UDC 004.855.5(045)
DOI:10.18372/1990-5548.75.17553

[1]V. M. Sineglazov,
[2]A. O. Samoshyn

# SEMI-SUPERVISED SUPPORT VECTOR MACHINE

[1]Aviation Computer-Integrated Complexes Department, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine
[2]Educational and Scientific Institute of Applied System Analysis, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua ORCID 0000-0002-3297-9060, [2]samoshyn.andriy@lll.kpi.ua

***Abstract*—***The article considers a new approach to constructing a support vector machine with semi-supervised learning for solving a classification problem. It is assumed that the distributions of the classes may overlap. The cost function has been modified by adding elements of a penalty to it for labels not in their class. The penalty is represented as a linear function of the distance between the label and the class boundary. To overcome the problem of multicriteria, a global optimization method known as continuation is proposed. For a combination of predictions, it is suggested to use the voting method of models with different kernels. The Optuna framework was chosen as the tool for configuring hyperparameters. The following were considered as training samples: type_dataset, banana, banana_inverse, c_circles, two_moons_classic, two_moons_tight, two_moons_wide.*

**Index Terms—**Support vector machine; semi-supervised learning; multi-class classification; multicriteria; method of global optimization.

## I. INTRODUCTION

Support Vector Machine (SVM) it is one of the most effective machine learning methods used to solve classification and regression problems, widely used in many fields, including computer vision, natural language processing, bioinformatics, and others.

An important property of support vector machines is that the determination of model parameters corresponds to a convex optimization problem, and therefore any local solution is also a global optimum.

This method is based on finding the hyperplane in the feature space that best separates the two classes of data. The main idea of the method is the translation of the original vectors into a higher-dimensional space and the search for a separating hyperplane with the largest gap in this space. Two parallel hyperplanes (support vectors) are built on both sides of the hyperplane that separates the classes. *The separating hyperplane* will be the hyperplane that creates the greatest distance to two parallel hyperplanes. The algorithm is based on the assumption that the greater the difference or distance between these parallel hyperplanes, the smaller the average classifier error will be.

The special property of the support vector machine is the continuous decrease in the empirical classification error and the increase in the gap, so the method is also known as ***the maximum gap classifier method.***

## II. FORMALIZATION OF THE METHOD SVM

Let's enter the notation. In general, the problem is solved for which class labels can take values $Y = \{-1, +1\}$. The object is a vector with $n$ features $x = (x_1, x_2, \ldots, x_n)$ in space $R^n$. During training, the algorithm must build a function $F(x) = y$, which takes an argument $x$ (object in space $R^n$) and returns the label of the class $y$.

The main purpose of SVM as a classifier is to find the equation of the separating hyperplane $w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = 0$ in space $R^n$, which will divide the two classes in the most optimal way. General view of the transformation $F$ object $x$ in the class label $Y$: $F(x) = \text{sign}(w^T x - b)$. After adjusting the algorithm weights $w$ and offset $b$, all objects that fall on one side of the constructed hyperplane will be predicted as the first class, and objects that fall on the other side as the second class.

There are different methods for constructing the separating hyperplane, but in the case of SVM, the weights $w$ and $b$ are adjusted so that the objects of the classes are as far away from the separating hyperplane as possible. In other words, the algorithm maximizes the margin between the hyperplane and the objects of the classes that are closest to it [1].

If the sample is linearly separable, then simple geometric understanding leads to the following

problem: it is necessary to find the values of the parameters $w$ and $b$, at which the norm of the vector $w$ is minimal. This is formalized into the following problem:

$$\begin{cases} \dfrac{w^{\mathrm{T}}w}{2} \to \min, \\ y_i\left(w^{\mathrm{T}}x_i + b\right) \ge 1, \ i = 1, ..., k, \end{cases}$$

where $k$ are number of training sample objects.

In order to generalize SVM to the case of linear non-separability, let's allow the algorithm to make errors on training examples, but at the same time try to reduce the number of errors. For this, we introduce an additional set of variables $\xi_i$, which characterize the magnitude of the error on the objects $x_i$ and to the functional that is minimized, we will add a penalty for the general error and its regularization by the coefficient $C$ (adjustable parameter).

