

Graph-Based Semi-Supervised Learning

Mark Culp, George Michailidis

Abstract—Graph-based learning provides a useful approach for modeling data in classification problems. In this modeling scenario, the relationship between labeled and unlabeled data impacts the construction and performance of classifiers, and therefore a semi-supervised learning framework is adopted. We propose a graph classifier based on kernel smoothing. A regularization framework is also introduced, and it is shown that the proposed classifier optimizes certain loss functions. Its performance is assessed on several synthetic and real benchmark data sets with good results, especially in settings where only a small fraction of the data are labeled.

Index Terms—Semi-supervised Graph-based Learning, Constraint Loss Optimization

I. INTRODUCTION

LEARNING in the *semi-supervised* context has received a lot of attention over the past few years [1; 2; 3; 4]. The main idea is that labeled observations (cases) are hard and/or expensive to acquire, whereas unlabeled ones are easy and inexpensive. The task of semi-supervised learning algorithms is to use the information available in labeled data, in conjunction with their relationship to unlabeled data, in order to complete the labeling of the data at hand.

Several aspects of semi-supervised learning have been addressed in the literature; for a comprehensive survey, see [4] and references therein. One direction of work involves approaches based on extensions of the EM algorithm from the supervised to the present setting [5]; these techniques rely on the concept of *self training*. In this setting, the classifier first concentrates on unlabeled cases that can be learned with high confidence, and then iteratively adds these estimates to the labeled set. After a prespecified number of iterations the algorithm provides a learner that has trained from both the labeled data and confident unlabeled data. Along similar lines is extending the boundary established by a supervised learner to a semi-supervised setting, such as the naive Bayes [1] and the EM algorithms [5]. One technique, in particular, is the Transductive SVM, which extends the supervised SVM to the semi-supervised setting. The rationale is based on the cluster assumption, that is the minimal generalization error is on a border established between highly dense unlabeled regions, and the challenge is finding it [4].

More recently, the focus has shifted towards algorithms that utilize the graph structure obtained by capturing pairwise similarities between the labeled and unlabeled cases [4]. The main idea is to cut the minimum number of edges between vertices (cases) such that each disjoint graph cluster has only one class in it. Upon termination this algorithm classifies (hard labels)

each unlabeled case, while the known labeled cases retain their original labels, which is known as *clamping* [4; 6]. The idea behind clamping is that the output of the algorithm needs to exhibit perfect agreement with the training data. To extend this technique, the proposed methodology borrows ideas from physics, by treating the graph as an electric circuit, with the labeled nodes representing positively or negatively charged sources. The objective is to optimize an energy function that labels all nodes; for example, the above approach corresponds to the absolute energy optimization function [4].

The label propagation algorithm [6] provides a softer approach, where the fitted probability class estimate of each vertex (as opposed to labels) propagates to neighboring vertices while the originally labeled vertices (sources) are clamped. The Gaussian harmonic field extension uses the Laplacian matrix [7] of the graph (where the off-diagonal elements correspond to the negative edge weights, while the diagonal ones are the degrees of the vertices of the graph), and the resulting optimization problem is solved by a generalized eigenvalue decomposition of this matrix. The eigenvectors of the Laplacian matrix correspond to the smooth functions of each vertex that optimize a quadratic energy function [8]. The Laplacian matrix can be thought of as a way of regularizing the problem at hand, an approach adopted by other more recent techniques, such as the regularized propagation approach in [7] or the spectral graph transducer in [2]. In practice, the proximity graph is typically constructed from a similarity measure on the data, which requires tuning. One approach to address this key issue is to add several Laplacian matrices from the tuned proximity graphs in a regularized setting [3; 9].

An advantage of working with a graph structure is its ability to naturally incorporate diverse types of information and measurements. In the area of multi-view learning [1; 2], the objective is to incorporate several distinct views for prediction purposes, where a view is a set of variables that summarizes one particular source of data. For example, consider a data set stemming from pharmacology [10], involving the study of adverse side effects of chemical compounds that are described by their chemical descriptors (in the form of binary features) and biological descriptors (noisy numerical features). The graph-based learning approaches discussed above have been successfully applied to such multi-view learning problems.

In this paper, we propose an algorithm for semi-supervised learning that employs both self-training and regularization. It utilizes the information provided by the labeled data, as well as the relationship between labeled and unlabeled cases. The algorithm relies on the fact that neighbor cases should belong to the same class, and the relationships between data points are captured in the form of a similarity graph with vertices corresponding to the cases and edge weights to their similarity. Specifically, we introduce a graph-based classifier that weighs

Mark Culp is an Assistant Professor in the Department of Statistics at West Virginia University. George Michailidis is an Associate Professor in the Department of Statistics at The University of Michigan.

