International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 445-458
© MIR Labs, www.mirlabs.net/ijcisim/index.html

RELIEF Algorithm and Similarity Learning for k-NN

Ali Mustafa Qamar[1] **and Eric Gaussier**[2]

[1] Assistant Professor, Department of Computing
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Islamabad, Pakistan
mustafa.qamar@seecs.edu.pk

[2]Laboratoire d'Informatique de Grenoble
Université de Grenoble
France
eric.gaussier@imag.fr

Abstract: **In this paper, we study the links between *RELIEF*, a well-known feature re-weighting algorithm and *SiLA*, a similarity learning algorithm. On one hand, *SiLA* is interested in directly reducing the leave-one-out error or $0-1$ loss by reducing the number of mistakes on unseen examples. On the other hand, it has been shown that *RELIEF* could be seen as a distance learning algorithm in which a linear utility function with maximum margin was optimized. We first propose here a version of this algorithm for similarity learning, called *RBS* (for *RELIEF*-Based Similarity learning). As *RELIEF*, and unlike *SiLA*, *RBS* does not try to optimize the leave-one-out error or $0-1$ loss, and does not perform very well in practice, as we illustrate on several UCI collections. We thus introduce a stricter version of *RBS*, called *sRBS*, aiming at relying on a cost function closer to the $0-1$ loss. Moreover, we also developed Positive, semi-definite (PSD) versions of *RBS* and *sRBS* algorithms, where the learned similarity matrix is projected onto the set of PSD matrices. Experiments conducted on several datasets illustrate the different behaviors of these algorithms for learning similarities for *kNN* classification. The results indicate in particular that the $0-1$ loss is a more appropriate cost function than the one implicitly used by *RELIEF*. Furthermore, the projection onto the set of PSD matrices improves the results for *RELIEF* algorithm only.**

Keywords: similarity learning, *RELIEF* algorithm, positive, semi-definite (PSD) matrices, *SiLA* algorithm, *kNN* classification, machine learning

I. Introduction

The k nearest neighbor (*kNN*) algorithm is a simple yet efficient classification algorithm: to classify an example x, it finds its k nearest neighbors based on the distance or similarity metric, from a set of already classified examples and assigns x to the most represented class in the set of these nearest neighbors. Many people have improved the performance of kNN algorithm by learning the underlying geometry of the space containing the data e.g. learning a Mahalanobis distance instead of the standard Euclidean one. This has paved the way for a new reasearch theme termed *metric learning*. Most of the people working in this research area are more interested in learning a distance metric (see e.g. [1, 2, 3, 4]) as compared to a similarity one. However, as argued by several researchers, similarities should be preferred over distances on some of the data sets. Similarity is usually preferred over the distance metric while dealing with text, in which case the cosine similarity has been deemed more appropriate as compared to the various distance metrics. Furthermore, studies reported in [5, 6, 7, 8] have proved that cosine should be preferred over the Euclidean distance over non-textual data sets as well. Furthermore, cosine similarity has been compared with the Euclidean distance on 15 different datasets. Umugwaneza and Zou [9] have combined cosine similarity and Euclidean distance for Trademarks retrieval whereby they fine tune the proportions for each of the two measures. Similarly Porwik et al. [10] have compared many different similarity and distance measures such as Euclidean distance, Soergel distance, cosine similarity, Jaccard and Dice coefficients etc.

RELIEF (originally proposed by Kira and Rendell [11]) is an online feature reweighting algorithm successfully employed in various different settings. It learns a vector of weights for each of the different features or attributes describing their importance. It has been proved by Sun and Wu [12] that it implicitly aims at maximizing the margin of a linear utility function.

SiLA [8] is a similarity metric learning algorithm for nearest neighbor classification. It aims at moving the nearest neighbors belonging to the same class nearer to the input example (termed as *target* neighbors) while pushing away the nearest examples belonging to different classes (described as *impostors*). The similarity function between two examples x and y can be written as:

$$s_A(x,y) = \frac{x^t A y}{N(x,y)} \tag{1}$$

where t represents the transpose, A is a $(p \times p)$ similarity matrix and $N(x,y)$ is a normalization function which depends on x and y (this normalization is typically used to map the similarity function to a particular interval, as [0, 1]). Equation 1 generalizes several standard similarity functions e.g. the cosine measure which is widely used in text retrieval, is obtained by setting the A matrix to the identity matrix I, and $N(x, y)$ to the product of the L2 norms of x and y. The aim here is to reduce the $0 - 1$ loss which is dependent on the number of mistakes made during the classification phase. For the remainder of the paper, a matrix is sometimes represented as a vector as well (e.g. a $p \times p$ matrix can be represented by a vector having p^2 elements).

The rest of the paper is organized as follows: Section 2 describes the *SiLA* algorithm. This is followed by a short introduction of *RELIEF* algorithm along with its mathematical interpretation, its comparison with $\hat{\text{S}}$iLA and a *RELIEF*-Based Similarity learning algorithm (*RBS*) in section 3. Section 4 introduces a strict version of *RBS* followed by the experimental results and conclusion.

II. *SiLA* - A Similarity Learning Algorithm

The *SiLA* algorithm is described in detail here. It is a similarity algorithm and is a variant of the voted perceptron algorithm of Freund and Schapire [13], later used in Collins [14].

SiLA - Training (k=1)

Input: training set $((x^{(1)}, c^{(1)}), \cdots, (x^{(n)}, c^{(n)}))$ of n vectors in R^p, number of epochs J; A_{ml} denotes the element of A at row m and column l

Output: list of weighted $(p \times p)$ matrices $((A_1, w_1), \cdots, (A_q, w_q))$

```
Initialization $t = 1, A^{(1)} = 0$ (null matrix), $w_1 = 0$
Repeat $J$ times (epochs)
  1. for $i = 1, \cdots, n$
    2. if $s_A(x^{(i)}, y) - s_A(x^{(i)}, z) \leq 0$
      3. $\forall (m, l), 1 \leq m, l \leq p$,
         $A_{ml}^{(t+1)} = A_{ml}^{(t)} + f_{ml}(x^{(i)}, y) - f_{ml}(x^{(i)}, z)$
      4. $w_{t+1} = 1$
      5. $t = t + 1$
    6. else
      7. $w_t = w_t + 1$
```

Whenever an example $x^{(i)}$ is not separated from differently labeled examples, the current A matrix is updated by the difference between the coordinates of the target neighbors (denoted by y) and the impostors (represented by z) as described in line 4 of the algorithm. This corresponds to the standard perceptron update. Similarly, when current A correctly classifies the example under focus, then its weight is increased by 1, so that the weights finally correspond to the number of examples correctly classified by the similarity matrix over the different epochs.

The worst-time complexity of *SiLA* is $0(Jnp^2)$ where J stands for the number of iterations, n is the number of training examples while p stands for the number of dimensions or attributes.

The functions f_{ml} allows to learn different types of matrices and therefore different types of similarities: in the case of a diagonal matrix, $f_{ml}(x, y) = \frac{\delta(m, l) x_m^t y_l}{\text{N}(x,y)}$ (with δ the Kronecker symbol), for a symmetric matrix, $f_{ml}(x, y) = \frac{x_m^t y_l + x_l^t y_m}{\text{N}(x,y)}$, and for a square matrix (and hence, potentially, an asymmetric similarity), $f_{ml}(x, y) = \frac{x_m^t y_l}{\text{N}(x,y)}$.

III. RELIEF and its mathematical interpretation

Sun and Wu [12] have shown shown that the *RELIEF* algorithm solves convex optimization problem while maximizing a margin-based objective function employing the *kNN* algorithm. It learns a vector of weights for each of the features, based on the nearest *hit* (nearest example belonging to the class under consideration, also known as the nearest *target* neighbor) and the nearest *miss* (nearest example belonging to other classes, also known the the nearest *impostor*).

In the original setting for the *RELIEF* algorithm, it only learns a diagonal matrix. However, Sun and Wu [12] have learned a full distance metric matrix and have also proved that *RELIEF* is basically an online algorithm.

In order to describe the *RELIEF* algorithm, we suppose that $x^{(i)}$ is a vector in R^p with $y^{(i)}$ as the corresponding class label with values $+1, -1$. Furthermore, let A be a vector of weights initialized with 0. The weight vector learns the qualities of the various attributes. A is learned on a set of training examples. Suppose an example $x^{(i)}$ is randomly selected. Then two nearest neighbors of $x^{(i)}$ are found: one from the same class (termed as the *nearest hit* or H) while the second one from a class other than that of $x^{(i)}$ (termed as the *nearest miss* or M). The update rule in case of *RELIEF* does not depend on any condition unlike *SiLA*.

