Semi-supervised Learning using Kernel Self-consistent Labeling

Keywords: semi-supervised learning, kernel methods, support vector machine, leave-one-out cross validation, Gaussian process.

Abstract

We present a new method for semi-supervised learning based on any given valid kernel. Our strategy is to view the kernel as the covariance matrix of a Gaussian process and predict the label of each instance conditioned on all other instances. We then find a self-consistent labeling of the instances by using the hinge loss on the predictions on labeled data and the ε insensitive loss on predictions on the unlabeled data. Computationally, this leads to a quadratic programming problem with only bound constraints. We examine the similarities and differences between the algorithm and other semi-supervised learning algorithms. Evaluation of the method on both synthetic and real-world datasets gives encouraging results.

1. Introduction

In many applications, labeled data are expensive but unlabeled data can be easily obtained. In these applications, it is desirable to exploit the unlabeled data to improve performance of learning algorithms. Methods that learn from both labeled and unlabeled data are called transductive or semi-supervised learning algorithms.

Kernel methods have been used successfully in various machine learning problems including classification, regression, ranking and dimensionality reduction. The use of kernels allows non-linear functions to be used efficiently in these problems by mapping the instances into a higher dimensional feature space. By designing the kernels appropriately, a rich family of high dimensional mappings is available to suit various learning problems. In view of the advantages and flexibility of kernel methods, effective semi-supervised kernel learning algorithms could be quite useful.

We use the *self-consistency principle* to design a semi-supervised kernel classification method. The idea behind this principle is to label the unlabeled data in such a way that the classifier trained using a subset of the (labeled and artificially labeled) data would have low validation error on the (labeled and artificially labeled) data that is not used in training. This principle was used in designing the mincut based semi-supervised learner in (Blum & Chawla, 2001) where the algorithm was shown to able to generate a labeling that minimizes the leave-one-out cross validation (LOOCV) error of the 1-nearest neighbor algorithm. A normalized version of the algorithm based on minimizing the ratiocut (Spectral Graph Transducer) was proposed in (Joachims, 2003).

We propose a support vector style algorithm with ε-insensitive regression cost constraints for self-consistency on unlabeled points and hinge loss penalties on labeled points. The problem can be formulated as a *quadratic programming* problem with only bound constraints, for which global optimum can be found efficiently. As the method is based on leave-one-out Gaussian process prediction using kernels, we call the method *kernel self-consistent labeling*.

Kernel self-consistent labeling has interesting connections with many existing algorithms. Without the ε-insensitive self-consistency constraints, the method is very similar to a leave-one-out version of the support vector machine. The Representer Theorem states that the support vector machines can be represented by linearly combining only kernel functions that are associated with the labeled data, regardless of whether unlabeled data is present. Interestingly, kernel self-consistent labeling utilizes kernel functions of both the labeled and unlabeled instances even when only the hinge loss penalties on labeled points are used.

Like many semi-supervised learning methods, it is also possible to view the kernel self-consistent labeling as a graph-based semi-supervised learning method. Both the mincut and the Spectral Graph Transducer algorithms are based on minimizing edge costs on graphs. Similarly, (Zhu, Ghahramani & Lafferty, 2003) minimizes the sum of squared differences of the function on all pairs of points (edges), weighted by the pair-wise similarity in the graph. Instead of minimizing weighted cost on edges, a second class of graph algorithms minimizes the node cost. They express the predicted label of one instance based on the average of other instances' labels, weighted by mutual similarity. In (Roweis & Saul, 2000), the cost function is defined on each *node* as the squared difference between the node's own function value and the weighted average values of its neighbors in a certain locality. Similarly, we can view kernel self-consistent labeling as a method for

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

minimizing node costs on a graph by using what is known as the *equivalent kernel* (Silverman, 1984) of the original kernel as the weight matrix in a graph.

Conversely, in some graph-based semi-supervised learning methods such as in the harmonic energy minimization (Zhu et al., 2003), the *Laplacian* of the graph (Chung, 1999) can be viewed as the pseudo-inverse of a kernel (Ham et al., 2004). Kernels can be interpreted as the covariance matrix of a Gaussian process except with independent Gaussian noise added to the output (Seeger, 2004). Thus, relationship between given labels, the probability distribution of unknown labels, and pair-wise similarity among examples can be represented under the framework of multivariate Gaussian distributions. Interestingly, in Gaussian distributions, the expected label of an unlabeled point, conditioned on known labels, turns out *not* to utilize the other unlabeled data once the kernel is fixed. In the harmonic energy minimization (Zhu et al., 2003), the unlabeled data is utilized to construct the Laplacian and changing an unlabeled instance would result in a different kernel and hence different predictions on the other unlabeled data. In contrast, kernel self-consistent labeling uses a fixed kernel and allows the change of an unlabeled instance to affect predictions on other unlabeled instance by self-consistency and optimizing an appropriate cost function.

The rest of this paper is organized as follows. Section 2 gives some background on Gaussian process and introduces the leave-one-out prediction algorithm which establishes the relationships among all data points based on any valid kernel. Section 3 presents the proposed mixed classification and regression leave-one-out style algorithm that uses the hinge loss on labeled data and the ε-insensitive loss function on unlabeled data. The model's connections to existing learning algorithms are discussed in Section 4. Computational efficiency is considered in Section 5. Section 6 contains experimental results. Finally the paper is concluded in Section 7.

