Efficient Max-Margin Metric Learning

Caiming Xiong
cxiong@buffalo.edu

David M. Johnson
davidjoh@buffalo.edu

Jason J. Corso
jcorso@buffalo.edu

ABSTRACT

Efficient learning of an appropriate distance metric is an increasingly important problem in machine learning. However, current methods are limited by scalability issues or are unsuited to use with general similarity/dissimilarity constraints. In this paper, we propose an efficient metric learning method based on the max-margin framework with pairwise constraints that has a strong generalization guarantee. First, we reformulate the max-margin metric learning problem as a structured support vector machine which we can optimize in linear time via a cutting-plane method. Second, we propose a kernelized extention and an approximation method based on matching pursuit that allows linear-time training even in the kernel case.We find our method to be comparable to or better than state of the art metric learning techniques at a number of machine learning and computer vision classification tasks.

KEYWORDS

metric learning, structured svm, cutting-plane, matching pursuit

1. INTRODUCTION

Distance metric learning is a powerful technique for a variety of machine learning problems, and has seen use in various machine learning applications such as object classification, text document retrieval and face verification . Current methods are limited, however, by scalability constraints that make them impractical for use with large datasets. In this paper, we present a new metric learning method capable of training in linear time and testing in constant time, and thus applicable to even large-scale learning problems.

The motivation behind distance metric learning lies ultimately in the limitations of the Euclidean distance. Although the Euclidian distance presents a very simple and convenient metric for comparison, in many cases it is not an accurate representation of the underlying shape of the data [Frome et al., 2007]. Unfortunately, the formulation of a good problem-specific metric is often far from obvious, especially for high dimensional data or multiple feature types, as is common in many applications.

In this paper, we propose an efficient maximum margin metric (and its kernelized form) for automatically learning a Mahalanobis distance metric from training data, which takes the form of a number of pairwise constraints with 'similar' or 'dissimilar' labels. Max-margin methods have demonstrated impressive successes on a broad range of tasks and are appealing because of their strong generalization guarantees. In our case, we adopt a max-margin approach to separating similar and dissimilar point-pairs, using the Frobenius norm of the Mahalanobis metric matrix as a regularizer. Under this formulation, solving the metric learning problem is the same as solving a support vector machine for binary classification, with an additional positive-semidefinite constraint for the metric matrix.

However, the additional constraint makes optimizing this metric learning formulation directly a semidefinite programming problem, and thus impractical for large-scale datasets. We overcome this obstacle by reformulating our max-margin metric learning problem as a structured support vector machine and solving via a cutting-plane algorithm [Kelley Jr, 1960]. This iterative approach allows us to guarantee convergence to a PSD solution in $\mathbf{O}(n)$ time, where n is number of pairwise constraints.

The Mahalanobis distance metric we learn in the original feature space can be seen as a Euclidean distance metric in the linear transform of the original space, but in some cases a linear transform is not appropriate for the problem. We have extended our algorithm to a kernelized formulation, but doing so increases the time complexity to $\mathbf{O}(n^2)$. Therefore, we propose an example-based matching pursuit method for approximate kernel-based metric learning, which yields a $\mathbf{O}(n)$ learning algorithm even in the kernel space.

This matching pursuit method treats the data distribution as a low-rank manifold in kernel space that is spanned by a constant (and small) number of elements. We cannot obtain these spanning elements explicitly because in kernel-based methods all features are mapped onto an implicit high-dimension space. Thus, we use the matching pursuit method to find a basis from the input data set that allows us to approximately reconstruct the implicit elements. Using this basis, we can transfer the data points from the implicit high dimensional kernel space to the explicit low-dimensional manifold space, then learn the metric efficiently in this low-dimensional space. Matching pursuit takes $\mathbf{O}(n)$ time, so the entire training process retains its linear complexity.