The *RELIEF* algorithm is presented next:

RELIEF (k=1)

Input: training set $((x^{(1)}, c^{(1)}), \cdots, (x^{(n)}, c^{(n)}))$ of n vectors in R^p, number of epochs J;

Output: the vector A of estimations of the qualities of attributes

```
Initialization $1 \leq m \leq p, A_m = 0$
Repeat $J$ times (epochs)
  1. randomly select an instance $x^{(i)}$
  2. find nearest hit $H$ and nearest miss $M$
  3. for $l = 1, \cdots, p$
  4. $A_l = A_l - \frac{\text{diff}(l, x^{(i)}, H)}{J} + \frac{\text{diff}(l, x^{(i)}, M)}{J}$
```

where J represents the number of times *RELIEF* has been executed, while *diff* finds the difference between the values of an attribute l for the current example $x^{(i)}$ and the nearest hit H or the nearest miss M. If an instance $x^{(i)}$ and its nearest hit, H have different values for an attribute l, then this means that it separates the two instances in the same class which is not desirable, so the quality estimation A_l is decreased. On the other hand, if the instances $x^{(i)}$ and M have different values for an attribute l then this attribute separates two instances pertaining to different classes which is desirable, so the quality estimation A_l is increased. In the case of discrete attributes, the value of difference is either 1 (the values are different) or 0 (the values are the same). However, for continuous attributes, the difference is the actual difference normalized to the closed interval $[0,1]$ which is given by the following equation:

$$\text{diff}(l,x,x') = \frac{|x_l - x'_l|}{max(l) - min(l)}$$

The complexity of *RELIEF* algorithm can be given as $O(Jpn)$. Moreover, the complexity is fixed for all of the scenarios unlike *SiLA*.

In the original setting, *RELIEF* can only deal with binary class problems and cannot work with incomplete data. As a work around, *RELIEF* was extended to *RELIEFF* algorithm [15]. Rather than just finding the nearest hit and miss, it finds k nearest hits and the same number of nearest misses from each of the different classes.

A. Mathematical Interpretation of RELIEF algorithm

Sun and Wu [12] have given a mathematical interpretation for the *RELIEF* algorithm. The margin for an instance $x^{(i)}$ can be defined in the following manner:

$$p = d(x^{(i)} - M(x^{(i)})) - d(x^{(i)} - H(x^{(i)}))$$

where $M(x^{(i)})$ and $H(x^{(i)})$ represent the nearest miss and the nearest hit for $x^{(i)}$ respectively, and $d(.)$ represents a distance function which is defined as $d(x) = \sum_l^p |x_l|$ in a similar fashion as the original *RELIEF* algorithm. The margin is greater than 0 if and only if $x^{(i)}$ is closer to the nearest hit as compared to the nearest miss, or in other words, is classified correctly as per the $1NN$ rule. The aim here is to scale each feature so that the leave-one-out error $\sum_{i=1}^n I(p_i(A) < 0)$ is minimized, where I(.) is the indicator function and $p_i(A)$ is the margin of $x^{(i)}$ with respect to A. As the indicator function is not differentiable, a linear utility function is used instead so that the averaged margin in the weighted feature space is maximized:

$$\begin{aligned}&\max_A \textstyle\sum_{i=1}^n p_i(A) = \sum_{i=1}^n \{\sum_{l=1}^p A_l \left| x_l^{(i)} - M^{(i)}(x_l)\right| \\ &- \textstyle\sum_{l=1}^p A_l \left| x_l^{(i)} - H^{(i)}(x_l)\right| \}, \\ &\text{subject to } \|A\|_2^2 = 1, \text{ and } A \geq 0,\end{aligned} \tag{2}$$

where $A \geq 0$ makes sure that the learned weight vector induces a distance measure. Equation 2 can be simplified by defining $z = \sum_{i=1}^n \left(|x^{(i)} - M(x^{(i)})| - |x^{(i)} - H(x^{(i)})|\right)$ which can be expressed as:

$$\max_A \; A^t z \;\; \text{subject to} \;\; \|A\|_2^2 = 1, \; A \geq 0$$

Taking the Lagrangian of the above equation, we get:

$$L = -A^t z + \lambda(\|A\|_2^2 + 1) + \sum_{l=1}^p \theta_i(-A_i)$$

where both λ and $\theta \geq 0$ are Lagrangian multipliers. In order to prove that the optimum solution can be calculated in a closed form, the following steps are performed: the derivative of L is taken with respect to A, before being set to zero:

$$\frac{\partial L}{\partial A} = -z + 2\lambda A - \theta = 0 \;\; \Rightarrow \;\; A = \frac{z+\theta}{2\lambda}$$

The values for λ and θ are deduced from the KKT (Karush-Kuhn-Tucker) condition giving way to:

$$A = \frac{(z)^+}{\|(z)^+\|_2} \tag{3}$$

where $(z)^+ = [max(z_1,0), \cdots, max(z_n,0)]^t$, z_i represents the margin of example i. If we compare the above equation with the weight update rule for *RELIEF*, it can be noted that *RELIEF* is an online algorithm which solves the optimization problem given in equation 2. This is true except when $A_l = 0$ for $z_l \leq 0$ which corresponds to the irrelevant features.

B. RELIEF-Based Similairty Learning Algorithm - RBS

In this subsection, a *RELIEF*-Based Similarity Learning algorithm (*RBS*) [16] is proposed which is based on *RELIEF* algorithm. However, the interest here is in similarities rather than distances.

Our aim here is to maximize the margin $\mathcal{M}(A)$ between *target* neighbors (represented by y) and *impostors* (represented by z). However, as the similarity defined through matrix A can be arbitrarily large through multiplication of A by a positive constant, we impose that the Frobenius norm of A be equal to 1. The margin, for $k=1$, in the *kNN* algorithm can be written as:

$$\begin{aligned}\mathcal{M}(A) &= \textstyle\sum_{i=1}^n \left(s_A(x^{(i)}, y^{(i)}) - s_A(x^{(i)}, z^{(i)})\right) \\ &= \textstyle\sum_{i=1}^n (x^{(i)^t} A y^{(i)} - x^{(i)^t} A z^{(i)}) \\ &= \textstyle\sum_{i=1}^n x^{(i)^t} A (y^{(i)} - z^{(i)})\end{aligned}$$

where A is the similarity matrix.

The optimization problem derived from the above considerations thus takes the form:

$$\begin{aligned}&\arg\max_A && \mathcal{M}(A) \\ &\text{subject to} && \|A\|_F^2 = 1,\end{aligned}$$

Taking the Lagrangian of the similarity matrix A:

$$\mathcal{L}(A) = \sum_{i=1}^n x^{(i)^t} A (y^{(i)} - z^{(i)}) + \lambda(1 - \sum_{l=1}^p \sum_{m=1}^p a_{lm}^2)$$

where λ is a Lagrangian multiplier. Moreover, after taking the derivative w.r.t. a_{lm} and setting it to zero, one obtains:

$$\begin{aligned}\frac{\partial \mathcal{L}(A)}{\partial a_{lm}} &= \textstyle\sum_{i=1}^n x_l^{(i)}(y_m^{(i)} - z_m^{(i)}) - 2\lambda a_{lm} = 0 \\ \Rightarrow a_{lm} &= \frac{\sum_{i=1}^n x_l^{(i)}(y_m^{(i)} - z_m^{(i)})}{2\lambda}\end{aligned}$$

Furthermore, since the Frobenius norm of matrix A is 1:

$$\begin{aligned}&\sum_{l=1}^p \sum_{m=1}^p a_{lm}^2 = 1 \\ &\Rightarrow \sum_{l=1}^p \sum_{m=1}^p a_{lm}^2 = \sum_{l=1}^p \sum_{m=1}^p \left(\frac{\sum_{i=1}^n x_l^{(i)}(y_m^{(i)} - z_m^{(i)})}{2\lambda}\right)^2\end{aligned}$$

leading to:

$$2\lambda = \sqrt{\sum_{l=1}^p \sum_{m=1}^p \left(\sum_{i=1}^n x_l^{(i)}(y_m^{(i)} - z_m^{(i)})\right)}$$

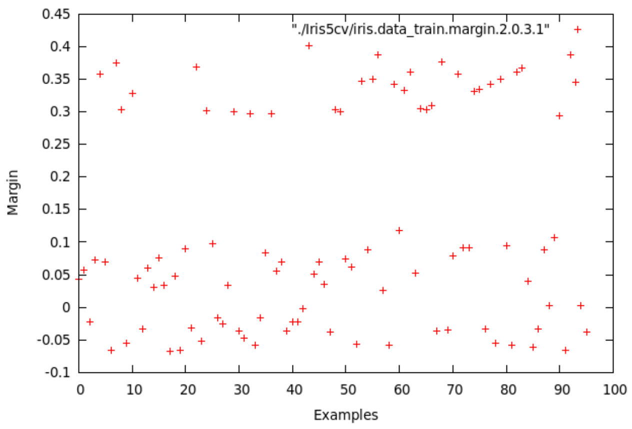

Figure. 1: Margin for *RELIEF*-Based similarity learning algorithm on *Iris* dataset

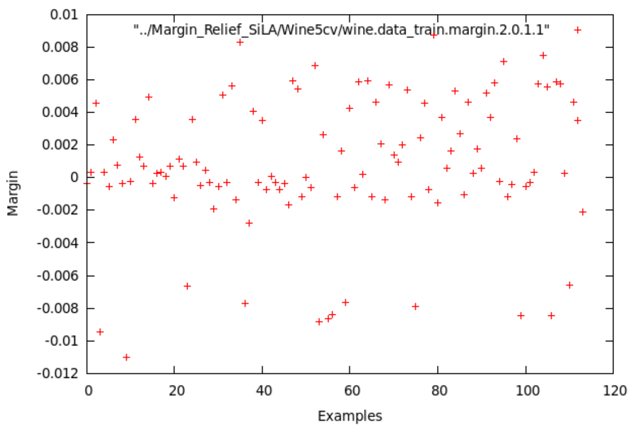

Figure. 2: Margin for *RELIEF*-Based similarity learning algorithm on *Wine* dataset

