A neural network algorithm for semi-supervised node label learning from unbalanced data

Marco Frasca[a], Alberto Bertoni[a], Matteo Re[a], Giorgio Valentini[a,*]

[a]*Dipartimento di Informatica, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy.*

Abstract

Given a weighted graph and a partial node labeling, the graph classification problem consists in predicting the labels of all the nodes. In several application domains, from gene to social network analysis, the labeling is unbalanced: for instance positive labels may be much less than negatives. In this paper we present *COSNet* (COst Sensitive neural Network), a neural algorithm for predicting node labels in graphs with unbalanced labels. *COSNet* is based on a 2-parameters family of Hopfield networks, and consists of two main steps: 1) the network parameters are learned through a cost-sensitive optimization procedure; 2) a suitable Hopfield network restricted to the unlabeled nodes is considered and simulated. The reached equilibrium point induces the classification of the unlabeled nodes. The restriction of the dynamics leads to a significant reduction in time complexity and allows the algorithm to nicely scale with large networks. An experimental analysis on real-world unbalanced data, in the context of the genome-wide prediction of gene functions, shows the effectiveness of the proposed approach.

Keywords: Hopfield neural networks, Semi-supervised learning in graphs, Learning from unbalanced data, Node label prediction

*Corresponding author. DI - Dipartimento di Informatica Università degli Studi di Milano, Via Comelico 39, 20135 Milano, Italy, Tel +39 (02) 503.16225, Fax +39 (02) 503.16373

Email addresses: frasca@di.unimi.it (Marco Frasca), bertoni@di.unimi.it (Alberto Bertoni), re@di.unimi.it (Matteo Re), valentini@di.unimi.it (Giorgio Valentini)

1. Introduction

The increasing interest in application domains where data are naturally represented as connected nodes in a graph, i.e. biological networks (Wuchty et al., 2003; Pena-Castillo et al., 2008), social networks (Borgatti et al., 2009) and the World-Wide-Web (Dorogovtsev & Mendes, 2003), motivated the design and analysis of novel network-based learning algorithms. Indeed several problems can be modeled as supervised or semi-supervised machine learning problems in undirected graphs. For instance, in label prediction problems nodes represent partially labeled instances, the edges pairwise similarities among nodes and the aim is to label the unlabeled part of the graph by exploiting the topology of the network and the a priori knowledge coded in the labeled nodes (Bengio et al., 2006).

Several methods have been proposed for node classification in networked data. Algorithms based on the *guilt-by-association* principle set unlabeled nodes according to the majority of the labels in their direct neighborhoods (Marcotte et al., 1999; Oliver, 2000). By extending this approach, indirect neighbours, that account for pairs of nodes connected through intermediate ones, have been used to extend the notion of pairwise-similarities among nodes (Chua et al., 2006; Li et al., 2010; Bogdanov & Singh, 2010). Other methods focused on clustering nodes into functional modules based on the graph topology, and assigning to unlabeled nodes the most common labels in a given module (Sharan et al., 2007; Zhu et al., 2010). Furthermore, nodes can propagate labels to their neighbors with an iterative process until convergence (Zhu et al., 2003; Zhou et al., 2004). Markov Random Walks have been applied to tune the amount of propagation we allow in the graph, by setting the length of the walk across the graph (Szummer & Jaakkola, 2001; Azran, 2007), and methods based on both local and global learning strategies, exploiting both local guilt-by-association rules and the overall global topology of the graph have been recently proposed (Re & Valentini, 2012).

Other approaches are based on graph regularization (Belkin et al., 2004; Delalleau et al., 2005), and on the exploitation of the properties of the graph Laplacian associated to the weight matrix of the graph (Belkin & Niyogi, 2003). Methods based on the amount of functional flow through the nodes (Nabieva et al., 2005), on global graph consistency (Vazquez et al., 2003; Karaoz et al., 2004), on Markov (Deng et al., 2004) and Gaussian Random Fields (Tsuda et al., 2005; Mostafavi et al., 2008), and recently on kernelized score functions (Re et al., 2012) have been applied to the predic-

tion of gene functions.

For their common characteristics, many of the described approaches can be cast into a common framework where a quadratic cost objective function is minimized (Bengio et al., 2006). In this context, global graph optimization techniques usually require time/space intensive procedures, whereas it is not ensured a relevant improvement of the predictive capabilities. For instance, in the gene function prediction problem (GFP), Murali et al. (2006) showed that a minimum cut algorithm, that achieves a global optimum of the objective function, does not provide a substantial advantage over simple local consistency procedures, while substantially increases the computational time (its complexity is cubic with respect to the number of nodes in dense graphs). From this point of view, it seems natural a neural approach based on Hopfield networks, that can reach a local minimum of the quadratic objective function, but with a relevant gain in computational time complexity (Hopfield, 1982).

Unfortunately, both local and global methods tend to suffer a decay in the quality of solutions when input data are highly unbalanced, that is when positive examples (resp. negative) are significantly less than negatives (resp. positives). This issue is particularly relevant in problems like GFP, where the imbalance in data requires the adoption of cost-sensitive strategies (Mostafavi et al., 2008; Cesa-Bianchi & Valentini, 2010; Valentini, 2011). Moreover, many of the described approaches may not preserve the prior knowledge coded in the initial labeling and in the pairwise similarities, and this is a relevant issue when we assume that the prior knowledge is not affected by noise. Finally, many approaches based on neural networks do not distinguish between the node labels and the values of the neuron states (Karaoz et al., 2004), thus resulting in a lower predictive capability of the network.

In order to address these issues, we propose *COSNet* (*COst Sensitive neural Network*)[1], a novel neural algorithm based on Hopfield 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

[1] A preliminary version of *COSNet* has been proposed in Bertoni et al. (2011)

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.

In order to motivate our approach, in Sect. 2 the GFP problem is described as a semi-supervised classification problem characterized by very unbalanced classes. Hopfield networks and the main issues related to this type of recurrent neural network are discussed in Sect. 3, and the "sub-network property" by which we can safely restrict the dynamics of the Hopfield network to its unlabeled part is introduced in Sect 4. *COSNet* and its regularized version are described and discussed in Sect. 5. Finally, in Sect. 6 we apply *COSNet* to the genome-wide prediction of gene functions in a model organism, including 232 functional classes of the FunCat taxonomy (Ruepp et al., 2004), and using six different types of biomolecular data. The paper ends with the conclusions.

2. An unbalanced semi-supervised learning problem in graphs: gene function prediction

To motivate our approach, we introduce a relevant unbalanced learning problem in computational biology: the *gene function prediction problem* (GFP). GFP is a multi-class, multi-label classification problem that can be modeled as a set of two-class classification problems, since each gene may belong (positive example) or not (negative example) to a given functional class. Functional classes are usually very unbalanced, with negative examples that largely outnumber positives.

In our setting GFP is formalized as a semi-supervised problem of label learning in graphs (Bengio et al., 2006). Genes are represented by a set of nodes $V = \{1, 2, \ldots, n\}$, and relationships between genes are encoded through a symmetric $n \times n$ real weight matrix $\boldsymbol{W}$, whose elements w_{ij} represent functional similarities between pairs (i, j) of genes.

For a given functional class c, the nodes V are labeled with $\{+, -\}$, leading to the subsets P and N of positive and negative vertices for class c.

For most model organisms usually the functional labeling is known only for a subset $S \subset V$, while is unknown for $U = V \setminus S$. Let be $S^+ = S \cap P$ and $S^- = S \cap N$: we can refer to S^+, S^- and $\boldsymbol{W}$ as the "prior information".

The *gene function prediction problem* consists in finding a bipartition (U^+, U^-) of genes in U on the basis of the prior information. Genes in U^+ are then considered candidates for the class $P \cap U$. From this standpoint, GFP is set as a semi-supervised learning problem on graphs, since gene functions can be predicted by exploiting both labeled and unlabeled nodes/genes and the weighted connections between them.

Finally, to simplify the problem, we assume that the prior knowledge is not affected by noise. Nevertheless, it is worth noting that in the functional taxonomies negative examples for a class c in general are simply genes that are not classified for c, and may correspond to false negatives due to lack of knowledge about their biological function.

3. Hopfield Networks for gene function prediction

A Hopfield network is a recurrent neural network whose dynamics admits a Lyapunov function (Hopfield, 1982). This model has been used in many different areas, including content-addressable memory (Wang, 2003; Liu & Hu, 2009; Zhang & Zhang, 2005), discrete nonlinear optimization (Tsirukis et al., 1989), binary classification for GFP (Karaoz et al., 2004).

Here we introduce Hopfield networks with binary neurons as binary classifier for the GFP problem. The classical Hopfield network has neurons with activation values -1 and 1; in this paper we consider as activation values $\sin\alpha$ and $-\cos\alpha$, where α is a real number ($0 \leq \alpha \leq \frac{\pi}{2}$) enclosed in the definition of the network. The main motivation of this parametric setting of activation values consists in extending the classical Hopfield network model in order to deal with unbalanced classification problems, where one of the classes is heavily under-represented in comparison to the other class. In particular, in the GFP problem the negative class samples largely outnumbers the positive ones, and our hypothesis is that, by putting more "strength" on positive neurons, we may obtain network dynamics characterized by a larger influence of positive neurons, thus "counterbalancing" the bias in favour of the negative class. Analogously, we can handle the case in which positive instances outnumber negatives by putting more "strength" on negative neurons. This can be achieved by conceptually separating labels and neuron states: neuron states are set to $\sin\alpha$ for "positive" neurons (i.e. those labeled with $+1$)

and to $-\cos\alpha$ for -1 labeled "negative" neurons. By automatically learning the α parameter from the data (Section 5), we obtain different absolute activation values for positive and negative labeled neurons: for instance, when $\alpha > \frac{\pi}{4}$, the activation value $\sin\alpha$ of positive neurons is larger than the absolute value of the activation $-\cos\alpha$ of negative neurons, and the opposite is true when $\alpha < \frac{\pi}{4}$. In this way, by decoupling labels from activation values, labels belonging to the under-represented class may still propagate across the networks, even if the large majority of neurons are labeled with the opposite class, thus avoiding convergence to trivial cases (e.g. "all negative" or "all positive" labelings) or more in general improving the sensitivity of the classification.

