Multitask Hopfield Networks

Marco Frasca^{1}✉$^{[0000-0002-4170-0922]}$, Giuliano Grossi$^{1[0000-0001-9274-4047]}$ and Giorgio Valentini$^{1[0000-0002-5694-3919]}$

Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, 20135 Milano, Italy
{frasca,grossi,valentini}@di.unimi.it

Abstract. Multitask algorithms typically use task similarity information as a bias to speed up and improve the performance of learning processes. Tasks are learned jointly, sharing information across them, in order to construct models more accurate than those learned separately over single tasks. In this contribution, we present the first multitask model, to our knowledge, based on Hopfield Networks (HNs), named HoMTask. We show that by appropriately building a unique HN embedding all tasks, a more robust and effective classification model can be learned. HoMTask is a transductive semi-supervised parametric HN, that minimizes an energy function extended to all nodes and to all tasks under study. We provide theoretical evidence that the optimal parameters automatically estimated by HoMTask make coherent the model itself with the prior knowledge (connection weights and node labels). The convergence properties of HNs are preserved, and the fixed point reached by the network dynamics gives rise to the prediction of unlabeled nodes. The proposed model improves the classification abilities of singletask HNs on a preliminary benchmark comparison, and achieves competitive performance with state-of-the-art semi-supervised graph-based algorithms.

Keywords: Multitask Hopfield networks, multitask learning

1 Introduction

Multitask learning is concerned with simultaneously learning multiple prediction tasks that are related to one another. It has been frequently observed in the recent literature that, when there are relations between the tasks, it can be advantageous to learn them simultaneously instead of learning each task separately [7, 11]. A major challenge in multitask learning is how to selectively screen the sharing of information so that unrelated tasks do not end up influencing each other. Sharing information between two unrelated tasks can worsen the performance of both tasks.

Multitasking thus plays an important role in a variety of practical situations, including: the prediction of user ratings for unseen items based on rating information from related users [32], the simultaneously forecasting of many related financial indicators [19], the categorization of genes associated with a genetic disorder by exploiting genes associated with related diseases [15].

There is a vast literature on multitask learning. The most important lines of work include: regularizers biasing the solution towards functions that lie geometrically close to each other in a RKHS [12, 11], or lie in a low dimensional subspace [3, 26]; structural risk minimization methods, where multitask relations are established by enforcing predictive functions for the different tasks to belong to the same hypothesis set [2]; spectral [10, 4] and cluster-based [23, 38] assumptions on the task relatedness; Bayesian approaches where task parameters share a common prior [39, 9, 42]; methods allowing a small number of outlier tasks that are not related to any other task [40, 8]; approaches attempting to learn the full task covariance matrix [41, 20]. To our knowledge, no multitask attempts have been proposed for Hopfield networks (HNs) [21], whereas several studies investigated HNs as singletask classifier [24, 27, 17, 22]. Indeed, HNs are efficient local optimizers, using the local minima of the energy function determined by network dynamics as a proxy to node classification.

In this paper we develop *HoMTask*, *Hopfield multitask Network*, an approach to multitask learning based on exploiting a family of parametric HNs. Our approach builds on COSNet [6], a singletask HN proposed to classify instances in a transductive semi-supervised scenario with unbalanced data. A main feature of HoMTask is that the energy function is extended to all tasks to be learned and to all instances (labeled and unlabeled), so as to learn the model parameters and to infer the node labels simultaneously for all tasks. The obtained network can be seen as a collection of singletask HNs, appropriately interconnected by exploiting the task relatedness. In particular, each task is associated with a couple of parameters determining the neuron activation values and thresholds, and we theoretically prove that in the optimal case, the learning procedure adopted is able to learn the parameters so as to move the multitask state of the labeled sub-network to a minimum of the energy. This is an important result, which allows the model to better fit the input data, since the classification of unlabeled nodes is based upon a minimum of the unlabeled subnetwork. Another interesting feature of HoMTask is that the complexity of the learning procedure linearly increases with the number of tasks, thus allowing the model to nicely scale on settings including numerous tasks. Finally, a proof of convergence of the multitask dynamics to a minimum of the energy is also supplied.

Experiments on a real-world classification problem have shown that HoMTask remarkably outperforms singletask HNs, and has competitive performance with state-of-the-art graph-based methods proposed in the same context.

2 Methods

2.1 Problem definition

The problem input is composed of an undirected weighted graph $G(V, \boldsymbol{W})$, where $V = \{1, 2, \ldots, n\}$ is the set of instances and the non negative symmetric matrix $\boldsymbol{W} = (w_{ij})$ denotes the degree of functional similarity between each pair of nodes i and j. A set of binary learning tasks $C = \{c_k | k = 1, 2, \ldots, m\}$ over G is given, where for every task c_k, V is labelled with $\{+, -\}$. The labeling is known only for the subset $L \subset V$, whereas it is unknown for $U := V \setminus L$. Moreover, the subsets of vertices labelled with $+$ (positive) and $-$ (negative) are denoted by $L_{k,+}$ and $L_{k,-}$, respectively, for each task $c_k \in C$. Without loss of generality, we assume $U = \{1, 2, \cdots, h\}$ and $L = \{h+1, h+2, \cdots, n\}$. As further assumption, task labelings are highly unbalanced, that is $\frac{|L_{k,+}|}{|L_{k,-}|} \ll 1$, for each $k \in \{1, 2, \dots, m\}$. In the multitask scenario, a $m \times m$ symmetric matrix $\boldsymbol{S} = s_{kr}|_{k,r=1}^{m}$ is also given, where $s_{kr} \in [0, 1]$ is an index of relatedness/similarity between the tasks c_k and c_r, and $s_{kk} = 0$ for each $k \in \{1, 2, \dots, m\}$, to learn just from the other tasks.

The aim is determining a set of bipartitions $(U_{k,+}, U_{k,-})$ of vertices in U for each task $c_k \in C$ by jointly learning tasks in C, on the basis of the prior information encoded in G and $\boldsymbol{S}$. In the following, the bold font is adopted to denote vectors and matrices, and the calligraphic font to denote multitask Hopfield networks. Moreover, we denote by $\boldsymbol{W}_{LL}$ and $\boldsymbol{W}_{UU}$ the submatrices of $\boldsymbol{W}$ relative to nodes L and U, respectively.

