THE SUPPORT VECTOR MACHINED KERNEL

Khaled S. Refaat[1], *Member*, *IEEE*

Abstract: **In this paper, we propose the so-called "SVM'ed-kernel function" and its use in SVM classification problems. This kernel function is itself a support vector machine classifier that is learned statistically from data. We show that the new kernel manages to change the classical methodology of defining a feature vector for each pattern. One will only need to define features representing the similarity between two patterns allowing many details to be captured in a concise way. The new proposed kernel shows very promising results. It opens the door for new feature definitions that could be created in various machine learning problems where similarity between patterns can be formulated more suitably.**

Index Terms**: Support Vector Machine, Kernel, Similarity**

I. INTRODUCTION

In the classical classification framework, training patterns are first converted to feature vectors which are then used to train the classifier. At the point of replacing the pattern by a feature vector representing it, a significant amount of information is lost.

In addition, sometimes representing the pattern by a feature vector could be problematic. For example if we would like to represent a document by a feature vector, we could use a dictionary to create a feature vector of word. This could lead to a huge number of features. An alternative and seemingly more efficient way is to define a feature vector that represents the similarity between a pair of documents. In such case we could just define a vector that consists of a few simple and effective high level features. This similarity vector could, for example, consists of the number of common stemmed words between the pair of documents, the number of common named entities, the number of common semantic relations, and finally a binary feature showing whether the two documents were extracted from the same source or not. This suggests that a significant achievement could be acquired, if we could change the classification framework to using feature vectors that represent the similarity between a pair of patterns rather than using feature vectors that represent single patterns.

SVM is a suitable classifier for applying this new framework. In SVM, the classical kernels take two feature vectors as input (each feature vector represents a pattern) and return a real number representing the similarity between them [1]. In order to make use of high level similarity features as stated previously, a domain expert is required to invent a user defined kernel which is an algorithm that measures the similarity between two patterns without converting them to feature vectors. The domain expert is required in order to determine the contribution of each component similarity feature to the final similarity measure. This is a time consuming task since it has to be done for each problem. Moreover, a hard quantitative approach would lead to more consistent performance, and allows the use of cutting edge optimization methods.

In this paper we propose a new kernel function that is learned statistically from data. The input of this new kernel function will be only one feature vector representing the similarity between the two input patterns. We name our new proposed kernel the SVM'ed-Kernel for a reason that will be clear.

We propose a method to automatically generate a recreated training set from the original training set. The recreated training set is then used to learn the SVM'ed-Kernel. Interestingly, the SVM'ed-Kernel will be learned as a separate SVM classification problem. Once trained, the SVM'ed-Kernel will then be used as a kernel function in the original classification SVM problem.

Using the SVM'ed-Kernel, we need not define features to represent a single pattern. We will only need to define features that represent the similarity between a pair of patterns. This allows novel features to be defined that could not have been defined using the classical feature definition framework.

Moreover, a simple similarity feature between a pair of patterns could eliminate a large number of features representing a single pattern as it was shown in the example of representing a document by a feature vector. This contributes to dimensionality reduction.

Figures 1 and 2 show the classical kernel and the SVM'ed-Kernel block diagrams respectively.

In the SVM'ed-Kernel, the contribution of each similarity feature to the final similarity measure is learned statistically from the recreated training set. This eliminates the need for a domain expert, allows the definition of novel high level similarity features, and leads to optimizing the contributions of the different similarities.

The proposed kernel could be used in Natural Language Processing, Machine Vision, and Bioinformatics applications that suffer from the loss of a significant amount of information at the point

[1] Khaled S. Refaat is with the Computer Engineering Department, Faculty of Engineering, Cairo University, Egypt. (e-mail: khaled.saeed84@gmail.com)

where the pattern is replaced by a feature vector representing it. We tested our SVM'ed-Kernel in two classification problems and showed that it gives very promising results.

