Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science

Master`s Thesis

COMPARATIVE ANALYSIS OF BINARY CLASSIFICATION ALGORITHMS

Zulfiia Galimzianova

Supervisor: Ing. Miroslav Bureš, Ph.D.

Study Program: Open Informatics

Field of Study: Software Engineering

May 24, 2019

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Galimzianova Zulfiia**	Personal ID number:	**484743**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science and Engineering**		
Study program:	**Open Informatics**		
Specialisation:	**Software Engineering**		

II. Master’s thesis details

Master’s thesis title in English:

Comparative analysis of binary classification algorithms

Master’s thesis title in Czech:

Comparative analysis of binary classification algorithms

Guidelines:

Consider the classification problem into two non-intersecting classes. Let there be some set of objects U, each of which belongs to one of the two represented classes C0 and C1, and is described by an n-dimensional vector of characteristics(x1, …, xn). Also, suppose there is a function μ(u) that calculates these vectors for any object u U.
The goal is to build classifiers and to analyze the performance of these classification algorithms.
The tested methods include types of support vector machines (SVM) such as standard SVM (minimizing the sum of squared errors) and SVM which uses uniform objective function(minimizing the maximum of squared errors).
The experiments on several applications will be conducted to demonstrate the strengths and weaknesses of the classifiers and quantitatively evaluate their performance.

Bibliography / sources:

Konnov, Igor, Selective Bi-Coordinate Variations for Resource Allocation Type Problems (November 5, 2014). Available at SSRN: https://ssrn.com/abstract=2519662 or http://dx.doi.org/10.2139/ssrn.2519662
I.V. Konnov (2015) Sequential threshold control in descent splitting methods for decomposable optimization problems, Optimization Methods and Software, 30:6, 1238-1254, DOI: 10.1080/10556788.2015.1030015
C. Cortes и V. Vapnik, «Support-Vector Networks», Mach. Learn., т. 20, вып. 3, сс. 273–297, сен. 1995.
В. Вапник и А. Червоненкис, Теория распознавания образов. Главная редакция физико-математической литературы издательства «Наука», 1974.
V. N. Vapnik, Statistical Learning Theory, 1 edition. New York: Wiley-Interscience, 1998.

Name and workplace of master’s thesis supervisor:

doc. Ing. Miroslav Bureš, Ph.D., Software Testing Intelligent Lab, FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **05.04.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until: **19.02.2021**

doc. Ing. Miroslav Bureš, Ph.D.	Head of department’s signature	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

Throughout the writing of this thesis I have received a great deal of support and assistance. I would first like to thank my supervisor, Prof. Dr. Miroslav Bureš, for his support and help during my education at Czech Technical University in Prague. I would like to express my sincere gratitude to Prof. Dr. Igor Konnov, whose expertise was invaluable in the formulating of the research topic and development of methodology in particular. I would also like to appreciate Prof. Dr. Ekaterina Turilova, Prof. Dr. Arslan Enikeev, and Prof. Dr. Galim Vakhitov for the opportunity to participate in the Double Degree program (CVUT, KFU) and the CTU mentors and Prof. Dr. Karel Frajták as part of it for the great experience of studying in CVUT.

Finally, thanks a lot to my family and friends for their unconditional support and faith in me.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

Prague, May, 2019

Abstract

GALIMZIANOVA, Zulfiia: Comparative analysis of binary classification algorithms. [Master Thesis] – Czech Technical University in Prague. Faculty of Electrical Engineering, Department of Computer Science. Supervisor: Ing. Miroslav Bureš, Ph.D.

In this work a new modification of the SVM algorithm that employs a new regularization term is proposed. We propose to formulate the constraints of the classification model in a uniform manner and demonstrate an algorithm to find the solution of the new optimization problem in an efficient manner. The performance was compared to the baseline SVM algorithm on a public dataset. The experiments were conducted following the Monte-Carlo cross-validation scheme and the performance metrics were chosen as accuracy, recall, precision, and f1-score. The computational performance analysis was done in terms of the number of iterations that was required by the algorithms to converge. The resulting metrics vectors were compared using Wilcoxon's signed rank test to identify the statistical significance of the findings. Our results on a public dataset demonstrated statistically significant improvements both in terms of accuracy and computational performance over the baseline SVM algorithm. As such, the method has the potential to be implemented in real-world applications in an effective and cost-efficient manner.

Keywords: optimization problem, binary classification, machine learning, support vector machine, SVM, comparative analysis.

Contents

List of figures

List of tables

Abbreviations

AI: artificial intelligence

GPU: graphics processing unit

CV: computer vision

NLU: natural language understanding

ML: machine learning

DL: deep learning

SVM: support vector machine

MCCV: Monte-Carlo cross-validation

QP: quadratic programming

SMO: sequential minimal optimization

1 Introduction

The AI methods are revolutionizing many fields of industry, medicine, business and academia through a variety of solutions becoming available to solve complex tasks in an effective and efficient way thanks to the recent methodological (e.g., deep learning) and technical (e.g., GPU computing) advancements [1]. A considerable boost to the popularization of the AI tools was an open visual recognition challenge ImageNet, where a deep convolutional neural network resulted in an unprecedented performance improvement of the traditional machine learning methods [2].

Regardless of the application, machine learning methods are the core of the AI tools (see Figure 1.1). In particular, many practical tasks, such as object detection in surveillance video or treatment response prediction based on genetic testing, are solved through formulating them as a classification problem. In such formulation, for each object of interest a vector of features is formed and the categorical labels are assigned to each of them. Although the classification problem is in general multiclass problem, it can and usually is re-formulated in terms of binary classification, using techniques such as one-against-many.

A significant contribution to the theory of machine learning, specifically, to object classification, was done by V. Vapnik, who proposed to consider the learning as optimization problem [3]. This, in turn, facilitates the use of optimization theory in AI systems development. At its current state, machine learning is an interdisciplinary field of research that combines disciplines such as statistics, information theory, theory of algorithms, probability theory and functional analysis.

In 1963, Vapnik and Chervonenkis proposed a method of separating hyperplanes, which has become a basis for the development of SVMs [4]. The method has become one of the most popular ML algorithms for supervised classification. SVMs have been applied to multitude of applications, including image analysis, digit recognition and bioinformatics.

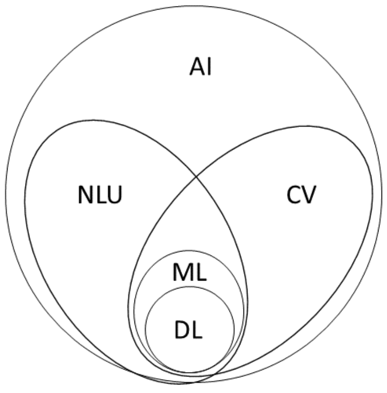


Figure 1.1. A schematic representation of an AI-based tool. Adapted from NVIDIA’s CEO J. Huang’s keynote speech at GTC 2019 Technology Conference.