2.2 Previous singletask model

In this section we recall the basic model proposed in [6, 13] for singletask modeling, named *COSNet*, that has inspired the multitask setting presented here. Essentially, it relies on a parametric family of the Hopfield model [21], where the network parameters are learned to cope with the label imbalance and the network equilibrium point is interpreted to classify the unlabeled nodes. A COSNet network over $G = \langle V, \boldsymbol{W} \rangle$ is a triple $H = \langle \boldsymbol{W}, \lambda, \rho \rangle$, where $\lambda \in \mathbb{R}$ denotes the neuron activation threshold (unique for all neurons), and $\rho \in [0, \frac{\pi}{2})$ is a parameter which determines the two neuron activation (state) values $\{\sin\rho, -\cos\rho\}$. The model parameters are appropriately learned in order to allow the algorithm to counterbalance the large imbalance towards negatives (see [13]). The initial state of a neuron $i \in V$ is set to $x_i(0) = \sin\rho$, if i is positive, $x_i(0) = -\cos\rho$, if i is negative, and $x_i(0) = 0$ when i in unlabeled. The network evolves according to the following asynchronous dynamics:

$$x_i(t) = \begin{cases} \sin\rho & \text{if } \sum\limits_{j=1}^{i-1} w_{ij} x_j(t) + \sum\limits_{k=i+1}^{n} w_{ik} x_k(t-1) - \lambda > 0 \\ -\cos\rho & \text{if } \sum\limits_{j=1}^{i-1} w_{ij} x_j(t) + \sum\limits_{k=i+1}^{n} w_{ik} x_k(t-1) - \lambda \le 0 \end{cases} \tag{1}$$

where $x_i(t)$ is the state of neuron $i \in V$ at time t. At each time t, the vector $\boldsymbol{x}(t) = (x_1(t), x_2(t), \dots, x_h(t))$ represents the state of the whole network. The network admits a state function named *energy function*:

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

The convergence properties of the dynamics (1) depend on the weight matrix $\boldsymbol{W}$ and the rule by which the nodes are updated. In particular, if $\boldsymbol{W}$ is symmetric and the dynamic is asynchronous, it has been proved that the network converges to a stable state in polynomial time. As a major result, it has been shown that (2) is a Lyapunov function for the Hopfield dynamical systems with asynchronous dynamics, i.e., for each $t > 0$, $E(\boldsymbol{x}(t+1)) \leq E(\boldsymbol{x}(t))$ and there exists a time $\bar{t}$ such that $E(\boldsymbol{x}(t)) = E(\boldsymbol{x}(\bar{t}))$, for all $t \geq \bar{t}$. Moreover, the reached fixed point $\bar{\boldsymbol{x}} = \boldsymbol{x}(\bar{t})$ is a local minimum of (2). Then, a neuron i in U is classified as positive if $\bar{x}_i = \sin\rho$, as negative otherwise. COSNet has also a dynamics control to avoid trajectories towards trivial equilibrium minima (see [13] for details).

2.3 Multitask Hopfield networks

A *Hopfield multitask network*, named *HoMTask*, with neurons V is a quadruple $\mathcal{H} = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S} \rangle$, where $\boldsymbol{S}$ is the task similarity matrix, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m) \in \mathbb{R}^m$, $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_m) \in [\frac{\pi}{4}, \frac{\pi}{2})^m$. The couple of parameters (γ_k, ρ_k) is associated with task c_k, for each $k \in \{1, 2, \ldots, m\}$: by leveraging the approach adopted in COSNet, for a task c_k, the neuron activation values are $\{\sin\rho_k, -\cos\rho_k\}$, whereas γ_k is the neuron activation threshold (the same for every neuron). Such a formalization allows to keep the absolute activation value in the range $[0, 1]$, and to calibrate it by suitably learning $\rho_k \in [\frac{\pi}{4}, \frac{\pi}{2})$. For instance, in presence of a large majority of negative neurons, ρ_k close to $\frac{\pi}{2}$ would prevent positive neurons to be overwhelmed during the net dynamics.

The state of the network is the $n \times m$ matrix $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)})$, where $\boldsymbol{x}^{(k)} = (x_{1k},\ x_{2k},\ \ldots, x_{nk}) \in \{\sin\rho_k, -\cos\rho_k\}^n$ is the state vector corresponding to task c_k. When simultaneously learning related tasks c_k and c_r, an usual approach consists in expecting that the higher the relatedness s_{rk}, the closer the corresponding states. In our setting, this can be achieved by minimizing

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(r)}\|^2 \ ,$$

for any couple of tasks $c_k, c_r \in C$, with $k \neq r$. To this end, we incorporate a term proportional to $\sum_k \sum_r s_{kr} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(r)}\|^2$ into the energy of $\mathcal{H}$, thus obtaining:

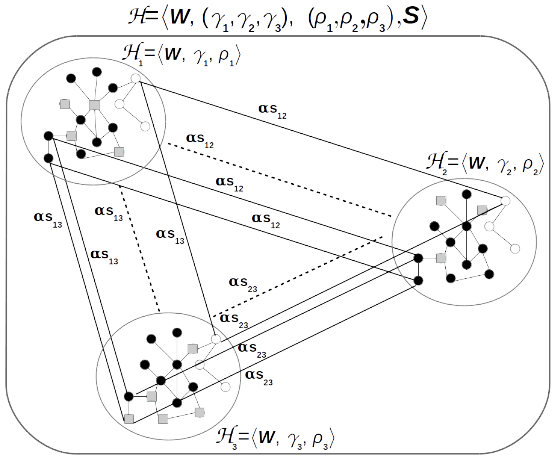
$$E_{\mathcal{H}}(\boldsymbol{X}) = \sum_{k=1}^{m} \left(E\big(\boldsymbol{x}^{(k)}\big) + \frac{\alpha}{4} \sum_{\substack{r=1 \\ r \neq k}}^{m} s_{kr}\ \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(r)}\|^2 \right) , \tag{3}$$

where $E\big(\boldsymbol{x}^{(k)}\big) = -\frac{1}{2}{\boldsymbol{x}^{(k)}}^T \boldsymbol{W} \boldsymbol{x}^{(k)} + {\boldsymbol{x}^{(k)}}^T \gamma_k \boldsymbol{e}_n$, $\boldsymbol{e}_n$ is the n-dimensional vector made by all ones, and α is a real hyper-parameter regulating the multitask contribution. Without the second additive term in brackets, energy (3) would be the summation of the energy functions of m independent singletask Hopfield networks, as recalled in the previous section.

By using the equality $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(r)}\|^2 = \|\boldsymbol{x}^{(k)}\|^2 + \|\boldsymbol{x}^{(r)}\|^2 - 2\boldsymbol{x}^{(k)} \cdot \boldsymbol{x}^{(r)}$, where $\cdot$ denotes the inner product, and giving that

$$\sum_{k=1}^{m} \sum_{\substack{r=1 \\ r \neq k}}^{m} s_{kr} \big(\|\boldsymbol{x}^{(k)}\|^2 + \|\boldsymbol{x}^{(r)}\|^2 \big) = 2 \Big(\sum_{k=1}^{m} S_k \|\boldsymbol{x}^{(k)}\|^2 \Big) \ ,$$

Fig. 1. Topology of $\mathcal{H}$ in the case $m = 3$. Black circles, gray squares and white circles represent elements of L_-, L_+ and U respectively. The local topology is the same across sub-networks $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, but the labeling varies with the task.


