UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI SCIENZE DELL'INFORMAZIONE
SETTORE SCIENTIFICO DISCIPLINARE INF/01 INFORMATICA

DOTTORATO IN INFORMATICA
XXIV CICLO

Graph-based approaches for imbalanced data in functional genomics

Tesi di Dottorato di Ricerca di:
Marco Frasca

Relatore:
Prof Alberto Bertoni

Correlatore:
Prof. Giorgio Valentini

Coordinatore del Dottorato:
Prof. Ernesto Damiani

Anno Accademico 2010/11

Aknowledgements

I wish to thank you...

Alberto Bertoni and Giorgio Valentini for their invaluable support;

Matteo Re for his pleasant presence,
and for aid that he gave me;

my parents for their peerless help;

my girlfriend Maryam for being by my side each moment;

my family,
my sister, my nephew and my niece.

Dedicated to my grandparents.

Contents

Abstract

The Gene Function Prediction (GFP) problem consists in inferring biological properties for the genes whose function is unknown or only partially known, and raises challenging issues from both a machine learning and a computational biology standpoint.

The GFP problem can be formalized as a semi-supervised learning problem in an undirected graph. Indeed, given a graph with a partial graph labeling, where nodes represent genes, edges functional relationships between genes, and labels their membership to functional classes, GFP consists in inferring the unknown functional classes of genes, by exploiting the topological relationships of the networks and the available a priori knowledge about the functional properties of genes.

Several network-based machine learning algorithms have been proposed for solving this problem, including Hopfield networks and label propagation methods; however, some issues have been only partially considered, e.g. the preservation of the prior knowledge and the unbalance between positive and negative labels.

A first contribution of the thesis is the design of a Hopfield-based cost sensitive neural network algorithm (*COSNet*) to address these learning issues. The method factorizes the solution of the problem in two parts: 1) the subnetwork composed by the labelled vertices is considered, and the network parameters are estimated through a supervised algorithm; 2) the estimated parameters are extended to the subnetwork composed of the unlabeled vertices, and the attractor reached by the dynamics of this subnetwork allows to predict the labeling of the unlabeled vertices.

The proposed method embeds in the neural algorithm the "a priori" knowledge coded in the labelled part of the graph, and separates node labels and neuron states, allowing to differentially weight positive and negative node labels, and to perform a learning approach that takes into account the "unbalance problem" that affects GFP.

A second contribution of this thesis is the development of a new algorithm (*LSI*) which exploits some ideas of *COSNet* for evaluating the predictive capability of each input network. By this algorithm we can estimate the effectiveness of each source of data for predicting a specific class, and then we can use this information to appropriately integrate multiple networks by weighting them according to an appropriate integration scheme.

Both *COSNet* and *LSI* are computationally efficient and scale well with the dimension of the data.

COSNet and *LSI* have been applied to the genome-wide prediction of

gene functions in the yeast and mouse model organisms, achieving results comparable with those obtained with state-of-the-art semi-supervised and supervised machine learning methods.

Introduction

Functional annotation of genes is an important goal in post-genomics research. However, despite the many recent technological advances that have allowed the production of various types of molecular data at a genome-wide scale, the function of large numbers of genes in fully sequenced genomes still remains unknown. This is true even for six of the most-studied model species, *S. cerevisiae, C. elegans, D. melanogaster, A. thaliana, M. musculus, H. sapiens* in which the proportion of genes whose functions are unknown varies between around 10% for S. cerevisiae and around 75% for M. musculus [41]. Accordingly, a fundamental problem in this context is determining an effective and reliable strategy for discovering the biological properties of uncharacterized genes.

In this context, for genes we also mean their products, like proteins and RNAs; accordingly, in the following we will talk mainly about genes, and about proteins when the biochemical, physiological and structural characteristics of the translated proteins are considered. The problem of inferring biomolecular functions for uncharacterized genes is called *Gene Function Prediction problem* (GFP). The input data are usually represented through an undirected weighted graph $G = (V, W)$, where nodes $v \in V$ correspond to instances to be classified (genes), and W defines the weights of the edges according to the "strength" or the evidence of the relationships between pairs of nodes. Moreover, a classification of nodes in positives and negatives is known for a subset $S \subset V$, and the aim is to find a classification for the remaining nodes $U = V \setminus S$.

Many different machine learning approaches have been proposed for GFP, including decision trees [10, 12], kernel based methods like SVMs [14, 15] and module-assisted approaches [42, 43, 49]. In particular, in recent years several network-based approaches have been proposed, in which genes relationships are represented through a network whose nodes represent genes and whose edges represent the detected similarities between genes [25, 26, 31, 32].

The first and simplest algorithms proposed were based on "guilt-by-association" methods, by which unlabeled nodes are set according to the majority or the weighted majority of the labels in their neighborhoods [26, 27]. By extending this approach, nodes can "propagate" their labels to their neighbors iteratively by repeating this "label propagation" process until convergence [33, 34, 35]. Algorithms based on the evaluation of the functional flow in graphs [25, 32], on Markov [62] and Gaussian Random Fields [35, 36] have been applied to the prediction of gene functions in biological networks.

Hopfield networks [87] shares common elements with label propagation algorithms. Indeed labels are iteratively propagated across the neighbors of each node and a quadratic cost function related to the consistency of the labeling of the nodes w.r.t. the network topology is minimized by the network dynamics. From this standpoint Hopfield networks and most of the proposed network-based algorithms for the prediction of node labels can be cast into a recently proposed common framework where a quadratic cost objective function is minimized [38]. Nevertheless, there are some issues that have been only partially considered in classifying networked data.

Many of the network-based approaches do not preserve prior information coded in nodes labeling, providing as output labeling that might be "inconsistent" with the initial labeling. Moreover, traditional machine learning algorithms usually suffer a decay in performance when applied on highly unbalanced data sets [70, 71], and the problem of learning in presence of unbalanced data is the so called *Class Imbalance Problem*.

The class imbalance problem has been tackled by sampling strategies, that is either the minority class is oversampled or the majority class is undersampled or some combination of both [73], and by Cost-Sensitive strategies [72, 74]. In the *GFP* context, in order to address this problem some simple class rescaling have been proposed so that the respective weights over unlabeled examples match the prior class distribution estimated from labeled examples [33, 36], and very few cost-sensitive approaches including a Gaussian field approach [36] and Bayesian hierarchical ensemble method [75].

Finally, many approaches based on neural networks do not distinguish between the node labels and the values of the neuron states [31], thus resulting in a lower predictive capability of the network.

To address these issues, a first contribution of the thesis is a cost-sensitive neural algorithm (*COSNet*), based on Hopfield neural networks, whose main characteristics are the following:

1. Available a priori information is embedded in the neural network and

preserved by the network dynamics.

2. Labels and neuron states are conceptually separated. In this way a class of Hopfield networks is introduced, having as parameters the values of neuron states and the neuron thresholds.

3. The parameters of the network are learned from the data through an efficient supervised algorithm, in order to take into account the unbalance between positive and negative node labels.

4. The dynamics of the network is restricted to its unlabeled part, preserving the minimization of the overall objective function and significantly reducing the time complexity of the learning algorithm.

COSNet has been validated with a genome-wide/ontology-wide prediction in the *S.Cerevisiae* model organism, by considering the FunCat ontology [6] and five different types of yeast biomolecular data. The results shows that the proposed algorithm achieves significantly better results in terms of F-score (Wilcoxon signed-ranks test at 10^{-15} significance level) than other state-of-the-art methods for GFP problem when high imbalanced data are considered.

Each network used for validating *COSNet* describes a different type of relationship between genes, ranging from genetic or physical interactions to gene expression correlation. Unfortunately, single biological networks often cover just a restricted set of proteins and are able to capture only a partial view of their properties. Moreover, a network might be highly predictive for a functional class but not for the others. Therefore, another open challenge and fundamental aspect in *GFP* is the appropriate integration of different data sources to construct high-coverage high-reliability networks. To this aim, an efficient and effective automatic strategy to weight each single source and to evaluate its reliability is needed.

Several approaches have been proposed to deal with this topic, e.g. functional linkage networks integration [64], kernel fusion [66], vector space integration [61], and ensemble systems [89].

A second contribution of this thesis is the design of a new algorithm (*LSI*) which exploits some ideas of *COSNet* for evaluating the predictive capability of each input network. Each network is associated with a classification problem into a two-dimensional space: each labeled node, representing a characterized biological entity of the network, becomes an instance to be classified. By applying an efficient linear classifier to the transformed

two-dimensional data, we obtain a measure of the linear separability that is used to weight the corresponding network. The found weights can then be used to integrate the input networks by adopting an appropriate integration scheme.

We applied the proposed method to the integration of multiple sources of biomolecular data to predict the functional classes of genes in the yeast and mouse model organisms at genome-wide level, using the FunCat and Gene Ontology [5] ontologies. The results of the performance comparison with other state-of-the-art methods shows the effectiveness of the proposed approach.

In Chapter 1 we introduce the preliminary notions that characterize the GFP context and describe the main machine learning approaches proposed in literature for predicting gene functions.

In Chapter 2 we describe the algorithm *COSNet* (*COst Sensitive neural Network*), its computational complexity and its validation on yeast organism. Moreover, we also report the result relative to the comparison of *COSNet* with the several algorithms proposed in the literature.

Chapter 3 is devoted to the description of LSI (*Linear Separability Integration*), a new algorithm for finding network reliability weights to integrate different biomolecular sources, and to the relative experimental analysis on two model organism (yeast and mouse).

The conclusions end the thesis.

Chapter 1

Background

In this Chapter we provide the preliminary notions related to the problem of predicting functions of genes, and we describe the main machine learning algorithms proposed for the GFP problem and for integrating different biological data sources.

1.1 Gene Function Prediction: basic notions

After the publication of the first complete genome sequence in 1995, the bacterium *Haemophilus influenzae* Rd [1], followed by the first full eukaryote genome, the baker's yeast, released in 1996 [2], many genome sequencing projects have been completed, including human genome sequencing (2001) [3, 4]. This have provided researchers with a large amount of new biomolecular data (e.g. genes/proteins), and it has facilitated the transition from molecular genetics, the study of single genes, to genomics, which involves the study of genotype, transcriptome and proteome in a genome-wide scale.

Many of the discovered proteins remain uncharacterized, i.e. we have no information about their function, about their location in the cell and about which proteins they interact with. In this context, an open challenge is determining the function of DNA at the levels of genes, RNA transcripts and gene functions. Recent studies try to ascertain the biochemical, cellular and physiological properties of DNA, including genes and nongenic elements, with a genome-wide approach.

This part of genomics has been coined functional genomics, and has spanned a whole generation of technologies and databases to provide data necessary to make inferences characterized by high reliability. Analysis of these data is now the key problem and we need computational techniques

which can scale to the size of the problems and are able to extract knowledge from the data. Due to the nature of the newly available large-scale data obtained from novel high-throughput technologies, networks of genes are commonly used for describing the different biological sources. In these networks nodes represent genes and edges represent the detected similarities between genes. For example, protein-protein interaction (PPI) measurements have created large-scale data on protein interaction across human and many model species. Accordingly, gene networks have been largely used for studying biomolecular properties of uncharacterized genes by exploiting their topological relationships with already studied genes.

In this context, we first discuss the concept of gene function, then we develop and apply machine learning methods to infer functions for uncharacterized genes.

The concept of gene function can be seen from different points of view: it can be described as the chemical activity of its products, or as those partners it interacts with, or where in the cell (or outside the cell) its products linger. Functional ontologies represent an attempt at defining gene functions by considering all these aspects.

1.1.1 Functional Ontologies

In order to support gene function definition, a number of structured ontologies has been developed, for example gene ontology (GO) [5], and FunCat (Functional Categories) [6].

The GO ontology covers three domains: *molecular function*, *biological process* and *cellular component*. Each domain is represented by a directed acyclic graph (DAG) where nodes (terms) describe a particular aspect of a molecular function, biological process or cellular component at different levels of generality, and whose edges denote relations between nodes, such as *is-a* or *part-of* relations.

The FunCat hierarchy instead is a simpler ontology represented by a forest of trees, where again nodes represent the biomolecular properties of genes and the edges represent hierarchical relationships between nodes.

An *annotation* in this context is simply an association between a gene or gene product and a term. For both GO and FunCat, annotations follow the *true path rule*: when a gene is annotated to an ontology term, it has to be annotated to all the nodes on paths from this term to the root. Accordingly, the main purpose is finding the correct set of functions for a given gene, possibly reaching the leaf nodes in the ontology. Usually, a gene annotated to a term (class) is considered a "positive example" for that term (class).

1.1.2 Gene/protein Function Prediction Problem

Gene function prediction (GFP) in its general formulation is a complex classification problem characterized by the following items:

- each gene can be assigned to multiple classes in a multiclass, multilabel classification scenario;
- classes are structured according to a predefined hierarchy;
- classes are usually unbalanced, with positive examples usually much less than negatives, where a negative example is an instance not annotated to the considered class and annotated to at least another class;
- multiple sources of data can be used to predict gene functions.

"Flat" approaches for *GFP* consider one class at a time, and they do no take into account the decisions for the other classes. Because of the true path rule, ontology-wide predictions with a flat approach may introduce inconsistencies in parent-child relationships between classes, and a hierarchical approach may correct flat predictions in order to improve the accuracy and the consistency of the overall annotations of genes [8]. In this work we just concentrate on flat approaches for binary classification, relying on future works the extension to hierarchical methods.

1.2 Machine Learning Methods for GFP

Due to the huge amount of data in modern databases, machine learning methods have been introduced in the last years for dealing with gene function prediction problem. In the following we briefly describe some approaches proposed in the past decade for GFP, focusing mainly on network-based methods.

Decision tree-based methods. Decision trees classifiers (DTC) are supervised algorithms which recursively partition the data based on its attributes, until some stopping condition is reached [9]. This recursive partitioning gives rise to a tree-like structure. The decisions represent the internal nodes of the tree and they are usually simple attribute tests, using one attribute at a time to discriminate the data. New data can be classified by tracing the path from the root node to each leaf node in the tree.

DTCs are very efficient even with large volumes of data. This is due to the partitioning nature of the algorithm, each time working on smaller and

smaller pieces of the dataset and the fact that they usually only work with simple attribute-value data which is easy to manipulate. DTCs vary both on the criterion for choosing the best attribute for the decision at each node and on their stopping and pruning criteria, i.e. how they decide when to stop growing the tree and how far back to prune it afterwards.

Moreover, DTCs have been used also for a hierarchical multilabel classification, e.g. for predicting gene functions [10], by extending the classical C4.5 decision tree algorithm for multiclass classification [11]. In particular, in [12] was shown that separate decision tree models are less accurate than a single decision tree trained to predict all classes at once.

Kernel methods. Kernels methods are based on measures of similarity called kernel functions that allow us to perform classification, regression and related tasks [13]. They work implicitly by mapping input data into a (usually) higher-dimensional feature space and by finding a suitable hypothesis in this feature space. In the case of classification, this hypothesis is a hyperplane in feature space which separates two classes of input data; new data points can then be classified into one of these two classes, depending on the half-space they are located in. This so-called Support Vector Machine (SVM) classifier maximizes the margin, i.e. the minimum distance between the hyperplane and data points from both classes.

The transformation from input space into feature space shows three main advantages: first, two classes of data points that are not linearly separable in input space can become linearly separable if mapped into an adequate feature space. Second, all calculations in feature space can be performed implicitly via evaluating a kernel function on data points in input space; this kernel function is a dot-product in feature space and represents a measure of similarity between data points. Third, any type of data can be classified, as long as a kernel function can be defined on it. Kernels have been developed for data types such as vectors, graphs, trees and strings. Several kernel functions can be combined into one joint kernel which integrates several sources of information.

The enormous potential of kernel methods in gene function prediction is reflected by a large number of publications over recent years. One common approach is to define a kernel on genes to quantify their similarity based on certain characteristics, e.g. their sequences, structures, chemical features, special amino acid motifs or phylogenetic profiles. Cai *et al.* [14] represent proteins as feature vectors comprising approximate chemical characteristics of their sequences. Dobson and Doig [15] describe protein structures as feature vectors including information about molecular surface, secondary

structure, ligands, bonds and surface clefts. Borgwardt *et al.* [16] integrate both sequence and structure information into one graph model of proteins that is further enriched by approximate chemical properties. In all three studies, representations of proteins are then classified into functional classes using SVMs.

Vert [17] proposes a tree kernel to analyze phylogenetic profiles by incorporating knowledge about the phylogenetic relationship among species. Via SVMs and other kernel methods, Vert then detects functional relationships based on these profiles. Ben Hur and Brutlag [18] define kernels based on discrete functional motifs, i.e. proteins are deemed similar if they share sequence patterns that have been found to be associated with certain functions.

Kernel methods for structured output spaces. In this framework the multilabel hierarchical classification problem is solved globally: the multilabels are viewed as elements of a structured space modeled by suitable kernel functions [19, 20, 21]. In particular, given a feature space $\mathcal{X}$ and a space of structured labels $\mathcal{Y}$, the task is to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ by an induced joint kernel function k that computes the "compatibility" of a given input-output pair (x, y): for each test example $x \in \mathcal{X}$ we need to determine the label $\overline{y} \in \mathcal{Y}$ such that $\overline{y} = \arg\max_{y \in \mathcal{Y}} k(x, y)$ [22].

A structured Perceptron, and a variant of the structured support vector machine [20], have been implemented in the *GOstruct* system and successfully applied to the prediction of GO terms in mouse and other model organisms [23]. Structured output maximum-margin algorithms have been also applied to the tree-structured prediction of enzyme functions [19, 24].

Network-based methods. Also known in literature as label propagation methods or functional association or linkage networks, these methods usually represent each dataset through an undirected graph $G = (V, E)$, where nodes $v \in V$ correspond to genes and edges $e \in E$ are weighted according to the evidence of co-functionality implied by data source [25, 26]. By exploiting "proximity relationships" between connected nodes, these algorithms are able to transfer annotations from previously annotated (labeled) nodes to unannotated (unlabeled) ones through a learning process inherently transductive in nature.

Label propagation algorithms adopt different strategies to learn the unlabeled nodes. Simple "guilt-by-association" methods [27, 28] are based on the idea that nodes lying closer to one another in the network are more likely to have the same biological function; accordingly they transfer to a

node the function most common in its direct neighborhood [29], or among nodes within a particular radius [30]. Other approaches extends guilty-by-association principle by exploiting the global topological structure of the interaction network with the optimization of a global criterion. Vazquez *et al.* [25] aim at assigning functions to nodes in the network by minimizing the number of edges connecting nodes with different functions. This optimization problem, which generalizes the computationally hard problem of minimum multiway cut, has been heuristically solved using simulated annealing. Karaoz *et al.* [31] face the same problem by handling each function at a time. For each function they build a Hopfield network whose energy function correspond to a global consistency criterion and whose connections are weighted using gene expression data. The dynamic of the network locally minimizes the energy ensuring a solution with value at least half of the optimum.

A related approach proposed by Nabieva *et al.* [32] introduced the concept of "network flow" in order to take into account both local and global effects. By considering one function at a time, they treat each annotated node as source of the "functional flow", and the weight of each edge as its capacity. Then they simulate the spread of functional flow by using an iterative algorithm and assign to each unannotated node a score for having the function based on the amount of flow it received during the simulation.

Due to the inherent dependency on the neighborhood structure, approaches based on Markov Random Fields have been proposed by Deng *et al.* [62]. In order to define the probability for a gene to be assigned with a certain function, the authors build a model based on the assumption that the function of a protein is independent of all other proteins given the functions of its immediate neighbors. The model parameters are estimated using quasi-likelihood method and setting a particular combination of parameter values yields the optimization criterion used by Karaoz *et al.* [31].

Another related approach is a label propagation method based on Gaussian kernel proposed by Zhu and Ghahramani [33], in which they iteratively propagate labels from labeled nodes to unlabeled ones until convergence. During the propagation the initial labeling is preserved. Zhou *et al.* [34] modify this approach by allowing labeled nodes to change their initial label, with the addition of a penalty term to ensure consistency with initial labeling. Tsuda *et al.* [35] used this approach to integrate multiple networks. By minimizing the same quadratic cost criterion, they select the most informative networks and at the same time infer prediction scores for unlabeled nodes. Finally, Mostafavi *et al.* [36] introduced two different optimization problems for predicting scores and integrating multiple networks for a single

functional class and for groups of related classes [37].

Bengio *et al.* [38] showed that different graph-based algorithms, including those described above, can be cast into a common framework where a quadratic cost objective function is minimized. In this framework closed form solutions can be derived by solving a linear system of size equal to the cardinality of nodes (proteins), or using fast iterative procedures such as the Jacobi method [39].

Finally, a network-based approach, alternative to label propagation and exhibiting strong theoretical predictive guarantees in the so-called mistake bound model, has been recently proposed by [40].

Module-assisted methods. We want also to briefly mention some approaches not based solely on the guilty by association rule. As summarized by Sharan *et al.* [41], module-assisted approaches first cluster the network into modules according to some properties and then annotate genes inside a module based on known annotations of other genes in the module. The module detection phase can be based just on topological information [42], on hierarchical clustering approaches [43, 44], or on graph clustering methods [46, 45, 47]. Moreover, several approaches have not tried to find new complexes, but just to predict new members for partially known gene complexes [49, 48, 50]

Once the modules have been identified, simple strategies can be applied for assigning functions within the module, e.g. assigning to each gene in the module the functions shared by the majority of genes.

Hierarchical ensemble methods. These methods attempt to take advantage of the intrinsic hierarchical nature of GFP, explicitly considering the relationships between functional classes [12, 51, 52, 53, 54]. Hierarchical ensemble methods generally work via a two-step strategy:

1. Flat learning of the protein function on a per-term basis (a set of independent classification problems)

2. Combination of the predictions by exploiting the relationships between terms that govern the hierarchy of the functional classes.

In principle, any supervised learning algorithm can be used for step 1. Step 2 requires a proper combination of the predictions made at step 1.

Based on this algorithmic scheme, Barutcuoglu et al. [55] proposed an ensemble algorithm that initially provides flat (possibly inconsistent) predictions for each class, and then combine them through a Bayesian network

scheme acting as a "collaborative" error-correction step over all nodes. As an extension of this approach, two local strategies that take into account the relationships between GO nodes and a composite ensemble method have been proposed [106]. Different strategies to hierarchically reconcile the output of an ensemble of learning machines trained to predict separately each GO term have been proposed by [8]: the results demonstrated that hierarchical multilabel methods can play a crucial role in improving gene function prediction performances.

1.3 Data integration methods for GFP

The vast amount and variety of genomic and proteomic data generated from the study of model organisms provides the opportunity to predict the functional properties of genes by using different heterogeneous biological data sources/networks.

When using single sources for predicting gene functions, the predictions often cover just a restricted set of genes and take into account only the limited biomolecular standpoint characterizing that source. Some single sources are also affected by noise, due to the inherent nature of sequencing techniques [57]. Moreover, some sources may be useful for learning a specific functional class, while being irrelevant to others. Clearly all these aspects may lead to unreliable predictions. Therefore, another open challenge and fundamental aspect in GFP is the integration of different data sources to construct high-coverage high-reliability networks.

To this aim, an efficient and effective automatic strategy to weight each single source and to evaluate its reliability is needed. Several works pointed out that data integration plays a central role to improve the accuracy in GFP [58].

The main approaches proposed in the literature can be schematically grouped in four categories [59]:

1. Functional association networks integration
2. Vector subspace integration
3. Kernel fusion
4. Ensemble methods

Functional association networks integration. In functional association networks, different graphs are combined to obtain the composite

resulting network [31, 64]. This network is then processed by a transduction algorithm that assigns all missing labels. The first approaches have been conjunctive/disjunctive techniques [26], that is respectively adding an edge when in all the networks two genes are linked together or when a link between the two genes is present in at least one functional network, and probabilistic evidence integration schemes [63].

More recently, function specific composite networks have been constructed by weighting each data source: Tsuda *et al.* [35] solved this problem by simultaneously optimize the Gaussian Random Fields applied to each data set and the weights associated to each network, while Myers *et al.* [65] construct a combined network by applying a Naive Bayes classifier.

Another network-based approach models data fusion as a constrained linear regression problem [36]. Recently, the same authors showed that better performances can be achieved by optimizing weights on subsets of related GO terms exploiting the relationships between functional classes [37].

Vector Space Integration. In vector space integration vectorial data are concatenated to combine different data sources [7]. For instance, [61] concatenate different vectors, each one corresponding to a different source of genomic data, in order to obtain a larger vector that is used to train a standard SVM. A similar approach has been proposed in [106], but they separately normalized each data source in order to take into account the data distribution in each individual vector space.

Kernel fusion. Thanks to the closure property with respect to the sum and other algebraic operators, kernels provide another valuable research direction for the integration of biomolecular data. Besides combining kernels linearly with fixed coefficients [61], one may also use semi-definite programming to learn the coefficients [66]. As methods based on semi-definite programming do not scale well to multiple data sources, more efficient methods for multiple kernel learning have been recently proposed [67, 68]. Kernel fusion methods, both with and without weighting the data sources, have been successfully applied to the classification of gene functions [60, 69, 104]

Ensemble methods. Ensemble methods use multiple models to obtain predictive performances better than those that could be obtained from any of the constituent models. Even if it seems quite natural to apply ensemble methods to genomic data fusion [59], only a few ensemble methods have been so far applied to this task. Some examples include "late integration" of kernels trained on different sources [61], Naive Bayes integration

of the outputs of SVMs trained with multiple sources [106], and logistic regression for combining the output of several SVMs trained with different biomolecular data and kernels [8].

