Random Forests for Metric Learning with Implicit Pairwise Position Dependence

Caiming Xiong
Department of Computer Science and Engineering
SUNY at Buffalo
cxiong@buffalo.edu

David Johnson
Department of Computer Science and Engineering
SUNY at Buffalo
davidjoh@buffalo.edu

Ran Xu
Department of Computer Science and Engineering
SUNY at Buffalo
rxu2@buffalo.edu

Jason J. Corso
Department of Computer Science and Engineering,
SUNY at Buffalo,
jcorso@buffalo.edu

ABSTRACT

Metric learning makes it plausible to learn semantically meaningful distances for complex distributions of data using label or pairwise constraint information. However, to date, most metric learning methods are based on a single Mahalanobis metric, which cannot handle heterogeneous data well. Those that learn multiple metrics throughout the feature space have demonstrated superior accuracy, but at a severe cost to computational efficiency. Here, we adopt a new angle on the metric learning problem and learn a single metric that is able to implicitly adapt its distance function throughout the feature space. This metric adaptation is accomplished by using a random forest-based classifier to underpin the distance function and incorporate both absolute pairwise position and standard relative position into the representation. We have implemented and tested our method against state of the art global and multi-metric methods on a variety of data sets. Overall, the proposed method outperforms both types of method in terms of accuracy (consistently ranked first) and is an order of magnitude faster than state of the art multi-metric methods (16x faster in the worst case).

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; I.5.1 [**Computing Methodologies**]: Pattern Recognition—*models*

General Terms

Algorithms

Keywords

Metric learning, random forests, pairwise constraints

KDD'12, August 12–16, 2012, Beijing, China.

1. INTRODUCTION

Although the Euclidean distance is a simple and convenient metric, it is often not an accurate representation of the underlying shape of the data [11]. Such a representation is crucial in many real-world applications [5,31], such as object classification [10, 12], text document retrieval [16, 26], data clustering [29] and face verification [8, 20], and methods that learn a distance metric from training data have hence been widely studied in recent years. We present a new angle on the metric learning problem based on random forests [1,6] as the underlying distance representation.[1] The emphasis of our work is the capability to incorporate the absolute position of point pairs in the input space without requiring a separate metric per instance or exemplar. In doing so, our method, called random forest distance (RFD), is able to adapt to the underlying shape of the data by varying the metric based on the *position* of sample pairs in the feature space while maintaining the efficiency of a single metric. In some sense, our method achieves a middle-ground between the two main classes of existing methods—single, global distance functions and multi-metric sets of distance functions—overcoming the limitations of both (see Figure 1 for an illustrative example). We next elaborate upon these comparisons.

The metric learning literature has been dominated by methods that learn a global Mahalanobis metric, with representative methods [3,9,14,21,24,25,28,30]. In brief, given a set of pairwise constraints (either by sampling from label data, or collecting side information in the semi-supervised case), indicating pairs of points that should or should not be grouped (i.e., have small or large distance, respectively), the goal is to find the appropriate linear transformation of the data to best satisfy these constraints. One such method [30] minimizes the distance between positively-linked points subject to the constraint that negatively-linked points are separated, but requires solving a computationaly expensive semidefinite programming problem. Relevant Component Analysis (RCA) [3] learns a linear Mahalanobis transformation to satisfy a set of positive constraints. Discriminant Component Analysis (DCA) [14] extends RCA by exploring negative constraints. ITML [9] minimizes the LogDet di-

[1]An early version of this manuscript has been released on arXiv, article id:1201.0610

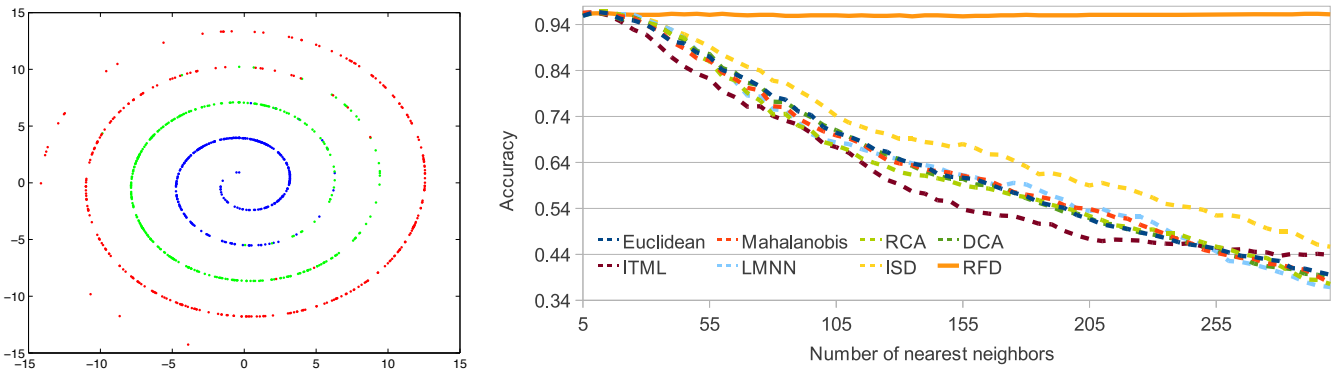

Figure 1: An example using a classic swiss roll data set comparing both global and position-specific Mahalanobis-based methods with our proposed method, RFD. All methods, including the baseline Euclidean, perform well at low K-values due to local linearity. However, as K increases and the global nonlinearity of the data becomes important, the monolithic methods' inability to incorporate position information causes their performance to degrade until it is little better than chance. The position-specific ISD method performs somewhat better, but even with a Mahalanobis matrix at every point it is unable to capture the globally nonlinear relations between points. Our method, by comparison, shows no degradation as K increases. (3 classes, 900 samples, validated using K-nearest neighbor classification, with varying K)

vergence under positive and negative linear constraints, and LMNN [24, 28] learns a distance metric through the maximum margin framework. Nguyen et al. [21] formulate metric learning as a quadratic semidefinite programming problem with local neighborhood constraints and linear time complexity in the original feature space. More recently, researchers have begun developing fast algorithms that can work in an online manner, such as POLA [23], MLCL [13] and LEGO [15].

