Journal of Machine Learning Research 10 (2009) 183-206 Submitted 1/08; Revised 9/08; Published 2/09

Using Local Dependencies within Batches to Improve Large Margin Classifiers

Volkan Vural VVURAL@ECE.NEU.EDU
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Glenn Fung GLENN.FUNG@SIEMENS.COM
Balaji Krishnapuram BALAJI.KRISHNAPURAM@SIEMENS.COM
Computer Aided Diagnosis and Therapy
Siemens Medical Solutions
Malvern, PA 19355

Jennifer G. Dy JDY@ECE.NEU.EDU
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

Bharat Rao BHARAT.RAO@SIEMENS.COM
Computer Aided Diagnosis and Therapy
Siemens Medical Solutions
Malvern, PA 19355

Editor: Lyle Ungar

Abstract

Most classification methods assume that the samples are drawn independently and identically from an unknown data generating distribution, yet this assumption is violated in several real life problems. In order to relax this assumption, we consider the case where batches or groups of samples may have internal correlations, whereas the samples from different batches may be considered to be uncorrelated. This paper introduces three algorithms for classifying all the samples in a batch jointly: one based on a probabilistic formulation, and two based on mathematical programming analysis. Experiments on three real-life *computer aided diagnosis* (CAD) problems demonstrate that the proposed algorithms are significantly more accurate than a naive support vector machine which ignores the correlations among the samples.

Keywords: batch-wise classification, support vector machine, linear programming, machine learning, statistical methods, unconstrained optimization

1. Introduction

Most classification systems assume that the data used to train and test the classifier are drawn from an independent and identically distributed underlying distribution. For example, samples are classified one at a time in a *support vector machine* (SVM), thus the classification of a particular test sample does not depend on the features from any other test sample. Nevertheless, this assumption

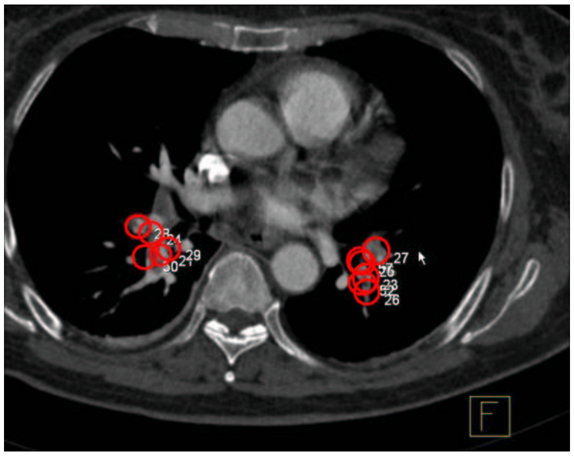


Figure 1: Two pulmonary emboli (blood clots) as they are detected by the Candidate Generation algorithm in a CT image. The candidates are shown as five circles for the left embolus & six circles for the right embolus. The disease status of spatially overlapping or proximate candidates is highly correlated.

is commonly violated in many real-life problems where sub-groups of samples have a high degree of correlation amongst both their features and their labels.

A sample problem with the characteristics described above is *computer aided diagnosis* (CAD), where the goal is to detect structures of interest to physicians in medical images: for example, to identify potentially malignant tumors in computed tomography (CT) scans, X-ray images, etc. In an almost universal paradigm for CAD algorithms, this problem is addressed by a three-stage system: (1) identification of potentially unhealthy candidates *regions of interest* (ROI) from a medical image, (2) computation of descriptive features for each candidate, and (3) classification of each candidate (e.g., normal or diseased) based on its features.

Figure 1 displays a CT image of a lung showing circular marks that point to potential diseased candidate regions that are detected by a CAD algorithm. There are five candidates on the left and six candidates on the right (marked by circles) in Figure 1. Descriptive features are extracted for each candidate and each candidate region is classified as normal or abnormal. In this setting, correlations exist among the candidates belonging to the same batch (image, in the case of CAD) both in the training data set and in the unseen testing data. Further, the level of correlation is a function of the pairwise distance between candidates: the disease status (class-label) of a candidate is highly correlated with the status of other spatially proximate candidates, but the correlations decrease as the distance is increased. Most conventional CAD algorithms classify one candidate at a time, ignoring the correlations amongst the candidates in an image. By explicitly accounting for the correlation structure between the labels of the test samples, classification accuracy can be improved.

In this paper, we present two large margin classifiers and a batch-wise version of logistic regression that take into account the local correlations among samples in a batch. We use the term *batch-wise classification* to mean training and testing classifiers in *batches* or groups (i.e., samples in the same batch are learned and classified collectively). Our experiments on three real CAD problems show that indeed incorporating the local dependencies among samples within a batch improve the classification accuracy significantly compared to standard SVMs.

Beyond the domain of CAD applications, our algorithms are quite general and may be used for batch-wise classification problems in many other contexts. In general, the proposed classifiers can be used whenever data samples are presented in independent batches. In the CAD example, a batch corresponds to an image and strong correlations among candidate samples are based on spatial proximity within an image. A similar relationship structure is true with geographical classification of regions in satellite images; an image would be a batch and spatial proximity corresponds to correlations among the pixel instances. In other contexts, a batch may correspond to data from the same hospital, the patients treated by the same doctor or nurse, etc. In non-medical contexts, a batch can be transactions from the same person, or documents from the same author. Another example is in credit card transaction approval where a single transaction has to be classified by taking into account both the entire set of available training transactions and other transactions made by the same person recently during the testing phase. In this credit card example, it might be preferable to test the recent transaction of this same person as a batch collectively rather than singly.

1.1 Related Work

We are not aware of previous work that exploit internal correlations among the samples to improve the performance of standard classifiers (such as, SVM). However, there exist classification algorithms that incorporate special relationships among the samples (the Markov property) into their model formulation. In natural language processing (NLP), *conditional random fields* (CRF) (Lafferty et al., 2001) and recently *maximum margin Markov* (MMM) networks (Taskar et al., 2004) are used to identify part-of-speech information about words by using the context of nearby words. CRF are also used in similar applications in spoken word recognition. For images, Markov random fields (MRF) (Besag, 1974; Geman and Geman, 1984) are applied to capture correlation among neighborhood pixels.

However, while these Markov models are also fairly general algorithms, they are both computationally demanding and are not easy to implement for problems where the relationship structure among the samples is in any form other than a linear chain (as in the text and speech processing applications). Certainly, their application would be difficult in many large-scale medical applications where run time requirements would be quite severe. For example, in the CAD applications shown in our experiments, the run-time of the testing phase usually has to be less than a second in order that the end user's (radiologist's) time would not be wasted.

Our algorithm is also related to the *multiple instance learning* (MIL) problem, where one is given bags (batches) of samples; class labels are provided only for the bags, not for the individual samples. A bag is labeled positive if we know that at least one sample from it is positive, and a negative bag is known to not contain any positive sample. In this manner, the MIL problem also encodes a form of prior knowledge about correlations between the labels of the training instances.

There are two differences between our algorithm and MIL. First, we want to classify each instance (candidate) in our algorithm; unlike MIL, we are not only trying to label a bag of related instances. Second, unlike the MIL problem which treats all the instances in a bag as equally related to each other, we account for more fine grained differences in the level of correlation between samples (via the similarity matrix $\mathbf{R}$).

Local learning algorithms such as, Wu and Schlkopf (2007) and Bottou and Vapnik (1992), also capture the correlation among data points. However, one major difference between our method and local learning based methods is that our methods build a global classifier that combines the global information extracted from the entire training data with the additional side-information extracted from correlations in the same batch, whereas the local learning algorithms rely on the local information only. Another major difference is that local learning algorithms classify a particular point based on its neighbors in the feature space, therefore the captured correlation is limited to the neighbors of that particular point. On the other hand, the methods that we propose in this paper can incorporate *any* type of correlation information that is not fully encoded in the features. The same difference is also valid between our method and semi-supervised algorithms such as, Vural et al. (2008) and Zhou et al. (2003) that exploit the idea of using local information in semi-supervised learning. The graph based semi-supervised algorithms such as, Krishnapuram et al. (2005) and Zhu et al. (2003), can be considered to be related to the methods introduced in this paper in the sense that they build weighted graphs based on the correlation among data points. However, the targeted problems in semi-supervised learning and this paper are substantially different. In semi-supervised learning problems, the goal is to make use of unlabeled data to improve classification accuracy. In our methods, on the other hand, we investigate the relations among the data points that are in the same batch only. These data points may not constitute a neighborhood in the feature space and every data point is labeled in our problem.

1.2 Organization of the Paper

After briefly describing the notation in Section 2, we build two SVM based methods for classifying batches of correlated samples in Section 3. We also provide a probabilistic interpretation of our approach in Section 4.