Chapter 2

COSNet: a cost sensitive algorithm for gene function prediction

2.1 Introduction

Many of the network-based methods described in Chapter 1 do not preserve prior information coded in the initial labeling, i.e. they can change the initial labeling for known genes [31, 34], and are unable to effectively predict node labels when data are unbalanced, that is when a category has much more examples than the other one [31, 33, 35]. In this chapter we cope with these learning issues by developing a new cost-sensitive neural algorithm (*COSNet*), based on parametrized Hopfield networks, which deals with the class imbalance problem and better exploits the prior knowledge.

In Section 2.2 the gene function prediction problem is formalized as a semi-supervised learning problem, then Hopfield networks and the main issues related to this type of recurrent neural network are discussed in Sections 2.3 and 2.4. In Section 2.5 we introduce the concept of "restriction" of network dynamics to a subset of nodes and analyze how it can be used either for preserving the prior information or for reducing time complexity, whereas our algorithm *COSNet* (*COst Sensitive neural Network*) is discussed in Section 2.6. In the same section we show that *COSNet* covers the main Hopfield networks learning issues, and in particular a statistical analysis highlights that the network parameters selected by *COSNet* lead to significantly lower values of the energy function w.r.t. the non cost-sensitive version of the Hopfield network. Finally, in Section 2.7 we describe the experimental validation

of *COSNet* in a genome-wide ontology-wide context for a model organisms in a classical unbalanced semi-supervised classification.

2.2 Gene Function Prediction as a semi-supervised learning problem

The gene function prediction problem (GFP) can be formalized as a semi-supervised learning problem in graphs [38]. The input of the problem consists of a set of genes V and their pairwise similarities w_{ij} deduced by specific biomolecular analysis. Moreover, for a given functional class c, each gene may belong or not to c: this provides a labeling of genes with + (positive example) and - (negative example). Nevertheless, this labeling is known just for a subset S of genes (annotated genes), and the problem is finding a labeling for the remaining genes $U = V \setminus S$ (unannotated genes).

More formally, the input to GFP problem consists in a quadruple $\langle V, \boldsymbol{W}, S, \Sigma^c \rangle$ where:

1. $V = \{1, 2, \ldots, n\}$ set of genes
2. $W : V \times V \longrightarrow \mathbb{R}$ symmetric matrix of gene similarities
3. $S \subset V$ set of annotated genes
4. $\Sigma^c : S \longrightarrow \{+, -\}$ labeling function according to class c

The GFP problem consists in extending the function Σ^c to V, i.e. finding a labeling function $\Psi^c : V \longrightarrow \{+, -\}$ such that $\Psi^c(i) = \Sigma^c(i)$ for each $i \in S$.

For what concerns the underlying model, we suppose that the function Ψ^c is fixed but is unknown, that the subset S of fixed size is uniformly drawn from V and that the restriction $\Psi^c|_S$ of Ψ^c to S is known. The aim is reconstructing the function Ψ^c relying on the prior knowledge $\langle W, \Sigma^c \rangle$.

Observe that knowing the restriction $\Psi^c|_S$ means having a bipartition of S in positive S^+ and negative S^- examples. Moreover, extending the function Σ^c to V means finding a bipartition (U^+, U^-) of genes in $U = V \setminus S$. Genes in U^+ are then considered candidates for the class c.

From this standpoint, GFP is set as a semi-supervised learning problem on graphs, since gene functions can be predicted by exploiting both labeled (annotated) and unlabeled (unannotated) nodes/genes and the weighted connections between them.

Finally, we point out that we assume that the prior knowledge is not affected by noise. This assumption simplify the model, nevertheless it is worth observe that in both GO and FunCat ontologies negative examples for a class c are simply genes that are not annotated for c, and may correspond

to true negatives or to false negatives due to lack of knowledge about their biological function.

2.3 Hopfield Networks

A discrete Hopfield network (DHN) [87, 88] is a dynamic system made up by units named neurons. Each neuron i has an activation threshold γ_i and each pair of neurons i and j is connected by an edge whose strength is w_{ij}, with $w_{ij} = w_{ji}$. Figure 2.1 shows a DHN with five neurons. The activation

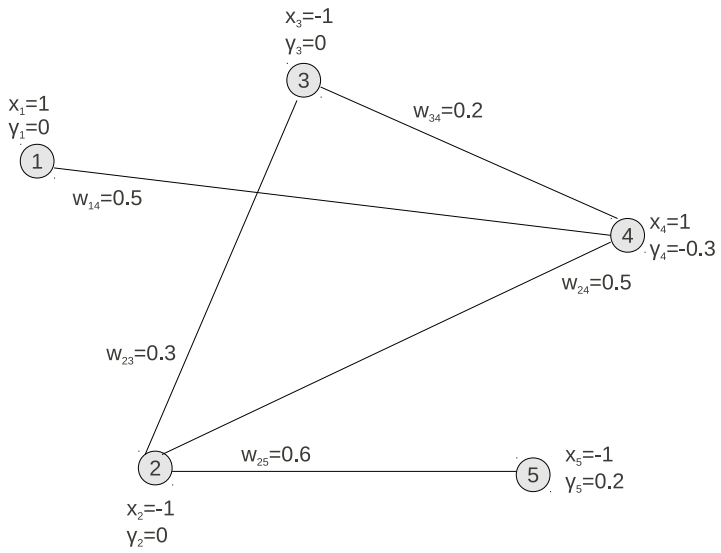


Figure 2.1: An example of DHN with five neurons and activation values 1, -1.

values of neurons are 1 (firing) and -1 (not firing) (or 1 and 0); during the network dynamics, at each discrete time $t \in \{0, 1, \cdots\}$ each neuron i has an activation value $x_i(t) \in \{1, -1\}$ (or $x_i(t) \in \{1, 0\}$). When the dynamics is asynchronous, the update rule at time $t+1$ for neuron i is

$$x_i(t+1) = \begin{cases} 1 & \text{if } \sum_{j=1}^{i-1} w_{ij} x_j(t+1) + \sum_{k=i+1}^{n} w_{ik} x_k(t) - \gamma_i > 0 \\ -1 & \text{if } \sum_{j=1}^{i-1} w_{ij} x_j(t+1) + \sum_{k=i+1}^{n} w_{ik} x_k(t) - \gamma_i \leq 0 \end{cases} \tag{2.1}$$

where n is the number of neurons and the update order is chosen randomly.

The state of the network at time t is $\boldsymbol{x} = (x_1(t), x_2(t), \cdots, x_n(t))$. The main feature of a Hopfield network is the existence of a quadratic state

function named *energy function*:

$$E(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^T \boldsymbol{W}\boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{\gamma} \tag{2.2}$$

where $\boldsymbol{W} = (w_{ij})$ and $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_n)$. This is a non increasing function w.r.t. the evolution of the network according to the activation rule (2.1), i.e.

$$E(\boldsymbol{x}(0)) \geq E(\boldsymbol{x}(1)) \geq \ldots \geq E(\boldsymbol{x}(t)) \geq \ldots$$

It is easy to show that every dynamics of the network converges to an equilibrium state $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$, where, by updating each neuron i, the value $\hat{x}_i$ does not change for any $i \in \{1, 2, \ldots, n\}$ [87]. In this sense a DHN is a local minimizer of the energy function, and $\hat{\boldsymbol{x}}$ is also called "attractor" of the dynamics.

It is worth noting that the state of a Hopfield network $(x_1, x_2, \cdots, x_n)$ denotes the subset (pattern) of neurons $\{i \mid x_i = 1\}$. This notion is independent of the activation values: for example the network states (1, -1, -1, 1) and (1, 0, 0, 1) denote the same pattern $\{1, 4\}$. Nevertheless, fixed the connections weights and the neuron thresholds, the dynamics with activation values $\{1, -1\}$ and $\{1, 0\}$ are generally different. Accordingly, it is possible to conceptually separate patterns and activation values of neurons.

We consider a new model of Hopfield network in which the activation values and thresholds are parameters. In particular we adopt $\{\sin\alpha, -\cos\alpha\}$ as neuron activation values, where $\alpha \in]0, \pi[$. Observe that for $\alpha = \frac{\pi}{2}$ we obtain the activation values $\{1, 0\}$, whereas for $\alpha = \frac{\pi}{4}$, up to a constant, we obtain the activation values $\{1, -1\}$. Below we give a formal definition of this network.

A *parametrized Hopfield network* H with neurons $V = \{1, 2, \ldots, n\}$ is a triple $H = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \alpha \rangle$, where :

- $\boldsymbol{W}$ is a $n \times n$ symmetric matrix in which $w_{ij} \in \mathbb{R}$ is the connection strength between neurons i and j, with $w_{ij} = w_{ji}$ for each i and j;
- $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ is variable in $\mathbb{R}^n$ whose assignment is a vector of activation thresholds
- α is a variable in $]0, \pi[$ whose assignment determines the two different neuron activation values $\{\sin\alpha, -\cos\alpha\}$.

In the following we will refer to neuron activation values also as neuron states or neuron values.

Fixed an assignment of α and $\boldsymbol{\gamma}$, the dynamics of the network is described as follows:

1. At time 0 an initial value $x_i(0) = a_i$ is given for each neuron i
2. At time $t+1$ each neuron is updated asynchronously (up to a permutation) by the following activation rule

$$x_i(t{+}1) = \begin{cases} \sin\alpha & \text{if } \sum_{j=1}^{i-1} w_{ij}x_j(t+1) + \sum_{k=i+1}^{n} w_{ik}x_k(t) - \gamma_i > 0 \\ -\cos\alpha & \text{if } \sum_{j=1}^{i-1} w_{ij}x_j(t+1) + \sum_{k=i+1}^{n} w_{ik}x_k(t) - \gamma_i \leq 0 \end{cases} \tag{2.3}$$

This dynamics is guaranteed reaching an equilibrium state $\hat{\boldsymbol{x}}$ which corresponds to a minimum of the energy function and which identifies the final pattern [87].

In this thesis, we simplify the model of parametrized Hopfield network by requiring that all the neurons have the same activation threshold, i.e. $\gamma_1 = \gamma_2 = \cdots = \gamma_n = \gamma$. As a consequence, we obtain a model with 2 real parameters α and γ.

The algorithm we propose for solving the GFP problem consists of two main steps. In the first step we learn the values of the parameters α, γ by exploiting the prior knowledge (Section 2.2); in the second step we simulate the network dynamics in order to obtain the solution of the problem as equilibrium state pattern.

2.4 Learning Issues in Hopfield Networks for GFP

Hopfield networks have been used in many different contexts including binary classification, and in particular Karaoz et al. proposed a method based on DHNs named *GAIN* (Gene Annotation using Integrated Networks) for GFP problem [31]. Considered the semi-supervised set-up described in section 2.2, with the set of genes V bi-partitioned into the sets S and U of annotated and unannotated genes, the authors consider a different DHN for each functional term c, by setting the thresholds to 0 and the neuron values to 1 and -1. The initial state of the network $\overline{\boldsymbol{x}} = (\overline{\boldsymbol{u}}, \overline{\boldsymbol{s}})$, where we emphasize the state $\overline{\boldsymbol{u}}$ of neurons in U and $\overline{\boldsymbol{s}}$ of neurons in S, is:

$$\overline{x}_i = \begin{cases} 0 & \text{if } \ i \in U \\ +1 & \text{if } \ i \in S^+ \\ -1 & \text{if } \ i \in S^- \end{cases} \tag{2.4}$$

for each $i \in \{1, 2, \ldots, n\}$, and at each discrete time $t+1$ each neuron is updated asynchronously in a random order by the following update rule

$$x_i(t+1) = \text{Sgn}\left(\sum_{j=1}^{i-1} w_{ij} x_j(t+1) + \sum_{k=i+1}^{n} w_{ik} x_k(t) \right). \tag{2.5}$$

The initial value 0 corresponds to an uncertain condition, and the aim is to change it to -1 or 1 during the network dynamics. The equilibrium state $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{s}})$ determines the bipartition of U as follows: the gene $i \in U$ belongs to U^+ if $\hat{x}_i = 1$ and to U^- if $\hat{x}_i = -1$. In other words, a final value $\hat{x}_i = 1$ is a clue that the gene i may be annotated with c.

From a biological standpoint, this approach is motivated by the fact that minimizing the overall energy (2.2) means maximizing the weighted sum of edges connecting neuron with the same activation value, since each pair neurons with the same value gives a negative contribution to the energy.

This approach leads to three main drawbacks:

1. *Preservation of the prior knowledge.* The network dynamics updates also neurons which are already labeled, that is the network evolution can change the initial states of neurons in S, and the available prior information coded in the bipartition (S^+, S^-) of S may not be preserved. In other words the original known labeling is not maintained by the algorithm. This happens when the reached state $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{s}})$ is such that $\hat{\boldsymbol{s}} \neq \overline{\boldsymbol{s}}$. Clearly this point is important because we have assumed that the prior information does not contain noise.

2. *Unbalance problem in functional classes.* By assigning the value 1 to positive labels, -1 to those negative and by setting to 0 the threshold of each neuron, when $|S^+| \ll |S^-|$ the network is likely to converge to a trivial state $(-1, -1, \ldots, -1)$. This is exactly the situation registered in both GO and FunCat ontologies, since only a small number of positive annotations is available for most of the functional categories and terms. As a consequence, the unbalance between positive and negative examples requires the adoption of cost-sensitive techniques to avoid predictions biased towards negative examples [36, 89].

3. *Incoherence of the prior knowledge coding.* Since the inference criterion is based on the minimization of the overall objective function, we expect that the initial state $\overline{\boldsymbol{s}}$ of labeled neurons is a subvector of a state $(\overline{\boldsymbol{s}}, \hat{\boldsymbol{u}})$ "close" to a minimum of the energy function. Unfortunately, in many cases this is not true.

To address these problems, we exploit a simple property which holds for sub-networks of a DHN, and that we discuss in the next section.

2.5 Sub-network Property

According to the semi-supervised setting of Section 2.2, let be $H = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \alpha \rangle$ a network with neurons $V = U \sqcup S = \{1, 2, \ldots, n\}$, where up to a permutation, $U = \{1, 2, \ldots, h\}$ and $S = \{h+1, h+2, \ldots, n\}$; each network state $\boldsymbol{x}$ can be decomposed in $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{s})$, where $\boldsymbol{u}$ and $\boldsymbol{s}$ are respectively the states of neurons in U and in S. The energy function of H can be written by separating the contributions due to U and S:

$$E(\boldsymbol{u}, \boldsymbol{s}) = -\frac{1}{2}\left(\boldsymbol{u}^T \boldsymbol{W}_{uu} u + \boldsymbol{s}^T \boldsymbol{W}_{ss} \boldsymbol{s} + \boldsymbol{u}^T \boldsymbol{W}_{us} \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{W}_{us}^T \boldsymbol{u}\right) + \boldsymbol{u}^T \boldsymbol{\gamma}^u + \boldsymbol{s}^T \boldsymbol{\gamma}^s, \tag{2.6}$$

where $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{uu} & \boldsymbol{W}_{us} \\ \boldsymbol{W}_{us}^T & \boldsymbol{W}_{ss} \end{pmatrix}$ is the weight matrix $\boldsymbol{W}$ decomposed in its submatrices $\boldsymbol{W}_{uu}$ connecting nodes in U, $\boldsymbol{W}_{ss}$ connecting nodes in S, $\boldsymbol{W}_{us}$ connecting each node in U with each node in S, and $\boldsymbol{W}_{us}^T$ its transpose, while $\boldsymbol{\gamma} = (\boldsymbol{\gamma}^u, \boldsymbol{\gamma}^s)$ represents the vector of the thresholds for respectively unlabeled ($\boldsymbol{\gamma}^u$) and labeled ($\boldsymbol{\gamma}^s$) nodes.

Suppose now that a state $\tilde{\boldsymbol{s}}$ of neurons in S is given. We are interested in the dynamics obtained by allowing the update just of neurons in U, without updating neurons in S. We denote with $H_{U|\tilde{\boldsymbol{s}}}$ the DHN which is characterized by this dynamics.

Proposition 1. . $H_{U|\tilde{\boldsymbol{s}}} = \langle \boldsymbol{W}_{uu}, \boldsymbol{\gamma}^u - \boldsymbol{W}_{us}\tilde{\boldsymbol{s}}, \alpha \rangle$.

Proof. The overall energy of network H is:

$$E(\boldsymbol{u}, \tilde{\boldsymbol{s}}) = -\frac{1}{2}\boldsymbol{u}^T \boldsymbol{W}_{uu} \boldsymbol{u} + \boldsymbol{u}^T(\boldsymbol{\gamma}^u - \boldsymbol{W}_{us}\tilde{\boldsymbol{s}}) - \frac{1}{2}\tilde{\boldsymbol{s}}^T \boldsymbol{W}_{ss}\tilde{\boldsymbol{s}} + \tilde{\boldsymbol{s}}^T \boldsymbol{\gamma}^s \tag{2.7}$$

Since we imposed that $\tilde{\boldsymbol{s}}$ is constant, the last two terms of equation (2.7) do not change during the subnetwork dynamics. Accordingly, we obtain the energy $E(\boldsymbol{u})$ due to the nodes that can change their states (i.e. the nodes in U):

$$E(\boldsymbol{u}) = -\frac{1}{2}\boldsymbol{u}^T \boldsymbol{W}_{uu} \boldsymbol{u} + \boldsymbol{u}^T(\boldsymbol{\gamma}^u - \boldsymbol{W}_{us}\tilde{\boldsymbol{s}}) \tag{2.8}$$

This energy by definition corresponds to the network $\langle \boldsymbol{W}_{uu}, \boldsymbol{\gamma}^u - \boldsymbol{W}_{us}\tilde{\boldsymbol{s}}, \alpha \rangle$. □

We observe that the network $H_{U|\tilde{\boldsymbol{s}}}$ has the same weights and activation values of H and the influence of the labeled nodes (nodes is S) on nodes in U is now embedded in the thresholds.

Finally, observe that by setting $\boldsymbol{\theta}^u = \boldsymbol{\gamma}^u - \boldsymbol{W}_{us}\tilde{\boldsymbol{s}}$, we obtain the energy of the subnetwork $H_{U|\tilde{\boldsymbol{s}}}$ in the usual form:

$$E(\boldsymbol{u}) = -\frac{1}{2}\boldsymbol{u}^T\boldsymbol{W}_{uu}\boldsymbol{u} + \boldsymbol{u}^T\boldsymbol{\theta}^u \tag{2.9}$$

It holds the following:

Proposition 2. (Sub-network property). *If $\tilde{\boldsymbol{s}}$ is part of a energy global minimum of H, and $\tilde{\boldsymbol{u}}$ is a energy global minimum of $H_{U|\tilde{\boldsymbol{s}}}$, then $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}})$ corresponds to a energy global minimum of H.*

Proof. From the equations (2.6) and (2.8), it follows that

$$E(\boldsymbol{u}, \boldsymbol{s}) = -\frac{1}{2}s^T\boldsymbol{W}_{ss}\boldsymbol{s} + \boldsymbol{s}^T\underline{\boldsymbol{\gamma}}^s + E(\boldsymbol{u}),$$

from which, by assuming $x = (\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}})$ is not a global minimum of E, there exists another state $\hat{\boldsymbol{u}}$ for neurons in U such that $E(\hat{\boldsymbol{u}}, \tilde{\boldsymbol{s}}) < E(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}})$. This means that $E(\hat{\boldsymbol{u}}) < E(\tilde{\boldsymbol{u}})$, which contradicts the hypothesis that $\tilde{\boldsymbol{u}}$ is a global minimum of $E(\boldsymbol{u})$. □

In our setting, we associate the state $\tilde{\boldsymbol{s}} = \boldsymbol{x}(S^+, S^-)$ with the given bipartition (S^+, S^-) of S:

$$x_i(S^+, S^-) = \begin{cases} \sin\alpha & \text{if } \ i \in S^+ \\ -\cos\alpha & \text{if } \ i \in S^- \end{cases}$$

for each $i \in S$. By the sub-network property, if $\boldsymbol{x}(S^+, S^-)$ is part of a energy global minimum of H, we can predict the hidden part relative to neurons U by minimizing the energy of $H_{U|\boldsymbol{x}(S^+,S^-)}$.

2.6 *COSNet*

In this Section we introduce the algorithm *COSNet* (COst-Sensitive neural Network) for dealing with the GFP problem presented in Section 2.2. In particular, our aim is to cope with the learning issues described in Section 2.4: preservation of the prior knowledge, data imbalance management, incoherence of the prior knowledge coding.

The input of the problem, as described in Section 2.2, is represented by $\langle V, \boldsymbol{W}, S^+, S^- \rangle$ and the aim is to extend to V the bipartition of S in positive S^+ and negative S^- examples, i.e. finding a bipartition of $U = V \setminus S$.

We consider the parametrized family $H = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \alpha \rangle$ of Hopfield networks on neurons V and parameters α and $\boldsymbol{\gamma}$. Our assumption is that do exist the couple $(\hat{\alpha}, \hat{\boldsymbol{\gamma}})$ such that the solution of the problem is denoted by the energy global minimum $\hat{x}$ of the Hopfield network $H = \langle \boldsymbol{W}, \hat{\boldsymbol{\gamma}}, \hat{\alpha} \rangle$ and the state $\boldsymbol{x}(S^+, S^-)$ is part of $\hat{x}$. Then, for the sub-network property, we can discover the hidden part $\hat{u}$ of $\hat{x}$ by minimizing the energy of the network $H_{U|\boldsymbol{x}(S^+,S^-)}$.

Since it is not guaranteed that $\boldsymbol{x}(S^+, S^-)$ is part of a energy minimum of H, our aim is finding the parameters $\alpha, \boldsymbol{\gamma}$ such that $\boldsymbol{x}(S^+, S^-)$ is as close as possible to an equilibrium state of the network $H_{S|x(U^+,U^-)}$. Observe that the bipartition (U^+, U^-) necessary for building this sub-network is not known, for this reason we first generate a temporary bipartition of U which resembles the proportion of positive examples in S.

The main steps of *COSNet* can be summarized as follows:

INPUT: a symmetric connection matrix $\boldsymbol{W} : V \times V \longrightarrow [0, 1]$, bipartition (U, S) of V and bipartition (S^+, S^-) of S.

OUTPUT: a bipartition (U^+, U^-) of $U = V \setminus S$.

Step 1. Randomly generate an initial temporary bipartition (U^+, U^-) of U such that $\frac{|U^+|}{|U|} \simeq \frac{|S^+|}{|S|}$.

Step 2. Find the optimal parameters $(\hat{\alpha}, \hat{\boldsymbol{\gamma}})$ of the Hopfield sub-network $H_{S|\boldsymbol{x}(U^+,U^-)}$ such that the state $\boldsymbol{x}(S^+, S^-)$ is "as close as possible" to an equilibrium state.

Step 3. Extend the parameters $(\hat{\alpha}, \hat{\boldsymbol{\gamma}})$ to the whole network and run the sub-network $H_{U|\boldsymbol{x}(S^+,S^-)}$ until an equilibrium state $\hat{\boldsymbol{u}}$ is reached. The

final solution (U^+, U^-) is:

$$U^+ = \{i \in U \mid \hat{u}_i = \sin\hat{\alpha}\}$$
$$U^- = \{i \in U \mid \hat{u}_i = -\cos\hat{\alpha}\}.$$

Below, each step of the algorithm is described in detail.

2.6.1 Generating a Temporary Solution

An initial bipartition of U is needed to build the sub-network $H_{S|\boldsymbol{x}(U^+,U^-)}$. We adopt a procedure that approximately maintains in U the same proportion of positive elements observed in S:

- generate a random number m according to the binomial distribution $B(|U|, \frac{|S^+|}{|S|})$

- assign to U^+ m elements uniformly chosen in U

- assign to U^- the set $U \setminus U^+$.

This criterion comes from the probabilistic model described below.

Suppose that V is biparted in positive and negative elements. We randomly draw a subset $S \subset V$, with $|S| = n - h$, and $|U| = h$. Moreover, we denote with S^+ the set of positive elements in S. Our aim is to infer the most likely cardinality of positive elements in $U = V \setminus S$.

By setting $P(z) = Prob\{|U^+| = z \mid S \text{ contains } |S^+| \text{ positives}\}$, the following equality holds:

$$\frac{|S^+|}{|S|} \cdot h = \underset{z}{\text{argmax}}\ P(z). \tag{2.10}$$

In fact, by setting $n_1 = |S^+|$, $n_2 = n - h - n_1$, and $p_s = \frac{n_1}{n-h}$, the probability $P(z)$ can be written as follows:

$$P(z) = \frac{\binom{n_1+z}{z}\binom{n_2+y}{y}}{\binom{n}{h}},$$

where $y = h - z$. The value of z which maximize $P(z)$ is such that $P(z) \simeq P(z+1)$, that is

$$\binom{n_1+z}{z}\binom{n_2+y}{y} = \binom{n_1+z+1}{z+1}\binom{n_2+y-1}{y-1},$$

from which it follows that $\frac{n_2+y}{y} = \frac{n_1+z+1}{z+1}$. By approximating $z+1$ with z we obtain $\frac{n_2}{y} = \frac{n_1}{z}$; since $n_1 = p_s(n-h)$ and $n_2 = (1-p_s)(n-h)$, it follows

that $z = p_s \cdot h$.