2. RELATED WORK

A number of previous works have addressed the problem of learning distance metrics from data. One such method [Xing et al., 2003] minimizes the distance between data points in the same class subject to the constraint that data points from different classes are separated, but requires solving a semidefinite programming problem, and thus is very slow. Relevant Component Analysis (RCA) [Bar-Hillel et al., 2003] learns the linear transformation under equivalence constraints and Kernel RCA [Tsang et al., 2005]) is kernel-based RCA, but both are limited by their use of only similarity constraints. Discriminant Component Analysis (DCA) [Hoi et al., 2006] extends RCA by exploring negative constraints, but in practice may yield worse results than using positive constraints alone. ITML [Davis et al., 2007] minimizes the LogDet divergence under linear constraints. More recently, researchers have begun developing fast algorithms which can work in an online manner, such as POLA [Shalev-Shwartz et al., 2004], MLCL [Globerson and Roweis, 2006] and LEGO [Jain et al., 2008].

Beside the above methods, there are several preexisting algorithms similar to our own. [Schultz and Joachims, 2004] defines relative distance constraints between sets of three points, and use the Frobenius inner product as a regularizer. However, their method assumes the metric matrix is diagonal, and thus is less general than methods using the full Mahalanobis matrix. LMNN [Shen et al., 2010, Weinberger and Saul, 2009] adopts similar constraints to [Schultz and Joachims, 2004], but seeks to minimize the sum of the distance of the pairs of points such that neighbors are in the same class. Nguyen et al. [Nguyen and Guo, 2008] formulate metric learning as a quadratic semi-definite programming problem with local neighborhood constraints and linear time complexity in the original feature space, but the kernelized version of their method still suffers from quadratic time complexity.

3. PROBLEM STATEMENT

Given a set of points $\{x_1, x_2, \cdots, x_N\}$, each $x_i \in \mathbb{R}^m$ is a vector of m features. Denote the set of pairwise equivalence and inequivalence constraints: $\mathcal{S}$ = $\{$ (x_i,x_j) | x_i and x_j are similar$\}$, $\mathcal{D}$ = $\{$ (x_i,x_j) | x_i and x_j are dissimilar$\}$. The total number of constraints ($\#(\mathcal{S} \cup \mathcal{D})$) is n, and we denote y_{ij} as the relation between the constrained points x_i and x_j. If (x_i,x_j) $\in \mathcal{S}$, then $y_{ij} = 1$. If (x_i,x_j) $\in \mathcal{D}$, then $y_{ij} = -1$. Our goal is to seek the PSD matrix A, which specifies a Mahalanobis distance metric. The form of the distance defined by matrix A between points x_i and x_j is:

$$d_A(x_i, x_j) = (x_i - x_j)^T A (x_i - x_j) = \langle A, (x_i - x_j)(x_i - x_j)^T \rangle_F \ , \tag{1}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, such that $\langle P, Q \rangle_F = \sum_{i,j} P_{ij} Q_{ij}$. Our objective is to find the PSD matrix A and the corresponding distance threshold b such that for any pair $(x_i,x_j) \in \mathcal{S}$ the distance between them is smaller than b and for any pair $(x_i,x_j) \in \mathcal{D}$ the distance between them is greater than b, after accounting for slack. If we consider $\mathcal{S}$ and $\mathcal{D}$ as two classes, this problem is equivalent to the binary classification problem. Here we adopt a max-margin approach as our objective for metric learning because of its ability to minimize generalization error. Therefore we formulate our metric learning problem as:

$$\begin{aligned} &\min_{A,b,\xi} \frac{1}{2}\|A\|_F^2 + \frac{C}{n} \sum_{(x_i,x_j)\in(\mathcal{S}\cup\mathcal{D})} \xi_{ij} \\ &s.t. \quad \forall (x_i,x_j) \in (\mathcal{S}\cup\mathcal{D}) : y_{ij}[\langle A, -(x_i - x_j)(x_i - x_j)^T \rangle_F + b] \geq 1 - \xi_{ij} \\ &\xi_{ij} \geq 0, A \geq 0 \ . \end{aligned} \tag{2}$$

where ξ_{ij} is a slack varible. We can obtain the metric matrix A and threshold b by solving the above formulation. Obviously, because of the non-negative constraint on A, this is a semidefinite programming problem. To efficiently solve the above formulation for large-scale datasets, we transfer our metric learning formulation to a structured SVM [Franc and Sonnenburg, 2008, Joachims et al., 2009, Shalev-Shwartz et al., 2007] and use a cutting-plane algorithm to find an approximate solution within the PSD cone in linear time.