One of the main innovations brought by the support vector machine is the **kernel trick**.

It can be shown that the linear function used in the support vector machine can be rewritten as:

$$w^{\mathrm{T}}x + b = b + \sum_{i=1}^{m}\alpha_i x^{\mathrm{T}}x^{(i)} = b + \sum_{i=1}^{m}\alpha_i k(x, x^{(i)}), \quad (1)$$

where $\boldsymbol{x^{(i)}}$ is the training example, α is a vector of coefficients, and $k(x, x^{(i)})$ is the kernel function (kernel).

The most commonly used functions are:

- *a linear kernel*: $K(x_i,x_j) = x_i^{\mathrm{T}}x_j$, which corresponds to the classifier on the support vectors in the original space;
- *polynomial kernel with degree p*: $K(x_i,x_j) = \left(1 + x_i^{\mathrm{T}}x_j\right)^p$;
- *Gaussian kernel with radial basis function (RBF)*: $K(x_i,x_j) = \exp(\gamma\|x_i - x_j\|2)$;
- *sigmoid kernel*: $K(x_i,x_j) = \tanh\left(\gamma x_i^{\mathrm{T}}x_j + \beta_0\right)$.

Each core is characterized by parameters ($p$, γ, $\beta_0$ etc.), which are subject to optimization

The main idea of using kernels is that when mapping data to a higher-dimensional space, the original set of points can become linearly separable [2].

The kernel trick is useful for two reasons. First, it allows (as discussed above) to train non-linear x models using convex optimization methods that are known to converge efficiently. Second, the kernel function $k$ often allows for a much more computationally efficient implementation. The most common is the Gaussian kernel

$$k(\boldsymbol{u}, \boldsymbol{v}) = N(u - \boldsymbol{v}; 0, \sigma^2\boldsymbol{I}), \quad (2)$$

where $N(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the standard density function of the normal distribution.

## III. Semi-supervised Learning in the Problem of SVM Synthesis

Labeled data are used to tune the SVM. However, they can be expensive, time-consuming, and difficult to access in many applications. Semi-supervised learning (SSL) aims to take advantage of large amounts of unlabeled data to improve learning performance. Empirical evidence suggests that in certain favorable situations unlabeled data may help, while in other situations it may not. As a result, several attempts have recently been made to develop a theoretical understanding of semi-supervised learning. It is generally accepted that unlabeled data can only help when there is a relationship between the marginal distribution of the data and the objective function to be studied.

## IV. Problem Statement

The transductive support vector machine (TVSM) algorithm is used as the base in this work.

Let the training set consist of $l$ labeled pairs ($x_i$, $y_i$), where $x_i$ – feature vector, $y_i$ – the label of the class that belongs to the set {1, –1}. Let there also exist an unlabeled sample $x_1^*,...,x_k^*$, which belongs to the same distribution as the marked objects. Then transductive learning based on the support vector method can be described by the following optimization problem [3]:

$$\begin{cases} \dfrac{w^{\mathrm{T}}w}{2} + C\sum\limits_{i=1}^{l}\xi_i + C^*\sum\limits_{j=1}^{k}\xi_i^* \to \min, \\ y_i\left(w^{\mathrm{T}}x_i + b\right) \ge 1 - \xi_i, \ i = 1,\ldots,l, \\ y_j^*\left(w^{\mathrm{T}}x_j^* + b\right) \ge 1 - \xi_j^*, \ j = 1,\ldots,k, \\ \xi_i \ge 0, \ i = 1,\ldots,l, \\ \xi_j^* \ge 0, \ j = 1,\ldots,k. \end{cases}$$

Solving this problem means finding a label $y_1^*,...,y_k^*$, labeled data $x_1^*,...,x_k^*$, and get a hyperplane $(w, b)$, so that this hyperplane separates both training and test data with maximum margin. $\xi_i$ and $\xi_j^*$ is the changing marked and unmarked objects, respectively, to handle linear non-separability. $C$ and $C^*$ – these are options that the user sets. They allow you to trade the size of the margin against the misclassification of training and unmarked objects.

