Reconstructing Big Semantic Similarity Networks

Ai He, Shefali Sharma
*Information Sciences Institute
University of Southern California
4676 Adminralty Way
Marina del Rey, CA 90292
{aihe|sharma}@isi.edu

Chun-Nan Hsu*,†
†Division of Biomedical Informatics
Department of Medicine
University of California, San Diego
La Jolla, CA 92093
chunnan@ucsd.edu

Abstract

Distance metric learning from high (thousands or more) dimensional data with hundreds or thousands of classes is intractable but in NLP and IR, high dimensionality is usually required to represent data points, such as in modeling semantic similarity. This paper presents algorithms to scale up learning of a Mahalanobis distance metric from a large data graph in a high dimensional space. Our novel contributions include random projection that reduces dimensionality and a new objective function that regularizes intra-class and inter-class distances to handle a large number of classes. We show that the new objective function is convex and can be efficiently optimized by a stochastic-batch subgradient descent method. We applied our algorithm to two different domains; semantic similarity of documents collected from the Web, and phenotype descriptions in genomic data. Experiments show that our algorithm can handle the high-dimensional big data and outperform competing approximations in both domains.

1 Introduction

According to Yang (2006), distance metric learning learns a distance metric from data sets that consists of pairs of points of the same or different classes while at the same time preserving the adjacency relations among the data points. Usually, it is easier to let the user label whether a set of data is in the same class than directly assign a distance between each pair or classify whether a pair of data points is a match or not. Learning a good distance metric in the feature space is essential in many real-world NLP and IR applications. For example, Web news article clustering applying hierarchical clustering or k-means requires that the distance between the two feature vectors extracted from the news articles faithfully reflect the semantic similarity between them for these algorithms to perform well.

Studies on distance metric learning over the past few years show that the learned metric can outperform Euclidean distance metric. The constraints of training examples in learning usually comes from the global or local adjacency information. Data points with the same class labels are supposed to be connected while those with different classes labels disconnected. Supervised algorithms aim to learn the distance metric to make the adjacency relationships in the training examples preserved.

One of the most common approaches to distance metric learning is to learn a Mahalanobis distance metric. Example algorithms to learn a Mahalanobis distance metric include (Xing et al., 2002; Goldberger et al., 2004; Shaw et al., 2011).

A common limitation shared by these algorithms is that they fail to scale up to high dimensional data sets. When those algorithms run on high dimensional data sets, they usually run out of memory. However, many NLP applications depend on tens of thousands of features to perform well. Dimensionality reduction and approximation have been suggested, but they usually degrade performance. Other issues occur when the data sets consists of a large number of disjoint classes. In this case, the learned distance metric must map the data points to a space where the data points cluster unevenly into a large

Proceedings of the TextGraphs-8 Workshop, pages 20–28,
Seattle, Washington, USA, 18 October 2013.

number of small groups, which makes the learning problem harder and may require special treatments.

In this paper, we present a new scalable approach to distance metric learning that addresses the scalability issues mentioned above. To deal with high dimensionality, one approach is to factorize the metric matrix in the Mahalanobis distance metric into low-rank matrix. This reduces the number of parameters that must be learned during the learning phase. However, the learning problem becomes non-convex. Different initializations may result in drastically different performance due to local optima. We solve this problem by introducing random projection, which projects data points to a low dimensional space before learning a low dimensional full-rank metric matrix. We show that this strategy not only is more robust than the low-rank approximation, but also outperforms the Principal Component Analysis (PCA), a common approach to dimensionality reduction.

Another contribution that our approach offers is new regularization terms in the objective function of learning. The new terms specify that one should learn to minimize the distance for data points in the same classes and maximize those in different ones. We found that minimization but not maximization would lead to the best performance, so we kept only the minimization term.

We evaluated our new approach with data sets from two problem domains. One domain is about learning semantic similarity between Web pages. This domain was studied in (Shaw et al., 2011) and involves moderately high dimensional data sets of bag-of-words. The other is about matching semantically related phenotype variables across different genome-wide association studies (GWAS) (Hsu et al., 2011). This problem domain requires extremely high dimensional data for a learner to perform well. Our experimental results show that our new algorithm consistently outperform the previous ones in both the domains.

