Propagating Distributions on a Hypergraph by Dual Information Regularization

Koji Tsuda KOJI.TSUDA@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076, Tübingen, Germany, and Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-42 Aomi, Koto-ku, Tokyo, 135-0064, Japan

Abstract

In the information regularization framework by Corduneanu and Jaakkola (2005), the distributions of labels are propagated on a hypergraph for semi-supervised learning. The learning is efficiently done by a Blahut-Arimoto-like two step algorithm, but, unfortunately, one of the steps cannot be solved in a closed form. In this paper, we propose a dual version of information regularization, which is considered as more natural in terms of information geometry. Our learning algorithm has two steps, each of which can be solved in a closed form. Also it can be naturally applied to exponential family distributions such as Gaussians. In experiments, our algorithm is applied to protein classification based on a metabolic network and known functional categories.

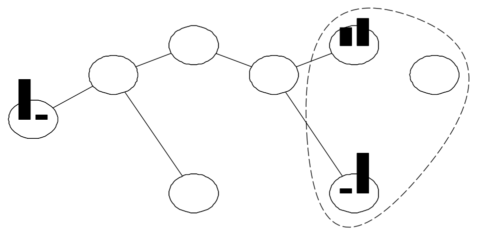
Figure 1. Propagating distributions on a hypergraph. This hypergraph contains three labeled and five unlabeled nodes. Each labeled node has a binomial distribution, which is depicted as a histgram, where two bars indicate the probabilities of positive and negative labels. This hypergraph contains six edges and one hyperedge including three nodes, which is written as a closed contour. Our task is to infer the binomial distributions on unlabeled nodes based on neighborhood relations specified by edges and hyperedges.

1. Introduction

In recent years, we have seen a significant progress of graph-based semi-supervised learning methods in the machine learning community (Zhou et al., 2004; Belkin & Niyogi, 2003; Zhu et al., 2003; Chapelle et al., 2003). In a typical setting, data points are represented as an undirected graph. For a limited number of nodes, their *labels* are known (i.e. labeled nodes). Our task is to infer the labels of the remaining *unlabeled nodes*. Intuitively, it is done by propagating labels from labeled to unlabeled nodes (i.e. *label propagation*). In binary classification problems, the label is either $+1$ or -1, and, in regression problems, it is real-valued. The main assumption is that an edge represents association of two data points, thus the labels of two adjacent nodes are likely to be the same. Each edge can have a positive weight, representing the degree of association. Typically, a regularization term is designed using a graph Laplacian matrix, and it is minimized together with a loss term which corresponds to the discrepancy between predictions and labels at labeled nodes.

Recently, Corduneanu and Jaakkola (2005) addressed a more general semi-supervised learning problem, where 1. a deterministic label is generalized to a distribution of labels, and 2. distributions are propagated on a *hypergraph*. A typical problem is depicted in Figure 1. Using distributions, *uncertainty* of labels can be described, for example, the probabilities of label A and B are 0.7 and 0.3, respectively. In regression, one can employ Gaussian distributions to augment the prediction with an error bar. A hypergraph has *hyperedges*, which can connect two or more nodes, instead of conventional edges connecting two nodes. In comparison

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

to conventional graphs, hypergraphs have more flexibility in describing prior knowledge, because known clusters can be directly encoded as hyperedges. It might be possible to convert a hypergraph to a conventional graph, for example, by converting a hyperedge to a complete subgraph. However, one drawback is that many edges are additionally introduced, increasing the computational cost.

For semi-supervised learning on the hypergraph, Corduneanu and Jaakkola (2005) proposed a regularization term based on the Kullback-Leibler (KL) divergences among distributions in a hyperedge. Then, the distributions are propagated by minimizing the regularization term together with the likelihood functions on labeled nodes, which is done efficiently by a two-step optimization algorithm like the Blahut-Arimoto algorithm (Cover & Thomas, 1991). But, one drawback of this approach, called *distributed information regularization*, is that the optimal solution of one step cannot be solved analytically even for simple multinomial distributions. So one has to rely on an iterative algorithm such as Newton's method. Also, when exponential family distributions are employed, the algorithm has a non-convexity problem inside a step.

