Research Article

Improve the Efficiency of Support Vector Machine Classifier with Fractional Gradient Descent

Dian Hapsari[1] , Aeri Rachmad[2], Imam Utoyo[3*], Santi Wulan Purnami[4]

[1] Department of Informatics, Faculty of Electronic Engineering and Information Technology, Institute of Technology Adhi Tama, Indonesia
[2] Department of Informatics, Faculty of Engineering, University of Trunojoyo Madura, Indonesia
[3*]Department of Mathematic, Faculty of Science and Technology, Airlangga University, Indonesia
[4]Department of Mathematic, Faculty of Science and Technology, Institute of Technology Sepuluh Nopember, Indonesia

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

Abstract: This study aims to improve the efficiency of the SVM classifier using the fractional descent gradient algorithm, so that the speed during the data training process increases. There are seven numerical datasets (iris, abalone, wine quality, pima Indian diabetes, ionosphere, wheat seeds, sonar) used to test the performance of the SVM classifier with fractional gradient descent which has been optimized with fractional order Caputo Type. The seven datasets have a binary class and a balanced class for the number of labels from one class to another. The problem of minimizing the objective function of the SVM classifier is the problem of infinite minimization which is a convex function that can be solved with Fractional Gradient Descent. Fractional order makes it easier to determine the amount of learning rate used to update weights so as to increase computation time. In this study SVM classifier optimization using a fractional order derivative of the Caputo type, so that the convergence speed is much faster than using a sequence of integers. The test results show that the SVM Classifier with fractional gradient descent, it reaches a convergence point of approximately 60% smaller than traditional SVM gradient descent. SVM classifier with Fractional Gradient Descent has the highest convergence speed in the seven datasets, with a small learning speed in the process of reaching the point of convergence of computing time increases. For the future study we want to use fractional gradient descent SVM for unbalanced class case and sparse dataset that many of the values are zero for text classification.

Keywords: Machine Learning**,** Support Vector Machine, Classification, Fractional Gradient Descent.

1. Introduction

The Support Vector Machine (SVM) linear classifier is a linear prediction method that has several advantages, namely first, SVM has regularization capabilities, so that SVM has good generalizability which prevents it from over-fitting problems [R. Fernandes de Mello. et. al., 2018]. The problem of over-fitting can be easily controlled by selecting an appropriate margin for separating the data. Second, SVM is able to efficiently handle non-linear data patterns using a kernel trick [S. Guo. et. al., 2017]. For sample data that cannot be resolved in linear space, SVM can transform into non-linear space using a kernel trick in a high-dimensional space, without increasing computational complexity [J. Tao. *et al.*, 2019]. Third, SVM is able to solve classification and regression problems, Support Vector Classification (SVC) is used for classification problems, while Support Vector Regression (SVR) is used for regression problems. Fourth, SVM has stability: small changes in data do not really affect the hyperplane and hence SVM becomes a stable classifier model.

SVM as a linear classification algorithm has constraints on the slow convergence and the small amount of data on large-scale data training activities [T. Watanabe and H. Iima., 2019] [J. Wang. et. al., 2017]. SVM also has the disadvantage of the difficulty of determining optimal parameters. So that the SVM linear classifier optimization problem uses convex optimization which is closely related to the large-scale data training process. In the large-scale data training process, it usually involves matrix operations that require a lot of time. This is related to the size of the matrix which keeps increasing or becomes impossible due to memory limitations [A. Cutkosky and R. Busa-Fekete, 2018]. Large-scale data training on the SVM linear classifier is a convex optimization problem that scales with the training set size rather than the feature space dimension. In the case of large-scale data training, it can cause data training activities to be impractical [S. Vakili. et. al.,2019].

The completion of the optimization of the SVM objective function using Quadratic Programming requires computation time or worse-case time O (N3) where N is the number of variables or constraints of the data. The solution to this problem, John Platt, in 1998 introduced SMO. SVM linear classifier aims to find the maximum margin with the objective function being primal or dual. SMO only optimizes two variables at once rather than trying to optimize all n variables, but it is still considered slow for large-scale data training processes [J. Shawe-

Taylor and S. Sun, 2011]. Optimization algorithms for machine learning are procedures that are carried out iteratively by comparing various solutions until an optimal solution is found. The optimization algorithm helps to minimize or maximize the objective function with respect to the internal parameters of the model which maps a set of predictor variables to target values [J. K. Liu and S. J. Li, 2014].