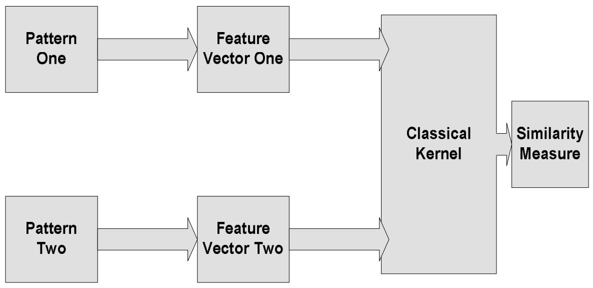
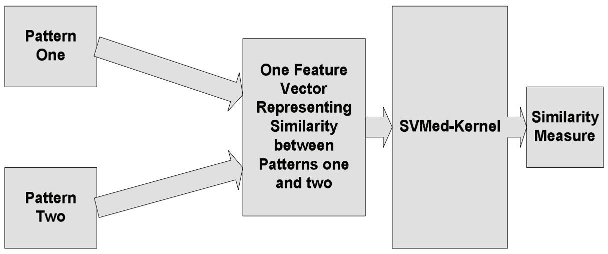


Fig.1 –Classical Kernel block diagram.


Pattern One
Pattern Two
One Feature Vector Representing Similarity between Patterns one and two
SVMed-Kernel
Similarity Measure
Fig.2 –SVM'ed-Kernel block diagram.

II. RELATED WORK

Many Kernels have been proposed in the SVM literature. We divide the related work into general kernels and specific user-defined kernels. The general kernels are not defined for a specific problem. On the other hand, the user defined kernels are domain dependent and they are defined specifically for the problem at hand. Our proposed SVM'ed-Kernel falls in the general kernels class.

From among the general proposed kernels, Thadani *et al* [2] creates a kernel function suitable for the training data using a genetic algorithm mechanism. They showed that their genetic kernel has good generalization abilities when compared with the polynomial and the radial basis kernel functions. Kong *et al* [3] proposed the autocorrelation kernel by borrowing this concept from signal processing. The autocorrelation functions give comparable results to the RBF kernel when used to classify some UCI datasets. Ye *et al* [4] proposed an orthogonal Chebyshev kernel function. Chebyshev polynomials are first constructed through Chebyshev formulae. Then based on these polynomials Chebyshev kernels are created satisfying Mercer condition. They showed that it is possible to reduce the number of support vectors using this kernel. In addition, they require the features to be normalized from -1 to 1. George *et al* [5] proposed a Sinc-Cauchy hybrid wavelet kernel and shows that it is admissible which means that it is positive definite [1]. They used it for the classification of Cardiac Single Photon Emission Computed Tomography images and Cardiac Arrhythmia signals. Their experimental results showed that promising generalization can be achieved with the hybrid kernel compared to conventional kernels. Wang *et al* [6] proposed the Weighted Mahalanobis Distance Kernels. They first find the data structure for each class in the input space via agglomerative hierarchical clustering and then construct the weighted Mahalanobis distance kernels which are affected by the size of clusters they reside in. They showed that, although WDM kernels are not guaranteed to be positive definite or conditionally positive definite, satisfactorily classification results can still be achieved because regularizes in SVMs with WDM kernels are empirically positive in pseudo-Euclidean spaces. Boughorbel *et al* [7] proposed the log kernel which seemed particularly interesting for images. They proved that the log kernel is conditionally positive definite. Moreover, they showed from experimentations that using conditionally positive definite kernels allows us to outperform classical positive definite kernels.

From among the specific user-defined kernels, XU *et al* [8] proposed using the weighted Levenshtein distance as a kernel function for strings. They used the UCI splice site recognition dataset for testing their proposed specific kernel which got the best results in this problem. Wu *et al* [9] proposed a new user-defined kernel for RNA classification. They showed that the new kernel takes advantage of both global and local structural information in RNAs. Their experimental results showed that the new kernel outperforms existing kernels when used to classify non-coding RNA sequences. Siolas *et al* [10] proposed using a new metric between documents based on a priori semantic knowledge about words. They incorporated this metric into the definition of radial basis function which improved the performance. Yan *et al* [11] proposed the position weight subsequences kernel (PWSK) that could be used for identifying gene sequences. This kernel was used for splice site identification and the performance was better than that of the string subsequences kernel (SSK). Cuturi *et al* [12] proposed a mutual information kernel for strings which borrows techniques from information theory and data compression. They showed that their kernel reported encouraging classification results on a standard protein homology detection experiment.

