

ON THE USE OF A NEW ADDITIVE KERNEL FOR CHANGE DETECTION USING SVM

Tarek HABIB^{†,‡}, Jordi INGLADA[‡], Grégoire MERCIER^², Jocelyn CHANUSSOT[†]

[†] GIPSA-Lab (Signal and Images Department), CNRS, INP Grenoble,

[‡] Centre national d'études spatiales (Cnes - DCT/SI/AP),

^² Institut Telecom ; Telecom Bretagne / CNRS FRE 3167 LabSTICC/CID.

ABSTRACT

In the context of change detection and due to the multitude of change scenarios, the objective is to build a generic change detection system. For many technical and operational reasons the Support Vector Machines (SVM) algorithm is used. One of the crucial steps when using the SVM algorithm is the choice of the kernel function. With the lack of *a priori information* the choice of the kernel function may be difficult for the user. In this paper several techniques for constructing a suitable kernel function obtained from the data are proposed.

1. INTRODUCTION

In SVM optimization the degrees of freedom for the human operator are very scarce. Once the algorithm is initialized, the operation of the SVM is automatic. Before starting the SVM optimization, different parameters could be adjusted according to the need of the user and the type of SVM used. For example in *C - SVM* the *C* parameter could be either automatically computed or designated by the user. In the *fuzzy - SVM* the different weights are also computed by the user [1]. However the most important and crucial task is the choice of the kernel function. Choosing different kernel functions will produce different solutions for the optimization problems, and hence different SVMs, and may result in different performances.

In order to generalize the SVM equations for the case where the input data is not linearly separable, the notion of kernel functions was introduced. The use of a kernel function, provides much flexibility to the SVMs in the sense that using the kernel trick (*i.e.* only substituting the kernel function), the behavior of the SVMs can be completely altered without changing the initial structure and the optimization equations of the SVMs.

The kernel trick was first introduced in [2] to extend the Generalized Portrait hyperplane classifier to non-linear Support Vector Machines. The authors in [3] used it in the context of the potential function classification method to express the dot product between elements in a higher dimensional space in terms of elements of the input space. This procedure is intended to add much operational flexibility without increasing

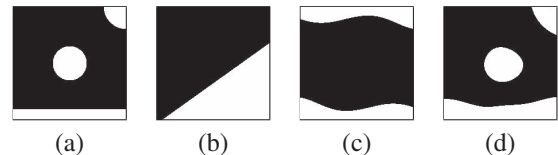


Fig. 1. (a) ground truth, classification results using (b) Linear kernel, (c) Polynomial kernel (degree = 4) and (d) radial basis kernel (gamma = 5).

the complexity of the system since only the vectors in the input space are used. With increase in flexibility the computed SVMs are more adapted to the learning data and should provide more accurate classification results.

In the literature, there exist several types of SVMs. Some of the most widely used are the polynomial SVMs and radial basis function SVMs. These correspond to simply using the appropriate kernel function, thus the polynomial SVM corresponds to the use of the polynomial kernel in the SVMs equations, and so on. With the lack of prior information on the available data and its separability, the difficulty resides in the choice of a specific kernel function that will provide the best classification results.

In order to highlight the importance of the use of an adequate kernel function, figure 1 shows different cases of kernels applied on the same data and the corresponding classification result. The test data is a synthetic two-band image, with the components of each band simply representing the coordinate of the vector along one of the two axis (*i.e.* the first band contains the x-axis coordinate and the second band contains the y-axis coordinate). The image is composed of 40000 vectors (200*200), and 100 randomly selected learning vectors were chosen for testing purposes. From the ground truth image it can be noted that there are three distinct areas representing the first class, learning vectors representing each one of these areas were among the randomly selected learning vectors.

From the above figure it can be noted that the choice of the kernel function highly affects the classification accuracy. For instance in the above example, the use of a radial basis kernel (classification accuracy = 92.12%) gives much better

results than the use of the polynomial kernel (classification accuracy = 80%) or the linear kernel (classification accuracy = 64.97%). This shows that the choice of the kernel function is a key step for obtaining adequate classification results.

