On Learning Asymmetric Dissimilarity Measures

Krishna Kummamuru, Raghu Krishnapuram
IBM India Research Lab
New Delhi 110016, INDIA
kkummamu,kraghura@in.ibm.com

Rakesh Agrawal
IBM Almaden Research Center
San Jose, CA 95120, USA
ragrawal@us.ibm.com

Abstract

Many practical applications require that distance measures to be asymmetric and context-sensitive. We introduce Context-sensitive Learnable Asymmetric Dissimilarity *(CLAD) measures, which are defined to be a weighted sum of a fixed number of dissimilarity measures where the associated weights depend on the point from which the dissimilarity is measured. The parameters used in defining the measure capture the global relationships among the features. We provide an algorithm to learn the dissimilarity measure automatically from a set of user specified comparisons in the form "x is closer to y than to z," and study its performance. The experimental results show that the proposed algorithm outperforms other approaches due to the context sensitive nature of the CLAD measures.*

1. Introductions

Distance measures play a vital role in many applications such as supervised and unsupervised learning, information retrieval, and product recommendations. Typically, the distances are assumed to be of a specific type such as Euclidean, Mahalanobis, Manhattan, or inverse-cosine. In some cases, the distance measure is specified by the domain expert. Even in the case of adaptive norms [6], the distance measure is symmetric, and does not adapt based on the point from which it is measured. However, many practical applications need the measures to be asymmetric and vary with the context. For example, the measure used to find documents similar to a document on Botany should be different from the measure used in the case of a document on Zoology. Similarly, the measure of inclusion of meaning of one sentence in another is not symmetric [10].

We propose a general class of dissimilarity[1] measures that are asymmetric and spatially-varying. They are defined as a weighted sum of a set of simple dissimilarity measures where the associated weights vary (linearly in the simplest case) within feature space. Since the weights are dependent on the point from which the dissimilarity is measured, i.e., they are spatially varying, we say that the proposed measures are context-sensitive. Moreover, the linear relationship between the weights capture the global semantic relationships among the features. We refer to the proposed measures as *Context-sensitive Learnable Asymmetric Dissimilarity* (CLAD) measures.

We present an algorithm to learn the measure, given a set of relative comparisons of the form "x is closer to y than to z". We refer to such relative comparisons as *triplet constraints*. We convert the triplet constraints into an objective function and use gradient descent for optimization. Three variants of the objective function yield three variants of the learning algorithm. We empirically evaluate our algorithm against prior approaches, including the one based on SVMs [15]. The results show that our algorithm outperforms the other approaches due the context sensitive nature of the CLAD measure.

1.1. Related work

Learning the distance measure that a user implicitly employs to arrive at the relevance of search results has been investigated earlier in multimedia retrieval (e.g., [14]). This problem has also been explored in machine learning and data mining communities. Though not all (e.g., [2, 9, 15]), most of these efforts have been targeted toward learning distance measures for the purpose of improving the performance of clustering algorithms (e.g. [3, 4, 12, 13, 17]).

The distance learning algorithms can be categorized according to the type of information they require to learn the underlying distance measure. Some algorithms learn the distances from a given set of unlabeled samples (e.g., [12]), while other algorithms need pair-wise constraints in terms of pairs of instances that are similar or dissimilar, or belong to same or different classes (e.g., [17, 3, 4]). For learning asymmetric distances, it is not meaningful to use pair-wise

[1]We prefer the term dissimilarity over distance because distance could mean a metric but the measure defined here is not a metric.

constraints. We therefore employ triplet constraints of the form "x is closer to y than to z". The triplet constraints have been used by Schultz and Joachims in [15], although for learning symmetric distances. They formulate distance learning from relative comparisons as a quadratic optimization problem, in which the triplet constraints directly become constraints in the optimization problem. They also propose an approach based on SVMs to learn the distances. Our experimental results show that our algorithm consistently outperforms their proposal.

Like CLAD, the SVaD measures in [12] are also defined as a weighted sum of m simple dissimilarity measures. However, in SVaD, the feature space is divided into K regions, the weights change only across the regions and they remain constant within a region. The K regions can be thought of as representing K topics and each region is associated with a specific weight vector. In non-clustering scenarios, dividing the feature space into K regions may not be appropriate. We, therefore, use continuously varying weights.

1.2. Paper Organization

We formally define CLAD measures in Section 2 and formulate the problem of learning CLAD measures from relative comparisons in Section 3. In Section 4, we describe the learning algorithm. We present the experimental results in Section 5 and summarize our contributions in Section 6.