## V. REVIEW OF SVM IN SSL

Among the popular approaches of semi-supervised learning S3VM [4] – [6] is based on the low-density assumption and try to learn a low-density separator that favors the solution boundary passing through low-density regions in the feature space [7]. These approach has already been applied to various applications such as text classification [6], image search [8], bioinformatics [9], natural language processing [10] natural language processing etc. However, as with other semi-supervised learning approaches, it has been found that S3VM can degrade performance when using unlabeled data [11], [8], [12], [13].

Paper [14] is devoted to concider SVM with partial teacher training based on the use of the continuation method, which is an effective tool for solving optimization problems with complex constraints that take into account uncertainty in semi-supervised learning. The authors note that SVM is one of the most popular classification methods, however, in many situations, data collection with labeled data can be a very complex or expensive process.

The work [1] considers the SVM, which is one of the most popular machine learning techniques using kernels. The SVM method is considered as a solution to an optimization problem that reduces to maximizing the width of the gap between classes in the discriminant function. The notion of regularization and soft separating hyperplane are discussed in order to avoid overfitting.

## V. DEVELOPMENT OF SSL SVM

Currently, there are a number of learning algorithms with SSL SVM:

- Proxy Labeling SVM;
- Noisy Student SVM;
- Co-training SVM;
- MixMatch SVM;
- S3VM;
- Transductive SVM.

Consider their features.

*The first proxy-tagged* SVM approach involves using an initial SVM to create proxy labels for unlabeled data, and then training a new SVM on the combined labeled and proxy labeled datasets. This approach uses information about labeled data to improve the accuracy of predictions on unlabeled data.

*The second approach* involves iteratively training the SVM model on a growing labeled dataset, adding the highest confidence predictions from the previous iteration and adding noise to the new labeled data. This approach uses the information in the unlabeled data to generate additional labeled examples and to improve the accuracy of the final SVM model.

*The third approach* involves training two SVMs on two different subsets of labeled data, and then iteratively adding examples to each subset using a different SVM to label them. This approach uses unlabeled data to improve the accuracy of SVM models by generating additional labeled data.

*The fourth approach* involves combining labeled and pseudolabeled examples into a mixed dataset and applying data extension techniques to improve model robustness. This approach uses unobserved data to generate additional examples used to train the SVM model and improve its accuracy.

*The fifth approach* uses both labeled and unlabeled data for SVM training. Unlabeled data is used to learn the structure of the data, while labeled data is used to configure SVM parameters. This method is shown to be effective in reducing the amount of labeled data required for training.

*The sixth approach* also uses both labeled and unlabeled data for SVM training. The approach is similar to S3VM, but instead of treating unlabeled data as a pool of candidates for future labeling, TSVM treats all untagged data as test data and tries to find a decision limit that better separates labeled data while minimizing error on unlabeled data.

Each of these approaches has disadvantages. In our opinion, the TSVM has the greatest advantages over the others, so this algorithm is taken as a basis. One way to improve it is to use the voting method of several algorithms: aggregating the results of each individual classifier and determining the major prediction based on the largest majority of votes. The idea is that instead of creating separate specialized models and looking for accuracy for each of them, we create a single model that learns from these models and predicts the result based on their overall majority vote for each class of results. Thus, as several different algorithms in nature, it is possible to use several different kernel functions (eg, radial basis, linear, sigmoidal, polynomial) to train different SVMs.

Optimization of hyperparameters is an important stage when working with TSVM, as well-tuned hyperparameters can improve the stability of the

algorithm. Some possible hyperparameter optimization techniques for TSVM:

- *Grid Search:* This method consists of choosing a set of hyperparameters and iterating through all possible combinations of these hyperparameter values to find the best parameters. The disadvantage of this method is that it is computationally expensive for large numbers of hyperparameters and their values.
- *Random Search:* This method is to randomly select a set of hyperparameters and perform model training on this set of hyperparameters. This method is less time consuming than Grid Search, but may require more iterations to find the best parameters.
- *Bayesian Optimization*: This method is more complicated, but it can help reduce the number of iterations to find the best parameters. It is based on a model of Gaussian processes, which allows estimation of the cost function based on information about the accuracy of the model for certain values of the hyperparameters. From this model, new sets of hyperparameters are generated for the next iteration.