2. Leave-one-out Gaussian Process Prediction

We assume that either a proper kernel function $k_0(x, x')$ that satisfies Mercer's theorem or a valid *Gram* matrix K_0 (symmetric and positive semi-definite) (Schölkopf & Smola, 2002) for both labeled and unlabeled data is given. By Mercer's Theorem, we have

$$k_0(x,x') = \sum_i \lambda_i \phi_i(x)\phi_i(x')$$

where $\{\phi_i(\cdot)\}$ are the eigenfunctions of $k_0(\cdot,\cdot)$, or eigenvectors of *Gram* matrix. Then we define Bayesian Linear Regression:

$$z(x) = \theta^T \phi(x), \qquad y(x) = z(x) + \varepsilon = \theta^T \phi(x) + \varepsilon$$

where $\phi(x) = (\phi_1(x), \phi_2(x), ...)^T$, $\theta \sim N(0, \Sigma)$, $\Sigma = diag(\lambda_1, \lambda_2, ...)$, and noise $\varepsilon \sim N(0, \sigma^2)$ which is independent between different x. Then covariance

$$\begin{aligned} \mathrm{cov}(z(x), z(x')) &= \phi(x)^T \Sigma \phi(x') = k_0(x,x') \\ \mathrm{cov}(y(x), y(x')) &= \phi(x)^T \Sigma \phi(x') + \sigma^2 \delta(x,x') \\ &= k_0(x,x') + \sigma^2 \delta(x,x') \end{aligned}$$

where $\delta(x,x')$ is *Kronecker* delta. So we can view $\{Z(x)\}$ as Gaussian Process (GP) with covariance matrix K_0 if K_0 is nonsingular. Suppose we have L labeled examples $\{(x_i, c_i)\}_{i=1}^{L}$ with $c_i \in \{-1, 1\}$. Then the log posterior of the Gaussian process $Z(x)$ is given by

$$J = \frac{1}{2\sigma^2}(c - z)^T (c - z) + \frac{1}{2} z^T K_0^{-1} z \qquad (1)$$

where the first term is log likelihood and the second term is the log of the prior, acting as a regularization factor. Assume there are U unlabeled examples $\{(x_i, \cdot)\}_{i=L+1}^{L+U}$ and denote $N = U + L$. We can view $\{Y(x)\}$ as a GP as well with covariance matrix $K = K_0 + \sigma^2 \cdot I$. Suppose K can be written as

$$K = \begin{pmatrix} K_{LL} & K_{LU} \\ K_{UL} & K_{UU} \end{pmatrix}.$$

As K_0 is positive semi-definite, K must be positive definite if $\sigma \neq 0$. Conditioning on the given labels c, the expected value of unlabeled data's label y_U is $K_{UL} K_{LL}^{-1} c$.

Unfortunately, the result shows that unlabeled data are *not* used in predicting y_U, i.e., only K_{LL} and the row corresponding to the single test example k_x^T are involved. Unlabeled data are not mutually related, directly or indirectly via labeled data, in the process of classifying other unlabeled data.

Our strategy of utilizing unlabeled data is to do leave-one-out prediction for each instance assuming that the labels of the other $N - 1$ instances are known and impose self-consistency.

We compute a decision vector y with each element associated with one example and then assign labels based on y. For labeled data, we do not fix its associated element in y to any predefined value, allowing it to vary. Assume we know $z_i = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_N)^T$ then the prediction of y_i is $k_i^T T_i^{-1} z_i$, where k_i^T is the i^{th} row vector of K, without the i^{th} element. T_i is the matrix after removing the i^{th} row and i^{th} column of K. Let b_i^T be the vector created by in-

serting 0 at the i^{th} element of $k_i^T T_i^{-1}$, then the prediction of y_i based on all other elements is $b_i^T y$. The function $b(\cdot,\cdot)$ is called *equivalent kernel* or *dual kernel* and is very similar to the equivalent kernel in (Silverman, 1984; Sollich, 2004), defined in a supervised learning settings. Let $B = (b_1, \ldots, b_N)$. We call B the *GP* prediction coefficient matrix.

3. Leave-one-out ρ-ε Learning

Having built up the relationship among all examples through the leave-one-out prediction formula, we can implement the self-consistency principle. For labeled data, since we are only concerned about their class and it is a classification problem, there is no need to force a strong match to target value like in regression. Rather, the maximum margin principle should be preferable. On unlabeled data, if we still use margin constraints, as in Transductive SVM (Joachims, 1999), to find an assignment such that

$$y_j \cdot b_j^T y \geq 1 - \xi_j \text{, where } y_j \text{ is variable,}$$

then we inevitably come up with a non-convex optimization problem. One simple alternative is using regression constraints such as the ε-insensitive loss function (Vapnik, 1995). Finally, regularization is incorporated in the same way as in (1). Putting it all together, suppose the label for the *first L* instances is known and *K* has already incoporated Gaussian noise, the problem is formulated as:

Minimize with respect to $\{y, \xi, \xi'\}$:

$$\frac{1}{2} y^T K^{-1} y + C_1 \sum_{i=1}^{L} \xi_i + C_2 \sum_{j=L+1}^{N} \left(\xi_j + \xi_j'\right) \quad (2)$$

s.t.

