Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Learning Node Labels with Multi-Category Hopfield Networks

Marco Frasca · Simone Bassis · Giorgio Valentini

Received: date / Accepted: date

Abstract In several real-world node-label prediction problems on graphs, in fields ranging from computational biology to World-Wide-Web analysis, nodes can be partitioned into categories different from the classes to be predicted, on the basis of their characteristics or their common properties. Such partitions may provide further information about node classification that classical machine learning algorithms do not take into account. We introduce a novel family of parametric Hopfield networks (*m-Category Hopfield Networks*) and a novel algorithm (*Hopfield Multi-Category – HoMCat*), designed to appropriately exploit the presence of property-based partitions of nodes into multiple categories. Moreover, the proposed model adopts a cost-sensitive learning strategy to prevent the remarkable decay in performance usually observed when instance labels are unbalanced, that is when one class of labels is highly under-represented than the other one. We validate the proposed model on both synthetic and real-world data, in the context of multi-species function prediction, where the classes to be predicted are the Gene Ontology terms and the categories the different species in the multi-species protein network. We carried out an intensive experimental validation, which on the one hand compares *HoMCat* with several state-of-the-art graph-based algorithms, and on the other hand reveals that exploiting meaningful prior partitions of input data can substantially improve classification performances.

Keywords Multi-Category Hopfield Network · Binary Classification · Unbalanced Graphs · Protein Function Prediction · Biological Networks

Marco Frasca · Simone Bassis · Giorgio Valentini
Dept. of Computer Science, University of Milano
Via Comelico 39/41, 20135, Milano, Italy
E-mail: {frasca,bassis,valentini}@di.unimi.it

1 Introduction

In many problems where instances can be directly associated according to different levels of semantic relationship, and where vector-based representations can be very expensive to use, graph-based learning is an effective and well-exploited alternative to traditional feature-based learning [6]. In graph-based representations, instances are nodes and their pairwise relationships the edges connecting them. When classifying web pages, for example, edge weights may incorporate information about hyperlinks; or when predicting a user location starting from a few known positions obtained from users who allow geo-tagging their tweets, edge weights describe the follower relationships in the Twitter follower network [13].

In this context, several problems can be modeled as supervised or semi-supervised label prediction problems in partially labeled graphs, where node labels are known only for a subset of instances; the aim is to predict the labels of unlabeled instances by exploiting both the known labels and the topology of the network [6,51]. Many methods have been proposed through the years to learn node labels in graphs. Early approaches exploited the so called *guilt-by-association* (GBA) rule, which makes predictions based on the majority or weighted majority of labels in the direct neighborhood, assuming that interacting nodes are likely to share similar properties [38,57]. Analogously, *k*-nearest neighborhood (*kNN*) methods consider only the labels of the *k* most similar neighbors [32]; in turn, shared similarity metrics, as those proposed in [28,18], can be introduced to generalize the notion of pairwise-similarity among nodes by taking into account the contribution of shared neighbors [14, 9]. Other methodologies predict labels by propagating node labels to neighbors with an iterative process until convergence [70,69], or by evaluating the functional flows through the nodes of the graph [62,49]. Random Walks (*RW*) have also been applied to tune the probability to reach a given node through a probabilistic path starting from positive instances [59,4,30]. Other relevant studies used techniques based on Markov [15,11] and Gaussian Random Fields [60, 44,43], Support Vector Machines [66], kernelized score functions [54,55], communities [42] and co-citations [8]. Hopfield networks (HNs) have also been adopted in this context for their efficient local optimization capabilities [29,7,21].

Despite their proven effectiveness, these methods neglect the existence of “prior partitions” of input instances into categories (apart from the class being predicted), which is naturally present in several real-world problems. For instance, when analyzing the impulse buying for a given population of consumers, impulse purchases depend on several factors, including the consumers’ emotional and cognitive reactions, and the shopping/marketing environment. In this context, individuals can be naturally partitioned into categories according to their demographics and socio-cultural characteristics, which are a central factor influencing the impulse buying [48]. Analogously, in personalized medicine, patient profiles naturally induce categorical partitions of patients useful for determining the prognosis and/or the diagnosis of patients. For instance, the prediction of in-hospital mortality of Intensive Care Unit (ICU) patients depends on several features, such as age, gender, height, and several time series variables for which multiple observations could be available (e.g. cholesterol, glucose, blood pressure). When predicting which patients survived their hospitalizations, a key information is the categorization of patients according to their ICUType (i.e. Coronary Care Unit, Cardiac Surgery Recovery Unit, etc.), defined as the reason for admitting patients at recovery [58]. Moving to an ecological scenario, soil types represent significant categories in the context of the classification of forest cover types (for instance detecting a given bush under examination) on the basis of physical features describing the forest area (elevation, slope, etc.) [24].

Besides this problem, most of the mentioned machine learning methods tend to suffer a decay in performance when labelings in input data are unbalanced, e.g. when positive examples are significantly fewer than those negative, and imbalance-aware strategies are required [16,67,35,34]. Unfortunately, several graph-based prediction problems are characterized by strongly unbalanced labelings [13,53,21].

We propose *HoMCat* (Hopfield Multi-Category network), a novel semi-supervised method for predicting node labels in graphs, which is designed to explicitly take into account both the prior partition of input instances into categories and the imbalance of node labeling. *HoMCat* is a parametric Hopfield network with discrete binary neurons, where, unlike classical HNs, neurons are divided *a priori* in $m \geq 2$ disjoint groups, each one associated with a dedicated couple of activation values, that become parameters to be automatically learned. This allows the model to exploit the different topology of every category sub-graph, while preserving the global energy minimization property. The parameters are learned through a cost-sensitive strategy, which explicitly increases the penalties of positive misclassification, allowing the overall labeling imbalance of input data to be handled. In particular, it extends the cost-sensitive setting adopted in [21] to the multi-category context, and introduces a novel learning strategy which optimizes a criterion based on a “fuzzy-like” generalization of the classical notions of positive and negative memberships to the class being predicted. Moreover, we show that the time complexity of *HoMCat* grows only linearly with the number of categories, allowing its application in contexts where several categories can be detected.

Exhaustive experimental procedures have been carried out to validate the effectiveness of the learning procedure, the predictive capabilities of the model, and its robustness to noise. *HoMCat* has been compared with several supervised and semi-supervised state-of-the-art methods in the multi-species prediction of protein functions, involving many eukaryotic and prokaryotic model organisms. The results suggest that properly exploiting the partition of the input instances may considerably improve the accuracy and the precision of label prediction methodologies.

The paper is organized as follows: in Sec. 2 we formalize the *Label Prediction in partially labeled Graphs* (*LPG*) problem; Sec. 3 is devoted to the description and properties of the Multi-Category Hopfield Network modeling the *LPG* problems. *HoMCat* and its learning properties are introduced in Sec. 4, while in Sec. 5 we analyze the effectiveness of the proposed method on both synthetic data and real-world computational biology problems. Concluding remarks end the paper, indicating also some perspectives for future research work.

2 Label prediction in partially labeled graphs characterized by multiple categories

Let $G = (V, \boldsymbol{W})$ be a weighted graph, where $V = \{1, 2, \ldots, n\}$ is the set of vertices, $\boldsymbol{W}$ is the $n \times n$ weight matrix, and $W_{ij} \in [0, 1]$ represents a pre-computed similarity between instances i and j. Nodes in V are divided into $m \geq 2$ disjoint categories $V_1, V_2, \ldots, V_m$, with $\bigcup_{k=1}^{m} V_k = V$ and $V_i \cap V_j = \emptyset$, for $i, j = 1, \ldots, m$ and $i \neq j$.

For a given class to be predicted, the vertices in V are labeled with $\{+, -\}$, but the labeling is known only for the subset $S \subset V$, whereas it is unknown for $U = V \setminus S$. Moreover, labeled vertices are partitioned into positive S_+ and negative S_- vertices. Without loss of generality, we consider graphs with prevalence of negative vertices, since we can get back to this condition by simply flipping node labels, when positive vertices are the majority. The input labeling is considered *unbalanced* when $\varepsilon = |S_+|/|S_-| \ll 1$.

The *Label Prediction in partially labeled Graphs* (*LPG*) consists in determining a bipartition (U_+, U_-) of vertices in U. Vertices in U_+ are then considered candidates for the positive class. In principle, different solutions can be detected for an instance of the *LPG* problem, as well as several criteria may be chosen to assert the quality of each solution. In the following we will consider the minimization of the energy function of Hopfield networks as criterion for evaluating the quality of solutions.

3 Hopfield networks with neurons partitioned into multiple categories

In this section we introduce a class of Hopfield networks with binary neurons designed to exploit different types of instances (i.e. categories) in the input graph. We assume that a partition of input instances in $m \geq 2$ disjoint categories $V_1, V_2, \ldots, V_m$, is known *a priori* (see Sec. 2), owing to specific instance properties/characteristics. We further remark that these categories are distinguished from the classes to be predicted: within each category we may have nodes belonging to different classes (see Fig. 1).

Classical HNs do not consider any prior partition of neurons, which are characterized by two activation levels for every neuron: -1 (or 0) for negative neurons and $+1$ for positive ones. Several approaches have been proposed in the literature to extend the HN model, ranging from associative memories [71,56,37], to optimization problems [17,40,12] and system identification tasks [2]. The generalized HN models focused mainly on the output functions of neurons, which have been extended to have multiple inflection points. In other words, they adopt neurons which can assume N different values, with $N > 2$ [71]. Other approaches extended

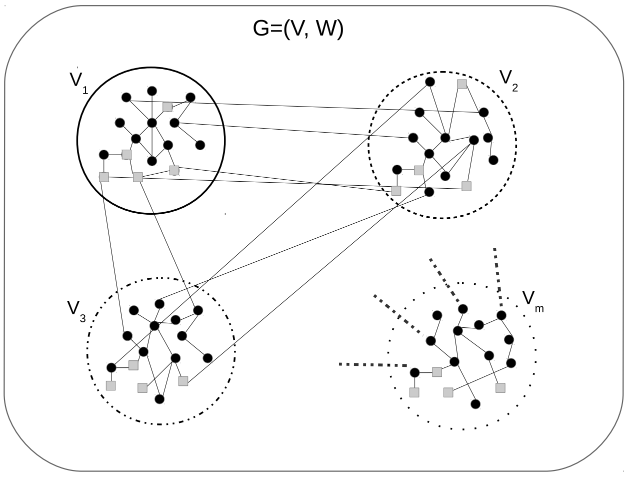

Fig. 1 Partition of the input graph $G = (V, \boldsymbol{W})$ into node categories $V_1, V_2, \ldots, V_m$. Square and circle nodes denote respectively positive and negative instances to be predicted.

HNs by considering neuron outputs as parametric functions, which are learned according to the structure of the graph [7]. Nevertheless, in all these approaches no prior partitions of neurons have been investigated.

In the proposed model, we extend HNs in another direction: in order to better fit the underlying "categorical" partition of the graph, we maintain binary neurons, but we allow their output levels to vary with the category they belong to. In other words, every category has its own pair of binary outputs, which is assigned to all neurons in that category. More precisely, we introduce a Hopfield model in which every category V_k is associated with a dedicated pair of activation values $\{y_1^{(k)}, y_2^{(k)}\}$, with $y_1^{(k)} < y_2^{(k)}$, which in general are different from the activation levels $\{-1(0), +1\}$ of classical HNs. In order to reduce the number of parameters, and accordingly the complexity of the model, each category V_k is associated with a single parameter $\alpha_k \in (0, \pi/2)$, and the output levels for neurons in V_k are defined as $y_1^{(k)} = -\cos\alpha_k$, for negative neurons, and $y_2^{(k)} = \sin\alpha_k$ for positive ones. It is worth nothing that this approach also allows to improve the results when unbalanced data are considered, as shown for the "vanilla" parametric HN without categories [21].