Unlike the previous methods such as CRF, MMM and MRF, the proposed algorithms are easy to implement for arbitrary relationships between samples, and further we are able to run these fast enough to be viable in commercial CAD products. In Section 5, we provide experimental evidence from three different CAD problems to show that the proposed algorithms are more accurate in terms of the metrics appropriate to CAD as compared to a naive SVM which is routinely used for these problems as the state-of-the-art in the current literature and commercial products. We conclude with a review of our contributions in Section 6.

2. Notations

Throughout this paper, we will use the following notations. The capital letters shown in bold font represent matrices whereas lower case letters in bold correspond to vectors. The notation $\mathbf{A} \in \mathbb{R}^{m\times n}$ will signify a real $m \times n$ matrix. For such a matrix, $\mathbf{A}'$ will denote the transpose of $\mathbf{A}$ and $\mathbf{A}_i$ will denote the i-th row of $\mathbf{A}$. All vectors will be column vectors.

For $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_p$ denotes the p-norm, $p = 1, 2, \infty$. A vector of ones in a real space of arbitrary dimension will be denoted by $\mathbf{e}$. Thus, for $\mathbf{e} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{e}'\mathbf{y}$ is the sum of the components of $\mathbf{y}$. A vector of zeros in a real space of arbitrary dimension will be denoted by 0. The identity matrix of arbitrary dimension will be denoted by $\mathbf{I}$. A *separating hyperplane*, with respect to two given point sets **A** and **B**, is a plane that attempts to separate $\mathbb{R}^n$ into two half spaces such that each open half space contains points mostly of **A** or **B**.

3. A Mathematical Programming Approach

In this section, we formulate the problem of learning for batch-wise prediction as an SVM-like mathematical program.

3.1 Standard Support Vector Machine

In a standard SVM hyperplane classifier, $f(\mathbf{x}) = \mathbf{x}'\mathbf{w} - \gamma$ is learned from the training instances individually, ignoring the correlations among them. Consider the problem of classifying m points in the n-dimensional real space $\mathbb{R}^n$, represented by the $m \times n$ matrix $\mathbf{A}$, according to class membership of each point $\mathbf{x}_i$ (i^{th} row of $\mathbf{A}$) in the classes $\mathbf{A}+$, $\mathbf{A}-$ as specified by a given $m \times m$ diagonal matrix $\mathbf{D}$ with $+1$ or -1 along its diagonal. The standard 1-norm support vector machine with a linear kernel (Vapnik, 1995; Bradley and Mangasarian, 1998) is given by the following linear program with parameter $\nu > 0$:

$$\begin{aligned} \min_{(\mathbf{w},\gamma,\xi,\mathbf{v})\in\mathbb{R}^{n+1+m+n}} \quad & \nu\mathbf{e}'\xi + \mathbf{e}'\mathbf{v} \\ \text{s.t.} \quad \mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{e}\gamma) + \xi & \geq \mathbf{e} \\ \mathbf{v} \geq \mathbf{w} & \geq -\mathbf{v} \\ \xi & \geq 0 \end{aligned} \tag{1}$$

where, $\xi \in \mathbb{R}^m$ is a vector of slack variables, ν is the cost parameter, and at a solution, $\mathbf{v} = |\mathbf{w}|$ is the absolute value of $\mathbf{w}$. We used the 1-norm linear programming formulation instead of the more popular 2-norm quadratic programming (QP) formulation since the 1-norm formulation is known to produce solutions that depend on fewer features. In other words, by solving the 1-norm formulation, an implicit feature selection is obtained automatically. This is an advantage in CAD applications because the initial pool typically contains a large set of experimental features, although often only a small subset of these features is typically found to be very useful for the classification task at the end of design (i.e., training) stage. It is important to note that all the batch-based classification approaches presented in this paper are equally applicable even when the standard QP SVM formulation is considered.

3.2 Batch Support Vector Machine

The idea in our model is to encode that the classification of any sample (i.e., data point) depends on two factors: (a) the features that encode information about each candidate sample in a batch; and (b) additional side-information—not already fully encoded in the features—that is available to describe the level of correlations among the set of samples within a batch.[1]

Let the covariance matrix $\mathbf{R}^j$ represent the correlations among the samples withing the j^{th} batch. As mentioned above, in CAD applications, $\mathbf{R}^j$ can be defined in terms of $\mathbf{S^j}$, the matrix of Euclidean distances between candidates inside a medical image (from a patient) as:

1. We define a batch to be a set of possibly correlated samples that occur together naturally, for example, the set of all candidate locations from the image of the same patient.

$$\mathbf{R}^j_{pq} = \begin{cases} 0 & , p = q \\ \exp(-\alpha \mathbf{S}^j_{pq}) & , o.w. \end{cases}$$

with $\alpha > 0$ as a model parameter. Accordingly, $\mathbf{B}^j \in \mathbb{R}^{m_j \times n}$ represents the m_j training points that belong to the j^{th} batch and the labels of these training points are represented by the $m_j \times m_j$ diagonal matrix $\mathbf{D}^j$ with positive or negative ones along its diagonal. Using the new notations, the standard SVM set of constraints: $\mathbf{D}(\mathbf{A}\mathbf{w} - \mathbf{e}\gamma) + \xi \geq \mathbf{e}$ can be presented as:

$$\mathbf{D}^j(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma) + \xi^j \geq \mathbf{e}, \text{ for } j = 1, \ldots, k. \tag{2}$$

Here, k is defined as the number of batches in $\mathbf{A}$ and the vector of slack variables, ξ, in Equation (1) corresponds to $\xi = [\xi^{1'}, \xi^{2'}, \ldots, \xi^{k'}]'$. In Equation (2), every data point in $\mathbf{B}^j$ sets a constraint on $\mathbf{w}$ and γ *independently*, that is, the i^{th} constraint is set by only the i^{th} data point, $\mathbf{B}^j_i$, ignoring the correlations between $\mathbf{B}^j_i$ and the rest of the samples in $\mathbf{B}^j$. Therefore, we modify the standard SVM set of constraints in order to take into account the correlations among samples in the same batches, using the $\mathbf{R}^j$ matrix as:

$$\mathbf{D}^j \left[\left(\mathbf{I} + \theta \mathbf{R}^j \right) (\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma) \right] + \xi^j \geq \mathbf{e}, \text{ for } j = 1, \ldots, k. \tag{3}$$

Here, θ is a parameter that helps us control the contribution of the correlation information encoded by $\mathbf{R}^j$. We can find the optimum value of θ by tuning over a validation set. However, we also present a method where we can learn θ automatically in Section 3.3.

In Equation (3), the class membership prediction for any single sample in batch j is a weighted average of the batch members' prediction vector $\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma$, and the weighting coefficients depend on the pairwise similarity between samples. Using constraints (3) in the SVM formulation (1), we obtain the optimization problem for learning *BatchSVM* with parameters ν and θ:

$$\begin{aligned} & \min_{(\mathbf{w},\gamma,\xi,\mathbf{v}) \in \mathbb{R}^{n+1+m+n}} \nu \mathbf{e}'\xi + \mathbf{e}'\mathbf{v} \\ \text{s.t.} \quad & \mathbf{D}^j \left[\left(\theta \mathbf{R}^j + \mathbf{I} \right) (\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma) \right] + \xi^j \geq \mathbf{e}, \quad \text{for } j = 1, \ldots, k \\ & \mathbf{v} \geq \mathbf{w} \geq -\mathbf{v} \\ & \xi \geq 0. \end{aligned} \tag{4}$$

When $\theta = 0$, the constraints in Equation (4) become standard SVM constraints. Therefore, if the information provided in $\mathbf{R}^j$ matrix does not help our algorithm improve the accuracy performance of SVM, we expect our method to turn into the standard SVM.

Unlike standard SVMs, the hyperplane ($f(\mathbf{x}) = \mathbf{w}'\mathbf{x} - \gamma$) produced by *BatchSVM* is *not* the final decision function. We refer to $f(\mathbf{x})$ as a pre-classifier that will be used in the next stage to make the final decision on a batch of instances. While testing an arbitrary data point $\mathbf{x}^j_i$ in batch $\mathbf{B}^j$, the *BatchSVM* algorithm accounts for the pre-classifier prediction $\mathbf{w}'\mathbf{x}^j_p - \gamma$ for every member in the batch. The final prediction $\hat{f}(\mathbf{x}^j_i)$ is given by:

$$sign(\hat{f}(\mathbf{x}^j_i)) = sign(\mathbf{w}'\mathbf{x}^j_i - \mathbf{e}\gamma + \theta \mathbf{R}^j_i \left[\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma \right]). \tag{5}$$

Consider the two dimensional example in Figure 2, showing batches of training points. The data points that belong to the same batch are indicated by the elliptical boundaries in the figure. Figure 2b displays the correlations amongst the training points given in Figure 2a using an edge.