In case of a diagonal matrix, m is replaced with l. The margin for $k > 1$ can be written as:

$$\begin{aligned}\mathcal{M}(A) &= \sum_{i=1}^{n}\left(\sum_{q=1}^{k} s_A(x^{(i)}, y^{(i),q}) - \sum_{q=1}^{k} s_A(x^{(i)}, z^{(i),q})\right) \\ &= x^{(i)^t} A \sum_{q=1}^{k} (y^{(i),q} - z^{(i),q})\end{aligned}$$

where $y^{(i),q}$ represents the qth nearest neighbor of $x^{(i)}$ from the same class while $z^{(i),q}$ represents the qth nearest neighbor of $x^{(i)}$ from other classes. Furthermore, a_{lm} and 2λ can be written as:

$$a_{lm} = \frac{\sum_{i=1}^{n} x_l^{(i)} \sum_{q=1}^{k} (y_m^{(i),q} - z_m^{(i),q})}{2\lambda}$$

$$2\lambda = \sqrt{\sum_{l=1}^{p}\sum_{m=1}^{p}\left(\sum_{i=1}^{n} x_l^{(i)} \sum_{q=1}^{k} (y_m^{(i,q)} - z_m^{(i,q)})\right)}$$

The problem with the *RELIEF* based approaches (*RELIEF* and *RBS*) is that as one strives to maximize the margin, it is quite possible that the overall margin is quite large but in reality the algorithm has made a certain number of mistakes (characterized with negative margin). This concept was verified on a number of standard UCI datasets [17] e.g. *Iris*, *Wine*, *Balance* and *Pima* as can be seen from figure 1, 2, 3, 4 respectively. It can be observed from all of these figures that the average margin remains positive despite the presence of a number of mistakes, since the positive margin is much greater than the negative one for the majority of the test examples. For example, in figure 1, the values of negative margin is in the range of $0.05 - 0.10$ whereas most of the positive margin values are greater than 0.15. Similarly, for *Wine* (figure 2), most of the negative margin values lie in the range between 0 and -0.04 while the positive margin values are dispersed in the range $0 - 0.08$. So, despite the fact that the overall margin is large, a lot of examples are still misclassified. This explains why the algorithms *RELIEF* and *RBS* did not perform quite well on different standard test collections (see section VII).

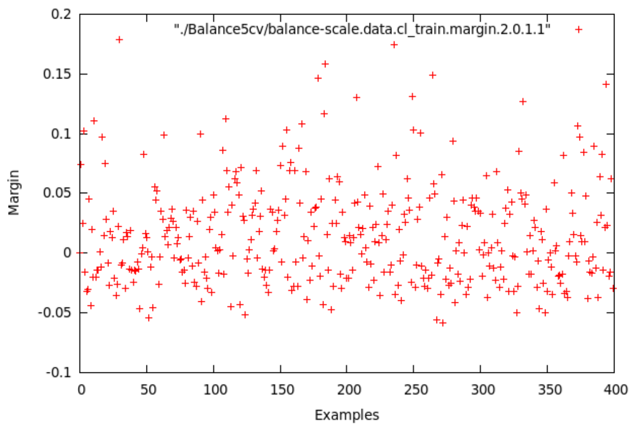

Figure. 3: Margin for *RELIEF*-Based similarity learning algorithm on *Balance* dataset

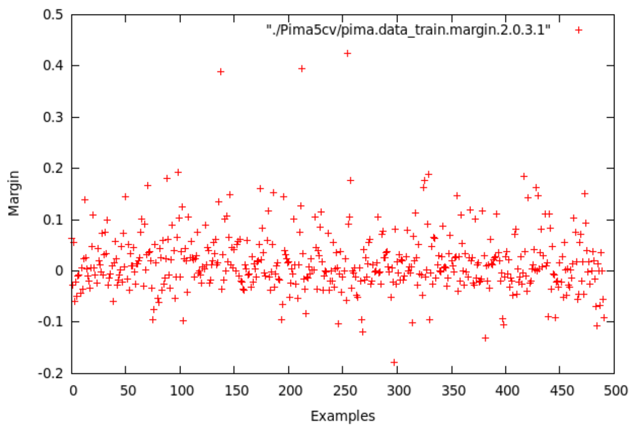

Figure. 4: Margin for *RELIEF*-Based similarity learning algorithm on *Pima* dataset

C. Comparison between SiLA and RELIEF

While comparing the two algorithms *SiLA* and *RELIEF*, it can be verified that *RELIEF* learns a vector of weights while *SiLA* learns a sequence of vectors where each vector has got a corresponding weight which signifies the number of examples correctly classified while using that particular vector. Furthermore, the weight vector is updated systematically in case of *RELIEF* while a vector is updated for *SiLA* if and

only if it has failed to correctly classify the current example $x^{(i)}$ (i.e. $s_A(x^{(i)},y) - s_A(x^{(i)},z) \leq 0$). In this case, a new vector A is created and its corresponding weight is set to 1. However, in case of correct classification for *SiLA*, the weight associated with the current A is incremented by 1. Moreover, the two algorithms find the nearest hit and the nearest miss to update A: *RELIEF* selects an instance randomly whereas *SiLA* uses the instances in a systematic way. Another difference between these two algorithms is that in case of *RELIEF*, the vector A is updated based on the difference (distance) while it is updated based on the similarity function for *SiLA*. This explains why the impact of nearest hit is subtracted for *RELIEF* while the impact for nearest miss is added to the vector A. For *SiLA*, the impact of the nearest hit is added while that of the nearest miss is subtracted from current A.
SiLA tries to directly reduce the leave-one-out error. However, *RELIEF* uses a linear utility function in such a way that the average margin is maximized.

IV. A stricter version: *sRBS*

A work around to improve the performance of *RELIEF* based methods is to directly use the leave-one-out error or $0-1$ loss like the original *SiLA* algorithm where the aim is to reduce the number of mistakes on unseen examples. The resulting algorithm is a stricter version of *RELIEF*-Based Similairty Learning Algorithm and is termed as *sRBS*. It is called as a *stricter* version as we do not try to maximize the overall margin but are interested in reducing the individual errors on the unseen examples.
The cost function for *sRBS* can be described in terms of a sigmoid function:

$$\sigma_A(x^{(i)}) = \left(\frac{1}{1+exp(\beta x^{(i)^t} A(y^{(i)} - z^{(i)})}\right)$$

As β approaches ∞, the sigmoid function starts representing the $0-1$ loss: it approaches 0 where the margin ($x^{(i)}A(y^{(i)} - z^{(i)}$) is positive and approaches 1 in the case where the margin is negative. Let $g_A(i)$ represents $exp(\beta x^{(i)^t} A(y^{(i)} - z^{(i)})$ while v represents $y-z$. The cost function we are considering is based on the above sigmoid function, regularized with the Frobenius norm of A:

$$\begin{aligned}
\arg\min_A \varepsilon(A) &= \sum_{i=1}^{n} \sigma_A(x^{(i)}) + \lambda \|A\|_2^2 \\
&= \sum_{i=1}^{n} \left[\frac{1}{1+g_A(i)} + \frac{\lambda}{n}\sum_{lm} a_{lm}^2\right] \quad (4) \\
&= \sum_{i=1}^{n} Q_i(A)
\end{aligned}$$

where λ is the regularization parameter. Taking the derivative of $\varepsilon(A)$ with respect to a_{lm}:

$$\begin{aligned}
\frac{\partial \varepsilon(A)}{\partial a_{lm}} &= -\beta \sum_{i=1}^{n} \frac{x_l^{(i)} v_m^{(i)} g_A(i)}{(1+g_A(i))^2} + 2\lambda a_{lm} \\
&= \sum_{i=1}^{n} \frac{\partial Q_i(A^t)}{\partial a_{lm}}
\end{aligned}$$

$\forall\ l, m, 1 \geq l \geq p, 1 \geq m \geq p$. Setting this derivative to 0 leads to:

$$2\lambda a_{lm} = -\beta \sum_{i=1}^{n} \frac{x_l^{(i)} v_m^{(i)} g_A(i)}{(1+g_A(i))^2}$$

We know of no closed form solution for this fixed point equation, which can however be solved with gradient descent methods, through the following update:

$$A_{lm}^{t+1} = A_{lm}^{t} - \frac{\alpha^t}{n} \sum_{i=1}^{n} \frac{\partial Q_i(A^t)}{\partial a_{lm}}$$

where α^t stands for the learning rate, is inversely proportional to time t and is given by: $\alpha^t = \frac{1}{t}$.

```
sRBS - Training
Input: training set $((x^{(1)},c^{(1)}),\cdots,(x^{(n)},c^{(n)}))$ of $n$ vectors in $R^p$, $A^1_{lm}$ denotes the element of $A^1$ at row $l$ and column $m$
Output: Matrix $A$
Initialization $t = 1, A^{(1)} = 1$ (Unity matrix)
Repeat $J$ times (epochs)
  1. For all of the features $l, m$
       2. Minus$_{lm} = 0$
  3. for $i = 1, \cdots, n$
     4. For all of the features $l, m$
       5. Minus$_{lm} += \frac{\partial Q_i(A^t)}{\partial a_{lm}}$
       6. $A^{t+1}_{lm} = A^t_{lm} - \frac{\alpha^t}{n} *$ Minus$_{lm}$
  7. If $\sum_{lm} |A^{t+1}_{lm} - A^t_{lm}| \leq \gamma$
       8. Stop
```

During each epoch, the difference between the new similarity matrix A_{lm}^{t+1} and the current one A_{lm}^t is computed. If this difference is less than a certain threshold (γ in this case), the algorithm is stopped. The range of γ is between 10^{-3} and 10^{-4}. Figure 5, 6, 7 and 8 show the margin values on the training data for *sRBS* for the datasets *Iris*, *Wine*, *Balance* and *Pima* respectively. Comparing these figures with the earlier ones (i.e. for *RBS*) reveals the importance of using a cost function closer to the 0-1 loss. One can see that the average margin is positive for most of the training examples for *sRBS*. There are only a very few errors although a lot of examples have a margin just slightly greater than zero.