$$c_i \cdot b_i^T y \geq 1 - \xi_i \qquad i = 1 \ldots L \quad (3)$$

$$b_j^T y - y_j \leq \varepsilon + \xi_j \qquad j = L+1, \ldots, N \quad (4)$$

$$y_j - b_j^T y \leq \varepsilon + \xi_j' \qquad j = L+1, \ldots, N \quad (5)$$

$$\xi_i \geq 0, \xi_j' \geq 0 \qquad i = 1 \ldots N,\ j = L+1, \ldots, N \quad (6)$$

The constraint (3) is margin constraint. The constraints (4) and (5) are for ε-insensitive regression. The first term in (2) is for regularization. We name this mixed classification-regression algorithm as ρ-ε learning because ρ is usually used to denote margin. The whole problem is a quadratic programming (*QP*) problem with *K* (thus K^{-1}) being positive definite.

The *Lagrangian* is:

$$\mathcal{L} = \frac{1}{2} y^T K^{-1} y + C_1 \sum_{i=1}^{L} \xi_i + C_2 \sum_{j=L+1}^{N} \left(\xi_j + \xi_j'\right) - \sum_{i=1}^{L} \alpha_i \left(c_i b_i^T y + \xi_i - 1\right) - \sum_{j=L+1}^{N} \beta_j \left(y_j - b_j^T y + \varepsilon + \xi_j\right) - \sum_{j=L+1}^{N} \gamma_j \left(b_j^T y - y_j + \varepsilon + \xi_j'\right) - \sum_{i=1}^{N} \lambda_i \xi_i - \sum_{j=L+1}^{N} \sigma_j \xi_j' \quad (7)$$

So

$$\partial \mathcal{L} / \partial \xi_i = C_1 - \alpha_i - \lambda_i = 0 \qquad i \in 1 \ldots L \quad (8)$$

$$\partial \mathcal{L} / \partial \xi_j = C_2 - \beta_j - \lambda_j = 0 \qquad j \in L+1, \ldots, N \quad (9)$$

$$\partial \mathcal{L} / \partial \xi_j' = C_2 - \gamma_j - \sigma_j = 0 \qquad j \in L+1, \ldots, N \quad (10)$$

$$\frac{\partial}{\partial y} \mathcal{L} = K^{-1} y - \sum_{i=1}^{L} \alpha_i c_i b_i + \sum_{j=L+1}^{N} \beta_j b_j - \beta_{ex} - \sum_{j=L+1}^{N} \gamma_j b_j + \gamma_{ex} = 0$$

where $\beta_{ex} = (\overbrace{0, \ldots, 0}^{L \text{ 0's}}, \beta^T)^T$, $\gamma_{ex} = (\overbrace{0, \ldots, 0}^{L \text{ 0's}}, \gamma^T)^T$. So

$$y = K \left(\sum_{i=1}^{L} \alpha_i c_i b_i - \sum_{j=L+1}^{N} \beta_j b_j + \beta_{ex} + \sum_{j=L+1}^{N} \gamma_j b_j - \gamma_{ex} \right) \quad (11)$$

Define $x = (\alpha^T, \beta^T, \gamma^T)^T$, $R_1 = \left(c_1 b_1, c_2 b_2, \ldots, c_L b_L\right)_{N \times L}$,

$$R_2 = -\left(b_{L+1}, \ldots, b_N\right) + \left(0_{U \times L}, I_U\right)^T, \quad R = (R_1, R_2, -R_2) \quad (12)$$

where I_U is $U \times U$ identity matrix and $0_{U \times L}$ is $U \times L$ zero matrix. We obtain

$$y = K(R_1 \alpha + R_2 \beta - R_2 \gamma) = KRx \quad (13)$$

Substituting (8)~(13) into (7), we obtain the dual problem:

Maximize with respect to $\{\alpha, \beta, \gamma\}$:

$$-\frac{1}{2} x^T R^T K R x + \sum_{i=1}^{L} \alpha_i - \varepsilon \sum_{j=L+1}^{N} \left(\beta_j + \gamma_j\right) \quad (14)$$

s.t.

$$\alpha_i \in [0, C_1] \qquad i \in 1 \ldots L \quad (15)$$

$$\beta_j \in [0, C_2],\ \gamma_j \in [0, C_2] \qquad j \in L+1, \ldots, N \quad (16)$$

and the prediction formula is (13).

Now the constraints are all bound constraints and the bound constrained *QP* can be solved very efficiently. In our experiments using the TRON algorithm (Lin & Moré, 1999), the optimization usually converges to global optimum within 10 seconds for 4000 variables, converging after about 10 to 20 iterations. As in standard SVM which employs L_1 norm penalty, the solutions to ρ-ε learning also enjoys a sparse structure.

4. Model Interpretation and Connections to Other Works

The principle idea for our model is to minimize the leave-one-out (LOO) cross validation error. This is embodied by both Gaussian process prediction formula and mixed classification-regression ρ-ε model.