2 Methods

2.1 Binary classification problem

Let us formulate the task of binary classification. Given a set of objects U, each of which belongs to one of the two given classes C_{-1} or C_1, we will describe them as a feature vector $x = (x_1, \dots, x_m)$.

Let us have n objects $u_1, \dots, u_n \in U$. Based on these, the training dataset T, containing n feature vectors $\{x^i\}$, where $i = 1, \dots, n,$ is formed. In the training dataset, for each input vector x^i its label is stored as the class index $y_i \in \{1, -1\}$, which contains object u_i. Thus, the training dataset is obtained as $T = ((x^1, y_1), \dots, (x^n, y_n))$.

The binary classification task is to find which of the given classes, i.e., C_{-1} or C_1 object u belongs, given the m-dimensional input features, or to find the mapping $f(x, w) \rightarrow \{-1, 1\}$, where x is the feature vector and w is the model parameters vector.

2.2 SVM Formulations

SVM is a machine learning method for supervised learning. It could be used for the classification as well as regression tasks.

In its linear form, the algorithm builds a hyperplane in the space of the features, such that to separate the sets of points from the two given classes. Its objective function is

$$f(x) = w^T x + \beta, \tag{2.1}$$

where x is the input vector of features, and w is a vector orthogonal to the separating hyperplane defined as $f(x) = 0$, and the hyperplanes that contain the points in the feature space closest to it with $f(x) = \pm 1$. The distance to these points from the hyperplane is defined as

$$l = \frac{1}{||w||}. \tag{2.2}$$

Since we want to apply this to a binary classification problem, we will ultimately predict y = 1 if f(x) ≥ 0 and y = −1 if f(x) < 0.

Multiple separating hyperplanes can exist (see Figure 2.1).

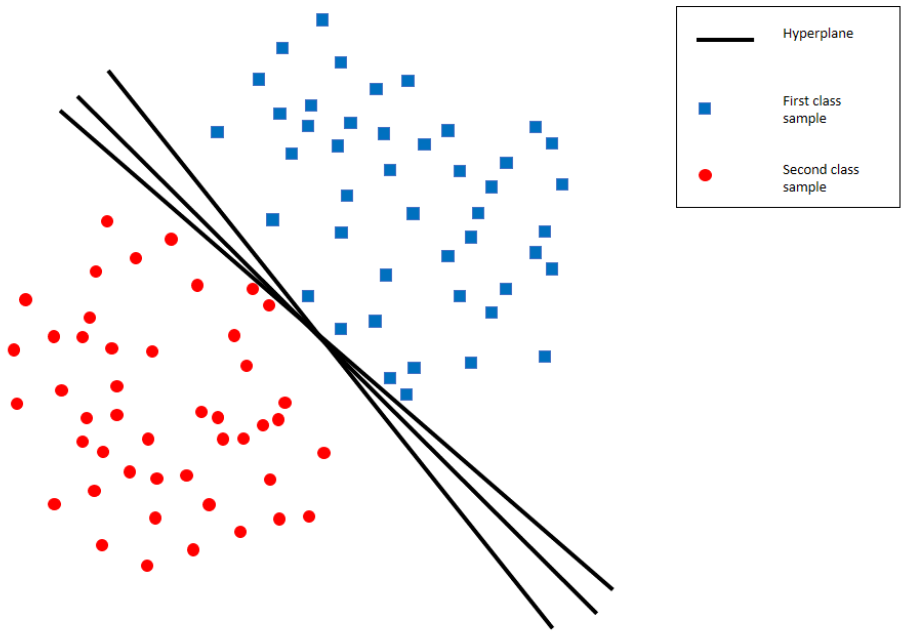


Figure 2.1 An example of separating hyperplanes in two-dimensional space.

The SVM problem is the search of an optimal separating hyperplane such that the distance between it and the closest points of the two classes is the largest, such as illustrated in Figure 2.2.

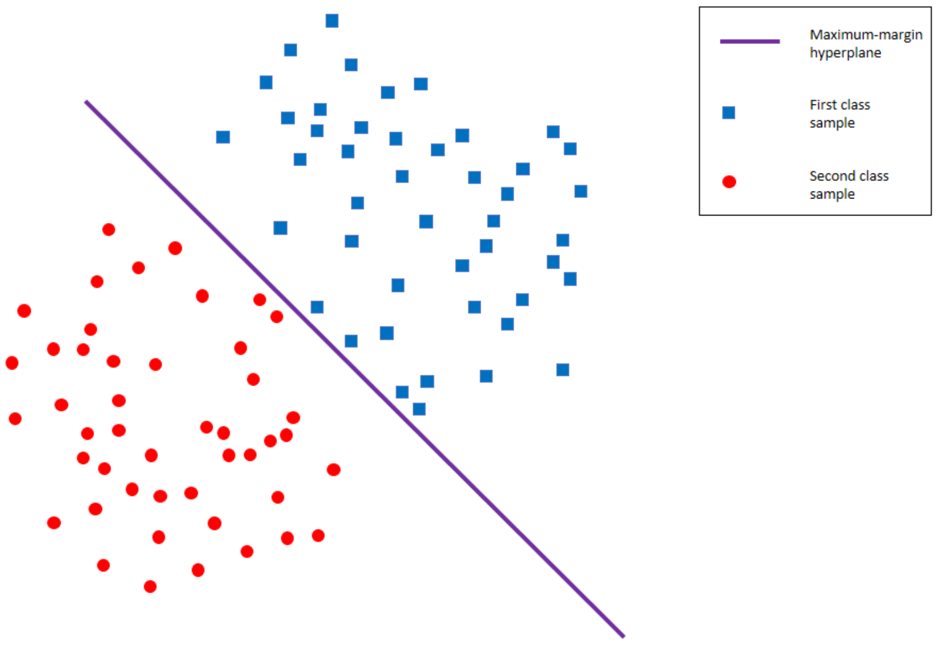


Figure 2.2 An example of an optimal hyperplane in two-dimensional space.

To build such optimal hyperplanes we can consider only small portion of data from the training sample. These points form the support vectors and define the margins of a hyperplane. Computationally, the method is then formulated as an optimization task with linear constraints:

$$\begin{gathered}\min_{w,b}\frac{1}{2}\|w\|^2,\\ 1+\ y_i(\beta-<w,x^i>)\ \le\ 0, i=\ \overline{1,n},\end{gathered} \tag{2.3}$$

where x^i is the i-th feature vector from the training dataset, and y_i is its class label.

Assuming the linear separability of the classes, let us define the penalties. To do so, we first introduce the variables $\xi_i > 0$:

$$\begin{gathered}\min_{w,b,\xi}\frac{1}{2}\|w\|^2+\mathrm{C}\sum_{i=1}^{n}\xi_i,\\ 1+\ y_i*\left(\beta-\ \langle w,x^i\rangle\right)\le\ \xi_i,\\ \xi_i\ge 0, i=\ \overline{1,n},\end{gathered} \tag{2.4}$$

where C is regularization parameter.

