A Randomized Algorithm for Learning Mahalanobis Metrics: Application to Classification and Regression of Biological Data

Christopher James Langmead[*†]

July 2005
CMU-CS-05-164

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA 15213.
[†] Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA, USA 15213

E-mail: cjl@cs.cmu.edu

This research is supported by a Young Pioneer Award to C.J.L. from the Pittsburgh Lifesciences Greenhouse.

Keywords: computational biology, metric learning, classification, regression

Abstract

We present a randomized algorithm for semi-supervised learning of Mahalanobis metrics over $\mathbb{R}^n$. The inputs to the algorithm are a set, U, of unlabeled points in $\mathbb{R}^n$, a set of pairs of points, $S = \{(x, y)_i\}; x, y \in U$, that are known to be similar, and a set of pairs of points, $D = \{(x, y)_i\}; x, y \in U$, that are known to be dissimilar. The algorithm randomly samples S, D, and m-dimensional subspaces of $\mathbb{R}^n$ and learns a metric for each subspace. The metric over $\mathbb{R}^n$ is a linear combination of the subspace metrics. The randomization addresses issues of efficiency and overfitting. Extensions of the algorithm to learning non-linear metrics via kernels, and as a pre-processing step for dimensionality reduction are discussed. The new method is demonstrated on a regression problem (structure-based chemical shift prediction) and a classification problem (predicting clinical outcomes for immunomodulatory strategies for treating severe sepsis).

1 Introduction

Many classification, clustering, and regression algorithms depend on a metric over the input space. For example, k-means clustering, k-nearest-neighbor classification/regression, radial basis function networks, and kernel methods, such as SVMs, need to be given good metrics that accurately reflect the important relationships between points. Many common distance metrics, such as the Euclidean metric, assume that each feature is not only independent of the others, but also equally important. Both assumptions are routinely violated in real-world applications. To address this problem, a number of (semi)supervised distance metric learning algorithms have been proposed (e.g., [14, 15]). Here, the user supplies a set of pairs of instances that are known to be similar and/or dissimilar to each other; the distance metric learning algorithm finds a metric that respects these relationships. In contrast to techniques like Multidimensional Scaling [6] and Principal Components Analysis (PCA) [9], which simply find an embedding of the test points, a distance metric learning algorithm learns a true metric, that can be incorporated into *any* learning algorithm that uses metrics, pseudo-metrics, or (dis)similarity measures.

This paper introduces a randomized approach to metric learning. In particular, given any algorithm, λ, for learning a Mahalanobis metric, M, the new method can construct a new metric $\hat{M}$, by calling λ as a subroutine on samples of the training data and subspaces of $\mathbb{R}^n$. The randomization serves two purposes: (1) The computational complexity of metric learning algorithms generally depend on the number of training samples, z, and the dimensionality of the input space, n. The new algorithm builds a metric by combining a constant number of metrics over random subspaces of $\mathbb{R}^n$, each trained on a small sample of the training data. While our algorithm has the same computational complexity as λ, in practice, the decrease in training times is dramatic. Moreover, the subspace metrics can be trained in parallel, yielding another constant factor speed-up. (2) In the language of statistical learning theory (e.g., [8]), random sampling of training instances and features reduces variance and thus guards against overfitting. Our approach to metric learning is influenced by the Random Forest algorithm [2] which uses similar strategies for classification and regression, but is not a technique for learning metrics.

We demonstrate the accuracy and efficiency of the new algorithm for representative problems in biology and medicine. In particular we construct distance metrics over (i) nuclear electronic environments and (ii) clinical data from patients with severe sepsis. We then use these metrics in k-nearest neighbor regression and classification, respectively. Our results demonstrate that the randomized algorithm is not only much faster than the deterministic algorithm (λ), but often produces metrics that improve the accuracy of k-nearest neighbor regression and classification. This indicates that the random algorithm resists overfitting and produces metrics that better generalize to novel instances.

2 Distance Metric Learning

Consider a set of points, $U = \{x_i\}_{i=1}^{l} \subseteq \mathbb{R}^n$, and a distance metric of the form

$$d_M(x_i, x_j) = \|x_i - x_j\| = \sqrt{(x_i - x_j)^T \mathbf{M}(x_i - x_j)}, \tag{1}$$

where **M** is a symmetric positive semi-definite $n \times n$ matrix. If **M** is an identity matrix, then Eq. (1) is simply the un-weighted Euclidean distance. If **M** is merely diagonal, then Eq. (1) is a diagonally weighted Euclidean distance. More generally, **M** parameterizes a family of Mahalanobis distances over $\mathbb{R}^n$ and encodes the weights and correlations among variables.

