Improving SVM Performance Using a Linear Combination of Kernels

Laura Dioşan[1,2], Mihai Oltean[1], Alexandrina Rogozan[2], and Jean-Pierre Pecuchet[2]

[1] Computer Science Department, Babeş-Bolyai University, Cluj-Napoca, Romania
{lauras, moltean}@cs.ubbcluj.ro
[2] LITIS, Institut National des Sciences Appliquées, Rouen, France
{arogozan, pecuchet}@insa-rouen.fr

Abstract. Standard kernel-based classifiers use only a single kernel, but the real-world applications and the recent developments of various kernel methods have emphasized the need to consider a combination of multiple kernels. We propose an evolutionary approach for finding the optimal weights of a combined kernel used by the Support Vector Machines (SVM) algorithm for solving some particular problems. We use a genetic algorithm (GA) for evolving these weights. The numerical experiments show that the evolved combined kernels (ECKs) perform better than the convex combined kernels (CCKs) for several classification problems.

1 Introduction

Various classification techniques have been used for assigning the correct labels associated to each input item. Kernel-based techniques (such as Support Vector Machines (SVMs) [1]) are one of the intensively explored classifiers used for solving this problem. Standard kernel-based classifiers use only a single kernel, but the real-world applications and the recent developments of various kernel methods have emphasized the need to consider a combination of kernels. If this combination is a linear one, than one has to determine the weights associated to each kernel.

The optimal values for these weights could be determined with different mathematical methods (such as Sequential Minimal Optimisation (SMO) [2], Semidefinite Programming [3]). Recently, the evolutionary methods (such as Evolutionary Algorithms (EAs) [4]) were used as an alternative approach for solving various optimization problems. EAs can solve most problems without taking into account the constraints regarding the continuity and the derivability of the functions that encode the core of the problems. Also, the evolutionary methods can be faster than the convex techniques in some situations (for problems with many variables or many instances). The convex technique can solve very well these problems, but the running time is larger than that of evolutionary approach. Even if the evolutionary techniques are able to find in some cases only an approximation of solution, this approximation is quickly found, compared to the convex tools. It is a trade-off between the solution accuracy and the computing time.

Therefore, EAs could be used for finding the optimal weights of a combined kernel. In our research we have used Genetic Algorithms (GAs) [5] for evolving the kernel weights in order to obtain a more complex kernel. A real encoding is used for representing the GA chromosomes. Each gene of a chromosome is associated to a kernel weight. For computing the fitness of a chromosome, we embed the complex kernel into a SVM algorithm and we run this algorithm for a particular classification problem. The accuracy rate computed by the SVM algorithm (on a validation set) represents the quality of the current chromosome. The accuracy rate (on the test dataset) computed by the SVM algorithm, which uses the evolved kernel, is compared to those computed by a single-kernel SVM algorithm for several classification problems. Numerical experiments show that an ensemble of multiple kernels is always superior to individual kernels, for the considered problems, in terms of classification accuracy (the number of correctly classified items over the total number of items). The obtained results also show the ability of the GAs to evolve better weights for the combination of kernels then the convex method (proposed in [3], [6]).

The paper is organized as follows: Sect. 2 describes some related work in the field of combined kernel generation. Section 3 describes the proposed technique for evolving combined kernels. This is followed by a special section (Sect. 4) where the results of the experiments are presented. Finally, Sect. 5 concludes.

2 Related Work

Only two attempts for finding the weights of a combined kernel for SVM algorithm were found in the literature. Recently, Lanckriet et al. [3] and Sonnenburg [6] considered the conic combinations of kernel matrices for the SVM and showed that the optimization of the coefficients of such a combination reduces to a convex optimization problem known as a quadratically constrained quadratic program. They shown how the kernel matrix can be learnt from data via semi-definite programming [3] and via semi-infinite linear programming [6]. The other attempt is also made by Lanckriet [7] and it is an improved approach of [3]. He proposed a novel dual formulation of the QCQP as a second-order cone-programming problem.

Both approaches are matrix-based techniques because, in both cases, the algorithms learn the kernel matrix (or the Gram matrix) associated to the multiple kernel. The model selection can be viewing in terms of Gram matrices rather than kernel functions.

