Online and Batch Learning of Generalized Cosine Similarities

Ali Mustafa Qamar and Eric Gaussier
Laboratoire d'Informatique de Grenoble (LIG)
Grenoble University, France
ali-mustafa.qamar@imag.fr, eric.gaussier@imag.fr

***Abstract*—In this paper, we define an online algorithm to learn the generalized cosine similarity measures for *k*-NN classification and hence a similarity matrix A corresponding to a bilinear form. In contrary to the standard cosine measure, the normalization is itself dependent on the similarity matrix which makes it impossible to use directly the algorithms developed for learning Mahanalobis distances, based on positive, semi-definite (PSD) matrices. We follow the approach where we first find an appropriate matrix and then project it onto the set of PSD matrices, which we have adapted to the particular form of generalized cosine similarities, and more particularly to the fact that such measures are normalized. The resulting online algorithm as well as its batch version is fast and has got better accuracy as compared with state-of-the-art methods on standard data sets.**

***Keywords*-Generalized cosine; Similarity learning; *k*-NN classification**

I. Introduction

Many people have used the underlying geometry of the data to improve the *k*-NN algorithm (e.g. by learning Mahanalobis distance metric instead of standard euclidean distance), thus paving the way for a new research area termed *metric learning*. Most of these works have based their approaches on distance learning [1], [2], [3], [4], [5]. However, some other works have constated that similarity should be preferred over distance. A similarity is dramatically different from the distance since it is not necessarily positive and it also does not obey the triangle inequality. Similarity measures are usually preferred over distance ones while dealing with textual data, where cosine measure has been proved to be more appropriate as compared with the various distance metrics. Still many approaches [6], [7], [8], [9], including ours where we learn generalized cosine similarities, show that cosine similarity should be considered as opposed to the euclidean distance, over non-textual collections (e.g. *Balance*, *Ionosphere*, *Iris* and *Wine*) in addition to the textual ones. This explains the importance of learning appropriate similarity measures, instead of distance metrics, for *k*-NN classification over various data collections. If several works have partially addressed this problem (as for example [10], [11], [12], [9]) for different applications, we know of no work which has fully addressed it in the context for generalized cosine similarities. This is exactly the goal of the present research.

The rest of the paper is organized as follows: Section 2 describes the problem we are dealing with, where we define the generalized cosine similarity. The online algorithm, along with the projections onto the set of positive, semi-definite (PSD) matrices, for both the separable and inseparable cases, are discussed in Section 3. Theoretical justification for the algorithm is described in Section 4. Finally, section 5 provides experimental validation along with a comparison with other methods.

II. Problem Setting

Let x and x' be two examples in $\mathbb{R}^p$. We consider similarities of the form:

$$s_A(x, x') = \frac{x^t A x'}{\sqrt{x^t A x}\sqrt{x'^t A x'}} \tag{1}$$

where $A \geq 0$ is a positive, semi-definite matrix. By choosing A as the identity matrix, equation 1 becomes the standard cosine similarity. Other positive, semi-definite matrices define different scalar products and norms, so that equation 1 corresponds to a cosine in a new basis of the underlying vector space. Because of this property, we will say that equation 1 corresponds to the family of Generalized Cosine Similarities.

The examples we consider are in the form of tuples, (x, x', y) where each example is composed of the instance pair (x, x') and a label y which is +1 when x and x' are similar and is -1 in the case when they are dissimilar. When the data is separable, the margin of a sample, S, denoted by 2γ, is defined as the minimum separation between all pairs of similar $(x_1, x'_1, +1)$ and dissimilar $(x_2, x'_2, -1)$ examples:

$$s_A(x_1, x'_1) - s_A(x_2, x'_2) \geq 2\gamma \tag{2}$$

By introducing a threshold $b \in \mathbb{R}$, the above equation can be rewritten as:

$$\forall (x, x', y) : y = +1 \Rightarrow s_A(x, x') \geq b + \gamma$$

$$\forall (x, x', y) : y = -1 \Rightarrow s_A(x, x') \leq b - \gamma$$

where $\gamma > 0$ and $-1 + \gamma \leq b \leq 1 - \gamma$. The two inequalities can be combined to form a single linear constraint:

$$y(b - s_A(x, x')) \leq -\gamma \tag{3}$$

Considering tuples of the form $(x_\tau, x'_\tau, y_\tau)$, at each time step, or round τ, we can compute the loss incurred by the current matrix-threshold pair (A, b) as:

$$l_\tau(A, b) = max\{0, y_\tau(b - s_A(x_\tau, x'_\tau)) + \gamma\}$$

which is a variant of the hinge loss. Our goal is thus to find a matrix-threshold pair (A, b) which minimizes the overall loss. When the data is separable, there exists a matrix-threshold pair such that the overall loss is 0 (as inequality 3 holds for matrix-threshold pairs separating the data). We now present an online algorithm to learn a matrix-threshold pair, reviewing the cases where the data is separable and is inseparable.

III. Online Algorithm

In the case where the data is separable, $\exists A \geq 0,\ \exists b, -1 + \gamma \leq b \leq 1 - \gamma$ such that the matrix-threshold pair (A, b) completely separates the data, i.e. has zero loss for all time steps. Because the matrix A should separate the data and be, at the same time, positive, semi-definite, one can rely on a strategy based on first finding a matrix-threshold pair with zero loss and close to the current matrix-threshold pair, and second on projecting the obtained matrix on the set of positive, semi-definite matrices (an approach reminiscent of the one defined in [4]). The first step aims at finding matrix-threshold pairs with small loss, whereas the second step ensures the fact that the obtained matrix is positive, semi-definite and hence defines a valid generalized cosine similarity.

Let $C_\tau \subset \mathbb{R}^{n^2+1}$ be the set of all matrix-threshold pairs having zero loss on example $(x_\tau, x'_\tau, y_\tau)$:

$$C_\tau = \{(A, b) \in \mathbb{R}^{n^2+1} : l_\tau(A, b) = 0\}$$

We then define C_a as the set of all admissible matrix-threshold pairs:

$$C_a = \{(A, b) \in \mathbb{R}^{n^2+1} : A \geq 0, -1 + \gamma \leq b \leq 1 - \gamma\}$$

The update step of our algorithm is thus based on two projections:

1) First, project the current matrix-threshold pair (A_τ, b_τ) on C_τ. The matrix threshold pair thus obtained is denoted by $(A_{\hat{\tau}}, b_{\hat{\tau}})$,
2) Then project $(A_{\hat{\tau}}, b_{\hat{\tau}})$ onto C_a to get $(A_{\tau+1}, b_{\tau+1})$

We now review these two projections.

A. Projection onto C_τ

The set of matrix-threshold pairs having zero loss on (x, x') can be rewritten as:

$$C_\tau = \{(A, b) \in \mathbb{R}^{n^2+1} : y[\frac{x^t A x'}{\sqrt{x^t A x}\sqrt{x'^t A x'}} - b] \geq \gamma\}$$

Let us introduce the following two quantities, which will allow us to define a simple projection:

$$\frac{1}{R_{-1}} = min[x^t A x, x'^t A x']$$

$$\frac{1}{R_{+1}} = max[x^t A x, x'^t A x']$$

We have:

$$R_{+1}(x^t A x') \leq \frac{x^t A x'}{\sqrt{x^t A x}\sqrt{x'^t A x'}} \leq R_{-1}(x^t A x')$$

By subtracting b from all terms and multiplying by y, we get:

$$\begin{aligned} y[R_{+1}(x^t A x') - b] \quad &\leq \quad y[\frac{x^t A x'}{\sqrt{x^t A x}\sqrt{x'^t A x'}} - b] \\ &\leq y[R_{-1}(x^t A x') - b] \end{aligned}$$

Substituting $\mathcal{A}$ to $y[\frac{x^t A x'}{\sqrt{x^t A x}\sqrt{x'^t A x'}} - b]$, we arrive at:

$$\begin{cases} R_{+1}(x^t A x') - b \leq \mathcal{A} \leq R_{-1}(x^t A x') - b & \text{if } y = 1 \\ R_{-1}(x^t A x') - b \leq \mathcal{A} \leq R_{+1}(x^t A x') - b & \text{if } y = -1 \end{cases}$$