Formally, a *parametric Hopfield network* H with neurons $V = \{1, 2, \ldots, n\}$ is a triple $H = < \boldsymbol{W}, \boldsymbol{\gamma}, \alpha >$, where:

$\boldsymbol{W} = (w_{ij})$ is a $n \times n$ symmetric matrix whose elements $w_{ij} \in [0, 1]$ represent the connection strength between neurons i and j, with $w_{ij} = w_{ji}$ for each pair $(i,\, j)$ and $w_{ii} = 0$

$\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n)$ is a real vector of activation thresholds

α is a real variable in $[0, \frac{\pi}{2}]$ that determines the two different neuron activation values $\{\sin\alpha,\, -\cos\alpha\}$

The dynamics of the network is described as follows:

1. At time 0 an initial value $x_i(0) \in \{\sin\alpha,\, -\cos\alpha, 0\}$ 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

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

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

the dynamics. In particular, consider the following quadratic state function (*energy function*):

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

This is a non increasing function which guarantees that every dynamics of the network converges to an equilibrium state $\hat{\boldsymbol{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$, which corresponds to a local minimum of the energy function. We extended the original convergence proof to the more general case in which the neuron activations values may assume values $\{-\cos\alpha, \sin\alpha\}$, for $0 \leq \alpha \leq \frac{\pi}{2}$ (see Appendix A for the proof).

In Karaoz et al. (2004) Hopfield networks have been applied to GFP; the corresponding algorithm for binary classification is named *GAIN* (Gene Annotation using Integrated Networks). A brief outline of *GAIN* is given in the following.

Given the set of genes V and their similarity matrix, we consider the Hopfield network $H = < \boldsymbol{W}, 0, \frac{\pi}{4} >$; the activation thresholds are 0 and the activation values are $\{\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\}$ (that is $\{1, -1\}$ up to a constant term).

Fixed a functional class c, let S^+ be the set of vertices already classified as positive, S^- the set of vertices classified as negative, U the set of vertices whose classification is unknown. In order to classify vertices in U, the following initial state of H is considered:

$$x_i(0) = \begin{cases} 0 & \text{if } \; i \in U \\ \frac{\sqrt{2}}{2} & \text{if } \; i \in S^+ \\ -\frac{\sqrt{2}}{2} & \text{if } \; i \in S^- \end{cases} \tag{3}$$

The dynamics (1) of H is applied to this state until the equilibrium point $\hat{\boldsymbol{x}}$ is reached; a gene $k \in U$ is classified as "positive" iff $\hat{x}_k = \frac{\sqrt{2}}{2}$.

From a biological standpoint, this approach is motivated by the fact that minimizing the overall energy (2) means maximizing the weighted sum of edges connecting neurons with the same activation value. In other words, genes "strongly" connected with other genes having a given function tend to be labeled with that same function, while the opposite is true when "strong" connections with negative neighbours prevail.

Nevertheless, this approach is characterized by a main drawback. By assigning the same absolute activation value $\frac{\sqrt{2}}{2}$ to both positive and negative neurons, and by setting to 0 the threshold of each neuron, when $|S^+| \ll |S^-|$ or $|S^+| \gg |S^-|$, the network is likely to converge to a trivial state made up

by all negative or all positive neurons. This is exactly the situation registered in the taxonomies for gene functions, since only a small number of positive examples is available for most of the functional categories. To address this problem, we first exploit a simple property which holds for sub-networks of a Hopfield network (Section 4), and then we show that, by learning the parameters of the Hopfield network, our proposed algorithm is able to solve unbalanced classification problems (Section 5).

4. Sub-network Property

According to the semi-supervised setting of Section 2, let be $H = < \boldsymbol{W}, \boldsymbol{\gamma}, \alpha >$ a network with neurons $V = U \cup S = \{1, 2, \ldots, n\}$, 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:

$$\begin{aligned} E(\boldsymbol{u}, \boldsymbol{s}) = & -\frac{1}{2}\left(\boldsymbol{u}^T \boldsymbol{W}_{uu} \boldsymbol{u} + \boldsymbol{s}^T \boldsymbol{W}_{ss} \boldsymbol{s} + \boldsymbol{u}^T \boldsymbol{W}_{us} \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{W}_{us}^T \boldsymbol{u}\right) + \boldsymbol{u}^T \boldsymbol{\gamma}^u + \boldsymbol{s}^T \boldsymbol{\gamma}^s \\ = & -\frac{1}{2} \boldsymbol{u}^T \boldsymbol{W}_{uu} \boldsymbol{u} + \boldsymbol{u}^T (\boldsymbol{\gamma}^u - \boldsymbol{W}_{us} \boldsymbol{s}) + C \end{aligned} \tag{4}$$

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. $\boldsymbol{\gamma} = (\boldsymbol{\gamma}^u, \boldsymbol{\gamma}^s)$ represents the vector of the thresholds for respectively unlabeled ($\boldsymbol{\gamma}^u$) and labeled ($\boldsymbol{\gamma}^s$) nodes and $C = -\frac{1}{2} \boldsymbol{s}^T \boldsymbol{W}_{ss} \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{\gamma}^s$ is a term constant w.r.t. $\boldsymbol{u}$.

Suppose now that a state $\tilde{\boldsymbol{s}}$ of neurons in S is given. To preserve the "a priori" knowledge, 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 Hopfield network with neurons U which realizes this dynamics; from equation (4) it holds:

Fact 1. $H_{U|\tilde{\boldsymbol{s}}} = < \boldsymbol{W}_{uu}, \boldsymbol{\gamma}^u - \boldsymbol{W}_{us} \tilde{\boldsymbol{s}}, \alpha >$.

Given a state $\tilde{\boldsymbol{s}}$ of neurons in S, we say that $\tilde{\boldsymbol{s}}$ is part of global minimum of the energy E of H if there is a state $\boldsymbol{u}$ of neurons in U s.t. $(\boldsymbol{u}, \tilde{\boldsymbol{s}})$ is a

global minimum of E. The introduction of the network $H_{U|\tilde{\boldsymbol{s}}}$ is motivated by the following property:

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

Proof. From (4) it follows that

$$E(\boldsymbol{u}, \boldsymbol{s}) = -\frac{1}{2} s^T \boldsymbol{W}_{ss} \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{\gamma}^s + E_{|\boldsymbol{s}}(\boldsymbol{u}) \tag{5}$$

where $E_{|\boldsymbol{s}}(\boldsymbol{u}) = -\frac{1}{2}\boldsymbol{u}^T \boldsymbol{W}_{uu} \boldsymbol{u} + \boldsymbol{u}^T(\boldsymbol{\gamma}^{\boldsymbol{u}} - \boldsymbol{W}_{us}\boldsymbol{s})$. By hypothesis $\tilde{\boldsymbol{u}}$ is a global minimum of $E_{|\tilde{\boldsymbol{s}}}(\boldsymbol{u})$. By assuming that $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}})$ is not a global minimum of E, there exists another state $\hat{\boldsymbol{u}}$ for neurons in U such that $E(\hat{\boldsymbol{u}}, \tilde{\boldsymbol{s}}) < E(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{s}})$. By (5) this implies that $E_{|\tilde{\boldsymbol{s}}}(\hat{\boldsymbol{u}}) < E_{|\tilde{\boldsymbol{s}}}(\tilde{\boldsymbol{u}})$, which contradicts the hypothesis that $\tilde{\boldsymbol{u}}$ is a global minimum of $E_{|\tilde{\boldsymbol{s}}}(\boldsymbol{u})$. □

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

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

for each $i \in S$. Suppose, for a suitable α, that $\tilde{\boldsymbol{s}} = \boldsymbol{x}(S^+, S^-)$ is part of a energy global minimum of $H =< \boldsymbol{W}, \boldsymbol{\gamma}, \alpha >$. Fact 2 assures that $(\boldsymbol{u}, \tilde{\boldsymbol{s}})$ is a global minimum when $\tilde{\boldsymbol{u}}$ is a global minimum of the energy of $H_{U|\tilde{\boldsymbol{s}}}$. In other words the subnetwork property suggests that the optimization problem can be factorized: we can minimize the energy of the subnetwork $H_{U|\tilde{\boldsymbol{s}}}$ instead of the overall network H to predict the hidden part U of neurons.

5. The *COSNet* Algorithm

In this Section we introduce the algorithm *COSNet* (COst-Sensitive neural Network) for dealing with the learning issues presented in the previous sections. We consider Hopfield networks $H =< \boldsymbol{W}, \gamma \cdot \mathbf{e}, \alpha >$, where $\mathbf{e} = (1, 1, \ldots, 1)$, i.e. networks having the same real threshold γ for all the neurons.

For a given $\boldsymbol{W}$, the class of networks $H =< \boldsymbol{W}, \gamma \cdot \mathbf{e}, \alpha >$ is a family with two real parameters γ, α ($0 \leq \alpha \leq \frac{\pi}{2}$). Given the"prior information" $\boldsymbol{W}$, S^+, S^-, suppose that there is a couple $(\hat{\alpha}, \hat{\gamma})$ such that:

1. The solution of the problem corresponds to an energy global minimum of $H =< \boldsymbol{W}, \hat{\gamma} \cdot \mathbf{e}, \hat{\alpha} >$

2. $\boldsymbol{x}(S^+, S^-)$ is part of an energy global minimum of H

Then, by Fact 2, we can discover the hidden states $\hat{\boldsymbol{u}}$ of the neurons by minimizing the energy of the network $H_{U|\boldsymbol{x}(S^+,S^-)}$. Unfortunately, Fact 2 holds only for global minima, and finding the global minimum of the energy requires time/memory intensive procedures. Accordingly, we run Hopfield networks on the subnetwork $H_{U|\boldsymbol{x}(S^+,S^-)}$: our aim is to move toward the factorization of the optimization problem suggested by the subnetwork property by finding a minimum of the energy that, in the general case, is a local minimum. This approach is also motivated by two main reasons: first, we experimentally observed (Section 5.7) that in several cases with high probability the local minima achieved by the Hopfield network are very close to energy global minima; second, in the context of GFP it has been shown that in most cases global minima solutions do not significantly improves performances w.r.t local minima solutions (Murali et al., 2006).

Accordingly, the procedure for solving the GFP problem can be factorized into two main steps:

Step 1. Determine the parameters $(\hat{\alpha}, \hat{\gamma})$ such that the state $\boldsymbol{x}(S^+, S^-)$ is part of a global minimum (or at least of a point "near" to a global minimum) by finding the parameters (α, γ) for which the state $\boldsymbol{x}(S^+, S^-)$ is "as close as possible" to a part of an equilibrium state of H.

Step 2. Minimize the energy function of the network $H_{U|\boldsymbol{x}(S^+,S^-)}$ with the estimated parameters $(\hat{\alpha}, \hat{\gamma})$ by reaching an equilibrium state $\hat{\boldsymbol{u}}$ in a dynamics generated by a suitable initial state.

Finally, the solution (U^+, U^-) of GFP is:

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

A possible realization of step 1 consists firstly in extending the state $\boldsymbol{x}(S^+, S^-)$ to neurons in U by generating a random bipartition (U^P, U^N) of U, and then in optimizing the parameters (α, γ). Below, we discuss in detail each step of *COSNet*. More precisely, in Section 5.1 we describe the procedure adopted

for generating a temporary bipartition of U, then the parameter optimization step is discussed in Section 5.2. The step 2 of *COSNet* is explained in Section 5.3, followed by the analysis of the time complexity of the overall algorithm (Section 5.4), and by the discussion of the strengths and limitations of the supervised two-steps procedure to learn the parameters of the network (Section 5.5). In Section 5.6 we introduce a regularized version of *COSNet*, useful when we need to deal with extremely unbalanced classification tasks. Finally in Section 5.7 we apply a statistical test to show that *COSNet* leads to significantly lower values of the global network energy (2) w.r.t. the "vanilla" version of the Hopfield network.