2 Distance Metric Learning

Let $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the feature matrix of input data points. For any two data points $x_i,\ x_j \in \mathbb{R}^{d \times 1}$ in $\mathbf{X}$, the (square of) the Mahalanobis distance between x_i and x_j is defined as

$$D_M^{i,j} = (x_i - x_j)^\intercal \mathbf{M} (x_i - x_j),$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a metric matrix. The distance is always non-negative because $\mathbf{M}$ is required to be positive semidefinite (PSD).

Xing (2002) used semidefinite programming to learn a Mahalanobis distance metric for clustering. It was a convex optimization problem, which allowed them to derive local optima free algorithms. Weinberger (2005) learned a Mahalanobis distance metric for the k-nearest neighbor classifier by maintaining a margin between data points in different classes, *i.e.*, enforcing the neighbors of the same class to be closer than all others. As in the support vector machines, the learning problem was reduced to convex optimization based on hinge loss. Yang (2006) presented a comprehensive survey on distance metric learning.

Recently, Shaw *et al.* (2011) followed the preceding approaches but reformulated the problem, as an instance of the on-line learning algorithm PEGASOS (Shalev-Shwartz et al., 2011), albeit a complex construction. In a way, it scaled up the traditional metric learning method to a larger amount of data points. They also reformulated the margin mentioned above as triplets over the data set and clarify the derivation of the objective function. Each training example used here is a triplet (x_i, x_j, x_k), consisting of a pair x_i and x_j in the same class and x_k that is in a different class. Learning in this case ensures that the learned distance between x_i and x_j is less than their distance to x_k. In comparison, one may formulate the distance learning problem as binary classification, where the objective is to minimize the match probability of data point pairs in different classes and maximize those in the same ones. Since there will always be much more data point pairs in different classes than those in the same ones, this formulation always lead to an unbalanced classification problem.

3 Strategies for Scaling Up Learning

Shaw *et al.* (2011) suggested various strategies to scale up the algorithm for high dimensional multi-class data. In this section, we will review these strategies and their weaknesses, and then, present our approach that addresses these weaknesses.

3.1 Dimensional Reduction

Four strategies were suggested to handle the high dimensional data sets:

- Diagonal Approximation,
- Principal Component Analysis (PCA),
- Low Rank Decomposition,
- Random Projection.

Diagonal approximation requires the metric matrix $\mathbf{M}$ to be diagonal, thus it consists of only d parameters to learn, instead of $d \times d$. It indeed shrinks the number of parameters to learn, but also ignores the feature-feature interaction and might harm the expressiveness of the model.

Dimensionality reduction using PCA as a preprocessing step is a common way to deal with high dimensional data. However, it does not always work satisfactorily, especially when the data set does not have an apparent intrinsic low dimensional structure. It usually fails to perform well for those data sets.

Shaw *et al.* (2011) suggested a low-rank decomposition to scale up distance metric learning. The idea is to decompose $\mathbf{M}$ into $\mathbf{L}^\intercal\mathbf{L}$, where $\mathbf{L}$ is a $r \times d$ matrix and $r \ll d$ is a predefined low-rank degree. This approach reduces computational cost but results in a non-convex problem that suffers from local minima. Previously, low-rank decomposition has been proposed for other machine learning problems. For example, Rennie *et al.* (2005) applied it to scale up Maximum Margin Matrix Factorization (MMMF) (Srebro et al., 2004). Originally, MMMF was formulated as a convex semi-definite programming (SDP) problem and solved by a standard SDP solver, but it is no longer applicable for the non-convex low-rank decomposition. Therefore, they solved the non-convex problem by Conjugate Gradient (CG) (Fletcher and Reeves, 1964), but still there is no guarantee that CG will converge at a global minimum.

Our choice is to use *random projection* $\mathbf{L}^\intercal\mathbf{RL}$, where $\mathbf{L}$ is a random $r \times d$ ($r \ll d$) matrix, with all of its element in $(0, 1)$. Random projection theory has been developed by Johnson and Lindenstrauss(1984). The theory shows that a set of n points in a high dimensional (d) Euclidean space can be mapped into a low-dimensional ($r \ll d$) Euclidean space such that the distance between any two points will be well-persevered (*i.e.*, changes by only a tiny factor ϵ if r is greater than a function of n and ϵ). Let $\mathbf{R}$ be a $r \times r$ PSD matrix to be learned from data. Distance between x_i and x_j becomes

$$D_R^{i,j} = (x_i - x_j)^\intercal\mathbf{L}^\intercal\mathbf{RL}(x_i - x_j).$$

There are two possible strategies to generate a random projection matrix $\mathbf{L}$. One is completely random projection, where all elements are generated independently; the other one is orthonormal random projection, which requires that $\mathbf{L}_{r\times d}$ be a matrix with r orthonormal random column vectors. The Gram-Schmidt process (Golub and Van Loan, 1996) generates such a matrix, in which one starts by generating a random column and then the next columns, though generated randomly, too, must be orthonormal with regard to other columns. In both strategies, all of the elements must be in $(0, 1)$.