Classifiers such as SVM for the data training process require an unconstrained optimization method. Based on the research that has been done, gradient-based unconstrained optimization method is very effective in terms of the speed of computation time for large-scale data sets [J. Liu and X. Wu, 2014]. There are several studies that have been carried out to optimize the gradient-based SVM model, including optimization with the Stochastic Gradient Descent approach [J. Konečný. et. al.,2016]. Furthermore, the optimization method for the gradient-based SVM model includes sub-gradient descent and coordinate descent [A. S. Bedi and K. Rajawat, 2018]. Both techniques have been shown to offer significant advantages over traditional approaches when dealing with large data sets. In a seamless optimization apart from stochastic gradient descent, there is an optimization method based on fractional gradient descent [F. Oberdorf. et. al.,2019]. Fractional gradient descent is a seamless optimization method that has been proven to be effective in solving large-scale training data optimization problems in linear classification algorithms such as neural networks [J. Wang. et. al.,2017]. In the fractional gradient descent optimization method, it involves mathematical theory, namely fractional calculus theory. Fractional Calculus is believed to be a good tool to improve the performance of traditional gradient method. Fractional decrease in the gradient method has been successfully used for convex problems, but related research is still in its early stages and needs to be investigated further. There are obstacles in its implementation, namely the convergence problem that cannot meet the requirements and convergence speed which must also be taken into account [S. Ruder.,2016].

2. Material and Method

2.1 Material

Generalization in machine learning, the classification algorithm uses training data to build several prediction models, where the training data consists of pairs of attributes in one data record along with several targets. The purpose of building a prediction model is that in the future there will be new data and unknown target values that are expected by the classification algorithm with the resulting model can predict the target label on the new data. The objectives of this study include how to construct and implement a Caputo type fractional gradient descent algorithm to optimize the SVM classifier in order to increase the speed during the data training process with a number of lines of approximately five thousand lines and a number of features or features of approximately 15 features, and simulate using secondary data.

The problem limitation of this research is to use a training data optimization method based on fractional gradient descent with a fractional-order derivative of the Caputo type. In this study, using a data group with two classes (binary class) and the number of class balanced labels (class balanced). The data group used for simulating the performance of the SVM classifier with fractional gradient descent has a variation in the number of attributes with the number of data lines ranging from 150 to approximately 5000 rows of data with features between 8 and 15. The data group used in the simulation can be accessed from the UCI engine web page. study. There are seven data groups used, among others the iris flower, abalone, wine quality, pima Indians diabetes, ionosphere, wheat seeds, and sonar data group.

In this study using a dataset from the UC Machine Learning Repository (URL:
https://archive.ics.uci.edu/ml/index.php). UC machine learning is a collection of databases, as well as data generators, used by the machine learning community for empirical analysis of machine learning algorithms. There are seven datasets used to test the performance of the SVM classifier which has been optimized with fractional gradient descent. The seven datasets have a binary class and a balanced class for the number of labels from one class to another. Table 2 below shows the number of records and the number of attributes of each of the datasets used.

No.	Dataset Name	Number of Instance	Number of Attribute
1	Abalone	4.177	8
2	Wine Quality	1.599	12
3	Pima Indians Diabetes	768	8
4	Ionosphere	351	34
5	Wheat Seeds	210	7

6	Sonar	207	60
7	Iris Flower	150	4

Tabel 2. Datasets for Fractional Gradient Descent SVM

For each dataset used only use negative or positive class labels as in the Pima Indians Diabetes dataset. For other datasets the class labels are only 0 and 1, from table 2 it can be seen that the Abalone dataset has the largest number of records. The Abalone dataset with the largest number of records will form a large matrix, as well as the Sonar dataset which has the largest number of attributes 60 among other datasets. In this experiment we will see the performance of the SVM classifier with fractional gradient descent in these datasets.