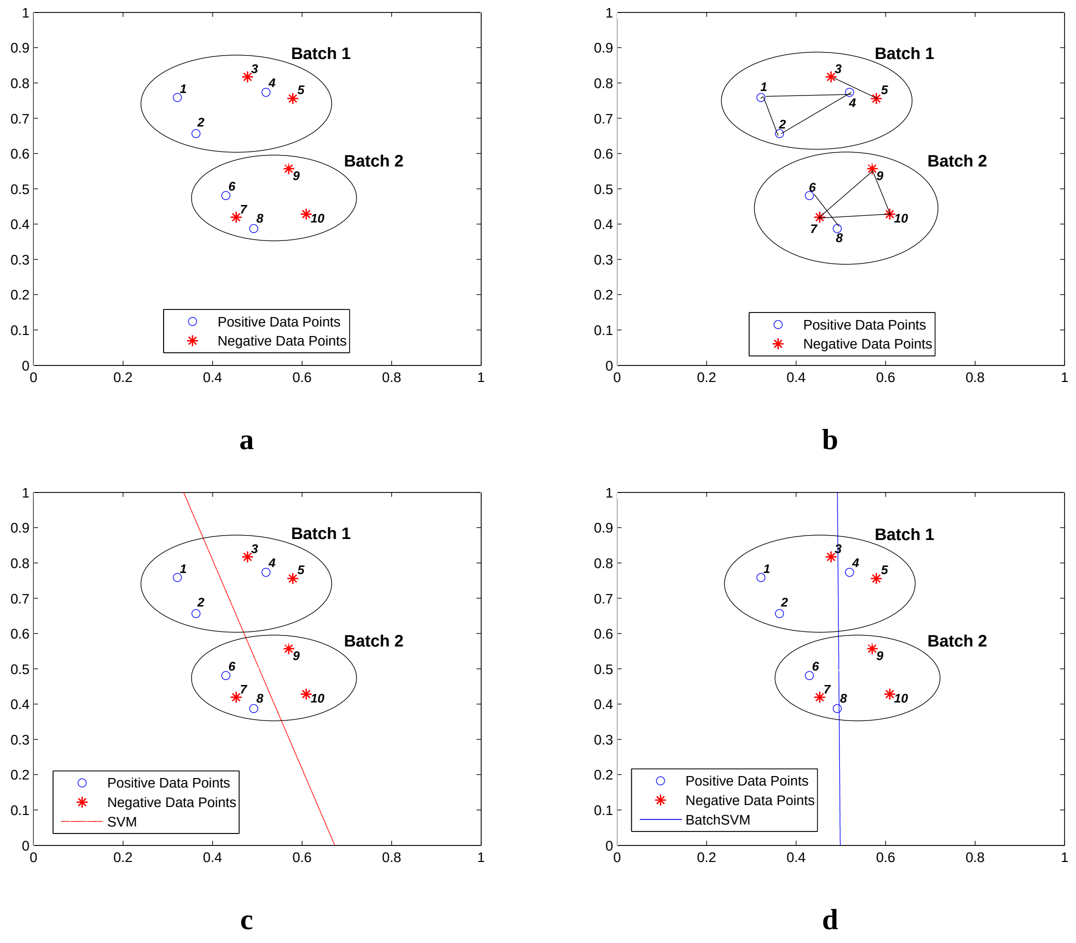


Figure 2: An illustrative example for batch learning. **a**) Training data points are displayed in batches. **b**) Relations within training points are displayed as a linked graph. **c**) Classifier produced by SVM. **d**) Pre-classifier produced by *BatchSVM*. Unlike standard SVMs, the hyperplane, $f(\mathbf{x})$, produced by *BatchSVM* (preclassifier) is not the decision function. Instead, the decision of each test sample $\mathbf{x}_i$, is based on a weighted average of the $f(\mathbf{x})$ values for the points linked to $\mathbf{x}_i$.

In Figure 2c, the hyperplane $f_{svm}(\mathbf{x})$ is the final decision function for standard SVM and gives the results displayed in Table 1 where we observe that the fourth and the seventh instances are misclassified. In Figure 2d, the pre-classifier produced by *BatchSVM*, $f_{batch}(\mathbf{x})$ gives the results displayed in the fifth column of Table 1 for the training data. If this pre-classifier were to be considered as the decision function, then three training points would be misclassified. However, during batch-testing (Equation 5), the predictions of those points are corrected as seen in the sixth column of Table 1.

Point	Batch	Label	SVM	Pre-classifier	Final classifier
1	1	+	0.2826	0.1723	0.1918
2	1	+	0.2621	0.1315	0.2122
3	1	-	-0.2398	**0.0153**	-0.0781
4	1	+	**-0.3188**	**-0.0259**	0.2909
5	1	-	-0.4787	-0.0857	-0.0276
6	2	+	0.2397	0.0659	0.0372
7	2	-	**0.2329**	**0.0432**	-0.0888
8	2	+	0.1490	0.0042	0.0680
9	2	-	-0.2525	-0.0752	-0.1079
10	2	-	-0.2399	-0.1135	-0.1671

Table 1: Outputs of the classifier produced by SVM, pre-classifier and the final classifier produced by *BatchSVM*. The outputs are calculated for the data points presented in Figure 2. The first column of the table indicates the order of the data points as they are presented in Figure 2a and the second column specifies the corresponding labels. Misclassified points are displayed in bold. Notice that the combination of the pre-classifier outputs at the final stage corrects the mistakes.

3.3 A Faster Proximal Batch Approach

Support vector machines are known to be slow to train. In Fung and Mangasarian (2001), Fung and Mangasarian introduced proximal support vector machine which is a fast and simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations. To yield a faster training method, we present a batch approach to proximal support vector machine in this section.

Similar to the proximal support vector machine formulation (PSVM) in Fung and Mangasarian (2001), we can slightly modify the set of inequalities (3) and consider a Gaussian prior (instead of a Laplacian) for both the error vector ξ and the regularization term, to obtain the following unconstrained formulation:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \nu\frac{1}{2}\sum_{j=1}^{k}\|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2). \tag{6}$$

Note that formulation (6) is a strongly convex (quadratic) unconstrained optimization problem and it has a unique solution. Then, obtaining the solution to formulation (6) requires only solving a single system of linear equations; therefore, it should be substantially faster than the linear programming approach of the standard QP approach while maintaining approximate accuracy results (Fung and Mangasarian, 2001). Moreover, this simple formulation allows us to introduce a very small modification to learn the parameter θ in a very simple but effective manner:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}} \nu\frac{1}{2}\sum_{j=1}^{k}\|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2). \tag{7}$$

This formulation is strongly convex in all its variables $(\mathbf{w},\gamma,\theta)$ and it has a unique solution as well. The optimization problem (7) can be solved in many different ways, however we chose to use an alternating optimization approach similar to the one used in Fung et al. (2004), since it suits the structure of the problem perfectly and it converged rapidly in our experiments. It took nine iterations on average for the algorithm to converge in our experiments. The alternate optimization method used can be summarized as follows:

0 *initialization*: Given an initial θ_0, the parameter ν, the training data $A = \bigcup_{j=1}^{k} \mathbf{B}^j$ and the corresponding labels $\mathbf{D}$.

1. Solve optimization problem (6) for $\theta = \theta^{i-1}$ obtain $\mathbf{w}^i$ and γ^i.

2. For the obtained $\mathbf{w}^i$ and γ^i, solve the following linear system of equations:

$$\theta^i = \frac{-\sum_{j=1}^{k} M^j}{2\sum_{j=1}^{k} N^j}$$

where

$$M^j \quad = \mathbf{S}^{j\prime}\mathbf{Z}_j\mathbf{S}^j + 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{S}^j \text{ , for } \mathbf{S}^j = \left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right) \text{ and } \mathbf{Z}_j = (\mathbf{R}^{j\prime} + \mathbf{R}^j),$$

$$N^j \quad = \mathbf{Y}^{j\prime}\mathbf{Y}^j + \tfrac{1}{\nu} \text{ , for } \mathbf{Y}^j = \mathbf{R}^j\left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right).$$

This is an iterative method with two steps. At the first step, it finds the optimum values of $\mathbf{w}$ and γ $(\mathbf{w}_{opt},\gamma_{opt})$ for a given θ. At the second step, for the given $\mathbf{w}_{opt}$ and γ_{opt}, it finds the optimum value of θ (θ_{opt}), which will be used at the first leg of the next iteration. Appendix A.2 provides a derivation of the update equations. Initially, we assign a random value for θ, the procedure is repeated until it converges to the global optimum solution for $\mathbf{w}$, γ and θ.

4. A Probabilistic Batch Classification Model

Let $\mathbf{x}_i^j \in \mathbb{R}^n$ represent the n features for the i^{th} candidate in the j^{th} image, and let $\mathbf{w} \in \mathbb{R}^n$ be the weight parameters and $\gamma \in \mathbb{R}^1$ be the offset of some hyperplane classifier. Traditionally, classifiers label samples one at a time (i.e., independently) based only on the features that describe a given sample $\mathbf{x}_i^j$:

$$z_i^j \quad = \mathbf{w}'\mathbf{x}_i^j - \gamma \quad = (\mathbf{x}_i^j)'\mathbf{w} - \gamma \quad , z_i^j \in \mathbb{R}^1.$$

For example, in logistic regression, the posterior probability of the sample $\mathbf{x}_i^j$ belonging to class $+1$ is obtained using the sigmoid function $P(y_i^j = 1|\mathbf{x}_i^j) = \frac{1}{1+\exp(-\mathbf{w}'\mathbf{x}_i^j+\gamma)}$.