4. MAX-MARGIN METRIC LEARNING VIA CUTTING PLANE ALGORITHM

In this section, we introduce how to transfer the metric learning problem to a structured SVM. We then explain how to use the cutting-plane algorithm to optimize the structured SVM to get the metric matrix A, and how to enforce the PSD constraint during optimization. Finally, we explain how to kernelize this problem formulation.

4.1 Max-Margin Metric Learning Revisited

To solve the metric learning problem efficiently, we begin by rewriting equation 2 into an unbiased form. First define a set $\mathcal{U}$, $u_k \in \mathcal{U}$ such that $u_k = (1; \vec{(}-(x_{k_1} - x_{k_2})(x_{k_1} - x_{k_2})^T)$, where $\vec{(}\cdot)$ represents vectorization of the input matrix, and $(x_{k_1}, x_{k_2}) \in (\mathcal{S}\cup\mathcal{D})$, so $\#(\mathcal{U}) = \#(\mathcal{S}\cup\mathcal{D}) = n$. The values k_1 and k_2 are the indices of points in the input dataset. $\mathcal{U}$ then serves as the new set of input data points for the SVM, and we define y_k as the label for u_k. If $(x_{k_1}, x_{k_2}) \in \mathcal{S}$, $y_k = 1$, otherwise -1. We define $w = [b; \vec{(}A)]$. The biased formulation in equation 2 can then be rewritten as follows:

$$\begin{aligned} &\min_{w,\xi_i} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{k=1}^{n} \xi_k \\ &s.t. \quad \forall k \in \{1, \cdots, n\} : y_k(w^T u_k) \geq 1 - \xi_k, \xi_k \geq 0, A \geq 0 \ . \end{aligned} \tag{3}$$

The structured SVM is a generalization of multi-class SVM that replaces simple label outputs with structured outputs (for example, parse trees or ordered point sets). In our case, however, we are not utilizing the potential for more complex outputs. We are instead simply using the structured SVM formulation to yield a one-slack binary SVM classifier that can be learned in an iterative manner in linear time. In accordance with [Joachims, 2006], we formulate the structured SVM (with additional PSD constraint) below:

$$\begin{aligned} &\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\xi \\ &s.t. \quad c \in \{0,1\}^n : \frac{1}{n} w^T \sum_{k=1}^{n} c_k y_k x_k \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi, \xi \geq 0, A \geq 0 \ . \end{aligned} \tag{4}$$

For the above formulation, there is only one slack ξ, but there are 2^n constraints rather than n. Each of these new compound constraints corresponds to the combination of a subset of the constraints in the unbiased form. The solution to the unbiased formulation in equation 3 is equivalent to the solution of the structured SVM with PSD constraint in equation 4. This can be easily proven via the same process as used in [Joachims, 2006].

4.2 Cutting Plane Algorithm

At first glance, the structured SVM with PSD constraint is more difficult to solve than the unbiased formulation, with the exponential number of constraints (2^n) requiring exponential space and time. However, the structured SVM has desirable sparseness properties, which are independent of the size of the training set. We do not need to use all of the exponential number of constraints—a small subset is sufficient. Here we use the cutting-plane algorithm to iteratively refine the objective function by means of linear equalities. We begin with no constraints, and simultaneously add constraints and optimize our solution in an iterative manner. In each iteration we first find unsatisfied pair constraints and combine them into one new compound constraint, then optimize the current metric matrix with respect to the current compound constraints. In this way, the cutting-plane algorithm can find an ϵ-approximate solution in a constant number of iterations [Joachims et al., 2009]. This leads to the linear time algorithm in the original space shown in Algorithm 1.