Our proposed kernel falls in the general kernels class while having the ability of defining similarity features which have been only used in specific user defined kernels. Moreover, it does not need a domain expert to determine the contribution of each similarity feature to the similarity measure since the kernel is learned statistically from data which is extracted automatically from the original training set.

III. PRELIMINARIES

The basic idea of SVM classifiers is to map a given data set from input space into higher dimensional feature space F, called dot product space, via a map function ϕ, where

$$\phi : R^N \to F \tag{3.1}$$

Then, it performs a linear classification in the higher dimensional space F. This requires the evaluation of dot products:

$$K(x, y) = (\phi(x), \phi(y)), \tag{3.2}$$

Where $K(x, y)$ is called the kernel function. Since F is high dimensional, then the right hand side of equation (3.2) will be very expensive to compute [1]. Therefore, kernel functions are used to compute the dot product in the feature space using the input parameters which means that the mapping to F is done implicitly. A kernel function returns a real number representing the similarity of its two input patterns. There are many types of kernels such as the RBF kernel, given by:

$$K(x, x_i) = e^{-\|x-xi\|^2 / 2\sigma^2} \tag{3.3}$$

Other similar kernels are also widely used.

The function used for the assignment of new objects to one of the two classes is called the decision function which takes the form:

$$f(x) = \begin{cases} +1 & if \quad \sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b > 0 \\ -1 & if \quad Otherwise \end{cases} \tag{3.4}$$

Where, l denotes the number of training patterns
x denotes unseen pattern vector
x_i denotes the i^{th} training pattern vector
y_i denotes label of the i^{th} training pattern
b denotes constant offset (or threshold)
1 and -1 are the labels of decision classes

The parameters αi are computed as the solution of a quadratic programming problem of the form:

$$\underset{w \in \aleph, b \in R}{\text{minimize}} \quad \tau(\mathrm{w}) = \frac{1}{2} \| w \|^2 \tag{3.5}$$

Subject to $y_i(\langle w, x_i \rangle + b) \geq 1$ for all $i = 1, \ldots, l$

Where,
w denotes weight vector in feature space
$\aleph$ denotes feature space
R denotes set of real
τ denotes objective function

The computed non-zeros αi's correspond to training patterns known as support vectors. Finally, substituting the values of αi in (3.4) produces the decision function hyper-plane in the feature space that corresponds to a nonlinear function in the input space as shown in figure 3. Thus, the classification problem becomes easier to be solved in the higher dimensional space than in the lower dimensional space [1].

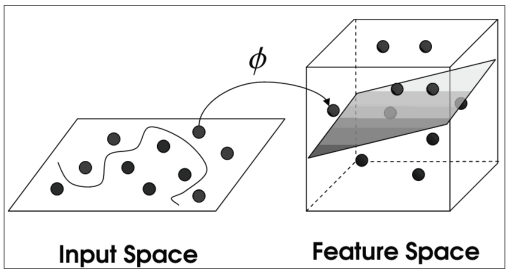


Fig.3 –Mapping data to the higher dimensional feature space.

IV. THE SVM'ED-KERNEL

The SVM'ed-Kernel could be used in any machine learning task that requires a kernel function. In this paper we illustrate its use as a kernel function for support vector machine classification problems.

Internally, the SVM'ed-Kernel is constructed as a support vector machine classification problem. Therefore we have two SVMs; the first one is the original SVM classification problem which we will call it the original SVM, while the other is the one used as a kernel function which we call it the SVM'ed-Kernel.

This SVM'ed-Kernel will be trained using a recreated training set extracted from the original one. The steps to create and use the SVM'ed-Kernel are: *A*. Define a feature vector representing the similarity between a pair of patterns, *B*. Automatically generate the recreated training set from the original one. *C*. Train the SVM'ed-Kernel as a normal classification problem using the recreated training set in *B*. *D*. Use the trained SVM'ed-kernel as a kernel function in the original SVM problem. *E*. Train the original SVM using the SVM'ed-Kernel.