Consider $\mathbf{L}$ to be a matrix which compresses x_i with dimension d by $v_i = \mathbf{L}x_i$ with dimension r. Hence,

$$D_R^{i,j} = (v_i - v_j)^\intercal\mathbf{R}(v_i - v_j) \tag{1}$$

This distance metric can be learned by searching for $\mathbf{R}$ that minimizes the following objective function:

$$\mathcal{F}(\mathbf{R}) = \frac{1}{|S|}\sum_{(i,j,k)\in S} Z_+(D_R^{i,j} - D_R^{i,k} + \xi), \tag{2}$$

where $Z_+(x)$ is the hinge loss function. It can be shown that this new objective function is convex. As in Shaw *et al.*, the training examples are given as a hinge loss function over triplets.

$$S = \{(i, j, k)|\mathbf{A}_{ij} = 1, \mathbf{A}_{ik} = 0\}$$

is the set of all triplets where $\mathbf{A}$ is the adjacency matrix of $\mathbf{X}$. ξ is a predefined constant margin. The hinge loss function will penalize a candidate $\mathbf{R}$ when the resulting distance between x_i and x_j plus the margin is greater than the distance between x_i and x_k.

3.2 Intra and Inter-Class Distance

One challenge in distance metric learning is dealing with data sets that can be clustered into a large

number of distinct classes. The hinge loss term in Eq. (2) does not consider this because it only keeps a margin of data points in one class against points in other classes. In our approach, we would like to learn a distance metric such that the entire set data points in the same class are close to each other and away from those in other classes. This idea is realized by adding additional regularization terms to the objective function. These new regularization terms are proved to be useful to handle problem domains where data points form a large number of mutually-exclusive classes.

The formula for intra-class distance regularization is given as

$$\mathcal{I}_1(\mathbf{R}) = \frac{1}{|S_1|} \sum_{(i,j)\in S_1} D_R^{i,j},\ S_1 = \{(i,j)|\mathbf{A}_{ij} = 1\}, \tag{3}$$

while the formulation for inter-class distance regularization is

$$\mathcal{I}_2(\mathbf{R}) = \frac{1}{|S_2|} \sum_{(i,k)\in S_2} D_R^{i,k},\ S_2 = \{(i,k)|\mathbf{A}_{ik} = 0\}. \tag{4}$$

3.3 Algorithm

Combining the regularization, the hinge loss, intra-class and the inter-class item, we get the complete objective function as Eq. (5).

$$\begin{aligned}\mathcal{F}(\mathbf{R}) = \frac{\lambda}{2}\|\mathbf{R}\|_F^2 + \\ \frac{1}{|S|}\sum_{(i,j,k)\in S} Z_+(D_R^{i,j} - D_R^{i,k} + \xi) \\ + \frac{1}{|S_1|}\sum_{(i,j)\in S_1} D_R^{i,j} - \frac{1}{|S_2|}\sum_{(i,k)\in S_2} D_R^{i,k}\end{aligned} \tag{5}$$

It is also possible to assign weights to terms above. According to Vandenberghe (1996), Eq. (5) can be expressed as a convex semidefinite programming problem. By construction, Eq. (5) can also be reformulated as an instance of PEGASOS algorithm, which basically employs a sub-gradient descent algorithm to optimize with a stochastic batch selection, and a smart step size selection. The sub-gradient of $\mathcal{F}$ in terms of $\mathbf{R}$ is then:

$$\begin{aligned}\nabla\mathcal{F} = \lambda\mathbf{R} + \frac{1}{|S^+|}\sum_{(i,j,k)\in S^+} \mathbf{L}\mathbf{X}\mathbf{C}^{(i,j,k)}\mathbf{X}^\intercal\mathbf{L}^\intercal \\ + \frac{1}{|S_1|}\sum_{(i,j)\in S)1} \mathbf{L}\mathbf{X}\mathbf{C_1}^{(i,j)}\mathbf{X}^\intercal\mathbf{L}^\intercal \\ - \frac{1}{|S_2|}\sum_{(i,k)\in S_2} \mathbf{L}\mathbf{X}\mathbf{C_2}^{(i,k)}\mathbf{X}^\intercal\mathbf{L}^\intercal, \\ S^+ = \{(i,j,k)|D_R^{i,j} + \xi - D_R^{i,k} > 0\}\end{aligned} \tag{6}$$