2.2 Optimization with Fractional Gradient Descent

The ability of the classification algorithm to predict the target value of new data is closely related to the problem of Under-fitting and over-fitting. What is meant by the problem of over-fitting is when learning conducted by the classification algorithm produces a prediction model that is only suitable for training data only []. This can occur when the prediction model is complex and flexible only for the noise contained in the training data. Problems arise due to noise in training data not necessarily the same as new data. The formal definition of the problem of over-fitting is a prediction model in the form of a function that is over-fit for a group of data if another f prime function is sought where the function makes more errors or errors in the training data E_{train} (F`) > E_{train} (F) but fewer mistakes are made for new data.

In contrast to the under-fitting problem is when a prediction model in the form of a function that is applied to training data is too general or simple to recognize patterns in a group of data. The formal definition of an under-fitting problem is a prediction model in the form of functions that are over-fit for a group of data if another f prime function is sought where the function is smaller in error value for training data and new data.

Each group of data needs different levels to understand the flexibility of the prediction model. Each classification algorithm has a way to adjust the flexibility of the prediction model. The regression classification algorithm uses polynomial derivatives, the decision tree classification algorithm uses pruning confidence, the kNN classification algorithm uses the closest neighbor value, and the SVM classification algorithm uses kernels and cost parameters.

Actual error training error that's the error that you get on a training set on the general form for the training set the general form for the training error is sum up over all training examples the individual errors that you get for each example x_i.
Training error:

$$E_{train} = \frac{1}{n}\sum_{i=1}^{n} error(f_D(x_i), y_i) \quad (2)$$

The error xi is the input that we get fD of xi that's our prediction that's what our classifier system predicts as the output for x_i, y_i is the true value that should have gone to xi. The error function is some measure of this error inside is some measure that tells you how close was the predicted value to the true value. The simplest case when in classification it usually just measures were they the same or were they not the same did we predict the right class. Training error and the sum is over the training set. The generalization error the generalization that is how well we do on the data that you don't have today on the data that's going to come tomorrow. The problem with that is we don't know what tomorrow's data would look like. We don't know what future data xi will be, we don't know what labels y_i it will have, otherwise if we did there wouldn't be much need in predicting those values. We will be building a classifier in the first place but you can still come up with an equation for the generalization error because we know the range of all possible {x,y}.

We can write on an equation for a generalization error:

$$E_{gen} = \int error\,(f_D(x), y)p(x, y)dx \quad (3)$$

we going to integral and that integral moves over all possible x's and all possible y's. Why it's integral and not a sum when we have real-valued data we need to integrate over all possible values for every possible attribute. That integral captures every possible value of every attribute over all the possible classes that we are trying to predict that's such a big space.

A particular label y together in the future. That's a probability distribution for the future data coming down the line. Same way as we had it before right so we have the y the x our function f the predictor would compute some value for the x and then we see that the same as the true label. As shown in Figure 1 the generalization error process

is carried out in the training data after the hyperparameter optimization process for the model generated by the classifier algorithm.

The SVM classifier which aims to estimate the value of w and b uses an algorithm or solver to solve optimization problems. We can use a quadratic solver with the first step of minimizing quadratic functions which are subject to linear constraints. The problem is that the solver is not efficient for large scale data, so we use an alternative approach. We can minimize the convex function.

$$f(w,b) = \frac{1}{2}\sum_{j=1}^{d}\left(w^{(j)}\right)^2 + C\sum_{i=1}^{n}\max\{0,1 - y_i\,(w \cdot x_i + b)\} \qquad (4)$$