Our model can be considered an evolutionary alternative to these approaches. We will call, in what follows, the learnt kernel of Lanckriet and Sonnenburg *convex combined kernel (CCK)* – because it is found using a convex method – and we will call our kernel *evolved combined kernel (ECK)* – because it is learnt using evolutionary techniques.

3 Proposed Model

3.1 Representation

Standard SVM algorithm works with a particular kernel function, which is empirically fixed, on the problem, but the real-world applications and the recent developments of various kernel methods have emphasized the need to consider a combination of multiple kernels. A combined kernel can perform a more fine transformation of the initial data space into a larger (with more dimensions) linear separable space.

Following the Lanckriet approach [3], the combined kernel can be obtained as a linear combination of basic kernels:

$$K^* = \sum_{i=1}^{k} w_i \times K_i \tag{1}$$

where k represents the cardinal of the considered kernel set, K_i the i^{th} kernel and w_i the weight of the kernel K_i in the linear combination, $i \in \{1, \ldots k\}$. Because the obtained combination must be a SVM kernel function, it has to satisfy the Mercer conditions regarding the positivity and the symmetry of Gram matrix. In [3] is proved that the combination from (1) is a SVM-adapted kernel only if the sum of weights is equal to the number of kernels embedded in that combination and if each weight is less or equal to the number of kernels.

There are two possibilities for choosing these weights: a general approach (see (2)) and a particular approach with non-negative weights (see (3)):

$$w_i \in [-k, k], \text{with} \sum_{i=1}^{k} w_i = k \tag{2}$$

$$w_i \in [0, k], \text{ with } \sum_{i=1}^{k} w_i = k \tag{3}$$

In our model we used a GA [5] for evolving the kernel weights involved into a linear combination. Each GA individual is a fixed-length string of genes. The length of a chromosome is equal to the number of standard kernels that are involved into the combination. Each gene is a real number from $[0, 1]$ range. Because each gene must be associated with a kernel weight, several transformations must be performed:

1. Transform each gene g_i into a gene-weight gw_i using the formula:

$$gw_i = \frac{g_i}{\sum_{j=1}^{k} g_j}, i = \overline{1, k} \tag{4}$$

2. Scale each gene-weight into the corresponding weight domain. Here, we used two approaches that correspond to those presented by (2) and (3):

– for obtaining the correct weights of a general linear combination with coefficients (negative or positive) whose sum is equal to k, we must scale each gene-weight using the formula:

$$w_i = -k + gw_i \times 2 \times k, i = \overline{1,k} \tag{5}$$

– for obtaining the correct weights of a particular linear combination with non-negative coefficients whose sum is equal to k, we must scale each gene-weight using the formula:

$$w_i = gw_i \times k, i = \overline{1,k} \tag{6}$$

3.2 Model

The proposed approach is a hybrid technique structured on two levels: a macro level and a micro level. The macro level is a GA that evolves the kernel weights for a combined kernel. The micro level is a SVM algorithm used for computing the quality of a GA individual on the validation dataset.

When we compute the quality of a GA chromosome we actually have to compute the quality of the combined kernel whose weights are encoded in that individual. For assessing the performance of a combined kernel we have to embed that kernel within a SVM algorithm and we have to run the obtained algorithm for a particular problem (classification problem in our case). The accuracy rate computed by the SVM algorithm (on the validation set) represents the quality of a GA individual.

3.3 Algorithms

The algorithms used for evolving the kernel coefficients are described in this section. As we said before, we are dealing with a hybrid technique which has a macro level and a micro level.

Macro-level Algorithm. The macro level algorithm is a standard GA [5] used for evolving the kernel's coefficients. We use steady-state evolutionary model [8] as underlying mechanism for our GA implementation. The GA starts by creating a random population of individuals. The following steps are repeated until a given number of generations is reached: two parents are selected by using a standard selection procedure. The parents are recombined in order to obtain two offspring by using an uniform arithmetical crossover [9]. The offspring are considered for a Gaussian mutation [10], [11]. The best offspring O replaces the worst individual W in the current population if O is better than W.