We claim that z_i^j is only a noisy observation of the underlying, unobserved variable $u_i^j \in \mathbb{R}^1$ that actually influences classification (as opposed to the traditional classification approach, where classification directly depends on z_i^j). Beside the features $\mathbf{x}_i^j$ that are available to influence the classification of a sample point (via u_i^j), we have side information about the extent of correlation amongst the quantities u_i^j. We mathematically model the side information as an a-priori guess or intuition about u_i^j even before we observe any $\mathbf{x}_i^j$ (therefore before z_i^j).

For example, in the context of CAD, features describe the shape, size, texture, and intensity profile of a candidate (sample) in an image. However, besides the above information encoded in the form of features that describe each candidate, we also know that correlation amongst the samples in an image (batch) purely based on the proximity of the spatial locations of samples in the j^{th} image. In CAD applications, this spatial adjacency induces correlations in the predictions for the labels, but in other applications the nature of side-information that accounts for the correlations would be different. Nevertheless, in general (in a domain independent way) we model this side-information as a Gaussian prior on u_i^j.

$$P(\mathbf{u}^j \in \mathbb{R}^{n_j}) \quad = N(\mathbf{u}^j|0, \mathbf{R}^j) \tag{8}$$

where n_j is the number of the candidates in the j^{th} image, and the covariance matrix $\mathbf{R}^j$ encodes the correlations. As mentioned above, in CAD applications, $\mathbf{R}^j$ can be defined in terms of $\mathbf{S}$, the matrix of Euclidean distances between candidates inside a medical image (from a patient) as $\mathbf{R}^j = \exp(-\alpha \mathbf{S})$, with $\alpha > 0$ as a model parameter.

Having defined a prior, next we define the likelihood as follows:

$$P(z_i^j|u_i^j) \quad = N(z_i^j|u_i^j, \sigma^2). \tag{9}$$

After observing $\mathbf{x}_i^j$ and therefore z_i^j, we can modify our prior intuition about $\mathbf{u}^j$ in (8), based on our observations from (9) to obtain the Bayesian posterior:

$$P(\mathbf{u}^j|\mathbf{z}^j) \quad = N\left(\mathbf{u}^j|(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I})^{-1}\mathbf{z}^j; (\mathbf{R}^{j^{-1}} + \tfrac{1}{\sigma^2}\mathbf{I})^{-1}\right) \tag{10}$$

The class-membership prediction for the i^{th} candidate in the j^{th} image is controlled exclusively by u_i^j. Let $\mathbf{B}^j \in \mathbb{R}^{m_j \times n}$ represent the m_j training points that belong to the j^{th} batch. Exact computation of the integral required for $P(\mathbf{y}^j = 1|\mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$ is analytically infeasible because of the nonlinear sigmoid terms. However, by using the *maximum-a-posteriori* (MAP) estimate of u_i^j, the prediction probability for class label, y_i^j is then determined approximately as:

$$\begin{gathered} P(y_i^j = 1|\mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2) \approx 1/\left(1 + \exp\left(-\mathbf{V}_i^j[\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma]\right)\right), \\ \text{where} \\ \mathbf{V}^j = \left[\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right]^{-1}. \end{gathered} \tag{11}$$

Note, however, that for each batch j, the probabilistic method requires calculating two matrix inversions to compute $\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1}$. Hence, training and testing using this method can be time consuming for large batch sizes. On the other hand, $\left(\mathbf{R}^j\theta + \mathbf{I}\right)$ that we introduced in Equation (3) is relatively less time consuming to compute.

In general, $\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1} \neq \left(\mathbf{R}^j\theta + \mathbf{I}\right)$. However, the intuition behind both are similar: *the class prediction for each sample is a weighted sum of the prediction values of the test sample and the samples that are correlated with the test instance*. In the appendix A.1, we show that when σ is small and we are considering a least squared loss function as in Section 3.3, the constraint (3) results in an approximation to the following constraint:

$$\mathbf{D}^j\left[\left(\mathbf{R}^{j^{-1}}\sigma^2 + \mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma)\right] + \xi^j \geq \mathbf{e}, \text{ for } j = 1, \ldots, k \tag{12}$$

when the sum of the labels of neighboring data points according to each row of $\mathbf{R}^j$ is the same or they have similar values. Note that, this is particularly true when $\mathbf{R}^j$ only contains information about correlations for a constant number of nearest neighbors for each sample (k-nearest neighbors) and when the assumption that neighboring segments should have similar labels holds.

4.1 Learning in this Model

For batch-wise prediction using (11), $\mathbf{w}, \alpha$ and σ^2 can be learned from a set of N training images via *maximum-a-posteriori* (MAP) estimation of $\mathbf{w}$ and *maximum-likelihood* estimation of α, σ^2 as follows:

$$[\hat{\mathbf{w}}, \hat{\gamma}, \hat{\alpha}, \hat{\sigma^2}] \quad = \arg\max_{\mathbf{w},\gamma,\alpha,\sigma^2} P(\mathbf{w}) \prod_{j=1}^{N} P(\mathbf{y}^j | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$$

where, $P(\mathbf{y}^j | \mathbf{B}^j, \mathbf{w}, \gamma, \alpha, \sigma^2)$ is defined as in (11) and $P(\mathbf{w})$ may be assumed to be Gaussian $N(\mathbf{w}|0, \lambda)$. The regularization parameter λ is typically chosen by cross-validation. General purpose numerical optimizers such as conjugate gradient algorithms may be used for the above optimization problem.

4.2 Intuition about Batch Classification

Equations (10) and (11) imply that $\mathbb{E}[\mathbf{u}^j | \mathbf{z}^j] = (\mathbf{R}^{j^{-1}} \sigma^2 + \mathbf{I})^{-1} \mathbf{z}^j$. In other words, the class membership prediction for any single sample is a weighted average of the noisy prediction quantity $\mathbf{z}^j$ (distance to the hyperplane), where the weighting coefficients depend on the pairwise Euclidean distances between samples. Hence, the intuition presented above is that we predict the classes for all the n_j candidates in the j^{th} image together, as a function of the features for all the candidates in the batch (here a batch corresponds to an image from a patient). In every test image, *each* of the candidates is classified using the features from *all* the samples in the image. Note that although we classify each instance in a batch at the same time, our algorithm outputs a class for each instance and the class membership for these samples need not be the same.

5. Experiments

In this section, we compare regular SVM, probabilistic batch learning (*BatchProb*), and *BatchSVM*. Note that in order to make the comparison consistent, *BatchProb* was implemented by using the same linear programming formulation as *BatchSVM* but using Equation (12) instead of Equation (3). We also compare PSVM based techniques introduced in Section 3: (1) the proximal version of batch learning presented in (6) where θ is determined as a parameter via cross-validation ($BatchPSVM - cv$), and (2) the alternating optimization method presented in (7), where θ is learned automatically ($BatchPSVM - learned$).

5.1 Implementation Details

We present the details of the similarity function that we used in our experiments and how we tuned the parameters of our methods in the next sections.

5.1.1 THE SIMILARITY FUNCTION

As mentioned earlier, the matrix $\mathbf{R}^j$ represents the level of correlation between all pairs of candidates from a batch (an image in our case) and it is a function of the pairwise-similarity between the candidates. One needs to produce the $\mathbf{R}^j$ matrix using the appropriate similarity metric that represents the pairwise-similarities the best for a particular application.

In our CAD applications, we have used the Euclidean distances between candidates inside a medical image as our similarity metric to define the covariance matrix $\mathbf{R}^j$. Let $\mathbf{r}_p$ and $\mathbf{r}_q$ represent the coordinates of two candidates, $\mathbf{B}_p^j$ and $\mathbf{B}_q^j$ on the j^{th} image. For our experiments, we used the Euclidean distance between $\mathbf{r}_p$ and $\mathbf{r}_q$ to define the pairwise-similarity, $s(p,q)$, between $\mathbf{B}_p^j$ and $\mathbf{B}_q^j$ as following:

$$s(p,q) = e^{-\frac{\|\mathbf{r}_p - \mathbf{r}_q\|^2}{\varsigma^2}}.$$

If the matrices $s(p,q)$ are sparse, it aids the computational efficiency of our algorithms, both in terms of memory use and speed of implementation. Note also that the matrix is naturally sparse, due to the rapid decrease of the exponential function. Nevertheless, to make the implementation stable and memory efficient (by avoiding storing all the elements of these matrices), we found it useful to discretize the continuous similarity function, $s(p,q)$ to the binary similarity function, $s^*(p,q)$ by applying a threshold as following:

$$s^*(p,q) = \begin{cases} 0 & s(p,q) < e^{-4}, \\ 1 & s(p,q) \geq e^{-4}. \end{cases} \tag{13}$$

In all experiments, we set the threshold at e^{-4} to provide us with a similarity of one if the neighbor is at a 95% confidence level of belonging to the same density as the candidate assuming that the neighborhood is a Gaussian distribution with mean equal to the candidate and variance $\varsigma^2 = \frac{1}{\alpha}$. Each element of the matrix $\mathbf{R}$ is given by:

$$\mathbf{R}_{pq} = s^*(p,q).$$

In order to observe the effects of binarization of the $\mathbf{R}$ matrix, we also run experiments on *BatchSVM* without applying the threshold in Equation (13), that is, $\mathbf{R} = s(p,q)$. We refer to this version of *BatchSVM* as $BatchSVM_{cont}$ in our experiments.