In this paper, we propose a dual version of their regularization term, which is considered to be more natural in terms of information geometry (Amari & Nagaoka, 2000). We obtain a similar alternation algorithm, where the optimal solution of each step can be found analytically for general exponential family distributions, including multinomial and Gaussian distributions. We applied our method to the classification of proteins on a metabolic network (Tsuda & Noble, 2004). In prediction accuracy, our method was comparable to that of Corduneanu and Jaakkola (2005). Furthermore, our method was faster in computation time due to the simplicity of our learning algorithm.

The paper is organized as follows: In Section 2, we introduce the basic definitions about information geometry. In Section 3, we first review the regularizer by Corduneanu and Jaakkola (2005) and then propose our dual regularizer. Section 4 explains how to propagate the exponential family distributions on a hypergraph. In Section 5, the two regularizers are compared in protein classification experiments. We conclude in Section 6 with discussions and future works.

2. Preliminaries

Let us start from introducing the exponential family distributions and the *center* of distributions in terms of the Kullback-Leibler divergence.

2.1. Exponential Family

A distribution $q(y)$ defined on a set $\mathcal{Y}$ belongs to the *exponential family* iff it can be written in the following form (Amari & Nagaoka, 2000),

$$q(y) = \frac{1}{Z}\exp\left(\sum_{j=1}^{J}\theta_j\phi_j(y)\right). \tag{1}$$

where $\phi_j : \mathcal{Y} \to \Re$ is called the *sufficient statistics*, and the normalization factor is written as

$$Z = \sum_{y\in\mathcal{Y}}\exp\left(\sum_{j=1}^{J}\theta_j\phi_j(y)\right).$$

In information geometry, one distribution is described in two coordinate systems. In the e-affine coordinate system, q is represented by the *natural parameters* θ_j. On the other hand, in the m-coordinate system, it is represented by *expectation parameters*,

$$\eta_j = E_q[\phi_j(y)] = \sum_{y\in\mathcal{Y}} q(y)\phi_j(y),$$

where $E_q[\cdot]$ means the expectation with respect to q. These dual coordinate systems will play an essential role in the learning algorithm introduced in Section 4.

2.2. Center of Distributions

The discrepancy of two distributions p, q is commonly described by the Kullback-Leibler divergence,

$$D(p,q) = \sum_{y\in\mathcal{Y}} p(y)\log\frac{p(y)}{q(y)},$$

which is not symmetric, i.e $D(p,q) \neq D(q,p)$. Let $q_1, \cdots, q_n$ be arbitrary distributions defined on $\mathcal{Y}$. Using the KL divergence, the *center* of distributions can be formulated in two ways:

$$q^E = \mathrm{argmin}_q \sum_{i=1}^{n} a_i D(q, q_i), \tag{2}$$

$$q^M = \mathrm{argmin}_q \sum_{i=1}^{n} a_i D(q_i, q) \tag{3}$$

where $\{a_i\}_{i=1}^n$ are predetermined nonnegative constants summing to 1 (i.e., $\sum_{i=1}^n a_i = 1$). Solving the minimization problems, we get the following (see e.g. Akaho, 2004),

$$q^E(y) = \frac{1}{Z}\exp\left(\sum_i a_i \log q_i(y)\right), \tag{4}$$

$$q^M(y) = \sum_i a_i q_i(y), \tag{5}$$

where Z is the normalization factor. Those center distributions are called the *exponential center* and the *mixture center*, respectively. Notice that, if $q_1, \dots, q_n$ belong to the exponential family, q^E also belongs to the exponential family, but q^M not.

3. Propagation of Distributions

In this section, we present two methods to propagate distributions on a hypergraph. First, the information regularization method by Corduneanu and Jaakkola (2005) is reviewed, and then our new method will be presented as its dual version.