5.1. Generating a Temporary Solution

For generating the temporary bipartition of U, we adopt a procedure that maintains in U about 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^P m elements uniformly chosen in U

- assign to U^N the set $U \setminus U^P$.

This criterion is justified by the probabilistic model described below.

Suppose that V is divided 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:

$$\frac{|S^+|}{|S|} \cdot h = \underset{z}{\text{argmax}}\ P(z). \tag{6}$$

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 $z = p_s \cdot h$.

Since the parameter learning step (Section 5.2) depends on the temporary assignment of the labels, we studied whether our proposed initial random assignment may introduce noise or may degrade the prediction performance of the neural network system. To this end we experimented two other procedures to initialize the unlabeled nodes. The first one simply avoids the generation of the temporary bipartition of U and projects the nodes into the plane only by considering nodes in S: in this way we achieved slightly worse results (data not shown). The second version consists in iterating the algorithm twice: generating a temporary bipartition of U according to the probabilistic method described above, predicting U by running the Hopfield network on the U subgraph, and then reusing the predicted labels of U as a novel temporary bipartition for another round of the algorithm. This version of the algorithm is computationally intensive, whereas the performance are just slightly better (but usually the difference is not statistically significant – data not shown). These experiments show that it is reasonable to adopt the probabilistic model proposed in this section to initialize the labeling of U when no a priori information about nodes in U is available. It is worth noting that the initial labeling of U contributes only to the computation of the point coordinates (7), while only the known labels associated to the nodes in S are used in the supervised process to learn the parameters α and $\boldsymbol{\gamma}$ of the network, as explained in the next section.

5.2. Finding the Optimal Parameters

The main goal of this step is to find the values of the parameters α and γ such that the state $\boldsymbol{x}(S^+, S^-)$ is "as close as possible" to an equilibrium state. To this end we project the nodes of the network into points of a plane by translating their neighborhood connections to positive and negative nodes into respectively the abscissa and the ordinate of the "projected" points (Figure 1 (a)). Then we learn from the data a line able to separate positive points

(those corresponding to S^+) from the negative ones (those corresponding to S^-). More precisely, we embed the Hopfield network parameters into the parametric representation of the line: α corresponds to the slope and $-\frac{\gamma}{\cos\alpha}$ to the intercept of the parametric line. In particular, we choose the line that minimizes the $F_{score}(\alpha, \gamma)$ to separate positive from negative labeled projected points (Figure 1 (b)). Then we show that finding the paramet-

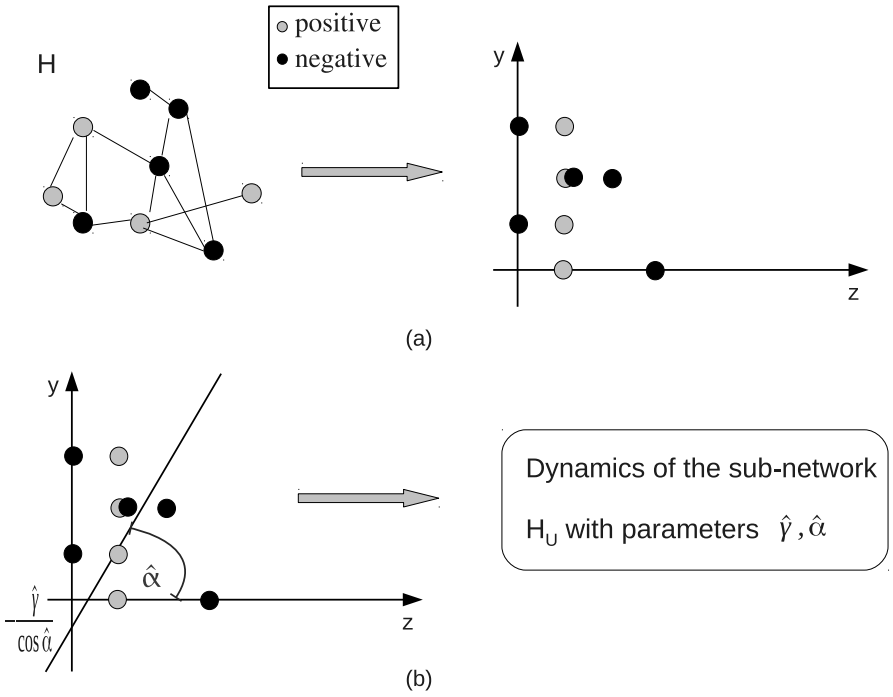


Figure 1: Graphical representation of the COSNet main steps. (a) Network to plane projections: (b) Learning of optimal network parameters and their usage for the subnetwork dynamics

ric line $(\hat{\alpha}, \hat{\gamma})$, which corresponds to the maximum F_{score} ($F_{score} = 1$), leads to an equilibrium state (minimum energy) of the subnetwork H_S, when its activation states are set to $\sin\hat{\alpha}$ and $-\cos\hat{\alpha}$.

5.2.1. Network to Plane Projections

Let consider the sub-network $H_{S|\boldsymbol{x}(U^P,U^N)} = < \boldsymbol{W}_{ss}, \boldsymbol{\gamma}^s - \boldsymbol{W}_{us}^T \boldsymbol{x}(U^P, U^N), \alpha >$, where $\gamma_i^s = \gamma \in \mathbb{R}$ for each $i \in S$, and (U^P, U^N) is the temporary biparti-

tion found in the previous step. We associate with each node $k \in S$ a point $\Delta(k) \equiv (\Delta^+(k), \Delta^-(k))$ in the plane, where

$$\Delta^+(k) = \sum_{j \in S^+ \cup U^P} w_{kj}, \qquad \Delta^-(k) = \sum_{j \in S^- \cup U^N} w_{kj}. \tag{7}$$

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

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

5.2.2. Learning a Line to Separate Positive from Negative Examples

Consider now an arbitrary straight line in the plane of equation:

$$f_{\alpha,\gamma}(z, y) = \cos\alpha \cdot y - \sin\alpha \cdot z + \gamma = 0 \tag{8}$$

where (z, y) correspond to the coordinates of point $((\Delta^+(k), \Delta^-(k))$ projected into the plane from node k, according to (7). This line separates the points of I in $I^+_{\alpha,\gamma}$ and $I^-_{\alpha,\gamma}$:

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

Figure 2 provides a graphical representation of this separation. Fixed the parameters (α, γ), we set:

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

- $FN(\alpha, \gamma) = |I^-_{\alpha,\gamma} \cap I^+|$, 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 $recall(\alpha, \gamma) = \frac{TP(\alpha,\gamma)}{TP(\alpha,\gamma)+FN(\alpha,\gamma)}$.

Observe that $0 \leq F_{score}(\alpha, \gamma) \leq 1$ and $F_{score}(\alpha, \gamma) = 1$ iff there are no classification errors.

The parameters (α, γ) are optimized by adopting the F_{score} maximization criterion:

$$(\hat{\alpha}, \hat{\gamma}) = \underset{\alpha,\gamma}{\operatorname{argmax}} \; F_{score}(\alpha, \gamma). \tag{9}$$

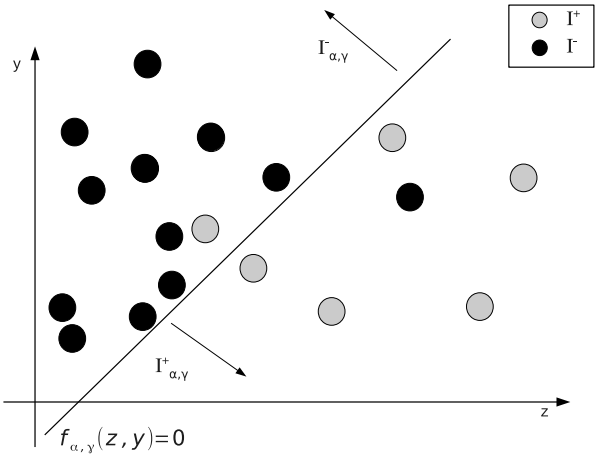


Figure 2: Graphical representation of the line separating positive and negative labeled nodes.

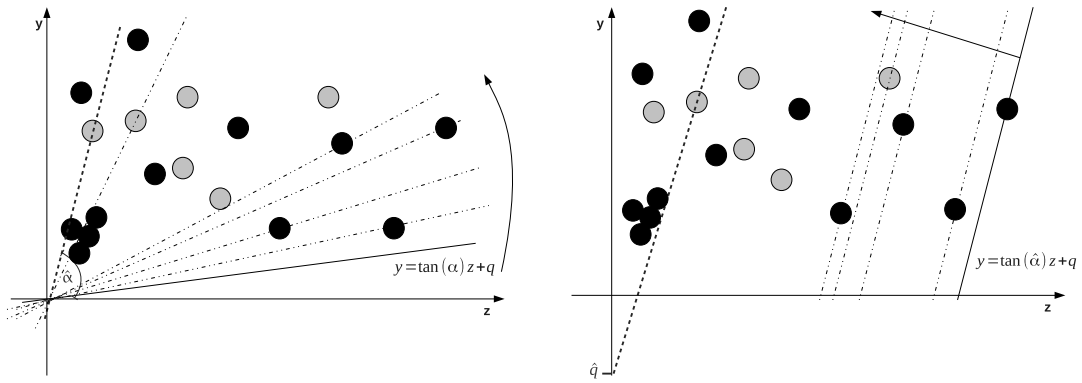
To actually compute the optimal parameters $(\hat{\alpha}, \hat{\gamma})$, although does exist an exact algorithm for this problem working in time $\mathcal{O}(|S|^2 \cdot \log |S|)$, we adopt a more efficient two-step approximation algorithm:

a) *Compute* $\hat{\alpha}$. The procedure computes the slopes of the lines crossing the origin and each point $\Delta(k) \in I$. Then it searches the line which maximizes the F_{score} criterion by scanning 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 $[0, \frac{\pi}{2}]$.

b) *Compute* $\hat{\gamma}$. Compute the intercepts of the lines whose slope is $\tan \hat{\alpha}$ and crossing each point belonging to 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 $y = \tan \hat{\alpha} \cdot z + q$, then we set $\hat{\gamma} = -\hat{q} \cos \hat{\alpha}$.

Figure 3 shows the two steps of the proposed approximation procedure.

Both step a) and step b) can be computed in $\mathcal{O}(|S| \log |S|)$ computational time.



a) b)

Figure 3: Graphical view of the α (a) and γ (b) parameters estimation procedure.