2. Context-sensitive Learnable Asymmetric Dissimilarity Measures

We first provide the formal definition and then explain the underlying intuition.

2.1. CLAD Definition

DEFINITION 2.1 Let $g_1, g_2, \ldots,$ and g_m be m dissimilarity measures that measure m different aspects of dissimilarity. Let the feature space be $\Re^n$. Given a set of m n-dimensional vectors $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ and an m dimensional vector $\mathbf{w}$, the CLAD measure of $\mathbf{y}$ from $\mathbf{x}$, $d_{A,W}(\mathbf{x}, \mathbf{y})$ is defined as:

$$d_{A,\mathbf{w}}(\mathbf{x}, \mathbf{y}) \equiv \sum_{i=1}^{m} (\mathbf{a}_i^T \mathbf{x} + w_i) g_i(\mathbf{x}, \mathbf{y}). \qquad (1)$$

Note that a_{ij} represents the effect of occurrence of the jth term in $\mathbf{x}$ on the weight of the ith term when computing the distance from $\mathbf{x}$. The larger the value of a_{ij}, the larger is the weight and the dissimilarity.

When $\mathbf{a}_i = 0, \forall i$, $m = n$ and $g_l(\mathbf{x}, \mathbf{y}) = (x_l - y_l)^2$ then CLAD is identical to the weighted Euclidean distance measure. Similarly, when $\mathbf{a}_i = 0, \forall i$, $m = n$ and $g_l(\mathbf{x}, \mathbf{y}) = (1/n - x_l y_l)$ then it is identical to the weighted cosine distance measure. It may be noted that the Kullback-Leibler divergence measure [11] is a special case of CLAD measures when $\mathbf{w} = 0$, $\mathbf{A} = \mathbf{I}$, and $g_l(\mathbf{x}, \mathbf{y}) = \log(x_l/y_l)$.

2.2. Motivation

The motivation for CLAD comes from the domain of text documents, although the following observations hold in other domains too. When we are measuring the dissimilarity of a document y from a document x on Botany, we want to measure the dissimilarity with respect to the words that are specific to Botany. The words that are not related to Botany even though they occur in both the documents should not contribute to the dissimilarity. This behavior can be best captured when the weights on the features change with the point from which the dissimilarity is being measured.

Assume that the documents are represented by a vector of length equal to n, the number of words in the vocabulary. The element corresponding to a word in the vector is proportional to the frequency of occurrence of the word in the document. That is, $x_i = f_i/F$, where f_i is the frequency of the ith word in the document $\mathbf{x}$ and $F = \sqrt{\sum f_i^2}$. Consider $m = n$ and $g_l(\mathbf{x}, \mathbf{y}) = (1/n - x_l y_l)$. Let $\mathbf{x}$ be a document on Botany. As mentioned, it would be desirable to compute the dissimilarity of $\mathbf{x}$ with other documents only with respect to those words relating to Botany. Suppose $\mathbf{x}$ contains a noisy word such as "network". Then, the traditional measure would yield a dissimilarity less than 1 (or a non-zero similarity) with any document $\mathbf{y}$ containing the word "network" even though $\mathbf{y}$ may not be directly related to Botany. Suppose $\mathbf{a}_i$ corresponding to "network" is such that its entries corresponding to Botany related words are large, then the effective weight $\mathbf{a}_i^T\mathbf{x} + w_i$ would be large and hence the dissimilarity. Thus, by employing A, we can mitigate the problem of noisy words. Note that $\mathbf{w}$ is applied uniformly in feature space to give more importance to globally significant words and less to stop-words.

For illustration, we have in the Appendix A the words related to some example words automatically extracted using matrix $\mathbf{A}$ obtained by applying our proposed algorithm on a textual data set. We find that the most of the words corresponding to the least values of a_{ij} are indeed related to the ith word.

3. Learning using Relative Comparisons

Assume that the relative comparisons are given in the form of triplets $\tau_i = (\mathbf{x}_1^i, \mathbf{x}_2^i, \mathbf{x}_3^i); i = 1, \ldots, r$, meaning that $\mathbf{x}_1^i$ is closer to $\mathbf{x}_2^i$ than to $\mathbf{x}_3^i$. That is, τ_i implies that

$$d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_2^i) < d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_3^i); i = 1, \ldots, r. \qquad (2)$$

We, therefore, interchangeably use constraints and inequalities. Given these triplets, the objective is to find A and $\mathbf{w}$ such that the number of unsatisfied inequalities is minimized.