4.1 LOO Gaussian Process prediction

Semi-supervised learning relies heavily on similarity between instances. Since kernels can also be viewed as description of similarity, it is interesting to interpret existing semi-supervised learning algorithms from the perspective of kernels, or Gaussian process. In (Zhu et al., 2003), RBF kernel W is used as similarity measure:

$$w_{ij} = \exp\left(-\left\|x_i - x_j\right\|^2 \Big/ 2\sigma^2\right) \tag{17}$$

Defining a decision vector $f = (f_L^T, f_U^T)^T$ with f_L fixed to the given labels 0/1, we minimize the weighted sum of the squared differences of f on each edge, also interpreted as a plausible cost for a one-dimensional embedding of nodes:

$$\sum_{i,j=1}^{N} w_{ij}(f_i - f_j)^2 \tag{18}$$

Let $D = diag(d_1, ..., d_N)$, where $d_i = \sum_{j=1}^{N} w_{ij}$ and define graph *Laplacian* as $L = D - W$. The algorithm essentially minimizes the quadratic form $f^T L f$. Suppose

$$L = \begin{pmatrix} L_{LL} & L_{LU} \\ L_{UL} & L_{UU} \end{pmatrix}$$

Then the closed-form solution is: $f_U = -L_{UU}^{-1} L_{UL} f_L$. (19)

Locally linear embedding (LLE) (Roweis & Saul, 2000) established cost functions on each node. Suppose for node x_i, the combined contribution from all other nodes is $\sum_{j \neq i} w_{ij} f_j$, subject to $\sum_j w_{ij} = 1$, $w_{ij} \geq 0$ (if requiring reconstruction of each point lying within the convex hull of its neighbors), and $w_{ij} = 0$ if x_j does not belong to the set of neighbors of x_i. Then we would like the weighted average to be close to f_i. So we have the following problem:

Minimize $$\sum_i \left(f_i - \sum_{j \neq i} w_{ij} f_j \right)^2 \tag{20}$$

Using notation above, it is essentially minimizing $f^T L^T L f$ and a similar closed-form solution can be given.

(Ham et al., 2004) pointed out that both Laplacian algorithm (18) and LLE are kernel PCA with kernels being the pseudo-inverse of L (denoted as $L^\dagger$) and $\lambda_{\max} I - L^T L$ (or its centered form) respectively, where $\lambda_{\max}$ is the largest eigenvalue of $L^T L$. In the minimization problems (18, 20) for classification tasks, it is $L^T L$ in LLE that plays a similar role as L in Laplacian method. To make $L^\dagger$ and $(L^T L)^\dagger$ valid covariance matrices of the associated Gaussian process, we add very small noise $\sigma^2 I$. Surprisingly, *from this perspective*, the prediction, say (19), does *not* make use of information *in the lower-right $U \times U$ sub-matrix of covariance matrix* (kernel) associated with unlabeled data. This is because for a multivariate Gaussian distribution of $(x_L^T, x_U^T)^T$ with zero mean and covariance

$$\begin{pmatrix} \Sigma_{LL} & \Sigma_{LU} \\ \Sigma_{UL} & \Sigma_{UU} \end{pmatrix}$$

the expected mean of x_U given x_L is $\Sigma_{UL}\Sigma_{LL}^{-1} x_L$. What (18) seeks is the x_U which minimizes the logarithm of Gaussian probability distribution function with x_L fixed, and its solution (19) should be exactly the expected conditional mean of x_U. Note, however, that when these methods are used for semi-supervised learning, it is the Laplacian that is constructed from the unlabeled data, hence the kernel changes when the unlabeled data change.

One restriction of the Laplacian algorithm is the requirement of nonnegative similarity. Otherwise the Laplacian will no longer be positive semi-definite. In LLE, weights are decided by minimizing a function and it may also require positive weights for certain applications. As some kernel functions may assume negative value, this restriction precludes a rich source of similarity functions. Using our leave-one-out Gaussian process prediction, we can obtain a naturally scaled formula for combining the contributions from all other nodes while utilizing kernel functions, even those that may assume negative values. The objective function now becomes minimizing $\sum_i \left(f_i - b_i^T f\right)^2$ with respect to f_U in $f = (f_L^T, f_U^T)^T$, clamping f_L to given labels ±1 or 0/1. This LOO Gaussian process prediction algorithm makes it possible to utilize general kernels as similarity for graph algorithms.

4.2 LOO Mixed ρ-ε Learning

It is interesting to investigate the model's behavior as an interpolation between putting unlabeled data on similar footing as the labeled data and ignoring unlabeled data completely. In min-cut class graph algorithms, the reweighting of labeled and unlabeled data can usually be carried out by modifying the weight between nodes, adjusting the original weight by a factor η $(0 \leq \eta \leq 1)$ when an edge connects unlabeled nodes. The simplest corresponding manipulation in our model is to convert matrix B as follows:

$$B = \begin{pmatrix} B_{LL} & B_{LU} \\ B_{UL} & B_{UU} \end{pmatrix} \rightarrow B' = \frac{1}{1+\eta} \begin{pmatrix} B_{LL} & \eta B_{LU} \\ B_{UL} & \eta B_{UU} \end{pmatrix}$$

In general cases, $\eta \leq 1$ as unlabeled points are less reliable and less informative than labeled data.

Another method is to tune ε to provide the tradeoff. By applying Karush-Kuhn-Tucker (KKT) conditions, we have:

$$\beta_j \left(\varepsilon + \xi_j - b_j^T y + y_j\right) = 0$$
$$\gamma_j \left(\varepsilon + \xi_j' + b_j^T y - y_j\right) = 0$$

When ε is very large, the constraints (4) and (5) are unlikely to be active, so β_j and γ_j are almost certainly 0. The only remaining constraints are on labeled data. In this sense, tuning ε can re-weight the effect of unlabeled data on our model.