5.2.3. Optimal Line Parameters Correspond to Optimal Network Parameters

The F-score maximization criterion to optimize the parameters (α, γ) is justified by the following:

Fact 3. *$F_{score}(\alpha, \gamma) = 1$ iff $x(S^+, S^-)$ is an equilibrium state of the sub-network $H_{S|\boldsymbol{x}(U^P,U^N)}$.*

Proof. ($\Rightarrow$). To simplify the notation, by considering $H_{S|\boldsymbol{x}(U^P,U^N)} = < \boldsymbol{W}_{ss}, \boldsymbol{\gamma}^s - \boldsymbol{W}_{us}^T \boldsymbol{x}(U^P, U^N), \alpha >$, we set the threshold $\boldsymbol{\gamma}^s - \boldsymbol{W}_{us}^T \boldsymbol{x}(U^P, U^N) \equiv \boldsymbol{\theta}^s$. If $F_{score}(\alpha, \gamma) = 1$ there are no misclassification errors, i.e $I^+ = I^+_{\alpha,\gamma}$ 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) from (1), (7) and (8) implies that

$$\sin\alpha \cdot \sum_{j \in S^+ \cup\, U^P} w_{kj} - \cos\alpha \cdot \sum_{j \in S^- \cup\, U^N} w_{kj} - \theta_k^s > 0 \;\; (*)$$

for each $k \in S^+$, and the condition (ii) implies that

$$\sin\alpha \cdot \sum_{j \in S^+ \cup\, U^P} w_{kj} - \cos\alpha \cdot \sum_{j \in S^- \cup\, U^N} w_{kj} - \theta_k^s \leq 0 \;\; (**)$$

for each $k \in S^-$, where θ_k^s is the k^{th} component of the threshold vector $\boldsymbol{\theta}^s$. Hence, $\boldsymbol{x}(S^+, S^-)$ is an equilibrium state for the network $H_{S|\boldsymbol{x}(U^P,U^N)}$, since

it is easy to see that, when both (*) and (**) conditions hold, the state of the network does not change.

($\Leftarrow$). Suppose $x(S^+, S^-)$ is an equilibrium state of the sub-network $H_{S|\boldsymbol{x}(U^P,U^N)}$. This means that the condition (*) holds for each element $k \in S^+$ and the condition (**) holds for each element $k \in S^-$. Accordingly, the conditions (i) and (ii) hold and $I^+ = I^+_{\alpha,\gamma}$ and $I^- = I^-_{\alpha,\gamma}$. So there are no misclassification errors during the F-score optimization, and hence $F_{score}(\alpha, \gamma) = 1$. □

5.3. Finding the unknown labels by Network Dynamics

After the computation of the optimal parameters $(\hat{\alpha}, \hat{\gamma})$, we consider the sub-network $H_{U|\boldsymbol{x}(S^+,S^-)}$:

$$H_{U|\boldsymbol{x}(S^+,S^-)} = < \boldsymbol{W}_{uu}, \hat{\boldsymbol{\gamma}}^u - \boldsymbol{W}^T_{su}\boldsymbol{x}(S^+, S^-), \hat{\alpha} > = < \boldsymbol{W}_{uu}, \boldsymbol{\theta}^u, \hat{\alpha} > \quad (10)$$

where $\hat{\gamma}^u_i = \hat{\gamma}$ for each $i \in \{1, 2, \ldots, h\}$, and $\boldsymbol{\theta}^u \equiv \hat{\boldsymbol{\gamma}}^u - \boldsymbol{W}^T_{su}\boldsymbol{x}(S^+, S^-)$.

Fixed an initial state $u_i = 0$ for each $i \in \{1, 2, \ldots, h\}$, 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. Note that the resulting update rule is similar to (1), substituting γ_i with θ^u_i and restricting the dynamics only to unlabeled neurons U:

$$u_i(t+1) = \begin{cases} \sin\alpha & \text{if } \sum_{j=1}^{i-1} w_{ij}u_j(t+1) + \sum_{k=i+1}^{h} w_{ik}u_k(t) - \theta^u_i > 0 \\ -\cos\alpha & \text{if } \sum_{j=1}^{i-1} w_{ij}u_j(t+1) + \sum_{k=i+1}^{h} w_{ik}u_k(t) - \theta^u_i \leq 0 \end{cases} \quad (11)$$

If $\hat{\boldsymbol{u}}$ is the stable state reached by this dynamics, the final solution (U^+, U^-) is:

$$\begin{aligned} U^+ &= \{i \in U \mid \hat{u}_i = \sin\hat{\alpha}\} \\ U^- &= \{i \in U \mid \hat{u}_i = -\cos\hat{\alpha}\} \end{aligned} \quad (12)$$

As explained in the previous sections, also the update rule (11) converges to a local minimum of the energy for the network H, corresponding to the the network state $(\bar{\boldsymbol{u}}, \boldsymbol{x}(S^+, S^-))$.

5.4. Analysis of COSNet

Figure 4 shows the pseudocode of the *COSNet* algorithm. The Lines

Figure 4: Pseudocode of the *COSNet* algorithm.

`Input`:
- neuron set V;
- bipartition (U, S) of V;
- bipartition (S^+, S^-) of S
- connection matrix $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{uu} & \boldsymbol{W}_{su}^T \\ \boldsymbol{W}_{su} & \boldsymbol{W}_{ss} \end{pmatrix}$

`begin algorithm`

01: `Generate a random number` m drawn from the distribution $B(\mid U \mid, \frac{|S^+|}{|S|})$
02: `Randomly assign to` U^P m `elements of` U `and set` $U^N := U \setminus U^P$
03: `for each` $k \in S$ `do`
04: $\quad \Delta_k^+ := \sum_{j \in S^+ \cup U^P} w_{kj}$
05: $\quad \Delta_k^- := \sum_{j \in S^- \cup U^N} w_{kj}$
06: `end for`
07: $[\hat{\alpha}, \hat{\gamma}] :=$ `FindOptimalLine`$\big(S^+, S^-, \Delta^+, \Delta^-\big)$
08: `for each` k in S `do`
09: $\quad$ `if` k in S^+ `then` $x_k(S^+, S^-) := \sin\hat{\alpha}$
10: $\quad$ `else` $x_k(S^+, S^-) := -\cos\hat{\alpha}$
11: `end for`
12: $\boldsymbol{\theta}^u := \hat{\gamma} \cdot \boldsymbol{e}^u - \boldsymbol{W}_{su}^T \boldsymbol{x}(S^+, S^-)$
13: $[\hat{\boldsymbol{u}}] :=$ `RunSubNet`$\big(\boldsymbol{W}, U, \hat{\alpha}, \boldsymbol{\theta}^u\big)$

`end algorithm`

`Output`: the equilibrium state $\hat{\boldsymbol{u}}$.

$1-2$ generates a temporary bipartition of U (Section 5.1) and it takes time $\mathcal{O}(|U|)$.

The lines $3-6$ compute for each node in S the weighted sum of its positive and negative neighbours, that is the coordinates of the projection of the nodes into the plane, taking time $\mathcal{O}(|\boldsymbol{W}_{su}| + |\boldsymbol{W}_{ss}|)$, where with $|\boldsymbol{W}|$ we mean the number of non-null components of the matrix $\boldsymbol{W}$.

The procedure `FindOptimalLine` at line 7 computes the optimal parameters $(\hat{\alpha}, \hat{\gamma})$ with $\mathcal{O}(|S| \log |S|)$ time complexity, by finding the straight line which best separates the points in I^+ from those in I^- (Section 5.2.2).

The lines $8-11$ takes time $\mathcal{O}(|S|)$, whereas line 12, which computes the thresholds for the sub-network $H_{U|\boldsymbol{x}(S^+,S^-)}$, has a time complexity $\mathcal{O}(|\boldsymbol{W}_{su}|)$.

Finally, line 13 realizes the dynamics of the sub-network $H_{U|\boldsymbol{x}(S^+,S^-)}$. The complexity of this step depends on the number of iterations needed for convergence, and each iteration takes time $\mathcal{O}(|\boldsymbol{W}_{uu}|)$. We empirically observed that the network in average converges in few iterations, confirming the observations of Karaoz et al. (2004).

Overall, recalling that $|S| < |V| = n$, the *COSNet* algorithm takes time $\mathcal{O}(|S| \log |S| + |\boldsymbol{W}|)$ which is almost linear in the number of neurons when the connection matrix is sparse, that is when the number of non zero entries in $\boldsymbol{W}$ is $|\boldsymbol{W}| = \mathcal{O}(|V|)$.

5.5. Strengths and Drawbacks of the supervised two-step approximate algorithm

Our proposed supervised algorithm described in the previous section can correctly learn the near-optimal parameters of the Hopfield network in most cases, as shown in the experimental section (Section 6), and its $\mathcal{O}(|S| \log |S|)$ computational complexity allows its application to large networks, for instance complex biomolecular networks or very large social networks.

On the other hand in some special cases it fails, or may incur in poor estimates of the network parameters. Consider for instance the first two cases depicted in Fig. 5: in both cases it is easy to see that the first step of the algorithm cannot find the right parameter $\alpha \simeq 0$ (Fig. 5 (a)) or $\alpha \simeq \frac{\pi}{2}$ (Fig. 5 (b)). By moving the intercept in the second step of the algorithm we can only partially correct the error. Note that this happens when the discrimination between positive and negative nodes depends respectively only on the "negative" (Fig. 5 (a)) or "positive" (Fig. 5 (b)) neighborhoods of each node. This means that the effect of the positive nodes in Fig. 5 (a) is marginal and symmetrically that negative nodes are uninfluential in Fig. 5 (b). Regarding these cases, we point out two main aspects: first, in our GFP experiments (Section 6) we never found similar cases; second, the optimum values for parameter α in such cases would lead to poor solutions, since the dynamics of the network with these parameters may likely converge to the trivial all negative (when $\alpha = 0$) or all positive ($\alpha = \frac{\pi}{2}$) states. Moreover, in most problems (including GFP) it is unlikely that connections only towards positive or only toward negative nodes play an exclusive role in the prediction of node labels. Two other interesting cases are represented in Fig. 5 (c), (d). In Fig. 5 (c) the large majority of points lie close to the abscissa, meaning that the main contribution for each node comes from "positive" neighbours $\Delta(k^+)$ (that is only positively labeled nodes are connected with relatively