Besides from the power of the mathematical tool that is the kernel, an interesting aspect of the use of these functions in the framework of the SVMs should be highlighted. Consider the decision function of the SVMs represented as follows:

$$f(x_{test}) = \text{sgn}\left(\sum_{i=1}^m \alpha_i y_i k(x_{test}, x_i) + b\right) \quad (1)$$

where: α_i and b are the solution of the SVMs equations, $y_i \in \{+1, -1\}$ is the vector's class, $x_i \in \mathbb{R}^n$ is a learning vector, m is the number of learning vectors, x_{test} is the new vector to be classified and $k(\cdot, \cdot)$ is the kernel function.

Assume that the kernel function used is the linear one, that is:

$$k(x_i, x_j) = \langle x_i, x_j \rangle \quad (2)$$

where $\langle \cdot, \cdot \rangle$ is the dot product

Since the scalar product between two vectors is simply the projection of the first vector onto the second one, hence the summation in equation 1 can be thought of as the weighted projection of the different learning vectors x_i into the direction of the new test vector x_{test} . The value of this weighted projection is then added to the threshold b , if the result is positive then the projections of the x_i 's with positive y_i 's are considered as *more resembling* to the new x_{test} and vice versa. Or in other words, the new vector x_{test} is considered as closer or in the same direction as the learning vectors with positive y_i 's. These ideas similarly applies to the non-linear kernels, since the role of the non-linear kernel is to compute this same dot product but into a space of higher dimension than that of the input space.

Hence apart from the fact that they are powerful mathematical tools that provides a high degree of flexibility to the SVMs without increasing their complexity, kernel functions can be considered as a similarity measure.

According to Mercer's theorem, reproducing kernels have to be symmetric positive definite functions. This is an interesting property since positive definite functions have pleasant algebra, and thus the obtainment of a new positive definite function from existing ones is relatively simple.

In this paper, positive definite functions' properties are used in order to develop additive kernels. Several techniques and cost measures are proposed with the objective of automatically building an adaptive kernel. This kernel is intended to overcome the lack of prior information concerning the data separability and hence the choice of the most adapted kernel function.

In section 2 the properties that were used to develop the additive kernels are highlighted. In section 3 the proposed

approach to automatically construct the kernels is described. In section 4 the results obtained using this approach are presented and finally the conclusions are drawn in section 5.

2. THEORETICAL BACKGROUND

As mentioned earlier, according to Mercer's theorem, reproducing kernels (*i.e.* Mercer kernels) are symmetric positive definite functions. Mercer proved that a symmetric positive definite function $f(x, y)$ satisfies the following condition:

$$\int f(x, y)g(x)g(y) \geq 0 \quad \forall g \in L^2 \quad (3)$$

Using this result, one can demonstrate that if $\psi(\cdot)$ is a function on \mathbb{R}^p and K' is a kernel on $\mathbb{R}^p \times \mathbb{R}^p$, then:

$$K(x, y) = K'(\psi(x), \psi(y)) \quad (4)$$

is a kernel. This is intuitive because the transformation function $\psi(\cdot)$ does not affect the properties of the kernel function itself, since it is applied on both of the input vectors.

Again using the result proved by Mercer, it can be proved that the non-negative linear combination of kernels is also a kernel. Combining both these properties, the following kernel combination is a valid Mercer kernel:

$$K(x, y) = \sum_{i=1}^M a_i K_i(\psi_i(x), \psi_i(y)) \quad (5)$$

The result presented in equation 5 is of a particular interest when using multiple features with different nature. For instance, if in the input features there are spectral, geometrical and textural feature groups then according to the specifications of each group of features, the most adequate kernel could be used. This procedure is discussed in the following section.

3. PROPOSED APPROACH

The aim of using a kernel function is to project the input data that is non-linearly separable in the input space into a higher dimensional space where they are linearly separable. If the kernel function is flexible enough, it will be able to perform this task for complex input feature vectors. In this section semi-automatic methods for the obtainment of adaptive additive kernels are presented. The objective is the increase of the classification accuracy by increasing the kernel's adaptability to the available data.