If $\beta_j \approx 0$ and $\gamma_j \approx 0$, then $Rx \approx R_1\alpha$ and the prediction formula is:

$$y = KR_1\alpha \tag{21}$$

Substituting (21) into (14), we have a simplified model:

$$\textit{Maximize} \qquad -\frac{1}{2}\alpha^T R_1^T K R_1 \alpha + \sum_{i=1}^{L}\alpha_i \tag{22}$$

$$\textit{s.t.} \qquad \alpha_i \in [0, C_1] \quad i \in 1...L \tag{23}$$

Specifically, for data point x, the prediction

$$y_x = k_x^T R_1 \alpha = \sum_{i=1}^{L} k_x^T b_i \cdot \alpha_i c_i = \left(\sum_{i=1}^{L} b_i \cdot \alpha_i c_i\right)^T k_x \tag{24}$$

where k_x is the $N \times 1$ vector $\left(k(x, x_1), ..., k(x, x_N)\right)^T$ corresponding to unlabeled instance x in K. Since $\sum_{i=1}^{L} b_i \cdot \alpha_i c_i$ is independent of x, this result shows that the optimal solution must lie in the span of kernels evaluated at the labeled *and unlabeled* data points. In comparison, according to the *Representer Theorem* (Schölkopf & Smola, 2002), the solution of the Support Vector Machine can be represented entirely with kernels associated with labeled data, even when unlabeled data is present. Note also that the Representer Theorem, applied to ρ-ε learning also shows that the solution for ρ-ε learning can be represented using only kernels associated with the labeled and unlabeled data and do not require kernels associated with unseen data. Hence, when additional examples are given for testing, the model can perform in an inductive fashion rather than only in the transductive style.

There is a natural connection between our mixed ρ-ε model and leave-one-out SVM (LOOSVM) (Herbrich, 2002). The LOOSVM is formulated as:

$$\textit{Minimize} \qquad \sum_{i=1}^{L}\xi_i \tag{25}$$

$$\textit{s.t.} \qquad c_i\left(\sum_{j=1, j\neq i}^{L} \alpha_j K(x_j, x_i) + b\right) \geq 1 - \xi_i \quad i \in 1...L \tag{26}$$

$$\xi_i \geq 0 \quad \alpha_i \geq 0 \qquad i \in 1...L \tag{27}$$

The constraints for our mixed ρ-ε model has the form

$$c_i b_i^T y \geq 1 - \xi_i$$

which is different to (26) but is still a leave-one-out prediction constraint. The objective function (2) has an additional regularization term compared to (25) which just minimizes the L_1 norm of penalty. Despite these differences, both LOOSVM and our model are minimizing the leave-one-out error, using similar frameworks to produce the prediction of one instance's label assuming the others' labels are known.

5. Efficiency Considerations

Two costly operations involved in this algorithm are matrix inversion for building the *GP* prediction coefficient matrix B, and the calculation of the *Hessian* matrix for quadratic programming. To calculate B, one needs $k_i^T T_i^{-1}$ for all $i = 1...N$ so the bottleneck lies in T_i^{-1}. Here we can apply *Matrix Inversion Lemma* in its simplest form:

$$(A + uv^T)^{-1} = A^{-1} - A^{-1}u \cdot vA^{-1} / (1 + v^T A^{-1} u)$$

where A is a $N \times N$ invertible matrix, u and v are $N \times 1$ vectors. We first calculate T_1^{-1}. Noticing that T_{i+1} ($i \geq 1$) is different from T_i only by the i^{th} row and i^{th} column, we have $T_{i+1} = T_i + \mathbf{1}_i \cdot u_i^T + v_i \cdot \mathbf{1}_i^T$, where $\mathbf{1}_i$ is a $(N-1) \times 1$ vector with all elements being 0 except the i^{th} element being 1. So $T_2^{-1}, ..., T_N^{-1}$ can be calculated one after one. As each application of matrix inversion lemma has computational complexity $O(N^2)$, the total cost of building B is $O(N^3)$.

The *QP* itself can be solved efficiently. The calculation of the *Hessian* matrix in (14) with R defined in (12) can also be done quickly. But if one-against-all approach is adopted for multi-class tasks, calculating $R^T K R$ for multiple times can be costly. However, as $R = (R_1, R_2, -R_2)$ and only R_1 is related to labels which change in the one-against-all process, we only need to calculate $R_1^T K R_2$ and $R_1^T K R_1$ repeatedly. They both cost $O(U^2 L)$, under the typical settings of semi-supervised learning that $L \ll U$. Therefore, both can be done efficiently. The $O(U^3)$ step for producing $R_2^T K R_2$ needs to be done only once.