$\mathcal{H}=\langle \boldsymbol{W}, (\gamma_1,\gamma_2,\gamma_3), (\rho_1,\rho_2,\rho_3), \boldsymbol{S}\rangle$
$\mathcal{H}_1=\langle \boldsymbol{W}, \gamma_1, \rho_1\rangle$
$\mathcal{H}_2=\langle \boldsymbol{W}, \gamma_2, \rho_2\rangle$
$\mathcal{H}_3=\langle \boldsymbol{W}, \gamma_3, \rho_3\rangle$
αs_{12}
αs_{12}
αs_{12}
αs_{12}
αs_{13}
αs_{13}
αs_{13}
αs_{13}
αs_{23}
αs_{23}
αs_{23}
αs_{23}

with $S_k = \sum_{r=1}^{m} s_{kr}$, the energy (3) can be rewritten as:

$$E_{\mathcal{H}}(\boldsymbol{X}) = \sum_{k=1}^{m} \left(E\big(\boldsymbol{x}^{(k)}\big) + \frac{\alpha}{2}\Big(S_k \sum_{i=1}^{n} x_{ik}^2 - \sum_{\substack{r=1 \\ r\neq k}}^{m} s_{kr} \sum_{i=1}^{n} x_{ik}x_{ir} \Big) \right). \tag{4}$$

Informally, $\mathcal{H}$ can be thought as m interconnected singletask parametric Hopfield networks $H_1 = \langle \boldsymbol{W}, \boldsymbol{\gamma}_1, \boldsymbol{\rho}_1\rangle, \dots, H_m = \langle \boldsymbol{W}, \boldsymbol{\gamma}_m, \boldsymbol{\rho}_m\rangle$ on V, having all the same topology given by $\boldsymbol{W}$. In addition, the multitask energy term introduces self loops for all neurons, and a novel connection for each neuron $i \in V$ with i itself in the network H_r, $r \in C \setminus \{c_k\}$, whose weight is αs_{kr} (see Fig.1). It is worth nothing that usually in Hopfield networks there are no self-loops; nevertheless, we show that it does not affect the convergence properties of the overall network.

Update rule and dynamics convergence. Starting from an initial state $\boldsymbol{X}(0)$ and adopting the asynchronous dynamic, in nm steps all neurons are updated in random order according to the following update rule:

$$x_{ik}(t+1) = \begin{cases} \sin\rho_k, & \text{if } \phi_{ik}(t) > 0 \\ -\cos\rho_k, & \text{if } \phi_{ik}(t) \leq 0 \end{cases} \tag{5}$$

where $x_{ik}(t+1)$ is the state of neuron $i \in X$ in task c_k (ik-th) at time $t+1$, and

$$\phi_{ik}(t) := A_{ik}(t) - \theta_{ik} + \alpha B_{ik}(t) \tag{6}$$

is the input of the ik-th neuron at time t, whose terms are $A_{ik}(t) = \sum_{j=1}^{n} w_{ij} x_{jk}(t)$, $\theta_{ik} = \gamma_k + \frac{\alpha S_k}{2}\big(\sin\rho_k - \cos\rho_k\big)$, and $B_{ik}(t) = \sum_{\substack{r=1 \\ r\neq k}}^{m} s_{kr} x_{ir}(t)$. A_{ik} represents the singletask input (Eq. (1)), $B_{ik}(t)$ is the multitask contribution, and θ_{ik} is the activation threshold for neuron ik, including also the 'singletask' threshold. The form of θ_{ik} derives from the following theorem, stating a HoMTask Hopfield network preserves the convergence properties of a Hopfield network.

Theorem 1. *A HoMTask Hopfield network $\mathcal{H} = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S}\rangle$ with n neurons and the asynchronous dynamics (5), which starts from any given network state, eventually reaches a stable state at a local minimum of the energy function (4).*

Proof. Let $E_{ik}(t)$ be the energy contribution to (4) of the ik-th neuron at time t, with

$$E_{ik}(t) = -\frac{1}{2}x_{ik}(t)\sum_{j=1}^{h}(w_{ij}+w_{ji})x_{jk}(t) + \gamma_k x_{ik}(t) + \frac{\alpha S_k}{2}x_{ik}^2(t) - \frac{\alpha}{2}x_{ik}(t)\sum_{\substack{r=1 \\ r\neq k}}^{m}(s_{kr}+s_{rk})x_{ir}(t) \; .$$

Let $\Delta_{ik}E(t+1) = E_{ik}(t+1) - E_{ik}(t)$ be the energy variation after updating the state x_{ik} at time $t+1$ according to (5). Due to the symmetry of $\boldsymbol{W}$ and $\boldsymbol{S}$, it follows

$$\Delta_{ik}E(t+1) = -\big(x_{ik}(t+1) - x_{ik}(t)\big) \Big(A_{ik}(t) - \gamma_k - \frac{\alpha S_k}{2}\big(x_{ik}(t+1) + x_{ik}(t)\big) + \alpha B_{ik}(t)\Big). \tag{7}$$

Since (4) is lower bounded, to complete to proof we need to prove that after updating x_{ik} at time $t+1$ according to (5), it holds $\Delta_{ik}E(t+1) \leq 0$. From (7), when $x_{ik}(t+1) = x_{ik}(t)$ (no neuron state change) it follows $\Delta_{ik}E(t+1) = 0$. Accordingly, we need to investigate the remaining two cases: (a) $x_{ik}(t) = \sin\rho_k$ and $x_{ik}(t+1) = -\cos\rho_k$; (b) $x_{ik}(t) = -\cos\rho_k$ and $x_{ik}(t+1) = \sin\rho_k$. In both cases it holds (by definition of θ_{ik}) $\gamma_k + \frac{\alpha S_k}{2}\big(x_{ik}(t+1) + x_{ik}(t)\big) = \theta_{ik}$.

(a) $(x_{ik}(t+1) - x_{ik}(t)) = (-\cos\rho_k - \sin\rho_k) < 0$, and, according to (5), $A_{ik}(t) - \theta_{ik} + \alpha B_{ik}(t) \leq 0$. It follows $\Delta_{ik}E(t+1) \leq 0$.
(b) $(x_{ik}(t+1) - x_{ik}(t)) = (\sin\rho_k + \cos\rho_k) > 0$, and $A_{ik}(t) - \theta_{ik} + \alpha B_{ik}(t) > 0$. Thus $\Delta_{ik}E(t+1) < 0$.

Every neuron update thereby does not increase the network energy, and, since the energy is lower bounded, there will be a time $t' > 0$ from which the update of any neuron will not change the current state, which is the definition of equilibrium state of the network, and which makes $\boldsymbol{X}(t')$ a local minimum of (4). $\square$

Learning the model parameters. Considered the subnetwork $\mathcal{H}_{\mathcal{L}} = \langle \boldsymbol{W}_{\mathrm{LL}}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S} \rangle$ restricted to labeled nodes L, its energy is:

$$E_{\mathcal{H}_{\mathcal{L}}}(\boldsymbol{L}) = \sum_{k=1}^{m} \left(E_L\big(\boldsymbol{l}^{(k)}\big) + \frac{\alpha}{2}\Big(S_k \sum_{i \in L} l_{ik}^2 - \sum_{\substack{r=1 \\ r \neq k}}^{m} s_{kr} \sum_{i \in L} l_{ik} l_{ir} \Big) \right) , \tag{8}$$

where $\boldsymbol{L} = (\boldsymbol{l}^{(1)}, \boldsymbol{l}^{(2)}, \ldots, \boldsymbol{l}^{(m)})$ with components $\boldsymbol{l}^{(k)} = (l_{1k}, l_{2k}, \ldots, l_{(n-h)k})$ belonging to the set $\{\sin\rho_k, -\cos\rho_k\}^{(n-h)}$, and $E_L\big(\boldsymbol{l}^{(k)}\big) = -\frac{1}{2}\boldsymbol{l}^{(k)^T} \boldsymbol{W}_{\mathrm{LL}} \boldsymbol{l}^{(k)} + \boldsymbol{l}^{(k)} \cdot \gamma_k \boldsymbol{e}_{(n-h)}$.