Formally, a *m-Category Hopfield Network* $\mathscr{H}$ with neurons $V = \{1, 2, \ldots, n\}$ is a $(m+3)$-tuple $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda} \rangle$, where:

- $\boldsymbol{W} = (W_{ij})|_{i,j=1}^{n}$ is a $n \times n$ symmetric matrix with null diagonal, whose elements $W_{ij} \in [0,1]$ represent the connection strength between neurons i and j;
- $\boldsymbol{b} \in \{1, 2, \ldots, m\}^n$ is a vector partitioning neurons into m categories:
$$V_k = \{i \in V | b_i = k\} \text{ for each } k \in \{1, 2, \ldots, m\}$$
- $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ are real values in $(0, \pi/2)$ denoting the neuron activation values; more precisely, $\{-\cos\alpha_k, \sin\alpha_k\}$ are the activation values for neurons i such that $b_i = k$, for $k \in \{1, 2, \ldots, m\}$;
- $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a real vector representing neuron activation thresholds.

The dynamics of the network is the following:

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

$$x_i(t+1) = \begin{cases} \sin\alpha_{b_i} & \text{if } E_i(t+1) > 0 \\ -\cos\alpha_{b_i} & \text{if } E_i(t+1) \leq 0 \end{cases} \quad (1)$$

where $E_i(t+1) = \sum_{j=1}^{i-1} W_{ij} x_j(t+1) + \sum_{r=i+1}^{n} W_{ir} x_r(t) - \lambda_i$, and b_i is the i^{th} element of the vector $\boldsymbol{b}$, denoting the membership of i^{th} neuron to the category V_{b_i}.

The state of the network at time t is $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$. The main feature of a classical Hopfield network is that it admits a Lyapunov function of the dynamics. In particular, let us consider the following quadratic state function (*energy function*):

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

Analogously to classical HNs, a multi-category Hopfield network converges towards a local minimum of the energy function E, as stated by the following fact:

Fact 1. *A m-Category Hopfield Network $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}\rangle$ with neurons $V = \{1,2,\ldots,n\}$ and asynchronous dynamics (1), starting from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

The proof is provided in Appendix A.

In the final part of the section, we extend the *Sub-network property*, holding for sub-networks of a Hopfield network [21], to the proposed multi-category Hopfield networks. This property allows to focus learning and inference on a specific sub-network of the global HN, while preserving the overall convergence properties of the algorithm.

Given a *m-Category Hopfield Network* $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}\rangle$, we recall that, according to the functional class to be predicted, the set of neurons V is partitioned in the subsets U and S of unlabeled and labeled neurons, as described in Sec. 2. Every state $\boldsymbol{x}$ of the network is thereby composed of the sub-vectors $\boldsymbol{x}_u$ and $\boldsymbol{x}_s$ representing states of neurons in U and S, respectively, that is $\boldsymbol{x} = (\boldsymbol{x}_u, \boldsymbol{x}_s)$. We can distinguish the contributions of $\boldsymbol{x}_u$ and $\boldsymbol{x}_s$ to the energy function (2):

$$\begin{aligned} E(\boldsymbol{x}_u, \boldsymbol{x}_s) = & -\frac{1}{2}\left(\boldsymbol{x}_u^T\boldsymbol{W}_{uu}\boldsymbol{x}_u + \boldsymbol{x}_s^T\boldsymbol{W}_{ss}\boldsymbol{x}_s + \boldsymbol{x}_u^T\boldsymbol{W}_{us}\boldsymbol{x}_s \right. \\ & \left. + \boldsymbol{x}_s^T\boldsymbol{W}_{us}^T\boldsymbol{x}_u\right) + \boldsymbol{x}_u^T\boldsymbol{\lambda}_u + \boldsymbol{x}_s^T\boldsymbol{\lambda}_s \\ = & -\frac{1}{2}\boldsymbol{x}_u^T\boldsymbol{W}_{uu}\boldsymbol{x}_u + \boldsymbol{x}_u^T(\boldsymbol{\lambda}_u - \boldsymbol{W}_{us}\boldsymbol{x}_s) + D \end{aligned} \tag{3}$$

where $D = -\frac{1}{2}\boldsymbol{x}_s^T\boldsymbol{W}_{ss}\boldsymbol{x}_s + \boldsymbol{x}_s^T\boldsymbol{\lambda}_s$ is a constant term with respect to $\boldsymbol{x}_u$, $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{uu} & \boldsymbol{W}_{us} \\ \boldsymbol{W}_{us}^T & \boldsymbol{W}_{ss} \end{pmatrix}$, and $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_u, \boldsymbol{\lambda}_s)$.

When a state $\bar{\boldsymbol{x}}_s$ of neurons in S is given, we denote with $\mathscr{H}_{U|\bar{\boldsymbol{x}}_s}$ the Hopfield network restricted to neurons in U, whose dynamics updates only such neurons and clamps to $\bar{\boldsymbol{x}}_s$ the state of neurons in S. From the energy decomposition (3) and the definition of Hopfield network, it is straightforward to prove the following fact:

Fact 2. $\mathscr{H}_{U|\bar{\boldsymbol{x}}_s} = \langle \boldsymbol{W}_{uu}, \boldsymbol{b}_u, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}_u - \boldsymbol{W}_{us}\bar{\boldsymbol{x}}_s\rangle$.

Here $\boldsymbol{b}_u$ is the sub-vector of $\boldsymbol{b}$ determining the partitions of U into the given m categories. Moreover, we denote with $E_{|\bar{\boldsymbol{x}}_s}(\boldsymbol{x}_u) = -\frac{1}{2}\boldsymbol{x}_u^T\boldsymbol{W}_{uu}\boldsymbol{x}_u + \boldsymbol{x}_u^T(\boldsymbol{\lambda}_u - \boldsymbol{W}_{us}\bar{\boldsymbol{x}}_s)$ the energy function of $\mathscr{H}_{U|\bar{\boldsymbol{x}}_s}$.

The state $\bar{\boldsymbol{x}}_s$ is *part of a global minimum* of the energy E if there is a state $\boldsymbol{x}_u$ of neurons in U such that $(\boldsymbol{x}_u, \bar{\boldsymbol{x}}_s)$ is a global minimum of E. The following property holds:

Fact 3. (Sub-network property) *If $\bar{\boldsymbol{x}}_s$ is part of an energy global minimum of $\mathscr{H}$, and $\bar{\boldsymbol{x}}_u$ is a global minimum of the energy $E_{|\bar{\boldsymbol{x}}_s}(\boldsymbol{x}_u)$, then $(\bar{\boldsymbol{x}}_u, \bar{\boldsymbol{x}}_s)$ is a energy global minimum of $\mathscr{H}$.*

Proof. From (3) it follows that

$$E(\boldsymbol{x}_u, \boldsymbol{x}_s) = -\frac{1}{2}\boldsymbol{x}_s^T\boldsymbol{W}_{ss}\boldsymbol{x}_s + \boldsymbol{x}_s^T\boldsymbol{\lambda}_s + E_{|\boldsymbol{x}_s}(\boldsymbol{x}_u) \tag{4}$$

By hypothesis, $\bar{\boldsymbol{x}}_u$ is a global minimum of $E_{|\bar{\boldsymbol{x}}_s}(\boldsymbol{x}_u)$ and, by assuming $(\bar{\boldsymbol{x}}_u, \bar{\boldsymbol{x}}_s)$ is not an energy global minimum of $\mathscr{H}$, there exists a state $\hat{\boldsymbol{x}}_u$ of neurons in U such that $E(\hat{\boldsymbol{x}}_u, \bar{\boldsymbol{x}}_s) < E(\bar{\boldsymbol{x}}_u, \bar{\boldsymbol{x}}_s)$. Then by (4), we have that $E_{|\bar{\boldsymbol{x}}_s}(\hat{\boldsymbol{x}}_u) < E_{|\bar{\boldsymbol{x}}_s}(\bar{\boldsymbol{x}}_u)$, which contradicts the hypothesis that $\bar{\boldsymbol{x}}_u$ is a global minimum of $E_{|\bar{\boldsymbol{x}}_s}(\boldsymbol{x}_u)$. $\square$

In our setting, we associate the given bipartition (S_+, S_-) of S with the sub-vector $\tilde{\boldsymbol{x}}_s$ of the state vector $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}_s, \tilde{\boldsymbol{x}}_u)$, where for each $i \in S$:

$$\tilde{x}_i = \begin{cases} \sin\alpha_{b_i} & \text{if } i \in S_+ \\ -\cos\alpha_{b_i} & \text{if } i \in S_- \end{cases} \tag{5}$$

When for suitable $\boldsymbol{b}, \alpha_1, \alpha_2, \ldots, \alpha_m$ the state $\tilde{\boldsymbol{x}}_s$ is part of a energy global minimum of $\mathscr{H}$, by the sub-network property we may discover the hidden part relative to neurons in U by minimizing the energy of $\mathscr{H}_{U|\tilde{\boldsymbol{x}}_s}$.

4 An algorithm for node label prediction with multi-category Hopfield networks

In this section we describe *HoMCat* (**Ho**pfield **M**ulti-**Cat**egory network) a novel method for dealing with the *LPG* problem based on *m-Category Hopfield networks*. In the following, we first provide a sketch of *HoMCat*, then we describe in detail each step of the algorithm.

Given the similarity matrix $\boldsymbol{W}$ and the vector $\boldsymbol{b}$, which determines the partition of neurons $V = \{1,2,\ldots,n\}$ into m non empty categories $V_1, V_2, \ldots, V_m$, we consider the class of *m-Category Hopfield networks* $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}\rangle$ where $\boldsymbol{\lambda} = \lambda \cdot \mathbf{e}$ and $\mathbf{e} = (1,1,\ldots,1)$, that is all neurons have the same real activation threshold λ.

An instance of the *LPG* problem is given by the matrix $\boldsymbol{W}$ and the sets S_+ and S_- of positive and negative examples. We hypothesize that there exists a $(m+1)$-tuple $(\alpha_1, \ldots, \alpha_m, \lambda)$ such that:

- the solution of the problem corresponds to an energy global minimum of $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}\rangle$;

- $\tilde{\boldsymbol{x}}_s$ is part of an energy global minimum $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}_u, \tilde{\boldsymbol{x}}_s)$ of $\mathscr{H}$.

Then, by Fact 3, we can discover the hidden part $\tilde{\boldsymbol{x}}_u$ of $\tilde{\boldsymbol{x}}$ by minimizing the energy of the sub-network $\mathscr{H}_{U|\tilde{\boldsymbol{x}}_s}$, provided that the reached equilibrium state is a global minimum of $E_{|\tilde{\boldsymbol{x}}_s}(\boldsymbol{x}_u)$. Our aim is to exploit this property and, at the same time, preserve the scalability of the algorithm on large-sized data. From this standpoint, a Hopfield network represents a suitable choice, since it can efficiently determine a minimum of the energy, although it does not ensure the achievement of a global minimum. Moreover, the choice of a local optimizer is motivated by previous investigations about the effectiveness of global methodologies (e.g. the *min-cut* algorithm [21]) in the context of the automated inference of protein functions: global optimizers have not significantly improved the performance with regard to local methods, while increasing the time complexity [21,47].