Let us assume a hypergraph with n nodes. Denote by $\mathcal{Y}$ the set of ℓ ($< \infty$) labels. In this section, a node distribution is defined as a *multinomial distribution* $q_i(y)$, which is an ℓ dimensional vector,

$$(q_i(y=1), q_i(y=2), \dots, q_i(y=\ell))^\top,$$

of nonnegative elements summing to one. It can be used in a classification problem with ℓ classes. The multinomial distribution belongs to the exponential family, but it has additional favorable properties such as the joint convexity of the divergence (Cover & Thomas, 1991). General exponential family distributions including Gaussians will be considered in the next section.

Each of the labeled nodes has a distribution $p_i(y), y \in \mathcal{Y}$. Hyperedges are denoted as $R_1, \cdots, R_m$, each of which is a set of node indices. Notice that these node sets are called *regions* in (Corduneanu & Jaakkola, 2005). We will derive the predicted distributions $q_i(y)$ for unlabeled nodes based on the given distributions $p_i(y)$ and the neighborhood relations R_k. Our assumption is that, inside each R_k, distributions $q_i(y)$ should be similar to each other.

3.1. Mixture-type Information Regularization

To encode our assumption, let us design a regularizer function which tends to be small if the assumption is satisfied. Denote by λ_k the nonnegative weight of R_k. Corduneanu and Jaakkola (2005) proposed the following regularizer,

$$\sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_i, q_k^M), \tag{6}$$

where $q_k^M(x)$ is the mixture center of R_k,

$$q_k^M(y) = \frac{1}{|R_k|} \sum_{i \in R_k} q_i(y). \tag{7}$$

We call it the *mixture-type information regularizer* or m-regularizer in short. To obtain the distributions $q_k(x)$, the following optimization problem is solved.

$$\begin{aligned} \mathrm{argmin}_{q_i} \quad & -\sum_{i=1}^{n} w_i \sum_{y \in \mathcal{Y}} p_i(y) \log q_i(y) \\ & + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_i, q_k^M). \end{aligned}$$

The first term consists of the negative likelihood of $q_i(y)$, when $p_i(y)$ is regarded as the sample distribution on node i. The parameter w_i indicates whether the i-th node is labeled or not, i.e., $w_i = 1$ if node i is labeled and $w_i = 0$ otherwise. Notice that this problem is equivalently rewritten as

$$\mathrm{argmin}_{q_i} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_i, q_k^M). \tag{8}$$

using the KL divergences only.

The optimization problem (8) is efficiently solved by an alternating two-step learning algorithm. The key observation is that the problem can be relaxed as

$$\mathrm{argmin}_{q_i, h_k} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_i, h_k). \tag{9}$$

by replacing q_k^M with a free distribution h_k. The objective function in (9) is jointly convex with respect to h_k and q_i (Corduneanu & Jaakkola, 2005). Furthermore, the solution of the relaxed problem agrees with that of (8), because q_k^M is the solution of the following subproblem,

$$\mathrm{argmin}_{h_k} \sum_{i \in R_k} D(q_i, h_k). \tag{10}$$

The learning algorithm consists of the following two steps. In the first step, all node distributions q_i are fixed, and h_k is obtained as (7). In the second step, the center distributions h_k are fixed, and q_i is obtained as

$$\mathrm{argmin}_{q_i} w_i D(p_i, q_i) + \sum_{\{k : i \in R_k\}} \lambda_k D(q_i, h_k), \tag{11}$$

where $\{k : i \in R_k\}$ is the index set of R_k's that include the i-th node. These two steps are repeated until convergence. Unfortunately, the solution of (11) cannot be obtained in a closed form. But one can use any iterative optimization method, e.g. Newton's method, to minimize (11).