Micro-level Algorithm. The micro level algorithm is a SVM algorithm [12], [13], [14] used for computing the fitness of each GA individual from the macro level. The SVM algorithm[1] solves a binary classification problem. As we know, the

[1] It is taken from libsvm [12].

method can be easily extended to multi-classes classification problems, but we performed the numerical experiments for binary classification problems because our approach could be an alternative to Lanckriet approach [3] and we wanted to performed a fairly comparison of the obtained results.

Original implementation of the SVM algorithm from [12] uses one out of the standard kernels (linear, polynomial and Radial Basis Function (RBF) - see Table 1). In more cases, the choice of a kernel is empirically performed and it is not problem-adapted. The algorithm proposed in this paper uses a combined kernel (whose coefficients are encoded in the GA chromosome) and a modified version of SVM implementation proposed in [12]. For computing the quality of each GA individual, we run the SVM embedding the combined kernel. The accuracy rate obtain by the SVM algorithm on the validation dataset will represent the quality of that combined kernel. Figure 1 presents the main structure of our model.

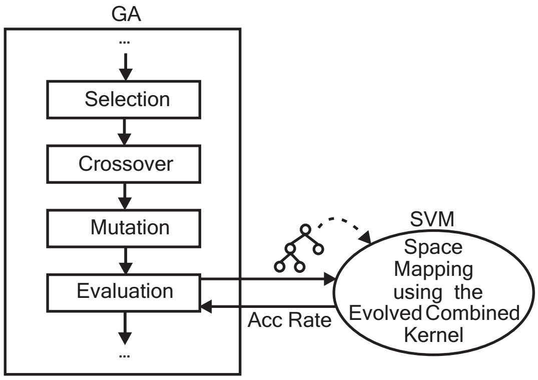

Fig. 1. The sketch of the hybrid approach

4 Experiments

4.1 Test Problems

The datasets were obtained from UCI repository [15]. Each dataset contains instances labelled with one of two class labels (we must solve a binary classification problem). The *breast* dataset contains 683 instances, the *heart* 270 instances, the *ionosphere* 351 instances and the *sonar* dataset contains 208 instances. Each data set was randomly divided into two sub-sets: learning sub-set (80%) and testing sub-set (20%). The learning sub-set was randomly partitioned into training (2/3) and validation (1/3) parts. In Fig. 2 is depicted this partitioning of a dataset.

The SVM algorithm uses the training subset for learning the model of the SVM and the validation subset for computing the accuracy rate. The best GA chromosome - the best evolved combined kernel (ECK^*) - found during learning stage is tested by running a SVM algorithm on the test subset.

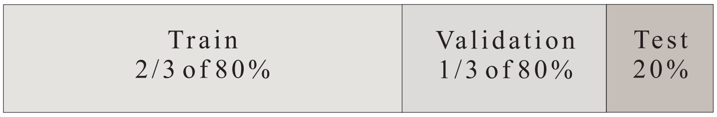

Fig. 2. Partitioning of a dataset

4.2 Numerical Experiments

Experiment 1. In this experiment, a single-objective GA is used for evolving the kernel coefficients. We used a real encoding for GA chromosomes representation that corresponds to the details presented into Sect. 3.1. Also, the transformation presented in that section (see (4), (5) and (6)) are perform in order to obtain valid kernel weights within the corresponding range.

We used a GA population with 20 individuals that are evolved during 50 generations. For obtaining a new generation, we performed a binary tournament selection, a convex crossover and a Gaussian mutation. The values for crossover and mutation probabilities ($p_c = 0.8$ and $p_m = 0.1$) were chosen for ensuring a good diversity of the population. The crossover and mutation type is specific for real encoding and the values used for the population size and for the number of generation were empirically chosen based on the best results obtained during several tests performed with different values for these parameters.

The standard kernels embedded into the combined kernel K^* are the linear kernel, the polynomial kernel and the Radial Basis Function (RBF) kernel. These kernels (and their parameters) were chosen for comparison purposes with the results obtained by Lanckriet. Therefore, each GA chromosome will contain three genes, one for each kernel weight. Table 1 contains the expressions of these kernels and the values of their parameters used in our experiments.