These global methods learn a single Mahalanobis metric using the relative position of point pairs:

$$\text{Dist}\,(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^{\mathsf{T}} \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j) \qquad (1)$$

Although the resulting single metric is efficient, it is limited in its capacity to capture the shape of complex data(see Figure 1 as example). In contrast, a second class, called multi-metric methods [11, 12, 29], distributes distance metrics throughout the input space; in the limit, they estimate a distance metric per instance or exemplar, e.g. [11,12] for the case of Mahalanobis metrics. Zhan et al. [32] extend [11] by propagating metrics learned on training exemplars to learn a metric matrix for each unlabeled point as well. However, these point-based multi-metric methods all suffer from high time and space complexity due to the need to learn and store $\mathbf{O}(N)$ d by d metric matrices. Wu et al. [29] learns a Bregman distance function which does not need to store the metric matrices for each data points, but it needs to take $\mathbf{O}(N)$ time to calculate the distance of single pair of points which is impractical for large scale data sets. A more efficient approach to this second class is to divide the data into subsets and learn a metric for each subset [2, 27]. However, these methods have strong assumptions in generating these subsets; for example, [2] learns at most one metric per category, forfeiting the possibility that different samples within a category may require different metrics.

We propose a metric learning method that is able to achieve both the efficiency of the global methods and specificity of the multi-metric methods. Our method, the random forest distance (RFD), transforms the metric learning problem into a binary classification problem and uses random forests as the underlying representation [1, 4, 6, 17]. In this general form, we are able to incorporate the position of samples implicitly into the metric and yet maintain a single and efficient global metric. To that end, we use a novel point-pair mapping function that encodes both the position of the points relative to each other and their absolute position within the feature space. Our experimental analyses demonstrate the importance of incorporating position information into the metric (Section 3).

We use the random forest as the underlying representation for several reasons. First, the output of the random forest algorithm is a simple "yes" or "no" vote from each tree in the forest. In our case, "no" votes correspond to positively constrained training data, and "yes" votes correspond to negatively constrained training data. The number of yes votes, then, is effectively a distance function, representing the relative resemblance of a point pair to pairs that are known to be dissimilar versus pairs that are known to be similar. Second, random forests are efficient and scale well, and have been shown to be one of the most powerful and scalable supervised methods for handling high-dimensional data [7]—in contrast to instance-specific multi-metric methods [11, 12], the storage requirement of our method is independent of the size of the input data set. Our experimental results indicate RFD is at least 16 times faster than the state of the art multi-metric method. Third, because random forests are non-parametric, they make minimal assumptions about the shape and patterning of the data [6], affording a flexible model that is inherently nonlinear. In the next section, we describe the new RFD method in more detail, followed by a thorough comparison to the state of the art in Section 3.

2. RANDOM FOREST DISTANCE: IMPLICITLY POSITION-DEPENDENT METRIC LEARNING

Our random forest-based approach is inspired by several

other recent advances in metric learning [2,23] that reformulate the metric learning problem into a classification problem. However, where these approaches restricted the form of the learned distance function to a Mahalanobis matrix, thus precluding the use of position information, we adopt a more general formulation of the classification problem that removes this restriction.

Given the instance set $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, each $\mathbf{x}_i \in \mathbb{R}^m$ is a vector of m features. Taking a geometric interpretation of each $\mathbf{x}_i$, we consider $\mathbf{x}_i$ the *position* of sample i in the space $\mathbb{R}^m$. The value of this interpretation will become clear throughout the paper as the learned metric will implicitly vary over $\mathbb{R}^m$, which allows it to adapt the learned metric based on local structure in a manner similar to the instance-specific multi-metric methods, e.g., [11]. Denote two pairwise constraint sets: a must-link constraint set $S = \{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are similar}\}$ and a cannot-link constraint set $D = \{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are dissimilar}\}$. For any constraint $(\mathbf{x}_i, \mathbf{x}_j)$, denote y_{ij} as the ideal distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. If $(\mathbf{x}_i, \mathbf{x}_j) \in S$, then the distance $y_{ij} = 0$, otherwise, their distance $y_{ij} = 1$. Therefore, we seek a function $\text{Dist}\,(\cdot,\cdot)$ from an appropriate function space H:

$$\text{Dist}\,(\cdot,\cdot)^* = \underset{\text{Dist}\,(\cdot,\cdot)\in H}{\text{argmin}} \frac{1}{|S \cup D|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S\cup D} l(\text{Dist}\,(\mathbf{x}_i, \mathbf{x}_j), y_{ij}) \ , \tag{2}$$

where $l(\cdot)$ is some loss function that will be specified by the specific classifier chosen. In our random forests case, we minimize expected loss, as in many classification problems. So consider $\text{Dist}\,(\cdot,\cdot)$ to be a binary classifier for the classes 0 and 1. For flexibility, we redefine the problem as $\text{Dist}\,(\mathbf{x}_i, \mathbf{x}_j) = F(\phi(\mathbf{x}_i, \mathbf{x}_j))$, where $F(\cdot)$ is some classification model, and $\phi(\mathbf{x}_i, \mathbf{x}_j)$ is a mapping function that maps the pair $(\mathbf{x}_i, \mathbf{x}_j)$ to a feature vector that will serve as input for the classifier function F. To train F, we transform each constraint pair using the mapping function $\{(\mathbf{x}_i, \mathbf{x}_j), y_{ij}\} \rightarrow \{\phi(\mathbf{x}_i, \mathbf{x}_j), y_{ij}\}$ and submit the resulting set of vectors and labels as training data. We next describe the feature mapping function ϕ.

2.1 Mapping function for implicitly position-dependent metric learning

In actuality, all metric learning methods implicitly employ a mapping function $\phi(\mathbf{x}_i, \mathbf{x}_j)$. Standard Mahalanobis-based methods all learn a (positive semidefinite) metric matrix $\mathbf{W}$, and a distance function of the form $\text{Dist}\,(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)$, which can be reformulated as $\text{Dist}\,(\mathbf{x}_i, \mathbf{x}_j) = \vec{[}\mathbf{W}]^\mathsf{T}\vec{[}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}]$, where $\vec{[}\cdot]$ denotes vectorization or *flattening* of a matrix. Mahalanobis-based methods can thus be viewed as using the mapping function $\phi(\mathbf{x}_i, \mathbf{x}_j) = \vec{[}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\mathsf{T}]$. This function encodes only relative position information, and adapting it to include absolute position information yields significant theoretical problems for Mahalanobis methods based on global weights (see Section 2.2).