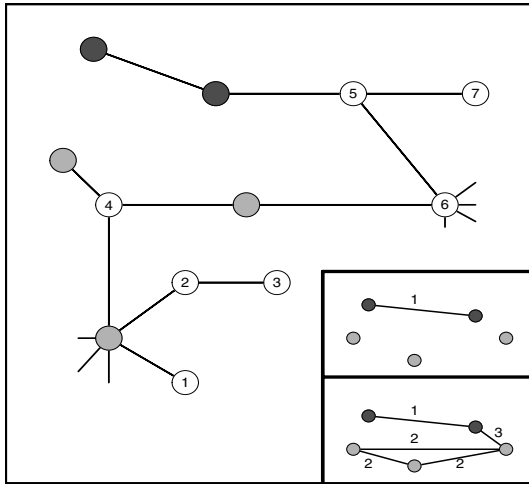


Fig. 1. The plot is a cross-section of a graph, where the labels are either light or dark gray and the unlabeled cases are given as 1 to 7. The shortest path distances are provided in the lower right when not accounting for the unlabeled cases and when accounting for the unlabeled cases. The mere act of constructing a learner from a graph places the problem in the transductive setting, since the unlabeled data fundamentally affect the underlying topology.

the neighboring information available using a kernel function. It also employs a penalty term that ‘corrects’ the predicted labels of cases farther away from labeled data towards the prior distribution of each class. The rationale for this feature is that as the classifier moves away from the labeled cases, uncertainty about predictions increases and the class prior becomes more informative. Another key aspect of the proposed algorithm is that it applies to both class (hard) labels and probability class estimates (soft labels). A connection between the type of loss function involved for soft labels (squared error) and hard ones (exponential or logistic) is also established. This approach is considered both in the multi-view setting and classical proximity graph setting with competitive results to the energy optimization approaches given above. In addition the bias/variance trade-off for this classifier as it relates to regularization is quantitatively assessed.

The remainder of the paper is organized as follows: the problem is formulated in Section III, while the proposed sequential predictions algorithm is introduced in Section IV. Section V assesses the performance of the algorithm on a number of synthetic and real data sets. Some concluding remarks are drawn in Section VI.

II. TRANSDUCTIVE LEARNING ON GRAPHS

In a Euclidean learning problem it is common to observe a data matrix X with n cases and p features, and the interest is in constructing a learner on X to establish a rule for the response Y . Learning in this paradigm relies on the following rationale: if one restricts the construction of a learner to a subset of the data (e.g. the training set) then the processing of new cases (i.e. predicting) preserves the distances/associations between all n cases. When deriving a learner on Euclidean space, this realization equates to the notion that one can easily train on the available data and predict unlabeled data without worrying about updating the learner when a new unlabeled case is processed.

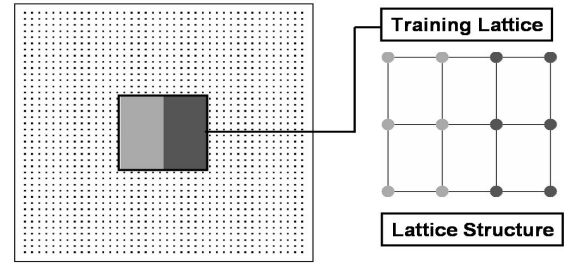


Fig. 2. Illustration of the simple lattice, where the center cases are labeled while the cases off center are unlabeled.

For graphs the problem is inherently more complex as it relates to proximity. An unlabeled/testing case provides intrinsic structural information, and its presence should be accounted for, even though the corresponding label is unknown. For example, consider defining the shortest path distance metric on a *simple* graph, which gives the number of links between two vertices. Figure 1 (a) illustrates the fact that unlabeled cases (case 4 in the figure) can influence the distances between almost all other labeled and unlabeled cases. This association between labeled and unlabeled data greatly affects how one builds a learner on a graph. The goal is to use as much of the underlying graph topology as possible to make associations that best preserve structure (e.g. distance, degree of vertices, etc.) between all vertices, whether they are labeled or not. In applications, this equates to using unlabeled cases during training, which places the problem of defining a suitable learner on a graph in the transductive (semi-supervised) construct.

In this work, the similarity weighted matrix, $W = \{W_{ij}\}$, $i, j = 1, \dots, n$ with $W_{ij} \in [0, 1]$ and $W_{ii} = 1$, serves the purpose of providing the transductive structure necessary for deriving the graph-based classifier/learner. Large values indicate a higher degree of similarity between two nodes and small values the opposite. There are many options for obtaining W in practice, and the optimal choice depends in part on the nature of the underlying graph (e.g. sparse or dense), and data size/computational considerations. For example, consider the lattice shown in Figure 2 (b), where the training data are clustered in the center with a linear boundary separating the two classes. In this case, there are two possibilities for defining W ; the first one uses the observed lattice adjacency matrix as W directly. The second choice relies on first computing the shortest path metric over the lattice, and subsequently applying a kernel function to convert dissimilarities to similarities, which results in a new adjacency matrix $W \equiv W_\tau$. The kernel function needs to be specified in order to use this option, which in this work is chosen to be $K_\tau(i, j) = \exp(-\frac{d(i, j)}{\tau})$, with $d(\cdot, \cdot)$ denoting the shortest path metric and $\tau > 0$ a tuning parameter. An analogous discussion applies to the case of obtaining proximity graphs from a feature space X [9].

Remark: An important non-trivial problem under investigation is inductive learning, in which we wish to predict a vertex that was not available in any capacity during training [4].

III. PROBLEM FORMULATION

Let $G_F = (V_F, E_F)$ denote a weighted graph with vertex set V_F and edge set E_F . The vertices correspond to the data observations (cases) and the weighted edges capture the relationship between them. Further, let $Y(v)$ denote the response – either a class label or a probability class estimate – associated with node v . The vertex set V_F can be partitioned into a labeled set V_L of size m and an unlabeled set V_U of size $n-m$, which translates into a similar partition for the response vector Y . The goal is to predict the response Y_U for the nodes in the unlabeled set V_U . The matrix W , discussed in Section II, is the $n \times n$ weighted similarity matrix corresponding to G_F .