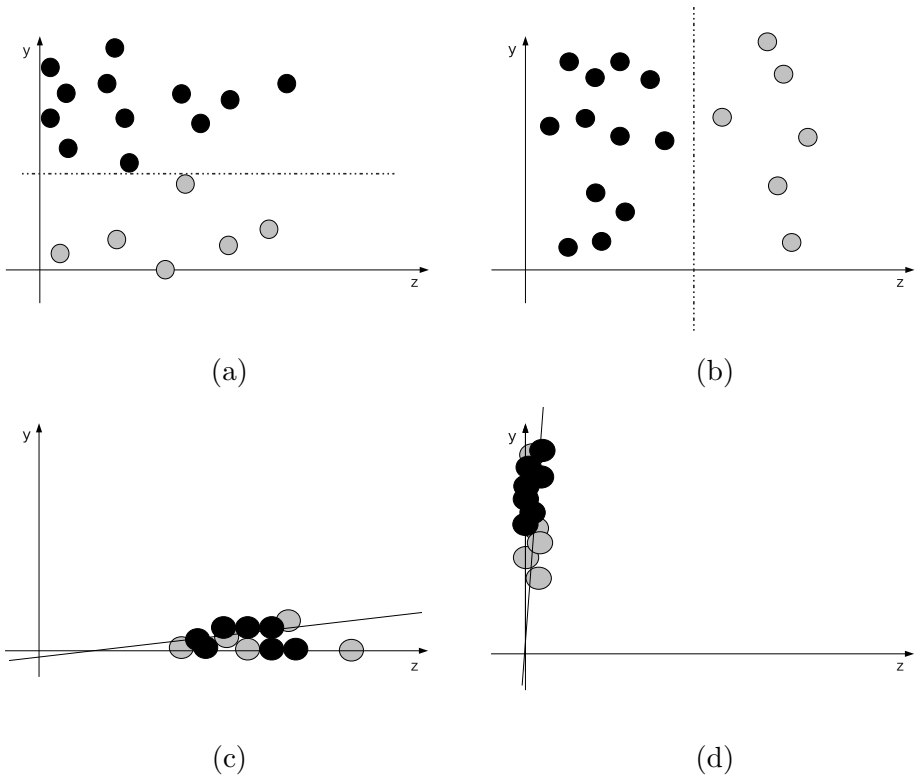


Figure 5: Limit cases of the optimization algorithm. The optimum line is horizontal (a) or vertical (b), or the large majority of points lie close to the abscissa (c) or to the ordinate (d).

large weights to other nodes), while "negative" contributions $\Delta(k^-)$ (see Section 5.2) are close to 0. This may happen, for instance, when we have a highly unbalanced data set with a large prevalence of positive nodes. The symmetric situation is represented in Fig. 5 (d): the large majority of points lie close to the ordinate, and the contribution of positive neighborhood is negligible. This may happen when we have a high unbalance in favour of negative examples. It is easy to see that both these extreme conditions may lead to trivial solutions (i.e. to "all positive" or "all negative" labelings) when we run the Hopfield network $H_{U|\boldsymbol{x}(S^+,S^-)}$, since in the first case the algorithm correctly estimates α values close to 0, leading to uninfluential activation values for positives ($\sin\alpha \simeq 0$) and very strong activation values for negatives ($-\cos\alpha \simeq -1$), while in the second case the algorithm correctly estimates

$\alpha \simeq \frac{\pi}{2}$, leading to a pair of activation values close to $\{1, 0\}$ respectively for positive and negative nodes. Unfortunately, while the estimation of α is correct in both cases, this situation leads to poor results. In our GFP experiments we observed for several functional classes a situation similar to that depicted in Fig. 5 (d), since in several cases negative genes largely outnumber positives. To effectively deal with these cases we introduced a regularized version of *COSNet*, discussed in the next Section.

5.6. Model regularization

As shown in the previous section, when the optimal value $\hat{\alpha}$ of the parameter α is very close to $\frac{\pi}{2}$ (resp. 0), the network dynamics is characterized by a too strong influence of positive neurons (resp. negative neurons), leading to trivial solutions.

To prevent this behaviour, we add to the energy function $E_{|s}(\boldsymbol{u})$ of the network $H|_{U,\boldsymbol{x}(S^+,S^-)}$ the regularization term

$$\left(\sum_{i=1}^{h}(au_i + b) - \nu_u\right)^2, \tag{13}$$

where $a = \frac{1}{\sin\hat{\alpha}+\cos\hat{\alpha}}$, $b = \frac{\cos\hat{\alpha}}{\sin\hat{\alpha}+\cos\hat{\alpha}}$ and $\nu_u = p_s \cdot h$, with $p_s = \frac{|S^+|}{|S|}$.

Since $\sum_{i=1}^{h}(au_i + b)$ is the number of positive neurons in $\boldsymbol{u}$, the term (13) is minimized when the number of positive neurons in $\boldsymbol{u}$ is ν_u. The choice of this regularization term is justified from the fact that, according to the model described in Section 5.1, $\nu_u = \underset{z}{\text{argmax}}\ P(z)$.

By adding the regularization term to $E_{|s}(\boldsymbol{u})$, we obtain a new energy function $E(\boldsymbol{u})$:

$$E(\boldsymbol{u}) = -\frac{1}{2}\sum_{i\neq j} w_{ij}u_iu_j + \sum_{i=1}^{h} u_i\theta_i^u + \eta\left(\sum_{i=1}^{h}(au_i + b) - \nu_u\right)^2, \tag{14}$$

where η is a regularization parameter and $\theta_i^u = \gamma - \sum_{j\in S} w_{ij}$ (Section 5.3). By few simplifications, we obtain up to a constant:

$$E(\boldsymbol{u}) = -\frac{1}{2}\sum_{i=1}^{h}\sum_{\substack{j=1\\ j\neq i}}^{h}\tilde{w}_{ij}u_iu_j + \sum_{i=1}^{h} u_i\tilde{\theta}_i^u \tag{15}$$

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

In principle, η should be significantly different from 0 when $\hat{\alpha} \simeq \frac{\pi}{2}$ or $\hat{\alpha} \simeq 0$. A possible choice of η which provides this behaviour is:

$$\eta = \ \beta|\tan((\hat{\alpha} - \tfrac{\pi}{4}) * 2)| \tag{16}$$

where β is a non negative real constant. By tuning β we can finely control the influence of the new energy term on the network dynamics.

5.7. Validation of COSNet optimization

In this section we show that the optimal parameters computed in step 2 of *COSNet* put the known labeling closer to a global minimum of the network energy with respect to the default parameters of a non cost-sensitive Hopfield network. To this end, we consider a real data set of protein-protein interactions (Von Mering et al., 2002) in which a labeling $\overline{\boldsymbol{x}} \in \{1, -1\}^{|V|}$ of V is known. By applying *COSNet* we approximate the optimal parameters $(\hat{\alpha}, \hat{\gamma})$. Accordingly, 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 "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 $P_{\boldsymbol{z}}$ 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 $B(|V|, \rho_z)$, where ρ_z is the rate of positive components in $\boldsymbol{z}$.

To estimate $P_{\boldsymbol{z}}$, 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 $P_{\boldsymbol{z}}$, and in our setting $t = 1000$. Since many studies suggest that for determining a reliable confidence interval of a binomial proportion we need $tP_{\boldsymbol{z}}, t(1 - P_{\boldsymbol{z}}) \geq 5$(or 10) (Brown et al., 2001), in order to define the confidence interval of $P_{\boldsymbol{z}}$ 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

Table 1: Confidence interval estimation for the probabilities $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$ at a confidence level 0.95.

Class	Confidence interval					Class	Confidence interval			
	$P_{\overline{\mathbf{x}}(\hat{\alpha})}$		$P_{\overline{\boldsymbol{x}}}$				$P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$		$P_{\overline{\boldsymbol{x}}}$	
	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

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 (Agresti & Coull, 1998), which is more stable for values of Y closer to the outliers (Brown et al., 2001). 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 $\delta = 0.05$, we estimated the confidence interval for both $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$. In Table 1 we report the comparison of the confidence intervals of $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ and $P_{\overline{\boldsymbol{x}}}$ for some functional classes. We distinguish two main cases: a) both the confidence intervals coincide with the minimum interval $[0, 0.0030]$, meaning that the known labeling is a global minimum with high probability; b) both lower and upper bounds of $P_{\overline{\boldsymbol{x}}(\hat{\alpha})}$ are less than the corresponding bounds of $P_{\overline{\boldsymbol{x}}}$. 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 $P_{\overline{\boldsymbol{x}}}$. 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.

6. Experiments

We applied *COSNet* to a typical unbalanced classification problem in the domain of computational biology: the gene function prediction problem (Sect. 2).

In our setting we decomposed the multi-label multi-class classification problem in a set of two-class classification problems, i.e. one distinct dichotomic classification problem for each class of the ontology, according to a widely adopted approach to this problem (Friedberg, 2006). For most classes the set of negative genes largely outnumbers the set of positives, thus leading to very unbalanced dichotomic classification problems.

We performed predictions of gene functions at genome-wide level in the *S.cerevisiae* organism (yeast), using the whole FunCat ontology (Ruepp et al., 2004) [2]. FunCat (Functional Categories) is one of the most used biological taxonomy to classify gene functions. The FunCat hierarchy is represented by a forest of trees, in which the functional categories represent nodes of trees and the edges represent hierarchical relationships between classes. It consists of 28 main functional categories (or branches) that cover general fields like cellular transport, metabolism and cellular communication/signal transduction. These main functional classes are divided into a set of subclasses with up to six levels of increasing specificity, according to a tree-like structure that accounts for different functional characteristics of genes and gene products.

6.1. Data sets

We predicted gene functions by means of six different biomolecular data sets, using both each data set separately and the data sets all together by integrating them through suitable data fusion techniques (Sect. 6.4.2).

[2] 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.

The main characteristics of the data can be summarized as follows (Table 2):

- *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 (Finn et al., 2010). 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 (Eddy, 1998). This dataset contains 3528 genes and 5724 features.

- *Expr* data contains 250 gene expression measures of 4523 genes relative to two experiments described in (Spellman et al., 1998) and (Gasch et al., 2000).

- *PPI-BG* data set contains protein-protein interaction data downloaded from the BioGRID database (Stark et al., 2006). 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 (Von Mering et al., 2002).

- *SP-sim* data set contains pairwise similarities between 3527 yeast genes represented by Smith and Waterman log-E values between all pairs of yeast sequences (Lanckriet et al., 2004).

In our experiments, we discarded classes with less than 20 positive examples, in order to avoid folds with no positives in the cross validation procedure and to preserve the generalization capabilities of the resulting classifiers.

6.2. Preprocessing and Normalization

For protein-protein interaction data (PPI) we adopted the scoring function used by Chua et al. (2007), which assigns to genes i and j the similarity

Table 2: Data sets

Name	n genes	n features	type	Classes	notes
Pfam-1	3529	4950	binary	211	sparse (7060 domains annotations)
Pfam-2	3528	5724	real	211	dense (20183019 scores > 0)
Expr	4532	250	real	230	fold change, dense (1076648 scores $\neq 0$)
PPI-BG	4531	5367	binary	232	sparse (149016 interactions)
PPI-VM	2338	2559	binary	176	sparse (22266 predicted interactions)
Sp-sim	3527	6349	real	211	dense (close to full coverage)

score

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

where N_k is the set of the neighbors of gene k (k is included). In this way the functional similarity between a pair of genes depends also on the their common shared neighbours: i.e. two genes are similar if they are directly connected or if they are connected through an intermediate node (that is a common neighbour).