Let

$$e_{A,\mathbf{w}}(\tau_i) = \begin{cases} 0 \text{ if } d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_3^i) - d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_2^i) > 0, \\ 1 \text{ otherwise.} \end{cases}$$

Then, the objective of learning process is to minimize E, which is defined as: $E(A, \mathbf{w}) = \sum_{i=1}^{r} e_{A,\mathbf{w}}(\tau_i)$.

4. Learning Algorithm

Note that $E(A, \mathbf{w})$ is a discontinuous function. We approximate $E(A, \mathbf{w})$ by a function that can be differentiated with respect to a_{ij} and w_i, and use a gradient descent approach to find optimal A and $\mathbf{w}$. We approximate $E(A, \mathbf{w})$ using the following three methods.

4.1. Cumulative Error

In this scheme, we approximate E by the cumulative error of each unsatisfied inequality. Let $u_{A,\mathbf{w}}(\tau^i) = h(d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_3^i) - d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_2^i))$, for $i = 1, \ldots, r$, where

$$h(z) = \begin{cases} -z \text{ if } z < 0, \\ 0 \text{ otherwise.} \end{cases} \quad (3)$$

Note that $u_{A,\mathbf{w}}(\tau^i)$ is an approximation of $e_{A,\mathbf{w}}(\tau_i)$. Then, the cumulative error is defined as:

$$J_C(A, \mathbf{w}) = \sum_{i=1}^{r} u_{A,\mathbf{w}}(\tau^i). \quad (4)$$

Let $I(A, \mathbf{w}) = \{i : u_{A,\mathbf{w}}(\tau^i) > 0\}$ denote the set of inequalities not satisfying (2). Then,

$$J_C(A, \mathbf{w}) = \sum_{i \in I(A,\mathbf{w})} [d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_2^i) - d_{A,\mathbf{w}}(\mathbf{x}_1^i, \mathbf{x}_3^i)]. \quad (5)$$

From the definition of the dissimilarity measure, we obtain

$$J_C(A, \mathbf{w}) = \sum_{i \in I(A,\mathbf{w})} \sum_{l=1}^{m} (\mathbf{a}_l^T \mathbf{x}_1^i + w_l)[g_l(\mathbf{x}_1^i, \mathbf{x}_2^i) - g_l(\mathbf{x}_1^i, \mathbf{x}_3^i)].$$

Let $y_{il} = [g_l(\mathbf{x}_1^i, \mathbf{x}_2^i) - g_l(\mathbf{x}_1^i, \mathbf{x}_3^i)]$ then,

$$J_C(A, \mathbf{w}) = \sum_{i \in I(A,\mathbf{w})} \sum_{l=1}^{m} (\mathbf{a}_l^T \mathbf{x}_1^i + w_l) y_{il}. \quad (6)$$

When $\mathbf{a}_l = 0, \forall l$, and $m = n$, the above problem reduces to the classical perceptron learning problem [7]. That is, finding a hyperplane containing the origin such that all $\mathbf{y}_i$ lie on the same side of the plane. In [15], the authors solve this problem using SVMs.

4.2. Maximum Error

$E(A, \mathbf{w})$ can also be approximated with the maximum error of unsatisfied inequalities. That is, the aim is to minimize the maximum error made by the unsatisfied inequalities. Let

$$i_m = \arg \max_{i : u_{A,\mathbf{w}}(\tau^i) > 0} u_{A,\mathbf{w}}(\tau^i).$$

Then the corresponding objective function is given by

$$J_M(A, \mathbf{w}) = \sum_{l=1}^{m} (\mathbf{a}_l^T \mathbf{x}_1^{i_m} + w_l) y_{i_m l}. \quad (7)$$

In SVM terminology [7], in this formulation, the objective is to find a feasible solution that maximizes the margin of separation.