V. Effect of Positive, Semi-Definitiveness on *RELIEF* based algorithms

The similarity x^tAx in the case of *RELIEF* based algorithms does not correspond to a symmetric bi-linear form, and hence a scalar product. In order to define a proper scalr product, and hence a cosine-like similarity, one can project the similarity matrix A onto the set of positive, semi-definite (PSD) matrices. A similarity matrix can be projected onto the set of PSD matrices by finding an eigenvector decomposition followed by the selection of positive eigenvalues. A PSD matrix A is written as:

$$A \succeq 0$$

In case, where a diagonal matrix is learned by *RELIEF*, positive semi-definitiveness can be achieved by selecting only the

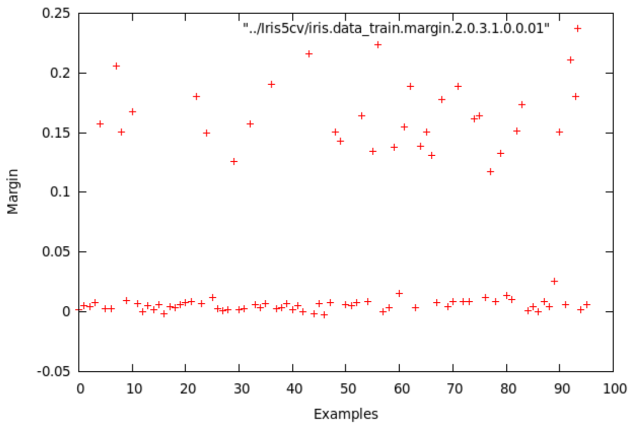

Figure. 5: Margin for *sRBS* on *Iris* dataset

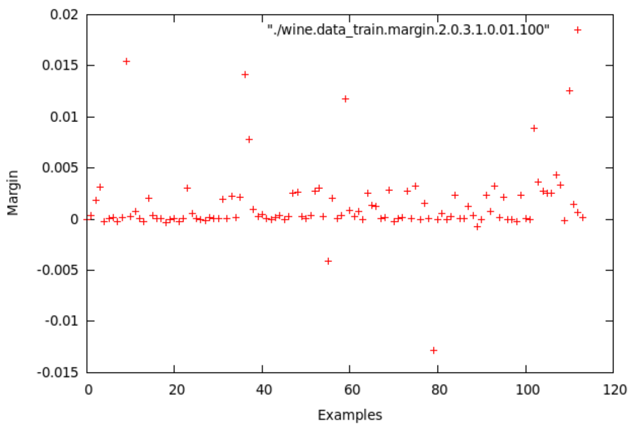

Figure. 6: Margin for *sRBS* on *Wine* dataset

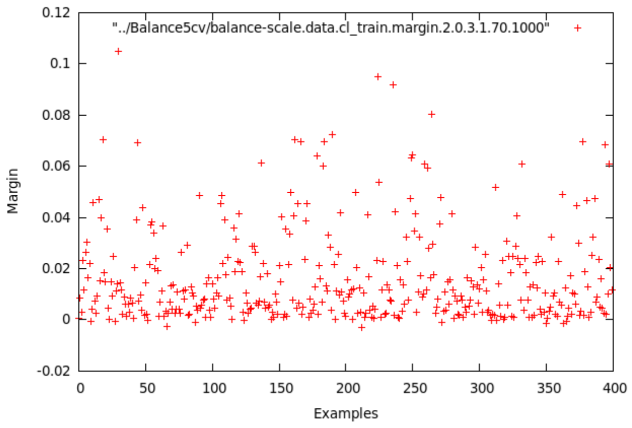

Figure. 7: Margin for *sRBS* on *Balance* dataset

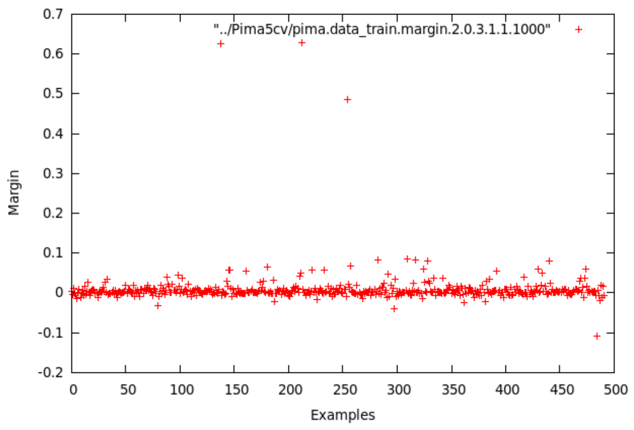

Figure. 8: Margin for *sRBS* on *Pima* dataset

positive entries of the diagonal. Moreover for learning a full matrix with *RELIEF*, the projection can be performed in the following manner:

$$A = \sum_{j,\lambda_j>0} \lambda_j u_j u_j^t$$

where λ_j and u_j are the eigenvalues and eigenvectors of A. In this case, only the positive eigenvalues are retained while the negative ones are discarded. It is important to note here that all eigenvalues of A may be negative, in which case the projection on the PSD cone will result on the null matrix. In such a case, PSD versions of the above alorithms, i.e. versions including a PSD constraint in the optimization problem, are not defined as the final matrix does satisfy the constraint but is not interesting from a classification point of view.
We now introduce the PSD versions of *RELIEF*, *RBS* and *sRBS* [18].

A. RELIEF-PSD Algorithm

The *RELIEF-PSD* algorithm for $k = 1$ is presented next.

```
RELIEF-PSD (k=1)
Input: training set $((x^{(1)}, c^{(1)}), \cdots, (x^{(n)}, c^{(n)}))$ of $n$ vectors in $R^p$, number of epochs $M$;
Output: diagonal matrix $A$ of estimations of the qualities of attributes
Initialization $\forall m\ 1 \leq m \leq p,\ A_m = 0$
Repeat $J$ times (epochs)
  1. randomly select an instance $x^{(i)}$
  2. find nearest hit $H$ and nearest miss $M$
  3. for $l = 1, \cdots, p$
    4. $\hat{A}_l = A_l - \frac{\text{diff}(l,x^{(i)},H)}{J} + \frac{\text{diff}(l,x^{(i)},M)}{J}$
  5. If there exist strictly positive eigenvalues of $\hat{A}$, then
  $A = \sum_{r,\lambda_r>0} \lambda_r u_r u_r^t$, where $\lambda_r$ and $u_r$ are the eigenvalues and eigenvectors of $\hat{A}$
  6. Error otherwise
```

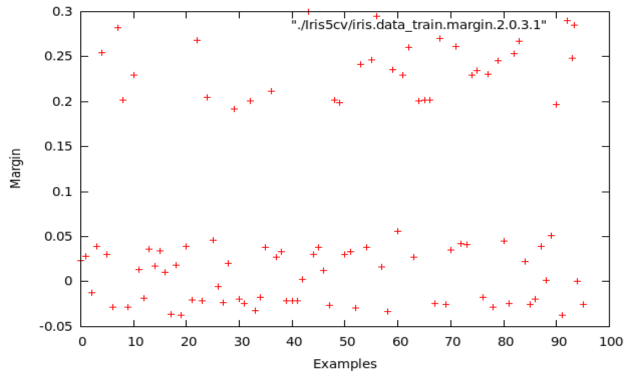

Figure. 9: Margin for *RBS*-PSD on *Iris* dataset

B. *RELIEF-Based Similarity Learning Algorithm with PSD matrices- RBS-PSD*

In this subsection, a *RELIEF*-Based Similarity Learning algorithm based on the PSD matrices (*RBS-PSD*) is proposed which is based on the *RELIEF* algorithm. However, the interest here is in similarities rather than distances. Furthermore, this algorithm is also based on the *RBS* algorithm discussed earlier. However, in this approach the similarity matrix is projected onto the set of PSD matrices. The aim here also, is to maximize the margin $\mathcal{M}(A)$ between the *target* neighbors (represented by y) and the examples belonging to other classes (called *impostors* and given by z). The margin, for $k = 1$ in *kNN* algorithm can be written as:

$$\begin{aligned}\mathcal{M}(A) &= \sum_{i=1}^{n} \left(s_A(x^{(i)}, y^{(i)}) - s_A(x^{(i)}, z^{(i)})\right)\\ &= \sum_{i=1}^{n} ({x^{(i)}}^t A y^{(i)} - {x^{(i)}}^t A z^{(i)})\\ &= \sum_{i=1}^{n} {x^{(i)}}^t A (y^{(i)} - z^{(i)})\end{aligned}$$

where A is the similarity matrix. The margin is maximized subject to two constraints: $A \succeq 0$ and $\|A\|_F^2 = 1$. There was only a single constraint ($A \succeq 0$) in the case of *RBS* algorithm. We proceed in two steps: in the first step, we maximize the margin subject to the constraint $\|A\|_F^2 = 1$, whereas in the second step, we find the closest PSD matrix of A normalized with its Frobenius norm:

$$\begin{aligned}\arg\max_{A} \quad & \mathcal{M}(A)\\ & \|A\|_F^2 = 1,\end{aligned}$$