3.2. Exponential-type Information Regularization

In this paper, we propose to use another regularizer,

$$\sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_k^E, q_i), \tag{12}$$

where $q_k^E(x)$ is the *exponential center* of node distributions,

$$q_k^E(y) = \frac{1}{Z_k} \exp\left(\frac{1}{|R_k|} \sum_{i \in R_k} \log q_i(y) \right). \tag{13}$$

It is called the *exponential-type information regularizer*, or e-regularizer. When the e-regularizer is used, the optimization problem is written as

$$\mathrm{argmin}_{q_i} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_k^E, q_i). \tag{14}$$

As in the previous section, one can relax it as

$$\mathrm{argmin}_{q_i, h_k} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(h_k, q_i), \tag{15}$$

where we replaced the central distribution q_k^E by a new distribution h_k. The optimal solution of (15) agrees with that of (14). The relaxed problem is solved by the following two-step learning algorithm. In the first step, the central distributions are updated as

$$\mathrm{argmin}_{h_k} \sum_{i \in R_k} D(h_k, q_i), \tag{16}$$

which is solved as (13). In the second step, the node distributions are obtained as

$$\mathrm{argmin}_{q_i} \, w_i D(p_i, q_i) + \sum_{\{k: i \in R_k\}} \lambda_k D(h_k, q_i). \tag{17}$$

This optimization problem is solved as

$$q_i(x) = \frac{1}{w_i + d_i} (w_i p_i(x) + \sum_{\{k: i \in R_k\}} \lambda_k h_k(x)).$$

where $d_i = \sum_{\{k: i \in R_k\}} \lambda_k$.

Unlike the m-regularization, the e-regularization has the desirable property that the second step (17) can be solved in a closed form, because the directions of divergences are aligned. Namely, the variable q_i is in the second argument of each divergence. In the m-regularizer (11), q_i appears both in first and second arguments, making the problem more difficult to solve.

3.3. Convexity

Theorem 1. *The e-regularizer (12) is convex with respect to node distributions q_i.*

Proof. To prove the theorem, it is sufficient to prove the convexity of each component in (12),

$$\sum_{i \in R_k} D(q_k^E, q_i).$$

As mentioned in Section 3.2, it is written as the minimum of the following function,

$$\min_h \sum_{i \in R_k} D(h, q_i). \tag{18}$$

So it is sufficient to show that (18) is jointly convex with respect to h and all q_i's. By Theorem 2.7.2 in (Cover & Thomas, 1991), $D(h, q_i)$ are jointly convex, hence the sum of divergences (18) is also jointly convex. □

4. Learning Algorithm for Exponential Family

In the following, we deal with the cases that the distributions $\{q_i\}_{i=1}^n$ are exponential family distributions with common sufficient statistics $\{\phi_j(y)\}_{j=1}^J$, which include important models such as Gaussian, Boltzmann, etc (Amari & Nagaoka, 2000). Denote by $\mathcal{S}$ the set of all possible exponential family distributions with $\{\phi_j(y)\}_{j=1}^J$.

Using the e-regularizer, the learning problem is written as

$$\mathrm{argmin}_{q_i \in \mathcal{S}} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(q_k^E, q_i),$$

with additional constraints $q_i \in \mathcal{S}$. As in the multinomial cases, we can relax it as

$$\mathrm{argmin}_{q_i \in \mathcal{S}, h_k \in \mathcal{S}} \sum_{i=1}^{n} w_i D(p_i, q_i) + \sum_{k=1}^{m} \lambda_k \sum_{i \in R_k} D(h_k, q_i),$$

where the centers h_k also belong to $\mathcal{S}$. Thus we have the following two steps,

$$\mathrm{argmin}_{h_k \in \mathcal{S}} \sum_{i \in R_k} D(h_k, q_i), \tag{19}$$

$$\mathrm{argmin}_{q_i \in \mathcal{S}} \, w_i D(p_i, q_i) + \sum_{\{k: i \in R_k\}} \lambda_k D(h_k, q_i) \tag{20}$$

Using the dual coordinates, the solution of the first step (19) is

$$\theta_{kj}^E = \frac{1}{|R_k|} \sum_{i \in R_k} \theta_{ij}, \tag{21}$$

where θ_{ij} and θ^E_{kj} are the natural parameters of q_i and h_k, respectively. Also, the solution of the second step (20) is

$$\eta_{ij} = \frac{1}{w_i + d_i}(w_i \eta^p_{ij} + \sum_{k|i \in R_k} \lambda_k \eta^E_{kj}), \quad (22)$$

where η_{ij}, η^p_{ij} and η^E_{kj} are the expectation parameters of q_i, p_i and h_k, respectively. Therefore, the two steps of our algorithm are equivalent to weighted averages in dual representations.