We now explain each step in details. In step *A*, We define a feature vector that represents the similarity between two patterns. For example in a text categorization classification problem where we need to classify a document according to whether it is related to either sport or politics. One could define a similarity feature vector of two features. The first feature could be the number of common words after stemming, while the second one could be the number of common semantic relations.

In step *B*, assume that we have an original simple training set similar to that in table one.

To create the recreated training set that will be used to train the SVM'ed-Kernel, we select every pair of patterns from the original training set (order is not important). So we have pattern 1 and pattern 2, pattern 1 and pattern 3, pattern 2 and pattern 3, pattern 2 and pattern 4, and so on. We label each pair as being matching (1) if the two patterns have the same label in the original training set or not matching (-1) if they have different labels. Table two illustrates the recreated training set. One can see here that the recreated training set is of larger size than the original training set.

Table 1. The original training set

Patterns	Class label (1 or-1)
pattern 1	1
pattern 2	-1
pattern 3	1
pattern 4	-1
pattern 5	1
pattern 6	1

Table 2. The recreated training set

Patterns	Class label (1 or-1)
pattern 1 and pattern 2	not matching (-1)
pattern 1 and pattern 3	matching (1)
pattern 1 and pattern 4	not matching (-1)
pattern 1 and pattern 5	matching (1)
pattern 1 and pattern 6	matching (1)
pattern 2 and pattern 3	not matching (-1)
pattern 2 and pattern 4	matching (1)
pattern 2 and pattern 5	not matching (-1)
pattern 2 and pattern 6	not matching (-1)
pattern 3 and pattern 4	not matching (-1)
pattern 3 and pattern 5	matching (1)
pattern 3 and pattern 6	matching (1)
pattern 4 and pattern 5	not matching (-1)
pattern 4 and pattern 6	not matching (-1)
pattern 5 and pattern 6	matching (1)

In step *C*, we first convert each pair in the recreated training set, created in step *B,* to a feature vector using the similarity feature vector definition we have defined in step *A*. After that, we train the SVM'ed-Kernel as a normal SVM classification problem using the recreated training set and any arbitrary kernel. The output of this SVM classifier will be a label indicating whether the two input patterns are matching or not. After training, we save the SVM trained as our SVM'ed-Kernel after removing the decision component from the function in equation (3.4) to be in the form

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b \tag{4.1}$$

The decision component was removed because we are interested in the real value returned from (4.1), which represents how confident we are in the match. A larger returned value represents a better match (high similarity) and vice versa. Figure 4 shows three pairs, from the recreated training set, and their locations from the decision boundary of the SVM'ed-Kernel. When substituting in equation (4.1), pair one's similarity feature vector will return a positive number indicating high similarity between this pair. On the other hand, pair two's similarity feature vector will return a smaller positive number indicating that this pair is less similar than pair one. Finally, pair three's vector will return a negative number indicating low similarity between this pair.

The training patterns in the recreated training set were used to determine the maximal margin classifier. The distance of a new pair from the maximal margin classifier decision boundary is a direct measure of the similarity between the two patterns of this pair due to the way we formed the recreated training set in *B* and the definition of the similarity feature vector in *A*.

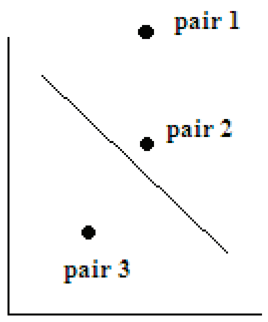


Fig.4 –Three pairs and their locations from the decision boundary of the SVM'ed-Kernel.

In step *D*, we use our trained SVM in *C* as our kernel function in the original SVM problem. In our work we have added an offset to the output of the SVM'ed-Kernel to have all the patterns on the positive side of the SVM'ed-Kernel decision boundary.

In step *E*, we train our original SVM using our SVM'ed-Kernel and the original training set. After training the original SVM, the original SVM model is ready for the real operation phase.