We briefly discuss next some challenges posed by the graph structure, which the *Sequential Predictions Algorithm* (SPA) is designed to overcome. In a supervised learning context, only the labeled cases (in our setting those in the set V_L) are used for constructing a classifier. However, the relationship between labeled and unlabeled cases on a graph is more complex, as previously indicated, which in turn is going to affect the construction of nearest neighbor type classifiers. For the SPA, the topology of the underlying graph is taken into consideration during training, along with the problem of predicting the unlabeled cases. The algorithm employs a graph-based nearest neighbor classifier and a sequence of vertex sets for label propagation purposes, together with a regularization mechanism. We define next several quantities necessary for subsequent developments.

Let $d(u, v)$ denote the shortest path distance between two nodes u and v . Further, let $d(u, A)$ denote the shortest path distance between node u and a vertex set A ; formally, $d(u, A) = \min_{v \in A} \{d(u, v)\}$. Consider next a sequence of sets $\mathcal{V} = \{V_r\}_{r=0}^{\kappa}$, such that $V_L = V_0 \subseteq V_1 \subseteq V_2 \subseteq \dots \subseteq V_{\kappa} = V_F$. The sets V_r contain all the unlabeled nodes that are within a certain distance of labeled ones and are formally defined as: $V_r = \{u \in V_F : d(u, V_L) \leq q_r\}$, where q_r a prespecified nonnegative threshold with $q_0 = 0$ and $q_{\kappa} = \max_{u \in V_U} d(u, V_L)$. In the case of an unweighted graph, the sets V_r contain all the unlabeled nodes that are at most q_r hops away from labeled ones. Finally, define the sequence of symmetric similarity matrices W_r of size $|V_r| \times |V_r|$, whose (u, v) -th element $W_r(u, v) \in [0, 1]$ contains the similarity measure between nodes u and $v \in V_r$. This matrix W_r is a submatrix of the weighted similarity matrix W , whose values contain similarities between nodes in V_F .

IV. SEQUENTIAL PREDICTIONS ALGORITHM

The proposed algorithm employs a graph-based kernel smoother classifier together with a regularization mechanism and proceeds sequentially to predict the labels of nodes in V_U . We start by defining the classifier. Given a similarity matrix W_r , define the smoother matrix S_r , whose (u, v) -th element is given by

$$S_r(u, v) = \frac{W_r(u, v)}{\sum_{w \in V_r} W_r(w, v)}.$$

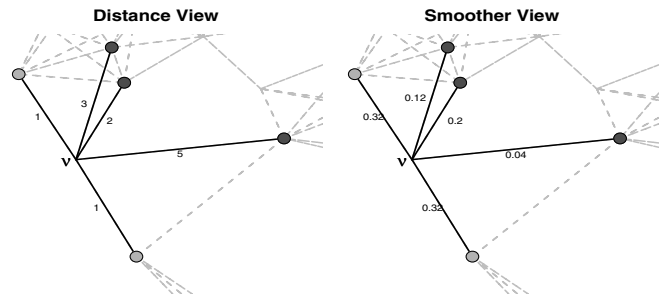


Fig. 3. The above graph cross-section provides a simple example of applying the local smoother in (1) for predicting a case ν . For this example, the first plot provides the labeled cases that are within 5 hops of ν , where the distance is taken over labeled and unlabeled cases. The second plot provides the normalized kernel weights, which classifies ν as light gray with probability 0.64.

Notice that S_r is a stochastic matrix; i.e. all its rows sum up to 1. A classifier $\mathbf{K}_r(v)$ that is designed to reflect the local graph structure is defined for any $v \in V_r$ as

$$\mathbf{K}_r(v) = \sum_{u \in V_r} S_r(u, v) Y_r(u). \quad (1)$$

In compact notation, we have that $\mathbf{K}_r(V_r) = S_r Y_r$; i.e. the classifier is *linear* in the response Y_r .

For illustration purposes, consider using this classifier on the graph segment highlighted in Figure 3, using only the labeled information contained in V_0 with $W_0(u, v) = \exp(-d(u, v)/\tau)$, and τ set to 2. In this case, $\mathbf{K}_0(v)$ classifies the unknown vertex (observation) as light gray with probability 0.64. The tuning parameter τ regulates the decrease in similarity as a function of distance and could be estimated from the data using generalized cross-validation [11].

The SPA (focus on SPA(s) given in Algorithm 4) proceeds in an iterative fashion and at step r uses the labeled cases Y_{r-1} (both known and estimated ones) in set V_{r-1} to predict those in set V_r . The final output of the algorithm is an estimate of the entire vector Y_F . In many cases, it is common to *clamp the response* at each iteration, by setting $Y_r(v) = Y_{r-1}(v)$ for all nodes $v \in V_{r-1}$.