The distance metric learning problem is to learn **M** given U and a set, $S = \{(x, y)_i\}; x, y \in U$. The set S represents pairs of points in U that a domain expert has asserted are "similar", but has not defined precisely how or why they are similar. Optionally, the domain expert may also provide a set of dissimilar pairs of points, $D = \{(x, y)_i\}; x, y \in U; S \cap D = \{\}$.

There are a number of approaches to learning **M**, given U, S, and, D. Xing *et al* [15] pose the following optimization problem:

$$\hat{M} = \underset{\mathbf{M}}{\operatorname{argmin}} \sum_{(x_i,x_j)\in S} \| x_i - x_j \|^2_{\mathbf{M}} \tag{2}$$

$$s.t. \quad \sum_{(x_i,x_j)\in D} \| x_i - x_j \|_{\mathbf{M}} \geq \alpha \tag{3}$$

$$\mathbf{M} \succeq 0, \tag{4}$$

where α is an arbitrary constant ≥ 1. The constraints are convex and the objective is linear in **M**, thus Eq. 2 can be solved using convex-optimization techniques. Xing *et al* introduce two iterative algorithms solving for $\hat{M}$. The first algorithm deals with the case when **M** is diagonal, the second algorithm solves the problem when **M** is an arbitrary positive semi-definite matrix. Tsang and Kwok [14] pose a different optimization problem (not shown here due to space considerations) and derive a quadratic programming problem using the technique of Lagrangian multipliers. The convex programming formulation of the problem has no user parameters (other than the sizes of S and D), but requires an iterative solution. The quadratic programming formulation of the problem can be solved more efficiently but has several user parameters.

Both approaches learn a metric over the entire input space. This raises two possible concerns: computational complexity and overfitting. The complexity of [15] is $O(I^2n^3)$, where I^2 reflects the nested iterations required to converge, and the n^3 term is the time needed to diagonalize an $n \times n$ matrix; the complexity of [14] is $O(z^3 + n^2)$, where $z = |D|$ and the z^3 term is the time to solve a quadratic programming problem with z variables. The second concern is overfitting; if the sizes of S and D are small, it becomes difficult to estimate the parameters of the metric. Similarly, if the input features are particularly noisy, spurious correlations among features may unduly influence the parameter estimates. Both issues can be addressed using randomization.

3 Algorithm

Our algorithm is presented in Algorithm 1. Briefly, the sets S and D are randomly sampled (with replacement) to construct sets S^b and D^b, where b is the number of samples in the subsets. Next, S^b and D^b are modified using the function `Random-Features`(S^b, D^b, m), where m is the dimensionality of the subspace ($m < n$). The resulting sets, S^b_m and D^b_m, contain points in the same

Algorithm 1 Random Metric Learning Algorithm pseudocode. See text for details.

Input:

U: The unlabeled set of points in $\mathbb{R}^n$
$S = \{(x, y)_i\}; x, y \in U$: a set of pairs of similar points
$D = \{(x, y)_i\}; x, y \in U$: a set of pairs of dissimilar points (*optional*)
λ: an algorithm for learning metrics
c: the number of subspace metrics to learn
b: the number of subspace training samples
m: the dimensionality of the subspace

/ Initialize the metric */*
$\hat{\mathbf{M}}_{n\times n} \leftarrow \mathbf{I}_{n\times n}$: an $n \times n$ identity matrix

for $i = 1$ to c **do**
  / Select random training instances */*
  $S^b \leftarrow$ `Random-Sample(`S, b`)`
  $D^b \leftarrow$ `Random-Sample(`D, b`)`

  / Select a random subspace */*
  $S^b_m, D^b_m, f \leftarrow$ `Random-Feature(`S^b, D^b, m`)`

  / Learn the subspace metric */*
  $\mathbf{M}^f_{m\times m} \leftarrow \lambda(S^b_m, D^b_m)$