To apply our algorithm, one needs to transform one representation to its dual. For example, in the first step, the natural parameters θ^E_{kj} are obtained, but the dual parameters η^E_{kj} are needed in the next step. This coordinate transformation is called *Legendre transform* (Amari & Nagaoka, 2000). Therefore, our algorithm contains the following four steps.

1. Computation of the natural parameters of the central distributions θ^E_{kj} as in (21)
2. Legendre transform from θ^E_{kj} to η^E_{kj}.
3. Computation of the expectation parameters of the node distributions η_{ij} as in (22).
4. Legendre transform from η_{ij} to θ_{ij}.

4.1. Gaussian Distribution

The Legendre transform is not always obtained in a closed form. However, in several cases, especially for Gaussian distributions, the Legendre transform boils down to simple computations. Let us consider the following Gaussian distribution with mean μ and standard deviation σ,

$$q(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}).$$

It can be rearranged in the form of exponential family (1) with sufficient statistics

$$\phi_1(x) = x^2, \quad \phi_2(x) = x,$$

and natural parameters,

$$\theta_1 = -\frac{1}{2\sigma^2}, \quad \theta_2 = \frac{\mu}{\sigma^2}.$$

Also, the expectation parameters are written as

$$\eta_1 = \mu^2 + \sigma^2, \quad \eta_2 = \mu.$$

Legendre transforms between natural and expectation parameters can thus be easily done through the conventional parameters μ and σ.

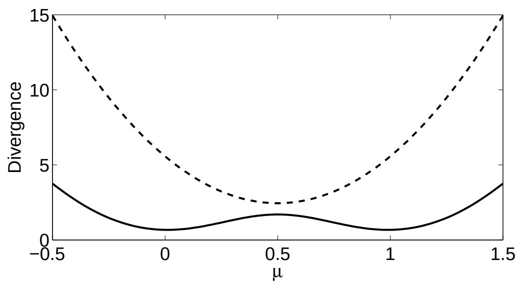

Figure 2. Nonconvexity of the m-regularizer. The solid curve describes the divergence $D(q,h)$ between a Gaussian mixture h and a Gaussian distribution q with variable center μ. The dashed curve shows the flipped version $D(h,q)$. See the text for details.

4.2. Convexity

When exponential family distributions are introduced, the e-regularizer (12) is no longer guaranteed to be convex, though each step is formulated as a convex optimization and therefore solved in a closed form. This situation happens also in the *em* algorithm (Amari, 1995), where a nonconvex function is minimized by alternating convex optimization steps.

If the m-regularizer term is used for exponential family distributions, the second step (11) is non-convex, making the optimization even more difficult. Suppose there is only one edge including two nodes with Gaussian distributions. The first likelihood term in (9) is always convex, but the m-regularizer $\sum_{i \in R_k} D(q_i, h_k)$ is not convex with respect to parameters of q_i. Let us determine h as the mixture of two Gaussians with mean 0 and 1, respectively, and standard deviation 0.2. The solid curve in Figure 2 plots $D(q,h)$ where q is a Gaussian with the standard deviation 0.2 with variable center μ. Notice that the flipped version $D(h,q)$ is convex as shown in the dashed curve.

If $q_i(y)$ is a mixture model,

$$q_i(y) = \sum_{j=1}^{J} \theta_{ij} h_j(y),$$

where h_j is a fixed distribution without any parameter, the situation is the opposite. For the m-regularizer, both steps are convex, while the e-regularizer suffers from the non-convexity problem.