Using gradient descent on the convex function is very simple, half ½ the sum of all the coordinates over all dimensions squared of the value w. The first part is margin maximization and the second part is the empirical loss plus the penalty slack C times the sum of all our training examples 1 to N then the slack penalty value = 0 if we classify correctly or the penalty slack value = -1 our distance from the limit. To optimize the function, it can be done by calculating the gradient (derivative). The function we are going to calculate the gradient for is basically a convex function so we can calculate it with gradient descent. If we start from a certain point then we calculate the derivative / gradient of the function with that particular point / point data so that we can make small steps leading to the gradient of the function to the desired minimum point. To count $\nabla(j)\ w \cdot r \cdot t \cdot w^{(j)}$

$$\nabla(j) = \frac{\partial f(w,b)}{\partial w^{(j)}} = w^{(j)} + C\sum_{i=1}^{n}\frac{\partial L(x_i,y_i)}{\partial w^{(j)}} \qquad (5)$$

The way to calculate the derivative of the empirical gradient is if our classification is correct which means that it is correct to classify the training data then the value of the derivative is 0 and if we misclassify then this is just the j coordinate of the i-th training data times the class of that training data. When you have obtained the gradient from the objective function, you can use the gradient descent (Batch) method to estimate w.

For the pseudocode Batch gradient descent as follows;

The iteration will repeat until the convergence condition:

For j = 1 … d

Evaluation: $\nabla(j) = \frac{\partial f(w,b)}{\partial w^{(j)}} = w^{(j)} + C\sum_{i=1}^{n}\frac{\partial L(x_i,y_i)}{\partial w^{(j)}}$

Update: $w^{(j)} \leftarrow w^{(j)} - \eta\nabla(j)$ where η learning rate parameter

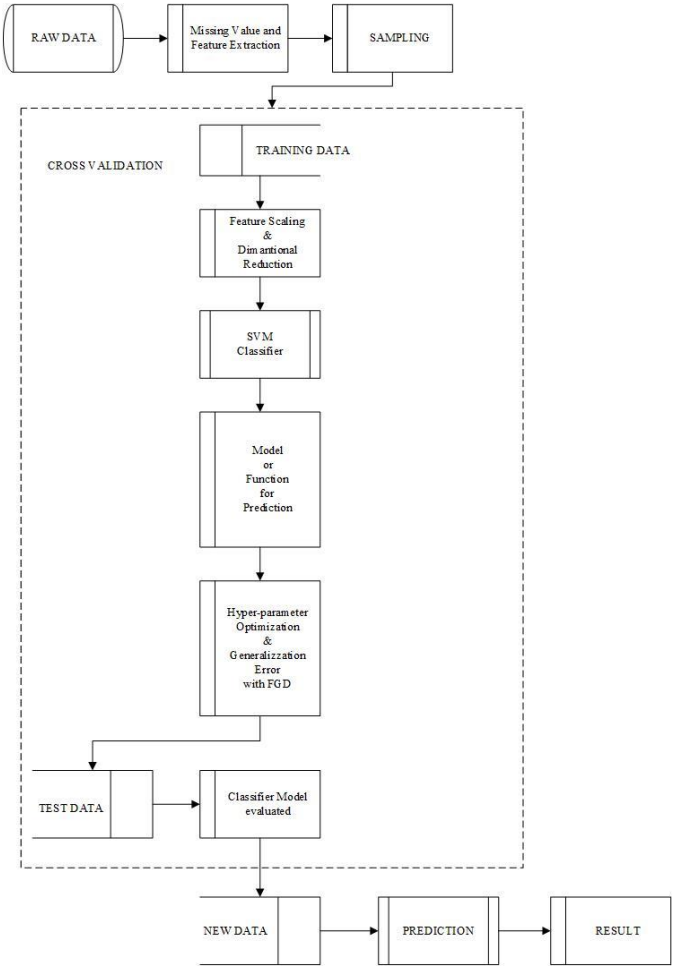

Figure 1. Supervised Learning processes.

The problem arises that calculating ∇ (j) will take O (n). The computation time will increase as the size of the training data increases. It can be concluded that using Batch gradient descent is not suitable for use in the case of big data. Now that we have the gradient to use for the gradient descent method, we will start at some location x and evaluate the gradient for the given dimension so we will repeat all the dimensions we will evaluate the gradient on the given dimension by simply going through all the data to calculate the number of previous interests. Then we update the vector coordinates in reverse to the gradient. We have a parameter called the Learning Rate parameter which basically tells us how big of a move we are going to make. However, there is a problem that the gradient descent computation time is linear with the amount of training data [Y. Wei. et. al.,2017].