6. Experimental Results

We evaluate our algorithm on eight datasets. Three are from the 20 newsgroup dataset (19997 instances) (Lang, 1995): **baseball-hockey** (1993 instances / 2 classes), **religion-atheism** (1424 / 2), and **pc-mac** (1943 / 2). Two are from handwritten digits recognition task: **odd-even** and **10 digits** (10-way), for which we used a simplified form available at http://www.cs.colorado.edu/~grudic/data, with 196 features each ranging between 0 and 9. We extracted 200 examples from each class (so 2000 instances in all). The other three datasets are synthetic **2-**

spiral (194 / 2 / 2 features), **wine** (178 / 3 / 13) (UCI Repository), and **yeast** (1484 / 10 / 8). All input feature vectors are normalized to have length 1, except the three 20 newsgroup datasets and the 2-spiral dataset. We randomly pick training sets and test prediction accuracy on the remaining examples under the constraint that all classes, including each digit as subclasses for the odd-even dataset, must be present in labeled set. For multi-class tasks, the one-against-all heuristic is used and we pick the class whose function has the largest value. The program is written in Matlab, except the *QP* solver which is implemented more efficiently by applying TRON algorithm provided by the *Toolkit for Advanced Optimization* (TAO) (Benson et al., 2004).

We compare our algorithm with the SVM. To ensure that comparison is fair we use RBF kernel for both models. For SVM, we choose the RBF kernel bandwidth σ_{SVM}^2 (as in (17)) and tradeoff parameter C that yield the best average *generalization* performance, i.e., maximizing the average testing accuracy of all choices of training dataset size. For kernel self-consistent labeling, we fix Gaussian noise variance to 10^{-4}. After some experiments, we find that $C_1 = C_2 = 10$, $\varepsilon = 1$ performs well throughout all datasets and all possible number of labeled data. In fact, the performance is not very sensitive to the exact value of C_1 and C_2, as long as they are at a proper magnitude. This is similar to the property of tradeoff parameter C in SVM. However, the RBF kernel bandwidth σ^2 needs to be tuned for each dataset. To demonstrate that good values for σ^2 can be found with reasonable effort, we experimented with four bandwidths for all datasets: $\sigma^2 = \sigma_{SVM}^2 / r$, where r = 1, 10, 20, 100. In practice, when enough labeled data exists, k-fold cross validation can be used to select good settings for the parameters.

Figure 1 to *Figure* 4 demonstrate the resulting accuracies. Each result is averaged over 30 trials of randomly picking labeled data. We observe that self-consistent labeling with r = 20 outperformed SVM for 6 out of the 8 datasets. For the other datasets, *yeast* and 10-*digit* (both 10-way multi-class tasks), we tried more settings to see if settings that can outperform the SVM exist and indeed we found better settings as shown in *Figure* 4. For yeast dataset, r = 2 yields consistently better results, while for 10-digit dataset, the result given by $r = 5^{-1}$ is of little statistical difference from SVM in most cases. Whether the preference for smaller r is due to the larger number of classes or just due to the property of the dataset itself requires further investigation, because the effect of one-against-all heuristic on kernel self-consistent labeling is less known. One avenue of investigation, as future work, is to try to transplant strategies that extend SVM for multi-class classification to our algorithm.

For all the datasets except the 10-digit dataset, we find that $r > 1$ yields better results than $r < 1$ (not shown because of the already consistently poor performance at r = 1 and the trend shown). In normal supervised learning, larger training set size usually allows the use of smaller bandwidths to improve performance. This suggests that kernel self-consistent labeling is able to take advantage of the unlabeled data and use kernels with smaller bandwidth effectively.

It is interesting to investigate the influence of the ratio between labeled positive and negative data. For all the 5 binary tasks, the performance of SVM grows worse under the sequence #postive/#negative = 10/10, 10/20, 10/30, 10/40, though the total number of training data is increasing. Kernel self-consistent labeling has a different trend when using unbalanced training data in the 2-spiral dataset. With the number of labeled data growing in that sequence, albeit increasingly unbalanced, there is consistent improvement in accuracy.

The σ_{SVM}^2 and optimal C for SVM are given in Table 1.

Table 1. Optimal parameters for SVM

	baseball - hockey	religion - atheism	pc - mac	odd - even
σ_{SVM}^2	40000	169000	10^5	0.5
C	100	10^6	10^3	10^5
	10 digits	2-spiral	wine	yeast
σ_{SVM}^2	0.05	5×10^{-4}	50	0.5
C	100	10^4	10^4	100

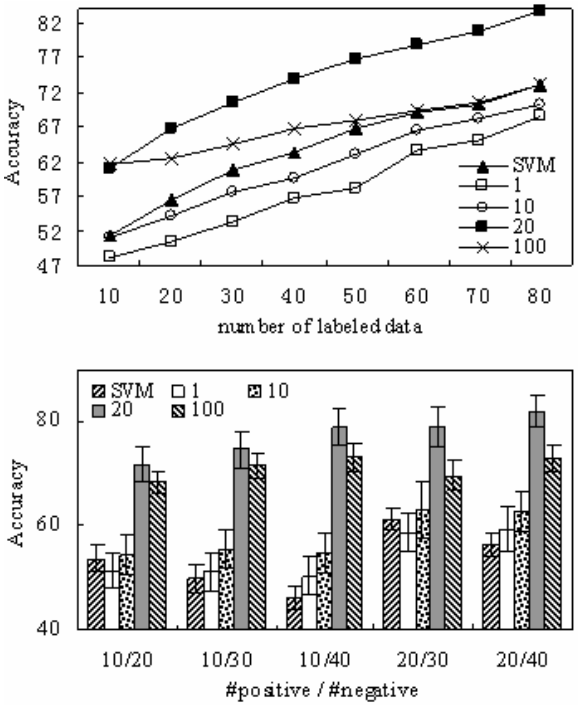

Figure 1. Accuracy results (in percentage) for 2-spiral data. In the upper figure, there is equal number of positive and negative labeled data. The lower figure shows different numbers of pos/neg labeled data. 1, 10, 20, 100 in legend are the values of r.