4.3. Cumulative of k Maximum Errors

This scheme is a compromise between the previous two schemes. Here, the objective is to minimize the cumulative of the top k errors of unsatisfied inequalities. Let $I_{CM}(A, \mathbf{w})$ contain the indices of k unsatisfied inequalities with maximum errors. Then, the objective function would be

$$J_{CM}(A, \mathbf{w}) = \sum_{i \in I_{CM}(A,\mathbf{w})} \sum_{l=1}^{m} (\mathbf{a}_l^T \mathbf{x}_1^i + w_l) y_{il}. \quad (8)$$

4.4. Avoiding trivial solutions

In all the above schemes, we impose the following constraints on A and $\mathbf{w}$ to avoid trivial solutions to the minimization problem:

$$\|\mathbf{a}_i\|_2 = 1, \forall i \text{ and } \|\mathbf{w}\|_2 = 1.$$

4.5. Algorithm

We propose a gradient descent algorithm to solve the minimization problem. Differentiating $J_C(A, \mathbf{w})$ with respect to w_l we obtain:

$$\frac{\partial J_C}{\partial w_l} = \sum_{i \in I(A,\mathbf{w})} y_{il}. \quad (9)$$

With respect to a_{jl}, we obtain:

$$\frac{\partial J_C}{\partial a_{jl}} = \sum_{i \in I(A,\mathbf{w})} x_{1j}^i y_{il}. \quad (10)$$

The gradients corresponding to J_M and J_{CM} also look similar to the above equations.

```
Input:
  1. Type of distance measure $g_l$
  2. Relative qualitative feedback, $\tau_i, i = 1, \ldots, m$
  3. Parameters: $\eta < 1, \rho(0) < 1$
Output: $A$ and $\mathbf{w}$ that minimize $J(A, \mathbf{w})$
Learning Algorithm:
  1. Set t=0;
  2. Initialize $\mathbf{a}_i$ to zero vector and $w_l$ to $1/\sqrt{m}$.
  3. Compute $y_{il} \forall i$, and $l$.
  4. Determine the set of unsatisfied inequalities $I(A, \mathbf{w})$.
  5. Update $w_l$ using (9) and $w_l = w_l - \rho(t)(\partial J/\partial w_l)$ for
     $l = 1, \ldots, m$.
  6. Normalize $\mathbf{w}$
  7. Update $a_{jl}$ using (10) and $a_{jl} = a_{jl} - \rho(t)(\partial J/\partial a_{jl})$
     for $j = 1, \ldots, n$, and $l = 1, \ldots, m$.
  8. Normalize $\mathbf{a}_l$ for $l = 1, \ldots, m$.
  9. $\rho(t+1) = \eta\rho(t), t = t + 1$.
  10. Go to STEP 4 until the termination criterion is satisfied.
  11. Exit.
```

Figure 1. Learning Algorithm

We summarize the algorithm in Figure 1. The algorithm starts with a uniform weights and all-zero vectors $\mathbf{a}_i$. In each iteration, the set of unsatisfied inequalities is found, the gradients in (9) and (10) are computed, and the weights and vectors are updated. The algorithm stops after reaching a termination criterion. The most often used termination condition is the completion of a fixed number of iterations. We denote the algorithms that optimize J_C, J_M and J_{CM} by Algo$_C$, Algo$_M$ and Algo$_{CM}$, respectively.

The update equations are quite intuitive. For example, if g_l is dimension-dependent as in the case of weighted Euclidean, it can be observed from (9) that the weight corresponding to the lth dimension is decreased by the cumulative error made by all the unsatisfied inequalities in that dimension. Note that the update for w_l is the same as in the perceptron learning algorithm [7]. A similar interpretation follows the update for a_{jl}.

5. Experimental Study

We present the results of two sets of experiments that illustrate the superior performance of the proposed CLAD-based algorithms over traditional approaches. In the first set of experiments, we study the relative performance of the three variants of the proposed algorithm. In the second set of experiments, we compare the performance of the proposed algorithm with the SVM-based distance learning algorithm [15].

As in [15], we measure the performance of algorithms by the number of constraints satisfied by the dissimilarity measure after convergence. The bigger the number of satisfied constraints, the better the performance of the algorithm.

5.1. Data sets

We used three data sets in our experimental studies. In the first set of experiments, we used the “Car” data set. In the second set of experiments we used the well-known 20 NewsGroup and WebKB data sets in addition to the Car data set. The Car data set consists of numerical attributes. We used additional textual data sets in the second set of experiments because the algorithm [15] with which we compare the proposed algorithm, was applied on textual data (viz., WebKB).