  / Update the complete metric */*
  $\hat{\mathbf{M}}_{n\times n} \leftarrow$ `update(`$\hat{\mathbf{M}}_{n\times n}, \mathbf{M}^f_{m\times m}, f$`)`
end for
return $\hat{\mathbf{M}}$

random m-dimensional subspace of $\mathbb{R}^n$. The function `Random-Features` also returns a vector, **f**, whose components encode which dimensions were sampled. S^b_m and D^b_m comprise a subspace training set. This training set is passed to the given metric-learning algorithm, λ, which returns a metric, $\mathbf{M}^f$, over the random subspace. $\mathbf{M}^f$ is, by definition, an $m \times m$ symmetric positive semi-definite matrix. The choice of λ dictates the properties of the subspace metric. For example, the algorithms presented in [15] and [14] are guaranteed to converge to an optimal solution for a given training set.

The next step is to update the matrix encoding the metric over $\mathbb{R}^n$. The matrix $\hat{\mathbf{M}}$ is initialized as a $n \times n$ identity matrix. Thus, $\hat{\mathbf{M}}$ is also a symmetric positive semi-definite matrix. Recall that the components of vector **f** indicate which dimensions of $\mathbb{R}^n$ were sampled. That is, the components of **f** correspond to rows and columns of **M**. Consider the auxiliary $n \times n$ matrix, $A : A(\mathbf{f}(i), \mathbf{f}(j)) = \mathbf{M}^f(i, j); \forall\ i, j \leq m$. The function `update(`$\hat{\mathbf{M}}, \mathbf{M}^f, \mathbf{f}$`)` returns the matrix $\hat{\mathbf{M}} + A$. This whole procedure is repeated c times, with different random samples of the training

data and features. It can be shown that $\hat{\mathbf{M}} + A$ is a symmetric positive semi-definite matrix. That is, the symmetric, positive semi-definiteness of $\hat{\mathbf{M}}$ is an invariant property of the algorithm. Thus, $\hat{\mathbf{M}}$ satisfies all the properties of a metric.

The computational complexity of the algorithm is a function of the size of U, the parameters c, b, and m, and the complexity of the underlying metric learning algorithm, λ. Let $z = |U|$. The parameters c and b are generally set so that the product $cb = O(z)$. The parameter m is generally set so that the product $cm = O(n)$. Let $g_\lambda(b, m)$ be the computational complexity of λ on b instances of m-dimensional data. Note that $g_\lambda(b, m)$ varies by the choice of learning algorithm but is trivially lower-bounded by $\Omega(max(b^2, m^2))$ since the $m \times m$ matrix $\mathbf{M}$ must be explicitly realized and b^2 pairwise similarities must be computed. The complexity of our algorithm is $O(cn^2 + cg_\lambda(b, m))$. In practice, we set $b \approx z/10$, $m \approx n/2$, and $c \geq 10$. Thus, c is a constant and since $b = O(z)$ and $m = O(n)$, our algorithm has the same complexity as λ which, effectively, has $c = 1$, $b = z$ and $m = n$. However, in practice, the randomized algorithm is much faster because b and m are significantly smaller than z and n, respectively. Finally, note that each subspace metric can be trained in parallel giving a constant factor speedup for p processors (i.e., $O(cn^2 + \lceil \frac{c}{p} \rceil g_\lambda(b, m))$).

An important feature of the algorithm is that the c subspace metrics are trained on different samples and different features. Each subspace metric is optimal for that subspace and the data used to construct it. The global metric is *not* optimal for U, despite being built from optimal subspace metrics. However, the random algorithm can be thought of as an *ensemble* method for metric learning. Ensemble methods in machine learning are well studied and have a number of desirable properties (see, e.g., [7]). In particular, ensemble methods often generalize better to novel instances than non-ensemble methods. Briefly, if the members of the ensemble are diverse, that is, not highly correlated in their erroneous predictions, then it becomes very unlikely that two "bad" predictors can conspire to affect the outcome. Conversely, two or more "good" predictors have a better chance of affecting the outcome. In our algorithm, diversity is enforced by sampling both the training instances and the features. We present empirical evidence in Sec. 5 that demonstrates that the randomized algorithm always produces metrics that are as good as, and often better than, the metrics produced using deterministic algorithms.

4 Experiments

We tested our metric learning algorithm in the contexts of regression and classification problems. The data for each experiment were split into randomly selected, non-overlapping training and test sets. The training set was used to learn the metric over the feature space. That metric was then used to compute the distance of each test instance to the training instances. The predicted value (or class) for each test instance was then computed using k-nearest neighbor regression (or classification). A variety of parameter settings (i.e., b, c, m, z) were tested. The entire procedure was repeated 20 times to obtain error estimates for a given configuration of parameters.