Algorithm 1 Max Margin Metric learning via Cutting Plane algorithm

Input: training pairs $(x_i, x_j) \in (\mathcal{S} \cup \mathcal{D})$ C, ϵ
Initialization:
$\mathcal{U} : \forall u_k \in \mathcal{U}, (u_k, y_k) : u_k = [1; \vec{(} - (x_{k_1} - x_{k_2})(x_{k_1} - x_{k_2})^T)]$;
if $(x_{k_1}, x_{k_2}) \in \mathcal{S}$ **then**
  $y_k = 1$
else
  $y_k = -1$
end if
Set $l = 0$, $\mathcal{C} = \emptyset$, $\mathbf{B} = \emptyset$, $w =$ [arbitrary]
repeat
  for i=1,$\cdots$,n **do**

$$c_i = \begin{cases} 1 & y_i(w^T u_i) \leq 1, \\ 0 & \text{otherwise} \end{cases}$$

  end for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{c}\}$
  $(w, \xi) = \text{argmin}_{w,\xi} \frac{1}{2}\|w\|^2 + C\xi$
    s.t $\forall \mathbf{c} \in \mathcal{C} : \frac{1}{n} w^T \sum_{k=1}^{n} \mathbf{c}_k y_k u_k \geq \frac{1}{n} \sum_{i=1}^{n} \mathbf{c}_i - \xi, \xi \geq 0, A \geq 0, w = [b; \vec{(}A)]$
  $l = l + 1$
  $\Psi_l = \frac{1}{n} w^T \sum_{i=1}^{n} \mathbf{c}_i^l y_i u_i = \frac{1}{n} w^T \sum_{i=1}^{n} \mathbf{c}_i^l y_i ([1; \vec{(} - (x_{i_1} - x_{i_2})(x_{i_1} - x_{i_2})^T)])$
until $\Psi_l \leq \xi + \epsilon$
Output A according to $w = [b; \vec{(}A)]$. .

4.3 Implementation

At each iteration of Algorithm 1, there is an optimization problem to be solved. This can be accomplished easily via a primal subgradient method. First, reformulate the optimization problem into the primal form:

$$\begin{aligned} (w) = \underset{w}{\text{argmin}}\, f(w) = \frac{1}{2}\|w\|^2 + C \max_{\mathbf{c} \in \mathcal{C}} (\frac{1}{n} \sum_{i=1}^{n} \mathbf{c}_i - \frac{1}{n} w^T \sum_{k=1}^{n} \mathbf{c}_k y_k x_k, 0) \\ s.t. A \geq 0, w = [b; \vec{(}A)] \ . \end{aligned} \tag{5}$$

In this form, we can obtain the gradient of $f(w)$ analytically via:

$$\frac{\partial f}{\partial w} = w - C \cdot (\frac{1}{n} w^T \sum_{k=1}^{n} \mathbf{c}_k^* y_k x_k) \ . \tag{6}$$

where $\mathbf{c}^* \in \mathcal{C}$ is the single compound constraint that achieves the current largest margin violation.

After solving this gradient step, we obtain a new w. Since $w = [b; \vec{(}A)]$, we also obtain a new metric matrix A. Because the metric matrix must be a PSD matrix, we adapt spectral decomposition to project the output metric matrix A into the PSD cone: first eigendecompose the metric matrix, set the negative eigenvalues v to zero, then reconstruct the metric matrix using the remaining positive eigenvalues and corresponding eigenvectors λ via

$$A = \sum_{i=1}^{m} v_i \lambda_i^T \lambda_i \ . \tag{7}$$

This yields a new metric matrix A that is guaranteed to be PSD. We then vectorize this matrix and place it in w, replacing the old A. We then optimize w again and repeat this process until convergence, taking $\mathbf{O}(n)$ at each iteration and constant iterations.

4.4 Complexity

At each iteration in Algorithm 1, we use the new w to find the pairs in the training set that are not satisfied by the current metric matrix and combine them into one new compound constraint for the structured SVM for metric learning. This requires computing dot products for each of the n constraints, and thus obviously consumes $\mathbf{O}(n)$ time ($\mathbf{O}(nl)$ time over all iterations). We also have to find the compound constraint $\mathbf{c}$ with the current largest margin violation at each iteration. This requires $\mathbf{O}(nl)$ time at each iteration (where l is the current number of iterations/compound constraints), and thus $\mathbf{O}(nl^2)$ time over all iterations.