Here the sparse symmetric matrix $\mathbf{C}$ is defined such that

$$\begin{aligned}\mathbf{C}_{jj}^{(i,j,k)} = \mathbf{C}_{ik}^{(i,j,k)} = \mathbf{C}_{ki}^{(i,j,k)} = 1, \\ \mathbf{C}_{ij}^{(i,j,k)} = \mathbf{C}_{ji}^{(i,j,k)} = \mathbf{C}_{kk}^{(i,j,k)} = -1,\end{aligned}$$

and zero elsewhere. Similarly, $\mathbf{C_1}$ is given by

$$\begin{aligned}\mathbf{C_1}_{ii}^{(i,j)} = \mathbf{C_1}_{jj}^{(i,j)} = 1, \\ \mathbf{C_1}_{ij}^{(i,j)} = \mathbf{C_1}_{ji}^{(i,j)} = -1,\end{aligned}$$

and zero elsewhere. $\mathbf{C_2}$ is

$$\begin{aligned}\mathbf{C_2}_{ii}^{(i,k)} = \mathbf{C_2}_{kk}^{(i,k)} = 1, \\ \mathbf{C_2}_{ik}^{(i,k)} = \mathbf{C_2}_{ki}^{(i,k)} = -1,\end{aligned}$$

and zero elsewhere. It is easy to verify that

$$tr(\mathbf{C}^{(i,j,k)}\mathbf{X}^\intercal\mathbf{L}^\intercal\mathbf{R}\mathbf{L}\mathbf{X}) = D_R^{i,j} - D_R^{i,k}.$$

The derivation is from (Petersen and Pedersen, 2006):

$$\frac{\partial tr(\mathbf{C}^{(i,j,k)}\mathbf{X}^\intercal\mathbf{L}^\intercal\mathbf{R}\mathbf{L}\mathbf{X})}{\partial\mathbf{R}} = \mathbf{L}\mathbf{X}\mathbf{C}^{(i,j,k)}\mathbf{X}^\intercal\mathbf{L}^\intercal$$

Using the same method for intra-class and inter-class terms, the subgradient of $\mathcal{F}$ at $\mathbf{R}$ can be derived as

$$\begin{aligned}\nabla\mathcal{F} = \lambda\mathbf{R} + \mathbf{L}\mathbf{X}\Bigg(\sum_{(i,j,k)\in S^+}\mathbf{C}^{(i,j,k)} \\ + \frac{1}{|S_1|}\sum_{(i,j)\in S_1}\mathbf{C_1}^{(i,j)} \\ - \frac{1}{|S_2|}\sum_{(i,k)\in S_2}\mathbf{C_2}^{(i,k)}\Bigg)\mathbf{X}^\intercal\mathbf{L}^\intercal.\end{aligned} \tag{7}$$

According to the PEGASOS algorithm, instead of using all elements in S, S_1 and S_2 to optimize $\mathcal{F}(\mathbf{R})$, we randomly sample subsets of S, S_1 and S_2 with size of B, B_1 and B_2 in each iteration. The full detail of the algorithm is given as procedure LEARN_METRIC.