With Stochastic, it is expected to accelerate the running of the algorithm. The idea is that we will only evaluate the gradient on each individual example. For coordinates or dimensions j and training data *i* only *w j* plus C times the gradient for the training data given *i*. So there are 4 loops, the first loop is for all training data, the second loop is for all dimensional coordinates and the third loop evaluates the coordinates (*j, i*) for the processed data points, then the last loop is the direction of the gradient from the individual data points. For improve the efficient of SVM classifier we use fractional calculus, However, when the first order generalized gradient direction by a first-order gradient fractions, the corresponding algorithm will converge to the extreme point fractions of the target function are not the same as the actual extreme point [Y. Chen. et. al.,2018]. This weakness is critically impeding the application of this method. To solve the problem of convergence, this paper analyzes the specific reasons and proposed three possible solutions. Taking into consideration the characteristics of the long memory of fractional derivatives, short-memory principle is the previous selection. In addition to cutting the length of memory, two new methods were developed to achieve convergence.

In this research, we will use Caputo's fractional order derivative for the optimization of the gradient-based SVM classifier algorithm. The Caputo derivative has the advantage that it allows traditional initial and boundary conditions to be included in the problem formulation and its derivative for constants is zero [M. Caputo and M. Fabrizio.,2015]. Caputo derivatives are especially useful when dealing with real-world problems because, they allow

traditional initial and boundary conditions to be included in the problem formulation and in addition the constant derivative is zero. Caputo's fractional-order of order α is defined as follows;

$$ {}_{a}^{Caputo}D_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-\tau)^{n-\alpha-1} f^{(n)}(\tau) d\tau, \quad (6) $$

where ${}_{a}^{Caputo}D_t^{\alpha}$ is a Caputo derivative operator, α is a fractional order derivative. Specifically, when $\alpha \in (0,1)$, the expression for the Caputo derivative is as follows;

$$ {}_{a}^{Caputo}D_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-\tau)^{-\alpha} f'(\tau) d\tau . \quad (7) $$

Since the initial values between fractional differential equations with Caputo derivatives and integer differential equations are the same: these derivatives have a wide variety of applications in physical processes and engineering problems.

3.Result and Discussion

In this section we carry out tests that demonstrate the effectiveness of an SVM classifier with a fractional gradient descent. The test conducted aims to evaluate the accuracy and error rate of the SVM classifier with fractional gradient descent. With the seven datasets used we use a Core i5-7200U 2.50GHz processor and 8GB RAM memory and the SVM classifier uses a linear kernel. Fractional order makes it easier to determine the amount of learning rate used to update weights so as to increase computation time. In this study SVM classifier optimization using a fractional order derivative of the Caputo type, so that the convergence speed is much faster than using an integer order.

Dataset Name	Accuration	Error for 1000 iteration			
		α= 0.00001	α= 0.0001	α= 0.001	α= 0.01
Abalone	0,49132	9.69417e-06	1.9326e-21	7.49966e-41	8.64068e-61
Wine Quality	0,52272	0,33994	0,336	0,336	0,336
Pima Diabetes	0,34375	0,34267	0,248	0,248	0,248
Ionosphare	0,61363	1,09394	0,059	0,059	0,059
Seeds	0,48571	2,80764	0,034	0,034	0,034
Sonar	0,47115	2,00595	0,056	0,056	0,056
Iris	0,40000	2,70962	0,022	0,020	0,020

Table 1. Simulation result SVM classifier with Fractional Gradient Descent

In table 1, it can be seen that the SVM classifier accuracy value for seven datasets, clearly seen in Figure 2, the highest accuracy value is the Ionosphere dataset with 351 instances and 34 features. For the lowest accuracy value is the Pima Indian Diabetes dataset with 768 instances and eight the number of features. For the accuracy of the SVM classifier, it will depend on the kernel used. In this study, the kernel that we used is a polynomial kernel which has a weakness, a polynomial kernel not suitable for large orders of power.