This simple property shows that the probability $P(z)$ is maximized when U has the same proportion of positives in S; accordingly, we generate the number of positive elements in U according the binomial distribution $B(|U|, \frac{|S^+|}{|S|})$. The initial labeling of U contributes to the assessment of the parameters α and $\boldsymbol{\gamma}$ of the network, as explained in the next section.

2.6.2 Finding the Optimal Parameters

The main goal of this step is to find the values of the parameters α and γ of H, such that the state $\boldsymbol{x}(S^+, S^-)$ is "as close as possible" to an equilibrium state. As a first simple approach, we compute a unique activation threshold γ for all the neuron, relying on future studies the analysis of more complex systems with different activation thresholds.

We consider the parametrized sub-network $H_{S|\boldsymbol{x}(U^+,U^-)} = \langle \boldsymbol{W}_{ss}, \boldsymbol{\gamma}^s - \boldsymbol{W}_{us}^T \boldsymbol{x}(U^+, U^-), \alpha \rangle$, where $\gamma_i^s = \gamma$ for each $i \in \{h+1, h+2, \ldots, n\}$, and (U^+, U^-) is the temporary bipartition found in the previous step.

We associate each node k of the network with a labeled point $\Delta(k)$ in the plane, whose coordinates depend on the topology of the network and on the weights of the edges connecting k to its neighbours. More precisely, for each node k in S we define $\Delta(k) \equiv (\Delta^+(k), \Delta^-(k))$, where

$$\Delta^+(k) = \sum_{j \in S^+ \cup U^+} w_{kj}, \qquad \Delta^-(k) = \sum_{j \in S^- \cup U^-} w_{kj}$$

From a computational complexity standpoint, note that the sums extended to $S^+ \cup U^+$ and to $S^- \cup U^-$ can be realized just considering the positive and negative neighbours of k respectively, since when a node j is not neighbour of k we have $w_{kj} = 0$.

The bipartition (S^+, S^-) of S induces in a natural way a bipartition (I^+, I^-) of the points $I = \{\Delta(k), k \in S\}$, where:

$$I^+ = \{\Delta(k), k \in S^+\} \qquad I^- = \{\Delta(k), k \in S^-\}$$

In an analogous way, an arbitrary straight line in the plane of equation $f_{\alpha,\gamma}(z, y) = \cos\alpha \cdot y - \sin\alpha \cdot z + \gamma = 0$ separates the points of I in $I^+_{\alpha,\gamma}$ and $I^-_{\alpha,\gamma}$, where:

$$I^+_{\alpha,\gamma} = \{\Delta(k) \mid f_{\alpha,\gamma}(\Delta(k)) < 0\} \qquad I^-_{\alpha,\gamma} = \{\Delta(k) \mid f_{\alpha,\gamma}(\Delta(k)) \geq 0\}.$$

Figure 2.2 provides a graphical view of the described scenario.

Fixed the parameters (α, γ), we set:

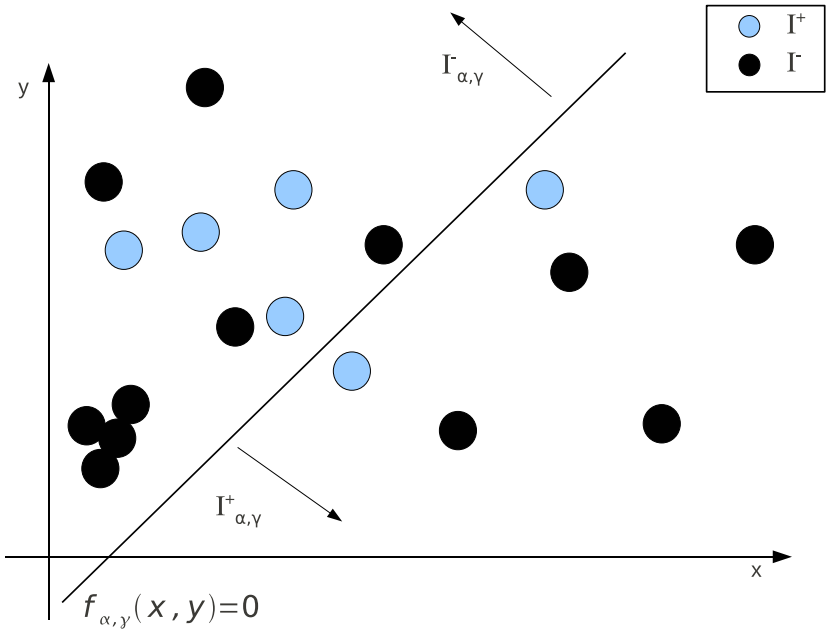


Figure 2.2: Example which shows the sets I^+, I^-, $I^+_{\alpha,\gamma}$ and $I^-_{\alpha,\gamma}$.

- $TP(\alpha,\gamma) = |I^+_{\alpha,\gamma} \cap I^+|$, i.e. the number of positive examples correctly classified by the line $f_{\alpha,\gamma}$

- $FN(\alpha,\gamma) = |I^-_{\alpha,\gamma} \cap I^+|$, i.e. the number of positive examples classified as negative

- $FP(\alpha,\gamma) = |I^+_{\alpha,\gamma} \cap I^-|$, i.e. is the number of negative examples classified as positive

$F_{score}(\alpha,\gamma)$ is the harmonic mean of $precision(\alpha,\gamma) = \frac{TP(\alpha,\gamma)}{TP(\alpha,\gamma)+FP(\alpha,\gamma)}$ and $recall(\alpha,\gamma) = \frac{TP(\alpha,\gamma)}{TP(\alpha,\gamma)+FN(\alpha,\gamma)}$. It holds $0 \leq F_{score}(\alpha,\gamma) \leq 1$ and $F_{score}(\alpha,\gamma) = 1$ iff there are no classification errors.

Observe that we define the positive halfplane below the line, since this choice is strictly related to the following proposition, which expresses the condition of equilibrium state of the network in terms of relation between the bipartitions (I^+, I^-) and $(I^+_{\alpha,\gamma}, I^-_{\alpha,\gamma})$ of I.

Proposition 3. *: The state $\boldsymbol{x}(S^+, S^-)$ is an equilibrium state for the network $H_{S|\boldsymbol{x}(U^+,U^-)}$ if and only if $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$.*

Proof. ($\Rightarrow$). Suppose (α, γ) is a couple such that $\boldsymbol{x}(S^+, S^-)$ is an equilibrium state for the network $H_{S|\boldsymbol{x}(U^+,U^-)}$. It means from eq.(2.3) that

$$\sin\alpha \cdot \sum_{j\in S, s_j=\sin\alpha} w_{ij} - \cos\alpha \cdot \sum_{j\in S, s_j=-\cos\alpha} w_{ij} - {\theta_i}^s > 0 \text{ if } i \in S^+ \qquad (*)$$

$$\sin\alpha \cdot \sum_{j\in S, s_j=\sin\alpha} w_{ij} - \cos\alpha \cdot \sum_{j\in S, s_j=-\cos\alpha} w_{ij} - {\theta_i}^s \leq 0 \text{ if } i \in S^- \qquad (**)$$

where $\theta_i^s = \boldsymbol{\gamma}_i - \sum_{k\in U} w_{ik} x_k(U^+, U^-)$. By setting $\boldsymbol{\gamma}_i = -q\cos\alpha$, the condition (*) is equivalent to $-f_{\alpha,\gamma}(\Delta(i)) > 0$ when $i \in S^+$, that is $f_{\alpha,\gamma}(\Delta(i)) < 0$. This means that $I^+ \subseteq I^+_{\alpha,\gamma}$. Analogously the condition (**) leads to $I^- \subseteq I^-_{\alpha,\gamma}$. Since the number of points in $I^+ \cup I^-$ is equal to the number of neurons in the network $H_{S|\boldsymbol{x}(U^+,U^-)}$, it follows that $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$.

($\Leftarrow$) Suppose there is a couple (α, γ) such that $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$. This means that for each node $k \in S^+$ it holds $f_{\alpha,\gamma}(\Delta(k)) < 0$ (i), and for each node $k \in S^-$ it holds $f_{\alpha,\gamma}(\Delta(k)) \geq 0$ (ii). The condition (i) implies condition (*) for each $k \in S^+$ and the condition (ii) implies condition (**) for each $k \in S^-$. Hence, $\boldsymbol{x}(S^+, S^-)$ is an equilibrium state for the network $H_{S|\boldsymbol{x}(U^+,U^-)}$. □

From Proposition 3, if we can find a linear classifier able to correctly classify points of $I^+ \cup I^-$ we can obtain the optimal parameters (α, γ) of the network. Of course, it is not guaranteed that the points in I are linearly separable, but in any case we can try to find the "best performing" linear classifier, thus obtaining "near-optimal" parameters for the network.

To optimize the parameters (α, γ) we adopt the F-score maximization criterion:

$$(\hat{\alpha}, \hat{\gamma}) = \underset{\alpha,\gamma}{\operatorname{argmax}} \; F_{score}(\alpha, \gamma), \qquad (2.11)$$

which gives to the misclassications of elements in S^+ (false negative) an higher cost than misclassications of elements in S^- (false positive). As mentioned in Section 1.1.2, in GFP most classes are seriously unbalanced, it means that simply adopting the accuracy as objective function can lead to meaningless "all negative" predictions. Moreover, this choice is also supported by the following corollary of the Proposition 3:

Corollary 1. *$F_{score}(\alpha, \gamma) = 1$ iff $x(S^+, S^-)$ is an equilibrium state of $H_{S|\boldsymbol{x}(U^+,U^-)}$.*

Proof. ($\Rightarrow$). If $F_{score}(\alpha, \gamma) = 1$ there are no misclassification errors, i.e $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$. According to the Proposition 3 this means that $x(S^+, S^-)$ is an equilibrium state of $H_{S|\boldsymbol{x}(U^+,U^-)}$.

($\Leftarrow$). Suppose $x(S^+, S^-)$ is an equilibrium state of $H_{S|\boldsymbol{x}(U^+,U^-)}$. The Proposition 3 states that $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$. So there are no misclassification errors during the F-score optimization, which means $F_{score}(\alpha, \gamma) = 1$. □

Although there is an exact algorithm for linearly separating points in I^+ from those in I^- working in time $\mathcal{O}(|S|^2 \cdot \log |S|)$, we adopt a more efficient two-step approximation algorithm `FindOptimalLine`, that at first computes the optimum line (in terms of the F_{score} criterion) among the ones crossing the origin of the axes, and then computes the optimal intercept:

1. *Compute* $\hat{\alpha}$. The algorithm computes the slopes of the lines crossing the origin and each point $\Delta(k) \in I^+ \cup I^-$. Then it searches the line which maximizes the F_{score} criterion by sorting the computed lines according to their slopes in an increasing order. Since all the points lie in the first quadrant, this assures that the angle $\hat{\alpha}$ relative to the optimum line is in the interval $[0, \frac{\pi}{2}[$. Figure 2.3 graphically shows this step.

2. *Compute* $\hat{\gamma}$. Compute the intercepts of the lines whose slope is $\tan \hat{\alpha}$ and crossing each point belonging to $I^+ \cup I^-$. The optimum line is identified by scanning the computed lines according to their intercept in an increasing order. Let $\hat{q}$ be the intercept of the optimum line $y = \tan \alpha \cdot z + q$, then we set $\hat{\gamma} = -\hat{q} \cos \hat{\alpha}$. In Figure 2.4 a graphical description of this step.

Both step 1 and step 2 can be computed in $\mathcal{O}(n \log n)$ computational time (due to the sorting), where n is the number of points.

Fig. 2.5 and 2.6 respectively shows the pseudocode of the procedures `ComputePoints` to compute the points $\Delta(k)$ and `FindOptimalLine` to compute the near-optimal line.

Lines $1 - 3$ of the procedure `FindOptimalLine` compute for each point in $I^+ \cup I^-$ the slope of the lines crossing that point and the origin of the axes.

Line 4 sorts the computed slopes in an increasing order and initialize the number of true positives (TP), false positives (FP) and false negative (FN). In the lines $6 - 12$ the optimum F_{score} and the relative slope (mOpt) are computed by scanning all the lines by increasing slope and updating the values TP, TN, FP according to the label of the current point. The notation x_i represents the i^{th} element of the vector $\boldsymbol{x}$, i.e. $ordInd_s$ represents the s^{th} elements of the vector $ordInd$.

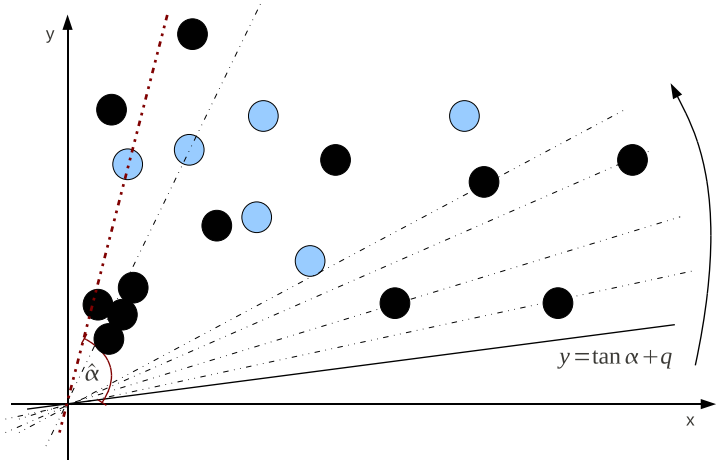


Figure 2.3: Step 1 of the optimization algorithm `FindOptimalLine` which computes the best angle $\hat{\alpha}$.

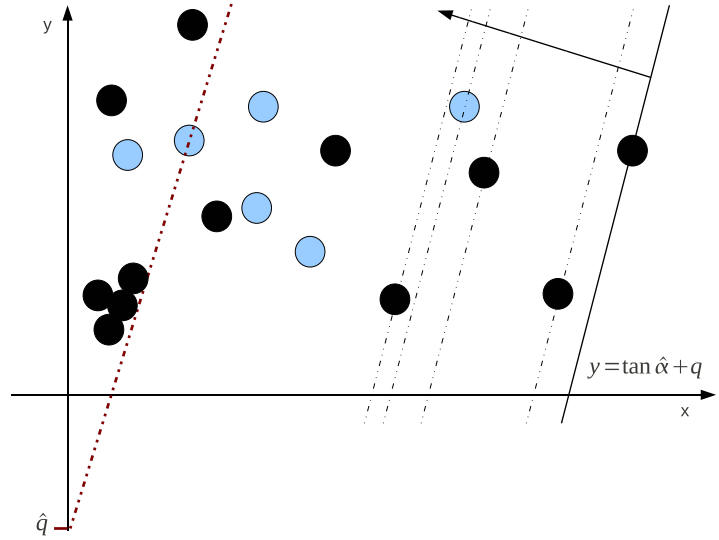


Figure 2.4: Step 2 of the optimization algorithm `FindOptimalLine` which computes the best intercept $\hat{q}$ given the best angle $\hat{\alpha}$.

In the lines $13 - 15$ the intercept for each line with slope mOpt and crossing each point of $I^+ \cup I^-$ is computed and the computed intercepts are sorted in an increasing order (line 16).

Figure 2.5: Pseudocode procedure `ComputePoints`

`Input`:
1) neuron set V; 2) bipartition (U, S) of V
3) connection matrix W
4) bipartition (U^+, U^-) of U; 5) bipartition (S^+, S^-) of S

```
- ComputePoints:begin algorithm
01:    for each k ∈ S do Δ_k^+ ← 0; Δ_k^- ← 0;
02:       for each j ∈ Neighbourhood_k do
03:         if j ∈ U^+ ∪ S^+ then Δ_k^+ ← Δ_k^+ + w_kj
04:         else Δ_k^- ← Δ_k^- + w_kj
05:       end for
06:    end for
end algorithm
```

`Output`: the vectors Δ^+ and Δ^-

Finally, lines $18-23$ compute the optimum F_{score} and the relative intercept (cOpt) by scanning all the lines by increasing intercept and updating the values TP, TN, FP according to the label of the current point.

To improve the effectiveness of the algorithm, we also distinguish the case in which the positive halfplane is above the separation line. This can be done by the procedure `FindOptimalLine` by adding few simple instructions that have not been specified in order not to make more complicated the relative pseudocode. When this case is characterized by a higher F_{score}, we have empirically verified that in almost all the cases there is at least one positive point which lies on the Y-axis. This is the case of extremely unbalanced data, and the procedure `FindOptimalLine` cannot correctly classify these points. We call the alternative procedure that manages this case `FindOptimalLine2` and it is made up by three steps:

1. *Choose a positive point $\Delta(\hat{k})$ which lies on the Y-axis.* The algorithm at first selects all the points which lie on the Y-axis. The aim is to choose a positive point which will be the center of the line bundle we consider in the next step. We sort the selected points by ordinate and for each positive point we compute the F_{score} of the almost vertical line (but with negative slope) crossing this point considering solely the selected points. Then we choose the point $\Delta(\hat{k})$ which correspond to the highest F_{score}.

Figure 2.6: Pseudocode procedure `FindOptimalLine`

`Input`:
1) neuron set V; 2) bipartition (U, S) of V
3) connection matrix W
4) bipartition (U^+, U^-) of U; 5) bipartition (S^+, S^-) of S
`- FindOptimalLine:begin algorithm`
01: `for each` $k \in S$ `do`
02: $m_k \leftarrow \frac{\Delta_k^-}{\Delta_k^+}$
03: `end for`
04: $ordInd \leftarrow \text{order}(m)$; $\text{sort}(m)$; $TP \leftarrow 0$; $FP \leftarrow 0$; $FN \leftarrow \mid S^+ \mid$
05: $mOpt \leftarrow 0$; $Fmax \leftarrow \text{ComputeFscore}(TP, FP, FN)$
06: `for` s from 1 `to` $\mid S \mid$ `do`
07: `if` $ordInd_s \in S^+$ `then` $TP \leftarrow TP + 1$; $FN \leftarrow FN - 1$
09: `else` $FP \leftarrow FP + 1$
10: $F \leftarrow \text{ComputeFscore}(TP, FP, FN)$
11: `if` $F > Fmax$ `then` $mOpt \leftarrow m_{ordInd_s}$; $Fmax \leftarrow F$
12: `end for`
13: `for each` $k \in S$ `do`
14: $c_k \leftarrow \Delta_k^- - mOpt \cdot \Delta_k^+$
15: `end for`
16: $ordInd \leftarrow \text{order}(c)$; $\text{sort}(c)$; $TP \leftarrow 0$; $FP \leftarrow 0$; $FN \leftarrow \mid S^+ \mid$;
17: $cOpt \leftarrow 0$; $Fmax \leftarrow \text{ComputeFscore}(TP, FP, FN)$
18: `for` s from 1 `to` $\mid S \mid$ `do`
19: `if` $ordInd_s \in S^+$ `then` $TP \leftarrow TP + 1$; $FN \leftarrow FN - 1$
20: `else` $FP \leftarrow FP + 1$
21: $F \leftarrow \text{ComputeFscore}(TP, FP, FN)$
22: `if` $F > Fmax$ `then` $cOpt \leftarrow c_{ordInd_s}$; $Fmax \leftarrow F$
23: `end for`
`end algorithm`
`Output`: $[\hat{\alpha} \leftarrow \arctan(mOpt), \hat{\gamma} \leftarrow -cOpt \cdot \cos\hat{\alpha}]$

2. *Compute* $\hat{\alpha}$. The algorithm computes the slopes of the lines crossing the point $\Delta(\hat{k})$ and each point not lying on the Y-axis. Then it searches the line, among those with negative slope, which maximizes the F_{score} criterion by sorting the computed lines according to their slopes in an increasing order. Consequently, the angle $\hat{\alpha}$ relative to the optimum line is in the interval $]\frac{\pi}{2}, \pi[$.

3. *Compute* $\hat{\gamma}$. Compute the intercepts of the lines whose slope is $\tan\hat{\alpha}$ and crossing each point belonging to $I^+ \cup I^-$. The optimum line is identified by scanning the computed lines according to their intercept in an increasing order. Let $\hat{q}$ be the intercept of the optimum line $y = \tan\alpha \cdot z + q$, then we set $\hat{\gamma} = -\hat{q}\cos\hat{\alpha}$.

All the three steps can be computed in $\mathcal{O}(n \log n)$ computational time (due to the sorting), where n is the number of points.

Fig. 2.7 shows the pseudocode of the procedure `FindOptimalLine2`.

Lines $1-2$ find the points of $I^+ \cup I^-$ with abscissa 0 which are sorted by ordinate in line 3.

In the lines $4-11$ the optimum line (considering solely the selected points) among those crossing a positive point is computed; this line is detected by the index $\hat{k}$ of the chosen positive point.

Lines $12-14$ compute the slopes of the lines crossing the point $\Delta_{\hat{k}}$ and each point not lying on Y-axis, whereas the best line is computed in lines $15-24$ by scanning these lines in increasing order of slopes and considering just those with negative slope. This ends the second step.

The third step of this procedure is exactly the step 2 of the procedure `FindOptimalLine`.

Note that in the rare cases in which there are no positive points on the Y-axis we adopt as optimal parameters those computed by the procedure *FindOptimalLine*, as this procedure in this case can correctly classify the positive nodes.

2.6.3 Finding the unknown labels by Network Dynamics

In the last step of *COSNet*, we run the sub-network $H_{U|\boldsymbol{x}(S^+,S^-)}$ to learn the unknown labels of neurons U, preserving the prior information coded in the labels of neurons in S.

The sub-network $H_{U|\boldsymbol{x}(S^+,S^-)} = \langle \boldsymbol{W}_{uu}, \hat{\boldsymbol{\gamma}}^u - \boldsymbol{W}_{su}^T \boldsymbol{x}(S^+,S^-), \hat{\alpha}\rangle$, where $\hat{\gamma}_i^u = \hat{\gamma}$ for each $i \in \{1,2,\ldots,h\}$, adopts the optimum parameters $(\hat{\alpha},\hat{\gamma})$ computed in the previous step (Section 2.6.2) to better fit the topology and achieve near-optimal "low-energy" states of the network (Section 2.6).
The initial state of the network is set to $u_i = 0$ for each $i \in \{1,2,\ldots,h\}$ and the update rule for node i at time $t+1$ is

$$u_i(t+1) = \begin{cases} \sin\hat{\alpha} & \text{if } \sum_{j=1}^{i-1} w_{ij}u_j(t+1) + \sum_{k=i+1}^{h} w_{ik}u_k(t) - \theta_i > 0 \\ -\cos\hat{\alpha} & \text{if } \sum_{j=1}^{i-1} w_{ij}u_j(t+1) + \sum_{k=i+1}^{h} w_{ik}u_k(t) - \theta_i \leq 0 \end{cases} \tag{2.12}$$

Figure 2.7: Pseudocode of the procedure `FindOptimalLine2`

`Input`:
1) neuron set V; 2) bipartition $(U,\, S)$ of V;
3) bipartition $(U^+,\, U^-)$ of U; 4) bipartition $(S^+,\, S^-)$ of S
5) vectors Δ^+ and Δ^-

`begin algorithm`

01: Assign to S_0^+ elements $k \in S^+$ such that $\Delta_k^+ == 0$
02: Assign to S_0^- elements $k \in S^-$ such that $\Delta_k^+ == 0$
03: $\Delta_0^- \leftarrow \Delta^-_{S_0^+ \cup S_0^-}$; $ordInd \leftarrow \text{order}(\Delta_0^-)$; $\text{sort}(\Delta_0^-)$;
04: $TP \leftarrow 0$; $FP \leftarrow \mid \Delta_0^+ \mid$; $FN \leftarrow 0$;
05: $\hat{k} \leftarrow -1$; $Fmax \leftarrow \text{ComputeFscore}(TP, FP, FN)$
06: `for each` $k \in S_0^+$ `do`
07: $TP \leftarrow TP + 1$; $FP \leftarrow ordInd_k - TP$; $FN \leftarrow FN - 1$
08: $F \leftarrow \text{ComputeFscore}(TP, FP, FN)$
09: `if` $F > Fmax$ `then`
10: $\hat{k} \leftarrow k$; $Fmax \leftarrow F$; $TP_1 \leftarrow TP$; $FP_1 \leftarrow FP$
11: `end for`
12: `for each` $i = S \setminus \{S_0^+ \cup S_0^-\}$ `do`
13: $m_i \leftarrow \frac{\Delta_i^- - \Delta_{\hat{k}}^-}{\Delta_i^+ - \Delta_{\hat{k}}^+}$
14: `end for`
15: $ordInd \leftarrow \text{order}(m)$; $\text{sort}(m)$;
16: $TP \leftarrow TP_1$; $FP \leftarrow FP_1$; $FN \leftarrow \mid S^+ \mid -TP$
17: $mOpt \leftarrow \tan(\frac{\pi}{2} + \epsilon)$; $Fmax \leftarrow \text{ComputeFscore}(TP, FP, FN)$
18: `for` s from 1 `to` $\mid S \setminus \{S_0^+ \cup S_0^-\} \mid$ `do`
19: `if` $m_{ordInd_s} < 0$ `then`
20: `if` $ordInd_s \in S^+$ `then` $TP \leftarrow TP + 1$; $FN \leftarrow FN - 1$
21: `else` $FP \leftarrow FP + 1$
22: $F \leftarrow \text{ComputeFscore}(TP, FP, FN)$
23: `if` $F > Fmax$ `then` $mOpt \leftarrow m_{ordInd_s}$; $Fmax \leftarrow F$
24: `end for`

25-35: `Step 3. lines 13-23 of ‘‘FindOptimalLine" computes cOpt`

`end algorithm`

`Output`: $[\hat{\alpha} \leftarrow \arctan(mOpt), \hat{\gamma} \leftarrow -cOpt \cdot \cos\hat{\alpha}]$

where $\theta_i = \hat{\gamma} - \sum_{j \in S} w_{ij} x_j(S^+, S^-)$. The thresholds θ_i embeds the influence of the labeled nodes S, whose fixed states are $\sin\hat{\alpha}$ for positive and $-\cos\hat{\alpha}$ for negative nodes, on the subnetwork $H_{U|\boldsymbol{x}(S^+,S^-)}$.