procedure LEARN_METRIC($\mathbf{A} \in \mathbb{B}^{n\times n}$, $\mathbf{X} \in \mathbb{B}^{n\times d}$, λ, S, S_1, S_2, B, B_1, B_2, T, ξ)
  $\mathbf{L} \leftarrow rand(r, d)$
  $\mathbf{R}_1 \leftarrow zeros(r, r)$
  for $t \leftarrow 1 - T$ **do**
    $\eta_t \leftarrow \frac{1}{\lambda t}$
    $\mathbf{C}, \mathbf{C}_1, \mathbf{C}_2 \leftarrow zeros(n, n)$
    for $b \leftarrow 1 \ \ to \ \ B$ **do**
      $(i, j, k) \leftarrow$ Sample from S
      if $D_R^{i,j} - D_R^{i,k} + \xi \geq 0$ **then**
        $\mathbf{C}_{jj} \leftarrow \mathbf{C}_{jj} + 1, \mathbf{C}_{ik} \leftarrow \mathbf{C}_{ik} + 1$
        $\mathbf{C}_{ki} \leftarrow \mathbf{C}_{ki} + 1, \mathbf{C}_{ij} \leftarrow \mathbf{C}_{ij} + 1$
        $\mathbf{C}_{ji} \leftarrow \mathbf{C}_{ji} + 1, \mathbf{C}_{kk} \leftarrow \mathbf{C}_{kk} + 1$
      end if
    end for
    for $b \leftarrow 1 \ \ to \ \ B_1$ **do**
      $(i, j) \leftarrow$ Sample from S_1
      $\mathbf{C}_{1,ii} \leftarrow \mathbf{C}_{1,ii} + 1, \mathbf{C}_{1,jj} \leftarrow \mathbf{C}_{1,jj} + 1$
      $\mathbf{C}_{1,ij} \leftarrow \mathbf{C}_{1,ij} - 1, \mathbf{C}_{1,ji} \leftarrow \mathbf{C}_{1,ji} - 1$
    end for
    for $b \leftarrow 1 \ \ to \ \ B_2$ **do**
      $(i, k) \leftarrow$ Sample from S_2
      $\mathbf{C}_{2,ii} \leftarrow \mathbf{C}_{2,ii} + 1, \mathbf{C}_{2,kk} \leftarrow \mathbf{C}_{2,kk} + 1$
      $\mathbf{C}_{2,ik} \leftarrow \mathbf{C}_{2,ik} - 1, \mathbf{C}_{2,ki} \leftarrow \mathbf{C}_{2,ki} - 1$
    end for
    $\nabla_t \leftarrow \lambda\mathbf{R} + \mathbf{L}\mathbf{X}(\mathbf{C} + \mathbf{C}_1 - \mathbf{C}_2)\mathbf{X}^\intercal\mathbf{L}^\intercal$
    $\mathbf{R}_{t+1} \leftarrow \mathbf{R}_t - \eta_t\nabla_t$
    $\mathbf{R}_{t+1} \leftarrow [\mathbf{R}_{t+1}]^+$ Optional PSD projection
  end for
  return $\mathbf{L}, \ \mathbf{R}_T$
end procedure

4 Experimental Results

We applied our approach in two different problem domains. One involves a small amount of data points with moderately high dimensions (more than 1,000 and less than 10,000); the other involves a large number of data points with very high dimensions (more than 10,000). The results show that our approach can perform well in both cases.

4.1 Wikipedia Articles

In this domain, the goal is to predict semantic distances between Wikipedia documents about "Search Engine" and "Philosophy Concept". The data sets are available from Shaw *et al.* (2011). They manually labeled these pages to decide which pages should be linked, and extracted bag-of-words features from Web documents after preprocessing steps. Each data set forms a sub-network of all related documents.

The problem here is to learn the metric matrices $\mathbf{L}^\intercal\mathbf{R}\mathbf{L}$ according to the sub-network described above. The random projection matrix $\mathbf{L}$ was randomly initialized in the beginning and $\mathbf{R}$ was obtained by optimization, as described in Section 3.

Tables 1 shows the statistics about the Wikipedia documents data sets.

Table 1: Wikipedia pages dataset

Data set	n	m	d
Search Engine	269	332	6695
Philosophy Concept	303	921	6695

In this table, n denotes the number of data points, m, the number of edges, and d, feature dimensionality.

We split this data set $80/20$ for training and testing, where 20% of the nodes are randomly chosen for evaluations. The remaining 80% of data were used in a five-fold cross-validation to select the best performing hyper-parameters, *e.g.*, λ in Eq. (5). We then trained the model with these hyper-parameters.

With the held-out evaluation data, we applied the learned distance metric to compute the distance between each pair of data points. Then we ranked all distances and measured the quality of ranking using the Receiver Operator Characteristic (ROC) curve. The area under the curve (AUC) was then used to compare the performance of various algorithms.

The list below shows the alternatives of the objective functions used in the algorithms compared in the experiment. Note that HL denotes the hinge loss term, parameterized by either $\mathbf{M}$, the full-rank metric matrix, $\mathbf{L}$, the low-rank approximation matrix, or $\mathbf{R}$, dimensionally-reduced metric matrix after random projection. $\mathcal{I}_1$ denotes the intra-class distances as given in Eq. (3) and $\mathcal{I}_2$ inter-class distances in Eq. (4). Algorithms A, B, C and D are diagonal approximation. Algorithm E implements the low rank decomposition with reduced dimension $r = 300$. Algorithms F, G, H, and I are variations of our approach with combinations of random projection and/or intra/inter-class regularization. The reduced dimension of the new feature space was also 300 and $\mathbf{R}$ is a full matrix. Algorithm J uses PCA to reduce the dimension to 300 and $\mathbf{M}$ here is a full matrix.