Hence, matrix-threshold pairs (A, b) such that:

$$(R_{+1}(x^t A x') - b) \geq \gamma \wedge (b - R_{-1}(x^t A x') \geq \gamma$$

will have zero loss. The two inequalities correspond to the two different types of data we are dealing with: similar examples $(y = +1)$, and dissimilar examples $(y = -1)$. Using the aforementioned inequalities, we can define two subsets of C_τ on which the current matrix-threshold pair should be projected according to the value of y:

$$\begin{aligned} C'^{+}_\tau &= \{(A, b) \in \mathbb{R}^{n^2+1} : R_{+1}(x^t A x') - b \geq \gamma\} \quad \text{if } y = 1 \\ C'^{-}_\tau &= \{(A, b) \in \mathbb{R}^{n^2+1} : -R_{-1}(x^t A x') + b \geq \gamma\} \quad \text{if } y = -1 \end{aligned}$$

which can be conveniently rewritten:

$$C'^{y}_\tau = \{(A, b) \in \mathbb{R}^{n^2+1} : yR_y(x^t A x') - yb \geq \gamma\},\ y \in \{-1, +1\}$$

The orthogonal projection of (A_τ, b_τ) (the current matrix-threshold pair) on C'^{y}_τ, i.e. the closest element from (A_τ, b_τ) in C'^{y}_τ, takes the form:

$$\begin{cases} A_{\hat{\tau}} \quad = \quad A_\tau + ya(xx'^t), \text{ with } a \in \mathbb{R}, yR_y(x^t A_{\hat{\tau}} x') - yb = \gamma \\ b_{\hat{\tau}} \quad = \quad b_\tau + ya \end{cases}$$

where

$$a = \frac{\gamma - yR_y(x^t A_\tau x') + yb}{R_y(||x||^2||x'||^2)}$$

B. Projection onto C_a

In order to describe the projection onto C_a, we can note that $A_{\tau+1}$ is the projection of $A_{\hat{\tau}}$ onto the set of all positive, semi-definite matrices, and $b_{\tau+1}$ the one of $b_{\hat{\tau}}$ onto the set $b \in \mathbb{R} : -1+\gamma \leq b \leq 1-\gamma$.

In order to project $A_{\hat{\tau}}$ onto the set of all positive, semi-definite matrices, we use the decomposition, $A_{\hat{\tau}} = \sum_j \lambda_j u_j u_j^T$, where λ_j and u_j are the eigenvalues and eigenvectors of the matrix $A_{\hat{\tau}}$. The matrix $A_{\tau+1}$ is the projection of $A_{\hat{\tau}}$ onto the set of PSD matrices (see for example [13]). Knowing the eigenvalues and eigenvectors of $A_{\hat{\tau}}$, we can write $A_{\tau+1}$ as:

$$A_{\tau+1} = \sum_{j,\lambda_j>0} \lambda_j u_j u_j^T$$

If the matrix $A_{\hat{\tau}}$ is already symmetric, we use symmetric Householder reduction to convert it into a tridiagonal matrix followed by QR transformation. On the contrary, we convert it to Hessenberg form before converting to real Schur form. These forms make it easier to find the eigenvalues and the eigenvectors. Template Numerical Toolkit *TNT* [1] was used to find the eigenvalues and eigenvectors for the projections. Alternatively, one can also use Lanczos method [13] along with symmetric tridiagonal QR algorithm or bisection method to find the eigenvalues and the eigenvectors of $A_{\hat{\tau}}$.

C. Algorithm

We denote the algorithm based on the above method as **gCosLA** for Generalized Cosine Learning Algorithm. The update rule consists of projecting the matrix A onto the set of PSD matrices. For each example (in the form of a pair), the loss is calculated based on the similarity s_A. The update is performed only in case the loss is greater than zero.

---

gCosLA - Training

Input: training set of the form (x, x', y), of n vectors in $\mathbb{R}^p$, number of epochs M; b represents the threshold

Output: list of $(p \times p)$ matrices $((A_1, b_1), \cdots, (A_q, b_q))$

Initialization $t = 1, A_1 = I$ (identity matrix), $b = 0$, $\gamma > 0$

Repeat M **times (epochs)**

for $i = 1, \cdots, n$

get triplet $(x, x', \pm 1) \in R^n \times R^n$

$l_\tau(A, b) = max\{0, y(b - s_A(x, x')) + \gamma\}$

if $(l_\tau(A, b) > 0)$

$\frac{1}{R_{+1}} = max[(x^t Ax), (x'^t Ax')]$

$\frac{1}{R_{-1}} = min[(x^t Ax), (x'^t Ax')]$

$a = \frac{\gamma - yR_y(x^t A_\tau x') + yb}{R_y(||x||^2||x'||^2)}$

$A_{\hat{\tau}} = A_\tau + ya(xx'^t)$

$A_{\tau+1} = \sum_{j,\lambda_j>0} \lambda_j u_j u_j^T$ (where λ_j and u_j are the eigenvalues and eigenvectors of matrix $A_{\hat{\tau}}$)