4.3. Illustrative Example

Figure 3 shows an example of propagation of Gaussian distributions. The upper figure shows the initial setting with two labeled nodes, which have Gaussian distributions with means 0 and 1, respectively, and

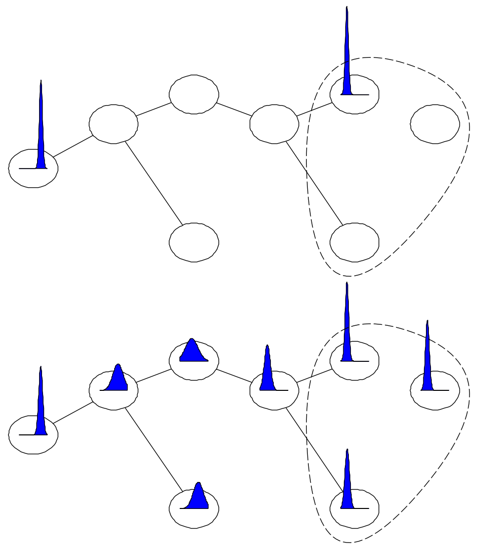

Figure 3. Propagation of Gaussian distributions. Upper and lower figures correspond to the initial and final status, respectively. See the text for details.

standard deviation 0.2. The lower figure shows the result after convergence. From one labeled node to the other, the mean is gradually changed. In the nodes distant from labeled nodes, the standard deviation tends to be large, showing uncertainty of predictions.

5. Experiments

We will compare the prediction performance of the exponential-type and mixture-type regularization using a protein classification dataset.

5.1. Dataset Description

The "metabolic network" dataset available at `noble.gs.washington.edu/proj/maxent/` contains 755 proteins, and each protein has 36 different binary labels. Each label indicates whether the protein belongs to a certain *functional category* or not (Tsuda & Noble, 2004). In this dataset, the relations among proteins are represented by the *metabolic network* containing 7860 edges. The network was derived by Vert and Kanehisa (2003) from the LIGAND database of chemical reactions in biological pathways (`www.genome.ad.jp/ligand`). In this network, two proteins are linked if they catalyze two successive reactions, in which the primary product of the first reaction is the primary substrate of the second reaction. Thus, a path in this graph represents a possible series of reactions catalyzed by proteins along the path.

The nodes (i.e. proteins) are divided into 50% training and 50% test nodes. Using the exponential-type and mixture-type regularizers, the labels of the test nodes are predicted. Since we solve a binary classification problem with respect to each label, the label set $\mathcal{Y}$ is discrete and contains only two elements. In a labeled node, the probability of the given label is set to 0.99, and the opposite label 0.01. We did not use 0/1 probabilities to avoid the computation of $\log 0$. The prediction accuracy for each label is measured by the ROC score. So, each regularizer is evaluated by a vector of 36 ROC scores.

5.2. Adding Clusters

The metabolic network of this dataset contains binary edges only. However, to test our methods in various settings, it would be meaningful to use a hypergraph with clusters (i.e. hyperedges containing more than three nodes) as well. So, in prediction of a specific binary label, we created clusters using the other 35 labels. From each label, a cluster including all the positive nodes is made and added to the hypergraph. However, we found that the proteins are classified almost perfectly, if clusters are made from all the remaining labels. To make the task difficult, we chose the remaining labels whose correlation coefficients with the target label are below 0.3. [1]

In the following experiments, the regularization parameters λ_k were set to constant λ for all hyperedges. We tried five different values $\lambda = \{0.001, 0.01, 0.1, 1, 10\}$, and recorded the best ROC score.

5.3. Results

Figure 4 shows the standard box plots of ROC scores over all 36 functional categories. This experiment was performed in three different settings: In 'Combined', the network was combined with the clusters, whereas in 'Network only' and 'Clusters only', only one of them was used for constructing the hypergraph. Comparing the two regularizers (i.e. exponential-type and mixture-type), the difference in box plots is not large for all three cases, but, in 'Clusters only', our exponential-type regularizer performed slightly worse. In fact, the p-values of Wilcoxson signed rank test were 0.118, 0.196 and 1.05e-05 respectively for 'Combined', 'Network only' and 'Clusters only'. So the difference of medians is statistically significant in the third case

[1]The correlation coefficient of two labels are computed as follows: Let $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ be two vectors of class labels where positive labels correspond to one and negative labels zero. Then, the correlation coefficient is defined as $\boldsymbol{y}_1^\top \boldsymbol{y}_2 / (\|\boldsymbol{y}_1\| \|\boldsymbol{y}_2\|)$.