The responses Y_r are obtained through the following iterative procedure, which corresponds to the so-called *local phase* of the SPA (Algorithm 4). Initialize the responses Y_r^0 and then clamp those in set V_{r-1} . Now, at iteration k , predict $Y_r^k = S_r Y_r^{k-1}$ and clamp $Y_r^k(v) = Y_{r-1}(v)$ for each $v \in V_{r-1}$, until $\|Y_r^k - Y_r^{k-1}\| < \delta$, with $\delta > 0$ being a prespecified tolerance parameter. The final response produced during the local phase of the algorithm is denoted by Y_r^* . Convergence is guaranteed by the fact that S_r is a stochastic matrix and hence its $V_r - V_{r-1}$ partition is substochastic with norm less than one. In the local phase, the algorithm successively adjusts the response employing the local graph topology structure.

In the *global phase* of the SPA, the algorithm incorporates a regularization component that takes into account the fact that classification of unlabeled nodes farther away from labeled ones is more uncertain, since they are based on predicted responses at previous steps. The proposed regularization drives the predictions towards the mean response of the labeled nodes

for $r \in \{1, \dots, \kappa\}$ **do**
 Local Phase
 Initialize, $P_r^0 = \mathbf{K}_{r-1}(V_r)$, $Y_r^0 = \xi(P_r^0)$ and fix $Y_r^0(v) = Y_{r-1}(v)$
 for $v \in V_{r-1}$ (clamp).
 Repeat estimation of $P_r^k = \mathbf{K}_r(V_r)$ using the response Y_r^{k-1} , set
 $Y_r^k = \xi(P_r^k)$ and then clamp Y_r^k .
 Until $\|Y_r^k - Y_r^{k-1}\| < \delta$ (i.e. convergence)
 Denote the convergent response as Y_r^* .
 Global Phase
 Set $Y_r(v) = Y_r^*(v) + \alpha(\rho(r))(\mu - Y_r^*(v))$ for each $v \in V_r - V_{r-1}$.
end for

Fig. 4. The Sequential Predictions Algorithm (SPA). For soft labels, SPA(s), set $\xi(p) = p$ and $\alpha(\rho) = \rho$, and for hard labels, SPA(h), set $\xi(p) = \mathbf{1}_{\{p \geq 0.5\}}$ and $\alpha(\rho) = \frac{\rho}{\sqrt{\mu(1-\mu)(1-\rho)+\rho}}$.

as a function of distance. Specifically, the response at step r is given by $Y_r = Y_r^* + \rho(r)(\mu - Y_r^*)$ where μ is the average response obtained from the labeled cases and $\rho(r)$ is a monotone increasing rate function in $[0, 1]$. As the procedure processes cases farther away from the known data, the global phase puts more emphasis on the global point estimate μ to counteract the use of previously predicted cases for the local estimate, Y_r^* .

A. A loss function framework for SPA

The SPA(s) algorithm optimizes the following function in the local phase when predicting probability class estimates (soft labels)

$$Y_r(v) = \arg \min_{h(v) \in \mathbb{R}} J(h, Y_r^*, W_r) + \lambda_r P(h, \mu), \quad (2)$$

where J corresponds to the quadratic loss function and is given by $J(h, Y, W) = \sum_u W(u, v)(Y(u) - h(v))^2$ and $\lambda_r \equiv 0$. Repeated optimization of this problem together with clamping results in the convergent response Y_r^* for the local phase given in Algorithm 4. In the global phase, a penalty term $P(h, \mu) = (\mu - h)^2$ with $\lambda_r \geq 0$ for SPA(s) is supplied for (2). The procedure is designed so that an increase in r results in an increase in λ_r . Notice that this preserves the monotone property of the rate function since $\rho(r) = \frac{\lambda_r}{1+\lambda_r}$ is identified by the equivalence between optimization of (2) and the global phase of SPA(s). Therefore, the procedure is applying an increase in regularization to pull the local estimates towards the global estimate μ to account for extrapolation using predicted responses.

Next, we investigate an extension of the above optimization problem to the exponential loss function, with $J(h, Y, W) = \sum_u W(u, v) \exp(-g(Y(u))h(v))$ and $g(y) = 2y - 1$. In this case, the response corresponds to hard labels taking values in the set $\{0, 1\}$. For this loss function, we consider a symmetric penalty given by $P(h, \eta) = \exp(h - \eta) + \exp(\eta - h)$ (hyperbolic penalty). Optimization of exponential loss with the hyperbolic penalty leads to the SPA(h) procedure given by Algorithm 4.

Proposition 1: Let $V_r \in \mathcal{V}$ with response Y_{r-1} assumed known. The local phase of SPA(h) corresponds to optimization of $\min_{h(v) \in \mathbb{R}} \sum_{u \in V_r} W_r(u, v) e^{-g(u)h(v)} + \lambda_r (e^{h(v)-\eta} + e^{\eta-h(v)})$, with response $g = 2Y_r^k - 1$, $\eta(\mu) = 0.5 \log\left(\frac{\mu}{1-\mu}\right)$ and $\lambda_r \equiv$

0. For the global phase, the problem is solved with $\lambda_r \geq 0$ and response $g = 2Y_r^* - 1$.

This result shows that the exponential loss function regularized with the symmetric hyperbolic penalty fits naturally in the SPA framework. Interestingly, the specific optimization problem introduces an adjustment function α that updates the curvature of the rate ρ to account for μ .

V. ILLUSTRATIVE EXAMPLES

The examples in this section illustrate the SPA on both real and simulated data sets. Results are provided for both SPA(s) (soft labeled version of SPA) and SPA(h) (hard labeled version of SPA). Recall that for hard labels, classification using the exponential loss function together with a hyperbolic penalty term is used, while for soft labels the quadratic loss function and a quadratic penalty are employed.