$b_{\hat{\tau}} = b_\tau + ya$

if $(b_{\hat{\tau}} > 0)$

$b_{\tau+1} = min(b_{\hat{\tau}}, 1 - \gamma)$

else

$b_{\tau+1} = max(b_{\hat{\tau}}, -1 + \gamma)$

---

[1] Can be obtained from http://math.nist.gov/tnt/index.html

The above algorithm assumes that the data is separable. In the case where the data is inseparable, the loss becomes non-zero, which can be dealt with by introducing a new parameter γ_1 which is used to decrease the previously introduced margin γ (this affects only the projection onto C_τ, the projection onto C_a being left unchanged). The set C_τ thus becomes:

$$C_\tau = \{(A, b) \in \mathbb{R}^{n^2+1} : y[\frac{x^t Ax'}{\sqrt{x^t Ax}\sqrt{x'^t Ax'}} - b] \geq \gamma - \gamma_1\}$$

Setting $\beta = \gamma - \gamma_1$ leads to:

$$C'^y_\tau = \{(A, b) \in \mathbb{R}^{n^2+1} : yR_y(x^t Ax') + yb \geq \beta\},\ y \in \{-1, +1\}$$

This finally yields the modified value for a:

$$a = \frac{\beta - yR_y(x^t A_\tau x') + yb}{R_y(||x||^2||x'||^2)}$$

However, the rest of the algorithm remains the same.

IV. Analysis

The following theorem provides a loss bound for the algorithm *gCosLA* in the separable case. It assumes the existence of a PSD matrix A which separates the data in a strict sense, as well as the existence of an upper bound on the scalar product between all basic instance pairs. The inseparable case is treated in exactly the same way by replacing the positive real number γ with an arbitrary real number, not necessarily positive, β.

Theorem 1: Let $(x_1, x'_1, y_1), \cdots (x_\tau, x'_\tau, y_\tau), \cdots$ be a sequence of examples. For any PSD matrix A, let:

$$\frac{1}{R_{-1}} = min[{x_i}^t Ax_i, {x'_i}^t Ax'_i],\ 1 \leq i$$

and

$$\frac{1}{R_{+1}} = max[{x_i}^t Ax_i, {x'_i}^t Ax'_i],\ 1 \leq i$$

Assume that there exists a positive, semi-definite matrix A, a real number b and a positive real number γ such that:

$$(R_{+1}(x^t Ax') - b) \geq \gamma \wedge (b - R_{-1}(x^t Ax') \geq \gamma$$

Using the notations introduced previously, let $R \in \mathbb{R}^+$ be an upper bound such that:

$$R_y||x||^2||x'||^2 \leq R,\ y \in \{-1, +1\}$$

Then the following bound holds for any $M \geq 1$:

$$\sum_{\tau=1}^{M}(l_\tau(A,b))^2 \leq R\left(||A-I||_2^2+(b)^2\right)$$

A proof of theorem 1 can be established along the same lines as the proof of the loss bound provided for the *POLA* algorithm in [4]. As it is mainly technical, we omit it here. In [4], the only requirement made is the fact that the data should lie in a sphere of radius R. This requirement is translated in the case of a generalized cosine similarity by the fact that the scalar product between data points, normalized by its maximum or minimum values, is bounded. This leads to a stricter notion of separation. It however allows one to rely on simple projections.

As the inseparable case can be treated in exactly the same way, by directly replacing the positive scalar γ by β, a scalar not necessarily positive, one can see that the condition imposed is not really restrictive, and leads to an algorithm with an explicit bound on the loss function.