Car data set

The Car data set consists of a description of 1506 car models with 23 numeric attributes each, and was obtained from the popular website *Edmunds.com*. Examples of features used are: FrontHipRoom, BaseNumberof-Cylinders, CityRange, FrontHeadroom, RearHipRoom, Price, FrontLegRoom, Height, Weight, RearLegRoom, RearShoulderRoom, BaseEngineSize, FrontShoulder-Room, Wheelbase, Width, Torque, CityMileage, HighwayRange, Length, RearHeadroom, Horsepower, FuelTankCapacity, and HighwayMileage. After eliminating models with missing data, we retained a feature set associated with 906 models. (We plan to upload this data set onto the UCI machine learning repository soon.) Since the values of the attributes vary over a wide range, they were normalized into the range [0, 1]. In the second set of experiments, we used a subset of this data set containing the data corresponding to three of the most populous models viz., Compact Sedan, Midsize SUV and Mini SUV which contain 71, 119 and 94 data points respectively. We consider 70% of this subset to generate training inequalities and the remaining 30% to generate testing inequalities.

20 News Group data set

We considered three different subsets (all of size 500) of 20 News Group data [1] that are known to contain clusters of varying degrees of separation [16]. We denote the three subsets by *Binary*, *Multi5* and *Multi10*. *Binary* had 250 randomly sampled documents from the group talk.politics.mideast and another 250 randomly sampled documents from talk.politics.misc. *Multi5* had 100 randomly-sampled documents from each of the groups comp.graphics, rec.motorcycles, rec.sport.baseball, sci.space, and talk.politics.mideast. Finally, *Multi10* had 50 randomly-sampled documents from each of the groups alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, and talk.politics.gun. We used two versions of each of the subsets *Binary*, *Multi5* and *Multi10*, one to generate the training inequalities and the other to generate testing inequalities.

Documents were represented using a set of vocabulary terms. Vocabulary sets were generated by stopword removal, stemming and thresholding on the frequency of occurrence of terms. We used normalized term-frequency vectors to represent the documents. The size of the vocabulary used to represent the documents in the *Binary* data set was about 2300, in *Multi5* about 4000 and in *Multi10* about 1400.

WebKB data set

The WebKB data set [5] contains 8145 Web pages pertaining to computer science departments of various universities. The collection includes the entirety of four departments, and additionally, an assortment of pages from other universities. The pages are divided into seven categories: student, faculty, staff, course, project, department and other. In this paper, we used the four most populous non-other categories: student, faculty, course and project, all together containing 4199 pages. This data set was used in the experiments reported in [15]. As in [15], we split the data set into a training set and a testing set containing 70% and 30% of the documents, respectively. As in the case of the 20 News-Group data sets, the vocabulary set used to represent the documents was generated by stopword removal, stemming and thresholding on the term-frequency, and the documents were represented by normalized term frequency vector.

5.2. Comparison of Algo$_C$, Algo$_M$, and Algo$_{CM}$

The first set of experiments that used the Car data set was aimed at testing the ability of the algorithms to converge to a solution with and without noisy relative comparisons. This was done by assuming some w_{target} to generate the inequalities. In this case, we initialized the weights to a random vector in all algorithms. We report the average values over 10 runs. Since we know the target weight, apart from the number of unsatisfied constraints, we measure the performance of the algorithms by l_2 distance, $\|\mathbf{w}_{target} - \mathbf{w}_{predicted}\|$, between the target weight and the predicted weight (also referred to as prediction error).

Without noise

Figure 2 shows the progress of Algo$_C$ in a typical run. The top graph in the figure shows the evolution of number of

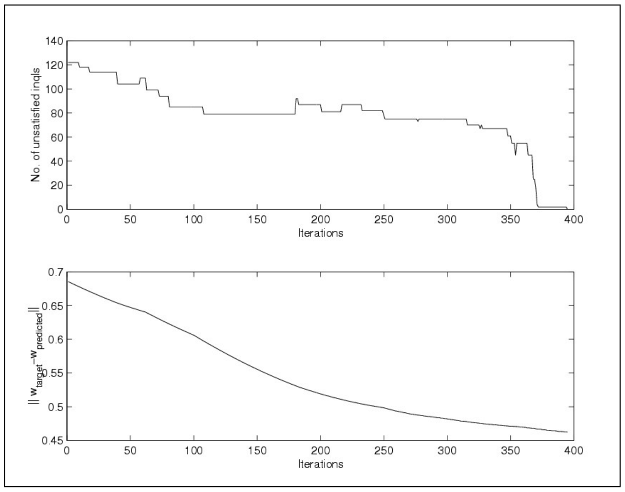

Figure 2. Convergence of Algo$_C$.

unsatisfied inequalities and the bottom the prediction error with the number of iterations. As it is expected of a gradient descent algorithm, the error is monotonically decreasing. Even though the number of unsatisfied inequalities is not strictly monotonically decreasing, as the iterations progress, it eventually becomes zero.