Therefore,

$$\langle w,x^i\rangle-\beta\ge 1,\text{ if } y_i=1 \tag{2.5}$$

and

$$\langle w,x^i\rangle-\beta\le -1,\text{ if } y_i=-1. \tag{2.6}$$

The search of the minimum of this function with respect to w, b and ξ in the system (2.4) is equivalent to the search of saddle points of its Lagrangian:

$$\mathrm{L}(\mathrm{w},\beta,\xi,\alpha)=\frac{1}{2}\|w\|^2+\mathrm{C}\sum_{i=1}^{n}\xi_i+\sum_{i=1}^{n}\alpha_i(1+\ y_i\left(\beta-\ \langle w,x^i\rangle\right)-\xi_i)), \tag{2.7}$$

where

$$\xi_i \geq 0, \alpha_i \geq 0, i = \overline{1, n}, \tag{2.8}$$

We differentiate the Lagragian, and, taking into account the constraints, obtain

$$\begin{gathered}
\frac{\partial L(w, \beta, \xi, \alpha)}{\partial w} = w - \sum_{i=1}^{n} y_i \, \alpha_i x^i = 0, \\
\frac{\partial L(w, \beta, \xi, \alpha)}{\partial \beta} = \sum_{i=1}^{n} \alpha_i y_i = 0, \\
\frac{\partial L(w, \beta, \xi, \alpha)}{\partial \xi_i} = C - \alpha_i = 0.
\end{gathered} \tag{2.9}$$

This way, by incorporating the constraints as penalty terms into Lagrangian, a dual optimization problem is defined, which is a quadratic optimization problem with respect to the Lagrangian multipliers α_i.

The dual problem takes the following form:

$$\max_{\alpha \geq 0} \rightarrow \varphi(\alpha),$$

$$\varphi(\alpha) = -\frac{1}{2} \left\| \sum_{i=1}^{n} y_i \, \alpha_i x^i \right\|^2 + \sum_{i=1}^{n} \alpha_i, \tag{2.10}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{2.11}$$

$$0 \leq \alpha_i \leq \mathrm{C}, i = \overline{1, n}. \tag{2.12}$$

By looking at the dual problem, we see that (2.1) can also be expressed using inner products (2.9) as

$$f(x) = \sum_{i=1}^{n} \alpha_i^* y_i \langle x^i, x \rangle + \beta. \tag{2.13}$$

Thus, the SVM training comes down to optimization problem (2.10)-(2.12).

Consider another formulation of the SVM learning problem. When the points from the different classes are not separable, the system might not have a solution and we could consider minimization of the following penalty:

$$\begin{gathered} \min \to \xi \\ 1+\ y_i\big(\beta-\ \langle w,x^i\rangle\big) \le \xi, i=\ \overline{1,n}; \end{gathered} \tag{2.14}$$

Since this penalty is uniform, we solve the following problem:

$$\min \max_{i=\ \overline{1,n}}\{1+\ y_i\big(\beta-\ \langle w,x^i\rangle\big),0\}. \tag{2.15}$$

Let us consider a case of a smooth function:

$$\begin{gathered} \min\frac{1}{2}\|w\|^2+C\xi \\ 1+\ y_i\big(\beta-\ \langle w,x^i\rangle\big) \le \xi, i=\ \overline{1,n}, \xi \ge 0 \end{gathered} \tag{2.16}$$

Its Lagrangian takes the following form:

$$\mathrm{L}(\mathrm{w},\upbeta,\xi,\upalpha)=\ \frac{1}{2}\|w\|^2+\mathrm{C}\xi+\sum_{i=1}^{n}\alpha_i\{1+\ y_i\big(\beta-\ \langle w,x^i\rangle\big)-\xi\}), \tag{2.17}$$

and the dual problem becomes

$$\begin{gathered} \max_{\alpha\ge 0} \to \varphi(\alpha), \\ \varphi(\alpha)=\min_{w,\beta} L(w,\beta,\xi,\alpha)=L(w(\alpha),\beta(\alpha),\xi,\upalpha) \\ \frac{\partial L(w,\beta,\xi,\alpha)}{\partial w}=w-\sum_{i=1}^{n} y_i\,\alpha_i x^i=0, \\ \frac{\partial L(w,\beta,\xi,\alpha)}{\partial \beta}=\sum_{i=1}^{n}\alpha_i y_i=0, \\ \frac{\partial L(w,\beta,\xi,\alpha)}{\partial \xi}=C-\sum_{i=1}^{n}\alpha_i=0, \end{gathered} \tag{2.18}$$

$$\xi \geq 0, \frac{\partial L}{\partial \xi} \geq 0, \xi \frac{\partial L}{\partial \xi} = 0.$$

Upon simplification, the dual problem takes the following form:

$$\begin{gathered}
\max_{\alpha \geq 0} \rightarrow \varphi(\alpha), \\
\varphi(\alpha) = -\frac{1}{2}\left\|\sum_{i=1}^{n} y_i\, \alpha_i x^i\right\|^2 + \sum_{i=1}^{n} \alpha_i, \\
\sum_{i=1}^{n} \alpha_i \leq C, \\
\sum_{i=1}^{n} \alpha_i y_i = 0, \\
\alpha_i \geq 0, i = \overline{1, n}.
\end{gathered} \tag{2.19}$$

2.3 SVM training

For the model training we will use an iterative decomposition method. In each iteration the set of variables splits into two subsets: the subset B of active variables with respect to which the optimization problem is solved in this iteration, and subset N containing the rest of the variables.

The algorithm is as follows:

Step 1. Find a value α_1 satisfying the constraints of the problem (2.20). Assign $q \leq n$ the number of active varibles. Set $k = 1$.

Step 2. If the point α_k is an optimal solution of (2.20), then return α_k. Otherwise, define the set $B \subset \{1, \dots, n\}$ of cardinality q, and the define the set $N \equiv \{1, \dots, n\} \backslash B$. According to these sets, define vectors α_B^k and α_N^k,

Step 3. Solve the following problem with respect to α_B:

$$\max_{\alpha \geq 0} \to -\frac{1}{2}\left\|\sum_{i \in B} y_i \alpha_i x^i\right\|^2 + \sum_{i \in B} \alpha_i - \frac{1}{2}\left\|\sum_{i \in N} y_i \alpha_i x^i\right\|^2 + \sum_{i \in N} \alpha_i \tag{2.20}$$

$$\alpha_i \geq 0, i \in B,$$

$$\sum_{i \in B} \alpha_i \leq C - \sum_{i \in N} \alpha_i, \tag{2.21}$$

$$\sum_{i \in B} \alpha_i y_i = -\sum_{i \in N} \alpha_i y_i. \tag{2.22}$$

Step 4. Let α_B^{k+1} be the optimal solution of the problem (2.19) and $\alpha_N^{k+1} \equiv \alpha_N^k$. Update $k \leftarrow k + 1$. Continue to Step 2.