The base metric learning algorithm (i.e., λ) in our experiments was the technique of Xing *et al* [15]. We will refer to our algorithm as λ_r for the purpose of comparison with λ. Both algorithms were implemented in Matlab and the experiments were conducted on a 3 GHz Pentium 4 workstation running Linux. We did not use the parallel training option for λ_r. Finally, λ_r and λ

were trained and tested on the same training and test sets, respectively, to ensure a fair comparison.

4.1 Chemical Shift Prediction

The first study comprised 5 separate experiments for predicting the real-valued chemical shifts for backbone nuclei (H^N, H^α, $^{13}C^\alpha$, $^{13}C'$, ^{15}N) given a structural model of the protein.

4.1.1 Background

Chemical shifts are the most fundamental of spectral parameters in Nuclear Magnetic Resonance (NMR) spectroscopy. Briefly, a spin $I = 1/2$ nucleus in an external magnetic field will have two spin states whose energy difference is linearly proportional to the strength of the applied magnetic field. Each nucleus, however, is influenced by the electrons in its vicinity. Relevant factors include covalent bonding, the orientations of nearby aromatic rings, hydrogen bonding, solvent interactions, and ionization constants. Thus, each nucleus feels a slightly different field because it has a different local electronic environment. The experimentally measured chemical shift for a given nucleus is proportional to the field felt by that nucleus.

Chemical shifts can most accurately be predicted from structure models using quantum mechanics. A purely quantum mechanical approach to chemical shift prediction involves determining the wave function of the molecule; this is computationally intractable for large molecules, like proteins. Consequently, hybrid methods for chemical shift prediction are used in practice; these methods combine quantum calculations for local interactions and either classical or empirical calculations for long-range interactions. There are a number of practical and important applications of structure-based chemical shift prediction including: resonance assignment (e.g. [10]), structure determination and model refinement (e.g., [3, 5]), high-throughput fold recognition (e.g. [11, 12]), as well as a variety of assays for probing mutations and structure-activity relationships.

4.1.2 Features and Training/Test Data

The regression model for these experiments included variables (i.e., features) that represent quantum and empirical factors. These contributions were computed in the manner of [16] and include the effects of torsion angles, residue type, ring-currents, electrostatics, and hydrogen bonding. The experimental data for these experiments were obtained from the REFDB [17] database. REFDB is a carefully re-referenced subset of the BioMagResBank (BMRB) [13] that corrects some systematic errors within the BMRB. The REFDB also includes a mapping to specific Protein Data Bank [1] (PDB) IDs; these structures were downloaded from the PDB and used to compute the quantum and empirical features.

The resulting data set contained between 20,000 and 47,000 instances, depending on nucleus type, across 454 different proteins[1]. Each instance comprised an experimentally measured chemical shift (re-referenced, as needed), and a feature vector containing the quantum and empirical

[1]RefDB actually contains data from 601 proteins. We used a subset of proteins that did not include protein complexes.

calculations. The various nuclei types are sensitive to different affects, thus the specific feature-vector size varied per nucleus type; the nuclei H^N, H^α, $^{13}C^\alpha$, ^{13}C', and ^{15}N had feature vectors of size 5, 7, 6, 7, and 9, respectively. The goal is to learn a metric over these features.

4.2 Patient Outcome

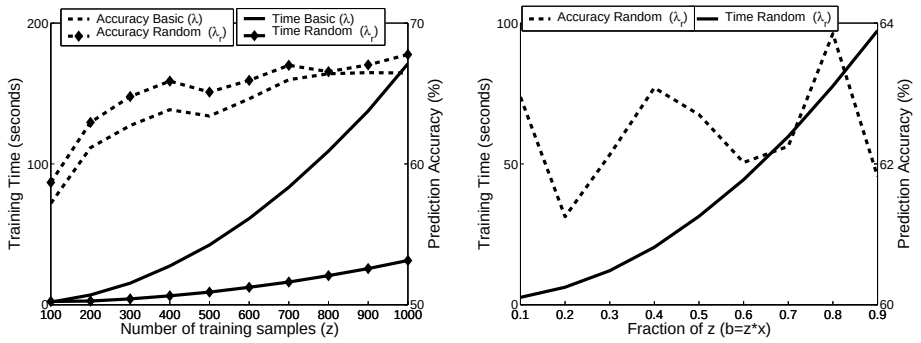
In a different experiment, we examined the performance of our algorithm in a classification task. In particular, we used a data set created by Clermont *et al* [4] for *in silico* modeling of immunomodulatory interventions for severe sepsis. The goal here is to predict an outcome for each of 1000 simulated patients. The four possible outcomes are (i) helped by intervention; (ii) harmed by intervention; (iii) lives regardless of intervention; (iv) dies regardless of intervention. The input space consists of 26 features that include the levels, ratios, and products of a number of biomarkers at the time of the disease detection and one hour following detection. The Clermont data set is divided into separate training and test sets. The partitions between these two sets were maintained during our experiments.