Table 1 compares performance of the three versions of the proposed algorithm in learning the target weights. In all the three cases, the algorithms converged with no unsatisfied inequalities. However, the predicted weights differed from the target weights to some extent. The middle column in the table shows the number of iterations taken by the algorithms to converge to zero unsatisfied inequalities averaged over 10 random initializations of the weights. We note that Algo$_C$ converges faster but results in a larger prediction error and Algo$_{CM}$ finds the target weights with high accuracy, but relatively slowly when compared with Algo$_C$.

Note that Algo$_C$ tries to find a feasible **w** that satisfies all the inequalities. On the other hand, Algo$_M$ not only finds a feasible **w** but also tries to find a **w** that maximizes the margin. This is why we see that the prediction error of Algo$_M$ is less than that of Algo$_C$. However, Algo$_M$ takes more iterations to converge to a feasible solution. From equations (7), (8) and (9), it may be noted that Algo$_M$ moves the weight toward the $\mathbf{y}_{i_m}$ vector corresponding to most unsatisfied inequality whereas, Algo$_{CM}$ moves the weight toward the average of **y**'s corresponding to the top few unsatisfied inequalities. Thus, Algo$_{CM}$ moves the weight more cautiously. This is the reason for Algo$_{CM}$ to be faster and more accurate than Algo$_M$.

With noise

The training data consisting of relative comparisons is generally prone to noise. To simulate this scenario, we in-

Table 1. Comparison of Algo$_C$, Algo$_M$ and Algo$_{CM}$.

Algorithm	Average # of iterations	$\|\mathbf{w}_{target} - \mathbf{w}_{predicted}\|$
Algo$_C$	419	0.21
Algo$_M$	677	0.17
Algo$_{CM}$	601	0.14

Table 2. Performance on Noisy Data.

Algorithm	% of Unsatisfied Inequalities	Average # of Iterations
Algo$_C$	6.26	794
Algo$_M$	11.29	560
Algo$_{CM}$	10.09	879

troduced noise in our training data by randomly reversing some of the preferences generated using $\mathbf{w}_{target}$ in the training data. This situation may occur when two objects are almost equally preferable and the user may have preferred one over the other by a narrow margin. We made the probability of a particular preference being reversed a decreasing function of the difference between the evaluations. Thus, preferences based on narrow margins were more likely to be inconsistent.

In this case, due to inconsistencies in the preference, the algorithms do not find the set of weights leading to zero unsatisfied inequalities. The algorithms are terminated if there is no improvement in the number of unsatisfied inequalities for a fixed number of iterations. Table 2 summarizes the results for noisy data. The middle column of the table shows the percentage of unsatisfied inequalities. We see that Algo$_C$ converges to a more accurate set of weights even though it takes more iterations than Algo$_M$. This behavior of Algo$_C$ is mainly due to the fact that it averages all $\mathbf{y}$'s corresponding to unsatisfied inequalities, thus minimizing the effect of noise. For these reasons, and since in general we do not believe in the existence of a set of A and $\mathbf{w}$ that would result in no inequalities being violated, we use Algo$_C$ in the next set of experiments.

Execution Speed

We ran our experiments on a 2GHz Pentinum 4 server with 1 GB RAM. In the case textual data sets, we found that the time taken to complete 40 iterations by Algo$_C$ increased very slowly (from about 44 minutes to 48 minutes) with the number of training constraints varying from 1,000 to 200,000.

5.3. Comparison with SVM and other Algorithms

In the second set of experiments, we use all the data sets to compare the performance of the proposed algorithm with that of other algorithms. From these data sets, we generated relative comparisons using the following procedure. A triplet $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ from the training set was first randomly chosen and was added to the set of training triplets only if the class labels of the first two are the same and the label of the third is different. Similarly, we generated the testing triplets from the testing set. We kept the number of training samples equal to the number of testing samples. The training and testing constraints were generated from two different subsets. In these experiments, we consider $m = n$ and $g_l(\mathbf{x}, \mathbf{y}) = (x_l - y_l)^2$. It may be noted that, in case of 20 NewsGroup and WebKB data, since we represent the documents as unitized term-frequency vectors, the resulting measure would be equivalent to the weighted cosine dissimilarity measure.

