THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Empirical Evaluation of Support Vector Machine Algorithms In Machine Learning

Nitesh Gupta

Research Scholar,School of Computer Science and Engineering,

Lovely Professional University,Phagwara Punjab ,India

niteshguptabrs@gmail.com

Atul Malhotra

Assistant Professor,School Of Computer Science and Engineering,

Lovely Professional University,Phagwara, Punjab ,India

atul.18011@lpu.co.in

Amandeep Nagpal

Associate Professor,School of Computer Science and Engineering,

Lovely Professional University ,Phagwara Punjab ,India

amandeep.nagpal@lpu.co.in

Sahil Verma

Associate Professor,School of computer Science and Engineering

Lovely Professional University,Phagwara,Punjab ,India

sahil.21915@lpu.co.in

***Abstract*—**In the domain of Machine Learning and Deep Learning, classification is the widely applied task for a variety of class-based problems such as classification of image, sentiments prediction, pattern recognition, hand-digit recognition, and other class prediction based problems. In this paper, the internal working of supervised learning technique, Support Vector Machine (SVM), is discussed with its application on different types of classification problems. In addition, this paper discusses the problematic approach to use different levels of SVMs depending on the type of problem.

Keywords— soft SVM, Support Vectors, kernel function, loss function, hinge-loss, outliers.

I. INTRODUCTION

In the era of Machine Learning, linear models were mostly used for predictive analysis. The competition was to generate a linear model that is better than existing models. The key idea of the linear model is to generate a hyper-plane, which is capable of separating classes within the dataset and map the input

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

vectors to high dimensional feature space. The technical problem arises was the computational complexity of high-dimensional feature mapping by generating the hyper-planes. The proposed SVM model by Vapnik [1] was capable of producing better results than existing linear models as it uses the concept of margin for separation of the classes within the data and maps the input vectors from the data to high dimensional feature space to produce generalized results. The SVM uses the optimal hyperplanes to separate the two classes. The optimal hyperplane is nothing but the linear decision function that separates the two-class as widely as possible by using the margin concept as shown in Figure.1. It explains the maximal margin separating hyper-plane in two-dimensional space. It contains the two classes which are separated by the hyper-planes p1 and p2 with maximum separation between them. The hyper-plane p1 and p2 are on either side of the classes and pass through the support vectors.

Till now, SVM is a widely used model for supervised learning and has a wide range of applications in the field of pattern recognition, image classification, object detection, and data classification [5]. It has the ability to deal with high-dimensional feature space so, with an increase in the dimensionality of data, it gives good results as compared to other linear models. The improvement in computational field and updates in SVM has given the birth to the kernel function, which we will learn in section 5. We will derive the optimal algorithm of SVM using the kernel function.

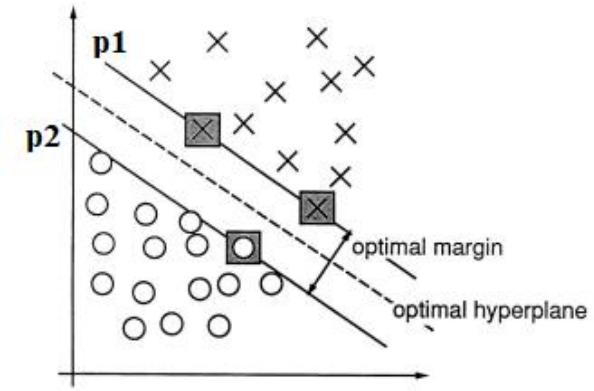
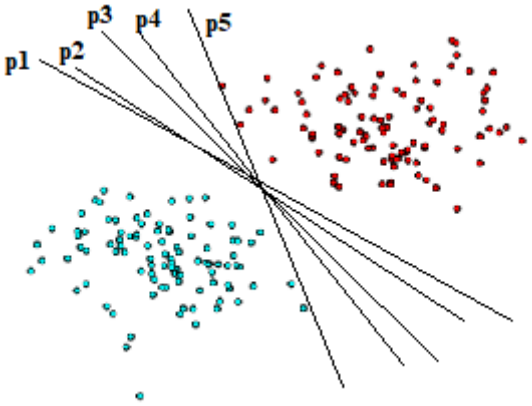


Figure.1: SVM with maximal margin separation

II. OUTLINES OF SVM

In this section, we are going to introduce the key terms related to SVM. Support Vector Machine (SVM) is the linear model used for supervised classification and regression problems. The main concept of SVM is to generate an optimal hyper-plane that has the ability to separates the classes, within the data, and generalize the model. Consider the data in two-dimensional feature space as shown in Figure.2.



Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Figure.2: Hyper-plane separating two-class data

The data in Figure.2 is two-class data which can be separated using 'n' number of hyper-planes. The hyper-planes p1, p2, p3, p4, and p5 are a subset of 'n'. The main idea of SVM is to find a hyper-plane 'p', which is able to separate the two-classes "as-widely-as possible". To achieve the optimal hyper-plane 'p', SVM uses the concept of margin. We will explain the margin concept later in this section.
Before introducing a margin, you should know about support vectors.
Consider Figure.3. Let 'p' is the optimal hyper-plane that separates the positive class from negative class in two-dimensional data space. Let 'p+' and 'p-' are two hyperplanes parallel to each other, and parallel to hyper-plane 'p' as well. The hyper-plane 'p+' and 'p-' passes through the outer-most positive and negative data-points respectively. These points through which, the hyper-plane 'p+' and 'p-' passes, are called "Support Vectors". The distance between the hyper-planes 'p+' and 'p-' is called "Margin".
The optimization problem of SVM is to maximize the margin distance so as to separate the hyper-plane 'p' from support vectors, as-widely-as possible, to provide a generalization of the model.