The given bipartition $(L_{k,+}, L_{k,-})$ for each task c_k naturally induces the labeling $\bar{\boldsymbol{l}}^{(k)} = \{\bar{l}_{1k}, \bar{l}_{2k}, \ldots, \bar{l}_{(n-h)k}\}$, defined as it follows:

$$\bar{l}_{ik} = \begin{cases} \sin\rho_k, & \text{if } i \in L_{k,+} \\ -\cos\rho_k, & \text{if } i \in L_{k,-} \end{cases} ,$$

and constituting the known 'multitask' state $\bar{\boldsymbol{L}} = (\bar{\boldsymbol{l}}^{(1)}, \bar{\boldsymbol{l}}^{(2)}, \ldots, \bar{\boldsymbol{l}}^{(m)})$.

Given $\bar{\boldsymbol{L}}$ as known components of a final state $\bar{\boldsymbol{X}}$ of the multitask network $\mathcal{H} = \langle \boldsymbol{W}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S} \rangle$, the purpose of the learning step is to compute the pair $(\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\rho}})$ which makes $\bar{\boldsymbol{X}}$ an energy global minimizer of (3), the energy function associated with $\mathcal{H}$. Since our aim is also keeping the model scalable on large sized data, and finding the global minimum of the energy requires time/memory intensive procedures, we employ a learning procedure leading $\bar{\boldsymbol{L}}$ towards an fixed point of $\mathcal{H}_L$, being in general a local minimum of (8). We provide the details of the learning procedure in the following, showing that such an approach also helps to handle the label imbalance at each task.

Maximizing a cost-sensitive criterion. When the parameters $\boldsymbol{\gamma}, \boldsymbol{\rho}$ are fixed, each neuron ik has input

$$\phi_{ik}^{L}(\boldsymbol{\gamma}, \boldsymbol{\rho}) = \sum_{j \in L} w_{ij} \Big(\sin\rho_k \chi_{jk} - \cos\rho_k \big(1 - \chi_{jk}\big) \Big) - \theta_{ik} + \alpha \sum_{\substack{r=1 \\ r \neq k}} s_{kr} \Big(\sin\rho_k \chi_{ir} - \cos\rho_k \big(1 - \chi_{ir}\big) \Big) ,$$

where, for each $k \in \{1, \ldots, m\}$ and $j \in L$, $\chi_{jk} = 1$ if $j \in L_{k,+}$, 0 otherwise. ϕ_{ik}^{L} corresponds to ϕ_{ik} of equation (6) restricted to L; to simplify the notation, in the following ϕ_{ik}^{L} is thereby denoted by ϕ_{ik}. Since the subnetwork is labeled, it is possible to define the set of *true positive* $tp_k(\boldsymbol{\gamma}, \boldsymbol{\rho}) = \{i \in L_{k,+} | \phi_{ik}(\boldsymbol{\gamma}, \boldsymbol{\rho}) > 0\}$, *false negative* $fn_k(\boldsymbol{\gamma}, \boldsymbol{\rho}) = \{i \in L_{k,+} | \phi_{ik}(\boldsymbol{\gamma}, \boldsymbol{\rho}) \leq 0\}$, and *false positive* $fp_k(\boldsymbol{\gamma}, \boldsymbol{\rho}) = \{i \in L_{k,-} | \phi_{ik}(\boldsymbol{\gamma}, \boldsymbol{\rho}) > 0\}$, for every task c_k. Following the approach proposed in [16], a set of membership functions can be defined, extending the crisp memberships introduced above:

$$\begin{aligned}
&\mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = f(\tau\phi_{ik}(\boldsymbol{\gamma},\boldsymbol{\rho})), && i \in L_{k,+}\\
&\mathrm{FN}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = 1 - f(\tau\phi_{ik}(\boldsymbol{\gamma},\boldsymbol{\rho})), && i \in L_{k,+}\\
&\mathrm{FP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = f(\tau\phi_{ik}(\boldsymbol{\gamma},\boldsymbol{\rho})), && i \in L_{k,-}
\end{aligned} \tag{9}$$

where $f : \mathbb{R} \to [0, 1]$ is a suitable monotonically increasing membership function. For instance $f_1(x) = {}^1\!/\!\left(1+e^{-x}\right)$ or $f_2(x) = \frac{1}{2}\left(\frac{2}{\pi}\,\mathrm{arctg}(x) + 1\right)$. $\tau > 0$ is a real parameter. If f is the Heaviside step function, we obtain the crisp memberships. For example, when $f = f_1$ or $f = f_2$, if $i \in L_{k,+}$ and $\tau\phi_{ik}(\boldsymbol{\gamma},\boldsymbol{\rho}) = 0$, if follows $\mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = \mathrm{FN}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = 0.5$; if $i \in L_{k,+}$ and $\tau\phi_{ik}(\boldsymbol{\gamma},\boldsymbol{\rho}) \to \infty$, it follows $\mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = 1$ and $\mathrm{FN}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) = 0$. The intermediate cases lead to $0 < \mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}), \mathrm{FN}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) < 1$.

Such a generalization, in a different setting (singletask, multi-category) increased both the learning capability of the model and its classification performance [16]. By means of the membership functions (9), we can define the objective F:

$$F(\boldsymbol{\gamma},\boldsymbol{\rho}) = \sigma\big(F_1(\boldsymbol{\gamma},\boldsymbol{\rho}), F_2(\boldsymbol{\gamma},\boldsymbol{\rho}), \ldots, F_m(\boldsymbol{\gamma},\boldsymbol{\rho})\big)\ , \tag{10}$$

where $F_k(\boldsymbol{\gamma},\boldsymbol{\rho}) = \frac{2\sum_{i\in L_{k,+}} \mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho})}{2\sum_{i\in L_{k,+}} \mathrm{TP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) + \sum_{i\in L_{k,-}} \mathrm{FP}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho}) + \sum_{i\in L_{k,+}} \mathrm{FN}(i,k,\boldsymbol{\gamma},\boldsymbol{\rho})}$ and σ is an appropriately chosen function, e.g. the mean, the minimum, or the harmonic mean function. The property σ must satisfy is that

$$F(\boldsymbol{\gamma},\boldsymbol{\rho}) = 1 \Longrightarrow F_1(\boldsymbol{\gamma},\boldsymbol{\rho}) = F_2(\boldsymbol{\gamma},\boldsymbol{\rho}) = \ldots = F_m(\boldsymbol{\gamma},\boldsymbol{\rho}) = 1.$$

By definition, F_k (a generalization of the F-measure) is penalized more by the misclassification of a positive instance than by the misclassification of a negative one. By maximizing $F(\boldsymbol{\gamma},\boldsymbol{\rho})$ we can thereby cope with the label imbalance. To this end, the learning criterion for the model parameters adopted here is $(\hat{\gamma},\hat{\rho}) = \arg\max_{\boldsymbol{\gamma},\boldsymbol{\rho}} F(\boldsymbol{\gamma},\boldsymbol{\rho})$, which also leads to the following important result.