In all algorithms, elements in the random projec-

tion matrix $\mathbf{L}$ were generated at random independently except for algorithm L, where $\mathbf{L}$ was generated such that its columns are orthonormal.

A $\frac{\lambda}{2}\|\mathbf{M}\|_F^2 + HL(\mathbf{M})$

B $\frac{\lambda}{2}\|\mathbf{M}\|_F^2 + HL(\mathbf{M}) + \mathcal{I}_1(\mathbf{M})$

C $\frac{\lambda}{2}\|\mathbf{M}\|_F^2 + HL(\mathbf{M}) - \mathcal{I}_2(\mathbf{M})$

D $\frac{\lambda}{2}\|\mathbf{M}\|_F^2 + HL(\mathbf{M}) + \mathcal{I}_1(\mathbf{M}) - \mathcal{I}_2(\mathbf{M})$

E $\frac{\lambda}{2}\|\mathbf{L}\|_F^2 + HL(\mathbf{L})$

F $\frac{\lambda}{2}\|\mathbf{R}\|_F^2 + HL(\mathbf{R})$

G $\frac{\lambda}{2}\|\mathbf{R}\|_F^2 + HL(\mathbf{R}) + \mathcal{I}_1(\mathbf{R})$

H $\frac{\lambda}{2}\|\mathbf{R}\|_F^2 + HL(\mathbf{R}) - \mathcal{I}_2(\mathbf{R})$

I $\frac{\lambda}{2}\|\mathbf{R}\|_F^2 + HL(\mathbf{R}) + \mathcal{I}_1(\mathbf{R}) - \mathcal{I}_2(\mathbf{R})$

J $\frac{\lambda}{2}\|\mathbf{M}\|_F^2 + HL(\mathbf{M})$ (PCA dimension reduction)

L $\frac{\lambda}{2}\|\mathbf{R}\|_F^2 + HL(\mathbf{R}) + \mathcal{I}_1(\mathbf{R})$ (orthonormal projection)

Table 2: Performance Comparison on Wikipedia Documents

	Search Engine	Philosophy Concept
A	0.7297	0.6935
B	0.6169	0.5870
C	0.5817	0.6808
D	0.6198	0.6704
E*	0.6952	0.4832
F	0.6962	0.6849
G	**0.7909**	**0.7264**
H	0.5744	0.6903
I	0.5984	0.6966
J	0.3509	0.4525
L	**0.7939**	**0.7174**

* Algorithm E performed unstably in this experiment even when all initial hyper-parameters were the same.

The running time for combined training and evaluation for A, B, C, D was around 15 seconds(s), for E about 300s, for F, G, H and I 200s and J 300s. All ran on a MacBook Pro notebook computer with 8GB of memory. Learning full-rank $\mathbf{M}$ is impossible because it ran out of memory.

Table 2 shows that algorithm G performs better than others in terms of the AUC metric, suggesting that the distances of two points in the same class is likely to be small and similar to some degree while the ones in the different classes vary a lot, as we speculated. Algorithm A also performs well probably because the bag-of-words features tend to be independent of each other and lack for word-to-word (feature-to-feature) interactions. As a result, diagonal approximation is sufficient to model the semantic similarity here. The low rank approximation algorithm E is unstable. Different trial runs resulted in drastically different AUCs. The AUC reported here is the best observed, but still not the best compared to other algorithms. In contrast, random projection algorithms rarely suffer this problem. PCA with dimensionality reducing to 300 hurt the performance significantly. This might be because there is too much information loss for such a low dimensionality, compared to the original one. However, using random $\mathbf{L}$ to project the feature vectors to 300 dimensions did not seem to have the same problem, according our results of algorithm F.

Comparing different strategies to generate the random projection matrix $\mathbf{L}$, we found that algorithms F and L basically performs similarly, yet variations of the performance of algorithm F in different trials are slightly higher than L, though the variations for both algorithms are negligible.

