A Power Index Based Framework for Feature Selection Problems

Corrado Mio

Corso di Dottorato in Informatica
Dipartimento di Informatica "Giovanni Degli Antoni"
Università degli Studi di Milano
XXXII Ciclo

Supervisore: Prof. Ernesto Damiani
Supervisore: Dr. Gabriele Gianini
Coordinatore Programma di Dottorato: Prof. Paolo Boldi

Anno Accademico: 2018-2019
Settore scientifico-disciplinare: INF/01

January 30, 2020

This thesis is dedicated to
Dr. Jean-Luc Widlowski
per avermi spinto ad intraprendere questa avventura
Dr. Gabriele Gianini e Prof. Alessandro Rizzi
per aver creduto nella fattibilità di questa avventura
Dr. Jiany Lin e Dr. Leopold Ghemmogne Fossi
per avermi aiutato a superare gli infiniti dubbi che questa avventura ha generato
i miei genitori,
per avermi supportato in questa avventura

Abstract

One of the most challenging tasks in the *Machine Learning* context is the *feature selection*. It consists in selecting the best set of features to use in the training and prediction processes. There are several benefits from pruning the set of actually operational features: the consequent reduction of the computation time, often a better quality of the prediction, the possibility to use less data to create a good predictor.

In its most common form, the problem is called *single-view* feature selection problem, to distinguish it from the feature selection task in *Multi-view* learning. In the latter, each view corresponds to a set of features and one would like to enact feature selection on each view, subject to some global constraints.

A related problem in the context of *Multi-View Learning*, is *Feature Partitioning*: it consists in splitting the set of features of a single large view into two or more views so that it becomes possible to create a good predictor based on each view. In this case, the best features must be distributed between the views, each view should contain synergistic features, while features that interfere disruptively must be placed in different views.

In the semi-supervised multi-view task known as *Co-training*, one requires also that each predictor trained on an individual view is able to *teach* something to the other views: in classification tasks for instance, one view should learn to classify unlabelled examples based on the guess provided by the other views.

There are several ways to address these problems. A set of techniques is inspired by *Coalitional Game Theory*. Such theory defines several useful concepts, among which two are of high practical importance: the concept of *power index* and the concept of *interaction index*. When used in the context of feature selection, they take the following meaning: the *power index* is a (context-dependent) synthesis measure of the prediction's capability of a feature, the *interaction index* is a (context-dependent) synthesis measure of the interaction (constructive/disruptive interference) between two features: it can be used to quantify how the collaboration between two features enhances their prediction capabilities. An important point is that the power

index of a feature is different from the predicting power of the feature in isolation: it takes into account, by a suitable averaging, the context, i.e. the fact that the feature is acting, together with other features, to train a model. Similarly, the interaction index between two features takes into account the context, by suitably averaging the interaction with all the other features.

In this work we address both the single-view and the multi-view problems as follows.

The *single-view feature selection problem*, is formalized as the problem of maximization of a pseudo-boolean function, i.e. a real valued set function (that maps sets of features into a performance metric). Since one has to enact a search over (a considerable portion of) the Boolean lattice (without any special guarantees, except, perhaps, positivity) the problem is in general NP-hard. We address the problem producing candidate maximum coalitions through the selection of the subset of features characterized by the highest power indices and using the coalition to approximate the actual maximum. Although the exact computation of the power indices is an exponential task, the estimates of the power indices for the purposes of the present problem can be achieved in polynomial time.

The *multi-view feature selection problem* is formalized as the generalization of the above set-up to the case of multi-variable pseudo-boolean functions.

The *multi-view splitting problem* is formalized instead as the problem of maximization of a real function defined over the partition lattice. Also this problem is typically NP-hard. However, candidate solutions can be found by suitably partitioning the top power-index features and keeping in different views the pairs of features that are less interactive or negatively interactive. The sum of the power indices of the participating features can be used to approximate the prediction capability of the view (i.e. they can be used as a proxy for the predicting power). The sum of the feature pair interactivity across views can be used as proxy for the *orthogonality* of the views.

Also the capability of a view to pass information (to teach) to other views, within a *co-training procedure* can benefit from the use of power indices based on a suitable definition of *information transfer* (a set of features – a coalition – classifies examples that are subsequently used in the training of a second set of features).

As to the feature selection task, not only we demonstrate the use of state of the art power index concepts (e.g. Shapley Value and Banzhaf along the

lines described above Value), but we define new power indices, within the more general class of *probabilistic power indices*, that contains the Shapley and the Banzhaf Values as special cases. Since the number of features to select is often a predefined parameter of the problem, we also introduce some novel power indices, namely k-Power Index (and its specializations k-Shapley Value, k-Banzhaf Value): they help selecting the features in a more efficient way.

For the feature partitioning, we use the more general class of *probabilistic interaction indices* that contains the Shapley and Banzhaf Interaction Indices as members.

We also address the problem of evaluating the teaching ability of a view, introducing a suitable *teaching capability index*.

The last contribution of the present work consists in comparing the Game Theory approach to the classical Greedy Forward Selection approach for feature selection. In the latter the candidate is obtained by aggregating one feature at time to the current maximal coalition, by choosing always the feature with the maximal marginal contribution.

In this case we show that in typical cases the two methods are complementary, and that when used in conjunction they reduce one another error in the estimate of the maximum value.

Moreover, the approach based on game theory has two advantages: it samples the space of all possible features' subsets, while the greedy algorithm scans a selected subspace excluding totally the rest of it, and it is able, for each feature, to assign a *score* that describes a context-aware measure of *importance* in the prediction process.

Corrado Mio

Milano, January 2020

Abstract

Una delle principali attività richieste nell'utilizzo degli algoritmi di Machine Learning è la *feature selection*. Questa attività consiste nel selezionare l'insieme delle feature migliori da utilizzare sia nella fase di training degli algoritmi, sia durante il processo di predizione. Si ottengono molti benefici nel ridurre l'insieme delle feature da utilizzare: una riduzione dei tempi di elaborazione, spesso una migliore qualità della predizione, la possibilità di utilizzare una quantità inferiore di dati per ottenere un buon predittore.

Nella sua forma più comune, il problema è chiamato *single-view feature selection*, per distinguerlo dallo stesso problema ma nel contesto del *multi-view learning*. In quest'ultimo caso, ciascuna vista corrisponde ad un insieme di feature le quali devono essere selezionate soddisfacendo un certo numero di contraint globali.

Un problema correlato, nel contesto del *multi-view learning* è il *feature partitioning*: l'operazione consiste nel suddividere le feature, appartenenti ad un'unico dataset, in due o più view in modo tale che con i dati di ogni view sia possibile creare un buon predittore. In questo caso, le feature migliori devono essere distribuite tra le diverse view, inoltre, ogni view deve contenere feature che collaborano in modo sinergico, mentre feature che interferiscono negativamente devono essere poste in view diverse.

Nel contesto del *co-training*, una delle attività nell'ambito del *semi supervised multi-view learning*, oltre al partizionamento delle feature in diverse view, è richiesto anche che ogni view sia in grado di *insegnare* qualcosa alle altre view: in particolare le istanze del dataset non ettichettate di una view possono essere classificate usando le predizioni offerte dalle altre view.

Ci sono molti modi per affrontare le attività appena descritte. Diverse tecniche sono ispirate dalla *Coalitional Game Theory* (*Teoria dei Giochi Cooperativi*). Questa teoria mette a disposizione molti concetti utili, tra i quali due sono di importanza pratica: il concetto di *Power Index* ed il concetto di *Interaction Index*. Quando usati nel contesto della feature selection, prendono il seguente significato: il *power index* è una misura sintetica (dipedente dal contesto) della capacità di predizione, mentre l'*interaction index* e' una misura sintetica (sempre dipendente dal contesto) dell'interazione (positiva

o negativa) tra due o più feature: per questo motivo, può essere usato per quantificare come, la collaborazione tra due feature, migliora (o peggiora) la loro capacità di predizione. Un punto importante da tenere in considerazione è che il power index è differente dalla capacità predittiva della featura considerata singolarmente: il power index prende in considerazione, mediante un'opportuna media pesata, il contesto, cioè l'interazione con le altre feature. Similmente, l'interaction index tra due o più feature prende in considerazione il contesto, anche in questo caso mediante una media pesata, dell'interazione con le altre feature.

In questo lavoro affrontiamo i problemi associati al single-view ed al multi-view learning nel seguente modo.

La feature selection nel contesto del *single-view learning* è formalizzata come un problema di massimizzazione di una funzione pseudo-booleana, cioè una funzione con dominio i possibili sottoinsiemi di feature, e con codominio i numeri reali. La funzione ha il compito di mappare ogni sottoinsieme di feature in una misura della qualità della predizione. Dal momento che si deve mettere in atto una ricerca (su una parte considerevole) del reticolo booleano (senza garanzie particolari, tranne, forse, la positività), il problema è generalmente NP-hard. Quindi, affrontiamo il problema identificando le coalizioni migliori attraverso la selezione del sottoinsieme di feature caratterizzate dai più alti power index, ed utilizzando tale coalizione per approssimare il massimo effettivo. Sebbene l'esatto calcolo dei power index sia un compito esponenziale, le stime, ai fini del presente problema, possono essere raggiunte in tempi polinomiali.

La feature selection, nel contesto del *multi-view learning*, e' formalizzata come una generalizzazione del problema precedente.

Il problema del *multi-view splitting*, invece, è formalizzato come un problema di massimizzazione di una funzione reale definita sul reticolo delle partizioni. Anche questo problema è tipicamente NP-hard. In ogni caso, soluzioni candidate possono essere trovate mediante un opportuno utilizzo delle feature con il miglior power index e mettendo le feature che non interagiscono, o interagiscono negativamente, in differenti view. La somma dei power index associati alle singole feature può essere usata per approssimare la capacità di predizione della view. La somma degli interaction index di coppie di feature può essere usata per valutare l'*ortogonalità* delle view.

Anche la capacità di una view di trasferire informazioni (di *insegnare*) ad

un'altra view, nel contesto del *co-training*, può beneficiare dell'uso di power index basati su un'opportuna definizione di *trasferimento di informazione*.

Per quanto riguarda l'attività di selezione delle feature, non solo dimostriamo l'uso dei concetti di power index maggiormente conosciuti (lo Shapley Value e Banzhaf Value), ma definiamo nuovi power index, all'interno della classe più generale dei *probabilistic power index*, che contiene Shapley e Banzhaf Value come casi speciali. Poichè il numero di feature da selezionare è spesso un parametro predefinito del problema, introduciamo anche alcuni nuovi power index, vale a dire l'indice k-Power Index (e le sue specializzazioni k-Shapley Value, k-Banzhaf Value): questi indici permettono di selezionare le feature in un modo più efficiente.

Per il feature partitioning, utilizziamo la classe più generale dei *probabilistic interaction index* che contiene, come casi speciali, gli Interaction Index di Shapley e Banzhaf.

Affrontiamo anche il problema della valutazione della capacità di insegnamento di una vista, introducendo un opportuno *teaching capability index*.

L'ultimo contributo del presente lavoro consiste nel confrontare l'approccio basato sulla Game Theory con il classico approccio Greedy Forward Selection per la selezione delle feature. In quest'ultimo caso l'insieme delle feature viene ottenuto aggregando una feature alla volta, aggiungendola alle feature già trovate, scegliendo sempre la feature con il massimo contributo marginale.

In questo caso mostriamo che, in casi tipici, i due metodi sono complementari e che, se usati insieme, si riducono a vicenda l'errore nella stima del valore massimo.

Inoltre, l'approccio basato sulla Game Theory presenta due vantaggi: campiona lo spazio di tutti i possibili sottoinsiemi di feature, mentre l'algoritmo greedy esegue la scansione di un sottospazio selezionato, escludendo totalmente il resto, ed è in grado, per ogni feature, di assegnare un *punteggio* che descrive una misura dell'*importanza* della feature nel processo di predizione.

Corrado Mio

Milano, January 2020

Contents

List of Figures

List of Tables

Notation

$\mathbb{N}$	set of natural integers
$\mathbb{N}^+$	set of positive integers (excluding 0)
$\mathbb{Z}$	set of integer numbers
$\mathbb{R}$	set of real numbers
$\{0,1\}$	set of boolean/binary values
$[a,b]$	closed interval of integers or reals
$[a,b)$	semi-open interval of integers or reals
$[A,B]$	closed interval of subsets $C : A \subseteq C \subseteq B$
$\{a,b,\ldots\}$	set of elements
$\langle a,b,\ldots\rangle$	sequence of elements
$\|S\|, s$	set or sequence cardinality
N	set $\{1,2,\ldots,n\}$
2^N	powerset of N
$N^{[0,k]}$	family of sets with cardinality $\leq k$
i, ij	when used in set operations, sets $\{i\}$ and $\{i,j\}$
$S_i, S(i)$	boolean value that says if i is in the set S or not
$\mathcal{P}^m(N)$	partitions of N in m blocks
π	permutation

$\Pi(N)$ permutations of N

$\Pi(N^k)$ permutations of N of length k

$S_i(\pi)$ set of elements preceding i in the permutation π

$S_i, S[i]$ i^{th} element of the sequence S

$S[l:u]$ sequence of elements with indices in the range $[l, u)$

$\mathbf{x}$ column vector of binary values

$\mathbf{x}^T$ row vector of binary values

$\mathbf{1}_S$ binary representation of the set S

$\langle \cdot, \cdot \rangle$ inner product

$\langle \cdot, \cdot \rangle_\mu$ weighted inner product

$(n)_k$ failing factorial

$\binom{n}{k}$ combinatorial coefficient

$\binom{n}{k}^*$ sum of combinatorial coefficients

$i \oplus j$ modular + with integer values in the range $[1, n]$

M maximal chain

$C(N)$ set of maximal chains

$M_S, M[S]$ smallest set in M containing S

ξ, ζ set functions with domain 2^N

f, g pseudo boolean functions with domain $\{0,1\}^n$

$\Delta_K \xi(S)$ discrete derivative

$\mathcal{L}(2^N)$ space of set functions defined on N

$\mu(S)$ probability distribution used in inner products

f_ξ pseudo boolean function induced by ξ

ξ_f set function induced by f

$\delta_T(\mathbf{x})$ Dirac basis

$e_T(\mathbf{x})$ Unanimity Game basis

$w_T(\mathbf{z})$ Walsh basis

$\Delta_K f(\mathbf{x})$ derivative of a pseudo boolean function

G TU game

$\mathcal{G}^N$ space of all TU games defined on N

$\mathcal{S}$ solution concept

$p_i(T)$ weights used in power indices

$\phi_\xi(i)$ probabilistic value

$\phi_\xi^{Sh}(i)$ Shapley Value

$\phi_\xi^{Ch}(i)$ Chaining Value

$\phi_\xi^{B}(i)$ Banzhaf Value

$\phi_\xi^{\mathcal{P}}(i)$ Weighted Banzhaf Value

$p_S(T)$ weights used in interaction indices

$I_\xi(S)$ probabilistic interaction index

$I_\xi^{Sh}(S)$ Shapley Interaction Index

$I_\xi^{Ch}(S)$ Chaining Interaction Index

$I_\xi^{B}(S)$ Banzhaf Interaction Index

$I_\xi^{\mathcal{P}}(S)$ Weighted Banzhaf Interaction Index

$m_\xi(S)$ Möbius transform

a_S Möbius coefficient

G graph

$\mathcal{T}_\tau$ teaching function

V graph vertices

E graph edges

`Ncut` graph edges normalized cut

w graph weight function

$\deg(v)$ vertex's degree

`vol`(V) volume of the vertex set

`cut` graph edges cut

`RatioCut` graph edges ratio cut

`Wcut` graph edges weighted cut

L Laplacian matrix

W Adjacent matrix

D Degree matrix

$\mathcal{Q}(\cdot)$ prediction's quality

$H(\mathbf{p})$ entropy

$\mathcal{D}$ dataset

$\mathcal{L}$ labelled dataset

$\mathcal{U}$ unlabelled dataset

$\mathcal{A}_{\mathcal{L}}$ Machine Learning algorithm

$\mathbf{d}$ dataset instance

$\mathbf{p}$ classification prediction

$\mathcal{D}^p$ dataset restricted to the view p

$\mathcal{L}^p$ labelled dataset restricted to the view p

$\mathcal{U}^p$ unlabelled dataset restricted to the view p

$\mathcal{A}^p_{\mathcal{L}}$ Machine Learning algorithm restricted to the view p

$\mathbf{d}^p$ dataset instance restricted to the view p

$\mathbf{p}^{(p)}$ classification prediction restricted to the view p

$I^{pq}_{\Xi}(ij)$ mixed interaction index of a partition function

Chapter 1

Introduction

1.1 Context and Problems

1.1.1 Machine Learning algorithms

The last years have seen a large development of *Machine Learning*, the area of the *Artificial Intelligence* responsible for creating algorithms capable of *learning*, in an automatic way, from some provided *data* (the *dataset*) and making autonomous decisions after the acquisition of such experience.

There are a large number of Machine Learning algorithms that can be classified in several different ways. A criterion for such classification is based on the presence/absence in the training of the desired outcome of the learning process, the so called *target* (e.g. the labels for a classification process):

- **supervised algorithms**: the algorithm uses for its *training* a *labeled dataset*, a set of data containing the required result. The task of the algorithm is to learn the best function capable of replicating the behaviour described in the dataset. They can be further subdivided into *classification algorithms* if the result is a *category*, and *regression algorithms* if the result is a numerical value
- **unsupervised algorithms**: the algorithm does not require labels for training. It tries to extract specific properties from the data. This class of algorithms contains, for example the *clustering* algorithms.

- **semi-supervised algorithms**: only part of the target information is available. Typically only some instances in the dataset have a label: the algorithm uses them for its initial *training*, then it utilizes this *knowledge* to compute and propagate the labels to the remaining unlabeled part of the dataset. Examples of these algorithms are self-training algorithms and co-training algorithms.

1.1.2 Feature selection

Motivations. One of the main problems with the supervised and semi-supervised learning algorithms is the selection of the *best features* to use in the training and prediction phases. Such a selection is necessary for the following reasons

- the quality of the training depends on the samples' quality in the features' domain. The problem is that the number of samples for having a good domain's sampling increases exponentially with the number of feature (the *curse of the dimensionality*)
- the presence of useless features increases the computation time and the space required for storing the data, without increasing the quality of the learning and predictions
- the presence of poor quality features reduces the overall quality of the training and the prediction capability of the algorithm

In short, in the task of feature selection we are given a large set of features, and we would like to select a small subset of it so as to retain most of the information.

General framing. This task can be formalized as the task of the maximization of a set function. A *set function* is a function that maps a subset to a real number: in the present case the set is the set of features whereas the real number represents a performance metric of the algorithm (e.g. the accuracy of a classification algorithm).

Such problem is in general NP-hard: it requires to analyze the 2^n possible subsets, operation feasible only for small number of features.

The problem of feature selection is typically formalized as a *cardinality-constrained monotone sub-modular* (positive) set function maximization problem in the *value oracle* setting. Indeed typically out of n features one aims at the selection:

- of a fixed number k of features (*cardinality constraint*),
- in a setting where adding a new feature to an given feature-set always increments (never decrements) the modeling power (*monotonicity*)
- where, as we consider larger sets, the marginal benefit of adding a new feature decreases (*sub-modularity*)
- and where the function is accessible through a "black box" (the *value oracle*) returning the value $f(S)$ for a given set S (the analytic form – boolean polynomial form – of the function is not known and would require exponentially many queries to the oracle).

An analogous problem is the so called Sensor Placement problem [58, 59], where one is given a large number n of locations, and would like to choose k locations where to place sensors so as to maximize some objective function, such as the coverage.

A wide literature addresses optimization problems for sub-modular objective functions: in the case of unconstrained minimization, polynomial time algorithms are available [60, 61]; for instance the Min-Cut problem in graphs is polynomial-time solvable.

On the contrary, even the simplest maximization problems of sub-modular functions turn out to be NP-hard [62–65]; for instance, the Max-Cut and Max-Direct-Cut problems (not necessarily monotones), and the Max-k-Cover problem (of which Sensor Placement is an instance) are known to be NP hard. As a consequence, in those optimization problems one has to resort to approximated/sub-optimal solutions.

This is even more motivated by the fact that with real datasets and with typical learned models the set function (a performance metric) is not in general endowed by strict monotonicity, sub-modularity or other *nice* properties. Only in very rare cases, the dataset has *nice* properties and the algorithms work very well. The literature tends to deal only with *nice* datasets.

Embedded, filter and wrapper methods. There are several *approximate* methods for addressing the feature selection task: Blum and Mitchel [123] classify them into the following categories:

- *filter methods*: these methods use the statistical properties of the features (the correlation between a feature and the target or among the features themselves) or use a projection of the space of the features onto a space with a lower dimension
- *embedded methods* the selection of the best set is embedded into the algorithm itself during the phase of predictor's creation (this is the case of the decision trees, which choose the feature for spitting based on Gini index, entropy, information gain, etc.)
- *wrapper methods*: the methods use the machine learning algorithm as a black box that returns a measure of the prediction's quality computed by means of a given metric (for example the *accuracy* in the classification algorithms). The goal is to find the set of features with the highest value for that metric. The selection of the *best set* is based on an optimization algorithms over that metric.

Notable examples of algorithms using the wrapper approach are Greedy Forward Selection (GFS) and Greedy Backward Elimination (GBE). Both apply a greedy hill climbing search strategy.

In GFS one evaluates sets of features of increasing size, adding one feature at time: each time the feature chosen is the feature that best enhances the performance of the current feature set over the training dataset. The algorithm starts with one feature and stops when the number k of desired features has been reached.

In GBE one evaluates sets of features of decreasing size, starting from the set of all features and removing one feature at time the number k of desired features has been reached: each time the feature chosen to be dropped is the one whose removal minimally reduces the performance of the remainder feature set.

Frame of Contributions. The present work studies an approach to feature selection that can be framed within the area of the wrapper methods, but – differently from GFS and GBE – performs a complex evaluation of

the individual features, based on *Coalitional Game Theory* (CGT) concepts, then assembles the most performing features to yield a candidate "coalition" of features. The performance of individual features is not computed by considering the predictive power of the feature in isolation (the *naïve* approach), but rather by averaging its contributions to an adequately large number of coalitions: this average corresponds to the CGT concept of the *power index*, a measure of the feature/player *ability* to be useful in the prediction. The most well-known power indices are the Shapley Value and the Banzhaf Value: we return on their definition below. A CGT concept related to the one of power index is the one of *interaction index*: one such index quantifies the effectiveness of pairs of features/players in the context of all the possible coalitions comparative to their individual effectiveness; it measures whether two features are redundant, positively or negatively synergistic.

Related work. The use of power indices in feature selection has been developed in the latest fifteen years by a small number of authors starting from a paper by Cohen, Dror and Ruppin [125, 126] (variations around the same ideas can be found in [130]), however the concept has a long history in the Economic literature and has been used very often in the Computer Science literature. The Shapley Value, particularly, since its introduction in 1953, has generated a large number of papers [4], where it was alternatively interpreted as a solution of the fare division problem [168], as a power index [18], as a centrality measure [3] or as a transform endowed of desirable properties within the Dempster-Shafer evidence theory [2] (a.k.a. theory of belief functions). By analogy to its interpretation as a power index, it has been used to assess the importance of components in a composite entity (a system or a process); among the recent examples, we can mention its application in algorithm portfolio selection [16], tag sense disambiguation [15], neural network pruning [5]. It has also been used in model interpretation [10, 12].

The Shapley Value has been used also in the context of explanation and interpretation of prediction models. The earliest work using the Shapley Value to address such problem of interpretability is the work [12] by Lipovetsky and Conklin, who use the SV to quantify the relative importance of the predictors in linear regression and show that the approach is robust with respect to the presence of collinearity. In the work [10] Lundberg and Lee address the problem of interpretability of the results of a prediction model. They consider an explanation model g to be a simple approximations of a prediction models f and focus on local additive explanation models: the local aspect corresponds to the fact that they aim at explaining a prediction

$f(x)$ based on a single input x; the additivity implies that the explanation model attributes an effect to each feature and summing the effect of each feature attribution one gets the approximation of the original model; they show that the Shapley Value allows deriving a class of additive importance measures fulfilling a set of desirable requirements. The accent of the paper is on formulating explanation models: the authors define a class of explanation models which unifies six major methods (the class is named additive feature attribution methods) and they validate their work by means of a user study showing that the approach is aligned with human intuition. An earlier work by Strumbelj and Kononenko [14] also address the problem of the explanation of prediction models. They focus on the situational importance of a feature: this is defined as the difference between what a feature contributes in average what it contributes when it takes a specific value (the average contribution represents the contribution when the value of a feature is not known or missing). Such a concept represents a useful explanation only if the prediction model is additive. The authors find that a convenient explanation model for non-additive cases is provided by a weighted average of the situational importance with respect to all the possible sets of missing features. The solution turns out to correspond to the Shapley Value of the situational importance.

The Shapley Value has several useful properties and one of them is that it can be used as linear approximation of the set function. Mikenina and Zimmermann [124] use the *Möbius transform*, one of several possible transforms for the set functions, limited to the terms of the first and second degree. Liu and Wang [131] use the Shapley Value in a two phases' algorithm: in the first phase, they use the mutual information to identify a good super-set of features, then, they use the Shapley Value for selecting the best subset. Sun et al. [128] use the Shapley value and a discretized version of the *conditional mutual information*. Gore et al. [130] use the Shapley Value applied to the Relief algorithm [156]. Mokdadl et al. [132] use the Shapley Value and a *ranking agreement*. Sun et al. ([127]), in another article, use the Banzhaf Value as an alternative to the Shapley Value. Kulynych et al. [129] use the Banzhaf Value in combination with a neural network.

Contributions in a nutshell. With respect to the above concepts, the contributions of the present work can be synthesized as follows. We model the feature selection problem as a problem of optimization on the Boolean lattice and carry a systematic experimental study (by real data-sets and simulation) on the the effectiveness of the most prominent existing power

indices, such as the Shapley and Banzhaf values (in the simulation we use also different assumptions on the property of the set function). We highlight the rationale behind their use in optimization: while the existing literature is based only on the analogy between the CGT concept of power and the related concept of influence on the results, we provide a frame of interpretation which involves pseudo-boolean function optimization. We define new power and interaction indices and study their CGT properties. Furthermore we use the power indices in conjunction with their interaction index (e.g. Shapley Value with Shapley Interaction Value) to build a bi-dimensional view of the features and assess the quality of information that pairs bring together (to be used in multi-view learning problems). Those contributions are reported in a more detailed summary below. We also contributed some methods for the approximate estimate of the power indices (whose exact computation would be exponential in the number of features): the novelty of our methods (which have polynomial complexity) is that they are tailored to achieve a value sufficient for supporting the feature selection process.

1.1.3 Feature Partitioning and Co-training

A common issue in the context of the Machine Learning is the high cost of providing the required *labeled dataset* for training. Very often dataset are only partially labeled, and finding labels for the missing instances can be very expensive, or even impossible, because experts capable to provide examples that can be used for labeling are not available.

Semi-supervised learning. self-training. To address this issue a class of *semi-supervised* algorithms has been designed. These algorithms initially learn a model using the labeled data (*initial labelled training set*), then the learned algorithm itself is used to predict the label for the unlabeled data. Subsequently, the best predictions (called *weakly labelled training set*) are used to extend the collection of labeled instances and to extend the *current labelled training set*. The process is iterated until all the data have been labelled. The simplest semi-supervised procedure of this kind is known as *self-training* (see for instance [57]).

Multi-view learning, Co-training. In self-training the learning process involves all the features considered as a single unified view of the data. How-

ever, often the different statistical properties of subsets of features can be better exploited splitting the feature set into two or more "views", and having each view train different models, that can subsequently be made to interact or can be integrated. This area goes under the name of *multi-view* learning [1].

A class of multi-view algorithms that proved to be effective are the co-training algorithms, which represent a simple generalization of self-training to two or more views: each view is used to train a prediction algorithm, then the prediction capability of an algorithm is used for "teaching" the other ones how classify instances on which they might have weak confidence.

Blum and Mitchell [148] specify the properties of the views for obtaining a good co-training algorithm:

- each view must be *sufficient*: using this view it is possible to obtain a good predictor: thus one of the objectives is to maximize the *predicting power* of individual views
- the views must be *consistent*: the predictors obtained from each view must predict the same class, if the prediction has a high confidence
- *conditional independence* in relation to the target, that is, the views must be independent

Multi-view splitting. The co-training algorithms assume the existence two or more views. Sometimes the features are already organized into distinct views naturally available and distinguishable in the application field. This is the case for instance for the text properties and the links' properties of a web page. Other times a single large view has to be split into smaller views, not only for practical computational reasons but sometimes for improving the learning process. Indeed, one may decide to split a view in two – instead of training a predictor on the whole set of available features – as a consequence of the fact that some features interact with some other destructively: in that case, isolating a subset of features (the view) from the negative influence of other features (the other view) can be beneficial for the training.

The simplest method to enact the splitting consists in selecting the features for each view randomly, as described in Nigam and Ghani [150]. Zhang and Zheng [154] describe the TSFS (Two-view Subspace Feature Splitting) algorithm. Chen et al. [137] describe the *Pseudo Multi-view Co-training*,

an algorithm that selects the features such that the ϵ-*expanding* property is ensured. We go beyond the random splitting and model the problem of view splitting as a problem of optimization on the partition lattice.

Multi-view feature selection. Related to the splitting of the feature set into two or more views is the problem of dropping the features that are less useful, to enhance the computational performance of the procedure. We model also this problem as a problem of optimization on the partition lattice, with the special constraint that one partition block has no consequence (is assigned zero weight) on the overall objective function.

Both in multi-view splitting and in multi-view feature selection the three requirements listed above represent a guide to the construction to the relevant objective function, so it is worth considering their rationale and their relationships. We examine first the second property, then the first and the third. For the sake of simplicity here we assume that the task consists in splitting the whole set of features in exactly two views.

Rationale of requirements: 2 - Consistency. Consistency is a natural requirement, we would like the views to yield the same predictions over any given target. Considering that often to a prediction can be associated a confidence level, one would like that at least the views' predictions agree when they are highly confident. A way for resolving conflicts is that of choosing the prediction of the view endowed with the highest confidence.

Rationale of requirements: 1 - Predicting Power The first property, sufficiency, corresponds to the requirement that each individual view has a non-negligible "predicting power" on its own. This is the basic requirement for any classifier/predictor and it is reasonable that it should hold also for the individual views. In the two views case, in principle the predicting power of the pair of views – if the aggregation mechanics is reasonable – should be monotonically increasing (or at least non-decreasing) w.r.t. each individual view's predicting power. So, in principle, "the higher the predicting power of each view, the better the multi-view predicting power".

However, as an objective function one should not just consider simply the sum (or the probabilistic sum) of the two views' predicting power: that

sum does not represent fully the actual predicting power of the whole. This happens only in the extreme case in which the two views have strictly unique information. In the oppositely extreme case in which the views are completely redundant, on the contrary, the power of the whole is equal to the power of one part.

Ideally the same value of the sum of predicting powers can correspond to different configurations more or less valuable in terms of overall performance. This leaves room for expressing further requirements.

Rationale of requirements: 3 - Independence and conditional independence. In the process of splitting one would like to reduce redundancy. Redundancy, in turn, materializes a form of probabilistic dependence. Reducing probabilistic dependence (by suitably partitioning the features) from some level of redundancy to a state of independence of the views, would be beneficial. Consider accuracy. In principle, should one achieve the independence of two views, one could count on the "probabilistic sum" of the individual accuracies.[1]

The *unconditional independence* requirement is however at odds with the consistency requirement: one would like the views to agree as much as possible in their predictions! So when a view says something the other is expected to say the same. This of course implies high redundancy. The requirement of *conditional independence* of the views given the value of the target[2] en-

[1]Consider this stylized example, where we will aggregate the prediction of the two views by choosing the most confident one. Let model 1 and model 2 be trained respectively on view 1 and 2.
For the sake of illustration suppose the models are linear separator based classifiers and that the views are two dimensional euclidean plane representations of a nearly linearly separable dataset. It is reasonable to assume that when a point is far from the separator line, it has a higher confidence of being correctly classified, whereas if it is close to the separator line it has a smaller confidence: so correct classification and high confidence are highly correlated.
Let the models trained on view 1 and 2 have respectively probability p_1 and p_2 of being correct. Then the probability that either one or the other are correct is $P = 1 - (1 - p_1)(1 - p_2)$. Since, for the sake of simplicity, we assume that correct predictions have higher confidence than wrong predictions, if we choose the highest-confidence prediction from the two views we get the right result, thus P is also the predicting power of the multi-view classifier.

[2]While the requirement of independence can also be translated by saying that knowing the prediction of one view does help in guessing the prediction of the other, the requirement of conditional independence can be spelled by saying even when the target is known, knowing the prediction of one view does help in guessing the prediction of the other. So

forces a redundancy reduction compatible with the consistency requirement, by demanding that each view brings independent information to each target value. This also enhances the possibility that the diverse information from the two views brings different confidence.

Relative teaching power. The requirement of independence conditional to the target is rather strong and very difficult to obtain in real data sets. However Balcan et al. [149] argue that this condition is not strictly necessary: it is sufficient for a view to be capable of predicting the correct class with a confidence higher than a wrong class predicted by the other (the authors call this the ϵ-*expanding* property). Indeed this is one of the many ways in which one can require that a view holds unique information, i.e. information non available to the other view. We will return to the facets of this definition in the appropriate chapter. For now, we informally define the *teaching power* of a first view w.r.t. a second view, as *the fraction of times the former is capable to predict the correct class with a confidence higher than a wrong class predicted by the latter*. Notice that this teaching relation is directed and that in general is not symmetric. Since it is a relative property we call it overall *relative teaching power*. Notice also that this definition, pragmatically, displaces the emphasis from the relative properties of the views given a target value, to the effectiveness of the views when the target is the right one.

Synthesis of contributions. In this work we address the problems of view splitting and the related problem of feature selection in a multi-view context. We model the view splitting problem as a problem of optimization of partition functions, i.e. as optimization over the partition lattice. In this context the requirement of optimizing the views' prediction power can be approached again using the a power index as a proxy for individual prediction power of the individual features and the two-feature interaction index as a measure of what is gained from synergy within a view, or lost when two features are assigned to distinct views. Those contributions are reported in a more detailed summary below.

---

in a sense, to every target value the views bring unique information.

1.2 Summary of the Contributions

1.2.1 Feature selection

Based on the analogy between the individual power in determining the outcome of the collaborative effort of a coalition and the individual role of a feature in contributing to the performance of a learning model, the power indices from Game Theory have been used in recent years to assign to each feature a share of the total power: then the features with the highest power have been selected to build the "candidate best" coalition. The papers that used this approach offer anecdotic evidence of its effectiveness.

We go beyond the game theoretical analogy and – modeling the feature selection task as an optimization problem on the Boolean lattice, in the space of real-valued set functions – we argue that the effectiveness of the approach lies in the fact that the power indices are the coefficients of a first-order expansion of the relevant function. It is known, for instance that the Banzhaf Values of the elements with respect to a given set function are the first-degree coefficients of the approximating polynomial which minimizes the Euclidean distance (L2 distance or *mean square error*). Selecting the k elements corresponding to the top k coefficients for building the candidate best coalition of size k is equivalent to choose the set maximizing the approximating function and use it as a candidate of maximum set of the target set function.

We demonstrate that the different power indices are the *best* first order approximation, based on a *weighted mean square error*. Using different weights, yields different indices.

Based on this observation we develop a number of new power index definitions and frame them into a larger picture. It turns out that the power indices relevant to the problem of feature selection are instances of a general class of *probabilistic indices*. This class is very wide, and would be difficult to use in practice because it needs a lot of configuration parameters. However, two if its sub-classes, the *cardinal-probabilistic power indices* and the *player-probabilistic power indices* which contain Shapley and Banzhaf Value require only a little number of parameters. We focus on them.

Furthermore, we observe that very often, we need to select a predefined number of features. This means that we are interested on the behavior of the set function defined only on a level, say level k of the Boolean lattice.

In this case the approximating function must not approximate the entire function, but only the selected part. Based on this idea, we have defined a new class of power indices, the "K-Power Indices". We have extended the previous algorithms to support these new indices.

Those extensions of the list of available power indices introduces the problem of choosing the most appropriate for a given context. The choice can be made base on the axioms that each index satisfies, and on their practical interpretation. The choice can also be based on practical experimentation.

We carry a systematic experimental study (by real data-sets and simulation) on the the effectiveness of the most prominent existing power indices (in the simulation we try different assumptions on the property of the set underlying function).

In passing, during the experimental study we show using benchmark datat-sets, that the properties of monotonicity, non-additivity and submodularity do not hold strictly. We also characterize the behavior of the set functions with readily defined metrics which quantify the degree of compliance with the conditions of monotonicity non-additivity and submodularity.

We also contribute some methods for the approximate estimate of the power indices (whose exact computation would be exponential in the number of features): the novelty of our methods (which have polynomial complexity) is that they are tailored to achieve a value sufficient for supporting the feature selection process. Indeed, in the power index based features selection, it is not necessary to know the exact or even approximate value of the power index. Power index values are used *only* to order the features. This allows to simplify the implementation of the indices' algorithms. For example, given we are interested only on the order, the implementation can continue the computation only to obtain a *stable order*. Also this goal can be relaxed: very often it is important to separate some features with a high index value from the rest, so that the exact order of the selected features is not important. Using these ideas, we have defined some algorithms focused on keeping efficient the searching order, disregarding the value of the indices.

Furthermore, we use the power indices in conjunction with their interaction index (e.g. Shapley Value with Shapley Interaction Value) to build a bi-dimensional view of the features and assess the quality of information that pairs bring together (to be used in multi-view learning problems). The power index provides a measure of the predicting power of the feature in a context,

the interaction index tells whether this power is redundant or synergistic.

1.2.2 Feature Partitioning for Multi-view Learning and Co-training

In the context of the *Multi View Learning*, when starting from a single view, the first step to take consists in splitting the dataset in two or more parts that meet the above mentioned Blum and Mitchell or Balcan conditions.

To this purpose, considering power indices is not enough, since we need to fulfill some extra properties, which involve at least the relationship between pairs of features:

- features should be assigned to the same view when they work constructively together
- features interfering destructively, or with a low level of constructive interference, should be assigned to different views
- each view should hold enough unique information to be able to "teach" the others, i.e. to help other views to improve

Interaction and the interference can be modelled using the *interaction indices*, such as the Shapley Interaction Value or the Banzhaf Interaction Value. Interaction indices can be obtained as coefficients of the second degree terms of the second order approximation of the set function polynomial. We can use the interaction indices to evaluate:

- the level of collaboration collaboration between the features in the same view
- the destructive interference of the features assigned to different views
- the ability of a view to "teach" to another view

The first and second point relate to the maximization of the predicting power of the individual views and are relevant to all the multi-view learning processes, the third refers specifically to the co-training process. We summarize

the formalization of the problem and of the solution involving the interaction indices that we adopted w.r.t. the two problems.

We model both maximization problems as problems of optimization of partition function over the partition lattice.

Maximization of the views' predicting power. Two-view case. The prediction power of an individual view (measured for instance in terms of a classification performance metrics such as the accuracy) is in principle the results of the power of individual features, plus interactions at pair level, at triplet level and so on, up to the level k, where k is the size of the view. If we split the whole set of n features in two views, the second view's prediction power will be the result of the power of individual features, plus interactions at pair level, at triplet level and so on, up to the level $(n-k)$, which is also the size of the view. By splitting we deactivate all the cross-interactions between pairs of features belonging to distinct views and all the cross-interactions within sets which turn out to be split by the partition.

In principle finding the best splitting in this case corresponds to a search over the Boolean lattice, where for every set (a view) is associated a unique complementary set (the other view). Suppose for the sake of simplicity that the objective function is the sum of the predicting power of the two views: the problem of maximizing the sum can be mapped into the problem of removing the least useful interactions (among which, the most damaging interactions). Being the search over the Boolean lattice non feasible in general, one can resort to the approximation approach. One can project the set function on the first two levels, i.e. we can find the second degree approximation of the set function: the higher-order interactions of the graph will disappear from the approximation and be summarized by the interaction levels up to two in the approximating function. At this point the value of the set function for any subset (view) is approximated by the sum of the zero-th order coefficient, the first order coefficients of the elements belonging to the set and the second order interactions of the pairs that they define.

Thus the problem of maximizing the sum of the predicting power of the views is the problem of maximizing the sum of the set function values and can be mapped into the problem of removing the smallest value second order interactions. This is a problem of Min-Cut. In this problem the zero-th order, constant, term does not play any role, but the first order terms make the formulation slightly different from the classical one of the Weighted Min-

Cut problem, where the nodes of the graph do not have any weight, and only the edges have a weight. By interpreting the first order coefficients as weights of a self-loop edge we can restore the classical form of the problem and apply an algorithm for Weighted Min-Cut. In this way the problem of finding the best splitting becomes polynomial problem.

Maximization of the views' predicting power. Many-view case. The approach can be generalized to an finite number of views: the problem of splitting in k views is mapped into a k-way weighted Min-Cut problem: again polynomial algorithms exist for this task.

Maximization of the views' "teaching power". When one deals with the relatively complex process of co-training, maximizing the predicting power of the views is not enough. The (models trained on the) views will interact during several iterations by progressively improving one's another prediction power. Thus one can stipulate that also the diversity of the knowledge held by the different views is important: thus one can try also to maximize their "teaching power": to this purpose we count the number of times that a view can "teach" to the other one (how many times a view makes a correct prediction with a confidence higher than the one by which the other view makes a wrong prediction on the same instance). From the count one can obtain a rate: the *teaching power of a view w.r.t. another*. Using this metric, the power indices and the interaction indices, we can define an integer programming optimization problem to distribute the features between the views that satisfy the required list of properties.

An issue to consider, w.r.t. the the ability of a view to "teach" to another view, is the number of interactions among views to consider. If we have only two views, it's simple: the first view teaches the second and the second one to the first. If we have v views, the number of interactions is $v(v-1)$. A simple solution, to reduce the number of interactions, is to assign an order to the views and consider only the interactions between one view and the next one (and the last view with the first one). In this way, the number of interactions to consider is only v.

1.2.3 Effectiveness of Power Index based methods

The last problem is to understand when Power Index based methods represent a comparative advantage in terms of classification performance and computational performance. The greedy approach, used in several Machine Learning algorithms, is very efficient, because it considers only the best local choices at each step. But it totally excludes a lot of other possibilities. Greedy methods work well only if the best local choices are also the best global ones. But this is true only in a very small number of cases. We have observed, in some experiments, that very often the behaviour of the quality metrics used to evaluate the Machine Learning algorithms has no special properties. Thus, the hypothesis used by greedy algorithms are not necessary satisfied.

Contrary to Greedy methods the CGT methods based on power indices have a *global approach*: they sample the whole support of the function using suitable weight, then synthesize the global behaviour by a low degree approximation function. This function is used to resolve the problems in feature selection and feature partitioning.

In practice, set functions can be organized in four sub-spaces:

- a subspace where greedy methods work well, because the functions satisfy the greedy hypotheses, and power-index based methods fail
- the sub space where and power-index based methods work well, but the greedy approach fails
- the subspace where and power-index based methods and greedy methods return the same result
- the subspace where both and power-index based methods and greedy methods fail

Identifying precisely those sub-spaces is an open challenge, however simulation based investigations seem to indicate that the two approaches are to a good extent complementary. Thus, for instance using both (since both admit polynomial time implementations or approximations), can lead to a polynomial time algorithm with improved prediction power.

The design of a suitable Monte Carlo generator for set function is one of the challenges undertaken by this thesis.

1.3 Structure of the document

The structure of this document is the following.

The Chapter 2 introduces the definitions of the mathematical objects used in the rest of the document. After the standard notations for integers, reals, intervals, sets, etc, it defines the concepts of *set function*, *pseudo boolean function* and *set function space*. Since the *set function space* is a *vector space*, it defines some very often used *basis*. There is also the discrete version of *function derivative*. The set functions are defined on sets, but there exist also functions defined on *partitions of a set*. Here it is used a special definition: a partition function *induced* by a partition and by the set functions assigned to each partition block. The last part of the chapter introduces a minimal set of definitions for graphs.

The Chapter 3 introduces the main objects used in the document: the *power indices* and the *interaction indices*. The classes of these indices can be organized in a hierarchical structure where the class of *probabilistic indices* is the root, the classes of *cardinal-probabilistic indices* and the *player-probabilistic indices* are two children, and the popular Shapley Value, Banzhaf Value (and related interaction indices) are instances of the previous ones. The chapter shows also that there is a bidirectional relation between set functions and power/interaction indices: they form an invertible *transformation*. The last part of the chapter shows that the indices can be defined using an axiomatic approach: selected a set of axioms, these define in an unique way a specific index. To know the axioms that define a index is important to understand which index to select for a specific problem.

The Chapter 4 shows the algorithms used to compute the power and interaction indices. Since the indices are combinatorial objects (to compute their values it is necessary to consider all possible subsets of a given set), it is not possible to compute the exact values, but it is necessary to use approximated methods. The chapter shows how these approximations are been implemented. There are described also some more efficient algorithms variants used when it is not necessary to know the power index values but only the values order.

The Chapter 5 shows why it is possible to use the power and interaction indices for feature selection. Here it is demonstrated that the power indices are the *best linear approximation* of the set function based on a *weighted mean square error*. Using different weights we obtain different power indices.

If it is necessary to approximate the original set function with a reduced number of terms, the best terms to use are the features with the highest power index value.

The Chapter 6 defines a new class of indices. The original indices try to approximate the *entire* set function. But in the problems of feature selection, we are interested on the values of the function defined on sets in limited cardinality range. The idea is to approximate a restricted set function to the selected sets. The chapter shows how to change the definitions of the original indices to consider only these cardinality. We have also derived the closed formula for the most simple power index (the Banzhaf Value) starting from scratch. The same approach can be used to find the closed formulas for the other indices.

The Chapter 7 analyzes another problem: the *feature partitioning*. In the *Multi View Learning* based on a single dataset, a problem is to split the dataset in two o more views such as: each view is able to predict well, each view contains collaborating features and features that interfere negatively are in different views. There are several algorithms available but here we use the concepts of Game Theory: the power and interaction indices. The chapter shows how to use the indices to define an optimization problem where the solution is the required views. It shows also that the feature partitioning problem can be defined as a graph partitioning problem. It describes how to adapt two of most used approaches (the spectral clustering and the multilevel graph partitioning) to the original problem. The chapter extends the feature partitioning in two extra directions: feature partitioning applied to the co-training, and when each view uses a different prediction algorithm.

In the feature partitioning for co-training, we have two goal to satisfy: each view must predict well, each view must teaches something to the other ones. The chapter defines the concept of *ability of teaching* of a view respect to another, and how to evaluate this ability. Then, the problem is modelled as an integer optimization problem. At the moment, it is not possible to convert the problem in a graph partitioning problem because the ability of teaching is a very strange object: it has a direction and must be considered only with the cut-edges selected in the partitioning. Furthermore, it change the original undirected graph in a directed one, and this change totally the type of the problem to resolve and the graph's algorithms to use.

The last extension is how to model the behaviour of a multi view learning algorithm when each view has a different prediction algorithm, and how to

define the *ability of teaching*. The chapter introduces the concept of *partition function induces by a partition and set function*, and the concept of *mixed derivative*, necessary to measure the teaching ability.

The Chapter 8 tries to answer, in some extremely simple cases, to the question: *when the Game Theory is better than a Greedy methods*. The question was born reading the articles, that use power indices (Shapley Value, Banzhaf Value, Möbius coefficients) that say: "using this index we obtain better results than with a greedy method", "this index is better because it has nice properties", etc. We have observed that this is not true. There is not an evidence that an index is better than another. The quality of the results depends totally by the data. Very often, the results obtained using different indices are very similar. However it is sure that there are cases where an approach based on the Game Theory is better than one based on Greedy method. This is related to how a greedy algorithm works: it extends the current solution only with solutions that contain the current one, excluding all others. Instead, a game theoretic method uses a global approach: it analyzes several possible solutions an tries to collect information about the importance of the solution's elements, to create a final solution that contains the most important elements. In another way, the greedy method works well when the function has nice properties (and in this case a game theoretic method returns the same results), where a game theoretic approach doesn't require special properties in the function.

The second part of the chapter contains a list of plots and properties (monotonicity, sub/super additivity, sub/super modularity) obtained tabulating the set function *accuracy* of a Decision Tree, applied to real datasets. These plots show that it is not possible to assume nice properties, as for example, the monotonicity. All functions are *sub additive* and *sub modular*, but these properties are too generic to be useful.

The Chapter 9 contains the conclusions of this work, and the open problems. The main open problems are: to understand when a game theoretic method is better than a greedy method, how to select the indices to use.

The last Chapter B contains a published paper on a clustering using a not conventional approach: using a transposed version of the dataset.

Chapter 2

Definitions

2.1 Introduction

This chapter introduces the main mathematical concepts used in the rest of the document.

2.1.1 Notations

The set of natural integers in denoted by $\mathbb{N}$, the positive integers (excluding 0) by $\mathbb{N}^+$, the set of integers by $\mathbb{Z}$, the real numbers by $\mathbb{R}$. With

$$i \oplus j = ((i + j - 1) \mod n) + 1$$

we denote the *modular* + with integer values in the range $[1, n]$.

The *closed interval* is denoted by $[a, b]$, the *open interval* by (a, b), and the *semi-open interval* by $(a, b]$ and $[a, b)$. We use the same syntax for real and integer intervals. If a and b are integers, also c in $a \leq c \leq b$ is an integer value.

The set is denoted by $\{a, b, \ldots\}$ or by an uppercase letter, as S, N, etc. N is used also for the set $\{1, 2, \ldots, n\}$. The cardinality of the set S is denoted by $|S|$ or by the correspondent lowercase letter ($s = |S|$, $n = |N|$, etc).

The *powerset* of N, the family of all possible its subsets, is denoted by $2^N = \{S : S \subseteq N\}$ (note that $S \subseteq N$ is equivalent to $S \in 2^N$). The family of sets with cardinality exactly k is denoted by $N^k = \langle S : S \subseteq N, |S| = k\rangle$. The family of sets with cardinality less or equal k by $N^{[0,k]} = \langle S : S \subseteq N, |S| \leq k\rangle$. The family of sets with cardinality in the range $[k_{\min}, k_{\max}]$ by $N^{[k_{\min},k_{\max}]}$ (2^N is equivalent to $N^{[0,n]}$). The cardinality of 2^N is 2^n. Given A and B with $A \subseteq B$, the family of all subsets between A and B is denoted as $[A, B] = [C : A \subseteq C \subseteq B]$. The notation is reminiscent of the fact that in set space the family is the analog to an interval. Note that if $A \cap B = \emptyset$, the family $[A, B]$ is empty.

We will use i, ij, etc, as notation for sets containing one ($\{i\}$), two ($\{i, j\}$) (or a small number of) element(s). Union, intersection and difference of two sets are denoted by: $A \cup B, A \cap B, A \setminus B$.

The symmetric difference is denoted by

$$A \Delta B \equiv (A \setminus B) \cup (B \setminus A) = (A \cup B) \setminus (A \cap B).$$

Some other operations are

$$A \setminus B \setminus C = (A \setminus B) \setminus C$$
$$A \setminus B \cup C = (A \setminus B) \cup C$$

In a set, the order of the elements is not important: $\{a, b, c\} = \{c, a, b\}$. A *sequence* is a list of elements where the order is important: $\langle a, b, c\rangle \neq \langle c, a, b\rangle$: they are called also ordered *tuples* (ordered pairs, ordered triplets and so on). In general, we will use sequences without duplicated elements. Elements in the sequence have an index starting from 1. The i-th element of the sequence S is denoted as S_i or $S[i]$. The length of a sequence is denoted by $|S|$. The *subsequence* $S[l : u]$ is the sequence composed of elements with index in the range $[l, u) = [l, u - 1]$ (with $l \leq u$). Note that $S[i : i]$ is the empty sequence, $S[i : i+1]$ is the sequence of one element, where $S[i]$ is the element in position i. The *concatenation* of two sequences is denoted by $S \cup T$.

The family of *partitions* of N in m blocks is denoted by $\mathcal{P}^m(N)$: it is a list of subsets $P_i \subseteq N$ such that

$$P_i \cap P_j = \emptyset \quad \forall i \neq j$$
$$\bigcup_{i=1}^{m} P_i = N$$

often denoted as $\mathbf{P} = \langle P_1, \ldots, P_m \rangle$.

The *Stirling numbers of the second kind* [173] is the number of ways in which one can partition a set of n elements into k subsets

$$\begin{aligned} \left\{ {n \atop k} \right\} &= \left\{ {n-1 \atop k-1} \right\} + n \left\{ {n-1 \atop k} \right\} \\ &= \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n \end{aligned}$$

Note that

$$\left\{ {n \atop 1} \right\} = \left\{ {n \atop n} \right\} = 1$$

The *Bell numbers* [174] counts the total number of partitions of the set

$$B_n = \sum_{k=0}^{n} \left\{ {n \atop k} \right\}$$

The *indicator function* of a set $S \subseteq N$) is

$$\mathbf{1}_S : N \to \{0, 1\}$$
$$\mathbf{1}_S(i) = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise.} \end{cases}$$

Another simple representation of the indicator function is

$$S_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathbf{1}_S = \langle S_i : i \in N \rangle$$

The *inverse* of the indicator function is denoted by:

$$N_{\mathbf{x}} = \{i : x_i = 1, i \in N\}$$

A *permutation* of a set is a *sequence* made of the elements from the set, without duplication. There are $n!$ *permutations* of n elements. The number of different *permutations* of k elements taken from n is $\frac{n!}{(n-k)!}$. The collection of all *permutations* of the set N is denoted by $\Pi(N)$. The collection of permutations of length k is denoted by $\Pi(N^{[0,k]})$. A *permutation* is denoted by $\pi \in \Pi(N)$. The sequence of elements preceding i in the permutation π is denoted by $S_i(\pi)$.

A *column vector* is denoted by a boldface lowercase letter $\mathbf{x}$. The *row vector* is denoted by $\mathbf{x}^T$. The vector composed by zeros is denoted by $\mathbf{0}$, whereas $\mathbf{1}$ denoted the vector composed by ones. The elements of the vector by x_i. The *dot/inner product* is denoted by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \cdot \mathbf{y} = \sum_{i=1}^{n} x_i \cdot y_i$$

By convention:

- $\sum_{i \in \emptyset} x_i = 0$
- $\prod_{i \in \emptyset} x_i = 1$
- $\sum_{i \in \emptyset} \prod_{i \in \emptyset} x_i = 1$

The *factorial* of n is denoted $n!$. The *falling factorial* $(n)_k$ is defined as:

$$(n)_k = \prod_{t=1}^{k} (n - t - 1) = \prod_{t=0}^{k-1} (n - t) = \frac{n!}{(n-k)!}$$

The *binomial coefficient* $\binom{n}{k}$ is defined as

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

It can be defined with the recurrent formula

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \tag{2.1}$$

If $k < 0$ or $n < k$, $\binom{n}{k} = 0$.

The *sum of binomial coefficient from 0 to k* is denoted

$$\binom{n}{k}^{*} = \sum_{i=0}^{k} \binom{n}{i}$$

with the property

$$\binom{n}{n}^{*} = \sum_{k=0}^{n} \binom{n}{k} = 2^n$$

2.1.2 Subset lattice

The relation *proper subset* $A \subset B$, where $A, B \subseteq N$, is a *partial order* between the sets in the family 2^N. The relation can be represented as a lattice ($L(N)$) where the nodes represent the sets and the edges the inclusion relation between pairs of sets. It can be visualized using the *Hasse diagram*, a *graph* where sets are ordered by increasing cardinality (each cardinality corresponds to one *level*). The bottom element is Ø (the empty set, with cardinality 0) and the top element is N (the full set, with cardinality n).

A *maximal chain* is an ordered collection of $n + 1$ nested distinct sets such that

$$M = \langle \emptyset = S_0 \subset S_1 \subset \ldots \subset S_n = N \rangle$$

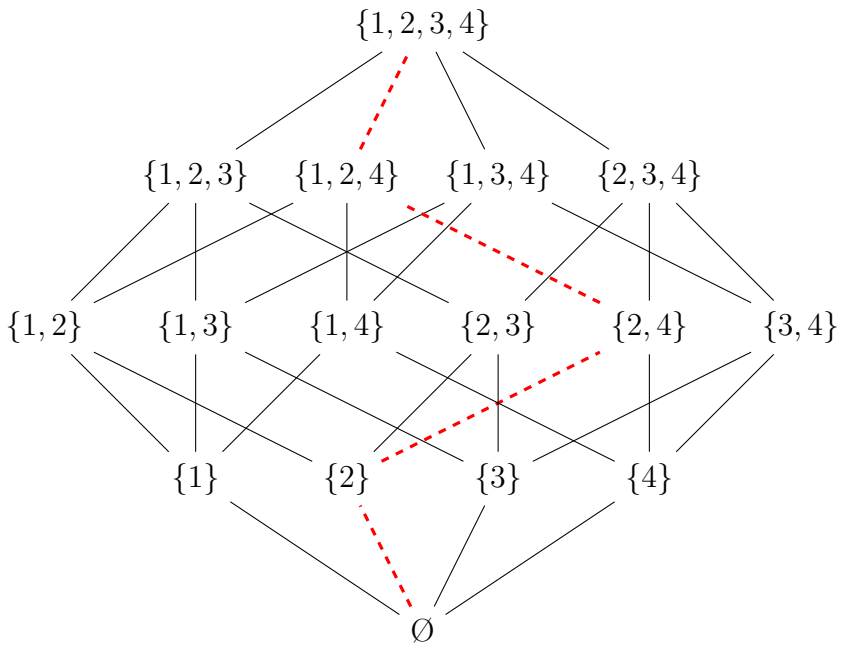


Figure 2.1: The Hasse diagram of $\{1, 2, 3, 4\}$

The maximal chain can be defined also using a permutation π with the convention

$$M = \langle S_i = \pi[1 : i + 1] : i \in [0, n] \rangle$$

(in the figure: $\langle 2, 4, 1, 3 \rangle$).

The set of maximal chains is denoted by $C(N)$. Its cardinality is $n!$.

Given $M \in C(N)$ a maximal chain and $S \subseteq N$ a set, M_S (or $M[S]$) is the smallest set in M containing S.

2.1.3 Set representation

The indicator function is a map between a set $S \subseteq N$ and the integers $l \in \mathbb{N}$ with n bits:

- we use the *natural* ordering of the elements i

- this order defines a unique map between the element i and its bit position $(i-1)$. The element 1 has position 0.

The operations on sets or between pairs of sets can be mapped onto operations on or between binary numbers:

- set union $\rightarrow$ `bitwise or`
- set intersection $\rightarrow$ `bitwise and`
- set complement $\rightarrow$ `bitwise not`
- set difference $\rightarrow$ `bitwise and` between the first set and the complement of the second set

The cardinality of the set is the number of bits with value 1.

Natural vs. Combinatorial Number system ordering. The problem with this representation is that the sets with a lower cardinality are mixed with sets with higher cardinality.

integer	binary rep.	set	cardinality
0	0000	$\emptyset$	0
1	0001	$\{1\}$	1
2	0010	$\{2\}$	1
3	0011	$\{1,2\}$	2
4	0100	$\{3\}$	1
5	0101	$\{1,3\}$	2
...	...	...	

Table 2.1: Cardinality in the binary representation of a set

In some algorithms we need to order the sets based on their cardinality, that is, we need to have first the empty set, then the sets with cardinality 1, the sets with cardinality 2, and so on.

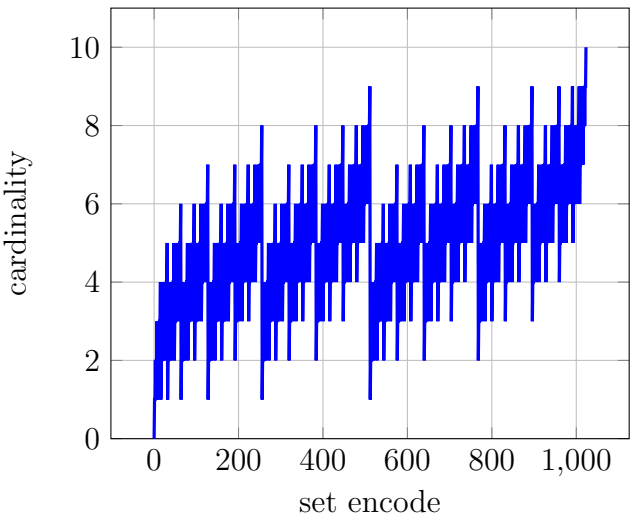


Figure 2.2: Cardinality in the binary representation of a set

To achieve this result, we can use another integer representation, based on the *Combinatorial Number System* (CNS) [66]. Using this representation, any integer l can be represented in an unique way as *ordered sequence of digits* $\langle C_1, C_2, \dots C_r \rangle$ where each *digit* C_i is an integer in the range $[0, n-1]$, with the property

$$0 \leq C_i < C_j \quad \forall\, i < j$$

i.e. the *lexicographic* order property, and

$$l = \sum_{i=1}^{r} \binom{C_i}{i}$$

The sequence can be converted into the correspondent set changing the digit C_i with the element $C_i + 1$.

integer	CNS rep.	cardinality
0	$\langle\rangle$	0
1	$\langle 1\rangle$	1
2	$\langle 2\rangle$	1
3	$\langle 3\rangle$	1
4	$\langle 4\rangle$	1
5	$\langle 1, 2\rangle$	2
...	...	...

Table 2.2: Cardinality in the CNS representation of a set

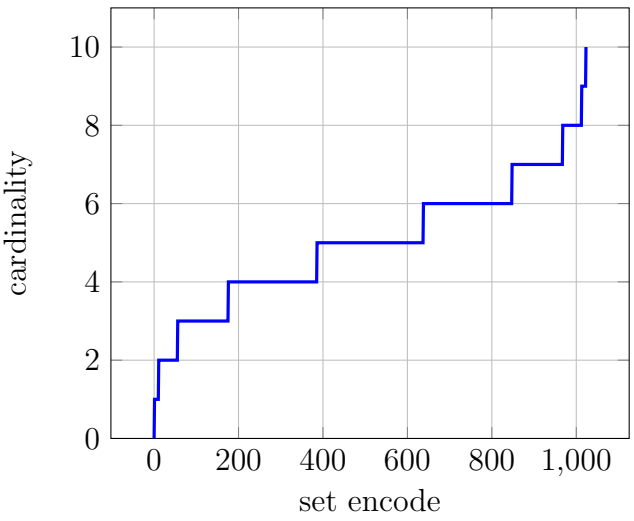


Figure 2.3: Cardinality in the CNS representation of a set

The following algorithm implements the direct transform (from the set S to

integer l).

Function LexicographicIndex(S, n)

input : S a set
input : n number of elements in the full set

$S \leftarrow$ sort the elements of S in ascending order
$s \leftarrow |S|$
$l \leftarrow 0$
for $i \in [1, s]$ **do**
  $c_i \leftarrow S[i]$
  $l \leftarrow l + \binom{c_i}{i}$
end
return l

To compute the inverse transform (from the integer l to the set S) we use a simple greedy algorithm.

Function LexicographicSet(l, n)

input : l an integer
input : n number of elements in the set

$r \leftarrow$ the largest integer such that $l \leq \binom{n}{r}$
$S \leftarrow \langle\rangle$
for $i \in [r, 1]$ **do**
  $c_i \leftarrow$ the largest integer such that $\binom{c_i}{i} \leq l$
  $S \leftarrow S \cup c_i$
  $l \leftarrow l - \binom{c_i}{i}$
end
return S

2.2 Set Functions

2.2.1 Definitions

A *set function* ξ on N is a mapping between 2^N and $\mathbb{R}$:

$$\xi : 2^N \to \mathbb{R}$$

It assigns a real value to each subset of N.

A set function can be:

- *monotone* if $\xi(A) \leq \xi(B)$ for $A \subseteq B$
- *grounded* if $\xi(\emptyset) = 0$
- *normalized* if $\xi(N) = 1$
- *non-negative* if $\xi(S) \geq 0$ for $\forall S \subseteq N$

For $\forall A, B \subseteq N$, and $A \cap B = \emptyset$, the set function can be:

- *additive* if $\xi(A \cup B) = \xi(A) + \xi(B)$
- *sub-additive* if $\xi(A \cup B) \leq \xi(A) + \xi(B)$
- *super-additive* if $\xi(A \cup B) \geq \xi(A) + \xi(B)$

For $\forall A, B \subseteq N$ (including also $A \cap B \neq \emptyset$), the set function can be:

- *modular* if $\xi(A \cup B) + \xi(A \cap B) = \xi(A) + \xi(B)$
- *sub-modular* if $\xi(A \cup B) + \xi(A \cap B) \leq \xi(A) + \xi(B)$
- *super-modular* if $\xi(A \cup B) + \xi(A \cap B) \geq \xi(A) + \xi(B)$

Note that the *modularity* is equivalent to the *additivity* when $A \cap B = \emptyset$.

There are some special terms for set functions with the selected properties:

- *measure*: a *grounded*, *non-negative*, *additive* set function
- *probability measure*: a *grounded*, *non-negative*, *additive*, *normalized* set function
- *capacity*: a *grounded*, *monotone* set function. The capacity can be *normalized*. Other terms for the capacity are: *fuzzy measure*, *non-additive measure*, *monotonic measure*.
- *game*: a *grounded* set function

2.2.2 Discrete Derivative

Let ξ a set function on N, a set $S \subseteq N$ and an element $i \in N$. The (first order) *derivative* of ξ on S with respect to i is defined as:

$$\Delta_i \xi(S) = \xi(S \cup i) - \xi(S \setminus i)$$

In this definition, it is irrelevant the fact that i is in S or not.

Some more common definitions are:

$$\Delta_i \xi(S) = \xi(S \cup i) - \xi(S)$$

if it is ensured that i is not member of S ($i \notin S$), and

$$\Delta_i \xi(S) = \xi(S) - \xi(S \setminus i)$$

if it is ensured that i is member of S ($i \in S$).

This value is also known as the *marginal contribution* of i in S with respect to ξ.

The *second order* derivative of ξ on S with respect to $\{i, j\}$ is defined as:

$$\begin{aligned}\Delta_{ij}\xi(S) &= \xi(S \cup ij) - \xi(S \setminus i \cup j) - \xi(S \setminus j \cup i) + \xi(S \setminus ij) \\ &= \Delta_i(\Delta_j\xi(S)) \\ &= \Delta_j(\Delta_i\xi(S))\end{aligned}$$

This is also known as the *net marginal contribution* of the pair $\{i, j\}$ on S with respect to ξ.

It is possible to generalize to the k^{th} *order* derivative, where k is the cardinality of the set $K \subseteq N$ of elements used to compute the derivative:

$$\Delta_K\xi(S) = \Delta_i\Big(\Delta_{K\setminus i}\xi(S)\Big) = \sum_{L\subseteq K}(-1)^{|K\setminus L|}\xi(S \setminus K \cup L)$$

with the convention

$$\Delta_\emptyset\xi(S) = \xi(S)$$

This is the *net marginal contribution* of the elements in K on S with respect to ξ.