However, our formulation affords a more general mapping function:

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = \begin{bmatrix}\mathbf{u}(\mathbf{x}_i, \mathbf{x}_j) \\ \mathbf{v}(\mathbf{x}_i, \mathbf{x}_j)\end{bmatrix} = \begin{bmatrix}|\mathbf{x}_i - \mathbf{x}_j| \\ \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)\end{bmatrix} \ , \tag{3}$$

which considers both the relative location of the samples $\mathbf{u}$ as well as their absolute position $\mathbf{v}$. The output feature vector is the concatenation of these two and in $\mathbb{R}^{2m}$.

The relative location $\mathbf{u}$ represents the same information as the Mahalanobis mapping function. Note that we take the absolute value of $\mathbf{u}$ to enforce symmetry within the learned metric. The primary difference between our mapping function and that of previous methods is thus the information contained in $\mathbf{v}$—the mean of the two point vectors. These features localize each mapped pair to a region of the space, which allows our method to adapt to heterogeneous distributions of data. It is for this reason that we consider our learned metric to be implicitly position-dependent. Note the earlier methods that learn position-based metrics, i.e. the methods that learn a metric per instance such as [11], incorporate absolute position of each instance only, whereas we incorporate the absolute position of each instance pair, which adds additional modeling versatility.

We note that alternate encodings of the position information are possible but have shortcomings. For example, we could choose to simply concatenate the position of the two points rather than average them, but this approach raises the issue of ordering the points. Using $\mathbf{v} = \begin{bmatrix}\mathbf{x}_i^\mathsf{T} & \mathbf{x}_j^\mathsf{T}\end{bmatrix}^\mathsf{T}$ would again yield a nonsymmetric feature, and an arbitrary ordering rule would not guarantee meaningful feature comparisons. The usefulness of position information varies depending on the data set. For data that is largely linear and homogenous, including $\mathbf{v}$ will only add noise to the features, and could worsen the accuracy. In our experiments, we find that for many real data sets (and particularly for more difficult data sets) the inclusion of $\mathbf{v}$ significantly improves the performance of the metric (see Section 3).

2.2 Why not Mahalanobis?

Before investigating the use of wholly new metric representations, it is reasonable to consider the use of absolute position information in a traditional Mahalanobis or Mahalanobis-like metric based on global weights. Here we show that such a formulation has immediate theoretical shortcomings.

Assume there exists a Mahalanobis metric $M > 0$ utilizing our position-embedded mapping function. Consider that $M = \begin{bmatrix}M_{uu} & M_{uv} \\ M_{vu} & M_{vv}\end{bmatrix}$. Since $M > 0$, we know $M_{uu} > 0$ and $M_{vv} > 0$.

Now, for any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ such that $\mathbf{x}_i = k \cdot \mathbf{x}_j$ where k is a scalar, let us build two pairs $(\mathbf{x}_i, \mathbf{x}_i)$ and $(\mathbf{x}_j, \mathbf{x}_j)$. Obviously the distance between each of these point pairs should be small, so the difference between these two distances should also be small:

$$\begin{aligned} Dist(\mathbf{x}_i, \mathbf{x}_i) - Dist(\mathbf{x}_j, \mathbf{x}_j) = &\begin{bmatrix}\mathbf{u}(\mathbf{x}_i, \mathbf{x}_i) \\ \mathbf{v}(\mathbf{x}_i, \mathbf{x}_i)\end{bmatrix}^T \cdot M \cdot \begin{bmatrix}\mathbf{u}(\mathbf{x}_i, \mathbf{x}_i) \\ \mathbf{v}(\mathbf{x}_i, \mathbf{x}_i)\end{bmatrix} \\ &- \begin{bmatrix}\mathbf{u}(\mathbf{x}_j, \mathbf{x}_j) \\ \mathbf{v}(\mathbf{x}_j, \mathbf{x}_j)\end{bmatrix}^T \cdot M \cdot \begin{bmatrix}\mathbf{u}(\mathbf{x}_j, \mathbf{x}_j) \\ \mathbf{v}(\mathbf{x}_j, \mathbf{x}_j)\end{bmatrix} \end{aligned}$$

Since $\mathbf{u}(\mathbf{x}_i, \mathbf{x}_i) = \mathbf{u}(\mathbf{x}_j, \mathbf{x}_j) = 0$,

$$\begin{aligned} Dist(\mathbf{x}_i, \mathbf{x}_i) - Dist(\mathbf{x}_j, \mathbf{x}_j) = \ &\mathbf{v}(\mathbf{x}_i, \mathbf{x}_i)^T M_{vv} \mathbf{v}(\mathbf{x}_i, \mathbf{x}_i) \\ &- \mathbf{v}(\mathbf{x}_j, \mathbf{x}_j)^T M_{vv} \mathbf{v}(\mathbf{x}_j, \mathbf{x}_j) \end{aligned}$$

Since $\mathbf{x}_i = k \cdot \mathbf{x}_j$,

Table 1: UCI data sets used for KNN-classification testing

Dataset	Size	Dim.	No. Classes	Dataset	Size	Dim.	No. Classes
Balance	625	4	3	Iris	150	4	3
BUPA Liver Disorders	345	6	2	Pima Indians Diabetes	768	8	2
Breast Cancer	699	10	2	Wine	178	13	3
Image Segmentation	2310	19	7	Sonar	208	60	2
Semeion Handwritten Digits	1593	256	10	Multiple Features Handwritten Digits	2000	649	10

$$Dist(\mathbf{x}_i, \mathbf{x}_i) - Dist(\mathbf{x}_j, \mathbf{x}_j) = (k^2 - 1)(\mathbf{v}(\mathbf{x}_j, \mathbf{x}_j))^T M_{vv}(\mathbf{v}(\mathbf{x}_j, \mathbf{x}_j))$$

Thus, as $k \to +\infty$, $Dist(\mathbf{x}_i, \mathbf{x}_i) - Dist(\mathbf{x}_j, \mathbf{x}_j) \to +\infty$.

This is a nonsensical result, and clearly undesirable in a metric. For this reason, traditional Mahalanobis formulations based on global covariance weights are not suitable for use with our position-sensitive mapping function, and we thus pursue alternative distance representations.