The procedure for solving the *LPG* problem is thereby factorized into the following main steps:

Step A: learning the network parameters. Estimate the parameters $(\alpha_1, \ldots, \alpha_m, \lambda)$ such that the state $\tilde{\boldsymbol{x}}_s$ is part of a minimum of the energy E, or as close as possible to a part of a global minimum. This approximation is efficiently performed by learning the parameters for which the state $\tilde{\boldsymbol{x}}_s$ is a fixed point (or close to a fixed point) of the sub-network of labeled neurons.

Step B: discovering the unknown labels through the sub-network dynamics. After choosing a suitable initial state, run the sub-network $\mathscr{H}_{U|\tilde{\boldsymbol{x}}_s}$ with the estimated parameters and reach a minimum $\tilde{\boldsymbol{x}}_u$ of the energy.

Finally, the solution (U_+, U_-) of *LPG* is:

$$U_+ = \{i \mid \tilde{x}_{u,i} > 0\}$$
$$U_- = \{i \mid \tilde{x}_{u,i} \leq 0\}$$

In the following we discuss in more details *Step A* (Sec. 4.1) and *Step B* (Sec. 4.2) of the algorithm.

4.1 Learning the network parameters

The main goal of this step is to find the values of the parameters $(\alpha_1, \ldots, \alpha_m, \lambda)$ such that the state $\tilde{\boldsymbol{x}}_s$ is "close" to an equilibrium state of the sub-network restricted to nodes in S. In this section we describe the objective function adopted in the learning step and the relative optimization procedure.

Objective function. When the network parameters $\boldsymbol{p} = (\alpha_1, \ldots, \alpha_m, \lambda) = (p_1, p_2, \ldots, p_{m+1})$ are fixed, every neuron $i \in S$ has an *internal energy* $A_i(\boldsymbol{p})$ defined as follows:

$$A_i(\boldsymbol{p}) = \sum_{k \in S} (\sin p_{b_k} W_{ik} \chi_k - \cos p_{b_k} W_{ik}(1 - \chi_k)) - p_{m+1} \quad (6)$$

where $\boldsymbol{\chi}$ is the characteristic vector of S_+ (i.e. $\chi_k = 1$ if $k \in S_+$, $\chi_k = 0$ otherwise). Note that $A_i(\boldsymbol{p})$ represents how much a node i is linked with positive nodes with respect to negative ones, taking also into account the category b_k of the neighboring nodes. The choice of the *sine* and *cosine* functions ensures on the one hand that the neuron activation values are limited, and on the other hand that, to counterbalance the large presence of negatives when the label imbalance in category V_k becomes higher, the activation value for positive neurons increases with α_k, due to the behaviour of the *sine* function in the range $(0, Pi/2)$, whereas the activation value for negative neurons decreases, considering the opposite trend of the *cosine* function on the same interval. Although effective, this choice is not the only possible, and in principle any other couple of functions having a similar behaviour may be used in lieu of *sine* and *cosine*.

By means of $A_i(\boldsymbol{p})$, we can define the following set of membership functions that provide an extension of the notion of *true positive* $TP = \{i \in S_+ | A_i(\boldsymbol{p}) > 0\}$, *false negative* $FN = \{i \in S_+ | A_i(\boldsymbol{p}) \leq 0\}$, and *false positive* $FP = \{i \in S_- | A_i(\boldsymbol{p}) > 0\}$:

$$\begin{aligned}
\mu_{TP}(i, \boldsymbol{p}) &= \tfrac{1}{2}\left(\tfrac{2}{\pi}\operatorname{arctg}(\tau A_i(\boldsymbol{p})) + 1\right), \quad i \in S_+ \\
\mu_{FN}(i, \boldsymbol{p}) &= \tfrac{1}{2}\left(1 - \tfrac{2}{\pi}\operatorname{arctg}(\tau A_i(\boldsymbol{p}))\right), \quad i \in S_+ \\
\mu_{FP}(i, \boldsymbol{p}) &= \tfrac{1}{2}\left(\tfrac{2}{\pi}\operatorname{arctg}(\tau A_i(\boldsymbol{p})) + 1\right), \quad i \in S_-
\end{aligned} \quad (7)$$

where τ is a positive real parameter. Indeed, it is easy to see that μ_{TP}, μ_{FN} and μ_{FP} may assume values in the continuous unitary range $[0, 1]$: for instance, if $i \in S_+$ and $\tau A_i(\boldsymbol{p}) = 0$, we consistently obtain $\mu_{TP}(i, \boldsymbol{p}) = \mu_{FN}(i, \boldsymbol{p}) = 0.5$, while if $i \in S_+$ and $\tau A_i(\boldsymbol{p}) \to \infty$, we have that $\mu_{TP}(i, \boldsymbol{p}) = 1$ and $\mu_{FN}(i, \boldsymbol{p}) = 0$; in the intermediate cases we will have $0 < \mu_{TP}(i, \boldsymbol{p}) < 1$ and $0 < \mu_{FN}(i, \boldsymbol{p}) < 1$.

Definition (7) provides an extension of the classical notion of crisp membership function; indeed, we obtain the traditional crisp membership values $\{0, 1\}$ when $\tau \to \infty$; whereas, when τ is close to 0, the membership values tend to 0.5. By means of this generalization, we can increase not only the learning capability of the model, but even its overall performance; indeed, in Sec. 5.2.4 we will show that too large values of τ decrease the predictive capability of the model.

In order to deal with data imbalance, we aim at learning the parameters $\boldsymbol{p}$ by maximizing the following cost-sensitive criterion, based on a *F-score* defined in terms of the above membership functions:

$$\textit{F-score}(\boldsymbol{p}) = \frac{2 \sum_{i \in S_+} \mu_{TP}(i, \boldsymbol{p})}{2 \sum_{i \in S_+} \mu_{TP}(i, \boldsymbol{p}) + \sum_{i \in S_-} \mu_{FP}(i, \boldsymbol{p}) + \sum_{i \in S_+} \mu_{FN}(i, \boldsymbol{p})} \quad (8)$$

By observing that $0 \leq \textit{F-score}(\boldsymbol{p}) \leq 1$, this criterion is justified by the following:

Fact 4. *If F-score$(\boldsymbol{p}) = 1$, then $\tilde{\boldsymbol{x}}_s$ is an equilibrium state of the sub-network $\mathscr{H}_S = \langle \boldsymbol{W}_{ss}, \boldsymbol{b}_s, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda}_s \rangle$.*

Proof. *F-score*$(\boldsymbol{p}) = 1$ means that:

$$\sum_{i \in S_-} \mu_{FP}(i, \boldsymbol{p}) = 0, \quad \sum_{i \in S_+} \mu_{FN}(i, \boldsymbol{p}) = 0 \tag{9}$$

since both μ_{FP} and μ_{FN} are non negative by definition. From (9) we have that $\mu_{FP}(i, \boldsymbol{p}) = 0$ for each $i \in S_-$ (*), and $\mu_{FN}(i, \boldsymbol{p}) = 0$ for each $i \in S_+$ (**). By definition, condition (*) implies that $A_i(\boldsymbol{p}) \leq 0$ (precisely $\tau A_i(\boldsymbol{p}) \to -\infty$) for each $i \in S_-$, and accordingly the update rule (1) does not change its state; condition (**) leads to $\tau A_i(\boldsymbol{p}) \to \infty$ and hence $A_i(\boldsymbol{p}) > 0$ for each $i \in S_+$, and again the update rule (1) does not change its state. Since $S = S_+ \cup S_-$, the sub-network $\mathscr{H}_S$ is at equilibrium.

□

Optimization procedure. We aim at determining the values $\hat{\boldsymbol{p}}$ of parameters $\boldsymbol{p} = (p_1, p_2, \ldots, p_{m+1})$ that maximize the *F-score* criterion, that is:

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{p}}{\operatorname{argmax}}\ \textit{F-score}(\boldsymbol{p})$$

The idea is to adopt an iterative and incremental procedure aimed at improving the estimation of the single parameters until a suitable criterion is met (e.g. convergence, number of iterations, etc.). More precisely, for a fixed assignment of parameters $(p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_{m+1})$, we estimate $\hat{p}_i$ with the value $\overline{p}_i = \operatorname{argmax}_{p_i}$ *F-score*$(\boldsymbol{p})$, $i \in \{1, \ldots, m+1\}$. The learning procedure can be summarized as follows:

1. Random permute the vector $\boldsymbol{p}$ to define a learning order, and fix an initial random assignment of parameters $(p_1, \ldots, p_{m+1})$;
2. Determine an estimate $\overline{p}_i$ of $\hat{p}_i$ with a standard line search procedure for optimizing continuous functions of one variable, and fix $p_i = \overline{p}_i$;
3. Iterate Step 2 for each $i \in \{1, 2, \ldots, m+1\}$;
4. Repeat Step 3 till a stopping criterion is satisfied.

Such approach, named *simplest search method*, has already been successfully applied to neural network training [31]. At Step 2 we adopt the Brent algorithm, which combines golden section search and successive parabolic interpolation to determine a sequence of gradually more accurate estimations of a local optimum of a continuous function in a specified interval [10]. The algorithm can be stopped when the difference of the estimates $\overline{\boldsymbol{p}}$ at two subsequent iterations falls below a given threshold, or when *F-score* improvements are lower than a fixed tolerance, or alternatively when a maximum number (chosen *a priori* or adaptively) of iterations of Step 2 is reached.

Fig. 2 summarizes the high-level pseudocode of the optimization procedure. The function `optimize` realizes the procedure adopted at Step 2. The time complexity of this function depends on the number of iterations to achieve convergence, which in turn depends on both the convergence rate of the procedure `optimize` and the desired tolerance. To speed-up the method, during the first iteration, we compute and fix the terms in the objective function (8) (*F-score*) constant with regard to the parameter to be learned. The cost of this operation is $O(|\boldsymbol{W}_{ss}|)$, where $|\boldsymbol{W}_{ss}|$ is the number of non-zero entries in the connection matrix $\boldsymbol{W}_{ss}$. Then, from the second iteration till convergence, the update of the *F-score* function can be done in $O(|S|)$ time, and accordingly the complexity of this procedure is $O(|\boldsymbol{W}_{ss}|)$ ($O(|S|)$ when $\boldsymbol{W}_{ss}$ is sparse). Finally, the overall complexity of the procedure `OptimizeParameters` is $O(|\boldsymbol{W}_{ss}| \cdot m)$, which increases just linearly with the number of categories.