Note that

$$\Delta_K\xi(S) \neq \xi(S \cup K) - \xi(S)$$

We can observe that

$$\xi(A \cup S) = \sum_{T\subseteq S}\Delta_T\xi(A)$$

and

$$\xi(S) = \sum_{T\subseteq S}\Delta_T\xi(\emptyset)$$

2.2.3 Set Functions representation

Let ξ a set function on N. There exist several methods to represent it [78]. The simplest method is to consider the function's value $\xi(S)$ directly, that is, to use a map between S and $\xi(S)$

$$S \to \xi(S)$$

The second method uses the *Möbius coefficients* (or *Harsanyi dividends*) a_T [76, 109]

$$\xi(S) = \sum_{T \subseteq S} a_T \quad S \subseteq N$$

were a_T is computed as

$$a_T = \sum_{S \subseteq T} (-1)^{t-s} \xi(S)$$

The two definitions form the *Möbius Transform*.

The Möbius representation of a set function is used very often. Several other representations are available and some of them are defined and used in the following chapters.

2.2.4 Set Functions Space

Let $\mathcal{L}(2^N)$ the space of the set functions defined on N. This is a 2^n-dimensional *linear space*. Let $\xi, \zeta \in \mathcal{L}(2^N)$, we can define

$$\begin{aligned} \Delta_{ij}\xi(S)(\xi + \zeta)(S) &= \xi(S) + \zeta(S) \quad \forall S \subseteq N \\ (\alpha\xi)(S) &= \alpha \cdot \xi(S) \quad \forall S \subseteq N, \forall \alpha \in \mathbb{R} \end{aligned}$$

The *inner product* is defined as

$$\langle \xi, \zeta \rangle = \sum_{S \subseteq N} \xi(S)\zeta(S)$$

and the *norm*, induced by the inner product, is defined as

$$\|\xi\| = \sqrt{\langle \xi, \xi \rangle}$$

Let μ a *probability distribution function*

$$\mu : 2^N \to [0, 1]$$
$$\sum_{S \subseteq N} \mu(S) = 1$$

with the condition that

$$\mu(S) > 0 \quad \forall S \subseteq N$$

the *weighted inner product* is defined as

$$\langle \xi, \zeta \rangle_\mu = \sum_{S \subseteq N} \mu(S)\xi(S)\zeta(S)$$

and the *weighted norm*, induced by the weighted inner product, is defined as

$$\|\xi\|_\mu = \sqrt{\langle \xi, \xi \rangle_\mu}$$

Note that if $\mu(S) = 0$ for some $S \neq \emptyset$ the function $\langle \cdot, \cdot \rangle_\mu$ may not have the properties of a *inner product*. However, if $\mu(S) = 0$ and $\xi(S) = 0$, properties are not lost.

2.3 Pseudo Boolean Functions

2.3.1 Definitions

As defined previously, a set function ξ on N is a function from the 2^N into $\mathbb{R}$

$$\xi : 2^N \to \mathbb{R}$$

Using the *indicator function*, each set $S \subseteq N$, can be mapped into a vector of bits. This allow to define the function also as a *pseudo boolean function* from $\{0,1\}^n$ into $\mathbb{R}$:

$$\begin{gathered} f_\xi : \{0,1\}^n \to \mathbb{R} \\ f_\xi(\mathbf{x}) = \xi(\{i \in N : x_i = 1\}) \end{gathered}$$

The inverse is

$$\xi_f(S) = f(\mathbf{1}_S)$$

Because there is an equivalence between $S \subseteq N$ and $\mathbf{x} \in \{0,1\}^n$, we can write $f(S)$ instead of $f(\mathbf{1}_S)$.

The space $\{0,1\}^n$ is a 2^n-dimensional *vector space* equivalent to the set functions' vector space [75].

For this space, there are several bases available.

2.3.2 Dirac basis

The most simple basis is *Dirac* basis [75], defined as

$$\delta_T(\mathbf{x}) = \prod_{i \in T} x_i \prod_{i \in N \setminus T} (1 - x_i)$$

$$\delta_T(S) = \prod_{i \in T} S_i \prod_{i \in N \setminus T} (1 - S_i)$$

where

$$S_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$

with the property

$$\delta_T(S) = \begin{cases} 1, & \text{if } T = S \\ 0, & \text{otherwise.} \end{cases}$$

(sometimes $1 - x_i$ is written as $\bar{x}_i$). Using the Dirac basis, the function can be defined as

$$f(\mathbf{x}) = \sum_{T \subseteq N} \xi(T) \delta_T(\mathbf{x})$$

This basis is *orthonormal*

$$\langle \delta_T, \delta_T \rangle = 1$$
$$\langle \delta_T, \delta_U \rangle = 0 \quad T \neq U$$

Other properties are available in the following table.

	value	note
$\delta_{\emptyset}(S)$	1	$\forall S \subseteq N$
$\delta_S(\emptyset)$	0	$S \supset \emptyset$
$\delta_S(S)$	1	
$\delta_S(T)$	0	$\forall T \neq S$
$\delta_S(T)\delta_S(T)$	$\delta_S(T)$	
$\delta_S(R)\delta_T(R)$	0	$\forall S \neq T$
$\Delta_i\delta_S(T)$	$(-1)^{\lvert\{i\}\setminus S\rvert}\delta_{S\setminus i}(T)$	
$\Delta_K\delta_S(T)$	$(-1)^{\lvert K\setminus S\rvert}\delta_{S\setminus K}(T)$	

Table 2.3: Dirac basis properties

2.3.3 Unanimity Game basis

Another useful basis is the *Unanimity Game* basis [75], defined as

$$e_T(\mathbf{x}) = \prod_{i \in T} x_i$$
$$e_T(S) = \prod_{i \in T} S_i$$

where

$$S_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$

The *Unanimity Game* basis has the property that

$$e_T(S) = \begin{cases} 1, & \text{if } S \supseteq T \\ 0, & \text{otherwise.} \end{cases}$$

Using this basis, the function can be defined as

$$f(\mathbf{x}) = \sum_{T \subseteq N} a_T e_T(\mathbf{x})$$

where a_T are the *Möbius coefficients*.

The basis **is not** *orthonormal*:

$$\langle e_T, e_T \rangle = \sum_{S \supseteq T} e_T(S) = 2^{n-t}$$
$$\langle e_T, e_U \rangle = \sum_{S \supseteq T \cup U} e_{T \cup U}(S) = 2^{n-|T \cup U|}$$

Other properties are available in the following table.

	value	note
$e_\emptyset(S)$	1	$\forall S \subseteq N$
$e_S(\emptyset)$	0	$S \neq \emptyset$
$e_S(S)$	1	
$e_S(T)$	1	$\forall T \supseteq S$
$e_S(T)e_S(T)$	$e_S(T)$	
$e_S(R)e_T(R)$	$e_{S \cup T}(R)$	
$\Delta_i e_S(T)$	$e_{S \setminus i}(T)$	$i \in S$
	0	$i \notin S$
$\Delta_K e_S(T)$	$e_{S \setminus K}(T)$	$S \cap K \neq \emptyset$
	0	$S \cap K = \emptyset$

Table 2.4: Unanimity Game basis properties

2.3.4 Walsh basis

Another basis is the *Walsh function* [75]. We define

$$z_i = 2x_i - 1 = \begin{cases} +1, & \text{if } x_i = 1 \\ -1, & \text{if } x_i = 0 \end{cases}$$

then, the function basis w_T is defined as

$$w_T(\mathbf{z}) = \prod_{i \in T} z_i$$

If we use S_i in a similar way as in $e_T(S)$ but with the convention that

$$S_i = \begin{cases} +1, & \text{if } i \in S \\ -1, & \text{otherwise} \end{cases}$$

the property of the basis is that

$$w_T(S) = \prod_{i \in T \cap S} S_i \prod_{i \in T \setminus S} = (-1)^{|T \setminus S|}$$

Note that:

$$\begin{aligned} w_S(\mathbf{z}) w_T(\mathbf{z}) &= \prod_{i \in S} z_i \prod_{i \in T} z_i \\ &= \prod_{i \in S \cap T} z_i \prod_{i \in S \setminus T} z_i \prod_{i \in T \setminus S} \\ &= \prod_{i \in S \setminus T} z_i \prod_{i \in T \setminus S} \\ &= w_{S \Delta T}(\mathbf{z}) \end{aligned}$$

Using the *weighted inner product* with the weight function $\mu(S) = \frac{1}{2^n}$, the basis is orthonormal

$$\langle w_T, w_T \rangle_\mu = 1$$
$$\langle w_T, w_U \rangle_\mu = 0$$

Using the Walsh functions, the function can be defined as

$$f(\mathbf{x}) = \sum_{T \subseteq N} \left(\sum_{S \supseteq T} \frac{a_S}{2^s} \right) w_T(\mathbf{z})$$

where a_S are the Möbius coefficients. Other properties are available in the following table.

	value	note
$w_\emptyset(S)$	1	
$w_S(\emptyset)$	$(-1)^s$	
$w_S(S)$	1	
$w_S(T)$	$(-1)^{\lvert T \setminus S \rvert}$	
$w_S(T)w_S(T)$	1	
$w_S(R)w_T(R)$	$w_{S \Delta T}(R)$	
$\Delta_i w_S(T)$	$w_{S \setminus i}(T)$	$i \in S$
	0	$i \notin S$
$\Delta_K w_S(T)$	$w_{S \setminus K}(T)$	$S \cap K \neq \emptyset$
	0	$S \cap K = \emptyset$

Table 2.5: Walsh function properties

2.3.5 Discrete Derivatives

The *first*-order derivative of a pseudo boolean function f on i is defined as

$$\Delta_i f(\mathbf{x}) = f(\mathbf{x})|_{x_i=1} - f(\mathbf{x})|_{x_i=0}$$

The *second*-order derivative (on ij) is defined as:

$$\begin{aligned} \Delta_{ij} f(\mathbf{x}) &= \Delta_{ji} f(\mathbf{x}) = \Delta_i \Delta_j f(\mathbf{x}) = \Delta_j \Delta_i f(\mathbf{x}) \\ &= f(\mathbf{x})|_{x_i=1,x_j=1} - f(\mathbf{x})|_{x_i=0,x_j=1} - f(\mathbf{x})|_{x_i=1,x_j=0} + f(\mathbf{x})|_{x_i=0,x_j=0} \end{aligned}$$

It is possible to generalize to the k^{th} order derivative as

$$\begin{aligned}\Delta_{i_1\dots i_k} f(\mathbf{x}) &= \Delta_{i_1} \cdots \Delta_{i_k} f(\mathbf{x}) \\ \Delta_K f(S) &= \sum_{T \subseteq K} (-1)^{|K \setminus T|} f(S \setminus K \cup T)\end{aligned}$$

2.4 Partition functions

2.4.1 Definitions

Let $R \subseteq N$, we define the *restricted set function ξ on R* as

$$\xi^R(T) = \xi(T \cap R)$$

A property of ξ^R is that the derivative on S, when $S \cap R = \emptyset$ is zero. For example, let $i \notin R$

$$\begin{aligned}\Delta_i \xi^R(T) &= \xi^R(T \cup i) - \xi^R(T) \\ &= \xi((T \cup i) \cap R) - \xi(T \cap R) \\ &= \xi(T \cap R) - \xi(T \cap R) \\ &= 0\end{aligned} \tag{2.2}$$

This property can be extended to all elements in S.

Let $\mathcal{P}^m(N)$ the partitions space of N in m blocks, and $P \in \mathcal{P}^m(N)$ a partition. We suppose to assign a set function ξ_r to each partition's block. We define the *partition function Ξ^P, induced by the set functions ξ_r and the partition P* as

$$\begin{aligned}\Xi^P &: \mathcal{P}^m(N) \to \mathbb{R} \\ \Xi^P(T) &= \sum_{r=1}^{m} \xi_r^{P_r}(T) = \sum_{r=1}^{m} \xi_r(T \cap P_r)\end{aligned}$$

2.4.2 Derivative on a partition

The main property of Ξ^P is that the derivative on S, when S is a subset of a partition's block P_t, for the 2.2, is equal to the derivative on ξ_t

$$\begin{aligned}\Delta_S \Xi^P(T) &= \sum_{r=1}^{m} \Delta_S \xi_r^{P_r}(T) \\ &= \Delta_S \xi_t^{P_t}(T) \\ &= \Delta_S \xi(T \cap P_t) \\ &= \Delta_S \xi(T)\end{aligned}$$

Now, we must define the *mixed derivative*, the derivative on S when S has not empty intersection with some partition's blocks. It can be defined as

$$\begin{aligned}\Delta_S \Xi^P(T) &= \sum_{r=1}^{m} \Delta_S \xi_r^{P_r}(T) \\ &= \sum_{\substack{r=1 \\ P_r \cap S \neq \emptyset}}^{m} \Delta_S \xi_r^{P_r}(T) \\ &= \sum_{\substack{r=1 \\ P_r \cap S \neq \emptyset}}^{m} \Delta_S \xi_r(T \cap P_r)\end{aligned} \tag{2.3}$$

If we compute the derivative on $\{i, j\}$, with $i \in P_s$ and $j \in P_t$, we have

$$\begin{aligned}\Delta_{ij} \Xi^P(T) &= \sum_{r=1}^{m} \Delta_S \xi_r^{P_r}(T) \\ &= \Delta_i \xi_r^{P_r}(T) + \Delta_j \xi_s^{P_s}(T) \\ &= \Delta_i \xi_r(T \cap P_r) + \Delta_j \xi_s(T \cap P_s)\end{aligned}$$

2.4.3 Mixed derivative

The previous definitions of derivative is based on a selected partition P. We must consider all partitions (with m blocks) but with an extra constraint: we

want to select the blocks where S (the set used to compute the derivative) has a not empty intersection.

For now, let $S = \{i, j\}$ and we want to use i with P_r and j with P_s. We use the symbol

$$\Delta_{ij}^{rs}$$

with the meaning:

- i is used *only* with the block P_r: it is added to, and removed from P_r
- j is used *only* with the block P_s: it is added to, and removed from P_s

Using this symbol, we define the *mixed derivative on* $i \in P_r, j \in P_s$ *and the permutation* P as

$$\begin{aligned}\Delta_{ij}^{rs}\Xi^P(T) &= \sum_{t=1}^{m} \Delta_{ij}^{rs}\xi_t^{P_t}(T) \\ &= \Delta_i\xi_r^{P_r}(T) + \Delta_j\xi_s^{P_s}(T) \\ &= \Delta_i\xi_r(T \cap P_r) + \Delta_j\xi_s(T \cap P_s)\end{aligned}$$

The *mixed derivative on* $i \in P_r, j \in P_s$ can be defined as

$$\Delta_{ij}^{rs}\Xi(T) = \sum_{P \in \mathcal{P}^m(N \setminus ij)} p_{ij}(P)\Delta_{ij}^{rs}\Xi^P(T)$$

where

$$\langle p_{ij}(P) : P \in \mathcal{P}^m(N \setminus ij)\rangle$$

is a probability distribution. The most simple distribution is

$$p_{ij}(P) = \frac{1}{|\mathcal{P}^m(N \setminus ij)|} = \frac{1}{\left\{{n-2 \atop m}\right\}}$$

This definition has two properties

- it has a direction
- the partition's blocks used are only the blocks with not empty intersection with the set used for the derivative.

Its generalization is

$$\Delta_K \Xi(T) = \sum_{P \in \mathcal{P}^m(N)} p_K(P) \Delta_K \Xi^P(T)$$

where

$$\Delta_K \Xi^P(T) = \sum_{i=1}^{m} \Delta_{K \cap P_i} \xi_i(T \cap P_i)$$

2.5 Graphs

A graph $G = (V, E, w)$ [178] is an mathematical object composed by

1. $V = \{v_1, \ldots, v_n\}$ a set of *vertices*
2. $E = \{e_{ij} = (v_i, v_j)\}$ a set of *edges*, where an *edge* describes the existence of a relation between two vertices
3. $w : V \cup E \to \mathbb{R}$ a function that assign a *weight* to vertices and edges. The most simple function assign 1 to all objects

A edge $e_{ij} = (v_i, v_j)$ can be

- an *undirected edge* if it describes a *symmetric* relation: $e_{ij} = e_{ji}$. In this case, v_i and v_j are the *endpoints* and can be extracted from e_{ij} with $v_1(e_{ij}) \to v_i$ and $v_2(e_{ij}) \to v_j$

- a *directed edge* if it has a direction (the relation is not symmetric), that is $e_{ij} \neq e_{ji}$. In this case, v_i is the *tail* and v_j the *head* of the edge. They can be extracted with $tail(e_{ij}) \rightarrow v_i$ and $head(e_{ij}) \rightarrow v_j$
- a *loop* (or a *self-edge*) if the two vertices are the same: $e_{ii} = (v_i, v_i)$

A graph is

- an *undirected graph* if it contains only undirected edges
- a *directed graph* if it contains only directed edges
- a *simple graph* if it contains only undirected edges and there is at most one edge between two vertices
- a *complete graph* if each vertex is connected to all the others
- a *multi graph* if there multiple edges between two vertices
- a *pseudo graph* if it contains loops or multiple edges

An *undirected graph* can be converted in a *directed graph* replacing the *undirected edges* with two *directed edges* in the opposite directions.

A graph can be modelled in several way. A common method is to use the *adjacent matrix* A, a squared matrix where the rows and the columns are the vertices and the element at the coordinates A_{ij} contains the weight of the edge e_{ij}. The matrix has several properties:

- the element at the coordinates ij is 0 (zero) if there is not an edge between the vertices v_i and v_j
- it is *symmetric* if the graph is undirected
- the elements of the diagonal are 0 (zero) if the graph has no loops

The matrix creates a bridge between the world of the graphs and the world of the linear algebra. The *Spectral Graph Theory* studies the properties of the graphs analyzing the properties of the adjacent matrices [187, 188].

2.6 Conclusions

In this chapter, we introduced the standard terminology and notation used in the rest of the document. We started with the standard notation for numbers, sets and set operations. Then, we described the subset lattice, the objects' domain of this work. We described some representations of sets, useful in the algorithms' implementation. We also defined the concepts of set function, pseudo boolean function, the relation between the two functions' classes, the concept of derivative and some basis used to represent the pseudo boolean functions. We have seen that the basis have specific properties. The demonstration of these properties and some representation transformations are available in the appendices.

Chapter 3

Coalitional Game Theory

3.1 Introduction

Coalitional Game Theory with *Transferable Utilities* (TU) [162, 163] is the study of the statics and dynamics of *coalitions of players* when the coalition's worth (e.g. the *value* generated by the coalition) can be divided and distributed, in some way, as *payoff* (a *reward*), to the members.

This branch of Game Theory focuses on the problems related to the coalition formation and stability and on the mechanisms to distribute the worth to the coalition's members. In the terminology of set functions, a *coalition* is a subset S of a set N (the *grand coalition*), the *players* are the set's members.

If the worth of the coalition depends only on the members, it can be modelled by a set function (the *utility function*, that in this context is called *characteristic function*). If the worth of a coalition depends also on the way in which the non-members are organized, it can be modeled in general using partition functions.

There exist several methods to distribute the worth to individual players (i.e. to assign a *value* to a player), typically aiming at a fair division goal. Most methods are based on the concept of power index or, in addition, to a generalization of the concept to the concept of interaction index. [1]

[1] In most of them, to determine the value to be assigned to a member one evaluates the *marginal contribution* of the player in each possible coalition, then computes a *weighted mean* of the contributions. The different power indices definitions follow this pattern: they

There exist different types of power and interaction indices [171]: we are interested only in the class of *probabilistic values* [81] and *probabilistic interaction indices* [85] that includes some of the most important power indices: **Shapley Value** [79,82], **Shapley Interaction Index** [96], **Banzhaf Value** [80], **Banzhaf Interaction Index** [89], **Chaining Interaction Index** [86].

3.2 Coalitional Games

Let N a set of *players*, a *coalitional game with transferable utilities* G (also *TU game*, or simply *game*) on N is a pair composed by N (the *grand coalition*) and a set function ξ (the *characteristic function*) that maps each subset $S \subseteq N$ (the *coalition*) to a real value (the *worth* of S). The function must be *grounded* ($\xi(\emptyset) = 0$), *non-negative* ($\xi(S) \geq 0 \quad \forall S \subseteq N$). Sometimes, but it is not mandatory, *normalized* ($\xi(N) = 1$).

$$\begin{aligned} G &= \langle N, \xi \rangle \\ \xi &: 2^N \to \mathbb{R} \\ \xi(\emptyset) &= 0 \\ \xi(S) &\geq 0 \quad \forall S \subseteq N \end{aligned}$$

The set function can have other properties (super/sub/-additive, super/sub/-modular, monotone, normalized, etc), but this is not necessary.

The space of all games on N (all set functions ξ defined on N) is denoted by $\mathcal{G}^N$.

---

are specified by different domain dependent constraints.

Based on the different coalitions' worth, it is also possible to evaluate the *interaction* between the subsets of players, that measures the degree by which the players collaborate or interfere when all of them are in the same coalition. Most of the time

- the method used to evaluate the interaction index is valid also for a single player, that is, the power index is a special case of the interaction index
- it is possible to recreate the original function using the interaction indices evaluated on all subsets of the grand coalition N

since the interaction indices are defined using the function and the function can be recreated using the interaction indices, this bidirectional mapping defines an *interaction transform*.

3.3 Power Indices

For the class of coalitional games $\mathcal{G}^N$, a *solution concept* is a method to divide the worth of the coalitions among their members

$$\mathcal{S} : \mathcal{G}^N \to \mathbb{R}^n$$

The *solution* of the solution concept is the *power index*, a vector that assign to each player a *score* based on the influence that the player hold inside the coalitions.

Two well known power indices are the **Shapley Value** and the **Banzhaf Value**: they are instances of the more larger class of *probabilistic values* [81].

The solution concept does not only assign a score to a single players. It is possible to define rules to assign scores to each possible player's subset. For those cases one uses the term *interaction index*. The class of solutions related to the probabilistic values is the class of *probabilistic interaction indices* [85].

3.3.1 Probabilistic Values

The *probabilistic value* [81] of a game $G = \langle N, \xi \rangle$ is a class of solution concepts based on the general formula

$$\phi_\xi(i) = \sum_{T \subseteq N \setminus i} p_i(T) \Delta_i \xi(T) \tag{3.1}$$

with $\langle p_i(T) : T \subseteq N \setminus i \rangle$ a probability distribution.

That is:

- $p_i(T)$ is the weight assigned to the coalition T
- $\Delta_i \xi(T)$ is the marginal contribution of i inside the coalition T
- $\phi_\xi(i)$ is the *expected payoff* of the i's marginal contributions

It is useful to change slightly the definition as follows

$$\phi_\xi(i) = \sum_{t=0}^{n-1} \phi_\xi(t,i)$$

$$\phi_\xi(t,i) = \sum_{\substack{T \in N \setminus i \\ |T|=t}} p_i(T) \Delta_i \xi(T)$$

where $\phi_\xi(t,i)$ is the component of the probabilistic value evaluated on the coalitions with cardinality t. This allow to consider the power index as a sum of indices evaluated on sets with selected cardinality.

The probability distribution $\langle p_i \rangle$, used in the definition, is not endowed with a special structure: all 2^{n-1} values can be different. If the number of players is very high, it is impossible to obtain all possible values. A solution is to assign a structure to the distribution such that the value for each coalition can be computed using a function and a small number of parameters.

The literature describes two simple structures:

1. *cardinal-probabilistic values* [118]: the distribution depends only on the coalition cardinality
2. *player-probabilistic values* [102]: the distribution depends only on the coalition members

3.3.2 Cardinal-Probabilistic Values

The *cardinal-probabilistic value* is a class of probabilistic values where the weight assigned to a coalition depends only on the coalition cardinality [118]:

$$p_i(T) = p_i(t)$$

with

$$\sum_{T \subseteq N \setminus i} p_i(T) = \sum_{t=0}^{n-1} \binom{n-1}{t} p_i(t) = 1$$

Note that the definition uses n probability distributions (one for each player) and each distribution is composed of $n-1$ values, that is, we need $n(n-1)$ values. In general this is not necessary: all players can use the same probability distribution.

3.3.3 Shapley Value

The most famous member of cardinal-probabilistic value is the **Shapley Value**, introduced by Lloyd Shapley in the 1953 [79, 82], defined as

$$\phi_{\xi}^{Sh}(i) = \sum_{T \subseteq N \setminus i} \frac{t!(n-1-t)!}{n!} \Delta_i \xi(T) \tag{3.2}$$

where the weight $p_i(t)$ is

$$p_i(t) = \frac{t!(n-1-t)!}{n!}$$

This definition can be interpreted in the following way: the weight is composed by two factors:

$$p_i(t) = \frac{t!(n-1-t)!}{n(n-1)!} = \frac{1}{n} \cdot \binom{n-1}{t}^{-1}$$

1. the first factor $(1/n)$ is the probability to select a coalition with cardinality t: there are n different cardinality (from 0 to $n-1$) and the probability is *uniform* (same probability for all cardinality)
2. the second factor is the probability to select a coalition with the selected cardinality: there are $\binom{n-1}{t}$ coalitions with cardinality t

There exists another definition based on permutations:

$$\phi_{\xi}^{Sh}(i) = \sum_{\pi \in \Pi(N)} \frac{1}{n!} \Delta_i \xi(S_i(\pi))$$

where $S_i(\pi)$ is the set compose by the permutation's elements that precede the element i.

The definition can be interpreted in the following way: we divide the permutation in three parts:

$S_i(\pi)$	i	$R_i(\pi)$

Figure 3.1: Parts of a permutation

1. $S_i(\pi)$: the part *before* the element i
2. i: the part composed *only* by i
3. $R_i(\pi)$: the remainder part *after* the element i

With $S_i(\pi) = T$, the weight is composed by:

- $t!$: number of different configurations of $S_i(\pi)$
- $(n-t-1)!$: number of different configurations of $R_i(\pi)$, set composed by $N \setminus (S_i(\pi) \cup i)$
- $n!$: total number of permutations

this is the number of coalitions having i in that specific position.

3.3.4 Chaining Value

Another example of cardinal-probabilistic value is the **Chaining Value** introduced by Marichal et al. in [86], and defined as

$$\phi_\xi^{Ch}(i) = \sum_{T \subseteq N \setminus i} \frac{t!(n-1-t)!}{n!} \Delta_i \xi(T) \tag{3.3}$$

Note that this definition is the same as the Shapley Value. As we will see later, the two indices differ on the *interaction indices*' definitions.

3.3.5 Player-probabilistic values

The *player-probabilistic value* is a class of probabilistic values introduced by Marichal et al. in [102] with the name **Weighted Banzhaf Value**. The reason of the more standard denomination *player-probabilistic value* is the following: in this case each player is assigned a probability to join a coalition, and the weight of the coalitions are just the product of the individual player probabilities.

Let $\langle p_j : j \in N \rangle$ the *probability* of participation. We assume that those probabilities are independent from one another. The probability assigned to a coalition T, containing i, is

$$p_i(T) = \prod_{k \in T} p_k \prod_{k \in N \setminus i \setminus T} \Big(1 - p_k\Big) \tag{3.4}$$

It is straightforward to show that $\langle p_i(T) : T \subseteq N \setminus i \rangle$ is a probability distribution. Indeed, we can remove the element j from the expression:

$$\begin{aligned}
\sum_{T \subseteq N \setminus i} p_i(T) &= \sum_{T \subseteq N \setminus i} \prod_{k \in T} p_k \prod_{k \in N \setminus i \setminus T} \Big(1 - p_k\Big) \\
&= \sum_{T \subseteq N \setminus ij} p_j \prod_{k \in T} p_k \prod_{k \in N \setminus ij \setminus T} \Big(1 - p_k\Big) \\
&+ \sum_{T \subseteq N \setminus ij} (1 - p_j) \prod_{k \in T} p_k \prod_{k \in N \setminus ij \setminus T} \Big(1 - p_k\Big) \\
&= \sum_{T \subseteq N \setminus ij} (p_j + 1 - p_j) \prod_{k \in T} p_k \prod_{k \in N \setminus ij \setminus T} \Big(1 - p_k\Big) \\
&= \sum_{T \subseteq N \setminus ij} \prod_{k \in T} p_k \prod_{k \in N \setminus ij \setminus T} \Big(1 - p_k\Big)
\end{aligned}$$

Using the same method, we can remove all elements, with result 1.

3.3.6 Banzhaf Value

The most famous member of player-probabilistic values is the **Banzhaf Value** [80], defined as

$$\phi_\xi^B(i) = \sum_{T \subseteq N \setminus i} p_i(T) \Delta_i \xi(T) \tag{3.5}$$

with

$$p_i(T) = \frac{1}{2^{n-1}}$$

This value is exactly the value obtained using $p_i = \frac{1}{2}$ for each player

$$p_i(T) = \prod_{j \in T} \frac{1}{2} \prod_{j \in N \setminus i \setminus T} \left(1 - \frac{1}{2}\right) = \prod_{j \in N \setminus i} \frac{1}{2} = \frac{1}{2^{n-1}}$$

3.3.7 Weighted Banzhaf Value

The **Weighted Banzhaf Value** is the generalization of the **Banzhaf Value**, and has the same definition of the *player probabilistic value*

$$\phi_\xi^{\mathcal{P}}(i) = \sum_{T \subseteq N \setminus i} p_i(T) \Delta_i \xi(T) \tag{3.6}$$

with the weights defined as in 3.4

3.4 Interaction Indices

The *interaction indices* [85,86,88,88,89,91,102] are the extensions of 2^{nd} and higher degree of the correspondent power indices.

To understand the usefullness and the meaning of a interaction index, we consider two players i and j, and we compare their behaviour as single players and as pair [118].

We consider the function's value of the pair compared to the function's value of the single players. We can have

$$\xi(\{i,j\}) = \xi(i) + \xi(j)$$

the players are *independent*, because the value of the pair is exactly the sum of the single players' values.

Alternatively, we can have

$$\xi(\{i,j\}) < \xi(i) + \xi(j)$$

the players have a *redundant* collaboration, or they *interfere* because their value as couple is less than the values as single elements. These means that, if possible, it is better to keep the players separated.

Otherwise, we can have

$$\xi(\{i,j\}) > \xi(i) + \xi(j)$$

the player *collaborates* constructively, that is, the value of the pair is great than the single players' value. In this case, if possible, it is better to keep the players together.

The coefficient measuring the interaction can be evaluated as

$$\xi(\{i,j\}) - (\xi(i) + \xi(j))$$

But this coefficient consider only the pair and the single players. Instead we need to consider the behaviour of i and j when they join the coalitions $S \subseteq N \setminus ij$. That is, we are interested on the marginal contribution of j when i is present minus the marginal contribution of j when i is not present, in presence of the players in S. Because the marginal contribution of a player h is $\Delta_h \xi(S) = \xi(S \cup h) - \xi(S)$, the interaction between i and j can be measured as

$$\Big(\xi(S \cup ij) - \xi(S \cup i)\Big) - \Big(\xi(S \cup j) - \xi(S)\Big)$$

but this value is exactly the derivative on ij:

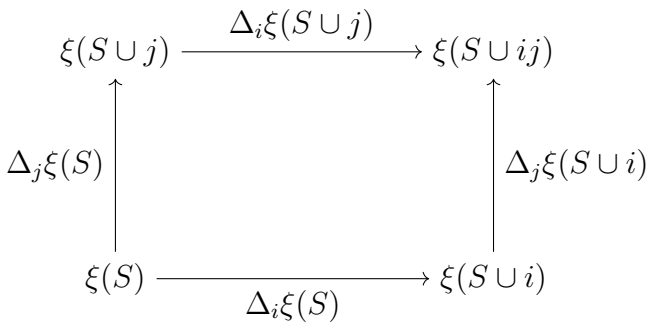


Figure 3.2: Interactions between i and j in the coalition S

$$\Delta_j\xi(S \cup i) - \Delta_j\xi(S) = \Delta_i\Big(\Delta_j\xi(S)\Big) = \Delta_{ij}\xi(S)$$

The next step is to compute a *weighted mean* of this contribution in presence of all possible coalitions $S \subseteq N \setminus ij$

$$I_\xi(ij) = \sum_{T \subseteq N \setminus ij} p_{ij}(T)\Delta_{ij}\xi(T)$$

For extension, it is possible to define the interactions among all players' subsets

$$I_\xi(S) = \sum_{T \subseteq N \setminus S} p_S(T)\Delta_S\xi(T) \quad \forall S \subseteq N$$

As before, we are interested into the *probabilistic interaction indices*, that is, indices where the probability distributions $\langle p_{ij} \rangle$ and $\langle p_S \rangle$ have a structure.

3.4.1 Probabilistic Interaction Indices

By analogy with the *probabilistic values*, a *probabilistic interaction index* is defined as [118]

$$I_\xi(S) = \sum_{T \subseteq N \setminus ij} p_S(T)\Delta_S\xi(T) \quad \forall S \subseteq N \tag{3.7}$$

with $\langle p_S(T) : T \subseteq 2^{N \setminus S} \rangle$ a probability distribution. That is:

- $p_S(T)$ is the weight assigned to the coalition T
- $\Delta_S \xi(T)$ is the marginal contribution of players in S in presence of the players in T
- $I_\xi(S)$ is the *expected payoff* of the S's marginal contributions

It is useful to change a little the definition in the following way

$$I_\xi(S) = \sum_{t=0}^{n-s} I_\xi(t, S)$$

$$I_\xi(t, S) \sum_{\substack{T \subseteq N \setminus S \\ |T|=t}} p_S(T) \Delta_S \xi(T)$$

where $I_\xi(t, S)$ is the component of the index evaluated on the coalitions with cardinality t.

Adding the same structures as before to $\langle p_S \rangle$, we can define two sub-classes:

1. *cardinal-probabilistic interaction indices* [118]: the weight depends only on the coalition cardinality
2. *player-probabilistic interaction indices* [102]: the weight depends only on the coalition members

3.4.2 Cardinal-Probabilistic Interaction Index

The *cardinal-probabilistic interaction index* is the class of interaction indices where the weight assigned to each coalition depends only on the coalition cardinality

$$p_S(T) = p_S(t)$$

with

$$\sum_{T \subseteq N \setminus S} p_S(T) = \sum_{t=0}^{n-s} \binom{n-s}{t} p_S(t) = 1$$

3.4.3 Shapley Interaction Index

The most famous member of this class is the **Shapley Interaction Index**. It is the extension of the Shapley Value, introduced in [84, 118] and defined as

$$I_\xi^{Sh}(S) = \sum_{T \subseteq N \setminus S} p_S(T) \Delta_T \xi(S) \tag{3.8}$$

with

$$p_S(T) = \frac{(n-s-t)!t!}{(n-s+1)!} = \frac{1}{n-s+1} \binom{n-s}{t}^{-1}$$

where

$$\frac{1}{n-s+1}$$

is the probability to select a level in the range $[0, n-s]$.

3.4.4 Chaining Interaction Index

The *Chaining Interaction Index* [86] is another member of this class. It is defined as

$$I_\xi^{Ch}(S) = \sum_{T \subseteq N \setminus S} p_S(T) \Delta_T \xi(S) \tag{3.9}$$

with

$$p_S(T) = \frac{s(n-s-t)!(s+t-1)!}{n!} = \binom{s-1+t}{s-1}\binom{n-s}{t}^{-1}$$

It differs by the Shapley Interaction Index on the probability to select a set's cardinality, that it is

$$\binom{s-1+t}{s-1}$$

3.4.5 Player-Probabilistic Interaction Index

The *player-probabilistic interaction index* is the extension to a generic player's coalition of the player-probabilistic value. It was introduced by Marichal et al in [86] with the name **Weighted Banzhaf Interaction Index**.

The weight assigned to each coalition is computed as in the player-probabilistic value:

$$p_S(T) = \prod_{j \in T} p_j \prod_{j \in N \setminus S \setminus T} (1 - p_j) \tag{3.10}$$

3.4.6 Banzhaf Interaction Index

The most famous member of this class in the **Banzhaf Interaction Index**. It is the extension of the Banzhaf Value, introduced in [115,118], and defined as

$$I^B_\xi(S) = \sum_{T \subseteq N \setminus S} p_S(T) \Delta_T \xi(S) \tag{3.11}$$

with

$$p_S(T) = \frac{1}{2^{n-s}}$$

3.4.7 Weighted Banzhaf Interaction Index

The **Weighted Banzhaf Interaction Index** is the generalization of the **Banzhaf Interaction Index** and it is defined as

$$I_\xi^{\mathcal{P}}(S) = \sum_{T \subseteq N \setminus S} p_S(T) \Delta_T \xi(S) \tag{3.12}$$

with the weights defined in Section 3.10.

3.5 Set Functions Transforms

The Möbius Transform, defined in 2.2.3, is only one of several possible transforms. Several other transformations are available, as, for example, the **Co-Möbius Transform** [167], the **Fourier Transform** [75], and the transforms based in power and interaction indices.

A *transform* $\mathcal{T}$ is an *invertible* high-order function such that, for each $\xi \in \mathcal{L}(2^N)$ exists a function ζ such that

$$\zeta = \mathcal{T}(\xi)$$
$$\xi = \mathcal{T}^{-1}(\zeta)$$

The transform can be defined in terms of the function or its derivatives. We are interested *only* on the definitions based on the derivatives: this because its implementation is more efficient.

The function used to compute the Möbius coefficients is the *Möbius Transform* [109]. It is a set function defined as

$$m_\xi(S) = \sum_{T \subseteq S} (-1)^{s-t} \xi(T) = a_S \tag{3.13}$$

The original set function can be recreated summing the Möbius coefficients assigned to the subsets of the set where the function must be evaluated

$$\xi(S) = \sum_{T \subseteq S} m_\xi(T) \tag{3.14}$$

The *first*-order derivative, for $S \subseteq N \setminus i$, can be expressed in terms of Möbius coefficients

$$\begin{aligned}
\Delta_i \xi(S) &= \xi(S \cup i) - \xi(S) \\
&= \sum_{T \subseteq S \cup i} m_\xi(T) - \sum_{T \subseteq S} m_\xi(T) \\
&= \sum_{T \subseteq S} m_\xi(T) + \sum_{T \subseteq S} m_\xi(T \cup i) - \sum_{T \subseteq S} m_\xi(T) \\
&= \sum_{T \subseteq S} m_\xi(T \cup i)
\end{aligned}$$

The k-order derivative, for $S \subseteq N \setminus K$, is

$$\Delta_K \xi(S) = \sum_{T \subseteq S} m_\xi(T \cup K) \quad S \cap K = \emptyset$$

3.5.1 Function Transforms based on Interaction Indices

In the previous paragraphs, we defined several power and interaction indices, based on the set function ξ. We have seen that the interaction indices are the generalization of the correspondent power index, and the power index is the special case of interaction index when the last one is applied to the set with only one element.

Starting from interaction indices evaluated on all possible subsets, it is possible to recreate the original function using an expression like the following

$$\xi(S) = \sum_{T \subseteq N} b(S,T) \cdot I_\xi(T)$$

where $b(S,T)$ can be considered as a *basis* of $\mathcal{L}(2^N)$.

3.5.2 Shapley Transform

The **Shapley Transform** is composed by the Shapley Interaction Indices, defined in Section 3.2 and Section 3.8, and the inverse transform, defined as

$$\xi(S) = \sum_{T \subseteq N} \beta^{|K|}_{|S \cap T|} \cdot I^{Sh}_\xi(T) \tag{3.15}$$

with

$$b(S,T) = \beta^s_t = \sum_{j=0}^{t} \binom{t}{k} B_{s-j} \quad t \leq s$$

where B_m is the m-th *Bernoulli number*.

The *Bernoulli number* is defined as [17, 96, 175]

$$B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0, B_4 = -\frac{1}{30}, B_5 = 0, B_6 = -\frac{1}{42}, \ldots$$

and, more in general, as:

$$B_m = 1 - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k}{m-k+1} = -\frac{1}{m+1} \sum_{k=0}^{m-1} \binom{m+1}{k} B_k$$

Note: the relation between the Bernulli numbers and the Shapley Interaction Indices can be found in **??** and [161], section 2.12.5, pag. 65.

3.5.3 Chaining Transform

The **Chaining Transform** is composed by Chaining Interaction Indices, defined in Section 3.3 and Section 3.9, and the inverse transform, defined as

$$\xi(S) = \sum_{T \subseteq N} \left(\frac{1}{t} \sum_{j=0}^{|T \cap S|} \left(\binom{|T \cap S|}{j} (-1)^{t-j} j \right) \right) \cdot I_\xi^{Ch}(T) \tag{3.16}$$

with

$$b(S,T) = \frac{1}{t} \sum_{j=0}^{|T \cap S|} \left(\binom{|T \cap S|}{j} (-1)^{t-j} j \right)$$

3.5.4 Banzhaf Transform

The **Banzhaf Transform** is composed by Banzhaf Interaction Indices, defined in Sections 3.5 and 3.11, and the inverse transform, defined as

$$\xi(S) = \sum_{T \subseteq N} \frac{1}{2^t} (-1)^{|T \setminus S|} \cdot I_\xi^B(T) \tag{3.17}$$

with

$$b(S,T) = \frac{1}{2^t} (-1)^{|T \setminus S|}$$

3.5.5 Weighted Banzhaf Transform

The **Weighted Banzhaf Transform** is composed by Banzhaf Interaction Indices, defined in Sections 3.6 and 3.12, and the inverse transform, defined as

$$\xi(S) = \sum_{T \subseteq N} \prod_{i \in T} (S_i - p_i) \cdot I_\xi^{\mathcal{P}}(T) \tag{3.18}$$

where

$$S_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$

with

$$b(S,T) = \prod_{i \in T} (S_i - p_i)$$

3.6 Axiomatization of Power and Interaction Indices

The power and interaction indices are *solution concepts* for the games in $\mathcal{G}^N$, that is, they are *methods* to distribute the worth of the coalitions to the players. These methods must have some *properties* as, for example, the *linearity*.

It is possible to demonstrate that, with specific properties (defined as mathematical *axioms*), the solution concept **must be** a specific power or interaction index [115, 118].

The list of axioms is not unique: there are several possible alternatives [112, 113, 120]. However, we are interested only on the more *classic* ones.

3.6.1 Definitions

A *dummy player* i is a player such that

$$\xi(S \cup i) - \xi(S) = \xi(i) \quad \forall S \subseteq N \setminus i$$

that is

$$\Delta_i \xi(S) = \xi(i) \quad \forall S \subseteq N \setminus i$$

A *null playeri* is a dummy player with $\Delta_i \xi(S) = 0$

Two players $\{i, j\}$ are *symmetric* if

$$\xi(S \cup i) = \xi(S \cup j) \quad \forall S \subseteq N \setminus ij$$

that is

$$\Delta_i \xi(S) = \Delta_j \xi(S) \quad \forall S \subseteq N \setminus ij$$

For extension, a *dummy coalition* P is a coalition such that

$$\xi(S \cup P) - \xi(S) = \xi(P) \quad \forall S \subseteq N \setminus P$$

and a *null coalition* P is a dummy coalition with $\xi(P) = 0$.

A *partnership* is a coalition P such as

$$\xi(S \cup T) - \xi(S) = 0 \quad \forall T \subset P$$

that is, if S does not contain all members of P, the worth of the coalition $S \cup T$ does not change.

A *permuted game*, denoted by $\pi\xi$ (it is a name composed by two symbols) is the game defined as

$$\pi\xi(\pi(S)) = \xi(S) \quad \forall S \subseteq N$$

where π is the *permutation function*:

$$\pi : N \to N$$
$$\pi(S) = \langle \pi(i) : i \in S \rangle$$

A *restricted game* ξ^R on $R \subset N$ is a game member of $\mathcal{G}^R$ such that

$$\xi^R(S) = \xi(S) \quad \forall S \subseteq R$$

Let P a coalition of players, and $\xi \in \mathcal{G}^N$ a game. A *reduced game* $\xi_{[P]}$ with respect to P is a game defined as follow:

- let $[P]$ a new symbol and $N_{[P]}$ the new grand coalition defined as
$$N_{[P]} = (N \setminus P) \cup \{[P]\}$$
- let $\xi_{[P]}$ a new game in $\mathcal{G}^{N_{[P]}}$, defined as follow:
$$\xi_{[P]}(S) = \begin{cases} \xi(S), & \text{if } [P] \notin S \\ \xi(S \cup P), & \text{otherwise.} \end{cases}$$

An alternative definition is

$$\xi_{[P]}(S) = \begin{cases} \xi(S), & \text{if } S \cap P = \emptyset \\ \xi(S \cup P), & \text{otherwise.} \end{cases}$$

Note that $N_{[P]}$ contains $n - p + 1$ elements.

That is, the function $\xi_{[P]}$ considers each subset of P as representative of P: if some player of P is in S, the function automatically aggregates all players in P.

A power index, defined as

$$\phi_\xi(i) = \sum_{T \subseteq N \setminus i} \Big(\xi(S \cup i) - \xi(S) \Big)$$

when applied to $[P]$ became

$$\phi_{\xi_{[P]}}([P]) = \sum_{T \subseteq N \setminus P} \Big(\xi(S \cup P) - \xi(S) \Big)$$

Remember that $\xi(S \cup P) - \xi(S) \neq \Delta_P \xi(S)$.

3.6.2 Axioms for power indices

Let $\xi, \zeta \in \mathcal{G}^N$ two games, and ϕ_ν a *power index*. Some of the most used *axioms* [81, 115, 118] are

- *linearity* (L): ϕ_ν is a linear function on $\mathcal{G}^N$, that is

$$\phi_{\xi+\zeta} = \phi_\xi + \phi_\zeta$$

$$\phi_{c\cdot\xi} = c \cdot \phi_\xi$$

- *dummy* (D): if i is a *dummy player*, then

$$\phi_\xi(i) = \xi(i)$$

- *monotonicity* (M): if ξ is monotonic, then

$$\phi_\xi(i) \geq 0 \quad \forall i \in N$$

- *symmetry* (S): for $\forall \xi \in \mathcal{G}^N$, for $\forall \pi \in \Pi(N)$, we have

$$\phi_\xi(i) = \phi_{\pi\xi}(\pi(i))$$

- *efficiency* (E): for $\forall \xi \in \mathcal{G}^N$

$$\sum_{i \in N} \phi_\xi(i) = \xi(N)$$

- *2-efficiency* (2-E), a.k.a. *delegation neutrality* or *collusion neutrality*: for $\forall i, j \in N$, we have

$$\phi_{\xi_{[ij]}}([ij]) = \phi_\xi(i) + \phi_\xi(j)$$

- *equal treatment* (ET) or *marginal contributions* (M) if i and j are *symmetric*, then

$$\phi_\xi(i) = \phi_\xi(j)$$

3.6.3 Axioms for interaction indices

Let I_ν an *interaction index*. Some of the most used *axioms* [81, 115, 118] are

- *linearity* (LI): I_ν is a linear function on $\mathcal{G}^N$, that is
$$I_{\xi+\zeta} = I_\xi + I_\zeta$$
$$I_{c\cdot\xi} = c \cdot I_\xi$$
- *dummy* (DI): if i is a *dummy player*, then
$$I_\xi(S \cup i) = 0 \quad \forall \emptyset \subset S \subseteq N \setminus i$$
$$I_\xi(\{i\}) = \xi(i)$$
- *dummy partnership* (DP): if P is a *dummy partnership*, then
$$I_\xi(S \cup P) = 0 \quad \forall \emptyset \subset S \subseteq N \setminus P$$
$$I_\xi(P) = \xi(P)$$
This axiom is the generalization of the dummy axiom D.
- *partnership-allocation* (PA): if P is a partnership in $\xi \in \mathcal{G}^N$, then
$$I_\xi(P) I_{e_P}(i) = I_\xi(i) \quad \forall i \in P$$
- *k-monotonicity* (MK): if ξ is k-monotonic, then
$$I_\xi(S) \geq 0 \quad \forall S \subseteq N^{[0,k]}$$
- *symmetry* (SI): for $\forall \xi \in \mathcal{G}^N$, for $\forall \pi \in \Pi(N)$, we have
$$I_\xi(i) = I_{\pi\xi}(\pi \circ S)$$
- *recursion* (RI): I_ξ can be defined by the recursive formula
$$I_\xi(S) = I_{\xi_{[S]}}([S]) - \sum_{\emptyset \subset K \subset S} I_{\xi^{N\setminus K}}(S \setminus K)$$
- *2-efficiency* (2-EI): for $\forall i, j \in N$, we have
$$I_{\xi_{[ij]}}(S \cup [ij]) = I_\xi(S \cup i) + I_\xi(S \cup j)$$
- *limit condition* (LIM): for $\forall S \subset N$
$$I_{\xi^S}(S) = \Delta_S \xi^S(\emptyset)$$

3.6.4 Characterization of the Probabilistic Values and Interaction Indices

In the article [118], Fujimot et al. showed that

- I_ξ satisfies the axiom LI if and only if there exists a list of constants $\langle \mu_S(T) \rangle$ such that, for $\forall \xi \in \mathcal{G}^N$

$$I_\xi(S) = \sum_{T \subseteq N} \mu_S(T) \xi(T)$$

- I_ξ satisfies the axioms LI and DI if and only if there exists a list of constants $\langle \mu_S(T) \rangle$ such that, for $\forall \xi \in \mathcal{G}^N$

$$I_\xi(S) = \sum_{T \subseteq N \setminus S} \mu_S(T) \Delta_S \xi(T)$$

- I_ξ satisfies the axioms LI, DI and SI if and only if there exists a list of constants $\langle \mu_s(t) \rangle$ such that, for $\forall \xi \in \mathcal{G}^N$

$$I_\xi(S) = \sum_{T \subseteq N \setminus S} \mu_s(t) \Delta_S \xi(T)$$

- I_ξ satisfies the axioms LI, DI and MK if and only if there exists a list of constants $\langle \mu_S(T) \rangle$ such that, for $\forall \xi \in \mathcal{G}^N$

$$I_\xi(S) = \sum_{T \subseteq N \setminus S} \mu_S(T) \Delta_S \xi(T)$$

$$\sum_{T \subseteq N \setminus S} \mu_S(T) = 1$$

- I_ξ satisfies the axioms LI, DI, SI and MK if and only if there exists a list of constants $\langle \mu_S(T) \rangle$ such that, for $\forall \xi \in \mathcal{G}^N$

$$I_\xi(S) = \sum_{T \subseteq N \setminus S} \mu_s(t) \Delta_S \xi(T)$$

$$\sum_{T \subseteq N \setminus S} \mu_s(t) = 1$$

3.6.5 Characterization of Shapley, Banzhaf, Chaining Values and Interaction Indices

It the article [81], Weber shows that

- the unique power index that satisfy the axioms L, D, S and E is the Shapley Value

In the article [115], Grabish et al. show that

- the unique power index that satisfy the axioms L, D, S and 2-E is the Banzhaf Value

In the article [115], Grabish et al. show that

- the unique interaction index that satisfy the axioms LI, DI, SI and RI is the Shapley Interaction Index
- the unique interaction index that satisfy the axioms LI, DI, SI, 2-EI and LIM is the Banzhaf Interaction Index

In the article [118], Fujimot et al. show that

- the unique interaction index that satisfy the axioms LI, MK, DP, SI, EI and PA is Chaining Interaction Index

3.6.6 How to use the axioms: an open problem

In theory, axioms should be used to decide the index to use with a specific Machine Learning problem. At the moment this is an *open problem*. As we have seen previously, power indices are used to *order* the features, while specific index properties (for example, the Shapley Value properties) are not used.

Currently, it is unclear how to map and axiom with a problem's requirement. Some axioms, such as, *linearity*, have a direct consequence: we are using a linear approximation of the set function. But for the other ones, it is necessary to have a deeper understanding on their role in the definitions.

3.7 Interpretation of the Banzhaf and Shapley Value

We consider the definition of the Banzhaf Value:

$$\begin{aligned}\phi_\xi^B(i) &= \sum_{S\subseteq N\setminus i} \frac{1}{2^{n-1}}\Delta_i\xi(S) \\ &= \frac{1}{2^{n-1}}\sum_{S\subseteq N\setminus i}\Delta_i\xi(S)\end{aligned}$$

this is *the mean of the marginal contribution of i among all the coalitions*. Because the weight depends on the elements in the coalition and on the elements not contained in it, in some sense, when we analyze a coalition, we are considering all elements of the grand coalition. The Weighted Banzhaf Value is a simple generalization.

Now, we consider the definition of Shapley Value:

$$\begin{aligned}\phi_\xi^{Sh}(i) &= \sum_{S\subseteq N\setminus i} \frac{s!(n-1-s)!}{n!}\Delta_i\xi(S) \\ &= \sum_{s=0}^{n-1}\sum_{\substack{S\subseteq N\setminus i \\ |S|=s}} \frac{1}{n}\binom{n-1}{s}^{-1}\Delta_i\xi(S) \\ &= \frac{1}{n}\sum_{s=0}^{n-1}\binom{n-1}{s}^{-1}\sum_{\substack{S\subseteq N\setminus i \\ |S|=s}}\Delta_i\xi(S)\end{aligned}$$

it is the *the mean of the marginal contributions' average of i for each lattice's level*. Indeed, from the equation, we can see that

- for each level, we sum the marginal contributions of the player i
- then we compute the mean by dividing the previous sum by the cardinality of the level

- as last step we sum the previous means and we divide it by the number of levels

These ideas can be applied to all player and cardinal probabilistic indices, replacing the mean with the *expected value* based on some probability distribution, and replacing the first-order derivative with the derivatives of higher order.

3.8 Conclusions

In this chapter we introduced some of the most used power and interaction indices. We have seen that Shapley Value, Banzhaf Value and related interaction indices, are members of the more general classes *cardinal-probabilistic indices* and *player-probabilistic indices*. We have seen that these indices satisfies specific axioms. These axioms are useful to understand when to select an index or another for the *feature selection*.

Chapter 4

Approximate algorithms for power and interaction indices

4.1 Introduction

From the analysis of the definition of power and interaction indices, we can observe that to compute the *exact* value it is necessary to consider all 2^n subsets or $n!$ permutations: the complexity of these computations are respectively $\mathcal{O}(2^n)$ and $\mathcal{O}(n!) \approx \mathcal{O}(2^{n \log n})$.

If the set function has specific properties as, for example, a binary value ($\{0, 1\}$, the class of *voting games* and *weighted voting games*), there are several efficient algorithms available [98, 101, 103, 107].

Unfortunately, in this work, the set functions return an arbitrary positive real value. In this case, the solution is to use *approximate algorithms*, based on the random generation of *subsets* or *permutations*, with some selected distributions.

The computational complexity of a approximation algorithm $\mathcal{A}_\xi$ depend, obviously, on the complexity of ξ. The *global* complexity can be computed as:

$$\mathcal{O}(\mathcal{A}_\xi) = \mathcal{O}(\mathcal{A}) \cdot \mathcal{O}(\xi)$$

where $\mathcal{O}(\mathcal{A})$ is the *net* complexity of $\mathcal{A}$, excluding the evaluation of ξ.

The analysis of this approach can be found in [97, 99, 100, 105, 108]

However, in this work, we are not interested on the index values, but on the elements' order induced by the values. This simplify the requirements on approximation's quality.

4.2 General structure of the algorithms

As specified previously:

- Shapley Value, Chaining Value, and related interaction indices, are members of *cardinal-probabilistic* indices
- Banzhaf Value, Weighted Banzhaf Value, and related interaction indices, are members of *player-probabilistic* indices
- cardinal and player-probabilistic indices are members of the more general class *probabilistic indices*

By using these regularities, we can implement all indices with algorithms having the same structure.

The general definition of a power index is

$$\phi_\xi(i) = \sum_{T \subseteq N \setminus i} p_i(T) \cdot \Delta_i \xi(T)$$

since it is not possible to use all 2^{n-1} sets, we can use a sampling of the subsets space. There are two possible ways to implement this definition

- we sample the space in a *uniform* way (we must be sure not generating the same set more than once), then we multiply the marginal contribution $\Delta_i \xi(T)$ by $p_i(T)$
- we sample the space in a *weighted* way, where the coefficient $p_i(T)$, is converted into a probability to select T

The general structure of the algorithms using these approaches is:

Function ProbabilisticValue(ξ, n, m, p)

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : p weight function
output: array of probabilistic values

$v \leftarrow \mathbf{0} \in \mathbb{R}^n$;; *cumulative values*
for $t \in [1, m]$ **do**
  $T \leftarrow$ RandomSubset(n)
  $R \leftarrow N \setminus T$
  for $i \in R$ **do**
    $v_i \leftarrow v_i + p_i(T) \cdot \Delta_i \xi(T)$
  end
end

return v

Function ProbabilisticValue($\xi, n, m, \mathcal{D}$)

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ distribution used by the set generator
output: array of probabilistic values

$v \leftarrow \mathbf{0} \in \mathbb{R}^n$;; *cumulative values*
$c \leftarrow \mathbf{0} \in \mathbb{Z}^n$;; *set counts*
for $t \in [1, m]$ **do**
  $T \leftarrow$ RandomSubset($n, \mathcal{D}$)
  $R \leftarrow N \setminus T$
  for $i \in R$ **do**
    $v_i \leftarrow v_i + \Delta_i \xi(T)$
    $c_i \leftarrow c_i + 1$
  end
end

return v/c ;; *element wise division*

where $\mathcal{D}$ is the parameter used by the set generator to generate the sets with a selected *distribution*.

The computational complexity of these algorithms is

$$\mathcal{O}(\texttt{ProbabilisticValue}) = \mathcal{O}(m \cdot n)$$

and the complexity of `RandomSubset` is linear ($\mathcal{O}(n)$).

The interaction index has a similar definition:

$$I_\xi(ij) = \sum_{T \subseteq N \setminus ij} p_{ij}(T) \cdot \Delta_{ij}\xi(T)$$

The structure of algorithms is the same.

Function ProbabilisticInteractionIndex(ξ, n, m, p)

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : p weights function
output: array of probabilistic indices

$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$;; *cumulative values*
for $t \in [1, m]$ **do**
  $T \leftarrow$ RandomSubset(n)
  $R \leftarrow N \setminus T$
  for $ij \subseteq R$ **do**
    $v_{ij} \leftarrow v_{ij} + p_{ij}(T) \cdot \Delta_{ij}\xi(T)$
  end
end
return v

Function ProbabilisticInteractionIndex($\xi, n, m, \mathcal{D}$)

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ parameters for the set generator
output: array of probabilistic indices

$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$;; *cumulative values*
$c \leftarrow \mathbf{0} \in \mathbb{Z}^{n \times n}$;; *set counts*
for $t \in [1, m]$ **do**
  $T \leftarrow \text{RandomSubset}(n, \mathcal{D})$
  $R \leftarrow N \setminus T$
  for $ij \subseteq R$ **do**
    $v_{ij} \leftarrow v_{ij} + \Delta_{ij}\xi(T)$
    $c_{ij} \leftarrow c_{ij} + 1$
  end
end
return v/c ;; *element wise division*

Because we must consider the $\frac{1}{2}t(t-1)$ ($\approx t^2$) subsets (with cardinality 2) of a given set T with cardinality t, the computational complexity of the algorithms is

$$\mathcal{O}(\texttt{ProbabilisticInteractionIndex}) = \mathcal{O}(m \cdot n^2)$$

4.3 Banzhaf Value and Interaction Index

The Banzhaf Value and Banzhaf Interaction Index are the most simple indices to implement. The weight functions used are

$$p_i(T) = \frac{1}{2^{n-1}}$$

and

$$p_{ij}(T) = \frac{1}{2^{n-2}}$$

Because the number of subsets of $N \setminus i$ is 2^{n-1}, the weight used in the definition, we have a simple way to generate the sets: we can use an *uniform* random set generator.

4.4 Weighted Banzhaf Value and Interaction Index

The Weighted Banzhaf Value and related Interaction Indices are the direct generalization of the Banzhaf Indices. The weight functions used are

$$p_i(T) = \prod_{k \in T} p_k \prod_{k \in N \setminus i \setminus T} (1 - p_k)$$

and

$$p_{ij}(T) = \prod_{k \in T} p_k \prod_{j \in N \setminus ij \setminus T} (1 - p_k)$$

where $\langle p_k : k \in N \rangle$ is the probability of each element $k \in N$ to be inside a coalition (in the Banzhaf Value, $p_k = .5$).

The implementation can use a simple generalization of the uniform random set generator: a random set generator where the probability of each member is p_k and not .5.

4.5 Shapley Value and Interaction Index

The Shapley Value and Shapley Interaction Index can be implemented using the weight functions

$$p_i(T) = \frac{t!(n-1-t)!}{n!} = \frac{1}{n} \binom{n-1}{t}^{-1}$$

and

$$p_{ij}(T) = \frac{t!(n-2-t)!}{(n-1)!} = \frac{1}{n-1}\binom{n-2}{t}^{-1}$$

As for the Banzhaf Value/Index, we can use an *uniform* random set generator.

The main problem of the weight functions is that the computed value decrease very quickly with the values of n and t, introducing numerical problems. In the following table one can observe this behaviour.

n/t	1	2	3	4	5	6	$n/2$
1	1.						1.
2	.5	.5					.5
3	.3333	.1667	.3333				.1667
4	.25	.0833	.0833	.25			.0833
5	.2	.05	.0333	.05	.2		.0333
10	.1	.0111	.0028	.0012	.0008	.0008	.0008
40	.025	.0006	3.37e-5	2.74e -6	3.04e -7	4.34e -8	3.63e-13
100	.01	.0001	2.06e-6	6.38e -8	2.66e -9	1.40e-10	1.98e-31

Table 4.1: Function values for $p_i(T)$ and some values of n and t

However, there are two alternative implementations. The first one is the

classical implementation based on permutations

Function ShapleyValue(ξ, n, m)

input : ξ set function
input : n number of elements in the set
input : m number of permutations to use
output: array of Shapley Values

$v \leftarrow \mathbf{0} \in \mathbb{R}^n$;; *cumulative values*
$c \leftarrow \mathbf{0} \in \mathbb{N}^n$;; *set counts*
for $t \in [1, m]$ **do**
  $P \leftarrow$ RandomPermutation(n)
  for $k \in [1, n]$ **do**
    $T \leftarrow P[1:k]$
    $i \leftarrow P_k$
    $v_i \leftarrow v_i + \Delta_i \xi(T)$
    $c_i \leftarrow c_i + 1$
  end
end
return v/c ;; *element wise division*

The second one is based on the structure of $p_i(T)$ that can be interpreted in the following way

1. $1/n$ is the (*uniform*) probability to select a set with a cardinality in the range $[0, n-1]$ (n distinct values)
2. $\binom{n-1}{t}^{-1}$ is the (*uniform*) probability to select a set with the selected cardinality t ($\binom{n-1}{t}$ distinct sets)

The algorithm's implementation is based on a *weighted random set generator*

that select the cardinality with the specified distribution.

Function ShapleyValue(ξ, n, m)

input : ξ set function
input : n number of elements in the set
input : m number of permutations to use
output: array of Shapley Values

$\mathcal{C} \leftarrow \langle 1/n : k \in [0, n-1] \rangle \cup \langle 0 \rangle$;; *probability distribution for cardinality*
$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$;; *cumulative values, distinguished by cardinality*
$l \leftarrow \mathbf{0} \in \mathbb{R}^{n}$;; *cardinality counts*
for $k \in [1, m]$ **do**
  $T \leftarrow \text{RandomSubset}(n, \mathcal{C})$
  $t \leftarrow |T|$
  $R \leftarrow N \setminus T$
  for $i \in R$ **do**
    $v_{t,i} \leftarrow v_{t,i} + \Delta_i \xi(T)$
    $l_t \leftarrow l_t + 1$
  end
end
$s \leftarrow \mathbf{0} \in \mathbb{R}^{n}$
for $i \in N$ **do**
  $s_i \leftarrow \langle v_{\cdot,i}/l, \mathbf{1} \rangle$;; *element wise division and elements sum*
end
return s

The (2^{nd} order) Interaction Index has similar implementations

Function ShapleyInteractionIndex(ξ, n, m)

input : ξ set function
input : n number of elements in the set
input : m number of permutations to use
output: array of Shapley Interaction Indices

$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$;; *cumulative values*
$c \leftarrow \mathbf{0} \in \mathbb{N}^{n \times n}$;; *set counts*
for $t \in [1, m]$ **do**
  $P \leftarrow \text{RandomPermutation}(n)$
  for $k \in [1, n-1]$ **do**
    $T \leftarrow P[1:k]$
    $i \leftarrow P_k$
    $j \leftarrow P_{k+1}$
    $v_{ij} \leftarrow v_{ij} + \Delta_{ij}\xi(T)$
    $c_{ij} \leftarrow c_{ij} + 1$
  end
end
return v/c ;; *element wise division*

Function ShapleyInteractionIndex(ξ, n, m)

input : ξ set function
input : n number of elements in the set
input : m number of permutations to use
output: array of Shapley Values

$\mathcal{C} \leftarrow \langle 1/(n-1) : k \in [0, n-2] \rangle \cup \langle 0, 0 \rangle$;; *probability distribution for cardinality*
$v \leftarrow \mathbf{0} \in \mathbb{R}^{(n-1) \times n \times n}$;; *cumulative values, distinguished by cardinality*
$l \leftarrow \mathbf{0} \in \mathbb{R}^{n-1}$;; *cardinality counts*
for $k \in [1, m]$ **do**
  $T \leftarrow \text{RandomSubset}(n, \mathcal{C})$
  $t \leftarrow |T|$
  $R \leftarrow N \setminus T$
  for $ij \subseteq R$ **do**
    $v_{t,ij} \leftarrow v_{t,ij} + \Delta_{ij}\xi(T)$
    $l_t \leftarrow l_t + 1$
  end
end
$s \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$
for $ij \subseteq N$ **do**
  $s_{ij} \leftarrow \langle v_{\cdot,ij}/l, \mathbf{1} \rangle$;; *element wise division and elements sum*
end
return s

In all implementations, the value is a simple *mean* (sum of values divided by the number of terms) and this resolve the numerical problems.

The key in this approach is the *weighted random set generator*, but we will show, in the following sections, that its implementation is very simple.

The computation complexity is the same than the power and interaction indices:

$$\begin{aligned}\mathcal{O}(\texttt{ShapleyValue}) &= \mathcal{O}(m \cdot n)\\ \mathcal{O}(\texttt{ShapleyInteractionIndex}) &= \mathcal{O}(m \cdot n^2)\end{aligned}$$

4.6 Chaining Value and Interaction Index

Chaining Value (and related Interaction Index), has the same implementation problems than Shapley Value. However, it can be implemented using the following weight function

$$\begin{aligned}p_S(T) &= s\frac{(n-s-t)!(s+t-1)!}{n!}\\ &= \binom{n}{s}^{-1}\binom{s+t-1}{s-1}\end{aligned}$$

By substituting s with 1 and 2, we obtain

$$p_i(T) = \binom{n}{1}^{-1}\binom{t}{0}$$

and

$$p_i(T) = \binom{n}{2}^{-1}\binom{t+1}{1}$$

4.7 Probabilistic Value and Interaction Index

In general, it is not possible to use a generic probabilistic index, since it requires the knowledge of all 2^n weights, and this number can be very large. But the main problem is that it is very difficult, or impossible, to find all values.