2.3.1 SMO algorithm

Let us consider a Sequential Minimal Optimization (SMO) method for solving the SVM problem [5]. The method was developed by John Platt and is a method of decomposition with the number of active variables $q = 2$. The SMO algorithm splits the QP problem into subproblems by following Osuna’s theorem to guarantee the convergence [6].

Thus, at each step, SMO finds a solution to a simpler problem. For the standard QP problem of SVM the smallest possible subproblem includes two Lagrange multipliers, in order to conform to the constraints in (2.11). At each step of the algorithm, two Lagrange multipliers are selected to solve the subproblem for these multipliers and updates the parameters of SVM.

The advantage of SMO is that the search for the two Lagrange multipliers can be performed analytically. Therefore, the necessity to solve the QP problem is eliminated.

2.3.1.1 Search of Lagrange multipliers α_i and α_j.

The α_i and α_j need to conform to (2.11) and (2.12). As can be seen in Figure 2.3, the inequality in (2.12) enforces the variables to the location within the square defined by the $a_j = C$, $a_i = C$, $\alpha_j = 0$ and $\alpha_i = 0$. At the same time, the constraint in (2.11) enforces them to be located on the diagonals defined as follows:

If $y_1 \neq y_2$,

$$\alpha_i - \alpha_j = \sum_{\substack{k=1,\\ k\neq i,\\ k\neq j}}^{n} \alpha_k y_k. \tag{2.23}$$

If $y_1 = y_2$,

$$\alpha_i + \alpha_j = \sum_{\substack{k=1,\\ k\neq i,\\ k\neq j}}^{n} \alpha_k y_k. \tag{2.24}$$

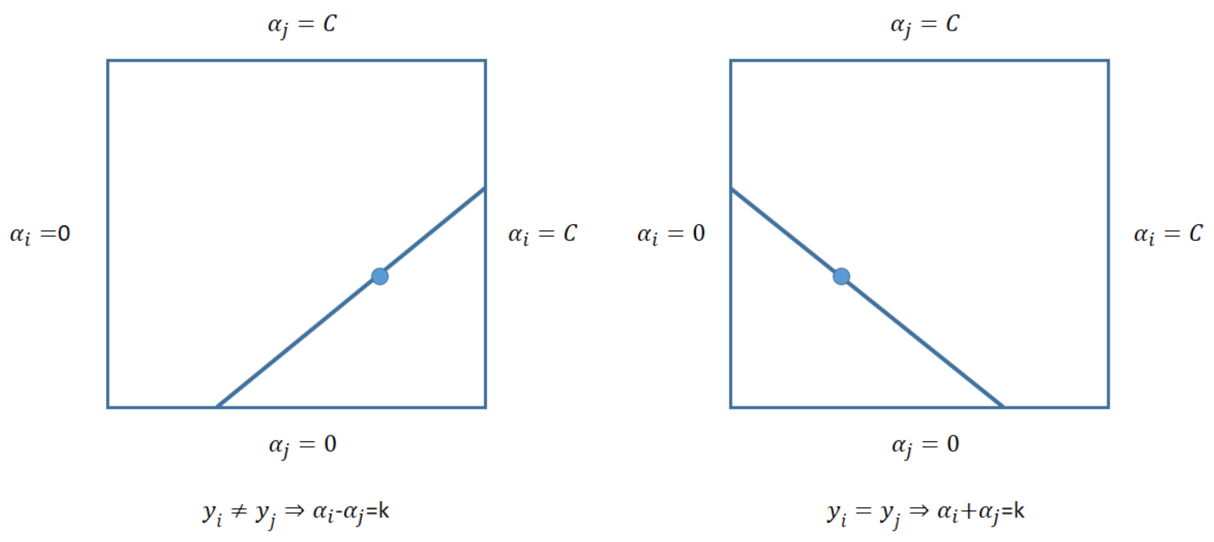


Figure 2.3 An illustration of two Lagrangian multipliers conforming to the constraints of the baseline SVM problem.

Therefore, the variables need to be constrained by the values at the ends of the respective diagonal lines: L for lower, and H, for the higher bound, which can be found in a simple manner. First, the algorithm computes one of the Lagrangian multipliers without conforming to the constraints, e.g., α_j. Next the bounds L and H are found:

$$\mathrm{L} = \max(0, \alpha_j - \alpha_i), \mathrm{H} = \min(\mathrm{C}, \mathrm{C} - \alpha_j + \alpha_i), \text{ if } y_i \neq y_j, \tag{2.25}$$

$$\mathrm{L} = \max(0, \alpha_j + \alpha_i - \mathrm{C}), \mathrm{H} = \min(\mathrm{C}, \alpha_j + \alpha_i), \text{ if } y_i = y_j. \tag{2.26}$$

Update the value of α_j:

$$\alpha_j = \alpha_j - \frac{y_i(E_i - E_j)}{\eta} \tag{2.27}$$

where

$$E_k = f(x^k) - y_k \tag{2.28}$$

$$\eta = 2\langle x^i, x^j\rangle - \langle x^i, x^i\rangle - \langle x^j, x^j\rangle. \tag{2.29}$$

Here E_k is the classification error at the k-th feature vector x^k of the training set, and η is the second derivative of the objective function.

Next, the bounds $[L, H]$ are checked for α_j

$$\alpha_j = \begin{cases} H, if\ \alpha_j > H \\ \alpha_j, if\ 0 \le \alpha_j \le H \\ L, if\ \alpha_j < L. \end{cases} \tag{2.30}$$

The α_i is updated as

$$\alpha_i = \alpha_i + y_i y_j (\alpha_j^{(old)} - \alpha_j) \tag{2.31}$$

where $\alpha_j^{(old)}$ is the value of α_j found following (2.27).