5.1.2 PARAMETER TUNING

All our parameters in these experiments are tuned by 10-fold patient cross-validation on the training data (i.e., the training data is split into ten folds). During cross-validation, a range of parameters $(\theta, \sigma, \varsigma)$ were evaluated for the proposed methods. In $BatchPSVM - learned$, the parameter θ is learned automatically through equation (7).

5.2 Evaluation Method

As we explained earlier, $\mathbf{R}$ used in the proposed methods is produced from the spatial locations of the candidates. These spatial locations can be considered as extra information provided to our methods. In order to make fair comparisons between our methods and standard SVM and PSVM, we added the spatial locations (x,y,z coordinates of the candidates in 3D image) to the feature vectors of the candidates.

Receiver Operating Characteristic (ROC) plots are used to study the classification accuracy of these techniques on three CAD applications for detecting pulmonary embolism, colon cancer, and

	PE		Colon		Lung	
	train	test	train	test	train	test
SVM	16.87	0.00	7.22	0.00	1.58	0.00
BatchProb	19.26	4.71	105.14	18.63	46.42	30.02
BatchSVM	11.39	0.23	13.45	0.91	11.73	2.81
$BatchSVM_{cont}$	13.42	0.74	61.09	5.93	14.69	3.01
PSVM	0.22	0.00	0.13	0.00	0.05	0.00
$BatchPSVM - cv$	1.02	0.23	3.67	0.89	4.80	2.82
$BatchPSVM - learned$	1.39	0.33	4.08	0.89	5.44	2.79

Table 2: Training and testing times for each classifier are displayed in seconds.

lung cancer. In clinical practice, CAD systems are evaluated on the basis of a somewhat domain-specific metric: to maximize the fraction of *positives* that are correctly identified by the system while displaying at most a clinically acceptable number of false-marks per image. We report this domain-specific metric in an ROC plot, where the y-axis is a measure of sensitivity and the x-axis is the number of false-marks per patient (in our case, per image is also per patient). Sensitivity is the number of patients diagnosed as having the disease divided by the number of patients that has the disease. High sensitivity and low false-marks are desired. All classification algorithms are trained on a training data set and evaluated on a sequestered (held-out) test set.

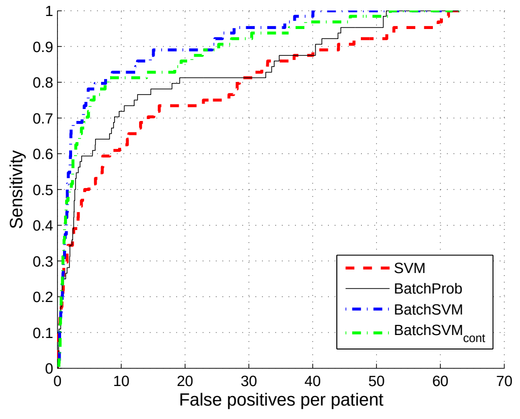
The area under the ROC curve is a commonly used criteria to evaluate the performance of an algorithm. Table 3 displays the area under the ROC curve of each classifier. It is important to note that the areas under the curve presented in Table 3 are calculated considering the entire false positive range which tends to underestimate the superiority of our proposed approach with respect to traditional approaches around the range of false positives acceptable for CAD applications.

PSVM, $BatchPSVM - cv$ and $BatchPSVM - learned$ algorithms require solving a set of linear equations while training, therefore we expect them to be faster than the other methods mentioned in this paper, namely SVM, *BatchProb* and *BatchSVM*. In order to compare the time performances of the methods, we measured the CPU time consumed by each classifier. Table 2, displays the training and testing times required by the algorithms. The training times obtained for the parametric methods (SVM, *BatchProb*, *BatchSVM*, PSVM and $BatchPSVM - cv$) are the total times divided by the number of cross-validations to find the best value of θ. On the other hand, the training time for $BatchPSVM - learned$ is the total time divided by the number of iterations that $BatchPSVM - learned$ required while converging to the optimum value of θ. We measured the training times this way to be able to make a fair comparison between the algorithms.

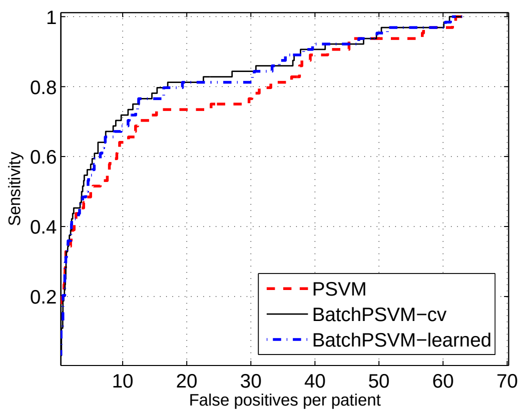
We ran our experiments on an Intel Pentium 4 CPU with 2.8 GHz clock frequency and 1GB RAM and the algorithms were implemented in Matlab.

5.3 Data Sources and Domain Description

When searching for descriptive features, researchers often deploy a large amount of experimental image features to describe the identified candidates, which consequently introduces irrelevant or redundant features. The initial large set of features is based on medically relevant characteristics:



a



b

Figure 3: Experimental results for PE data. ROC curves obtained by (a) SVM, *BatchProb*, *BatchSVM* and $BatchSVM_{cont}$ (b) PSVM, $BatchPSVM-cv$ and $BatchPSVM-learned$

the pixel intensity descriptive statistics, texture, contrast, 2D and 3D shape descriptors, and the location of the corresponding structure.

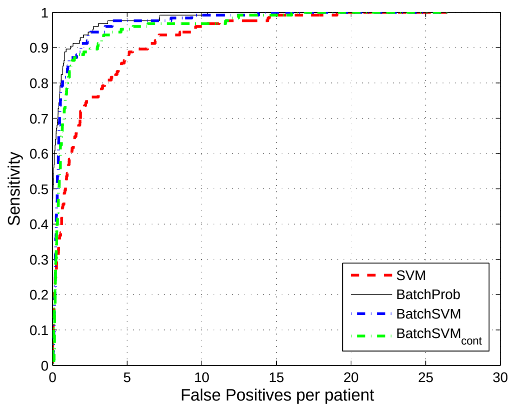
5.3.1 EXAMPLE: PULMONARY EMBOLISM

Pulmonary embolism (PE), a potentially life-threatening condition, is a result of underlying venous thromboembolic disease. An early and accurate diagnosis is the key to survival. Computed tomography angiography (CTA) has emerged as an accurate diagnostic tool for PE. However, there are hundreds of CT slices in each CTA study. Manual reading is laborious, time consuming and complicated by various PE look-alikes (false positives) including respiratory motion artifact, flow-related artifact, streak artifact, partial volume artifact, stair step artifact, lymph nodes, vascular bifurcation among many others (Masutani et al., 2002; Wittram et al., 2004). Several CAD systems are developed to assist radiologists in this process by helping them detect and characterize emboli in an accurate, efficient and reproducible way (Quist et al., 2004; Zhou et al., 2003).

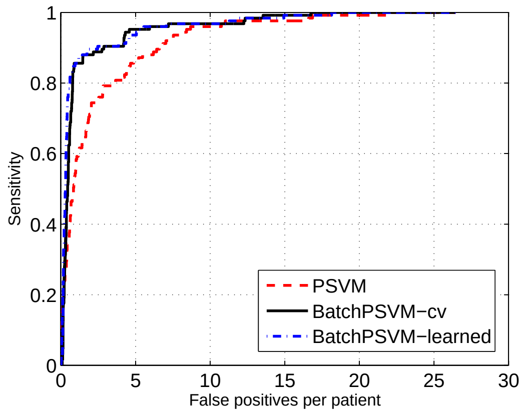
We have collected 72 cases with 242 PEs marked by expert chest radiologists at four different institutions (two North American sites and two European sites). For our experiments, they were randomly divided into two sets: a training and a testing set. The training set was used to train and validate the classifiers and consists of 48 cases with 173 PEs and a total of 3655 candidates. The testing set consists of 24 cases with 69 true PEs out of a total of 1857 candidates. This set was only used to evaluate the performance of the final system. A combined total of 70 features were extracted to describe the shape, texture, size and intensity profile of each candidate.