Fig. 2 Pseudocode of the `OptimizeParameters procedure.`

```
Input:
- neuron set $V$, bipartition ($U$, $S$) of $V$
- bipartition ($S_+$, $S_-$) of $S$
- vector partitioning neurons into $m$ categories $\boldsymbol{b}$
- parameter vector $\boldsymbol{p}$
- connection matrix $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_{uu} & \boldsymbol{W}_{us} \\ \boldsymbol{W}_{us}^T & \boldsymbol{W}_{ss} \end{pmatrix}$

Function OptimizeParameters
begin algorithm
01:   Initialize($\boldsymbol{p}$)
02:   repeat
03:       for each $i \in \{1, 2, \ldots, m+1\}$ do
04:           $\overline{p}_i \leftarrow$ optimize($\boldsymbol{p}, i, \boldsymbol{b}, \boldsymbol{W}_{ss}$)
05:           $p_i \leftarrow \overline{p}_i$
06:       end for
07:   until stopping criterion is met
08:   $\overline{\boldsymbol{p}} \leftarrow \boldsymbol{p}$
end algorithm

Output: $\overline{\boldsymbol{p}}$
```

4.2 Discovering the unknown labels by sub-network dynamics

After the computation of the optimal parameters $\overline{\boldsymbol{p}} = (\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_{m+1})$, we consider the sub-network $\mathscr{H}_{U|\tilde{\boldsymbol{x}}_s} = \langle \boldsymbol{W}_{uu}, \boldsymbol{b}_u, \alpha_1 = \overline{p}_1, \ldots, \alpha_m = \overline{p}_m, \boldsymbol{\lambda}_u - \boldsymbol{W}_{us}\tilde{\boldsymbol{x}}_s \rangle$, where $\boldsymbol{\lambda}_u$ is a vector of size $|U|$ whose components are set to $\overline{p}_{m+1}$, and $\tilde{\boldsymbol{x}}_s$ is defined according to (5). Fixed the initial state $x_{u,i} = 0$ for each $i \in \{1, 2, \ldots, |U|\}$, we run the sub-network $\mathscr{H}_{U|\tilde{\boldsymbol{x}}_s}$ till convergence, to learn the unknown labels of neurons U (see Appendix A for the convergence proof). The state $x_{u,i}$ at time $t+1$ is updated by the following rule:

$$x_{u,i}(t+1) = \begin{cases} \sin \alpha_{b_{u,i}} & \text{if } E_{|\tilde{\boldsymbol{x}}_s,i}(t+1) > 0 \\ -\cos \alpha_{b_{u,i}} & \text{if } E_{|\tilde{\boldsymbol{x}}_s,i}(t+1) \leq 0 \end{cases}$$

where $E_{|\tilde{\boldsymbol{x}}_s,i}(t+1) = \sum_{j=1}^{i-1} W_{ij} x_{u,j}(t+1) + \sum_{r=i+1}^{|U|} W_{ir} x_{u,r}(t) - \theta_i$, $\boldsymbol{\theta} \equiv \boldsymbol{\lambda}_u - \boldsymbol{W}_{us}\tilde{\boldsymbol{x}}_s$ and $\boldsymbol{b}_u$ is the sub-vector of $\boldsymbol{b}$ corresponding to the instances in U.

If $\tilde{\boldsymbol{x}}_u$ is the stable state reached by this dynamics, we obtain the final solution (U_+, U_-) by setting $U_+ = \{i|\tilde{\boldsymbol{x}}_{u,i} > 0\}$ and $U_- = \{i|\tilde{\boldsymbol{x}}_{u,i} \leq 0\}$. The time complexity of this step is $O(|W_{uu}| \cdot h)$, where h is the number of iterations needed to achieve convergence (on average $h \simeq 2$ in our experiments).

Finally, the overall time complexity of *HoMCat* procedure is $O(|W_{ss}| \cdot m + |W_{uu}| \cdot h)$, which is $O(|S| \cdot m + |U| \cdot h)$ when the connection matrix W is sparse.

5 Results and Discussion

In this section we first study the behavior of *HoMCat* on synthetic data sets suitably generated to test its performance on easy to complex scenarios, then we compare its predictive performances on real-world data sets with state-of-the-art methodologies.

5.1 Experiments with synthetic data

To validate the proposed method and to test its effectiveness against perturbations as well, we performed a series of experiments based on synthetic data sets. In particular, we considered two main families of experiments: while the former is devoted to highlight the efficacy of the learning algorithm in terms of the *F-score* in both a non-perturbed and noise-perturbed scenario, the main focus of the latter is to check the stability of the fixed point reached during the Hopfield network evolution, when various perturbations are injected into the learned parameters.

5.1.1 Effectiveness of HoMCat in perturbed and non perturbed scenarios.

The aim of the experiments is twofold: on the one hand we want to check whether the proposed learning algorithm can achieve *F-score* approaching 1 in case of separable patterns, for a suitable selection of the number m of categories; on the other hand, we want to stress the advantages emerging by working with multiple categories, when weights and labels are affected by random perturbations. We run the optimization procedure of *HoMCat* (`OptimizeParameters`) on a set of instances generated so as to guarantee perfect separability when working with m categories (see Appendix B for the generative process), and subsequently evaluate its performance on a lower number $\widetilde{m} < m$ of them. We conduct several experiments with different initial conditions. The number of nodes is set to $n = 2000$ and the labeling imbalance to $\varepsilon = 0.01$ (Sec. 2). We fix the number m of categories to $\{4, 6, 8\}$, and randomly draw initial values of parameters α_k, for $k \in \{1, \ldots, m\}$. Then we run the procedure `OptimizeParameters` up to convergence. The *F-score* has been estimated through 5-fold cross-validation techniques.

Fig. 3(a) shows the smooth histogram of the *F-score*s obtained on 45 repetitions of the experiment with $m = 8$ and $\widetilde{m} \in \{4, 2, 1\}$. As expected, we observe that the learning procedure is able to achieve values of *F-score* near equal to 1 only when the nodes are subdivided into 8 categories, while the performance degrades as they are reduced in number, till becoming indistinguishable when $\widetilde{m} \leq 2$. In the meanwhile, the high number of parameters to be learned, coupled with an inadequate training set size, prevents the algorithm to achieve *F-score* exactly equal to 1.

Furthermore, we analyzed how different parameter initializations may affect the attained *F-score* values. In Table 1 we report the *mean* and the *standard deviation* of the *F-score* achieved across the 45 replicas with different random parameter initializations (for each value of m), and the *coefficient of variation*, i.e. the ratio of standard deviation to mean. We observe a small dispersion when choosing the optimal number $m = 8$ of categories, with standard deviation value dropping significantly down, compared to those obtained with other values of m, thus providing an indicator of the robustness of the approach against various parameter initializations. This is further accentuated by the coefficient of variation, which is a more suitable indicator of variability when the data to be compared have a different mean or order of magnitude.

Finally, it is worth noting that the proposed iterative learning algorithm outperformed global maximization techniques based on Nelder Mead, Differential Evolution, and Simulated Annealing [64] in a fraction of 3 out of 10 repetitions, while producing the same output on the remaining ones (data not shown).

To confirm the benefit of working with more categories, we checked the performance of the algorithm against random perturbations, by considering three different kinds of noise, while maintaining the same experimental setting used so far:

Table 1 Mean, standard deviation (std) and coefficient of variation (CVar) of *F-score* values achieved by the learning procedure on 45 repetitions with different parameter initializations.

Measure	m = 8	m = 4	m = 2	m = 1
Mean	0.9727	0.6408	0.5831	0.5717
Std	0.0243	0.0872	0.0918	0.0902
CVar	0.0249	0.1361	0.1572	0.1579

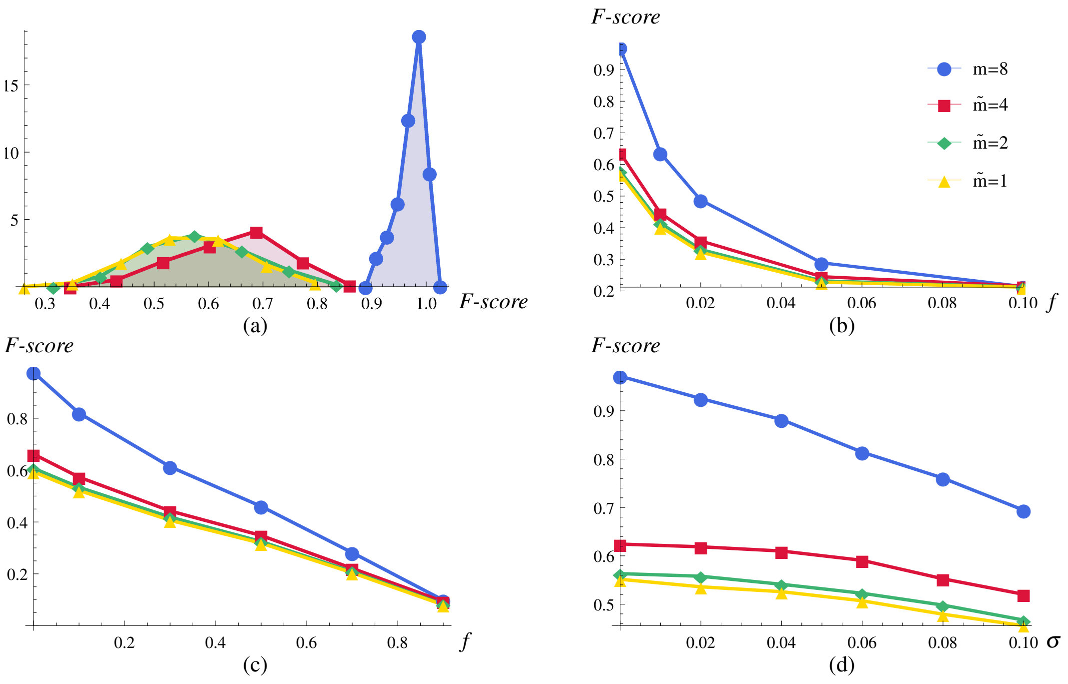

Fig. 3 Learning algorithm analysis. (a) Smooth Histogram of *F-score* values; (b-d) course of *F-score* values with regard to random perturbations generated with noise model N1, N2, and N3, respectively, for a different number of categories.

N1: *label switch.* We randomly select a fraction f of points and switch their labels, with $f \in \{0.01, 0.02, 0.05, 0.1\}$.

N2: *label swap.* We randomly select a fraction f of positive points and swap their labels with randomly selected negative points, with $f \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

N3: *weight noise.* We add white noise to the weights W_{ij} with a standard deviation $\sigma \in \{0.02, 0.04, 0.06, 0.08, 0.1\}$.

Figs. 3(b-d) show the course of *F-score*s as a function of the perturbation level for the three kinds of noise N1, N2 and N3, respectively. In all scenarios, the *F-score* ranking is preserved, with values of *F-score* increasing with the number of categories. The enhancement in *F-score* when $m = 8$ is more visible in Fig. 3(d), even though a similar trend is shown also in Figs. 3(b) and (c). Moreover, in all the experiments the improvements in *F-score* when larger number of categories is adopted is statistically significant (p-value $< 10^{-5}$, Wilcoxon signed-rank test [63]). Furthermore, since when injecting too much noise in a classification task we tend to predict noise, the values of *F-score* become nearly indistinguishable for high perturbation levels (i.e. when f approaches 0.1 in Fig. 3(b) and 1 in Fig. 3(c)), but their differences still remain statistically significant.

Summarizing, these experiments confirm that when the input space can be partitioned into meaningful categories, *m-Category Hopfield Networks* can properly fit the data, improving performances in both noisy and noise-free scenarios.

5.1.2 Fixed point stability analysis.

In this section we check whether the fixed point reached during the evolution of the Hopfield network can be substantially affected by moderate perturbations of the estimated parameters.