Bayesian optimization can be less computationally demanding than lattice and random search because it considers fewer combinations of hyperparameters. In addition, Bayesian optimization allows the model to focus on those hyperparameters that are important to achieve the best results.

The Optuna framework was chosen as the hyperparameter tuning tool, which is an open source framework for automatic hyperparameter optimization that uses Bayesian optimization to efficiently find optimal hyperparameter values. This framework has become popular due to its ease of use, speed and flexibility [15].

Optuna has several advantages that make it the optimal choice for SVM hyperparameter tuning using Bayesian optimization. Below are a few of them:

- *Ease of use:* Optuna has a simple and intuitive interface that makes it easy to configure and run SVM hyperparameter optimization.
- *Efficiency:* Optuna uses Bayesian optimization, which is an efficient method for finding optimal hyperparameter values. This allows you to focus on the hyperparameters that are important to achieve the best results.
- *Flexibility:* Optuna allows you to use various optimization methods, including Bayesian optimization, which allows you to find optimal hyperparameter values in a wide range of machine learning problems.

To configure TSVM hyperparameters, you need to define a range of possible hyperparameter values to search for. In our case, ranges of values were defined:

- $C$ [0.5, 5];
- $C^*$ [0.5, 5];
- kernel [‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’];
- gamma [0.05, 0.75, ’scale’].

Let's define the function of evaluating the quality of the adjusted model on the validation sample. The F1-metric is selected, which is maximized.

After optimization, the following hyperparameters were obtained:

- $C = 1$;
- $C^* = 0.5$;
- Kernel = ‘sigmoid’
- Gamma = 0.5

Due to the fact that TSVM is quite sensitive to outliers in the training sample, it is proposed to create a combination of several TSVM algorithms with different settings. For a combination of predictions, it is suggested to use the voting method of models with different kernels [16]. In addition to the already mentioned reduction in sensitivity to outliers, the voting method may have several other advantages compared to the prediction of a separate algorithm:

- *Reducing the risk of overfitting*: when only one algorithm is used, there is the possibility of overfitting, when the model becomes overly complex and fits the training data precisely, which reduces its ability to generalize to new data. Using multiple algorithms in voting reduces this risk, as each algorithm solves the problem with its own approach and hyperparameters.
- *Improved prediction accuracy:* Provided the algorithms used have different methods and hyperparameters, better accuracy can be achieved than using a single algorithm alone. Each algorithm can highlight different aspects in the data, so a combination of them can give a better result.
- *Increasing the diversity of results*: If the algorithms used have significant differences in their methods and hyperparameters, then voting can produce diverse results for different inputs, providing a wider coverage of possible answers and reducing the probability of false predictions.

A separate TSVM model with different kernel parameters is created for each kernel (['linear', 'poly', 'rbf', 'sigmoid']) $F_{lin}, F_{poly}, F_{rbf}$ and $F_{sigm}$, which are trained on the same data set. After training the defined algorithms, predictions are made $y^*_{lin}, y^*_{poly}, y^*_{rbf}$ and $y^*_{sigm}$) (on marked data $x^*_1,\ldots,x^*_k$ and decision functions are calculated

$\left(dec^*_{lin}, dec^*_{poly}, dec^*_{rbf}$ and $dec^*_{sigm}\right)$, which are proportional to the distances from the predicted objects to the trained separating hyperplane.

The resulting decision functions from different TSVM models are combined using median voting:

$$dec^*_{comb} = \text{median}\left(\left[dec^*_{lin}, dec^*_{poly}, dec^*_{rbf}, dec^*_{sigm}\right]\right).$$

Each prediction of the TSVM model counts as a vote, and the final decision is made based on the median of these votes. The final prediction is formed according to the rule that if the median is greater than or equal to 0, then it is classified as class 1, otherwise it is classified as class –1:

$$y^*_{comb} = \begin{cases} 1, & \text{if } dec^*_{comb} \geq 0, \\ -1, & \text{else.} \end{cases}$$

After voting, a final prediction is obtained, which can be compared with the actual values.