We take the Lagrangian of the similarity matrix in a similar manner as used for *RBS* algorithm before finding the elements of A matrix.
Once we have obtained the matrix A, its closest PSD matrix $\hat{A}$ is found. Since we want the Frobenius norm of $\hat{A}$ to be equal to 1, hence $\hat{A}$ is normalized with its Frobenius norm in the following manner:

$$\tilde{A} = \frac{\hat{A}}{\sqrt{\sum_{l,m}(\hat{A}_{l,m})^2}}$$

The problem with maximizing the margin approach was verified on *RBS*-PSD and was found to be equivalent to that encountered for *RBS*. A number of UCI datasets [17] i.e. *Iris, Wine, Balance* etc. were used for the verification as seen in figures 9, 10, 11 and 12. It can be observed for all of the

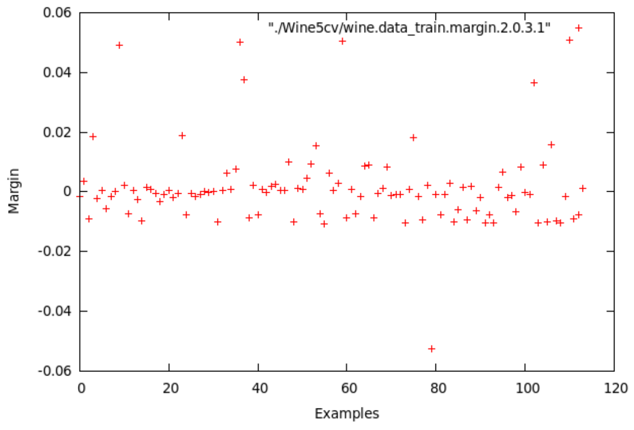

Figure. 10: Margin for *RBS*-PSD on *Wine* dataset

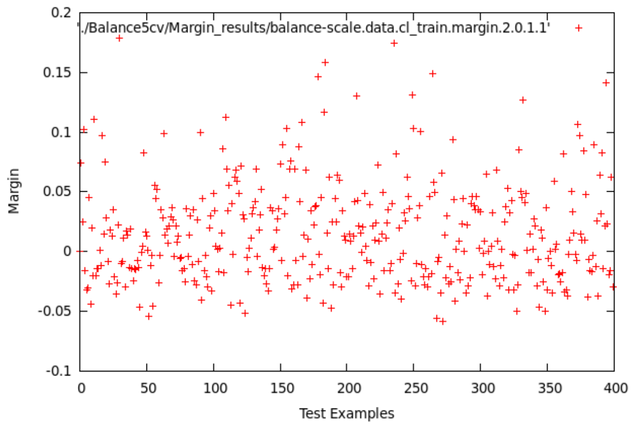

Figure. 11: Margin for *RBS*-PSD on *Balance* dataset

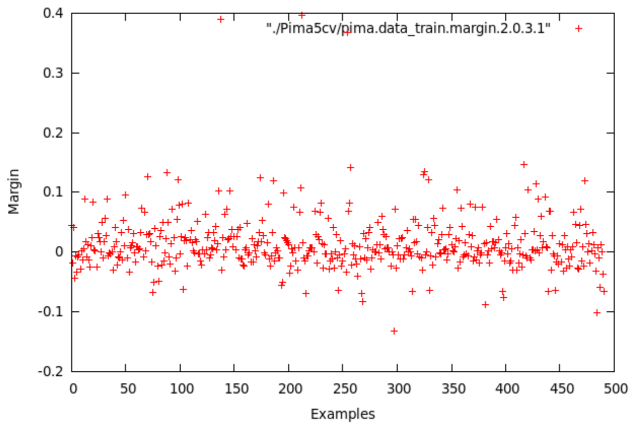

Figure. 12: Margin for *RBS*-PSD on *Pima* dataset

datasets that the average margin remains positive despite the presence of a number of mistakes, since the positive margin is much greater than the negative one for the majority of the test examples. For example, the values of negative margin in the case of *Iris* is in the range of $0.05 - 0.10$ whereas most of the positive margin values are greater than 0.15. Similarly, for *Wine*, most of the negative margin values lie in the range between 0 and -0.04 while the positive margin values are dispersed in the range $0 - 0.08$. So, despite the fact that the overall margin is large, a lot of examples are misclassified. This explains the fact that the algorithms *RELIEF-PSD* and *RBS-PSD* did not perform quite well on different standard test collections as can be seen in Section VII.

VI. A stricter version of *RBS*-PSD: *sRBS*-PSD

The same work around used in the case of *sRBS* algorithm to improve the performance of *RELIEF* based methods was used in the case of *RELEIF*-PSD and *RBS*-PSD: to directly use the leave-one-out error or $0 - 1$ loss like the *SiLA* algorithm [8] where the aim is to reduce the number of mistakes on unseen examples. The resulting algorithm is a stricter version of *RBS*-PSD and is termed as *sRBS-PSD*. It is called as a *stricter* version as we do not try to maximize the overall margin but are interested in reducing the individual errors on the unseen examples.
sRBS-PSD algorithm is exactly the same as *sRBS* except the fact that in the former case, the similarity matrix is projected onto the set of PSD matrices.
We proceed in two steps in the case of *sRBS*-PSD: in the first step we find the A matrix for which the cost function is minimized. In the second step, we find the closest PSD matrix of A which should have a low cost.

```
sRBS-PSD - Training
Input: training set $(x^{(1)}, c^{(1)}), \cdots, (x^{(n)}, c^{(n)})$ of $n$ vectors in $R^p$, $A^1_{lm}$ denotes the element of $A^1$ at row $l$ and column $m$
Output: Matrix $A$
Initialization $t = 1, A^{(1)} = 1$ (Unity matrix)
Repeat $J$ times (epochs)
  for all of the features $l, m$
      Minus$_{lm} = 0$
  for $i = 1, \cdots, n$
    for all of the features $l, m$
      Minus$_{lm} += \frac{\partial Q_i(A^t)}{\partial a_{lm}}$
  $\alpha^t = \frac{1}{t}$
  $\hat{A}^{t+1} = A^t - \frac{\alpha^t}{n} * $Minus
  If there exist strictly positive eigenvalues of $\hat{A}^{t+1}$, then
  $A^{t+1} = \sum_{r, \lambda_r > 0} \lambda_r u_r u_r^t$ (where $\lambda_r$ and $u_r$ are the eigenvalues and eigenvectors of $\hat{A}^{t+1}$)
  Error otherwise
  if $\sum_{lm} |A^{t+1}_{lm} - A^t_{lm}| \leq 0.001$
     Stop
```

Figures 13, 14, 15 and 16 show the margin values on the training data for *sRBS-PSD* for the datasets *Iris*, *Wine* and *Balance*. Comparing these results with the earlier ones for