Note: some results are inaccurate due to bugs found after ICML submission.

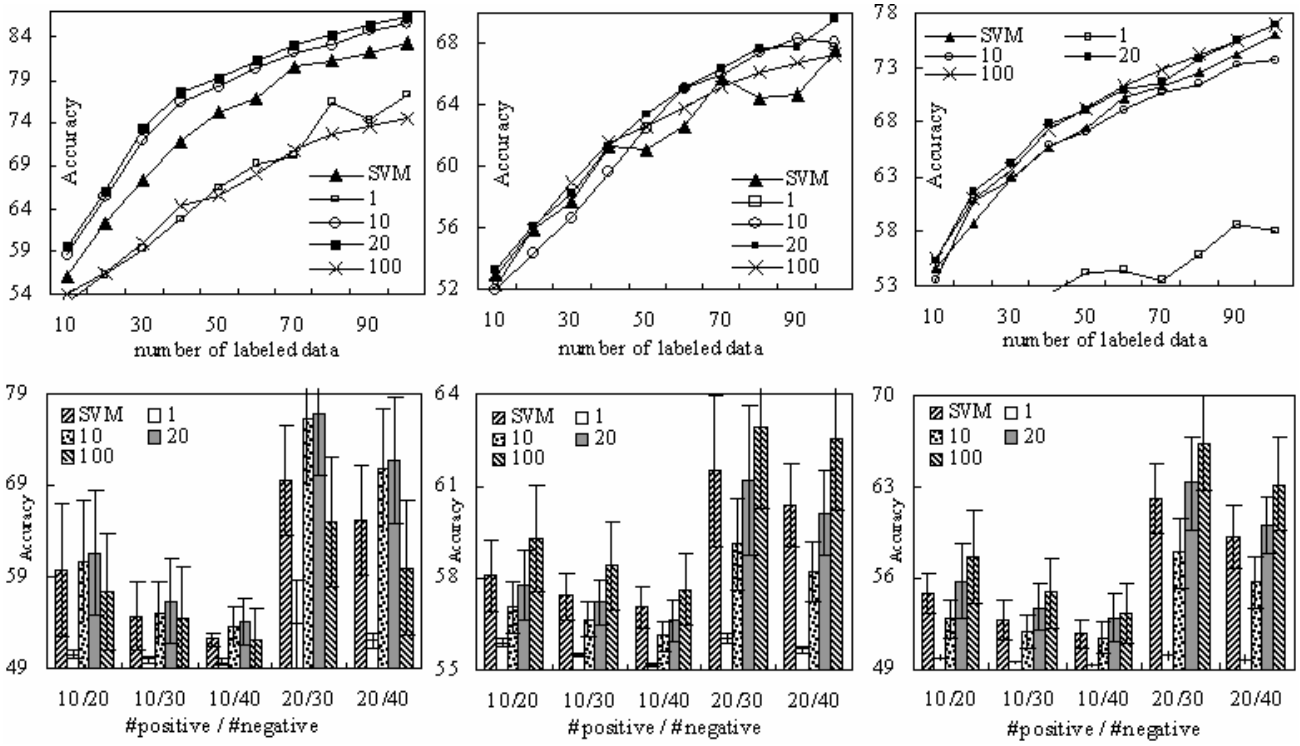

Figure 2. Accuracy results for baseball vs hockey (left), religion vs atheism (middle), and pc vs mac (right). The upper middle figure has no curve for $r = 1$ because its accuracy is always below 52%.

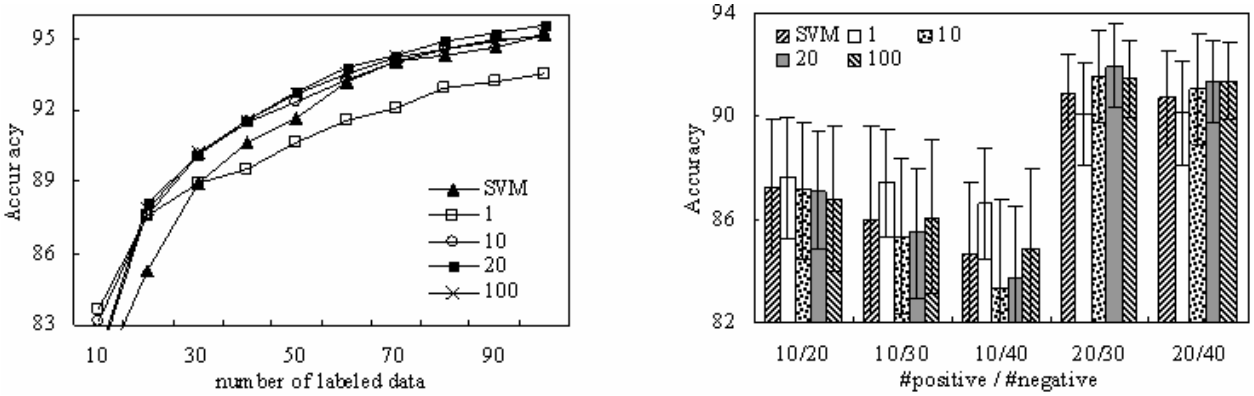

Figure 3. Accuracy results for odd vs even.