As a result, we have the following algorithm

1. Data preparation.

1.1. Dividing the dataset into separate parts: training and validation labeled samples, unlabeled data.

1.2. Application of data preprocessing: filling gaps, normalization of features.

2. TSVM training and optimization.

2.1. Select initial kernel for TSVM (linear, radial, polynomial or sigmoid).

2.2. Set initial values of hyperparameters such as *C*, *Cu* and gamma.

2.3. Initial training.

2.3.1. We initialize the parameters by setting the positive class label to +1 and the negative class label to –1.

2.3.2. We calculate the ratio of positive labeled examples in X1 to the total number of labeled examples. This ratio is then used to determine the number of positive examples to mark in X2.

2.3.3. We then calculate the sample weight for each example by setting the weight Cl for each labeled example in X1 and 0 for each example in X2. Cl is a hyperparameter that defines the penalty for misclassifying the labeled example.

2.3.4. We train a binary SVM classifier on X1 with the sample weights assigned in the previous step. We classify the num_plus examples with the highest value as +1, and the rest as –1.

2.3.5. We predict the labels for X2 using the trained classifier, and label the num_plus examples with the highest values of the decision function +1 and the rest with –1.

2.3.6. We set the initial weight of each example in X2 to C_minus if it is marked as –1, and C_plus if it is marked as +1. C_minus and C_plus are hyperparameters that define the penalty for misclassifying an unlabeled example.

2.3.7. We create a new data set X3, which is a concatenation of X1 and X2. We also create a new set of labels, Y3, which is a concatenation of Y1 and the predicted labels for X2.

2.3.8. We enter a cycle that continues until C_minus and C_plus reach the specified maximum value (Cu).

2.3.9. We train a binary SVM classifier on X3 using the sample weights assigned in the previous step.

2.3.10. We calculate slack variables that represent the degree to which each example violates the classification boundary.

2.3.11. We then calculate the epsilon slack for the labeled examples, which is the sag variable for the labeled examples.

2.3.12. We check whether there are any unobserved examples with epsilon-slack greater than zero, which indicates the condition when an example is misclassified.

2.3.13. If such an example exists, we enter another loop where we identify the positive and negative set of examples with the highest epsilon-slack values.

2.3.14. We change the labels for these two examples and update the sample weights for each example.

2.3.15. We then retrain the binary SVM classifier on X3 with the updated sampling weights.

2.3.16. We repeat steps 11-16 until there are no unlabeled examples with epsilon slack greater than zero.

2.3.17. Then we increase C_minus and C_plus by a factor of 2 or until they reach their maximum value, *Cu*.

2.3.18. We update the sampling weights for each example and repeat steps 10-18 until C_minus and C_plus reach *Cu*.

2.3.19. Finally, we return the predicted labels for X2 as the result of the TSVM algorithm.

2.4. Determine the acceptable range of hyperparameter values for optimization and use Bayesian optimization (using the Optuna library) to find optimal hyperparameter values based on quality metrics.

2.4.1. Iterating over the allowable range of hyperparameter values.

2.4.2. Training a TSVM model based on previous steps 2.3.1–2.3.19 and a combination of hyperparameters from the current iteration.

2.4.3. Using the trained model, make a forecast on the control sample.

2.4.4. Evaluate the quality of the forecast using the F1 metric.

2.5. Select the best set of hyperparameters by selecting the best estimate of the prediction.

3. Create a combined forecast by training multiple TSVM models.

3.1. Take as a basis the best hyperparameters from the results of step 2.4 and train separate TSVM models for different kernels (linear, radial, polynomial or sigmoid).

3.2. After training each TSVM model, predictions are made on unlabeled data and a decision function is computed, which returns the distance to the separating hyperplane.

3.3. The resulting decision functions are combined using median voting. Each prediction of the TSVM model counts as a vote, and the final decision is made based on the median of these votes. If the median is greater than or equal to 0, then it is classified as a positive class, otherwise it is classified as a negative class.

4. After performing training, optimization and combining the predictions, a final prediction is obtained, which can be compared with the actual values.

5. Evaluate the quality of the forecast using the F1 metric.

## V. RESULTS

After voting, a final prediction is obtained, which can be compared with the actual values. The obtained results are shown in the Tables 1–3.