4.2 Phenotype Similarity

The second dataset comes from a project supported by NIH, with the objective of matching semantically related phenotype variables across studies. For training, phenotype variables that are semantically similar are categorized to be in the same class. The data set that we used here contained annotated data from the harmonization effort in the CARe (the Candidate Gene Association Resource) consortium, led by Prof. Leslie Lange from the University of North Carolina. This data set is comprised of 3,700 phenotypes that were manually classified into 160 categories. We note that previous works in distance metric learning usually used test data sets with less than 30 classes or categories.

A phenotype variable is given as a tuple of a condition description and an expert-assigned category. For example

1. ⟨ Forced Vital ...
   ... Capacity (Litres),
   Forced Expiratory Volume ⟩

2. ⟨ MAX MID-EXPIR'Y FLOW RATE ...
   ... FR MAX CURVE,
   Forced Expiratory Volume ⟩

3. ⟨ Age at natural menopause,
   Menopause ⟩

If we regard condition descriptions as data points and expert-assigned categories as class labels, each tuple above can be considered to be a node with a label in a graph. Hence, nodes with the same labels should be a match (be linked) to form a semantic network. For example, phenotype 1 and 2 forms a match because they have the same category ``Forced Expiratory Volume.''

Previously, we applied the Maximum Entropy model (MaxEnt) to predict the match on pairs, which were labeled according to the annotated data by considering variables in the same category as semantically similar (*i.e.*, a match), and those across categories as dissimilar (Hsu et al., 2011). We extracted features from each pair, such as whether a keyword appear in both phenotype descriptions, and trained a MaxEnt model to predict whether a given pair is a match. To evaluate the performance, we split variables into training and test sets and paired variables within each set as the training and test data sets.

After careful observation of the descriptions of the phenotype variables, we found that they are in general short and non-standardized. To standardize them, and detect their semantic similarity, we expanded the descriptions using UMLS (Unified Medical Language System, http://www.nlm.nih.gov/research/umls/). We search UMLS with a series of n-grams constructed from the variable descriptions. For example, for a variable description: "Age Diagnosed Coronary Bypass". The n-gram words will be "age diagnosed coronary bypass", "age diagnosed coronary", "diagnosed coronary bypass", "age diagnosed", "diagnosed coronary", "coronary bypass". We query UMLS for concepts corresponding to these n-grams. The definitions thus returned are appended to the original variable description. We reused the feature set described in (Hsu et al., 2008) for the BioCreative 2 Gene Mention Tagging task, but removed all char-gram features. This features set was carefully designed and proved to be effective for the gene mention tagging task, but results in a very high dimensional data set. Note that this feature set is different from the one used in our previous work (Hsu et al., 2011).

To make the configurations of our experiment exactly comparable with what described in Sharma *et al.* (2012), we modified the algorithms to be pairwise oriented. We randomly divided the annotated phenotype variable description data into three folds and created pairs for this experiment. The statistics of these folds is shown in Table 3. We trained our models on connected and disconnected training pairs in two folds and then evaluated with the pairs in the other fold. Again, after we used the trained model to predict the distances of all test pairs, we ranked them and compared the quality of the ranking using the ROC curves as described earlier. The evaluation metric is the average of their AUC scores for the three folds.

Table 3: Phenotype dataset

Fold	train+	train−	test+	test−
Fold_1	98,964	2,824,398	24,550	705,686
Fold_2	99,607	2,823,755	25,386	704,850
Fold_3	98,013	2,825,349	23,892	706,344

a. The number of data points is 2,484 and dimension here is 18,919.

b. In this table, train+ denotes number of connected (positive) pairs in training data, $train-$ number of disconnected (negative) pairs in training data, test+ number of connected pairs in testing data, test− number of disconnected pairs in testing data.

We compared algorithms A, E, G, H, I, J and L listed earlier with the same configurations except that the reduced dimensionality $r = 500$. Performance of the MaxEnt algorithm was also reported as K. Their AUC scores are shown in Table 4.

Training and evaluation processes took each fold for algorithm A around 500 seconds(s) , for algorithm E around 1400s , for algorithms F, G, H, I and L around 900s , for J 1400s, and for K 1200s.