The additive kernel as presented in equation 5 is composed of a set of *sub-kernels* shown by the kernels K_i , $i = 1, \dots, M$. Each one of these M kernels operates on a subset of the available input features.

The proposed kernel construction technique as described here is semi-automatic since it requires a human operator's

intervention for the algorithm initialization. The role of the human operator is to identify the following:

1. Group the features according to their nature, this is an intuitive procedure based on the operator's experience. This means that the operator should group the features that he judges having a similar behavior (e.g. increases linearly with a specific class, decreases logarithmically with another class, ...).
2. Specify the different types of kernels that are required for testing. Based on the performance of these different kernels during the learning phase, the additive kernel will be constructed.

The generic framework for the construction of the additive kernel works as follows: 1. Initialization, 2. test kernels on subsets, 3. select the best kernel/subset, 4. build the additive kernel.

Initialization: This block is the responsibility of the human operator where he identifies the feature subsets as well as the kernels to be tested.

Apply kernels on subsets: This block *learns* all the subsets using every kernel identified by the operator.

Select the best kernel/subset: Based on the performance during this learning phase and using the suitable measure, identify the best kernel for every subset.

Construct the additive kernel: The additive kernel will be finally constituted of the addition of the kernels identified from the last block.

In this paper several measures for the choice of the most suitable kernel were tested, namely:

- The generalization error in the leave-one-out sense: this is the measure provided by the $\xi\alpha$ -estimator of SVM-light [4].
- The number of misclassified vectors during the learning phase.
- A combination of different parameters (KCost):

$$KCost = \frac{\text{Nb.Mis.} * \text{Nb.SV}}{|w|} \quad (6)$$

Where Nb.Mis. is the number of misclassified vectors during the learning phase, Nb.SV is the number of the obtained support vectors and $|w|$ is the margin of separation between the two classes.

The kernel with the lowest $KCost$ is considered as the best for the given subset. The $KCost$ makes the hypothesis that a kernel that produces a high number of support vectors is not capable of *simply* separating the data vectors. Also the high number of support vectors is penalized since it will require a high classification processing time during the classification phase.

- A combination of different parameters including time constraints (KTCost):

$$KTCost = \frac{\text{Nb.Mis.} * \text{Nb.SV} * \text{time}}{|w|} \quad (7)$$

Where time is the cpu-units passed during the learning phase using the kernel.

This is the same measure as the $KCost$ but it takes into consideration the processing time during the learning phase, since it is indicative of the required learning time that will be required by the additive kernel on the following phase.

In the following section the results obtained using the different schemes are presented.

4. RESULTS

The kernel construction algorithm was first tested on the Goma data set. For this data set two images are available, the *before* and the *after* image, and the goal is to identify the zones that have undergone an abrupt change between the two images. Hence the objective is to classify the data set into two classes (*i.e.* change and no change) using the available images. Since using the original imagery to properly classify the image does not provide the required accuracy, a set of features were computed using the original images. The features are: difference, ratio, ratio of means, ratio of medians correlation, mean squares, Kullback-Leibler distance, mutual information, cardinality match, gradient difference, entropy and energy.

These features were divided into three groups as follows:

Group 1: Difference, ratio, ratio of means and ratio of medians.

Group 2: Correlation, mutual information, entropy and energy.

Group 3: Mean squares, Kullback-Leibler distance, cardinality match and gradient difference.

For testing purposes, 900 random training vectors were chosen for each iteration, which is 0.28 % of the total number of vectors (*i.e.* the image size is 320000 vectors). The additive kernel was constructed using the most commonly used kernels, namely the linear the polynomial and the radial basis function kernels. A combination of several kernels of the same type could be also used, for example a number of radial basis function kernels with different γ or polynomials of different degrees. However in this paper the spot was made on the combinations that could be obtained using the linear, the polynomial and the radial basis function kernels whose functions and fixed parameters are shown as follows:

$$\begin{aligned}
\text{Linear: } K(X, Y) &= \langle X, Y \rangle \\
\text{Polynomial: } K(X, Y) &= (\langle X, Y \rangle)^4 \\
\text{Radial basis: } K(X, Y) &= \exp(-0.5\|X - Y\|^2)
\end{aligned}$$

After generating random masks, table 1 shows the mean performance of the additive kernel. The mean was calculated on 10 different iterations, generating a new random mask for each one of them.