2.3 Random forests for metric learning

Random forests are well studied in the machine learning literature and we do not describe them in any detail; the interested reader is directed to [1, 6]. In brief, a random forest is a set of T decision trees $\{f_t\}_{t=1}^{T}$ operating on a common feature space, in our case $\mathbb{R}^{2m}$. To evaluate a point-pair $(\mathbf{x}_i, \mathbf{x}_j)$, each tree independently classifies the sample (based on the leaf node at which the point-pair arrives) as similar or dissimilar (0 or 1, respectively) and the forest averages them, essentially regressing a distance measure on the point-pair:

$$\text{Dist}(\mathbf{x}_i, \mathbf{x}_j) = F(\phi(\mathbf{x}_i, \mathbf{x}_j)) = \frac{1}{T}\sum_{t=1}^{T} f_t(\phi(\mathbf{x}_i, \mathbf{x}_j)) \ , \qquad (4)$$

where $f_t(\cdot)$ is the classification output of tree t.

It has been found empirically that random forests scale well with increasing dimensionality, compared with other classification methods [7], and, as a decision tree-based method, they are inherently nonlinear. Hence, our use of them in RFD as a regression algorithm allows for a more scalable and more flexible metric than is possible using Mahalanobis methods. Moreover, the incorporation of position information into this classification function (as described in Section 2.1) allows the metric to implicitly adapt to different regions over the feature space. In other words, when a decision tree in the random forest selects a node split based on a value of the absolute position $\mathbf{v}$ sub-vector (see Eq. 3), then all evaluation in the sub-tree is *localized* to a specific half-space of $\mathbb{R}^m$. Subsequent splits on elements of $\mathbf{v}$ further refine the sub-space of emphasis $\mathbb{R}^m$. Indeed, each path through a decision tree in the random forest is localized to a particular (possibly overlapping) sub-space.

The RFD is not technically a *metric* but rather a *pseudosemi-metric*. Although RFD can easily be shown to be non-negative and symmetric, it does not satisfy the triangle inequality (i.e., $\text{Dist}(\mathbf{x}_1, \mathbf{x}_2) \leq \text{Dist}(\mathbf{x}_1, \mathbf{x}_3) + \text{Dist}(\mathbf{x}_2, \mathbf{x}_3)$) or the implication that $\text{Dist}(\mathbf{x}_1, \mathbf{x}_2) = 0 \implies \mathbf{x}_1 = \mathbf{x}_2$, sometimes called identity of indiscernibles. It is straightforward to construct examples for both of these cases. Although this point may appear problematic, it is not uncommon in the metric learning literature. For example, by necessity, no metric whose distance function varies across the feature space can guarantee the triangle inequality is satisfied. [11, 12] similarly cannot satisfy the triangle inequality. Our method *must* violate the triangle inequality in order to fulfill our original objective of producing a metric that incorporates position data. Moreover, our extensive experimental results demonstrate the capability of RFD as a distance (Section 3).

2.4 Theoretic analysis of RFD performance

For simplicity, we assume that the classes of must-link and cannot-link are balanced $|S| == |D|$. For the case of a single random decision tree, we suppose each leaf has a true probabilistic margin γ with a total of C training pairs that reach the leaf during the training process. Thus, a fraction $\frac{1}{2} + \gamma$ of C training pairs belongs to one class (e.g. must-link pair constraints), and a fraction $\frac{1}{2} - \gamma$ of them belong to the other class (e.g. cannot-link pair constraints). We denote a leaf as a must-link leaf if a majority of the training pairs in the leaf are must-link, and likewise denote it a cannot-link leaf if the majority of pairs are cannot-link. According to [19], it is straightforward to show that the probability that must-link pair will end up in a must-link leaf during testing is $\frac{1}{2} + \gamma$, and similarly for cannot-link pairs.

For the purposes of establishing error bounds, we can relax RFD to a binary classification algorithm that discriminates between must-link and cannot-link pairs. We do this by employing an *evidence voting* [22] procedure, wherein we drop a test pair through the T learned random decision trees and take the mode of the $C \cdot T$ training pairs stored in the leaves reached by the testing pair as a classification result. Given this setup, we can establish an upper bound on pair classification error in RFD:

PROPOSITION 1. *The probability of pair classification error in RFD* $P(\epsilon)_{RFD}$ *is* $\leq \exp(-8CT\gamma^4)$.

PROOF. Details in [22]. □

Computational complexity Since all trees are generated independently, we can build these trees in parallel. Thus the major computational cost for our method is the generation of single tree based on training data. For building single tree, it takes $\mathcal{O}(n \log n)$ time. And the learning processes of nodes from the same depth of one tree are also independent, so we can use the parallel computing technique for learning the nodes of same depth of tree to make our method more efficient.

3. EXPERIMENTS AND ANALYSIS

In this section, we present a set of experiments comparing our method to state of the art metric learning techniques on both a range of UCI data sets (Table 1) and an image data set taken from the Corel database. To substantiate our claim of computational efficiency, we also provide an analysis of running time efficiency relative to an existing position-dependent metric learning method. We also include results illustrating the effect of varying forest sizes on our algorithm's accuracy.

For the UCI data sets, we compare performance at the **K**-nearest neighbor classification task against both standard Mahalanobis methods and point-based position-dependent methods.[2] For the former, we test K-NN classification accuracy at a range of **K**-values (as in Figure 1), while the latter relies on results published by other methods' authors, and thus uses a fixed **K**. For the image data set, we measure accuracy at **K**-NN retrieval,

[2]We note that previous work has established a connection between random forests and nearest neighbor analysis [18], but this relationship does not apply to RFD. In RFD, the "points" within the forest are actually point pairs, and thus potential nearest neighbors identified via the forest would represent similar pair constraints, rather than similar data elements.

rather than **K**-NN classification. We compare our results to several Mahalanobis methods.

The following is an overview of the primary experimental findings to be covered in the following sections.

1. RFD has the best overall performance on ten UCI data sets ranging from 4 to 649 dimensions against four state of the art and two baseline global Mahalanobis-based methods (Figure 2 and Table 2).
2. RFD has comparable or superior accuracy to state of the art position-specific methods (Table 3).
3. RFD is 16 to 85 times faster than the state of the art position-specific method (Table 4).
4. RFD outperforms the state of the art in nine out of ten categories in the benchmark Corel image retrieval problem (Figure 4).