5.3.2 EXAMPLE: COLON CANCER DETECTION

Colorectal cancer is the third most common cancer in both men and women. It is estimated that in 2004, nearly $147,000$ cases of colon and rectal cancer will be diagnosed in the US, and more than $56,730$ people would die from colon cancer (Jemal et al., 2004). In over 90% of the cases colon



a



b

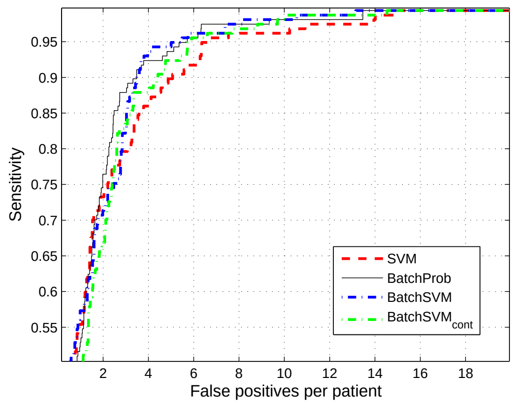
Figure 4: Experimental results for Colon Cancer data. ROC curves obtained by (a) SVM, *BatchProb*, *BatchSVM* and *BatchSVM*$_{cont}$ (b) PSVM, $BatchPSVM - cv$ and $BatchPSVM - learned$

cancer progressed rapidly is from local (polyp adenomas) to advanced stages (colorectal cancer), which has very poor survival rates. However, identifying (and removing) lesions (polyp) when still in a local stage of the disease, has very high survival rates, thus illustrating the critical need for early diagnosis. The sizes of a polyp in the colon can vary from 1mm all the way up to 10cm. Most polyps no matter how small they are, are represented by two candidates; one obtained from the prone view and the other one from the supine view. Moreover, for large polyps or so called masses a typical candidate generation algorithm generates several candidates across the polyp surface. Therefore most polyps in the training data are inherently represented by multiple candidates.

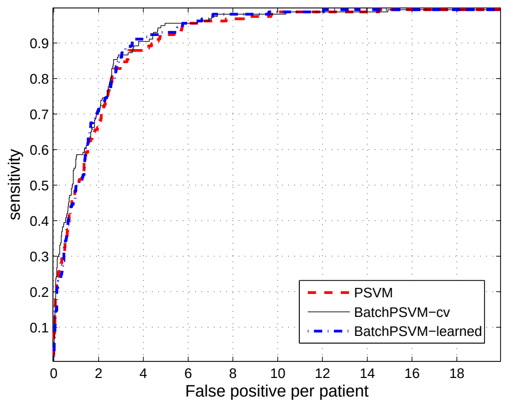
For the sake of clinical acceptability, it is sufficient to detect one of these candidates during classification. Unlike a standard classification algorithm where the emphasis is to accurately classify each and every candidate, here we seek to classify at least one of the candidates representing a polyp accurately. The database of high-resolution CT images used in this study were obtained from seven different sites across US, Europe and Asia. The 188 patients were randomly partitioned into a training and a test set. The training set consists of 65 cases containing 127 volumes. Fifty polyps were identified in this set out of a total of 6748 candidates. The testing set consists of 123 cases containing 237 volumes. There are 103 polyps in this set from a total of 12984 candidates. A total of 75 features were extracted to describe the shape, size, texture and intensity profile of each candidate.

5.3.3 EXAMPLE: LUNG CANCER

LungCAD is a computer aided detection system for detecting potentially cancerous pulmonary nodules from thin slice multi-detector computed tomography (CT) scans. The final output of LungCAD is provided by a classifier that classifies a set of candidates as positive or negative. This is a very hard classification problem: most patient lung CTs contain a few thousand structures (candidates),



a



b

Figure 5: Experimental results for Lung Cancer data. ROC curves obtained by (a) SVM, *BatchProb*, *BatchSVM* and *BatchSVM$_{cont}$* (b) PSVM, *BatchPSVM* − *cv* and *BatchPSVM* − *learned*

and only a few (≤ 5 on average) of which are potential nodules that should be identified as positive by LungCAD, all within the run-time requirements of completing the classification on-line during the time the physician completes their manual review.

The training set consists of 60 patients. The number of candidates labeled as nodules in the training set are 157 and the total number of candidates is 9987. The testing set consists of 26 patients. In this testing set, there are 79 candidates labeled as nodules out of 6159 generated candidates. A total of 15 features were used to describe the shape, size, texture and intensity profile of each candidate region.

5.4 Discussion and Results

In our experiments, we would like to find out whether the batch methods are better than the standard SVM, whether *BatchSVM* is better than *BatchProb* in terms of accuracy and speed, whether *BatchPSVM* is as accurate as *BatchSVM*, whether *BatchPSVM* is faster than *BatchSVM*, and whether the automated tuning works by comparing *BatchPSVM* − *learned* with *BatchPSVM* − *cv*.

5.4.1 SVM VERSUS BATCH METHODS

Figures 3, 4, and 5 show the ROC curves for pulmonary embolism, colon cancer, and lung cancer data respectively. In our medical applications high-sensitivity is critical as early detection of lung and colon cancer is believed to greatly improve the chances of successful treatment (Bogoni et al., 2005). Furthermore, high specificity is also critical, as a large number of false positives will vastly increase physician load and lead (ultimately) to loss of physician confidence.

In Figure 3a, corresponding to the comparison of the ROC curves on the PE data set, we observe that standard SVM can only achieve 53% sensitivity for six false positives. However, *BatchSVM* achieves 80% with a remarkable improvement of 27%. *BatchProb* also outperforms SVM with a

	PE	Colon	Lung
SVM	0.79	0.91	0.98
BatchProb	0.83	**0.98**	**0.99**
BatchSVM	**0.89**	0.96	**0.99**
$BatchSVM_{cont}$	0.88	0.95	**0.99**
PSVM	0.77	0.91	**0.99**
$BatchPSVM - cv$	0.82	0.96	**0.99**
$BatchPSVM - learned$	0.83	0.96	**0.99**

Table 3: Area under the ROC curves for each classifier. (Best in bold).

64% sensitivity. As seen from the figure, the two proposed methods are substantially more accurate than standard SVMs at any specificity level. When we compare the performances of PSVM and $BatchPSVM - learned$ in Figure 3b, we observe that the proposed batch method ($BatchPSVM - learned$) dominates over PSVM. However, they do not improve the accuracy of PSVM as much as *BatchProb* and *BatchSVM* do.

Colon cancer data is a relatively easier data set than pulmonary embolism since standard SVM can achieve 54.5% sensitivity at one false positive level as illustrated in Figure 4a. However, *BatchSVM* improved SVM's performance to 84% sensitivity for the same number of false positives. Note that *BatchProb* improved the sensitivity further, giving 89.6% for the same specifity. In one to ten false positives region which constitutes the region of interest in our applications, our proposed methods outperform standard SVM significantly. PSVM's performance is substantially worse than that of our proposed batch methods, $BatchPSVM - cv$ and $BatchPSVM - learned$.

Although SVM is very accurate for the lung cancer application, Figure 5a shows that *BatchProb* and *BatchSVM* could still improve SVM's performance further. *BatchProb* method is superior to the other methods at two and three false positives per image. Both *BatchProb* and *BatchSVM* outperform SVM in the 2-6 false positives per image region, which is the region of interest for commercial clinical lung CAD systems. All three of the methods are comparable at other specificity levels. On the other hand, we observe in Figure 5b that $BatchPSVM - cv$ and $BatchPSVM - learned$ methods did not achieve significant improvements, yet they are still comparable to PSVM.

When we consider the area under the ROC curve for each of the methods in Table 3, we observe that our proposed methods are better than SVM and PSVM in every case except one where PSVM performs the same as our proposed methods in the lung cancer data.

In order to understand if the proposed methods are statistically significantly better than standard SVM and PSVM, we applied Z-test to compare the areas under the ROC curves derived from the same set of patients as explained in Hanley and McNeil (1983). The corresponding P-values that indicate the level of confidence of the paired comparisons are presented in Table 4. The smaller the P-values, the higher the level of confidence. Looking at the results in Table 4, we can say that the improvements provided by the proposed methods are statistically significant for PE and Colon data sets.

	SVM vs *BatchProb*	SVM vs *BatchSVM*	PSVM vs *BatchPSVM* − *cv*	PSVM vs *BatchPSVM* − *learned*
PE	0.05	0.01	0.06	0.05
Colon	0.01	0.01	0.05	0.05
Lung	0.81	0.81	1	1

Table 4: P-values that indicate the confidence level of significance when the proposed algorithms are compared to SVM and PSVM in terms of the area under the ROC curves.