We report the results when Algo$_C$ was used to learn CLAD. In all the experiments, the values of $\rho(0)$ and η were set to 0.001 and 0.95 respectively. Algo$_C$ was stopped after completing 40 iterations in each case. We denote the results obtained by Algo$_C$ as CLAD.

The baseline accuracy we consider is that obtained by uniformly weighting all the features. We refer to the corresponding results as "Uniform Weights". When A is considered to be the set of zero vectors, Algo$_C$ becomes identical to the perceptron learning algorithm. Hence, we report the corresponding results of Algo$_C$ as that of the "Perceptron" learning algorithm.

We also compared Algo$_C$ to the approach proposed in [15]. We used SVM^{light} [8] for this purpose and fixed the value of the regularization parameter C to 1. We refer to the corresponding results as "SVM".

To study how various algorithms generalize from different amounts of training data, we performed the experiment in which the number of training constraints are changed from 1,000 to 200,000. Figure 5.3 shows the learning curves of various algorithms on Car, Binary, Multi5 and WebKB data sets. Since the graphs corresponding to Multi10 looked very similar to that of Multi5, we omit the Multi10 graph here.

CLAD performed consistently better than Uniform Weights, Perceptron and SVMs on all data sets because of its context-sensitive nature. As shown in Appendix A, the parameters in CLAD measure show the desired properties mentioned in Section 2.

6. Summary

We defined a general class of dissimilarity measures and proposed three variants of a learning algorithm to learn

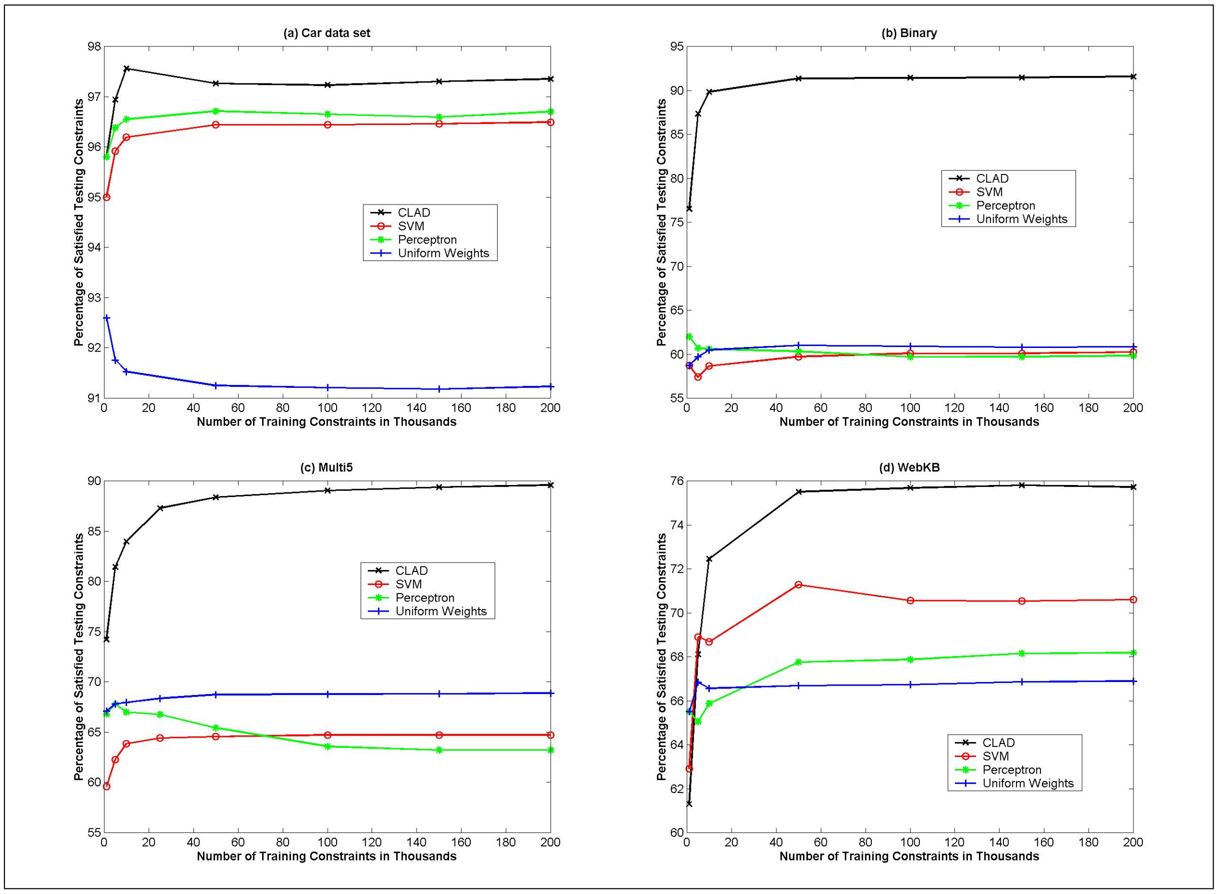

Figure 3. Learning curves of various algorithms on different data sets.