We conclude here that the original SVM classifier component does not require the pattern to be extracted to a feature vector on its own. It takes the form:

$$f(pattern) = \sum_i^{l} \alpha i.yi.SVK(pattern, patterni) + b$$

(4.2)

where SVK is the SVM'ed-Kernel in the form:

$$SVK(patternx, patterny) = \sum_j^{l'} \alpha j.yj.k(m(x,y), mj) + b'$$

(4.3)

Where,

mj denotes a similarity feature vector of a support vector pattern for the SVM'ed-Kernel.

$m(x,y)$ denotes the feature vector representing similarity between $patternx$ and $patterny$.

yi, yj denote the labels of the i^{th} and j^{th} patterns respectively.

b , b' denote constant offsets (or thresholds).

k denotes an arbitrary kernel function.

l l' denote the number of training patterns in the original and the recreated training sets respectively.

The parameters αi and αj are computed as the solutions of quadratic programming problems.

Figure 5 shows a block diagram of the original SVM classifier that makes use of the SVM'ed-Kernel which in turn makes use of an arbitrary kernel.

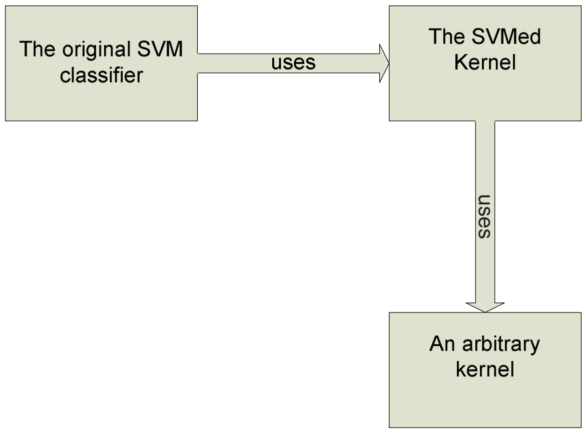


Fig.5 –A block diagram showing the original SVM classifier that makes use of the SVM'ed-Kernel which in turn makes use of an arbitrary kernel.

V. EXPERIMENTAL SETUP

We define the experimental setup of two classification problems. In the first one we show that the SVM'ed-Kernel is capable of using the already existing feature definitions. In the second, we show its ability to define features that could not have been defined using the classical kernels.

The first classification problem is a hand drawn circle-triangle classification problem. The input is a hand drawn shape and the output is its classification as a circle or a triangle. We assumed that the unseen shape should be either a circle or a triangle. Figure 6 shows a hand-drawn circle.

Fig.6 –A hand-drawn circle.

The data set used was formed from the circles and triangles patterns created by Refaat *et al* [13] after increasing its size to be 222 circles and 222 triangles. We divided the dataset set randomly into 400 patterns for training and 44 patterns for testing. We compared our SVM'ed-Kernel to different kernels with different parameters. We used SVM Light [14] in our simulations.

A number of features are extracted from each shape for the purpose of serving as inputs to the classifier. The features were selected to be size and orientation independent [13]. The basic shape features used are Ach, the area of the convex hull; Pch, its perimeter; Aer the area of the rectangle enclosing the convex hull of the shape and having the minimum area; Alq, the area of the maximum area inscribed quadrilateral that fits inside the convex hull of the shape [15]; Alt, the area of the maximum area inscribed triangle that fits inside the convex hull of the shape [15]; Plt, its perimeter; Her, the height of the rectangle enclosing the convex hull of the shape and having the minimum area; and , Wer its width.

The features used for all kernels except the SVM'ed-Kernel are

Pch^2 / Ach (Thinness ratio)

Ach / Aer

Alq / Aer

Alq / Ach

Alt / Alq .

Alt / Ach

Plt / Pch

Her / Wer

The similarity feature vector definition that is defined for the sake of the SVM'ed-Kernel was defined simply to be the subtraction of the two feature vectors of the two input patterns. In this problem, we did not define new similarity features that make use of the benefit of the SVM'ed- Kernel to evaluate our kernel power of making use of the already existing feature definition that was defined for a single pattern.