Given the set of nodes V ($|V| = 3000$), their bipartition into positive and negative instances (V_+, V_-), the partition of nodes into $m \in \{2, 4, 6, 8\}$ categories, and the vector of corresponding parameters $(\alpha_1, \ldots, \alpha_m, \lambda)$, we compute the initial labeling $\tilde{\boldsymbol{x}}$ by extending (5) to V. Having also fixed the labeling imbalance $\varepsilon = 0.01$ and the rate of labeled nodes $|S|/|V| \in \{0.3, 0.5, 0.7\}$, we generate a Hopfield network having $\tilde{\boldsymbol{x}}$ as fixed point together with other randomly generated patterns as allowed by its maximal capacity [26] (see Appendix B for details about the generation of the synthetic data). After hiding the labels of nodes in $U = V \setminus S$, we inject into the parameters a zero mean Gaussian noise with a standard deviation $\sigma \in \{0.01, 0.02, 0.04, 0.07, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and compute the corresponding *F-score* (8) on the labeled part S. We observe that the *F-score* value changes after each perturbation, since for each node $i \in S$ the corresponding internal energy A_i changes. Then, we let the perturbed network evolve to a fixed point $\boldsymbol{x}'_u$, clamping the state of neurons in S, and compute the perturbation in the reached fixed point as the normalized Hamming distance $d_{\tilde{H}}$ between $\boldsymbol{x}'_u$ and $\tilde{\boldsymbol{x}}_u$ (sub-vector of $\tilde{\boldsymbol{x}}$ having neurons in U), averaged on 200 repetitions of the dynamics. Given two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

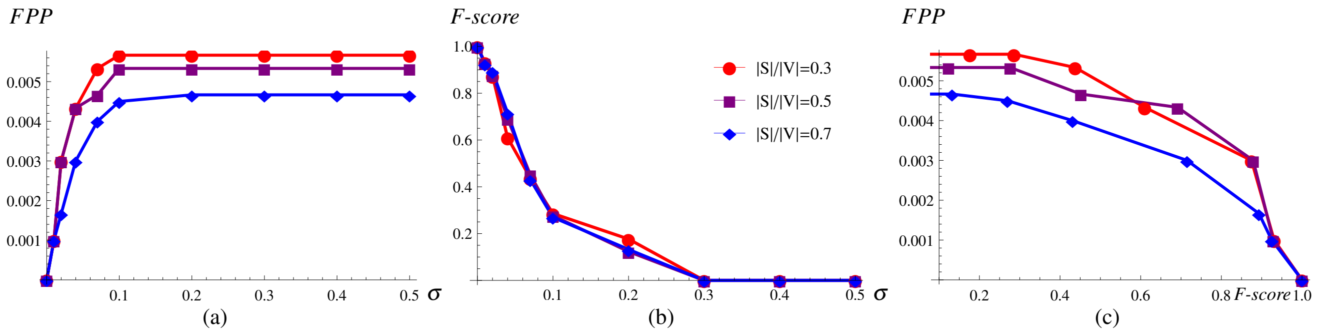

Fig. 4 Fixed point stability analysis. Course of: (a) fixed point perturbation (FPP), and (b) *F-score* vs. noise level σ; (c) fixed point perturbation with regard to *F-score*, for different $|S|/|V|$ ratios.

of length n, $d_{\tilde{H}}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is defined as the Hamming distance between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, divided by n.

Fig. 4 shows the results with $m = 8$ categories (similar results are obtained for $m = 2, 4, 6$). Interestingly, we can note that the perturbation of the fixed point and the corresponding *F-score* decay are negligible for reasonable noise levels ($\sigma < 0.04$), and for larger noise levels the perturbation of the fixed point increases till a normalized Hamming distance around 0.005, getting almost constant for $\sigma > 0.1$ (Fig. 4(a)). The *F-score* quickly decreases for $\sigma \geq 0.04$ (Fig. 4(b)), since injecting noise with such standard deviations considerably changes the initial value of the parameters. More remarkably, we can observe a slight, and almost constant, perturbation suffered by the fixed point in conjunction with very low values of *F-score* (Fig. 4(c)). Low *F-score* values are frequent in real-world *LPG* problems, where the lack of positive examples makes the learning tasks very difficult for any learning algorithm, and the observed property ensures in such cases robustness of the achieved fixed point to parameter noise. Finally, as expected, the model seems more robust to perturbations when the available prior information increases (larger ratio $|S|/|V|$), indeed the lowest fixed point perturbation is achieved when $|S|/|V| = 0.7$ for all the considered noise levels.

5.2 Experiments with real-world data

5.2.1 Experimental Setup.

We applied our algorithm to the Automated protein Function Prediction (*AFP*) problem, a challenging and central problem in computational biology [53,50]. In this setting, nodes represent proteins and connections their pairwise relationships deriving from different sources of information, including gene co-expression, genetic and physical interactions, protein ontologies and phenotype annotations. In particular, recent approaches focused on the multi-species *AFP* problem, relying on the availability of a collection of orthology relationships across inter-species proteins [65,41]. In this context, the sets of proteins belonging to different species constitute the different categories in the model, while the classes to be predicted are the biomolecular functions of the proteins. For instance, if we consider a network composed of the proteins of two species of rodents, *Mus musculus* (house mouse) and *Rattus norvegicus* (brown rat), mouse proteins form the first and rat proteins the second category of the model.

To assess the effectiveness of the proposed *m-Category Hopfield Network* in the context of multi-species function prediction problem, we considered at first two evolutionarily related species, i.e. *Danio rerio* (a fish) and *Xenopus laevis* (a frog) and then two (*Escherichia coli, Salmonella enterica*) and three (*Escherichia coli, Salmonella typhimurium, Bacillus subtilis*) species of Bacteria, thus obtaining models with two and three categories (Sec. 3). We predicted the functional classes (terms) of the Gene Ontology (GO) [1], the most used taxonomy for biomolecular functions, covering three ontologies: Biological Process (BP), Molecular Function (MF) and Cellular Component (CC). In order to validate our model on both balanced and unbalanced data, we considered two different experiments. In the first one, to predict relatively unbalanced classes, we selected BP terms with annotated proteins (belonging to the positive class) ranging from a minimum of 20 to a maximum of 200, obtaining a set of 272 GO terms for the fish and the frog, and respectively 281 and 289 GO terms for the two species and three species of bacteria. In the second experiment, on the same networks we selected the BP terms with more than 200 annotations, obtaining 35, 49 and 50 total terms for the three considered data sets.

5.2.2 Data.

At first, we collected data from the different databases reported in Table 2 to obtain different profiles for each protein. More precisely, each protein has been associated to a binary feature vector, representing a protein profile, whose

Table 2 Databases used to construct protein profiles and protein networks

Database	**Description**
PROSITE [27]	Protein domains and families
SMART [33]	Simple Modular Architecture Research Tool (database annotations)
PRINTS [3]	Motif fingerprints
InterPro [45]	Protein families, domains and functional sites
EggNOG [46]	Non-supervised Orthologous Groups
Pfam [19]	Protein domain
Swiss-Prot [5]	Manually curated keywords describing the function of the proteins
Protein super-families [23]	Structural and functional annotations

elements are 1 when the protein is annotated for a specific feature (e.g. includes a specific domain, or a specific motif), and 0 otherwise. Then the protein profiles have been used to construct a set of similarity networks (one for each data type) with edge scores based on the computation of the classical Jaccard similarity coefficient between each possible pair of protein profiles, thus obtaining $r = 8$ different protein networks.

The protein networks constructed according to the previously described steps have been integrated in a *consensus network* using the *Unweighted Average* (UA) network integration scheme (see, e.g. [61]): the weight of each edge is computed by averaging across the available r networks:

$$\overline{W}_{ij} = \frac{1}{r}\sum_{d=1}^{r} W_{ij}^{d} \tag{10}$$

where $\overline{W}_{ij}$ is the weight of the integrated network and W_{ij}^{d} represents the weight associated to the edge (i, j) of the d^{th} network.

The resulting integrated network for *Danio rerio* and *Xenopus laevis* (*Danxen* network) includes 6250 nodes (proteins) and about one million of edges, while the net with 2 species of bacteria (*Bacteria2* network) includes 5313 nodes and about 5 millions of edges, and the net with 3 species of bacteria (*Bacteria3*) includes 10319 nodes and about 8 millions of edges. Data are downloadable from the web repository `http://frasca.di.unimi.it/data/homcat`.

5.2.3 Performance metrics.

The generalization performances have been assessed through a stratified 5-fold cross validation procedure, and to properly take into account the unbalance that characterizes *AFP* problem, the evaluation of the performance has been measured in terms of Area Under the Precision-Recall Curve (AUPRC) and *F-score*, the harmonic mean of Precision and Recall. Fixed a GO term c, Precision and Recall are defined as:

$$\text{Precision} = \frac{TP}{NP_{+}}, \quad \text{Recall} = \frac{TP}{N_{+}}$$

where TP is the number of positive instances for c predicted as positive, NP_{+} the total number of instances predicted as positive, and N_{+} is the total number of positive instances for the term c. Moreover, we also adopted the classical *Area Under the ROC Curve* (AUC), pointing out that AUC by definition is a metric not well-suited for unbalanced problems [68].

We also note that *HoMCat* is essentially a classifier, that is it provides discrete predictions, and to compute the decision score for every neuron $i \in U$, necessary to compute AUC and AUPRC, we adopted the approach proposed in [20, 22], where a score is computed on the basis of the internal energy $E_{|\tilde{\mathbf{x}}_s,i}$ of the neuron i at the end of the dynamics (see Sec. 4.2).

5.2.4 Model parameter selection.

In this section we report the results of the experiments we conducted to evaluate the impact of parameter τ in equation (7) on the predictive capability of *HoMCat*. We compared the achieved average AUC, AUPRC and *F-score* on the data set *Bacteria2* by varying τ between 0.1 and 3 with steps of 0.1. The overall results are shown in Fig. 5. Interestingly, while AUC just slightly varies with τ, AUPRC and *F-score* values increase till $\tau \simeq 2$, then slightly decrease with similar behavior. Experiments not reported here show an analogous progressive decay for all the performance measures when $\tau > 3$, showing that the adopted membership functions (7) perform better than classical crisp membership (corresponding to $\tau \to \infty$). Similar results have been obtained also with the other considered data sets (*Bacteria3* and *Danxen* – data not shown). Summarizing, in our

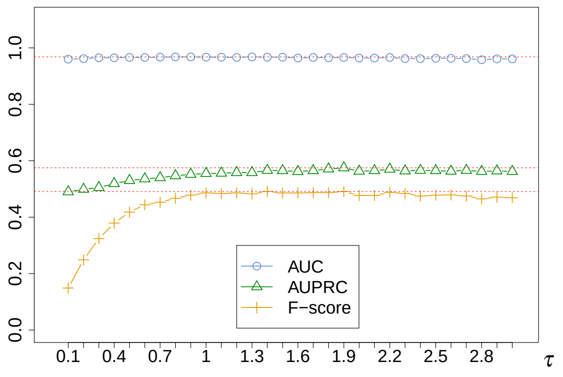

Fig. 5 Mean AUC, AUPRC and *F-score* achieved by *HoMCat* on *Bacteria2* data by varying the parameter τ. The horizontal dotted lines correspond to the maximum values achieved for each performance measure.