The stable state $\hat{\boldsymbol{u}}$ reached by this dynamics is used to classify unlabeled data. If the known state $\boldsymbol{x}(S^+, S^-)$, with the parameters found according to the procedure described in Section 2.6.2, is part of a global minimum of the energy of H, and $\hat{\boldsymbol{u}}$ is an energy global minimum of $H_{U|\boldsymbol{x}(S^+,S^-)}$, the sub-network property (Section 2.5) guarantees that $(\hat{\boldsymbol{u}}, \boldsymbol{x}(S^+, S^-))$ is an energy global minimum of H.

Furthermore, our aim is to provide a score for each neuron associated with the "strength" of predictions. In this regard, a consistent way to define these scores is considering the energy contribution $E(u_i)$ of a single node $i \in U$ to the overall energy $E(\boldsymbol{u})$ (2.9):

$$E(u_i) = -u_i \sum_{j \neq i} (w_{ij} u_j - \theta_i) \tag{2.13}$$

Indeed, low values of $E(u_i)$ correspond to stable states u_i for the node i, and can be interpreted as more reliable predictions. From (2.13) we can derive a score $\phi(i)$ associated to each node i:

$$\phi(i) = \sum_{j \neq i} (w_{ij} u_j - \theta_i) \tag{2.14}$$

It is easy to see that for positive predictions (corresponding to $u_i = \sin \hat{\alpha}$), large values of the score $\phi(i)$ correspond to low values of the energy $E(u_i)$. Note that this is true when we have a large number of strongly connected positive nodes in the neighbour of node i, that is when we have positively annotated genes strongly linked to the gene i. The opposite is true for negative predictions: low values of $\phi(i)$ correspond to low energy states for the node i. In our experiments we obtained the scores $\phi(i)$ from the energy $E(u_i)$ reached by each node at the convergence of the *COSNet* algorithm: large values of $\phi(i)$ correspond to stable positive states interpreted as reliable positive predictions. In other words, ϕ values provide scores associated to the "strength" of the prediction.

Fig. 2.8 provides the pseudocode of the network dynamics. Lines $2 - 10$ realize the dynamics until convergence.

Line 5 checks if node i is not stable and in such case the lines $6 - 7$ update the value of the node.

Finally in the lines $9 - 10$ the new energy values is computed and compared with the previous one.

Fig. 2.9 shows the pseudocode of the overall *COSNet* algorithm.

The described *COSNet* procedures assumes that no isolated nodes are present in the network, i.e. each node has at least one neighbour. If the network contains isolated nodes, we discard these nodes from the network and

Figure 2.8: Pseudocode of the procedure `RunSubNet` that implements the third step of the *COSNet* algorithm.

`Input`:
- neuron set V;
- bipartition (U, S) of V;
- connection matrix W
- optimal parameters $\hat{\alpha}$ and $\hat{\gamma}$
- threshold vector $\theta^u = \hat{\boldsymbol{\gamma}}^u - \boldsymbol{W}_{su}^T \boldsymbol{x}(S^+, S^-)$

```
begin algorithm
```
01: $E(\boldsymbol{u}) \leftarrow -\frac{1}{2}\sum_{i\in U}\sum_{j\in U} u_i w_{ij} u_j + \sum_{i\in U} u_i\theta_i$
02: `repeat`
03: $E_{old} \leftarrow E(u)$
04: `for each` $i \in U$ `do`
05: `if` $u_i\left(\sum_{j\in U} w_{ij}u_j - \theta_i\right) \leq 0$ `then`
06: `if` $\sum_{j\in U} w_{ij}u_j - \theta_i > 0$ `then` $u_i \leftarrow \sin\hat{\alpha}$
07: `else` $u_i \leftarrow -\cos\hat{\alpha}$
08: `end for`
09: $E(\boldsymbol{u}) \leftarrow -\frac{1}{2}\sum_{i\in U}\sum_{j\in U} u_i w_{ij} u_j + \sum_{i\in U} u_i\theta_i$
10: `until` $E_{old} = E(\boldsymbol{u})$
11: `for each` $i \in U$ `do`
12: $\phi_i \leftarrow \sum_{j\in U} w_{ij}u_j - \theta_i$
13: `end for`

```
end algorithm
```
`Output`: the vectors $\boldsymbol{u}$ and $\boldsymbol{\phi}$
- $\boldsymbol{u}$ vector of predictions
- $\boldsymbol{\phi}$ vector of scores

adopt a different procedure for these nodes. Specifically, let p_s be the rate of positive nodes in S and n_{iso} the number of isolated nodes. We generate a random number m_{iso} according to the binomial distribution $B(n_{iso}, p_s)$, and then m_{iso} isolated nodes randomly chosen are predicted as positive, the remaining as negative. To define a score for these nodes, we consider the scores predicted by the algorithm on the non-isolated nodes.

Let n_{neg} be the number of negative predictions and n_{pos} the number of positive predictions inferred by the algorithm for not isolated nodes. The score for isolated nodes predicted as negative is the set as the $(\lceil n_{neg} \cdot p_s \rceil)^{th}$ order statistic of the negative score sample, i.e. the $(\lceil n_{neg} \cdot p_s \rceil)^{th}$ element of the vector of n_{neg} negative scores increasingly ordered, and the score for

Figure 2.9: Pseudocode of the overall *COSNet* algorithm.

`Input`:
- neuron set V;
- bipartition (U, S) of V;
- bipartition (S^+, S^-) of S
- connection matrix W

`begin algorithm`
02: `Generate random number` $m \in B(\mid U \mid, \frac{|S^+|}{|S|})$
03: `Assign to` U^+ m `elements of` U `randomly chosen`
04: $[\Delta^+, \Delta^-] \leftarrow$ `ComputePoints`$\big((U,S),\ W,\ (U^+,U^-),\ (S^+,S^-)\big)$
05: $[\hat{\alpha}, \hat{\gamma}] \leftarrow$ `FindOptimalLine`$\big((U,S),\ W,\ (U^+,U^-),\ (S^+,S^-),\ \Delta^+,\ \Delta^-\big)$
06: `if positive halfplane ‘‘above" the line then`
07: $\quad [\hat{\alpha}, \hat{\gamma}] \leftarrow$ `FindOptimalLine2`$\big((U,S),\ W,\ (U^+,U^-),\ (S^+,S^-),\ \Delta^+,\ \Delta^-\big)$
08: $\theta^u = \hat{\boldsymbol{\gamma}}^u - \boldsymbol{W}_{su}^T \boldsymbol{x}(S^+, S^-)$
09: $[\hat{\boldsymbol{u}}, \hat{\boldsymbol{\phi}}] \leftarrow$ `RunSubNet`$\big((U,S),\ W,\ (S^+,S^-),\ \hat{\alpha},\ \hat{\gamma},\ \theta^u\big)$
`end algorithm`
`Output`: the vectors $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{\phi}}$
$\hat{\boldsymbol{u}}$ vector of predictions
$\hat{\boldsymbol{\phi}}$ vector of scores

isolated nodes predicted as positive is the $(\lceil n_{pos} \cdot p_s \rceil)^{th}$ order statistic of the positive score sample.

2.6.4 Time complexity

In this section we analyze the time complexity of *COSNet* in order to show that it nicely scales when large-scale data are used. *COSNet* is made up by three steps (see Section 2.6) and we analyze the complexity of each step separately.

The first step generates a temporary bipartition of U (Section 2.6.1) and it easy to show that it takes time $\mathcal{O}(|U|)$.

The second step is described by the procedures in Fig. 2.5 and 2.6. The procedure *ComputePoints* for each node in S computes the weighted sum of its positive and negative neighbours, taking time $\mathcal{O}(|W|)$, where with $|W|$ we mean the number of non-null components of the matrix W. The lines $1-3$, $6-12$, $13-15$ and $18-23$ of the procedure *FindOptimalLine* take time $\mathcal{O}(|S|)$, whereas $\mathcal{O}(|S| \log |S|)$ is the complexity of lines 4 and 16 due to the sorting. Lines 5 and 17 takes constant time. A similar analysis can be done for the procedure *FindOptimalLine2*. Accordingly the step 2 of *COSNet* has

complexity $\mathcal{O}(|S| \log |S| + |W|)$.

The third step run the dynamics of the network restricted to nodes in U. It's clear that the complexity of this step depends on the number of iterations needed for convergence. We empirically observed the network needs two or three iterations for reaching a stable state, as also observed in [31]. Each iteration takes time $\mathcal{O}(|W_{uu}|)$. Moreover this steps needs also time $\mathcal{O}(|W_{us}|)$ for computing neuron thresholds.

Overall, the *COSNet* algorithm takes time $\mathcal{O}(|S| \log |S| + |W|)$ which is almost linear in the input size (number of neurons) when the connection matrix is sparse, constraint that can be easily satisfied by adopting suitable thresholds for connection weights during the preprocessing of input data.

2.6.5 Model regularization

When the distribution of the points $\Delta(k)$ (Section 2.6.2) is such that $\hat{\alpha}$ is very close to $\frac{\pi}{2}$ (that is, when points $\Delta(k), k \in S$ can be separated by an almost vertical line), since $\lim_{\hat{\alpha} \to (\frac{\pi}{2})^-} \tan \hat{\alpha} = \infty$, the network dynamics is characterized by a huge influence of positive neurons, leading to a degenerate state where almost all neurons are positive. This is true also in the opposite case, i.e. when $\tan \hat{\alpha} = 0$, leading to the "all negatives" degenerate state.

To prevent this behavior, we add to the energy function of the network $H|_{U,\boldsymbol{x}(S^+,S^-)}$ a regularization term that is minimized when the number of positive neurons in U is close to $\nu_u = p_s \cdot h$, that is when the probability (2.10) is maximized (Section 2.6.1). Here $p_s = \frac{|S^+|}{|S|}$. This component can be considered as a penalty for the current state of the network depending on how much different is the actual number of positives from ν_u. By adding a regularization term to (2.9), we obtain:

$$E(\boldsymbol{u}) = -\frac{1}{2} \sum_{i \neq j} w_{ij} u_i u_j + \sum_{i=1}^{h} u_i \theta_i + \eta \left(\sum_{i=1}^{h} (a u_i + b) - \nu_u \right)^2, \qquad (2.15)$$

where η is a regularization parameter, $a = \frac{1}{\sin \hat{\alpha} + \cos \hat{\alpha}}$ and $b = \frac{\cos \hat{\alpha}}{\sin \hat{\alpha} + \cos \hat{\alpha}}$. It is easy to see that $\sum_{i=1}^{h} (a u_i + b)$ is the number of positive neurons in U, and hence the penalty term is the quadratic difference between the number of positive neurons in U predicted by the algorithm and the number of expected positive labeled neurons in U (Section 2.6.1). Focusing just on the added

function, up to constants terms, we have:

$$\left(\sum_{i=1}^{h}(au_i+b)-\nu_u\right)^2 \longrightarrow \left(\sum_{i=1}^{h}(au_i+b)\right)^2 - 2\nu_u\sum_{i=1}^{h}(au_i+b)$$
$$= \sum_{i\neq j}(au_i+b)(au_j+b)+\sum_{i=1}^{h}(au_i+b)^2$$
$$- 2\nu_u\sum_{i=1}^{h}(au_i+b)$$

As $(au_i+b)\in\{0,1\}$, $(au_i+b)=(au_i+b)^2$, it follows

$$\left(\sum_{i=1}^{h}(au_i+b)-\nu_u\right)^2 \simeq a^2\sum_{i\neq j}u_iu_j+ab\sum_{i\neq j}(u_i+u_j)+(1-2\nu_u)\sum_{i=1}^{h}(au_i+b)$$
$$\simeq a^2\sum_{i\neq j}u_iu_j+2ab(h-1)\sum_{i=1}^{h}u_i+a(1-2\nu_u)\sum_{i=1}^{h}u_i$$
$$= a^2\sum_{i\neq j}u_iu_j+a(2b(h-1)+(1-2\nu_u))\sum_{i=1}^{h}u_i$$

Considering the whole energy, we have

$$E(\boldsymbol{u}) = -\frac{1}{2}\sum_{i=1}^{h}\sum_{\substack{j=1\\j\neq i}}^{h}(w_{ij}-2\eta a^2)u_iu_j+\sum_{i=1}^{h}u_i\left[\theta_i+\eta a(2b(h-1)+(1-2\nu_u))\right]$$
$$= -\frac{1}{2}\sum_{i=1}^{h}\sum_{\substack{j=1\\j\neq i}}^{h}\tilde{w}_{ij}u_iu_j+\sum_{i=1}^{h}u_i\tilde{\theta}_i \tag{2.16}$$

where $\tilde{\theta}_i=\theta_i+\eta a\left[2b(h-1)+(1-2p_sh)\right]$ and $\tilde{w}_{ij}=(w_{ij}-2\eta a^2)$.

The parameter η regulates the influence of the new energy component on the dynamics of the network; a large value of η generates a dynamics mainly controlled by the regularization term. In principle, we need a value of η such that the influence on the dynamics of the added term is low when data are quite balanced, and increases when the value of $\hat{\alpha}$ is close to limit cases (e.g. when $\hat{\alpha}$ is close to $\frac{\pi}{2}$ or 0). An empirical choice of η which provides this behavior is:

$$\eta = \beta|\tan((\hat{\alpha}-\tfrac{\pi}{4})*2)| \tag{2.17}$$

where β is a non negative real constant. This choice guarantees an increasing penalty when a too large influence is given to positive ($\hat{\alpha}\simeq\frac{\pi}{2}$) or to negative

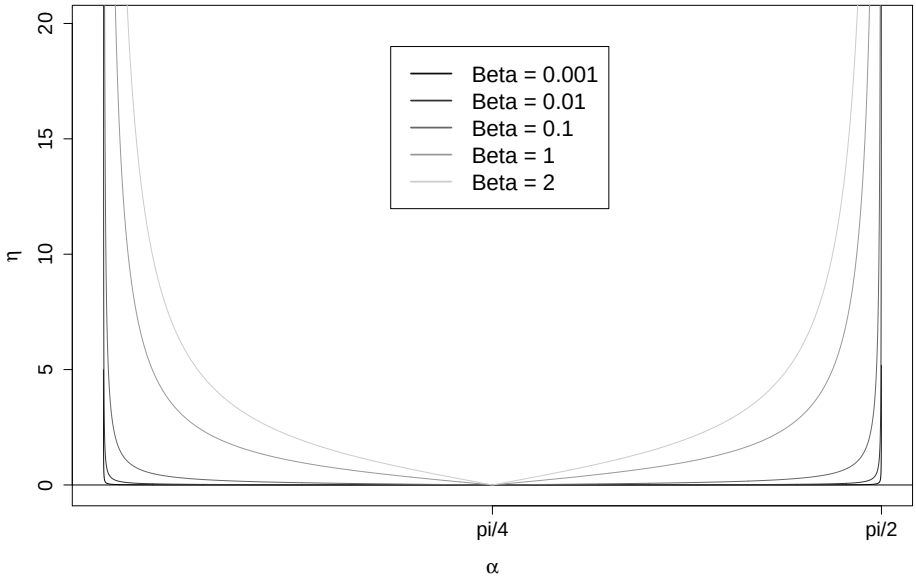


Figure 2.10: η graph with different values of β.

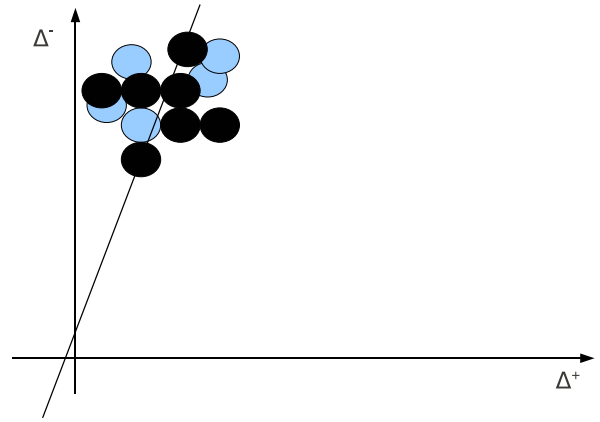
($\hat{\alpha} \simeq 0$) neurons, while no penalty is added when positive and negative states are identical ($\hat{\alpha} = \frac{\pi}{4}$). Figure 2.10 shows the values of the parameter η corresponding to different choices of β. Low values of β compress the curve close to the x axis. By choosing the suitable value of β we can finely control the influence of the new energy term on the network dynamics, and, by fixing β after a tuning procedure on sample data sets, we can obtain a regularization that automatically fits the data by augmenting its influence coming close to the limit cases.

2.6.6 *COSNet* covers GFP learning issues

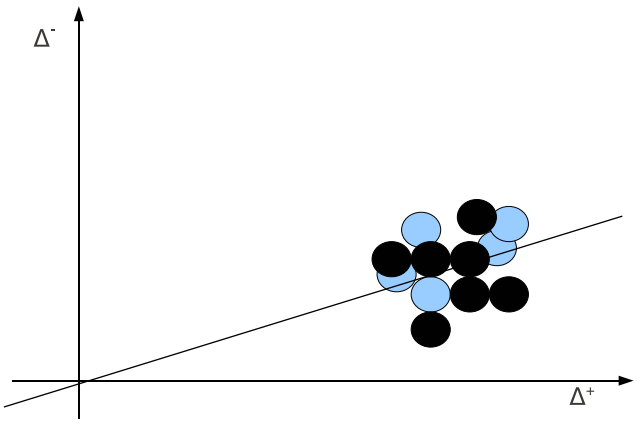
In this section we analyze the effectiveness of the proposed algorithm w.r.t the learning issues described in Section 2.4.

1. Preservation of the prior knowledge. The restriction of the dynamics to the unlabeled data assures the preservation of the prior knowledge coded in the connection matrix and in the bipartition of the labeled data. Note that a similar approach has been proposed in [33], even if in that case the known labels are simply restored at each iteration of the algorithm, without an actual restriction of the dynamics.

2. Unbalance problem in functional classes. This problem naturally



(a)



(b)

Figure 2.11: Placement of points in the set I when $|S^+| \ll |S^-|$ (a), and when $|S^+| \gg |S^-|$ (b).

arises in GFP, since often GO terms have a small number of annotated genes. This is true especially for the most specific terms (i.e. "deep" nodes in the ontology w.r.t. the root) that are the most interesting from a biological standpoint [76], since they better characterize the functions of genes and gene products.

When $|S^+| \ll |S^-|$, the points $\Delta(k) \equiv (\Delta^+(k), \Delta^-(k))$ (Section 2.6.2)

are such that $\Delta^-(k) \gg \Delta^+(k)$. Figure 2.11 (a) shows the placement of points $\Delta(k)$ in this case. Accordingly, a separation angle $\frac{\pi}{4} \leq \hat{\alpha} \leq \frac{\pi}{2}$ is computed by the algorithm `FindOptimalLine` described in Section 2.6.2. In our setting, such an angle determines a value of the positive states greater than the negative ones, yielding the network dynamics to converge towards non trivial attractors.

On the other hand, when $|S^+| \gg |S^-|$, the points are located as in Figure 2.11 (b) and the computed separation angle $0 \leq \hat{\alpha} \leq \frac{\pi}{4}$ is such that negative points have a higher activation value than those positive.

In this way annotated genes can actually propagate their positive state across the biological network, without surrendering to the prevalence of negative nodes leading to trivial states. Indeed for several GO terms (and especially for those characterized by a small number of positive annotations) Hopfield networks [31], or more in general non cost-sensitive label propagation algorithms [38, 33, 36], may incur trivial "all negative" predictions, while *COSNet* is able to correctly predict (at least in part) positive annotated genes.

3. Incoherence of the prior knowledge coding. We would like to show that the parameters (α, γ) automatically selected by *COSNet* can yield to a "more coherent" state w.r.t. the prior knowledge, in the sense that this state corresponds to a lower energy of the underlying network.

To this end, we consider a data set of binary protein-protein interactions of 2338 yeast proteins (*PPI-VM*) [86] in which a labeling $\overline{\boldsymbol{x}} \in \{1, -1\}^{|V|}$ of V is known. By applying *COSNet* we approximate the optimal parameters $(\hat{\alpha}, \hat{\gamma})$ and we define the state $\overline{\boldsymbol{x}}(\hat{\alpha})$ by setting $\overline{x}_k(\hat{\alpha}) = \sin\hat{\alpha}$ if $\overline{x}_k = 1$ and $\overline{x}_k(\hat{\alpha}) = -\cos\hat{\alpha}$ if $\overline{x}_k = -1$, for each $k \in \{1 \dots |V|\}$. We show that the state $\overline{\boldsymbol{x}}(\hat{\alpha})$ is "more coherent" with the prior knowledge than $\overline{\boldsymbol{x}}$, by studying whether $\overline{\boldsymbol{x}}(\hat{\alpha})$ is "closer" than $\overline{\boldsymbol{x}}$ to a global minimum of the energy function $E(\boldsymbol{x})$.

As measure of "closeness" of a given state $\boldsymbol{z}$ to a global minimum of $E(\boldsymbol{x})$, we consider the probability P_z that $E(\boldsymbol{x}) < E(\boldsymbol{z})$, where $\boldsymbol{x} = (x_1, x_2, \dots, x_{|V|})$ is a random state generated according to the binomial distribution $B(|V|, \rho_z)$, and ρ_z is the rate of positive components in $\boldsymbol{z}$.

Estimation of P_z. To estimate P_z, we independently generate t random states $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, ..., $\boldsymbol{x}^{(t)}$ and we set $Y = \sum_{i=1}^{t} \beta(E(\boldsymbol{z}) - E(\boldsymbol{x}^{(i)}))$, where $\beta(x) = 1$ if $x \geq 0$, 0 otherwise. The variable $\frac{Y}{t}$ is an estimator of p_z, and in our setting $Y << t$. For determining the confidence interval of P_z at a $1 - \delta$ confidence level, we need to consider three cases:

1. $Y = 0$. We can directly compute the confidence interval $[0, 1 - \delta^{\frac{1}{t}}]$.

2. $1 \leq Y \leq 5$. Y is approximately distributed according to the Poisson distribution with expected value $\lambda = Y$. Accordingly, the confidence interval is $\left[\frac{1}{2n}\chi^2_{2Y,1-\frac{\delta}{2}}, \frac{1}{2n}\chi^2_{2(Y+1),\frac{\delta}{2}}\right]$, where χ^2_k is a chi squared random variable with k degrees of freedom.

3. $Y > 5$. The random variable Y is approximately distributed according to a normal distribution with expected value Y and variance $\frac{Y(1-Y)}{t}$. We adopt the Agresti-Coull interval estimator [77], which is more stable for values of Y closer to the outliers [78]. The resulting confidence interval is $\frac{Y+2}{t+4} \pm \frac{1}{t+4}\sqrt{(Y+2)(t-Y-2)}z_{1-\frac{\delta}{2}}$, where $z_{1-\alpha}$ is the $1-\alpha$ percentile of the standard normal distribution.

By setting $\delta = 0.05$ and $t = 1000$, we estimated the confidence interval for both $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$. In Table 2.1 we report the comparison of the confidence intervals of $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$ in the *PPI-VM* data set and for the some of the considered FunCat classes. Since the results were similar for all the functional classes, we report just the first classes in a lexicographic order. Moreover, in Table 2.2 we report a description of these classes. We point out that similar results are obtained also with other data sets.

We distinguish two main cases: a) both the confidence intervals coincide with the minimum interval $[0, 0.0030]$, case coherent with the prior information; b) both lower and upper bounds of $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ are less than the corresponding bounds of $P_{\overline{x}}$. It is worth noting that, in almost all cases, the probability $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ has an upper bound smaller than the lower bound of $P_{\overline{x}}$. This is particularly evident for classes "01.03.16.01", "02.13" and "11.02.01"; in the latter the lower bound of $P_{\overline{\boldsymbol{x}}}$ is 0.7761, while the corresponding upper bound of $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ is $\backsimeq 0$.