the dissimilarity measures from triplet constraints. We analyzed the performance of these variants in the presence noise. We also compared the performance of the proposed algorithm with that of SVMs by comparing the accuracies of the corresponding dissimilarity in satisfying the triplet constraints. We observed that the proposed algorithm performs remarkably better than other competing algorithms, particularly on textual data sets. Moreover, the parameters used in defining the measure are shown to capture the relationship between the features.

In the present work, we addressed a fundamental problem in data mining, viz., learning the dissimilarity measure. In the future, we would like to explore the use of CLAD in clustering, classification, information retrieval and product recommendation.

References

[1] *http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.tar.gz*.

[2] C. C. Aggarwal. Towards systematic design of distance functions for data mining applications. In *Proceedings of SIGKDD*, pages 9–18. ACM Press, 2003.

[3] M. Bilenko, S. Basu, and R. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of ICML*, pages 81–88, 2004.

[4] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Technical Report TR2003-1892, Cornell University*, 2003.

[5] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[6] E. E. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings of IEEE Conference on Decision and Control*, pages 761–766, Piscataway, NJ, 1979.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 1999.

[8] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD*, pages 133–142. ACM Press, 2002.

[10] K. Krishna and R. Krishnapuram. A clustering algorithm for asymmetrically related data with applications to text mining. In *Proceedings of CIKM*, pages 571–573. ACM, 2001.

[11] S. Kullback. *Information Theory and Statistics*. Cloucester, MA: Peter Smith, 1968.

[12] K. Kummamuru, R. Krishnapuram, and R. Agrawal. Learning spatially variant dissimilarity (svad) measures. In *Proceedings of SIGKDD*, pages 611–616, 2004.

[13] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transaction on SMC - Part B: Cybernetics*, 27(5), Oct 1997.

[14] Y. Rui, T. S. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 25–36, 1998.

[15] M. Schultz and T. Joachims. Learning a distance metric with relative comparisons. *Advances in Neural Information Processing Systems*, 16, 2003.

[16] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of SIGIR*, pages 208–215, 2000.

[17] E. Xing, A. Ng, M. Jordon, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, 16:505–512, 2003.

A. Appendix

We list below the words corresponding to the first 40 least values of a_{ij} along with the ith word for some i where the A matrix was obtained by applying Algo$_C$ on *Multi5* data set. We also show the most probable class associated with the root words within parentheses. It may be noted that most of the words corresponding to the least a_{ij} are indeed related to the ith word.

air (sci.space): space orbit sky nasa hst thing launch internet moon idea mission night nicho time cost rocket dseg toronto billboard isn mass work earth actual zoo fred power design spencer talk vnet ibm find sunset loss put high cso star engin

aircraft (sci.space): space nasa pat orbit time think access digex hst fund long sky idea don mission prb make earth billboard gov thing flight internet cso moon launch interest uiuc design plant toronto mass project sunset mccall engin island problem major zoo

acm (comp.graphics): file graphic program format map point bit color code sphere gif data convert think run iff version email public virtual ibm fast siggraph user center read mode thank video polygon system engin plane group fax interest book umich ftp screen

ascii (comp.graphics): graphic bit file group program convert point siggraph color cost video format run help make think map copyright look post code polygon email ibm center interest don version set window robert motorola algorithm support server system bezier fax dec call

bike (rec.motorcyles): bmw dod rider drive bnr dog duke org tool ride tek deal drink acpub moa shaft andrew msf owner mike road switch beavington sale adjust biker max parr behanna driver respect act buck log std tank honda ama morgan david

bmw (rec.motorcyles): bike dod ride dog sun wheel behanna stop road drive biker bnr max nec hydro back car yamaha sure steer green front mike org shaft seca effect similar honda duke pipe hole thread drink learn ursa csundh30 contact patch seat