After each iteration of our algorithm, we check the error loss of the unsatisfied pairs on the current metric matrix. If it is greater than $\xi + \epsilon$, then we go back to the loop, otherwise we stop and extract the output Mahalanobis metric matrix A from w. It has been proven in [Joachims, 2006] that this process requires a constant number of iterations that doesn't depend on the size of the training set for a given error accuracy ϵ. l is thus a constant, so the overall time complexity for this algorithm is $\mathbf{O}(n)$.

4.5 Kernelized Metric Learning

The metric matrix resulting from the method described above can be seen as a linear transform of the original feature space. However, in some cases the nonlinear nature of the data demands a nonlinear transformation. Thus, it is necessary that the metric learning algorithm be applicable to the kernelized case.

In applying a kernel to a given dataset, we implicitly map each point x_{k_1}, x_{k_2} in each constraint pair to a high dimensional feature space via $\phi(x_{k_1})$,$\phi(x_{k_2})$, where $\phi(x.)$ is the implicit mapping function. We then build the training set sample (u_k, y_k) shown in the initialization part of in Algorithm 1. Unfortunately, because the mapping function ϕ is not explicit when using a nonlinear kernel, we cannot obtain the explicit input $u_k = [1; \vec{(} - (\phi(x_{k_1}) - \phi(x_{k_2}))(\phi(x_{k_1}) - \phi(x_{k_2}))^T)]$. Moreover, because the kernel function only allows us to compute dot products in the kernel space (as opposed to other operations such as vector subtraction), we cannot even compute the normal u elements implicitly in kernel space. One potential solution to this problem is to instead apply the kernel function to the transformed point pairs via $\phi(u)$, directly learning a kernelized distance function, rather than a distance metric within the kernel space. This is feasible, but has inherent scalability problems.

In order to learn a kernelized distance function using the cutting plane optimization framework described previously, there are multiple points during in each iteration where we must evaluate each pair constraint with respect to the current SVM model: first when we determine which constraints are not satisfied in order to construct a new compound constraint, and second when we must find the current compound constraint with the largest margin violation. Evaluating a single pair-constraint in the kernelized case requires computing $w \cdot \phi(u_k) = \sum_{i=1}^{n} \alpha_i y_i K(u_k, u_i)$, where $K(\cdot, \cdot)$ indicates the kernel function. Because of the required kernel comparison to each of the n pair constraints, this obviously requires $\mathbf{O}(n)$ time. Evaluating the SVM for all n pairs thus costs $\mathbf{O}(n^2)$ time, thus requiring that the overall optimization algorithm also be $\mathbf{O}(n^2)$.

Directly learning a kernelized max-margin metric might thus be practical for small datasets. However, as the complexity of data increases, the number of pair constraints required to provide sufficient information to learn a good distance metric also increases. Challenging datasets requiring significant numbers of pair constraints

will thus quickly become impractical for kernelized max-margin metric learning, given the quadratic time complexity. Hence, we seek an approximation that will allow for more efficient and scalable computation.

4.6 Approximated Method based on Matching Pursuit Algorithm

Because of the high computational complexity of the kernelized structured SVM metric learning method, we want to find a new algorithm to approximate the problem. We know the key problem is the representation $\phi(x_i)$ of data point x_i in kernel space. $\phi(x)$ is implicit and generally high dimensional. If $\phi(x_i)$ can be represented in an explicit and low-dimensional way, then we can use this representation not only to learn a metric within the kernel space (rather than a kernelized distance in the original space) but we can do so in linear time using Algorithm 1.