In an experiment conducted to evaluate the error rate of the SVM classifier with fractional gradient descent, it uses four different alpha values. The alpha value shows the learning rate which is a tuning parameter in an optimization algorithm like fractional gradient descent that determines the step size at each iteration while moving toward a minimum of a loss function. With four different learning rate values (0.01; 0.001; 0.0001; 0.00001) will show that fractional gradient descent has the ability to process training data and makes it easier to determine the amount of learning rate used to update weights so as to increase computation time, as seen in Figure 3.

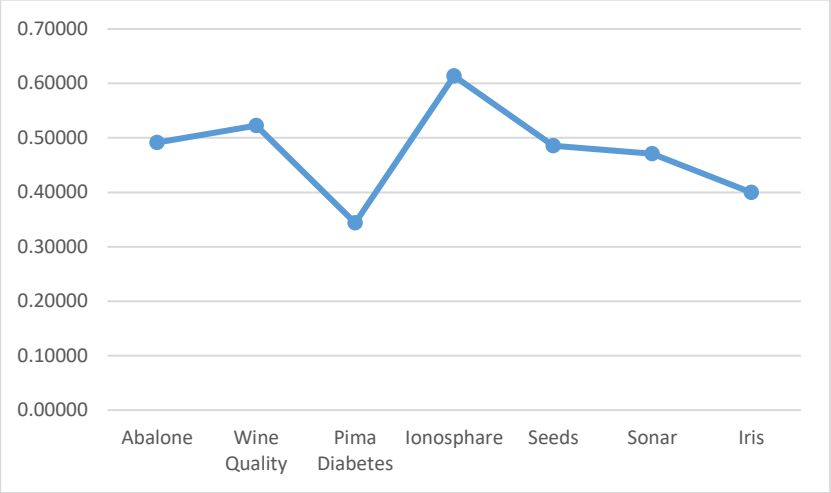

Figure 2. The Accuracy of SVM Fractional Gradient Descent

From the Figure 3, it is clear that with a small learning rate of 0.00001 the SVM classifier with a Fractional gradient descent moving toward a minimum of a loss function for each dataset. It can be seen in the table 1, for the SVM classifier with fractional gradient descent which has the highest number of instances of the seven datasets used, the error value NaN (Not a Number) is the value of a numeric data type that is not represented in floating point. With the fractional gradient descent SVM the Abalone dataset has an error value of = 9,69417e-06 by setting a learning rate of 0.00001, on 4177 data records.

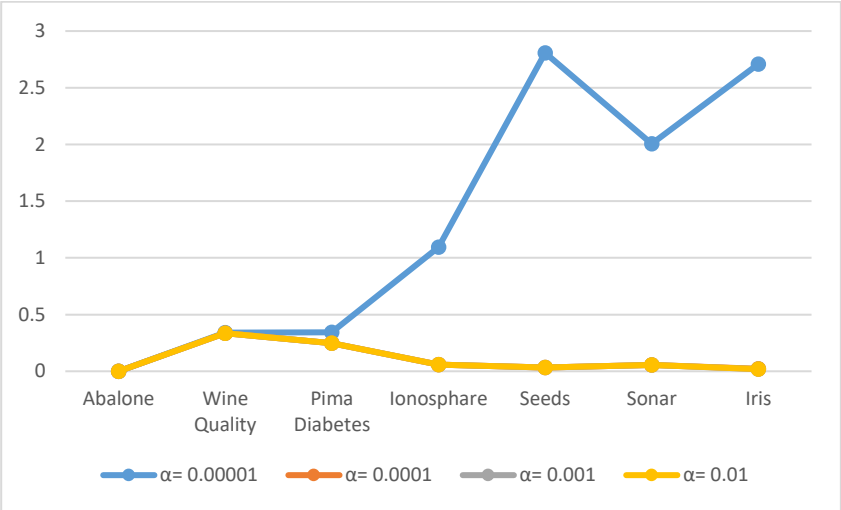

Figure 3. Error Rate for four different value of alfa

For the error rate value in table 1 on seven datasets is the error rate value in the thousandth iteration. For error rate values of seven datasets with iterations below a thousand, only the SVM classifier with fractional gradient descent was able to achieve convergence. We also conducted experiments on SVM classifiers optimized perceptron algorithm and stochastic gradient algorithm decline. As shown in Figure 3, the SVM classifier using fractional gradient descent algorithm that gives a small error rate and not much different from the optimized SVM classifier with stochastic gradient descent algorithm. Even the SVM classifier error value with the decline smaller gradient fractions.