3.1 Comparison with global Mahalanobis metric learning methods

We first compare our method to a set of state of the art Mahalanobis metric learning methods: RCA [3], DCA [14], Information-Theoretic Metric Learning (ITML) [9] and distance metric learning for large-margin nearest neighbor classification (LMNN) [24, 28]. For our method, we test using the full feature mapping including relative position data, $\mathbf{u}$, and absolute pairwise position data, $\mathbf{v}$, (RFD (+P)) as well as with only relative position data, $\mathbf{u}$, (RFD (−P)). To provide a baseline, we also show results using both the Euclidean distance and a heuristic Mahalanobis metric, where the $\mathbf{W}$ used is simply the covariance matrix for the data. All algorithm code was obtained from authors' websites, for which we are indebted (*our code is available, as well*).

We test each algorithm on a number of standard small to medium scale UCI data sets (see Table 1). All algorithms are trained using 1000 positive and 1000 negative constraints per class, with the exceptions of RCA, which used only the 1000 positive constraints and LMNN, which used the full label set to actively select a (generally much larger) set of constraints. In each case, we set the number of trees used by our method to 400 (*see Section 3.2 for a discussion of the effect of varying forest sizes*).

Testing is performed using 5-fold cross validation on the **K** nearest-neighbor classification task. In each fold, metric constraints, as well as neighbors used for classification, are drawn only from the training data. Rather than selecting a single **K**-value for this task, we test with varying **K**s, increasing in increments of 5 up to the maximum possible value for each data set (i.e. the number of elements in the smallest class). By varying **K** in this way, we are able to gain some insight into each method's ability to capture the global variation in a data set. When **K** is small, most of the identified neighbors lie within a small local region surrounding the query point, enabling linear metrics to perform fairly well even on globally nonlinear data by taking advantage of local linearity. However, as **K** increases, local linearity becomes less practical, and the quality of the metric's representation of the global structure of the data is exposed. Though the accuracy results at higher **K** values do not have strong implications for each method's efficacy for the specific task of **K**-NN classification (where an ideal **K** value can just be selected by cross-validation), they do indicate overall metric performance, and are highly relevant to other tasks, such as retrieval.

Figure 2 shows the accuracy plots for ten UCI datasets. RFD is consistently near the top performers on these various data sets. In the lower dimension case (Iris), most methods perform well, and RFD without position information outperforms RFD with position information (this is the sole data set in which this occurs), which we attribute to the limited data set size (150 samples) and the position information acting as a distractor in this small and highly linear case. In all other cases, the RFD with absolute position information significantly outperforms RFD without it. In many of the more difficult cases (Diabetes, Segmentation, Sonar), RFD with position information significantly outperforms the field. This result is suggestive that RFD can scale well with increasing dimensionality, which is consistent with the findings from the literature that random forests are one of the most robust classification methods for high-dimensional data [7].

Table 2 provides a summary statistic of the methods by computing the mean-rank (lower better) over the ten data sets at varying **K**-values. For all but one value of **K**, RFD with absolute position information has the best mean rank of all the methods (and for the off-case, it is ranked second). RFD without absolute position information performs comparatively poorer, underscoring the utility of the absolute position information. In summary, the results in Table 2 show that RFD is consistently able to outperform the state of the art in global metric learning methods on various benchmark problems.

3.2 Varying forest size

One question that must be addressed when using RFD is how many trees must or should be learned in order to obtain good performance. Increasing the size of the forest increases computation and space requirements, and past a certain point yields little or no improvement and may possibly over-train. It is beyond the scope of this paper to provide a full answer as to how many trees are needed in RFD, but we have made some observations.

First, the addition of absolute position information noticeably increases the benefit that may be obtained from additional trees (see Figure 3). This result is unsurprising, considering the increased size of the feature vector, as well as the increased degree of fine-tuning possible for a metric that can vary from region to region. Second, in our experiments we observe significant improvements in accuracy up to about 100 trees, even without position information, and would recommend this as a reasonable minimum value. Intuition suggests that larger constraint-sets and more complex data distributions may require larger forests, but these two points have not yet been thoroughly explored by our group.

3.3 Comparison with position-specific multi-metric methods

We compare our method to three multi-metric methods that incorporate absolute position (by way of instance-specific metrics): FSM, FSSM and ISD. FSM [11] learns an instance-specific distance for each labeled example. FSSM [12] is an extension of FSM that enforces global consistency and comparability among the different instance-specific metrics. ISD [32] first learns instance-specific distance metrics for each labeled data point, then uses metric propagation to generate instance-specific metrics for unlabeled points as well.

We again use the ten UCI data sets, but under the same conditions used by these methods' authors. Accuracy is measured on the **K**-NN task (**K**=11) with three-fold cross validation. The parameters of the compared methods are set as suggested in [32]. Our RFD method chooses 1% of the available positive constraints and 1% of the available negative constraints, and constructs a random forest with 1000 trees. We report the average result of ten different runs on each data set, with random partitions of training/testing data generated each time (see Table 3). These results show that our RFD method yields performance better than or comparable to state of the art explicit multi-metric learning methods. Additionally, because we only learn one distance function and random forests are an inherently efficient technique, our method offers significantly better computational efficiency than these instance-specific approaches (see Table 4)—between 16 to 85 times faster than ISD.

The comparable level of accuracy is not surprising. While our method is a single metric in form, in practice its implicit position-dependence allows it to act like a multi-metric system. Notably, because our method learns using the position of each point-pair rather than each point, it can potentially encode up to n^2 implicit position-specific metrics, rather than the $\mathbf{O}(n)$ learned by existing position-dependent methods, which learn a single metric per instance/position. RFD is a stronger way to learn a position-dependent metric, because even explicit multi-metric methods will fail over global distances in cases where a single (Mahalanobis) metric cannot capture the relationship between its associated point and every other point in the data.