At each step of the algorithm the value of β is updated. In case of $0 < \alpha_i < C$, the value of β is found as

$$\beta_1 = \beta - E_i - y_i(\alpha_i - \alpha_i^{(old)})\langle x^i, x^i\rangle - y_i(\alpha_j - \alpha_j^{(old)})\langle x^j, x^j\rangle. \tag{2.32}$$

Similarly, when $0 < \alpha_j < C$, the value of β is found as follows:

$$\beta_2 = \beta - E_j - y_i(\alpha_i - \alpha_i^{(old)})\langle x^i, x^j\rangle - y_i(\alpha_j - \alpha_j^{(old)})\langle x^j, x^j\rangle. \tag{2.33}$$

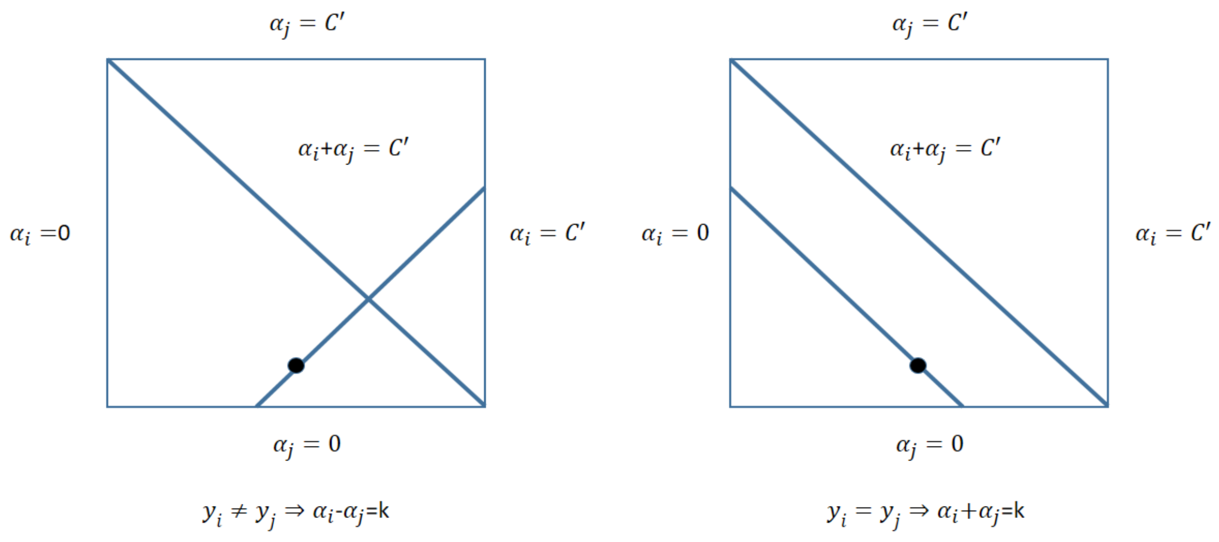
If neither $0 < \alpha_i < C$ nor $0 < \alpha_j < C$ satisfy and both β_1 and β_2 defined, then β is found as their average value:

$$\beta = \begin{cases} \beta_1, if\ 0 < \alpha_i < C \\ \beta_2, if\ 0 < \alpha_j < C \\ \frac{(\beta_1 + \beta_2)}{2}, otherwise. \end{cases} \tag{2.34}$$

Then the pseudo-code of the algorithm will read as follows:

```
• Generate initial $\alpha = \theta$ and set initial $\beta = 0$
• While the number of the maximal iterations has not exceeded:
    o For each data vector in the dataset:
        ▪ If the data vector can be optimized:
            • Randomly select another data vector
            • Jointly optimize with respect to the two vectors
            • If the objective not improving, break the loop
    o If no vectors were optimized, increment the iteration count
```

To solve the problem with the uniform penalties proposed above in (2.19), we now propose the following modifications to SMO. The number of active variables will still set to $q = 2$, but since in this formulation the constraint $\sum_{i=1}^{n} \alpha_i \leq C$, is added, we modify the constraints to the Lagrangian multipliers accordingly. The ends of the diagonal lines on which α_i and α_j should lay in order to conform to the constraints will be computed in a different manner (see also illustration in Figure 2.4).



$\alpha_j = C'$
$\alpha_i + \alpha_j = C'$
$\alpha_i = 0$
$\alpha_i = C'$
$\alpha_j = 0$
$y_i \neq y_j \Rightarrow \alpha_i - \alpha_j = k$
$\alpha_j = C'$
$\alpha_i + \alpha_j = C'$
$\alpha_i = 0$
$\alpha_i = C'$
$\alpha_j = 0$
$y_i = y_j \Rightarrow \alpha_i + \alpha_j = k$

Figure 2.4 An illustration of two Lagrangian multipliers conforming to the constraints of the proposed SVM problem (2.19).

Let us assume $C' = C - (\alpha_i + \alpha_j)$, then the bounds L and H can be found as follows:

$$\mathrm{L} = \max(0, \alpha_j - \alpha_i), \mathrm{H} = \min\left(\mathrm{C}', \frac{1}{2}(\mathrm{C}' - \alpha_j + \alpha_i)\right), \text{if } y_i \neq y_j, \tag{2.35}$$

$$\mathrm{L} = \max(0, \alpha_j + \alpha_i - \mathrm{C}'), \mathrm{H} = \min(\mathrm{C}', \alpha_j + \alpha_i), \text{if } y_i = y_j. \tag{2.36}$$

3 Experiments

We will analyze the performance of the baseline and proposed methods on the publicly available Iris plants dataset [7], [8].

In this chapter we provide the description of the experimental setup and the validation methodology employed in this work.

3.1 Experimental design

Monte Carlo cross-validation (MCCV) scheme in our experiments. This method randomly selects (without replacement) some part of the data to create a training data set, and then assigns the rest of the points to the test data set. This process is repeated several times and each iteration we generate new training and test sets (see Figure 3.1).

