An evolutionary method for constructing complex SVM kernels

DANA SIMIAN
Computer Science Department
Faculty of Sciences
University “Lucian Blaga” Sibiu
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu
ROMANIA
d_simian@yahoo.com

FLORIN STOICA
Computer Science Department
Faculty of Sciences
University “Lucian Blaga” Sibiu
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu
ROMANIA
florin.stoica@ulbsibiu.ro

Abstract: - The aim of this paper is to construct and analyze multiple SVM kernels. The construction is based on a genetic algorithm which uses a new co-mutation operator called LR-Mijn, capable of operating on a set of adjacent bits in one single step.

Key-Words: - SVM kernel, Genetic algorithms, Co-mutation

1 Introduction
In recent years SVMs have gained increasing attention and have become a very popular tool for machine learning tasks. Applications of SVM have been done in various fields for accomplishing tasks as classification, regression and novelty detection ([14]).

The task of classification is to find a rule, which based on external observations assigns an object to one of several classes. A classification task supposes the existence of training and testing data given in the form of data instances. Each instance in the training set contains one target value, named class label and several attributes named features. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes. Training involves optimization of a convex cost function If the data set is separable we obtain an optimal separating hyperplane with a maximal margin. In order to avoid the difficulties for the non separable data the kernel method is used. The kernel method is a very powerful idea. Using an appropriate kernel, the data are projected in a space with higher dimension in which they are separable by a hyperplane ([2], [14]). Under certain conditions, kernel functions can be interpreted as representing the inner product of data objects implicitly mapped into a nonlinear feature space. The "kernel trick" is to calculate the inner product in the feature space without knowing explicit the mapping function. There are many standard kernels: linear, polynomial, RBF, sigmoidal ([2]). Standard kernel-based classifiers use only a single kernel, but practical applications require consideration of a combination of kernels ([2], [3]). Recent developments are oriented in finding complex kernels and studying their behavior for different problems. In [3], [4], [11], are presented methods for solving this problem, using multiple kernels and genetic programming techniques.

The aim of this paper is to construct and analyze multiple SVM kernels. The construction is based on a genetic algorithm which uses a new co-mutation operator called LR-Mijn, capable of operating on a set of adjacent bits in one single step.

In present, there is a major interest in design of powerful mutation operators, in order to solve practical problems which can not be efficiently resolved using standard genetic operators. These new operators are called co-mutation operators. In [7] was presented a co-mutation operator called Mijn, capable of operating on a set of adjacent bits in one single step. In [13] we introduced and studied a new co-mutation operator which we denoted by LR-Mijn and we proved that it offers superior performances than Mijn operator.

The paper is organized as follows. In Section 2 we make a brief presentation of SVM and kernel method. We also present the basic idea of evolutionary method adopted for multiple SVM kernel construction. In section 3 is introduced the co-mutation operator LR-Mijn. The evolutionary algorithm based on LR-Mijn operator which we use for multiple kernel construction is presented in section 4. Section 5 contains the main results: the model for constructing multiple SVM kernels and the experimental results. Conclusions and further directions of study can be found in section 6.

2 Support Vector Machines and kernels

2.1 Problem definition
SVM algorithm can solve the problem of binary or multiclass classification. There are many known
methods to generalize the binary classifier to a n - class classifier ([14]). Therefore we consider in this section only the problem of binary classification.

Let be given the data points \( x_i \in \mathbb{R}^n \), i=1,\ldots,m and their labels \( y_i \in \{ -1, 1 \} \). We are looking for a decision function \( f \), which associates to each input data \( x \) its correct label \( y = f(x) \). The form of the decision function is

\[
f(x) = \text{sign}(\langle w, x \rangle + b) \quad (2.1)
\]

If the data set is separable then the conditions for classification without training error are

\[
y_i (\langle w, x_i \rangle + b) > 0 \quad (2.2)
\]

For the data sets which are not linearly separable we use the kernel method, which makes a projection of the input data \( X \) in a feature Hilbert space \( F \):

\[
\phi : X \to \mathcal{F}; \quad x \to \phi(x) \quad (2.3)
\]

The functional form of the mapping \( \phi(x_i) \) does not need to be known. It is implicitly defined by the choice of kernel:

\[
K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad (2.4)
\]

The kernel represents the inner product in the higher dimensional Hilbert space of features. Feasible kernels must satisfy Mercer’s conditions.

### 2.2 Multiple kernels

Usually the choice of the kernel is made empirically and the standard SVM classifiers use a single kernel. Recent papers proved that multiple kernels give better results than the single ones. It is known from the algebra of kernels that the set of operations

\[
(+, *, \exp) \quad (2.5)
\]

preserves the Mercer’s conditions and therefore we can obtain multiple kernels using these operations. One possibility is to use a linear combination of simple kernels and to optimize the weights ([11]). For optimization the weights two different kind of approaches can be found. One of them reduces the problem to a convex optimization problem. Other uses evolutionary methods for optimizing the weights. In [3] and [11] a hybrid approach using a genetic algorithm and a SVM algorithm is proposed. Every chromosome codes the expression of a multiple kernel. The quality of a chromosome is represented by the classification accuracy (the number of correctly classified items over the total number of items) using the multiple kernel coded in this chromosome and it is obtained running the SVM algorithm. The hybrid techniques from [3] is structured in two levels: a macro level and a micro level.