In the remaining data sets each gene is associated with a feature vector, and the score for each gene pair is set to the Pearson's correlation coefficient of the corresponding feature vectors. We removed edges with negative values of the correlation coefficient, except for gene expression data (*Expr* data), for which we computed the squared correlation coefficient in order to take into account relationships of "inverse regulation" between genes (i.e. genes that are up-regulated when others are down-regulated). Then we adopted a technique to sparsify the resulting network. More precisely, given the correlation matrix $\boldsymbol{W}$, for each gene i we set $t_i = \max_j w_{ij}$, and then we set a general threshold $t^* = \min_i t_i$: elements w_{ij} such that $w_{ij} < t^*$ were set to 0. In this way we can assure that there are no isolated "singleton" nodes/genes in the network.

Finally, each obtained network $\boldsymbol{W}$ underwent a graph Laplacian normalization (Smola & Kondor, 2003): each element w_{ij} of $\boldsymbol{W}$ has been divided 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 $\boldsymbol{D}$ is a $n \times n$ diagonal matrix such that $d_{ii} = \sum_j w_{ij}$, then the normalized matrix $\hat{\boldsymbol{W}}$ is:

$$\hat{\boldsymbol{W}} = \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2} \tag{17}$$

6.3. Experimental setup

We compared *COSNet* with semi-supervised and supervised machine learning methods proposed in the literature for the gene function prediction problem in a "flat" setting, that is predicting each functional class separately, without considering its hierarchical relationships with the other classes. We considered the *GAIN* algorithm (Karaoz et al., 2004), based on Hopfield networks (Section 3), since our proposed method is a regularized cost-sensitive generalization of an Hopfield network. We applied also *Zhu-LP*, a popular semi-supervised label propagation learning algorithm based on Gaussian random fields over a continuous state space, and its class mass normalized version *Zhu-LP-CMN*, that takes into account the unbalance between positive and negative examples (Zhu et al., 2003). Finally we considered a supervised learning algorithm, i.e. linear (*SVM-l*) and Gaussian (*SVM-g*) Support Vector Machines (SVMs), since it has been shown that SVMs are among the best supervised algorithms for predicting gene function (Brown et al., 2000; Pavlidis et al., 2002).

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, we adopted as the main performance measure the *F-score*, that is the harmonic mean between *precision* and *recall*; we did not consider the classification accuracy because with unbalanced classes a naive classifier which predicts all the genes as negative obtains yet a high accuracy.

The selection of the β parameter for the regularized version of *COSNet* (Section 5.6) has been performed through a tuning procedure on model data sets. Moreover, since the *Zhu-LP* algorithm provides just continuous prediction scores, to obtain a classifier we set the threshold for the positive class to $\frac{1}{2}$, as suggested by the authors (Zhu et al., 2003).

6.4. Results and discussion

In this Section we report and discuss the results of the compared methods using both single and integrated data sources.

6.4.1. Classification using single data sets

Table 3 shows for each dataset the performances in terms of average precision, recall and F-score across all the functional classes.

Table 3: Average precision, recall and F-score of the compared methods using single data sets.

	Pfam-1			Pfam-2		
Method	Prec	Rec	F	Prec	Rec	F
Zhu-LP	0.407	0.067	0.104	0.614	0.151	0.226
Zhu-LP-CMN	0.286	0.545	0.336	0.210	0.524	0.273
GAIN	0.504	0.174	0.250	0.526	0.103	0.162
SVM-g	0.246	0.491	0.235	0.103	0.229	0.027
SVM-l	0.298	0.497	0.272	0.079	0.450	0.105
COSNet	0.322	0.399	0.347	0.445	0.371	0.375
R-COSNet	0.381	0.383	0.375	0.458	0.353	0.380
	Expr			PPI-BG		
Method	Prec	Rec	F	Prec	Rec	F
Zhu-LP	0.015	0.0001	0.0002	0.575	0.104	0.158
Zhu-LP-CMN	0.098	0.215	0.074	0.209	0.518	0.273
GAIN	0	0	0	0.232	0.033	0.049
SVM-g	0.023	0.210	0.019	0.157	0.342	0.118
SVM-l	0.070	0.287	0.053	0.173	0.437	0.155
COSNet	0.057	0.808	0.085	0.359	0.412	0.358
R-COSNet	0.107	0.177	0.130	0.383	0.378	0.371
	PPI-VM			Sp-sim		
Method	Prec	Rec	F	Prec	Rec	F
Zhu-LP	0.426	0.148	0.205	0.619	0.166	0.241
Zhu-LP-CMN	0.176	0.646	0.270	0.202	0.524	0.268
GAIN	0.342	0.064	0.097	0.502	0.104	0.161
SVM-g	0.243	0.518	0.235	0.449	0.347	0.190
SVM-l	0.196	0.477	0.189	0.070	0.588	0.110
COSNet	0.382	0.437	0.390	0.445	0.376	0.376
R-COSNet	0.392	0.425	0.396	0.458	0.351	0.383

COSNet and *R-COSNet* achieve the best average F-score w.r.t. all the other compared methods, and the difference is always significant at 10^{-6} significance level, according to the Wilcoxon signed-ranks test (Wilcoxon, 1945). This is the result of a good balance between precision and recall, since in terms of average precision *Zhu-LP* obtains quite always the best results (but at the expense of a low recall), and *SVMs* (in particular linear SVMs) achieves the best average recall results (but paying in terms of a relatively low average precision). It is worth noting that the regularized version of *COSNet* obtains slightly better results than its "vanilla" counterpart. On the average *R-COSNet* achieves a better precision, while *COSNet* a better recall. This is particularly noticeable with the least informative data sets (i.e. the *Expr* data set), where the regularization plays a significant role to "re-equilibrate" the wrong predictions of the unregularized version *COSNet*.

In order to analyze the behaviour of *COSNet* with functional classes at different levels of specificity, we also computed the F-score performances averaged per hierarchy level (Figure 6): level 1 refers to the root nodes of the FunCat forest, level i to nodes at distance $i-1$ from the root. Note that the number of positive examples decreases at higher levels: level 1 contains 18 classes with about 862 positive examples per class on the average, whereas level 5 includes 16 classes with about 39 positive examples per class. Level 3 is the level with the largest number of classes (92).

In Figure 6 we reported only the results of *R-COSNet*, since *COSNet* achieves similar but slightly worse results.

R-COSNet exhibits a reduced decay in F-score w.r.t. the other methods when the level increases, that is when more specific classes that better characterize the functions of a given gene are considered. Only the *Zhu-LP-CMN* method has a similar behaviour, but with worse performances (note that *Zhu-LP-CMN* is a cost-sensitive variant of *Zhu-LP*). These results show that cost-sensitive strategies maintain a relatively high F-score also with the most specific classes, reducing the decay in performance due to the lower number of positive examples associated to the most specific classes, confirming recent findings achieved in the context of cost-sensitive hierarchical ensemble methods (Cesa-Bianchi et al., 2012).

6.4.2. Classification using integrated data sets

We integrated the six biomolecular data sets (Section 6.1) using a disjunctive approach, i.e. we integrated all the data sets considering the union of all genes included in the data sets. In this way we obtained a set of 4665

genes and 232 functional classes with at least 20 positive examples. More precisely, we extended each of the six available adjacency matrices to the the size 4665×4665 of the integrated "consensus" network, by adding rows and columns of zeros when a given gene is not included in the corresponding data set. Then we integrated the resulting "extended" 4665×4665 adjacency matrices of each data set by adopting three different strategies for unweighted integration described below.

Unweighted sum (US). This is the simple sum of the individual adjacency matrices (corresponding each one to a different data set) divided by the number of the m available networks:

$$\boldsymbol{W}^* = \frac{1}{m} \sum_{d=1}^{m} \boldsymbol{W}^{(d)} \tag{18}$$

Max fusion (MF). The consensus matrix $\boldsymbol{W}^*$ is obtained element by element by taking the maximum edge weight ($w_{ij}^{(d)}$ is the weight associated to the edge (i, j) of the d^{th} data set):

$$w_{ij}^* = \max_d w_{ij}^{(d)} \tag{19}$$

Per edge unweighted sum (EUS). Each entry w_{ij}^* of the consensus matrix is the average of the corresponding entries $w_{ij}^{(d)}$ in the data sets for which both genes i and j are present at the same time. Formally, if $g^{(d)}$ is the set of genes of network $\boldsymbol{W}^{(d)}$, then we define $N_{ij} = \{d \in \{1, 2, \cdots, m\} \mid i, j \in g^{(d)}\}$. The "consensus weight" for each edge (i, j) is:

$$w_{ij}{}^* = \frac{1}{|N_{ij}|} \sum_{d \in N_{ij}} w_{ij}{}^{(d)} \tag{20}$$

With SVMs we cannot directly apply the integration methods described above, since the SVM algorithm needs a valid kernel to learn the functional classes. To this end we first constructed the Gram matrices $\boldsymbol{K}^{(d)}$ using the linear kernel function for each single data set $D^{(d)}$, and then we applied two different kernel fusion methods to integrate the sources of data, i.e. *unweighted kernel fusion (UKF)* and *Multiple kernel learning (MKL)* (see below).

Unweighted kernel fusion (UKF). The integrated kernel matrix $\boldsymbol{K}^*$ is obtained by averaging across the kernel matrices $\boldsymbol{K}^{(d)}$ constructed from the m sources of data:

$$\boldsymbol{K}^* = \frac{1}{m} \sum_{d=1}^{m} \boldsymbol{K}^{(d)} \tag{21}$$

Multiple kernel learning (MKL). MKL is an approach for integrating kernels to learn concurrently both the parameters of the classifier and the weights associated to each kernel, by solving a semi-infinite programming problem (Sonnenburg et al., 2006; Kloft et al., 2009). In this way we can obtain an integrated kernel matrix $\boldsymbol{K}^*$ through a weighted sum of the component kernels:

$$\boldsymbol{K}^* = \sum_{d=1}^{m} h^d \boldsymbol{K}^{(d)} \tag{22}$$

where the weights h^d are automatically learned from the data (Sonnenburg et al., 2006).

Table 4 shows the performances in terms of average precision, recall and F-score across classes for the methods *COSNet* (both unregularized and regularized versions), *Zhu-LP*, *Zhu-LP-CMN* and *GAIN* for each unweighted network integration method. The same table reports also the average precision, recall, and F-score for the *SVM-UKF* and *SVM-MKL* kernel fusion methods.