The solution is to use a *weight function* that can calculate the weight based only on the player (or the pair), on the set used and on some extra parameters.

The cardinal-probabilistic indices and the player-probabilistic indices meet this condition. The cardinal-probabilistic indices need only a probability for each possible set's cardinality (n values because the full set is not used), the player-probabilistic indices need a probability for each player (n values). Eventually, we can have an index that combine cardinal and player indices: it needs $2n$ parameters.

Let $\langle p_i : i \in N \rangle$ the weights assigned to the players, the coefficients to use with the player-probabilistic value are:

$$p_i(T) = \prod_{l \in T} p_l \prod_{l \in N \setminus i \setminus T} (1 - p_l)$$

and for the related interaction index, the coefficients are:

$$p_{ij}(T) = \prod_{l \in T} p_l \prod_{l \in N \setminus ij \setminus T} (1 - p_l)$$

4.8 Considerations on the Probabilistic Indices approximations

The power indices (Shapley Value, Banzhaf Value, their Interaction Indices and more in general the probabilistic indices) are concepts with a rich literature. The algorithms to implements the indices are well known. However, there are only a limited number of articles that analyze the approximation algorithms for games based on *real set functions*. The majority of the ar-

ticles analyze the implementations for *voting games*, based on a binary set functions.

In their article, Castro et al. [99] analyze the approximation of the Shapley Value based on random permutations.

In their article, Maleki et al. [105] improve the bounds founded by Castro et al. always for the Shapley Value based on random permutations.

The article of Krishna et al. [107] analyze the approximation of the Banzhaf Value based on random sets.

We can summarize the previous articles in the following way

- all probabilistic indices are *weighted means*
- weights can be converted in a *weighted sampling* and the weighted average became a simple average
- the error on the mean decreases with an exponential law $e^{-\lambda m}$: it decreases very quickly at the begin but slowly for large values of m (m number of samples) [105–107]

The value of λ depends on the properties of the set function, but, in general, these properties are totally unknown, and very rarely the function has *nice* properties as monotonicity, super modularity or super additivity.

The last note is that, for use in this work, we are not interested on the (approximated) value, but in the induced order. This allows us to use values with a higher approximation error and to use a smaller number of samples.

4.9 Selecting the random generator

To generate *random sets* or *random permutations* it is necessary having a *good uniform random generator* of integer or real values.

Each programming language, in its standard library, has a random generator, and there are good third-part libraries. However, it is necessary to check that the *period* of the generator is long enough for the application. If the

set/permutation is composed of 20 elements, it is necessary to ensure that the generator has a period of $2^{20} \approx 10^6$ or, better, 10^7 states.

In general, the standard random generator is based on the *linear congruential generator* [159, 160] with a period of $2^{31} - 1$ states (sometimes with period or $2^{15} - 1$ states). However, there are better generators, as, for example, the *Mersenne Twister MT19937* [158], with a period of $2^{19937} - 1 \approx 4 \cdot 10^{6000}$ states.

Function RandomInteger(l, u)
input : l,u lower and upper integers **output:** a random integer, selected in the uniform way, in the range $[l, u]$

Function RandomReal()
output: a real number, selected in the uniform way, in the range $[0, 1]$

There exists a little problem with `RandomReal()` to consider: in general, the implementation is based on `RandomInteger()` as

$$\text{RandomReal}() = \frac{\text{RandomInteger}(0, M)}{M}$$

where M is the maximum integer value that the generator can generate. This means that there exist a minimum distance between two adjacent real values, equals to $1/M$.

The implementation of these algorithm is very efficient and the computational complexity is constant ($\mathcal{O}(1)$).

4.10 Generating a random permutation

A *random permutation* of length n is a random ordering of integers in the range $[1, n]$ (or $[0, n-1]$ depends on the context). A simple method to generate the permutation is to use the **Knuth shuffle** algorithm (computational

complexity $\mathcal{O}(n)$) [157][1]:

Function RandomPermutation(n)

input : n number of elements in the permutation
output: a random permutation

$P \leftarrow \langle 1, \ldots, n \rangle$;; *permutation*
for $i \in [1, n-1]$ **do**
  $j \leftarrow$ RandomInteger(i, n)
  swap(P_i, P_j)
end
return P

Sometimes, it is necessary to generate a random permutation of length $k < n$. In this case, the algorithm is very simple: to generate a random permutation, and to take the prefix of length k:

Function RandomPermutation(n, k)

input : n number of elements in the permutation
input : k number of elements of the prefix
output: a random permutation of length k

$P \leftarrow$ RandomPermutation(n)
return $P[1 : k+1]$

Sometimes, it can be useful to keep the list of previous generated prefixes and if the last prefix was already generated, generate a new one. This is important when the length of the prefix is very small because there is a high probability to generate the same prefix.

4.11 Generating a random subset

There are two algorithms to generate a *random subset* of the set $N = \{1, \ldots, n\}$ (or $\{0, \ldots, n-1\}$ it depends on the context), based on the map between the set and the natural integers:

1. generating a random integer (in the uniform way) in the range $[0, 2^n - 1]$

[1] In general, it is a good idea reusing the last permutation to generate the new one.

2. generating a sequence of n bits, where each bit is 1 with probability $\frac{1}{2}$

The first approach is possible only if the integers random generator has a period several order of magnitude greater than 2^n.

Function RandomSubset(n)

input : n number of elements in the set
output: a random subset

$S \leftarrow \{\}$
$r \leftarrow \text{RandomInteger}(0, 2^n - 1)$
for $i \in [1, n]$ **do**
  if $m \mod 2^{i-1} = 1$ **then**
    $S \leftarrow S \cup i$;; *remember that* $N = \{1, \ldots, n\}$
  end
end
return S

The second approach is valid for each n: given the probability of a bit to have the value 1 is $\frac{1}{2}$, the probability of a sequence with n bits is $\frac{1}{2^n}$, where 2^n is exactly the number of subsets for a set with n elements.

Function RandomSubset(n)

input : n number of elements in the set
output: a random subset

$S \leftarrow \{\}$
for $i \in N$ **do**
  $r \leftarrow \text{RandomReal}()$
  if $r \leq \frac{1}{2}$ **then**
    $S \leftarrow S \cup i$
  end
end
return S

It is possible to extend the last algorithm for the generation of subsets having

a predefined cardinality k:

Function RandomSubset(n, k)

input : n number of elements in the set
input : k cardinality of the subset
output: a random set of cardinality k

$S \leftarrow \{\}$
$i \leftarrow \text{RandomInteger}(1, n)$;; *it starts from a random position*
while $|S| \neq k$ **do**
  $r \leftarrow \text{RandomReal}()$
  if $r \leq \frac{1}{2}$ **then**
    $S \leftarrow S \cup i$
  end
  $i \leftarrow i \oplus 1$;; *it cycles between 1 and n*
end
return S

Note that the cardinality of S do not change by adding an element already present.

The last algorithm is the *random cardinality generator* for generating a random cardinality in the range $[k_{\min}, k_{\max}]$. In doing this, it is necessary consider the *numerosity* of the sets with a given cardinality, that it is $\binom{n}{k}$:

Function RandomCardinality($k_{\min}, k_{\max}$)

input : n number of elements in the set
input : $[k_{\min}, k_{\max}]$ minimum/maximum cardinality of the subsets
output: a random cardinality

$m \leftarrow \sum_{k=k_{\min}}^{k_{\max}} \binom{n}{k}$
$k \leftarrow k_{\min}$
$c \leftarrow \binom{n}{k_{\min}}$
$r \leftarrow \text{RandomReal}()$
while $r > \frac{c}{m}$ **do**
  $k \leftarrow k + 1$
  $c \leftarrow c + \binom{n}{k}$
end
return k

The computational complexity of these algorithms is linear ($\mathcal{O}(n)$).

4.12 Generating a random subset with a selected distribution

In the previous algorithms we have used a random set generator able of generating set following a selected distribution $\mathcal{D}$. The problem is how to describe this distribution.

A simple approach, compatible with the previous algorithms, is to split the distribution into two parts:

1. $\mathcal{C}$: the probability distribution assigned to the set cardinality
2. $\mathcal{P}$: the probability assigned to the set elements

The *uniform random set generator* uses the probabilities based on the multiplicity of the sets with a selected cardinality

$$\mathcal{C} = \left\langle \frac{1}{2^n}\binom{n}{k} : k \in [0, n] \right\rangle$$

and the same probability for the elements of the set

$$\mathcal{P} = \left\langle \frac{1}{2} : i \in N \right\rangle$$

The algorithm for generating a set based on $\mathcal{P}$ is very similar to the original

one:

Function RandomSubset($n, \mathcal{P}$)

input : n number of elements in the set
input : $\mathcal{P}$ probabilities assigned to the elements
output: a random subset

$S \leftarrow \{\}$
for $i \in N$ **do**
  $r \leftarrow$ RandomReal()
  if $r \leq \mathcal{P}_i$ **then**
    $S \leftarrow S \cup i$
  end
end
return S

Function RandomSubset($n, k, \mathcal{P}$)

input : n number of elements in the set
input : k cardinality of the subset
input : $\mathcal{P}$ probability assigned to the players
output: a random subset

$S \leftarrow \{\}$
$i \leftarrow$ RandomInteger$(1, n)$;; *start from a random position*
while $|S| \neq k$ **do**
  $r \leftarrow$ RandomReal()
  if $r \leq \mathcal{P}_i$ **then**
    $S \leftarrow S \cup i$
  end
  $i \leftarrow i \oplus 1$;; *it cycles between 1 and n*
end
return S

Also, the algorithm for generating a random cardinality, based on the cardinality probability distribution, is very similar to the original one

Function RandomCardinality($k_{\min}, k_{\max}, \mathcal{C}$)

input : n number of elements in the set
input : $[k_{\min}, k_{\max}]$ minimum/maximum cardinality of the subset
input : $\mathcal{C}$ probability distribution assigned to the cardinality
output: a random cardinality

$m \leftarrow \sum_{k=k_{\min}}^{k_{\max}} \mathcal{C}_k$
$k \leftarrow k_{\min}$
$c \leftarrow \mathcal{C}_{k_{\min}}$
$r \leftarrow$ RandomReal()
while $r > \frac{c}{m}$ **do**
  $k \leftarrow k + 1$
  $c \leftarrow c + \mathcal{C}_k$
end

return k

Composing the previous algorithms, the generalized random set generator is the following:

Function RandomSubset($n, k_{\min}, k_{\max}, \mathcal{C}, \mathcal{P}$)

input : n number of elements in the set
input : $[k_{\min}, k_{\max}]$ minimum/maximum cardinality of the subset
input : $\mathcal{C}$ probability distribution assigned to the cardinality
input : $\mathcal{P}$ probabilities assigned to the elements
output: a random subset

$k \leftarrow$RandomCardinality($k_{\min}, k_{\max}, \mathcal{C}$)
$S \leftarrow$RandomSubset($n, k, \mathcal{P}$)

return S

As for the previous implementations, the computational complexity of these algorithms is linear ($\mathcal{O}(n)$).

4.13 Approximate algorithms based on elements order

As specified at the beginning of the section, it is not possible computing the *exact* value of the power indices. But it is not a problem because in general, the value of the power index is used only to *order*, in a descending way, the elements of the set. It is possible to do some extra observations:

- the order of elements with *similar* values is not important
- the order of elements with *small* values is not important
- it is important only the order of elements with *highest* index values

Sometimes, the problem consist of selecting the first k elements, with k inside a little range $[k_{\min}, k_{\max}]$. In this case, the order of the elements can be used in two different ways:

1. the order of the elements is computed **only** once, then the application selects the first k_1, k_2, etc, elements
2. the order of the elements is computed for each k, **but** k is used **only** to separate the first k elements from the rest. The *exact* order of the selected elements is not important.

Another note is about how to evaluate the function ξ. In some works the function is the performance (*accuracy*, *precision*, *mean square error*, etc.) of a machine learning algorithm (*Naive Bayes Classifier*, *Decision Tree Classifier*, *Linear Regression*, etc): computing this function can be very expensive.

These observations permit to relax the properties of the computed values: it is not necessary to ensure specific properties on the *values* of the index, but only on the *order* among the values.

To implement these ideas, we can use the following steps:

1. we subdivide the power index's approximation in *epochs*, and in each epoch we use a predefined number of random samples

2. after each epoch, we order the elements based on the power index, and compare this order with the previous one

3. if the two orders are different, we continue the computation, otherwise we check the *stability* of the order, that is, if the order is the same in the last p epochs

4. if the order is *stable* for the specified number of epochs, we can consider the order *definitely stable* and we stop the computation

---

Function SelectElements($\xi, n, m, \mathcal{D}, n_k, n_e, n_p$)

---

input : ξ set function
input : n number of elements in the set
input : m number of sets/permutations per epoch
input : $\mathcal{D}$ distribution to use

input : n_k number of elements to consider in the order
input : n_e number of epochs
input : n_p stability counter
output: elements order
output: power/interaction indices (optional)

$p \leftarrow 0$;; *patient counter*
$o \leftarrow \langle 1, \ldots, n \rangle$;; *default order*
$o_p \leftarrow o[1 : n_k]$;; *previous selected elements order*
for $e \in [1, n_e]$ ***and*** $p < n_p$ **do**
    ;; *loop n_e times of until $p \geq n_p$*
    $v \leftarrow \text{ProbabilisticValue}(\xi, n, m, \mathcal{D})$
    $ii \leftarrow \text{ProbabilisticInteractionIndex}(\xi, n, m, \mathcal{D})$;; *(optional)*
    $o \leftarrow \text{argsort}(v)$
    $o_c \leftarrow o[1 : n_k]$;; *current selected elements order*
    are-equals $\leftarrow \text{compare}(o_p, o_c)$
    if *are-equals* **then**
        $p \leftarrow p + 1$
    else
        $p \leftarrow 0$
        $o_p \leftarrow o_c$
    end
end

return $o(, ii)$

---

where `compare`(o_p, o_c) can be defined as

1. the *sequences* o_p and o_c (considering the order) are equals or
2. the *sets* o_p and o_c (independently by the order) are equals

The value of n_k is *problem dependent*. In general, it is important to have a *stable order* only for the most important elements. The computational complexity is

$$\begin{aligned}\mathcal{O}(\texttt{SelectElements}) &= n_e \cdot \mathcal{O}(\texttt{ProbabilisticValue}) \\ &= \mathcal{O}((n_e \cdot m) \cdot n)\end{aligned}$$

We can use the same approach for the computation of the interaction indices, because it is simple to compute the power indices during the same computation. The complexity is

$$\begin{aligned}\mathcal{O}(\texttt{SelectElements}) &= n_e \cdot \mathcal{O}(\texttt{ProbabilisticInteractionIndex}) \\ &= \mathcal{O}((n_e \cdot m) \cdot n^2)\end{aligned}$$

4.14 Approximate algorithms using parallelism and local maximum

To improve the performance of the algorithms it is possible to evaluate the function on the sets in a *parallel* way. In doing this, we generate *previously* the set to use in computing the function values. Then, we evaluate the function in a parallel way, and the results are saved in a data structure that doesn't need locks. At the end, we aggregate the results to obtain the final value.

In general, it is not possible to generate the random sets in parallel, because the random number generator has a state that changes after each generated

number.

Function ParallelProbabilisticValue($\xi, n, m, \mathcal{D}$)

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ probability distribution
output: array of probabilistic values

$L \leftarrow$ RandomSetsToUse($n, m, \mathcal{D}$)
$v \leftarrow \mathbf{0} \in \mathbb{R}^{m \times n}$;; *cumulative values, distinguished by set*
parallel for $t \in [1, m]$ **do**
  $T \leftarrow L_t$
  $v_{t,\cdot} \leftarrow$ EvaluateFunction(ξ, n, T)
end
$p \leftarrow$ AggregateValues(v)
return p

Because it is possible to evaluate ξ on different sets independently, we can get a *linear* speedup using the parallel approach:

$$\mathcal{O}(\texttt{ParallelProbabilisticValue}) \approx \frac{1}{n_t}\mathcal{O}(\texttt{ProbabilisticValue})$$

with n_t the number of threads.

Function RandomSetsToUse($n, m, \mathcal{D}$)

input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ distribution to use
output: list f random sets to use

$L \leftarrow \langle\rangle$
for $t \in [1, m]$ **do**
  $T \leftarrow \text{RandomSubset}(n, \mathcal{D})$
  $L \leftarrow L \cup \langle T \rangle$
end
return L

Function EvaluateFunction(ξ, n, T)

input : ξ set function
input : n number of elements in the set
input : T set to use
output: list f random sets to use

$v \leftarrow \mathbf{0} \in \mathbb{R}^n$
$R \leftarrow N \setminus T$
for $i \in R$ **do**
  $v_i \leftarrow \Delta_i \xi(T)$
end
return v

Function AggregateValues(v)

input : $v \in \mathbb{R}^{m \times n}$ values to aggregate
output: aggregated values

$p \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $i \in [1, n]$ **do**
  for $t \in [1, m]$ **do**
    $p_i \leftarrow p_i + v_{t,i}$
  end
end
return p

Given we are interested into the best set of elements, an alternative way to compute the indices is to use the *best random sets*. The idea is

- generate the list of sets to use
- for each set, search a *closed* set that it is a *local maximum*
- replace the original set with its *best set*

Function ProbabilisticValueByBestSet($\xi, n, m, \mathcal{D}$)

---

input : ξ set function
input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ probability distribution
output: array of probabilistic values

$L \leftarrow$ BestSetsToUse$(n, m, \mathcal{D})$
$v \leftarrow \mathbf{0} \in \mathbb{R}^{m \times n}$;; *cumulative values, distinguished by set*

parallel for $t \in [1, m]$ **do**
  $T \leftarrow L_t$
  $v_{t,\cdot} \leftarrow$ EvaluateFirstDerivOnBestSet(ξ, n, T)
end

;; *it aggregates the distinguished values in a single value*
$p \leftarrow$ AggregateValues(v)

return p

Function BestSetsToUse($\xi, n, m, \mathcal{D}$)

input : ξ set function
input : n number of elements in the set
input : m number of samples to generate
input : $\mathcal{D}$ distribution used by the set generator
output: list of *best* sets

$L \leftarrow \langle\rangle$
for $i \in [1, m]$ **do**
  $S \leftarrow \text{RandomSubset}(n, \mathcal{D})$
  $T \leftarrow \text{find-best-set}(\xi, S)$
  $L \leftarrow L \cup \langle S \rangle$
end
return L

where `find-best-set()` is a function that, starting from S, searches a *closed set*, with the same cardinality, where the function has a *local maximum*. It can be implemented using any algorithm in the family of *discrete hill climbing algorithms*.

Using this approach, there are two observations to do:

1. starting from two different sets, we can find the same *best set*. And this is a good news
2. the derivative must be computed in a little different way because now the *best set* has the role of $S \cup i$.

Function EvaluateFirstDerivOnBestSet(ξ, n, T)

input : ξ set function
input : n number of elements in the set
input : T set to use
output: list f random sets to use

$v \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $i \in T$ **do**
  $v_i \leftarrow \Delta_i \xi(T \setminus i)$
end

return v

These approaches can be extended, in the obviously way, to compute the interaction indices.

Function EvaluateSecondDerivOnBestSet(ξ, n, T)

input : ξ set function
input : n number of elements in the set
input : T set to use
output: list f random sets to use

$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$
for $ij \subseteq T$ **do**
  $v_{ij} \leftarrow \Delta_{ij} \xi(T \setminus ij)$
end

return v

4.15 Conclusions

In this chapter we described the algorithms used to compute the approximated values of power and interaction indices. Actually the algorithms implement only the 2^{nd} degree interaction indices, but it is simple to extend the implementation for indices of any degree.

We have, also, introduced some extension to improve the computation performances, using the parallel computation, and the approximations, using the *best sets*.

Chapter 5

The approximation of set functions

5.1 Introduction

In Chapter 3 we introduced many power and interaction indices. We have seen that the power indices are interaction indices applied to coalitions with only one player. We have seen, also, that set function and interaction indices form a *transform*, that is, a couple of invertible functions. We have also seen that the power indices are a *score* that quantify the marginal contribution of a player inside its coalitions.

Most of the time, we are interested of searching the coalition, with a chosen cardinality, where the function has the highest value. To look for this set we can use the *heuristic combinatorial* optimization algorithms. These algorithms have a computational complexity similar to the complexity of the algorithms used to compute the indices. Furthermore, these algorithms use the (discrete) derivative, as the definition of the indices. The questions are: does the set, with the maximum function's value, contain the players with the highest score? Is it reasonable to select the k players with the highest score as an alternative to search the set with the highest function's value?

There is not a definitive answer. The main problem is, and it is possible to demonstrate, that the power indices are the *best linear approximation* of the set function using a *weighted mean square error*. And the interaction indices are the approximations of a higher degree. Different weights describe different

indices. Obviously, in general, there is not a simple relation between the maximum on the original function and the maximum on the approximated one. But, we can find a relationship if we can *conjecture* that

- if a player *collaborates* with a coalition, it collaborates *almost always*
- if a player *interferes* with a coalition, it interferes *almost always*

Based on these conjectures, there are good chances that the best set of the function contains the players with the highest scores.

In the following sections we will demonstrate the relationship between the power and interaction indices and the approximation problem. To analyze the function on selected sets' cardinality, we will use some *families* of sets, where a *family* of sets is a subset of 2^N containing sets with some selected property.

5.2 Set families

A set family F is a subset of 2^N: it is a collection of sets $S \subseteq N$, with some *selected* property. We are interested to the following families:

- $F = 2^N$ the family of all subsets of N
- $F = N^k$ the family of subsets of N with cardinality exactly k
- $F = N^{[0,k]}$ the family of subsets of N with cardinality in the range $[0, k]$

We have already seen that $\mathcal{L}(2^N)$ is a linear space. Also the set of functions defined on N^k and $N^{[0,k]}$ is a linear space, because any linear combination of functions in these spaces returns a function in the same space. Using the Möbius representation of the set function, we have

$$\begin{aligned}\xi^{(3)}(S) &= \alpha_1\xi^{(1)}(S) + \alpha_2\xi^{(2)}(S) \\ &= \alpha_1\left(\sum_{T\subseteq N} a_T^{(1)} e_T(S)\right) + \alpha_2\left(\sum_{T\subseteq N} a_T^{(2)} e_T(S)\right) \\ &= \sum_{T\subseteq N}\left(\alpha_1 a_T^{(1)} + \alpha_2 a_T^{(2)}\right) e_T(S) \\ &= \sum_{T\subseteq N} a_T^{(3)} e_T(S)\end{aligned}$$

Because we have used only *unanimity games* already used in the $\xi^{(1)}$ and $\xi^{(2)}$, the degree of $\xi^{(3)}$ will be equal or less than $\xi^{(1)}$ and $\xi^{(2)}$.

5.2.1 Properties

For each $i \in N$, we have

$$F = F_i \cup F_{-i}$$

that is, the family F is composed by the sub-families

- $F_i = \{S \in F : i \in S\}$, the family of sets containing i
- $F_{-i} = \{S \in F : i \notin S\}$, the family of sets don't containing i

The family F_i can be defined also as

$$F_i = F_{-i} \cup i = \{S \cup i : S \in F_{-i}\}$$

The family F_i^{-j} is the family of sets containing i but not j, and it is equal to the family F_{-j}^{i}, the family of sets not containing j but containing i:

$$F_i^{-j} = F_{-j}^{i} = \{S \in F : i \in S, j \notin S\}$$

For each $ij \supseteq N$, we have

$$F = F_{ij} \cup F_{-ij} \cup F^i_{-j} \cup F^j_{-i}$$

that is, the family F is composed by the sub-families

- $F_{ij} = \{S \in F : i, j \in S\}$ the family of sets containing i and j
- $F_{-ij} = \{S \in F : i, j \notin S\}$ the family of sets don't containing i nor j
- $F_i^{-j} = \{S \in F : i \in S, j \notin S\}$ the family of sets containing i but not j
- $F_j^{-i} = \{S \in F : i \notin S, j \in S\}$ the family of sets containing j but not i

More in general, the families F_T and F_{-T} are defined as

$$F_T = \{S \in F : S \supseteq T\}$$
$$F_{-T} = \{S \in F : S \cap T = \emptyset\}$$

and

$$F_T^U = \{S \in F_T : S \supseteq U\}$$
$$F_T^{-U} = \{S \in F_T : S \cap U = \emptyset\}$$

We denote with

- $\lfloor F \rfloor = \min_{S \in F} |S|$ the minimum cardinality of the sets in the family
- $\lceil F \rceil = \max_{S \in F} |S|$ the maximum cardinality of the sets in the family
- $F^k = \{S \in F : |S| = k\}$ the family of the sets with cardinality exactly k (with $\lfloor F \rfloor \leq k \leq \lceil F \rceil$)

	$\lfloor F \rfloor$	$\lceil F \rceil$
$F = 2^N$	0	n
$F = N^k$	k	k
$F = N^{[0,k]}$	0	k

Table 5.1: Family's lower/upper cardinality

Obviously

$$F = \bigcup_{k=\lfloor F \rfloor}^{\lceil F \rceil} F^k$$

One important property of these families is their cardinality

family	2^N	N^k	$N^{[0,k]}$	note
F	2^n	$\binom{n}{k}$	$\binom{n}{k}^*$	
F_i	2^{n-1}	$\binom{n-1}{k-1}$	$\binom{n-1}{k-1}^*$	
F_{-i}	2^{n-1}	$\binom{n-1}{k-1}$	$\binom{n-1}{k-1}^*$	$= \lvert F_i \rvert$
F_{ij}	2^{n-2}	$\binom{n-2}{k-2}$	$\binom{n-2}{k-2}^*$	
F_{-ij}	2^{n-2}	$\binom{n-2}{k-2}$	$\binom{n-2}{k-2}^*$	$= \lvert F_{ij} \rvert$
F_T	2^{n-t}	$\binom{n-t}{k-t}$	$\binom{n-t}{k-t}^*$	
F_{-T}	2^{n-t}	$\binom{n-t}{k-t}$	$\binom{n-t}{k-t}^*$	$= \lvert F_T \rvert$
$F_{-S}^{T \setminus S}$	2^{n-t}	$\binom{n-t}{k-t}$	$\binom{n-t}{k-t}^*$	$= \lvert F_T \rvert$

Table 5.2: Family's cardinality

5.3 Function approximation

As we have seen in the previous section, the families 2^N, N^k and $N^{[0,k]}$ are linear spaces with the unanimity games $e_T(\mathbf{x})$ as bases. The next results can be applied to each space.

Let $\mathcal{S} = \mathcal{L}(F)$ the space of linear functions defined on the family F, $f, g \in \mathcal{S}$ and μ a *probability distribution*. In this space we define the *pseudo inner product*

$$\langle f, g \rangle_\mu = \sum_{S \in F} \mu(S) f(S) g(S)$$

and its induced norm

$$\|f\|_\mu = \sqrt{\langle f, f\rangle_\mu} = \sqrt{\sum_{S\in F} \mu(S) f(S)^2}$$

We have already seen that if $\mu(S) > 0 \ \forall S \in F$ the *pseudo inner product* has the same properties than classic *inner product*.

We are interested of finding the *best approximation* of the function $f \in \mathcal{L}(2^N)$ using a function $g \in \mathcal{S}$. Using the defined norm, we are searching a function g such that

$$\min_{g\in\mathcal{S}} \|f - g\|_\mu$$

If $\mathcal{S} = \mathcal{L}(N^{[0,1]})$ $(F = N^{[0,1]})$ we are searching the *best linear* approximation, and with $\mathcal{S} = \mathcal{L}(N^{[0,2]})$ $(F = N^{[0,2]})$ the *best second order* approximation, etc.

We already know that the solution of the problem is the *orthogonal projection* g_f of f into $\mathcal{S}$, and the property of the projection is that the element $f - g_f$ is *orthogonal* to all g in $\mathcal{S}$:

$$\langle f - g, h\rangle_\mu = 0 \quad \forall h \in \mathcal{S}$$

In particular, it is orthogonal to all bases of $\mathcal{S}$, for example the *unanimity game* basis e_S:

$$\langle f - g, e_S\rangle_\mu = 0 \quad \forall e_S \in \mathcal{S}$$

but this means also that

$$\langle f, e_S\rangle_\mu = \langle g, e_S\rangle_\mu \quad \forall e_S \in \mathcal{S} \tag{5.1}$$

In the space $\mathcal{S}$ the function g has the form

$$g(\mathbf{x}) = \sum_{T\in F} \beta_T e_T(\mathbf{x})$$

For the linearity, the function g can be rewritten also as

$$g(\mathbf{x}) = \sum_{T \subseteq N} a_T g_T(\mathbf{x})$$

with $g_T(\mathbf{x})$ the best approximation of the unanimity game basis $e_T(\mathbf{x})$

$$g_T(\mathbf{x}) = \sum_{U \in F} \beta_U^T e_U(\mathbf{x})$$

In this case, the 5.1 can be rewritten as

$$\langle f, e_S \rangle_\mu = \langle g, e_S \rangle_\mu$$
$$\Big\langle \sum_{T \subseteq N} a_T e_T, e_S \Big\rangle_\mu = \Big\langle \sum_{T \subseteq N} a_T g_T, e_S \Big\rangle_\mu$$

and, for the linearity of the inner product

$$\sum_{T \subseteq N} a_T \langle e_T, e_S \rangle_\mu = \sum_{T \subseteq N} a_T \langle g_T, e_S \rangle_\mu \tag{5.2}$$

5.3.1 General approximation

In their paper, Ding et al. [71] demonstrate how to create a *weighted* approximation of order k of a set function, based on a *generic* probability distribution μ. To understand their approach we can analyze Equation 5.2 and to observe that

$$\langle e_T, e_S \rangle_\mu = \sum_{U \in F} \mu(U) e_T(U) e_S(U)$$

Given $e_T(U) = 1$ for each $U \supseteq T$ and $e_S(U) = 1$ for each $U \supseteq S$, we have

$$\langle e_T, e_S \rangle_\mu = \sum_{\substack{U \in F \\ U \supseteq T \cup S}} \mu(U) = \bar{\mu}(T \cup S)$$

The first part of Equation 5.1 can be rewritten as

$$\begin{aligned}
\langle f, e_S \rangle_\mu &= \sum_{U \in F} f(U) e_S(U) \mu(U) \\
&= \sum_{U \in F} \left(\sum_{T \subseteq N} a_T e_T(U) e_S(U) \mu(U) \right) \\
&= \sum_{T \subseteq N} a_T \left(\sum_{U \in F} e_T(U) e_S(U) \mu(U) \right) \\
&= \sum_{T \subseteq N} a_T \bar{\mu}(T \cup S) \\
&= \eta_S
\end{aligned}$$

The second part of Equation 5.1 can be rewritten as

$$\begin{aligned}
\langle g, e_S \rangle_\mu &= \sum_{U \in F} g(U) e_S(U) \mu(U) \\
&= \sum_{U \in F} \left(\sum_{T \subseteq N} \beta_T e_T(U) e_S(U) \mu(U) \right) \\
&= \sum_{T \subseteq N} \beta_T \left(\sum_{U \in F} e_T(U) e_S(U) \mu(U) \right) \\
&= \sum_{T \subseteq N} \beta_T \bar{\mu}(T \cup S)
\end{aligned}$$

where β_T are the unknown coefficients of the function g.

The general solution of the problem can be converted into the linear system

$$M_\mu \cdot \beta = \eta$$

with:

- M_μ a symmetric matrix $|\mathcal{S}| \times |\mathcal{S}|$ where

$$M_\mu[S,T] = \bar{\mu}(T \cup S) \quad \forall S,T \in \mathcal{S}$$

- $\beta = \langle \beta_S : S \in \mathcal{S} \rangle$ the *unknown* score's vector
- $\eta = \langle \eta_S : S \in \mathcal{S} \rangle$

To find the approximation of k-degree, the rows and columns of the matrix must be ordered using the *lexicographic order* of the sets, and to include in the approximation all sets with cardinality $|S| \leq k$.

Since the matrix is symmetric, there exist efficient algorithms to solve the linear system, for example the Cholesky's method.

5.4 Player-based approximation

5.4.1 Introduction

In the player-probabilistic value (and related interaction index), a probability p_i to participate in a coalition S, is assigns to each player $i \in N$:

$$\mathcal{P} = \langle p_i : i \in N \rangle$$

The analysis of the power and interaction index (and related transform), has been addressed by Ding et al. in [72] and in the articles [81, 102].

If the probability of participation of each player in a coalition is *independent*, the probability of the coalition S is computed as

$$\mu(S) = \mathcal{P}(S) = \prod_{i \in S} p_i \prod_{i \in N \setminus S} q_i \tag{5.3}$$

where $q_i = 1 - p_i$, that is: the product of players' probability participating in the coalition, multiplied by the probabilities of the remainder players do not to be in the coalition.

The idea is to consider each player i as a *independent* Bernoulli random variable X_i with $\mathcal{P}(X_i = 1) = p_i$ and $\mathcal{P}(X_i = 0) = q_i$. If the probability p_i is not available, but it is available the weight function μ, it can be computed as

$$p_i = \mathcal{P}(X_i = 1) = \sum_{S \in F_i} \mu(S)$$

(for a normalized μ over the Boolean lattice) and, obviously

$$q_i = \mathcal{P}(X_i = 0) = \sum_{S \in F_{-i}} \mu(S) = 1 - p_i$$

The *expected value* and *variance* of the Bernoulli random variable X_i are:

$$\begin{aligned} E_\mu[X_i] &= p_i \\ \mathrm{Var}_\mu[X_i] &= p_i q_i = E[X_i^2] \end{aligned}$$

Using the transform

$$Z_i = \frac{X_i - p_i}{\sqrt{p_i q_i}} \quad \forall i \in N$$

the random variable X_i can be *standardized* in a random variable Z_i with zero *mean* and unit variance:

$$\begin{aligned} E[Z_i] &= 0 \\ \mathrm{Var}[Z_i] &= 1 = E[Z_i^2] \end{aligned}$$

We can define the random variable Z_S as

$$Z_S = \prod_{i \in S} Z_i \quad \forall S \subseteq N$$

Because X_i are *mutually independent*, also Z_i and Z_S will be *mutually independent*, and we have (with $\mathbf{c} \in \{0, 1\}^s$)

$$\mathcal{P}(Z_S = \mathbf{c}) = \mathcal{P}(\prod_{i \in S} Z_i = c_i) = \prod_{i \in S} \mathcal{P}(Z_i = c_i)$$

and

$$E[Z_S] = \prod_{i \in S} E[Z_i] = 0$$
$$\mathrm{Var}[Z_S] = \prod_{i \in S} \mathrm{Var}[Z_i] = 1$$

5.4.2 Orthonormal basis for $\mathcal{L}(F)$

The 2^n functions $\langle z_S : S \subseteq N \rangle$, where z_S is defined as

$$z_S(\mathbf{x}) = \prod_{i \in S} \frac{x_i - p_i}{\sqrt{p_i q_i}}$$

form a *orthonormal basis* for $\mathcal{L}(F)$ with respect to $\langle \cdot, \cdot \rangle_\mu$, and μ defined as in 5.3. Indeed, we consider the inner product:

$$\langle z_R, z_S \rangle_\mu = \sum_{T \subseteq N} \mu(T) z_R(T) z_S(T)$$

this is the definition of *expected value* of the random variable $Z_R Z_S$. If $R = S$, we have

$$E[Z_S Z_S] = E[Z_S^2] = \mathrm{Var}[Z_S] = 1$$

otherwise

$$E[Z_R Z_S] = E[Z_R] E[Z_S] = 0$$

that is:

$$\langle z_S, z_S \rangle_\mu = 1 \quad \forall S \in F$$
$$\langle z_R, z_S \rangle_\mu = 0 \quad \forall R, S \in F, R \neq S$$

5.4.3 Function transforms

Given $\langle z_S : S \subseteq N \rangle$ is an orthonormal basis for $\mathcal{L}(F)$, each function $f \in \mathcal{L}(F)$ can be represented as a linear combination of this basis

$$f(\mathbf{x}) = \sum_{S \subseteq N} \langle f, z_S \rangle_\mu z_S(\mathbf{x})$$

since the basis is *orthonormal*, to approximate the function $f \in \mathcal{L}(F)$ with another function f^k of degree k, we can just consider the basis up to k^{th} degree:

$$f^k(\mathbf{x}) = \sum_{S \subseteq N^{[0,k]}} \langle f, z_S \rangle_\mu z_S(\mathbf{x})$$

The basis z_S can be represented also using the unanimity games e_T:

$$\begin{aligned}
z_S(\mathbf{x}) &= \prod_{i \in S} (a_i x_i + b_i) \\
&= \sum_{T \subseteq S} \left(\prod_{i \in T} a_i x_i \prod_{i \in S \setminus T} b_i \right) \\
&= \sum_{T \subseteq S} \left(\prod_{i \in T} a_i \prod_{i \in S \setminus T} b_i \right) e_T(\mathbf{x}) \\
&= \sum_{T \subseteq S} \frac{\prod_{i \in S \setminus T} (-p_i)}{\prod_{i \in S} \sqrt{p_i q_i}} e_T(\mathbf{x}) \\
&= \sum_{T \subseteq S} (-1)^{|S \setminus T|} \frac{\prod_{i \in S \setminus T} (p_i)}{\prod_{i \in S} \sqrt{p_i q_i}} e_T(\mathbf{x}) \\
&= \sum_{T \subseteq S} \alpha_T^S e_T(\mathbf{x})
\end{aligned}$$

with

$$a_i = \frac{1}{\sqrt{p_i q_i}}, \quad b_i = \frac{-p_i}{\sqrt{p_i q_i}}$$

and

$$\alpha_T^S = (-1)^{|S \setminus T|} \frac{\prod_{i \in S \setminus T}(p_i)}{\prod_{i \in S} \sqrt{p_i q_i}}$$

The definition of the function f can be rewritten as

$$\begin{aligned} f(\mathbf{x}) &= \sum_{S \subseteq N} \langle f, z_S \rangle_\mu z_S(\mathbf{x}) \\ &= \sum_{S \subseteq N} \langle f, z_S \rangle_\mu \sum_{T \subseteq S} \alpha_T^S e_T(\mathbf{x}) \\ &= \sum_{T \subseteq S} \left(\sum_{S \subseteq N} \langle f, z_S \rangle_\mu \right) \alpha_T^S e_T(\mathbf{x}) \\ &= \sum_{T \subseteq N} \left(\sum_{S \supseteq T} \alpha_T^S \langle f, z_S \rangle_\mu \right) e_T(\mathbf{x}) \\ &= \sum_{T \subseteq N} \alpha_T e_T(\mathbf{x}) \end{aligned}$$

with

$$\alpha_T = \sum_{S \supseteq T} \alpha_T^S \langle f, z_S \rangle_\mu$$

and its k^{th} approximation as

$$f^k(\mathbf{x}) = \sum_{T \subseteq N^{[0,k]}} \alpha_T^k e_T(\mathbf{x})$$

with

$$\alpha_T^k = \sum_{\substack{T \supseteq S \\ |T| \leq k}} \alpha_T^S \langle f, z_T \rangle_\mu$$

5.4.4 Player-probabilistic interaction transform

Let $I_f^{\mathcal{P}}(S)$ the *player-probabilistic interaction index* of order s defined as:

$$\begin{aligned} I_f^{\mathcal{P}}(S) &= \frac{1}{\prod_{i \in S} \sqrt{p_i q_i}} \langle f, z_S \rangle_\mu \\ &= \sum_{T \subseteq N} \frac{1}{\prod_{i \in S} \sqrt{p_i q_i}} \mu(T) f(T) z_S(T) \end{aligned}$$

It can be rewritten also as

$$I_f^{\mathcal{P}}(S) = \frac{1}{\prod_{i \in S} p_i q_i} \sum_{T \subseteq N} \mu(T) f(T) \prod_{i \in S} (T_i - p_i)$$

We have

$$\langle f, z_S \rangle_\mu = I_f^{\mathcal{P}}(S) \prod_{i \in S} \sqrt{p_i q_i}$$

5.4.5 Player-probabilistic based approximation

The definition of the function f can be rewritten as

$$
\begin{aligned}
f(\mathbf{x}) &= \sum_{T \subseteq N} \langle f, z_T \rangle_\mu z_T(\mathbf{x}) \\
&= \sum_{T \subseteq N} I_f^{\mathcal{P}}(T) \left(\prod_{i \in T} \sqrt{p_i q_i} \right) z_T(\mathbf{x}) \\
&= \sum_{T \subseteq N} I_f^{\mathcal{P}}(T) \left(\prod_{i \in T} \sqrt{p_i q_i} \right) \frac{\prod_{i \in T} x_i - p_i}{\prod_{i \in T} \sqrt{p_i q_i}} \\
&= \sum_{T \subseteq N} I_f^{\mathcal{P}}(T) \left(\prod_{i \in T} (x_i - p_i) \right)
\end{aligned}
\tag{5.4}
$$

and the best k^{th} approximation as

$$
f^k(\mathbf{x}) = \sum_{T \subseteq N^{[0,k]}} I_f^{\mathcal{P}}(T) \left(\prod_{i \in T} (x_i - p_i) \right) \tag{5.5}
$$

The equations 5.4 and 5.5 demonstrate that the *Weighted Banzhaf Interaction Index* can be used to construct the k-order approximation of the set function.

5.4.6 Banzhaf-based approximation

5.4.6.1 From Player-Probabilistic to Banzhaf Interaction Index

The Banzhaf Interaction Index is a player-probabilistic interaction index where the probability assigned to the players is

$$
\mathcal{P} = \left\langle \frac{1}{2} : i \in N \right\rangle
$$

We have

$$
\begin{aligned}
I_f^{\mathcal{P}}(S) &= \frac{1}{\prod_{i\in S} p_i q_i} \sum_{T\subseteq N} \mu(T) f(T) \prod_{i\in S} (T_i - p_i) \\
&= \frac{1}{\prod_{i\in S} \frac{1}{2^2}} \sum_{T\subseteq N} \frac{1}{2^n} f(T) \left(\sum_{R\subseteq S} \prod_{i\in R} T_i \prod_{i\in S\setminus R} \left(-\frac{1}{2}\right) \right) \\
&= \frac{2^{2s}}{2^n} \sum_{T\subseteq N} f(T) \sum_{R\subseteq S} \left(-\frac{1}{2}\right)^{s-r} e_R(T) \\
&= \frac{2^s}{2^{n-s}} \sum_{R\subseteq S} \sum_{T\supseteq R} f(T) \left(-\frac{1}{2}\right)^{s-r} e_R(T) \\
&= \frac{2^s}{2^{n-s}} \sum_{R\subseteq S} \sum_{T\supseteq R} f(T) \left(\frac{(-1)^{s-r}}{2^{s-r}} \right) \frac{2^r}{2^r} e_R(T) \\
&= \frac{1}{2^{n-s}} \sum_{R\subseteq S} \sum_{T\subseteq N\setminus R} 2^r f(T\cup R)(-1)^{s-r} e_R(T\cup R)
\end{aligned}
$$

We can use the trick:

$$
\sum_{T\subseteq N\setminus R} 2^r f(T\cup R) = \sum_{T\subseteq N} f(T\cup R)
$$

then

$$
I_f^{\mathcal{P}}(S) = \frac{1}{2^{n-s}} \sum_{T\subseteq N} \sum_{R\subseteq S} (-1)^{s-r} f(T\cup R) e_R(T\cup R)
$$

but

$$
\sum_{R\subseteq S} (-1)^{s-r} f(T\cup R) e_R(T\cup R) = \Delta_S f(T)
$$

is the derivative on S with respect to T, and replacing it in the previous equation, we have

$$
I_f^{\mathcal{P}}(S) = \frac{1}{2^{n-s}} \sum_{T\subseteq N} \Delta_S f(T)
$$

and this is exactly the Banzhaf Interaction Index $I_f^B(S)$.

5.4.6.2 Banzhaf-based approximation

We can use the Banzhaf Interaction Index in the equations 5.4 and 5.5 to obtain the definition of the function and its k^{th} order approximation

$$\begin{aligned} f(\mathbf{x}) &= \sum_{T \subseteq N} \left(\prod_{i \in T} \left(x_i - \frac{1}{2} \right) \right) I_f^B(T) \\ &= \sum_{T \subseteq N} \frac{1}{2^t} I_f^B(T) w_T(\mathbf{x}) \end{aligned} \tag{5.6}$$

and

$$f^k(\mathbf{x}) = \sum_{T \subseteq N^{[0,k]}} \frac{1}{2^t} I_f^B(T) w_T(\mathbf{x}) \tag{5.7}$$

where $w_T(\mathbf{x})$ are the Walsh basis.

5.5 Cardinal-based approximation

5.5.1 Shapley-based approximation

In the article [69] , Charnes et al. demonstrate that the Shapley Value and the Banzhaf Value are the *best linear approximation* of the function f, based on a *weighted mean square error*, selecting special weights.

We can start with the equality

$$\langle f, e_i \rangle_\mu = \langle g, e_i \rangle_\mu \tag{5.8}$$

where e_i is the *unanimity game* for $\{i\}$. Because g is a *linear* approximation, its structure is:

$$g(\mathbf{x}) = \alpha_0 + \sum_{j \in N} \beta_j e_j(\mathbf{x})$$

The equality 5.8 can be rewritten as

$$\langle f, e_i \rangle_\mu = \langle \alpha_0 + \sum_{j \in N} \beta_j e_j, e_i \rangle_\mu$$
$$\langle f, e_i \rangle_\mu - \langle \alpha_0, e_i \rangle_\mu = \langle \sum_{j \in N} \beta_j e_j, e_i \rangle_\mu \tag{5.9}$$

We add two constraints, because they are the properties of the Shapley Value. The first constraint is on the *efficiency* of the power index:

$$f(N) = \sum_{i \in N} \beta_i \tag{5.10}$$

The second constraint is on the property of the weight: the weight for a set depends only on its cardinality

$$\mu(S) = \mu(s)$$

Now, we can introduce the following substitution:

$$\lambda = \langle \alpha_0, e_i \rangle_\mu$$
$$\psi_i = \langle f, e_i \rangle_\mu$$

and to rewrite Equation 5.9 as

$$\zeta_i - \lambda = \sum_{S \in F_i} \mu(s) \left(\sum_{j \in N} \beta_j x_j^S \right)$$

5.5.1.1 Right side

Since $\mu(s)$ depends on the cardinality, we can write:

$$\begin{aligned}
\sum_{S\in F_i}\mu(s)\left(\sum_{i\in N}\beta_i\cdot x_i\right) &= \sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\sum_{S\in F_i^k}\sum_{i\in N}\beta_i\cdot x_i \\
&= \sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\sum_{S\in F_i^k}\left(\beta_i\cdot x_i+\sum_{j\in N\setminus i}\beta_j\cdot x_j\right) \\
&= \sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\sum_{S\in F_i^k}(\beta_i\cdot x_i)+\sum_{k=\lfloor F_i\rfloor+1}^{\lceil F_i\rceil-1}\mu(k)\sum_{S\in F_i^{k+1}}\sum_{j\in N\setminus i}(\beta_j\cdot x_j) \\
&= \sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-1}{k-1}\beta_i+\sum_{k=\lfloor F_i\rfloor+1}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-2}\sum_{j\in N\setminus i}\beta_j
\end{aligned} \tag{5.11}$$

Note that, for 5.10, we have

$$\sum_{j\in N\setminus i}\beta_j=\sum_{j\in N}\beta_j-\beta_i=f(N)-\beta_i$$

and using the property 2.1, the equation 5.11 can be rewritten as

$$\begin{aligned}
&\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-1}{k-1}\beta_i+\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-2}(f(N)-\beta_i) \\
&=\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\left(\binom{\lceil F_i\rceil-1}{k-1}-\binom{\lceil F_i\rceil-2}{k-2}\right)\beta_i+\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-2}f(N) \\
&=\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-1}\beta_i+\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-2}f(N)
\end{aligned} \tag{5.12}$$

Using the substitutions

$$\zeta=\sum_{k=\lfloor F_i\rfloor}^{\lceil F_i\rceil-1}\mu(k)\binom{\lceil F_i\rceil-2}{k-1} \tag{5.13}$$

$$\eta = \sum_{k=\lfloor F_i \rfloor}^{\lceil F_i \rceil - 1} \mu(k) \binom{\lceil F_i \rceil - 2}{k-2} \tag{5.14}$$

the expression 5.12 can be rewritten as

$$\begin{aligned} \psi_i - \lambda &= \zeta \cdot \beta_i + \eta \cdot f(N) \\ \lambda &= \psi_i - \zeta \cdot \beta_i - \eta \cdot f(N) \end{aligned} \tag{5.15}$$

5.5.1.2 Left side

The value of λ can be computed summing over all $i \in N$, and using Equations 5.10 and 5.15:

$$\begin{aligned} \sum_{i \in N} \lambda &= \sum_{i \in N} (\psi_i - \zeta \cdot \beta_i - \eta \cdot f(N)) \\ n\lambda &= \sum_{i \in N} \psi_i - \zeta \cdot f(N) - n \cdot \eta \cdot f(N) \\ \lambda &= \frac{1}{n} \left(\sum_{i \in N} \psi_i - \zeta \cdot f(N) \right) - \eta \cdot f(N) \end{aligned} \tag{5.16}$$

5.5.1.3 Two sides

By substituting Equation 5.16 in 5.15, we obtain

$$\begin{aligned} \zeta \cdot \beta_i &= \psi_i - \lambda - \eta \cdot f(N) \\ &= \psi_i - \frac{1}{n} \left(\sum_{i \in N} \psi_i - \zeta \cdot f(N) \right) + \eta \cdot f(N) - \eta \cdot f(N) \\ &= \frac{1}{n} \left(n \cdot \psi_i - \sum_{i \in N} \psi_i \right) + \frac{\zeta \cdot f(N)}{n} \\ \beta_i &= \frac{1}{n \cdot \zeta} \left(n \cdot \psi_i - \sum_{i \in N} \psi_i \right) + \frac{f(N)}{n} \end{aligned}$$

5.5.1.4 How simplifying $n \cdot \psi_i - \sum_{i \in N} \psi_i$

It is necessary simplifying

$$n \cdot \psi_i - \sum_{i \in N} \psi_i$$

We have

$$\begin{aligned}
\sum_{i \in N} \psi_i &= \sum_{i \in N} \sum_{S \in F_i} \mu(s) f(S) \\
&= \sum_{i \in N} \sum_{k=\lfloor F_i \rfloor}^{\lceil F_i \rceil} \mu(k) \sum_{S \in F_i^k} f(S) \\
&= \sum_{S \in F} s \cdot \mu(s) f(S) \\
&= \sum_{S \in F_i} s \cdot \mu(s) f(S) + \sum_{S \in F_{-i}} s \cdot \mu(s) f(S)
\end{aligned} \tag{5.17}$$

To understand the equality, we can consider the set $\{1, 2, 3, 4\}$ and the families $F_i = \{S \supseteq \{i\}\}$

	1	2	3	4
F_1	{1}	{12},{13},{14}	{123},{124},{134}	{1234}
F_2	{2}	{12},{23},{24}	{123},{124},{234}	{1234}
F_3	{3}	{13},{23},{34}	{123},{134},{234}	{1234}
F_4	{4}	{14},{24},{34}	{124},{134},{234}	{1234}

for each cardinality, the sets are present a number of times equal to the cardinality. We have also that the family F is composed by the family of sets containing i and the family of sets not containing i

$$\begin{aligned}
F &= F_i \cup F_{-i} \\
F_i &= \{S \cup i : S \in F_{-i}\}
\end{aligned}$$

Now

$$
\begin{aligned}
n \cdot \psi_i - \sum_{i \in N} \psi_i &= \sum_{S \in F_i} n \cdot \mu(s) f(S) - \sum_{S \in F_i} s \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \\
&= \sum_{S \in F_i} (n - s) \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \\
&= \sum_{S \in F_{-i}} (n - s - 1) \mu(s + 1) f(S \cup i) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \\
&= \sum_{S \in F_{-i}} ((n - s - 1) \mu(s + 1) f(S \cup i) - s \cdot \mu(s) f(S))
\end{aligned}
$$

We can observe that the sequence of pairs $\langle (n-s), s \rangle$ is equivalent to the sequence $\langle s, (n-s) \rangle$: we have just to reverse the order of the elements. This means that the previous equation can be rewritten also as

$$
\begin{aligned}
n \cdot \psi_i - \sum_{i \in N} \psi_i &= \sum_{S \in F_i} (n - s) \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \\
&= \frac{1}{2} \left(\sum_{S \in F_i} (n - s) \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \right) \\
&+ \frac{1}{2} \left(\sum_{S \in F_i} s \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} (n - s) \cdot \mu(s) f(S) \right) \\
&= \frac{1}{2} \left(\sum_{S \in F_i} n \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} n \cdot \mu(s) f(S) \right) \\
&= \frac{n}{2} \left(\sum_{S \in F_{-i}} (\mu(s + 1) f(S \cup i) - \mu(s) f(S)) \right)
\end{aligned}
$$

5.5.1.5 Determination of β_i

The value of β_i is

$$
\begin{aligned}
\beta_i &= \frac{1}{n \cdot \zeta} \left(n \cdot \psi_i - \sum_{j \in N} \psi_j \right) + \frac{f(N)}{n} \\
&= \frac{1}{n \cdot \zeta} \left(\sum_{S \in F_i} n \cdot \mu(s) f(S) - \sum_{S \subseteq N} s \cdot \mu(s) f(S) \right) + \frac{f(N)}{n} \\
&= \frac{1}{n \cdot \zeta} \left(\sum_{S \in F_i} n \cdot \mu(s) f(S) - \sum_{SF_i} s \cdot \mu(s) f(S) - \sum_{SF_{-i}} s \cdot \mu(s) f(S) \right) + \frac{f(N)}{n} \\
&= \frac{1}{n \cdot \zeta} \left(\sum_{S \in F_i} (n - s) \cdot \mu(s) f(S) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \right) + \frac{f(N)}{n}
\end{aligned}
\tag{5.18}
$$

5.5.1.6 Shapley Value

We consider the space 2^N, $\mu(s)$ defined as

$$
\mu(s) = \binom{n-2}{s-1}^{-1} = \frac{(s-1)!(n-s-1)!}{(n-2)!}
$$

and β_i redefined as

$$
\beta_i = \frac{1}{n \cdot \zeta} \left(\sum_{S \in F_{-i}} (n - s - 1) \cdot \mu(s+1) f(S \cup i) - \sum_{S \in F_{-i}} s \cdot \mu(s) f(S) \right) + \frac{f(N)}{n}
\tag{5.19}
$$

We have

$$\zeta = \sum_{s=1}^{n-1} \binom{n-2}{s-1}^{-1} \binom{n-2}{s-1} = n-1$$

$$s \cdot \mu(s) = s \cdot \frac{(s-1)!(n-s-1)!)}{(n-2)!} = \frac{s!(n-s-1)!}{(n-2)!}$$

$$(n-s-1)\mu(s+1) = (n-s-1)\frac{(s-1)!(n-s-2)!)}{(n-2)!} = \frac{s!(n-s-1)!}{(n-2)!}$$

replacing them in Equation 5.19

$$\begin{aligned}\beta_i &= \frac{1}{n \cdot (n-1)} \left(\sum_{S \in F_{-i}} \frac{s!(n-s-1)!}{(n-2)!} \left(f(S \cup i) - f(S) \right) \right) + \frac{f(N)}{n} \\ &= \sum_{S \in F_{-i}} \left(\frac{s!(n-s-1)!}{n!} \Delta_i f(S) \right) + \frac{f(N)}{n}\end{aligned}$$

We can observed that, except for the term $\frac{f(N)}{n}$, β_i is defined exactly as in the Shapley Value.

Now, we consider the space $N^{[0,k]}$, we have

$$\zeta = \sum_{s=1}^{k-1} \binom{n-2}{s-1}^{-1} \binom{n-2}{s-1} \quad = k-1$$

and substituting it in Equation 5.19

$$\begin{aligned}\beta_i &= \frac{1}{n \cdot (k-1)} \left(\sum_{S \in F_{-i}} \frac{s!(n-s-1)!}{(n-2)!} \left(f(S \cup i) - f(S) \right) \right) + \frac{f(N)}{n} \\ &= \frac{(n-1)}{(k-1)} \sum_{S \in F_{-i}} \left(\frac{s!(n-s-1)!}{n!} \Delta_i f(S) \right) + \frac{f(N)}{n}\end{aligned}$$

5.5.1.7 Banzhaf Value

We consider the space 2^N, $\mu(s) = 1$ (a constant), the definition of β_i can be rewritten as

$$\beta_i = \frac{1}{n \cdot \zeta} \left(\sum_{S \in F_i} (n - s) \cdot f(S) - \sum_{S \in F_{-i}} s \cdot f(S) \right) + \frac{f(N)}{n}$$

with

$$\zeta = \sum_{s=1}^{k-1} \binom{n-2}{s-1} = 2^{n-2}$$

The list of pairs $\langle (n-s), s \rangle$ is equivalent to the list of pairs $\langle s, (n-s) \rangle$, it is sufficient reversing the order of the list. By using the same trick as before, the difference inside the definition can be rewritten as

$$\begin{aligned}
&\sum_{S \in F_i} (n - s) \cdot f(S) - \sum_{S \in F_{-i}} s \cdot f(S) \\
&= \frac{1}{2} \left(\sum_{S \in F_i} (n - s) \cdot f(S) - \sum_{S \in F_{-i}} s \cdot f(S) + \sum_{S \in F_i} s \cdot f(S) - \sum_{S \in F_{-i}} (n - s) \cdot f(S) \right) \\
&= \frac{1}{2} \left(\sum_{S \in F_i} n \cdot f(S) - \sum_{S \in F_{-i}} n \cdot f(S) \right) \\
&= \frac{n}{2} \left(\sum_{S \in F_i} f(S) - \sum_{S \in F_{-i}} f(S) \right) \\
&= \frac{n}{2} \left(\sum_{S \in F_{-i}} \Delta_i f(S) \right)
\end{aligned}$$

The definition of β_i can be rewritten as

$$\beta_i = \frac{1}{n \cdot 2^{n-2}} \frac{n}{2} \left(\sum_{S \in F_{-i}} \Delta_i f(S) \right) + \frac{f(N)}{n}$$
$$= \frac{1}{2^{n-1}} \left(\sum_{S \in F_{-i}} \Delta_i f(S) \right) + \frac{f(N)}{n}$$

and this, except for the term $\frac{f(N)}{n}$, is the definition of the Banzhaf Value.

5.6 Conclusions

In this chapter we have seen that the power and interaction indices are the terms of the *best weighted approximations* of the set function. Different weights define different indices. There is another interpretation of indices: they are the coordinate of the function in $\mathcal{L}(2^N)$ when it is selected a specific basis: for example, the Möbius coefficients are the coordinates of function when it is used the *unanimity games* basis, the Shapley Interaction Indices are the coordinates when the basis are $\beta^{|K|}_{|S \cap T|}$, the Banzhaf Interaction Indices are the coordinates when the basis are $\frac{1}{2^t}(-1)^{|T \setminus S|}$, etc. We have seen, also, that if the basis is orthogonal, to approximate the function with another function degree k it is enough to use the basis with degree less or equal to k.

Chapter 6

New Interaction Indices

6.1 Introduction

In the Chapters 3 and 5 we have introduced some of the most used power and interaction indices (Shapley, Chaining, Banzhaf, Weighted Banzhaf Values and related Interaction Indices). All of them are members of probabilistic interaction indices. These indices are defined on the complete subset's lattice (on sets with all possible cardinality) and they define a transform, that is, it is possible to recreate the original function from the values of the indices computed on all lattice's nodes.

Sometimes, it is useful analyzing only a part of the lattice, and more often, this part is composed of the level k (that contains the sets with cardinality k) or the levels in the range $[k_{\min}, k_{\max}]$.

In this case, we can use a *reduced* version of the previous indices that we call K-*Interaction Indices*. In this chapter, we will prove the relation between the original indices and the new ones.

6.2 Selecting some lattice's levels

The general structure of a probabilistic interaction index is

$$I_\xi(S) = \sum_{T \subseteq N} p_S(T) \Delta_S \xi(T)$$

with $\langle p_S : S \subseteq N \rangle$ a probability distribution. This expression can be rewritten also as

$$I_\xi(S) = \sum_{k=0}^{n-s} I_\xi(k, S) \tag{6.1}$$

$$I_\xi(k, S) = \sum_{\substack{T \subseteq N \\ |T|=k}} p_S(T) \Delta_S \xi(T)$$

where we have split the sum into terms specific for each set's cardinality.

If we are interested only on sets with the cardinality in the range $[k_{\min}, k_{\max}]$, the 6.1 can be rewritten as

$$I_\xi(S) = \sum_{k=0}^{k_{\max}-1} I_\xi(k, S) + \sum_{k=k_{\min}}^{k_{\max}} I_\xi(k, S) + \sum_{k=k_{\max}+1}^{n-s} I_\xi(k, S)$$

If $I_\xi(k, S) = 0$ for $k < k_{\min}$ and $k > k_{\max}$ we obtain

$$I_\xi(S) = \sum_{k=k_{\min}}^{k_{\max}} I_\xi(k, S)$$

The excluded terms of I_ξ can be zero for several reason, but two reliable ones are:

1. $\Delta_S \xi(T) = 0$: the derivative is 0, that is, the function is constant, or zero
2. $p_S(T) = 0$: the weighs for these sets are zero

Obviously, in general, the function will be not zero, but we suppose to *force* it to be zero.

The problem with the approach (1.) is that, however, we are still considering the function defined on the entire lattice because the weights used are not a probability distribution

$$\sum_{\substack{T\subseteq N \\ k_{\min}\leq|T|\leq k_{\max}}} p_S(T) \neq 1 \quad \text{with } 0 < k_{\min} \text{ or } k_{\max} < n - s$$

The approach (2.), instead, is *partially* correct because $\langle p_S \rangle$ for *definition* must be a probability distribution, also if several weights are zero. The problem is that the weights are used in the definition of the *weighted inner product* and we know that, if there are weights with value zero, it has not the properties of an inner product.

But, if we *force* the function to be 0 on these sets *and* we assign them the weight 0, the *weighted inner product* became a valid inner product.

Now, we can change the definition of Interaction Indices to adapt them to the new approach.

6.3 K-Cardinal-Probabilistic Interaction Indices

The simplest indices to change are the cardinal-probabilistic indices.

In these indices, the weight depends only on the set's cardinality

$$p_S(K) = p_S(k)$$

with

$$\sum_{k=0}^{n-s} \binom{n-s}{k} p_S(k) = 1$$

The factor

$$\mathcal{C}_k = p_S(k)\binom{n-s}{k}^{-1}$$

is the *probability* to select the lattice's level k, with $\mathcal{C}$ the probability distribution assigned to all levels.

Excluding the levels not in the range $[k_{\min}, k_{\max}]$, we need to redistribute the excluded probabilities to the remaining levels. The most simple approach is to change the remaining probabilities proportionally to their values

$$\mathcal{C}'_k = \frac{\mathcal{C}_k}{\mathcal{C}_{[k_{\min},k_{\max}]}} \quad \forall k \in [k_{\min}, k_{\max}]$$

with

$$\mathcal{C}_{[k_{\min},k_{\max}]} = \sum_{k=k_{\min}}^{k_{\max}} \mathcal{C}_k$$

Another method could be to distribute the missing probability in a uniform way to the remaining levels

$$\mathcal{C}'_k = \mathcal{C}_k + \left(\frac{1 - \mathcal{C}_{[k_{\min},k_{\max}]}}{k_{\max} - k_{\min} + 1}\right) \quad \forall k \in [k_{\min}, k_{\max}]$$

but this is not a good solution, because it does not maintain the ratio between the levels weights.

6.3.1 K-Shapley Interaction Index

The weight used in the Shapley Interaction Index is

$$p_S(k) = \frac{(n-s-k)!k!}{(n-s+1)!} = \frac{1}{n-s}\binom{n-s}{k}^{-1}$$

where

$$\mathcal{C}_k = \frac{1}{n-s}$$

and

$$\mathcal{C}_{[k_{\min},k_{\max}]} = \frac{k_{\max} - k_{\min} + 1}{n-s}$$

The probability can be changed in

$$\mathcal{C}'_k = \frac{n-s}{k_{\max} - k_{\min} + 1}\mathcal{C}_k = \frac{1}{k_{\max} - k_{\min} + 1}$$

6.3.2 K-Chaining Interaction Index

The weight used in Chaining Interaction Index is

$$p_S(k) = \binom{s-1+k}{s-1}\binom{n-s}{k}^{-1}$$

where the probability to select the level k is

$$\mathcal{C}_k = \binom{s-1+k}{s-1}$$

We can change it in

$$\mathcal{C}'_k = \frac{n-s}{k_{\max} - k_{\min} + 1}\mathcal{C}_k$$

6.4 K-Player-Probabilistic Interaction Indices

In these indices, the weight depends only on the probability p_i assigned to the members of the set

$$p_S(T) = \prod_{j \in T} p_j \prod_{j \in N \setminus S \setminus T} (1 - p_j)$$

with

$$\sum_{T \subseteq N \setminus S} p_S(T) = 1$$

By selecting only the sets with cardinality in the range $[k_{\min}, k_{\max}]$, we are using only the quota

$$\mathcal{P}_{[k_{\min}, k_{\max}]} = \sum_{\substack{T \subseteq N \setminus S \\ k_{\min} \leq t \leq k_{\max}}} p_S(T)$$

of the total probability. We can change the weight assigned to the set in

$$p_S(T)' = \frac{p_S(T)}{\mathcal{P}_{[k_{\min}, k_{\max}]}}$$

The problem is to find a *compact* formula for $\mathcal{P}_{[k_{\min}, k_{\max}]}$. Unfortunately, this is not in general possible because the *closed* formula contains a number of terms similar to the number of sets analyzed.

6.4.1 K-Banzhaf Interaction Index

The Banzhaf Interaction Index is a player-probabilistic interaction index where the weight for each set is

$$p_S(T) = \frac{1}{2^{n-s}}$$

In this case, the value of $\mathcal{P}_{[k_{\min},k_{\max}]}$ is:

$$\mathcal{P}_{[k_{\min},k_{\max}]} = \sum_{k=k_{\min}}^{k_{\max}} \binom{n-s}{k}$$

6.5 Approximate algorithms

In Chapter 4 we have already described the general structure of the algorithms and their extension to be used with a selected range of cardinality.

Here we can observe that it is not necessary to change the probability distributions used as parameters in the algorithms because the normalization of the cumulative values is based on *counting* the number of sets or permutations used and this considers automatically the cardinality range.

6.6 Closed formula for K-Banzhaf Value

6.6.1 First order approximation

Starting from the general solution, it is possible to find the closed formulas for the K-Banzhaf Value.

