Hand-Drawn Shape Recognition Using the SVM'ed Kernel

Khaled S. Refaat and Amir F. Atiya

Computer Department, Faculty of Engineering, Cairo University, Egypt
khaled.saeed84@gmail.com, amir@alumni.caltech.edu

Abstract. We describe an application of the novel Support Vector Machined Kernel (SVM'ed Kernel) to the Recognition of hand-drawn shapes. The SVM'ed kernel function is itself a support vector machine classifier that is learned statistically from data using an automatically generated training set. 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. In addition, we illustrate that features describing a single pattern could also be used in this new framework. In this paper we show how the SVM'ed Kernel is defined and trained for the multiclass shape recognition problem. Simulation results show that the SVM'ed Kernel outperforms all other classical kernels and is more robust to hard test sets.

Keywords: Shape recognition, Support Vector Machine, Kernel, Similarity.

1 Introduction

Structured diagrams are very prevalent in many document types. Most people who need to create such diagrams use structured graphics editors such as Microsoft Visio [16]. Structured graphics editors are extremely powerful and expressive but they can be cumbersome to use [17]. It was shown through extensive timing experiments that structured diagrams drawn by hand takes only about 10% of the time it takes to draw one using a tool like Visio [10]. This indicates the value of automated recognition of hand-drawn diagrams.

One of the main steps in the problem of diagram understanding is the recognition of individual hand-drawn shapes. The input to the shape recognition system is a geometric hand-drawn shape. Whereas, the output is its classification to one of predefined classes.

In the classical classification framework, shapes are first converted to feature vectors which are then used to train the classifier. At the point of replacing the shape by a feature vector representing it, a significant amount of information is lost. This could be easily noticed when we discover that we could not recover the shape pattern once converted to a feature vector.

In other problems, sometimes representing the pattern by a feature vector could be problematic. For example if we would like to represent a document by

C. Alippi et al. (Eds.): ICANN 2009, Part II, LNCS 5769, pp. 275–284, 2009.
© Springer-Verlag Berlin Heidelberg 2009

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, consist 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.

We propose a novel kernel function that is extracted from data through a statistical learning procedure. 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.

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 do not need to 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.

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.

In this paper, we describe the application of the SVM'ed Kernel to the Recognition of hand-drawn shapes. The SVM'ed Kernel will allow adding a chain code similarity feature representing the similarity between a pair of patterns that will boost the accuracy significantly.

This paper is organized as follows: Section 2 describes the related work. In Section 3 we introduce preliminaries of SVM as a classifier. The SVM'ed Kernel will be presented in Section 4. Section 5 describes the shape recognition problem. Finally we introduce the experimental results in section 6. The paper ends with a conclusion and future work in section 7.

2 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.

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. George *et al* [4] 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* [5] 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.

From among the specific user-defined kernels, XU *et al* [6] 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* [7] 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. Yan *et al* [8] 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* [9] 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.

For the problem of shape recognition, Valveny and Marti discussed a method for recognizing hand-drawn architectural symbols [13] using deformable template matching. They achieved recognition rates around 85%, but did not discuss how the user might correct an incorrect recognition. Notowidigo and Miller [14] presented a novel approach to creating structured diagrams. There system aims to provide drawing freedom by allowing the user to sketch entirely off-line using a pure pen-and paper interface. The system can infer multiple interpretations for a given sketch to aid during the user's polishing stage. The UDSI program uses a novel recognition architecture that combines low-level recognizers with domain-specific heuristic filters and a greedy algorithm that eliminates incompatible interpretations. Refaat *et al* (2008) [10] has proposed a new approach for context-independent hand-written diagram recognition using support vector machines achieving an acceptable segmentation accuracy and approaching 90% recognition accuracy.

3 Preliminaries

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

$$\phi : R^N \longrightarrow F \tag{1}$$

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

$$k(x, y) = (\phi(x), \phi(y)) \tag{2}$$

Where k is called the kernel function. If F is high dimensional, the right hand side of equation (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 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:

$$k(x, x_i) = e^{-\|x-x_i\|^2/2\sigma^2} \tag{3}$$

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:

$$f(x) = \begin{cases} +1 & \text{if } \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b > 0, \\ -1 & \text{if otherwise.} \end{cases} \quad (4)$$

Where, l denotes the number of training patterns
x denotes the unseen pattern vector
x_i denotes training pattern vector
y_i denotes the label of the training pattern
b denotes constant offset (or threshold)
1 and -1 are the labels of the decision classes
The parameters $\alpha_i{}'s$ are computed as the solution of a quadratic programming problem.