We also know that, in the kernel space, data points are distributed as a manifold. Thus, we want to construct a low-dimensional subspace $\phi(x_i) = span(\phi(x_1^{'}), \phi(x_2^{'}), \cdots, \phi(x_t^{'})) = \sum_{i=1}^{t} \lambda_i \phi(x_i^{'}) : \lambda \in \mathcal{R}^l$, each $\phi(x_k^{'}) \in \mathcal{B}$ is a basis vector and each two basis vectors are orthogonal, thus we can provide a low dimension representation of $\phi(x_i)$ such that $\phi^{'}(x_i) = (K(x_i, x_1^{'}), K(x_i, x_2^{'}), \cdots, K(x_i, x_t^{'}))$ and $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \approx \phi^{'}(x_i) \cdot \phi^{'}(x_j)$. In essence, then, we wish to obtain a basis set for the kernel space, and use it to transform the original data into a new representation that encodes the position of points within the kernel space by describing them in terms of their kernel relationship to the points in the identified basis. Once we have this new representation, we can then use Algorithm 1 to efficiently learn a metric within this new space.

However, because the positions of vectors within the kernel space are implicit, we cannot find our orthogonal basis vectors directly. We instead design a example-based matching pursuit algorithm that selects instances from the input dataset that are approximately orthogonal in the kernel space. We begin with no vectors in our basis set ($\mathcal{B} = \emptyset$) and randomly choose one point $\phi(x_k)$ as the first basis vector and add it into basis set $\mathcal{B}$. Then, in order to find a new basis vector, we solve the following formulation:

$$\phi(x_i) = \underset{\phi(x_i): x_i \in \{x_1, x_2, \cdots, x_N\}}{\operatorname{argmax}} Residual(\phi(x_i)) = \underset{\phi(x_i)}{\operatorname{argmax}} (\min_{\beta} \|\phi(x_i) - \sum_{j=1}^{t} \beta_j \phi(x_j^{'})\|^2) \ . \tag{8}$$

The expression $\min_\beta \|\phi(x_i) - \sum_{j=1}^{t} \beta_j \phi(x_j^{'})\|^2$ represents the loss from the best possible reconstruction (in the kernel space) of point x_i using the current set of basis vectors. Because the previously selected basis vectors are always approximately orthogonal to each other, the values of each β_j that provide the best reconstruction in the kernel space can be approximately calculated via $\beta_j = \frac{K(x_j, x_i^{'})}{\sqrt{K(x_j, x_j) \cdot K(x_i^{'} x_i^{'})}}$. Using this approximation, $Residual(\phi(x_i))$ can be easily obtained. At each iteration of our algorithm, we compute the residual for each point in the dataset, then choose the point with the largest residual to add to our basis set $\mathcal{B}$, until we reach a maximal number of basis vectors or the maximal residual value is less than a predefined threshold. Obtaining each new basis vector requires $\mathbf{O}(t)$ computations for each point in the input dataset, so the process as a whole costs $\mathbf{O}(Nt^2)$ time, where N is the number of points in the dataset. The number of iterations can be set to a constant, reducing this to $\mathbf{O}(N)$ time.

Once we have our basis set, computing the new representation for the data takes $\mathbf{O}(Nt)$ time (again reducing to $\mathbf{O}(N)$). Once this is complete, our linear max-margin metric learning method can be applied to the new data representation, again taking $\mathbf{O}(n)$ time, for a final time complexity of $\mathbf{O}(max(N, n))$ even in the kernelized case.

5. EXPERIMENTS

In this section, we present our experiments using three datasets: the UCI [Frank and Asuncion, 2010] and USPS [Roweis, 2010] machine learning datasets and the Caltech-101 [Fei-Fei et al., 2007] image dataset. We obtain the metric matrix in the original space for the UCI and USPS datasets. Because image datasets tends to have non-linear decision boundaries, we learn the kernel based metric for the image data. To evaluate the performance of our metric learning algorithm, we use the k-Nearest-Neighobor classification algorithm (kNN). We assume that higher accuracy of kNN indicates better performance of the metric used.

Table 1: Classification error rate of kNN classifier on the seven UCI data sets, as well as the larger-scale USPS dataset, using four different metrics.