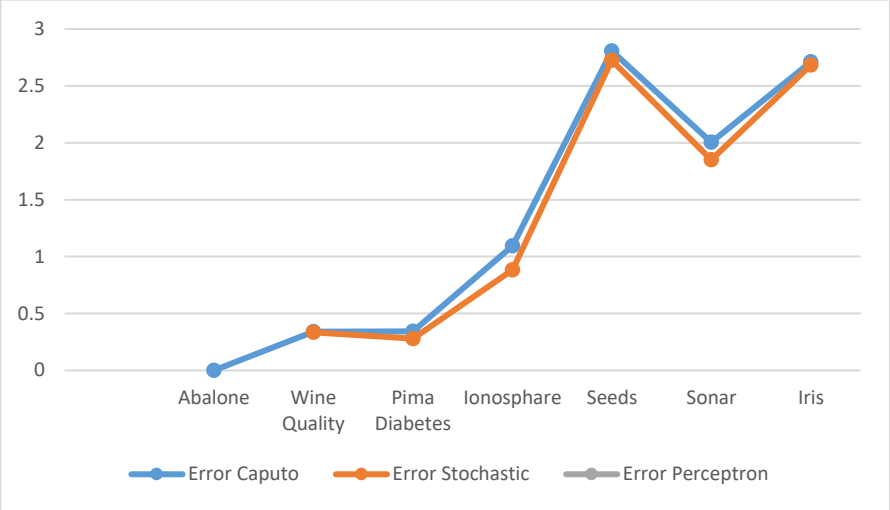

Figure 4. Error Rate for SVM classifier with optimization

We also conducted experiments on the SVM classifier which is optimized with the perceptron algorithm and the stochastic gradient descent algorithm. As shown in Figure 4, the SVM classifier with the fractional gradient descent algorithm which gives a small error rate and is not much different from the SVM classifier that is optimized with the stochastic gradient descent algorithm. Even the SVM classifier with fractional gradient descent error value is smaller. The simulation result show that the SVM Classifier with fractional gradient descent optimization, it reaches a convergence point of approximately 60% smaller than traditional SVM gradient descent.

4.Conclusion

In this study use seven datasets that have a binary class and a balanced class for the number of labels from one class to another. From seven numerical datasets (iris, abalone, wine quality, pima Indian diabetes, ionosphere, wheat seeds, sonar) used to test the performance of the SVM classifier with fractional gradient descent. The simulation result show that the SVM classifier with fractional gradient descent it reaches a convergence point of approximately 60% smaller than traditional SVM gradient descent. SVM classifier with fractional gradient descent has the highest convergence speed in the seven datasets, with a small number of learning rate in the process of reaching convergence point that increase computation time. The SVM classifier with fractional gradient descent is able to streamline computational procedures for training data. By using the Fractional type Caputo Gradient Descent optimization method, it is able to reduce the steps towards the convergent point. The desired convergence and convergence speed of the Fractional type Caputo Gradient Descent can be optimized with the smallest error value of = 9,69417e-06 from Abalone dataset by setting a learning rate of 0.00001 on 4177 data records by using the basic procedure of the fractional gradient descent with Caputo type. For the future study we want to use fractional gradient descent SVM for unbalanced class case and sparse dataset that many of the values are zero for text classification.