These results, reproduced with similar trends in other data sets (data not shown), point out the effectiveness of our method in approaching the problem of the incoherence of the prior knowledge coding.

2.6.7 Software Implementation

We have developed all the software which implements the *COSNet* algorithm. The software is written in R, a language and environment for statistical computing and graphics largely used in bioinformatics. Even if its im-

Table 2.1: Confidence interval estimation for the probabilities $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$ at a confidence level 0.95 (data set PPI-VM).

Data set PPI-VM										
Class	Confidence interval					Class	Confidence interval			
	$P_{\overline{\boldsymbol{x}}}(\hat{\alpha})$		$P_{\overline{\boldsymbol{x}}}$				$P_{\overline{\boldsymbol{x}}}(\hat{\alpha})$		$P_{\overline{\boldsymbol{x}}}$	
	min	max	min	max			min	max	min	max
"01"	0	0.0030	0	0.0030		"02"	0	0.0030	0	0.0030
"01.01"	0	0.0030	0	0.0030		"02.01"	0	0.0030	0.0638	0.0975
"01.01.03"	0.0001	0.0056	0.0433	0.0722		"02.07"	0	0.0030	0.0011	0.0102
"01.01.06"	0.0001	0.0056	0.0442	0.0733		"02.10"	0	0.0030	0.0522	0.0833
"01.01.06.05"	0.0210	0.0427	0.0702	0.1051		"02.11"	0.0002	0.0072	0.0939	0.1332
"01.01.09"	0	0.0030	0.0045	0.0174		"02.13"	0.0312	0.0565	0.3622	0.4226
"01.02"	0.0001	0.0056	0.0067	0.0212		"02.13.03"	0.7139	0.7681	0.7740	0.8236
"01.03"	0	0.0030	0.0620	0.0953		"02.19"	0.0001	0.0056	0.0006	0.0088
"01.03.01"	0.1452	0.1915	0.2232	0.2768		"02.45"	0.1022	0.1428	0.1815	0.2312
"01.03.01.03"	0	0.0030	0.0145	0.0333		"11"	0	0.0030	0	0.0030
"01.03.04"	0.5020	0.5637	0.6280	0.6867		"11.02"	0	0.0030	0	0.0030
"01.03.16"	0.0025	0.0135	0.1189	0.1619		"11.02.01"	0	0.0030	0.7761	0.8255
"01.03.16.01"	0	0.0030	0.3025	0.3608		"11.02.02"	0.2184	0.2716	0.8519	0.8931

plementation is efficient, R is an interpreted language, and especially when iterative-intensive constructs are needed, runs more slowly than compiled code. For this reason, we have written the computationally more expensive parts of the algorithms in C language, and loaded the compiled C code as a shared object in the R environment. The code is available on demand at frasca@dsi.unimi.it.

Table 2.2: Description of functional terms of FunCat ontology reported in Table 2.1.

Classes	Description
"01"	Metabolism
"01.01"	Amino acid metabolism
"01.01.03"	Assimilation of ammonia, metabolism of the glutamate group
"01.01.06"	Metabolism of the aspartate family
"01.01.06.05"	Metabolism of methionine
"01.01.09"	Metabolism of the cysteine - aromatic group
"01.02"	Nitrogen, sulfur and selenium metabolism
"01.03"	Nucleotide/nucleoside/nucleobase metabolism
"01.03.01"	Purin nucleotide/nucleoside/nucleobase metabolism
"01.03.01.03"	Purine nucleotide/nucleoside/nucleobase anabolism
"01.03.04"	Pyrimidine nucleotide/nucleoside/nucleobase metabolism
"01.03.16"	Polynucleotide degradation
"01.03.16.01"	RNA degradation
"02"	Energy
"02.01"	Glycolysis and gluconeogenesis
"02.07"	Pentose-phosphate pathway
"02.10"	Tricarboxylic-acid pathway (citrate cycle, Krebs cycle, TCA cycle)
"02.11"	Electron transport and membrane-associated energy conservation
"02.13"	Respiration
"02.13.03"	Aerobic respiration
"02.19"	Metabolism of energy reserves (e.g. glycogen, trehalose)
"02.45"	Energy conversion and regeneration
"11"	Transcription
"11.02"	RNA synthesis
"11.02.01"	rRNA synthesis
"11.02.02"	tRNA synthesis

2.7 Experimental setting

In this section we describe the experimental procedure used for validating the *COSNet* algorithm.

2.7.1 GFP in Yeast

We performed predictions of gene functions at genome-wide level in the *S.cerevisiae* organism, using the whole FunCat ontology [6] [1]. In order to compare our algorithm also with some hierarchical methods described in Section 3.4.1, we follow the same experimental setting proposed in the corresponding paper by selecting classes with at least 20 positive examples [104]. Nevertheless, our algorithm can be used for predicting classes with less than 20 positives, as done in Section 3.4.2.

We used five different biomolecular data sources, previously analyzed in [75]. The main characteristics of the data can be summarized as follows:

- *Pfam-1* data are represented as binary vectors: each feature registers the presence or absence of 4,950 protein domains obtained from the Pfam (Protein families) data base. This dataset contains 3529 genes.

- *Pfam-2* is an enriched representation of Pfam domains by replacing the binary scoring with log E-values obtained with the HMMER software toolkit [79].

- *Expr* data contains gene expression measures of 4523 genes relative to two experiments described in [80] and [81].

- *PPI-BG* data set contains protein-protein interaction data downloaded from the BioGRID database [82]. Data are binary: they represent the presence or absence of protein-protein interactions for 4531 proteins.

- *PPI-VM* is another data set of protein-protein interactions that collects binary protein-protein interaction data for 2338 proteins from yeast two-hybrid assay, mass-spectrometry of purified complexes, correlated mRNA expression and genetic interactions [86].

For *PPI* data we adopt the scoring function used by Chua et al [64], which assigns to genes i and j the similarity score

$$w_{ij} = \frac{2|N_i \cap N_j|}{|N_i \setminus N_j| + 2|N_i \cap N_j| + 1} \times \frac{2|N_i \cap N_j|}{|N_j \setminus N_i| + 2|N_i \cap N_j| + 1}$$

[1] We used the FunCat-2.1 scheme with the annotation data FunCat-2.1_data_20070316, available from: ftp://ftpmips.gsf.de/yeast/catalogues/funcat/funcat-2.1_data_20070316.

where N_k is the set of the neighbors of gene k (k is included).

In the remaining data sets, each gene is associated with a feature vector. Accordingly, the score for each gene pair is set to the Pearson's correlation coefficient of the corresponding feature vectors. For *Expr* data we computed the squared correlation coefficient in order to take in account the down regulation of a gene by another one, i.e. a gene over expressed inhibits the expression of another gene, resulting in a close relationship even though the corresponding correlation coefficient is negative.

To reduce the noise introduced by too small edge weights, we eliminated edges below a given threshold. We tune the edge threshold for each data set separately, by ensuring that each gene has at least one neighbour (no isolated nodes). Finally, each obtained networks $\boldsymbol{W}$ has been normalized by dividing each entry W_{ij} by the square root of the product of the the sum of the elements of row i and the sum of elements in column j. In other words, if D is a $n \times n$ diagonal matrix such that $D_{ii} = \sum_j W_{ij}$ then the normalized matrix $\hat{W}$ is:

$$\hat{\boldsymbol{W}} = D^{-1/2} \boldsymbol{W} D^{-1/2} \tag{2.18}$$

2.7.2 Results and Discussion

We compared *COSNet* with other semi-supervised label propagation algorithms and supervised machine learning methods proposed in the literature for the gene function prediction problem. We considered the classical *GAIN* algorithm [31], based on Hopfield networks; *LP-Zhu*, a semi-supervised learning method based on label propagation [33]; *SVM-l* and *SVM-g*, i.e. respectively linear and Gaussian kernel SVMs with probabilistic output [84]. SVMs had previously been shown to be among the best algorithms for predicting gene functions in a "FLAT" setting (that is without considering the hierarchical relationships between classes) [61, 85].

We tested two values of β for defining η (see Section 2.6.5): $\beta = 0$ which corresponds to the unregularized version of *COSNet* and $\beta = 0.0001$. This value is the result of a tuning procedure we used on several sample data sets and that we now extend to all the used data sets for the regularized version.

To estimate the generalization capabilities of the compared methods we adopted a stratified 10-fold cross validation procedure, by ensuring that each fold includes at least one positive example for each classification task. Considering the severe unbalance between positive and negative classes, beyond the classical accuracy, we computed *F-score*, *precision* and *recall* for each functional class and for each considered data set. Indeed in this context the

Table 2.3: Accuracy - F-score comparison between GAIN, LP-Zhu, *COSNet*, *COSNet*($\beta = 0.0001$), SVM-l, SVM-g.

Data set	Methods						Performance measures
	GAIN	*LP-Zhu*	COSNet	COSNet($\beta = 0.0001$)	*SVM-l*	*SVM-g*	
Pfam-1	0.9615	0.9613	0.9560	0.9570	0.7528	0.7435	Accuracy
	0.0277	0.0120	0.3937	0.3944	0.2722	0.2355	F-score
Pfam-2	0.9613	0.9656	0.9054	0.9330	0.7048	0.7515	Accuracy
	0.0296	0.2117	0.3299	0.3372	0.1054	0.0270	F-score
Expr	0.9655	0.9655	0.5022	0.8791	0.7496	0.7704	Accuracy
	0	0.0008	0.1007	0.1224	0.0531	0.0192	F-score
PPI-BG	0.9666	0.9704	0.9495	0.9580	0.7679	0.7597	Accuracy
	0.0362	0.1758	0.3528	0.3536	0.1546	0.1178	F-score
PPI-VM	0.9554	0.9560	0.9337	0.9440	0.7237	0.7222	Accuracy
	0.1009	0.2106	0.3911	0.3921	0.1888	0.2351	F-score

accuracy is only partially informative, since a classifier predicting always "negative" could obtain a very high accuracy. Table 2.3 shows the average F-score and accuracy across all the classes and for each data set.

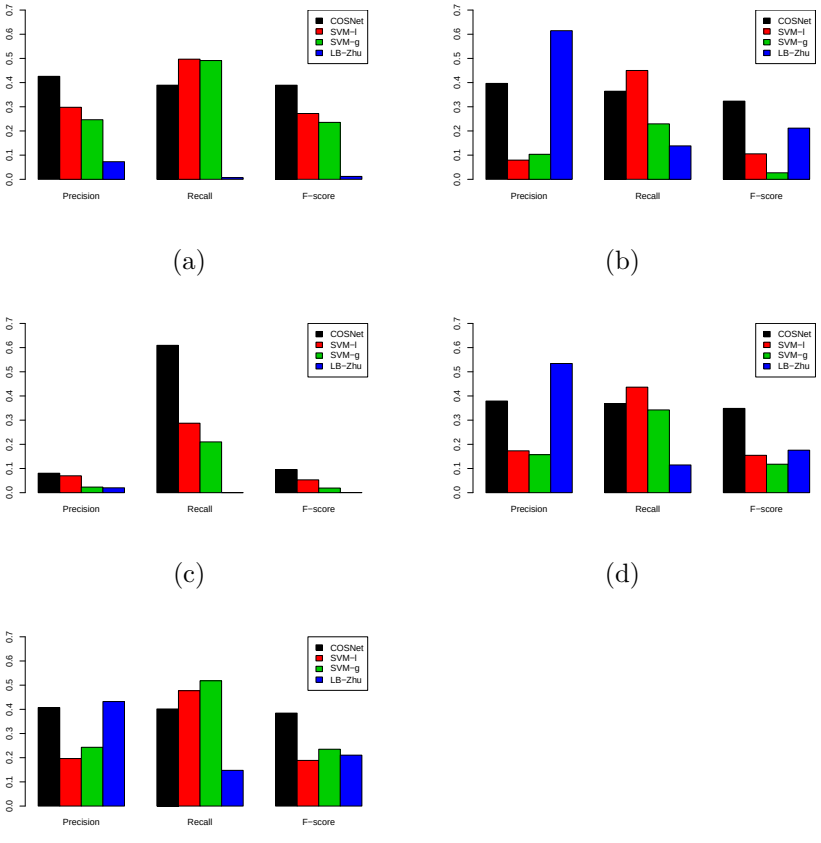
The results show that *COSNet* achieves the best performances (in terms of the F-score) w.r.t. all the other methods with or without regularization. Moreover, the regularization lead to better performances either in terms of accuracy or in terms of F-score and the effect of the regularization is more evident for the *Expr* dataset, in which the optimization phase often produce a value of α close to $\pi/2$.

The *LP-Zhu* method is the second best method in *Pfam-2* and *PPI-BG* data sets, but obtains very low F-scores with *Pfam-1* and *Expr* data, despite high accuracies. This is an example of the low informativeness of the accuracy in an unbalanced contexts as GFP. These overall results are confirmed by the Wilcoxon signed-ranks test [83], which is a non-parametric statistical hypothesis test used when comparing two related samples or repeated measurements on a single sample to assess whether their population means differ or which sample mean is greater than the other one. We applied this test to the F-score performance vectors: we registered a significant improvement in favour of *COSNet* with respect to all the other methods and for each considered data set at $\alpha = 10^{-15}$ significance level.

In order to understand the reasons for which our method works better, we compared also the overall precision and recall of the methods separately for each data set: we did not consider *GAIN*, since this methods achieved the worst results in almost all the data sets. Table 2.4 summarizes this

comparison. Moreover, to provide a visual clue of the performance of the adopted methods, in Figure 2.12 we graphically show the results contained in Table 2.4. It is more clear that while *COSNet* does not always achieve the best precision or recall, it obtains the best F-score as a result of a good balancing between them.

Finally, we want also to compare the adopted methods also in terms of the time needed for the computation of all the prediction tasks. We performed these algorithms on a machine with Intel processor 2.80 GHz and 16 GB of RAM memory. In Table 2.5 we report the time in seconds needed by *COSNet*, *Zhu-LP* and *GAIN* algorithms for computing 10-folds cross vali-



(a) (b)

(c) (d)

e)

Figure 2.12: Average precision, recall and F-score for each compared method (excluding *GAIN*) with data set Pfam-1 (a), Pfam-2 (b), Expr (c), PPI-BG (d) and PPI-VM (e).

Table 2.4: Average precision, recall, F-score and Accuracy for COSNet (β = 0.0001), SVM-l, SVM-g and LP-Zhu on single data sets.

Pfam-1				
Methods	Prec	Rec	F	Accuracy
COSNet	0.4270	0.3864	0.3951	0.9591
SVM-l	0.2979	0.4969	0.2722	0.7528
SVM-g	0.2465	0.4912	0.2355	0.7435
LP-Zhu	0.0729	0.0069	0.0120	0.9613
Pfam-2				
Methods	**Prec**	**Rec**	**F**	**Accuracy**
COSNet	0.3835	0.3327	0.3360	0.9224
SVM-l	0.0794	0.4501	0.1054	0.7048
SVM-g	0.1035	0.2294	0.0270	0.7515
LP-Zhu	0.6147	0.1383	0.2117	0.9656
Expr				
Methods	**Prec**	**Rec**	**F**	**Accuracy**
COSNet	0.1069	0.2206	0.1247	0.8775
SVM-l	0.0699	0.2874	0.0531	0.7496
SVM-g	0.0232	0.2101	0.0192	0.7704
LP-Zhu	0.0202	0.0004	0.0008	0.9655
PPI-BG				
Methods	**Prec**	**Rec**	**F**	**Accuracy**
COSNet	0.3803	0.3409	0.3532	0.9617
SVM-l	0.1732	0.4365	0.1546	0.7679
SVM-g	0.1572	0.3423	0.1178	0.7597
LP-Zhu	0.5343	0.1147	0.1758	0.9704
PPI-VM				
Methods	**Prec**	**Rec**	**F**	**Accuracy**
COSNet	0.4155	0.3915	0.3929	0.9442
SVM-l	0.1965	0.4771	0.1888	0.7237
SVM-g	0.2431	0.5181	0.2351	0.7222
LP-Zhu	0.4322	0.1477	0.2106	0.9560

Table 2.5: Overall time in seconds needed by each algorithm for computing 10-folds cross validation on each functional class on single data.

Method	Dataset				
	PPI-BG	Expr	Pfam-1	Pfam-2	PPI-VM
COSNet	894.634	1280.011	402.589	602.676	175.117
Zhu-LP	499.534	483.943	261.960	293.298	99.549
GAIN	1957.209	1314.215	780.184	824.335	548.871

dation on each functional class for each integrated network. We do not show the time needed by SVMs because these methods need several hours for this tasks. *COSNet* is faster GAIN on each dataset, in two cases needs less than the half of the time needed by GAIN (PPI-BG and PPI-VM). Zhu-LP is the fastest algorithm, because it reduces the dynamics to the unlabeled nodes like *COSNet*, but without the learning phase which characterizes *COSNet*.

Overall, we think that obtained performance improvements come from the *COSNet* cost-sensitive approach that allows to automatically find the "near-optimal" parameters of the network with respect to the distribution of positive and negative nodes (Section 2.6). It is worth noting that using only single sources of data *COSNet* can obtain a relatively high precision, without suffering a too high decay of the recall. This is of paramount importance in the gene function prediction problem, where "in silico" positive predictions of unknown genes need to be confirmed by expensive "wet" biological experimental validation procedures. From this standpoint the experimental results show that our method could be applied to predict the "unknown" functions of genes, considering also that data fusion techniques could in principle further improve the reliability and the precision of the results [37, 89], as we discuss in Chapter 3.

Chapter 3

LSI: an efficient cost-sensitive algorithm for network data integration

When different data sources are available in the same context it may happen that some of them are more informative than other ones. This fact is quite usual in the GFP context, since a particular type of biomolecular data can be informative or not depending on the functional class to be predicted.

In other words, the predictive capabilities of a learning algorithm might significantly change when different sources of data are used to predict functions of genes. In order to make more clear this basic concept, the heat map in Figure 3.1 represents the F-scores per class obtained by *COSNet* on yeast data set (results described in Section 2.7). Lighter colors means higher F-scores. It is clear that the informativeness of each data set depends on the considered functional class, and there are some classes for which the data set in average less informative (*Expr*, see Table 2.3) obtains the highest F-score. In Table 3.1 we also show the F-score obtained by *COSNet* for some specific classes where *Expr* (e.g. 01.01.06.05) or *Pfam-2* (e.g. 01.05.02) data sets are the most informative.

It should be interesting and of paramount importance knowing a priori which data sets are more informative in order to appropriately integrate them in a unique composite network. In this chapter we investigate a new approach for integrating different data sources/networks for obtaining a unique high-reliability/high-coverage network.

The proposed method exploits the linear separation of the nodes projected into a two dimensional space performed in the second step of *COSNet*, and for this reason we name it *Linear Separability Integration* (LSI).

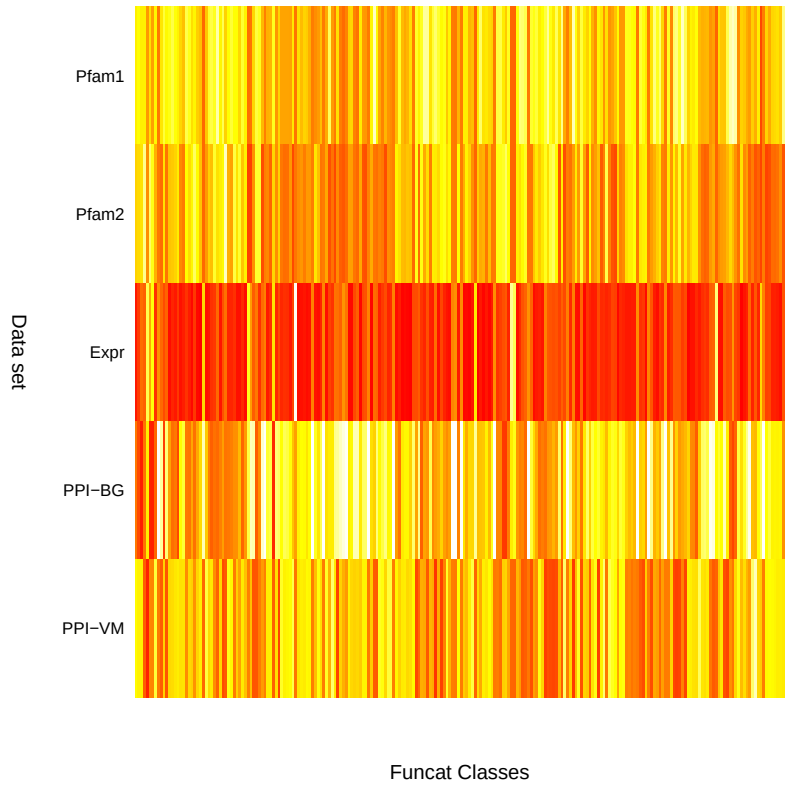


Figure 3.1: Heat map representing the F-scores obtained by *COSNet* on the yeast data sets (Section 2.7). The lighter the color the higher the corresponding value of F-score. The columns represent the 232 considered FunCat functional classes.

Table 3.1: F-score per class reached by *COSNet* on some functional classes in the experimental validation described in Section 2.7.

Class	Pfam-1	Pfam-2	Expr	PPI-BG	PPI-VM
Metabolism of Methionine (01.01.06.05)	0.227	0.346	0.350	0.074	0.193
Electron transport (02.11)	0.252	0.185	0.500	0.480	0.278
Sugar, glucoside metabolism (01.05.02)	0.412	0.453	0.039	0.313	0.388
Metabolism of glutamate (01.01.03.02)	0.320	0.454	0.008	0.046	0.000

This approach can be summarized as follows:

- Each network is associated with a *classification problem* in $\mathbb{R}^2$
- Each classification problem is solved through an efficient approximated algorithm, obtaining a measure of "*linear separability*" related to the

"predictiveness" of each network

- These measures are used to properly integrate the various networks by following a specific weighted integration schema

We do not focus on the development of new integration schemes, but just verify the effectiveness of LSI weighting by using those described in Section 3.2.

3.1 Estimating network weights

The main idea of the proposed approach consists in projecting the nodes of the network into a plane, in performing an efficient linear classification of positive and negative nodes in the projected bidimensional space, and in generating weights associated to the network according to the measured performances of the linear classifier.

More precisely, each network is represented by a weighted graph $G_d = \langle V_d, W^{(d)} \rangle$ for $d \in \{1, 2, \ldots, m\}$, where m is the number of networks, V_d set of genes and $w_{sk}^{(d)}$ a similarity measure between genes s and k in the network G_d.

The set V_d is halved in ($S^{(d)}$, $U^{(d)}$), where $S^{(d)}$ is the set of labeled nodes and $U^{(d)}$ is the set of unlabeled nodes. For each functional class c, a bipartition $(S_+^{(d)}, S_-^{(d)})$ of $S^{(d)}$ is known, where $S_+^{(d)}$ contains positive examples and $S_-^{(d)}$ contains those negative.

The algorithm can be set out in four main steps:

1. For each network G_d, each labeled node $k \in S^{(d)}$ is associated with a labeled point $\Delta^{(d)}(k) \equiv \left(\Delta_+^{(d)}(k), \Delta_-^{(d)}(k)\right) \in \mathbb{R}^2$, where

$$\Delta_+^{(d)}(k) = \sum_{j \in S_+^{(d)}} w_{kj}^{(d)}, \qquad \Delta_-^{(d)}(k) = \sum_{j \in S_-^{(d)}} w_{kj}^{(d)}.$$

   The label of $\Delta^{(d)}(k)$ is the label of k and we can define $I_+^{(d)}$ and $I_-^{(d)}$ as the set of positive and negative points respectively.

2. Positive and negative points are linearly separated by an efficient approximated algorithm which maximizes the *F-score* criterion.

3. The computed F-score $F_d(c)$ is then used to compute the weight $h^{(d)}$ for the network G_d when considering class c: $h^{(d)} = \frac{F_d(c)}{\sum_d F_d(c)}$.

4. The network weights found in the previous step are used for combining networks according to a specific integration schema.

First observe that the computation of points $\Delta^{(d)}(k)$ does not include nodes U and their connections with labeled nodes because we do not need in this case the restriction to subnetwork of labeled nodes.

Second, the algorithm at point 2 is slightly different than the second step of *COSNet* because we apply just the procedure `FindOptimalParameter` for computing the best F-score by considering both cases of positive half-plane above and below the separation line. Consequently, the linear separation algorithm at step 2 (see above) can be summarized as follows:

- The algorithm computes the slopes of the lines crossing the origin and each point $\Delta^{(d)}(k) \in I_+^{(d)} \cup I_-^{(d)}$. Then it searches the straight line which maximizes the F-score criterion by sorting the computed lines according to their slopes in an increasing order and by considering both positive halfplane cases (above and below the line).

- Let $\hat{m}$ be the computed best slope, then the algorithm computes the intercepts of the lines whose slope is $\hat{m}$ and crossing each point belonging to $I_+^{(d)} \cup I_-^{(d)}$.

- Finally the optimum line, and so the corresponding F-score $F_d(c)$, is identified by scanning the computed lines according to their intercept in an increasing order.

- Return the best computed F-score $F_d(c)$.