7. Conclusions

In this paper, we describe a new method of utilizing unlabeled data for classification based on leave-one-out Gaussian process prediction and mixed classification-regression. We motivate this approach using the principles of low training error on labeled data and low LOOCV error on whole dataset. The essence of the algorithm is to view any generic valid kernel or *Gram* matrix as covariance matrix of a Gaussian process and then establish the leave-one-out prediction formula based on conditional expectation under multivariate Gaussian distribution. This formula connects all instances and can also possibly be interpreted as a similarity measure. Built upon this connection, a support vector style model is proposed by minimizing the hinge loss on labeled data for classification, and the ε-insensitive cost on unlabeled data for regression. The model reduces to a local-optima-free quadratic programming problem with only bound constrain and can be solved efficiently. Both the *Hessian* matrix and *GP* prediction coefficient matrix can be computed

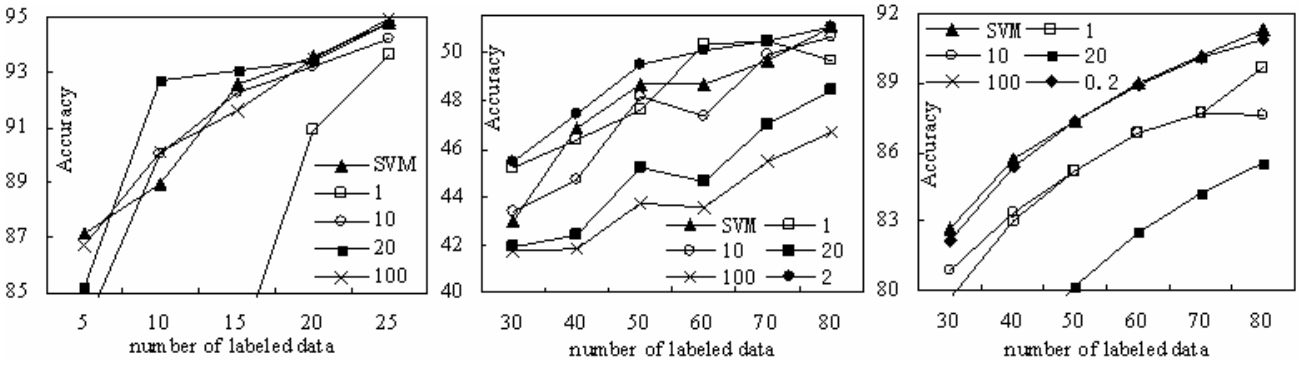

Figure 4. Accuracy results for wine (left), yeast (middle), and 10-digit (right). For these multi-class tasks, we do not show the influence of different ratio of labeled data from each class. There is no line for $r = 100$ in the right figure (10-digit) because its accuracy is always below 80%.

efficiently, by matrix block decomposition and matrix inversion lemma respectively. Experimental results illustrate the advantage of semi-supervised learning using kernel self-consistent labeling when compared to using the SVM.

Further work needs to be done to extend the current binary classification settings so that it uses an in-built mechanism for multi-class classification, rather than general one-against-all heuristics. Other variants of SVM such as γ-SVM or other loss functions like Huber's robust loss (Schölkopf & Smola, 2002) can also be utilized under the framework of self-consistency. Finally, it is desirable to explore and establish, under the settings of semi-supervised learning, stronger theoretical connections with RKHS models and learning theory, where many state-of-the-art learning models are rooted.

References

Benson, S., McInnes, L., Moré, J., & Sarich, J. (2004). *TAO User Manual* (*Revision* 1.7) ANL/MCS-TM-242. http://www.mcs.anl.gov/tao.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts**.** *Proc. of the 18th Int'l Conf. on Machine Learning* (pp. 19-26).

Chung, F. R. K. (1997). *Spectral graph theory*. No. 92 in Regional Conference Series in Mathematics. American Mathematical Society.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2002). On kernel-target alignment. In *Advances in Neural Information Processing Systems*, 14, 367-373.

Ham, J., Lee, D., Mika, S., & Schölkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. *Proceedings of the 20th International Conference on Machine Learning* (pp. 290-297).

Joachims, T. (1999). Transductive inference for text Classification using support vector machines. *Proceedings of the 16th International Conference on Machine Learning* (pp. 200-209).

Joachims, T. (2003). Transductive learning via Spectral Graph Partitioning. *Proceedings of the 20th International Conference on Machine Learning* (pp. 290-297).

Lin C.-J., & Moré, J. (1999). Newton's method for large bound-constrained optimization problems. *SIOPT*, 9(4), 1100-1127.

Herbrich, R. (2002). *Learning kernel classifiers: theory and algorithms*. Cambridge, Mass.: MIT Press.

Roweis, S., & Saul, K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2323-2326.

Seeger, M. (2004). Gaussian processes for machine learning. *Int'l Journal of Neural Systems*, 14(2), 69-106.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*: *support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press.

Silverman, B. (1984). Spline smoothing: the equivalent variable kernel method. *Annals of Statistics*, 12, 898-916.

Sollich, P., & Williams, C. (2004). Using the equivalent kernel to understand Gaussian Process regression. In *Advances in Neural Information Processing Systems*, 17.

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

UCI (2000). Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning* (pp. 912-919).