TABLE I. AVERAGE BY DATASET TYPE

| | Accuracy | | | Recall | | | Precision | | | f1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM |
| **type_dataset** | | | | | | | | | | | | |
| **banana** | 0.518 | 0.497 | 0.493 | 0.643 | 0.618 | 0.753 | 0.505 | 0.494 | 0.489 | 0.549 | 0.539 | 0.555 |
| **banana_inverse** | 0.541 | 0.635 | 0.554 | 0.719 | 0.814 | 0.738 | 0.527 | 0.632 | 0.547 | 0.598 | 0.689 | 0.610 |
| **c_circles** | 0.501 | 0.618 | 0.504 | 0.507 | 0.610 | 0.563 | 0.348 | 0.422 | 0.337 | 0.411 | 0.499 | 0.418 |
| **two_moons_classic** | 0.828 | 0.888 | 0.760 | 0.838 | 0.918 | 0.807 | 0.823 | 0.867 | 0.739 | 0.830 | 0.891 | 0.769 |
| **two_moons_tight** | 0.632 | 0.781 | 0.600 | 0.563 | 0.740 | 0.523 | 0.669 | 0.814 | 0.407 | 0.595 | 0.763 | 0.454 |
| **two_moons_wide** | 0.942 | 0.928 | 0.906 | 0.917 | 0.870 | 0.966 | 0.969 | 0.987 | 0.864 | 0.937 | 0.913 | 0.912 |

TABLE II. AVERAGE BY PERCENTAGE OF MARKING

| | Accuracy | | | Recall | | | Precision | | | f1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM |
| **perc_labeled** | | | | | | | | | | | | |
| **1** | 0.637 | 0.640 | 0.618 | 0.644 | 0.642 | 0.635 | 0.597 | 0.615 | 0.443 | 0.587 | 0.593 | 0.510 |
| **10** | 0.696 | 0.774 | 0.665 | 0.785 | 0.836 | 0.897 | 0.677 | 0.754 | 0.636 | 0.722 | 0.787 | 0.730 |
| **50** | 0.649 | 0.760 | 0.625 | 0.664 | 0.807 | 0.643 | 0.646 | 0.739 | 0.612 | 0.651 | 0.768 | 0.619 |

TABLE III. GENERAL RESULTS

| | | Accuracy | | | Recall | | | Precision | | | f1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM | TSVM | Voting TSVM | Base SVM |
| **type_dataset** | **perc_labeled** | | | | | | | | | | | | |
| **banana** | **1** | 0.577 | 0.512 | 0.498 | 0.966 | 0.852 | 1.000 | 0.542 | 0.506 | 0.498 | 0.694 | 0.635 | 0.665 |
| | **10** | 0.500 | 0.501 | 0.499 | 0.612 | 0.592 | 1.000 | 0.499 | 0.500 | 0.499 | 0.550 | 0.542 | 0.666 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **50** | 0.478 | 0.478 | 0.482 | 0.351 | 0.410 | 0.259 | 0.473 | 0.477 | 0.471 | 0.403 | 0.441 | 0.334 |
| **banana_inverse** | **1** | 0.549 | 0.496 | 0.496 | 0.974 | 1.000 | 1.000 | 0.525 | 0.496 | 0.496 | 0.682 | 0.663 | 0.663 |
| | **10** | 0.594 | 0.712 | 0.588 | 0.604 | 0.632 | 0.524 | 0.597 | 0.757 | 0.606 | 0.600 | 0.689 | 0.562 |
| | **50** | 0.480 | 0.698 | 0.578 | 0.579 | 0.809 | 0.689 | 0.458 | 0.642 | 0.540 | 0.511 | 0.716 | 0.606 |
| **c_circles** | **1** | 0.416 | 0.467 | 0.497 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | **10** | 0.633 | 0.757 | 0.494 | 0.946 | 1.000 | 1.000 | 0.579 | 0.670 | 0.494 | 0.718 | 0.803 | 0.662 |
| | **50** | 0.454 | 0.632 | 0.522 | 0.574 | 0.829 | 0.689 | 0.465 | 0.596 | 0.518 | 0.513 | 0.693 | 0.591 |
| **two_moons_classic** | **1** | 0.780 | 0.832 | 0.785 | 0.792 | 0.848 | 0.836 | 0.773 | 0.822 | 0.758 | 0.782 | 0.835 | 0.795 |
| | **10** | 0.843 | 0.907 | 0.776 | 0.866 | 0.960 | 0.886 | 0.827 | 0.867 | 0.724 | 0.846 | 0.911 | 0.797 |
| | **50** | 0.862 | 0.926 | 0.720 | 0.857 | 0.944 | 0.698 | 0.867 | 0.912 | 0.733 | 0.862 | 0.928 | 0.715 |
| **two_moons_tight** | **1** | 0.623 | 0.725 | 0.498 | 0.376 | 0.535 | 0.000 | 0.748 | 0.866 | 0.000 | 0.501 | 0.662 | 0.000 |
| | **10** | 0.636 | 0.789 | 0.726 | 0.683 | 0.830 | 0.975 | 0.622 | 0.765 | 0.649 | 0.651 | 0.797 | 0.780 |
| | **50** | 0.638 | 0.828 | 0.576 | 0.631 | 0.855 | 0.594 | 0.638 | 0.810 | 0.571 | 0.634 | 0.832 | 0.583 |
| **two_moons_wide** | **1** | 0.877 | 0.806 | 0.935 | 0.759 | 0.614 | 0.972 | 0.995 | 1.000 | 0.906 | 0.861 | 0.761 | 0.938 |
| | **10** | 0.968 | 0.981 | 0.908 | 1.000 | 1.000 | 0.998 | 0.939 | 0.963 | 0.845 | 0.969 | 0.981 | 0.915 |
| | **50** | 0.982 | 0.996 | 0.874 | 0.992 | 0.996 | 0.929 | 0.973 | 0.996 | 0.840 | 0.982 | 0.996 | 0.882 |