	Euclidean	RCA	DCA	ITML	LMNN	Our Method
Balance	18.18	15.89	16.58	**9.42**	18.53	11.87
Breast	38.58	4.19	4.63	3.69	**3.07**	3.5
Diabetes	30.56	29.42	**27.86**	28.42	28.52	28.52
Segmentation	24.86	47.14	16.82	11.0	11.62	**10.9**
Sonar	25.99	34.29	25.16	19.23	15.97	**14.52**
Wine	3.93	2.99	3.74	2.3	2.59	**1.97**
Breast(Scaled)	**3.28**	3.81	4.0	3.56	**3.28**	3.37
USPS	6.71	-	-	11.15	8.50	**3.85**

5.1 Linear Metric Learning on UCI Machine Learning and USPS Datasets

We tested our method against other metric learning techniques using a kNN-classification task over seven data sets from the UCI repository, as well as the larger USPS dataset: (1) breast-cancer, with 2 classes, 683 instances and 10 features; (2) segmentation, with 7 classes, 210 instances and 19 features; (3) wine, with 3 classes, 178 instances and 13 features; (4) balance, with 3 classes, 625 instances and 4 features; (5) Diabetes, with 2 classes, 768 instances and 8 features; (6) sonar, with 2 classes, 208 instances and 60 features; (7) breast-cancer_scaled, with 2 classes, 683 instances and 10 features; (8) USPS, with 10 classes, 11000 instances and 256 features.

We compare our method to the Euclidean distance metric (as a baseline), as well as four state of the art algorithms for metric learning: (1) RCA, (2) DCA, (3) Information-Theoretic Metric Learning (ITML) [Davis et al., 2007] and (4) Distance metric learning for large margin nearest neighbor classification (LMNN). All algorithms are from authors' websites.

In each dataset, we randomly designate half of the instances as training data, and half as testing. For DCA, our method and ITML, we only used this training data to generate a set of randomly chosen constraint pairs (200 for the UCI datasets, 2000 for USPS, in each case half similarity and half dissimilarity constraints). RCA is similar, but only takes the 100 positive constraints. LMNN, however, utilizes the full label set of the training data provided to pick its constraints and learn its metric.

Table 1 shows the classification error rate of the tested linear metric learning methods over the eight datasets, using $k = 3$ for the kNN classifier. We observe that our proposed metric learning algorithm performs comparably with the state of the art methods. Our method is surprisingly robust, never yielding a result significantly worse than the best tested algorithm, and performing noticably better at the large-scale USPS task.

5.2 Kernelized Metric Learning for Object Recognition in the Caltech-101 Dataset

We also apply the kernelized version of our method to the object recognition task using the Caltech-101 dataset, again using a kNN classifier to test classification effectiveness. We compute the distance between images using the learned kernel-based metric with two different image kernels: (1) RBF kernel applied to Sparse SIFT features [Sivic and Zisserman, 2004] (2) RBF kernel applied to GIST features [Oliva and Torralba, 2001].

We evaluate the performance of both our method and ITML applied to this task, with both methods learning a kernel-based distance metric. We set $k = 1$ for the kNN classifier. We choose 10 classes randomly from Caltech-101 and vary the number of training samples M per class. Considering the remaining samples as test samples, we measure the mean per-class recognition accuracy. The results shown are from 5 runs of this process. Figure 1 shows the results of this test. For both types of features, both metrics perform better than the unlearned kernel at most points, and our method performed better than ITML.