4 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 1. 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 pattern3, 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 2

Table 1. The original training set

patterns	class label 1 or -1
pattern 1	1
pattern 2	-1
pattern 3	1
pattern 4	-1

Table 2. The recreated training set

patterns	class label 1 or -1
pattern 1 and pattern 2	-1
pattern 1 and pattern 3	1
pattern 1 and pattern 4	-1
pattern 2 and pattern 3	-1
pattern 2 and pattern 4	1
pattern 3 and pattern 4	-1

illustrates the recreated training set. One can see here that the recreated training set is of larger size than the original training set.

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 (4), to be in the form

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i k(x, x_i) + b \tag{5}$$

The decision component was removed because we are interested in the real value returned from (5), which represents how confident we are in the match. A larger returned value represents a better match (high similarity) and vice versa. Figure 1 shows three pairs, from the recreated training set, and their locations from the decision boundary of the SVM'ed-Kernel. When substituting in equation (5), 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

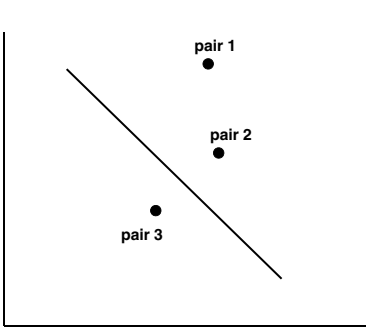

Fig. 1. Three pairs and their locations from the decision boundary of the SVM'ed-Kernel

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.

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 so that the similarity measure returned becomes always positive.

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:

$$f(pattern) = \sum_{i=1}^{l} \alpha_i y_i SVK(pattern, pattern_i) + b \tag{6}$$

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

$$SVK(pattern_x, pattern_y) = \sum_{j=1}^{l'} \alpha_j y_j k(m(x,y), m_j) + b' \tag{7}$$

Where, m_j 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 $pattern_x$ and $pattern_y$.
y_i denotes the label of the i^{th} pattern in the original training set.
y_j denotes the label of the j^{th} pair in the recreated training set.
b and b' denote constant offsets (or thresholds).
k denotes an arbitrary kernel function.
l and l' denote the number of training patterns in the original and the recreated training sets respectively.
The parameters ${\alpha_i}'s$ and ${\alpha_j}'s$ are computed as the solutions of quadratic programming problems.

5 Shape Recognition

In our previous work [10], a number of basic features were extracted from each shape for the purpose of serving as inputs to the classifier. The features were selected to be size and orientation independent. 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 [12];Alt, the area of the maximum area inscribed triangle that fits inside the convex hull of the shape [12];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 in the SVM model were: Pch^2/Ach, Ach/Aer, Alq/Aer, Alq/Ach, Alt/Alq, Alt/Ach, Plt/Pch,and Her/Wer. For the SVM'ed Kernel, we decided to make use of the features already designed before together while adding a novel high level similarity feature to show the power of the novel proposed kernel.

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 original feature vectors of the two input patterns. However, we have added a single similarity feature that represents the similarity between a pair of patterns. This feature was a modified chain code distance measure [15].

6 Testing Results

In our experiments, we trained the system using hand-drawn circles, triangles, rectangles, diamonds and ellipses from Refaat *et al* data set (2008) [10], we have added some new shapes to the set to increase its size. We divided the new extended data set into 750 patterns for training and we kept two test sets unseen. The first test set is a normal one of 236 patterns while the other one consists of 234 hard patterns. In the hard test set the shapes may be drawn similar to more than one shape class and it is the responsibility of the model to discover its true class. The hard test set was created in order to measure our models' robustness to hard patterns or shapes drawn carelessly. We used the SVM'ed Kernel with the pairwise classification method [1] to handle the multiclass problem. A recreated set was generated from the training set of each pair of classes. Each recreated set was then used to train an SVM'ed Kernel which was used subsequently as a kernel function for the corresponding binary classifier. All SVM'ed Kernels use the rbf kernel with the gamma parameter set to 2. We did not perform any tuning for the gamma of the rbf kernel used by the SVM'ed Kernels.

We compared the test accuracy of the SVM'ed Kernel to that achieved by Refaat *et al* (2008) SVM model and also to that achieved by using the rbf kernel with the pairwise classification method. In the last case, by trial and error, the gamma parameter was chosen to be set to 2. We used SVMlight [11] in all our simulations. Table 3 shows the testing accuracies of the three models for both the normal and the hard sets.

Table 3. Testing Accuracies

Test Set-Model	RBF gamma = 2 *pairwise*	Refaat 2008	SVM'ed Kernel *pairwise*
normal	81.355%	88.135%	92.796%
hard	61.96%	69.23%	85.89%

The testing results showed that the SVM'ed kernel outperforms Refaat *et al* 2008 in both the normal and the hard sets by about 4.6% and 16.5% respectively. The reason of this significant gain was that the SVM'ed kernel used the modified chain code similarity measure. This mutual feature could not have been used by the classical kernels because it represents a pair of patterns rather than only one. In addition, the SVM'ed kernel did not neglect the predefined classical features which made it act as a statistical integrator of all information about the task of shape recognition.

7 Conclusion and Future Work

In this paper, we proposed the application of the SVM'ed-Kernel function to the problem of shape recognition. We showed how the SVM'ed Kernel allows defining features of similarity between a pair of patterns. In addition, we showed that the old feature definitions for single patterns could also be used by just subtracting each two corresponding features. In the shape recognition problem, the enhancement was about 4.5 % in the normal test set and interestingly about 16.5% in the hard test set. In our future work, we are going to use the SVM'ed-Kernel in various real world applications in both Natural Language Processing and Bioinformatics.

References

1. Scholkopf, B., Smola, A.J.: Learning with Kernels. The MIT Press, Cambridge (2002)
2. Thadani, K., Jayaraman, A., Sundararajan, V.: Evolutionary Selection of Kernels in Support Vector Machines. In: International Conference on Advanced Computing and Communication (2006)
3. Kong, R., Zhang, B.: Autocorrelation Kernel Functions for Support Vector Machines. In: Third International Conference on Natural Computation (2007)
4. George, J., Rajeev, K.: SINC-CAUCHY Hybrid Wavelet Kernel for Support Vector Machines. In: IEEE Workshop on Machine Learning for Signal Processing. IEEE Press, Los Alamitos (2008)
5. Wang, D., Yeung, D., Eric, C.: Weighted Mahalanobis Distance Kernels for Support Vector Machines. IEEE Transaction on Neural Networks 18, 1453–1462 (2007)
6. Xu, J., Zhang, X.: Kernels Based on Weighted Levenshtein Distance. In: International Joint Conference on Neural Networks, pp. 3015–3018. IEEE Press, Budapest (2004)

7. Wu, X., Wang, J., Herbet, K.: A New Kernel Method for RNA Classification. In: Sixth IEEE International Symposium on BioInformatics and BioEngineering, Virginia, pp. 201–208 (2006)
8. Yan, C., Wang, Z., Gao, Q., Du, Y.: A Novel Kernel for Sequences Classification. In: IEEE International Conference on Natural Language Processing and Knowledge Engineering, Wuhan, pp. 769–773 (2005)
9. Cuturi, M., Vert, J.: A Mutual Information Kernel for Sequences. In: International Joint Conference on Neural Networks, pp. 1905–1910. IEEE Press, Budapest (2004)
10. Refaat, K., Helmy, W., Ali, A., Abdelghany, M., Atiya, A.: A New Approach for Context-Independent Handwritten Offline Diagram Recognition using Support Vector Machines. In: International Joint Conference on Neural Networks, pp. 177–182. IEEE Press, Hong Kong (2008)
11. Jaochims, T.: SVMlight is an implementation of Support Vector Machines in C
12. Boyce, J., Dobkin, D., Drysdale, R., Guibas, L.: Finding External Polygons. In: Annual Symposium on the Theory of Computing (1982)
13. Valveny, E., Martí, E.: Deformable template matching within a bayesian framework for hand-written graphic symbol recognition. In: Chhabra, A.K., Dori, D. (eds.) GREC 1999. LNCS, vol. 1941, pp. 193–208. Springer, Heidelberg (2000)
14. Notowidigo, M., Miller, C.: Offline sketch interpretation. In: AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural. Washington (2004)
15. Ahmad, M.B., Park, J.-A., Chang, M.H., Shim, Y.-S., Choi, T.-S.: Shape registration based on modified chain codes. In: Zhou, X., Xu, M., Jähnichen, S., Cao, J. (eds.) APPT 2003. LNCS, vol. 2834, pp. 600–607. Springer, Heidelberg (2003)
16. Microsoft software product for creating a wide variety of business and technical drawings, `http://www.office.microsoft.com`
17. Notowidigdo, M.: User-Directed Sketch Interpretation. MEng thesis, Massachusetts Institute of Technology (2004)