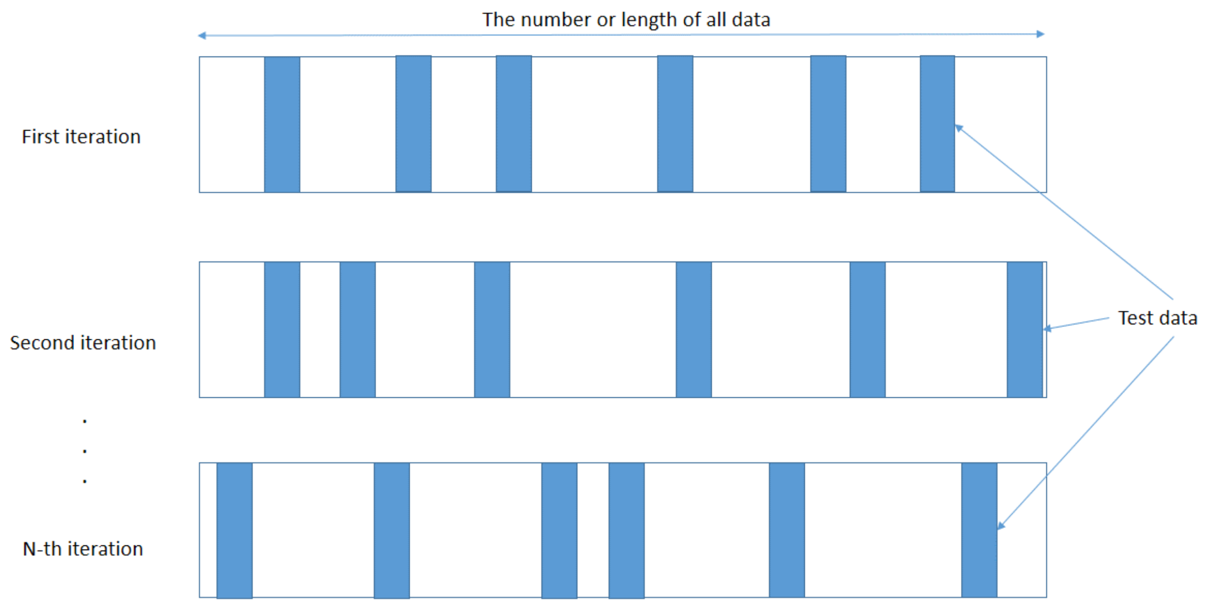


Figure 3.1 Data split in MCCV

Suppose we have a set of 100 data point and we want to use 10% of our data to test our model. Then on the first run our test set can contain these points: 35, 13, 21, 73, 50, 52, 18, 79, 39 и 63. On the next iteration, our test data might be 8, 21, 6, 59, 66, 97, 33, 79, 69 и 58. Since the partitions are performed independently for each iteration, the same point may appear in the test set several times, which is the main difference between the Monte-Carlo and traditional cross-validation. MCCV allows to explore several more possible data splits, although it is unlikely to consider all of them as there are $C_{100}^{50} \approx 10^{28}$ possible ways to split a set of 100 points equally.

Now we split the data set to training and test data sets. Training set is the data on which we fit the model, and the test set is the data that we will use to see how well the model works on unseen data.

3.2 Validation metrics

We consider a binary classification problem with two given classes C_{-1} or C_1 and we want to find the mapping $f(x,w) \rightarrow \{-1,1\}$.

As classification problems might be the most common type of machine learning problems, there is a lot of metrics that can be used to evaluate their performance.

Let us consider the following indicators (see also Figure 3.2):

- True Positives = $| \{u_i \in C_1 \,| y(x_i,w) = 1\} |$,
- True Negatives = $| \{u_i \in C_{-1} \,| y(x_i,w) = -1\}|$,
- False Positives =$| \{u_i \in C_{-1} \,| y(x_i,w) = 1\} |$,
- False Negatives = $| \{u_i \in C_1 \,| y(x_i,w) = -1\} |$,

where u belongs to given set of objects U.

For the sake of brevity further we shall use the following notations: TP, TN, FP, FN.

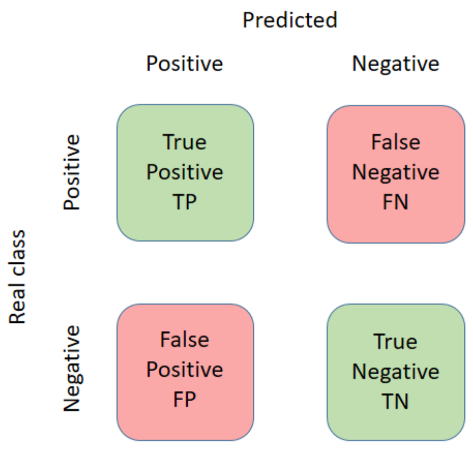


Figure 3.2 Illustration of the true positive, false negative, false positive and true negative indicators.

It is obvious, that TP + TN + FP + FN = N, где N = |U|.

There are some metrics that can be used to evaluate predictions for binary classification problem.

- Type I error:

Error of the first kind, false positive classification or false alarm, False Positive Rate:

$$\alpha = \frac{FP}{TN + FP}$$

- Type II error:

Error of the second kind, false negative classification or miss, False Negative Rate:

$$\beta = \frac{FN}{TP + FN}$$

- Accuracy:

$$Accuracy = \frac{TP + TN}{N}$$

- Precision:

The fraction of positively classified objects that truly belong to the positive class over the total number of positively classified objects:

$$Precision = \frac{TP}{TP + FP}$$

- Recall:

The fraction of positively classified objects that truly belong to the positive class over the total number of objects that belong to a positive class:

$$Recall = \frac{TP}{TP + FN}$$

In addition to the above performance metrics, we record the number of iterations that the methods require at each run of the MCCV experiment, which will serve us as the computational performance metric.

3.3 Statistical analyses

The statistical significance of the difference in classification performance and computational performance was done using Wilcoxon's signed rank test for two samples [9], [10]. In our experimental setting we were able to assume the two performance metrics to be paired since each MCCV experiment was conducted such that both methods were initialized with the same initial parameters, and were validated using the same training and test splits. The significance level was chosen as $\alpha = 0.001$.

4 Implementation details

In this Chapter, we provide the software solution specification that was developed as part of our experimental setup.

SVM(self, max_iter=10000, kernel_type='linear', C=1.0, epsilon=0.001**)**

Model initialization

Parameters:	**max_iter: int (default=10000)**
	Limit on the number of iteration without solving the optimization problem
	kernel_type: string (default='linear')
	Specifies the kernel type to be used in the model. It must be 'linear' or 'quadratic'. If nothing is specified, "linear" will be used.
	C: float (default=1.0)
	Correlation coefficient.
	epsilon: float (default=0.001)
	The minimum difference between the new and the old value of α

Methods.

Table 4.1 The methods and their description

fit(self, X, y, alphaInit = None, imodel = 0)	Training the both SVM algorithms
predict(self, X)	Object classification based on X characteristic vector
calc_b(self, X, y, w)	Calculation of the parameter of the model β
calc_w(self, alpha, y, X)	Calculation of the parameter of the model w
E(self, x_k, y_k, w, b)	Calculation of prediction error
compute_L_H(self, C, alpha_prime_j, alpha_prime_i, y_j, y_i)	Calculation of boundary values of α for standard SVM model
compute_L_H_2(self, C, alpha_prime_j, alpha_prime_i, y_j, y_i)	Calculation of boundary values of α for second type of SVM model
get_rnd_int(self, a,b,z)	Selection the random number between a and b that is not equal to z
kernel_linear(self, x1, x2)	Define linear kernel
kernel_quadratic(self, x1, x2)	Define RBF kernel

fit(self, X, y, alphaInit = None, imodel = 0**)**

Training the SVM model

Parameters:

X: {array-like}, shape (n, d)
The matrix of vectors of characteristics from the training set, where (n, d) is the expected shape of X

y : array-like, shape (n)
The vector of target values, expected shape of y is n

alphaInit: array-like, shape (n)
Initial value of α

Imodel: int (default = 0)
The type of SVM algorithm. imodel=0 – standard SVM, if imodel=1 – second type of SVM

Returns:

support_vectors: array-like
The list of support vectors

count: int
The number of support vectors

alpha: array-like, shape (n)
Lagrange multipliers α

predict(self, X**)**

Object classification based on X characteristic vector

In the case of a binary classification, returns -1 or 1.