Now we experimentally show that this approach is effective in defining reliability weights for networked data sources. Table 3.2 reports the F-scores obtained in the second step of *COSNet* in the FunCat gene function prediction in yeast for each source by averaging the F-scores of nodes belonging to the subtrees rooted at the first level of the FunCat hierarchy 2.7.1. Note that this F-score is different from the F-score computed by running the network, and to avoid misunderstandings we name it F-score$_2$. Moreover, since the five considered data sets have a different number of genes, the number of considered functional classes can change from a data set to another. Consequently, the empty positions in the table means that for the corresponding data set there are no classes in the functional tree with at least 20 annotated genes.

Some data sources are characterized by higher F-score$_2$, meaning that the corresponding classification in step 2 is "less difficult" than the other ones. Moreover, we can observe that the F-score$_2$ rank among data sources changes according to the considered functional tree.

Table 3.2: Average F-score$_2$ separated by data set and by functional tree.

FunCat Tree	Data Set				
	Pfam-1	*Pfam-2*	*Expr*	*PPI-BG*	*PPI-VM*
Metabolism	0.4553	0.4009	0.1702	0.2838	0.4493
Energy	0.4864	0.4255	0.2153	0.4080	0.4111
Cell Cycle and DNA Processing	0.3423	0.2774	0.1775	0.4795	0.4289
Transcription	0.3630	0.2985	0.1632	0.5346	0.5410
Protein Synthesis	0.5311	0.5264	0.2401	0.5059	0.5818
Protein Fate	0.5969	0.5379	0.1748	0.4268	0.4942
Protein with Binding Function or Cofactor Requirement	0.5040	0.4541	0.1803	0.3420	0.4079
Regulation of Metabolism and Protein Function	0.4822	0.4092	0.0834	0.1936	0.2396
Cellular Transport, Transport Facilitation and Routes	0.5130	0.4286	0.1472	0.4305	0.4945
Cellular Communication/Signal Transduction Mechanism	0.5342	0.4568	0.0749	0.4148	0.2933
Cell Rescue, Defense and Virulence	0.3804	0.3045	0.1359	0.2758	0.2560
Interaction with the Environment	0.3780	0.3162	0.1324	0.4058	0.4660
Transposable Elements, Viral and Plasmid Proteins			0.6087	0.5000	
Cell Fate	0.3548	0.2646	0.0969	0.4060	0.3962
Development	0.3636	0.2727	0.2174	0.2020	0.3077
Biogenesis of Cellular Components	0.3558	0.2510	0.1509	0.5049	0.4170
Cell Type Differentiation	0.4121	0.3155	0.1955	0.4251	0.4290

In Table 3.3 we show the average F-score computed by running the *COSNet* algorithm, corresponding to the average F-score$_2$ computed in Table 3.2. In this table too we can observe that a source can be much informative for a tree but less predictive for another one. We found that there is a strict correlation between the F-score and the F-score$_2$, i.e. the higher the F-score$_2$ the higher the predictive ability of the algorithm on that source. This is also confirmed by the Pearson correlation coefficients of the two vectors F-score and F-score$_2$: Pfam-1 0.99, Pfam-2 0.98, Expr 0.98, PPI-BG 0.996, PPI-VM 0.99. In Figure 3.2 we graphically report the results of this comparison for some functional trees.

These results show that we can know in advance which dataset is more informative for a functional class by evaluating the corresponding F-score$_2$, and we can also justify from an experimental standpoint the feasibility of our approach to compute weights that represent the effectiveness of each source of data for the prediction of a given functional class.

Table 3.3: Average F-score separated by data set and by functional tree.

FunCat Tree	Data set				
	Pfam-1	*Pfam-2*	*Expr*	*PPI-BG*	*PPI-VM*
Metabolism	0.3992	0.3684	0.1294	0.2251	0.3917
Energy	0.4587	0.3712	0.1774	0.3469	0.3471
Cell Cycle and DNA Processing	0.2901	0.2214	0.1574	0.4336	0.3751
Transcription	0.3207	0.2700	0.1278	0.4988	0.4966
Protein Synthesis	0.4762	0.4977	0.1207	0.4641	0.5090
Protein Fate	0.5441	0.4972	0.1429	0.3699	0.4562
Protein with Binding Function or Cofactor Requirement	0.4631	0.4155	0.1324	0.3086	0.3648
Regulation of Metabolism and Protein Function	0.4351	0.3564	0.0485	0.1395	0.2043
Cellular Transport, Transport Facilitation and Routes	0.4580	0.3882	0.1195	0.3834	0.4596
Cellular Communication/Signal Transduction Mechanism	0.4579	0.4113	0.0313	0.3602	0.2528
Cell Rescue, Defense and Virulence	0.3160	0.2569	0.0951	0.2264	0.2090
Interaction with the Environment	0.3458	0.2871	0.0830	0.3360	0.4001
Transposable Elements, Viral and Plasmid Proteins			0.6087	0.4489	
Cell Fate	0.3375	0.2274	0.0749	0.3671	0.3717
Development	0.2820	0.1739	0.1463	0.1417	0.2258
Biogenesis of Cellular Components	0.3031	0.1796	0.1265	0.4653	0.3773
Cell Type Differentiation	0.3806	0.2989	0.1405	0.4034	0.4135

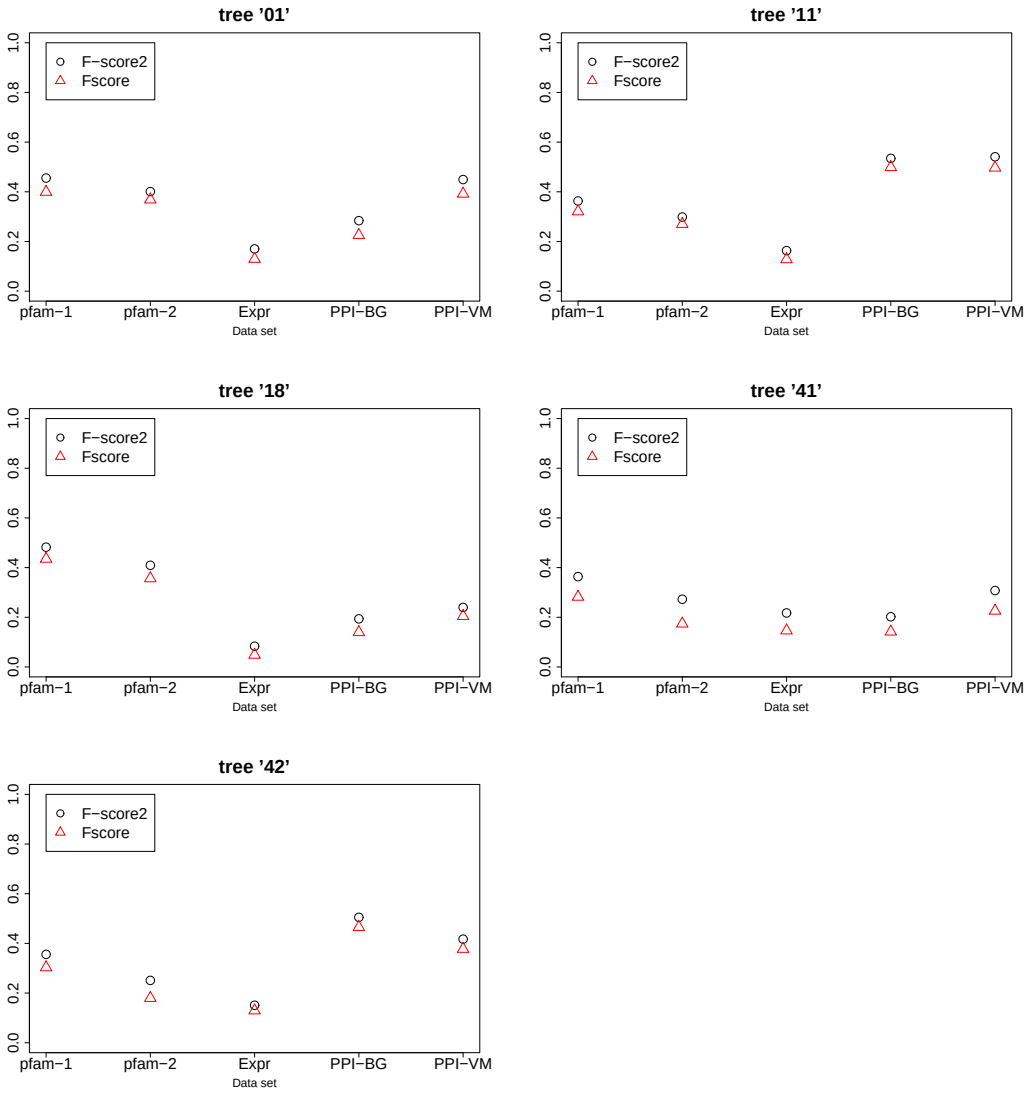


Figure 3.2: Relationship between F-score$_2$ and F-score averaged on the functional trees 01, 11, 18, 41 and 42.

3.2 Data Integration Schemes

The computed weights $h^{(d)}$ can be used in different weighted integration schemes to construct composite networks and to improve the predictive capability and the reliability of the predictions. In this section we briefly describe some weighted and unweighted schemes used in our experiments.

Weighted sum (WS). This scheme assumes that each network $\boldsymbol{W}^{(d)}$ is associated with a reliability weight $h^{(d)}$, previously computed. Each entry of the consensus matrix is obtained simply by the weighted sum, according to the $h^{(d)}$ values, of each entry of the m matrices:

$$\boldsymbol{W}^* = \sum_{d=1}^{m} h^{(d)} \boldsymbol{W}^{(d)} \tag{3.1}$$

Max fusion (MF). Each entry of the consensus matrix is chosen by selecting the corresponding entry of the data for which we have the maximum value of the coefficient $h^{(d)}$. Of course, only entries w_{ij}^d for which do exist data of the source d for both i and j are considered for the selection. That is for each edge (i, j) we compute:

$$w_{ij}^* = w_{ij}^{\bar{d}(i,j)} \tag{3.2}$$

where $\bar{d}(i,j) = \arg\max_d \left(h^{(d)} | w_{ij}^d \neq 0 \right)$

Unweighted sum (US). This is the simple sum of the single networks divided by the number of available networks:

$$\boldsymbol{W}^* = \frac{1}{m} \sum_{d=1}^{m} \boldsymbol{W}^{(d)} \tag{3.3}$$

Max unweighted fusion (MUF). The consensus matrix $\boldsymbol{W}^*$ is obtained element by element by taking the maximum edge weight

$$w_{ij}^* = \max_d w_{ij}^d \tag{3.4}$$

3.3 Experimental setting

COSNet with *LSI*-weight integration has been applied to genome wide prediction of gene functions using GO and FunCat ontologies with *S.cerevisiae*

(yeast) and *M. musculus* (mouse) model organisms. In this Section we describe the experimental setting for mouse organism, since for yeast we adopt the same set-up described in Section 2.7.

Mouse data has been collected from the *MouseFunc* benchmark data [76]. We used the same data and the same annotations (GO annotations 17 February 2006; version 1.612) available from the MouseFunc website [1], as well as the same experimental set-up, in order to perform a fair comparison with other state-of-the-art supervised and semi-supervised gene function prediction methods that participated to the MouseFunc challenge.

According to the protocol proposed for the MouseFunc challenge, different sources of data, including protein sequence pattern annotations, protein-protein interactions, phenotype annotations, phylogenetic profiles, gene expression across multiple tissues, disease associations have been applied to predict the functions of a set of genes belonging to 2815 GO terms with a number of annotations ranging form 3 to 300, to avoid classes with a too low number of positive examples, or too generic classes characterized by a too large number of annotations. Mouse Genomics Informatics (MGI) annotations for 21603 mouse genes have been considered, excluding GO annotations based solely on the "inferred from electronic annotation" (IEA) evidence code.

Prediction tasks. Following the MouseFunc protocol we performed the held-out genes task: a randomly selected set of 1718 genes, listed in the file TestSet.txt available from the MouseFunc website, is held-out and their annotations have to be predicted using the data and the annotations of the remaining genes.

We used as positives for a given GO term all the genes annotated to that functional class. As negatives, at first we simply selected all the genes not annotated for that class (Base strategy). Regarding the training set for each GO term we selected only genes labeled for that class.

Data sets. To perform our validation tests, we use different data sources from [76], which we briefly describe below:

- *Expression data.* Expression data from oligonucleotide arrays for 13,566 genes across 55 mouse tissues [91]; expression data from Affymetrix arrays for 18,208 genes across 61 mouse tissues [90]; tag counts at quality 0.99 cut-off from 139 SAGE libraries for 16,726 genes [92].

[1] $http://hugheslab.med.utoronto.ca/supplementary-data/mouseFunc\_I/$

- *Sequence patterns.* Protein sequence pattern annotations from Pfam-A (release 19) for 15,569 genes with 3,133 protein families [93]; protein sequence pattern annotations from InterPro (release 12.1) for 16,965 genes with 5,404 sequence patterns [94]

- *Protein interactions.* Protein-protein interactions from OPHID for 7,125 genes [95] (downloaded on 20 April 2006)

- *Phenotypes.* Phenotype annotations from MGI for 3,439 genes with 33 phenotypes [96] (downloaded on 21 February 2006 from [97])

- *Conservation profile.* Conservation pattern from Ensembl (v38) for 15,939 genes across 18 species [98]; conservation pattern from Inparanoid (v4.0) for 15,703 genes across 21 species [99]

- *Disease associations.* Disease associations from OMIM for 1,938 genes to 2,488 diseases/phenotypes [100, 101] (downloaded on 6 June 2006 from [102])

Pre-processing. When constructing functional association networks we distinguish three types of data: binary and continuous valued and PPI interaction data.

For binary data, if β is the proportion of genes for which a given feature has value equal to 1, then all ones were replaced with $-\log(\beta)$ and zeros with $\log(1-\beta)$. In this way the "weight" of very uncommon features is emphasized [36]. Then the score for each gene pair is set to the Pearson's correlation coefficient of the corresponding feature vectors.

For continuous data we adopt directly the pairwise Pearson's correlation coefficient, and the squared correlation for gene expression data in order to take in account the down regulation of a gene by the other one.

Finally for PPI interaction data we construct the pairwise interaction scores using the approach proposed in [64], where the similarity score for genes i and j is

$$S_{ij} = \frac{2|N_i \cap N_j|}{|N_i \setminus N_j| + 2|N_i \cap N_j| + 1} \times \frac{2|N_i \cap N_j|}{|N_j \setminus N_i| + 2|N_i \cap N_j| + 1}$$

where N_k is the set of the neighbors of gene k (k is included).

To maintain sparse the resulting association networks, we set for each network an edge threshold such that each node has at least one neighbour. Hence, having n genes we obtained $n \times n$ sparse matrices $\boldsymbol{W}^{(d)}$, $1 \leq d \leq m$, where m is the number of the different data sources considered.

Each network $\boldsymbol{W}^{(d)}$ has been normalized by dividing each entry $\boldsymbol{W}_{ij}^{(d)}$ by the square root of the product of the the sum of the elements of row i and the sum of elements in column j. In other words, if D is a $n \times n$ diagonal matrix such that $D_{ii} = \sum_j \boldsymbol{W}_{ij}^{(d)}$ then the normalized matrix $\hat{\boldsymbol{W}}^{(d)}$ is the normalized Laplacian of the graph:

$$\hat{\boldsymbol{W}}^{(d)} = D^{-1/2} \boldsymbol{W}^{(d)} D^{-1/2} \tag{3.5}$$

3.4 Results and Discussion

3.4.1 GFP in Yeast

Disjunctive approach. With regard to the experimental setting described in Section 2.7, we integrate the yeast networks, Pfam-1, Pfam-2, Expr, PPI-BG, PPI-VM with a disjunctive approach, i.e. by considering all the genes in at least one of the single networks, obtaining a set of 4665 genes and 232 functional classes with at least 20 annotated genes.

At first, each single network is extended to this size by adding rows and columns made up by zeros, and then the extended networks are integrated by adopting both weighted and unweighted techniques described in Section 3.2: weighted sum (*WS*), max fusion (*MF*), unweighted sum (*US*) and max unweighted fusion (*MUF*).

To define the network reliability weights needed by the weighted schemes we use both *LSI* and the integration algorithm *Simultaneous Weights* (SW) recently proposed by Mostafavi and coworkers [37], based on a previous algorithm for gene function prediction, *GeneMANIA* [36], and whose code can be downloaded at http://morrislab.med.utoronto.ca/ sara/SW/. The *GeneMANIA-SW* method is made up by two steps: 1) integration of multiple sources in a composite network by using a ridge regression approach; 2) gene function prediction based on a label propagation algorithm based on Gaussian Random Fields [34].

The input of the *GeneMANIA-SW* software is given by the networks to be integrated and by the corresponding label matrix, the output consists of either the vector of reliability weights or the precision at 20% recall and the area under the ROC curve (AUC) relative to the prediction step.

Note that this algorithm can provide a unique weight vector $\boldsymbol{v}$ relative to all the functional classes included in the label matrix or a specific weight vector $\boldsymbol{v}_c$ for each functional class c. We tested *GeneMANIA-SW* algorithm either by building a single composite network (using $\boldsymbol{v}$) for all the functional classes (in the following *strategy a*), or by building a composite network for each class c separately using v_c vector (in the following *strategy b*). Clearly the latter approach is more expensive from a computational standpoint but in general may lead to better results.

In order to compare *COSNet* with *GeneMANIA-SW*, we also compute precision at 20% recall level and AUC in addition to accuracy, F-score, precision and recall. These performance measures are computed by using the scores defined by equation (2.14) in Section 2.6.3. Finally we adopt a stratified 10-fold cross validation procedure for validating *COSNet*, by ensuring

Table 3.4: Comparison of *COSNet* performance on different weighted and unweighted integration strategies and the *GeneMANIA-SW* algorithm.

Method	Performance measures					
	Prec	Rec	F	AUC	P20R	
COSNet - US	0.501	0.508	0.490	0.854	0.709	$\beta = 0$
	0.514	0.498	0.497	0.854	0.722	$\beta = 0.0001$
COSNet - MUF	0.494	0.505	0.484	0.855	0.712	$\beta = 0$
	0.508	0.496	0.494	0.858	0.719	$\beta = 0.0001$
COSNet - WS_SW	0.163	0.626	0.197	0.753	0.304	$\beta = 0$
	0.304	0.375	0.330	0.820	0.420	$\beta = 0.0001$
COSNet - LSI_WS a)	0.510	0.514	0.497	0.861	0.738	$\beta = 0$
	0.517	0.498	0.499	0.864	0.742	$\beta = 0.0001$
COSNet - LSI_WS b)	0.519	0.509	0.499	0.846	0.743	$\beta = 0$
	0.529	0.499	0.505	0.850	0.744	$\beta = 0.0001$
COSNet - LSI_MF a)	0.394	0.352	0.343	0.732	0.519	$\beta = 0$
	0.413	0.322	0.350	0.732	0.534	$\beta = 0.0001$
COSNet - LSI_MF b)	0.462	0.451	0.438	0.800	0.611	$\beta = 0$
	0.495	0.446	0.456	0.802	0.677	$\beta = 0.0001$
GeneMANIA-SW a)				0.835	0.571	
GeneMANIA-SW b)				0.891	0.602	

that each fold includes at least one positive example for each classification task.

For *LSI* method we compute the network reliability weights by considering each functional class separately. It means that for each class c_i we obtain a weight vector $\boldsymbol{v}_i$. We adopt two strategies for integrating yeast networks using vectors $\boldsymbol{v}_i$:

a) generating a unique composite network using as network weights the vectors $\boldsymbol{v}_i$ averaged across the functional classes

b) generating one integrated network for each class c_i by using $\boldsymbol{v}_i$ as network weights

Table 3.4 shows the performance of *GeneMANIA-SW* and of *COSNet* applied to all the different integrated networks. The results are averaged across all the functional classes. Note that precision, recall and F-score for *GeneMANIA-SW* are not reported since the original algorithm provides in output only AUC, P20R and network weights.

First, for unweighted schemes, we observe that the *US* and *MUF* obtain similar performances, with a slightly better precision and F-score for *US* and a slightly better AUC for MUF. It is worth noting that in the GFP

context AUC is less important than F-score because AUC does not properly take into account the unbalance between positive and negative examples.

Second, the performance improvements of regularized version of *COSNet* w.r.t the unregularized one are confirmed, with better results in all the considered performance measures up to the recall one. In particular, having a higher precision and P20R is a valuable result in the GFP context as explained in Section 2.7.

The best performances are obtained with weighted schemes which use LSI weights. In particular, the integration strategy b), which considers an integrated matrix for each functional class, obtains the highest precision, F-score and P20R. Moreover, the performance improvement of the strategies *LSI_WS b)* and *LSI_MF b)* w.r.t. the strategies *LSI_WS a)* and *LSI_WS a)* respectively shows the effectiveness of LSI weights.

What seems really unexpected is the behaviour of *COSNet* on the weighted sum integration using SW weights (WS_SW), where the method obtains the worst results.

To understand the reasons of this behaviour, we observe that the reliability weights associated by SW (strategy a) with the five data sets are: 1.91, 17.91, 0, 0.55, 0.48 respectively for *PPI-BG*, *Expr*, *pfam-1*, *pfam-2*, *PPI-VM* data sets. In particular the data set *pfam-1* has weight 0 which means that it is discarded during the integration phase, whereas *COSNet* obtain the highest F-score just on this data set.

The performance decay for label propagation methods on pfam-1 data is also confirmed by the *LP-Zhu* method which has F-score 0.012, as though this methodology of semi-supervised learning was not able to exploit the information embedded in protein domain similarities.

We can also observe that *COSNet* markedly outperforms *GeneMANIA-SW* in P20R with both integration strategies a) and b), whereas it has a slightly lower AUC than *GeneMANIA-SW* when strategy b) is used. Figure 3.3 graphically show the AUC and P20R results contained in Table 3.4. It is immediate to recognize that, despite of similar AUC values for *COSNet* and *GeneMANIA-SW*, our method highly overcomes *GeneMANIA-SW* in precision at 20% of recall level. It means that *COSNet*, by exploiting its inherently cost-sensitive nature, is able to discover real positive annotations (high recall) by preserving a high precision.

Finally, we want to compare *COSNet* and *GeneMANIA-SW* also in terms of the time needed for the computation of all the prediction tasks. In Table 3.5 we report the time in seconds needed by the two algorithms for computing 10-folds cross validation on each functional class (for *COSNet* we report only the network integrated with LSI_WS scheme, since the

time needed by the other composite networks is very similar). We point out that, for having a fair comparison, we excluded from the counting for *GeneMANIA-SW* the time needed by the first step (integration of the multiple sources). *COSNet* is faster than *GeneMANIA-SW* in both integration approaches, with the integration strategy b) slower than the strategy a) for both the methods.

Conjunctive approach. We have also analyzed the performance of our algorithm on common genes, i.e. the genes belonging to each of the considered data sets, in order to compare *COSNet* with hierarchical methods that have been applied to the same prediction task. With this approach we obtain 1901 genes and 168 functional classes with at least 20 annotations for selected genes.

We compared *COSNet* with a hierarchical multi-label cost-sensitive algorithm based on kernel fusion techniques (*HB-CS*) [89] and with a hierarchical

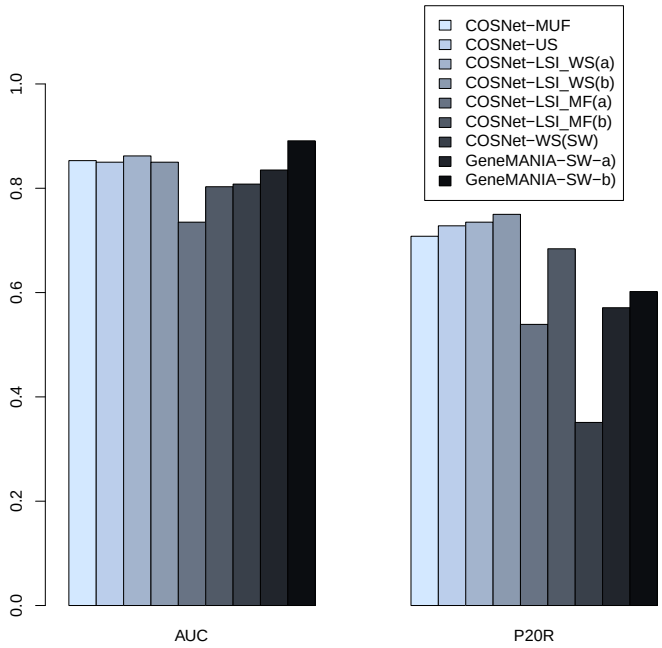


Figure 3.3: Average AUC and P20R of *COSNet* (regularized version) on each integrated network and *GeneMANIA-SW* methods.

Table 3.5: Overall time in seconds needed by *COSNet* and *GeneMANIA-SW* algorithm for computing 10-folds cross validation on each functional class on integrated data.

Method	Dataset	
	LSI_WS a)	LSI_WS b)
COSNet	1010.3	2321.3
	GeneMANIA-SW a)	GeneMANIA-SW b)
	2326.1	3758.3

ensemble method based on the "true path rule" (*TPR*) [103], which have been proven being among the best methods for this task [104].

The results are summarized in Table 3.6. For *HB-CS* and *TPR* we report only the averaged precision, recall and F-score because the remaining measures have not been provided by the authors of the corresponding paper.

COSNet has a higher F-score than both *HB-CS* and *TPR* and this is quite surprising if we observe that these are hierarchical methods, i.e. they use the information embedded in the hierarchical structure (forest of trees) of the functional classes and their constraints (true path rule). Moreover, *COSNet* obtains very balanced precision and recall values, while *HB-CS* has the lowest recall and *TPR* has the lowest precision.

The weighted sum strategies which use LSI weights (*LSI_WS*) work better that the other integration approaches, and the *MF* schema seems to be not appropriate for data integration, since it has performances worse that the unweighted approaches.

In conclusion, we tested LSI weights with two different weighted integration schemes with both conjunctive and disjunctive approach and the obtained performance improvement w.r.t the unweighted integration shows the effectiveness of LSI algorithm in finding for each network the corresponding level of "predictiveness". Clearly the potential of LSI algorithm can be exploited better by analyzing other appropriate integration schemes.

Moreover, we adopted just one optimization criterion (F-score) when finding the optimal line (see Section 3), but other criterions may be adopted and may be more effective in defining network reliability weights.

3.4.2 GFP in Mouse

In this section we report the results on mouse organism relative to the experimental set-up described in Section 3.3.

Table 3.6: Performance comparison of *COSNet*, *HB-CS* and *TPR* considering solely genes common to each data set.

Method	Performance measures					
	Prec	Rec	F	AUC	P20R	
COSNet - US	0.581	0.610	0.585	0.899	0.816	$\beta = 0$
	0.595	0.605	0.591	0.899	0.826	$\beta = 0.0001$
COSNet - MUF	0.583	0.602	0.579	0.899	0.817	$\beta = 0$
	0.596	0.589	0.583	0.898	0.817	$\beta = 0.0001$
COSNet - WS_SW	0.224	0.674	0.267	0.798	0.461	$\beta = 0$
	0.426	0.466	0.436	0.863	0.574	$\beta = 0.0001$
COSNet - LSI_WS (a)	0.603	0.613	0.596	0.907	0.847	$\beta = 0$
	0.605	0.606	0.597	0.908	0.851	$\beta = 0.0001$
COSNet - LSI_WS (b)	0.613	0.601	0.595	0.894	0.848	$\beta = 0$
	0.617	0.598	0.597	0.898	0.851	$\beta = 0.0001$
COSNet - LSI_MF (a)	0.459	0.375	0.388	0.745	0.617	$\beta = 0$
	0.479	0.361	0.392	0.745	0.629	$\beta = 0.0001$
COSNet - LSI_MF (b)	0.553	0.536	0.524	0.840	0.757	$\beta = 0$
	0.568	0.536	0.535	0.846	0.791	$\beta = 0.0001$
HB-CS	0.648	0.504	0.550			
TPR	0.480	0.634	0.503			

We integrate the mouse single networks by using a disjunctive approach, i.e. by considering all the genes in at least one of the single networks, obtaining a set of 21603 genes. Since the single networks have a size smaller than 21603, each single network is extended to this size by adding rows and columns made up by zeros; then the extended networks are integrated by adopting the weighted sum (*WS*), max fusion (*MF*) and unweighted sum (*US*) integration schemes described in Section 3.2. Moreover, among the 2815 GO classes with 3-300 annotations, we choose those with at least one annotation in the test set, obtaining 1174 BP, 442 MF and 231 CC terms.

To define the network reliability weights needed by the weighted schemes we compute the LSI network weights by considering each functional class separately. It means that for each class c_i we obtain a weight vector $\boldsymbol{v}_i$. We applied two strategies for integrating mouse networks using vectors $\boldsymbol{v}_i$:

a) generating a unique composite network for each GO ontology by using as network weights the vectors $\boldsymbol{v}_i$ averaged across the corresponding ontology classes

b) generating a integrated network for each class c_i by using $\boldsymbol{v}_i$ as network weights

We compared *COSNet* with the MouseFunc I challenge participant methods,

Table 3.7: MouseFunc I participants.

Group	Authors	Algorithm
Group A	G. Obozinski, C. Grant, J. Qiu, G. Lanckriet, M. I. Jordan and W. S. Noble	Calibrated ensembles of SVMs [8]
Group B	H. Lee, M. Deng, T. Chen, F. Sun	An Integrated Kernel-Logistic Regression Method for Protein Function Prediction [105]
Group C	S. Mostafavi, D. W. Farley, C. Grouios, D. Ray and Q. Morris	GeneMANIA [36]
Group D	Y. Guan, C. L. Myers, O. G. Troyanskaya	Multi-label hierarchical classification [55] and Bayesian integration of diverse data sources [106]
Group E	W. K. Kim, C. Krumpelman, E. Marcotte	Combination of classifier ensemble and gene network [107]
Group F	T. Joshi, C. Zhang, G. N. Lin, D. Xu	GeneFAS [108, 109]
Group G	W. Tian, M. Tasan, F. D. Gibbons, F. P. Roth	Funckenstein [110]
Group H	Y. Qi, J. K. Seetharaman and Z. B. Joseph	Protein Function Prediction Using 'Query Retrieval' Methods [111]
Group I	M. Leone, A. Pagnani	Function prediction with message passing algorithms [112]

whose prediction scores are available on the MouseFunc website [2]. Table 3.7 contains the description of the nine groups which participated to the MouseFunc I challenge. Note that for group I the available scores do not include all the functional classes needed for the comparison.

For each GO term we computed the area under the receiver operating characteristics (ROC) curve (AUC), the precision at different level of recall (e.g. the precision at 20% recall (P20R)) and the F-score measure. We point out that AUC and P20R are measures suitable for ranking methods, that is methods which provide real prediction scores, while F-score needs binary predictions for being computed. Accordingly, since *COSNet* is a binary classifier, we have adapted it for providing also prediction scores (see Section 2.6.3), but it is clear that *COSNet* results in P20R and AUC are suboptimal, and more fine strategies for computing scores can be studied.

On the other hand, the MouseFunc I participant algorithms are not binary classifiers, therefore for computing the F-scores these methods, since they provide for each gene i just a score $s_i \in [0, 1]$, we first scale these scores in the interval [0,1] by using the following equation:

$$s_i^* = \frac{s_i - min(s)}{max(s) - min(s)}$$

where $min(\boldsymbol{s}) = \min_i s_i$ and $max(\boldsymbol{s}) = \max_i s_i$. In this way the lowest

[2] $http://hugheslab.med.utoronto.ca/supplementary-data/mouseFunc\_I/$

Table 3.8: Performance comparison of MouseFunc I methods and regularized *COSNet* ($\beta = 0.0001$). The values are averaged across the three GO ontologies.

Group	Performance								
	BP			MF			CC		
	AUC	P20R	F	AUC	P20R	F	AUC	P20R	F
Group A	0.672	0.204	0.113	0.796	0.470	0.340	0.766	0.284	0.163
Group B	0.709	0.204	0.113	0.790	0.469	0.328	0.737	0.334	0.197
Group C	0.858	0.314	0.175	0.929	0.607	0.406	0.890	0.479	0.281
Group D	0.825	0.320	0.140	0.894	0.591	0.346	0.872	0.423	0.229
Group E	0.809	0.209	0.028	0.870	0.492	0.170	0.845	0.366	0.208
Group F	0.742	0.203	0.104	0.848	0.529	0.340	0.795	0.343	0.198
Group G	0.810	0.351	0.188	0.890	0.653	0.434	0.846	0.467	0.231
Group H	0.759	0.194	0.091	0.859	0.462	0.322	0.805	0.297	0.143
COSNet -US	0.726	0.265	0.158	0.842	0.466	0.316	0.810	0.384	0.260
COSNet -LSI_MF (a)	0.696	0.235	0.126	0.876	0.572	0.376	0.737	0.327	0.210
COSNet -LSI_WS (a)	0.716	0.272	0.164	0.867	0.547	0.387	0.803	0.413	0.280

score in $\boldsymbol{s}$ corresponds to 0 and the maximum score in $\boldsymbol{s}$ corresponds to 1. Then we set a threshold for scores at 0.5, i.e. genes corresponding to scores greater than 0.5 are predicted as positive and the remaining genes are predicted in the negative class. We outline that this is a naive technique for computing binary labels which might be suboptimal for some of the compared algorithms; method-specific techniques to set the thresholds may lead to significantly better results.

For each GO ontology, i.e. Biological Process (BP), Molecular Function (MF), Cellular Component (CC), according to the MouseFunc evaluation protocol, we considered four ranges of specificity, that is the number of genes in the training set with which each term is annotated: {3..10}, {11..30}, {31..100}, {101..300}, for a total of 12 evaluation categories. We also considered the average performances across the classes in the three ontologies.

In table 3.8 we show the overall average results of the MouseFunc I challenge participants and of the regularized version of *COSNet* ($\beta = 0.0001$) applied to the integration schemes unweighted sum, max weighted fusion and weighted sum (LSI weighting). First we note that, even with the simple unweighted sum integration, *COSNet* overcomes in P20R and F-score the methods A, B, E, F, H over CC and BP classes, which represent the around the 75% of the total number of classes.

Second, *COSNet* performance improves when using *WS* weighted scheme, whereas with *MF* scheme there is an improvement w.r.t. the *US* scheme just

Table 3.9: Statistically significant differences in F-score performance at $\alpha = 10^{-2}$ significance level between *COSNet* and the other methods. The symbol "+" means a difference statistically significant in favour of *COSNet*, "=" means no statistically significant difference, "-" means a difference statistically significant in favour of the other method.

Group	BP	MF	CC
Group A	+	+	+
Group B	+	+	+
Group C	=	=	=
Group D	=	+	+
Group E	+	+	+
Group F	+	+	+
Group G	=	=	+
Group H	+	+	+

in MF ontology. With regard to the *WS* scheme results, *COSNet* obtains the third top F-score in BP and MF ontologies and the second top F-score in CC ontology.

Moreover, in order to understand whether there is a statistically significant difference in performance, we performed the Wilcoxon signed-ranks test [83] on the F-score results. In Table 3.9 we report which difference is statistically significant (at $\alpha = 10^{-2}$ significance level) comparing *COSNet* with the other group methods in each domain separately. *COSNet* improvements are statistically significant w.r.t. all the methods and all the domains up to group D in domain BP. We think these positive results are due either to the cost-sensitive strategy or to the effective synergy between *COSNet* and LSI algorithms.

Finally, *COSNet* has competitive performance also w.r.t the P20R measure, where it is the 4^{th} top method over all the functional ontologies.

Overall, we point out that two out of three methods that perform better than *COSNet* in terms of P20R, Group D and Group G methods, are hierarchical methods: they also consider the hierarchical structure of classes, whereas our method predict with flat approach. Observe that applying simple hierarchical post-prediction reconciliating techniques for eliminating inconsistencies in parent-child predictions may significantly improve the performances [103]. The other method which outperforms *COSNet* in P20R, *GeneMANIA* (Group C), has been already compared with *COSNet* in Section 3.4.1, where our algorithm prominently outperformed *GeneMANIA*.

This means that on the task described in this section *COSNet* can less exploit its cost sensitive nature. Furthermore, w.r.t. the F-score results, there is no statistically significant difference between *COSNet* and the two methods (group C and G) that slightly outperform it, as shown in Table 3.9.

Finally, we observe that AUC results are less important in this context, as explained also in the previous sections.

In order to better analyze the obtained results, in figures 3.4, 3.5 and 3.6 we report the results averaged across the 12 considered GO categories in terms of AUC, P20R and F-score respectively. We point out that the problem of predicting gene functions has different difficulty across the considered categories, since more specific classes (i.e. classes with less positive examples) are more difficult to predict.

By considering P20R and F-score measures, an important aspect which arises is that *COSNet* tends to work better on 31..100 and 101..300 categories and suffers a slight decay in performance on 3..10 and 11..30 categories. In fact, our method obtains the best P20R in the BP 101..300 category and the second top P20R in the MF 31..100, 101..300 and CC 101..300 categories. Moreover, *COSNet* has the best F-score in the categories BP 31..100, 101..300, MF 101..300, CC 31..100 and CC 101..300, and it is the second top method in MF 31..100 and CC 11..30 categories. This may depend on the fact that the excessive low number of positive examples reduces the effectiveness of the cost-sensitive strategy adopted by *COSNet*.

Although the performance of our method is already comparable with that of the other methods, we observe that *COSNet* performance can be further improved by adopting the strategy b) for integrating input networks. This strategy is more expensive than strategy a) from a computational standpoint, since it generates an integrated network for each considered GO term, i.e. 1847 different integrated networks. Due to the large size of each network, our machine is forced to use swap memory for computing the weighted sum and the normalization of single networks, making the computation very slow. On the contrary, the computation of *LSI* weights takes a short time.

Nevertheless, in order to have a clue about the potential of this approach, we applied strategy b) integration to the CC 101..300 category, which is the smallest category as it contains just 30 GO terms. Table 3.10 shows the obtained results: there is an improvement in all the adopted performance measures when integration strategy b) is used. This result shows on one hand the effectiveness of LSI algorithm in defining network reliability weights, on the other hand that the averaged result in Table 3.8 relative to LSI_WS a)

Table 3.10: Performance comparison of MouseFunc I methods and regularized *COSNet* ($\beta = 0.0001$) averaged across the CC 101..300 category.

Group	Performance		
	AUC	P20R	F
Group A	0.827	0.456	0.287
Group B	0.855	0.478	0.265
Group C	0.842	0.559	0.310
Group D	0.876	0.452	0.290
Group E	0.797	0.218	0.171
Group F	0.748	0.410	0.237
Group G	0.867	0.465	0.133
Group H	0.723	0.214	0.061
COSNet -US	0.837	0.476	0.352
COSNet -LSI_MF (a)	0.788	0.386	0.286
COSNet -LSI_WS (a)	0.816	0.498	0.371
COSNet -LSI_WS (b)	0.826	0.503	0.375

can be substantially improved when considering integration strategy b).

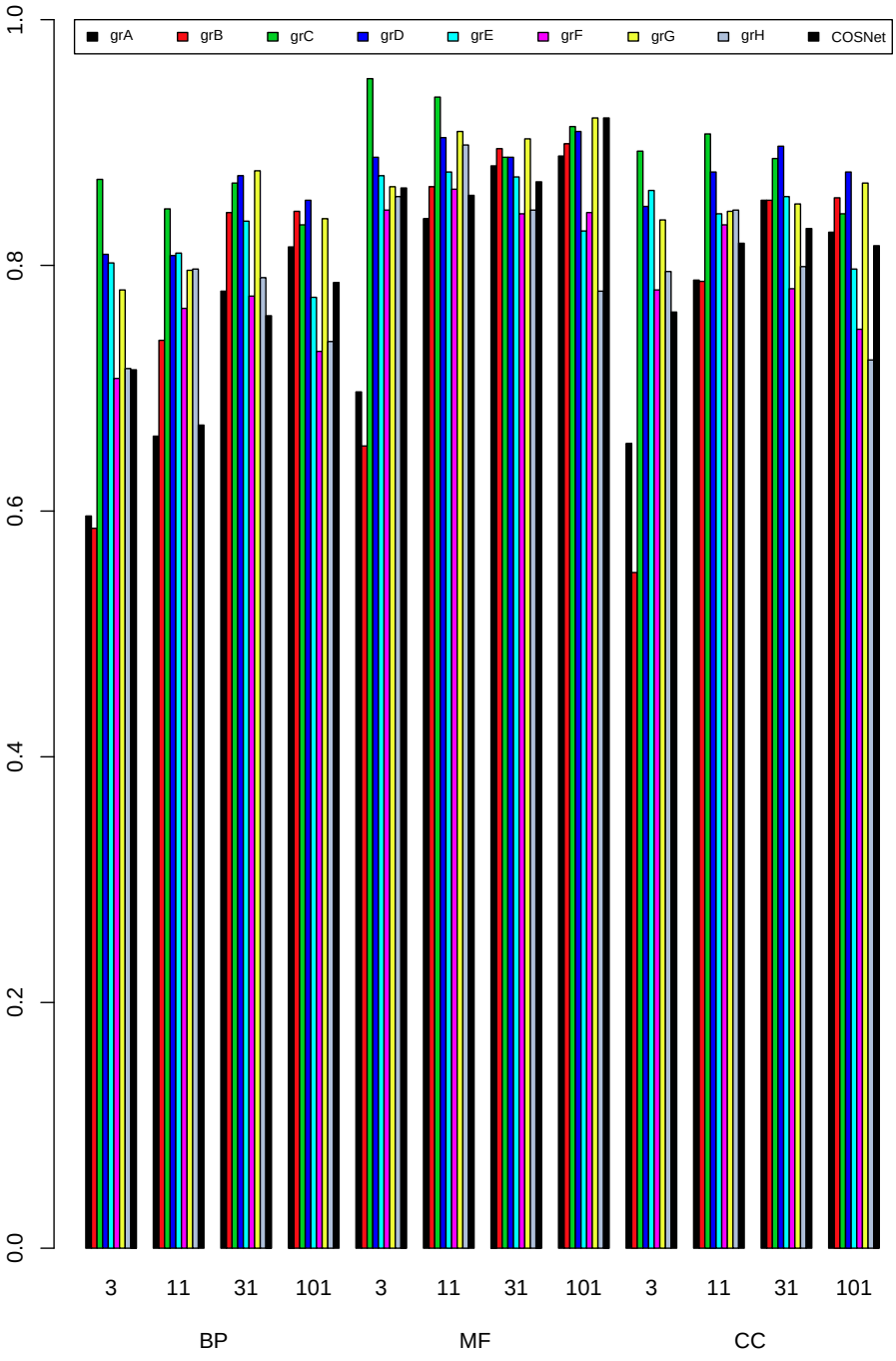


Figure 3.4: Comparison of the MouseFunc I challenge methods and regularized *COSNet* with strategy LSI-WS a) in terms of AUC averaged across all the twelve considered GO categories.

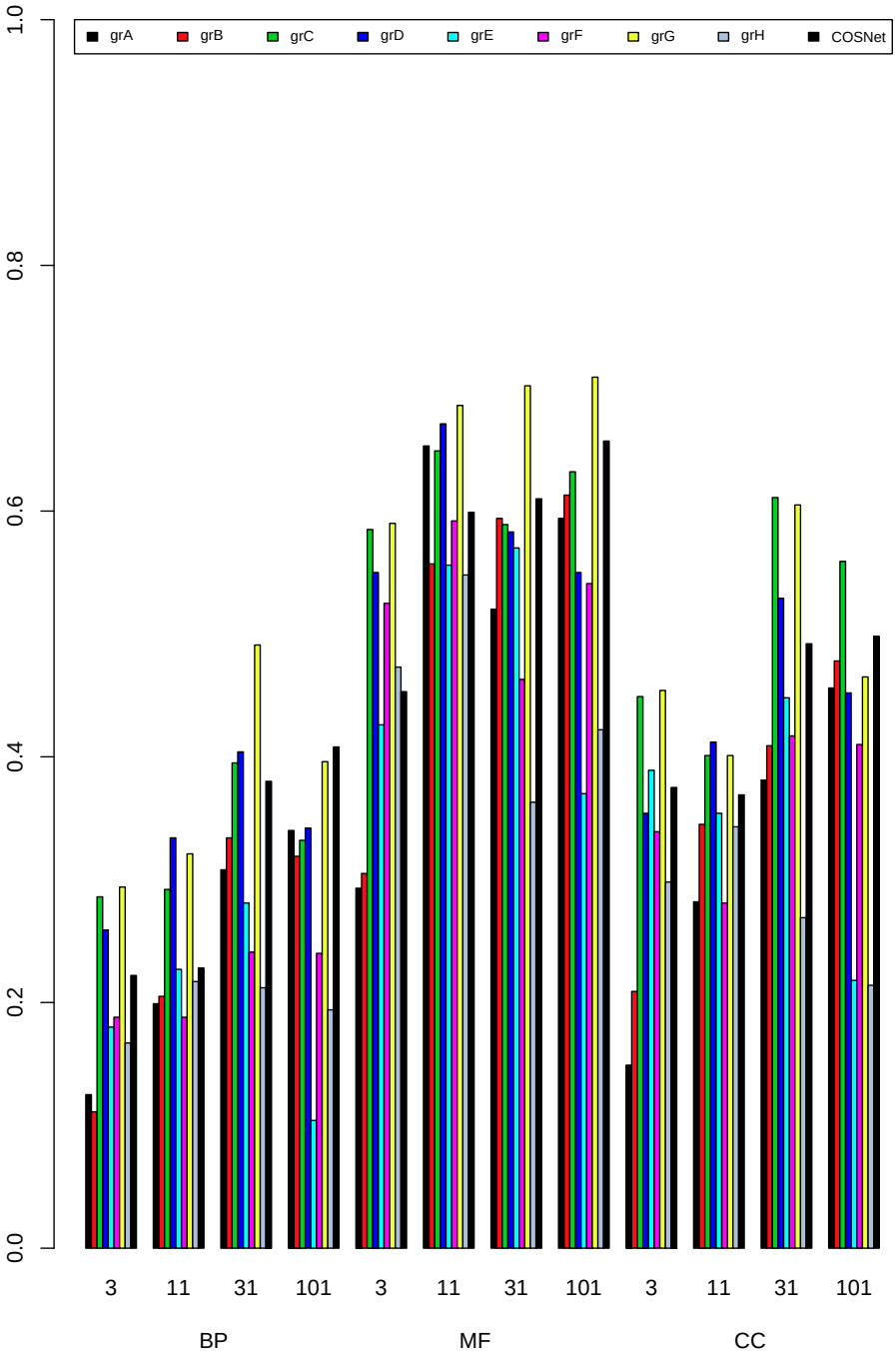


Figure 3.5: Comparison of the MouseFunc I challenge methods and regularized *COSNet* with strategy LSI-WS a) in terms of P20R averaged across all the twelve considered GO categories.

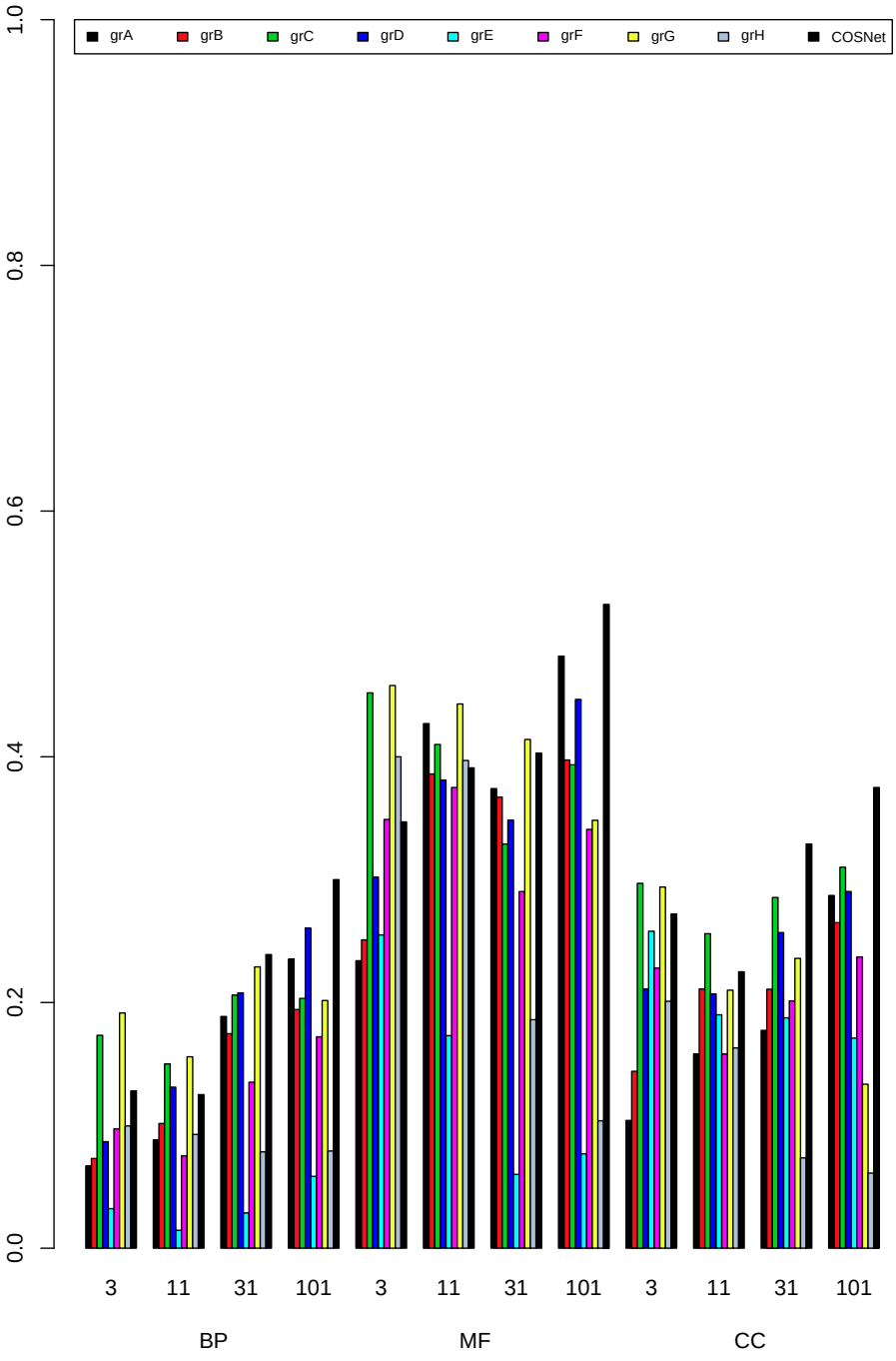


Figure 3.6: Comparison of the MouseFunc I challenge methods and regularized *COSNet* with strategy LSI-WS a) in terms of F-score averaged across all the twelve considered GO categories.