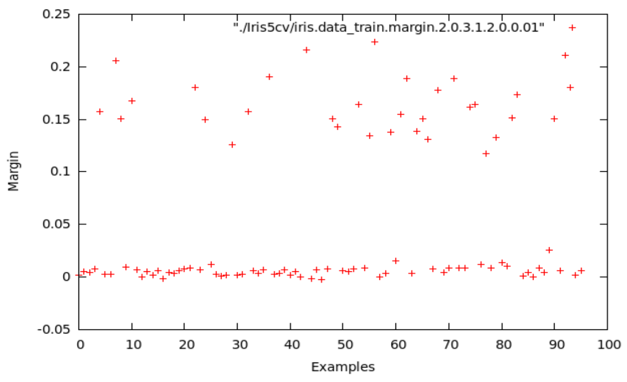

Figure. 13: Margin for *sRBS*-PSD on *Iris* dataset

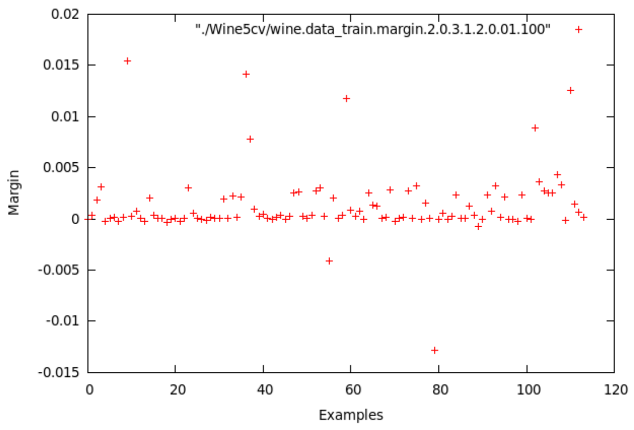

Figure. 14: Margin for *sRBS*-PSD on *Wine* dataset

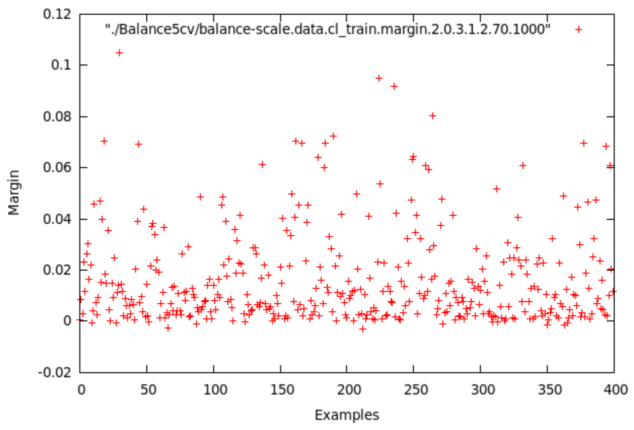

Figure. 15: Margin for *sRBS*-PSD on *Balance* dataset

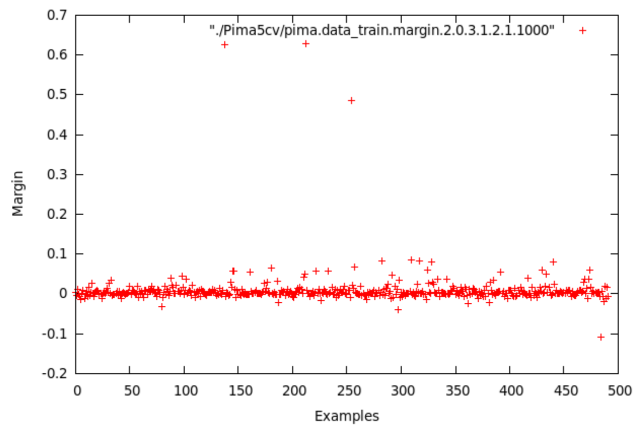

Figure. 16: Margin for *sRBS*-PSD on *Pima* dataset

	kNN-cosine	kNN-Euclidean
Soybean	1.0 ± 0.0	1.0 ± 0.0
Iris	0.987 ± 0.025	0.973 ± 0.029
Letter	0.997 ± 0.002	0.997 ± 0.002
Balance	$0.954 \pm 0.021 \gg$	0.879 ± 0.028
Wine	$0.865 \pm 0.050 \gg$	0.819 ± 0.096
Ionosphere	0.871 ± 0.019	0.854 ± 0.035
Glass	0.899 ± 0.085	0.890 ± 0.099
Pima	0.630 ± 0.041	$0.698 \pm 0.024 \gg$
Liver	0.620 ± 0.064	0.620 ± 0.043
German	0.594 ± 0.040	0.615 ± 0.047
Heart	0.670 ± 0.020	0.656 ± 0.056
Yeast	0.911 ± 0.108	0.912 ± 0.108
Spambase	0.858 ± 0.009	0.816 ± 0.007
Musk-1	0.844 ± 0.028	0.848 ± 0.018

Table 2: Comparison between cosine similarity and Euclidean distance based on s-test

RBS-PSD, one can see the importance of using a cost function closer to the $0-1$ loss. The margin is positive for most of the training examples in this case.

VII. Experimental Validation

Fifteen datasets from the UCI database ([17]) were used to assess the performance of the different algorithms. 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 Table 1 where *Bal* stands for *Balance* whereas *Iono* refers to *Ionosphere*.
Table 2 compares the performance of cosine similarity and the Euclidean distance for all of the datasets.
The matrices learned by all of the algorithms can be used to predict the class(es) to which a new example should be assigned. Two basic rules for prediction were considered: the standard *kNN* rule and its symmetric variant (*SkNN*). *SkNN* is based on the consideration of the same number of examples in the different classes. The new example is simply assigned to the closest class, the similarity with a class being defined as the sum of the similarities between the new example and its k nearest neighbors in the class.
Furthermore, all of the algorithms can be used in either a binary or multi-class mode. There are a certain number of advantages in the binary version. First, it allows using the two prediction rules given above. Moreover, it allows learning *local* matrices, which are more likely to capture the variety of the data. Finally, its application in prediction results in a multi-label decision.
5-fold nested cross-validation was used to learn the single weight vector in case of *RELIEF* and *RBS* along with their PSD counterparts, and the matrix sequence $(A_1, \cdots, A_n)$ in case of *sRBS* and *sRBS*-PSD for all of the datasets. 20 percent of the data was used for test purpose for each of the dataset. Of the remaining data, 80 percent was used for learning whereas 20 percent for the validation set. In case of *RELIEF*, *RBS* and their PSD versions, the validation set is used to find the best value of k, whereas in the case of *sRBS* and *sRBS*-PSD, it is used to estimate the values of k, λ and β. Micro-sign test (*s-test*), earlier used by Yang and Liu [19] was performed to assess the statistical significance of the different results.

A. *Comparison between different RELIEF algorithms based on kNN decision rule*

While comparing *RELIEF* with its similarity based variant (*RBS*) based on the simple *kNN* classification rule, it is evident that the later performs significantly much better only on *German* and slightly better on *Soybean* as shown in table 3. However *RELIEF* outperforms *RBS* for *Heart* while using *kNN*.
It can be further verified from table 3 that the algorithm *sRBS* performs significantly much better ($\gg$) than the *RELIEF* algorithm for eight out of twelve datasets i.e. *Soybean*, *Iris*, *Balance*, *Ionosphere*, *Heart*, *Pima*, *Glass* and *Wine*. This allows one to deduce safely that *sRBS* in general is a much better choice than *RELIEF* algorithm.

B. *Comparison between different RELIEF algorithms based on SkNN decision rule*

While comparing *RELIEF* with its similarity based variant (*RBS*) based on the *SkNN*-A rule, it can be seen from table 4 that the later performs significantly much better on *Ionosphere*, *German*, *Liver* and *Wine* collections. On the other hand, *RELIEF* performs significantly much better than *RBS* on *Heart* and *Glass*.
It can further observed that *sRBS* performed significantly much better than *RELIEF* on majority of the datasets (9 out of a total of 12) i.e. *Soybean*, *Iris*, *Balance*, *Ionosphere*, *Heart*, *German*, *Pima*, *Glass* and *Wine*. On *Liver*, *sRBS* performed slightly better than the *RELIEF* algorithm. Thus *sRBS* outperforms *RELIEF* in general for *SkNN* as seen previously for *kNN*.

C. *Performance of sRBS as compared to RBS*

Furthemore, the two *RELIEF* based similarity learning algorithms i.e. *RBS* and *sRBS* are compared using both *kNN* as well as *SkNN* as shown in table 5. On majority of the datasets, the algorithm *sRBS* outperforms *RBS* for both *kNN* and *SkNN*. *sRBS* performs significantly much better (as shown by $\ll$) than its counterpart on the following datasets: *Soybean*, *Iris*, *Balance*, *Ionosphere*, *Heart*, *Pima*, *Glass* and *Wine* for the two classification rules (*kNN* and *SkNN*). On the other hand, *RBS* was able to perform slighty better than its

	Iris	**Wine**	**Bal**	**Iono**	**Glass**	**Soy**	**Pima**	**Liver**	**Letter**	**German**	**Yeast**	**Heart**	**Magic**	**Spam**	**Musk-1**
Learn	96	114	400	221	137	30	492	220	12800	640	950	172	12172	2944	304
Valid.	24	29	100	56	35	8	123	56	3200	160	238	44	3044	737	77
Test	30	35	125	70	42	9	153	69	4000	200	296	54	3804	920	95
Class	3	3	3	2	6	4	2	2	26	2	10	2	2	2	2
Feat.	4	13	4	34	9	35	8	6	16	20	8	13	10	57	168

Table 1: Characteristics of datasets

	kNN-A (*RELIEF*)	kNN-A (*RBS*)	kNN-A (*sRBS*)
Soybean	0.711 ± 0.211	$0.750 \pm 0.197 >$	$\mathbf{1.0} \pm 0.0 \gg$
Iris	0.667 ± 0.059	0.667 ± 0.059	$\mathbf{0.987} \pm 0.025 \gg$
Balance	0.681 ± 0.662	0.670 ± 0.171	$0.959 \pm 0.016 \gg$
Ionosphere	0.799 ± 0.062	0.826 ± 0.035	$0.866 \pm 0.015 \gg$
Heart	0.556 ± 0.048	$0.437 \pm 0.064 \ll$	$\mathbf{0.696} \pm 0.046 \gg$
Yeast	0.900 ± 0.112	0.900 ± 0.112	0.905 ± 0.113
German	0.598 ± 0.068	$0.631 \pm 0.020 \gg$	0.609 ± 0.016
Liver	0.574 ± 0.047	0.580 ± 0.042	0.583 ± 0.015
Pima	0.598 ± 0.118	0.583 ± 0.140	$0.651 \pm 0.034 \gg$
Glass	0.815 ± 0.177	0.821 ± 0.165	$\mathbf{0.886} \pm 0.093 \gg$
Letter	0.961 ± 0.003	0.961 ± 0.005	$\mathbf{0.997} \pm 0.002$
Wine	0.596 ± 0.188	0.630 ± 0.165	$0.834 \pm 0.077 \gg$

Table 3: Comparison between different *RELIEF* based algorithms while using *kNN*-A method based on s-test

	SkNN-A (*RELIEF*)	SkNN-A (*RBS*)	SkNN-A (*sRBS*)
Soybean	0.756 ± 0.199	0.750 ± 0.197	$0.989 \pm 0.034 \gg$
Iris	0.673 ± 0.064	0.667 ± 0.059	$\mathbf{0.987} \pm 0.025 \gg$
Balance	0.662 ± 0.200	0.672 ± 0.173	$\mathbf{0.967} \pm 0.010 \gg$
Ionosphere	0.681 ± 0.201	$0.834 \pm 0.031 \gg$	$\mathbf{0.871} \pm 0.021 \gg$
Heart	0.526 ± 0.085	$0.430 \pm 0.057 \ll$	$\mathbf{0.685} \pm 0.069 \gg$
Yeast	0.900 ± 0.113	0.900 ± 0.112	$\mathbf{0.908} \pm 0.110$
German	0.493 ± 0.115	$\mathbf{0.632} \pm 0.021 \gg$	$0.598 \pm 0.038 \gg$
Liver	0.539 ± 0.078	$0.580 \pm 0.042 \gg$	$\mathbf{0.588} \pm 0.021 >$
Pima	0.585 ± 0.125	0.583 ± 0.140	$\mathbf{0.665} \pm 0.044 \gg$
Glass	0.833 ± 0.140	$0.816 \pm 0.171 \ll$	$\mathbf{0.884} \pm 0.084 \gg$
Letter	0.957 ± 0.047	0.961 ± 0.005	$\mathbf{0.997} \pm 0.002$
Wine	0.575 ± 0.198	$0.634 \pm 0.168 \gg$	$\mathbf{0.840} \pm 0.064 \gg$

Table 4: Comparison between different *RELIEF* based algorithms while using *SkNN*-A based on s-test

stricter version *sRBS* on *German* while using the *kNN* rule. Similarly *RBS* performs significantly much better than *sRBS* on only one dataset i.e. *German* while using the *SkNN* classification rule. The performance of *RBS* and *sRBS* is equivalent for *Yeast*, *Liver* and *Letter*. These results allows us to conclude that *sRBS* is a much better algorithm than *RELIEF*.