V. Experimental Validation

We used nine different datasets to assess *gCosLA* which are part of the UCI database ([14]), namely, *Ionosphere*, *Iris*, *Wine*, *Balance*, *Soybean (Small)*, *Glass Identification*, *Pima Indians Diabetes BUPA liver Disorders* and *Letter Recognition*. These are standard collections which have been used by different research communities (machine learning, pattern recognition, statistics etc.). The information about the datasets is summarized in the table I.

In order to create pairs of examples, we found 5 nearest neighbors for each example from the class it belongs to. Additionally, the same number of nearest neighbors from different classes was also found. We used a batch version of *gCosLA*, based on the approach described by [15], based on the average of the different matrices learnt: $A = \frac{\sum_{l=1}^{n} A_l}{n}$. We futhermore used two classification rules: the standard *kNN* rule and a symmetric version of it which consists in computing the *k*-nearest neighbors in each class and in selecting the class with the highest overall similarity.

5-fold nested cross-validation was used to learn the matrix sequence $(A_1, \cdots, A_n)$ for all of the datasets. The number of iterations and the number of matrices to be used was also learnt. As already stated, in a sequence of hypothesis, the last q elements may be more interesting than the earlier ones. In order to determine the optimum number of epochs and the value of q, a validation set was used. Another validation set was employed to determine the best value of β. 20 percent of the data was used for test purpose for each of the dataset. Of the remaining data, 70 percent was used for learning whereas 15 percent each for the two validation sets. It was observed that for each dataset, the best value of β is different. A greedy strategy was used, by generating a large number of matrices and ultimately selecting the one which produces maximum accuracy on the first validation set.

Combining the prediction rules mentioned above, with our algorithm, we get four different possibilities for comparison:

1) Standard *k*-NN rule with the cosine similarity by replacing A matrix by the Identity matrix. This rule is refered to as *kNN-cos*,
2) Standard *k*-NN rule with the similarity, based on the general cosine as in the equation 1, learned with the algorithm *gCosLA*. This method is termed as *kNN-A*,
3) The symmetric prediction rule with the cosine similarity having $A = I$, which we call as *SkNN-cos*,
4) Symmetric prediction rule with the similarity, based on the general cosine given in the equation 1, learned with *gCosLA*. This method appears as *SkNN-A*.

Unless otherwise stated, we used a binary version of *gCosLA*, in which a sequence of matrices is learned for each class (*one vs others*), and assessed the quality of a given method with its average accuracy (i.e. the accuracy averaged over the different classes).

A. Overall Results

We compared the four methods we retained on all of our collections. The comparison between *gCosLA* and cosine is given in table II. It is pertinent to mention here that we give only the best results obtained over k=1 and k=3. We can see that *gCosLA* performs much better than cosine on *Balance* (gain of 1.7%), *Wine* (gain of 2.4%), *Ionosphere* (gain of 3.6%) and *Liver* (better by 4.5%). The performance of all of the methods is comparable for *Iris*, *Letter*, *Soybean* and *Glass*. We can also observe that the symmetric version of *kNN* perform better than the standard *kNN*. Furthermore, it can be noted that *SkNN-A* method along with *gCosLA* algorithm has the best accuracy.

B. Comparison with previous approaches

We give here a detailed comparison between our approach and several state of the art methods. The first two learn similarity [7], [9] whereas the next three are interested in learning distances with kNN algorithm [2], [16], [5]. The algorithms are: Similarity Learning Algorithm *SiLA*, Similarity Learning with Neural Network *SNN*, Information Theoretic Metric Learning *ITML*, Maximally Collapsing Metric Learning *MCML*, Large Margin Nearest Neighbor *LMNN* and a multiclass version of SVMs. *SiLA* [9] also learns a similarity matrix which can be diagonal, symmetric or asymmetric. However, the normalization is independant of the similarity matrix learnt. They have used voted perceptron of [17] which cannot be used in our case. The similarity used in *SNN* is always positive, unlike *gCosLA*, owing to the use of sigmoidal function. *ITML* uses an information-theoretic approach based on Stein's loss to learn the Mahalanobis metric, where it imposes constraints on the distances between examples, constraints that can be

	Iris	**Wine**	**Balance**	**Ionosphere**	**Glass**	**Soybean**	**Pima**	**Liver**	**Letter**
Learning ex.	84	100	350	196	120	26	430	193	11200
Validation ex.	36	43	150	81	52	12	185	83	4800
Test ex.	30	35	125	70	42	9	153	69	4000
Classes	3	3	3	2	6	4	2	2	26
Features	4	13	4	34	9	35	8	6	16