5.4.2 *BatchSVM* VERSUS *BatchProb*

As we discussed in Section 3, *BatchProb* requires calculating two matrix inversions, which can be time consuming for large sizes of batches, whereas *BatchSVM* avoids matrix inversions, therefore *BatchSVM* should be faster than *BatchProb* in both training and testing. As expected, our experiments show us that *BatchSVM* is faster than *BatchProb* in every case. Note that the difference in run times is significant for testing in PE and Colon data sets, where *BatchSVM* is twenty times faster than *BatchProb*. On the other hand, *BatchProb* produced a better ROC curve in Colon data set. Therefore, although *BatchProb* is slow compared to *BatchSVM*, it may be preferable with respect to accuracy depending on the application.

5.4.3 *BatchPSVM* − *learned* VERSUS *BatchPSVM* − *cv*

BatchPSVM − *cv* requires cross-validated tuning over the parameter θ to achieve the optimum accuracy. One problem with tuning is that we can only tune the parameter in a limited range. Therefore a good guess of the range of the parameter values is crucial. Another problem is that we can only tune the parameter over discrete values which may not include the best parameter value. One way to obtain a close approximation of the best parameter value is to divide the range of the parameter into small steps. However this method will increase the training time since cross-validation will be repeated as many times as the possible parameter values (in our experiments, we cross-validated θ over 21 values). On the other hand, automated tuning (*BatchPSVM* − *learned*) is quite feasible and avoids the problems stated above. In our experiments, *BatchPSVM* − *learned* was able to automatically converge to the optimum value of θ in only nine iterations on average while producing comparable ROC plots with cross-validated tuning (*BatchPSVM* − *cv*) as seen in Figures 3b, 4b and 5b. The optimum θ values obtained by cross-validated tuning and automated tuning are presented in Table 5. As observed in the table, the optimum θ values found by both algorithms are very similar.

When we compare the two methods with respect to training times, we observe that *BatchPSVM* − *cv* is faster than *BatchPSVM* − *learned* for a fixed value of θ. However, since we repeated cross-validation 21 times for *BatchPSVM* − *cv* and *BatchPSVM* − *learned* required nine iterations on average, *BatchPSVM* − *learned* is actually faster than *BatchPSVM* − *cv* in overall while training.

5.4.4 *BatchPSVM* VERSUS *BatchSVM*

BatchSVM results in a linear programming problem, whereas *BatchPSVM* requires solving a set of linear equations. Therefore, we expect *BatchPSVM* to be faster than *BatchSVM* while training. Experimental results on training times displayed in Table 2 confirm that it takes less time for

	PE	Colon	Lung
$BatchPSVM-cv$	1	0.6	0.1
$BatchPSVM-learned$	1.11	0.68	0.03

Table 5: θ values obtained by $BatchPSVM-cv$ and $BatchPSVM-learned$.

$BatchPSVM$ in the training phase. However, despite the training time performance, $BatchSVM$ produced better ROC curves than that of $BatchPSVM$ in PE and lung cancer data sets as illustrated in Figures 3 and 5 respectively. The area under the ROC curve produced by $BatchSVM$ is also better than $BatchPSVM$ for these two data sets. For Colon data set, in Figure 4b, we observe that $BatchPSVM-cv$ and $BatchSVM$ performed slightly better than $BatchSVM-learned$ where both achieve 86% sensitivity at one false positive level. Furthermore, the total area under the ROC curves produced by $BatchSVM$ and $BatchPSVM$ are the same (0.96). From the experimental results, we can draw the conclusion that $BatchPSVM$ is a fast alternative to $BatchSVM$ and can be preferable in some applications.

6. Conclusions

Three related algorithms have been proposed for classifying batches of correlated data samples. Although primarily motivated by real-life CAD applications, the problem occurs commonly in many situations; our algorithms are sufficiently general to be applied in other contexts. Experimental results indicate that the proposed method can substantially improve the diagnosis of (a) early stage cancer in the Lung & Colon, and (b) pulmonary embolisms (which may result in strokes). With the increasing adoption of these systems in routine clinical practice, these experimental results demonstrate the potential of our methods to impact a large cross-section of the population.

In this paper, we validated the improvement in diagnosis sensitivity that batch-wise classification provides to CAD problems. The algorithms presented here are general and can be applied to other domains. In other domains, one may apply other similarity metric appropriate for the application. For data with no clear batch information, clustering can be performed to create the batches. Extending the batch algorithms to other data and automatically learning the similarity metric are topics for future work.

Acknowledgments

This research is supported by Siemens Medical Solutions USA, Inc. and NSF CAREER IIS-0347532.

Appendix A.

In this section, we present the connection between Equations (3) and (12), and the derivation steps for automatically tuning the parameters in BatchPSVM.

A.1 Connection between Equations (3) and (12)

We will show the connection between the optimization problems that result by considering the constraints (3) and (12) by proving that the resulting objective functions are closely related to each other.

First, let us show that the inverse $\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1}$ that appears in Equation (12) can be approximated by the expression $\mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}$.

By replacing $\mathbf{A}=\sigma^2\mathbf{R}^{j^{-1}}$, in the identity:

$$(\mathbf{I}+\mathbf{A})^{-1}=\mathbf{I}-\mathbf{A}+\mathbf{A}^2-\mathbf{A}^3+\ldots+,\forall\mathbf{A}$$

we have:

$$\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1}=\mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}+\sigma^4\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}-\ldots$$

where the terms $\sigma^4\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}-\sigma^6\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}\mathbf{R}^{j^{-1}}+\ldots$ can be ignored if σ is sufficiently small.

Then, we have:

$$\begin{aligned}\left(\mathbf{I}+\mathbf{R}^{j^{-1}}\sigma^2\right)^{-1} &\cong \mathbf{I}-\sigma^2\mathbf{R}^{j^{-1}}\\ &= -\sigma^2\mathbf{R}^{j^{-1}}\left(\mathbf{I}+\theta\mathbf{R}^j\right)\end{aligned}\tag{14}$$

where $\theta=\frac{-1}{\sigma^2}$ provided that $\sigma\neq 0$.

Let us recall that the quadratic unconstrained objective function that results from the constraint (12) is:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}}\nu\frac{1}{2}\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\sigma^2\mathbf{R}^{j^{-1}}+\mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2)\tag{15}$$

and that the quadratic unconstrained objective function that results from the constraint (3) is given by:

$$\min_{(\mathbf{w},\gamma,\theta)\in\mathbb{R}^{n+1+1}}\nu\frac{1}{2}\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2+\frac{1}{2}(\mathbf{w}'\mathbf{w}+\gamma^2+\theta^2).\tag{16}$$

Next, we will show that the objective functions (15) and (16) are related to each other under certain assumptions. In order to ease the notation, let's now define

$$g(\mathbf{w},\gamma)=\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[\left(\sigma^2\mathbf{R}^{j^{-1}}+\mathbf{I}\right)^{-1}(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2.$$

Then if σ^2 is small we have from (14) that:

$$\begin{aligned}g(\mathbf{w},\gamma) &\approx \textstyle\sum_{j=1}^{k}\left\|\mathbf{e}-\mathbf{D}^j\left[-\sigma^2\mathbf{R}^{j^{-1}}\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2\\ &= \textstyle\sum_{j=1}^{k}\left\|\left(-\sigma^2\mathbf{R}^{j^{-1}}\right)\left(\left(-\frac{1}{\sigma^2}\mathbf{R}^j\right)\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right)\right\|^2\\ &\leq \textstyle\sum_{j=1}^{k}\left\|\sigma^2\mathbf{R}^{j^{-1}}\right\|^2\left\|\left(-\frac{1}{\sigma^2}\mathbf{R}^j\right)\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2\\ &\leq \Psi\textstyle\sum_{j=1}^{k}\left\|-\frac{1}{\sigma^2}\mathbf{R}^j\mathbf{e}-\mathbf{D}^j\left[\left(\theta\mathbf{R}^j+\mathbf{I}\right)(\mathbf{B}^j\mathbf{w}-\mathbf{e}\gamma)\right]\right\|^2=f(\mathbf{w},\gamma)\end{aligned}$$

where $\Psi = max_j\{\left\|\sigma^2\mathbf{R}^{j^{-1}}\right\|^2\}$, then, if σ is small we can conclude that $f(\mathbf{w},\gamma)$ is approximately a "good" upper bound for $g(\mathbf{w},\gamma)$. Furthermore if $h(\mathbf{w},\gamma,\theta) = \mathbf{w}'\mathbf{w} + \gamma^2 + \theta^2$,then $F(\mathbf{w},\gamma,\theta) = \nu\frac{1}{2}f(\mathbf{w},\gamma) + \frac{1}{2}h(\mathbf{w},\gamma,\theta)$ is also a "good" upper bound for $G(\mathbf{w},\gamma,\theta) = \nu\frac{1}{2}g(\mathbf{w},\gamma) + \frac{1}{2}h(\mathbf{w},\gamma,\theta)$.