D. Effect of positive, semi-definitiveness on RELIEF based algorithms

In this subsection, the effect of learning PSD matrices is investigated for the *RELIEF* based algorithms.

1) RELIEF based approaches and positive, semi-definite matrices with kNN classification rule

In table 6, *RELIEF-PSD* is compared with *RELIEF*-Based Similarity learning algorithm *RBS-PSD* and its stricter version (*sRBS-PSD*) while using the *kNN* classification rule. It can be seen that *sRBS-PSD* performs much better than the other two algorithms on majority of the data sets. *sRBS-PSD* is statistically much better (as shown by the symbol $\gg$) than *RELIEF-PSD* for the following 10 datasets: *Soybean*, *Iris*, *Balance*, *Heart*, *Yeast*, *Pima*, *Glass*, *Wine*, *Spambase* and *Musk-1*. Similarly for *Ionosphere*, *sRBS-PSD* is slightly better than the *RELIEF-PSD* algorithm. On the other hand, *RELIEF-PSD* performs slightly better ($<$) than *sRBS-PSD* for *German* dataset.
Moreover, while comparing *RBS-PSD* with *RELIEF-PSD*, it can be observed that the former performs significantly better than the later for *Yeast*, *Pima* and *Musk-1*, and slightly better for *Glass* dataset. On the other hand, *RELIEF-PSD* was able to perform significantly better than *RBS-PSD* for *Heart* and *Spambase*, while slightly better for *German*.

2) RELIEF based approaches and positive, semi-definite matrices with SkNN classification rule

Table 7 compares different *RELIEF* based algorithms based on *SkNN* decision rule while using PSD matrices. It can be observed that *sRBS-PSD* performs much better than the other two algorithms on majority of the data sets as seen earlier while using the *kNN* rule. *sRBS-PSD* is statistically much better (as shown by the symbol $\gg$) than *RELIEF-PSD* for the following 10 datasets (out of 15): *Soybean*, *Iris*, *Balance*, *Heart*, *Yeast*, *Liver*, *Glass*, *Wine*, *Spambase* and *Musk-1*. *RELIEF-PSD* performs slightly better ($<$) than *sRBS-PSD* for only one dataset i.e. *German*.
Similarly, *RBS-PSD* outperforms *RELIEF-PSD* for 6 datasets (*Iris*, *Yeast*, *Liver*, *Glass*, *Spambase* and *Musk-1*) while the reverse is true for the following 3 datasets: *Balance*, *Ionosphere* and *Heart*.

3) Performance of sRBS-PSD as compared to RBS-PSD

Table 8 compares statistically the results obtained while using *RBS-PSD* and *sRBS-PSD* algorithms. The later outperforms the former for the following 7 datasets (out of 13 considered for comparison): *Soybean*, *Iris*, *Balance*, *Ionosphere*, *Heart*, *Glass* and *Wine* with both *kNN* as well as *SkNN*. *RBS-PSD* performs slightly better than its counterpart for *German* while using the *SkNN* rule. However, for the rest of the datasets, the two algorithms' performance is comparable.

4) Performance of RELIEF-PSD as compared to RELIEF

While comparing *RELIEF* (with no PSD matrices) with *RELIEF-PSD* algorithm (table 8), it can be observed that while using *kNN*, *RELIEF-PSD* performs significantly better than *RELIEF* on *German* and slightly better on *Ionosphere*. On the other hand, *RELIEF* was able to outclass its counterpart for *Yeast*. However, for rest of the datasets the performance of these two algorithms was comparable.
Table 8 also compares the effect of using PSD matrices with the *RELIEF* algorithm while using the *SkNN* decision rule. It can be observed that *RELIEF-PSD* performs significantly better than *RELIEF* on 4 datasets: *Balance*, *Ionosphere*, *German* and *Pima*. On the other hand, *RELIEF* was able to outclass its counterpart for 3 datasets: *Iris*, *Yeast* and *Glass*. The performance of these two algorithms was comparable for the remaining collections.

5) Effect of positive, semi-definitiveness on RBS and sRBS

Table 8 finds statistically the effect of positive, semi-definitiveness on *RBS* as well as *sRBS*. *RBS-PSD* outperforms *RBS* significantly for *Pima* with both *kNN* as well as *SkNN* while *Glass* for *SkNN* only. The use of PSD matrices improve the performance slightly for *Glass* with *kNN* rule.
The use of positive, semi-definitiveness neither improves nor degrades the performance for *sRBS* algorithm for both of the classification rules: *kNN* and its symmetric counterpart.

VIII. Conclusion

In this paper, we studied the links between *RELIEF* and *SiLA* algorithms. *SiLA* is interested in directly reducing the leave-one-out error or $0-1$ loss by reducing the number of mistakes on unseen examples. However, Sun and Wu have shown that *RELIEF* could be seen as a distance learning algorithm in which a linear utility function with maximum margin was optimized. We first proposed a version of *RELIEF* for similarity learning, called *RBS* (for *RELIEF*-Based Similarity learning). As *RELIEF*, and unlike *SiLA*, *RBS* does not try to optimize the leave-one-out error or $0-1$ loss, and does not perform very well in practice, as we illustrate on two UCI collections namely *Iris* and *Wine*. We thus introduce a stricter version of *RBS*, called *sRBS*, aiming at relying on a cost function closer to the $0-1$ loss. Moreover, we developed positive, semi-definite (PSD) versions of *RBS* and *sRBS* algorithms: *RBS*-PSD and *sRBS*-PSD whereby the similarity matrices were projected onto the set of PSD matrices.
The algorithms were validated on a number of UCI datasets. In order to find the statistical significance of the results, a micro-sign test, known as the s-test was employed. The algorithm *sRBS* performs statistically much better than its counterparts for most of the data sets i.e. *Soybean*, *Iris*, *Balance*, *Ionosphere*, *Pima*, *Glass* and *Wine* for both of the learning rules used: *kNN* and its symmetric variant *SkNN*. While comparing *RELIEF* with *RBS*, it can be noted that the results for the later are significantly better on *German* for both *kNN* and *SkNN*; while with only *kNN* for *Ionosphere*, *Liver* and *Wine*. On contrary, *RELIEF* performs significantly much better than *RBS* for *Heart* (with both *kNN* as well as *SkNN*) while only with *SkNN* on *Glass*. This shows that the $0-1$ loss is a more appropriate cost function than the linear utility

	kNN-A (RBS) / kNN-A (sRBS)	SkNN-A (RBS) / SkNN-A (sRBS)
Soybean	$\ll$	$\ll$
Iris	$\ll$	$\ll$
Balance	$\ll$	$\ll$
Ionosphere	$\ll$	$\ll$
Heart	$\ll$	$\ll$
Yeast	=	=
German	$>$	$\gg$
Liver	=	=
Pima	$\ll$	$\ll$
Glass	$\ll$	$\ll$
Letter	=	=
Wine	$\ll$	$\ll$

Table 5: Comparison between *RBS* and *sRBS* based on s-test

	kNN-A (*RELIEF-PSD*)	kNN-A (*RBS-PSD*)	kNN-A (*sRBS-PSD*)
Soybean	0.739 $\pm$ 0.192	0.733 $\pm$ 0.220	**1.0** $\pm$ 0.0 $\gg$
Iris	0.664 $\pm$ 0.058	0.667 $\pm$ 0.059	**0.987** $\pm$ 0.025 $\gg$
Balance	0.665 $\pm$ 0.193	0.670 $\pm$ 0.171	**0.959** $\pm$ 0.016 $\gg$
Ionosphere	0.839 $\pm$ 0.055	0.826 $\pm$ 0.035	**0.880** $\pm$ 0.015 $>$
Heart	0.556 $\pm$ 0.048	0.437 $\pm$ 0.036 $\ll$	**0.693** $\pm$ 0.047 $\gg$
Yeast	0.893 $\pm$ 0.132	0.900 $\pm$ 0.112 $\gg$	**0.911** $\pm$ 0.109 $\gg$
German	**0.637** $\pm$ 0.017	0.624 $\pm$ 0.015 $<$	0.609 $\pm$ 0.016 $<$
Liver	0.574 $\pm$ 0.034	0.580 $\pm$ 0.042	**0.606** $\pm$ 0.034
Pima	0.593 $\pm$ 0.077	**0.661** $\pm$ 0.024 $\gg$	0.651 $\pm$ 0.034 $\gg$
Glass	0.819 $\pm$ 0.164	0.835 $\pm$ 0.138 $>$	**0.886** $\pm$ 0.093 $\gg$
Letter	0.961 $\pm$ 0.005	0.961 $\pm$ 0.005	**0.997** $\pm$ 0.002
Wine	0.608 $\pm$ 0.185	0.630 $\pm$ 0.165	**0.834** $\pm$ 0.077 $\gg$
Magic	0.516 $\pm$ 0.085	0.360 $\pm$ 0.007	**0.767** $\pm$ 0.009
Spambase	**0.618** $\pm$ 0.031	0.611 $\pm$ 0.020 $\ll$	0.855 $\pm$ 0.009 $\gg$
Musk-1	0.698 $\pm$ 0.055	**0.851** $\pm$ 0.033 $\gg$	0.838 $\pm$ 0.024 $\gg$