## VI. CONCLUSIONS

A new algorithm of the support vector machine using semi-supervised learning was developed. The high efficiency of the algorithm is ensured by the use of the voting method, which makes it possible to aggregate the results of each individual classifier and determine the major prediction based on the largest majority of votes. As several different algorithms in nature, several different kernel functions (eg, rbf, linear, sigmoidal, polynomial) can be used to train different SVMs. Optuna is a machine learning hyperparameter optimization framework that provides a wide range of algorithms to find the best hyperparameters.

## REFERENCES


[1] C. M. Bishop, *Pattern recognition and machine learning*. 2006. Berlin: Springer.

[2] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000, Cambridge. https://doi.org/10.1017/CBO9780511801389

[3] A. Gammerman, V. Vapnik, and V. Vovk, "Learning by transduction," *In Uncertainty in Artificial Intelligence*, pp. 148–155, 1998.

[4] V. N. Vapnik, Statistical learning theory. New York: John Wiley & Sons, Inc.

[5] K. P. Bennett, A. Demiriz, and J. Shawe–Taylor, "A Column Generation Algorithm for Boosting," (http://www.recognition.mccme.ru/pub/papers/boosting/bennett00column.pdf). *In Pat Langley, editor, Proceedings of Seventeenth International Conference on Machine Learning*, pp. 65–72. Morgan Kaufmann, 2000.

[6] T. Joachims, "Transductive inference for text classification using support vector machines," *In ICML*, 1999.

[7] O. Chapelle, and A. Zien, "Semi-supervised learning by low density separation," *In AISTATS*, pp. 57–64, 2005. https://doi.org/10.7551/mitpress/9780262033589.001.0001

[8] F. Wang, & C. Zhang, "Label propagation through linear neighborhoods," *IEEE Transactions on Knowledge and Data Engineering*, 20(1), 2008, pp. 55–67. https://doi.org/10.1109/TKDE.2007.190672

[9] N. Kasabov and S. Pang, "Transductive support vector machines and applications in bioinformatics for promoter recognition," *In ICNNSP*, 2004, pp. 1–6. https://doi.org/10.1109/ICNNSP.2003.1279199

[10] C. Goutte, H. Derjean, E. Gaussier, N. Cancedda, and J.M. Renders, "Combining labelled and unlabelled data: A case study on fisher kernels and transductive inference for biological entity recognition," *In*

*CoNLL*, 2002, pp. 1–7. https://doi.org/10.3115/1118853.1118864

[11] T. Zhang and F. J. Oles, "A probability analysis on the value of unlabeled data for classification problems," *In ICML 00*, pp. 1191–1198, 2000.