Theorem 2. *If $F(\boldsymbol{\gamma},\boldsymbol{\rho}) = 1$, then $\bar{\boldsymbol{L}}$ is an equilibrium state of the sub-network $\mathcal{H}_{\mathcal{L}}\langle W_{LL}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S}\rangle$.*

Learning procedure. Denoted by $\boldsymbol{\delta} = (\boldsymbol{\gamma},\boldsymbol{\rho})$ the vector of model parameters, this procedure learns the values $\hat{\boldsymbol{\delta}}$ that maximize eq. (10), that is $\hat{\boldsymbol{\delta}} = \operatorname{argmax}_{\boldsymbol{\delta}} F(\boldsymbol{\delta})$. Following the approach in [16], we adopt the *simplest search method* [28], which employs an iterative and incremental procedure estimating in turn a single parameter at a time, by fixing the other ones, until a suitable criterion is met (e.g. convergence, or number of iterations). Thus, the complexity of the learning procedure just linearly increases with the number of tasks. In particular, fixed an assignment of parameters $(\delta_1, \ldots, \delta_{i-1}, \delta_{i+1}, \ldots, \delta_{2m})$, $\hat{\delta}_i$ is estimated through $\bar{\delta}_i = \operatorname{argmax}_{\delta_i}\ F(\boldsymbol{\delta})$, $i \in \{1, \ldots, 2m\}$. The learning procedure is sketched below:

1. Randomly permute the vector $\boldsymbol{\delta}$, and randomly initialize $\boldsymbol{\delta}$;

2. Determine an estimate $\overline{\delta}_i$ of $\hat{\delta}_i$ with a standard line search procedure for optimizing continuous functions of one variable, and fix $\delta_i = \overline{\delta}_i$;
3. Iterate Step 2 for each $i \in \{1, 2, \ldots, 2m\}$;
4. Repeat Step 3 till a stopping criterion is satisfied.

As stopping criterion we used a combination of the maximum number of iterations and of the maximum norm of the difference of two subsequent estimates $\overline{\boldsymbol{\delta}}$ (falling below a given threshold). As initial test, at Step 2 we simply adopted a grid search optimization algorithm, where a set of trials is formed for each parameter, and all possible parameter combinations are assembled and tested.

Label inference. Once the parameters $\hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\rho}}$ have been estimated, we consider the subnetwork $\mathcal{H}_{\mathcal{U}} = \langle \boldsymbol{W}_{\mathrm{UU}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\rho}}, \boldsymbol{S} \rangle$ restricted to the unlabeled nodes U, whose energy is

$$E_{\mathcal{H}_{\mathcal{U}}}(\boldsymbol{U}) = \sum_{k=1}^{m} \left(E_U\big(\boldsymbol{u}^{(k)}\big) + \frac{\alpha}{2} \Big(S_k \sum_{i=1}^{h} u_{ik}^2 - \sum_{\substack{r=1 \\ r \neq k}}^{m} s_{kr} \sum_{i=1}^{h} u_{ik} u_{ir} \Big), \right) \tag{11}$$

with $\boldsymbol{U} = (\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(m)})$ state of $\mathcal{H}_{\mathcal{U}}$, $\boldsymbol{u}^{(k)} = (u_{1k},\ u_{2k},\ \ldots, u_{hk}) = (x_{1k},\ x_{2k},\ \ldots, x_{hk}) \in \{\sin \hat{\rho}_k, -\cos \hat{\rho}_k\}^h$, $E_U\big(\boldsymbol{u}^{(k)}\big) = -\frac{1}{2} {\boldsymbol{u}^{(k)}}^T \boldsymbol{W}_{UU} \boldsymbol{u}^{(k)} + {\boldsymbol{u}^{(k)}}^T \overline{\boldsymbol{\theta}}_k$, and $\overline{\boldsymbol{\theta}}_k = \hat{\gamma}_k \boldsymbol{e}_h - W_{\mathrm{UL}} \overline{\boldsymbol{l}}^{(k)}$ is the vector of activation thresholds for task c_k, including the contribution of labeled nodes (which are clamped). In the case the learned parameters make $\bar{\boldsymbol{L}}$ a part of global minimum of $\mathcal{H}$, by determining the global minimum of $\mathcal{H}_{\mathcal{U}}$, it is possible to determine the global minimum of $\mathcal{H}$ (as stated by the following theorem), and consequently the problem solution.

Theorem 3. *Given a multitask Hopfield network $\mathcal{H} = \langle W, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S} \rangle$ on neurons V, bipartitioned into the sets L and U, if $\boldsymbol{L}$ is a part of a global minimum of the energy of $\mathcal{H}$, and $\boldsymbol{U}$ is a global minimum of the energy of $\mathcal{H}_U = \langle W_{UU}, \boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{S} \rangle$, then $(\boldsymbol{L}, \boldsymbol{U})$ is a global minimum of the energy of $\mathcal{H}$.*

On the other side, computing the energy global minimum of $\mathcal{H}_U$ would require time intensive algorithms; to preserve the model efficiency and scalability, we run the dynamics of $\mathcal{H}_U$ till an equilibrium state is reached, which, in general, is an energy local minimum. Given an initial state $\boldsymbol{U}(0)$, at each time t one neuron is updated, and in nm consecutive steps all neurons are updated asynchronously and in a randomly chosen order according to the following update rule:

$$u_{ik}(t+1) = \begin{cases} \sin \hat{\rho}, & \text{if } \phi_{ik}^U(t) > 0 \\ -\cos \hat{\rho}, & \text{if } \phi_{ik}^U(t) \leq 0 \end{cases}, \tag{12}$$

where $u_{ik}(t+1)$ is the state of neuron ik at time $t+1$, and $\phi_{ik}^U(t)$ is the restriction of $\phi_{ik}(t)$ to U. According to Theorem 1, the dynamics (12) converges to an equilibrium state $\bar{\boldsymbol{U}}$ of $\mathcal{H}_U$, and the predicted bipartition $(U_{k,+}, U_{k,-})$ for task k is: $U_{k,+} := \{i \in U | \bar{u}_{ik} = \sin \hat{\rho}\}$ and $U_{k,-} := \{i \in U | \bar{u}_{ik} = -\cos \hat{\rho}\}$.

Dynamics regularization. As shown by [13], the network dynamics might get stuck in trivial equilibrium states when input labeling are highly unbalanced —e.g. states made up by almost all negative neurons. To prevent this behaviour, they applied a dynamics regularization, with the aim to control the number of positive neurons in the current state. By extending that approach, and denoted by $p_{k,+} = \frac{|L_{k,+}|}{|L|}$ the proportion of positives in the training set for task c_k, the following regularization term is added to the energy function $E_{\mathcal{H}_\mathcal{U}}(\boldsymbol{U})$

$$\eta_k \left(\sum_{i=1}^{h} (a_k u_{ik} + b_k) - h p_{k,+} \right)^2 , \tag{13}$$

where $a_k = \frac{1}{\sin\hat{\rho}_k + \cos\hat{\rho}_k}$, $b_k = \frac{\cos\hat{\rho}_k}{\sin\hat{\rho}_k + \cos\hat{\rho}_k}$, and η_k is a real regularization parameter. Since a_k and b_k are such that $(a_k u_{ik} + b_k) = 1$ when $u_{ik} = \sin\hat{\rho}_k$, 0 otherwise, the $\sum_{i=1}^{h}(a_k u_i + b_k)$ is the number of positive neurons in $\boldsymbol{u}^{(k)}$. The term (13) is thereby minimized when the number of positive neurons in $\boldsymbol{u}^{(k)}$ is $hp_{k,+}$. This choice is motivated by the fact that