References

1. Cutkosky and R. Busa-Fekete, “Distributed stochastic optimization via adaptive SGD,” in Advances in Neural Information Processing Systems, 2018.
2. S. Bedi and K. Rajawat, “Wireless network optimization via stochastic sub-gradient descent: Rate analysis,” in IEEE Wireless Communications and Networking Conference, WCNC, 2018.
3. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” Data Min. Knowl. Discov., 1998.
4. F. Oberdorf, K. McFall, S. Moros, and J. Kempkes, “A gradient descent based method for maximum efficiency calculation,” in CANDO-EPE 2018 - Proceedings IEEE International Conference and Workshop in Obuda on Electrical and Power Engineering, 2019.
5. J. K. Liu and S. J. Li, “New hybrid conjugate gradient method for unconstrained optimization,” Appl. Math. Comput., 2014.
6. J. Konečný, J. Liu, P. Richtárik, and M. Takáč, “Mini-Batch Semi-Stochastic Gradient Descent in the Proximal Setting,” IEEE J. Sel. Top. Signal Process., 2016.
7. J. Liu and X. Wu, “New three-term conjugate gradient method for solving unconstrained optimization problems,” ScienceAsia, 2014.

8. J. Shawe-Taylor and S. Sun, "A review of optimization methodologies in support vector machines," Neurocomputing, 2011.
9. J. Tao, M. Dehmer, G. Xie, and Q. Zhou, "A Generalized Predictive Control-Based Path Following Method for Parafoil Systems in Wind Environments," IEEE Access, 2019.
10. J. Wang, G. Yang, B. Zhang, Z. Sun, Y. Liu, and J. Wang, "Convergence Analysis of Caputo-Type Fractional Order Complex-Valued Neural Networks," IEEE Access, 2017.
11. J. Wang, Y. Wen, Y. Gou, Z. Ye, and H. Chen, "Fractional-order gradient descent learning of BP neural networks with Caputo derivative," Neural Networks, 2017.
12. L. Bottou, "Large-scale machine learning with stochastic gradient descent," in Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers, 2010.
13. M. Caputo and M. Fabrizio, "A new definition of fractional derivative without singular kernel," Prog. Fract. Differ. Appl., 2015.
14. N. Zhang, D. Lei, and J. F. Zhao, "An Improved Adagrad Gradient Descent Optimization Algorithm," in Proceedings 2018 Chinese Automation Congress, CAC 2018, 2019.
15. R. Fernandes de Mello, M. Antonelli Ponti, R. Fernandes de Mello, and M. Antonelli Ponti, "Introduction to Support Vector Machines," in Machine Learning, 2018.
16. S. Guo, S. Chen, and Y. Li, "Face recognition based on convolutional neural network & support vector machine," in 2016 IEEE International Conference on Information and Automation, IEEE ICIA 2016, 2017.
17. S. J. Wright, "Coordinate descent algorithms," Math. Program., 2015.
18. S. Ruder, "Overview Optimization Gradients," arXiv Prepr. arXiv1609.04747, 2016.
19. S. Vakili, S. Salgia, and Q. Zhao, "Stochastic Gradient Descent on a Tree: An Adaptive and Robust Approach to Stochastic Convex Optimization," in 2019 57th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2019, 2019.
20. T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1998.
21. T. Watanabe and H. Iima, "Nonlinear Optimization Method Based on Stochastic Gradient Descent for Fast Convergence," in Proceedings - 2018 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, 2019.
22. W. Wu, J. Wang, M. Cheng, and Z. Li, "Convergence analysis of online gradient method for BP neural networks," Neural Networks, 2011.
23. Y. Chen, Y. Wei, Y. Wang, and Y. Q. Chen, "Fractional order gradient methods for a general class of convex functions," in Proceedings of the American Control Conference, 2018.
24. Y. Wei, Y. Chen, S. Cheng, and Y. Wang, "Discussion on fractional order derivatives," IFAC-PapersOnLine, 2017.
25. Y. Wei, Y. Kang, W. Yin, and Y. Wang, "Design of generalized fractional order gradient descent method," arXiv. 2018.
26. Y. Xu et al., "Large scale tissue histopathology image classification, segmentation, and visualization via deep convolutional activation features," BMC Bioinformatics, 2017.
27. Z. A. Khan, S. Zubair, H. Alquhayz, M. Azeem, and A. Ditta, "Design of Momentum Fractional Stochastic Gradient Descent for Recommender Systems," IEEE Access, 2019.