Table I
CHARACTERISTICS OF DATASETS

	kNN-cos	kNN-A(*SiLA*)	kNN-A(*gCosLA*)	SkNN-cos	SkNN-A (*SiLA*)	SkNN-A (*gCosLA*)
Soybean	**1.0**	0.994	**1.0**	0.989	0.989	**1.0**
Iris	0.987	0.987	0.987	0.982	0.987	**0.989**
Letter	**0.997**	0.962	0.994	**0.997**	0.962	0.995
Balance	0.959	0.979	**0.986**	0.969	0.983	**0.986**
Wine	0.905	0.916	**0.933**	0.909	0.916	**0.933**
Ionosphere	0.871	**0.911**	0.907	0.871	**0.911**	0.907
Glass	0.902	0.902	**0.905**	0.902	0.902	**0.905**
Pima	0.652	0.647	0.664	0.665	**0.678**	0.665
Liver	0.632	0.609	0.691	0.658	0.609	**0.703**

Table II
RESULTS FOR ALL COLLECTIONS

relaxed for finding an admissible solution. *MCML* uses a conditional distribution over data points which reflects the fact that, ideally, points in the same class should have a high conditional probability (the class is collapsed), whereas points from different classes should have a low one (e.g. 0). Looking for the probability distribution which minimizes the Kullback-Leibler divergence with the ideal distribution leads to a convex optimization problem, the solution of which involves computing the eigen-decomposition of the probability matrix. *LMNN* proposes the use of semi-definite programming for learning a Mahanalobis distance that ensures a separation with a certain margin between the examples. To compare our method based on *gCosLA* with previous approaches, we used a multiclass version and computed the global accuracy. The results of this comparison are given in table III. We compare the methods on three UCI datasets common to all of the approaches.

Comparing *gCosLA* with *SiLA*, we can see that for *Balance* and *Wine*, *gCosLA* not only outperformed *SiLA* but it also required much fewer iterations for convergence. The performance for *gCosLA* is on par with that of *SiLA* on *Iris*. We also compared the binary version of *gCosLA* with that of *SiLA* (part of table II). *gCosLA* performed significantly better than *SiLA* on *Wine* (93.3% vs 91.6%), *Liver* (70.3% vs 60.9%) and *Letter* (99.5% vs 96.2%) data sets with both *kNN-A* and *SkNN-A*. Furthermore, we can observe that the performance of *gCosLA* is on par with *SiLA* for the rest of the datasets. However *gCosLA* converged faster as compared with *SiLA* for all of these datasets.

While comparing *gCosLA* with *SNN*, we note that our method outperformed *SNN* for *Balance* and *Iris*. However for *Wine*, *SNN* has got a better performance as compared with our algorithm. The primary reason for this is that *SNN* was able to down-weigh an influential attribute for *Wine* whereas our method was unable to do so, since we do not perform feature selection while *SNN* does so.

gCosLA performed much better than *MCML* and *Multi-class SVM* for all of the three collections considered. Our method also outperformed *LMNN* and *ITML* on two out of three data sets, namely *Balance* and *Iris*. However they performed better than *gCosLA* on *Wine* because they were able to down-weigh an influential attribute like *SNN*.

VI. CONCLUSION

We have developed in this paper an algorithm, *gCosLA*, to learn generalized cosine similarity measures, for both online and batch modes. Because generalized cosine similarities are based on scalar products, they involve bilinear forms defined by positive, semi-definite matrices. However, the normalization introduced in the cosine similarity prevents one from directly re-using algorithms previously introduced for learning say Mahalanobis distances, also based on PSD matrices. We have followed in this work the approach defined in [4], which consists in first finding an appropriate matrix and then projecting it on the set of PSD matrices, which we have adapted to the particular form of generalized cosine similarities, and more particularly to the fact that such measures are normalized. We have then experimentally tested our algorithm on several standard collections from the UCI database.