Parameters:

X: {array-like}, shape (n, d)
The matrix of vectors of characteristics from the training set, where (n, d) is the expected shape of X

Returns:

y_pred: int
Class labels for samples in X.

calc_b(self, X, y, w**)**

Calculation of the parameter of the model β

Parameters: **X: {array-like}, shape (n, d)**

The matrix of vectors of characteristics from the training set, where (n, d) is the expected shape of X

y : array-like, shape (n)

The vector of target values, expected shape of y is n

w: array-like, shape (d)

Parameter of the model w, expected shape of w is d

Returns:

b: float

Parameter of the model β

calc_w(self, alpha, y, X**)**

Calculation of the parameter of the model w

Parameters: **alpha: array-like, shape (n)**

Lagrange multipliers α

y : array-like, shape (n)

The vector of target values, expected shape of y is n

X: {array-like}, shape (n, d)

The matrix of vectors of characteristics from the training set, where (n, d) is the expected shape of X

Returns:

w: array-like, shape (d)

Parameter of the model w, expected shape of w is d

E(self, x_k, y_k, w, b**)**

Calculation of prediction error

Parameters: **x_k: array-like, shape (d)**

The vectors of characteristics of k-th object, expected shape of x_k is d

y_k : int

Target values of the k-th object

w: array-like, shape (d)

Parameter of the model w, expected shape of w is d

b: float

Parameter of the model β

Returns: **E_k: float**

Classification error

compute_L_H(self, C, alpha_prime_j, alpha_prime_i, y_j, y_i**)**

Calculation of boundary values of α for standard SVM model

Parameters:

C: float

Correlation coefficient.

alpha_prime_j: float

j-th Lagrange multiplier α_j

alpha_prime_i: float

i-th Lagrange multiplier α_i

y_i : int

Target values of the i-th object

y_j : int

Target values of the j-th object

Returns:

L, H: float

Boundary values of α_j

compute_L_H_2(self, C, alpha_prime_j, alpha_prime_i, y_j, y_i**)**

Calculation of boundary values of α for second type of SVM model

Parameters:

C: float

Correlation coefficient.

alpha_prime_j: float

j-th Lagrange multiplier α_j

alpha_prime_i: float

i-th Lagrange multiplier α_i

y_i : int

Target values of the i-th object

y_j : int

Target values of the j-th object

Returns:

L, H: float

Boundary values of α_j

get_rnd_int(self, a, b, z**)**

Selection the random number between a and b that is not equal to z

Parameters:	**a: int**
	Lower boundary value
	b: int
	Upper boundary value
	z: int
	The number i cannot be equal to
Returns:	
	i: int
	Random number

kernel_linear(self, x1, x2**)**

Define linear kernel

Parameters:	**x1: array-like, shape (d)**
	The vectors of characteristics of the object
	x2: array-like, shape (d)
	The vectors of characteristics of another object
Returns:	
	kernel: float
	The value of kernel function

kernel_quadratic(self, x1, x2**)**

Define quadratic kernel

Parameters:	**x1: array-like, shape (d)**
	The vectors of characteristics of the object
	x2: array-like, shape (d)
	The vectors of characteristics of another object
Returns:	
	kernel: float
	The value of kernel function

5 Results

This chapter summarizes the results of experiments described in the previous chapters. In order to compare the baseline and proposed SVMs, the following steps were conducted:

1. the performance metrics were chosen as accuracy, recall, precision, and f1-score, as well as the number of iterations to conversion;

2. m=100 runs of MCCV were conducted with a random split of training and testing sets with 75% and 25% of data, respectively;

3. the resulting metrics vectors were compared using Wilcoxon's signed rank test to identify the statistical significance of the findings.

5.1 Performance

Table 5.1 summarizes the performance metrics for the two methods.

Table 5.1 The median of the metrics for each method

Median values	Baseline SVM	Uniform SVM
Accuracy	0.78	0.85
Precision	0.65	0.72
Recall	1.00	1.00
F1-Score	0.78	0.82
Number of iterations	10.00	9.00

As we can see from Table 5.1, the median values the performance metrics are consistently higher for the proposed method and the median value for the number of iterations is greater for the baseline method.

The differences of the metrics were as follows: the median difference of 10.8% at $p \ll 0.001$ for accuracy (see also Figure 5.1), the median difference of 9.0% at $p \ll 0.001$ for f1-score (see also Figure 5.2), the median difference of 14.0% at $p \ll 0.001$ for precision (see also Figure 5.3) and no statistically significant difference found for recall (see also Figure 5.4). Such pairwise comparison was possible, since our implementation of the MCCV provided both methods with the same sets of training and test data as well as the initial parameters (described in Chapters 3,4)

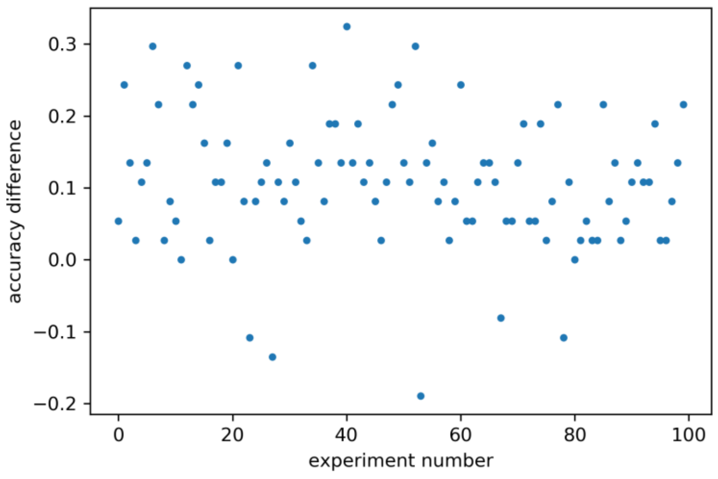


Figure 5.1 The difference of the accuracy between the proposed and baseline SVMs over the 100 MCCV experiments.

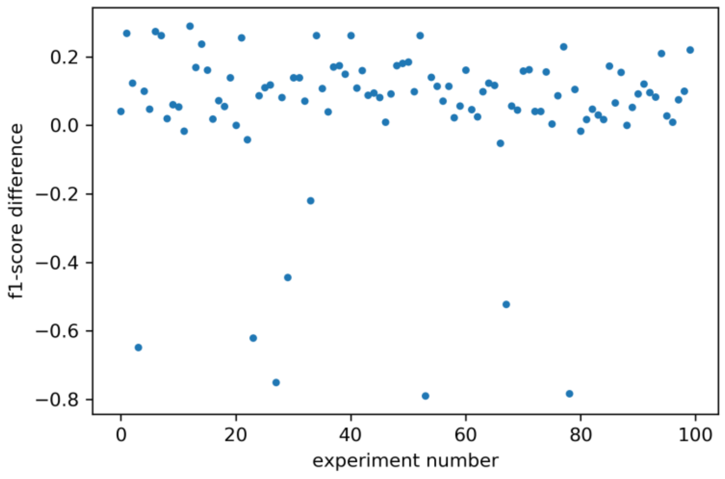


Figure 5.2 The difference of the f1-score between the proposed and baseline SVMs over the 100 experiments.