Table 1. Kernels parameters

Kernel name	**Kernel expression**	**Kernel parameters**
Linear	$K_1(u, v) = u^T \times v$	-
Polynomial	$K_2(u, v) = (\gamma \times u^T \times v + coef)^d$	$\gamma = 0.1$, $coef = 1$, $d = 3$
RBF	$K_3(u, v) = \exp -\gamma \times \lvert u - v \rvert^2$	$\gamma = 0.1$
ECK	$K^*(u, v) = \sum_{i=1}^{k} w_i \times K_i(u, v)$	$k = 3$, $w_i \in [a, b]$

Two sets of experiments were performed for evolving the weights w_i, ($i \in \{1, 2, 3\}$). In each experiment, we choose a different range for these weights:

– $w_i \in [0, k]$ and $\sum_i^k w_i = k$ (obtained according to the transformations presented in (4) and (6)) – we denote the obtained evolved combined kernel with $ECK_{[0,k]}$;

- $w_i \in [-k, k]$ and $\sum_i^k w_i = k$ (obtained according to the transformation presented in (4) and (5)) – we denote the obtained evolved combined kernel with $ECK_{[-k,k]}$.

The accuracy rates obtained on each test dataset and for each kernel are presented in Table 2. The results from columns that correspond to different ECKs represent the accuracy rate obtained by running a SVM algorithm (on the test dataset) which used the ECK encoded into the best GA individual (found during learning stage). In this experiment, each SVM algorithm (with a standard or with a combined kernel) is trained and tested on the same dataset.

Table 2. The accuracy rate (*Acc*) computed on the test dataset (TDS) by a SVM algorithm that uses different kernels (K): K_1, K_2, K_3, $ECK^*_{[0,k]}$ and $ECK^*_{[-k,k]}$, respectively. $ECK^*_{[0,k]}$ and $ECK^*_{[-k,k]}$ mean the best $ECK_{[0,k]}$ and $ECK_{[-k,k]}$ obtained during the learning stage. The weights of the ECKs are also given.

Dataset	K_1	K_2	K_3	$ECK^*_{[0,k]}$	$ECK^*_{[-k,k]}$	
breast	98.5401	98.5401	97.8102	98.5401	100	$Acc(\%)$
	–	–	–	1.30/0.25/1.45	2.02/-0.11/1.08	weights
heart	83.3333	81.4815	79.6296	85.1852	87.037	$Acc(\%)$
	–	–	–	0.27/2.03/0.70	-1.27/2.23/2.04	weights
ionosphere	97.1831	95.7746	84.507	98.5915	98.5915	$Acc(\%)$
	–	–	–	0.03/0.08/2.89	0.90/1.78/0.32	weights
sonar	57.1429	61.9048	0	61.9048	76.1905	$Acc(\%)$
	–	–	–	0.35/0.53/2.12	1.51/-0.02/1.51	weights

The results from Table 2 show that ECK performs better than a simple one. $ECK^*_{[0,k]}$ outperforms the standard kernels for two datasets and equalize the performance of a particular kernel for the other two problems (*breast* and *sonar*). Unlike this kernel, the $ECK^*_{[-k,k]}$ outperforms the standard kernels for all datasets (in *breast* case it classifies correctly all the test dataset instances).

Moreover, the evolved combined kernel whose weights are from a large range ($ECK^*_{[-k,k]}$) performs better than the evolved combined kernel whose weights are in $[0, k]$ range ($ECK^*_{[0,k]}$) because it allows a better adaptability of the combined kernel to the classification problem (the $[-k, k]$ search space is larger than $[0, k]$ interval =- from the computer point of view).

Experiment 2. We also compared our results against those obtained by Lanckriet [3] with a combined kernel learnt with convex methods (CCK). We computed the performance improvement for each dataset and for each kernel type as a percent difference Δ between the accuracy rate computed by the SVM algorithm with a combined kernel (Acc_{CK}) - evolved or not - and the accuracy rate computed by an algorithm with a standard kernel (Acc_{SK_i}):

$$\Delta_i = \frac{Acc_{CK} - Acc_{SK_i}}{Acc_{SK_i}}, i = \overline{1,k} \tag{7}$$

The obtained differences (from Table 3) show that the $ECK^*_{[-k,k]}$ improves the SVM performance in all cases, unlike the $CCK_{[-k,k]}$ which decreases the SVM performance in two cases (*breast* and *sonar*). Moreover, the ECK performance improvements are better than those obtained with the CCK in 6 cases (out of 12).