Table 1. Performance assessment on the Goma dataset

Kernel function	Accuracy	Precision
Linear	81.19%	33.65%
Polynomial (degree = 4)	72.64%	19.14%
RBF ($\gamma = 0.5$)	77.2%	27.82%
Add.(Generalization error)	76.9%	43.75%
Add.(Misclassified)	74.51%	46.88%
Add.(KCost)	86.32%	28.47%
Add.(KTCost)	65.22%	33.64%

From the inspection of these results it can be shown that however the criteria the performance of the additive kernel is acceptable. It can be noted that using the KCost criteria for constructing the additive kernel provides the best average accuracy.

Following the same procedure, the additive kernel technique was tested on a synthetic data set. Instead of the different measures as calculated for the Goma case, this dataset is built using one single ground truth image of size 320000 pixels from which 20 different images were derived. Each new image is a "noised" version of the ground truth image. The added noise, is additive gaussian noise with different standard deviations. The feature groups are the composed by randomly selected images. However the same number of groups (*i.e.* 3) and the same number of features per group (*i.e.* 4) was selected to match that of the Goma test.

The results obtained when applying the proposed approach (using 10 iterations) are shown in the following table 2.

Table 2. Performance assessment on the Synthetic dataset

Kernel function	Accuracy	Precision
Linear	85.476%	70.61%
Polynomial (degree = 4)	85.656%	70.80%
RBF ($\gamma = 0.5$)	85.574%	70.80%
Add.(Generalization error)	85.128%	78.78%
Add.(Misclassified)	85.262%	71.67%
Add.(KCost)	85.356%	71.35%
Add.(KTCost)	85.528%	71.27%

From the above results, it can be noted that all the applied kernels provide almost the same performance. This shows that even if the dataset does not favor a certain kernel type over the other, the use of the additive kernel can guarantee the solution's stability.

5. CONCLUSION

In this paper different additive kernel construction algorithms were proposed. The aim is to increase the adaptability of the kernel function onto available data in order to increase the classification accuracy. From a theoretical point of view, the KTCost measure is the most adapted measure, since it is a complete indicator that takes into consideration the classification accuracy during the learning phase as well as the operational time. However from the experimental results, this measure did not provide the best results, this may be due to the way the *time* factor is considered, a more suitable way for integrating the time considerations into the measure is studied (*i.e.* using $\log(\text{time})$ instead of a direct multiplication by the time for instance). The results also show that whatever the criteria chosen for the additive kernel, the results are acceptable and offers good stability when no prior information concerning the kernel to be chosen is available. However other considerations and difficulties that were encountered during the testing phase include 1. the relatively long time needed for the convergence of the additive kernel (*i.e.* since the available SVM algorithms are not optimized for the use of such custom kernels), and 2. problems of convergence where the learning algorithm can not converge for a solution of the SVM optimization problem. These difficulties are to be considered in future studies.

6. REFERENCES

- [1] Kui Wu and Kim-Hui Yap, "Fuzzy SVM For content-based image retrieval: a pseudo-label support vector machine framework," in *IEEE Computational Intelligence Magazine*, vol. 1, no. 2, May 2006, pp. 10–16.
- [2] B. E. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Computational Learning Theory*, 1992, pp. 144–152.
- [3] M.A. Aizerman and É.M. Braverman and L.I. Rozonoér, "Theoretical foundations of the potential function method in pattern recognition learning," in *Automation and Remote Control*, 1964, pp. 25:821–837.
- [4] Thorsten Joachims, "Estimating the generalization performance of a SVM efficiently," in *Proceedings of ICML-00, 17th International Conference on Machine Learning*, P. Langley, Ed. Stanford, US: Morgan Kaufmann Publishers, San Francisco, US, 2000, pp. 431–438.