$$hp_{k,+} = \arg\max_q \; Prob \left\{ |U_{k,+}| = q \;\middle|\; L \text{ contains } |L_{k,+}| \text{ positives} \right\} ,$$

when U and L are randomly drawn from V —see [13]. By simplifying eq. (13), up to a constant terms, we obtain the quadratic term:

$$\eta_k a_k \Big(a_k \sum_{i=1}^{h} \sum_{\substack{j=1 \\ j \neq i}}^{h} u_{ik} u_{jk} + \big(2b_k(h-1) + 1 - 2p_{k,+}\big) \sum_{i=1}^{h} u_{ik} \Big),$$

which can be included into $E_U(\boldsymbol{u}^{(k)}) = -\frac{1}{2} \sum_{i=1}^{h} \sum_{\substack{j=1 \\ j\neq i}}^{h} w_{ij}^{(k)} u_{ik} u_{jk} + \sum_{i=1}^{h} u_{ik} \tilde{\theta}_{ik}$, where $\tilde{\theta}_{ik} = \overline{\theta}_{ik} + \eta_k a_k \left[2b_k(h-1) + (1 - 2p_{k,+}h)\right]$ and $w_{ij}^{(k)} = (w_{ij} - 2\eta_k a_k^2)$. By adding a regularization term for each task c_k, the following energy is derived:

$$E_{\mathcal{H}_\mathcal{U}}(\boldsymbol{U}) = \sum_{k=1}^{m} \Bigg(-\frac{1}{2} {\boldsymbol{u}^{(k)}}^T \boldsymbol{W}_{\mathrm{UU}}^{(k)} \boldsymbol{u}^{(k)} + {\boldsymbol{u}^{(k)}}^T \tilde{\boldsymbol{\theta}}_k + \frac{\alpha}{2} \Big(S_k \sum_{i=1}^{h} u_{ik}^2 - \sum_{\substack{r=1 \\ r\neq k}}^{m} s_{kr} \sum_{i=1}^{h} u_{ik} u_{ir} \Big) \Bigg)$$

Informally, this regularization leads to a different network topology for each task, in addition to a modification of the neuron activation thresholds. Nevertheless, since the connection weights are modified by a constant value, from an implementation standpoint this regularization just need to memorize m different constant values, thus not increasing the space complexity of the model. As preliminary approach, and to have a fair comparison, the parameters η_k have been set as for the singletask case [13], that is $\eta_k = \beta \left| \tan\left((\hat{\rho}_k - \frac{\pi}{4}) * 2 \right) \right|$, where β is a non negative real constant. Another advantage of this choice is that we have to learn just one parameter β, instead of m dedicated parameters.

2.4 Model complexity

The time complexity of HoMTask depends in turn on the computational complexity of the learning procedure and the network dynamics. The learning procedure updates at each iteration $2m$ parameters, and each update requires computing eq. 10 for each of the z possible values of the grid search. Since the labeling is fixed, the weighted sum of positive and negative neighbors can be computed offline, thus the update of ϕ_{ik}^{L} can be performed in constant time, allowing computing eq. 10 in $\mathcal{O}(|L|)$ time. The time complexity of the learning procedure is thereby $\mathcal{O}(mz|L|I)$, where I is the number of iterations to converge (Step 4 of learning procedure). The complexity of the network dynamics depends on the number of iterations needed to converge, and each iteration takes time $\mathcal{O}(m|\boldsymbol{W}_{UU}|)$, where where $|\boldsymbol{T}|$ is the number of non-null entries in the matrix $\boldsymbol{T}$. We empirically observed that the network in average converges in few iterations (less than 10), confirming the notes in [27, 13]. Thus, the overall time complexity is $\mathcal{O}(mz|L|I + m|\boldsymbol{W}_{UU}|)$, which is $\mathcal{O}(mz|L|I + m|U|)$ when the connection matrix is sparse, that is when $|\boldsymbol{W}_{UU}| = \mathcal{O}(|U|)$.

Finally, the space complexity is $\mathcal{O}(nm + n^2)$, deriving from the storage of matrices $\boldsymbol{X}$ and $\boldsymbol{W}$, which becomes $\mathcal{O}(nm)$ when $\boldsymbol{W}$ is sparse.

3 Preliminary results and discussion

In this section we evaluate our algorithm on the prediction of the bio-molecular functions of proteins, a binary classification problem aiming at associating sequenced proteins with their biological functions. Next we describe the experimental setting, analyze the impact on performance of parameter configurations, and we compare HoMTask against other state-of-the-art graph-based methods.

3.1 Benchmark data

In our experiments we considered the Gene Ontology [5] terms, i.e. the reference functional classes in this context, and their annotations to the *Saccaromyces cerevisiae* (yeast) proteins, one of the most studied model organisms. The connection matrix $\boldsymbol{W}$ has been retrieved from the STRING database, version 10.5 [35], and contains 6391 yeast proteins. As common in this context, the GO terms with less than 10 and more than 100 yeast protein annotations (positives) have been discarded, in order to have a minimum of information and to avoid too generic terms —GO is a DAG, where annotations for a term are transferred to all its ancestors. We considered the UniProt GOA (release 87, 12 March 2018) experimentally validated annotations from all GO branches, Cellular Component (CC), Molecular Function (MF) and Biological Process (BP), for a total of 162, 227, and 660 CC, MF, BP GO terms, respectively.

3.2 Evaluation setting

To evaluate the generalization capabilities of our algorithm, we used a 3-fold cross validation (CV), and measured the performance in terms of Area Under the ROC curve (AUC) and Area Under the Precision-Recall curve (AUPR). The AUPR has been adopted in the recent CAFA2 international challenge for the critical assessment of protein functions [25], since in this imbalanced setting AUPR is more informative than AUC [33].

3.3 Model configuration

HoMTask has three hyper-parameters, τ, β and α, and two functions to be chosen: f in eq. (9), and σ in eq. (10). τ, β and α were learned through inner 3-fold CV, considering also the cases α and β in turn or together clamped to $= 0$, to evaluate their individual impact on the performance. A different discussion can be made for the τ parameter, since in our experimentations best performance correspond to large values of τ (e.g. $\tau > 500$), thus making the model less sensitive to this choice (the function f becomes a Heaviside function). This behaviour apparently conflicts with results reported in [16], where typically $0.5 < \tau < 2$ performed best. However, in that work the authors focused on a substantially different learning task, i.e. a singletask Hopfield model, where nodes were divided into categories, and the model parameters were not related to different tasks, but to different node categories. We still include τ in the formalization proposed in Section 2.3 because it permits also future analytic studies about the derivatives of σ, to determine close formulations for the optimal parameters. Further, We set $f(x) = \frac{1}{2}\big(\frac{2}{\pi}\,\mathrm{arctg}(x)+1\big)$, since this choice in a multi-category context leaded to excellent results [16], even if different choices are possible (Section 2.3).