[12] O. Chapelle, B. Schölkopf, and A. Zien, (eds.). *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006b.

[13] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *J. Mach. Learn. Res*., 9: 203–233, 2008.

[14] O. Chapelle, M. Chi, & A. Zien, "A continuation method for semi-supervised SVMsm," *In Proceedings of the 23rd international conference on machine learning*, 2006a, pp. 185–192. https://doi.org/10.1145/1143844.1143868

[15] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, Masanori Koyama, *Optuna: A Next-generation Hyperparameter Optimization Framework*, 2019, https://doi.org/10.48550/arXiv.1907.10902

[16] Victor Chukwudi Osamor & Adaugo Fiona Okezie, "Enhancing the weighted voting ensemble algorithm for tuberculosis predictive diagnosis," *Scientific Reports*, vol. 11, Article number: 14806, 2021, 4922, Accesses 19. https://doi.org/10.1038/s41598-021-94347-6

Received February 19, 2023

**Sineglazov Victor.** ORCID 0000-0002-3297-9060.
Doctor of Engineering Science. Professor. Head of the Department.
Aviation Computer-Integrated Complexes Department, Faculty of Air Navigation Electronics and Telecommunications, National Aviation University, Kyiv, Ukraine.
Education: Kyiv Polytechnic Institute, Kyiv, Ukraine, (1973).
Research area: Air Navigation, Air Traffic Control, Identification of Complex Systems, Wind/Solar power plant, artificial intelligence.
Publications: more than 700 papers.
E-mail: svm@nau.edu.ua

**Samoshyn Andrii.** Bachelor.
Educational and Scientific Institute of Applied System Analysis, National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," Kyiv, Ukraine.
Education: National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute," (2022).
Research interests: artificial neural networks, programming.
E-mail: samoshyn.andriy@lll.kpi.ua

**В. М. Синєглазов, А. О. Самошин. Напівкерована машина опорних векторів**
У статті розглянуто новий підхід побудови машини опорних векторів із напівкерованим навчанням для вирішення задачі багатокласової класифікації. Передбачається, що розподіли умовних класів можуть перекриватися. Зроблено модифікацію функції вартості за рахунок додавання до неї елементів штрафу за влучення міток не до свого класу. Штраф подається у вигляді лінійної функції відстані між міткою та межею класу. Для подолання проблеми багатокритеріальності запропоновано метод глобальної оптимізації, відомий як continuation. Для комбінації передбачень пропонується використати метод голосування моделей з різними ядрами. За інструмент для налаштування гіперпараметрів був обраний фреймворк Optuna. В якості навчальних вибірок було розглянуто наступні: type_dataset, banana, banana_inverse, c_circles, two_moons_classic, two_moons_tight, two_moons_wide.
**Ключові слова:** машина опорних векторів; напівкероване навчання; багатокласова класифікація; багатокритеріальність; метод глобальної оптимізації.

**Синєглазов Віктор Михайлович**. ORCID 0000-0002-3297-9060.
Доктор технічних наук. Професор. Завідувач кафедрою.
Кафедра авіаційних комп'ютерно-інтегрованих комплексів, Факультет аеронавігації електроніки і телекомунікацій, Національний авіаційний університет, Київ, Україна.
Освіта: Київський політехнічний інститут, Київ, Україна, (1973).
Напрям наукової діяльності: аеронавігація, управління повітряним рухом, ідентифікація складних систем, вітроенергетичні установки, штучний інтелект.
Кількість публікацій: більше 700 наукових робіт.
E-mail: svm@nau.edu.ua

**Самошин Андрій Олександрович.** Бакалавр.
Навчально-науковий інститут прикладного системного аналізу, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.
Освіта: Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна, (2022).
Напрям наукової діяльності: штучні нейронні мережі, програмування.
E-mail: samoshyn.andriy@lll.kpi.ua