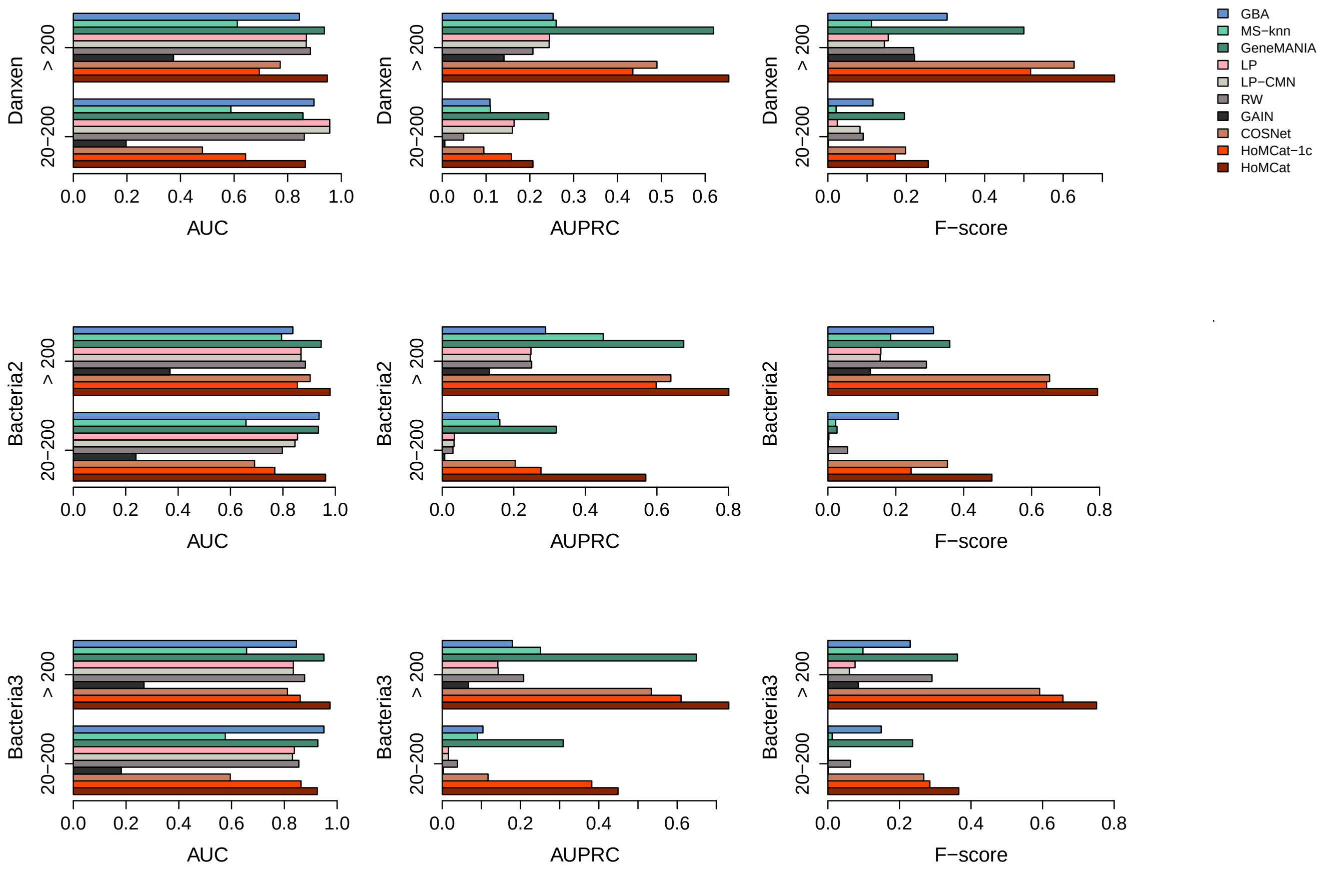

Fig. 6 Mean AUC, AUPRC and *F-score* achieved on *Danxen* and *Bacteria* data sets. $20-200$ corresponds to the setting with a number of positives between 20 and 200, whereas > 200 represents the more balanced setting, with more than 200 positives.

experiments we found that the optimal value of τ is in the interval $[0.9, 2.2]$. After tuning on a small subset of labeled data, in our experiments we fixed $\tau = 1, 2, 2$ respectively for *Danxen*, *Bacteria2* and *Bacteria3* data sets.

5.2.5 Comparison with state-of-the-art methods.

We compared our method with several state-of-the-art competitors: *GBA*, an algorithm based on the guilt-by-association principle [39]; *GeneMANIA* [44], the top method in the MouseFunc challenge [52]; *MS-kNN* [32], one of the top ranking algorithm in the recent CAFA challenge [53]; *LP*, a semi-supervised label propagation algorithm based on Gaussian random fields, and its class mass normalized version *LP-CMN* [70]; *RW*, the classical random walk algorithm without restart with at most 1000 random walk steps [36]; *COSNet*, a recently proposed algorithm for label prediction in graph, which is explicitly designed to cope with the imbalance in the instance labeling [21]. Furthermore, to assess the improvements introduced by partitioning proteins into categories, we also apply a different version of *HoMCat*, in which all the proteins are considered belonging to the same category (named *HoMCat-1c*). In Fig. 6 we show the obtained results averaged across the considered GO terms and in Fig. 7 the corresponding distribution for the dataset *Bacteria2*. In terms of average *F-score*, *HoMCat* significantly outperforms all the other methods (Wilcoxon signed-rank test, p-value $= 10^{-5}$), in both balanced and unbalanced settings. *COSNet* is almost always the second top method, whereas very poor results are achieved by GAIN and both LP methods. The proposed method achieves also the best AUPRC results (p-value $= 10^{-3}$) in all the experiments, except for Danxen 20-200 data, where it is the second top method. *HoMCat* obtains also the best AUC results when balanced setting are considered, and also on *Bacteria2* 20-200 data set. *GBA* and *LP* methods obtain the best AUC performance on unbalanced *Bacteria3* and *Danxen* data sets respectively.

It is worth noting that most of the compared algorithms, such as *LP*, *LP-CMN* and *RW*, are not inherently imbalance-aware, and this partially explains their poor performance in

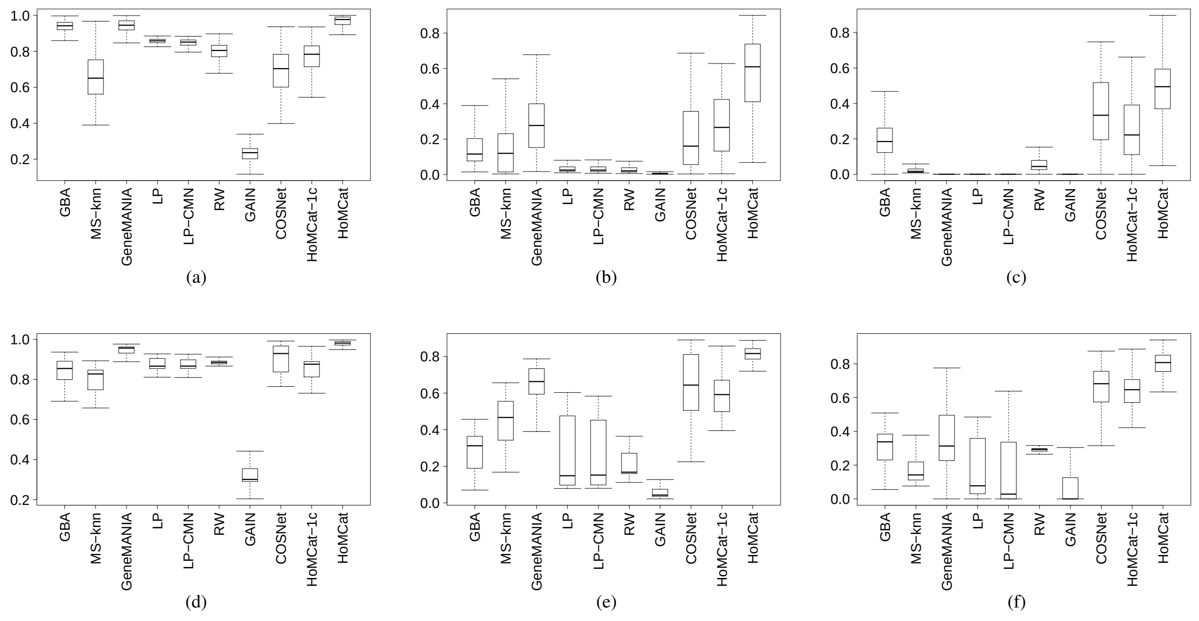

Fig. 7 Boxplots showing the performance distribution of the compared methods over classes with the *Bacteria2* dataset. The first row shows results relative to classes having from 20 to 200 positive examples (20-200), the second row results for classes with more than 200 positives ($>$ 200). AUC (a: 20-200, d: $>$ 200), AUPRC (b: 20-200, e: $>$ 200) and *F-score* (c: 20-200, f: $>$ 200).

terms of both *F-score* and AUPRC. This also explains why the same methods perform much better in terms of AUC, that is a measure which does not properly take into account the labeling imbalance (see [68] for further details). Moreover, these methods are rankers, that is they only provide for each node a membership score for the class being predicted, and this also may cause a decay in performance in terms of *F-score*. On the other hand, *COSNet* is a binary classifier, and this explains why it achieves better performance in terms of *F-score* than in terms of AUPRC and AUC.

Furthermore, we observe remarkable improvements for *HoMCat* when compared with classical HNs; indeed GAIN is the worst method in almost all the considered settings. Even when compared with COSNet, another algorithm based on Hopfield networks, our method always performs better. Finally, *HoMCat* achieves significantly better results than *HoMCat-1c* in both balanced and unbalanced settings (p-value $= 10^{-3}$), showing that exploiting meaningful prior partitions of input instances may largely improve the predictive capability of the model. Indeed, *HoMCat-1c* is the third or the fourth method in the majority of the experiments, and in terms of AUC is among the worst methods.

6 Conclusions

The partition of network nodes into categories, i.e. groups of related instances not attributable to the classes to be predicted, can be exploited to improve predictions in graph-based node label learning problems. Such problems arise naturally from the analysis of real-word data, including multi-species protein function prediction problems and personalized medicine.

To this end, we first introduced a new model of Hopfield network to embed node categories, namely the *m-Category Hopfield Network*, then we designed a novel algorithm, *HoMCat*, which can exploit prior partitions of input data on account of specific instance properties and also manage the imbalance between positive and negative instances.

We proved that the dynamics of this model converges to a stable state of the *m-Category Hopfield Network* (minimum of the corresponding energy), and we showed that the time complexity of the algorithm is linear in the number of nodes and categories when the graph is sparse. Moreover, our experiments suggest that the dynamics of the network usually converges after few iterations of node updates.

Experimental results in the context of multi-species function prediction problems show that *HoMCat* improves the performance of the base model with no partition of input instances, and favorably compares with several state-of-the-art algorithms for node label prediction in graph-structured data. Moreover the experimental results obtained with synthetic data show that the partition of input instances into multiple categories can induce noise resiliency and improve the stability of the underlying Hopfield network.

In perspective this approach can be extended also to cases where categories are partially or completely not known *a* *priori*, by inferring them from the data, e.g. through properly selected or designed unsupervised or semi-supervised methods. Moreover, we plan to extend *HoMCat* to multi-level functional classes, by exploiting the theory of *Generalized Hopfield networks* [71] which replaces bi-level activation function with their multilevel counterparts.