Table 6: Comparison between different *RELIEF* based algorithms using *kNN*-A and PSD matrices

	SkNN-A (*RELIEF-PSD*)	SkNN-A (*RBS-PSD*)	SkNN-A (*sRBS-PSD*)
Soybean	0.783 $\pm$ 0.163	0.733 $\pm$ 0.220	**0.983** $\pm$ 0.041 $\gg$
Iris	0.571 $\pm$ 0.164	0.667 $\pm$ 0.059 $\gg$	**0.987** $\pm$ 0.025 $\gg$
Balance	0.708 $\pm$ 0.175	0.672 $\pm$ 0.173 $\ll$	**0.967** $\pm$ 0.010 $\gg$
Ionosphere	0.886 $\pm$ 0.028	0.834 $\pm$ 0.031 $\ll$	**0.889** $\pm$ 0.011
Heart	0.533 $\pm$ 0.067	0.437 $\pm$ 0.036 $\ll$	**0.685** $\pm$ 0.069 $\gg$
Yeast	0.897 $\pm$ 0.122	0.900 $\pm$ 0.112 $\gg$	**0.914** $\pm$ 0.106 $\gg$
German	**0.625** $\pm$ 0.035	0.624 $\pm$ 0.015	0.598 $\pm$ 0.038 $<$
Liver	0.528 $\pm$ 0.085	0.580 $\pm$ 0.042 $\gg$	**0.609** $\pm$ 0.035 $\gg$
Pima	0.659 $\pm$ 0.027	0.658 $\pm$ 0.030	**0.665** $\pm$ 0.044
Glass	0.768 $\pm$ 0.235	0.835 $\pm$ 0.138 $\gg$	**0.884** $\pm$ 0.084 $\gg$
Letter	0.961 $\pm$ 0.008	0.961 $\pm$ 0.004	**0.997** $\pm$ 0.002
Wine	0.606 $\pm$ 0.177	0.634 $\pm$ 0.168	**0.840** $\pm$ 0.064 $\gg$
Magic	0.539 $\pm$ 0.109	0.360 $\pm$ 0.007	**0.777** $\pm$ 0.009
Spambase	0.583 $\pm$ 0.075	0.611 $\pm$ 0.020 $\gg$	**0.857** $\pm$ 0.010 $\gg$
Musk-1	0.712 $\pm$ 0.037	**0.857** $\pm$ 0.029 $\gg$	0.842 $\pm$ 0.010 $\gg$

Table 7: Comparison between different *RELIEF* based algorithms using *SkNN*-A and PSD matrices

Table 8: Comparison between different algorithms based on s-test

	RELIEF / RELIEF-PSD		*RBS / RBS-PSD*		*sRBS / sRBS-PSD*		*RBS-PSD / sRBS-PSD*	
	kNN-A	*SkNN-A*	*kNN-A*	*SkNN-A*	*kNN-A*	*SkNN-A*	*kNN-A*	*SkNN-A*
Soybean	=	=	=	=	=	=	$\ll$	$\ll$
Iris	=	$\gg$	=	=	=	=	$\ll$	$\ll$
Balance	=	$\ll$	=	=	=	=	$\ll$	$\ll$
Ionosphere	$<$	$\ll$	=	=	=	=	$\ll$	$\ll$
Heart	=	=	=	=	=	=	$\ll$	$\ll$
Yeast	$\gg$	$\gg$	=	=	=	=	=	=
German	$\ll$	$\ll$	=	=	=	=	=	$>$
Liver	=	=	=	=	=	=	=	=
Pima	=	$\ll$	$\ll$	$\ll$	=	=	=	=
Glass	=	$\gg$	$<$	$\ll$	=	=	$\ll$	$\ll$
Letter	=	=	=	=	=	=	=	=
Wine	=	=	=	=	=	=	$\ll$	$\ll$
Musk-1							=	=

function used by *RELIEF*. While comparing the PSD algorithms with their counterparts, *RELIEF*-PSD performed well as compared to *RELIEF*. However, for the rest of the algorithms, the effect of projection onto the set of PSD matrices was minimal e.g. in the case of *sRBS*, the use of PSD matrices neither improves neither degrades the performance.

Acknowledgments

Ali Mustafa would like to thank the French Ministry of Higher Education and Research for the PhD grant.

References

[1] M. Diligenti, M. Maggini, and L. Rigutini, "Learning similarities for text documents using neural networks," in *Proceedings of International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, Florence, Italy, 2003.

[2] L. Baoli, L. Qin, and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, no. 4, pp. 215–226, 2004.

[3] 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, p. 94.

[4] 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.

[5] 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.

[6] S. Melacci, L. Sarti, M. Maggini, and M. Bianchini, "A neural network approach to similarity learning," in *Proceeding of Third IAPR Workshop on Artificial Neural Networks in Pattern Recognition, ANNPR, Paris, France*, ser. Lecture Notes in Computer Science. Springer, 2008, pp. 133–136.

[7] M. R. Peterson, T. E. Doom, and M. L. 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.

[8] A. M. Qamar, E. Gaussier, J.-P. Chevallet, and J. H. Lim, "Similarity learning for nearest neighbor classification," in *Proceedings of International Conference on Data Mining (ICDM)*, 2008, pp. 983–988.

[9] M. P. Umugwaneza and B. Zou, "A novel similarity measure using a normalized hausdorff distance for trademarks retrieval based on genetic algorithm," *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, vol. 1, pp. 312–320, 2009.

[10] P. Porwik, R. Doroz, and K. Wrobel, "A new signature similarity measure based on windows allocation technique," *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, vol. 2, pp. 297–305, 2010.

[11] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *ML92: Proceedings of the ninth international workshop on Machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1992, pp. 249–256.

[12] Y. Sun and D. Wu, "A RELIEF based feature extraction algorithm," in *Proceedings of the SIAM International Conference on Data Mining, SDM, Atlanta, USA*, 2008, pp. 188–195.

[13] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Machine Learning*, vol. 37, no. 3, pp. 277–296, 1999.

[14] M. Collins, "Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms," in *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, Morristown, NJ, USA, 2002, pp. 1–8.

[15] I. Kononenko, “Estimating attributes: Analysis and extensions of relief.” Springer Verlag, 1994, pp. 171–182.

[16] A. M. Qamar and E. Gaussier, “Similarity Learning in Nearest Neighbor and RELIEF Algorithm,” in *Proceedings of the Ninth IEEE International Conference on Machine Learning and Applications (ICMLA)*, Washington DC, USA, December 2010.

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

[18] A. M. Qamar and E. Gaussier, “Similarity Learning in Nearest Neighbor; Positive Semi-Definitiveness and Relief Algorithm,” in *Proceedings of the Second IEEE International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, Cergy Pointoise / Paris, France, December 2010.

[19] Y. Yang and X. Liu, “A re-examination of text categorization methods,” in *SIGIR ’99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. NY, USA: ACM Press, 1999, pp. 42–49.

Author Biographies

Ali Mustafa Qamar Dr. Ali Mustafa Qamar is currently serving as an Assistant Professor at School of Electrical Engineering and Computer Science (SEECS) at National University of Sciences and Technology, NUST, Islamabad, Pakistan. He completed his PhD in Machine Learning from University of Grenoble, France in November 2010. During his PhD, he worked in Grenoble Informatics Laboratory where he developed similarity metric learning algorithms for nearest neighbor classification. After completing his PhD, he served as a Post-Doc scientist/teacher at University Joseph Fourier (UJF). He has done his Masters in Artificial Intelligence in 2007 from UJF and Bachelors of Software Engineering from Colleg of Signal, NUST in 2005.

Eric Gaussier Pr. Eric Gaussier graduated from École Centrale Paris in Applied Mathematics and from Universit Paris 7 in Computer Science, in 1990. He did his PhD in Computer Science at both Université Paris 7 and the IBM France Scientific Center, on probabilistic models for bilingual lexicon extraction. After having been Area Manager for Learning and Content Analysis at the Xerox Research Centre Europe, he joined the Université Joseph Fourier and the Grenoble Informatics laboratory as a Professor of Computer Science in September 2006. He is currently leading the AMA team, the research of which fits within the general framework of machine learning and information modeling. He is a member, since 2007, of the Information Sciences panel for starting grants of the European Research Council, a member, since 2005, of the Advisory Board of SIGDAT and was a member, from 2007 to 2010, of the Executive Board of the European Association for Computational Linguistics. Since 2010, he is deputy director of the Grenoble Informatics Laboratory, one of the largest Computer Science laboratories in France.