On the other side, we tested two choices for σ: the harmonic mean (σ_1) and mean functions (σ_2). Furthermore, another central factor of our model is the computation of the task similarity matrix $\boldsymbol{S}$, which can be computed by using several metrics (see for instance [15]), and how to group the tasks that should be learned together. We employed in this work the Jaccard similarity measure, since it performed nicely in hierarchical contexts [15, 37, 14], defined as follows:

$$s_{kr} = \begin{cases} \dfrac{\big|L_{k,+} \wedge L_{r,+}\big|}{\big|L_{k,+} \vee L_{r,+}\big|} & \text{if } L_{k,+} \vee L_{r,+} \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Thus, s_{kr} is the ratio between the number of instances that are positive for both tasks and the number of instances that are positive for at least one task. The higher the number of shared instances, the higher the similarity (up to 1); conversely, if two tasks do not share items, their similarity is zero. Due to the numerous experiments to be carried out, just for this analysis the focus is only on CC terms, which are less numerous than those in the MF and BP branches, while showing very similar trends, as shown in the benchmark comparison described in next section. Finally, we grouped tasks by GO branch, and by GO branch and number of positives: in the first case (*Branch*), all tasks within the CC branch are learned simultaneously; in the latter one (*Card*), CC tasks having 10-20 (76 tasks), 21-50 (60), or 51-100 (26) positives have been grouped together. Both approaches are quite usual when predicting GO terms [31, 14].

Table 1. Performance averaged across CC terms for different configuration of the model.

Configuration	AUC	AUPR
Branch, σ_1	0.961	0.439
Card, σ_1	0.959	0.439
Card, σ_1, $\alpha = 0$	0.959	0.431
Card, σ_1, $\beta = 0$	0.810	0.204
Card, σ_1, $\alpha = \beta = 0$	0.811	0.204
Card, σ_2	0.937	0.312

The Table 1 reports the obtained results. First, the two different strategies for grouping tasks led to similar results in this setting, with the *Branch* grouping being experimentally slower because the learning procedure needs more iterations to converge when the number of parameters increases (due to the max norm adopted here as stopping criterion). Nevertheless, we remark that no thresholding on the matrix $\boldsymbol{S}$ has been applied in both cases; thus, in the same model even tasks with small similarities can be included, which in principle might introduce noise in the learning and inference processes. Consequently, the advantage of jointly learning a larger number of similar tasks can be compensated by this potential noise; investigating other task grouping and similarity thresholding strategies could thereby give rise to further insights about model, which for lack of room we destine to future study.

Regarding the impact of parameter β, regulating the effect of dynamics regularization, a strong decay in performance is obtained when no regularization is applied ($\beta = 0$): this confirms the tendency of the network trajectory to be attracted in some limit cases by trivial fixed points, already observed in the singletask Hopfield model [13]. In this experiment, the contribution of regularization is even more dominating, since it allows to double the AUPR performance.

The parameter α, which regulates the multitask contribution in Eq. (3), has apparently less impact on the performance. Indeed, the performance reduces just around 2% when $\alpha = 0$; however, this behaviour should be further studied, because it can be strictly related to the noise we introduced by grouping tasks without filtering out connections between less similar task. Thus, further experiments with different organisms would help this analysis and potentially reveal novel and more clear trends. It is also important noting that by setting $\alpha = 0$, the overall multitask contribution is not cancelled: the learning procedure, by maximizing criterion (10), still learns tasks jointly, even when the multitask contribution in formula (9) is removed. For instance, choosing σ equal to the minimum function would mean learning individual task parameters in order to maximize the minimum performance ($\min_k F_k$) across tasks, even when $\alpha = 0$.

Finally, the function σ itself seems having a marked impact on the model. When using the mean function (σ_2) the AUPR decreases of around 25% with respect to the AUPR obtained using the harmonic mean (σ_1). To some extent such a result is expected, since the harmonic mean tends to penalize more the outliers towards 0, thus fostering the learning procedure to estimate the parameters in order not to penalize some tasks in favors of the remaining ones, which instead can happen when using the mean function. This preliminary model analysis suggested to adopt the configuration "Card, σ_1" in the comparison with the state-of-the-art methodologies, which is described in the next section.

3.4 Model performance

We compared our method with several state-of-the-art graph-based algorithms, ranging from singletask Hopfield networks and other multitask methodologies, to some methods specifically designed to predicting protein functions: *RW, random walk* [30], the classical t-step random walk algorithm, predicting a score corresponding to the probability that a t-step random walk in G, starting from positive nodes, ends in the node to be predicted; *RWR, random walk with restart*, since in RW after many steps the walker may forget the prior information coded in the initial probability vector (0 for negative nodes $1/|L_{k,+}|$ for positive nodes), RWR allows the walker to move another random walk step with probability $1-\theta$, or to restart from its initial condition with probability θ; *GBA, guilt-by-association* [34], a method based on the assumption that interacting proteins are more likely to share similar functions; *LP, label propagation* [43], a popular semi-supervised learning algorithm which propagates labels to unlabeled nodes through an iterative process based on Gaussian random fields over a continuous state space; *MTLP, MTLP-inv* [14], two recent multitask extensions of LP, exploiting task dissimilarities (MTLP) and similarities (MTLP-inv); *MS-kNN, Multi-Source k-Nearest Neighbors* [29], a method based on the k-Nearest Neighbours (kNN) algorithm [1], among the top-ranked methods in the recent CAFA2 international challenge for AFP [25]; *RANKS* [36], a recent graph-based method proposed to rank proteins, adopting a suitable kernel matrix to extend the notion of node similarity also to non neighboring nodes; *COSNet*, employing the neuron internal energy at equilibrium to compute node ranking, in order to properly calculating both AUC and AUPR, as done in [18]. Free parameters for compared methods have been learned through inner 3-fold cross-validation.

In Table 3.4 we show the obtained results. Our method achieves the highest AUPR in all the experiments, with statistically significant difference over the second top method (RWR) in 2 out of 3 experiments (Wilcoxon signed rank test, $\alpha = 0.05$). The performance improvement compared with COSNet is noticeable, showing the remarkable contribution supplied by our multitask extension. Interestingly, MTLP and MTLP-inv increase the AUPR results of LP not so remarkably as HoMTask: this means that the further information regarding task similarities should be appropriately exploited in order to achieve relevant gains. RANKS is the third method in all experiments, followed by MTLP(-inv), while MS-kNN is surprisingly the last method. Our method achieves good results also in terms of AUC (which however is less informative in this context), being close to top performing methods (RWR on CC and MF, and MTLP-inv on BP terms).

Table 2. Performance comparison averaged across GO branches. In bold the top results, underlined when statistically different from the second top result.

	RW	RWR	GBA	LP	MTLP	MTLP-inv	MS-kNN	RANKS	COSNet	HoMTask
						AUC				
CC	0.954	**0.966**	0.944	0.964	0.957	0.964	0.790	0.958	0.904	0.959
MF	0.934	**0.955**	0.931	0.951	0.939	0.953	0.742	0.945	0.859	0.945
BP	0.943	0.959	0.935	0.955	0.947	**0.961**	0.764	0.949	0.855	0.954
						AUPR				
CC	0.367	0.437	0.207	0.308	0.343	0.342	0.218	0.398	0.361	**0.439**
MF	0.199	0.272	0.125	0.201	0.229	0.234	0.090	0.236	0.214	<u>**0.291**</u>
BP	0.244	0.313	0.145	0.224	0.246	0.250	0.116	0.271	0.241	<u>**0.330**</u>

Conclusions

We have proposed the first multitask Hopfield Network for classification purposes, HoMTask, capable to simultaneously learn multiple tasks and to cope with the label imbalance. In our validation experiments, it significantly outperformed singletask HNs, and favorably compared with state-of-the-art single and multitask graph-based methodologies. Future investigations might reveal novel insights about the model, in particular regarding the choice of the task relatedness matrix, the task grouping strategy, the multitask criterion to be optimized during the learning phase, the optimization procedure itself, and the robustness against different proportions of labeled data.