5 Results

We first compared the rates of increase in training times for the random and deterministic algorithms. Figure 1(a) plots the accuracy and the number of seconds required to train λ and λ_r for $z = 100, 200, ..., 1000$ samples on the patient outcome data set. The test set consisted of 200 randomly selected instances and the classifications were made with $k = 1$ nearest neighbor classification. Similar results are obtained for the other data sets. The parameters for λ_r were $c = 20, b = z/10$, and $m = n/2$. Recall that the parameters for λ are, effectively, $c = 1, b = z$, and $m = n$. The rates of increase in training time are dramatically different, as expected. The accuracies under both metric increase with the size of z. We note that the accuracies under the λ_r metric are either statistically identical or *higher* than those under the λ metric. That is, λ_r provides the same level of accuracy, or better, than λ while drastically reducing training times. The accuracies under the λ_r metric were statistically higher than those under the λ metric for $z = 300, 400, 500, 1000$. These results are consistent with the notion that randomization may be an effective means for resisting overfitting a given training set.

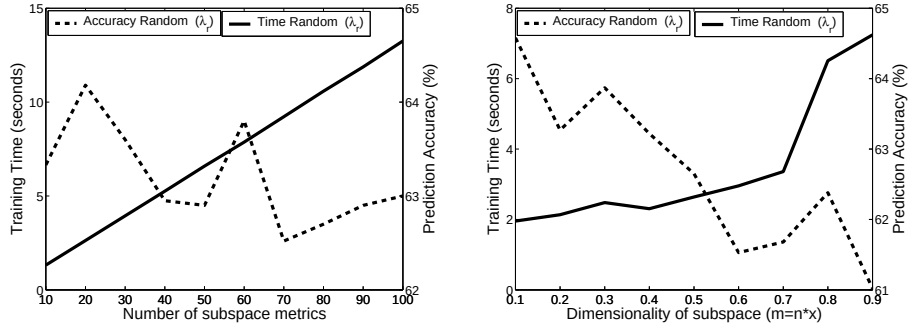
Figures 1(a), 2(a), and 2(b) plot the accuracy and training times for λ_r various values of b, c, and m, respectively, while fixing the remaining variables ($z = 200, c = 20, b = z/10$, and $m = n/2$). Training times increase dramatically with increasing b, but there is no statistical difference in prediction accuracy between $b = z \times 0.1$ and $b = z \times 0.9$, suggesting that small values of b are sufficient. Similarly, training times increase linearly with increasing c, but there is no statistical difference in accuracy between $c = 10$ and $c = 100$. Training times rise with increasing m, but not as quickly as for increasing z, c, or b. Interestingly, there is a statistically significant *decrease* in accuracy ($p < 0.001$) as m increases. This result further supports the hypothesis that randomization can lead to better metrics.

Finally, we evaluated the accuracy of the random algorithm on larger test sets. In these experiments the parameters of the random algorithm were $z = 1000, c = 20, b = z/10$, and $m = n/2$.



(a) (b)

Figure 1: Left: Training times (solid lines) and accuracies (dashed lines) for λ and λ_r for varying z. Right: Training times (solid line) and accuracies (dashed line) for λ_r for varying b. Note that left- and right hand axis scales are not the same in the two figures.



(a) (b)

Figure 2: Left: Training times (solid line) and accuracies (dashed line) for λ_r for varying c. Right: Training times (solid line) and accuracies (dashed line) for λ_r for varying m. Note that left- and right hand axis scales are not the same in the two figures.