We can start with the equalities

$$\langle e_T, e_\emptyset \rangle = \sum_{S \in F} e_T(S) = |F_T|$$

$$\langle e_T, e_i \rangle = \sum_{S \in F} e_T(S) e_T(i) = |F_{T \cup i}|$$

The second list of equalities is

$$\langle e_T, e_\emptyset \rangle = \langle g_T, e_\emptyset \rangle \tag{6.2}$$

$$\langle e_T, e_i \rangle = \langle g_T, e_i \rangle \tag{6.3}$$

Since the equality 6.2 is true for each T, it is true also for $T \cup i$ and for $T \setminus i$

$$\begin{aligned} \langle e_{T \cup i}, e_i \rangle &= \langle g_{T \cup i}, e_i \rangle \\ \langle e_{T \setminus i}, e_i \rangle &= \langle g_{T \setminus i}, e_i \rangle \end{aligned}$$

For the linearity of the inner product, the equality is true also for

$$\langle e_{T \cup i} - e_{T \setminus i}, e_i \rangle = \langle g_{T \cup i} - g_{T \setminus i}, e_i \rangle$$

but these differences are the *first order derivative* on i of e_T, and g_T

$$\langle \Delta_i e_T, e_i \rangle = \langle \Delta_i g_T, e_i \rangle \tag{6.4}$$

with $g_T(\mathbf{x})$ the *first order approximation* of the *unanimity base* $e_T(\mathbf{x})$:

$$g_T(\mathbf{x}) = \alpha_0^T + \sum_{i \in N} \beta_i^T x_i \tag{6.5}$$

6.6.2 Determination of β_i

6.6.2.1 Right side

Because $\Delta_i g_T(\mathbf{x}) = \beta_i^T$, we have:

$$\sum_{\mathbf{x} \in F_{-i}} \Delta_i g_T(\mathbf{x}) = \sum_{\mathbf{x} \in F_{-i}} \beta_i^T = |F_{-i}| \beta_i^T$$

6.6.2.2 Left side

We have already seen that $\Delta_i e_T(\mathbf{x}) = e_{T \setminus i}(\mathbf{x})$, hence

$$\sum_{\mathbf{x}\in F_{-i}} \Delta_i e_T(\mathbf{x}) = \sum_{\mathbf{x}\in F_{-i}} e_{T\setminus i}(\mathbf{x}) = |F_{-i}^{T\setminus i}| = |F_T| \tag{6.6}$$

6.6.2.3 Two sides

Now, we have

$$|F_{-i}|\beta_i^T = |F_T|$$
$$\beta_i^T = \frac{|F_T|}{|F_{-i}|}$$

It is possible to observe that the value of β_i^T depends only on the cardinality of T, i.e. β_i^T is a *level dependent* coefficient and it can be written as $\beta(t)$:

$$\begin{array}{ll} F = 2^N & \beta(t) = \frac{2^{n-t}}{2^{n-1}} = \frac{1}{2^{t-1}} \\ F = N^k & \beta(t) = \frac{\binom{n-k}{k-k}}{\binom{n-1}{k-1}} = \frac{(k-1)_{t-1}}{(n-1)_{t-1}} \\ F = N^{[0,k]} & \beta(t) = \frac{\binom{n-t}{k-t}^*}{\binom{n-1}{k-1}^*} \end{array}$$

Substituting $\beta(t)$ in 6.5, we have

$$g_T(\mathbf{x}) = \alpha_0^T + \sum_{i\in N} \beta_i^T x_i$$

6.6.3 Determination of α_0

6.6.3.1 Right side

We have

$$\sum_{\mathbf{x}\in F}\left(\alpha_0^T+\beta(t)\sum_{i\in T}x_i\right)$$
$$=\sum_{\mathbf{x}\in F}\alpha_0^T+\beta(t)\sum_{\mathbf{x}\in F}\sum_{i\in T}x_i$$

$$=|F|\cdot\alpha_0^T+\beta(t)\cdot t\cdot|F_i|$$
$$=|F|\cdot\alpha_0^T+\frac{|F_T|}{|F_{-i}|}\cdot t\cdot|F_i|$$
$$=|F|\cdot\alpha_0^T+t\cdot|F_T|$$

6.6.3.2 Left side

$$\sum_{\mathbf{x}\in F}e_T(\mathbf{x})=|F_T| \tag{6.7}$$

6.6.3.3 Two sides

Now, we have

$$|F|\alpha_0^T+t|F_T|=|F_T|$$
$$|F|\alpha_0^T=|F_T|-t|F_T|$$
$$\alpha_0^T=\frac{|F_T|}{|F|}-t\frac{|F_T|}{|F|}$$
$$=-(t-1)\frac{|F_T|}{|F|}$$

Given α_0^T depends only on t, we can write $\alpha(t)$

$$F=2^N \qquad \alpha(t)=-(t-1)\tfrac{1}{2^t}$$
$$F=N^k \qquad \alpha(t)=-(t-1)\frac{\binom{n-t}{k-t}}{\binom{n}{k}}=-(t-1)\frac{(k)_t}{(n)_t}$$
$$F=N^{[0,k]} \qquad \alpha(t)=-(t-1)\frac{\binom{n-t}{k-t}^*}{\binom{n}{k}^*}$$

Replacing $\alpha(t), \beta(t)$ in 6.5, we have

$$g_T(\mathbf{x}) = \alpha(t) + \beta(t) \sum_{i \in N} x_i$$

6.6.4 Wrap up

The function $g_T(\mathbf{x})$ is defined as

$$\begin{aligned} g_T(\mathbf{x}) &= \alpha(t) + \beta(t) \sum_{i \in N} x_i \\ \alpha(t) &= -(t-1)\frac{|F_T|}{|F|} \\ \beta(t) &= \frac{|F_T|}{|F_{-i}|} \end{aligned} \tag{6.8}$$

6.6.5 General expression of the first order approximation in the Möbius basis

The function:

$$f(\mathbf{x}) = \sum_{T \in F} a_T \cdot e_T(\mathbf{x})$$

can be approximated replacing $e_T(\mathbf{x})$ with the first order approximation 6.8:

$$
\begin{aligned}
g(\mathbf{x}) &= \sum_{T\in F} a_T \cdot g_T(\mathbf{x}) \\
&= \sum_{T\in F} a_T \cdot \Big(\alpha(t) + \beta(t) \sum_{i\in T} x_i\Big) \\
&= \sum_{T\in F} a_T \cdot \alpha(t) + \sum_{T\in F} a_T \cdot \beta(t) \sum_{i\in T} x_i \\
&= \sum_{T\in F} a_T \cdot \alpha(t) + \sum_{i\in N} \Big(\sum_{T\in F} \beta(t) \cdot a_T\Big) x_i \\
&= \alpha_0 + \sum_{i\in N} \beta_i x_i
\end{aligned}
$$

(note that if $i \notin T \rightarrow x_i = 0$) where

$$
\begin{aligned}
\alpha_0 &= \sum_{T\in F} a_T \cdot \alpha(t) = \sum_{T\in F} a_T \cdot \left(-(t-1)\frac{|F_T|}{|F|}\right) = -\frac{1}{|F|}\sum_{T\in F} a_T \cdot (t-1) \cdot |F_T| \\
\beta_i &= \sum_{T\in F_i} a_T \cdot \beta(t) = \sum_{T\in F_i} a_T \cdot \frac{|F_T|}{|F_{-i}|} = \frac{1}{|F_{-i}|}\sum_{T\in F_i} a_T \cdot |F_T|
\end{aligned}
$$

and

$$
\begin{aligned}
F = 2^N \quad \alpha_0 &= -\frac{1}{2^n}\sum_{T\in 2^N} a_T \cdot (t-1) \cdot 2^{n-t} = -\sum_{T\in 2^N} \frac{t-1}{2^t} a_T \\
\beta_i &= \frac{1}{2^{n-1}}\sum_{T\in 2_i^N} a_T \cdot 2^{n-t} = \sum_{T\in 2_i^N} \frac{1}{2^{t-1}} a_T \\
F = N^k \quad \alpha_0 &= -\frac{1}{\binom{n}{k}}\sum_{T\in N^k} a_T \cdot \binom{n-1}{k-1} = \frac{k}{n}\sum_{T\in N^k} a_T \\
\beta_i &= \frac{1}{\binom{n-1}{k-1}}\sum_{T\in N^k{}_i} a_T \cdot \binom{n-k}{k-k} = \frac{1}{\binom{n-1}{k-1}}\sum_{T\in N^k{}_i} a_T \\
F = N^{[0,k]} \quad \alpha_0 &= -\frac{1}{\binom{n}{k}^*}\sum_{T\in N^{[0,k]}} (t-1) \cdot \binom{n-t}{k-t}^* a_T \\
\beta_i &= \frac{1}{\binom{n-1}{k-1}^*}\sum_{T\in N^{[0,k]}{}_i} \binom{n-t}{k-t}^* a_T
\end{aligned}
$$

6.6.6 General expression of α and β_i in terms of f and $\Delta_i f$

6.6.6.1 Expression of β_i in terms of $\Delta_i f$

From Equation 6.4, we have:

$$\sum_{\mathbf{x}\in F_{-i}} \Delta_i g(\mathbf{x}) = \sum_{\mathbf{x}\in F_{-i}} \Delta_i f(\mathbf{x})$$

and, for the linearity of $g(\mathbf{x})$, $\Delta_i g(\mathbf{x}) = \beta_i$, so, the left side is $|F_{-i}|\beta_i$. It follows that

$$\sum_{\mathbf{x}\in F_{-i}} \Delta_i g(\mathbf{x}) = \sum_{\mathbf{x}\in F_{-i}} \beta_i = |F_{-i}|\beta_i$$

and

$$\beta_i = \frac{1}{|F_{-i}|} \sum_{\mathbf{x}\in F_{-i}} \Delta_i f(\mathbf{x})$$

where

$$\begin{aligned}
F = 2^N \quad & \beta_i = \frac{1}{2^{n-1}} \sum_{\mathbf{x}\in 2^{N\setminus i}} \Delta_i f(\mathbf{x}) \\
F = N^k \quad & \beta_i = \frac{1}{\binom{n}{k}} \sum_{\mathbf{x}\in N^k{}_{-i}} \Delta_i f(\mathbf{x}) \\
F = N^{[0,k]} \quad & \beta_i = \frac{1}{\binom{n-1}{k-1}^*} \sum_{\mathbf{x}\in N^{[0,k]}{}_{-i}} \Delta_i f(\mathbf{x})
\end{aligned}$$

when $k = n$ we have $N^{[0,k]} = 2^N$ and the definitions of β_i are equal.

It is possible to observe that for the family 2^N the definition of β_i is exactly the definition of the Banzhaf Value. This means that the Banzhaf Value is

(part) of the *best linear approximation* of the function, using the *mean square error* as error metric.

6.6.6.2 Expression of α in terms of f

The value of α can be computed from the equation 6.2

$$\begin{aligned}
\sum_{S \in F} f(S) &= \sum_{S \in F} g(S) \\
&= \sum_{S \in F} \Big(\alpha + \sum_{i \in N} \beta_i x_i \Big) \\
&= |F|\alpha + \sum_{S \in F} \Big(\sum_{i \in N} \beta_i x_i \Big) \\
|F|\alpha &= \sum_{S \in F} f(S) - \sum_{S \in F} \Big(\sum_{i \in N} \beta_i x_i \Big) \\
\alpha &= \frac{1}{|F|} \sum_{S \in F} \Big(f(S) - \sum_{i \in N} \beta_i x_i \Big)
\end{aligned}$$

where

$$\begin{aligned}
F = 2^N \quad & \alpha_0 = \frac{1}{2^n} \sum_{S \in 2^N} \Big(f(S) - \sum_{i \in N} \beta_i x_i \Big) \\
F = N^k \quad & \alpha_0 = \frac{1}{\binom{n}{k}} \sum_{S \in N^k} \Big(f(S) - \sum_{i \in N} \beta_i x_i \Big) \\
F = N^{[0,k]} \quad & \alpha_0 = \frac{1}{\binom{n}{k}^*} \sum_{S \in N^{[0,k]}} \Big(f(S) - \sum_{i \in N} \beta_i x_i \Big)
\end{aligned}$$

Obviously, when $k = n$, $N^{[0,k]} = 2^N$ and the definitions of α are equal.

6.6.7 Second order approximation

The 2^{nd} order approximation must satisfy the equalities 6.3, 6.4 and

$$\langle e_T, e_{ij} \rangle = \langle g_T, e_{ij} \rangle \tag{6.9}$$

with

$$g_T(\mathbf{x}) = \alpha_0^T + \sum_{i \in N} \beta_i^T x_i + \sum_{ij \subseteq N} \gamma_{ij}^T x_i x_j \tag{6.10}$$

6.6.8 Determination of γ_{ij}

6.6.8.1 Right side

Since $\Delta_{ij} g_T(\mathbf{x}) = \gamma_{ij}^T$, we have

$$\sum_{\mathbf{x} \in F_{-ij}} \Delta_{ij} g_T(\mathbf{x}) = \sum_{\mathbf{x} \in F_{-ij}} \gamma_{ij}^T = |F_{-ij}| \gamma_{ij}^T$$

6.6.8.2 Left side

We have already seen that $\Delta_{ij} e_T(\mathbf{x}) = e_{T \setminus ij}(\mathbf{x})$, hence

$$\sum_{\mathbf{x} \in F_{-ij}} \Delta_{ij} e_T(\mathbf{x}) = \sum_{\mathbf{x} \in F_{-ij}} e_{T \setminus ij}(\mathbf{x}) = |F_{-ij}^{T \setminus ij}| = |F_T| \tag{6.11}$$

6.6.8.3 Two sides

Now, we have

$$|F_{-ij}| \gamma_{ij}^T = |F_T|$$
$$\gamma_{ij}^T = \frac{|F_T|}{|F_{-ij}|}$$

It is possible to observe that the value of γ_{ij}^T depends only on the cardinality of T, i.e. γ_{ij}^T is a *level dependent* coefficient and it can be written as $\gamma(t)$:

$$\begin{aligned} F &= 2^N & \gamma(t) &= \frac{2^{n-t}}{2^{n-2}} = \frac{1}{2^{t-2}} \\ F &= N^k & \gamma(t) &= \frac{\binom{n-k}{k-k}}{\binom{n-2}{k-2}} = \frac{(k-2)_{t-2}}{(n-2)_{t-2}} \\ F &= N^{[0,k]} & \gamma(t) &= \frac{\binom{n-t}{k-t}^*}{\binom{n-2}{k-2}^*} \end{aligned}$$

By substituting $\gamma(t)$ in 6.10, we have

$$g_T(\mathbf{x}) = \alpha_0^T + \sum_{i \in N} \beta_i^T x_i + \sum_{ij \subseteq N} \gamma(t) x_i x_j$$

6.6.9 Determination of β_i

The value of β_i can be evaluated using 6.4

6.6.9.1 Left side

$$\begin{aligned} \sum_{\mathbf{x} \in F_{-i}} \Delta_i g_T(\mathbf{x}) &= \sum_{\mathbf{x} \in F_{-i}} \beta_i^T + \sum_{j \in T \setminus i} \sum_{\mathbf{x} \in F_{-i}^j} \gamma(t) \\ &= |F_{-i}| \beta_i^T + (t-1) |F_{-i}^j| \gamma(t) \\ &= |F_{-i}| \beta_i^T + (t-1) |F_{-ij}| \frac{|F_T|}{|F_{-ij}|} \\ &= |F_{-i}| \beta_i^T + (t-1) |F_T| \end{aligned}$$

6.6.9.2 Right side

We can use the 6.4

6.6.9.3 Two sides

Now, we have

$$|F_{-i}|\beta_i^T + (t-1)|F_T| = |F_T|$$
$$\beta_i^T = -(t-2)\frac{|F_T|}{|F_{-i}|}$$

β_i^T depends only on t, than, it can written as $\beta(t)$.

$$\begin{array}{ll} F = 2^N & \beta(t) = -(t-2)\frac{1}{2^{t-1}} \\ F = N^k & \beta(t) = -(t-2)\frac{(k-1)_{t-1}}{(n-1)_{t-1}} \\ F = N^{[0,k]} & \beta(t) = -(t-2)\frac{\binom{n-t}{k-t}^*}{\binom{n-1}{k-1}^*} \end{array}$$

By substituting $\beta(t), \gamma(t)$ in 6.10, we have

$$g_T(\mathbf{x}) = \alpha_0^T + \sum_{i \in N} \beta(t) x_i + \sum_{ij \subseteq N} \gamma(t) x_i x_j$$

6.6.10 Determination of α_0

6.6.10.1 Left side

$$\begin{aligned} \sum_{\mathbf{x} \in F} g_T(\mathbf{x}) &= \sum_{\mathbf{x} \in F} \left(\alpha_0^T + \sum_{i \in T} \beta(t) x_i + \sum_{ij \subseteq T} \gamma(t) x_i x_j \right) \\ &= \sum_{\mathbf{x} \in F} \alpha_0^T + \beta(t) \sum_{\mathbf{x} \in F} \sum_{i \in T} x_i + \gamma(t) \sum_{\mathbf{x} \in F} \sum_{ij \subseteq T} x_i x_j \\ &= |F| \alpha_0^T + \beta(t) \sum_{i \in T} |F_i| + \gamma(t) \sum_{ij \subseteq T} |F_{ij}| \\ &= |F| \alpha_0^T + \beta(t) t |F_i| + \gamma(t) \frac{1}{2} t(t-1) |F_{ij}| \end{aligned}$$

6.6.10.2 Right side

We can use the 6.7

6.6.10.3 Two sides

$$|F|\alpha_0^T + \beta(t)t|F_i| + \gamma(t)\frac{1}{2}t(t-1)|F_{ij}| = |F_T|$$

$$\begin{aligned}
|F|\alpha_0^T &= |F_T| - \beta(t)t|F_i| - \gamma(t)\frac{1}{2}t(t-1)|F_{ij}| \\
&= |F_T| + (t-2)\frac{|F_T|}{|F_{-i}|}t|F_i| - \frac{|F_T|}{|F_{-ij}|}\frac{1}{2}t(t-1)|F_{ij}| \\
&= |F_T| + t(t-2)|F_T| - \frac{1}{2}t(t-1)|F_T| \\
&= \frac{1}{2}(t-1)(t-2)|F_T|
\end{aligned}$$

$$\alpha_0^T = \frac{1}{2}(t-1)(t-2)\frac{|F_T|}{|F|}$$

Since α_0^T depends only on t, it can written as $\alpha(t)$.

$$\begin{array}{ll}
F = 2^N & \alpha(t) = \frac{(t-1)(t-2)}{2^{t+1}} \\
F = N^k & \alpha(t) = \frac{1}{2}(t-1)(t-2)\frac{(k)_t}{(n)_t} \\
F = N^{[0,k]} & \alpha(t) = \frac{1}{2}(t-1)(t-2)\frac{\binom{n-t}{k-t}^*}{\binom{n}{k}^*}
\end{array}$$

By substituting $\alpha(t), \beta(t), \gamma(t)$ in 6.10, we have

$$g_T(\mathbf{x}) = \alpha(t) + \sum_{i \in N}\beta(t)x_i + \sum_{ij \subseteq N}\gamma(t)x_i x_j$$

6.6.11 Wrap up

The function $g_T(\mathbf{x})$ is defined as

$$
\begin{aligned}
g_T(\mathbf{x}) &= \alpha(t) + \beta(t) \sum_{i \in N} x_i + \gamma(t) \sum_{ij \subseteq T} x_i x_j \\
\alpha(t) &= \frac{1}{2}(t-1)(t-2)\frac{|F_T|}{|F|} \\
\beta(t) &= -(t-2)\frac{|F_T|}{|F_{-i}|} \\
\gamma(t) &= \frac{|F_T|}{|F_{-ij}|}
\end{aligned}
\tag{6.12}
$$

6.6.12 General expression of the second order approximation in the Möbius basis

The function

$$
f(\mathbf{x}) = \sum_{T \in F} a_T \cdot e_T(\mathbf{x})
$$

can be approximated by replacing $e_T(\mathbf{x})$ with the second order approximation 6.12

$$
\begin{aligned}
g(\mathbf{x}) &= \sum_{T \in F} a_T \cdot g_T(\mathbf{x}) \\
&= \sum_{T \in F} a_T \cdot \Big(\alpha(t) + \beta(t) \sum_{i \in N} x_i + \gamma(t) \sum_{ij \subseteq T} x_i x_j\Big) \\
&= \sum_{T \in F} a_T \cdot \alpha(t) + \sum_{T \in F} a_T \cdot \beta(t) \sum_{i \in N} x_i + \sum_{T \in F} a_T \cdot \gamma(t) \sum_{ij \subseteq T} x_i x_j \\
&= \sum_{T \in F} a_T \cdot \alpha(t) + \sum_{i \in N} \Big(\sum_{T \in F} a_T \cdot \beta(t)\Big) x_i + \sum_{ij \subseteq N} \Big(\sum_{T \in F} a_T \cdot \gamma(t)\Big) x_i x_j \\
&= \alpha_0 + \sum_{i \in N} \beta_i x_i + \sum_{ij \subseteq N} \gamma_{ij} x_i x_j
\end{aligned}
$$

where

$$\alpha_0 = \sum_{T \in F} a_T \cdot \alpha(t) = \sum_{T \in F} a_T \cdot \frac{1}{2}(t-1)(t-2)\frac{|F_T|}{|F|}$$
$$\beta_i = \sum_{T \in F} a_T \cdot \beta(t) = \sum_{T \in F} -a_T \cdot (t-2)\frac{|F_T|}{|F_{-i}|}$$
$$\gamma_{ij} = \sum_{T \in F} a_T \cdot \gamma(t) = \sum_{T \in F} a_T \cdot \frac{|F_T|}{|F_{-ij}|}$$

6.7 Conclusions

In this chapter we have defined the *K-Power Indices*, a variant of the probabilistic power indices specialized to approximate the function only on sets with the selected cardinality k. We have seen how to modify the weights used in the original definitions to adapt them to the new definition. We have found the closed formula for the first and second degree of the *K-Banzhaf Value*. We have also seen that the current algorithms are already able to support the new class of indices.

Chapter 7

Feature Partitioning and Co-Training

7.1 Introduction

Coalitional Game Theory, and in particular the power indices (Shapley Value, Banzhaf Value, etc) have already been used – as already mentioned in the introductory chapter – for feature selection. Sun et al. [127], Kulynych et al [129] have used the Banzhaf Value, Cohen et al. in [125] and [126], Liu et al [131], Sun et al. [128], Gore et al. [130], Mokdad et al. [132] have used the Shapley Value, and Mikenina et al. [124] have used the first degree coefficients of the Möbius Transform.

This makes sense because the power indices are the *best first order* (weighted) approximation of the set function: a way to approximate the maximum of the function with a reduced number of elements is to select the elements with the highest values in the first order approximation.

In the area of the *Multi View Learning* sometimes it is necessary to split a dataset into two o more views, as explained in the introductory chapter. The challenge consists in selecting the features for each view. There are several approaches available [134, 144, 145] but in this chapter we will use an approach based on the Coalitional Game Theory.

The *feature partitioning* problem consists in subdividing a set of features in two o more subsets (the *views*) subject to some desirable conditions. A

commonly accepted synthesis of those conditions are the following, obtained from Blum and Mitchell [148] (for predicting power and consistency) and Balcan et al. [149] (for the relative teaching power):

- each view must be *sufficient*: using the view it is possible to obtain a good predictor, thus one of the objectives is to maximize the *predicting power* of individual views
- the views must be *consistent*: the predictors obtained from each view must predict the same class, if the prediction has a high confidence
- *relative teaching power*: (this denomination is ours) the views must be able to act as teachers to one another. The *relative teaching power* of a first view w.r.t. a second view, as *the fraction of times the former is capable to predict the correct class with a confidence higher than a wrong class predicted by the latter*; it corresponds to holding unique information and being able to transfer it.

Within semi-supervised learning, the teaching power of one feature over another can be estimated within each pair of features form the initially labelled examples. For each unordered pair of features there are two values of the teaching power, corresponding to the two ordered pairs (the first feature teaching to the second and the second teaching to the first). The view teaching power for a given arrangement of features in two views can be estimated equally.

Hereafter we consider separately

- the problem of optimizing the partitioning for a setting where the views act separately, i.e. are used to train separate models whose output (whose predictions) are then summarized by an aggregator (e.g. majority voting): in this case only the sufficiency (i.e. prediction power) requirement and the consistency requirement are considered; this case will be dealt with in Subsection 7.2
- from the problem of optimizing the partitioning when the views are expected to take part to co-training procedure: in this case also the teaching power requirement is considered; this case will be dealt with in Subsection 7.3

7.2 Feature partitioning in multi-view learning

7.2.0.1 Prediction power

Coalitional Game Theory offers two useful concepts to this respect:

1. the *power indices*, which quantify the prediction's ability of the features
2. the *interaction indices*, which quantify the constructive collaboration or disruptive interference on the prediction between two (or among three or more) features

Within semi-supervised learning, power indices and interaction indices of and between/among features can be estimated from the initially labelled examples.

The concepts of power indices and interaction indices can be used to support the prediction power requirement and the relative teaching power requirement.

With respect to the predicting power, the *feature partitioning* problem can be described as an *optimization problem* with the following general properties

1. each view must contain features with the best prediction ability (features with the best power index)
2. features that collaborate must stay in the same view (features with the best interaction index)
3. features that interfere disruptively must stay in different views (features with the worst interaction index)

As explained in the previous chapters, the power index and the interaction index correspond to a projection of the first or second degree of the overall n-degree polynomial representing the whole set function (the characteristic function of the game). By projecting the n-degree polynomial onto a second degree approximation, we achieve a reduction of the complexity of the problem: in the new representation the features become the nodes of a graph,

endowed with a weight (the coefficient of the first order terms) and their interactions become the weight of the arcs of that graph (they are the coefficients of the second order terms); the problem of splitting into two views is mapped into a graph-cut problem.

One can formulate the graph cut problem as follows. Each part of the cut represents a view, whose predicting power which can be approximated by the sum of the first and second order terms belonging to that view. This sum, for an individual view, consists of the first order weights of the nodes belonging to the view and of the second order interactions between all the pairs of those nodes. Maximizing the power of both views can be expressed as maximizing the sum (or any monotonic function of the arguments) of the predicting power of the individual views. As already mentioned in the introduction, this is equivalent to minimizing the second order terms that turn out to be cut because the join two nodes belonging to distinct views.

Notice that estimating/verifying the interaction among features or views (or the relative teaching power of pairs of features or of pairs of views) requires an effort which increases quadratically with the number of objects. A way for reducing this complexity, for instance for the views, is illustrated here after.

If we have $v \geq 2$ views, there are $v(v-1)$ relations to consider.

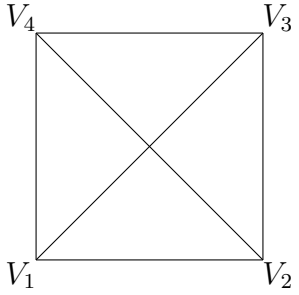


Figure 7.1: Views interactions

A possible solution is

1. to assign an arbitrary order to the views
2. each view is compared only with the next one

3. the last view is compared with the first one

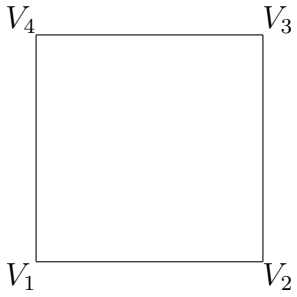


Figure 7.2: Reduced views interactions

This reduces the number of interactions to consider to n.

7.2.1 The optimization problem

The optimization problem can be expressed as

$$\max_{\mathcal{V}(N)} \quad \sum_{p=1}^{v} \left(\phi_\xi(V_p) + I_\xi^{(2)}(V_p) \right) - \sum_{\substack{p=1 \\ q=p\oplus 1}}^{v} \sum_{\substack{i\in V_p \\ j\in V_q}} I_\xi(ij)$$

where

$$\begin{aligned} &\mathcal{V}(N) = \langle V_1, \ldots, V_v \rangle \\ &V_p \cap V_q = \emptyset \quad \forall p \neq q \\ &\bigcup_{p=1}^{v} V_p = N \end{aligned}$$

is a partition of the elements in N in v views $V_1, \ldots, V_v$,

$$\phi_\xi(V_p) = \sum_{i\in V_p} \phi_\xi(i)$$

is the sum of power indices of the elements in the view V_p

$$I_\xi^{(2)}(V_p) = \sum_{ij \subseteq V_p} I_\xi(ij)$$

is the sum of the 2^{nd}-order interaction indices of the elements in the same view V_p (the previous one and this are the 2^{nd}-order approximation of the set function) and

$$\sum_{\substack{i \in V_p \\ j \in V_q}} I_\xi(ij)$$

is the sum of the 2^{nd}-order interaction indices of the elements in different views (V_p and V_q).

7.2.2 Splitting and feature selection

Another related problem consists in splitting the feature set in several views at the same time dropping the features that bring little or negative contribution to the predicting power of the views.

It is possible to extend the problem with an extra view used to *exclude* some elements. In this case $\mathcal{V}(N)$ will contain $v+1$ views, if $p = v$, $p \oplus 1$ will be 1, not $v+1$, and the constraints saying that an element can be present only in a view must consider $v+1$ views and not v.

7.2.3 Feature Partitioning as an integer programming problem

The problem can be defined as a *quadratic programming problem* in nv binary variables. It can be written as

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & \sum_{p=1}^{v} \left(\sum_{i \in N} \phi_\xi(i) x_{pi} + \sum_{ij \subseteq N} I_\xi(ij) x_{pi} x_{pj} \right) - \sum_{\substack{p=1 \\ q=p \oplus 1}}^{v} \sum_{ij \subseteq N} I_\xi(ij) x_{pi} x_{qj} \\
\text{s.t.} \quad & \sum_{p=1}^{v} x_{pi} = 1 \quad \forall i \in N \\
& x_{pi} \in \{0,1\} \quad \forall p \in [1,v], \forall i \in N \\
& \mathbf{x} \in \{0,1\}^{v \times n}
\end{aligned}
$$

The general structure of a *quadratic programming problem* is

$$
\begin{aligned}
\max_{\mathbf{x}} \quad & \frac{1}{2} \mathbf{x}^T \mathcal{Q} \mathbf{x} + \mathbf{c}^T \mathbf{x} \\
\text{s.t.} \quad & \mathcal{A} \mathbf{x} = \mathbf{a} \\
& \mathbf{x} \in \{0,1\}^{vn}
\end{aligned}
$$

When applied to our problem, $\mathbf{x}, \mathcal{Q}, \mathbf{c}, \mathcal{A}, \mathbf{a}$ became

$$\mathbf{x} = \langle x_{pi} : p \in [1,v], i \in N \rangle$$

$$\mathcal{Q} = \begin{bmatrix} \left[I_\xi(ij) \right]_{11} & \dots & \left[-I_\xi(ij) \right]_{1v} \\ \dots & \left[I_\xi(ij) \right]_{pp} & \dots \\ \left[-I_\xi(ij) \right]_{v1} & \dots & \left[I_\xi(ij) \right]_{vv} \end{bmatrix}$$

$$\mathbf{c} = \left[\langle \phi_\xi(i) : i \in N \rangle_p : p \in [1,v] \right]$$

$$\mathcal{A} = \left[I_n : p \in [1,v] \right]$$

$$\mathbf{a} = \mathbf{1} \in \mathbb{N}^n$$

where

- $\mathbf{x} \in \{0,1\}^{nv}$ is a block vector, a block for each view, of boolean variables x_{pi} that says if V_p contains the element i
- $\mathcal{Q}^T = \mathcal{Q} \in \mathbb{R}^{nv \times nv}$ is a squared symmetric block matrix, a block for each pair of views, where the block contains the matrix of interaction indices between the elements i in V_p and j in V_q. The main diagonal blocks contain the interaction indices of the elements in the same view
- $\mathbf{c} \in \mathbb{R}^{nv}$ is a block vector, a block for each view, where the block contains the power indices of the elements $i \in N$
- $\mathcal{A} \in \mathbb{N}^{n \times nv}$ is a rectangular block matrix, a block for each view, where the block is the *identity matrix* of n^{th} order. It is used to count how many views contain the element i
- $\mathbf{a} \in \mathbb{N}^n$ is a vector with the number of views that must contain the element i: exactly one view for each element

Sometimes can be useful to add the constraints

- $\mathbf{x} \geq \mathbf{0}$
- $\mathbf{x} \leq \mathbf{1}$

that can be modelled as

$$\mathcal{B} \leq \mathbf{b}$$

$$\mathcal{B} = \left[\frac{-I_n}{I_n}\right]$$

$$\mathbf{b} = \left[\frac{\mathbf{0}}{\mathbf{1}}\right]$$

where

- $\mathcal{B} \in \mathbb{N}^{2nv \times nv}$ is a rectangular block matrix composed by two stacked identity matrices, the first used to model the constraint $\mathbf{x} \geq \mathbf{0}$ $(-\mathbf{x} \leq \mathbf{0})$ and the second used to model the constraint $\mathbf{x} \leq \mathbf{1}$

- **b** is the vector used to model the previous constrains

This is a NP-hard problem, and can be resolved with one of the several algorithms available.

7.2.4 Partitioning as graph partitioning problem

7.2.4.1 Introduction

The partitioning of the features can be modelled also as a *graph partitioning* (or *graph clustering*) problem [179].

In this case, we can use an *graph* $G = (V, E, w)$ where

- the *vertices* V are the features
- the *edges* E are pair of features
- the *weight function* w assigns a weight to vertices and edges

The *graph partitioning* problem consists in dividing the graph G into k components such that the components have similar *weights*, and this weight must be maximized, and the sum of weights of the edges with endpoints in different parts is minimized. The relation with the *feature partitioning* problem is direct:

- the views must have similar prediction ability $\rightarrow$ the components must have similar *weights*
- features that interfere negatively (they have a lower or negative interaction index value) must be separated $\rightarrow$ the weight of the edges with endpoints in different parts must be minimized
- features that collaborate (they have a higher positive interaction index value) must be held together $\rightarrow$ the weight of edges in the same component must be maximized

The main difference with the *classic* graph partitioning problem is that, in this case, we have a weight assigned to the vertices/features that we must include in the solution.

7.2.4.2 Definitions

As specified previously, the graph $G = (V, E, w)$ is composed by:

- V (set of*vertices*) the set of features v_i
- E (set of *edges*) the set of all possible feature pairs $e_{ij} = \{v_i, v_j\}$
- w a *weight* function that assigns a weight to vertices and edges

The weight function assigns:

- the *power index value* to the vertices (the features)
- the *interaction index value* to the edge (pair of features)

Because we have an interaction index value between each possible pair of features, the graph is *complete* and, because the interaction index is a symmetric function, the graph is *undirected*.

We extend the definition of the weight function to compute the weight for subsets of V and E:

$$w(U) = \sum_{v \in U} w(v) \quad \forall U \subseteq V$$
$$w(D) = \sum_{e \in D} w(e) \quad \forall D \subseteq E$$

There are other two useful measures used in the graph algorithms. The first is the *degree* of a vertex, in this case defined as sum of the weights of the edges incident to the vertex

$$\deg(v_i) = \sum_{v_j \in V} w(\{v_i, v_j\})$$

The second is the *volume* of a subgraph, defined as sum of edge's weights of all edges that have one or two vertices in $U \subseteq V$

$$\texttt{vol}(U) = \sum_{\substack{v_i \in U \\ v_j \in V}} w(\{v_i, v_j\}) = \sum_{v \in U} \deg(v)$$

A *partition* of the graph is a list of subgraphs such that $P = \langle V_1, \ldots, V_k \rangle$ is a vertices partition, and

$$\begin{aligned} G_i &= (V_i, E_i) \\ E_i &= \{\{v_i, v_j\} \in E : v_i, v_j \in V_i\} \end{aligned}$$

is the *subgraph induced by* V_i, that is, G_i is composed by V_i and the edges in E that have both vertices in V_i. The edges *cut*/removed, to create the partition, have a weight that can be computed as

$$\begin{aligned} \texttt{cut}(U, V) &= \sum_{\substack{v_i \in U \\ v_j \in V}} w(\{v_i, v_j\}) \\ \texttt{cut}(P) &= \frac{1}{2} \sum_{i=1}^{k} \texttt{cut}(V_i, V \setminus V_i) \end{aligned}$$

The requirement for the view partitioning can be converted in requirements for the graph partitioning:

- each view must be predict well: since the power index is the prediction ability of a feature, we must maximize $w(V_i)$
- each view must contain collaborating features: since the interaction index is a measure of collaboration/interference between two features, we must maximize $w(E_i)$
- features that does not collaborate must be separated, we must minimize $\texttt{cut}(P)$

Then, the problem to resolve is to find the *minimum cut* [180] that permits to obtain k components

$$\min_{P \in \mathcal{P}^k(V)} \texttt{cut}(P)$$

The main drawback of this approach is that it can generate components with very different weights.

The literature defines two cut variants: the *ratio cut* [185] and the *normalized cut* [181], defined as

$$\texttt{RatioCut}(P) = \frac{1}{2}\sum_{V_i \in P} \frac{\texttt{cut}(V_i, V \setminus V_i)}{w(V_i)}$$

$$\texttt{Ncut}(P) = \frac{1}{2}\sum_{V_i \in P} \frac{\texttt{cut}(V_i, V \setminus V_i)}{\texttt{vol}(V_i)}$$

The `RatioCut` tries to create subgraphs with similar vertices weights, where `Ncut` tries to create subgraphs with similar edges weights.

Our problem is to create subgraphs with similar *total* weights. We can define a *weighted cut* as

$$\texttt{Wcut}(P) = \frac{1}{2}\sum_{V_i \in P} \frac{\texttt{cut}(V_i, V \setminus V_i)}{w(V_i) + \texttt{vol}(V_i)}$$

and to resolve

$$\min_{P \in \mathcal{P}^k(V)} \texttt{Wcut}(P)$$

The main property of this approach is the the functions `RatioCut`, `Ncut` and `Wcut` have a minimum when all terms have similar values.

There is a little problem: not all algorithms are able to use both weights. Very often they are able to use only the edge weights. A simple strategy to include the vertex weight is to add, to each vertex, a *loop* (an edge with that starts and ends on the same vertex) with the weight equals to the vertex weight. Using this trick, and, when necessary, using some special rules for loops, our problem can be reconverted in a problem based on the *normalized cut*

$$\min_{P \in \mathcal{P}^k(V)} \texttt{Ncut}(P) \tag{7.1}$$

7.2.4.3 Spectral Clustering

The *Spectral Clustering* is an approach to graph partitioning based on the linear algebra an the properties of the *autovectors* [186, 187].

The method needs the *unnormalized Laplacian matrix* defined as

$$L = D - W$$

where D is the diagonal *degree matrix*, defined as

$$D_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

and W is the symmetric *adjacent matrix* (or *similarity matrix*), defined as

$$W_{ij} = w(\{v_i, v_j\})$$

The loops need an special treatment. The matrix D has the property

$$\sum_{v \in V} \deg(v) = 2w(E)$$

that is, the sum of the vertices' degree is equal to the sum of the edges' weight multiplied by 2. To ensure that the property is maintained, the weight of the loop must be inserted into the diagonal of the matrix multiplied by two.

In W the loop's weight can be inserted as is or multiplied by 2. We need to insert it multiplied by 2, to ensure that it disappear in $D - W$. It will reappear with the *normalization*.

The next step is to use a *normalized Laplacian matrix*. The literature defines two normalized variants

$$\begin{aligned} L_{sym} &:= D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \\ L_{rw} &:= D^{-1} L = I - D^{-1} W \end{aligned}$$

Both matrices are able to resolve the optimization problem 7.1. The solution is the k eigenvectors $u_1, \ldots, u_k$ with the smallest eigenvalues.

Let $U \in \mathbb{R}^{n \times k}$ the matrix where the columns are the selected eigenvectors $u_1, \ldots, u_k$. The rows can be considered as the coordinates in $\mathbb{R}^k$ of the vertices/features. To improve the partition process, the rows of U must be normalized to have *euclidean norm* 1.

The last step is to use the clustering algorithm as, for example, *K-means*, to aggregate the nodes in k clusters. The content of each cluster will be the features assigned to each view.

Function SpectralClustering(G, k)

---

input : $G = (V, E, w)$ the graph
input : k number of partitions
output: $P = \langle V_1, \ldots, V_k \rangle$ the vertices partition

$W \leftarrow$ the adjacent matrix
$D \leftarrow$ the degree matrix
$L \leftarrow D - W$;; *the Laplacian matrix*
$L_{sym} \leftarrow D^{-1/2} L D^{-1/2}$;; *the normalized Laplacian matrix*
$U \in \mathbb{R}^{n \times k} \leftarrow$;; *the smallest k autovectors of* L_{sym}
$T \leftarrow$ to normalize the rows of U to have euclidean norm 1
$P \leftarrow$ to use *K-means* to create k clusters

return P

Note: a current notebook (2019), with CPU at 2.5GHz, is able to evaluate the eigenvectors of a 1000×1000 symmetric matrix in $0.2s$, and the eigenvectors of a 1000×1000 generic matrix in $1.2s$.

7.2.4.4 Multilevel Graph Clustering

One of the most effective method for graph partitioning actually existent, is the *Multilevel Graph Partitioning* [182, 183]. The main idea of this class of algorithms is to simplify the graph collapsing vertices and edges, partition the reduced graph, then expand the vertices and use some local partition improvement, until a partition for the original graph is obtained.

One of the properties of this method is to assign to *generated* vertices a weight that depends on the collapsed vertices and edges. This means also

that it is able to handle graphs with weights assigned also to the original vertices.

The general structure of the method is based on three phases:

1. *coarsening phase*: the graph $G = G_0$ is converted in a sequence of smaller graphs $G_1, \ldots, G_m$ with $w(V_0) > w(V_1), \ldots, w(V_m)$
2. *partitioning phase*: a k-way partition P_m is computed on $G_m = (E_m, V_m)$
3. *uncoarsening phase*: the partition P_m is projected to $G_{m-1}, \ldots$ until G_0 is reached

In the *coarsening phase*, the graph is analyzed to identify small subgraphs, with fewer vertices, with *nice properties*. How to identify these subgraphs is problem dependent. In our case, for example, we can search not intersecting feature's pairs with the highest interaction index value. These subgraphs will be collapsed to form a single vertex in the next level. In the original algorithm, to the collapsed vertex is assigned a weights that depends on the collapsed edges. But this weight can contains also the weights of the collapsed vertices.

In the *partition phase*, it is computed a partition of G_m. Here, it is possible to use any good algorithms available, for example *spectral clustering*. Since the weights of the edges in G_m reflect the original weights, the graph contains enough information to force the partitioner to find a good partition also for the finer graphs. Also, since G_m has a small number of vertices, the partition is very efficient.

In the *uncoarsening phase*, the partition of G_{i+1} is projected into the graphs G_i in the previous level. Since G_i has more vertices, it is possible that the optimal partition of G_{i+1} is not optimal in G_i. Here, it is possible to use some local refinement heuristic to try to improve it. The general structure of these algorithms is: select some vertices in two blocks of the partition, swap them and check if the new partition has a smaller cut weight. This is tried a selected number of times. Then the process restart with the previous level until the original graph is reached.

However, because the *spectral clustering* is enough fast, this approach is useful only when the graph (and the dataset) has several thousand of features.

7.3 Feature partitioning for Co-training

7.3.1 View teaching

In the previous section, we have split the elements of N in v views in the best possible way. Now we want to use these views in the context of the *co-training*.

The *co-training* is a mechanism used in the *Semi-Supervised Learning* algorithms to transfer the best prediction's ability of a view, on a selected instance of the dataset to another view. The views must have the following properties:

1. each view must contain features with the best prediction ability
2. collaborating features must stay in the same view
3. interfering features must be in different views
4. each view must be able to *teach* something to other ones as well as possible

The *ability of teaching* introduces some observations

1. the function is not commutative: if V_p is able to teach something to V_q, nothing is said on the ability of V_q to teach something to V_p
2. each view can teach to each the other ones, but this introduces $v(v-1)$ relations (it is a *clique* in a *directed graph*)
3. we can decrease the prediction ability of a view if this increase its ability of teaching

The point 2. is the reason of the rule *a view is related only with the next one* (and the last view with the first one): in this way, we need to consider only v relations and not v^2.

7.3.2 How to evaluate the ability of teaching

We consider a classification problem. We need to evaluate the ability of a view to teach something to another one. The idea is to follow the same approach used to evaluate the *accuracy* of a classification algorithm: we *count* how many times V_p is able to teach something to V_q. The method is

1. we select the instances in V_p with the *best* prediction's *quality*
2. we evaluate the prediction's *quality* in V_q on the same instances selected in V_p (each view uses its features on the instance to be analyzed)
3. if the prediction's *quality* of V_p is greater than the prediction's quality of V_q, of a selected threshold τ, V_p is able to teach to V_q, and this counts 1

The quantity

$$\mathcal{T}_\tau(V_p, V_q) = \frac{\#\ V_p \text{ is able to teach to } V_q}{\#\ \text{instances analyzed}}$$

is what we need: *the ability of V_p to teach to V_q*.

7.3.3 Prediction quality for classification

We must evaluate the *quality* of a classification prediction. We remember that the result of a classification algorithm is not the predicted category, but the vector $\mathbf{p}$ with the *prediction probability* for each category, and the predicted category is the category with the higher probability.

If there are c categories (to generalize, $c > 2$), we know that the *best* prediction is when the probabilities are

$$\mathbf{p} = \langle \ldots, 1, \ldots \rangle$$

that is, the category c_i is predicted with probability 1 and the others with probability 0. The *worst* prediction is when the probabilities are

$$\mathbf{p} = \langle \ldots, \frac{1}{c}, \ldots \rangle$$

that is, the prediction's probability is the same for each category.

The prediction's *quality* must be a function such that:

$$\mathcal{Q}(\langle \ldots, 1, \ldots \rangle) = 1$$
$$\mathcal{Q}(\langle \ldots, \frac{1}{c}, \ldots \rangle) = 0$$

The first candidate is, obviously, the *entropy*

$$H(\mathbf{p}) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

given $H(\langle \ldots, 1, \ldots \rangle)$ is zero and

$$H(\langle \ldots, \frac{1}{c}, \ldots \rangle) = -\log_2(\frac{1}{c}) = \log_2(c)$$

has the maximum value. The prediction's *quality* $\mathcal{Q}(\cdot)$ can be defined as

$$\mathcal{Q}(\mathbf{p}) = 1 - \frac{H(\mathbf{p})}{\log_2(c)}$$

Another candidate is the *Euclidean norm*

$$\|\mathbf{p}\| = \sqrt{\sum_{i=1}^{c} p_i^2}$$

since $\|\langle \ldots, 1, \ldots \rangle\| = 1$ is the point with the maximum distance from the origin, and $\|\langle \ldots, \frac{1}{c}, \ldots \rangle\| = \frac{1}{c}\sqrt{c}$ is the minimum distance. The prediction's *quality* can be defined as

$$\mathcal{Q}(\mathbf{p}) = \frac{\|\mathbf{p}\| - \frac{1}{c}\sqrt{c}}{1 - \frac{1}{c}\sqrt{c}}$$

Obviously, it is possible to use other *distances* defined in $\mathbb{R}^n$.

7.3.4 Prediction quality for regression

In general, the prediction of a regression algorithm is only the prediction value y: it is not sufficient to evaluate the quality of the prediction. If we obtain, in some way, a *mean value* μ and its *standard error* σ, around the predicted value, we can use the Normal Distribution $\mathcal{N}(\mu, \sigma^2)$ to evaluate the *quality* of the prediction as

$$\mathcal{Q}(v) = 1 - \int_{\mu-\delta}^{\mu+\delta} \mathcal{N}(x|\mu, \sigma^2)dx \quad \text{with } \delta = |\mu - v|$$

If the value is near to μ, the integral is near to zero, and the quality of prediction is near to 1. On the other hand, if the value is very far from μ, the integral is near to 1, and the quality of prediction is near to zero.

A simple method to obtain multiple predictions is, for example, to use the cross validation.

7.3.5 Prediction quality and ability of teaching

The *quality of prediction* $\mathcal{Q}(\cdot)$ can be used to compare the prediction's ability of the two views. Let τ the threshold to use. On a selected instance, V_p is able to teach to V_p if

$$\mathcal{Q}(\mathbf{p}^{(p)}) \geq \mathcal{Q}(\mathbf{p}^{(q)}) + \tau$$

where $\mathbf{p}^{(p)}$ and $\mathbf{p}^{(q)}$ are the prediction probabilities obtained using the features in V_p and V_q.

7.3.6 To evaluate the view's ability of teaching

Let $\mathcal{D}$ a dataset used with some *Semi Supervised Learning* algorithm $\mathcal{A}_{\mathcal{L}}$, with n features, some labelled instances $\mathcal{L}$ and m unlabelled instances $\mathcal{U}$:

$$\begin{aligned}\mathcal{D} &= \mathcal{L} \cup \mathcal{U} = \langle \mathbf{d}_j : j = [1, \ldots] \rangle \\ \mathbf{d}_j &= \langle d_{ji} : i \in N \rangle\end{aligned}$$

and let

$$\begin{aligned}\mathbf{d}_j^p &= \langle d_{ji} : i \in V_p \rangle \\ \mathcal{D}^p &= \mathcal{L}^p \cup \mathcal{U}^p = \langle \mathbf{d}_j^p : j = [1, \ldots] \rangle\end{aligned}$$

the instances and the dataset with the features in V_p (and in V_q).

Let $\mathcal{A}_{\mathcal{L}}^p(\cdot)$ and $\mathcal{A}_{\mathcal{L}}^q(\cdot)$ the algorithms trained with $\mathcal{L}^p$ and $\mathcal{L}^q$ and

$$\begin{aligned}\mathbf{p}_i^{(p)} &= \mathcal{A}_{\mathcal{L}}^p(\mathbf{d}_i^p) \\ \mathbf{p}_i^{(q)} &= \mathcal{A}_{\mathcal{L}}^q(\mathbf{d}_i^q)\end{aligned}$$

the predictions obtained from the algorithms on the unlabelled instance $\mathbf{d}_i \in \mathcal{U}$.

We can defined the function V_p *can teach to* V_q as

$$\mathcal{T}_\tau(V_p, V_q, i) = \begin{cases} 1, \text{ if } \mathcal{Q}(\mathbf{p}_i^{(p)}) \geq \mathcal{Q}(\mathbf{p}_i^{(q)}) + \tau \\ 0, \text{ otherwise} \end{cases}$$

The *the ability of* V_p *to teach to* V_q is

$$\mathcal{T}_\tau(V_p, V_q) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{T}_\tau(V_p, V_q, i)$$

and the *ability of teaching of the partition* $\mathcal{V}(N)$

$$\mathcal{T}_\tau(\mathcal{V}(N)) = \frac{1}{v}\sum_{p=1}^{v}\mathcal{T}_\tau(V_p, V_{p\oplus 1})$$

If we consider all possible pairs of views, this definition is

$$\mathcal{T}_\tau(\mathcal{V}(N)) = \frac{1}{v(v-1)}\sum_{\substack{p,q=1\\ q\neq p}}^{v}\mathcal{T}_\tau(V_p, V_q)$$

The *teaching function* $\mathcal{T}_\tau(\cdot)$ is a *partition function*

$$\mathcal{T}_\tau : \mathcal{P}^v(N) \to \mathbb{R}$$

given it is evaluated on $\mathcal{V}(N)$, a partitions of N in v blocks (the *views*).

7.3.7 Partition with the best ability of teaching

To find the partition with the best teaching ability we can check all possible partitions in v views

$$\max_{\mathcal{V}(N)} \mathcal{T}_\tau(\mathcal{V}(N))$$

The number of partitions of n elements in v blocks is computed by *Stirling numbers of the second kind* $\left\{ {n \atop v} \right\}$ [173]. This number grows with order $\mathcal{O}(n^n)$, more quickly than 2^n. This means that, as for the set functions, it is impossible to analyze all possible partitions. The solution is to approximate the function with a more simple one.

7.3.8 Approximation of the ability of teaching

The *ability of teaching* function $\mathcal{T}_\tau$ evaluates the ability of the view V_p to teach something to the view V_q:

$$\mathcal{T}_\tau(V_p, V_q)$$

the two views can be two arbitrary not-intersecting subsets of N.

To approximate the function, the most simple solution is to find a *first-order approximation* in 2 parameters, such that

$$\mathcal{T}_\tau(S,T) \approx \sum_{i \in T} \sum_{j \in S} \mathcal{T}_\tau(i,j)$$

where $\mathcal{T}_\tau(i,j)$ is the *ability* of i to teach something to j.

We can define the *Teaching Index* $\mathcal{T}_\tau(i,j)$ as

$$\mathcal{T}_\tau(i,j) = \sum_{T \subseteq N \setminus ij} \sum_{S \subseteq N \setminus ij \setminus T} \frac{1}{\left\{ {n-2 \atop 3} \right\}} \Delta_{ij} \mathcal{T}_\tau(T,S)$$

where

$$\begin{aligned} \Delta_{ij} \mathcal{T}_\tau(T,S) = & \mathcal{T}_\tau(T \cup i, S \cup j) \\ & - \mathcal{T}_\tau(T \cup i, S) \\ & - \mathcal{T}_\tau(T, S \cup j) \\ & + \mathcal{T}_\tau(T,S) \end{aligned}$$

and $\left\{ {n-2 \atop 3} \right\}$ is the number of partitions, of a set with $n-2$ elements, in 3 blocks: the two views and the set of excluded elements.

The expression is similar to the interaction index of the second degree: it measures the net ability of i to teach to j, excluding the other elements in

the blocks.

Function TeachingIndex($\mathcal{T}_\tau, n, m, \mathcal{D}$)

input : $\mathcal{T}_\tau$ teaching function
input : n number of elements in the set
input : m number of samples to use
input : $\mathcal{D}$ parameters for the set generator
output: matrix of teaching indices

$v \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$;; *cumulative values*
$c \leftarrow \mathbf{0} \in \mathbb{Z}^{n \times n}$;; *pair counts*
for $t \in [1, m]$ **do**
  $T \leftarrow \text{RandomSubset}(n, \mathcal{D})$;; *teacher set*
  $S \leftarrow \text{RandomSubset}(n, \mathcal{D})$;; *student set*
  $S \leftarrow S \setminus T$;; *the sets must have empty intersection*
  for $i \in T, j \in S$ **do**
    $v_{ij} \leftarrow v_{ij} + \Delta_{ij}\mathcal{T}_\tau(T \setminus i, S \setminus j)$
    $c_{ij} \leftarrow c_{ij} + 1$
  end
end
$t \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$
for $ij \subseteq N$ **do**
  $t_{ij} \leftarrow v_{ij}/c_{ij}$
end
return t

7.3.9 The co-training optimization problem

Now we have all components to define the complete optimization problem for the co-training: we must find a partition of N in v views such that

1. each view is able to predict as well as possible
2. each view interferes with the others as less as possible
3. each view is able to teach something to the other views as well as possible

The problem can be rewritten as:

$$\max_{\mathcal{V}(N)} \quad \sum_{p=1}^{v} \left(\phi_\xi(V_p) + I_\xi^{(2)}(V_p) \right) - \sum_{\substack{p=1 \\ q=p\oplus 1}}^{v} \sum_{\substack{i\in V_p \\ j\in V_q}} \left(I_\xi(ij) - \mathcal{T}_\tau(i,j) \right)$$

7.3.10 Co-training as quadratic programming problem

The problem can be rewritten as an *integer programming problem*

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{p=1}^{v} \left(\sum_{i\in N} \phi_\xi(i) x_{pi} + \sum_{ij\subseteq N} I_\xi(ij) x_{pi} x_{pj} \right) \\ & - \sum_{\substack{p=1 \\ q=p\oplus 1}}^{v} \sum_{ij\in N} \left(I_\xi(ij) - \mathcal{T}_\tau(i,j) \right) x_{pi} x_{qj} \end{aligned}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{p=1}^{v} x_{pi} = 1 \quad \forall i \in N \\ & x_{pi} \in \{0,1\} \quad \forall p \in [1,v], \forall i \in N \end{aligned}$$

The structure of this problem is *similar* to the previous one. The difference is in the matrix $\mathcal{Q}$

$$\mathcal{Q} = \begin{bmatrix} \left[I_\xi(ij) \right]_{11} & \dots & \left[-I_\xi(ij) + \mathcal{T}_\tau(i,j) \right]_{1v} \\ \dots & \left[I_\xi(ij) \right]_{pp} & \dots \\ \left[-I_\xi(ij) + \mathcal{T}_\tau(i,j) \right]_{v1} & \dots & \left[I_\xi(ij) \right]_{vv} \end{bmatrix}$$

this matrix is not symmetric because the *Teaching Index* is not symmetric

$$\mathcal{T}_\tau(i,j) \neq \mathcal{T}_\tau(j,i)$$

However, there are standard methods to convert this problem in an equivalent *quadratic programming* optimization problem [67].

7.3.11 Co-training as graph partitioning problem

To model the co-training problem as a graph partitioning problem is an open problem. To simplify the description, we can suppose to extend the graph with *teaching edges*, a new type of *direct edges* used to model the partition's ability of teaching. These edges introduce some complications

1. the *teaching edges* have a direction
2. the *teaching edges* are useful *only* in parallel with the cut-edges and only between V_p and $V_{p\oplus 1}$
3. the partitioning must minimize the cut-edges and maximize the teaching edges

Because the graph contains directed edges, it must considered as a *direct graph*. In this case, it is not possible to use the algorithms used for the undirected graphs, but it is necessary to consider the algorithms specific for direct ones [184].

For example, because the adjacent matrix is not symmetric, also the Laplacian matrix will be not symmetric, and eigenvectors and eigenvalues will be complex.

But this is not enough: since the teaching edges must be considered only with cut-edges, this breaks the conditions on which the algorithms were designed.

A possible solution is to use the *Multilevel graph partitioning* algorithm, because it is not difficult to consider the *teaching edges* during the *uncoarsening phase*.

7.4 Feature partitioning with different algorithms

7.4.1 The 2^{nd} degree mixed interaction index

In Section 7.2 we have used the same set function to compute the interaction indices for the elements in the same view and in different views. To use the

same set function is valid *only* if we are using the *same* Machine Learning algorithm, with the *same* configuration parameters, for all views.

If we use the same algorithm with different parameters or different algorithms, we must associate each view with a different set functions. Now, we need a function to evaluate the quality of prediction for a selected *partition* (list of v views). We can define the *partition function induced by the set functions* (set functions assigned to each view) as

$$\Xi^{\mathcal{V}(N)}(S) = \sum_{p=1}^{v} \xi_p(S \cap V_p)$$

as specified in section 2.4.

Now, we can define the 2^{nd} *degree mixed interaction index* between the views V_p and V_q, in the similar way as the probabilistic indices, and the *teaching index*

$$I_{\Xi}^{pq}(ij) = \sum_{T \subseteq N \setminus ij} p_{ij}^{pq}(T) \Delta_{ij}^{pq} \Xi^{\mathcal{V}(N)}(T) \tag{7.2}$$

with $\langle p_{ij}^{pq}(T) : T \subseteq N \setminus ij \rangle$ a probability distribution. Selecting the weights for the partitions and for the sets, we can define one of the several power indices based on partitions, as the Owen-Shapley [92], Owen-Banzhaf [93], Banzhaf-Banzhaf [94], or we can use the new classes of power indices.

7.4.2 Partitioning as quadratic programming problem

Using the mixed interaction indices, the problem can be rewritten as

$$\max_{\mathcal{V}(N)} \quad \sum_{p=1}^{v} \left(\phi_{\xi}(V_p) + I_{\xi}^{(2)}(V_p) \right) - \sum_{\substack{p=1 \\ q=p \oplus 1}}^{v} \sum_{\substack{i \in V_p \\ j \in V_q}} \left(I_{\Xi}^{pq}(ij) - \mathcal{T}_{\tau}(i,j) \right)$$

Using the same methods used in Section 7.3.9, it can be converted in a *Quadratic programming* optimization problem, and resolved using standard methods.

7.5 Conclusions

In this chapter we have extended the usage of the Game Theory in the context of the *feature partitioning*. We have sees that the concepts of power index and interaction index can be used to describe the importance of features in the predictions' quality of the Machine Learning algorithms, and can be used to split the features in two or more views in the best possible way. We have also used the Game Theory applied to partition functions to model the concepts of *ability of teaching* of a view toward another one, and to model the problem of view partitioning when they are used with different Machine Learning algorithms.

Chapter 8

Effectiveness of Power Index based methods

8.1 Introduction

To check the validity of our approach, we have applied the methods described in the previous chapters on a selected list of datasets downloaded from **UCI Machine Learning Repository** [176].

Because we are interested to compare the methods with the *true* results, we have selected datasets that can be used for classification with a maximum of 20 features. Then, we have tabulate the *accuracy* of *all* possible subsets (2^n) using a simple *Decision Tree*. We have selected the DT because it is the fastest algorithm available and has no limitations on the value types.

To implement the algorithms and to plot he results, we have used

- `Python 3` [177]
- `scikit-learn`, a popular Machine Learning library [189]
- `matplotlib`, a popular plotting library [190]
- several our libraries

We have used the classification algorithms `tree.DecisionTreeClassifier`

with the default configurations. The general structure of the results changes very little using other algorithms.

It is possible to improve the accuracy tuning the hyper-parameters, but this is not the goal of this analysis.

8.2 Accuracy behaviour

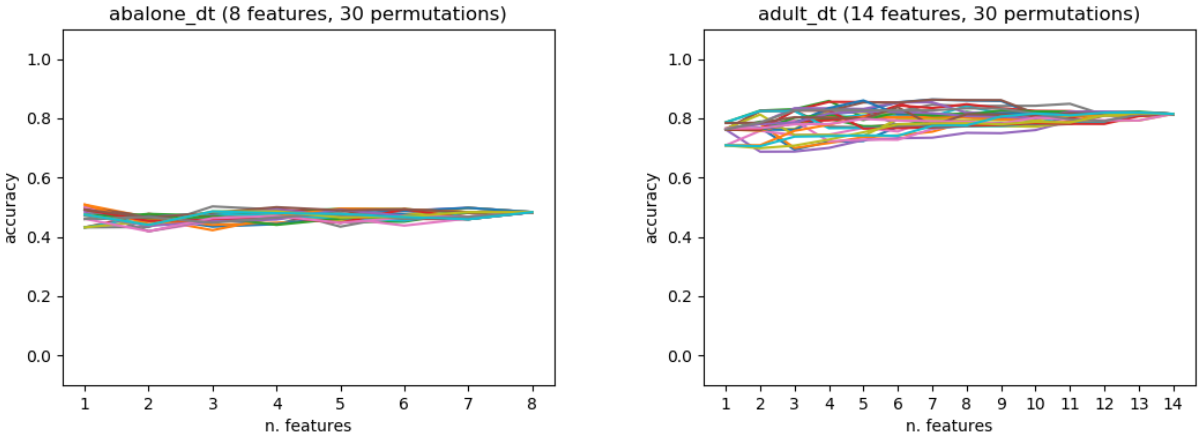
The first questions are

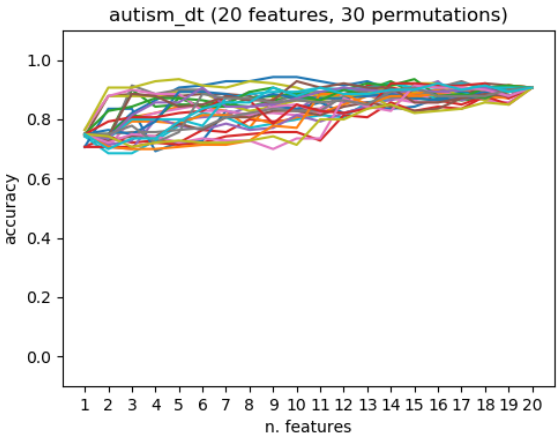
1. which is the *accuracy* of the algorithm, selecting 1, 2 or more features?
2. how does the accuracy change selecting the features in different order?

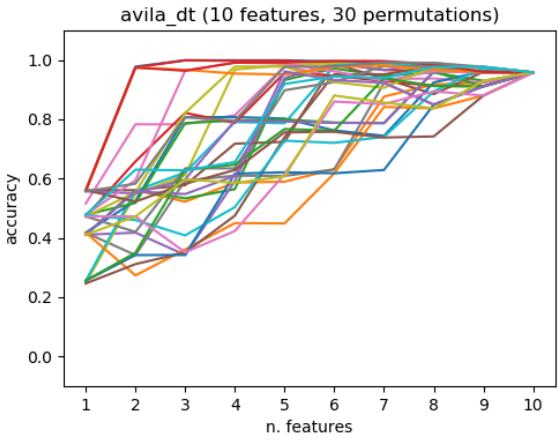
To answer to these questions we have generated the following plots: each line is generated selecting a random permutation of the features, selecting the first $1, 2, \ldots$ features and computing the accuracy on the obtained set.