Appendix A Convergence proof

The following fact adapts the convergence proof for Hopfield networks with neuron activation values $\{\sin\alpha, -\cos\alpha\}$ [21] to the case with $m \geq 2$ categories of neurons $V_1, V_2, \ldots, V_m$, each with a different couple of activation values $\{\sin\alpha_k, -\cos\alpha_k\}$, for neurons belonging to category V_k.

Fact 5. *A m-Category Hopfield Network $\mathscr{H} = \langle \boldsymbol{W}, \boldsymbol{b}, \alpha_1, \ldots, \alpha_m, \boldsymbol{\lambda} \rangle$ with neurons $V = \{1, 2, \ldots, n\}$ and asynchronous dynamics (1), starting from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

Proof. We observe that the energy (2) is equivalent to the following

$$E(\boldsymbol{x}) = -\frac{1}{2}\sum_{i,j=1}^{n} W_{ij}x_i x_j + \sum_{i=1}^{n} x_i\lambda_i \tag{11}$$

During the iteration $t+1$, each unit i is selected and updated according to the update rule (1). To simplify the notation we set $x_i' = x_i(t+1)$ and $x_i = x_i(t)$. If the unit i does not change its state, then the energy of the system does not change as well, and we set $x_i' = x_i$. Otherwise, the network reaches a new global state $\boldsymbol{x}' = (x_1, \ldots, x_i', \ldots, x_n)$ having energy $E(\boldsymbol{x}')$. Since by definition $W_{ii} = 0$, the difference between $E(\boldsymbol{x}')$ and $E(\boldsymbol{x})$ is given by all terms in the summation (11) which contain x_i' and x_i, that is

$$\begin{aligned} E(\boldsymbol{x}') - E(\boldsymbol{x}) &= -\sum_{x_i' \neq x_i}\left((x_i' - x_i)\left(\sum_{j=1}^{n} W_{ij}x_j - \lambda_i\right)\right) \\ &= -\sum_{x_i' \neq x_i}\left((x_i' - x_i)B_i\right) \end{aligned} \tag{12}$$

where $B_i = \sum_{j=1}^{n} W_{ij}x_j - \lambda_i$. The factor $1/2$ disappears from the computation because the terms $W_{ij}x_i x_j$ appear twice in the double summation in (11). If $B_i > 0$, it means that $x_i' = \sin\alpha_{b_i}$ and $x_i = -\cos\alpha_{b_i}$ (see (1)); since $0 \leq \alpha_{b_i} < \frac{\pi}{2}$, $(x_i' - x_i) > 0$ and $E(\boldsymbol{x}') - E(\boldsymbol{x}) < 0$. If $B_i < 0$, it means that $x_i' = -\cos\alpha_{b_i}$ and $x_i = \sin\alpha_{b_i}$ and again $E(\boldsymbol{x}') - E(\boldsymbol{x}) < 0$. If $B_i = 0$, the energy value does not change after the update. Hence the energy (11) is a monotonically decreasing function. Moreover, since the connections W_{ij} are non negative, the energy E is lower bounded by the value

$$E_{\mathrm{LB}} = -\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sin(\alpha_{b_i})\sin(\alpha_{b_j})W_{ij} - \lambda_i\right) \tag{13}$$

and the dynamics is guaranteed to converge to a fixed point, corresponding to a local minimum of the energy. □

Appendix B Generative process for synthetic data

We provide here detailed information about the generative process of the two families of experiments on synthetic data described in Sec. 5.1. The generator used in the experiments is available from the authors upon request.

The main prerequisite to perform the first family of experiments is the generation of separable patterns. For this purpose we decided to work in the Δ space, where each node $i \in V$ is mapped in a $2m$-dimensional point $\Delta(i) \equiv \{\Delta_1^+(i), \Delta_1^-(i), \ldots, \Delta_m^+(i), \Delta_m^-(i)\}$ where:

$$\Delta_k^+(i) = \sum_{j \in V_k \cap S^+} w_{ij}, \quad \Delta_k^-(i) = \sum_{j \in V_k \cap S^-} w_{ij}, \quad k = 1, \ldots, m \tag{14}$$

and, according to the notation used throughout the paper, V_k is the k-th category, S^+ (resp. S^-) the set of positive (resp. negative) labeled instances, and W_{ij} the weight connecting nodes i and j.

Namely, after having fixed a labeling imbalance ε, we randomly generated a set of n points $\Delta(i)$ in the unitary hypercube, partitioned them into m categories according to their Euclidean similarity in the Δ space. Then we drew α_k uniformly for each category k, and computed the threshold λ guaranteeing the correct labeling imbalance, i.e. satisfying the following equation:

$$\sum_{i=1}^{n} \mathrm{HS}\left(\sum_{k=1}^{m}\left(\sin\alpha_k\Delta_k^+(i) - \cos\alpha_k\Delta_k^-(i)\right) - \lambda\right) = \frac{n\varepsilon}{\varepsilon+1} \tag{15}$$

where HS is the Heaviside step function, i.e. $\mathrm{HS}(x) = 1$ if $x \geq 0$, 0 otherwise.

We run the proposed learning algorithm starting with m categories, and subsequently evaluated its performance on a lower number $\widetilde{m} < m$ of them. Although other strategies could be adopted, the decision of working directly in the Δ space was motivated by the need of exerting more control on the construction of separable instances. We select the number $\widetilde{m}$ of categories to be investigated as a divisor of m. For instance, in case $m = 4$, by choosing $\widetilde{m} = 2$ we may easily consider the reduced Δ space obtained by associating to node i the point $\{\Delta_1^+(i) + \Delta_2^+(i),\ \Delta_1^-(i) + \Delta_2^-(i),\ \Delta_3^+(i) + \Delta_4^+(i),\ \Delta_3^-(i) + \Delta_4^-(i)\}$ which, in turn, can be coupled with two parameters (α_1 and α_2) out of four. Of course, in order to preserve the concept of category, this aggregation should be performed by merging together the most similar categories, where similarity is again defined in terms of Euclidean metric in the Δ space.

given the set of nodes V, partitioned into positives V_+ and negatives V_-, the vector $\boldsymbol{b}$ partitioning V in m disjoint categories and the corresponding parameters $(\alpha_1, \ldots, \alpha_m, \lambda)$, we compute the initial labeling $\tilde{\boldsymbol{x}}$ as follows:

$$\tilde{x}_i = \begin{cases} \sin\alpha_{b_i} & \text{if } i \in V_+ \\ -\cos\alpha_{b_i} & \text{if } i \in V_- \end{cases} \tag{16}$$

Then, by exploiting the associative memory functionality of the Hopfield network, we generated the weight matrix $\boldsymbol{W}$ through the naive Hebbian rule [25] in order to store the pattern $\tilde{\boldsymbol{x}}$, i.e. $\tilde{\boldsymbol{x}}$ becomes a fixed point of the dynamics [26]. Moreover, to make the convergence to $\tilde{\boldsymbol{x}}$ as much complex as possible and hence to amplify the fixed point perturbations, we learned the weights $\boldsymbol{W}$ in order to store also other μ randomly generated patterns with the same label imbalance $\varepsilon = |V_+|/|V_-|$, where $\mu \simeq 0.13|V|$ is the capacity of the network when trained with the naive Hebbian rule. Then, after having hidden the labels for a subset $U \subset V$ chosen according to the fixed $|S|/|V|$ ratio, for each value of the standard deviation σ modulating the injected noise, we run the network restricted to U till convergence and finally we measure the perturbation of the attained stable state with respect to $\tilde{\boldsymbol{x}}_u$.

Acknowledgements Most of the main ideas behind this paper have been discussed with Alberto Bertoni. We would like to dedicate this paper to Alberto, who recently departed his life.

References

1. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature genetics **25**(1), 25–29 (2000)