References

1. Altman, N.S.: An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. The American Statistician **46**(3), 175–185 (1992)
2. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. **6**, 1817–1853 (2005)
3. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning **73**(3), 243–272 (2008)
4. Argyriou, A., et al.: A spectral regularization framework for multi-task structure learning. In: Advances in Neural Inf. Proc. Sys. pp. 25–32 (2007)
5. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. the gene ontology consortium. Nature genetics **25**(1), 25–29 (2000)
6. Bertoni, A., Frasca, M., Valentini, G.: COSNet: a cost sensitive neural network for semi-supervised learning in graphs. In: ECML PKDD 2011. Lecture Notes on Artificial Intelligence, vol. 6911, pp. 219–234. Springer (2011). https://doi.org/10.1007/978-3-642-23780-5_24
7. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
8. Chen, J., Zhou, J., Ye, J.: Integrating low-rank and group-sparse structures for robust multi-task learning. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 42–50. ACM (2011)

9. Daumé III, H.: Bayesian multitask learning with latent hierarchies. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 135–142. AUAI Press (2009)
10. Evgeniou, A., Pontil, M.: Multi-task feature learning. Advances in Neural Inf. Proc. Sys **19**, 41 (2007)
11. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. **6**, 615–637 (2005)
12. Evgeniou, T., Pontil, M.: Regularized multi–task learning. In: Proceedings of the Tenth ACM SIGKDD KDD '04. pp. 109–117. ACM (2004)
13. 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)
14. Frasca, M., Cesa-Bianchi, N.: Multitask protein function prediction through task dissimilarity. IEEE/ACM Trans. on Comp. Biology and Bioinf. pp. 1–1 (2018). https://doi.org/10.1109/TCBB.2017.2684127
15. Frasca, M.: Gene2disco: Gene to disease using disease commonalities. Artificial Intelligence in Medicine **82**, 34 – 46 (2017). https://doi.org/10.1016/j.artmed.2017.08.001
16. Frasca, M., Bassis, S., Valentini, G.: Learning node labels with multi-category hopfield networks. Neural Computing and Applications **27**(6), 1677–1692 (2016). https://doi.org/10.1007/s00521-015-1965-1
17. Frasca, M., Bertoni, A., Sion, A.: A Neural Procedure for Gene Function Prediction, pp. 179–188. Neural Nets and Surroundings,Springer Berlin Heidelberg (2013). https://doi.org/10.1007/978-3-642-35467-0_19
18. Frasca, M., Pavesi, G.: A neural network based algorithm for gene expression prediction from chromatin structure. In: IJCNN. pp. 1–8. IEEE (2013). https://doi.org/10.1109/IJCNN.2013.6706954
19. Greene, W.H.: Econometric Analysis. Prentice Hall, 5. edn. (2003)
20. Guo, S., Zoeter, O., Archambeau, C.: Sparse bayesian multi-task learning. In: Advances in Neural Inf. Proc. Sys. pp. 1755–1763 (2011)
21. Hopfield, J.J.: Neural networks and physical systems with emergent collective compatational abilities. Proc. Natl Acad. Sci. USA **79**, 2554–2558 (1982)
22. Hu, X., Wang, T.: Training the hopfield neural network for classification using a stdp-like rule. In: Neural Information Processing. pp. 737–744. Springer (2017)
23. Jacob, L., Vert, J.p., Bach, F.R.: Clustered multi-task learning: A convex formulation. In: Advances in Neural Inf. Proc. Sys. pp. 745–752 (2009)
24. Jacyna, G.M., Malaret, E.R.: Classification performance of a hopfield neural network based on a hebbian-like learning rule. IEEE Transactions on Information Theory **35**(2), 263–280 (March 1989). https://doi.org/10.1109/18.32122
25. Jiang, Y., Oron, T.R., et al.: An expanded evaluation of protein function prediction methods shows an improvement in accuracy. Genome Biology **17**(1), 184 (2016)
26. Kang, Z., Grauman, K., Sha, F.: Learning with whom to share in multi-task feature learning. In: Proceedings of the 28th ICML. pp. 521–528 (2011)
27. Karaoz, U., et al.: Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl Acad. Sci. USA **101**, 2888–2893 (2004)
28. Kordos, M., Duch, W.: Variable step search algorithm for feedforward networks. Neurocomput. **71**(13-15), 2470–2480 (2008)
29. Lan, L., Djuric, N., Guo, Y., S., V.: MS-kNN: protein function prediction by integrating multiple data sources. BMC Bioinformatics **14**(Suppl 3:S8) (2013)
30. Lovász, L.: Random walks on graphs: A survey. In: Miklós, D., Sós, V.T., Szőnyi, T. (eds.) Combinatorics, Paul Erdős is Eighty, vol. 2, pp. 353–398. Budapest (1996)

31. Mostafavi, S., Morris, Q.: Fast integration of heterogeneous data sources for predicting gene function with limited annotation. Bioinfo. **26**(14), 1759–1765 (2010)
32. Ning, X., Karypis, G.: Multi-task Learning for Recommender System. In: Proc. 2nd Asian Conf. on Mac. Lear. (ACML2010). vol. 13, pp. 269–284 (2010)
33. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE **10**, e0118432 (2015)
34. Schwikowski, B., Uetz, P., Fields, S.: A network of protein-protein interactions in yeast. Nature biotechnology **18**(12), 1257–1261 (Dec 2000)
35. Szklarczyk, D., et al.: String v10: protein–protein interaction networks, integrated over the tree of life. Nucleic Acids Research **43**(D1), D447–D452 (2015)
36. Valentini, G., et al.: RANKS: a flexible tool for node label ranking and classification in biological networks. Bioinformatics (2016)
37. Vascon, S., Frasca, M., Tripodi, R., Valentini, G., Pelillo, M.: Protein function prediction as a graph-transduction game. Pattern Recognition Letters (2018)
38. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. J. of Mach. Learn. Res. **8**, 35–63 (2007)
39. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian process from multiple tasks. In: Proc. of the 22nd Int. Conf. on Mac. lear. pp. 1012–1019. ACM (2005)
40. Yu, S., Tresp, V., Yu, K.: Robust multi-task learning with t-processes. In: Proc. of the 24th Int. Conf. on Machine learning. pp. 1103–1110. ACM (2007)
41. Zhang, Y., Schneider, J.G.: Learning multiple tasks with a sparse matrix-normal penalty. In: Advances in Neural Inf. Proc. Sys. pp. 2550–2558 (2010)
42. Zhou, J., Chen, J., Ye, J.: Clustered multi-task learning via alternating structure optimization. In: Advances in Neural Inf. Proc. Sys. pp. 702–710 (2011)
43. Zhu, X., et al.: Semi-supervised learning with gaussian fields and harmonic functions. In: Proc. of the 20th Int. Conf. on Machine Learning (2003)