Table 3. The Δ values. The table presents comparatively the performance improvement of the combined kernels with general weights ($ECK^*_{[-k,k]}$ and $CCK_{[-k,k]}$).

Dataset	$ECK^*_{[-k,k]}$			$CCK_{[-k,k]}$		
	K_1	K_2	K_3	K_1	K_2	K_3
breast	1%	1%	2%	9%	-1%	7%
heart	4%	7%	9%	1%	7%	43%
ionosphere	1%	3%	17%	14%	0%	3%
sonar	33%	23%	0%	15%	8%	-1%

Experiment 3. For a complete analysis of the ECKs performance, we tested the generalization ability of the ECKs. For this experiment we use the one of the previous ECKs obtained for a particular dataset. This combined kernel is embedded into a SVM algorithm which is run against other 3 datasets. The results obtained by running the SVM with the ECK based on *sonar* dataset – for both weight types (second column and third column) – and with the standard kernels (next three columns) are presented in Table 4.

Table 4. Generalization ability of the evolved kernel. *sonar* dataset [15] was used as training set and the other datasets were used as test dataset.

Dataset	$ECK^*_{[0,k]}$	$ECK^*_{[-k,k]}$	K_1	K_2	K_3
breast	97.8102	99.2701	98.5401	98.5401	97.8102
heart	79.6296	85.1852	83.3333	81.4815	79.6296
ionosphere	97.1831	92.9577	97.1831	95.7746	84.5070
sonar	**61.9048**	**76.1905**	57.1429	61.9048	0.0000

Table 4 show that the $ECK^*_{[0,k]}$ performs similarly with the standard kernels in 2 cases, but the $ECK^*_{[-k,k]}$ increases the SVM performance for all problems and for all kernel pairs (evolved, standard). Therefore, these results can be considered an empirically proof of the good ECKs generalization ability.

5 Conclusions

A hybrid technique for evolving kernel weights has been proposed in this paper. The model has been used for evolving the weights of a combined kernel embedded into a SVM algorithm that solves a binary classification problem.

We have performed several numerical experiments for comparing our ECKs to others kernels (simple and combined – whose weights are determined by convex methods). Numerical experiments have shown that the ECKs perform similarly and sometimes even better than the standard kernels (linear, polynomial and RBF kernels). Moreover, the ECKs performance also outperform the CCKs (obtain in [3]) for some of the test problems. Our evolved kernels are also more robust in comparison with that of Lanckriet (which works worst than standard kernels in some cases).

However, taking into account the No Free Lunch theorems for Search [16] and Optimization [17] we cannot make any assumption about the generalization ability of the evolved kernel weights. Further numerical experiments are required in order to assess the power of the evolved kernels.

Further work will be focused on:

– evolving better-combined kernels
– using multiple data sets or a multi-objective evolutionary approach for the training stage
– using Genetic Programming [18] technique for obtained more complex combined kernels.

References

1. Joachims, T.: Making large–scale SVM learning practical. In: Advances in Kernel Methods — Support Vector Learning, MIT Press (1999) 169–184
2. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, MSR (1998)
3. Lanckriet, G.R.G., et al.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research **5** (2004) 27–72
4. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press (1995)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley (1989)
6. Sonnenburg, S., Rtsch, G., Schfer, C., Schlkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research **7** (2006) 1531–1565
7. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Machine Learning, Proceedings of ICML 2004, ACM (2004)
8. Syswerda, G.: A study of reproduction in generational and steady state genetic algorithms. In: Proc. of FOGA, Morgan Kaufmann Publishers (1991) 94–101
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1992)
10. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming made faster. IEEE-EC **3**(2) (1999) 82

11. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. John Wiley & Sons (1966)
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
13. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (2000)
14. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
15. Newman, D., Hettich, S., Blake, C., Merz, C.: Uci repository of machine learning databases (1998)
16. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute (1995)
17. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 67–82
18. Koza, J.R.: Genetic programming II: automatic discovery of reusable programs. MIT Press (1994)