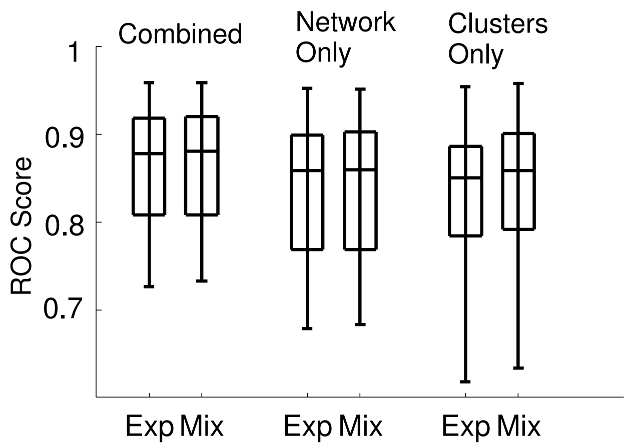

Figure 4. Box plots of the ROC scores for function category prediction of proteins. The box plot notation specifies marks at 95,75,50,25 and 5 percentiles of values. The mixture-type regularization ('Mix') is compared with the exponential-type regularization ('Exp') in three different settings: In 'Combined', the hypergraph was constructed from the network and the clusters. In 'Network only' and 'Clusters only', only one of the two sources was used.

only. In 'Clusters only', the number of hyperedges is less than 35, which is much smaller than the other two cases (more than 7000). Therefore, from our results, it is suggested that the choice of the regularizer is not crucially important when the nodes are densely connected. We have shown the comparison plots as well in Figure 5.

The advantage of our exponential-type regularizer lies in the simplicity of the learning algorithm. Its computation time was fast enough for practical use. At $\lambda = 0.1$, the average number of iterations was 35.5, and, in a standard PC with 2.4Ghz CPU, the time per iteration was 0.57 second on average. On the other hand, the mixture-type regularizer took 9.73 second per iteration using MATLAB's commmand `fminbnd` for solving (11).

6. Discussion

We have presented two dual regularizers for semi-supervised learning on a hypergraph. We conclude this paper with discussions from viewpoints of machine learning and computational biology.

6.1. Regularization vs. Joint Modeling

Our goal was to predict the distributions at unlabeled nodes based on the similarity relations represented by a hypergraph. In principle, this goal can also be achieved by modeling the joint distribution of all nodes. For example, in the Gaussian process, the covariance among all Gaussian variables are predetermined by the kernel matrix (Williams & Barber, 1998). Fixing the variables at labeled nodes, the marginal distributions of unlabeled nodes can be derived simply as the conditional distributions. In comparison to our regularization approach, the joint modeling is conceptually clearer, because it models the complete probabilistic mechanism of data generation. However, our concern is *overmodeling*, i.e., when unnecessary parts of the probabilistic mechanism are constrained to parametric models, it introduces additional risk of model misspecification. Since our primary aim is to obtain the distributions on the unlabeled nodes, we do not need to have a joint distribution. Also, when a hypergraph is given *a priori* as in our experiments, it is easier to directly describe relationships among individual variables, rather than trying to construct a joint model. Intuitively, our regularizers implicitly constrain the joint distribution, but their theoretical properties need to be investigated further.

6.2. Applications in Computational Biology

In computational biology, it is increasingly common that relationships among proteins are represented as *protein networks*. In such a network, nodes represent genes or proteins, and edges may represent physical interaction of the proteins (von Mering et al., 2002), gene regulatory relationships (Lee et al., 2002), similarities between protein sequences (Yona et al., 1999), etc. The metabolic network used in our experiments is just one of them. Our method can be applied to any type of protein networks to predict various properties of proteins (e.g., subcellular localization).

6.3. Future Directions

We used our distribution propagation algorithm for semi-supervised learning, but it can also be applied for *ranking* on a hypergraph (Weston et al., 2004). Also, it would be interesting to apply our algorithm to vectorial data to exploit manifold structures, because our hypergraph representation will provide additional flexibility in manifold modeling.

Acknowledgments

KT would like to thank B. Schölkopf, D. Zhou and M. Wu for fruitful discussions.