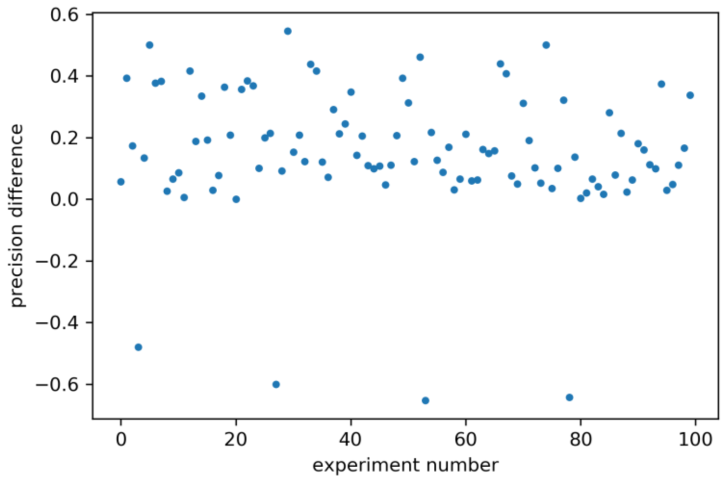


Figure 5.3 The difference of the precision between the proposed and baseline SVMs over the 100 experiments.

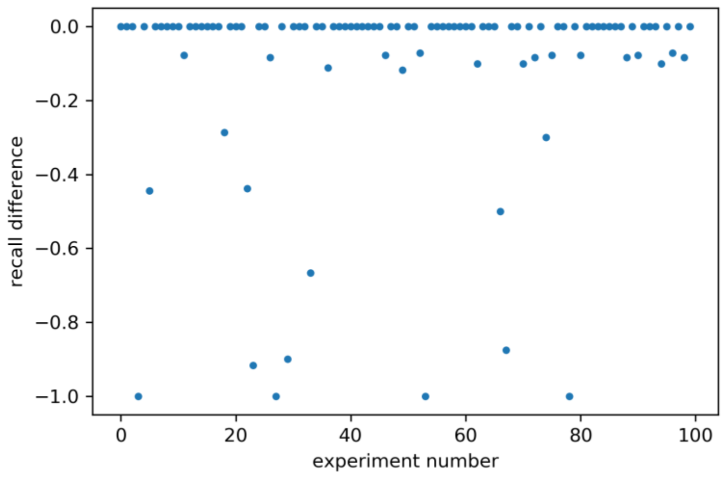


Figure 5.4 The difference of the recall between the proposed and baseline SVMs over the 100 experiments.

5.2 Computational performance analysis

The comparison of the computational performance was done in terms of the number of iterations that was required by the algorithms to converge. The difference of the number of iterations over the 100 experiments is illustrated in Figure 5.5. There was statistically significant reduction of the number of iterations when using the proposed method (median decrease of 3 iterations, $p \ll 0.001$, Wilcoxon's signed rank test).

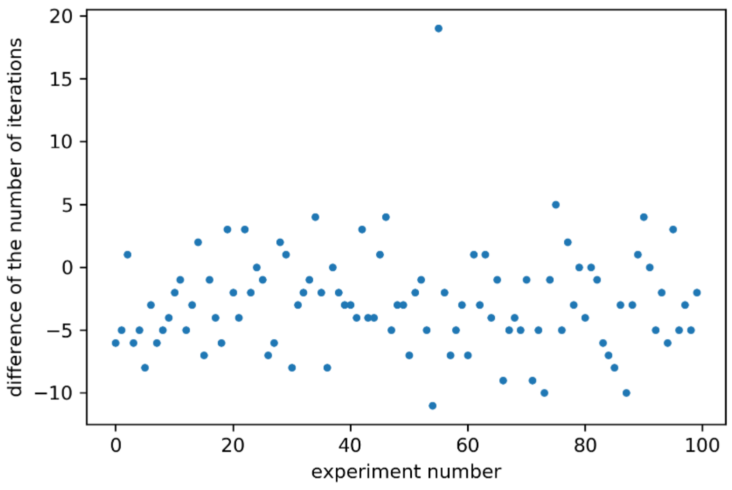


Figure 5.5 The difference of the number of iterations between the proposed and baseline SVMs over the 100 experiments.

5 Discussion

The machine learning tasks typically suggest a compromise between efficiency and accuracy, with accurate methods being more computationally expensive. In this work, we proposed a new modification of the SVM algorithm that provides improvements over the baseline SVM both in terms of accuracy and efficiency.

Our experiments on public data demonstrate statistically significant improvements in classification performance in terms of accuracy, precision and f1-score. The performance in terms of recall were at the same level as the baseline model without significant differences, indicating that source of the significant improvement of 9.8% for F1-score was the precision (statistically significant improvement of 13.1%). Improvement in terms of the latter can be impactful in applications such as computerized diagnostics in screening as lowering of false positives as it implies that there will be fewer subjects that require further clinical investigations, thus lowering the phycological pressure and financial costs associated with over-diagnosis [11].

The number of iterations also affects the applicability of the method. As our results suggest, the proposed method can lower the number of iterations needed to converge with a statistically significant difference. As such, its implementation in real-world applications and systems can potentially lower the time costs of the solutions.

The design of our experiments followed a probabilistic approach via the use of MCCV technique. As the comparative study of the two algorithms was the primary goal, we ensured fair comparison by providing same training and test splits as well as initial parameters to both baseline and proposed methods. By doing so, we enabled the pairwise statistical analysis (Wilcoxon's signed rank test) to identify the significance in the set of performance metrics.

Conclusion

To conclude, we proposed a new modification of the SVM algorithm that employs a new regularization term. The experiments on a public dataset demonstrated statistically significant improvements both in terms of accuracy and computational performance over the baseline SVM algorithm. As such, the method has a potential to be implemented in real-world applications.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[3] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[4] V. N. Vapnik, *Statistical Learning Theory*, 1 edition. New York: Wiley-Interscience, 1998.

[5] J. C. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization,” in *Advances in kernel methods*, MA, USA: MIT Press, 1999.

[6] E. Osuna, R. Freund, and F. Girosi, “An improved training algorithm for support vector machines.,” in *Proc. of IEEE NNSP’97*, 1997.

[7] “UCI Machine Learning Repository: Iris Data Set.” [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris. [Accessed: 21-May-2019].

[8] R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[9] F. Wilcoxon, “Individual Comparisons by Ranking Methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[10] A. Benavoli, G. Corani, J. Demsar, and M. Zaffalon, “Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis,” *arXiv:1606.04316 [cs, stat]*, Jun. 2016.

[11] S. Vaccarella, S. Franceschi, F. Bray, C. P. Wild, M. Plummer, and L. Dal Maso, “Worldwide Thyroid-Cancer Epidemic? The Increasing Impact of Overdiagnosis,” *N. Engl. J. Med.*, vol. 375, no. 7, pp. 614–617, Aug. 2016.