We discuss next a data driven estimation approach for determining the set $\{q_r\}_{r=1}^{\kappa}$, which is used in constructing the sets V_r . A good way of defining the q_r sequence is to specify it according to the partition of $d(v, V_L)$ for $v \in V_U$ into κ regions with area equal to $\frac{1}{\kappa}$. Therefore given κ the entire q_r sequence is determined. Next, we define the rate function by the corresponding shrunken ‘probability’ estimate: $\rho_{\kappa, \gamma}(r) = \sum_{i=0}^{\kappa} \frac{i}{(\gamma+1)^{\kappa}} \times \mathbf{1}_{(q_{i-1}, q_i]}(r)$ for $\gamma \geq 0$. The new parameter γ allows μ to influence the local estimate Y_r^* without forcing it to be identical to μ , especially for $r \approx \kappa$. The remaining parameters (κ , γ and τ) are estimated using cross validation unless otherwise specified. The implementation of the SPA is available on the main author’s website.

A. A synthetic data set

The graph G_F is given by a 40×40 node lattice with two classes (black and gray) and 144 cases reserved for training purposes with 72 in each class (see Figure 2). The planar nature of this graph allows a visual inspection into the workings of the SPA.

The SPA is evaluated without regularization (i.e. execute SPA with $\kappa = 1$ and $\gamma = \infty \implies \rho(r) \equiv 0$ and $\mathcal{V} = \{V_L, V_F\}$) on the two training configurations given in the first column of Figure 5 (a) for both SPA(s) and SPA(h). In the case of SPA(s) (second column) the intensity of each case is related to its proximity on the lattice to a labeled (training) case. Cases close to a labeled black case have a high PCE for the black class and will be dark on the plot. Similarly, cases close to a labeled gray case have a low PCE for the black class and will be lightly colored on the plot. The predictions become less certain for cases located farther away from labeled cases. On the other hand, for SPA(h) the intensity of each case is related to its proximity on the lattice to the classification border (not the training data). Therefore cases close to the classification border will be classified with less certainty and will be neither black (high PCE for black) nor white (high PCE for gray) in either training configuration.

As mentioned above, SPA(s) optimizes a squared error loss function, which leads to an internal regularization effect over the unlabeled data. The algorithm is internally decreasing the confidence of the PCEs based on their location with respect to

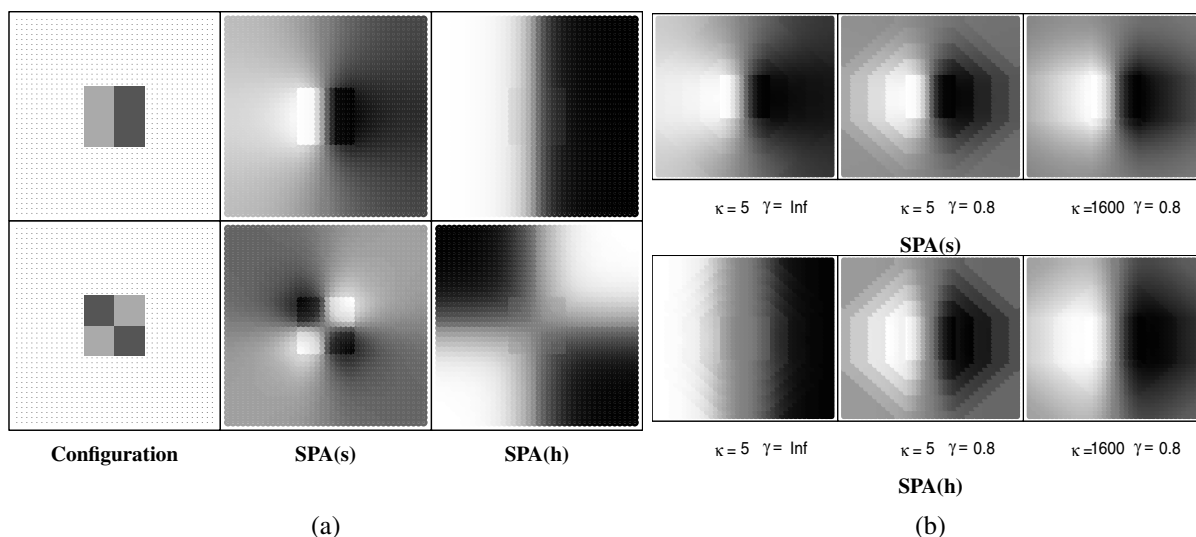


Fig. 5. (a) (First Column) The positioning of the training data over the lattice in each example. (Second Column) The PCEs for SPA (s) over the entire unknown lattice without regularization (i.e. $\kappa = 1$ and $\gamma = \infty$). At each point on the lattice, a class prediction (gray or black) is made. The intensity of the pixel at each point represents the probability that the point has been classified as “black”. The more intense the pixel, the larger the PCE for the black class. (Third Column) The analogous SPA(h) result for each training configuration. For SPA(s) and SPA(h), the training data were classified using $\mathbf{K}_0(v)$ as described in Section IV. (b) The grid of plots demonstrates regularization of SPA(s) and SPA(h) (i.e. vary κ and γ) for the first training configuration only.

the labeled data. On the other hand, SPA(h), has binary output and as a result there is a sharp boundary (reflected in the PCE) between the two classes. The results reflect the underlying loss function used for optimization in a fairly intuitive way. As a final comment, the PCEs for the labeled data seem to be on a different scale, and the reason for this will become more clear as we discuss regularization.

To assess the effect of regularization (i.e. vary κ and γ) refer to Figure 5 (b). In the case of $\kappa = 5$ we observe discrete jumps, which indicate the specific vertex grouping (i.e. cases in V_1 predict new cases in V_2 which in turn predict new cases in V_3 , etc.). As κ increases, the distinction between the labeled and unlabeled data becomes blurred, where the cases near the labeled data are increasing in confidence to become similar to the fixed training data (recall that the training data is unaffected by the rate function). As κ increases and γ decreases the procedure more dramatically accounts for the use of unlabeled data. In particular, the confidence of a case somewhat farther away from labeled data is adjusted with a stronger pull towards $\mu = 0.5$ (the class prior probability). In this plot we notice that the effect of regularization in the limit ($\kappa \rightarrow \infty$ and $\gamma \rightarrow 0$) for SPA(s) and SPA(h) are fairly similar.

B. Multi-view learning with SPA

In this section, we consider a multi-view data set involving compounds used in drug discovery [10]. The success of a compound in becoming an approved drug is highly related to its clinical side-effect status. The objective is to predict the side-effect status of compounds based on biological (1st view) and chemical (2nd view) features. The first view consists of 191 continuous noisy biological features that describe the association between the compound and a specific target (cell, protein) measured in a lab over a period of a few weeks. The second view consists of 151 chemical structural binary features

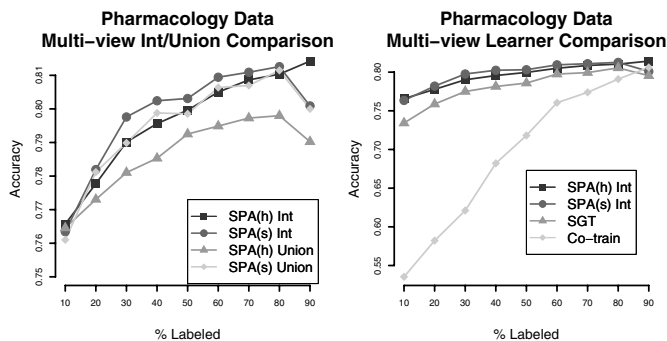


Fig. 6. (left) provides SPA(s) and SPA(h) performance results for these data using both the union and intersection graphs. The x axis provides the percent labeled, where the remaining cases are treated as unlabeled. (right) provides the performance of SPA compared to that of co-training and the SGT.

computed via software, which indicate the presence or absence of a particular molecular structure in the compound.

Both the SPA(s) and SPA(h) algorithms were employed with this data set, using $K = 1$ Nearest Neighbor graphs constructed independently from each view. The SPA operated on the union and intersection graph [12] computed element-wise as $\max(W_1, W_2)$ and $\sqrt{W_1 W_2}$, respectively (W_i is the weighted adjacency matrix for view i). The results in Figure 6 provide average testing performance over 50 replicates, where the training set size varied from 10-90% of the originally labeled cases, while the corresponding testing cases were treated as unlabeled. The intersection graph is interpreted as capturing the agreement between the biological and chemical views. The union graph provides an edge additive effect, and as a result, performance suffers since the association for specific chemistry and biological similarity is lost.

The results in Figure 6 (right) provide a comparison between the SPA and both co-training with the supervised K nearest neighbor classifier (K -NN) and the spectral graph transducer

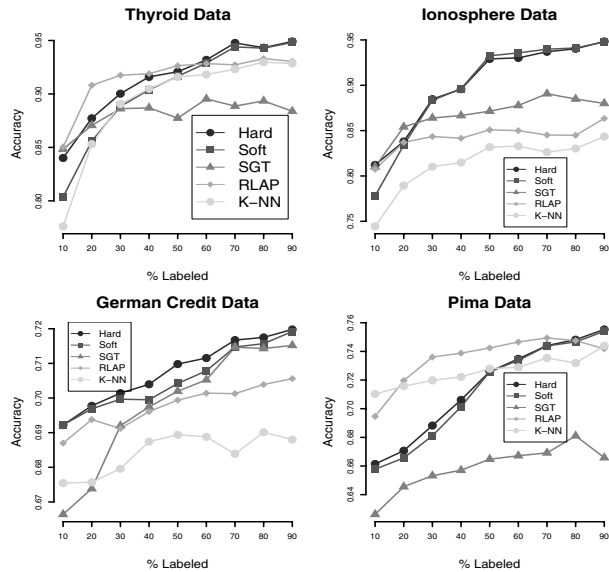


Fig. 7. Performance plots for various data sets. The x-axis is the % of labeled cases, where the remainder are treated as unlabeled, and the y-axis is accuracy average over 50 random samples.

(SGT) (default settings for SGT-lite software) [1; 2]. The co-training procedure iteratively and greedily adds confident unlabeled predictions from each view to the labeled set. However a closer inspection reveals that in many cases the confident biological and chemical predictions for the side-effect class are incorrect, which in turn leads to reinforcement of mistakes for co-training. The graph-based approaches have a significant advantage in this setting, since the procedures are not operating on the predictions of the K -NN classifier, but are operating on the relational information associated with the K -NN graphs. The SGT provides a competing graph-based approach similar to SPA; it uses the spectral values on the Laplacian matrix constructed by adding the adjacency matrices of each view. On this particular data set, the regularized version of SPA applied to the intersection graph exhibits the best performance.

C. Application to real data sets

A number of standard data sets taken from the UCI repository (Thyroid, Ionosphere German Credit, and Pima) are used to assess the performance of the SPA. The SGT procedure was used together with the regularized Laplacian (RLAP) [7]. The RLAP procedure solves a sparse system of equations using ridge regularization on the spectra of Laplacian. The supervised K -NN procedure was also provided since all procedures rely on the K -NN structure, and this provides a baseline for comparison. Notice that the co-training algorithm is specific to multi-view learning and could not be used in this context.

For SPA and supervised K -NN, the parameter $K = 5$ was sufficient. The RLAP and SGT were performed using the available software with K set to 10 and 50, respectively. In our experience, these parameter setting performed well; nevertheless, small changes in K can affect the performance of each technique [2; 4]. Approaches for estimating K in SPA along the lines of [3] are currently under investigation.

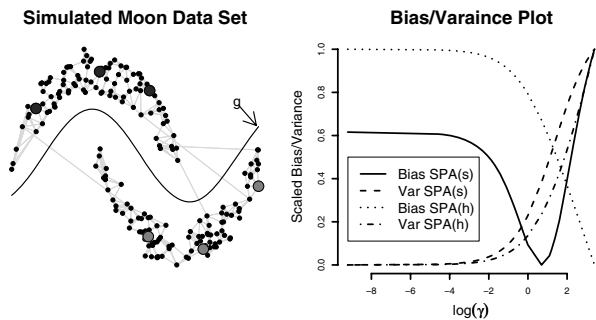


Fig. 8. A simulated data set with $|L| = 6$ (3 for each class). The adjacent plot is bias and variance (scaled to $[0, 1]$) averaged over 500 such data sets for several choices of γ .

The size of the cases retained for training purposes was varied between 10-90% and the results based on 50 replications are shown in Figure 7. It can be seen that for the Ionosphere and German Credit data sets, the SPA usually outperformed its competitors. It also exhibited good performance for the Thyroid data. The Pima data set provides a noisy scenario, where the SPA and SGT algorithms experienced difficulties with their performance, especially in the presence of a small size labeled set, while RLAP and K -NN performed well. The SPA catches up in performance when the size of the labeled set exceeds 50% of the total cases.

D. Bias/variance tradeoff

We briefly investigate the issue of bias and variance on a synthetic data set. The model is in general given by $\eta(Y) = f(\nu) + \epsilon$ where η is the link function, f a function over the vertices, and the error term ϵ has mean zero with constant variance σ^2 . The conditional expected error rate, $E[\|\eta - \hat{f}\|_2^2 | G]$, for a graph-based learner, \hat{f} , is decomposed in $\text{Var}(\epsilon) + \text{Bias}^2(\hat{f}) + \text{Var}(\hat{f})$, as discussed in [11]. Typically, a decrease in bias results in an increase in variance, and one is often interested in the tradeoff between the two. In our example, the data generation mechanism gives two moon-shaped components, whose relation is shown in Figure 8 (left). The true probability of the dark gray class reflects displacement from the border g defined on the x, y -plane, given by: $p = (1 + \exp(g(x) - y))^{-1}$ with true function, $f = \eta(p)$. Six labeled observations were randomly selected (three from each class) over 500 samples, and the average bias and variance of \hat{f} resulting from SPA was computed. The values for γ were varied over a grid with κ fixed to 100. From Figure 8 (right), it can be seen that the optimal values of γ are selected with $\gamma \approx 1$ for SPA(s) and $\gamma \approx 7$ for SPA(h). This suggests that the regularization imposed by SPA leads to a tradeoff between bias and variance.

VI. CONCLUSIONS

In this paper, we proposed the SPA as a graph-based algorithm which propagates labels across vertices. The framework encompasses both hard and soft labels as they relate to the underlying loss function and statistical regularization. The algorithm was evaluated on a number of synthetic and real

data sets. In particular, the pharmacology data emphasized how information from different views (assays and compound fingerprints) can be naturally incorporated through a graph structure. The SPA provides a useful algorithm for graph-based learning that is competitive compared to existing propagation algorithms.

APPENDIX

PROOF OF PROPOSITION 1

Proposition 1: Let $V_r \in \mathcal{V}$ with response Y_{r-1} assumed known. The local phase of SPA(h) corresponds to optimization of $\min_{h(v) \in \mathbb{R}} \sum_{u \in V_r} W_r(u, v) e^{-g(u)h(v)} + \lambda_r (e^{h(v)-\eta} + e^{\eta-h(v)})$, with response $g = 2Y_r^k - 1$, $\eta(\mu) = 0.5 \log\left(\frac{\mu}{1-\mu}\right)$ and $\lambda_r \equiv 0$. For the global phase, the problem is solved with $\lambda_r \geq 0$ and response $g = 2Y_r^* - 1$.

Proof: The proposition relies on optimization of (2), starting with a response Y_{r-1} defined on the vertex set $V_r \in \mathcal{V}$, and a parameter $\lambda \geq 0$. Further let $\eta(\mu) = 0.5 \log\left(\frac{\mu}{1-\mu}\right)$.

Find the initialized PCE $P_{r-1}^0(v)$ for some $v \in V_r$ by minimizing $J(h)$. For this, let $g = 2Y_{r-1} - 1$ and η be short for $\eta(\mu)$, $\mathbf{K}_{r-1}(v) = \frac{\sum_{u \in V_{r-1}} W_{r-1}(u, v) Y_{r-1}(u)}{\sum_{u \in V_{r-1}} W_{r-1}(u, v)}$, and then $J(h) = (\mathbf{K}_{r-1}(v) + \lambda e^\eta) e^{-h(v)} + (1 - \mathbf{K}_{r-1}(v) + \lambda e^{-\eta}) e^{h(v)}$. The solution to $\min_{h(v) \in \mathbb{R}} J(h)$ is $h(z) = \frac{1}{2} \log\left(\frac{\mathbf{K}_{r-1}(v) + \lambda e^\eta}{1 - \mathbf{K}_{r-1}(v) + \lambda e^{-\eta}}\right)$. The resulting PCE is: $P_r^0(v) = \eta^{-1}(h(v)) = \frac{\mathbf{K}_{r-1}(v) + \lambda e^\eta}{1 + \lambda(e^\eta + e^{-\eta})}$. If we set $\lambda = 0$, then we obtain the PCE, $P_r^0(v) = \mathbf{K}_{r-1}(v)$ for $v \in V_r$. The next step is to update the response by $Y_r^0 = \mathbf{1}_{\{P_r^0 \geq 0.5\}}$, with $Y_r^0(v) = Y_{r-1}(v)$ for $v \in V_{r-1}$. Then at iteration k , minimize $J(h)$ with response Y_r^{k-1} to obtain PCE, $P_r^k(v) = \mathbf{K}_r(v)$. This completes the local phase of SPA under exponential loss with final PCE, P_r^* and clamped response Y_r^* .

For the global phase we update the PCE with the response Y_r^* and define: $\nu(\lambda) = \frac{\lambda(e^\eta + e^{-\eta})}{1 + \lambda(e^\eta + e^{-\eta})} \in [0, 1]$. Notice that $\frac{\lambda e^\eta}{1 + \lambda(e^\eta + e^{-\eta})} = \nu(\lambda) - \frac{\lambda e^{-\eta}}{1 + \lambda(e^\eta + e^{-\eta})} = \nu(\lambda)\mu$ with $\mu = \frac{e^\eta}{e^{-\eta} + e^\eta}$. Therefore, we see that the final PCE is given by, $P_r(v) = (1 - \nu(\lambda))P_r^*(v) + \nu(\lambda)\mu$. Moreover using the identity $\rho = \frac{\lambda}{\lambda+1}$ we have for all $\lambda \geq 0$ that $\alpha(\rho) = \frac{\rho}{(1-\rho)\sqrt{\mu(1-\mu)} + \rho} = \nu(\lambda) \in [0, 1]$. ■

ACKNOWLEDGMENTS

The authors would like to thank the Associate Editor and three anonymous referees, whose useful and constructive comments led to a significant improvement of the presentation of the manuscript. The work of George Michailidis was partially supported by NIH grant 5P41RR018627 – 04.

REFERENCES

- [1] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Computational Learning Theory*, 1998, pp. 92–100.
- [2] T. Joachims, "Transductive learning via spectral graph partitioning," in *International Conference on Machine Learning*, 2003, pp. 290–297.
- [3] X. Argyriou, M. Herbster, and M. Pontil, "Combining graph laplacians for semi-supervised learning," in *Neural Information Processing Systems*, 2005.
- [4] X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep., 2006.

- [5] S. Abney, "Understanding the yarowsky algorithm," *Computational Linguistics*, vol. 30, no. 3, pp. 365–395, 2004.
- [6] X. Zhu, "Semi-supervised learning with graphs," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., 2005.
- [7] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Computational Learning Theory*, 2004, pp. 624–638.
- [8] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Neural Information Processing Systems*, 2005, pp. 1641–1648.
- [9] M. Herbster and M. Pontil, "Prediction on a graph with the perceptron," in *Neural Information Processing Systems*, 2006.
- [10] M. Kansy, F. Senner, and K. Gubemator, "Physicochemical high throughput screening: Parallel artificial membrane permeation assay in the description of passive absorption process," *Journal of Medicinal Chemistry*, vol. 44, no. 6, pp. 923–930, 2001.
- [11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning (Data Mining, Inference and Prediction)*. Springer Verlag, 2001.
- [12] R. Gould, *Graph Theory*. The Benjamin/Cummings Publishing Company, 1998.



Mark Culp received his Ph.D. in Statistics from The University of Michigan in 2007. In the fall of 2007, he will join West Virginia University as an Assistant Professor of Statistics. His research interests are in the areas of machine learning with emphasis on semi-supervised learning and ensemble methods, and in drug discovery applications involving non-clinical statistics.



George Michailidis received his Ph.D. in Mathematics from UCLA in 1996. He was a post-doctoral fellow in the Department of Operations Research at Stanford University from 1996 to 1998. He joined The University of Michigan in 1998, where he is currently an Associate Professor of Statistics and of Electrical Engineering and Computer Science. His research interests are in the areas of machine learning, data mining with emphasis on visualization and stochastic network modeling, and performance evaluation with emphasis on queuing analysis and

congestion control.