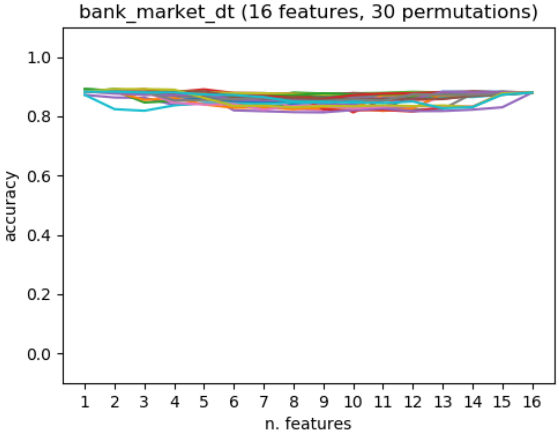
The interesting aspects of the diagrams are

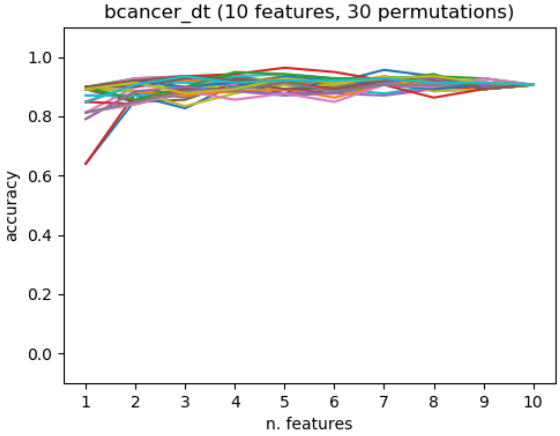
1. the *envelope* of the diagrams, that is, the minimum and maximum accuracy for each sets' cardinality selected
2. the *global* behaviour of the accuracy: flat, increasing or decreasing

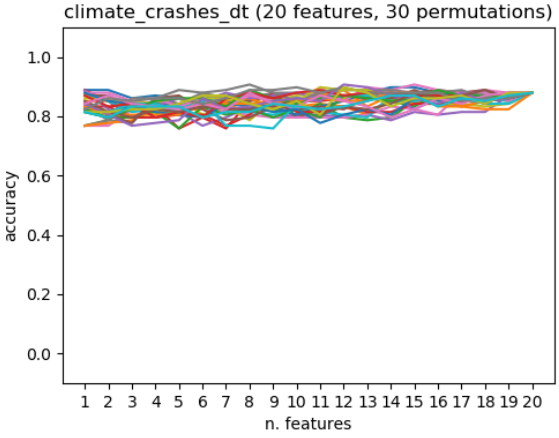


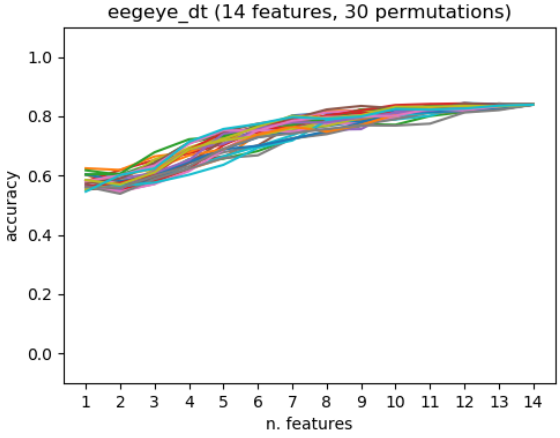

autism_dt (20 features, 30 permutations)
accuracy
n. features

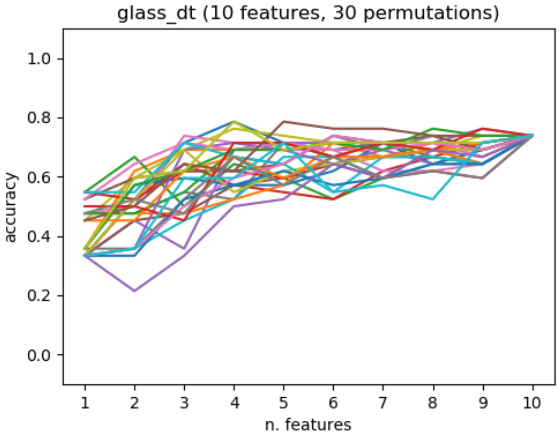

avila_dt (10 features, 30 permutations)
accuracy
n. features

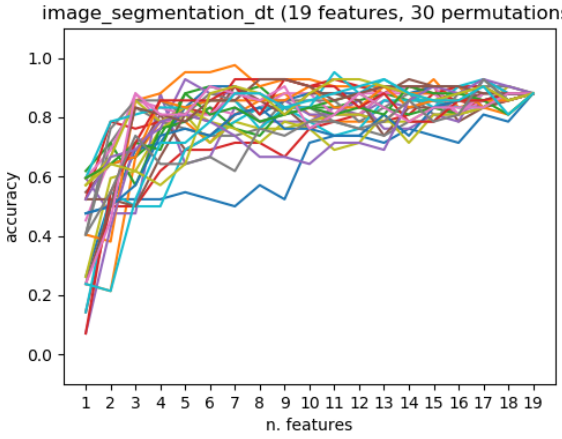

bank_market_dt (16 features, 30 permutations)
accuracy
n. features

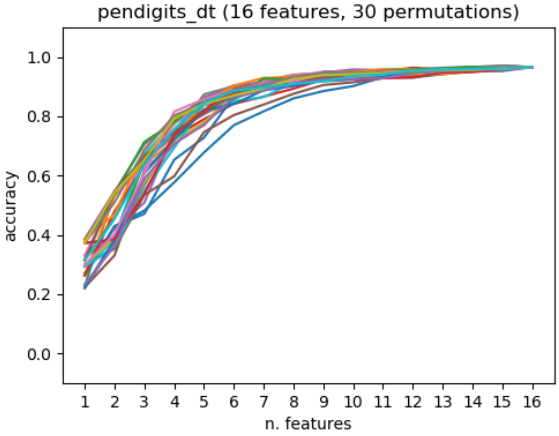

bcancer_dt (10 features, 30 permutations)
accuracy
n. features

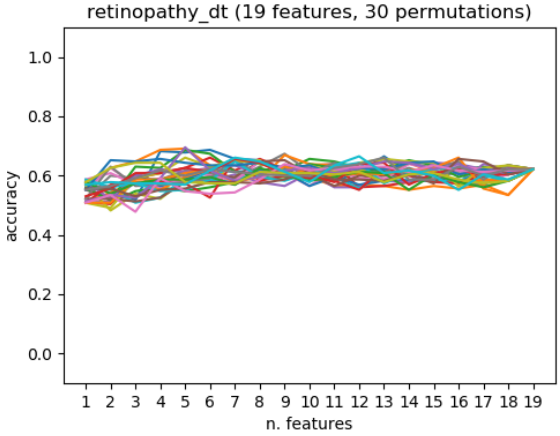

climate_crashes_dt (20 features, 30 permutations)
accuracy
n. features

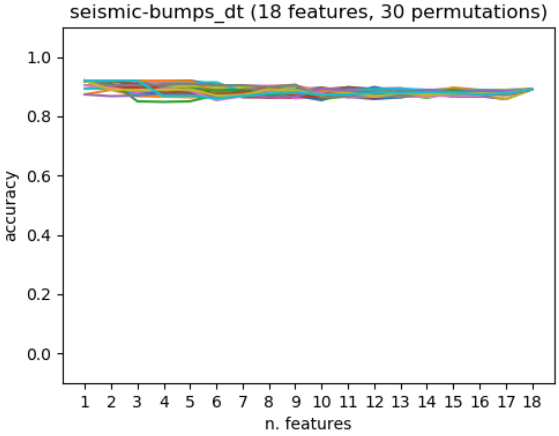

eegeye_dt (14 features, 30 permutations)
accuracy
n. features

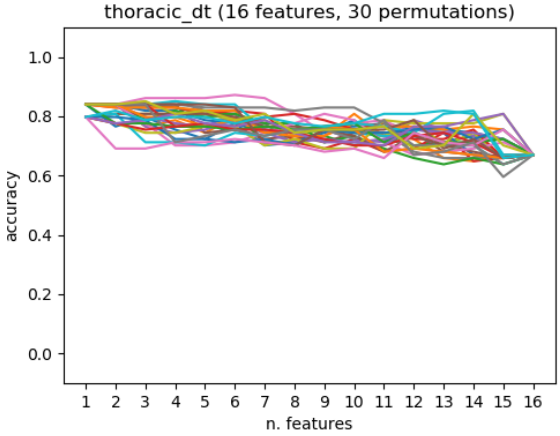

glass_dt (10 features, 30 permutations)
accuracy
n. features

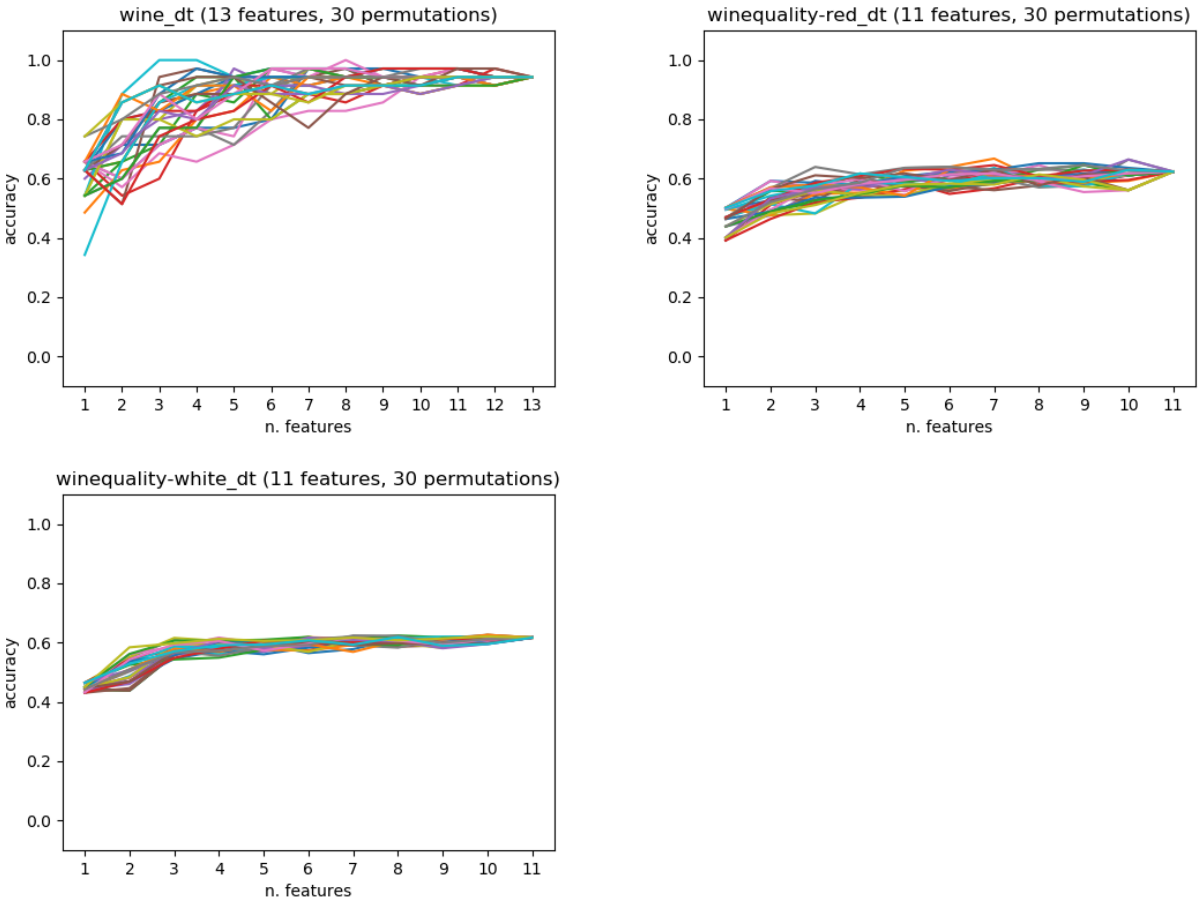

image_segmentation_dt (19 features, 30 permutations)
accuracy
n. features


pendigits_dt (16 features, 30 permutations)
accuracy
n. features


retinopathy_dt (19 features, 30 permutations)
accuracy
n. features


seismic-bumps_dt (18 features, 30 permutations)
accuracy
n. features


thoracic_dt (16 features, 30 permutations)
accuracy
n. features



8.3 Accuracy properties

For each dataset, we have tabulated the accuracy. Since the accuracy is a set function, it can be:

- monotone (increasing)
- sub/super/- additive
- sub/super/- modular

To evaluate the correspondence of the functions with the previous definitions, we have *count* how many times the functions satisfies the definitions, generating all possible subset pairs.

The following table shows that

- they are *sub-additive*
- they are *sub-modular* (more or less) at $60 - 80\%$
- they are *super-modular* (more or less) at $3 - 6\%$

name	nf	mon	add	suba	supa	mod	subm	supm
abalone	8	0.628	0.006	0.942	0.064	0.206	0.603	0.603
adult	14	0.665	0.0	0.989	0.011	0.038	0.625	0.413
autism	20	0.769	0.0	0.998	0.003	0.072	0.647	0.424
avila	10	0.849	0.001	0.895	0.106	0.114	0.803	0.311
bank-market	16	0.28	0.0	0.995	0.005	0.022	0.203	0.82
bcancer	10	0.747	0.005	0.968	0.037	0.172	0.757	0.415
climate-crashes	20	0.659	0.0	0.998	0.002	0.097	0.562	0.536
eegeye	14	0.994	0.0	0.982	0.018	0.037	0.949	0.088
glass	10	0.866	0.007	0.953	0.055	0.191	0.802	0.388
image-seg	19	0.792	0.002	0.992	0.01	0.105	0.715	0.391
pendigits	16	0.999	0.0	0.984	0.016	0.022	0.992	0.029
retinopathy	19	0.672	0.0	0.997	0.003	0.04	0.621	0.419
seismic-bumps	18	0.309	0.0	0.998	0.002	0.05	0.359	0.692
thoracic	16	0.253	0.001	0.998	0.002	0.1	0.487	0.613
winequality-red	11	0.864	0.001	0.966	0.035	0.104	0.765	0.339
winequality-white	11	0.89	0.001	0.965	0.036	0.093	0.807	0.286
wine	13	0.883	0.004	0.984	0.02	0.166	0.837	0.33

Table 8.1: Accuracy properties

with:

- `nf`: number of features
- `mon`: non-decreasing monotonicity

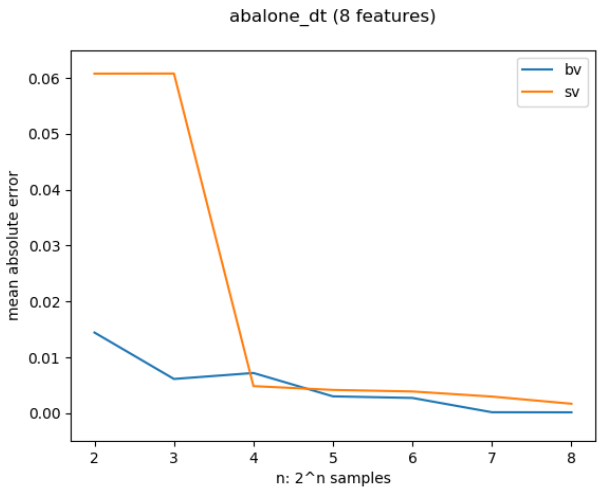
- `add`: additivity
- `suba`: sub additivity
- `supa`: super additivity
- `mod`: modularity
- `subm`: sub modularity
- `supm`: super modularity

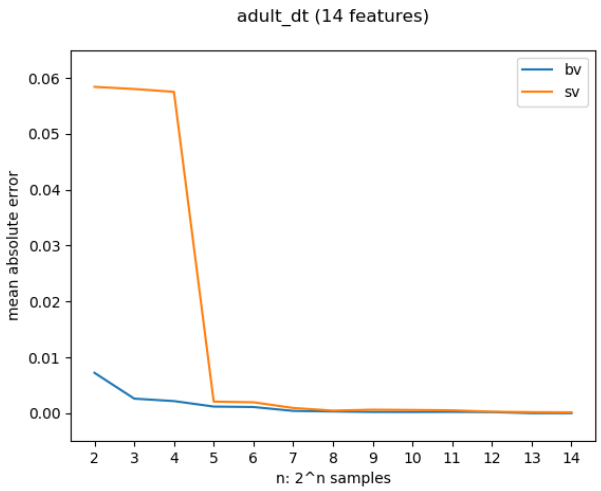
8.4 Approximation of Power Indices

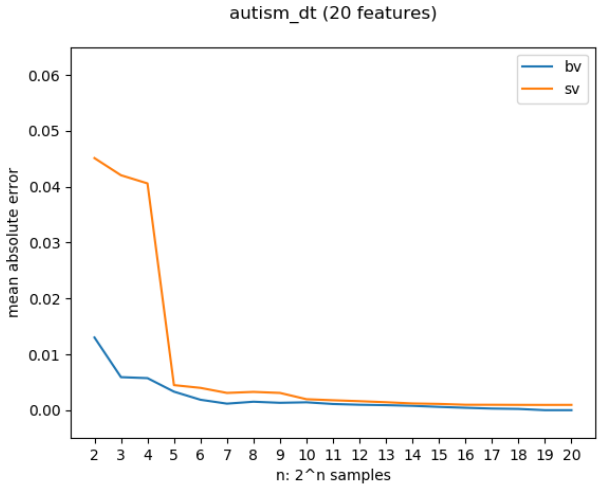
The following plots show how the *mean absolute error* between the *true* Power Index value ($\phi_\xi(i)$) and the *approximated* one ($\widetilde{\phi}_\xi(i)$), decreases with the number of samples. The approximation of the Shapley Value is computed using the definition based on sets.

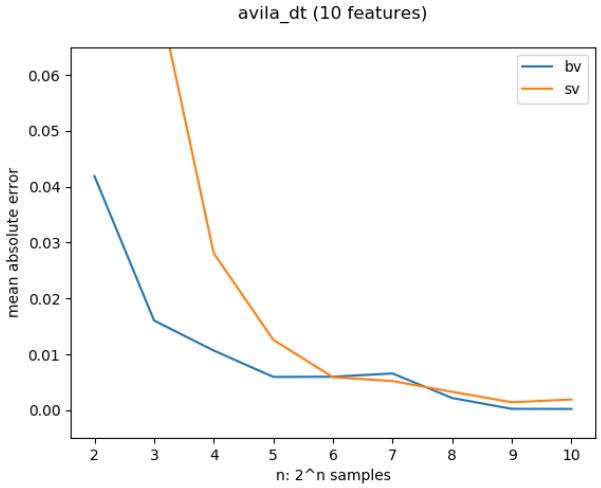
The *mean absolute error*, for a selected approximation, is computed as:

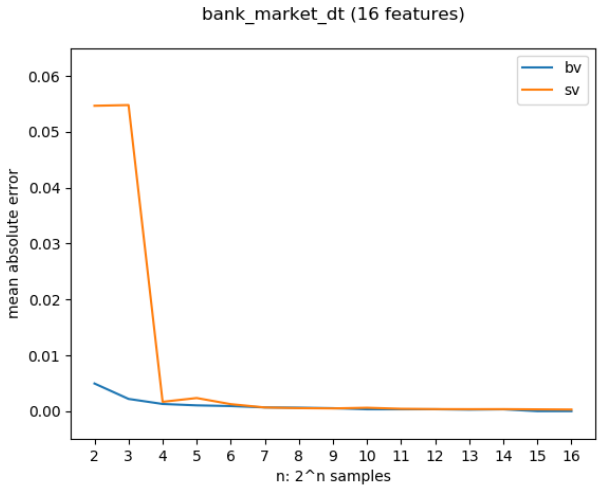
$$\texttt{mae} = \frac{1}{n}\sum_{i \in N} |\phi_\xi(i) - \widetilde{\phi}_\xi(i)|$$

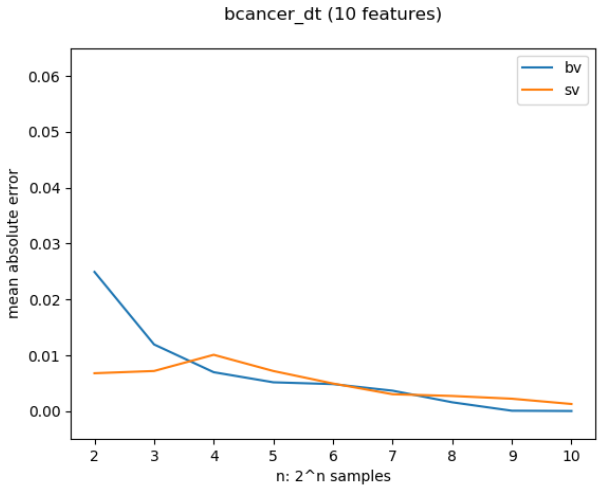


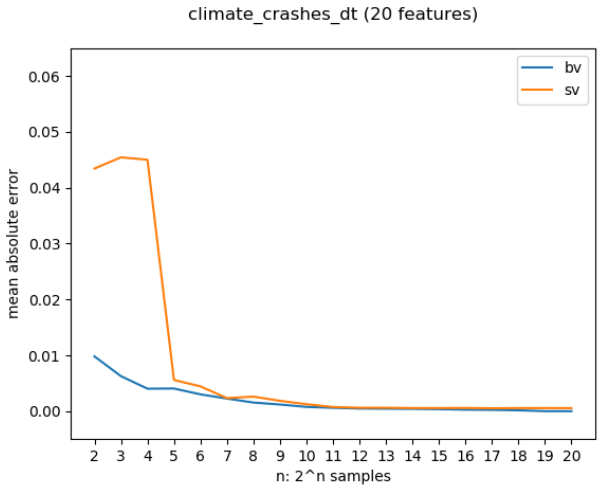


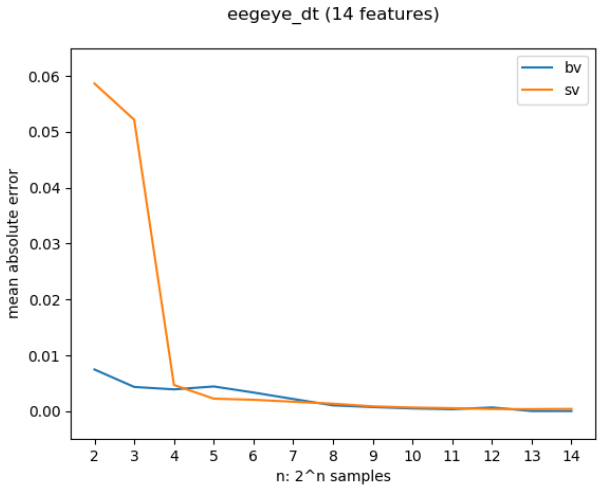

autism_dt (20 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
n: 2^n samples

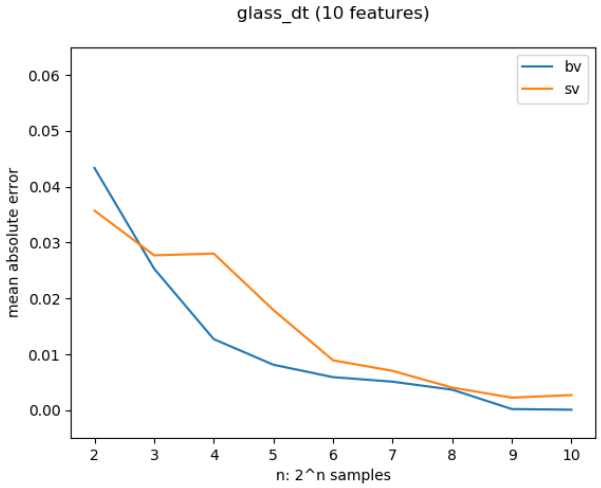

avila_dt (10 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10
n: 2^n samples

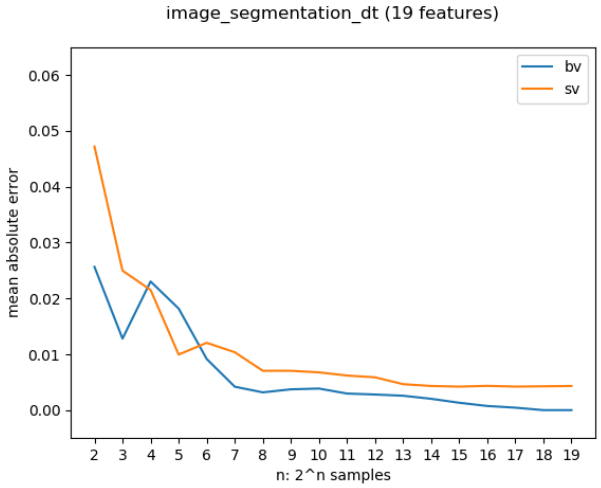

bank_market_dt (16 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
n: 2^n samples

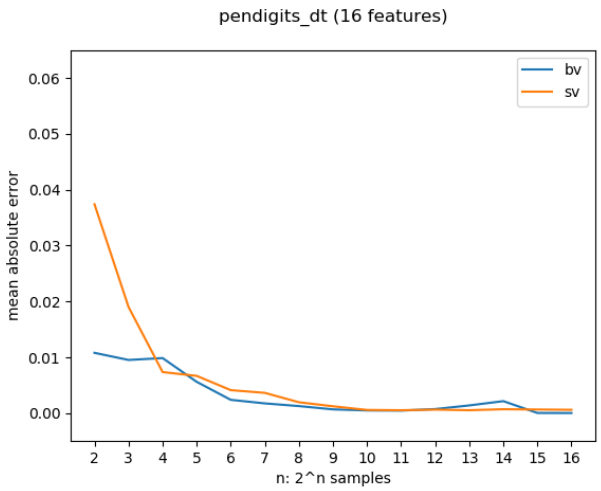

bcancer_dt (10 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10
n: 2^n samples

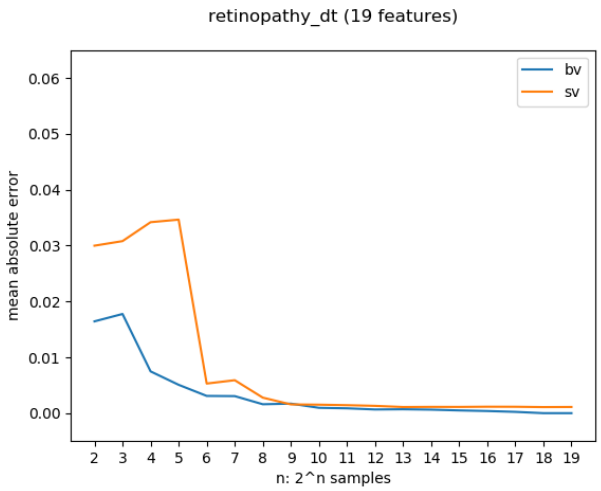

climate_crashes_dt (20 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
n: 2^n samples

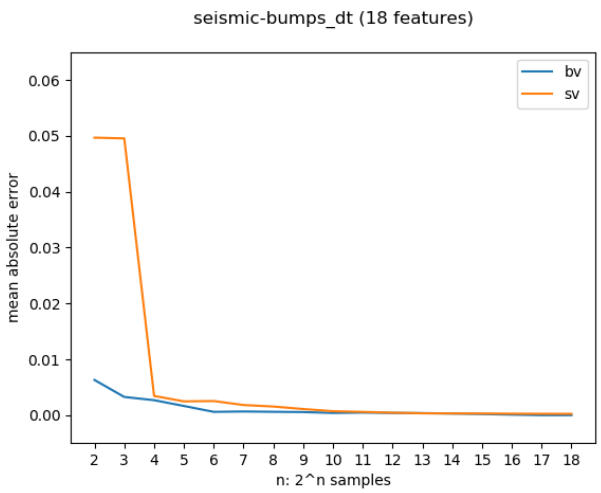

eegeye_dt (14 features)
bv
sv
mean absolute error
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2 3 4 5 6 7 8 9 10 11 12 13 14
n: 2^n samples

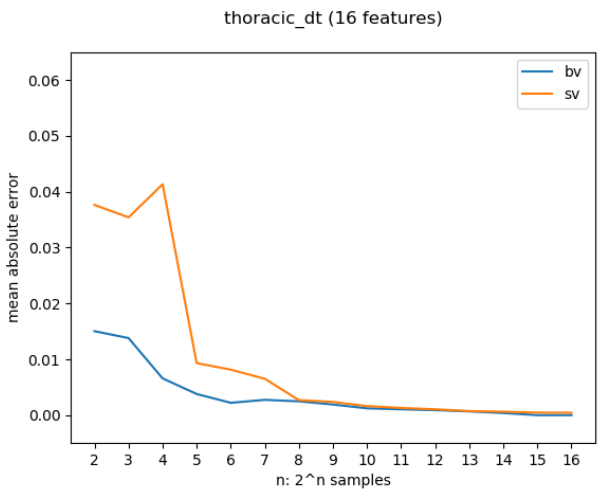

glass_dt (10 features)
bv
sv
mean absolute error
n: 2^n samples

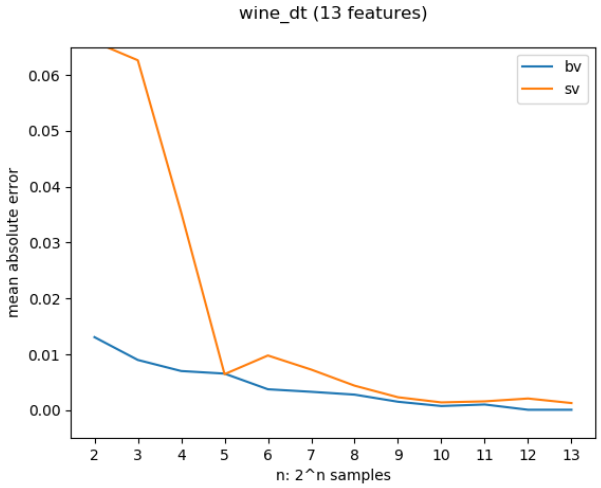

image_segmentation_dt (19 features)
bv
sv
mean absolute error
n: 2^n samples

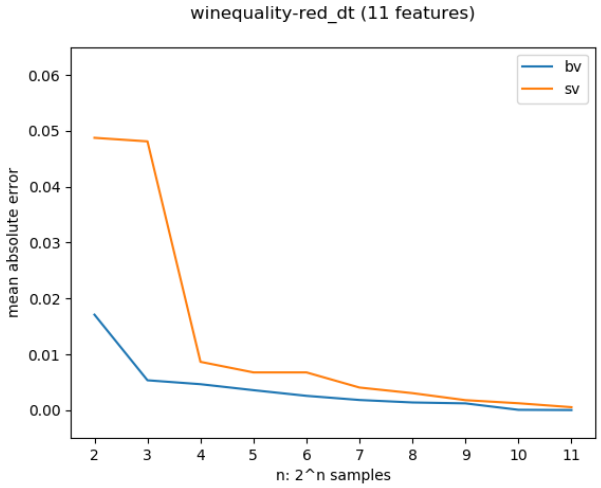

pendigits_dt (16 features)
bv
sv
mean absolute error
n: 2^n samples

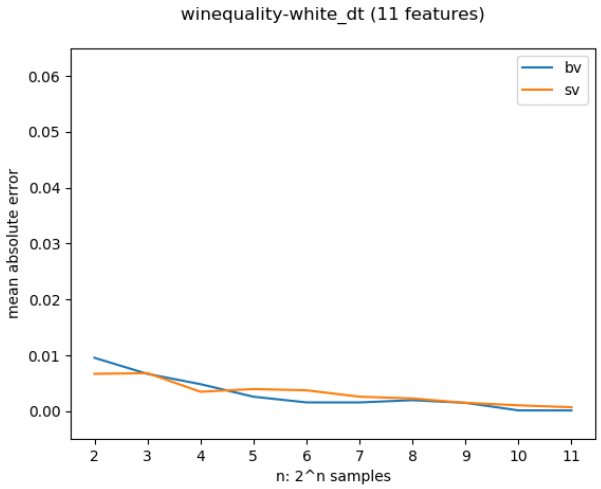

retinopathy_dt (19 features)
bv
sv
mean absolute error
n: 2^n samples


seismic-bumps_dt (18 features)
bv
sv
mean absolute error
n: 2^n samples


thoracic_dt (16 features)
bv
sv
mean absolute error
n: 2^n samples







Because the error decrease very quickly, then it remains approximately stable, we can reach a *stable* order very quickly. Indeed, in the following table, we can observe that a stable order is reached using a very limited number of samples (this is partially true for sets with a small number of feature, as for `abalone`).

The order of k elements is considered *stable* if it remain the same for a selected number of *epochs* (in this case 4).

Table 8.2: Stopping criteria

name	n	nsets	type	np	m	nsamples	stable
abalone	8	256	bv	4	10	240	8
abalone	8	256	sv	4	10	210	8
adult	14	16384	bv	4	100	800	14
adult	14	16384	sv	4	100	1000	14
autism	20	1048576	bv	4	100	3300	20
autism	20	1048576	sv	4	100	5400	20
avila	10	1024	bv	4	10	230	10
avila	10	1024	sv	4	10	100	10
bank-market	16	65536	bv	4	100	700	16
bank-market	16	65536	sv	4	100	1200	16
bcancer	10	1024	bv	4	10	300	10
bcancer	10	1024	sv	4	10	190	10
climate-crashes	20	1048576	bv	4	100	3700	20
climate-crashes	20	1048576	sv	4	100	3600	10
eegeye	14	16384	bv	4	100	1300	14
eegeye	14	16384	sv	4	100	900	14
glass	10	1024	bv	4	10	90	10
glass	10	1024	sv	4	10	170	10
image-segmentation	19	524288	bv	4	100	2400	19
image-segmentation	19	524288	sv	4	100	1800	19
pendigits	16	65536	bv	4	100	1900	16
pendigits	16	65536	sv	4	100	1700	16
retinopathy	19	524288	bv	4	100	4700	19
retinopathy	19	524288	sv	4	100	2700	19
seismic-bumps	18	262144	bv	4	100	4100	18
seismic-bumps	18	262144	sv	4	100	8400	18
thoracic	16	65536	bv	4	100	3600	16
thoracic	16	65536	sv	4	100	2300	16
wine	13	8192	bv	4	81	1134	13

name	n	nsets	type	np	m	nsamples	stable
wine	13	8192	sv	4	81	1053	13
winequality-red	11	2048	bv	4	20	400	11
winequality-red	11	2048	sv	4	20	320	11
winequality-white	11	2048	bv	4	20	380	11
winequality-white	11	2048	sv	4	20	280	11

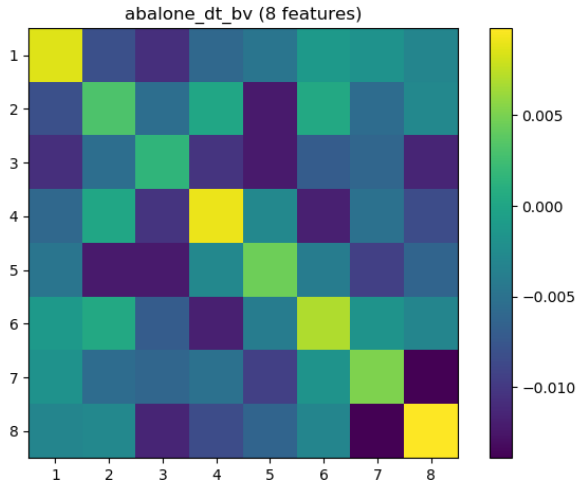
The columns are:

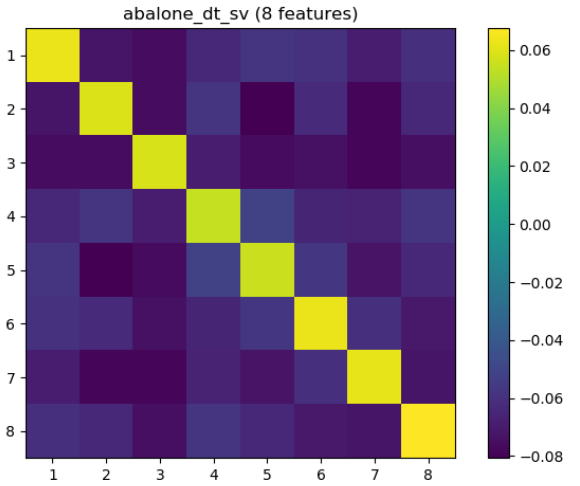
- `name`: name of the datase
- `n`: n. of features
- `nsets`: n. of different subsets (2^n)
- `type`: power index used to evaluate the order
- `np`: n. of times that the order must be *stable*
- `m`: n. of samples used in each *epoch*
- `nsamples`: n. of samples used to reach the stable order
- `stable`: n. of elements in the stable order

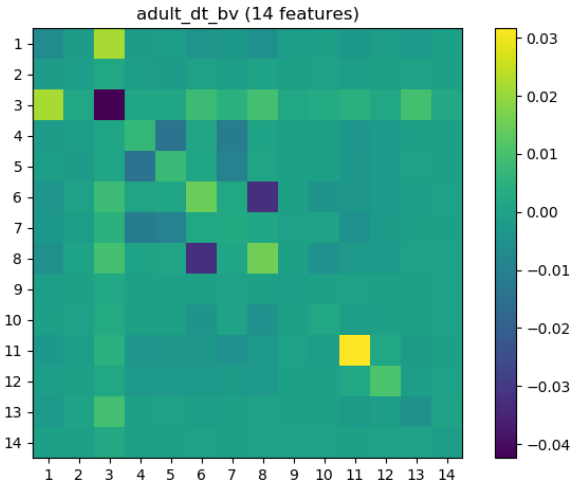
The last column (`stable`) contains the maximum number of stable elements: it is possible to observe that we have reached a stable order for *all* features.

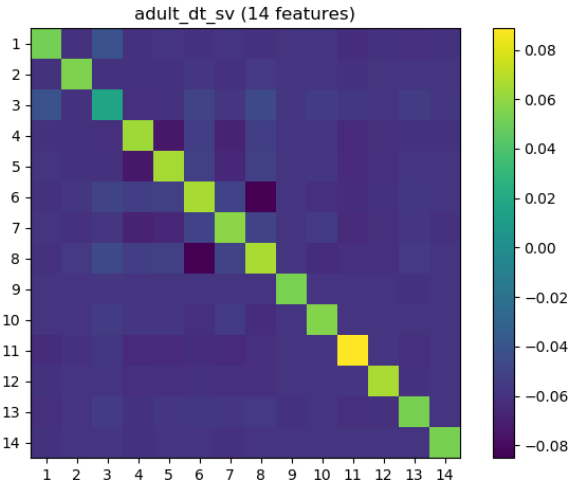
8.5 Power and Interaction Indices

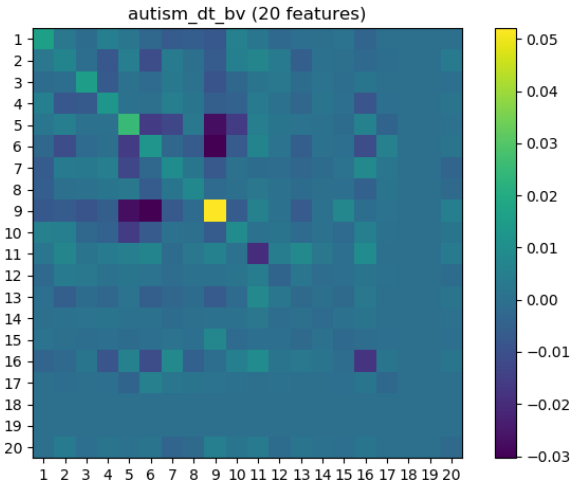
The following images show the distribution of the values between the power indices (the diagonal) and the interaction indices for Banzhaf (first column) and Shapley (second column).

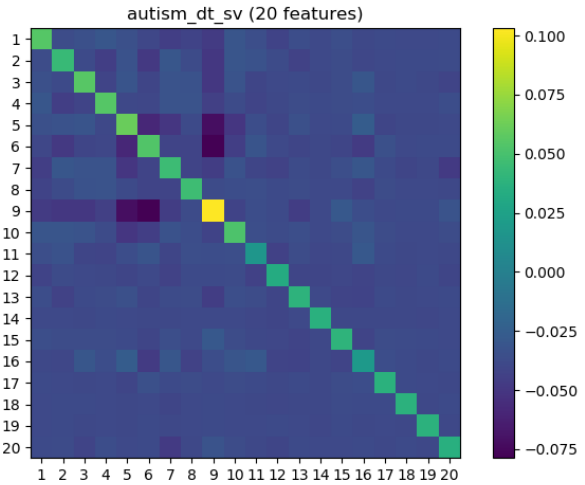

abalone_dt_bv (8 features)
1 2 3 4 5 6 7 8
0.005
0.000
−0.005
−0.010

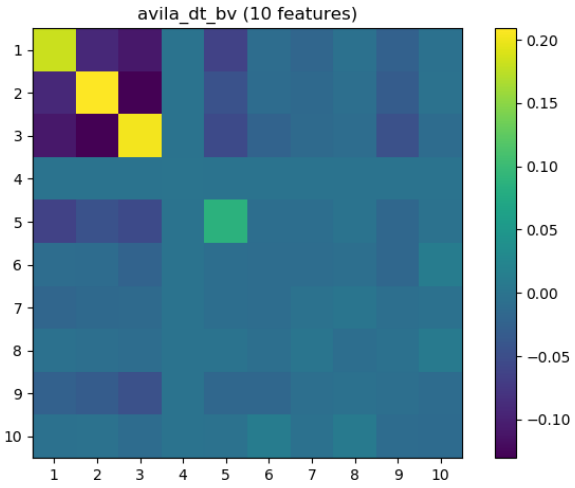

abalone_dt_sv (8 features)
1 2 3 4 5 6 7 8
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08

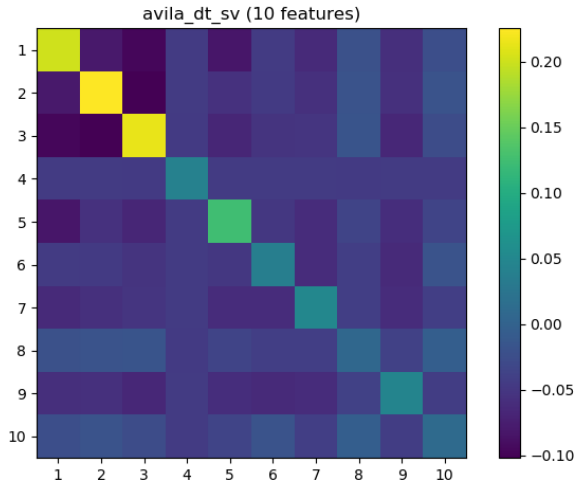

adult_dt_bv (14 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03
−0.04

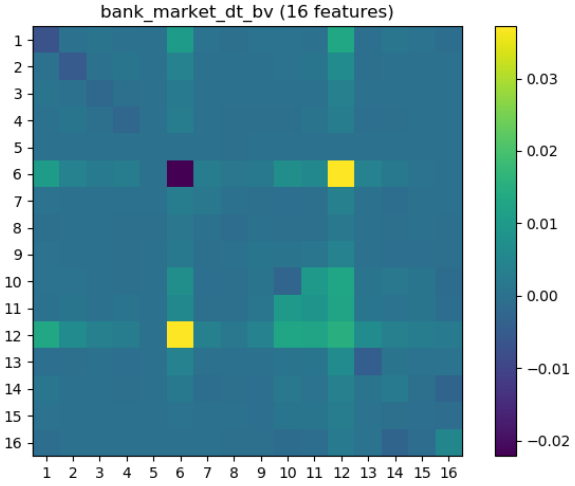

adult_dt_sv (14 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08

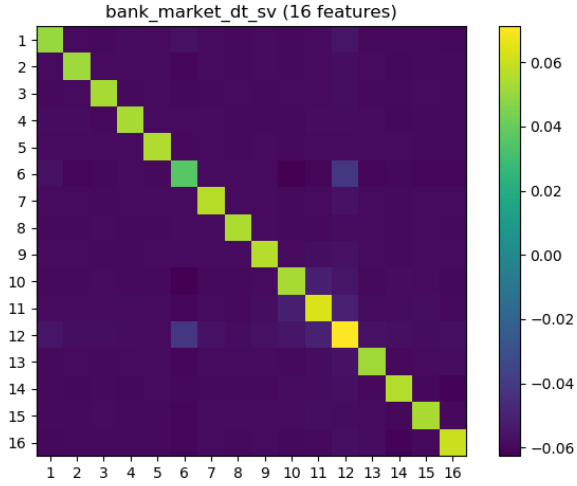

autism_dt_bv (20 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
0.05
0.04
0.03
0.02
0.01
0.00
−0.01
−0.02
−0.03

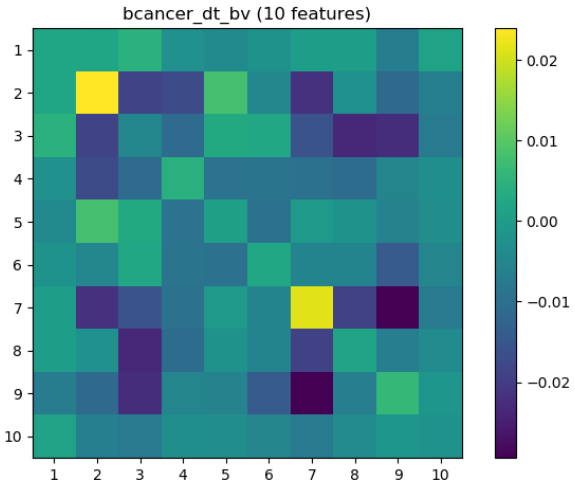

autism_dt_sv (20 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075

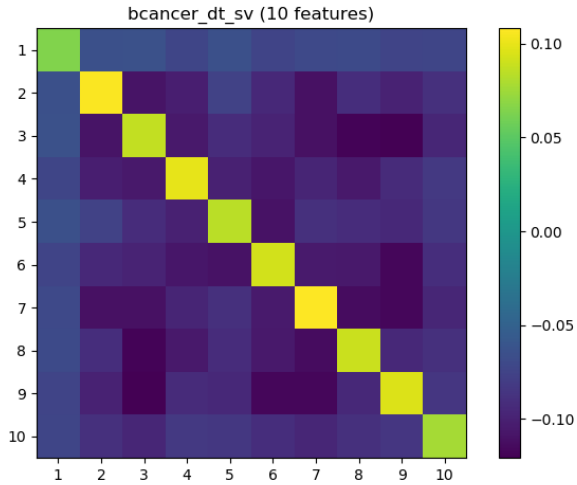

avila_dt_bv (10 features)
1 2 3 4 5 6 7 8 9 10
0.20 0.15 0.10 0.05 0.00 −0.05 −0.10

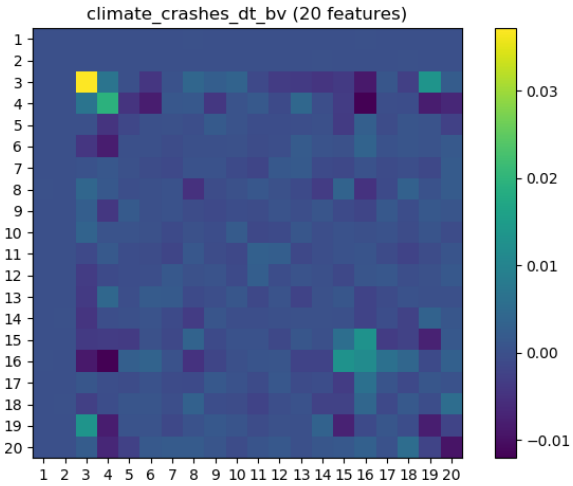

avila_dt_sv (10 features)
1 2 3 4 5 6 7 8 9 10
0.20 0.15 0.10 0.05 0.00 −0.05 −0.10

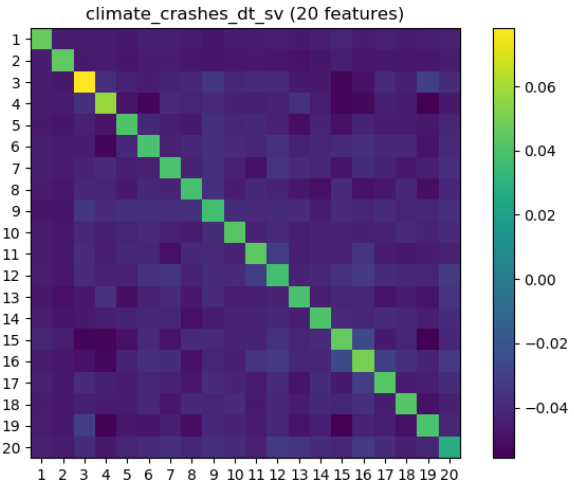

bank_market_dt_bv (16 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0.03 0.02 0.01 0.00 −0.01 −0.02

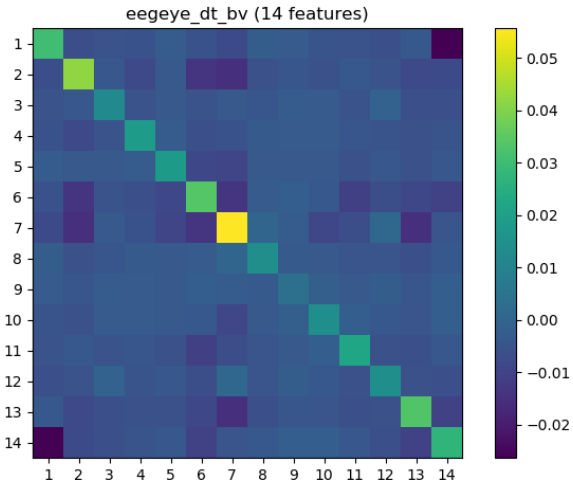

bank_market_dt_sv (16 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0.06 0.04 0.02 0.00 −0.02 −0.04 −0.06

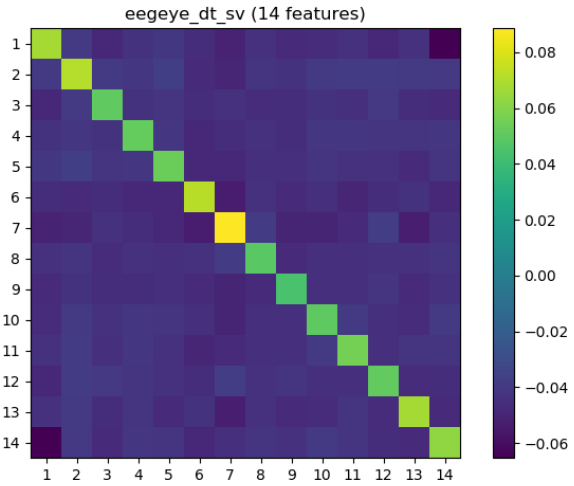

bcancer_dt_bv (10 features)
1 2 3 4 5 6 7 8 9 10
0.02 0.01 0.00 −0.01 −0.02

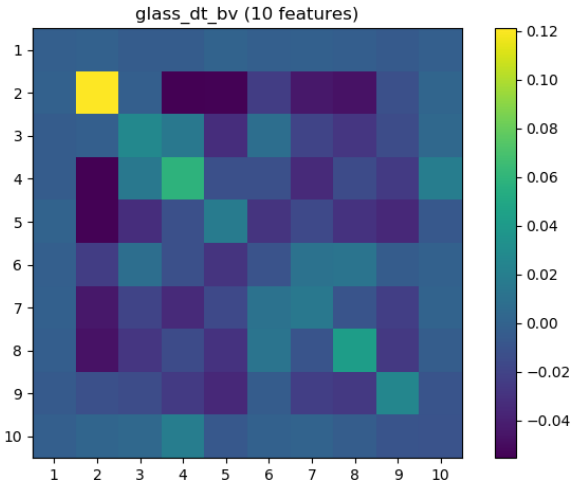

bcancer_dt_sv (10 features)
1 2 3 4 5 6 7 8 9 10
0.10 0.05 0.00 −0.05 −0.10

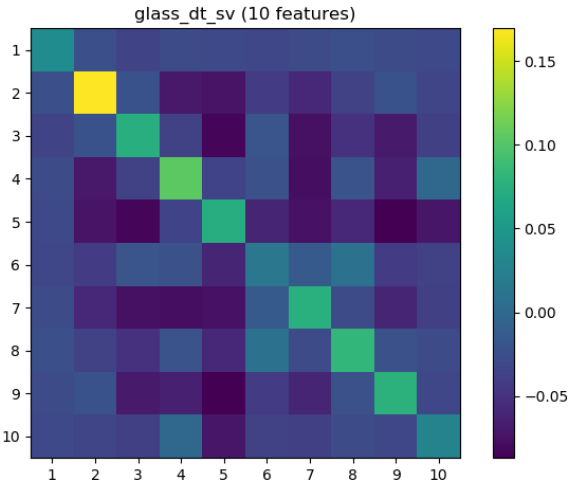

climate_crashes_dt_bv (20 features)

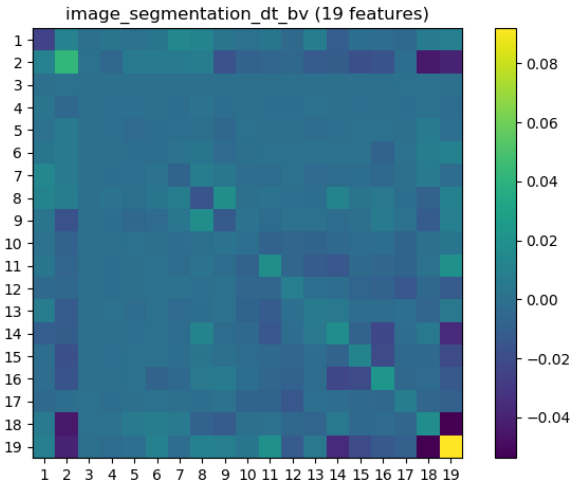

climate_crashes_dt_sv (20 features)

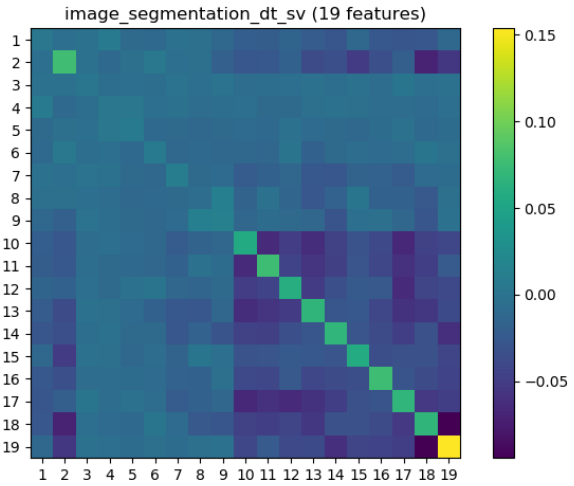

eegeye_dt_bv (14 features)

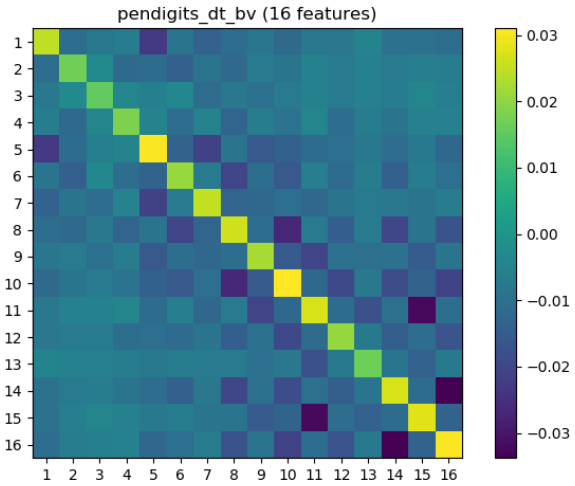

eegeye_dt_sv (14 features)

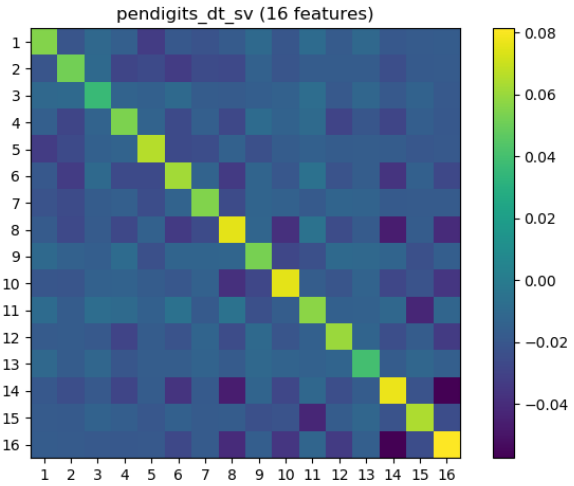

glass_dt_bv (10 features)

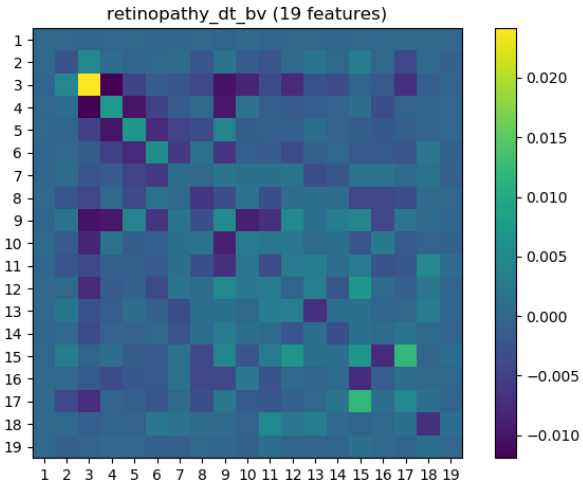

glass_dt_sv (10 features)

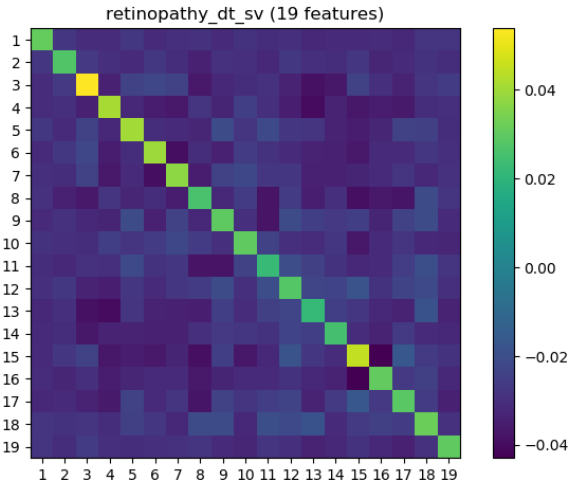

image_segmentation_dt_bv (19 features)

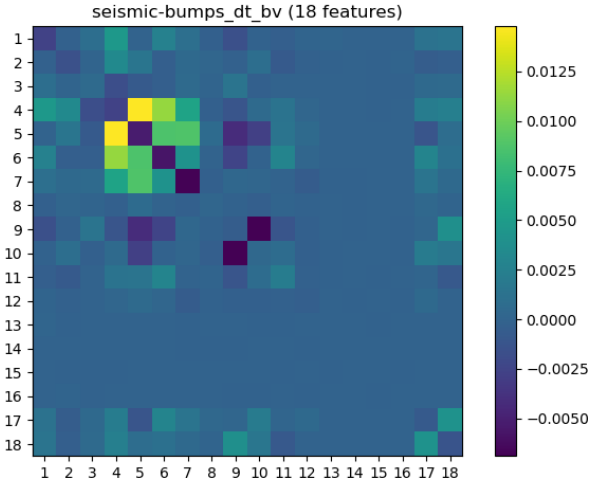

image_segmentation_dt_sv (19 features)

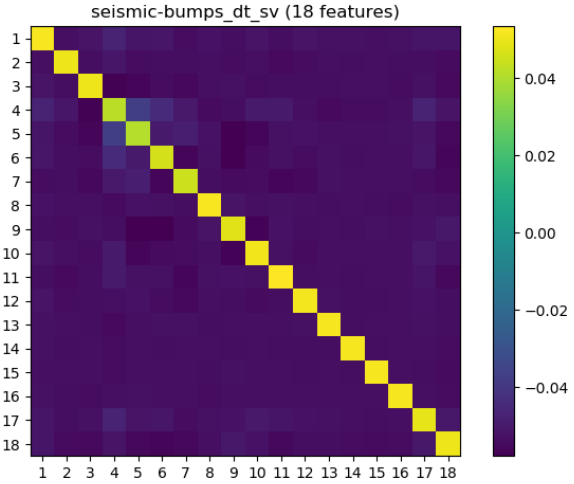

pendigits_dt_bv (16 features)

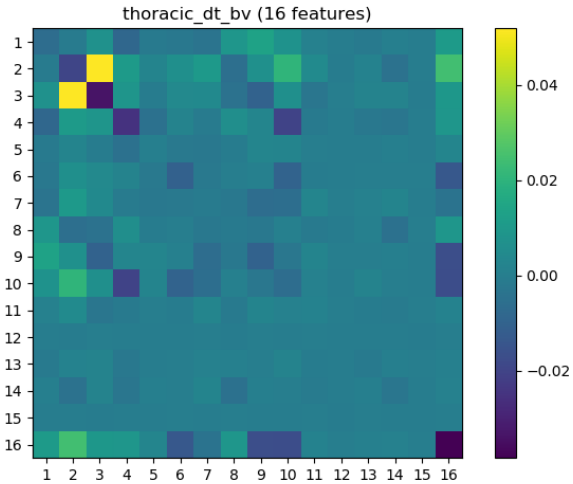

pendigits_dt_sv (16 features)

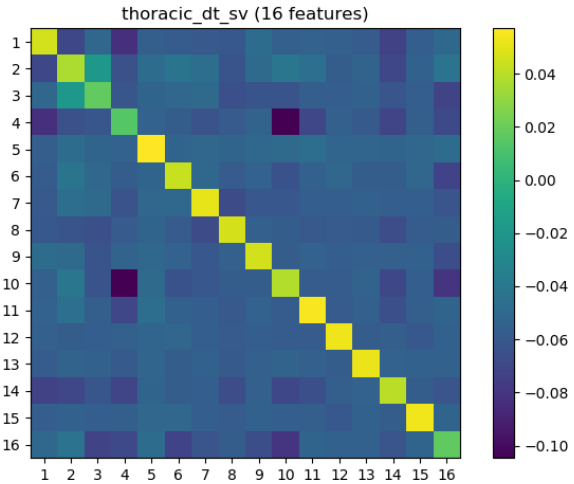

retinopathy_dt_bv (19 features)

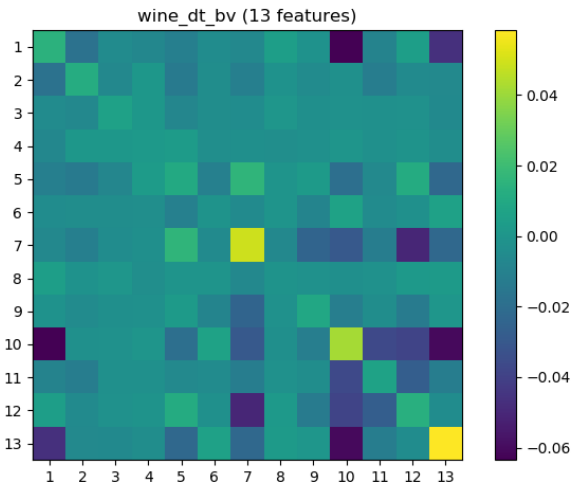

retinopathy_dt_sv (19 features)

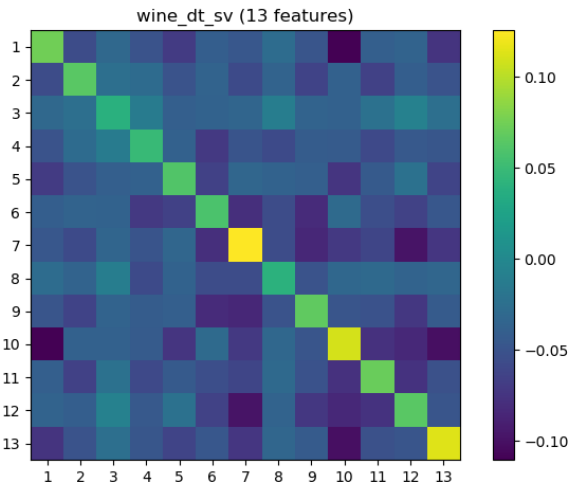

seismic-bumps_dt_bv (18 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050

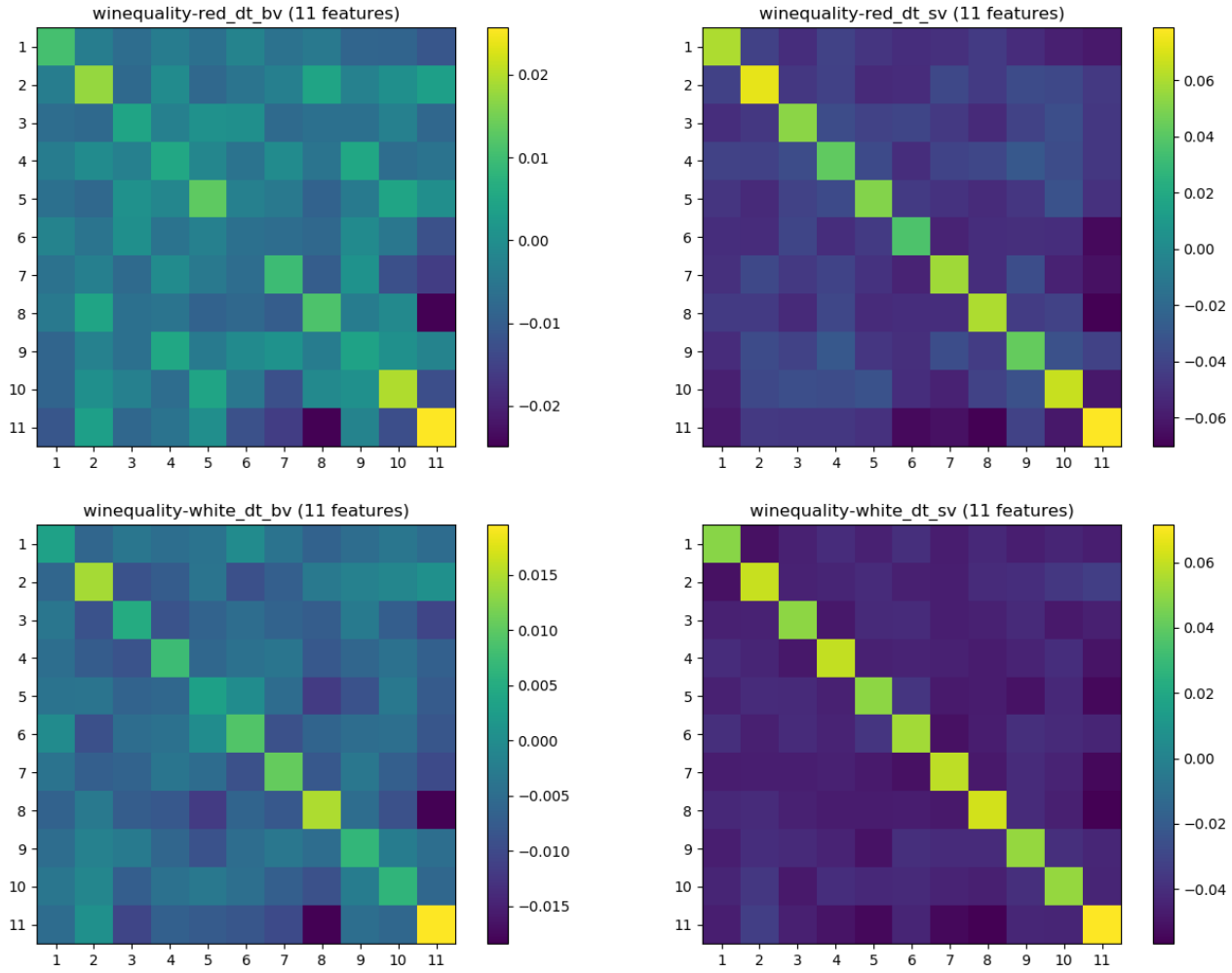

seismic-bumps_dt_sv (18 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
0.04
0.02
0.00
−0.02
−0.04


thoracic_dt_bv (16 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0.04
0.02
0.00
−0.02


thoracic_dt_sv (16 features)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.10


wine_dt_bv (13 features)
1 2 3 4 5 6 7 8 9 10 11 12 13
0.04
0.02
0.00
−0.02
−0.04
−0.06


wine_dt_sv (13 features)
1 2 3 4 5 6 7 8 9 10 11 12 13
0.10
0.05
0.00
−0.05
−0.10



8.6 Feature Selection

Another question is: is there an algorithm that works better than the others?