In the second classification problem, we used the Mushroom problem from the UCI Machine Learning Repository. This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

The attributes available for each pattern were the cap-shape, the cap-surface, the cap-color, the bruises, the odor, the gill-attachment, the gill-spacing, the gill-size, the gill-color, the stalk-shape, the stalk-root, the stalk-surface-above-ring, the stalk-surface-below-ring, the stalk-color-above-ring, the stalk-color-below-ring, the veil-type, the veil-color, the ring-number, the ring-type, the spore-print-color ,the population and the habitat.

Each pattern is described by 22 characters; each of them is describing an attribute. If we are going to define a feature vector for each pattern, we will need to encode such characters. Using the SVM'ed-Kernel, we do not need to define a feature vector for each pattern. We defined the similarity feature vector as a binary vector of 22 features. Feature i takes the value of one if attribute i is the same in the two input patterns, otherwise it takes zero. This eliminated the burden of encoding the character attributes using an elegant definition of the similarity feature vector in a binary simple form.

While creating the similarity feature vector, we handle the missing attributes in the dataset by putting the value of one if both attributes are missing. If only one attribute is missing we put zero. We divided the data set into 67 % for training and 33% for testing.

Interestingly, the recreated training set had more than twenty million combinations, so we have randomly sampled about 11% of this recreated training set to train our SVM'ed-Kernel due to size limitations in the SVM Light package.

In both classification problems the SVM'ed-Kernel uses internally the RBF kernel with the Gamma parameter set to two. We note here that any kernel could be used by the SVM'ed-Kernel.

VI. TESTING RESULTS

For the hand drawn circle-triangle problem, we used the SVM'ed-Kernel with just subtracting the two feature vectors of the two input patterns. We also used the polynomial kernel (Poly) with the d parameter set to 2, 3 and 4; the RBF kernel with the gamma parameter set to 1, 2, 0.1 and 0.01; and the linear kernel. Table three shows the testing results of the circle-triangle problem.

The SVM'ed-Kernel gives comparable testing accuracy to all other kernels with respect to the test set size and gives the least number of support vectors. The SVM'ed-Kernel has only 6 support vectors whereas all other kernels have at least 27 support vectors. This shows that the SVM'ed-Kernel ability of generalization outperforms all other classical kernels. In addition, the SVM'ed-Kernel outperforms all other kernels in the Joachim Xi alpha [16] estimate of the recall and shows very promising Joachim Xi alpha of the precision.

We note here that we did not make use of the benefit of defining features of similarity between a pair of patterns but only showed that the SVM'ed-Kernel is capable of using the original feature definitions that were already defined to all problems before.

In the Mushroom problem, Kim *et al* [17] reported 99.51 %, 96.61 %, and 99.53 % ten-fold cross validation accuracies using different SVM variations to avoid training with the whole dataset. In order to make the problem harder; we divided the data set into only 67 % for training and 33% for testing. After generating the recreated training set, we chose 11% of its patterns randomly to train our SVM'ed- Kernel. The accuracy on the test set using our trained SVM'ed-Kernel turned out to be 99.71 %. The precision was 99.06 % while the recall was 98.49 %. We showed here that the SVM'ed-Kernel was able to generalize well even while training it with only 11% of the recreated training set. In addition, we showed that the SVM'ed-Kernel eliminated the burden of encoding the character attributes using an elegant definition of the similarity feature vector in a binary simple form.

Table 3. Hand-drawn circle-triangle problem testing results

The kernel used	**Accur-acy on the test set**	**XiAlpha estimate of the recall**	**XiAlpha estimate of the precision**	**Num. of support vectors**
SVM'ed	97.73	99.5	99.5	6
Poly d=2	97.73	91.5	91.5	34
Poly d=3	97.73	90	90	40
Poly d=4	97.73	81	81	76
RBF g=1	100	97.5	100	68
RBF g=2	100	97	100	84
RBF g=.1	100	98.5	98.99	42
RBFg=.01	100	94.5	94.97	27
Linear	97.73	86	86	56

REFERENCES

[1]. B. Scholkopf and A. J. Smola. Learning with Kernels. The MIT press, Massachusetts London, England 2002.