The macro level is represented by the genetic algorithm which builds the multiple kernels. The micro level is represented by the SVM algorithm which computes the quality of chromosomes. The accuracy rate is computed by the SVM algorithm on a validation set of data.

### 3 The LR-Mijjn operator

In this section we define the co-mutation operator called LR-Mijjn. Our LR-Mijjn operator finds the longest sequence of \( \sigma_p \) elements, situated \( \text{in the left or in the right of the position } p \). If the longest sequence is in the left of \( p \), the LR-Mijjn behaves as Mijjn, otherwise the LR-Mijjn will operate on the set of bits starting from \( p \) and going to the right.

Let us consider a generic alphabet \( \mathcal{A} = \{ a_1, a_2, \ldots, a_s \} \) composed by \( s \geq 2 \) different symbols. The set of all sequences of length \( l \) over the alphabet \( \mathcal{A} \) will be denoted with \( \sum = \mathcal{A}^l \).

In the following we shall denote with \( \sigma \) a generic string, and \( \sigma = \sigma_{l \ldots 0} \in \sum = \mathcal{A}^l \), where \( \sigma_q \in \mathcal{A} \lor q \in \{ 0, \ldots, l-1 \} \). Through \( \sigma(q,i) \) we denote that on position \( q \) within the sequence \( \sigma \) there is the symbol \( a_i \) of the alphabet \( \mathcal{A} \).

\( \sigma_{p,j}^z \) denotes the presence of \( z \) symbols \( a_i \) within the sequence \( \sigma \), starting from the position \( p \) and going left and \( \sigma(p,i) \) specify the presence of symbol \( a_i \) on position \( p \) within the sequence \( \sigma \), between right symbols \( a_{n} \) on the right and left symbols \( a_{m} \) on the left. We suppose that \( \sigma = \sigma(l-1) \ldots \sigma(p+\text{left}+1,m) \sigma(p,i) \sigma(p-\text{right}-1,n) \ldots \sigma(0) \).

The Mijjn operator is the mutation operator defined in [7]:

\[
\text{Mijjn} : \sigma = \Sigma, \quad p \in \{ 0, \ldots, l-1 \} \to \sigma' = \Sigma' \subset \Sigma, \quad \text{where } p \text{ is randomly chosen}
\]

(i) \( \sigma = \sigma_{l \ldots 0} \sigma_{p+\text{left}+1,m} \sigma_{p,i} \sigma_{p-\text{right}-1,n} \ldots \sigma(0) \)

\[
\text{Mijjn} \to \quad \sigma' = \sigma_{l \ldots 0} \sigma_{p+\text{left}+1,m} \sigma_{p,i} \sigma_{p-\text{right}-1,n} \ldots \sigma(0), \quad \text{for } n < l - p + 1
\]

and

(ii) \( \sigma = \sigma_{p,j}^z \sigma_{p+\text{left}+1,m} \sigma_{p,i} \sigma_{p-\text{right}-1,n} \ldots \sigma(0) \)

\[
\text{Mijjn} \to \quad \sigma' = \sigma_{p,j}^z \sigma_{p+\text{left}+1,m} \sigma_{p,i} \sigma_{p-\text{right}-1,n} \ldots \sigma(0), \quad \text{for } n = l - p + 1
\]

with \( a_k \neq a_j \) randomly chosen in \( \mathcal{A} \).

In [13], we introduced and study the properties of LR-Mijjn co-mutation operator

**Definition 3.1** Formally, the LR-Mijjn operator is defined as follows:

(i) If \( p \neq \text{right} \) and \( p \neq l - \text{left} - 1 \),

\[
\text{LR-Mijjn}(\sigma) =
\]
\[
\begin{align*}
\sigma_{\text{left}} &= \sigma(l-1)...\sigma(p+1), \quad \text{for} \quad p \leq l - 1, \\
\sigma(p-1) &= \sigma(0) \quad \text{for} \quad p > l, \\
\sigma_{\text{right}} &= \sigma(l-1)...\sigma(p+m), \quad \text{for} \quad p \geq l, \\
\end{align*}
\]

As an example, let us consider the binary case, the string \( \sigma = 11110000 \) and the randomly chosen application point \( p = 2 \). In this case, \( \sigma_2 = 0, \) so we have to find the longest sequence of 0 within string \( \sigma, \) starting from position \( p. \) This sequence goes to the right, and because we have reached the end of the string, and no occurrence of 1 has been met, the new string obtained after the application of LR-Mijn is 11110111.