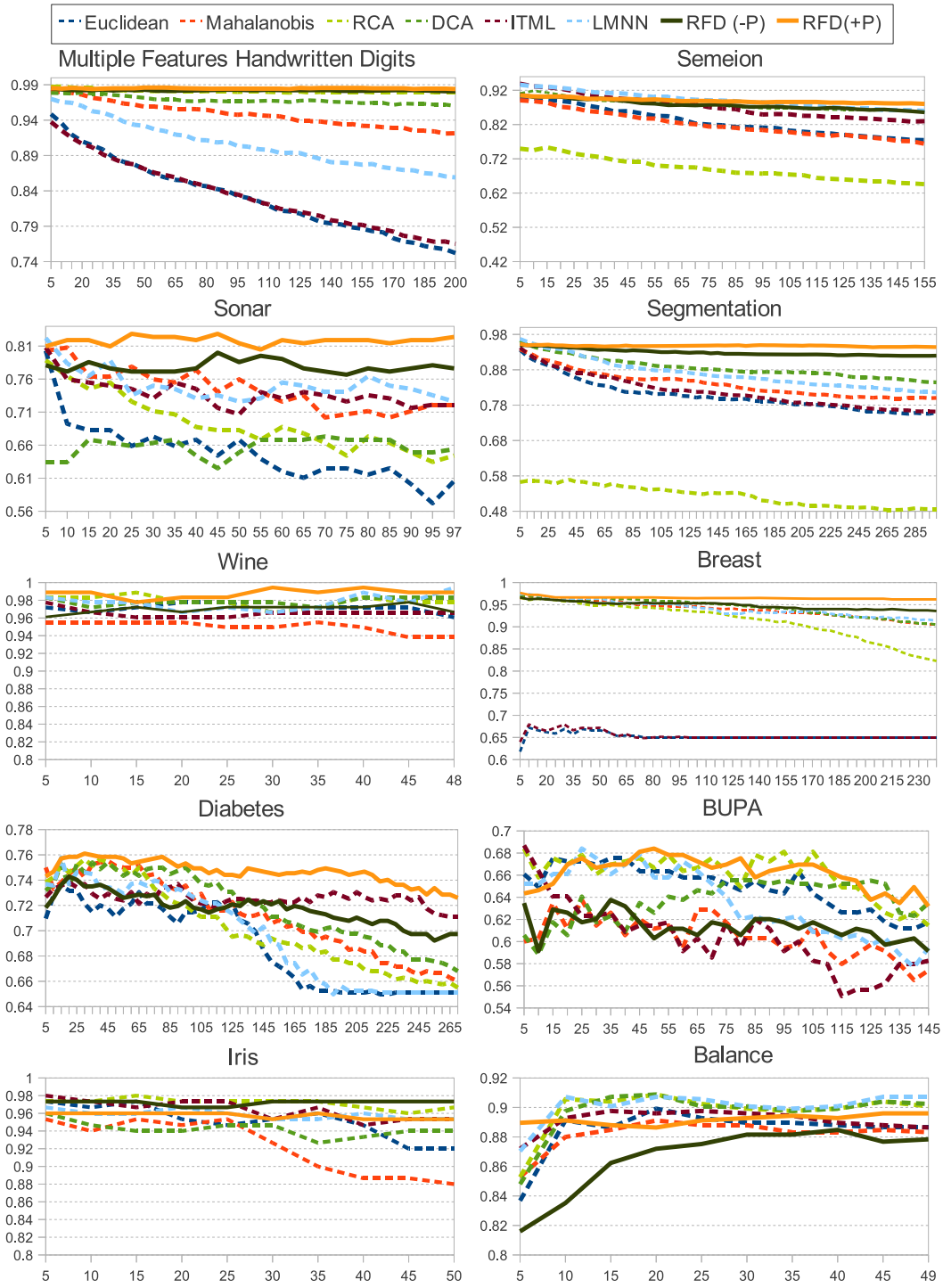

Figure 2: K-nearest neighbor classification results with varying K values of RFD versus assorted global Mahalanobis methods on 10 UCI data sets . Plots show K-nearest neighbor K-value versus accuracy. Note in particular the segmentation and breast datasets, where RFD shows little or no degradation over increasing distances, while other methods steadily decline in accuracy. Also note that the inclusion of position information in the RFD yields higher performance on all but the low-dimensional and highly linear iris dataset.

Table 2: Mean K-nearest neighbor classification accuracy ranking on 10 UCI data sets at varying K values (lower rank is better). The mean ranking is shown in each table cell as well as the corresponding overall rank, in parentheses. As expected, Euclidean distance nearly always has the worst rank. RFD with absolute position information attains the best rank in nearly all cases, and the relative performance of both RFD methods improves as K increases.

k-value	Euclid	Mahal	RCA	DCA	ITML	LMNN	RFD (−P)	RFD (+P)
5	5.8 (8)	5.7 (7)	4.3 (4)	4.8 (5)	3.9 (3)	3.2 (2)	5.4 (6)	**2.9 (1)**
10	6.1 (8)	5.6 (7)	3.7 (3)	4.6 (4)	4.8 (5)	**2.9 (1)**	5.1 (6)	3.2 (2)
15	5.7 (8)	5.4 (6)	3.9 (3)	4.7 (5)	5.6 (7)	3.1 (2)	4.6 (4)	**3 (1)**
20	5.6 (8)	5.4 (7)	3.8 (3)	5.2 (5)	5.3 (6)	3.7 (2)	4.5 (4)	**2.5 (1)**
25	6.1 (8)	5.3 (6)	4 (3)	4.5 (4)	5.4 (7)	3.4 (2)	4.8 (5)	**2.5 (1)**
30	5.8 (7)	5.9 (8)	4.5 (5)	4.3 (3)	5.3 (6)	3.5 (2)	4.3 (3)	**2.4 (1)**
35	5.8 (8)	5.4 (6)	4.3 (4)	4.9 (5)	5.5 (7)	4 (3)	3.8 (2)	**2.3 (1)**
45	6.6 (8)	5.5 (6)	4.4 (4)	4.4 (4)	5.9 (7)	3.3 (2)	4.1 (3)	**1.8 (1)**
Max	6.5 (8)	6.1 (7)	5.1 (5)	3.7 (3)	5.5 (6)	3.7 (3)	3.5 (2)	**1.9 (1)**

Table 3: Comparison of test error (mean ± STD) for position-dependent metric learning methods. The best performance on each data set is shown in bold. We note that our method yields the best accuracy on 3 out of 5 data sets tested, and is within 1% of the best on the remaining 2.