Using the complete training and test sets for the patient outcome data (1,000 instances each), the classification accuracy for $k = 1$ nearest neighbor was 73% using the λ_r metric versus 68% for using the λ metric. These results are in agreement with the smaller test cases, described above.

				SHIFTS	λ Metric	λ_r Metric
Nucleus	Instances	Mean Shift	Range	RMSE	RMSE	RMSE
H^N	47,401	8.29	5.44	0.647	0.611	**0.607**
H^α	39,009	4.41	4.19	1.13	0.340	**0.336**
^{15}N	34,196	119.62	35.18	5.16	3.87	**3.87**
$^{13}C^\alpha$	30,174	58.10	26.46	1.74	1.67	**1.67**
^{13}C'	19,877	176.15	15.40	1.88	1.54	**1.53**

Table 1: **Chemical Shift Prediction:** Accuracy and summary of training and test data for 5 different nuclei. Column 2: the total number of instances for a given nucleus; 1,000 randomly selected instances were selected for training, the remaining were used to test the learned metric using $k = 5$ nearest neighbor regression. Columns 3-4, the mean shift value, and range (in ppm). Columns 5-7, the root mean squared error (in ppm) for the program SHIFTS and k-NN under the λ metric and under the λ_r metric, respectively. Errors are estimated based 20 random partitions into training and test sets.

However, Clermont *et al* were able to obtain an 84% percent accuracy on the same training/test data using logistic regression [4]. This suggests that a linear metric may not be optimal for these data.

The test sets for the chemical shift predictions contained between 19,000 to 46,000 instances, depending on nucleus type. Accuracies are reported in terms of root mean squared error (RMSE) in *parts per million* (ppm), the standard units of chemical shifts. The RMSEs for the chemical shifts vary widely by nucleus type. This is due to the fact that the average magnitude and range of different nuclei varies widely. For example the H^α nucleus has an average chemical shift of $\approx$ 4.4 ppm and a range of $\approx$ 4.2 ppm; the ^{15}N nucleus, on the other hand, has an average chemical shift of $\approx$ 119.6 ppm and a range of $\approx$ 35.2 ppm. Table 5 summarizes the training sets and reports the RMSE for the program SHIFTS [16], and for $k = 5$ nearest neighbor regression under the λ and λ_r metrics. The predictions errors under the λ_r metric are statistically significantly lower than those of the SHIFTS program for all nuclei ($p \ll 0.01$). The predictions errors under the λ_r metric are statistically lower than the λ metric for ^{13}C', H^α, and H^N ($p < 0.01$); the errors for $^{13}C^\alpha$ and H^α are statistically identical under both metrics.

6 Discussion

The primary finding of these experiments is that the randomized algorithm produces metrics that are equivalent, and sometimes better, than those produced by a deterministic metric learning algorithm. The randomized algorithm, however, is much faster because the subspace metrics can be trained on smaller sets of data. Indeed, the size of the training set (U) has the greatest effect on both training time and accuracy. An interesting finding is that accuracy drops when m, the dimensionality of the subspace, increases. This suggests that the sampling of random subspaces may guard against overfitting.

There are a number of extensions to this work that are worthy of investigation. Our results

demonstrate that the new algorithm constructs good linear metrics for the chemical shift data, but that the linear metrics over the sepsis data produced by both λ and λ_r are not competitive with a logistic regression over the same variables. This suggests that a linear metric may not be optimal for these data. Xing *et al* have noted that it is also possible to consider non-linear metrics by introducing a non-linear feature map, ϕ,

$$d_M(x, y) = \sqrt{(\phi(x) - \phi(y))^T \mathbf{M}(\phi(x) - \phi(y))}. \tag{5}$$

This can also be thought of as learning a metric in a feature space associated with a kernel function $k(x_i, x_j) = \phi(x_i)\mathbf{M}\phi(x_j)$ [14]. The exploration of non-linear metrics is an interesting area for future research.