Both the regularized and unregularized version of *COSNet* show the highest F-score w.r.t. to the other compared methods, independently of the network integration technique. *COSNet* and *R-COSNet* average results are comparable between the different integration methods, and always significantly better (as expected) w.r.t. to the best "single source" results (Table 3). Moreover, according to the Wilcoxon rank-sums test, the average F-score is higher in *COSNet* and *R-COSNet* w.r.t. to all the other semi-supervised and supervised compared methods at 10^{-5} significance level. With the most informative integrated networks the need of the regularization is less apparent. Indeed, even if *R-COSNet* registers slightly better results than *COSNet*, the difference is not statistically significant. Also with the integrated data *R-COSNet* attains a slightly larger precision and *COSNet* a slightly larger recall. Analogously to the "single source" classification task (Section 6.4.1), the best results of *COSNet* and *R-COSNet* are due to a nice balancing between precision and recall (Table 4),

Table 4: Average precision, recall and F-score using integrated data.

US integration			
Method	Prec	Rec	F
Zhu-LP	0.614	0.078	0.129
Zhu-LP-CMN	0.367	0.606	0.411
GAIN	0.147	0.003	0.005
COSNet	0.513	0.506	0.494
R-COSNet	0.519	0.494	0.499
MF integration			
Method	Prec	Rec	F
Zhu-LP	0.289	0.017	0.029
Zhu-LP-CMN	0.384	0.593	0.411
GAIN	0.012	0.0002	0.0004
COSNet	0.504	0.503	0.489
R-COSNet	0.512	0.498	0.498
EUS integration			
Method	Prec	Rec	F
Zhu-LP	0.537	0.048	0.083
Zhu-LP-CMN	0.365	0.600	0.406
GAIN	0.084	0.002	0.003
COSNet	0.497	0.515	0.488
R-COSNet	0.505	0.500	0.494
SVM-KF integration			
Method	Prec	Rec	F
SVM-UKF	0.307	0.607	0.393
SVM-MKL	0.628	0.294	0.385

Interestingly enough, the other Hopfield network considered in the experiments (*GAIN*), achieves the worst results, worsening also its performances w.r.t. to "single source" data (Table 3). This is likely due to the problem of the trivial attractors described in Section 3, since in the integrated network we have a larger number of nodes and relatively more unbalanced classes. Also with integrated data the cost-sensitive version of *Zhu-LP* achieves significantly better than its "vanilla" counterpart, showing that methods that take into account the unbalance of the data are well-suited to *GFP*.

Figure 7 reports the F-score averaged by taxonomy level of the compared semi-supervised methods for each network integration technique applied in the experiments. *R-COSNet* obtains the highest F-score at each level and for each network integration method. It is worth noting that the F-score is relatively high also at the lowest levels of the hierarchy, where the classes are more unbalanced and difficult to be predicted. In the context of *GFP*, these results are particularly relevant, since functional classes belonging to the lowest level of the hierarchy are the most specific and informative about the functional characteristics of genes. The same behaviour is also maintained by the other cost-sensitive method (*Zhu-LP-CMN*), while both *Zhu-LP* and *GAIN* suffer of a relevant decay in performances when more specific classes need to be predicted. These results, according to the results of the previous section, witness for unbalanced data-aware methods for gene function prediction.

6.4.3. Comparison with the minimum cut algorithm

In order to further validate our method, in this section we compare *COSNet* with the minimum cut algorithm proposed in (Murali et al., 2006) to predict gene functions. The problem of minimizing the overall energy (2) can be optimally solved in $O(nm \log n)$ time using a min-cut/max-flow algorithm (Goldberg & Tarjan, 1988), where n is the number of nodes and m the number of edges in the network. We tested the min-cut algorithm with the US integrated network (Section 6.4.2), one of the most informative nets used in our experiments (Table 4). Due to its time complexity, we run the min-cut algorithm on the subset of the 47 descendant categories of the FunCat functional trees rooted in "01" (Metabolism) and "02" (Energy).

COSNet significantly outperforms the min-cut algorithm in terms of average precision, recall and F-score (Table 5). This is not surpising, since in this task we need cost-sensitive algorithms to take into account the unbalance between positive and negative examples. Indeed the min-cut algorithm

Table 5: Comparison of *COSNet* and min-cut algorithm in terms of precision, recall, F-score and accuracy averaged across the FunCat trees rooted in "01" and "02".

US integration				
`Method`	`Prec`	`Rec`	`F`	`Acc.`
Min-cut	0.028	0.048	0.032	0.938
COSNet	0.529	0.467	0.488	0.978

performs similarly to the GAIN algorithm, confirming the results obtained by Murali et al. (2006). Interestingly enough, the results in terms of the overall accuracy (the ratio of correctly predicted examples) are quite comparable between *COSNet* and the min-cut algorithm (Table 5), achieving both methods very high accuracy rates (but recall that in this unbalanced context the F-score is a more appropriate performance measure). Moreover, *COSNet* takes 413.3 seconds to compute the 10-folds CV over the 47 selected classes, whereas for the same task the min-cut algorithm needs 25080.7 seconds (Linux system with i7 processor and 8 Gb RAM). This is a relevant aspect when we need to consider large-size data and a high number of classes.

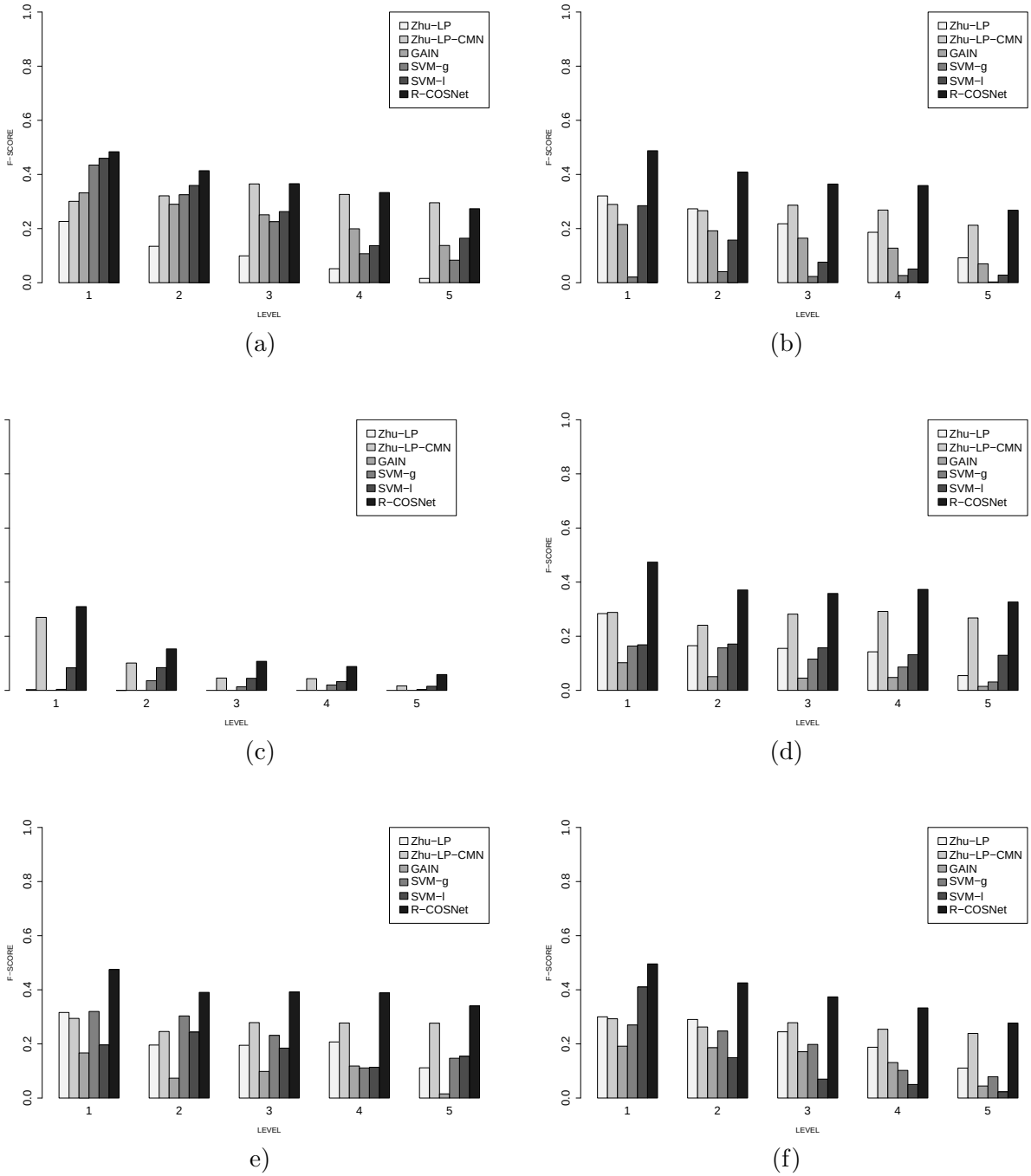
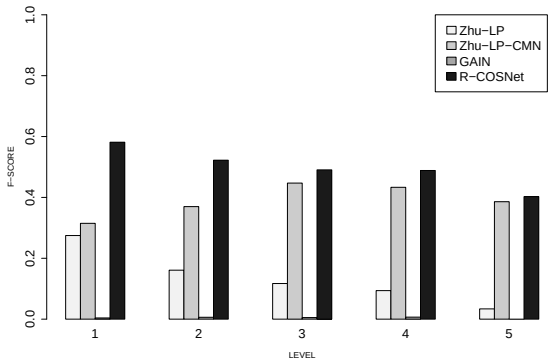
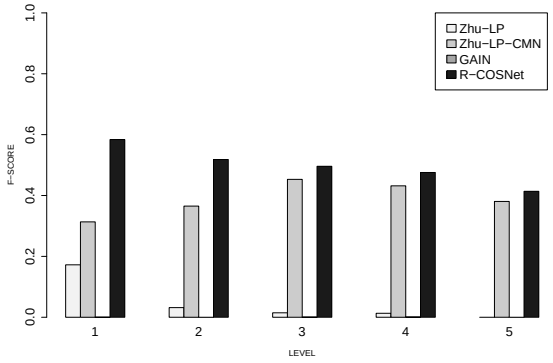


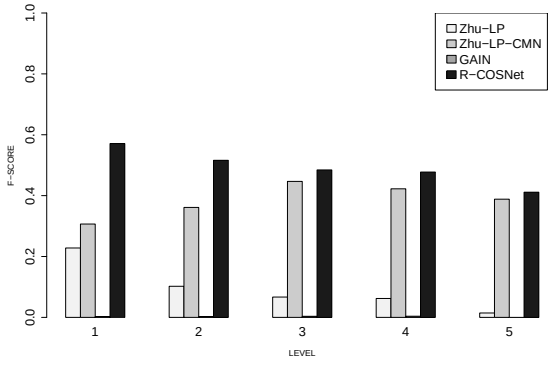
Figure 6: Compared F-scores between methods, averaged by FunCat level for Pfam-1 (a), Pfam-2 (b), Expr (c), PPI-BG (d), PPI-VM (e), Sp-sim (f) data sets.