The experimental results shows that random projection (algorithm F) outperformed diagonal approximation (algorithm A) by reserving feature-feature interactions with merely 500 dimensions, compared

Table 4: Performance Comparison on Phenotype Data

	AUC_1	AUC_2	AUC_3	AUC*
A	0.7668	0.7642	0.7627	0.7646
E*	0.9612	0.9617	0.9689	**0.9639**
F	0.8825	0.8953	0.8684	0.8820
G	0.9461	0.9545	0.9293	**0.9433**
H	0.7149	0.7272	0.7299	0.7240
I	0.8930	0.8888	0.8809	0.8876
J	0.7582	0.7357	0.7481	0.7473
K	0.8884	0.9107	0.8996	0.8996
L	0.9505	0.9580	0.9527	**0.9537**

$AUC_{1,2,3}$ denote the AUC performance of $Fold_{1,2,3}$ respectively. AUC* is the average AUC score.
* Results of algorithm E varied wildly in each running. For example, using $Fold_1$ with a fixed hyper-parameter setting, its AUC scores were 0.9612, 0.9301, 0.9017 and 0.8920 in different trails, respectively.

to the original dimensionality of nearly $19K$. Again, the best performer is algorithm G, which combines random projection with intra-class regularization. The intra-class term was beneficial but the inter-class term was not. One speculation is that the distance between data points in the same class may be confined in a small range while those in different classes may vary widely. Maximizing inter-class distance here might have distorted the learned feature space. The low rank decomposition (algorithm E) behaved unstably in this data sets. The result shown here is the best-chosen one. In fact, the non-convex low-rank decomposition results in drastically different AUC in each trial run. We speculate that the best result observed here is an overfitting. Diagonal approximation (algorithm A) and PCA (algorithm J) performed similarly, unlike the results for the Web documents.

Finally, when comparing different strategies to generate the random projection matrix $\mathbf{L}$, we had the same conclusion as for the Wikipedia domain that orthonormal random projection (algorithm L) has a slight advantage in terms of low variations in different trials over completely indepndent random project (algorithm F).

5 Discussions and Future Work

We have proposed a convex, input-size free optimization algorithm for distance metric learning. This algorithm combines random projection and intra-class regularization that addresses the weaknesses presenting in the previous works. When the dimension d is in tens of thousands, as in many NLP and IR applications, $\mathbf{M}$ will be hundreds of millions in size, too large and intractable to handle for any existing approaches. Our approach addresses these issues, making the learning not only scalable, but also more accurate.

In the future, we will investigate theoretical properties that explain why the synergy of random projection and intra-class regularization works well and robust. We would also like to investigate how to use unlabeled data to regulate the learning to accomplish semi-supervised learning.

Acknowledgments

Research reported in this publication was supported in part by the National Human Genome Research Institute of the National Institutes of Health (NIH) under Award Number U01HG006894. The content is solely the responsibility of the authors and does not necessarily represent the official views of NIH.

References

John Blitzer, Kilian Q Weinberger, and Lawrence K Saul. 2005. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480.

Reeves Fletcher and Colin M Reeves. 1964. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154.

Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. 2004. Neighbourhood components analysis.

G.H. Golub and C.F. Van Loan. 1996. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press.

Chun-Nan Hsu, Yu-Ming Chang, Cheng-Ju Kuo, Yu-Shi Lin, Han-Shen Huang, and I-Fang Chung. 2008. Integrating high dimensional bi-directional parsing models for gene mention tagging. *Bioinformatics*, 24(13):i286–i294.

Chun-Nan Hsu, Cheng-Ju Kuo, Congxing Cai, Sarah Pendergrass, Marylyn Ritchie, and Jose Luis Ambite.

2011. Learning phenotype mapping for integrating large genetic data. In *Proceedings of BioNLP 2011 Workshop*, pages 19–27, Portland, Oregon, USA, June. Association for Computational Linguistics.

William B Johnson and Joram Lindenstrauss. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.

Kaare Brandt Petersen and Michael Syskind Pedersen. 2006. The matrix cookbook.

Jasson DM Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. 2011. Pegasos: Primal estimated subgradient solver for svm. *Mathematical Programming*, 127(1):3–30.

Shefali Sharma, Leslie Lange, Jose Luis Ambite, Yigal Arens, and Chun-Nan Hsu. 2012. Exploring label dependency in active learning for phenotype mapping. In *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 146–154, Montréal, Canada, June. Association for Computational Linguistics.

Blake Shaw, Bert Huang, and Tony Jebara. 2011. Learning a distance metric from a network. In *Advances in Neural Information Processing Systems*, pages 1899–1907.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2004. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336.

Lieven Vandenberghe and Stephen Boyd. 1996. Semidefinite programming. *SIAM review*, 38(1):49–95.

Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Ng. 2002. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512.

Liu Yang and Rong Jin. 2006. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, pages 1–51.