We also compared the binary version of *gCosLA* with that of *SiLA*. The performance of *gCosLA* is significantly

	gCosLA (kNN-A)	SiLA (kNN-A)	SNN	MCML	LMNN	ITML	Multiclass SVM
Balance	**0.976**	0.952	0.879	0.925	0.916	0.920	0.922
Wine	0.880	0.863	0.951	0.837	**0.974**	**0.974**	0.801
Iris	0.973	**0.982**	0.934	0.967	0.953	0.961	0.956

Table III
DIFFERENT SIMILARITY AND METRIC LEARNING ALGORITHMS ON UCI DATASETS

better than that of *SiLA* on *Wine* (93.3% vs 91.6%), *Liver* (70.3% vs 60.9%) and *Letter* (99.5% vs 96.2%) data sets with both *kNN-A* and *SkNN-A*. Furthermore, we observed that the performance of *gCosLA* is on par with *SiLA* for the rest of the data sets. However *gCosLA* converged faster as compared with *SiLA* for all of these datasets.

We have also compared a multiclass version of our algorithm with other state of the art approaches on *Balance*, *Iris* and *Wine* (only 3 datasets were chosen since these were common to all methods). Comparing *gCosLA* with *SiLA*, we can see that for *Balance* and *Wine*, *gCosLA* not only outperformed *SiLA* but it also converged very rapidly. The performance for *gCosLA* is on par with that of *SiLA* on *Iris* but nevertheless, *gCosLA* is faster. While comparing *gCosLA* with *SNN*, we note that our method outperformed *SNN* for *Balance* and *Iris*. However for *Wine*, *SNN* has got a better performance as compared with our algorithm. The primary reason for this is that *SNN* was able to down-weigh an influential attribute for *Wine* whereas our method was unable to do so, since we do not perform feature selection while *SNN* does so. Our method also outperformed *LMNN* and *ITML* on two out of three data sets, namely *Balance* and *Iris*. However they performed better than *gCosLA* on *Wine* because they were able to down-weigh an influential attribute like *SNN*. *gCosLA* performed much better than *MCML* and *Multiclass SVM* for all of the three collections considered.

Since dot products can be easily expressed with kernels, *gCosLA* can be extended to employ kernels, which we plan to investigate in the future. Feature selection can also be incorporated to improve the performance of our algorithm.

REFERENCES

[1] L. Baoli, L. Qin, and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing (TALIP)*, 2004.

[2] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[3] L. R. M. Diligenti, M. Maggini, "Learning similarities for text documents using neural networks," in *Proceedings of IAPR – TC3 International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, 2003.

[4] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004.

[5] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, Feb 2009.

[6] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based classification: Concepts and algorithms," *Journal of Machine Learning Research*, vol. 10, pp. 747–776, March 2009.

[7] S. Melacci, L. Sarti, M. Maggini, and M. Bianchini, "A neural network approach to similarity learning," in *ANNPR*, ser. Lecture Notes in Computer Science, L. Prevost, S. Marinai, and F. Schwenker, Eds., vol. 5064. Springer, 2008, pp. 133–136.

[8] M. Peterson, T. Doom, and M. Raymer, "Ga-facilitated knn classifier optimization with varying similarity measures," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 3, Sept. 2005, pp. 2514–2521 Vol. 3.

[9] A. M. Qamar, É. Gaussier, J.-P. Chevallet, and J.-H. Lim, "Similarity learning for nearest neighbor classification," in *ICDM*, 2008, pp. 983–988.

[10] J.-P. Bao, J.-Y. Shen, X.-D. Liu, and H.-Y. Liu, "Quick asymmetric text similarity measures," *International Conference on Machine Learning and Cybernetics*, 2003.

[11] M. Grabowski and A. Szałas, "A technique for learning similarities on complex structures with applications to extracting ontologies," in *Proceedings of the 3rd Atlantic Web Intelligence Conf. 2005*, ser. LNAI. Springer Verlag, 2000.

[12] A. Hust, "Learning Similarities for Collaborative Information Retrieval," in *Proceedings of KI 2004 workshop "Machine Learning and Interaction for Text-Based Information Retrieval", TIR-04*, September 2004, pp. 43–54.

[13] G. Golub and C. V. Loan, *Matrix Computations (2nd ed.)*. John Hopkins University Press, 1989.

[14] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[15] O. Dekel and Y. Singer, "Data-driven online to batch conversions," in *NIPS*, 2005.

[16] A. Globerson and S. Roweis, "Metric learning by collapsing classes," in *NIPS*, 2005.

[17] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, no. 3, 1999.