Another interesting area of investigation is the use of dimensionality reduction schemes. Standard techniques for dimensionality reduction (e.g., via PCA) can be applied either before or after the metric is learned. Dimensionality reduction prior to learning the metric will reduce training times, as demonstrated above. However, applying dimensionality reduction after learning the metric is also worthy of consideration. Note that because the matrix $\mathbf{M}$ is positive semi-definite, we can write $\mathbf{M}$ as $\mathbf{A}\mathbf{A}^T$. The matrix $\mathbf{A}$ can be thought of as a projection into a different feature space. The potential advantage to doing the dimensionality reduction in the feature space is that the user implicitly defines the kinds of relationships that are important by constructing S and D. For some problems, there may be several kinds of relationships that are of interest. For example, the patient data defined "similar" in terms of mortality and the response to the intervention. But there may also be other kinds of similarity that may be of interest, such as comorbidity. Here, the expert can define a different S and D that reflect the similarity of interest; the same training data can be used, but a different metric will be learned. This additional flexibility is an attractive feature of any metric learning algorithm.

7 Conclusion

We have introduced a randomized algorithm for learning metrics over an input space. The new algorithm compares favorably with deterministic metric learning algorithms in terms of accuracy, generalization, and performance. Training times are drastically reduced due to the sampling of the training data and subspaces of $\mathbb{R}^n$. At the same time, the metrics produced by the random algorithm are equivalent to, and sometimes better than, the metrics produced by deterministic metric learning algorithms.

We have demonstrated the effectiveness of these metrics on diverse data sets and in both classification and regression contexts. Our metric over electronic environments resulted in significantly lower root mean squared errors than those of the program SHIFTS, which does not use a metric. Our results on a classification task were somewhat mixed. On the one hand, our metric outperformed a metric produced by a deterministic algorithm. Still, the prediction accuracies were not as high as those obtained using logistic regression. We believe that an investigation into non-linear metrics may address this issue.

Acknowledgments

We would like to thank Dr. Gilles Clermont for use of his data and Dr. Eric Xing for use of his code for his metric learning algorithm.

References

[1] H.M Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucl. Acids Res.*, 28:235–242, 2000.

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

[3] D.A. Case, T.A. Darden, T.E. Cheatham III, C.L. Simmerling, J. Wang, R.E. Duke, R. Luo, K.M. Merz, B. Wang, D.A. Pearlman, M. Crowley, S. Brozell, V. Tsui, H. Gohlke, J. Mongan, V. Hornak, G. Cui, P. Beroza, C. Schafmeister, J.W. Caldwell, W.S. Ross, and P.A. Kollman. AMBER 8. University of California, San Francisco, 2004.

[4] G. Clermont, J. Bartels, R. Kumar, G. Constantine, Y. Vodovotz, and C. Chow. In silico design of clinical trials: A method coming of age. *Crit. Care Med.*, 32(10):2061–2070, 2004.

[5] G. Cornilescu, F. Delaglio, and A. Bax. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *J Biomol NMR*, 13(3):289–302, 1999.

[6] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, 1994.

[7] T.G. Dietterich. Ensemble methods in machine learning. *Proc. of the First International conference on Multiple Classifier Systems, Lecture Notes in Computer Science*, pages 1–15, 2000.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[9] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1989.

[10] C. J. Langmead and B. R. Donald. An Expectation/Maximization Nuclear Vector Replacement Algorithm for Automated NMR Resonance Assignments. *J. Biomol. NMR.*, 29(2):111–138, 2004.

[11] C. J. Langmead and B. R. Donald. High-Throughput 3D Homology Detection via NMR Resonance Assignment. *Proc. IEEE Computer Society Bioinformatics Conference (CSB), Stanford University, Palo Alto, CA*, pages 278–289, 2004.

[12] S.P. Mielke and V.V. Krishnan. Protein structural class identification directly from NMR spectra using averaged chemical shifts. *Bioinformatics*, 19(16):2054–64, 2003.

[13] B.R. Seavey, E.A. Farr, W.M. Westler, and J.L. Markley. A Relational Database for Sequence-Specific Protein NMR Data. *J. Biom. NMR*, 1:217–236, 1991.

[14] I. W. Tsang and J.T. Kwok. Distance Metric Learning with Kernels. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 126–129, Cambridge, MA, June 2003.

[15] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance Metric Learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, Istanbul, Turkey, 2002. MIT Press.

[16] X.P Xu and D.A. Case. Automated prediction of 15N, 13C‘alpha‘, 13C‘beta‘ and 13C’ chemical shifts in proteins using a density functional database. *J. Biomol. NMR*, 21:321–333, 2001.

[17] H. Zhang, S. Neal, and D.S. Wishart. A Database of Uniformly Referenced Protein Chemical Shifts. *J. Biomol. NMR*, 25(3):173–195, 2003.