So intuitively, since $G(\mathbf{w},\gamma,\theta)$ and $F(\mathbf{w},\gamma,\theta)$ are non-negative convex quadratic functions, minimizing $F(\mathbf{w},\gamma,\theta)$ should provide good approximate solutions to the problem of minimizing $G(\mathbf{w},\gamma,\theta)$.

Let us assume now that the number of neighboring points that affect the classification for every point is more or less constant. This is true when $\mathbf{R}$ is binary and calculated only for the k nearest neighbors or when the rows of $\mathbf{R}$ are normalized and approximately true when $\mathbf{R}$ is defined as in (13). This is equivalent to assuming that the summation of all the rows of $\mathbf{R}$ is constant, or equivalently, for every batch j:

$$-\frac{1}{\sigma^2}\mathbf{R}^j\mathbf{e} \approx k\mathbf{e}$$

where k is a constant. hence:

$$f(\mathbf{w},\gamma) = \sum_{j=1}^{k} \left\|k\mathbf{e} - \mathbf{D}^j\left[\left(\theta\mathbf{R}^j + \mathbf{I}\right)\left(\mathbf{B}^j\mathbf{w} - \mathbf{e}\gamma\right)\right]\right\|^2.$$

Then, solving the problem:

$$min_{\mathbf{w},\gamma,\theta}F(\mathbf{w},\gamma,\theta)$$

is equivalent to solving the proximal batch formulation presented in Equation (7) for some $\nu = \bar{\nu}$ (the constant k only rescales the problem in the variables w and γ).

A.2 Derivation for Automatically Tuning the Parameters in BatchPSVM

The optimization problem in (7) is a bi-convex problem and can be solved in an iterative fashion with two steps. At the first step, we find the optimum values of $\mathbf{w}$ and γ $(\mathbf{w}_{opt},\gamma_{opt})$ for a given θ. At the second step, for $\mathbf{w}_{opt}$ and γ_{opt} held fixed, we find the optimum value of θ (θ_{opt}), which will be used at the first leg of the next iteration.

A.2.1 OPTIMIZING FOR $\mathbf{w}$ AND γ WITH θ FIXED

Given θ is held fixed, we can optimize (7) with respect to $\mathbf{w}$ and γ, and rewrite the equation as follows:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \frac{1}{2}\mu\sum_{j=1}^{k}\mathbf{x}'\mathbf{P}^j\mathbf{x} + \mathbf{q}^j\mathbf{x} + K_1,$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{w} \\ \gamma \end{bmatrix},$$

$$\mathbf{P}^j = \begin{bmatrix} \frac{1}{\mu}\mathbf{I}+\theta^2\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j+\theta\mathbf{B}^{j'}(\mathbf{R}^{j'}+\mathbf{R}^j)\mathbf{B}^j+\mathbf{B}^{j'}\mathbf{B}^j & -\theta^2\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e}-\theta\mathbf{B}^{j'}(\mathbf{R}^{j'}+\mathbf{R}^j)\mathbf{e}-\mathbf{B}^{j'}\mathbf{e} \\ -\theta^2\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j-\theta\mathbf{e}'(\mathbf{R}^{j'}+\mathbf{R}^j)\mathbf{B}^j-\mathbf{e}'\mathbf{B}^j & \theta^2\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e}+\theta\mathbf{e}'(\mathbf{R}^{j'}+\mathbf{R}^j)\mathbf{e}+\mathbf{e}'\mathbf{e}+\frac{1}{\mu} \end{bmatrix},$$

$$\mathbf{q}^j = 2\begin{bmatrix} -\theta\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j-\mathbf{e}'\mathbf{D}^j\mathbf{B}^j \\ -\theta\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}+\mathbf{e}'\mathbf{D}^j\mathbf{e} \end{bmatrix}$$

and K_1 is a constant term with respect to $\mathbf{w}$ and γ.

After taking the derivative of the objective function with respect to $\mathbf{x}$ and equating to zero, we can obtain the solution as following:

$$x = -(\sum_{j=1}^{k} \mathbf{P}^j)^{-1}(\sum_{j=1}^{k} \mathbf{q}^j)$$

A.2.2 OPTIMIZING FOR θ WITH $\mathbf{w}$ AND γ FIXED

When $\mathbf{w}$ and γ have fixed values, we can rewrite Equation (7) considering θ is the only variable as following:

$$\min_{(\mathbf{w},\gamma)\in\mathbb{R}^{n+1}} \frac{1}{2}\mu \sum_{j=1}^{k} \theta^2 N^j + \theta M^j + K_2$$

where K_2 is a constant with respect to θ and

$$N^j = \quad \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e}\gamma - \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \gamma^2\mathbf{e}'\mathbf{R}^{j'}\mathbf{R}^j\mathbf{e} + \frac{1}{\mu}$$

and

$$\begin{aligned} M^j = \quad & -\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}\gamma - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{D}^{j'}\mathbf{e} + \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^{j'}\mathbf{e}\gamma \\ & +\gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{D}^{j'}\mathbf{e} - \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{B}^j\mathbf{w} + \gamma\mathbf{e}'\mathbf{R}^{j'}\mathbf{e}\gamma + \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^j\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}\mathbf{R}^j\mathbf{e}\gamma \\ & -\gamma\mathbf{e}'\mathbf{R}^j\mathbf{B}^j\mathbf{w} + \gamma\mathbf{e}'\mathbf{R}^j\mathbf{e} \end{aligned} .$$

M^j can be further simplified as:

$$\begin{aligned} M^j = \quad & \mathbf{w}'\mathbf{B}^{j'}(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{B}^j\mathbf{w} - \mathbf{w}'\mathbf{B}^{j'}(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{e}\gamma - \gamma\mathbf{e}'(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{B}^j\mathbf{w} \\ & +\gamma\mathbf{e}'(\mathbf{R}^{j'} + \mathbf{R}^j)\mathbf{e}\gamma - 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{B}^j\mathbf{w} + 2\mathbf{e}'\mathbf{D}^j\mathbf{R}^j\mathbf{e}\gamma \end{aligned}$$

using the property that $A' = A$, if A is a scalar.

References

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

L. Bogoni, P. Cathier, M. Dundar, A. Jerebko, S. Lakare, J. Liang, S. Periaswamy, M. Baker, and M. Macari. CAD for colonography: A tool to address a growing need. *British Journal of Radiology*, 78:57–62, 2005.

L. Bottou and V.N. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992. URL `citeseer.ist.psu.edu/bottou92local.html`.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proc. 15th International Conf. on Machine Learning*, pages 82–90, San Francisco, CA, 1998. Morgan Kaufmann.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.

G. Fung, M. Dundar, J. Bi, and B. Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 40, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015409.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

J.A. Hanley and B.J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983. URL `http://radiology.rsnajnls.org/cgi/content/abstract/148/3/839`.

D. Jemal, R. Tiwari, T. Murray, A. Ghafoor, A. Samuels, E. Ward, E. Feuer, and M. Thun. Cancer statistics. *CA Cancer J. Clin.*, 54:8–29, 2004.

B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In *NIPS 17*, pages 721–728. MIT Press, 2005.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

Y. Masutani, H. MacMahon, and K. Doi. Computerized detection of pulmonary embolism in spiral CT angiography based on volumetric image analysis. *IEEE Transactions on Medical Imaging*, 21:1517–1523, 2002.

M. Quist, H. Bouma, C. Van Kuijk, O. Van Delden, and F. Gerritsen. Computer aided detection of pulmonary embolism on multi-detector CT. In *Proceedings of the 90th Meeting of the Radiological Society of North America (RSNA)*, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

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

V. Vural, G. Fung, J. Dy, and Rao B. Fast semi-supervised svm classifiers using a-priori metric information. *Optimization Methods and Software (OMS)*, 23(4):521–532, 2008.

C. Wittram, M. Maher, A. Yoo, M. Kalra, O. Jo-Anne, M. Shepard, and T. McLoud. CT angiography of pulmonary embolism: Diagnostic criteria and causes of misdiagnosis. *RadioGraphics*, 24: 1219–1238, 2004.

M. Wu and B. Schlkopf. Transductive classification via local learning regularization. In *11th International Conference on Artificial Intelligence and Statistics*, pages 628–635, Brookline, MA, USA, 03 2007. Microtome. URL `http://jmlr.csail.mit.edu/proceedings/papers/v2/`.

C. Zhou, L. M. Hadjiiski, B. Sahiner, H.-P. Chan, S. Patel, P. Cascade, E. A. Kazerooni, and J. Wei. Computerized detection of pulmonary embolism in 3D computed tomographic (CT) images: vessel tracking and segmentation techniques. In *Medical Imaging 2003: Image Processing. Edited by Sonka, Milan; Fitzpatrick, J. Michael. Proceedings of the SPIE, Volume 5032, pp. 1613-1620 (2003).*, pages 1613–1620, May 2003. doi: 10.1117/12.481369.

D. Zhou, O. Bousquet, T. Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*. MIT Press, 2003.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. 20th Int. Conf. on Machine Learning*, pages 912–919, 2003.