5 Main results. The model for constructing complex SVM kernels

Our goal is to build and analyze a multiple kernel starting from the simple polynomial kernels:

\[
K_d = (x_1 \cdot x_2 + r)^d, \quad \text{where} \quad r, d \in \mathbb{Z}_+, \quad (5.1)
\]

and having two parameters, the degree \( d \) and \( r. \) We use the idea of the model proposed in [3]. In a first level, we will build and evaluate multiple kernels obtained from (5.1) using a genetic algorithm and the set of operations \( \{+, \cdot, \exp\}, i = 1,3. \)

5.1 Representation of the multiple kernel

We consider the particular case in which parameter \( r \) is the same for all simple kernels used in our composition procedure. In the figure 1 is represented the kernel

\[
(K_{d_1,r} \cdot K_{d_2,r} \cdot K_{d_3,r}) \cdot (K_{d_1,r} \cdot K_{d_2,r} \cdot K_{d_3,r}): 
\]
If a node contains the operation $\exp$ only one of its descendants is considered (the “left” kernel).

The chromosome which codes the multiple kernel described above has the following structure:

```
| op1 | op2 | op3 | d1 | d2 | d3 | d4 | r |
```

Each operation $op_i$ is represented using two genes, for a degree $d_j$ are allocated four genes, and the variable $r$ is represented using twelve genes. Thus, our chromosome is composed from 34 genes.

The evaluation of the chromosome is made using the SVM algorithm for a particular set of data. To do this we divide the data into two subsets: the training subset, used for problem modeling and test subset used for evaluation. The training subset is also random divided into a subset for learning and a subset for validation. The SVM algorithm uses the data from the learning subset for training and the subset from the validation set for computing the classification accuracy which is used as fitness function for the genetic algorithm.

### 5.2 SVM algorithm

For the implementation/testing/validation of our method was used the “leukemia” data set from the page LIBSVM data sets page ([2]).

In order to replace the default polynomial kernel from libsvm, we extend the `svm_parameter` class with the following attributes:

```java
// parameters for multiple polynomial kernels
public long op[]; // operations
public long d[]; // degrees
public long r[]; // parameters
```

The class `svm_predict` was extended with the following method:

```java
public double predict(long op[], long d[], long r){
    try {
        BufferedReader input = new BufferedReader( new FileReader(test_file));
        DataOutputStream output = new DataOutputStream( new FileOutputStream("predict.out"));
        svm_model model = svm.svm_load_model(model_file);
        model.param.op = op;
        model.param.d = d;
        model.param.r = new long[4];
        for (int i=0; i<4; i++)
            model.param.r[i]=r;
        model.param.kernel_type = svm_parameter.POLY;
        result=predict(input,output,model,0);
        input.close();
        output.close();
        return result;
    } catch(FileNotFoundException e) {
        exit_with_help();
    }
    ...... return 0;
}
```

The `Kernel` class was modified to accomplish the kernel substitution. In the `k_function` method, our simple kernels are computed as follows:

```java
K = new double [4];
for (k=0; k<4; k++)
    K[k]=powi(dot(x,y)+param.r[k],param.d[k]);
```

Then, the kernels are combined using operation given in array `param.op[]`.

In the genetic algorithm, the operations, the degrees of simple kernels and the parameter $r$ are obtained from a chromosome, which is then evaluated using the result of the `predict` method presented above.

Thus, the SVM algorithm uses the learning subset of data for training the SVM model and the testing subset for computing the classification accuracy that is the fitness function for the genetic algorithm.

After the end of the genetic algorithm, the best chromosome gives the multiple kernel which can be evaluated on the test subset of data. The way of construction this multiple kernel assures that it is a veritable kernel, that is, it satisfies Mercer’s conditions.

### 5.3 Experimental results

Using the standard `libsvm` package, for the “leukemia” data set is obtained the following classification accuracy:

```java
java -classpath libsvm.jar svm_predict leu.t leu.model leu.output
Accuracy = 67.64705882352942% (23/34) (classification)
```

Multiple kernels obtained using genetic approach are
improving the classification accuracy up to 91.18%. In the figure 2 are presented results from three runs of our genetic algorithm based on LR-Mijn operator. For each execution, dimension of population was 35 and the number of generations was 30.

One “optimal” multiple kernel obtained is depicted in figure 3, where r = 2111.

![Fig. 3 Representation of an optimal multiple kernel](image)

If we impose the additional condition that $d_i > 0$, $i=1,\ldots,4$, we obtain the same accuracy after 30 generation, using only $+$ and $*$ operations.

![Fig. 4 Classification accuracy, for $d_i > 0$](image)

6. Conclusions and further directions of study

In this paper we presented a model for a SVM multiple kernels, which contains only polynomials simple kernels and is based on a genetic algorithm using a new co-mutation algorithm. Experimental results prove that the utilization of genetic algorithm based on LR-Mijn co-mutation operator, has a better convergence and improves the accuracy toward the classical genetic algorithm used in [7]. We use the idea introduced in [3], but our model is different, the structure of chromosomes which code the multiple kernels is different and the macro level that is the genetic algorithm used is different. The experimental results prove that the operations $+$ and $*$ have a better behavior with polynomial kernels than the exponential operation.

As a further direction of study we want to compare the results obtained using not only polynomial kernels and characterize the behavior of the composition operation $+,*$ and $\exp$ related to the kernel type.

References:


