Credit Data Fraud Detection using 
Kernel Methods with Support Vector 
Machine
Maryamsadat Hejazi\textsuperscript{a}, Yashwant Prasad Singh\textsuperscript{b} 
Faculty of Information Technology, Multimedia University, 
Cyberjaya, Malaysia
\textsuperscript{a}\texttt{hejazi\_m2002@yahoo.com}, \textsuperscript{b}\texttt{y.p.singh@mmu.edu.my}

ABSTRACT
This paper presents binary Support Vector Machine classifiers for detection of fraudulent credit card transactions. Support Vector Machine (SVM) is a well-known method in statistical learning theory based on the principle of structural risk minimization for classification and regression problems. The objective of this paper is about evaluation of accuracy and performance of different kernel methods using Sequential Minimal Optimization (SMO), \textit{C}-Support Vector Classification (\textit{C}-SVC) and \textit{\nu} -Support Vector Classification (\textit{\nu}-SVC) for fraudulent credit card transaction detection. A Comparison of computation time and classifiers’ accuracy for SMO classifier with \textit{C}-SVC in LIBSVM is provided. The simulation results show that accuracy of SMO classifier with different kernel functions is higher than \textit{C}-SVC on credit data dataset obtained from UCI machine learning database.

Keywords: Support Vector Machine; Kernel methods; \textit{C}-Support Vector Classification; \textit{\nu}-Support Vector Classification; LIBSVM; SMO.

1. Introduction
Fraud detection is known as illegal activities in commercial and industrial organization such as banks, insurance companies, and cell phone companies and so on (Chandola et al., 2009). Fraud is a serious ethical problem in credit card companies in banking systems. The primary purposes of credit card fraud detection are to detect credit card fraud transactions and reduce the losses due to such illegal activities. Classification is one of the techniques that can help credit card anomaly detection system which can learn a model from a set of training data and classify a test data well into one of the classes normal or anomalous.

Support Vector Machine (SVM) is a very popular classification technique which is related to statistical learning theory in machine learning and can be applied to classification and regression problems (Vapnik et al., 1997). Primary concept of SVM has been proposed by Vladimir Vapnik (Vapnik, 1995) as binary classification techniques to reduce generalization error and to enhance predictive accuracy for detection and classification (Falah, 2011). SVM is machine learning
method formulated as an optimal separating hyperplane with maximum margin between specified classes in data space or feature space. Kernel methods in Support Vector Machines has been applied for nonlinear SVM methods to map data points to a feature space (higher dimensional space) and learn optimal linear classifier based on kernel function in transformed space (Kumar et al., 2006). In recent years, SVM learning has found a wide range of real-world applications, including text categorization, character recognition, bioinformatics, spam categorization, bankruptcy, handwritten digit recognition, object recognition, speaker identification, face detection in images and so forth (Vapnik et al., 1992; Burges, 1998).

In this paper we propose the use of support vector machine (SVM) learning to detect fraudulent credit card transactions with different kernel functions. SVM classifiers with different kernels are investigated for credit card dataset to determine the best SVM classifier algorithm. We attempt to consider binary classification accuracy between SMO and LIBSVM classifiers with different kernel methods to determine which classifier has higher accuracy than others with the same parameters settings on the dataset from UCI repository (Frank and Asuncion, 2010).

The rest of the paper is organized as follows. Section 2 provides a brief review on SVMs classification and gives details of kernel methods and kernel types. C-SVC and γ-SVC are also described in the section. Section 3 describes SVM tools as SMO and LIBSVM. The experimental results are presented in section 4 along with a comparison of computation time and accuracy for C-SVC, γ-SVC and SVM. A conclusion is given in section 5.

2. Support Vector Machine

The SVM classification is based on selection of optimal separating hyperplane as decision boundary among many hyperplanes possible in input space or feature space to separate set of data point into negative and positive classes. The decision boundary is constructed with maximal distance (maximum margin) from the linear separator to the closest positive and negative samples (Kumar et al., 2006). Each decision boundary is related to a pair of hyperplanes. The distance between the hyperplanes is defined as margin of classifier. Learning a decision boundary with maximal margin affects in minimizing the generalization errors and performs well on the testing data (Kumar et al., 2006).

One such classifier is linear SVM that can classify the training data samples using learning a decision boundary in input space. The method used for solution of linear SVM in separable case is known as hard margin classifier and method applied in linear SVM for nonseparable case is known as soft margin classifier.

Nonlinear SVM employs kernel methods for mapping data from input data space into a higher dimensional feature space to design nonlinear decision boundary in input space. The above concept is shown in Fig. 1. Mathematical concepts for binary SVM for linear and nonlinear case are given in the following sections.
2.1 Linear SVMs Classifier: Separable Case

Each training sample (a \( n \)-dimensional vector) is shown by a tuple \((x_i, y_i) (i = 1, 2, \ldots, N)\) that \(x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T\) corresponding to the set of attributes for the \(i^{th}\) sample in a data set and \(y_i \in \{-1, 1\}\) is the class label related to \(x_i\). A linear decision boundary classifier can be written as Eq. 1:

\[
w \cdot x + b = 0
\]  

(1)

The vector \(w \in \mathbb{R}^n\) is as the weight vector and \(b\) is a bias term or offset of the hyperplane from the origin. \(x\) are points located on decision boundary that are known as support vector for binary class problem (Kumar et al., 2006; Campbell and Ying, 2011; Abe, 2005). Decision boundary and margin as \(d\) are shown in Fig. 2 (Kumar et al., 2006).

The parameters \(w\) and \(b\) of decision boundary can be estimated in the training phase with considering Eqs. 2 and 3:
Both inequalities constraints (2) and (3) can be simplified to Eq. 4:

\[ y_i (w \cdot x_i + b) \geq 1, \forall i \in \mathbb{N} \]  

(4)

The maximal margin of classifier is formulated as maximizing \( \frac{2}{||w||} \).

Optimization problems are selection of the optimal solution from feasible solutions that can minimize or maximize the value of a function as given below (Hamel, 2009):

\[ x \in \mathbb{R}^n : \max f(x) = \min \frac{1}{f(x)} \]  

(5)

Eq. 6 is an objective function that describes an equivalent maximizing margin function from Eq. 5 as:

\[ f(w) = \min \frac{||w||^2}{2} \]  

subject to \[ y_i (w \cdot x_i + b) \geq 1, \forall i \in \mathbb{N} \]  

(6)

Quadratic objective function (\( ||w||^2 \)) and linear constrains \( w \) and \( b \) in Eq. 6 is known as a convex optimization problem which can find the minimum of objective function. The Lagrange multiplier method is applied to solve a convex quadratic programming (QP) problem with constraints (equality and inequality) (Hamel, 2009).

The new objective function with the Lagrange multipliers (\( \lambda_i \geq 0 \)) is known as the Lagrangian for the optimization problem (Kumar et al., 2006) and is given below:

\[ L(w, b, \lambda) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \lambda_i (y_i(w \cdot x_i + b) - 1) \]  

(7)

The optimal solution for Eq. 6 problem is specified by the saddle point of the Lagrangian \( L(w, b, \lambda) \) that it must minimize with respect to \( w \) and \( b \), and is maximized with respect to \( \lambda \). Derivative of \( L(w, b, \lambda) \) with respect to \( w \) and \( b \) and set them to zero are computed to minimize the Lagrangian:

\[ \frac{\partial L(w, b, \lambda)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \lambda_i y_i x_i \]  

(8)
\[
\frac{\partial L(w, b, \lambda)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0
\]  

(9)

Transformation inequality to equality constraints lead to the following constraints on the Lagrange multipliers that are known as Karush-Kuhn-Tucker (KKT):

\[
\lambda_i (y_i (w \cdot x_i + b) - 1) = 0, \forall i \in \mathbb{N}, \lambda_i \geq 0
\]  

(10)

Training instances that reside on the hyperplanes are known as support vectors and satisfy \(y_i (w \cdot x_i + b) = 1\) and \(\lambda_i > 0\), but training samples which are not located on hyperplanes have \(\lambda_i = 0\) (Kumar et al., 2006).

The optimization problem can be reformulated as the dual problem as written below:

\[
L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i \cdot x_j
\]  

subject to \(\sum_{i=1}^{N} \lambda_i y_i = 0, \lambda_i \geq 0, \forall i \in \mathbb{N}\)  

(11)

(12)

The decision boundary is defined as following:

\[
f(x) = \left( \sum_{i=1}^{N} \lambda_i y_i x_i \cdot x \right) + b = 0
\]  

(13)

\(b\) is defined by the KKT condition and depends on the support vector \(x_i\), so it might not be unique and average of \(b\) is selected for the decision boundary equation (Abe, 2005):

\[
b = \frac{1}{|S|} \sum_{i \in S} (y_i - w \cdot x_i)
\]  

(14)

where \(S\) is the set of support vectors.

The unseen instance \(x\) can be classified as follows:

\[
f(x) = \text{sign} (w \cdot x + b) = \text{sign} \left( \sum_{i=1}^{N} \lambda_i y_i x_i \cdot x + b \right)
\]  

(15)

If \(f(x) = 1\) then the test example is specified as instance of positive class, otherwise, the instance is classified as negative class (Kumar et al., 2006).
2.2 Linear SVMs Classifier: Nonseparable

The training data in hard margin support vector machine is separated as linear classifier without misclassification. This section examines how SVM constructs a linear decision boundary for nonseparable data (outliers or noisy samples- are inside or on the wrong side of the margin (Falah, 2011) using a method known as soft margin. The method can be to classify correctly training data and considers the trade-off between size of margin and the number of training errors.

The soft margin introduces slack variables (\( \forall \xi_i \geq 0 \)) that provide estimation of training errors of classifier for nonseparable data into inequality constraints of optimization. If \( \xi_i = 0 \) the training data \( x_i \) is on margin or correct side of margin. If \( 0 < \xi_i < 1 \) the training data are inside margin and do not have maximal margin, but they can be still classified correctly. Otherwise, if \( \xi_i \geq 1 \) the optimal hyperplane classifies the training data incorrectly. These concepts are shown in Fig. 3 (Abe, 2005).

![Fig. 3. Slack variables for nonseparable data (Abe, 2005)](image)

Objective function is modified for soft margin as follows:

\[
f(x) = \min \left[ \frac{||w||^2}{2} + C \sum_{i=1}^{N} \xi_i^k \right] \tag{16}
\]

subject to \( y_i (w \cdot x + b) \geq 1 - \xi_i, \forall i \in \mathbb{N} \)

\( C \) is the positive margin parameter that specifies the trade-off between size of the margin and minimum number of training error (regularization term). The parameter \( k \) specifies type of classifier. \( k = 1 \) is assumed to simplify the problem. The dual Lagrangian for linear SVM for nonseparable case is as follows:

\[
L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j x_i \cdot x_j \tag{17}
\]
subject to \[ \sum_{i=1}^{N} \lambda_i y_i = 0 , 0 \leq \lambda_i \leq C , \forall i \in \mathbb{N} \]

Training data that are located on outside margin have conditions of \( \lambda_i = 0 \) and \( \xi_i = 0 \). The training data that reside along hyperplanes have terms of \( 0 < \lambda_i < C \) and \( \xi_i = 0 \). If training instances locate inside of the margin, they can be still classified correctly while they have conditions of \( \lambda_i = C \) and \( 0 < \xi_i < 1 \). The training points are misclassified while inside of margin and have conditions of \( \lambda_i = C \) and \( \xi_i \geq 1 \).

The bias term as:

\[
b = \frac{1}{|S|} \sum_{i \in S} (y_i - w \cdot x_i)
\]  

(18)

Decision boundary function is defined in the following equation:

\[
f(x) = \text{sign} (w \cdot x + b) = \text{sign} (\sum_{i=1}^{N} \lambda_i y_i x_i \cdot x + b)
\]  

(19)

2.3 Nonlinear SVMs Classifier (Kernel SVMs)

Nonlinear support vector machine applies a methodology to maximize the generalization ability for training data sets that have nonlinear decision boundary. The methodology is mapping of each vector \( x \) in input space with \( n \)-dimension into new space \( \Phi(x) \) with the \( l \)-dimensional feature space and use a linear classifier to separate the instances in the feature space as follows (Kumar et al., 2006; Abe, 2005; Schölkopf and Smola, 2002):

\[
f(x) = w \cdot \Phi(x) + b
\]  

(20)

The following Fig. 4 illustrates this mapping:
If the symmetric function \( k(x_i, x_j) \) satisfies the following Hilbert-Schmidt theorem for any input data set \( \{x_i\}_{i=1}^{N} \in X \) (input space) (Abe, 2005),

\[
\sum_{i,j=1}^{N} h_i h_j k(x_i, x_j) \geq 0, \forall h_i \in \mathbb{R}, N \in \mathbb{N}
\]  

(21)

then there exists a mapping function, \( \Phi(x) \), that maps any input data \( x_i \) in input space into feature sapace and \( \Phi(x) \) satisfies following condition (Abe, 2005):

\[
k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)
\]  

(22)

If Eq. 22 is satisfied, then

\[
\sum_{i,j=1}^{N} h_i h_j k(x_i, x_j) = \left( \sum_{i=1}^{N} h_i \Phi(x_i) \right) \cdot \left( \sum_{j=1}^{N} h_j \Phi(x_j) \right) \geq 0
\]  

(23)

The condition in Eqs. 21 or 23 is known as Mercer’s condition, and the function in this condition is called as the positive define kernel or Mercer kernel (Abe, 2005).

Kernel method is known as kernel trick that uses \( k(x_i, x_j) \) in classification and training instead of evaluation of the transformation \( \Phi(x_i) \). The dual Lagrangian for constrained optimization problem in feature space is modified as:

\[
L(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j k(x_i, x_j)
\]  

(24)
subject to \( \sum_{i=1}^{N} \lambda_i y_i = 0, \ 0 \leq \lambda_i \leq C, \forall i \in \mathbb{N} \)

Then the classifier or decision boundary is as below:

\[
f(x) = \sum_{i=1}^{N} \lambda_i y_i k(x, x_i) + b
\]  

(25)

One of advantages of support vector machine is improvement of generalization performance by appropriate selection of kernels, so selection of kernels to specific application is more significant. The common kernels that are used in SVM are given below (Abe, 2005; Vaerenbergh, 2009):

- Linear Kernel: \( k(x_i, x_j) = x_i \cdot x_j \)
- Polynomial Kernels: \( k(x_i, x_j) = (\gamma (x_i \cdot x_j) + r)^d, \ r \geq 0, \gamma > 0 \)
- Radial Basis Function Kernels (RBF): Vapnik, Boser and Guyon define Gaussian radial basis function kernels as following formulation:
  \[ k(x_i, x_j) = \exp(-||x_i - x_j||^2 / 2\sigma^2) \] where \( \sigma > 0 \)
  RBF kernel is described via Gaussian kernel where \( \sigma = 1 \) as follows:
  \[ k(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2), \gamma > 0 \]
- Hyperbolic Tangent Kernel: \( k(x_i, x_j) = \tanh(\alpha (x_i \cdot x_j) + r), r \geq 0 \)

2.4 C-Support Vector Classification (C-SVC)

C-SVC is a binary classification and the primal optimization problem in this formulation has been solved by Boser et al. in 1992 and Vapnik and Cortes in 1995 (Chang and Lin, 2011). The constrained optimization problem is defined for training vectors for binary classes \( x_i \in \mathbb{R}^n, i = 1, 2, ..., N \) and \( y_i \in \{-1, 1\} \):

\[
\min_{w, b, \xi} \left[ \frac{1}{2} ||w||^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i \right]
\]

subject to \( y_i (w \cdot \Phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \in \mathbb{N} \)

If \( \xi_i = 0 \) then the data points have no margin errors, otherwise all slack variables are margin errors. Parameter \( C \) is positive regularization parameter that is trade off maximum size of margin and minimum number of training error. The dual problem can be defined in general as follows:
\[
\min_{\lambda} \left[ \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \right] 
\]

subject to \( 0 \leq \lambda_i \leq \frac{C}{N}, \sum_{i=1}^{N} \lambda_i y_i = 0, C > 0, \forall i \in \mathbb{N} \)

The decision function is as follows:

\[
f(x) = \text{sign} \left( \sum_{i=1}^{N} y_i \lambda_i k(x,x_i) + b \right)
\]

The threshold \( b \) can be achieved for instances by averaging from the following equation:

\[
b = f(x) - \sum_{i=1}^{N} y_i \lambda_i k(x,x_i)
\]

### 2.5 \( \nu \)-Support Vector Classification (\( \nu \)-SVC)

\( \nu \)-SVC is binary classification and has been proposed by Schölkopf et al. in 2000 (Chang and Lin, 2011). \( C \) in \( C \)-SVC is a rather unintuitive and constant parameter and there is not any priori way to select it, so a parameter \( \nu \in (0,1] \) replaces parameter \( C \) to control number of margin errors and support vectors. Parameter \( \nu \) is an upper bound on the fraction of margin errors, and lower bound on the fraction of SVs (Schölkopf and Smola, 2002; Chang and Lin, 2011). Objective function is formulated with term \( \rho \) as margin parameter in the following:

\[
\min_{w, b, \xi, \rho} \left[ \frac{1}{2} ||w||^2 - \nu \rho + \frac{1}{N} \sum_{i=1}^{N} \xi_i \right]
\]

subject to \( y_i (w \cdot \Phi(x_i) + b) \geq \rho - \xi_i, \forall i \in \mathbb{N}; \xi_i \geq 0, \rho \geq 0 \)

If \( \xi_i = 0 \) then the margin between two classes is \( \frac{2\rho}{||w||} \), but if variable \( \xi_i > 0 \) then the margin error denotes points that are training errors with condition \( (\xi_i > \rho) \) or reside along the margin \( (\xi_i \in (0,\rho]) \) (Schölkopf and Smola, 2002). The decision boundary can be expressed as follows:

\[
f(x) = \text{sign} \left( \sum_{i=1}^{N} \lambda_i y_i k(x,x_i) + b \right)
\]
The bias term $b$ and margin parameter $\rho$ are computed as given below by two sets $S_{\pm}$ of identical size ($s > 0$) which include support vectors $x_i$ with respective to $0 < \lambda_i < 1$ and $y_i = \pm 1$:

\[
b = -\frac{1}{2s} \sum_{x \in S_+ \cup S_-} \sum_{i=1}^{N} \lambda_i y_i k(x, x_i)
\]

\[
\rho = \frac{1}{2s} \left( \sum_{x \in S_+} \sum_{i=1}^{N} \lambda_i y_i k(x, x_i) - \sum_{x \in S_-} \sum_{i=1}^{N} \lambda_i y_i k(x, x_i) \right) \tag{32}
\]

It may be noted that only $b$ is required for decision boundary.


There exist several algorithms and software packages to solve quadric optimization programs including SMO and LIBSVM. The present paper considers SMO and LIBSVM for SVM simulation on credit card data set.

3.1 Sequential Minimal Optimization (SMO)

This algorithm has been proposed by Platt in 1999 for training support vector machines. SMO is a decomposition method which breaks large quadratic optimization problem into a series of smallest possible quadric problem (Müller, 2001; Platt, 2000). These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop such as previous SVM learning algorithms (Platt, 2000). Because most of SMO time spends for evaluating the decision function, rather than performing QP (Platt, 2000). The original work was targeted at an SVM classification; there are now also approaches which implement variants of SMO for SVM regression and single-class SVMs (Müller, 2001).

3.2 Library of Support Vector Machine (LIBSVM)

Chang and Lin (Chang and Lin, 2011) have proposed LIBSVM algorithm in Java and C++ language based on simplification of both SMO and SVMLight by Joachims and a simplification of the modification of SMO by Keerthi et al (Chang and Lin, site link). Libsvm is a library of support vector machines that has been widely developed for support vector classification, such as $C$-SVC and $\nu$-SVC (binary case), Support Vector Regression (SVR) as $\varepsilon$-SVR and $\nu$-SVR and distribution estimation (one-class SVM) (Chang and Lin, 2011). Libsvm provides a simple interface with Java, Python, MATLAB, Perl, LabVIEW, Splus, Ruby and R.
4. Experimental Results

This section provides experimental results of a number of Support Vector Machines with LIBSVM and SMO in Weka software (Witten et al., 2011) and their computation time and accuracy for binary classification on German credit data dataset from UCI repository (Frank and Asuncion, 2010). German credit data dataset has two classes with 20 attributes. The number of instances of given data is 1000 which is divided into 900 samples as training and 100 as test data. Experimental results are provided for SVM classifiers with SMO and LIBSVM along with their tuning parameters. A comparative performance of different kernels is obtained to determine the best and efficient kernel.

4.1 Performance of different kernels in SMO

The selection of a kernel to map data from original input space to feature space affects the performance of SVM. There are no definite rules for this selection except satisfactory performance by simulation study (Falah, 2011).

Table 1 presents simulation results of the number of support vectors, accuracy and performance of SMO with three kernels for different values of C parameter (cost for each slack). The polynomial kernel is considered with degrees 3 and $\gamma = 1$, and the value of $\gamma$ parameter for RBF kernel is set to 0.01 and 0.8. The best result from Table 1 for all kernels is for $C$ (complexity parameter) with value of 60. In spite of better accuracy for polynomial kernels, the computation time and the number of support vectors for classifying the samples in RBF is better than polynomial kernels shown in Table 1.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>C</th>
<th>SV</th>
<th>Accuracy [%]</th>
<th>Computation Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Train</td>
<td>Test</td>
</tr>
<tr>
<td>Linear</td>
<td>0.1</td>
<td>0</td>
<td>77.66</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>78.44</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0</td>
<td>78.88</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0</td>
<td>79.11</td>
<td>72</td>
</tr>
<tr>
<td>Polynomial</td>
<td>0.1</td>
<td>513</td>
<td>100</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>502</td>
<td>100</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>502</td>
<td>100</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>502</td>
<td>100</td>
<td>73</td>
</tr>
<tr>
<td>RBF ($\gamma = 0.01$)</td>
<td>0.1</td>
<td>605</td>
<td>70.22</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>629</td>
<td>70.22</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>512</td>
<td>80.44</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>492</td>
<td>87.77</td>
<td>74</td>
</tr>
<tr>
<td>RBF ($\gamma = 0.8$)</td>
<td>0.1</td>
<td>897</td>
<td>70.22</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>898</td>
<td>99.66</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>898</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>898</td>
<td>100</td>
<td>70</td>
</tr>
</tbody>
</table>
4.2 Performance of different kernels in LIBSVM

Table 2 presents the results of different kernel in LIBSVM classifiers for different set of parameters $C$ and $\gamma$. It can be seen from Table 2 that better accuracy is obtained for polynomial kernels, but their computation time is more than RFB kernels. Table 3 shows that accuracy for RBF kernels is more than other kernels, but computation time is less than polynomial kernel and more than linear kernel.

Table 2. A performance Comparison of Different Kernels in $C$-SVC

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$C$</th>
<th>$\gamma$</th>
<th>Accuracy [%]</th>
<th>Computation Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Train</td>
<td>Test</td>
</tr>
<tr>
<td>Linear</td>
<td>0.1</td>
<td>1</td>
<td>76.44</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>77.22</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1</td>
<td>77.77</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>1</td>
<td>77.77</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
<td>99.77</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.01</td>
<td>70.22</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.01</td>
<td>74.88</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.01</td>
<td>83.88</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.8</td>
<td>70.22</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.8</td>
<td>99.66</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.8</td>
<td>83.33</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.8</td>
<td>100</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 3. A Performance Comparison of Different Kernels in $\nu$-SVC

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$\nu$</th>
<th>$\gamma$</th>
<th>Accuracy [%]</th>
<th>Computation Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Train</td>
<td>Test</td>
</tr>
<tr>
<td>Linear</td>
<td>0.01</td>
<td>1</td>
<td>51.77</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1</td>
<td>64.77</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>1</td>
<td>75.66</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>1</td>
<td>100</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>1</td>
<td>96.55</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>1</td>
<td>83.33</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.01</td>
<td>70.33</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.01</td>
<td>95</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.01</td>
<td>80.55</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.8</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.8</td>
<td>99.88</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.8</td>
<td>99.66</td>
<td>70</td>
</tr>
</tbody>
</table>
4.3 Comparison between SMO and C-SVC in LIBSVM with different kernels

Considering Tables 1 and 2, linear kernel in SMO has better accuracy than linear kernels in LIBSVM, but linear kernel in LIBSVM classifier takes computation time less than linear kernel in SMO.

Accuracy and computation time for polynomial kernel in SMO classifier are better than the polynomial kernel in LIBSVM as shown in Tables 1 and 2. Table 1 also shows that RBF kernels in SMO classifier have higher accuracy and lower computation time than RBF in C-SVC in LIBSVM.

5. Conclusion

This paper has presented details about binary SVM classification and has provided experimental results on credit card dataset classification. Comparative results for different kernels have been provided for credit data dataset with SMO and LIBSVM classifiers with different parameter settings. The experimental results have shown that accuracy of SMO classifier with different kernels is better than C-SVC as binary SVM of LIBSVM classifier with the same kernels, but computation time in LIBSVM classifier is always less than SMO classifier for German credit card dataset.

References