Conclusions

In this thesis we introduced a novel neural algorithm, *COSNet*, based on Hopfield networks for semi-supervised learning in graphs with high imbalanced data. *COSNet* adopts a cost sensitive methodology to manage the unbalance between positive and negative labels, and to preserve and coherently encode the prior knowledge. Moreover, the proposed algorithm has a low computational complexity and nicely scales on large-scale data.

We applied *COSNet* to the genome-wide prediction of gene function in *S.Cerevisiae* model organism, showing a large improvement of the prediction performances w.r.t. the compared state-of-the-art methods.

A second contribution of the thesis is an algorithm (*LSI*) to combine multiple sources of networked data in a "consensus" network, whose edges are the result of a weighted combination of multiple types of data. The algorithm associates each input network with an "measure of informativeness" for each considered biological property and source of data. These measures are then used in the weighted combination of the single input networks. The algorithm is fast and can be applied to the integration of a high number of different data sources.

We validated the proposed method in the integration of multiple sources of biomolecular data to predict the functional classes of genes in the yeast and mouse model organisms at genome-wide level, using the GO and FunCat ontologies. The results, compared with those of the best state-of-the-art methods, show the effectiveness of our approach.

Regarding the possible improvements on the proposed algorithms, we observe that the design of *COSNet* is based on two working hypothesis:

1. The model assumes that the input data are not affected by noise, but in both GO and FunCat ontologies genes may be not annotated for a class c simply due to lack of knowledge about their biological functions.

2. *COSNet* is based on parametrized Hopfield networks in which only

two parameters are fixed and estimated in the learning phase

Indeed, the assumption 1) simplifies the model and more complex techniques are needed to adapt the algorithm to noisy input data, e.g. neural networks with constraints on the activation values of labeled neurons.

With regard to the second working hypothesis, we verified (results not included in the thesis) that Hopfield networks with an excessive high number of parameters are affected by overfitting. An interesting extension would be increasing the number of parameters, e.g. by adopting different activation thresholds for neurons, and finding the optimal number of parameters with model selection techniques.

Moreover, *COSNet* considers as negative examples for a class c all the genes not annotated with c and annotated with any other class in the functional ontology. Clearly, among negative examples, there are some of them more informative than the others, for example because they are closer to c in the structure representing the functional hierarchy. Accordingly, a direct extension of the algorithm could be adopting appropriate strategies in selecting the negative examples for the biological property to be studied.

Finally, we have adopted in our experiments a "flat" approach in predicting gene functions, i.e. we predict each functional class at a time without considering the decisions for the other classes. Due to the true path rule, ontology-wide predictions may produce inconsistencies in parent-child relationships, leading to worse predictive performances. An extension in this sense is either adopting multitask approaches when predicting function of genes or applying techniques to reconcile predictions according to the hierarchy of the functional classes.

Bibliography

[1] Freshman R. D., Adams M. D., White O., Clayton R. A., Jerkiness E. F., Ker lavage A. R., Built C. J., Tomb J. F., Daugherty B. A., Merrick J. M., et al.: Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science* 1995, **269**(5223): 496–512.

[2] Goff A., Barr ell BG., Buss H., Davis RE., Dijon B., Fieldsman H., Gilbert F., Hoarsely JD., Jack C., Johnston M., Louis EH., Mew es HAW., Mariam Y., Philippine P., Tattling H., Oliver SIG.: Life with 6000 genes. *Science* 1996, **274**(546): 563–7.

[3] Lander ES., Linton LEM., Barren B., Niobium C., Ody MC., Baldwin J., Devon K., Dewar K., Doyle M., Fitz Hugh W., Funk R., Gage D., Harris K., Heard A., How land J., Kan L., Legacy J., Levine R., Merwin P., Keenan K., Meld rim J.: Initial sequencing and analysis of the human genome. *Nature* 2001, **409**: 860–921.

[4] Vent er KC., Adams MD., Myers WE., Li PW., Mural RAJ., Sutton G., Smith HO., Randell M., Evans CA., Holt RA., Cocaine JD., Emanates P., Bealle RM., Hus on DH., Wort man JR., Zhang Q., Kodiak CD., Zhang H., Chen L., Skips M.: The sequence of the human genome. *Science* 2001, **291**: 1304–51.

[5] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics* 2000, **25**: 25–29, doi:10.1038/75556. PM ID 10802651

[6] Ruepp A., Zollie A., Mai er D., Albertan K., Hanni J., Muckrakes M., TKO I., Gulden er U., Manhunt G., Menstruate M., and Mew es H.W.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research* 2004, **32**(18):5539–5545.

[7] discarding M., Karo P., Kronecker M., Lee T., and Ouzo unis C. : Prediction of enzyme classification from protein sequence without the use of sequence similarity. In *PRC. of the 5th IS MB* 1997, 92–99.

[8] Obelisk G., Lanckriet G., Grant C., Jordan M. I. and Noble, W.S. : Consistent probabilistic output for protein function prediction. *Genome Biology* 2008, **9**:(S6).

[9] Clare A.: Machine learning and data mining for yeast functional genomics. PhD thesis. University of Wales Abreast. PhD thesis 2003. University of Wales Abreast.

[10] Clare A. and King R. : Predicting gene function in *saccuracies cerevisiae*. *Bioinformatics* 2003, **19**(Supp.2): II42–II49.

[11] Quinlan J. : Induction of decision trees. *Machine Learning* 1986, **1**: 81–106.

[12] Venus C., Strife J., Architect L., Desirous S., and Bloc keel H.: Decision trees for hierarchical multi-label classification. *Machine Learning* 2008, **73**(2): 185–214.

[13] Schlep B. and Smile A. J.: Learning with Kernels. *MIT Press.*, 2002.

[14] CAI C. Z., Han L. Y., Ji Z. L. and Chen Y. Z.: Enzyme family classification by support vector machines. *Proteins* 2004, **55**(1): 66–76.

[15] Dob son P. D. and Doug A. J. Distinguishing enzyme structures from non-enzymes without alignments. *J Mil Biol* 2003, **330**(4):771–783.

[16] Backward K. M., ON C. S., Schnauzer S., Washington S.V.N., Smile A.J. and Rigel H.: Protein Function Prediction via Graph Kernels. *Bioinformatics* 2005, **21 Suppl 1**: i47–56.

[17] Vet J. P.: A tree kernel to analyse phylogenetic profiles. *Bioinformatics* 2002, **18 Suppl 1**: S276–84.

[18] Ben-Hour, A. and Brut lag, D. Remote homology detection: a motif based approach. *Bioinformatics* 2003, **19 Suppl 1**: i26–33.

[19] Rouse J., Saunders C., Seismic S. and Sh awe-Taylor J. : Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research* 2006, **7**: 1601–1626.

[20] Centroids I., Joachim's T., Hoffman T. and Al tun, Y. : Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 2005, **6**: 1453–1484.

[21] Lam pert C. and Black M. : Structured prediction by joint kernel support estimation. *Machine Learning* 2009, **77**: 249–269.

[22] Baker G., Hoffman T., Schoolkid B., Smile A.J., Task B. and Washington S. : *Predicting structured data*. MIT Press 2007, Cambridge, MA.

[23] Skoal A. and Ben-Hour A.: Hierarchical classification of Gene Ontology terms using the Construct method. *Journal of Bioinformatics and Computational Biology* 2010, **8**(2): 357–376.

[24] Astatine K., Helm L., Pitcairn E., Seismic S. and Rouse J.: Towards structured output prediction of enzyme function. *BC Proceedings* 2008, **2 Suppl 4**:S2.

[25] Vazquez A., Flam mini A., Mari tan A., Vespasian A.: Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology* 2003, **21**: 697 – 700.

[26] Marcotte E. M., Peregrine M., Thompson M.J., Ye ates T.O., Iceberg D.: A combined algorithm for genome-wide prediction of protein function. *Nature* 1999, **402**: 83–86.

[27] Oliver S.: Guilt-by-association goes global. *Nature* 2000, **403**(6770):601–3.

[28] Demott J., Bumgarner R., Samudrala R. : Functional annotation from predicted protein interaction networks. *Bioinformatics* 2005, **21**(15): 3217–3226.

[29] Schwikowski B. *et al.* : A Network of protein interaction in yeast. *Nat. Biotechnol.* 2000, **18**: 1257–1261.

[30] Hishigaki H., Nakai K., Ono T., Tanigami A., Takagi T.: Assessment of prediction accuracy of protein function from protein - protein interaction data. *Yeast* 2001, **18**: 523–531.

[31] Karaoz U., Murali T.M., Letovsky S., Yu Zheng, Chumming Ding, Cantor R.C. : Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc Natl Acad Sci USA* 2004, **101**: 2888–2893.

[32] Nabieva, E., Jim, K., Agarwal, A., Chazelle, B., and Singh, M.: Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics* 2005, **21**(S1): 302–310.

[33] Zhu X., Ghahramani Z., & Lafferty J.: Semi-supervised learning with gaussian fields and harmonic functions. *In Proc. of the 20th Int. Conf. on Machine Learning* 2003. Washintgton DC, USA.

[34] Zhou D., Bousquet O., Lal T.N., Weston J. and Schlkopf B.: Learning with Local and Global Consistency. *Advances in Neural Information Processing Systems* 2004, **16**:321–28.

[35] Tsuda K., Shin H.J. and Scholkopf B.: Fast protein classification with multiple networks. *Bioinformatics* 2005, **21(Suppl.2)**:ii59–ii5.

[36] Mostafavi S., Ray D., Farley D.W., Grouios C. and Morris Q.: Genemania: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biol.* 2008, **9(Suppl. 1)**:S4.

[37] Mostafavi S. and Morris Q.: Fast integration of heterogeneous data sources for predicting gene function with limited annotations. *Bioinformatics* 2010, **26**:1759–1765.

[38] Bengio Y., Delalleau O., and Roux N.: Label propagation and quadratic criterion. In *Semi-Supervised Learning* (Chapelle O., Schlkopf B. , and Zien A., eds.), MIT Press 2006, pp. 193–216.

[39] Saad Y. : *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company 1996, Boston, MA.

[40] Cesa-Bianchi N., Gentile C., Vitale F. and Zappella G.: Random spanning trees and the prediction of weighted graphs. In *Proceedings of the 27th International Conference on Machine Learning* 2010, Haifa, Israel.

[41] Sharan R., Ulitsky, I. and Shamir R.: Network-based prediction of protein function. *Molecular Systems Biology* 2007, **3**:3–88.

[42] Bader G.D. and Hogue C.W.: An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics* 2003, **4**:2.

[43] Rives A.W., Galitski T.: Modular organization of cellular networks. *Proc Natl Acad Sci U S A* 2003, **100**:1128–33.

[44] Goldberg D.S. and Roth F.P.: Assessing experimentally derived interactions in a small world. *Proc Natl Acad Sci U S A* 2003, **100**:4372–6.

[45] Spirin V., Mirny L.A.: Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A* 2003, **100**:12123–8.

[46] Przulj N., Wigle D.A., Jurisica I.: Functional topology in a network of protein interactions. *Bioinformatics* 2004, **20**:340–8.

[47] King A.D., Przulj N., Jurisica I.: Protein complex prediction via cost-based clustering. *Bioinformatics* 2004, **20**:3013–20.

[48] Bader J.S.: Greedily building protein networks with confidence. *Bioinformatics* 2003, **19**:1869–74.

[49] Asthana S., King O.D., Gibbons F.D. and Roth F.P.: Predicting protein complex membership using probabilistic network reliability. *Genome Res* 2004, **14**:1170–5.

[50] Wu D.D., Hu X.: An Efficient Approach to Detect a Protein Community from a Seed. *In CIBCB* 2005, 135–141.

[51] Eisner R., Poulin B., Szafron D. and Lu P: Improving protein prediction using the hierarchical structure of the Gene Oontology. In *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology* 2005.

[52] Blockeel H., Schietgat L. and Clare A.: Hierarchical multilabel classification trees for gene function prediction. In J. Rousu, S. Kaski, and E. Ukkonen, editors, *Probabilistic Modeling and Machine Learning in Structural and Systems Biology* 2006, Tuusula, Finland. Helsinki University Printing House.

[53] Shahbaba B. and Neal M.: Gene function classification using Bayesian models with hierarchy-based priors. *BMC Bioinformatics* 2006, **7**(448).

[54] Jiang X., Nariai N., Steffen M., Kasif S. and Kolaczyk E.: Integration of relational and hierarchical network information for protein function prediction. *BMC Bioinformatics* 2008, **9**(350).

[55] Barutcuoglu Z., Schapire R. and Troyanskaya O.: Hierarchical multi-label prediction of gene function. *Bioinformatics* 2006, **22**(7): 830–836.

[56] Guan Y., Myers C., Hess D., Barutcuoglu Z., Caudy A. and Troyanskaya O.: Predicting gene function in a hierarchical context with an ensemble of classifiers. *Genome Biology* 2008, **9**(S2).

[57] Labaj P.P., Leparc G.G., Linggi B.E., Markillie L. M., Wiley H.S. and Kreil D.P.: Characterization and improvement of RNA-Seq precision in quantitative transcript expression profiling. *Bioinformatics* 2011, **13**: i383–i391.

[58] Friedberg, I.: Automated protein function prediction-the genomic challenge. *Brief. Bioinformatics* 2006, **7**: 225–242.

[59] Noble W. and Ben-Hur A. : Integrating information for protein function prediction. In T. Lengauer, editor, *Bioinformatics - From Genomes to Therapies* 2007, **3**:1297–1314. Wiley-VCH.

[60] Lanckriet G.R., Deng M., Cristianini N., Jordan M.I., Noble W.S.: Kernel based data fusionand its application to protein function prediction in yeast. *Pac Symp Biocomput* 2004, 300–311.

[61] Pavlidis P., Weston J., Cai J. and Noble, W.: Learning gene functional classification from multiple data. *J. Comput. Biol.* 2002, **9**: 401–411.

[62] M. Deng, T. Chen, F. Sun.: An integrated probabilistic model for functional prediction of proteins. *J Comput Biol* 2004, **11**(2-3):463–475.

[63] Troyanskaya O. *et al.* : A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *saccharomices cerevisiae*). *Proc. Natl Acad. Sci. USA* 2003, **100**: 8348–8353.

[64] Chua H., Sung W. and Wong L.: An efficient strategy for extensive integration of diverse biological data for protein function prediction. *Bioinformatics* 2007, **23**(24): 3364–3373.

[65] Myers C. and Troyanskaya O.: Context-sensitive data inegration and prediction of biological networks. *Bioinformatics* 2007, **23**: 2322–2330.

[66] Lanckriet G., De Bie T., Cristianini N., Jordan M. and Noble, W.: A statistical framework for genomic data fusion. *Bioinformatics* 2004, **20**: 2626–2635.

[67] Sonnenburg S., Ratsch G., Schafer C. and Scholkopf B.: Large scale multiple kernel learning. *Journal of Machine Learning Research* 2006, **7**: 1531–1565.

[68] Rakotomamonjy A., Bach F., Canu S. and Grandvalet Y.: More efficiency in multiple kernel learning. In *ICML : Proceedings of the*

24th international conference on Machine learning 2007, 775–782. New York, NY, USA. ACM.

[69] Lewis D., Jebara T. and Noble, W.: Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. *Bioinformatics* 2006, **22**(22): 2753–2760.

[70] Japkowicz N. and Stephen S.: The class imbalance problem: A systematic study. *Journal Intelligent Data Analysis* 2002, **6**: i5.

[71] Japkowicz N.: The Class Imbalance Problem: Significance and Strategies. *in Proceedings of the 2000 International Conference on Artificial Intelligence*, 111–117.

[72] Elkan C.: The Foundations of Cost-Sensitive Learning. *in Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* 2001.

[73] Guo X., Yin Y., Dong C, Yang G. and Zhou G.: On the Class Imbalance Problem. *Proceeding ICNC '08 Proceedings of the 2008 Fourth International Conference on Natural Computation*, **4**:192–201.

[74] Ling C.X. and Sheng V.S.: Cost-sensitive Learning and the Class Imbalanced Problem. *In Encyclopedia of Machine Learning. C. Sammut (Ed.). Springer* 2007.

[75] Cesa-Bianchi N. and Valentini G.:. Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. *Journal of Machine Learning Research, W&C Proceedings, Machine Learning in Systems Biology* 2010, **8**: 14–29.

[76] Pena-Castillo L. et al.: A critical assessment of Mus musculus gene function prediction using integrated genomic evidence. *Genome Biology* 2008, **9**:S1.

[77] Agresti A., Coull B.A.: Approximate is better than exact for interval estimation of binomial proportions. *Statistical Science* 1998, **52**(2): 119–126.

[78] Brown L.D., Cai T.T., Dasgupta A.: Interval estimation for a binomial proportion. *Statistical Science* 2001 **16**: 101–133.

[79] Eddy S.R.: Profile hidden Markov models. *Bioinformatics* 1998, **14**(9): 755–763.

[80] Spellman P.T. et al.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell* 1998, **9**(12): 3273–3297.

[81] Gasch P. et al.: Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell* 2000, **11**(12): 4241–4257.

[82] Stark C., Breitkreutz B.J., Reguly T., Boucher L., Breitkreutz A., Tyers M.: Biogrid: a general repository for interaction datasets. *Nucleic acids research* 2006, **34**: D535–D539.

[83] Wilcoxon F.: Individual comparisons by ranking methods. *Journal of Computational Biology* 1945, **1**(6): 80–83.

[84] Lin H.T., Lin C.J., Weng R.: A note on platt's probabilistic outputs for support vector machines. *Machine Learning* 2007, **68**(3): 267–276.

[85] Brown M.P.S. et al.: Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences of the United States of America* 2000, **97**(1): 267–276.

[86] von Mering C., Krause R., Snel B., Cornell M., Oliver S., Fields S. and Bork P.: Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* 2002, **417**: 399–403.

[87] Hopfield J. J.: Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA* 1982, **79**(8): 2554.-2558.

[88] Hertz J., Krogh A. and Palmer R.G.: Introduction to the Theory of Neural Computation. *Addison Wesley* 1991.

[89] Re M. and Valentini G.: Simple ensemble methods are competitive with state-of-the-art data integration methods for gene function prediction. *Journal of Machine Learning Research, W&C Proceedings, Machine Learning in Systems Biology* 2010, **8**: 98–111.

[90] Su A.I., Wiltshire T., Batalov S., Lapp H., Ching K.A., Block D., Zhang J., Soden R., Hayakawa M., Kreiman G., Cooke M.P., Walker J.R., Hogenesch J.B.: A gene atlas of the mouse and human protein-encoding transcriptomes *Proc Natl Acad Sci USA* 2004, **101**:6062–6067.

[91] Zhang W., Morris Q.D., Chang R., Shai O., Bakowski M.A., Mitsakakis N., Mohammad N., Robinson M.D., Zirngibl R., Somogyi E., Laurin N., Eftekharpour E., Sat E., Grigull J., Pan Q., Peng W.T., Krogan N., Greenblatt J., Fehlings M., Kooy D., Aubin J., Bruneau B.G., Rossant J., Blencowe B.J., Frey B.J., Hughes T.R.: The functional landscape of mouse gene expression *J Biol* 2004, **9**:3–21.

[92] Siddiqui A.S., Khattra J., Delaney A.D., Zhao Y., Astell C., Asano J., Babakaiff R., Barber S., Beland J., Bohacec S., Brown-John M., Chand S., Charest D., Charters A.M., Cullum R., Dhalla N., Featherstone R., Gerhard D.S., Hoffman B., Holt R.A., Hou J., Kuo B.Y., Lee L.L., Lee S., Leung D., Ma K., Matsuo C., Mayo M., McDonald H., Prabhu A.L. et al.: A mouse atlas of gene expression: large-scale digital gene-expression profiles from precisely defined developing C57BL/6J mouse tissues and cells. *Proc Natl Acad Sci USA* 2005, **102**:18485–18490.

[93] Finn R.D., Mistry J., Schuster-Bockler B., Griffiths-Jones S., Hollich V., Lassmann T., Moxon S., Marshall M., Khanna A., Durbin R., Eddy S.R., Sonnhammer E.L., Bateman A.: Pfam: clans, web tools and services. *Nucleic Acids Res* 2006, **34**:D247–251.

[94] Mulder N.J., Apweiler R., Attwood T.K., Bairoch A., Bateman A., Binns D., Bradley P., Bork P., Bucher P., Cerutti L., Copley R., Courcelle E., Das U., Durbin R., Fleischmann W., Gough J., Haft D., Harte N., Hulo N., Kahn D., Kanapin A., Krestyaninova M., Lonsdale D., Lopez R., Letunic I., Madera M., Maslen J., McDowall J., Mitchell A., Nikolskaya A.N. et al.: InterPro. *Nucleic Acids Res* 2005, **33**:D201–205.

[95] Brown K.R., Jurisica I.: Online Predicted Human Interaction Database. *Bioinformatics* 2005, **21**:2076–2082.

[96] Eppig J.T., Blake J.A., Bult C.J., Kadin J.A., Richardson J.E.: The mouse genome database (MGD): new features facilitating a model system. *Nucleic Acids Res* 2007, **35**:D630–637.

[97] Phenotype Annotations from MGI [ftp.informatics.jax.org/pub/reports].

[98] Kasprzyk A., Keefe D., Smedley D., London D., Spooner W., Melsopp C., Hammond M., Rocca-Serra P., Cox T., Birney E.: EnsMart:a generic system for fast and flexible access to biological data. *Genome Res* 2004, **14**:160–169 .

[99] O'Brien K.P., Remm M., Sonnhammer E.L.: Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Res* 2005, **33**:D476–D480.

[100] Wheeler D.L., Barrett T., Benson D.A., Bryant S.H., Canese K., Chetvernin V., Church D.M., DiCuccio M., Edgar R., Federhen S., Geer L.Y., Kapustin Y., Khovayko O., Landsman D., Lipman D.J., Madden T.L., Maglott D.R., Ostell J., Miller V., Pruitt K.D., Schuler G.D., Sequeira E., Sherry S.T., Sirotkin K., Souvorov A., Starchenko G., Tatusov R.L., Tatusova T.A., Wagner L., Yaschenko E.: Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* 2007, **35**:5–12.

[101] Hamosh A., Scott A.F., Amberger J.S., Bocchini C.A., McKusick V.A.: Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Res* 2005, **33**:514–517.

[102] Disease Associations from OMIM [ftp.ncbi.nih.gov/repository/OMIM/] .

[103] Valentini G.: True Path Rule hierarchical ensembles for genome-wide gene function prediction. *IEEE ACM Transactions on Computational Biology and Bioinformatics* 2011, **8**(3):832–847.

[104] Cesa-Bianchi N., Re M., Valentini G.: Functional Inference in FunCat through the Combination of Hierarchical Ensembles with Data Fusion Methods. *ICML Workshop on learning from Multi-Label Data MLD'10* 2010, 13–20.

[105] Lee H., Tu Z., Deng M., Sun F. and Chen, T.: Diffusion Kernel-Based Logistic Regression Models for Protein Function Prediction. *OMICS: A Journal of Integrative Biology* 2006, **10**(1): 40–55.

[106] Guan Y., Myers C.L., Lu R., Lemischka I.R., Bult C.J., Troyanskaya O.G.: A genomewide functional network for the laboratory mouse. *PLoS Comput Biol* 2008, **4**(9):e1000165.

[107] Kim W.K., Krumpelman C., Marcotte E.M.: Inferring mouse gene functions from genomic-scale data using a combined functional network/classification strategy. *Genome Biol.* 2008, **9 (Suppl 1)**:S5.

[108] Chen Y., Xu D.: Global protein function annotation through mining genome-scale data in yeast Saccharomyces cerevisiae. *Nucleic Acids Res* 2004, **32**:6414–6424.

[109] Joshi T., Chen Y., Becker J.M., Alexandrov N., Xu D.: Genome-scale gene function prediction using multiple sources of high-throughput data in yeast Saccharomyces cerevisiae. *OMICS* 2004, **8**:322–333.

[110] Tian W., Zhang L. V., Taan M., Gibbons F. D., King O. D., Park J., Wunderlich Z., Cherry J. M. and Roth F. P.: Combining guilt-by-association and guilt-by-profiling to predict Saccharomyces cerevisiae gene function. *Genome Biology* 2008, **9(Suppl 1)**:S7.

[111] Qi Y., Klein-Seetharaman J., Bar-Joseph Z.: A mixture of feature experts approach for protein-protein interaction prediction. *BMC Bioinformatics* 2007, **8**(S10): S6.

[112] Leone M., Pagnani A.: Predicting protein functions with message passing algorithms. *Bioinformatics* 2005, **21**:239–247.