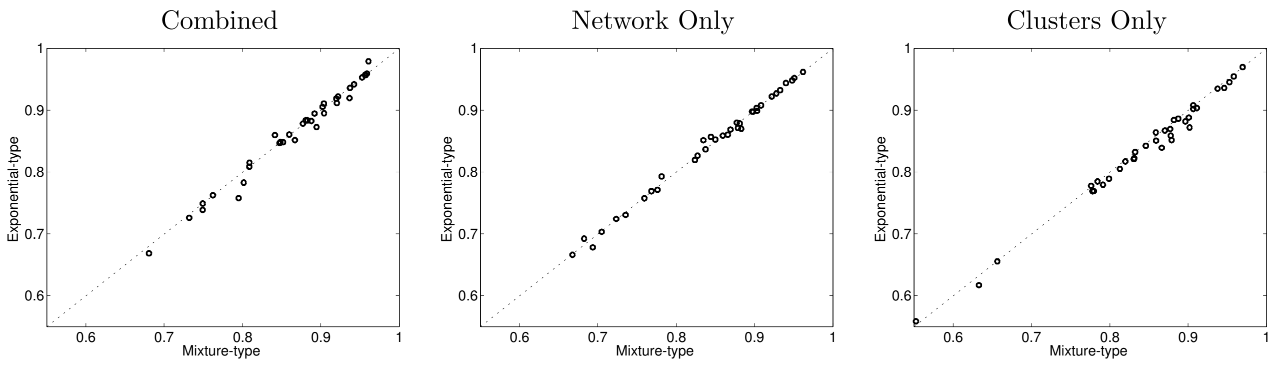

Figure 5. Individual comparison of the ROC scores.

References

Akaho, S. (2004). The e-PCA and m-PCA: Dimension reduction of parameters by information geometry. *Proceedings of 2004 IEEE International Joint Conference on Neural Networks* (pp. 129–134).

Amari, S. (1995). Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, *8*, 1379–1408.

Amari, S., & Nagaoka, H. (2000). *Methods of information geometry.* Oxford: Oxford University Press.

Belkin, M., & Niyogi, P. (2003). Using manifold structure for partially labelled classification. *Advances in Neural Information Processing Systems (NIPS) 15.* MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems (NIPS) 15* (pp. 585–592). MIT Press.

Corduneanu, A., & Jaakkola, T. (2005). Distributed information regularization on graphs. *Advances in Neural Information Processing Systems 17* (pp. 297–304). MIT Press.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory.* New York: Wiley.

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., Harbison, C. R., Thompson, C. M., Simon, I., Zeitlinger, J., Jennings, E. G., Murray, H. L., Gordon, D. B., Ren, B., Wyrick, J. J., Tagne, J., Volkert, T. L., Fraenkel, E., Gifford, D. K., & Young, R. A. (2002). Transcriptional regulatory networks in *Saccharomyces cerevisiae. Science*, *298*, 799–804.

Tsuda, K., & Noble, W. (2004). Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, *20*, i326–i333.

Vert, J., & Kanehisa, M. (2003). Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. *Advances in Neural Information Processing Systems 15* (pp. 1425–1432). MIT Press.

von Mering, C., Krause, R., Snel, B., Cornell, M., Olivier, S. G., Fields, S., & Bork, P. (2002). Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, *417*, 399–403.

Weston, J., Elisseeff, A., Zhou, D., Leslie, C., & Noble, W. (2004). Protein ranking: from local to global structure in the protein similarity network. *Proc. Nat. Acad. Sci. USA*, *101*, 6559–6563.

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with gaussian processes. *IEEE Trans. PAMI.*, *20*, 1342–1351.

Yona, G., Linial, N., & Linial, M. (1999). Protomap: Automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins: Structure, Function, and Genetics*, *37*, 360–678.

Zhou, D., Bousquet, O., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *Advances in Neural Information Processing Systems (NIPS) 16* (pp. 321–328). MIT Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *Proc. of the Twentieth International Conference on Machine Learning (ICML)* (pp. 912–919). AAAI Press.