To answer this question, we have compared the quality of the feature selection using the following algorithms:

- `ws` - *Worst Set*: it select the set with the worst accuracy
- `bs` - *Best Set*: it select the set with the best accuracy

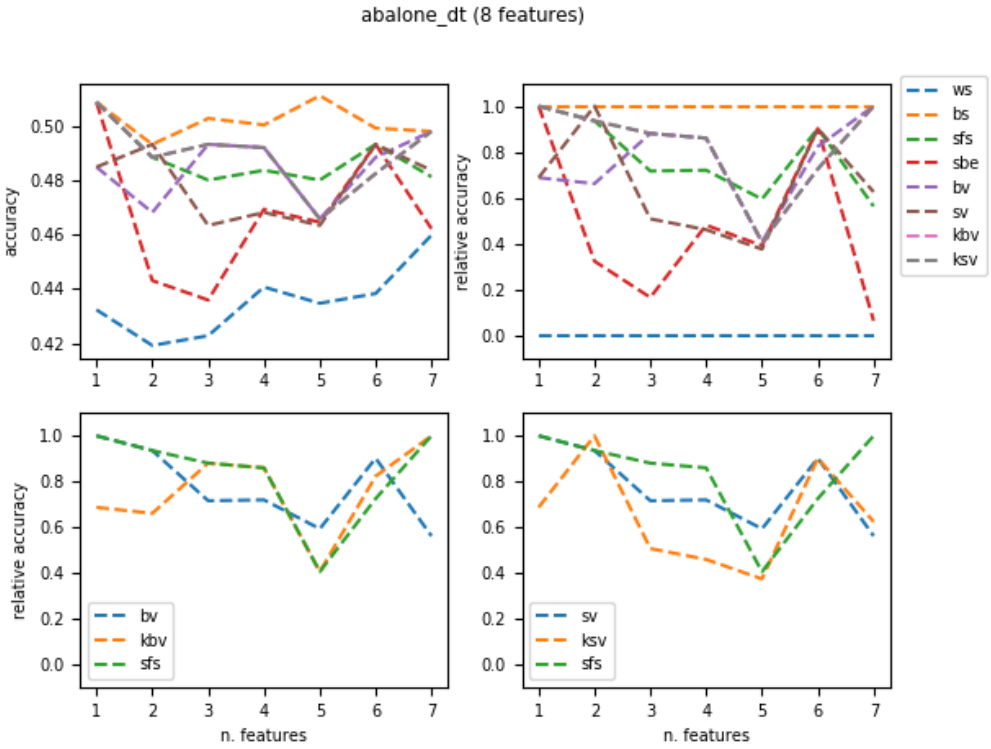
- `sfs` - *Sequential Forward Selection*: a classic greedy algorithm that, at each step, add the *best* feature
- `sbe` - *Sequential Backward Elimination*: a classic greedy algorithm that, at each step, remove the *worst* feature
- `sv` - *Shapley Value*: it select the features with the best Shapley Value
- `bv` - *Banzhaf Value*: it select the features with the best Banzhaf Value
- `ksv` - *K-Shapley Value*: it select the features with the best Shapley Value computed on sets with cardinality k
- `kbv` - *K-Banzhaf Value*: it select the features with the best Banzhaf Value computed on sets with cardinality k

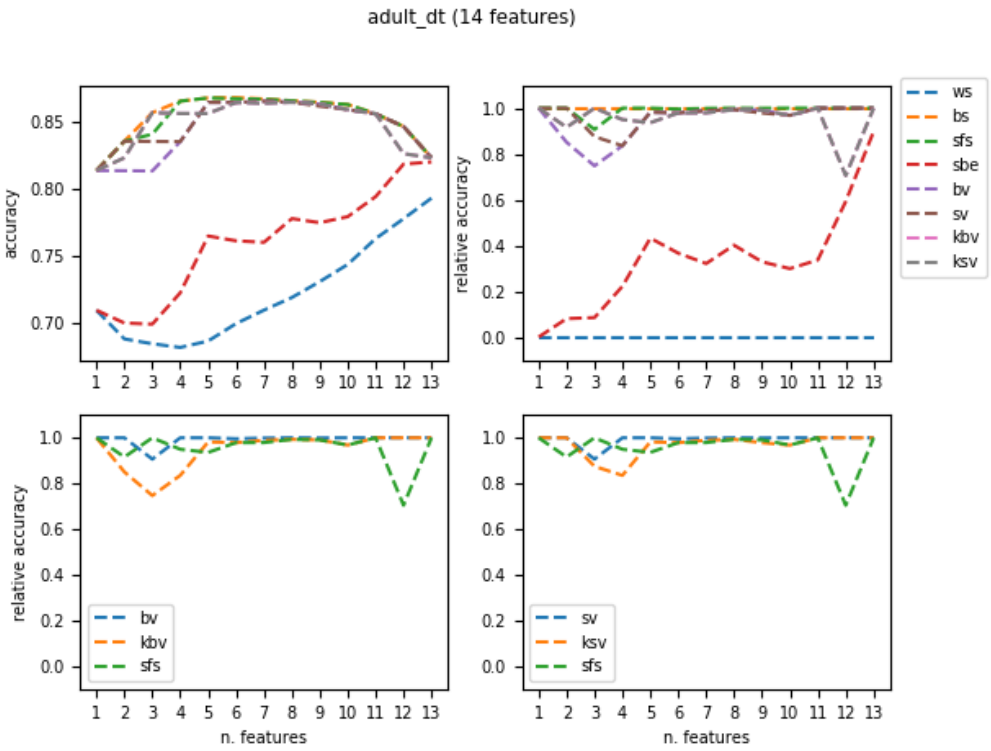
To find the *worst set* and the *best set* we have implemented a library with several *heuristic optimization algorithms* (containing several algorithms available in [164]). We have used two different algorithms (Tabu-Search and Genetic Algorithms) executed three times (with different parameters) to be sure to obtain the same results.

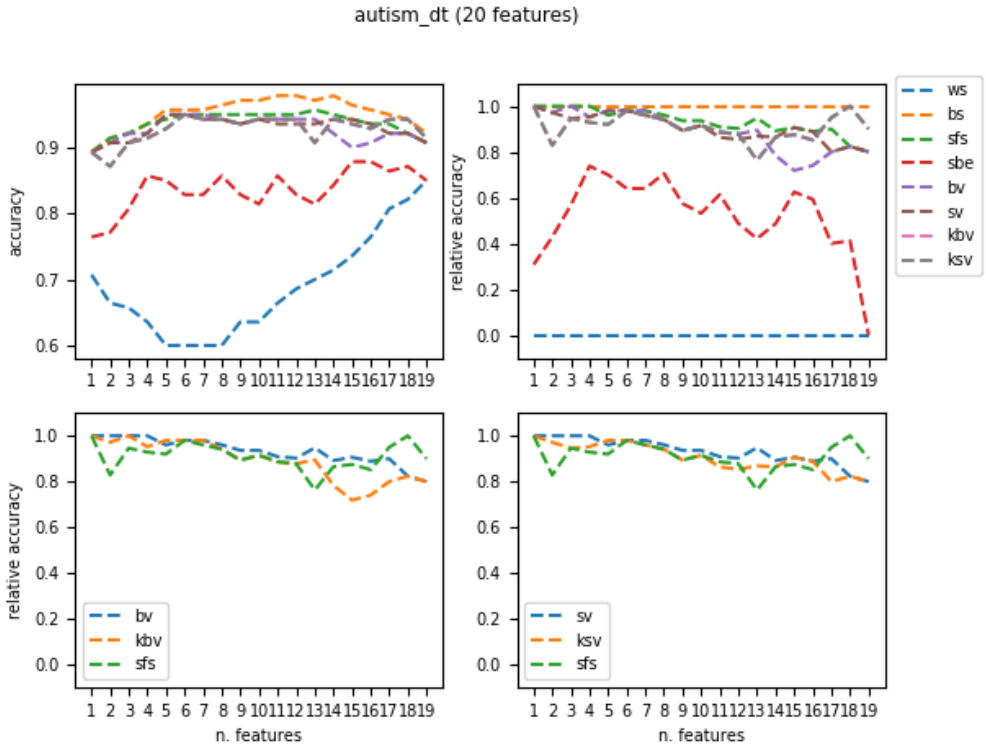
For each dataset there are 4 plots:

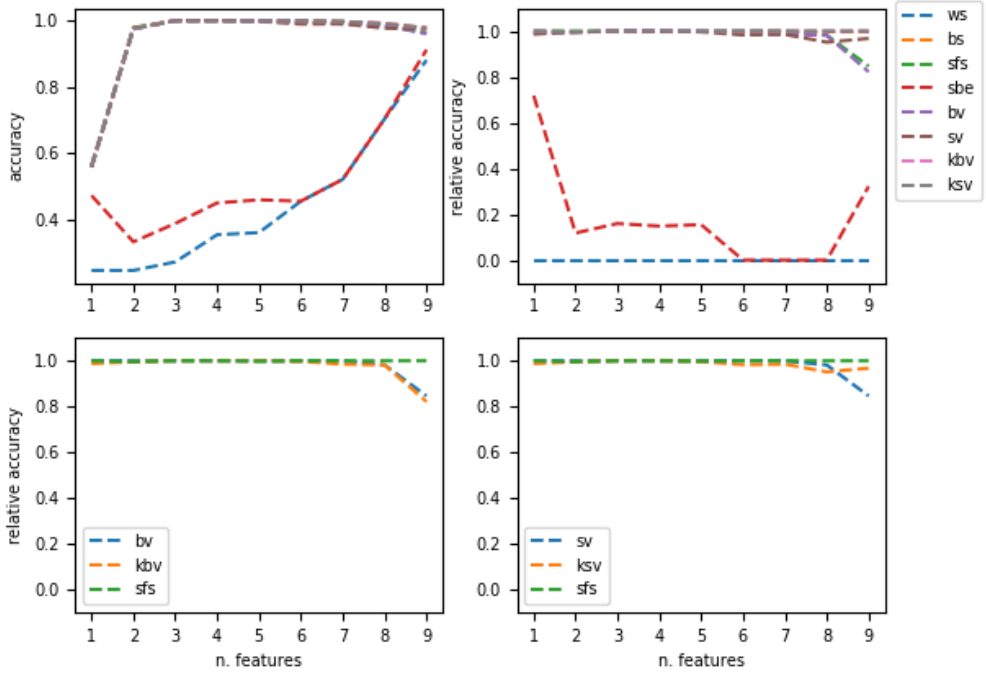
1. the accuracy of the set identified by the algorithm
2. the *relative* accuracy, where 0 is the *worst* accuracy, and 1 the *best* accuracy
3. the comparison between the relative accuracy obtained selecting the features using Banzhaf Value (blue) and K-Banzhaf Value (yellow)
4. the comparison between the relative accuracy obtained selecting the features using Shapley Value (blue) and K-Shapley Value (yellow)

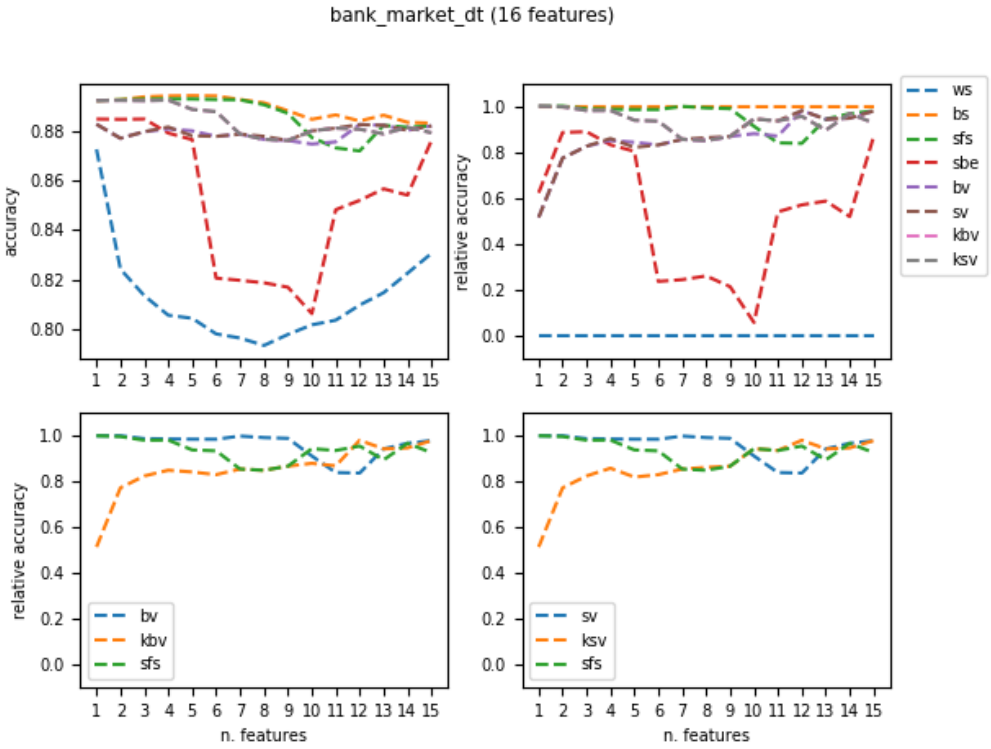
It is possible observe that there is not an algorithm that performs better than others. Also, the k variant of the power indices are not better than the original one. The last observation is that very often, when the greedy method (SFS) works well, the power index approach works bad, and when the greedy works bad, the power index works well. Sometimes, the methods are equivalent.

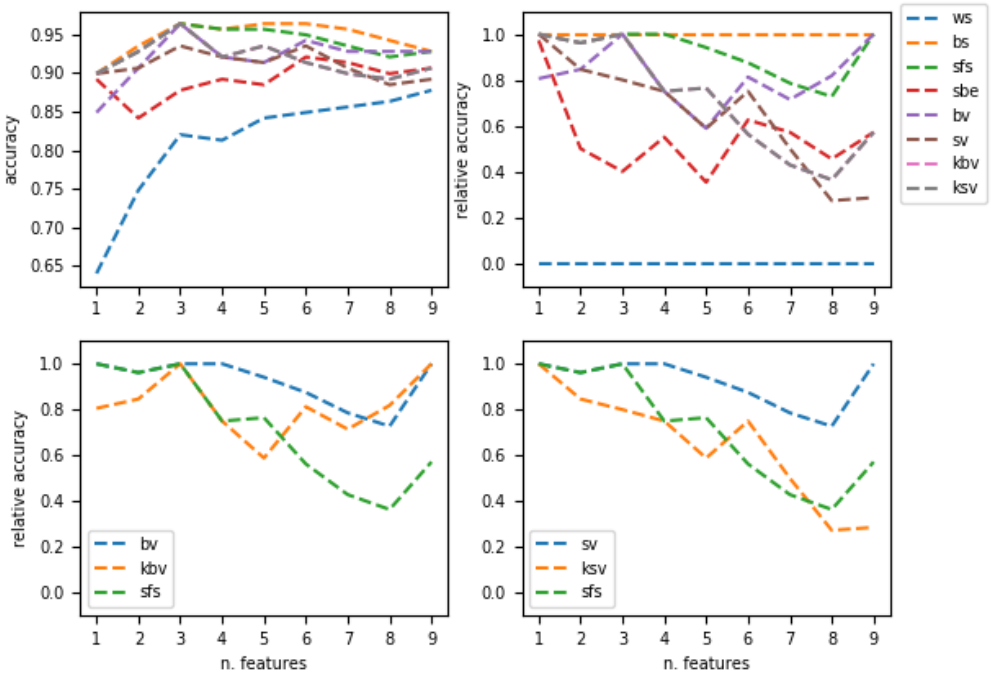

abalone_dt (8 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs

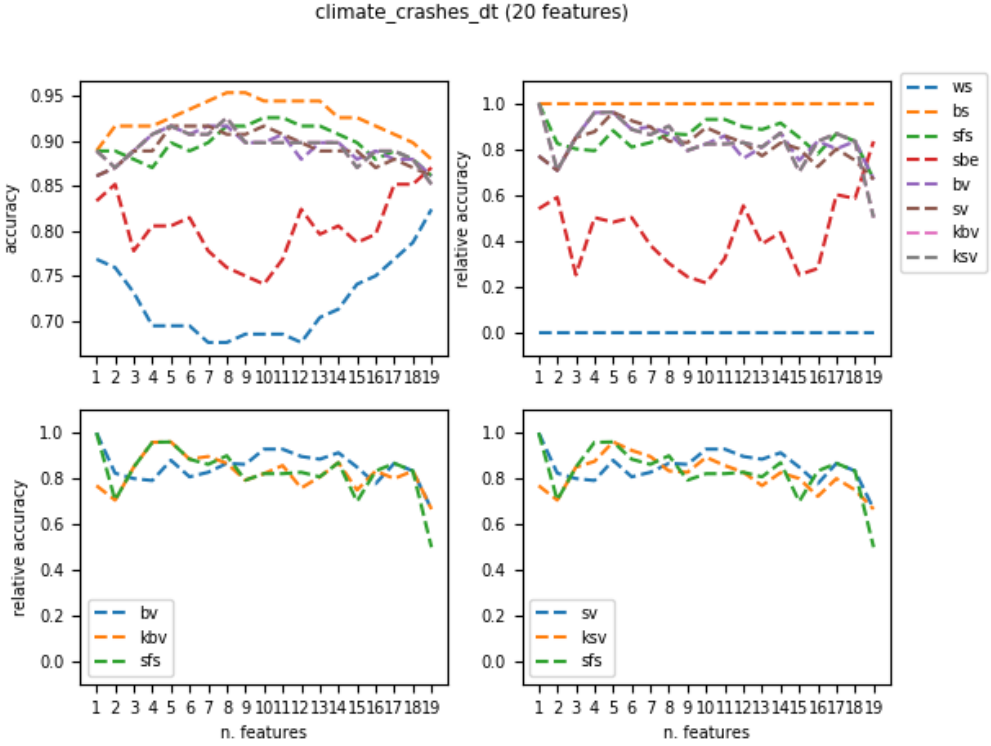

adult_dt (14 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs

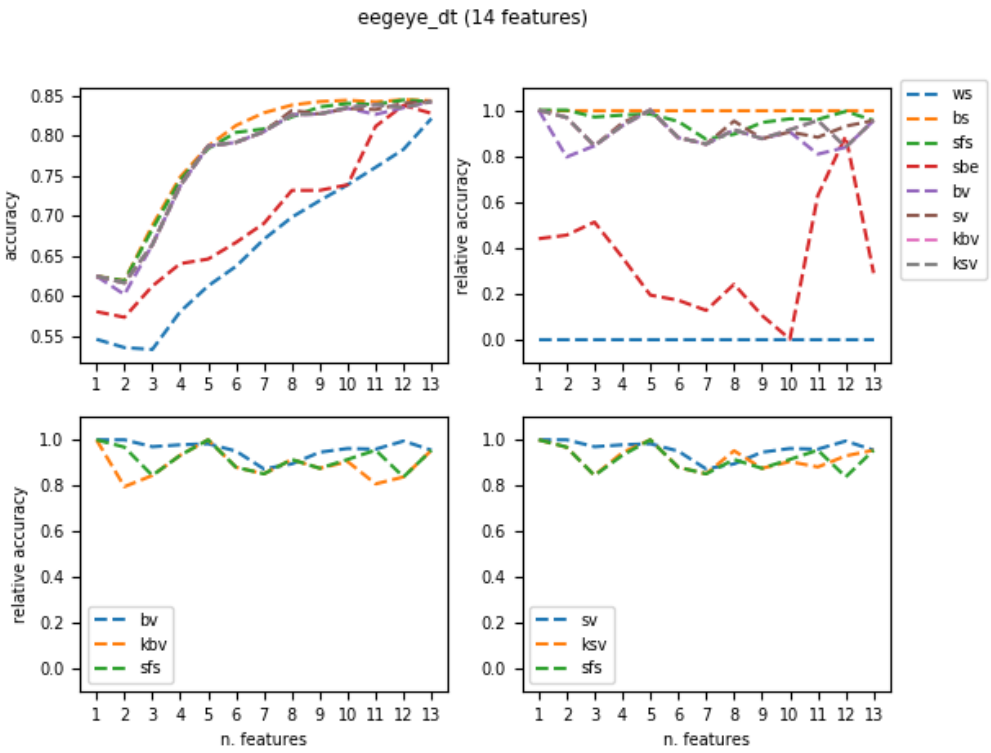

autism_dt (20 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv

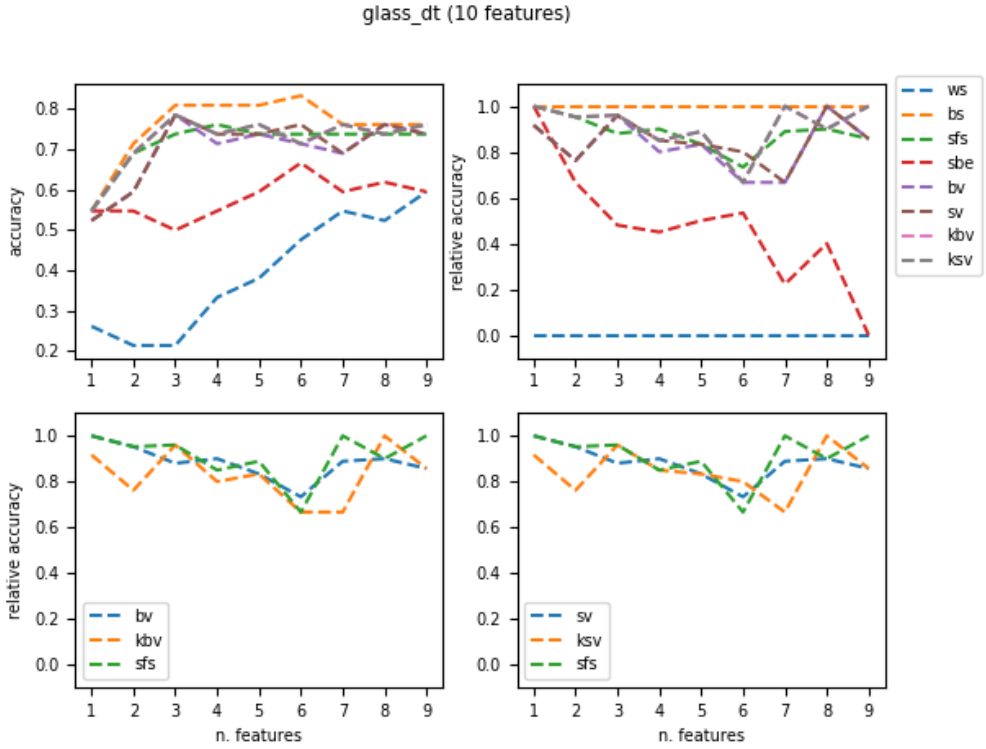

avila_dt (10 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv

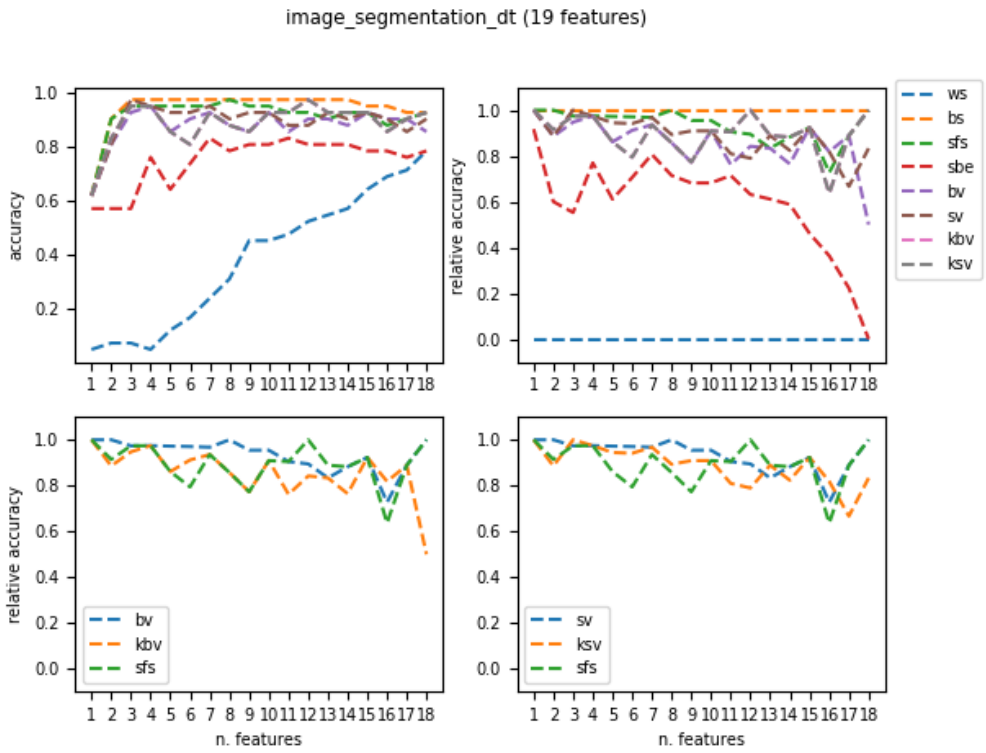

bank_market_dt (16 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features

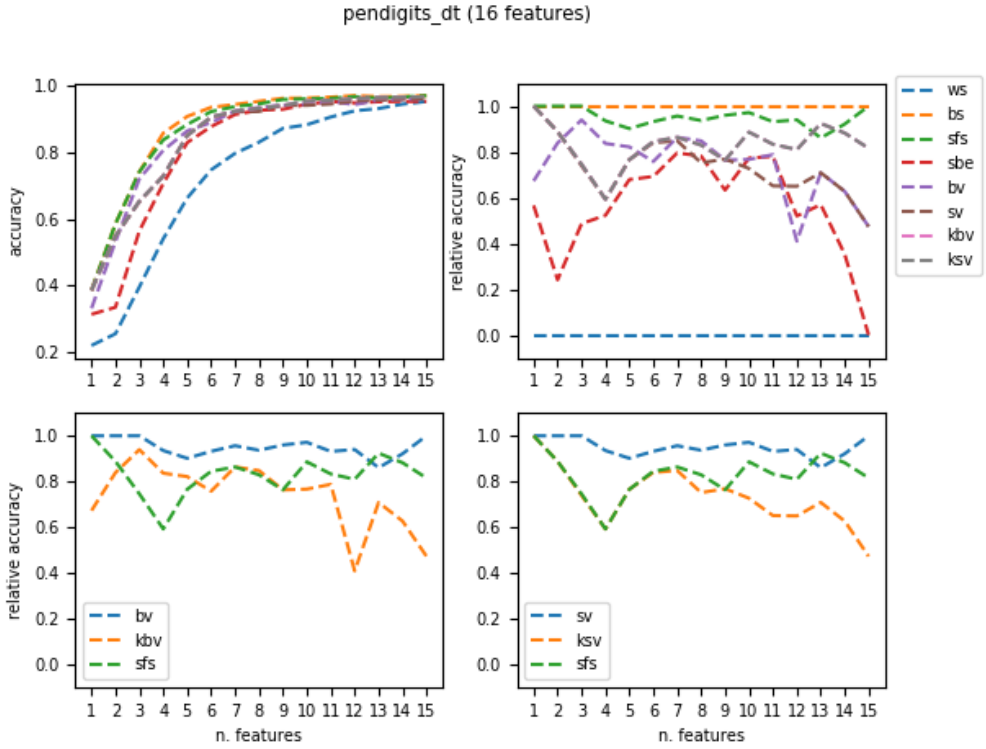

bcancer_dt (10 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features

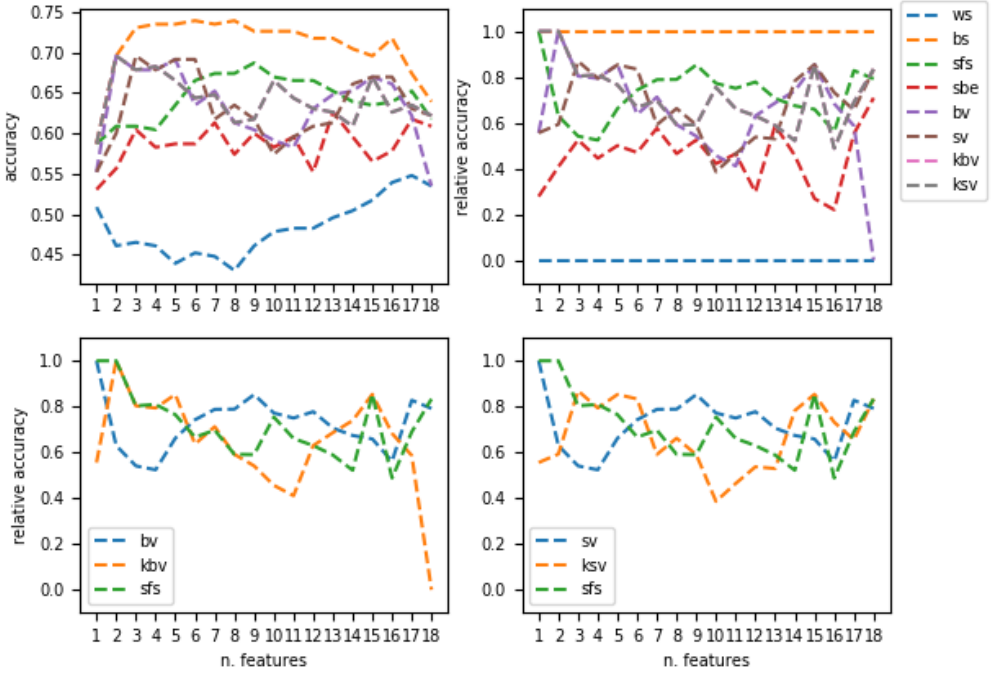

climate_crashes_dt (20 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv

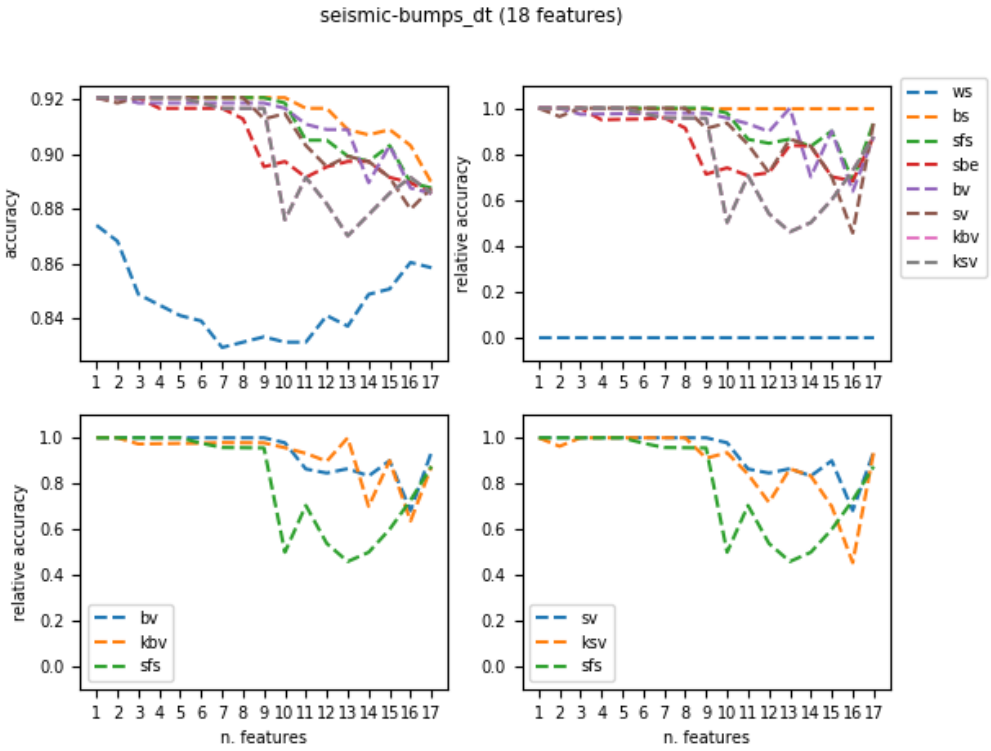

eegeye_dt (14 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv

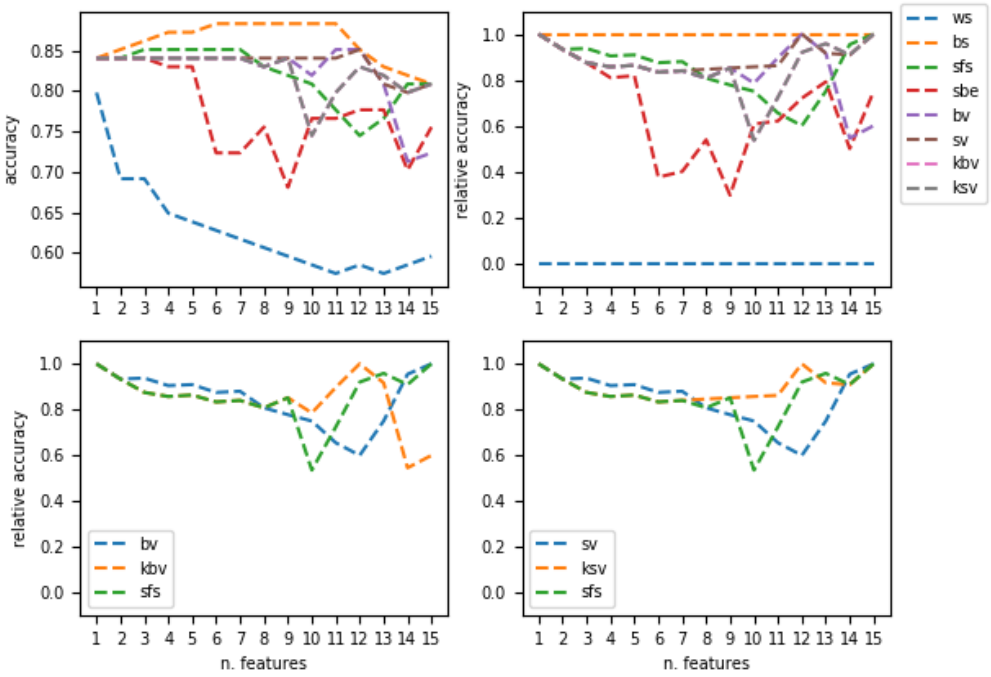

glass_dt (10 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features

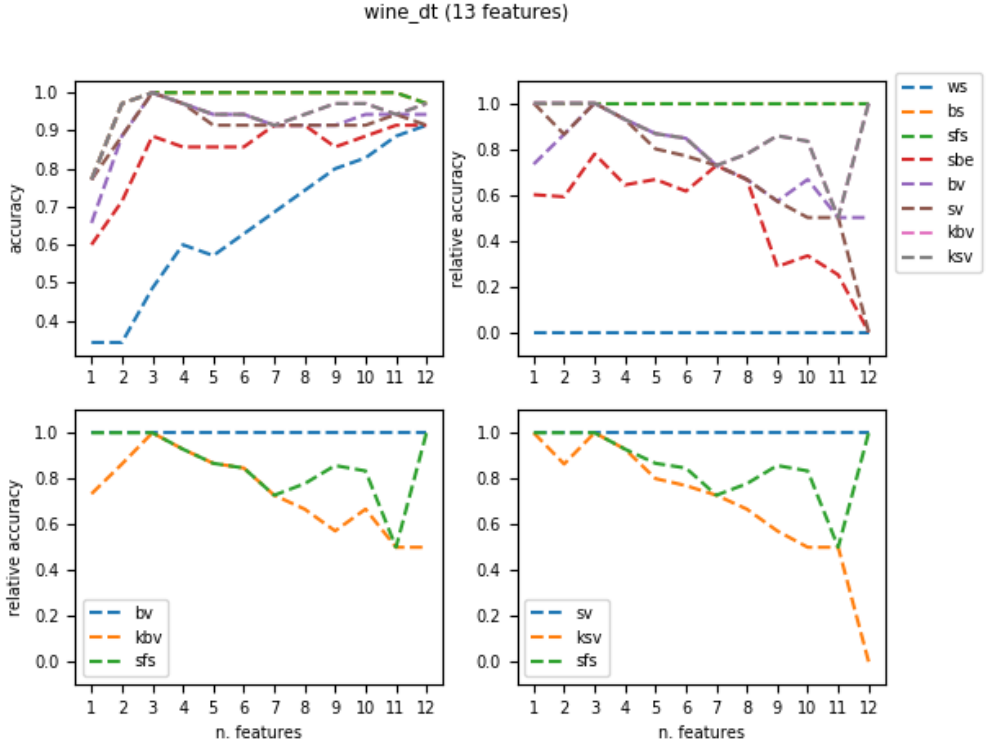

image_segmentation_dt (19 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features

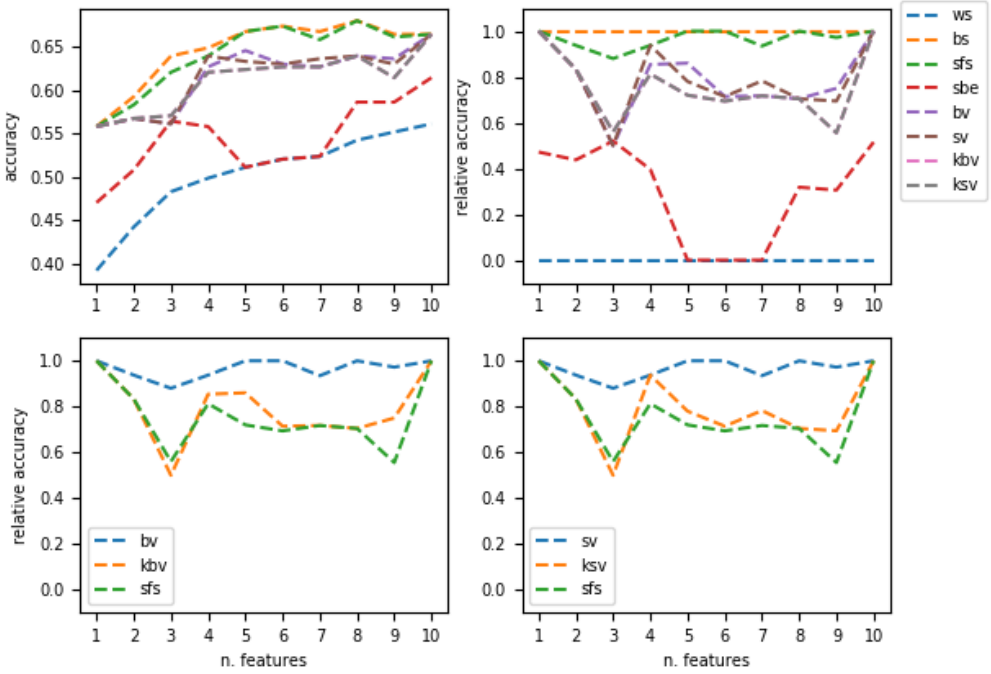

pendigits_dt (16 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv

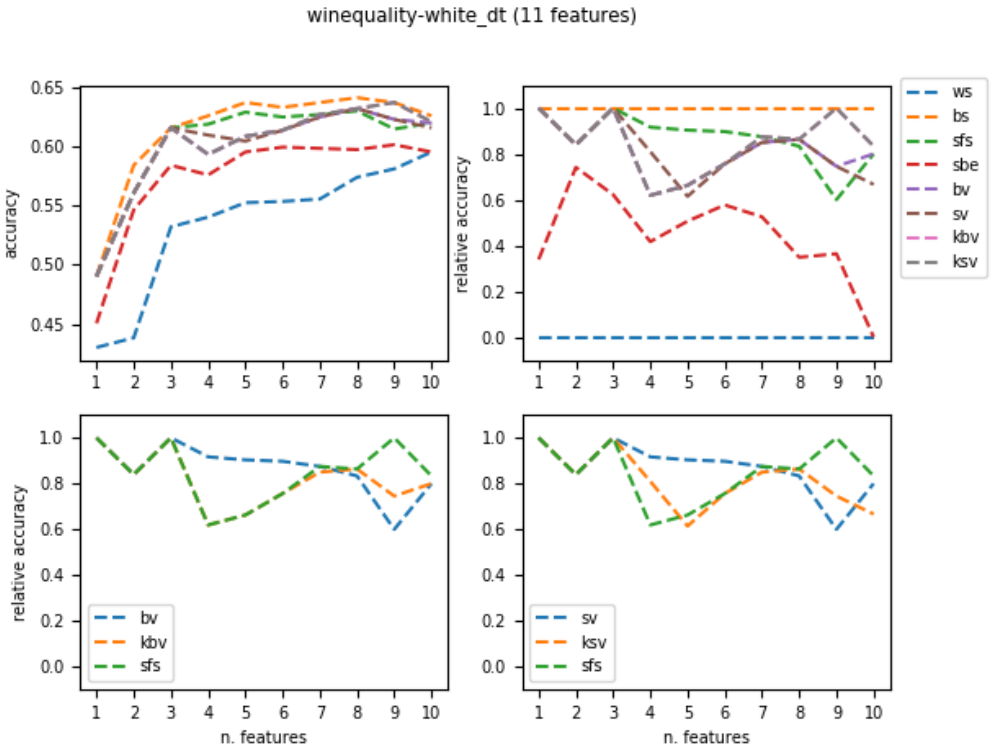

retinopathy_dt (19 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv


seismic-bumps_dt (18 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features


thoracic_dt (16 features)
accuracy
relative accuracy
ws
bs
sfs
sbe
bv
sv
kbv
ksv
bv
kbv
sfs
sv
ksv
sfs
n. features
n. features


wine_dt (13 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv


winequality-red_dt (11 features)
accuracy
relative accuracy
n. features
ws
bs
sfs
sbe
bv
sv
kbv
ksv



8.7 Power Indices vs Greedy Method

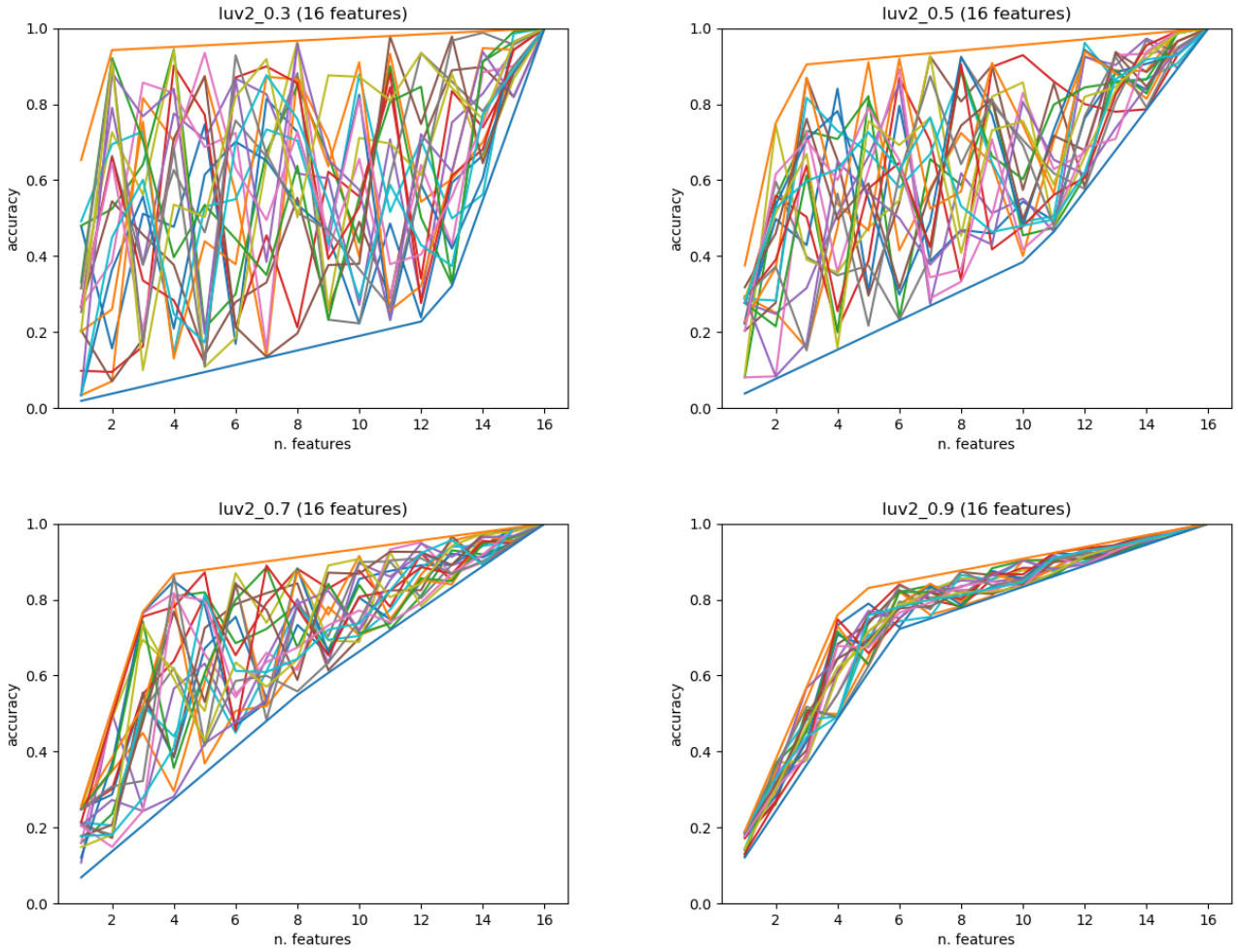
To evaluate when the approach based on power indices is better than the greedy algorithm, we must apply each approach to several different datasets. The main problem is to find these data sets. An alternative solution is to generate *synthetic set functions*.

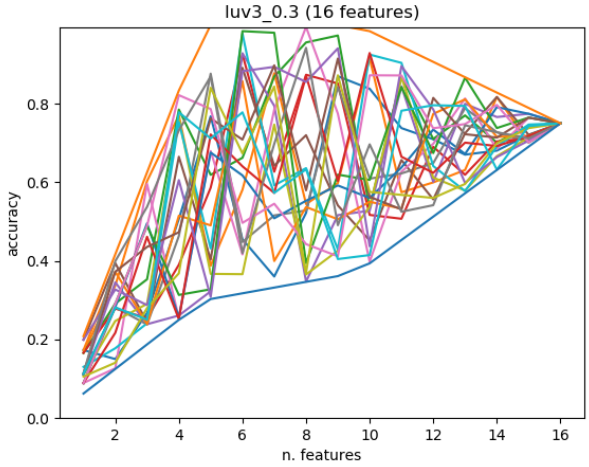
To generate *realistic* set functions, that is functions with a behaviour similar to real ones, is very difficult. We have tried several approaches, and some of them are:

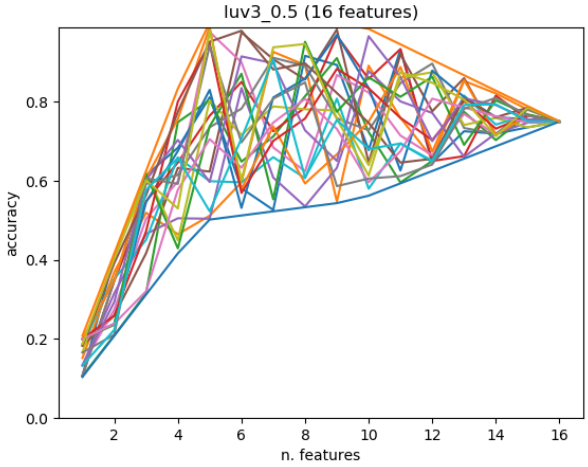
- it is defined a *lower bound* and an *upper bound* for each cardinality, and the value of the function on a set S is a random number between these bounds. We have used two bound's *profiles* and for each profile we have used some *squeezed* variants (images `luv2` and `luv3`)
- we can generate the function as sum of interaction indices until a max-

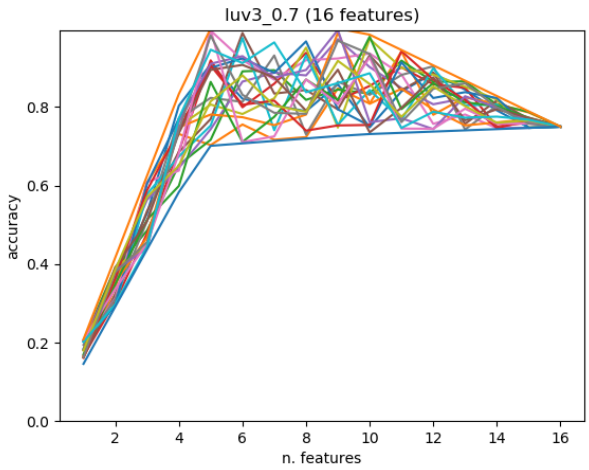
imum degree (in general this is n, but this is not necessary - images `collab`).

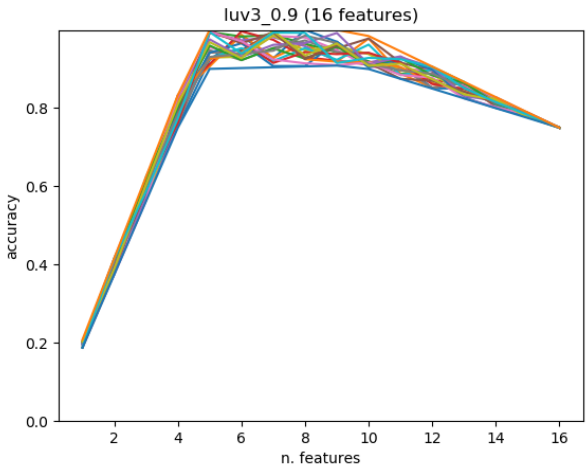
- we can generate the function from random Möbius coefficients (images `mobius`)

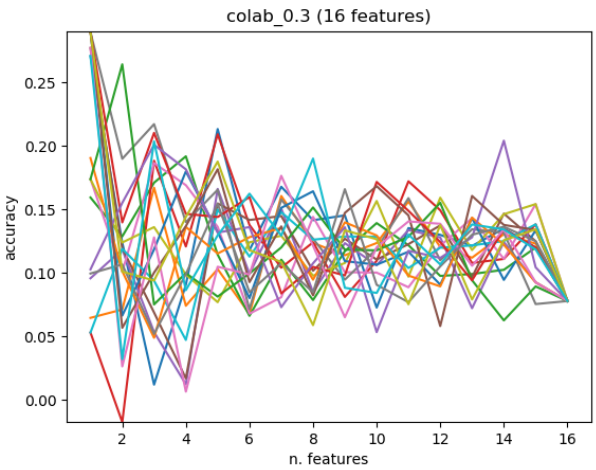


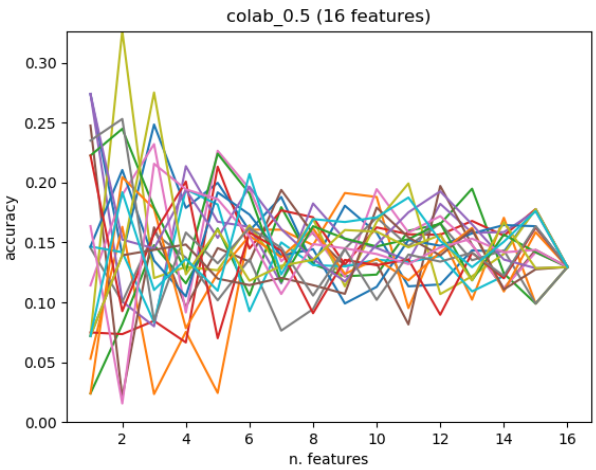

luv3_0.3 (16 features)
accuracy
n. features

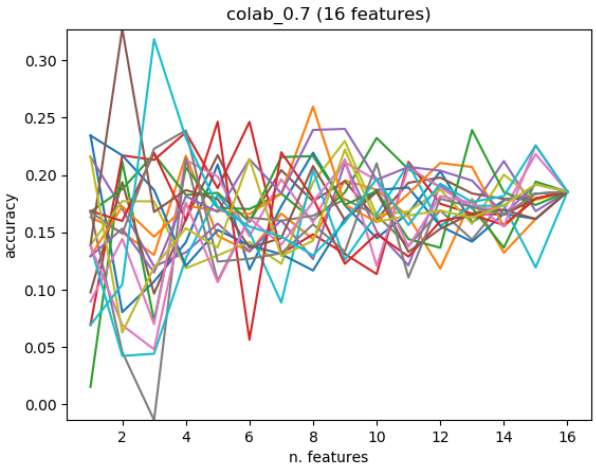

luv3_0.5 (16 features)
accuracy
n. features

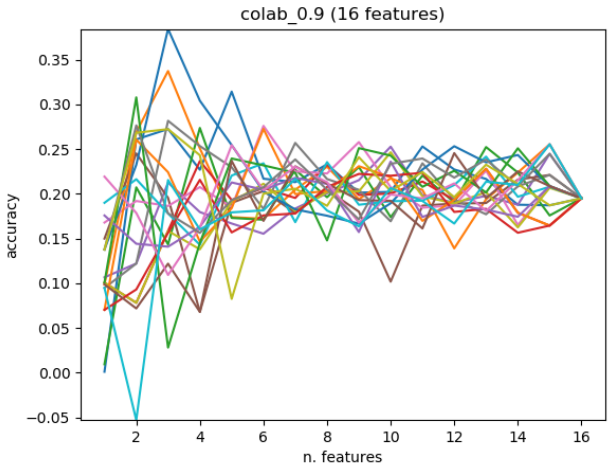

luv3_0.7 (16 features)
accuracy
n. features

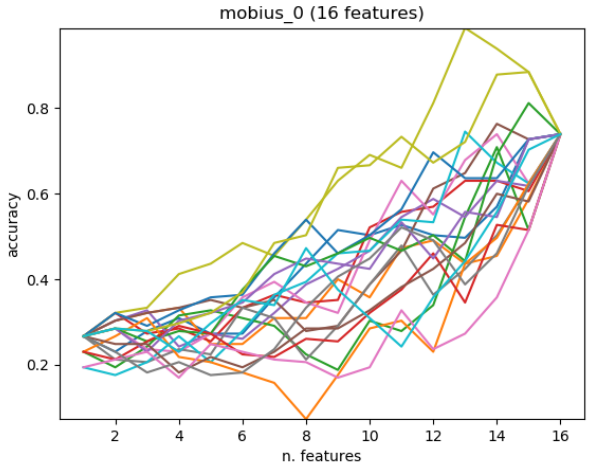

luv3_0.9 (16 features)
accuracy
n. features

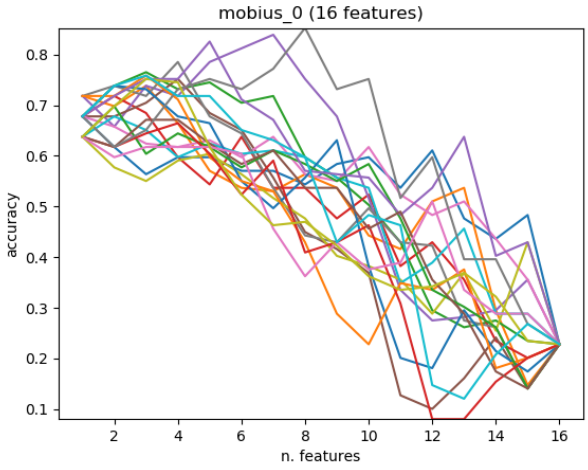

colab_0.3 (16 features)
accuracy
n. features


colab_0.5 (16 features)
accuracy
n. features









We have observed very *strange* behaviours. For example, consider the following function properties

name	nf	mon	k	ws	bs	pfs	pbv	psv	pkbv	pksv
f57760	16	0.5686	1	0.034	0.612	1	1	1	1	1
f57760	16	0.5686	2	0.0581	0.9342	0.9515	0.6916	0.8359	0.4835	0.4835
f57760	16	0.5686	3	0.0586	0.9445	0.9106	0.6752	0.1874	0.7572	0.7572
f57760	16	0.5686	4	0.0761	0.9485	0.9664	0.9935	0.1722	0.6055	0.6055
f57760	16	0.5686	5	0.0949	0.9545	0.9881	0.952	0.223	0.1593	0.1593
f57760	16	0.5686	6	0.114	0.9587	0.8275	0.0477	0.7012	0.002	0.002
f57760	16	0.5686	7	0.1329	0.9628	1	0.2974	0.7332	0.1985	0.1985
f57760	16	0.5686	8	0.1519	0.967	0.8212	0.7317	0.6112	0.3025	0.3025
f57760	16	0.5686	9	0.1709	0.971	0.8718	0.2286	0.4117	0.5806	0.5806
f57760	16	0.5686	10	0.1899	0.9752	0.9808	0.1029	0.8362	0.1029	0.1029
f57760	16	0.5686	11	0.2091	0.979	0.8523	0.4702	0.894	0.1101	0.1101
f57760	16	0.5686	12	0.2279	0.9829	0.9913	0.7856	0.7073	0.923	0.923
f57760	16	0.5686	13	0.3217	0.9854	0.9159	0.5141	0.8635	0.5028	0.5028
f57760	16	0.5686	14	0.5516	0.9879	0.9342	0.2585	0.1041	0.8964	0.8964
f57760	16	0.5686	15	0.775	0.9922	0.8683	0.8683	0.511	0.994	0.994

the columns are:

1. `name`: name of the function
2. `nf`: number of features
3. `k`: cardinality of the subset/n. of features selected
4. `ws`: (*worst set*) worst function's value for the selected cardinality
5. `bs`: (*best set*) best function's value for the selected cardinality
6. `pfs`: *relative* function's value on the set selected by *sequential forward selection* (in the range $[0, 1]$ where 0 is the worst value and 1 is the best value)
7. `pbv`: relative value of the set selected using the Banzhaf Value
8. `psv`: relative value of the set selected using the Shapley Value
9. `pkbv`: relative value of the set selected using the K-Banzhaf Value
10. `pksv`: relative value of the set selected using the K-Shapley Value

We can observe that:

- the greedy method is very good

- for some cardinality, the power index based approach is very bad (yellow cells)
- methods based on K-Banzhaf Value and K-Shapley Value identify the *same set* (the column values are the same). We have checked several times to be sure that the values calculated with the two methods are actually different

We have also cases where the approach based on the game theory is very good:

name	nf	mon	k	ws	bs	pfs	pbv	psv	pkbv	pksv
f58758_0	16	0.8183	1	0.1939	0.2667	0.9974	1	1	1	1
f58758_0	16	0.8183	2	0.1394	0.3212	0.9985	1	1	1	1
f58758_0	16	0.8183	3	0.0909	0.4061	0.8845	0.9423	0.9423	0.9613	0.9613
f58758_0	16	0.8183	4	0.0606	0.4788	0.8694	0.9275	0.9854	0.9275	0.9275
f58758_0	16	0.8183	5	0.0667	0.5636	0.8293	0.9756	0.9756	0.8414	0.8414
f58758_0	16	0.8183	6	0.0424	0.6303	0.866	0.9073	0.9073	0.9073	0.9073
f58758_0	16	0.8183	7	0.0242	0.7212	0.8608	0.9565	0.9565	0.9565	0.9565
f58758_0	16	0.8183	8	0.0061	0.8303	0.8162	0.9486	0.9486	0.9486	0.9486
f58758_0	16	0.8183	9	0.0121	0.9152	0.765	1	1	1	1
f58758_0	16	0.8183	10	0	1	0.7212	1	1	1	1
f58758_0	16	0.8183	11	0.0242	0.9939	0.7313	0.9626	0.9626	0.9938	0.9938
f58758_0	16	0.8183	12	0.097	0.9939	0.7703	0.8311	0.9662	1	1
f58758_0	16	0.8183	13	0.2	0.9939	0.8626	0.7328	1	0.939	0.939
f58758_0	16	0.8183	14	0.3576	0.9394	0.8125	0.677	1	1	1
f58758_0	16	0.8183	15	0.5152	0.8848	0.7541	0.8033	0.8033	1	1

We have observed the same behaviour on all generated functions, regardless of the algorithm used for generation.

It is not clear why there are these behaviours. The main hypothesis is the *randomness* of the function.

8.8 Feature Partitioning

To evaluate the quality of the partitioning, using power and interaction indices and spectral clustering, we compared them with the *worst* and the *best* partitioning identified using an library for *discrete optimization problems*. To be sure that the the solution found are correct, we applied the different

approaches, tuning the parameters, until we obtained the same solutions 3 times.

The results are available in the following table. The columns are:

1. `name`: name of the dataset
2. `nf`: number of features
3. `type`: method used to partition the features
   (a) `ws`: the *worst selected* partitioning possible, based on the set function
   (b) `bs`: the *best selected* partitioning possible, based on the set function
   (c) `bv`: the partitioning based on Banzhaf Value and Interaction Index
   (d) `sv`: the partitioning based on Shapley Value and Interaction Index
   (e) `spbv`: the partitioning based on Spectral Clustering, Banzhaf Value and Interaction Index
   (f) `spsv`: the partitioning based on Spectral Clustering, Shapley Value and Interaction Index
4. `q1/2...`: *accuracy* of the view 1 of the partition in 2 views, etc
5. `qp2...`: *quality* of the partition, as sum of the views' accuracy

A partition is a *good partition* if

- the accuracy of all views are similar
- the accuracy of all views are near the accuracy of the *best partition* (`bs`)

We have experimented a partitioning in 2 and 3 views.

Table 8.3: Feature Partitioning properties

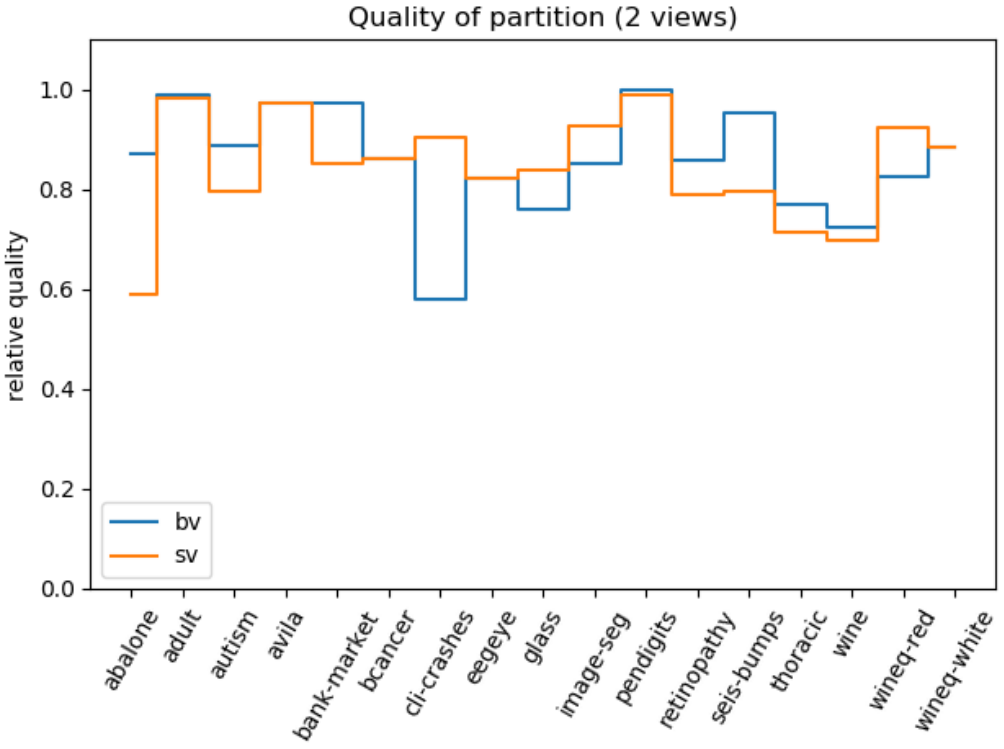
name	nf	type	q1/2	q2/2	qp2	q1/3	q2/3	q3/3	qp3
abalone	8	ws	0.4383	0.4371	0.8754	0.4335	0.4192	0.4599	1.3126
abalone	8	bs	0.4982	0.509	1.0072	0.4994	0.509	0.4994	1.5078
abalone	8	sv	0.479	0.4743	0.9533	0.4778	0.4707	0.4431	1.3916
abalone	8	bv	0.4934	0.497	0.9904	0.503	0.4527	0.4431	1.3988
abalone	8	spbv	0.4647	0.4802	0.9449	0.4539	0.4467	0.4886	1.3892
abalone	8	spsv	0.4766	0.4467	0.9234	0.4335	0.4467	0.4635	1.3437
adult	14	ws	0.8301	0.6841	1.5141	0.8137	0.6841	0.7625	2.2602
adult	14	bs	0.818	0.8348	1.6527	0.787	0.8306	0.835	2.4525
adult	14	sv	0.7853	0.8652	1.6505	0.7753	0.8225	0.8343	2.432
adult	14	bv	0.7838	0.8674	1.6512	0.7698	0.8459	0.8248	2.4405
adult	14	spbv	0.7582	0.7936	1.5518	0.7971	0.788	0.7582	2.3433
adult	14	spsv	0.7774	0.7648	1.5423	0.7625	0.7997	0.709	2.2712
autism	20	ws	0.8857	0.6	1.4857	0.8786	0.6429	0.6571	2.1786
autism	20	bs	0.9571	0.9143	1.8714	0.8857	0.9214	0.9286	2.7357
autism	20	sv	0.85	0.9429	1.7929	0.8786	0.8214	0.8857	2.5857
autism	20	bv	0.8857	0.9429	1.8286	0.8857	0.8714	0.8929	2.65
autism	20	spbv	0.6786	0.8929	1.5714	0.8429	0.9286	0.6786	2.45
autism	20	spsv	0.7643	0.85	1.6143	0.7571	0.8929	0.7214	2.3714
avila	10	ws	0.96	0.2559	1.2159	0.9741	0.2559	0.2466	1.4766
avila	10	bs	0.9717	0.7898	1.7616	0.9578	0.8267	0.4783	2.2629
avila	10	sv	0.9377	0.809	1.7467	0.6053	0.8133	0.63	2.0486
avila	10	bv	0.9377	0.809	1.7467	0.5974	0.7997	0.6302	2.0273
avila	10	spbv	0.9768	0.2737	1.2504	0.9741	0.2466	0.2559	1.4766
avila	10	spsv	0.9768	0.2737	1.2504	0.2559	0.2466	0.9741	1.4766
bank-market	16	ws	0.8279	0.8251	1.6529	0.8264	0.8091	0.8793	2.5148
bank-market	16	bs	0.8786	0.8926	1.7712	0.8786	0.8918	0.8859	2.6564
bank-market	16	sv	0.874	0.8799	1.7538	0.8665	0.8818	0.891	2.6393
bank-market	16	bv	0.8761	0.8921	1.7682	0.8747	0.8832	0.8921	2.65
bank-market	16	spbv	0.8628	0.8296	1.6924	0.878	0.8665	0.8922	2.6367

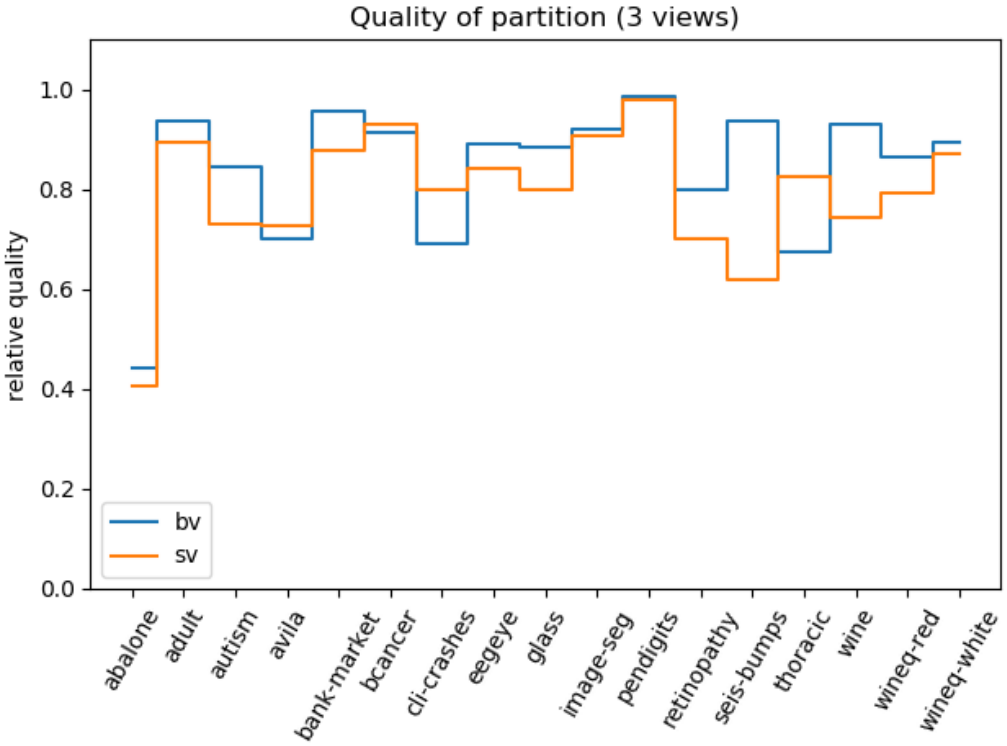
name	nf	type	q1/2	q2/2	qp2	q1/3	q2/3	q3/3	qp3
bank-market	16	spsv	0.8708	0.8243	1.6951	0.8142	0.8805	0.875	2.5698
bcancer	10	ws	0.6403	0.8921	1.5324	0.6403	0.8849	0.7914	2.3165
bcancer	10	bs	0.9568	0.9424	1.8993	0.9496	0.9353	0.9424	2.8273
bcancer	10	sv	0.9353	0.9137	1.8489	0.9353	0.9281	0.9281	2.7914
bcancer	10	bv	0.9353	0.9137	1.8489	0.9496	0.9065	0.9281	2.7842
bcancer	10	spbv	0.9065	0.8777	1.7842	0.8417	0.9065	0.9065	2.6547
bcancer	10	spsv	0.8129	0.8777	1.6906	0.8849	0.9137	0.9137	2.7122
cli-crashes	20	ws	0.8241	0.7037	1.5278	0.6759	0.75	0.787	2.213
cli-crashes	20	bs	0.9259	0.8889	1.8148	0.8981	0.9074	0.9167	2.7222
cli-crashes	20	sv	0.9074	0.8796	1.787	0.8704	0.8704	0.8796	2.6204
cli-crashes	20	bv	0.8148	0.8796	1.6944	0.8241	0.8796	0.8611	2.5648
cli-crashes	20	spbv	0.8333	0.8426	1.6759	0.8611	0.8333	0.8426	2.537
cli-crashes	20	spsv	0.8426	0.7778	1.6204	0.8148	0.8241	0.8148	2.4537
eegeye	14	ws	0.8284	0.536	1.3645	0.8074	0.5464	0.5391	1.8929
eegeye	14	bs	0.7707	0.8114	1.5821	0.7644	0.7814	0.6182	2.1639
eegeye	14	sv	0.7964	0.747	1.5434	0.6822	0.7387	0.6999	2.1208
eegeye	14	bv	0.7964	0.747	1.5434	0.7109	0.7387	0.6846	2.1342
eegeye	14	spbv	0.5738	0.8218	1.3955	0.5738	0.8164	0.5831	1.9733
eegeye	14	spsv	0.6335	0.8064	1.4399	0.5738	0.8061	0.5484	1.9282
glass	10	ws	0.2143	0.6905	0.9048	0.3333	0.6905	0.2143	1.2381
glass	10	bs	0.8333	0.6667	1.5	0.5952	0.7381	0.7381	2.0714
glass	10	sv	0.6905	0.7143	1.4048	0.5952	0.7381	0.5714	1.9048
glass	10	bv	0.5952	0.7619	1.3571	0.6905	0.6905	0.5952	1.9762
glass	10	spbv	0.5238	0.7381	1.2619	0.5238	0.6429	0.5238	1.6905
glass	10	spsv	0.2143	0.6905	0.9048	0.4762	0.6905	0.5238	1.6905
image-seg	19	ws	0.8333	0.0476	0.881	0.7857	0.0476	0.0714	0.9048
image-seg	19	bs	0.9286	0.9286	1.8571	0.9048	0.9048	0.9286	2.7381
image-seg	19	sv	0.881	0.9048	1.7857	0.8571	0.881	0.8333	2.5714
image-seg	19	bv	0.881	0.8333	1.7143	0.881	0.8333	0.881	2.5952
image-seg	19	spbv	0.2619	0.881	1.1429	0.4524	0.881	0.2381	1.5714
image-seg	19	spsv	0.1905	0.8571	1.0476	0.2381	0.881	0.1905	1.3095

name	nf	type	q1/2	q2/2	qp2	q1/3	q2/3	q3/3	qp3
pendigits	16	ws	0.9613	0.2197	1.1811	0.9554	0.2197	0.2266	1.4017
pendigits	16	bs	0.9522	0.9409	1.8931	0.8722	0.8967	0.9067	2.6756
pendigits	16	sv	0.9354	0.95	1.8853	0.8658	0.9258	0.8571	2.6488
pendigits	16	bv	0.9463	0.9459	1.8922	0.8722	0.9113	0.8744	2.6579
pendigits	16	spbv	0.8157	0.9031	1.7188	0.7038	0.8617	0.6187	2.1843
pendigits	16	spsv	0.8062	0.8931	1.6993	0.626	0.4763	0.9277	2.03
retinopathy	19	ws	0.4652	0.5478	1.013	0.4391	0.5696	0.4652	1.4739
retinopathy	19	bs	0.6609	0.7217	1.3826	0.6565	0.6522	0.7304	2.0391
retinopathy	19	sv	0.6696	0.6348	1.3043	0.6391	0.6652	0.5652	1.8696
retinopathy	19	bv	0.6739	0.6565	1.3304	0.6304	0.6478	0.6478	1.9261
retinopathy	19	spbv	0.5609	0.6087	1.1696	0.5217	0.5348	0.6087	1.6652
retinopathy	19	spsv	0.513	0.587	1.1	0.587	0.587	0.5609	1.7348
seis-bumps	18	ws	0.8605	0.845	1.7054	0.8605	0.8624	0.845	2.5678
seis-bumps	18	bs	0.9089	0.9205	1.8295	0.9167	0.9205	0.9186	2.7558
seis-bumps	18	sv	0.8876	0.9167	1.8043	0.8702	0.9012	0.9128	2.6841
seis-bumps	18	bv	0.9031	0.9205	1.8236	0.9031	0.9205	0.9205	2.7442
seis-bumps	18	spbv	0.9186	0.8915	1.8101	0.8818	0.9167	0.9205	2.719
seis-bumps	18	spsv	0.8663	0.8779	1.7442	0.874	0.8643	0.8798	2.6182
thoracic	16	ws	0.7021	0.6064	1.3085	0.6915	0.7128	0.7234	2.1277
thoracic	16	bs	0.8191	0.8617	1.6809	0.8404	0.8723	0.8404	2.5532
thoracic	16	sv	0.7872	0.7872	1.5745	0.8511	0.7872	0.8404	2.4787
thoracic	16	bv	0.8191	0.7766	1.5957	0.8511	0.7234	0.8404	2.4149
thoracic	16	spbv	0.8085	0.6809	1.4894	0.8085	0.8085	0.7553	2.3723
thoracic	16	spsv	0.6489	0.7553	1.4043	0.7234	0.8404	0.7128	2.2766
wineq-red	11	ws	0.627	0.3919	1.0188	0.5486	0.4765	0.3919	1.4169
wineq-red	11	bs	0.6552	0.6238	1.279	0.6144	0.6364	0.6301	1.8809
wineq-red	11	sv	0.5862	0.6144	1.2006	0.5674	0.627	0.5674	1.7618
wineq-red	11	bv	0.5799	0.627	1.2069	0.5987	0.6301	0.6207	1.8495
wineq-red	11	spbv	0.5799	0.5674	1.1473	0.5987	0.5549	0.5141	1.6677
wineq-red	11	spsv	0.5705	0.558	1.1285	0.5643	0.5705	0.5643	1.6991
wineq-white	11	ws	0.4382	0.5822	1.0204	0.5822	0.4413	0.4321	1.4556

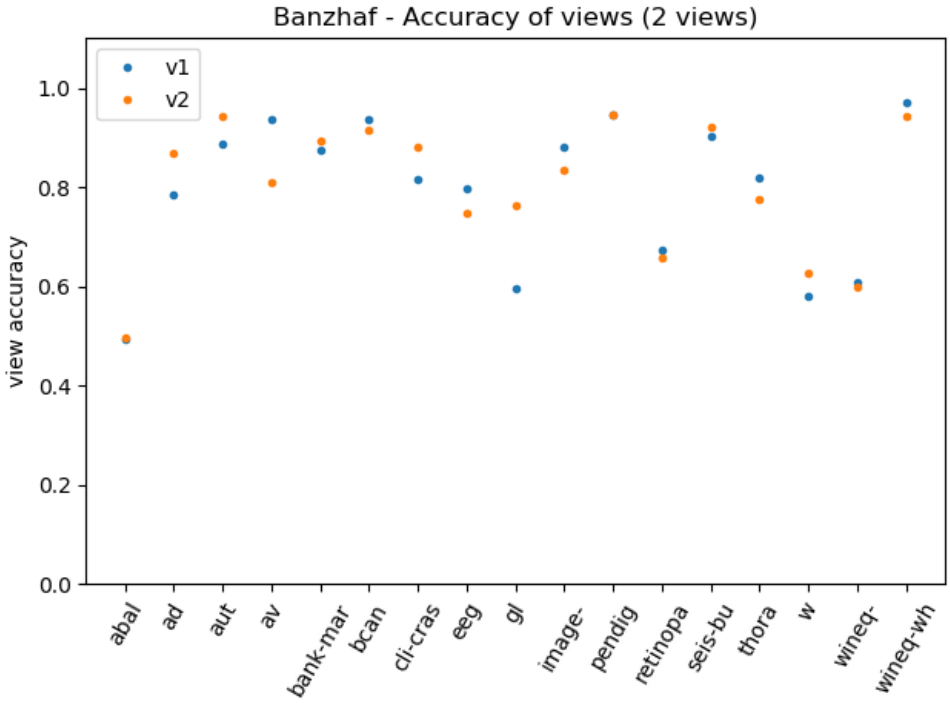
name	nf	type	q1/2	q2/2	qp2	q1/3	q2/3	q3/3	qp3
wineq-white	11	bs	0.621	0.6231	1.2441	0.6016	0.6108	0.62	1.8325
wineq-white	11	sv	0.5996	0.6272	1.2268	0.5751	0.6159	0.5628	1.7538
wineq-white	11	bv	0.6067	0.5986	1.2053	0.572	0.6159	0.5935	1.7814
wineq-white	11	spbv	0.5689	0.5853	1.1542	0.4688	0.5945	0.573	1.6364
wineq-white	11	spsv	0.5914	0.5863	1.1777	0.4688	0.4382	0.5945	1.5015
wine	13	ws	0.9143	0.3429	1.2571	0.9143	0.3429	0.3429	1.6
wine	13	bs	1	1	2	0.9714	0.9714	1	2.9429
wine	13	sv	0.9143	1	1.9143	0.9429	0.9429	0.8857	2.7714
wine	13	bv	0.9714	0.9429	1.9143	0.9714	0.9143	0.9143	2.8
wine	13	spbv	0.8571	0.6857	1.5429	0.9143	0.7143	0.7143	2.3429
wine	13	spsv	0.8571	0.6857	1.5429	0.8857	0.8	0.7429	2.4286

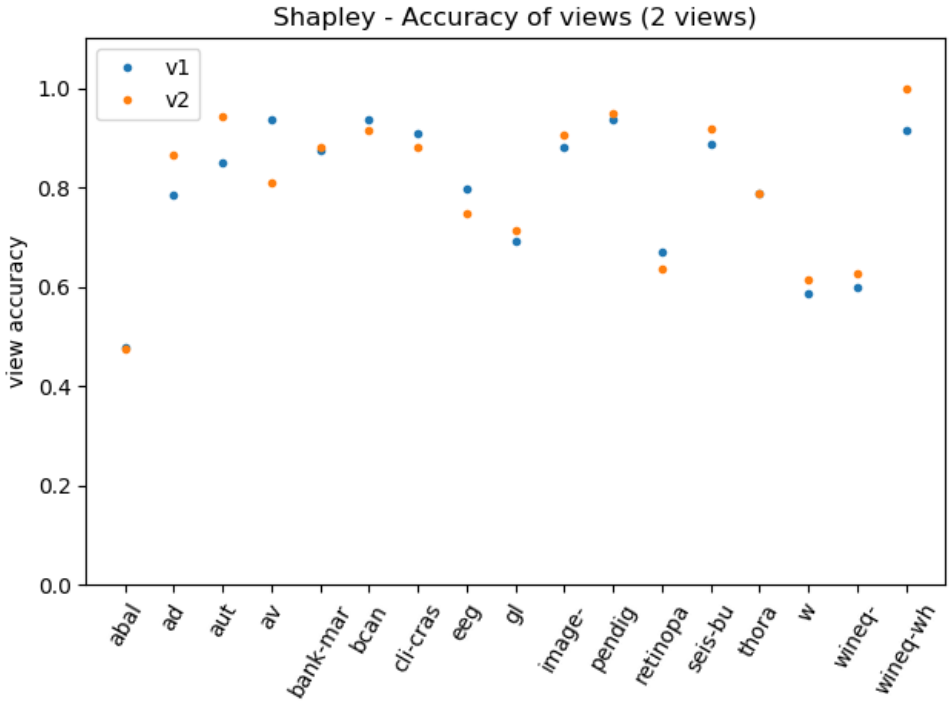
If we map the *worst partition* to 0 and the *best partition* to 1, the following plots show the quality of the partition obtained using the Banzhaf Value and Interaction Index and the Shapley Value and Interaction Index.

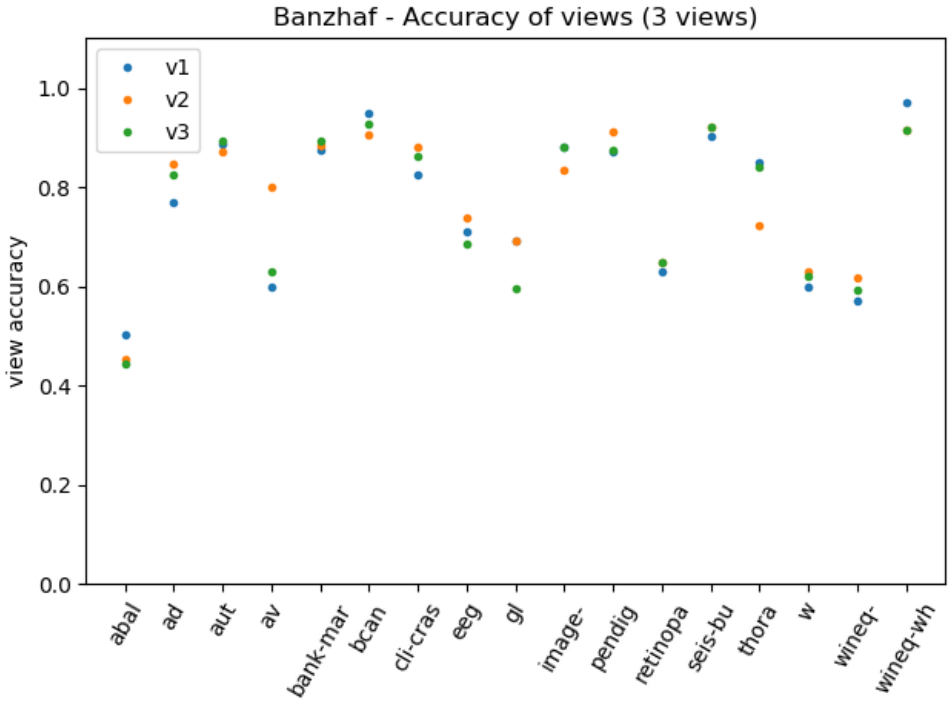

Quality of partition (2 views)
relative quality
1.0
0.8
0.6
0.4
0.2
0.0
bv
sv
abalone
adult
autism
avila
bank-market
bcancer
cli-crashes
eegeye
glass
image-seg
pendigits
retinopathy
seis-bumps
thoracic
wine
wineq-red
wineq-white

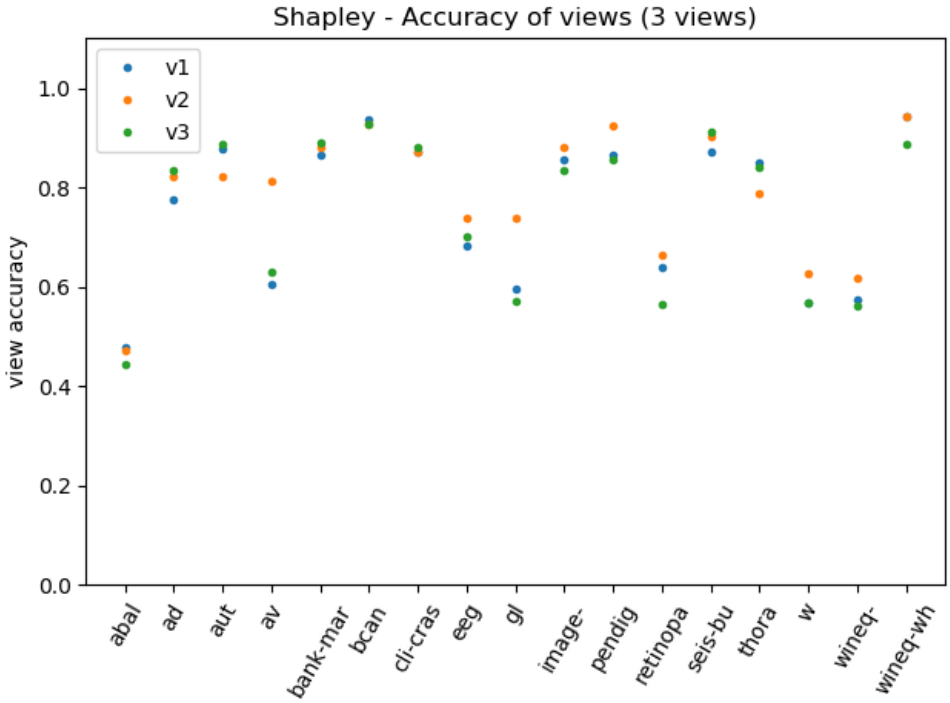

Quality of partition (3 views)
relative quality
1.0
0.8
0.6
0.4
0.2
0.0
bv
sv
abalone
adult
autism
avila
bank-market
bcancer
cli-crashes
eegeye
glass
image-seg
pendigits
retinopathy
seis-bumps
thoracic
wine
wineq-red
wineq-white

The following plots confirms that the views have very similar prediction's ability: each dot is the prediction's ability of a view for a selected dataset

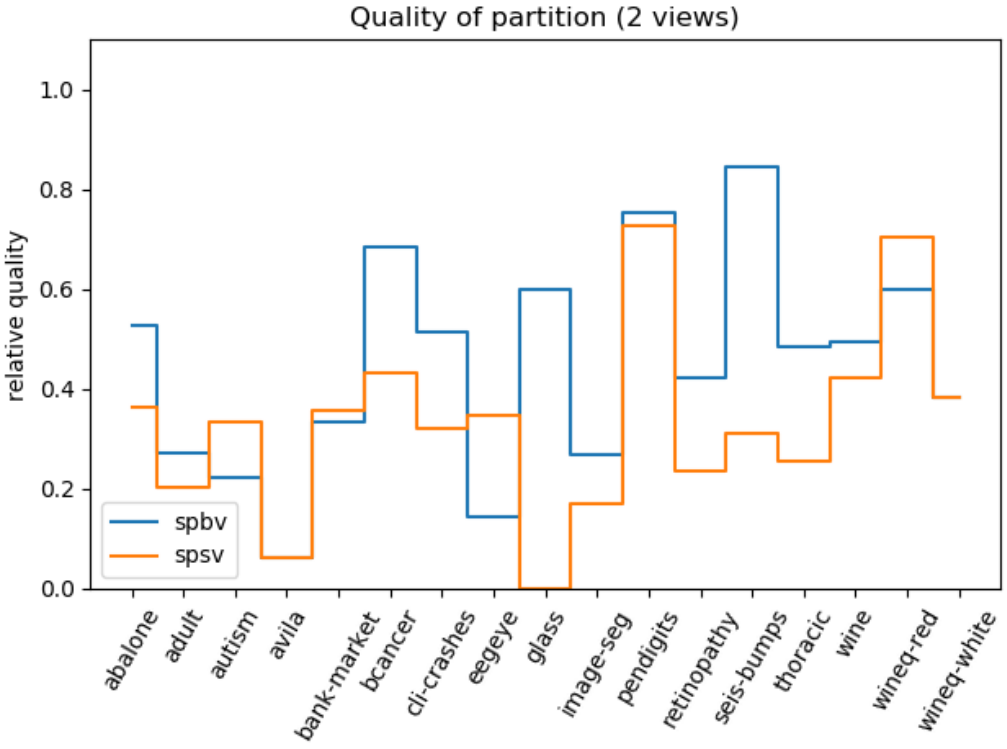


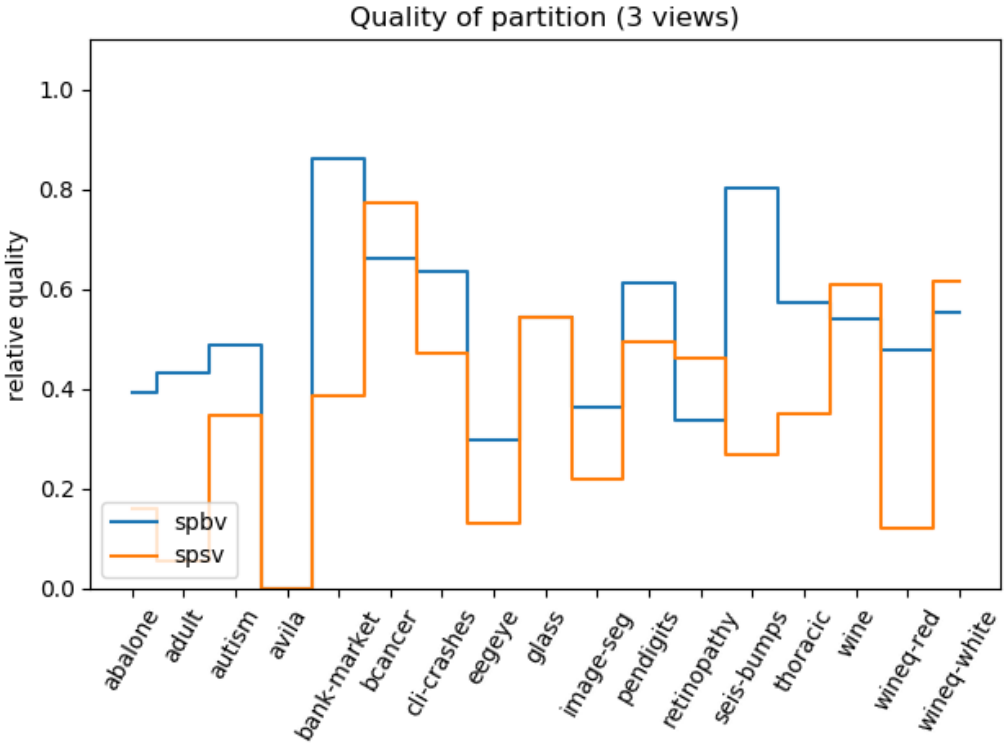

Shapley - Accuracy of views (2 views)
v1
v2
view accuracy
1.0
0.8
0.6
0.4
0.2
0.0
abal
ad
aut
av
bank-mar
bcan
cli-cras
eeg
gl
image-
pendig
retinopa
seis-bu
thora
w
wineq-
wineq-wh


Banzhaf - Accuracy of views (3 views)
v1
v2
v3
view accuracy
1.0
0.8
0.6
0.4
0.2
0.0
abal
ad
aut
av
bank-mar
bcan
cli-cras
eeg
gl
image-
pendig
retinopa
seis-bu
thora
w
wineq-
wineq-wh

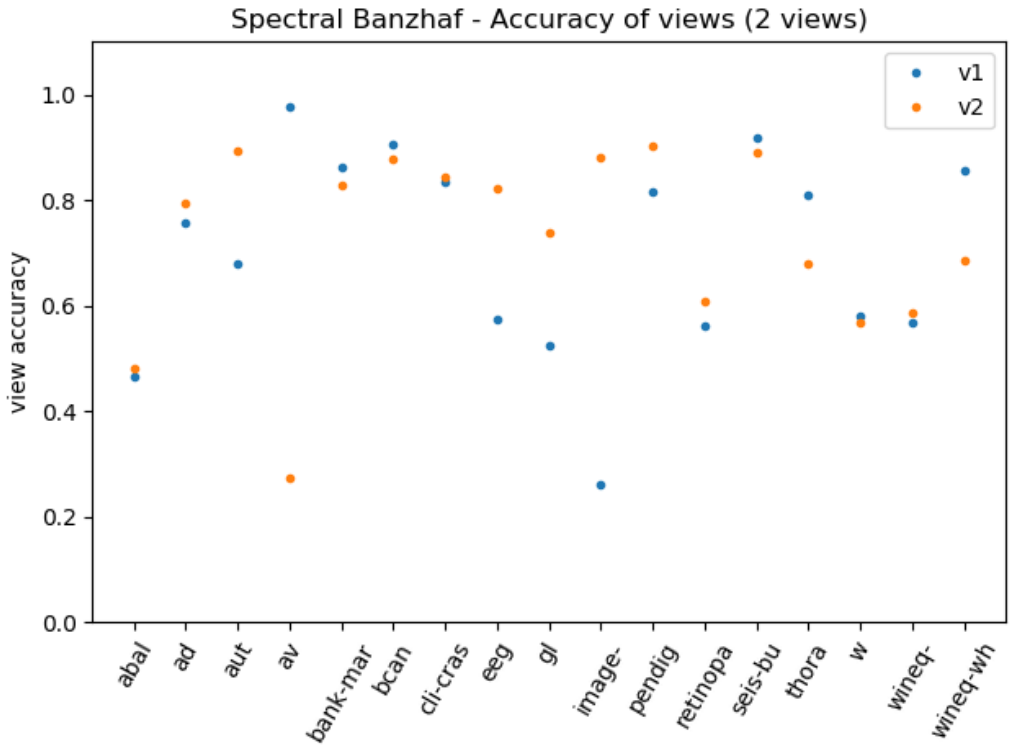


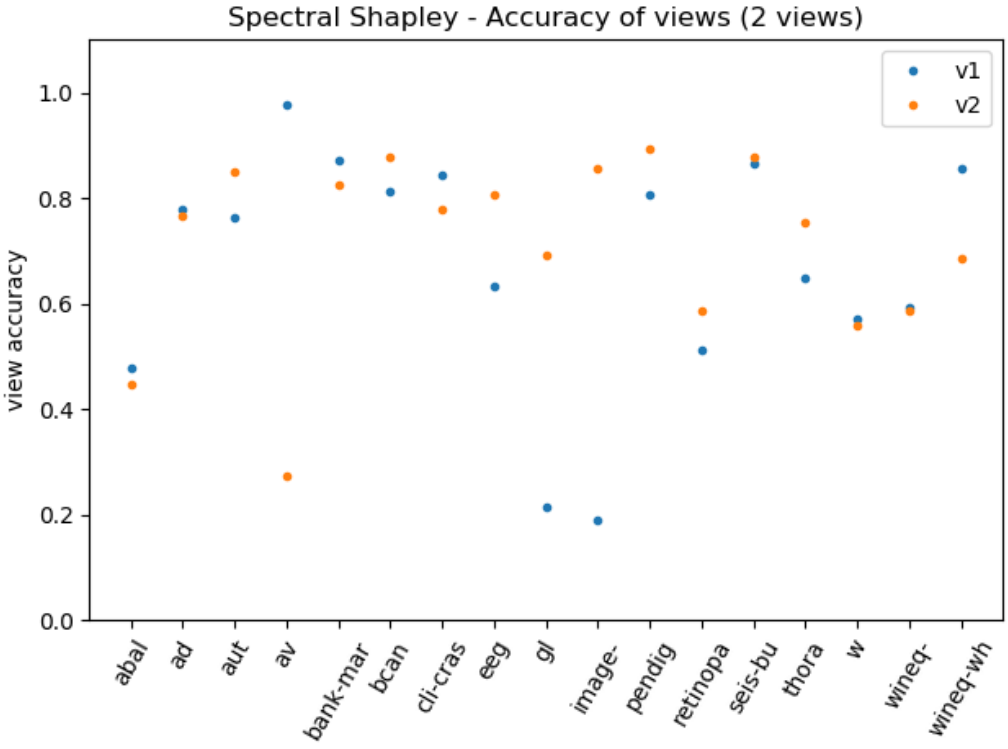
The spectral clustering approach is not good as the previous one. This is visible in the following plots

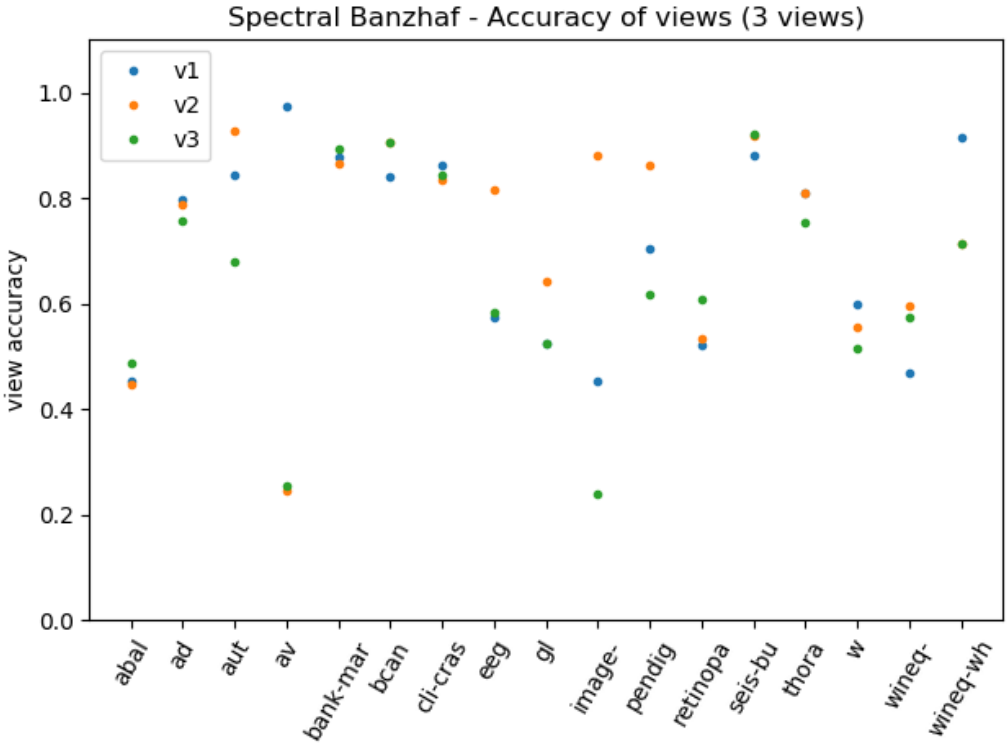

Quality of partition (2 views)
relative quality
1.0
0.8
0.6
0.4
0.2
0.0
spbv
spsv
abalone
adult
autism
avila
bank-market
bcancer
cli-crashes
eegeye
glass
image-seg
pendigits
retinopathy
seis-bumps
thoracic
wine
wineq-red
wineq-white

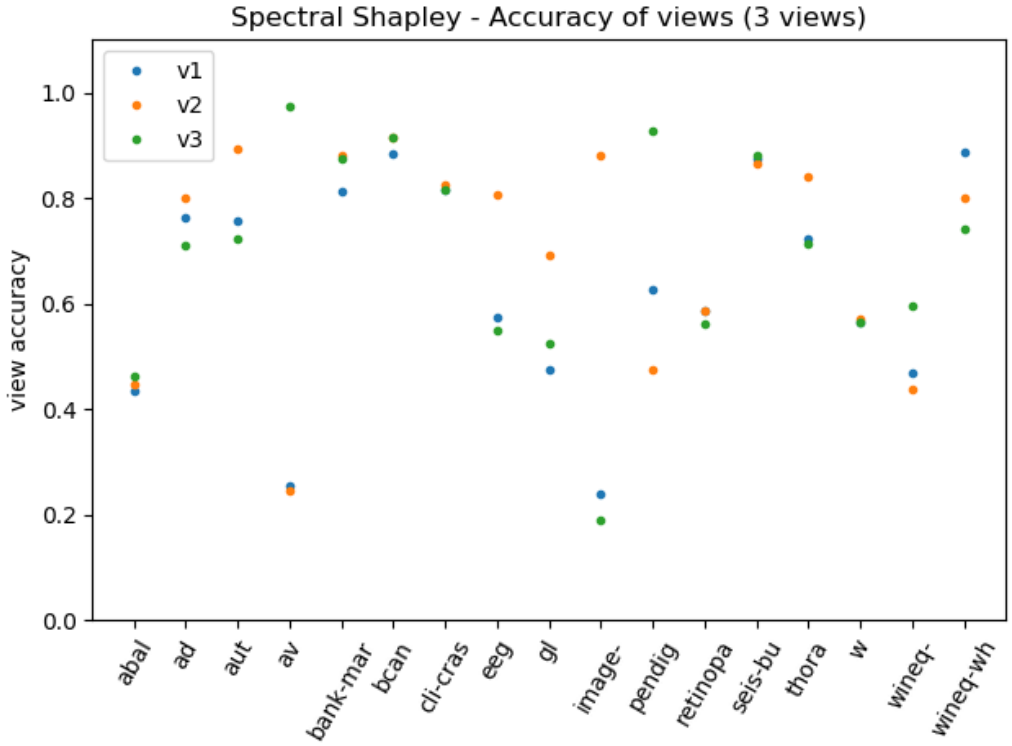

Quality of partition (3 views)
relative quality
1.0
0.8
0.6
0.4
0.2
0.0
spbv
spsv
abalone
adult
autism
avila
bank-market
bcancer
cli-crashes
eegeye
glass
image-seg
pendigits
retinopathy
seis-bumps
thoracic
wine
wineq-red
wineq-white

The partition found is very far from the optimal one. We have also another problem: very often the accuracy of the views is very different. This is visible in the following plots




Spectral Shapley - Accuracy of views (2 views)
view accuracy
1.0
0.8
0.6
0.4
0.2
0.0
v1
v2
abal
ad
aut
av
bank-mar
bcan
cli-cras
eeg
gl
image-
pendig
retinopa
seis-bu
thora
w
wineq-
wineq-wh


Spectral Banzhaf - Accuracy of views (3 views)
view accuracy
1.0
0.8
0.6
0.4
0.2
0.0
v1
v2
v3
abal
ad
aut
av
bank-mar
bcan
cli-cras
eeg
gl
image-
pendig
retinopa
seis-bu
thora
w
wineq-
wineq-wh



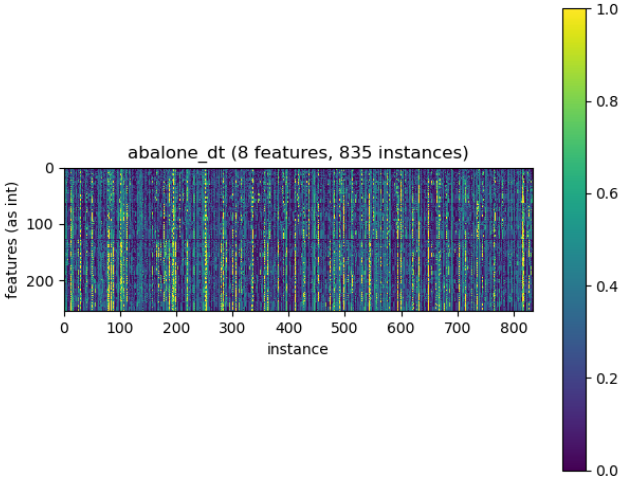
We can observe that the points are often not close to each other.

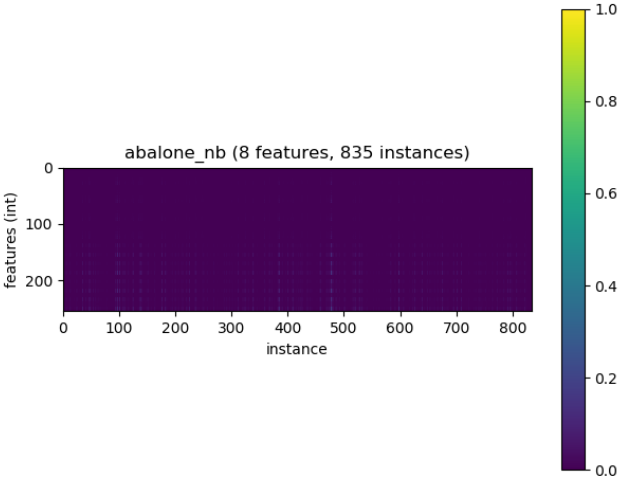
8.9 Generated predictions

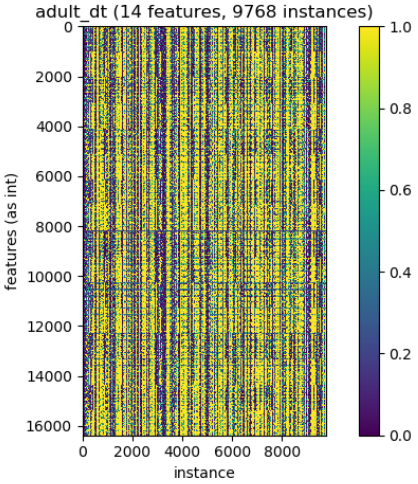
In the following images, each pixel is the *quality of prediction* for a instance (x axis) using a subset of features (y axis, feature's set represented as integer).

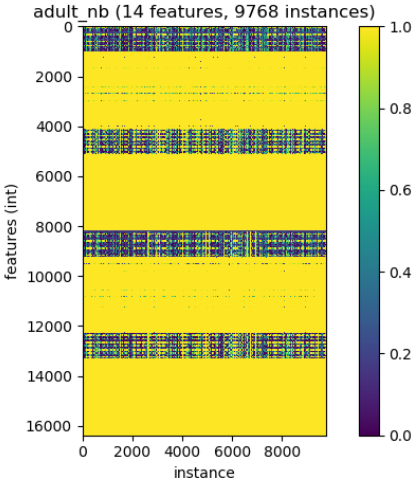
In the first column, we have used a *Decision Tree*, in the second column the *Naïve Bayes* method.

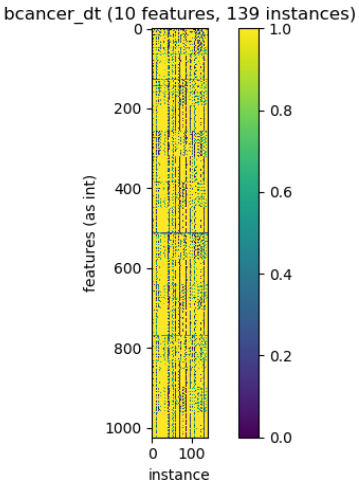
The visual effect of the increased prediction's quality, using more features, is a transition from dark magenta to yellow. Unfortunately, this transition is not visible in the following images, confirmed by the flat diagrams available in 8.2.

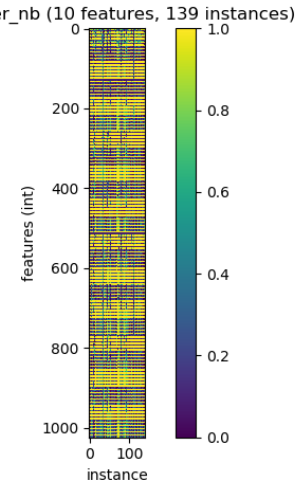

abalone_dt (8 features, 835 instances)
features (as int)
instance

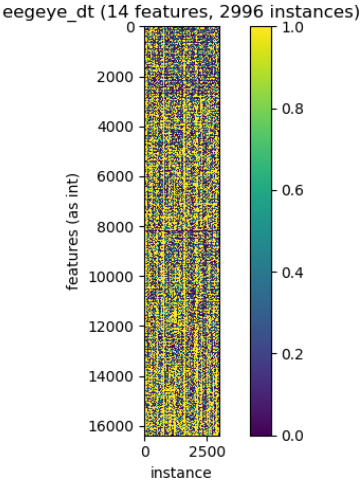

abalone_nb (8 features, 835 instances)
features (int)
instance

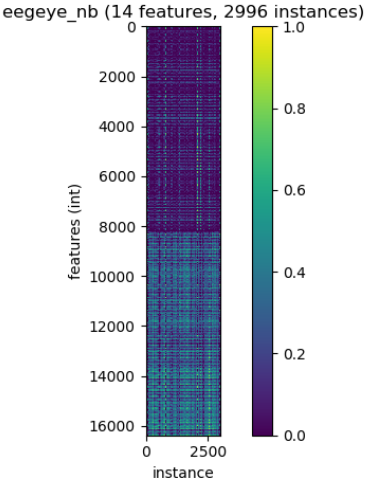

adult_dt (14 features, 9768 instances)
features (as int)
instance

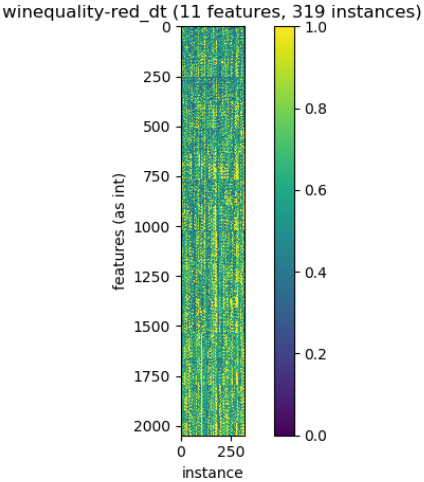

adult_nb (14 features, 9768 instances)
features (int)
instance

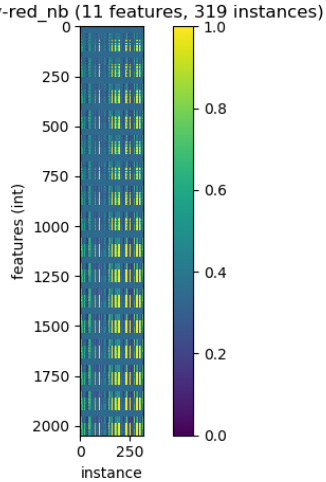

bcancer_dt (10 features, 139 instances)
features (as int)
instance

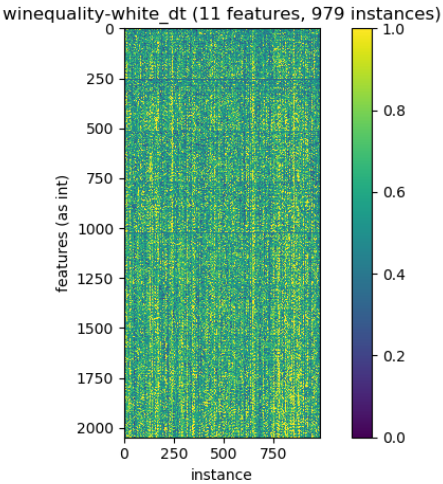

bcancer_nb (10 features, 139 instances)
features (int)
instance

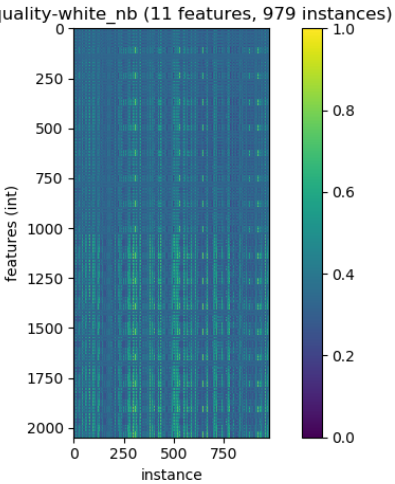

eegeye_dt (14 features, 2996 instances)
features (as int)
instance


eegeye_nb (14 features, 2996 instances)
features (int)
instance


winequality-red_dt (11 features, 319 instances)
features (as int)
instance


winequality-red_nb (11 features, 319 instances)
features (int)
instance


winequality-white_dt (11 features, 979 instances)
features (as int)
instance


winequality-white_nb (11 features, 979 instances)
features (int)
instance

8.10 Teaching ability

In section 8.8 we have seen that it is possible to split a dataset in two o more views with a good prediction's ability. A view is able to teach something to another view if in the first view there is a instance with good prediction and in the second view, the prediction, on the same instance, is very bad.

Unfortunately, with the used datasets, it is very difficult to find this condition. The following table shows a classical result obtained during the computation of the *teaching index*, and using a difference between the best and the worst prediction's quality of 0.1.

name	nf	ni	nt
abalone_dt	8	835	3
adult_dt	14	9768	1
avila_dt	10	4173	2
bank_market_dt	16	9039	1
bcance_dt	10	139	6
eegey_dt	14	2996	3
glass_dt	10	42	0
pendigits_dt	16	2198	7
thoracic_dt	16	94	1
winequality-red_dt	11	319	1
winequality-white_dt	11	979	4
wine_dt	13	35	7

Table 8.4: Available teaching instances

where

- `name`: name of the dataset
- `nf`: number of features
- `ni`: number of instances

- `nt`: number of usable teaching instances

The extremely small number of teaching instances (or its absence) prevents the possibility to evaluate the *teaching index* fore the features and the teaching ability of the views.

The main problem, obviously, is that the dataset contains features that are good or bad predictors on the same instances. To obtain a good teaching index it is necessary to collect features such that different features are good predictors on different instances.

8.11 When the CGT based methods are not useful

The computation of power indices is very expensive. This means that using the game theory to select the best players (or features) is useful only if the function has not *nice* properties. This occurs because, if the functions have *nice* properties, there exist more efficient algorithms to use.

For example, if the set function is *additive*, the worth of a coalition is

$$\xi(S) = \sum_{i \in S} = \xi(i)$$

In this case, Banzhaf Value is

$$\phi_\xi^B(i) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus i} \Delta_i \xi(S) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus i} \xi(i) = \frac{2^{n-1}}{2^{n-1}} \xi(i) = \xi(i)$$

and the order induces by the power index is the same induced by the value assigned to each player.

If the set function depends only on the coalition's cardinality ($\xi(S) = l(s)$), we have:

$$\phi_\xi^B(i) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus i} \Delta_i \xi(S) = \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} l(k)$$

and, because the Banzhaf Value doesn't depend on the coalition's members, all players have the same value.

If a player i interferes with all others, we have that $\Delta_i \xi(S)$ is negative for all coalitions, then the Banzhaf Value will be negative and, if we order the players (in decreasing way), i will be posted at the end of the list and it will be selected lastly.

If it is possible to assign some score $s(i)$ to each player $i \in N$ (here, the score is only a *fictional* numerical value useful to assign an order to the players),

and the function has the following property:

1. $\sum_{i \in S} s(i) \leq \sum_{i \in T} s(i) \rightarrow \xi(S) \leq \xi(T)$

it is not difficult to demonstrate that if $s(a) \leq s(b)$ we have $\phi_\xi^B(a) \leq \phi_\xi^B(b)$, that is, the order assigned by power indices is the same order based on the player's scores.

For example, we consider the coalition S that contains the players a and b with $s(a) < s(b)$, and the coalitions $S_a = S \setminus a$, $S_b = S \setminus b$:

$$\begin{aligned} \Delta_a \xi(S_a) &= \xi(S) - \xi(S_a) \\ \Delta_b \xi(S_b) &= \xi(S) - \xi(S_b) \\ \Delta_b \xi(S_b) - \Delta_a \xi(S_a) &= \xi(S) - \xi(S_b) - \xi(S) + \xi(S_a) = \xi(S_a) - \xi(S_b) \end{aligned}$$

but

$$\begin{aligned} \sum_{i \in S} s(i) &= \sum_{i \in S_a} s(i) + s(a) = \sum_{i \in S_b} s(i) + s(b) \\ \sum_{i \in S_b} s(i) - \sum_{i \in S_a} s(i) &= s(b) - s(a) \geq 0 \end{aligned}$$

this means that

$$\begin{aligned} \xi(S_b) &\geq \xi(S_a) \\ \xi(S_b) - \xi(S_a) &\geq 0 \\ \Delta_b \xi(S_b) - \Delta_a \xi(S_a) &\geq 0 \\ \Delta_b \xi(S_b) &\geq \Delta_a \xi(S_a) \end{aligned}$$

that is, if $s(a) \leq s(b)$ we have $\Delta_a \xi(S_a) \leq \Delta_b \xi(S_b)$ and, because the power indices are means of the first derivatives, we have $\phi_\xi(a) \leq \phi_\xi(b)$.

In a more general sense, if each local maximum is also global, there are more efficient algorithms to select the best elements.

8.12 When the CGT based methods are useful

The concepts used in the Coalitional Game Theory have a *little* problem: they are *weighted mean* of the marginal contributions of players, where the problem to solve is finding the coalition with the *maximum value*.

A possible approach, obviously, is to find explicitly this maximum, using one of several *heuristic optimization algorithms* available.

We consider the lattice

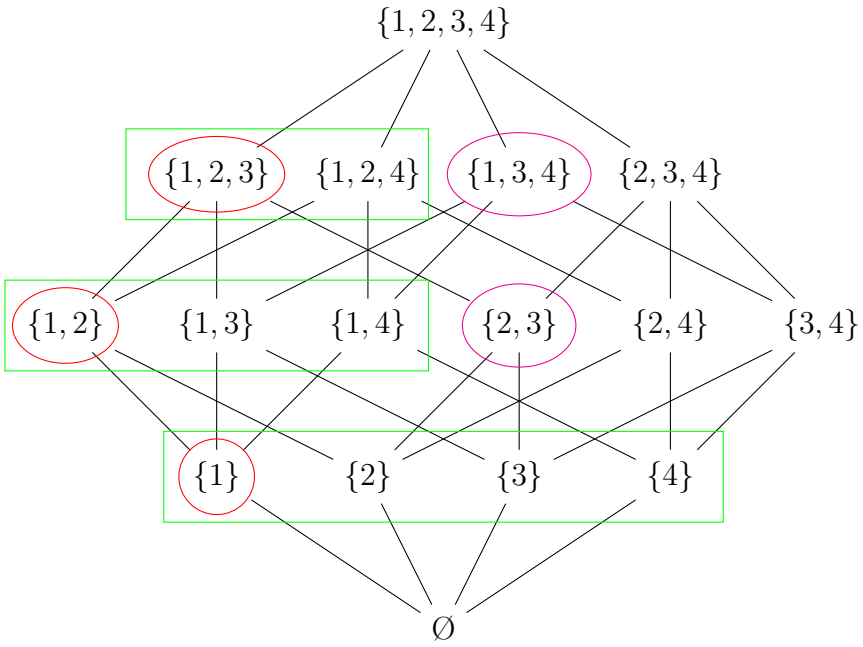


Figure 8.1: Subset relation

Suppose using the **Greedy Forward Selection** algorithm to select the best set of players with cardinality 1, 2 and 3. The algorithm starts from the empty set and check the function value on the sets with cardinality 1 ($\{1\}, \ldots, \{4\}$). We suppose that the maximum value is in $\{1\}$.

The next step is to select the best set, with cardinality 2, containing 1. To do this, the algorithm analyzes *only* the sets $\{1,2\}, \{1,3\}, \{1,4\}$, excluding the other sets ($\{2,3\}, \{2,4\}, \{3,4\}$). For example, the best set is $\{1,2\}$.

The last step is to select the best set with cardinality 3 containing the elements $\{1,2\}$, analyzing the sets $\{1,2,3\},\{1,2,4\}$. For example, the best set is $\{1,2,3\}$.

The problem is that the set $\{1\}$ is the only *global maximum*, the other sets, $(\{1,2\},\{1,2,3\})$ are *local maximum* and not *global maximum* that can be placed, for example, in $\{2,4\}$ and $\{1,3,4\}$.

Now, we suppose to use the Banzhaf Value to order and to select the players for the best set with cardinality 2. In this case, the order required must be

$$\phi_\xi^B(2),\phi_\xi^B(3) > \phi_\xi^B(1)$$

We consider the difference between two Banzhaf Values

$$\begin{aligned}
\phi_\xi^B(i)-\phi_\xi^B(j) &= \sum_{S\subseteq N\setminus i}\Delta_i\xi(S)-\sum_{S\subseteq N\setminus j}\Delta_j\xi(S)\\
&=\left(\sum_{S\subseteq N\setminus i}\xi(S\cup i)-\sum_{S\subseteq N\setminus i}\xi(S)\right)-\left(\sum_{S\subseteq N\setminus j}\xi(S\cup j)-\sum_{S\subseteq N\setminus j}\xi(S)\right)\\
&=\left(\sum_{S\subseteq N\setminus ij}\xi(S\cup ij)-\sum_{S\subseteq N\setminus ij}\xi(S\cup j)\right)-\left(\sum_{S\subseteq N\setminus ij}\xi(S\cup ij)-\sum_{S\subseteq N\setminus ij}\xi(S\cup i)\right)\\
&=\sum_{S\subseteq N\setminus ij}\xi(S\cup i)-\sum_{S\subseteq N\setminus ij}\xi(S\cup j)\\
&=\Big(\xi(i)-\xi(j)\Big)+\left(\sum_{S\in F^+_{-ij}}\xi(S\cup i)-\sum_{S\in F^+_{-ij}}\xi(S\cup j)\right)\\
&=D_1(ij)+D_+(ij)
\end{aligned}$$

with

$$\begin{aligned}
D_1(ij) &= \Big(\xi(i)-\xi(j)\Big)\\
D_+(ij) &= \left(\sum_{S\in F^+_{-ij}}\xi(S\cup i)-\sum_{S\in F^+_{-ij}}\xi(S\cup j)\right)
\end{aligned}$$

The greedy algorithm considers only the difference $D_1(ij)$: if the difference is greater than 0, the algorithm select i otherwise j. The selection based on the Banzhaf Value considers both terms (D_1 and D_+) and this sum can have a different sign.

The last observation is this: because a power index is the *mean* of the marginal contributions of a player, if the player *collaborates* very well with the other players *quite* often, and *interferes quite* rarely, its marginal contributions will be high and the related power index will be high. Conversely, if the player *interferes quite* often, and *collaborates quite* rarely, its marginal contributions will be low and the related power index will be low.

Chapter 9

Conclusions

9.1 Current work

In Machine Learning context, one of the first problems to resolve is to select the most useful set of features to use in the teaching phase of the algorithms. There exists several approaches available, and one of them is based on the Coalitional Game Theory. The theory offer two concepts that, when applied to the Machine Learning, can be described as follow: the power index, a measure of the feature's contribution in the correct prediction, and the interaction index, a measure of the collaboration or interference of two (or more) features during the prediction.

A power index is a weighted mean of the features' marginal contributions, where the marginal contribution is the difference between the correctness of the predictions when the features is in the set and when it is not present. But here there is a problem: we are searching the set of features with the highest correctness of the predictions, and to do this, we use a concept based on a mean. Why is it possible to use this method?

An interaction index is a weighted mean of the differences between marginal contributions when both are present in the set and when only one of them or none is present. This value is able to measure the collaboration and the interference on the prediction. Obviously, we prefer to keep together the features that collaborate and to keep separate the features that interfere.

In the Multi View Context, based on a single view, we are interested to

split the features in two o more views: for this work, the power index can be used to create views with a good prediction's correctness, and the interaction index can be used to decide if it is better to keep the features together or separated.

In the Co-training, based on a single view, we are interested not only to split the features in two o more views, but also in the ability of a view to teach something to the other ones. Here, the problem is how to measure this ability and how to consider all requirements.

This dissertation has answer to these questions.

After the standard notations defined in Chapter 2, in Chapter 3 we have introduced the concepts of power and interaction indices. In this chapter, we have seen that the popular Shapley Value and Banzhaf Value are example of the most general class of *probabilistic power indices*. This class is too general to be useful in practice: it is necessary to configure 2^n parameters and this can be very difficult. There are two sub classes that need only n parameters. The first one is the *cardinal-probabilistic values* class. It uses a probability distribution for each subset cardinality. Using the uniform distribution, we obtain the Shapley Value. The second one is the *player-probabilistic values* class, that assigns a collaboration's probability to each player. If the probability is $\frac{1}{2}$ for each player, we obtain the Banzhaf Value. These classes extend the list of power indices that be used with real problems.

To compute the exact value of these indices it is necessary to consider all 2^n subsets. This is possible only for n very small. In general, it is possible to compute only a approximated value, sampling the subsets' lattice. In the feature partitioning, the index values are used to order the features so that the most important are in the lead. We have also observed that the exact order is not so important. Indeed, we are interested only to separate some best features from the others. In this case, the order inside each group is not important. In Chapter 4 we have described the structure of these algorithms, and their implementation. We have also described some algorithms, variants as, for example, an alternative implementation for the Shapley Value, based on permutations, or the algorithm based on the usage of best sets.

In Chapter 5 we have demonstrate that the power indices are the *best linear approximation* of the set function, based on a *weighted mean square error*. Using different weights, we obtain different indices. We have seen also that it is possible to define an index based on a set of axioms that it must satisfy.

Using different axioms, we obtain different indices. Because the power indices are an approximation of the set function, this permits us to use the index values to approximate the set with highest function value: we can use the features with the highest index values.

During the feature selection phase, we must select a predefined number of features. In Chapter 5 we have seen that the indices are the best approximation of the *entire* function. This is not necessary: we must approximate a reduced version of the function, defined only on sets with the selected cardinality. In Chapter 6 we have defined a new class of indices, based on this reduced definition: the terms used in the original one are grouped according to the set's cardinality. The reduced definition uses only the terms for the selected cardinality.

Why can we approximate the set of features with the highest function value, with the set composed by the features with the best power index values? This is based on the *conjecture* that, if a feature is in the best set, it is a feature that collaborate very often, and its marginal contribution is very high. Since the power index is a (weighted) mean of marginal contributions, a feature with a contribution very high has also a index value very high.

A feature partitioning problem consists to split the features into two o more views such as each view has enough data to train a good predictor. The problem to resolve is how to distribute the features. We have added also some other constraint: we would like to have collaborating features in the same view and interfering features in different views. These requirements are satisfied by the power and interaction indices. In Chapter 7 we have used these indices to define an optimization problem where the solution is the required split. In the context of Co-training based on a single view, the feature partitioning is only the first part of the problem. The second part is to find views such as each view is able to teach something to each of the others. Another problem is how to define the *ability of teaching*. In that chapter we have used an approach similar to the accuracy used in the classification algorithms: we *count* the number of times that the prediction in a view can be transferred to another one. Using this definition, we have extended the splitting optimization problem to consider also the ability of teaching.

9.2 Future works

There are several aspects of Game Theory that require further study. For example it is not clear when to use a certain power index or another. Or if it is better to use an index where the user must select the parameters. And in this case, which values to assign to the parameters. Or if it is necessary to *find* the parameter's values, resolving an optimization problem for the best approximation of the set function. The answer to this question must be based on Machine Learning concepts, or metrics based on the data and the problem to solve, and not on Game Theory concepts. This to allow Machine Learning experts to use this approach without being experts on game theory.

Another problem to resolve is how to decide when an approach based on the Game Theory is better than one based on greedy methods. In Chapter 8 we have seen two simple cases, but it is necessary to analyze in a more extended way the space of all set functions. To generate samples of this space using real datasets and real Machine Learning algorithms is impracticable: there are available only a limited number of dataset and a limited number of algorithms. The solution is to create a good *random set function generator* and to use real set functions as *examples* of functions to generate. Using this generator, we can sample the functions space and to find the sub-spaces where an approach is better than another. We can start with small dimensions spaces, and extend the search into larger spaces.

How to define this generator is another problem. An approach is to synthesize the function based on concepts as capacity of prediction of each feature, collaboration and interference between features. Another approach is to use the function transforms, for example the Möbius transform, to use a distribution for each transform's coefficient, to generate the value for each coefficient, and to recreate the function.

To model the feature partitioning with teaching as a graph introduces another problem: in the graph there are two type of edges, one undirected and the other directed. The directed edges must be considered only in parallel with the cut-edges. The partition must minimize the cut-edges weights and maximize the teaching edges weights.

But this is another story.

Appendix A

Proofs

A.1 Dirac basis

The Dirac basis is defined as

$$\delta_S(T) = \prod_{i \in S} T_i \prod_{i \in S \setminus N} (1 - T_i)$$

If $i \in S$ but $i \notin T$, $T_i = 0$ and the first product is 0. If $i \notin S$ but $i \in T$, $(1 - T_i) = 0$ and the second product is 0.

Then $\delta_S(T) = 1$ iff $S = T$.

The basis is *orthonormal*:

$$\langle \delta_T, \delta_T \rangle = \sum_{S \subseteq N} \delta_T(S)\delta_T(S) = \delta_T(T)\delta_T(T) = 1$$

$$\langle \delta_T, \delta_U \rangle = \sum_{S \subseteq N} \delta_T(S)\delta_U(S) = \delta_T(T)\delta_U(T) + \delta_T(U)\delta_U(U) = 0 \quad \forall T \neq U$$

A.2 Unanimity Game basis

The Unanimity Game basis is defined as

$$e_S(T) = \prod_{i \in S} T_i$$

The product is 1 only if $T_i = 1$ for all $i \in S$, then $T \supseteq S$.

The basis **is not** orthonormal:

$$\begin{aligned}
\langle e_T, e_T \rangle &= \sum_{S \subseteq N} e_T(S) e_T(S) \\
&= \sum_{S \supseteq T} e_T(S) \\
&= \sum_{S \subseteq N \setminus T} e_T(S \cup T) \\
&= 2^{n-t}
\end{aligned}$$

$$\begin{aligned}
\langle e_T, e_U \rangle &= \sum_{S \subseteq N} e_T(S) e_U(S) \\
&= \sum_{S \supseteq T \cup U} e_{T \cup U}(S) \\
&= \sum_{S \subseteq N \setminus T \cup U} e_{T \cup U}(S \cup T \cup U) \\
&= 2^{n-|T \cup U|}
\end{aligned}$$

A.3 Walsh Function basis

Let

$$z_i = 2x_i - 1$$

and

$$S_i = \begin{cases} 1, & \text{if } i \in S \\ -1, & \text{otherwise} \end{cases}$$

The Walsh function is defined as

$$w_S(\mathbf{z}) = \prod_{i \in S} z_i$$

with the convention

$$w_\emptyset(\mathbf{z}) = \prod_{i \in \emptyset} z_i = 1$$

If we consider the *weighted dot product* with $\mu(S) = \frac{1}{2^n}$, the basis is orthonormal

$$\begin{aligned} \langle w_S, w_S \rangle_\mu &= \sum_{T \subseteq N} \frac{1}{2^n} w_S(T) w_S(T) \\ &= \frac{1}{2^n} \sum_{T \subseteq N} \prod_{i \in S} z_i^2 \\ &= \frac{1}{2^n} \sum_{T \subseteq N} 1 = \frac{1}{2^n} 2^n = 1 \end{aligned}$$

If $S \neq T$ the dot product is

$$\langle w_S, w_T \rangle_\mu = \sum_{U \subseteq N} w_{S \Delta T}(U)$$

but $S \Delta T$ can be any set. If $S \Delta T = \emptyset$ we have $S = T$ or S and T are $\emptyset$ and the value is 1 as specified previously. Otherwise $S \neq \emptyset$ and there exists an $i \in S$. Then, we have

$$\sum_{T \subseteq N} w_S(T) = \sum_{T \subseteq N \setminus i} (-1) w_{S \setminus i}(T) + \sum_{T \subseteq N \setminus i} (+1) w_{S \setminus i}(T) = 0$$

That is, if $S \neq T$, we have $S \Delta T \neq \emptyset$ and

$$\langle w_S, w_T \rangle_\mu = 0$$

A.4 The Möbius transform

We consider the function ξ definition based on the Dirac basis

$$\xi(A) = \sum_{S \subseteq N} \xi(S) \delta_S(A)$$

The definition can be changed in

$$\begin{aligned}
\xi(A) &= \sum_{S \subseteq N} \xi(S) \delta_S(A) \\
&= \sum_{S \subseteq N} \xi(S) \prod_{i \in S} A_i \prod_{i \in N \setminus S} (1 - A_i) \\
&= \sum_{S \subseteq N} \xi(S) \prod_{i \in S} A_i \left(\sum_{T \subseteq N \setminus S} \prod_{i \in T} 1 \prod_{i \in N \setminus S \setminus T} (-A_i) \right) \\
&= \sum_{S \subseteq N} \left(\sum_{T \subseteq N \setminus S} (-1)^{n-s-t} \xi(S) \prod_{i \in N \setminus S \setminus T} A_i \right) \prod_{i \in S} A_i
\end{aligned}$$

The expression $\prod_{i \in S} A_i = e_S(A) = 1$ for each $A \supseteq S$

$$\xi(A) = \sum_{S \supseteq A} \left(\sum_{T \subseteq N \setminus S} (-1)^{n-s-t} \xi(S) \prod_{i \in N \setminus S \setminus T} A_i \right)$$

The expression $\prod_{i \in N \setminus S \setminus T} A_i = 1$ for each $N \setminus S \setminus T \subseteq A$. The definition can be rewritten as

$$\xi(A) = \sum_{S \subseteq A} \left(\sum_{T \subseteq S} (-1)^{s-t} \xi(T) \right) e_S(A)$$

The set function

$$m_\xi(S) = \sum_{T \subseteq S} (-1)^{s-t} \xi(T)$$

is the *Möbius transform* of ξ. Based on the Möbius transform, the set function can be written as

$$\xi(A) = \sum_{S \subseteq A} m_\xi(S) e_S(A)$$

A.5 From function to derivative

The *probabilistic value* [81] is the generalization of Banzhaf Value and Shapley Value.

A *probabilistic value* is defined as:

$$\phi_\xi(i) = \sum_{S \subseteq N} p_i(S) \cdot \xi(S) \tag{A.1}$$

We will demonstrate that it is possible to convert this definition into the more classic definition

$$\begin{aligned} \phi_\xi(i) &= \sum_{S \subseteq N \setminus i} p_i(S) \cdot (\xi(S \cup i) - \xi(S)) \\ &= \sum_{S \subseteq N \setminus i} p_i(S) \cdot \Delta_i \xi(S) \end{aligned} \tag{A.2}$$

using the *dummy value* axiom.

To do this, we consider the following equalities:

$$\begin{aligned}\phi_\xi(i) &= \sum_{S\subseteq N} p_i(S)\cdot\xi(S) \\ &= \sum_{S\supseteq i} p_i(S)\cdot\xi(S) + \sum_{S\subseteq N\setminus i} p_i(S)\cdot\xi(S) \\ &= \sum_{S\subseteq N\setminus i} (p_i(S\cup i)\xi(S\cup i) + p_i(S)\cdot\xi(S))\end{aligned}$$

The last equality can be converted into the equation A.2 if $p_i(S\cup i) = -p_i(S)$. To demonstrate this step, we can consider the *Unanimity Game basis* functions e_T with $T\subseteq N\setminus i$

$$e_T(S) = \begin{cases} 1, & \text{if } S\supseteq T \\ 0, & \text{otherwise}\end{cases}$$

For these functions, i is a *dummy player*. Indeed, for $S\subset T$, we have that $S\cup i\not\subset T$, and the marginal contribution is

$$e_T(S\cup i) - e_T(S) = 0 - 0 = 0$$

for $S\supseteq T$, we have $S\cup i\supset T$, and the marginal contribution is

$$e_T(S\cup i) - e_T(S) = 1 - 1 = 0$$

Now, we consider the power index $\phi_{e_T}(i)$ defined for each e_T. Because i is a dummy player, we have

$$\begin{aligned}\phi_{e_T}(i) &= \sum_{S\subseteq N} p_i(S)\cdot e_T(S) \\ &= \sum_{S\subseteq N\setminus i} (p_i(S\cup i)\cdot e_T(S\cup i) + p_i(S)\cdot e_T(S)) \\ &= p_i(N)\cdot e_T(N) + p_i(N\setminus i)\cdot e_T(N\setminus i) \\ &= p_i(N) + p_i(N\setminus i) = 0\end{aligned} \tag{A.3}$$

For inductive purpose, we suppose that A.3 is valid for each T with cardinality $t > k$. The next step is to consider T with cardinality k:

$$\begin{aligned}
\phi_{e_T}(i) &= \sum_{S \subseteq N} p_i(S) \cdot e_T(S) \\
&= \sum_{S \in [T, N \setminus i]} (p_i(S \cup i) \cdot e_T(S \cup i) + p_i(S) \cdot e_T(S)) \\
&= (p_i(T \cup i) \cdot e_T(T \cup i) + p_i(T) \cdot e_T(T)) \\
&\quad + \sum_{S \in (T, N \setminus i]} (p_i(S \cup i) \cdot e_T(S \cup i) + p_i(S) \cdot e_T(S)) \\
&= p_i(T \cup i) + p_i(T) = 0
\end{aligned}$$

With this set of equalities we have a list of relations between the values of $p_i(T)$, that is

$$p_i(S \cup i) = -p_i(S) \quad \forall S \subseteq N \setminus i$$

Now, we can rewrite $\phi_\xi(i)$ as

$$\begin{aligned}
\phi_\xi(i) &= \sum_{S \subseteq N} -p_i(S) \cdot \xi(S) \\
&= \sum_{S \subseteq N \setminus i} (p_i(S \cup i) \cdot \xi(S \cup i) + -p_i(S) \cdot \xi(S)) \\
&= \sum_{S \subseteq N \setminus i} p_i(S \cup i) \left(\xi(S \cup i) - \xi(S)\right) \\
&= \sum_{S \subseteq N \setminus i} p_i(S) \cdot \Delta_i \xi(S)
\end{aligned}$$

with $p_S^i = p_i(S \cup i)$ and

$$\sum_{S \subseteq N \setminus i} p_i(S) = 1$$

Appendix B

K-means Clustering in Dual Space for Unsupervised Feature Partitioning in Multi-view Learning

B.1 Authors

Corrado Mio, Gabriele Gianini and Ernesto Damiani

EBTIC, Khalifa University of Science and Technology, Abu Dhabi, UAE
and Dipartimento di Informatica, Università degli Studi di Milano, Italy

Email: {firstname.lastname}@unimi.it

B.2 Published in

2018 - 14th International Conference on Signal-Image Technology &
Internet-Based Systems (SITIS)

B.3 Abstract

In contrast to single-view learning, multi-view learning trains simultaneously distinct algorithms on disjoint subsets of features (the views), and jointly optimizes them, so that they come to a consensus. Multi-view learning is typically used when the data are described by a large number of features. It aims at exploiting the different statistical properties of distinct views. A task to be performed before multi-view learning – in the case where the features have no natural groupings – is multi-view generation (MVG): it consists in partitioning the feature set in subsets (views) characterized by some desired properties. Given a dataset, in the form of a table with a large number of columns, the desired solution of the MVG problem is a partition of the columns that optimizes an objective function, encoding typical requirements. If the class labels are available, one wants to minimize the inter-view redundancy in target prediction and maximize consistency. If the class labels are not available, one wants simply to minimize inter-view redundancy (minimize the information each view has about the others). In this work, we approach the MVG problem in the latter, unsupervised, setting. Our approach is based on the transposition of the data table: the original instance rows are mapped into columns (the *pseudo-features*), while the original feature columns become rows (the *pseudo-instances*). The latter can then be partitioned by any suitable standard instance-partitioning algorithm: the resulting groups can be considered as groups of the original features, i.e. views, solution of the MVG problem. We demonstrate the approach using k-means and the standard benchmark MNIST dataset of handwritten digits.

B.4 Introduction

In several data analytic applications, data about each training example are gathered from diverse domains or obtained from various feature extractors and exhibit heterogeneous statistical properties. For instance in IoT environments, data are collected by many distinct devices, at the periphery, so that their feature-sets can be naturally endowed with a faceted structure [207]. Also in the web-data mining domain the intrinsic attributes of a page, describing its textual content, those that describe its multimedia content and the extrinsic attributes representing meta-data are endowed with very different and specific statistical properties. In those and in other cases, the features of each example can be naturally partitioned into groups: each feature group

is referred to as a particular *view*.

Most conventional machine learning algorithms concatenate all views into a single view, subsequently provided in input to the learning algorithms (*single-view learning*). In contrast to this approach, *multi-view learning* (MVL) uses a distinct learning model for each view, with the goal of better exploiting the diverse information of the distinct views. The different variants of MVL try to jointly optimize all the learning models, so that they come to a consensus [197, 200]. Given a multi-view description of a phenomenon, one can apply both supervised or semi-supervised learning (e.g. multi-view classification or regression [197,198,200,217,218]) and unsupervised learning (e.g. multi-view clustering [192, 193]).

B.4.1 Motivations and problem

Sometimes, the features do not hint at a natural partitioning. In this case, the first task to be performed in MVL is the one known as multi-view generation (MVG): it consists in partitioning the feature-set in subsets (each representing a view) characterized by some desired properties and relationships. For instance, among the requirements of this problem is that the *inter-view* redundancy is minimal. There are at least two forms in which the problem can be found: the *supervised* setting and the *unsupervised* setting.

In the first setting, the class labels are available: in this case one wants to minimize the inter-view redundancy in target prediction (maximize uniqueness of information about the target from each view).

If the second, unsupervised setting, class labels are not available. This occurs for example when the labels do not actually exist: this is the case for instance of multi-view clustering [193] or other multi-view unsupervised tasks. This situation can take place also in multi-view supervised or semi supervised tasks, when the labels are determined at a later time. The case applies also to deep multi-view representation learning: there one has access to multiple unlabeled views of the data for representation learning. The setting applies as well to the case of long data analytic pipelines, where at the early stages of analysis it is not known what are the detailed learning tasks for which the data will be used.

In the unsupervised MVG task, one aims at achieving minimal inter-view redundancy (minimize the information each view has about the others). In

this work we approach the MVG problem in the latter, unsupervised, setting.

B.4.2 General approach

Given a dataset, in the form of a table, the desired solution of the unsupervised MVG problem is thus a partition of the columns that optimizes some *least-redundancy* requirements.

Hereafter, we will refer to the following document-word-count example for the illustration of the method. Consider a corpus of documents, such as a literary corpus or a corpus of web pages (for now we disregard the hyper-links and the multimedia content and focus on text only). Each document, under the bag-of-words representation (that disregards the structure of the text [208–212]), can be represented by the count of the occurrences of each word of a reference dictionary. This representation can take the form of a table, where each document corresponds to a row and each word to a column (we call it the `[row=document,column=word]` representation): each table-cell contains the *count* of the number of occurrences of a word into a document. In this formulation the words play the role of *features* while the documents play the role of *instances*.

Suppose that we intend are to apply multi-view learning: each view would correspond to a subset of words. Unfortunately, in our example, a natural partition of the words into views is not available. Thus, before running multi-view learning, we need to perform multi-view generation. We assume that no labels are available for the documents: our problem corresponds to the *unsupervised MVG problem*. We aim at *partitioning the words into groups* that optimize the least inter-view redundancy requirements, without any reference to labels, but based only on the relative properties of the views. A solution to this problem takes the form of a partition of the feature-set (a partition of the columns).

Our approach to the unsupervised MVG problem consists a *dual-space method*, based on the transposition of the data table. The original instance rows are mapped into columns, that we call *pseudo-features*, while the original features columns become rows, that we call *pseudo-instances* (i.e. we passe to a `[row=word, column=document]` representation). After transposition, a solution of the MVG problem takes the form of a partition of the pseudo-instances.

The key idea of our approach is the following: consider the pseudo-instances (the rows after transposition, which are the instances of a different problem, the dual problem), to those rows one can apply a standard instance-partitioning algorithms. Once the partition of the rows is obtained, one can transpose the solution back into the original form, and get the multi-view partition of the original features.

For the sake of simplicity, we study the approach using the partitional clustering algorithm k-means, however any partitional clustering algorithm could be used to the purpose. We also chose, for demonstrative purposes, to limit ourselves to the most straightforward case of *numerical-only data*: the case of partially of fully categorical data could in principle be dealt with, by using suitable categorical to numerical encodings (such as the one-hot encoding). We validate the approach using the MNIST handwritten digits dataset.

Organization of the paper. The reminder of the paper is organized as follows. In the next section (Section B.5) we provide an overview of the method, then (Section B.6) we give a formal definition of the problem and of the approach. Subsequently (Section B.7), we show the results obtained from the benchmark dataset and provide a partial validation (Section B.8). The discussion of the outcomes concludes the paper (Section B.9).

B.5 Overview of the method and issues

Let us refer to our illustrative document-word data table example, with count values, i.e. *numerical* values in the table-cells. The original data table has the form `[row=document, column=word]`. Our method consists in taking the transpose of the data table, i.e. passing to a `[row=word, column=document]` representation: now the words (formerly acting as features) take the role of objects and are called *pseudo-instances*, while the documents (formerly acting as instances) take the role of attributes and are called *pseudo-features*.

We can apply k-means to the pseudo-instances to obtain a partitioning of the words. In the algorithm, the distances between two words, i.e. two points (pseudo-instances), are computed in the document space: the space in which each dimension corresponds to a document. Two words that have a similar (percentage) count in the same document are close along that (document) dimension. The k-means algorithm outputs k clusters of pseudo-instances.

At this point, one can transpose back the partition of the pseudo-instances and get a multi-view partition of the original features. The relation of this method with simple word clustering based on documents or with the co-clustering approach is developed in the Discussion, Conclusion and Outlook section.

B.5.1 Issues

The main issue, after the first transposition, is that, if the original dataset is large, the number of pseudo-features makes the problem very high-dimensional, and the clustering algorithm potentially less effective. E.g. in a large corpus, consisting of many documents, the transposed matrix has a very large number of columns.

We address this issue as follows:

i) we break the whole set of *pseudo-features* into r smaller disjoint subsets (in the original space they represented object batches);

ii) we run a distinct pseudo-instance clustering on each of the r pseudo-feature subset, so that each clustering yields its own partition; we are left with r partitions;

iii) we aggregate the r cluster partitions to produce an individual partition solution.

With respect to points i) and ii), the operation of breaking down the columns should be made by choosing at random the columns, so as to avoid possible biases resulting from the structure of the original dataset (in our example, the documents might have been listed by topic). Thanks to the randomness in the choice of the pseudo-features, running a distinct pseudo-instance clustering on each pseudo-feature subset should provide roughly consistent clustering solutions.

With respect to the point iii), we observe that it involves a non-trivial problem: the reconciliation of the different partitions. Though, this can be performed by standard partition consensus algorithms. The main approaches to this problem consist either in creating the partition that shares the maximum information with the ones available [206] or creating the solution partition

by aggregation e.g. by majority voting/boosting [203, 204] (a review of those *cluster ensemble methods* can be found in [205]). We choose the second approach.

We demonstrate the overall approach using the MNIST dataset of handwritten digits [201]. The instances of the dataset are $n = 60000$. The features of each image are determined by its pixels: each image has $m = 784$ pixels (they are square images of $m = p \times p$ pixels, with $p = 28$): to each image-pixel pair is associated the the gray-scale intensity of the pixel in that image (a numerical value in the interval $[0, 255]$). Using our approach, we obtain a partition of the set of pixels, into subsets, each corresponding to a view.

With respect to the document-word-count example – that we will continue using throughout the paper for illustrating the method – the relevant relationships are the following: the images of handwritten digits correspond to the documents (the original instances); the pixels correspond to the words (the original features); the count of the number of occurrences of a words in a document is substituted by the gray-scale intensity value of the pixel. The views consisting of subsets of words are replaced by views consisting of subsets of pixels.

In principle, the views issued by our method can be later used for multi-view learning (e.g. using the views separately to learn relatively weak classifiers, then having the views to coordinate into a multi-phase classification process). However the study of the multi-view learning phase of the process is out of the scope of the current work: the application of our approach on the mentioned example is aimed only at demonstrating the procedure. We return on the relation between view splitting phase and multi-view learning phase in the Results section.

B.6 Formalization of the Method

B.6.1 Notation

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote a set of objects/points/instances/examples. Each object corresponds to a point in a m-dimensional feature space: the i-th object can be represented as a row vector $x_i = x_{i*} = (x_{i1}, x_{i2}, \ldots, x_{im})$, each element of the vector corresponding to an *explanatory variable* or *feature*.

The row vectors make up a data matrix X. Each column vector of the data matrix X represents the values taken by a feature over the different objects: the j-th feature can be represented as $x_{*j} = (x_{1j}, x_{2j}, \ldots, x_{nj})$.

To represent the operations in the dual space, it is useful to denote the transpose $X^\top$ of X by a matrix $Y = X^\top$. We treat the m features of the dataset X as instances of the dataset Y, and call the m rows of Y *pseudo-instances*; similarly, we treat the n rows of the dataset X as features of the dataset Y, and call the n columns of Y *pseudo-features*. When using a single index, we refer to a whole array: x_i refers to the i-th instance of the data matrix X, while y_j refers to the j-th pseudo-instance of the data matrix Y. The set of *pseudo-instances* can be denoted by the collection of row vectors $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$.

B.6.2 A dual-space approach to unsupervised MVG

The multi-view generation task consists in the following problem. *Given* a dataset in the form of an $n \times m$ matrix X – with n, rows representing the instances, and m columns, representing the features – *find* a partition of the feature set consisting in k blocks, so as to optimize a specific objective function of the intra-view and inter-view similarity.

The objective functions typically used in relation to this task can encode several requirements. The main requirement considered in literature is the following: the information held by each view should be as much as possible *unique* (maximal inter-view diversity, minimal inter-view redundancy requirements). Those methods that have access also to the classifiers/regressors later used in the training, can consider also requirements such as sufficiency of the view (good predicting power) and compatibility (the classifiers trained on the different views, given an instance, should predict the same label with high probability). In our case we assume we do not have access to the classifier/regressor to be used in the training, therefore we consider only the maximum inter-view diversity, i.e. *minimum inter-view redundancy* requirement.

B.6.2.1 Requirements and distance definition

We pursue the attainment of the *minimum inter-view redundancy* requirement indirectly, by *maximizing the intra-view redundance*: features should be grouped together if they contain partially redundant information, or equivalently if one feature contains much information about the other. To this purpose we try to group together those features that are close in this prespecified sense: *two features are close if for many objects they have similar values (on a standardized scale)*: intuitively, knowing the values of one feature (on a collection of objects) can help guessing the value of the other feature on the other feature (on the same array of objects). This concept can be concretized in a variety of ways, each one dense of assumptions about the process that generated the data. We chose to use the above stylized definition: two features are close if they provide similar values on many objects.

In our reference example, where the objects are documents and the features are words, two words are considered close to one another if they have similar (percentage of) occurrence in several documents. Notice that we are not advocating this as a definition of distance between two words specially meaningful in many contexts: we just illustrate how the definition of distance between features would translate in terms of our example; the usefulness of this definition consists in providing a way of creating views with high intra-view redundance.

From this definition of pairwise distance between features one can build groupings of similar features, for instance as centroid-based clustering algorithms do.

To this purpose, we pass from the original data matrix X to its transpose $Y = X^\top$ and considering the rows of Y as new data-points (the pseudo-instances) we define the pairwise row distance as an L^2 distance, i.e. an Euclidean distance $d_E(\cdot, \cdot)$. The distance between *row* y_j and *row* $y_{j'}$ is defined as

$$d(y_j, y_{j'}) = d(j, j') = \left(\sum_{i=1}^{n} (y_{j,i} - y_{j',i})^2 \right)^{\frac{1}{2}}$$

where i runs over all the pseudo-features (i.e. the former objects). Based on this distance one can run the clustering algorithm k-means [219] or another algorithm belonging to the same family, such as k-medoid [202].

B.6.2.2 Output of a single pseudo-instance clustering and dimensionality problems

Running the k-means partitioning algorithm, one obtains a solution for the problem of pseudo-instance partitioning based on the dataset Y. This will also be a solution of the MVG (i.e. feature partitioning) problem based on X.

In practice, however when the number n of rows of X is large, after transposition, the number of pseudo-features (columns of Y) makes the clustering problem high-dimensional, and the clustering algorithm potentially less effective (e.g. a significant difference of two points along a dimension could be obfuscated by many non-significant differences along other dimensions).

One can address the issue by breaking the set of *pseudo-features* into r smaller redundant subsets of s elements each (approximately n/r elements each), then by running the clustering algorithm separately on the whole set of pseudo-instances, described only by a group of s pseudo-features. This yields r distinct cluster partitions. Eventually, the different partitions can be aggregated by a partition consensus algorithm, to produce an individual partition solution.

In terms of our reference example – in which the objects are the documents and the features are the words, and in which the pseudo-instances are the words, while the pseudo-features are the documents – this corresponds to breaking the corpus into r randomly chosen groups of documents and running the clustering algorithm r times over all the pseudo-instances (words), using only s pseudo-features at time, then aggregating the resulting r partitions into a single solution partition: e.g. a word will be assigned to the partition block to which it belongs most often.

This task is formally described in the next subsection.

B.6.3 The consensus clustering task

A *clusterer* Φ is a function that, given a set $\mathcal{Y}$, outputs a partition π under the form of a label vector λ. Different *clusterers* $\Phi^{(1)}, \Phi^{(2)}, \ldots, \Phi^{(r)}$, run over the same dataset $\mathcal{Y}$ will output, in general, different label vectors $\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(q)}, \ldots, \lambda^{(r)}$.

A collection of label vectors $\Lambda = \{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(r)}\}$ can be combined into a single label vector $\hat{\lambda}$, called *consensus labelling*, by using a *consensus function* Γ. Equivalently one can say that Γ combines the corresponding collection Π of partitions $\Pi = \{\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(q)}, \ldots, \pi^{(r)}\}$ into a single partition $\hat{\pi}$. Given r partitions, the $\lambda^{(q)}$ partition consisting in k clusters/blocks, a consensus function is defined as a mapping $\mathbb{N}^{n \times r} \to \mathbb{N}^n$ taking a set of partitions into an integrated partition, $\Gamma : \Lambda \to \hat{\lambda}$, or equivalently $\Gamma : \Pi \to \hat{\pi}$.

The *consensus clustering problem* consists in finding a new partition $\hat{\pi}$ of the data $\mathcal{Y}$, given the partitions in Π, such that the objects in a block/cluster of $\hat{\pi}$ are more similar (in some pre-specified sense) to each other, than the object in different clusters of $\hat{\pi}$. The solution of the problem can be defined in different ways [205]: some are based on minimization of information theoretic measures, some on different forms of aggregation, such as majority voting (bagging). We will use the latter approach. Beforehand, however we discuss a minor technical issue.

B.6.3.1 Logical equivalence

The reconciliation of the different partitions involves an ancillary issue: there are partitions that are denoted by different arrays of labels, but that are the same from the logical point of view (a suitable permutation of the symbols used to denote the labels is able to transform one in another). Indeed, for each unique partition there are $k!$ equivalent representations as integer label vectors: i.e. given two equivalent partitions there exists a permutation of the labels such that they become equal. Formally, given the set P of the $k!$ permutations of $[1, k]$, two label vectors $\lambda^{(a)}$ and $\lambda^{(b)}$ are said to be *logically identical* if there exist a permutation $p \in P$, taking $\lambda^{(a)}$ into $\lambda^{(a)'}$ such that $\lambda^{(a)'}(y_j) = \lambda^{(b)}(y_j),\ \forall j \in [1, m]$. One needs to account for those equivalences, in order not to make the task of partition reconciliation uselessly complex: this issue can be solved passing to a canonical form [206]. Indeed the solution to this potentially complex correspondence problem is very simple. The pseudo-instances are endowed by a numerical index, setting a natural ordering for the set of pseudo-instances. After obtaining the r different partitions from the r clusterers, one should rewrite, for each partition, the block labels, so that the index of the blocks is monotonic (e.g. monotonic non-decreasing) w.r.t. to the order of the objects. By rewriting each partition in this way, logically identical partitions will be mapped onto the same representation. Formally, one should enforce for each partition the

constraints (i) $\lambda(y_1) = 1$ (the first object defines the first block/cluster) and (ii) $\lambda(y_{i+1}) \leq \max_{j=1,\ldots,i}(\lambda(y_j)) + 1 \ \forall i = 1, 2, \ldots, (n-1)$ (the cluster label $\lambda(y_{i+1})$ either has a label that occurred before, or has a label that increases by one unit w.r.t. the highest used so far).

B.6.3.2 Partition reconciliation, by majority voting

Once the r partitions are in canonical form, it is straightforward to aggregate them into a single solution partition $\hat{\lambda}$: one assigns the pseudo-instance to its most frequently occurring label.

$$\hat{\lambda}(y_j) = \arg\max_{\lambda} count(\lambda(y_j))$$

In the reference example where the pseudo-instances are the words, this corresponds to assigning a word to the partition block in which it occurs most often. Ties can be resolved by random choice. The overall process is illustrated in Figure B.1

B.7 Results

We demonstrate the proposed approach using the MNIST dataset containing gray-scale images of handwritten digits [201]. With respect to the document-word reference example, where we had documents we now have pictures, where we had word occurrence counts we have pixel gray levels.

B.7.1 The dataset

The dataset contains $n = 60,000$ gray-scale images: each image instance represents a handwritten digit. Although the class label for each image is available (there are 10 classes, $\{0, 1, 2, \ldots, 9\}$), it is not used in our unsupervised setting. The features considered for each instance are the gray-scale intensities of the pixels, each intensity takes an integer value in the interval $[0, 255]$. In the original version of the dataset each image has $28 \times 28 = 784$ pixels (there are $m = 784$ features for each instance). Thus, the dataset can be represented in terms of a $n = 60000$ row and $m = 784$ column table.

B.7.2 The process

Our final goal was to obtain a partitioning of the columns into k views, i.e. the whole area of the image in k pixel regions (later to be used separately to train distinct classifiers).

Following the above described approach, we transposed the table to obtain a table with $m = 784$ rows (pseudo-instances, the pixels) and $n = 60000$ columns (pseudo-features, the images); then we broke this column set in r splits (each of $s = n/r$ columns, i.e. images, chosen at random *without restitution*); using each split we ran k-means using all the m pseudo-instances, thus we obtained r partitions of the pseudo-instances; finally we reconciled the r partitions into a single partition solution, by majority voting.

Transposing back the partitioned data table we got a view partitioning of the image features, i.e. a partitioning of the pixels in regions that share some similarity. The similarity defined by the application of k-means was the co-occurrence of equal or similar gray-levels.

B.7.3 The outcomes

We experimented different values of the number-of-views parameter k (number of pixel regions, $k = \{2, 3, 5, 9, 13, 17\}$) and the split-granularity parameter r (and consequently $s = n/r$, number of images/pseudo-features contained in a pseudo-feature split). We chose vales of r which together could represent almost the whole range, leaving out the extremities (the single "split" case, with $r = 1$ and the one-element split, with $s = 1$). The results are shown in Figure B.3.

One can see, in the first row of Figure B.3 (also with reference to Figure B.2) that for $k = 2$ views, the process neatly distinguishes between the active region and the non-active region (whose pixels are almost never used). For $k = 3$ views, one can distinguish further, inside the central active region, two sub-regions with different importance in detailing the digits. With $k = 5$ views, and up, the process issues views, which detail the difference of the regions even further; also the different values of s return varying shapes.

B.8 Validation

In principle, the views thus obtained could be later used for multi-view learning. This could consist either in unsupervised multi-view learning, e.g. multi-view clustering, or in supervised ore semi-supervised multi-view learning. In the latter case the learning phase would involve the class labels: in our case study the symbol of the represented digit.

For instance in an hypothetical semi-supervised setting, one might know the class label only for a subset of images and might want to predict the class for the reminder images. This would be carried out by training distinct models separately on each view, and then using them to create the missing labels by means of co-training [218]. In this case, a direct validation of the effectiveness of our multi-view generation method could consist in a study of the quality of the co-training phase resulting from the use of the proposed MVG phase.

Such direct validation, however, would apply only to the specific multi-view technique considered. On the other hand, a wider systematic study of different multi-view techniques, would go beyond the scope of the present paper.

Nevertheless, it is possible to perform a indirect validation of the method, endowed by a reasonable generality, based on the following considerations. In a supervised multi-view setting, one requires that the views issued by the MVG phase fulfill some natural requirements [218]:

1) each view must individually, be endowed with *predicting power*,

2) each view must hold *unique information* about the targets,

3) the different views should achieve prevalently consistent predictions.

An indirect validation of our method can be performed by checking that the views generated fulfill those base requirements. We opted for such an indirect validation. For the sake of simplicity, we did not consider the third requirement, which is more complex to account for with a high number of views and many classes. We focused on the first two requirements.

We ran two kinds of learning models on the views that were obtained by our method: a Naïve Bayesian classifier (NB) and a Decision Tree (DT). Since the findings from the two learners were in qualitative agreement, hereafter, for space reasons, we report only about the NB classifier.

For each parameter setting (each sub-figure in Figure B.3), we ran the learner(s) on all the views of the $n = 60000$ instances MNIST training set and measured the accuracy of the prediction using the $n = 10000$ instances MINT test set. We computed both the individual accuracy in the classification of each individual symbol/class (digits from 0 to 9) and the average accuracy over all the classes. The results are shown in Figure B.4 for some representative combination of parameters of the $k = 3$ and $k = 5$ views cases. For completeness we also computed the accuracy of the classifier defined by the bagged version of the different views. We also trained, for comparison, a single view NB classifier. The plots allow to appreciate both the predicting power of the individual views, and the fact that they are endowed with unique information about the targets.

B.8.1 Requirement 1: Predicting power.

The accuracy of the NB classifier trained on individual views, is always (up to the level studied of $k = 17$ views) much greater than the baseline random classifier (which would have accuracy $a = 0.10$ for each target), and for small number of views (large amount of information in each view) is often comparable to the reference single view classifier accuracy, $a = 0.84$ (this is the accuracy of the NB classifier applied to all the features/pixels, gathered into a single view). Looking at specific target classes one can observe that some views achieve a reasonably high performance at least on one class.

B.8.2 Requirement 2: Unique information on targets

As to this requirement, one could already qualitatively see, from Figure B.3, that the views concretize in pixel areas covering regions, approximately corresponding to constructive elements of the digits: for instance for $k = 9$ (e.g. with $r = 120$), one can see distinctly that including or omitting some patches one can build the digit **3** or the digit **9** or the digit **8**. Thus, each view holds information that is not available to others for determining the class of the image. This is confirmed in Figure B.4. The views have different efficiencies for different targets: each view is the top accuracy view for at least one target. In other words, each view would have something to teach to the others, for example within a co-training process.

In short, both main requirements are fulfilled.

B.9 Discussion, conclusions and outlook

In this work we approached the Multi View Generation problem in an unsupervised, setting. We proposed an approach based on the transposition of the data table: the original instance rows are mapped into columns (the *pseudo-features*), while the original feature columns become rows (the *pseudo-instances*); the latter can then be partitioned by any suitable standard instance-partitioning algorithm: the resulting groups can be considered as groups of the original features, i.e. views, solution problem. We demonstrated the approach using k-means.

With reference to our document-word reference example, notice, for the sake of comparison, that the task of "clustering words based on the documents to which they belong " and the task of "clustering documents based on the words that they contain" are commonly studied classical tasks. In the former task one assigns to the words the role of instances and the the documents the role of features; in the latter case it is the converse. However, in both classical cases, the final aim of the procedure is to come up with a clustering of the instances. Our approach, on the contrary, aims at producing a partition of the features into views (later to be used by independent prediction models).

Furthermore, the effectiveness of the two mentioned classical tasks is quantified based on the quality of the instance partition or in relation to another reference partition: either based on intrinsic criteria (such as the goodness of clustering Hubert's r statistics) or based on external criteria by comparing the clusters to ground truth classes (e.g. using purity, Rand index, Jaccard index and so on). In our case, on the contrary, the outcome of the partition is assessed in relation of the effectiveness of the multi-view partition in supporting a subsequent learning procedure.

Another technique worth mentioning, for comparison, is co-clustering [195, 196]. Co-clustering models the relation of words and documents as a bipartite graph: the co-clustering algorithms find sub-graphs of the initial connected component graph, using spectral methods. It is true that the output provides a clustering of the words and a clustering of the documents: the words (documents) belonging to the same subgraph are in the same word- (document-) cluster. It is also true that the method issues a simultaneous clustering of the features and of the instances. However, the method is radically different from ours, since it involves a *joint minimization* and in general will not

provide the same results.

Our method provides an unsupervised multi view partition. As for any unsupervised optimization task, whose output is used in input of a supervised (or semi-supervised) task, issue might arise that the solution of the former task is not necessarily optimal to the latter. This is a problem that can be found in many settings, it can take place for instance, when using a learner after having applied Principal Component Analysis, or any other representation learner.

The point that we wanted to make is that one can take methods designed for working in instance space and use them in feature space. The application of such dual-space approach can be extended to many other situations, that will be the object of future works.

B.10 Acknowledgements

The authors acknowledge the support of the Information and Communication Technology Fund (ICT Fund) at EBTIC/Khalifa University, Abu Dhabi, UAE (Project number 88434000029). The work was partially founded also by the European Union's Horizon 2020 research and innovation programme, within the projects Toreador (grant agreement No. 688797), Evotion (grant agreement No. 727521) and Threat-Arrest (Project-ID No. 786890).

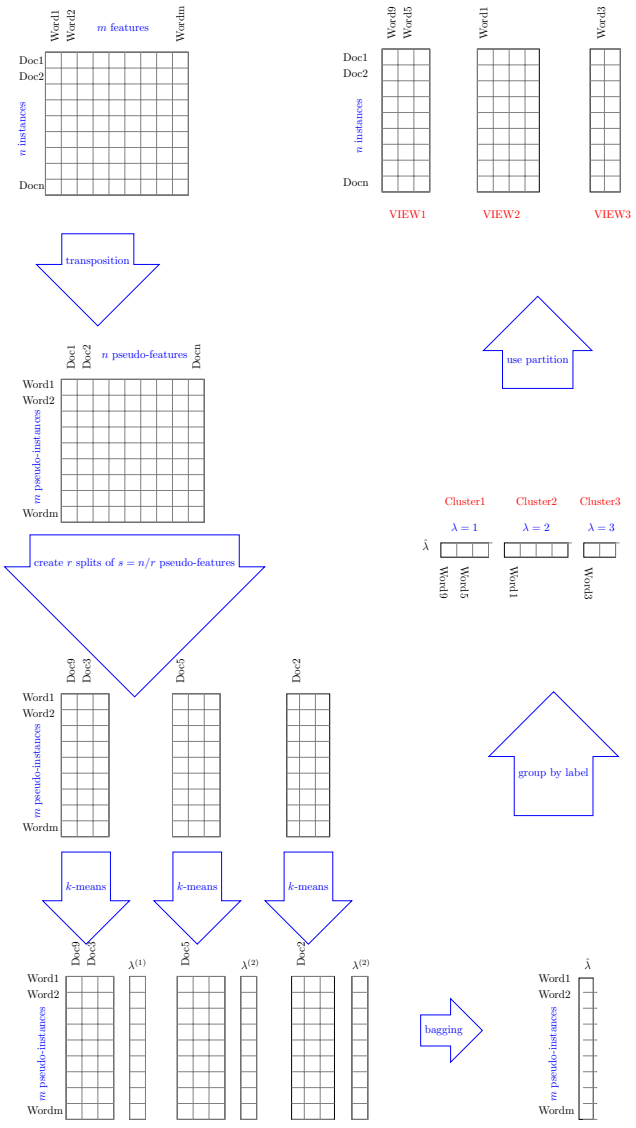


Figure B.1: Illustration of the overall process based on the reference example.

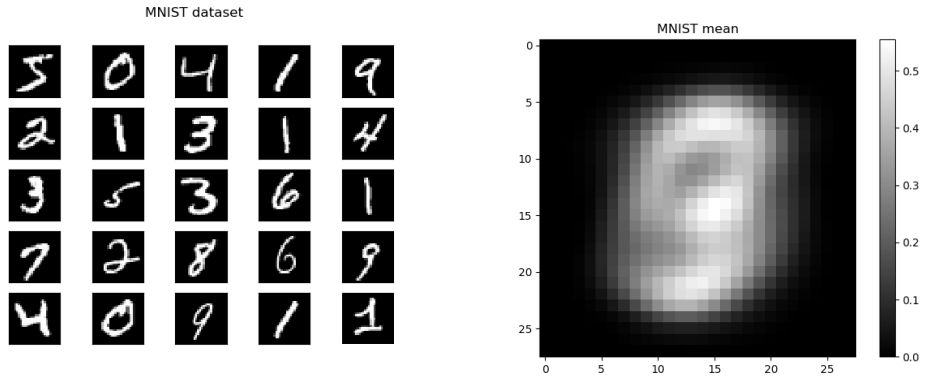


Figure B.2: Left: the first 25 images of the MNIST dataset. Right: the average gray-level taken over the whole set: the pictures hints at a "background" region little or not used by the handwritten digits.

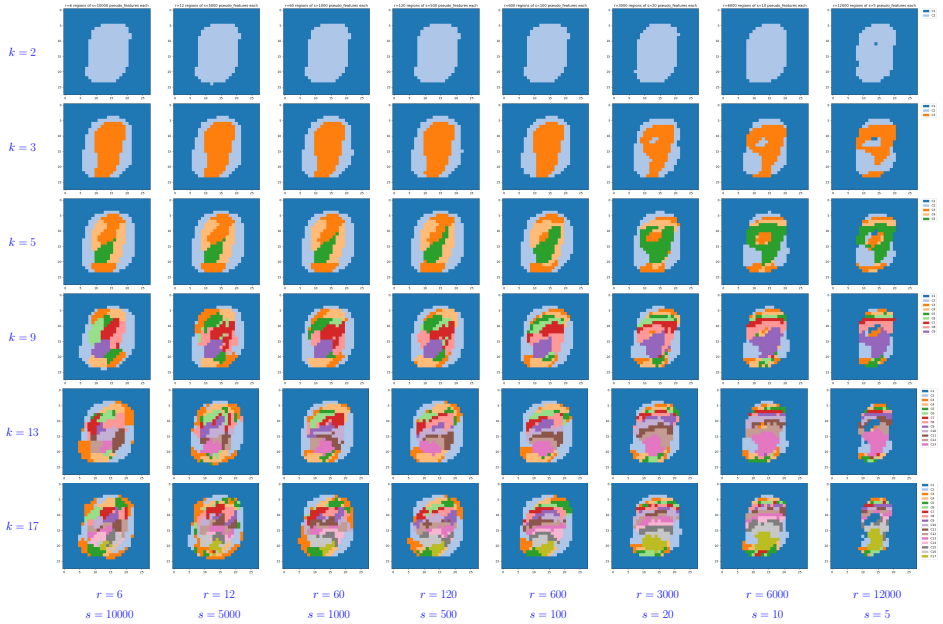


Figure B.3: Outcome of the view splitting process for three different resolutions. Each color corresponds to a cluster of pixels and represents a view. The parameter k is the number of views. The number of instances used for the task was $n = 60000$. The parameters r is the number of independent k-means clustering processes obtained by sectioning the data, then reconciliated in a single clustering partition. Finally, $s = n/r$. See also text of the Results Section.

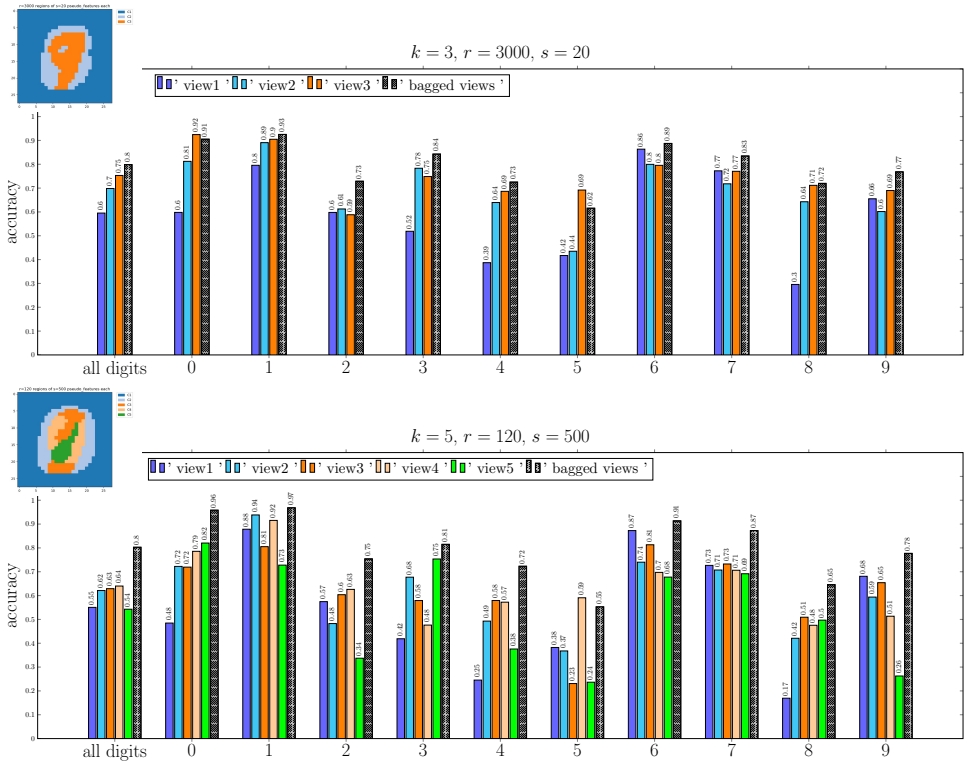


Figure B.4: Accuracies for Naïve Bayes classifiers trained on individual views (colored bars) and performance of the bagged classifier (gray bars), for the different targets (digits 0 to 9, ten rightmost groups) and averaged over all the targets (first, i.e. leftmost group of bars). From the top to the bottom: $k = 3$ views (with $r = 3000$ and $s = 20$); $k = 5$ views (with $r = 120$ and $s = 500$).

Bibliography

[1] Shiliang Sun, Liang Mao, Ziang Dong, and Lidan Wu. *Multiview machine learning*. Springer, 2019.

[2] Philippe Smets. *The Transferable Belief Model for Quantified Belief Representation*, pages 267–301. Springer Netherlands, Dordrecht, 1998.

[3] Oskar Skibski, Tomasz P Michalak, and Talal Rahwan. Axiomatic characterization of game-theoretic centrality. *Journal of Artificial Intelligence Research*, 62:33–68, 2018.

[4] Stefano Moretti and Fioravante Patrone. Transversality of the shapley value. *TOP*, 16(1):1, Apr 2008.

[5] Julian Stier, Gabriele Gianini, Michael Granitzer, and Konstantin Ziegler. Analysing neural network topologies: a game theoretic approach. *Procedia Computer Science*, 126:234 – 243, 2018. Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia.

[6] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613, 2011.

[7] Olivier Caelen, Gabriele Gianini, and Ernesto Damiani. Fr3065558a1, system and method to manage the detection of fraud in a system of financial transactions.

[8] Luis Vergara, Antonio Soriano, Gonzalo Safont, and Addisson Salazar. On the fusion of non-independent detectors. *Digital Signal Processing*, 50:24–33, 2016.

[9] Addisson Salazar, Gonzalo Safont, Antonio Soriano, and Luis Vergara. Automatic credit card fraud detection based on non-linear signal processing. In *2012 IEEE International Carnahan Conference on Security Technology (ICCST)*, pages 207–212. IEEE, 2012.

[10] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[11] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[12] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[13] Alon Keinan, Ben Sandbank, Claus C Hilgetag, Isaac Meilijson, and Eytan Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Neural computation*, 16(9):1887–1915, 2004.

[14] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[15] Meshesha Legesse, Gabriele Gianini, and Dereje Teferi. Selecting feature-words in tag sense disambiguation based on their shapley value. In *Signal-Image Technology and Internet-Based Systems (SITIS), 2016 12th International Conference on*, pages 236–240. IEEE, 2016.

[16] Alexandre Fréchette, Lars Kotthoff, Tomasz Michalak, Talal Rahwan, Holger H Hoos, and Kevin Leyton-Brown. Using the shapley value to analyze algorithm portfolios. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[17] Michel Grabisch. *Set functions, games and capacities in decision making*, volume 46. Springer.

[18] Lloyd S Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *American political science review*, 48(03):787–792, 1954.

[19] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[20] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[21] Pooria Joulani, András György, and Csaba Szepesvári. Online learning under delayed feedback. *arXiv preprint arXiv:1306.0686*, 2013.

[22] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[23] Alina Beygelzimer, Satyen Kale, and Haipeng Luo. Optimal and adaptive algorithms for online boosting. *arXiv preprint arXiv:1502.02651*, 2015.

[24] Nikunj C Oza. Aveboost2: Boosting for noisy data. In *Multiple Classifier Systems*, pages 31–40. Springer, 2004.

[25] Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Comparing boosting and bagging techniques with noisy and imbalanced data. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(3):552–568, 2011.

[26] Guangzhe Fan and Mu Zhu. Detection of rare items with target. *Statistics and Its Interface*, 4:11–17, 2011.

[27] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999.

[28] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[29] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484, 2012.

[30] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[31] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. When is undersampling effective in unbalanced classification tasks? In *Machine Learning and Knowledge Discovery in Databases*, pages 200–215. Springer, 2015.

[32] Haibo He, Edwardo Garcia, et al. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[33] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[34] Andrea Dal Pozzolo, Olivier Caelen, Yann-Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014.

[35] Maciej A Mazurowski, Piotr A Habas, Jacek M Zurada, Joseph Y Lo, Jay A Baker, and Georgia D Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks*, 21(2):427–436, 2008.

[36] Mahbod Tavallaee, Natalia Stakhanova, and Ali Akbar Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(5):516–524, 2010.

[37] Mahesh V Joshi, Vipin Kumar, and Ramesh C Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 257–264. IEEE, 2001.

[38] Andrea Dal Pozzolo, Reid Johnson, Olivier Caelen, Serge Waterschoot, Nitesh V Chawla, and Gianluca Bontempi. Using hddt to avoid instances propagation in unbalanced and evolving data streams. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 588–594. IEEE, 2014.

[39] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2007.

[40] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

[41] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[42] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[43] David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

[44] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Advances in information retrieval*, pages 345–359. Springer, 2005.

[45] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[46] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015.

[47] Alejandro Correa Bahnsen, Djamila Aouada, Aleksandar Stojanovic, and Björn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 2016.

[48] Bertrand Lebichot, Fabian Braun, Olivier Caelen, and Marco Saerens. A graph-based, semi-supervised, credit card fraud detection system. In *International Workshop on Complex Networks and their Applications*, pages 721–733. Springer, 2016.

[49] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 2019.

[50] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, and Gianluca Bontempi. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. *International Journal of Data Science and Analytics*, 5(4):285–300, 2018.

[51] Fabrizio Carcillo, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, and Gianluca Bontempi. Scarff: a scalable framework for streaming credit card fraud detection with spark. *Information fusion*, 41:182–194, 2018.

[52] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection: a realistic modeling and a novel learning strategy. *IEEE transactions on neural networks and learning systems*, 29(8):3784–3797, 2017.

[53] Yvan Lucas, Pierre-Edouard Portier, Léa Laporte, Sylvie Calabretto, Olivier Caelen, Liyun He-Guelton, and Michael Granitzer. Multiple perspectives hmm-based feature engineering for credit card fraud detection. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1359–1361. ACM, 2019.

[54] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 2018.

[55] Mathieu Garchery and Michael Granitzer. On the influence of categorical features in ranking anomalies using mixed data. *Procedia Computer Science*, 126:77–86, 2018.

[56] Konstantin Ziegler, Olivier Caelen, Mathieu Garchery, Michael Granitzer, Liyun He-Guelton, Johannes Jurgovsky, Pierre-Edouard Portier, and Stefan Zwicklbauer. Injecting semantic background knowledge into neural networks using graph embeddings. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 200–205. IEEE, 2017.

[57] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, volume 1, pages 29–36, Jan 2005.

[58] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd*

international conference on Machine learning, pages 265–272. ACM, 2005.

[59] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[60] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[61] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[62] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[63] Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[64] Shahar Dobzinski and Jan Vondrák. From query complexity to computational complexity. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1107–1116. ACM, 2012.

[65] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 42(1):265–304, 2013.

[66] Abu Bakar Siddique, Saadia Farid, and Muhammad Tahir. Proof of bijection for combinatorial number system. *arXiv preprint arXiv:1601.05794*, 2016.

[67] Gilbert Strang, Gilbert Strang, Gilbert Strang, and Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[68] Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

[69] A Charnes, B Golany, M Keane, and J Rousseau. Extremal principle solutions of games in characteristic function form: core, chebychev and shapley value generalizations. In *Econometrics of planning and efficiency*, pages 123–133. Springer, 1988.

[70] Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

[71] Guoli Ding, Robert F Lax, Jianhua Chen, and Peter P Chen. Formulas for approximating pseudo-boolean random variables. *Discrete Applied Mathematics*, 156(10):1581–1597, 2008.

[72] Guoli Ding, Robert F Lax, Jianhua Chen, Peter P Chen, and Brian D Marx. Transforms of pseudo-boolean random variables. *Discrete Applied Mathematics*, 158(1):13–24, 2010.

[73] Ronald R Yager and Janusz Kacprzyk. *The ordered weighted averaging operators: theory and applications*. Springer Science & Business Media, 2012.

[74] Petter Strandmark and Fredrik Kahl. Pseudo-boolean optimization: Theory and applications in vision. In *Swedish Symposium on Image Analysis (SSBA)*, 2012.

[75] Michel Grabisch. Bases and transforms of set functions. In *On Logical, Algebraic, and Probabilistic Aspects of Fuzzy Set Theory*, pages 215–231. Springer, 2016.

[76] Guillermo Owen. Multilinear extensions of games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pages 139–151, 1988.

[77] Lionel S Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

[78] Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

[79] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[80] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[81] Robert J Weber. Probabilistic values for games. *The Shapley Value. Essays in Honor of Lloyd S. Shapley*, pages 101–119, 1988.

[82] Lloyd S Shapley. A value for n-person games. *Classics in game theory*, page 69, 1997.

[83] Dov Monderer and Dov Samet. Variations on the shapley value. *Handbook of game theory with economic applications*, 3:2055–2076, 2002.

[84] Michel Grabisch. K-order additive discrete fuzzy measures and their representation. *Fuzzy sets and systems*, 92(2):167–189, 1997.

[85] Michel Grabisch and Marc Roubens. Probabilistic interactions among players of a cooperative game. In *Beliefs, Interactions and Preferences in Decision Making*, pages 205–216. Springer, 1999.

[86] Jean-Luc Marichal and Marc Roubens. The chaining interaction index among players in cooperative games. In *Advances in Decision Analysis*, pages 69–85. Springer, 1999.

[87] Michel Grabisch and Fabien Lange. A new approach to the shapley value for games on lattices. In *4th Logic, Game Theory and Social Choice meeting, Caen, France (June 2005)*, 2005.

[88] Michel Grabisch and Christophe Labreuche. Derivative of functions over lattices as a basis for the notion of interaction between attributes. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):151–170, 2007.

[89] Shujin Li, Xiaoning Li, and Qiang Zhang. The banzhaf interaction index in game with coalition structure. In *2008 Chinese Control and Decision Conference*, pages 1196–1200. IEEE, 2008.

[90] Stefano Moretti and Fioravante Patrone. Transversality of the shapley value. *Top*, 16(1):1, 2008.

[91] Fabien Lange and Michel Grabisch. The interaction transform for functions on lattices. *Discrete Mathematics*, 309(12):4037–4048, 2009.

[92] Guilliermo Owen. Values of games with a priori unions. In *Mathematical economics and game theory*, pages 76–88. Springer, 1977.

[93] Guillermo Owen. Modification of the banzhaf-coleman index for games with a priori unions. In *Power, voting, and voting power*, pages 232–238. Springer, 1981.

[94] José M Alonso-Meijide and M Gloria Fiestras-Janeiro. Modification of the banzhaf value for games with a coalition structure. *Annals of Operations Research*, 109(1-4):213–227, 2002.

[95] M Roubens. Interaction between criteria and definition of weights in mcda problems. In *44th Meeting of the European Working Group "Multicriteria Aid for Decisions", Brussels, Belgium*, 1996.

[96] Dieter Denneberg and Michel Grabisch. Interaction transform of set functions over a finite set. *Information Sciences*, 121(1-2):149–170, 1999.

[97] Ron Holzman, Ehud Lehrer, and Nathan Linial. Some bounds for the banzhaf index and other semivalues. *Mathematics of Operations Research*, 13(2):358–363, 1988.

[98] Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. A linear approximation method for the shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

[99] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[100] Yoram Bachrach, Evangelos Markakis, Ezra Resnick, Ariel D Procaccia, Jeffrey S Rosenschein, and Amin Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, 20(2):105–122, 2010.

[101] Shaheen S Fatima, Michael Wooldridge, and Nicholas R Jennings. An approximation method for power indices for voting games. In *Innovations in Agent-Based Complex Automated Negotiations*, pages 179–194. Springer, 2010.

[102] Jean-Luc Marichal and Pierre Mathonet. Weighted banzhaf power and interaction indexes through weighted approximations of games. *European Journal of Operational Research*, 211(2):352–358, 2011.

[103] Shaheen Fatima, Michael Wooldridge, and Nicholas R Jennings. A heuristic approximation method for the banzhaf index for voting games. *Multiagent and Grid Systems*, 8(3):257–274, 2012.

[104] Tomasz P Michalak, Karthik V Aadithya, Piotr L Szczepanski, Balaraman Ravindran, and Nicholas R Jennings. Efficient computation of the

shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research*, 46:607–650, 2013.

[105] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

[106] W Hoeffding. Probability inequalities for sums of bounded random variables. *Wiley StatsRef: Statistics Reference Online*, 2014.

[107] Krishna V Acharya, Himadri Mukherjee, and Jajati K Sahoo. Approximation of banzhaf indices and its application to voting games. *arXiv preprint arXiv:1801.08029*, 2018.

[108] Tjeerd van Campen, Herbert Hamers, Bart Husslage, and Roy Lindelauf. A new approximation method for the shapley value applied to the wtc 9/11 terrorist attack. *Social Network Analysis and Mining*, 8(1):3, 2018.

[109] Gian-Carlo Rota. On the foundations of combinatorial theory i. theory of möbius functions. *Probability theory and related fields*, 2(4):340–368, 1964.

[110] Pradeep Dubey, Abraham Neyman, and Robert James Weber. Value theory without efficiency. *Mathematics of Operations Research*, 6(1):122–128, 1981.

[111] Francesc Carreras and Antonio Magaña. The multilinear extension and the modified banzhaf-coleman index. *Mathematical Social Sciences*, 28(3):215–222, 1994.

[112] Vincent Feltkamp. Alternative axiomatic characterizations of the shapley and banzhaf values. *International Journal of Game Theory*, 24(2):179–186, 1995.

[113] Andrzej S Nowak. On an axiomatization of the banzhaf value without the additivity axiom. *International Journal of Game Theory*, 26(1):137–141, 1997.

[114] Rene Van den Brink and Gerard Van der Laan. Axiomatizations of the normalized banzhaf value and the shapley value. *Social Choice and Welfare*, 15(4):567–582, 1998.

[115] Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of Game Theory*, 28(4):547–565, 1999.

[116] Annick Laruelle and Federico Valenciano. *Shapley-Shubik and Banzhaf indices revisited*. Instituto Valenciano de Investigaciones Económicas, 2000.

[117] Michel Grabisch. An axiomatization of the shapley value and interaction index for games on lattices. Citeseer, 2004.

[118] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1):72–99, 2006.

[119] Jean-Luc Marichal, Ivan Kojadinovic, and Katsushige Fujimoto. Axiomatic characterizations of generalized values. *Discrete Applied Mathematics*, 155(1):26–43, 2007.

[120] José M Alonso-Meijide, Francesc Carreras, M Gloria Fiestras-Janeiro, and Guillermo Owen. A comparative axiomatic characterization of the banzhaf–owen coalitional value. *Decision Support Systems*, 43(3):701–712, 2007.

[121] Mustapha Ridaoui, Michel Grabisch, and Christophe Labreuche. An axiomatisation of the banzhaf value and interaction index for multichoice games. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 143–155. Springer, 2018.

[122] Kai Lai Chung. *Elementary probability theory with stochastic processes*. Springer Science & Business Media, 2012.

[123] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

[124] Lara Mikenina and H-J Zimmermann. Improved feature selection and classification by the 2-additive fuzzy measure. *Fuzzy sets and systems*, 107(2):197–218, 1999.

[125] Shay Cohen, Eytan Ruppin, and Gideon Dror. Feature selection based on the shapley value. *In other words*, 1:98Eqr, 2005.

[126] Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007.

[127] Xin Sun, Yanheng Liu, Jin Li, Jianqi Zhu, Huiling Chen, and Xuejie Liu. Feature evaluation and selection with cooperative game theory. *Pattern recognition*, 45(8):2992–3002, 2012.

[128] Xin Sun, Yanheng Liu, Jin Li, Jianqi Zhu, Xuejie Liu, and Huiling Chen. Using cooperative game theory to optimize the feature selection problem. *Neurocomputing*, 97:86–93, 2012.

[129] Bogdan Kulynych and Carmela Troncoso. Feature importance scores and lossless feature pruning using banzhaf power indices. *arXiv preprint arXiv:1711.04992*, 2017.

[130] Shounak Gore and Venu Govindaraju. Feature selection using cooperative game theory and relief algorithm. In *Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions*, pages 401–412. Springer, 2016.

[131] Jihong Liu and Guoxiong Wang. A hybrid feature selection method for data sets of thousands of variables. In *2010 2nd International Conference on Advanced Computer Control*, volume 2, pages 288–291. IEEE, 2010.

[132] Fatiha Mokdad, Djamel Bouchaffra, Nabil Zerrouki, and Azzedine Touazi. Determination of an optimal feature selection method based on maximum shapley value. In *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 116–121. IEEE, 2015.

[133] Mohammad Zaeri-Amirani, Fatemeh Afghah, and Sajad Mousavi. A feature selection method based on shapley value to false alarm reduction in icus a genetic-algorithm approach. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 319–323. IEEE, 2018.

[134] Iñigo Barandiaran. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8), 1998.

[135] Sally Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. In *ICML*, pages 327–334, 2000.

[136] Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *IJCAI*, volume 5, pages 908–913, 2005.

[137] Minmin Chen, Yixin Chen, and Kilian Q Weinberger. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 953–960, 2011.

[138] Ahmed Salaheldin and Neamat El-Gayar. Complementary feature splits for co-training. In *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, pages 1303–1308. IEEE, 2012.

[139] Corrado Mio, Gabriele Gianini, and Ernesto Damiani. K-means clustering in dual space for unsupervised feature partitioning in multi-view learning. In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 1–8. IEEE, 2018.

[140] Joseph St Amand and Jun Huan. Discriminative view learning for single view co-training. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2221–2226. ACM, 2016.

[141] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[142] Wei Wang and Zhi-Hua Zhou. Analyzing co-training style algorithms. In *European conference on machine learning*, pages 454–465. Springer, 2007.

[143] Nikos Karampatziakis and Paul Mineiro. Discriminative features via generalized eigenvectors. *arXiv preprint arXiv:1310.1934*, 2013.

[144] Jiye Liang, Feng Wang, Chuangyin Dang, and Yuhua Qian. An efficient rough feature selection algorithm with a multi-granulation view. *International Journal of Approximate Reasoning*, 53(6):912–926, 2012.

[145] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[146] Xuran Zhao, Nicholas Evans, and Jean-Luc Dugelay. A subspace co-training framework for multi-view clustering. *Pattern Recognition Letters*, 41:73–82, 2014.

[147] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[148] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[149] Maria-Florina Balcan, Avrim Blum, and Ke Yang. Co-training and expansion: Towards bridging theory and practice. In *Advances in neural information processing systems*, pages 89–96, 2005.

[150] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Cikm*, volume 5, page 3, 2000.

[151] Ulf Brefeld and Tobias Scheffer. Co-em support vector learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 16. ACM, 2004.

[152] Felix Feger and Irena Koprinska. Co-training using rbf nets and different feature splits. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1878–1885. IEEE, 2006.

[153] Zhi-Hua Zhou, De-Chuan Zhan, and Qiang Yang. Semi-supervised learning with very few labeled training examples. In *AAAI*, pages 675–680, 2007.

[154] Wen Zhang and Quan Zheng. Tsfs: A novel algorithm for single view co-training. In *2009 International Joint Conference on Computational Sciences and Optimization*, volume 1, pages 492–496. IEEE, 2009.

[155] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400, 2011.

[156] Ryan J Urbanowicz, Melissa Meeker, William La Cava, Randal S Olson, and Jason H Moore. Relief-based feature selection: introduction and review. *Journal of biomedical informatics*, 2018.

[157] Derek O'Connor. A historical note on shuffle algorithms, 2014.

[158] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[159] Charles N Zeeb, Patrick J Burns, et al. Random number generator recommendation. *Report prepared for Sandia National Laboratories, Albuquerque, NM. Available as a WWW document., URL= http://www. colostate. edu/˜ pburns/monte/documents. html*, 1997.

[160] Elaine B Barker and John Michael Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised).* US Department of Commerce, Technology Administration, National Institute of . . . , 2007.

[161] Michel Grabisch et al. *Set functions, games and capacities in decision making.* Springer, 2016.

[162] Bezalel Peleg and Peter Sudhölter. *Introduction to the theory of cooperative games*, volume 34. Springer Science & Business Media, 2007.

[163] Kevin Leyton-Brown and Yoav Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis lectures on artificial intelligence and machine learning*, 2(1):1–88, 2008.

[164] Michael Lones. Sean luke: essentials of metaheuristics, 2011.

[165] Peter L Hammer and Sergiu Rudeanu. *Boolean methods in operations research and related areas*, volume 7. Springer-Verlag New York Inc., 1968.

[166] Peter L Hammer and Ron Holzman. Approximations of pseudo-boolean functions; applications to game theory. *Zeitschrift für Operations Research*, 36(1):3–21, 1992.

[167] Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

[168] Lloyd S Shapley. *Additive and non-additive set functions.* Princeton University, 1953.

[169] Roger B Myerson. Conference structures and fair allocation rules. *International Journal of Game Theory*, 9(3):169–182, 1980.

[170] Ehud Kalai and Dov Samet. On weighted shapley values. *International Journal of Game Theory*, 16(3):205–222, 1987.

[171] Bart de Keijzer. A survey on the computation of power indices, 2008. http://www.pakvla.nl/bart/powerindexsurvey.pdf, Last accessed on 2019-07-01.

[172] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml/datasets.php, Last accessed on 2019-07-01.

[173] Wikipedia. Stirling numbers of the second kind, 2018. https://en.wikipedia.org/wiki/Stirling_numbers_of_the_second_kind, Last accessed on 2019-10-01.

[174] Wikipedia. Bell number, 2019. https://en.wikipedia.org/wiki/Bell_number, Last accessed on 2019-10-06.

[175] Wikipedia. Bernoulli number, 2019. https://en.wikipedia.org/wiki/Bernoulli_number, Last accessed on 2019-10-06.

[176] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[177] Python org. Python, 2019. https://www.python.org/, Last accessed on 2019-10-10.

[178] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[179] Christian Schulz. Graph partitioning and graph clusteringin theory and practice. *Institute for Theoretical Informatics Karlsruhe Institute of Technology (KIT).–May*, 20:24–187, 2016.

[180] Olivier Goldschmidt and Dorit S Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19(1):24–37, 1994.

[181] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[182] Bruce Hendrickson and R Leland. A multilevel algorithm for partitioning graphs, acm. In *IEEE conference on Supercomputing*, pages 435–446, 1995.

[183] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[184] Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.

[185] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[186] X Yu Stella and Jianbo Shi. Multiclass spectral clustering. In *null*, page 313. IEEE, 2003.

[187] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[188] Bogdan Nica. A brief introduction to spectral graph theory. *arXiv preprint arXiv:1609.08072*, 2016.

[189] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[190] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[191] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.

[192] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In *Advances in neural information processing systems*, pages 1413–1421, 2011.

[193] Steffen Bickel and Tobias Scheffer. Multi-view clustering. In *ICDM*, volume 4, pages 19–26, 2004.

[194] Stefanie Jegelka, Suvrit Sra, and Arindam Banerjee. Approximation algorithms for tensor clustering. In *International Conference on Algorithmic Learning Theory*, pages 368–383. Springer, 2009.

[195] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.

[196] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[197] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[198] Shiliang Sun and Feng Jin. Robust co-training. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(07):1113–1126, 2011.

[199] Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao. Bayesian co-training. *Journal of Machine Learning Research*, 12(Sep):2649–2680, 2011.

[200] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[201] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[202] Xue Li. *K-Means and K-Medoids*, pages 1588–1589. Springer US, Boston, MA, 2009.

[203] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[204] Hanan G Ayad and Mohamed S Kamel. Cumulative voting consensus method for partitions with variable number of clusters. *IEEE transactions on pattern analysis and machine intelligence*, 30(1):160–173, 2008.

[205] Joydeep Ghosh and Ayan Acharya. Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315, 2011.

[206] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[207] E. Damiani, G. Gianini, M. Ceci, and D. Malerba. Toward iot-friendly learning models. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1284–1289, July 2018.

[208] Meshesha Legesse, Gabriele Gianini, and Dereje Teferi. Selecting feature-words in tag sense disambiguation based on their shapley value. In *Signal-Image Technology & Internet-Based Systems (SITIS), 2016 12th International Conference on*, pages 236–240. IEEE, 2016.

[209] Meshesha Legesse, Gabriele Gianini, Dereje Teferi, Hatem Mousselly-Sergieh, David Coquil, and Elöd Egyed-Zsigmond. Unsupervised cue-words discovery for tag-sense disambiguation: comparing dissimilarity metrics. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, pages 24–28. ACM, 2015.

[210] Hatem Mousselly-Sergieh, Mario Döller, Elöd Egyed-Zsigmond, Gabriele Gianini, Harald Kosch, and Jean-Marie Pinon. Tag relatedness using laplacian score feature selection and adapted jensen-shannon divergence. In *International Conference on multimedia modeling*, pages 159–171. Springer, 2014.

[211] Hatem Mousselly-Sergieh, Elöd Egyed-Zsigmond, Gabriele Gianini, Mario Döller, Jean-Marie Pinon, and Harald Kosch. Tag relatedness in image folksonomies. *Document numérique*, 17(2):33–54, 2014.

[212] H Mousselly-sergieh, E Egyed-zsigmond, G Gianini, M Doller, H Kosch, and J Pinon. Tag similarity in folksonomies. In *INFORSID*, pages 277–291. INFORSID, 2013.

[213] Virgil Griffith and Tracey Ho. Quantifying redundant information in predicting a target random variable. *Entropy*, 17(7):4644–4653, 2015.

[214] Virgil Griffith and Christof Koch. Quantifying synergistic mutual information. In *Guided Self-Organization: Inception*, pages 159–190. Springer, 2014.

[215] Paul L Williams and Randall D Beer. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

[216] Masahiro Terabe and Kazuo Hashimoto. Evaluation criteria of feature splits for co-training. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2008, 2008.

[217] Felix Feger and Irena Koprinska. Co-training using rbf nets and different feature splits. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 1878–1885. IEEE, 2006.

[218] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[219] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[220] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304, 1998.

[221] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD)*, pages 21–34. Singapore, 1997.

[222] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[223] Fuyuan Cao, Jiye Liang, Deyu Li, Liang Bai, and Chuangyin Dang. A dissimilarity measure for the k-modes clustering algorithm. *Knowledge-Based Systems*, 26:120–127, 2012.

[224] Zengyou He, Xiaofei Xu, and Shengchun Deng. Scalable algorithms for clustering large datasets with mixed type attributes. *International Journal of Intelligent Systems*, 20(10):1077–1089, 2005.

[225] Zhexue Huang and Michael K Ng. A fuzzy k-modes algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, 7(4):446–452, 1999.

[226] Dae-Won Kim, Kwang H Lee, and Doheon Lee. Fuzzy clustering of categorical data using fuzzy centroids. *Pattern Recognition Letters*, 25(11):1263–1271, 2004.

[227] Basilis Boutsinas and T Papastergiou. On clustering tree structured data with categorical nature. *Pattern Recognition*, 41(12):3613–3623, 2008.

[228] NG De Bruijn, Ca van Ebbenhorst Tengbergen, and D Kruyswijk. On the set of divisors of a number. *Nieuw Arch. Wiskunde (2)*, 23:191–193, 1951.

[229] Patrick Bosc, Ernesto Damiani, and Mariagrazia Fugini. Fuzzy service selection in a distributed object-oriented environment. *IEEE Transactions on Fuzzy Systems*, 9(5):682–698, 2001.

[230] Ernesto Damiani, O D'Antona, and Francesco Regonati. Whitney numbers of some geometric lattices. *Journal of Combinatorial Theory, Series A*, 65(1):11–25, 1994.

[231] Ernesto Damiani, Barbara Olibini, and Letizia Tanca. Fuzzy techniques for xml data smushing. In *International Conference on Computational Intelligence*, pages 637–652. Springer, 2001.

[232] Ernesto Damiani, Claudio Ardagna, Paolo Ceravolo, and Nello Scarabottolo. Toward model-based big data-as-a-service: The toreador approach. In *Advances in Databases and Information Systems*, pages 3–9. Springer, 2017.

[233] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of machine learning research*, 12(Jul):2211–2268, 2011.

[234] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[235] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, 2011. ACM.

[236] D Loeb, Ernesto Damiani, and O D'Antona. Decompositions of bn and πn using symmetric chains. *Journal of Combinatorial Theory, Series A*, 65(1):151–157, 1994.

[237] Zdravko Markov. A lattice-based approach to hierarchical clustering. In *FLAIRS Conference*, pages 389–393, 2001.

[238] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[239] Zdzisław Pawlak. *Rough sets: Theoretical aspects of reasoning about data*, volume 9. Springer Science & Business Media, 2012.