Dataset	RFD	ISD L1	ISD L2	FSM	FSSM
Balance	.120±.024	**.114±.013**	.116±.014	0.134±.020	0.143±.013
Diabetes	**.241±.028**	.287±.019	.269±.023	.342±.050	.322±.232
Breast(Scaled)	**.030±.011**	.0.31±.010	**.030±.010**	.102±.041	.112±.029
German	.277±.039	.277±.015	**.274±.013**	.275±.021	0.275±.060
Haberman	**.273±.029**	.277±.029	**.273±.025**	.276±.032	.276±.029

Table 4: Run-time comparison of ISD and RFD (with position information, using 1000 trees) across several UCI data sets. All times are in seconds. Results were obtained by performing 5-fold cross validation and averaging the time for each fold. *Note that ISD is multithreading across 12 cores, while our implementation of RFD is fully sequential.

Dataset	ISD Time*	RFD Time	ISD:RFD Ratio
Iris	34.6	2.1	16.4
Balance	620.3	11.2	55.3
Breast (scaled)	657.4	7.8	84.6
Diabetes	849.5	14.7	57.8

3.4 Retrieval on the Corel image data set

We also evaluate our method's performance on the challenging image retrieval task because this task differs from **K**-NN classification by emphasizing the accuracy of individual pairwise distances rather than broad patterns. For this task, we use an image data set taken from the Corel image database. We select ten image categories of varying types (cats, roses, mountains, etc.—the classes and images are similar to those used by Hoi et al. to validate DCA [14]), each with a clear semantic meaning. Each class contains 100 images, for a total of 1000 images in the data set.

For each image, we extract a 36-dimensional low-level feature vector comprising color, shape and texture. For color, we extract mean, variance and skewness in each HSV color channel, and thus obtain 9 color features. For shape, we employ a Canny edge detector and construct an 18-dimensional edge direction histogram for the image. For texture, we apply Discrete Wavelet Transformation (DWT) to graylevel versions of original RGB images. A Daubechies-4 wavelet filter is applied to perform 3-level decomposition, and mean, variance and mode of each of the 3 levels are extracted as a 9-dimensional texture feature.

We compare three state of the art algorithms and a Euclidean distance baseline: ITML, DCA, and our RFD method (with absolute position information). Since retrieval problem is different from classification problem, there is no label for each point, only have labels for pairs such as must-link or cannot-link. But in the training process of LMNN, it needs the class label for each training points. Therefore we did not use the LMNN in the retrieval experiments.

For ITML, we vary the parameter γ from 10^{-4} to 10^4 and choose the best (10^{-3}). For each method, we generate 1% of the available positive constraints and 1% of the available negative constraints (as proposed in [14]). For RFD, we construct a random forest with 1500 trees. Using five-fold cross validation, we retrieve the 20 nearest neighbors of each image under each metric. Accuracy is determined by counting the fraction of the retrieved images that are the same class as the image that retrieved them. We repeat this experiment 10 times with differing random folds and report the average results in Figure 4. RFD clearly outperforms the other methods tested, achieving the best accuracy on all but the cougar category. Also note that ITML performs roughly on par with or worse than the baseline on 7 classes, and DCA on 5, while RFD fails only on 1, indicating again that RFD provides a better global distance measure than current state of the art approaches, and is less likely to sacrifice performance in one region in order to gain it in another.

4. CONCLUSION

In this paper, we have proposed a new angle on the metric learning problem utilizing random forests. Our method, called random forest distance (RFD), incorporates conventional relative position information as well as absolute position of point pairs into the learned metric, and hence implicitly adapts the positon-based metric through the feature space. Our evaluation has demonstrated the capability of RFD, which attains the best overall performance in terms of accuracy and speed on a variety of benchmarks.

There are immediate directions of further inquiry that have been paved with this paper. First, RFD further demonstrates the capability of classification methods as a foundation for metric learning. Similar feature mapping functions and other underlying forms for the distance function need to be investigated. Second, the usefulness of absolute pairwise position information is clear from our work—a good indication of the need for multiple metrics. Open questions remain about other representations for position information as well as the use of position in other metric forms. We are also investigating the use of RFD on larger-scale, more diverse data sets like the MIT SUN image classification data set.

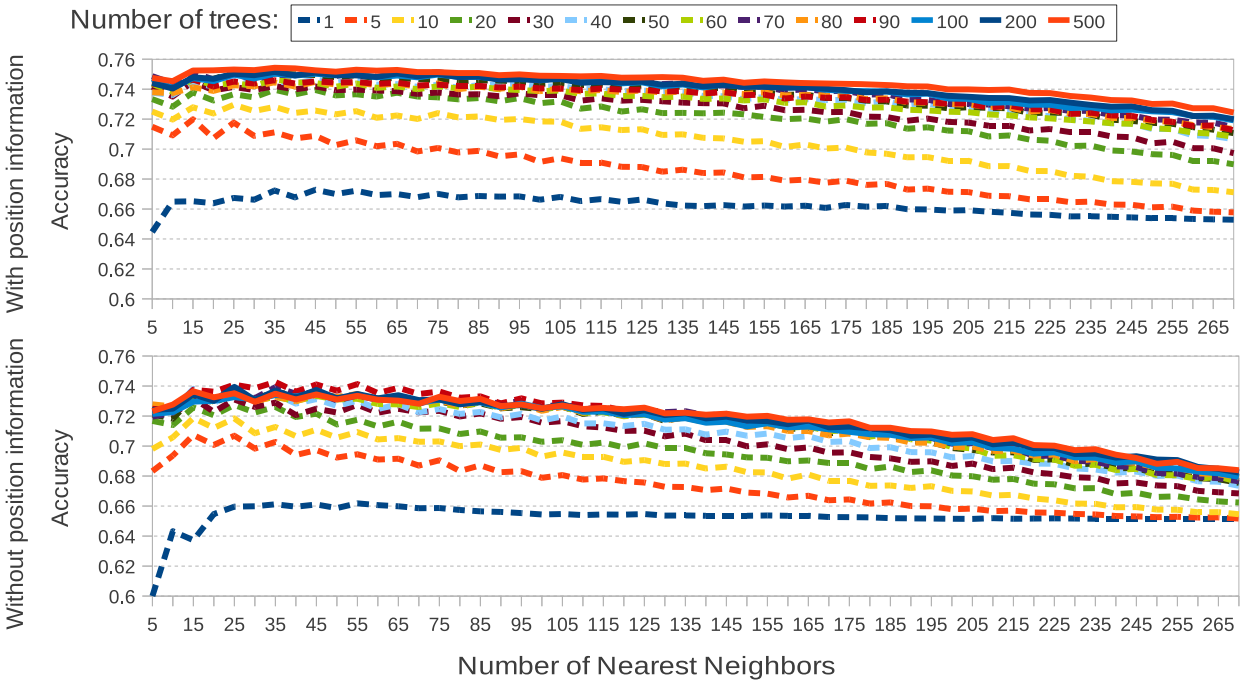

Figure 3: Effect of forest size on RFD performance on the UCI diabetes data set. Results were obtained by averaging results from 10 runs, each using 5-fold cross validation. Both with and without position information, increasing forest size yields notable improvements in accuracy up to about 100 trees. If no position information is included, then additional trees beyond this point provide modest gains at best. With position information, larger forests do appear to allow more fine-tuning, and can produce noticable improvements up to at least 500 trees.