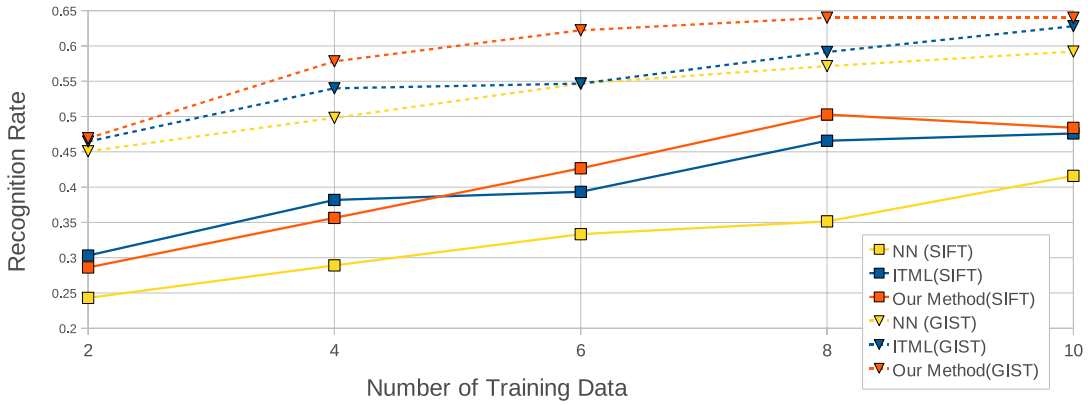

Figure 1: Object recognition on the Caltech-101 dataset, with the RBF kernel applied to sparse sift and GIST features. Metrics learned by our kernel-based method improve NN recognition accuracy over the distance metric learned via ITML and the original kernel.

6. Conclusion

In this paper we propose an efficient metric learning method based on the maximum margin framework with similar/dissimilar pairwise constraints and extend this to the kernel space. We first showed how to use structured SVM model with the cutting-plane algorithm for Max-Margin metric learning. We also proposed a example-based matching pursuit method which can make the kernel-based metric learning more computationally efficient. In experiments, results shows our metric is comparable to the state of the art techniques.While our method performed well on the medium-scale USPS dataset, we would like to apply it to truly large scale problems, such as document retrieval with 20 Newsgroups [Lang, 1995].

Acknowledgement

This work was partially supported by the National Science Foundation CAREER grant (IIS-0845282), the Army Research Office (W911NF-11-1-0090), DARPA CSSG (HR0011-09-1-0022 and D11AP00245). Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, ARO, NSF or the U.S. Government.

References

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *ICML*, volume 20, page 11, 2003.

J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216. ACM, 2007.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 106(1):59–70, 2007. ISSN 1077-3142.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *ICML*, pages 320–327. ACM, 2008.

A. Frank and A. Asuncion. UCI machine learning repository, 2010.

A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. *NIPS*, 19:417, 2007. ISSN 1049-5258.

A. Globerson and S. Roweis. Metric learning by collapsing classes. *NIPS*, 18:451, 2006. ISSN 1049-5258.

S.C.H. Hoi, W. Liu, M.R. Lyu, and W.Y. Ma. Learning distance metrics with contextual constraints for image retrieval. In *CVPR*, volume 2, pages 2072–2078. IEEE, 2006. ISBN 0769525970.

P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. *CVPR*, 2008.

T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD*, pages 217–226. ACM, 2006. ISBN 1595933395.

T. Joachims, T. Finley, and C.N.J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. ISSN 0885-6125.

J.E. Kelley Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, pages 703–712, 1960. ISSN 0368-4245.

Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

N. Nguyen and Y. Guo. Metric Learning: A Support Vector Approach. *ECML PKDD*, pages 125–136, 2008.

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3): 145–175, 2001. ISSN 0920-5691.

Sam Roweis. Usps handwritten digits dataset, 2010.

M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. *Advances in Neural Information Processing Systems (NIPS)*, page 41, 2004.

S. Shalev-Shwartz, Y. Singer, and A.Y. Ng. Online and batch learning of pseudo-metrics. In *ICML*, page 94. ACM, 2004.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML*, pages 807–814. ACM, 2007.

C. Shen, J. Kim, and L. Wang. Scalable Large-Margin Mahalanobis Distance Metric Learning. *Neural Networks, IEEE Transactions on*, 21(9):1524–1530, 2010. ISSN 1045-9227.

J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. In *CVPR*, volume 1. IEEE, 2004. ISBN 0769521584.

I.W. Tsang, P.M. Cheung, and J.T. Kwok. Kernel relevant component analysis for distance metric learning. In *IJCNN*, volume 2, pages 954–959. IEEE, 2005. ISBN 0780390482.

K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *JLMR*, 10: 207–244, 2009. ISSN 1532-4435.

E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *NIPS*, pages 521–528, 2003. ISSN 1049-5258.