2. Atencia, M., Joya, G., Sandoval, F.: Parametric identification of robotic systems with stable time-varying Hopfield networks. Neural Computing and Applications **13**(4), 270–280 (2004). DOI 10.1007/s00521-004-0421-4. URL `http://dx.doi.org/10.1007/s00521-004-0421-4`
3. Attwood, T.K., Bradley, P., Flower, D.R., Gaulton, A., Maudling, N., Mitchell, A., Moulton, G., Nordle, A., Paine, K., Taylor, P., et al.: Prints and its automatic supplement, preprints. Nucleic acids research **31**(1), 400–402 (2003)
4. Azran, A.: The rendezvous algorithm: Multi- class semi-supervised learning with Markov random walks. In: Proceedings of the 24th International Confer- ence on Machine Learning (ICML) (2007)
5. Bairoch, A., Apweiler, R.: ”the swiss-prot protein sequence data bank and its supplement trembl”. Nucl. Acids Res. **25**(1), 31–36 (1997)
6. Bengio, Y., Delalleau, O., Le Roux, N.: Label Propagation and Quadratic Criterion. In: O. Chapelle, B. Scholkopf, A. Zien (eds.) Semi-Supervised Learning, pp. 193–216. MIT Press (2006)
7. Bertoni, A., Frasca, M., Valentini, G.: Cosnet: A cost sensitive neural network for semi-supervised learning in graphs. In: ECML/PKDD (1), *Lecture Notes in Computer Science*, vol. 6911, pp. 219–234. Springer (2011)
8. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. CoRR **abs/1101.3291** (2011)
9. Bogdanov, P., Singh, A.K.: Molecular function prediction using neighborhood features. IEEE/ACM Trans. Comput. Biol. Bioinformatics **7**, 208–217 (2010)
10. Brent, R.: Algorithms for minimization without derivatives. Prentice-Hall (1973)
11. Chaudhari, G., Avadhanula, V., Sarawagi, S.: A few good predictions: Selective node labeling in a social network. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14, pp. 353–362. ACM, New York, NY, USA (2014). DOI 10.1145/2556195.2556241. URL `http://doi.acm.org/10.1145/2556195.2556241`
12. Chen, R.M., Huang, Y.M.: Multiprocessor Task Assignment with Fuzzy Hopfield Neural Network Clustering Technique. Neural Computing and Applications **10**(1), 12–21 (2001). DOI 10.1007/s005210170013. URL `http://dx.doi.org/10.1007/s005210170013`
13. Cheng, Z., Caverlee, J., Lee, K.: You are where you tweet: A content-based approach to geo-locating twitter users. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 759–768. ACM, New York, NY, USA (2010)
14. Chua, H.N., Sung, W.K., Wong, L.: Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions. Bioinformatics **22**, 1623–1630 (2006)
15. Deng, M., Chen, T., Sun, F.: An integrated probabilistic model for functional prediction of proteins. J. Comput. Biol. **11**, 463–475 (2004)
16. Elkan, C.: The foundations of cost-sensitive learning. In: In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 973–978 (2001)
17. Erdem, M.H., Ozturk, Y.: A new family of multivalued networks. Neural Networks **9**(6), 979–989 (1996)
18. Ertoz, L., Steinbach, M., Kumar, V.: A new shared nearest neighbor clustering algorithm and its applications. In: Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining (2002)
19. Finn, R.D., Mistry, J., Schuster-Böckler, B., Griffiths-Jones, S., Hollich, V., Lassmann, T., Moxon, S., Marshall, M., Khanna, A., Durbin, R., et al.: Pfam: clans, web tools and services. Nucleic acids research **34**(suppl 1), D247–D251 (2006)
20. Frasca, M.: Automated gene function prediction through gene multifunctionality in biological networks. Neurocomputing (2015). DOI http://dx.doi.org/10.1016/j.neucom.2015.04.007. URL `http://www.sciencedirect.com/science/article/pii/S0925231215004142`. In press.
21. Frasca, M., Bertoni, A., et al.: A neural network algorithm for semi-supervised node label learning from unbalanced data. Neural Networks **43**(0), 84 – 98 (2013)
22. Frasca, M., Pavesi, G.: A neural network based algorithm for gene expression prediction from chromatin structure. In: IJCNN, pp. 1–8. IEEE (2013)
23. Gough, J., Karplus, K., Hughey, R., Chothia, C.: Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure. Journal of molecular biology **313**(4), 903–919 (2001)
24. Guyon, I., Cawley, G., Dror, G. (eds.): Hands-On Pattern Recognition: Challenges in Machine Learning, *Challenges in Machine Learning*, vol. 1. Microtome Publishing, Brookline, MA (2011)
25. Hebb, D.O.: The Organization of Behavior:: A Neuropsychological Theory. Lawrence Erlbaum Associates Inc,US, Mahwah, NJ (2002). URL `http://www.loc.gov/catdir/enhancements/fy0659/2002018867-d.html`
26. Hopfield, J.: Neural networks and physical systems with emergent collective compatational abilities. Proc. Natl Acad. Sci. USA **79**, 2554–2558 (1982)
27. Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., De Castro, E., Langendijk-Genevaux, P.S., Pagni, M., Sigrist, C.J.: The prosite database. Nucleic acids research **34**(suppl 1), D227–D230 (2006)
28. Jarvis, R.A., Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. IEEE Trans. Comput. **22**(11), 1025–1034 (1973)
29. Karaoz, U., et al.: Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl Acad. Sci. USA **101**, 2888–2893 (2004)
30. Kohler, S., Bauer, S., Horn, D., Robinson, P.: Walking the interactome for prioritization of candidate disease genes. Am. J. Human Genetics **82**(4), 948–958 (2008)
31. Kordos, M., Duch, W.: Variable step search algorithm for feedforward networks. Neurocomput. **71**(13-15), 2470–2480 (2008). DOI 10.1016/j.neucom.2008.02.019. URL `http://dx.doi.org/10.1016/j.neucom.2008.02.019`
32. Lan, L., et al.: MS-kNN: protein function prediction by integrating multiple data sources. BMC Bioinformatics **14**(Suppl 3:S8) (2013)
33. Letunic, I., Copley, R.R., Pils, B., Pinkert, S., Schultz, J., Bork, P.: Smart 5: domains in the context of genomes and networks. Nucleic acids research **34**(suppl 1), D257–D260 (2006)
34. Ling, C., Sheng, V.: Class imbalance problem. In: C. Sammut, G. Webb (eds.) Encyclopedia of Machine Learning, pp. 171–171. Springer US (2010). DOI 10.1007/978-0-387-30164-8_110. URL `http://dx.doi.org/10.1007/978-0-387-30164-8_110`
35. Ling, C., Sheng, V.: Cost-sensitive learning. In: C. Sammut, G. Webb (eds.) Encyclopedia of Machine Learning, pp. 231–235. Springer US (2010). DOI 10.1007/978-0-387-30164-8_181. URL `http://dx.doi.org/10.1007/978-0-387-30164-8_181`
36. Lovász, L.: Random walks on graphs: A survey. In: D. Miklós, V.T. Sós, T. Szőnyi (eds.) Combinatorics, Paul Erdős is Eighty, vol. 2, pp. 353–398. János Bolyai Mathematical Society, Budapest (1996)
37. Ma, J.: The Object Perceptron Learning Algorithm on Generalised Hopfield Networks for Associative Memory. Neural Computing and Applications **8**(1), 25–32 (1999). DOI 10.1007/s005210050004. URL `http://dx.doi.org/10.1007/s005210050004`
38. Marcotte, E., Pellegrini, M., Thompson, M., Yeates, T., Eisenberg, D.: A combined algorithm for genome-wide prediction of protein function. Nature **402**, 83–86 (1999)
39. Mayer, M.L., Hieter, P.: Protein networks-built by association. Nat Biotechnol **18**(12), 1242–3 (2000)

40. Mérida-Casermeiro, E., Galán-Marín, G., Muñoz Pérez, J.: An Efficient Multivalued Hopfield Network for the Traveling Salesman Problem. Neural Process. Lett. **14**(3), 203–216 (2001). DOI 10.1023/A:1012751230791. URL `http://dx.doi.org/10.1023/A:1012751230791`
41. Mesiti, M., Re, M., Valentini, G.: Think globally and solve locally: secondary memory-based network learning for automated multi-species function prediction. GigaScience **3:5** (2014). DOI {http://doi:10.1186/2047-217X-3-5}
42. Mislove, A., Viswanath, B., Gummadi, K.P., Druschel, P.: You are who you know: Inferring user profiles in online social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM '10, pp. 251–260. ACM, New York, NY, USA (2010). DOI 10.1145/1718487.1718519. URL `http://doi.acm.org/10.1145/1718487.1718519`
43. Mostafavi, S., Morris, Q.: Fast integration of heterogeneous data sources for predicting gene function with limited annotation. Bioinformatics **26**(14), 1759–1765 (2010)
44. Mostafavi, S., Ray, D., Farley, D.W., et al.: GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. Genome Biology **9**(Suppl 1), S4+ (2008)
45. Mulder, N.J., Apweiler, R., Attwood, T.K., Bairoch, A., Bateman, A., Binns, D., Bork, P., Buillard, V., Cerutti, L., Copley, R., et al.: New developments in the interpro database. Nucleic acids research **35**(suppl 1), D224–D228 (2007)
46. Muller, J., Szklarczyk, D., Julien, P., Letunic, I., Roth, A., Kuhn, M., Powell, S., von Mering, C., Doerks, T., Jensen, L.J., et al.: eggnog v2. 0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations. Nucleic acids research **38**(suppl 1), D190–D195 (2010)
47. Murali, T.M., Wu, C.J., Kasif, S.: The art of gene function prediction. Nature Biotechnology **24**(12), 1474–1475 (2006). DOI 10.1038/nbt1206-1474. URL `http://dx.doi.org/10.1038/nbt1206-1474`
48. Muruganantham, G., Bhakat, R.S.: A review of impulse buying behavior. International Journal of Marketing Studies **5**(3), p149 (2013)
49. Nabieva, E., Jim, K., Agarwal, A., Chazelle, B., Singh, M.: Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. Bioinformatics **21**(S1), 302–310 (2005)
50. Neagu, D., Palade, V.: A neuro-fuzzy approach for functional genomics data interpretation and analysis. Neural Computing and Applications **12**(3-4), 153–159 (2003). DOI 10.1007/s00521-003-0388-6. URL `http://dx.doi.org/10.1007/s00521-003-0388-6`
51. Nie, F., Xiang, S., Liu, Y., Zhang, C.: A general graph-based semi-supervised learning with novel class discovery. Neural Computing and Applications **19**(4), 549–555 (2010). DOI 10.1007/s00521-009-0305-8. URL `http://dx.doi.org/10.1007/s00521-009-0305-8`
52. Pena-Castillo, L., Tasan, M., Myers, C., et al.: A critical assessment of Mus musculus gene function prediction using integrated genomic evidence. Genome Biology **9**, S1 (2008)
53. Radivojac, P., et al.: A large-scale evaluation of computational protein function prediction. Nature Methods **10**(3), 221–227 (2013)
54. Re, M., Mesiti, M., Valentini, G.: A fast ranking algorithm for predicting gene functions in biomolecular networks. IEEE/ACM Trans. Comput. Biol. Bioinformatics **9**(6), 1812–1818 (2012). DOI 10.1109/TCBB.2012.114. URL `http://dx.doi.org/10.1109/TCBB.2012.114`
55. Re, M., Valentini, G.: Cancer module genes ranking using kernelized score functions. BMC Bioinformatics **13**(Suppl 14/S3) (2012). DOI 10.1186/1471-2105-13-S14-S3. URL `http://www.biomedcentral.com/bmcbioinformatics/supplements/13/S14/S3`
56. Salavati, A.H., Kumar, K.R., Shokrollahi, A.: A non-binary associative memory with exponential pattern retrieval capacity and iterative learning: Extended results. CoRR **abs/1302.1156** (2013)
57. Schwikowski, B., Uetz, P., Fields, S.: A network of protein-protein interactions in yeast. Nature biotechnology **18**(12), 1257–1261 (2000)
58. Silva, I., Moody, G., Scott, D.J., Celi, L.A., Mark, R.G.: Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. Comput Cardiol **39**, 245–248 (2012). URL `http://www.biomedsearch.com/nih/Predicting-In-Hospital-Mortality-ICU/24678516.html`
59. Szummer, M., Jaakkola, T.: Partially labeled classification with Markov random walks. In: Advances in Neural Information Processing Systems (NIPS), vol. 14, pp. 945–952. MIT Press (2001)
60. Tsuda, K., Shin, H., Scholkopf, B.: Fast protein classification with multiple networks. Bioinformatics **21**(Suppl 2), ii59–ii65 (2005)
61. Valentini, G., Paccanaro, A., Caniza, H., Romero, A., Re, M.: An extensive analysis of disease-gene associations using network integration and fast kernel-based gene prioritization methods. Artificial Intelligence in Medicine **61**(2), 63–78 (2014). DOI {http://10.1016/j.artmed.2014.03.003}
62. Vazquez, A., Flammini, A., Maritan, A., Vespignani, A.: Global protein function prediction from protein-protein interaction networks. Nature Biotechnology **21**, 697–700 (2003)
63. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics **1**, 80–83 (1945)
64. Wolfram Research, Inc.: Mathematica (2012). URL `http://www.wolfram.com/mathematica/`. Version 9.0
65. Wong, A.K., Park, C.Y., Greene, C.S., Bongo, L.A., Guan, Y., Troyanskaya, O.G.: Imp: a multi-species functional genomics portal for integration, visualization and prediction of protein functions and networks. Nucleic acids research **40**(W1), W484–W490 (2012)
66. Xue, H., Chen, S.: Glocalization pursuit support vector machine. Neural Computing and Applications **20**(7), 1043–1053 (2011). DOI 10.1007/s00521-010-0448-7. URL `http://dx.doi.org/10.1007/s00521-010-0448-7`
67. Yoon, K., Kwek, S.: A data reduction approach for resolving the imbalanced data issue in functional genomics. Neural Computing and Applications **16**(3), 295–306 (2007). DOI 10.1007/s00521-007-0089-7. URL `http://dx.doi.org/10.1007/s00521-007-0089-7`
68. Youngs, N., Penfold-Brown, D., Drew, K., Shasha, D., Bonneau, R.: Parametric Bayesian Priors and Better Choice of Negative Examples Improve Protein Function Prediction. Bioinformatics **29**(9), btt110–1198 (2013). DOI 10.1093/bioinformatics/btt110
69. Zhou, D., et al.: Learning with local and global consistency. In: S. Thrun, L. Saul, B. Schölkopf (eds.) Advances in Neural Information Processing Systems 16, pp. 321–328. MIT Press (2004). URL `http://papers.nips.cc/paper/2506-learning-with-local-and-global-consistency.pdf`
70. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: In ICML, pp. 912–919 (2003)
71. Zurada, J.M., Cloete, I., van der Poel, E.: Generalized Hopfield networks for associative memories with multi-valued stable states. Neurocomputing **13**(24), 135 – 149 (1996)