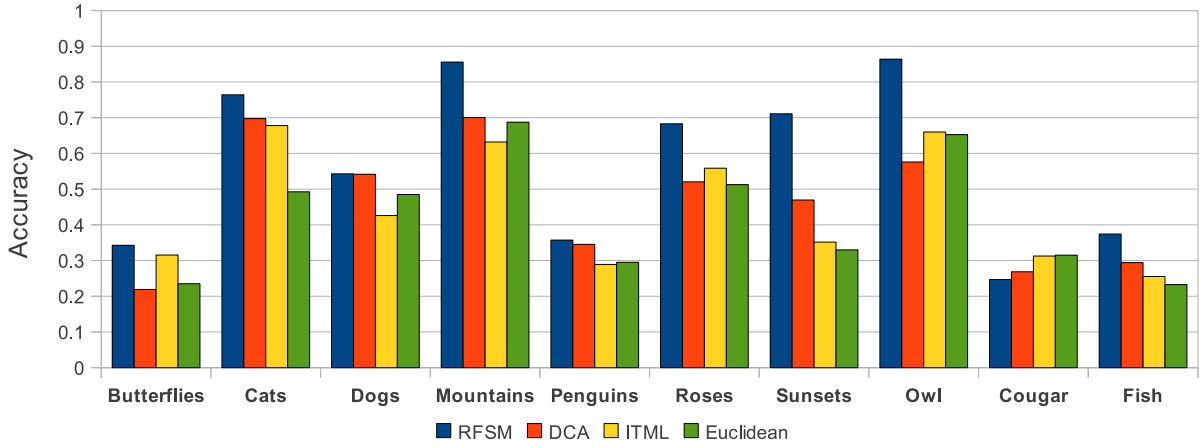

Figure 4: Average retrieval precision on top 20 nearest neighbors of images in the Corel data set. RFD outperforms DCA, ITML and the baseline Euclidean measure on all but one category.

5. 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.

6. REFERENCES

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[2] B. Babenko, S. Branson, and S. Belongie. Similarity

metrics for categorization: from monolithic to category specific. In *International Conference on Computer Vision*, pages 293–300, 2009.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *International Conference on Machine Learning*, volume 20, page 11, 2003.

[4] G. Biau and L. Devroye. On the layered nearest neighbour estimate, the bagged nearest neighbour estimate and the random forest method in regression and classification. *Journal of Multivariate Analysis*, 101(10):2499–2518, 2010.

[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1–8. IEEE, 2008.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] R. Caruana and A. Niculescu-Mizil. An empircal comparison of supervised learning algorithms. In *International Conference on Machine Learning*, 2006.

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition*, volume 1, pages 539–546. IEEE, 2005.

[9] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning*, pages 209–216, 2007.

[10] M. Fink. Object classification from a single example utilizing class relevance metrics. In *Advances in Neural Information Processing Systems*, volume 17, page 449. The MIT Press, 2004.

[11] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. *Advances in Neural Information Processing Systems*, 19:417, 2006.

[12] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[13] A. Globerson and S. Roweis. Metric learning by collapsing classes. *Advances in Neural Information Processing Systems*, 18:451, 2006.

[14] S. Hoi, W. Liu, M. Lyu, and W. Ma. Learning distance metrics with contextual constraints for image retrieval. In *Computer Vision and Pattern Recognition*, volume 2, pages 2072–2078. IEEE, 2006.

[15] P. Jain, B. Kulis, and K. Grauman. Fast image search for learned metrics. *Computer Vision and Pattern Recognition*, 2008.

[16] G. Lebanon. Metric learning for text documents. *Transactions Pattern Analysis and Machine Intelligence*, pages 497–508, 2006.

[17] C. Leistner, A. Saffari, J. Santner, and H. Bischof. Semi-supervised random forests. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 506–513. IEEE.

[18] Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[19] G. Martinez-Munoz, N. Larios, E. Mortensen, W. Zhang, A. Yamamuro, R. Paasch, N. Payet, D. Lytle, L. Shapiro, S. Todorovic, et al. Dictionary-free categorization of very similar objects via stacked evidence trees. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 549–556. IEEE, 2009.

[20] H. Nguyen and L. Bai. Cosine similarity metric learning for face verification. *Asian Conference on Computer Vision*, pages 709–720, 2010.

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

[22] N. Payet and S. Todorovic. (rf) 2 — random forest random field. Advanced in Neural Information Processing Systems, 2010.

[23] S. Shalev-Shwartz, Y. Singer, and A. Ng. Online and batch learning of pseudo-metrics. In *International Conference on Machine Learning*, page 94. ACM, 2004.

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

[25] Y. Shi, Y. Noh, F. Sha, and D. Lee. Learning discriminative metrics via generative models and kernel learning. *Arxiv preprint arXiv:1109.3940*, 2011.

[26] J. Wang, S. Wu, H. Vu, and G. Li. Text document clustering with metric learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 783–784. ACM, 2010.

[27] K. Weinberger and L. Saul. Fast solvers and efficient implementations for distance metric learning. In *International Conference on Machine Learning*, pages 1160–1167. ACM, 2008.

[28] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.

[29] L. Wu, R. Jin, S. Hoi, J. Zhu, and N. Yu. Learning bregman distance functions and its application for semi-supervised clustering. *Advances in Neural Information Processing Systems*, 22:2089–2097, 2009.

[30] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

[31] W. Yang, Y. Wang, and G. Mori. Learning transferable distance functions for human action recognition. *Machine Learning for Vision-Based Motion Analysis*, pages 349–370, 2011.

[32] D. Zhan, M. Li, Y. Li, and Z. Zhou. Learning instance specific distances using metric propagation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1225–1232. ACM, 2009.