(a)



(b)



(c)

Figure 7: F-scores averaged by FunCat level, compared between different semi-supervised methods. (a) *US*, (b) *MF*, (c) *EUS* network integration.

7. Conclusions

In this paper we introduced *COSNet*, an algorithm for learning node labels in graphs with unbalanced labels. *COSNet* is based on a family of parametrized Hopfield networks; it preserves the prior knowledge coded in the partially known labeling of the graph by restricting the dynamics solely to the unlabeled nodes, and adopts a cost sensitive strategy to manage the unbalance between positive and negative labels. To address extremely unbalanced classification problems, we presented also a regularized version of the algorithm.

The restriction of the network dynamics to the unlabeled part of the network makes the algorithm fast even on large input data. We validated the algorithm on the problem of gene function prediction at genome-wide level in yeast, considering more than two hundreds of functional classes. We compared *COSNet* with other state-of-the-art methods, and the results show that our proposed approach is able to automatically find neuron states and thresholds that better fit the unbalanced labeling of the network.

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their comments and suggestions, and gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

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

Azran, A. (2007). The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning* ICML '07 (pp. 49–56). New York, NY, USA: ACM.

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In *In COLT* (pp. 624–638). Springer.

Belkin, M., & Niyogi, P. (2003). Using Manifold Structure for Partially Labeled Classification. *Advances in Neural Information Processing Systems*, *15*, 929–936.

Bengio, Y., Delalleau, O., & Le Roux, N. (2006). Label Propagation and Quadratic Criterion. In O. Chapelle, B. Scholkopf, & A. Zien (Eds.), *Semi-Supervised Learning* (pp. 193–216). MIT Press.

Bertoni, A., Frasca, M., & Valentini, G. (2011). COSNet: a cost sensitive neural network for semi-supervised learning in graphs. In *Machine Learning and Knowledge Discovery in Databases - European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)* (pp. 219–234). Springer Berlin/Heidelberg.

Bogdanov, P., & Singh, A. K. (2010). Molecular function prediction using neighborhood features. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, *7*, 208–217.

Borgatti, S., Mehra, A., Brass, D., & Labianca, G. (2009). Network Analysis in the Social Sciences. *Science*, *232*, 892–895.

Brown, L. D., Cai, T. T., & Dasgupta, A. (2001). Interval estimation for a binomial proportion. *Statistical Science*, *16*, 101–133.

Brown, M. P. S. et al. (2000). 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*, *97*, 267–276.

Cesa-Bianchi, N., Re, M., & Valentini, G. (2012). Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, *88*, 209–241.

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

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

Chua, H. N., Sung, W.-K., & Wong, L. (2006). Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. *Bioinformatics*, *22*, 1623–1630.

Delalleau, O., Bengio, Y., & Le Roux, N. (2005). Efficient non-parametric function induction in semi-supervised learning. In R. G. Cowell, & Z. Ghahramani (Eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados* (pp. 96–103).

Deng, M., Chen, T., & Sun, F. (2004). An integrated probabilistic model for functional prediction of proteins. *J. Comput. Biol.*, *11*, 463–475.

Dorogovtsev, S., & Mendes, J. (2003). *Evolution of networks: From biological nets to the Internet and WWW*. Oxford: Oxford University Press.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, *14*, 755–763.

Finn, R., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J., Gavin, O., Gunesekaran, P., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E., Eddy, S., & Bateman, A. (2010). The Pfam protein families database. *Nucleic Acids Research*, *Database Issue 38*, D211–222.

Friedberg, I. (2006). Automated protein function prediction-the genomic challenge. *Brief. Bioinformatics*, *7*, 225–242.

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

Goldberg, A., & Tarjan, R. (1988). A new approach to the maximum flow problem. *Journal of the ACM (JACM)*, *35*, 921–940.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective compatational abilities. *Proc. Natl Acad. Sci. USA*, *79*, 2554–2558.

Karaoz, U. et al. (2004). Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc. Natl Acad. Sci. USA*, *101*, 2888–2893.

Kloft, M., Brefeld, U., Sonnenburg, S., Laskov, P., Müller, K.-R., & Zien, A. (2009). Efficient and accurate lp-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22* (pp. 997–1005).

Lanckriet, G. R., Deng, M., Cristianini, N., Jordan, M. I., & Noble, W. S. (2004). Kernel-based data fusion and its application to protein function prediction in yeast. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, (pp. 300–311).

Li, X., Chen, H., Li, J., & Zhang, Z. (2010). Gene function prediction with gene interaction networks: a context graph kernel approach. *Trans. Info. Tech. Biomed.*, *14*, 119–128.

Liu, H., & Hu, Y. (2009). An application of hopfield neural network in target selection of mergers and acquisitions. *Business Intelligence and Financial Engineering, International Conference on*, *0*, 34–37.

Marcotte, E., Pellegrini, M., Thompson, M., Yeates, T., & Eisenberg, D. (1999). A combined algorithm for genome-wide prediction of protein function. *Nature*, *402*, 83–86.

Mostafavi, S., Ray, D., Farley, D. W., Grouios, C., & Morris, Q. (2008). Genemania: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, *9*, S4+.

Murali, T. M., Wu, C.-J., & Kasif, S. (2006). The art of gene function prediction. *Nature Biotechnology*, *24*, 1474–1475.

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

Oliver, S. (2000). Guilt-by-association goes global. *Nature*, *403*, 601–603.

Pavlidis, P., Cai, J., Weston, J., & Noble, W. S. (2002). Learning gene functional classifications from multiple data types. *Journal of Computational Biology*, *9*, 401–411.

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

Re, M., Mesiti, M., & Valentini, G. (2012). A fast ranking algorithm for predicting gene functions in biomolecular networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*, 1812–1818.

Re, M., & Valentini, G. (2012). Cancer module genes ranking using kernelized score functions. *BMC Bioinformatics*, *13*, S14.

Rojas, R. (1996). *Neural Networks - A Systematic Introduction*. Berlin: Springer-Verlag.

Ruepp, A. et al. (2004). The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, *32*, 5539–5545.

Sharan, R., Ulitsky, I., & Shamir, R. (2007). Network-based prediction of protein function. *Molecular Systems Biology*, *3:88*.

Smola, A., & Kondor, I. (2003). Kernel and regularization on graphs. In B. Scholkopf, & M. Warmuth (Eds.), *Proc. of the Annual Conf. on Computational Learning Theory* Lecture Notes in Computer Science (pp. 144–158). Springer.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *J. Mach. Learn. Res.*, *7*, 1531–1565.

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

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

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 945–952). MIT Press volume 14.

Tsirukis, A. G., Reklaitis, G. V., & Tenorio, M. F. (1989). Nonlinear optimization using generalized hopfield networks. *Neural Comput.*, *1*, 511–521.

Tsuda, K., Shin, H., & Scholkopf, B. (2005). Fast protein classification with multiple networks. *Bioinformatics*, *21*, ii59–ii65.

Valentini, G. (2011). True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *8*, 832–847.

Vazquez, A., Flammini, A., Maritan, A., & Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks. *Nature Biotechnology*, *21*, 697–700.

Von Mering, C. et al. (2002). Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, *417*, 399–403.

Wang, D. (2003). Temporal pattern processing. In *The Handbook of Brain Theory and Neural Networks* (pp. 1163–1167).

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, *1*, 80–83.

Wuchty, S., Ravasz, E., & Barabsi, A. L. (2003). The architecture of biological networks. *Complex Systems in Biomedicine*, *5259*, 165–181.

Zhang, F., & Zhang, H. (2005). Applications of a neural network to watermarking capacity of digital image. *Neurocomputing*, *67*, 345–349.

Zhou, D. et al. (2004). Learning with local and global consistency. In *Adv. Neural Inf. Process. Syst.* (pp. 321–328). volume 16.

Zhu, W., Hou, J., & Chen, Y.-P. P. (2010). Semantic and layered protein function prediction from ppi networks. *J Theor Biol*, *267*, 129–36.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *In ICML* (pp. 912–919).

Appendix A. Convergence proof

The following fact adapts the convergence proof for Hopfield networks with neuron activation values $\{1, -1(0)\}$ (Rojas, 1996; Hopfield, 1982) to the more general case with activation values $\{\sin\alpha, -\cos\alpha\}$.

Fact 4. *A parametric Hopfield network* $H = < \boldsymbol{W}, \boldsymbol{\gamma}, \alpha >$ *with neurons* $V = \{1, 2, \ldots, n\}$ *and asynchronous dynamics, which starts from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

Proof. First of all, we observe that the energy (2) is equivalent to the following

$$E(\boldsymbol{x}) = -\frac{1}{2}\sum_{i,j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n} x_i\gamma_i \tag{A.1}$$

During the iteration *t+1* a random unit k is selected and updated according to the update rule (1). To simplify the proof we set $x'_k = x_k(t+1)$ and $x_k = x_k(t)$. If the unit k does not change its state, then the energy of the system does not change either. Otherwise, the network reaches a new global state $\boldsymbol{x}' = (x_1, \ \ldots, \ x'_k, \ldots, \ x_n)$ for which the new energy is $\mathrm{E}(\boldsymbol{x}')$. Since by definition $w_{ii} = 0$, the difference between $\mathrm{E}(\boldsymbol{x}')$ and $\mathrm{E}(\boldsymbol{x})$ is given by all terms in the summation (A.1) which contain x'_k and x_k, that is

$$E(\boldsymbol{x}') - E(\boldsymbol{x}) = -(x'_k - x_k)(\sum_{j=1}^{n} w_{kj}x_j - \gamma_k) = -(x'_k - x_k)B_k \tag{A.2}$$

where $B_k = \sum_{j=1}^{n} w_{kj}x_j - \gamma_k$. The factor $\frac{1}{2}$ disappears from the computation because the terms $w_{kj}x_k x_j$ appear twice in the double sum of (A.1). If $B_k > 0$, it means that $x'_k = \sin\alpha$ and $x_k = -\cos\alpha$; since $0 \leq \alpha \leq \frac{\pi}{2}$, $(x'_k - x_k) > 0$ and $E(\boldsymbol{x}') - E(\boldsymbol{x}) < 0$. If $B_k < 0$, it means that $x'_k = -\cos\alpha$ and $x_k = \sin\alpha$ and again $E(\boldsymbol{x}') - E(\boldsymbol{x}) < 0$. If $B_k = 0$, the energy value does not change after the update. Hence the energy (A.1) is a monotonically decreasing function. Moreover, since the connections w_{ij} are non negative, the energy E is lower bounded by the value $-\sin\alpha(\sum_{i=1}^{n}(\sum_{j=1}^{n}\sin\alpha \cdot w_{ij} - \gamma_i)$, and the dynamics is guaranteed to converge to a fixed point, corresponding to a local minimum of the energy. □