[2]. K. Thadani, A. Jayaraman, and V. Sundararajan. Evolutionary Selection of Kernels in Support Vector Machines. Proceedings of "The International Conference on Advanced Computing and

Communications (ADCOM’2006)”, 2006.
[3]. R. Kong and B. Zhang. Autocorrelation Kernel Functions for Support Vector Machines. Proceedings of “International Conference on Natural Computation (ICNC’2007)”, 2007.
[4]. N. Ye, R. Sun, Y. Liu and L. Cao. Support Vector Machines with Orthogonal Chebyshev Kernel. Proceedings of ”The International Conference on Pattern Recognition (ICPR’2006)”, 2006.
[5]. J. George and K. Rajeev. SINC-CAUCHY Hybrid Wavelet Kernel for Support Vector Machines. Proceedings of the Workshop “Machine Learning for Signal Processing (MLSP’2008)”, 2008.
[6]. D. Wang, D. Yeung, and C. Eric. Weighted Mahalanobis Distance Kernels for Support Vector Machines. IEEE Transaction on Neural Networks (2007).
[7]. S. Boughorbel, J. Tarel, and N. Boujemaa. Conditionally Positive Definite Kernels for SVM Based Image Recognition. Proceedings of ”IEEE International Conference on Multimedia and Expo (ICME’2005)”, 2005, pp.113-116.
[8]. J. Xu, and X. Zhang. Kernels Based on Weighted Levenshtein Distance. Proceedings of ”International Joint Conference on Neural Networks (IJCNN’2004)”, 2004, pp.3015-3018.
[9]. X. Wu, J. Wang, K. Herbert. A New Kernel Method for RNA Classification. Proceedings of ”International Symposium on Bioinformatics and Bioengineering (BIBE’2006)”, 2006, pp.201-208.
[10]. G. Siolas, and F. d’Alche-Buc. Support Vector Machines Based on a Semantic Kernel for Text Categorization. Proceedings of “International Joint Conference on Neural Networks (IJCNN’2000)”, 2000, pp.205-209.
[11]. C. Yan, Z. Wang, Q. Gao, and Y. Du. A Novel Kernel for Sequences Classification. Proceedings of “IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE’2005)”, 2005, pp.769-773.
[12]. M. Cuturi, and J. Vert. A Mutual Information Kernel for Sequences. Proceedings of “International Joint Conference on Neural Networks (IJCNN’2004)”, 2004, pp.1905-1910.
[13]. K. Refaat, W. Helmy, A. Ali, M. AbdelGhany, and A. Atiya. A New Approach for Context-Independent Handwritten Offline Diagram Recognition using Support Vector Machines. Proceedings of “International Joint Conference on Neural Networks (IJCNN’2008)”, pp.177-182.
[14]. SVMlight is an implementation of Support Vector Machines in C, by Thorsten Joachims.
[15]. J. E. Boyce, D. P. Dobkin, R. L. (Scot) Drysdale, and L. J. Guibas. Finding External Polygons. Proceedings of “Annual Symposium on the Theory of Computing (STOC’1982)”, 1982.
[16]. J. Joachims. The maximum-margin approach to learning text classifier: method, theory and algorithms, Ph.D. Department of Computer Science, University of Dortmund, 2000.
[17]. H. Kim and S. Park. Data Reduction in Support Vector Machines by a Kernalized Ionic Interaction Model. Proceedings of ”SIAM International Conference on Data Mining (SDM’2004)”, 2004.

Khaled S. Refaat was born in Cairo, Egypt on December 14, 1984. He received the BSc degree in Computer Engineering from Cairo University, Egypt in 2007. His graduation project entitled “A new approach for context independent handwritten offline diagram recognition using support vector machines” has won the best computer engineering graduation project in Egypt in the national competition (EED’2007). He was awarded IEEE CIS 2008 Student Travel Grant for IJCNN’08. From August 2007 to February 2009, he worked as a Research Engineer in the Human Language Technologies Group, IBM Cairo Technology Development Center, Egypt. In IBM, he worked in several research projects including event and relation extraction from unstructured text, Arabic named entity normalization, content filtering, and ontology learning. He is currently a teaching assistant and a master student in Cairo University, Egypt. His research interests include Machine Learning, Computational Linguistics, Time Series Forecasting, Machine Vision, and Support Vector Machines.

.