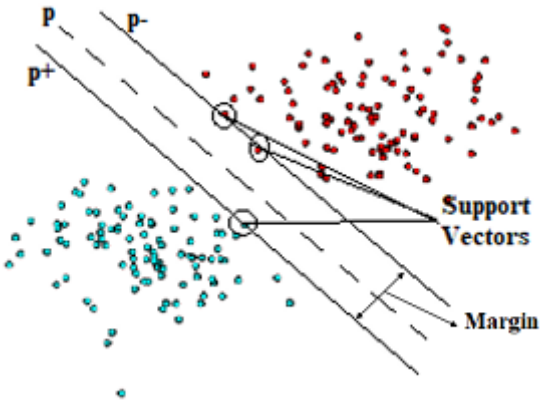


Figure.3: Optimal hyperplane with support vectors.

III. GEOMETRIC INTUITION OF SVM

Support Vector Machine (SVM) can be explained geometrically by using the convex hulls. In this section, we are going to introduce a convex-hull polygon and derive the optimal hyper-plane using the convex-hull [2].
A convex polygon is the polygons that have interior angles less than 180 degrees and all of its angle points towards outside. If we draw a line from any two points inside convex-polygon, all the portion of the line lies inside the polygon.
Concave-polygons are those polygons whose one or more interior angles are greater than 180 degrees and points towards inside. If we draw a line from two points inside the concave-polygon, the portion of the line has a chance to lie outside the polygon. Figure.4 explains the concave and convex polygon.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

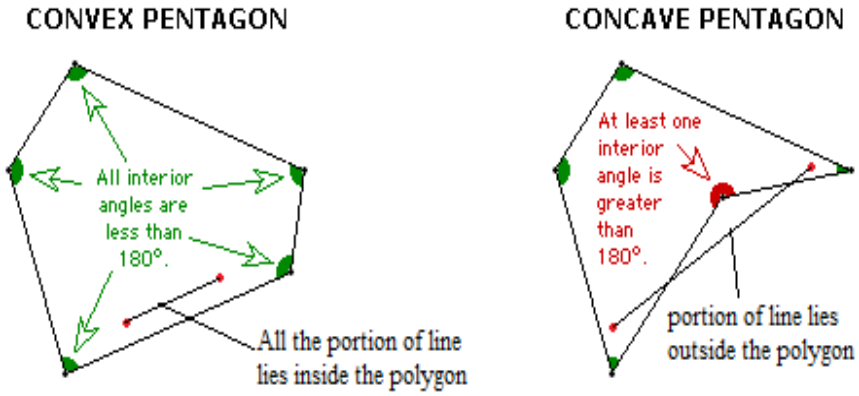


Figure.4: Convex VS Concave polygon.

A convex-hull is the smallest envelope or a convex-polygon, that contains all the set of points inside it. To derive the optimal hyper-plane, follow the steps shown below.

1. Draw the convex-hull for positive and negative points.
2. We'll get two hulls for the two-class problem.
3. Find the shortest line connecting the two convex-hulls.
4. Bisect the line.
5. The bisector plane is the optimal hyper-plane and maximizes the margin.

In Figure.5, by following the above steps, optimal hyper-plane is derived.

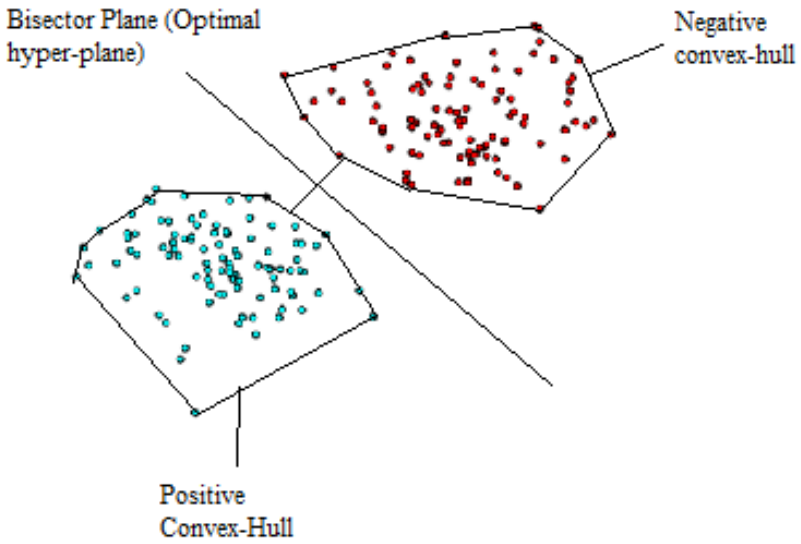


Figure.5: Derivation of Optimal Hyper-plane Geometrically

IV. MATHEMATICAL DERIVATION

From the hypothesis, we know that 'p' is the margin maximizing hyper-plane. From the equation of the plane, we have the following:

'p': $w^T x+b=0$

Let 'p+': $w^T x + b = 1$

So 'p-': $w^T x + b = - 1$

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Here 'w' and 'b' is the weight and bias vector respectively, and 'x' is the input vector.
From the distance formula between two hyper-plane, we can find out the distance ('d') between the plane 'p+' and 'p-'.
Margin-'d': distance ('p+', 'p-') = 2 / ||w||.
Where '||w||' is the L2 norm of w.
The optimization problem is to maximize the margin ('d').
The optimal 'w' and 'b' are found by maximizing the margin ('d'). Let w* and b* be the optimal weight and bias vectors.
Mathematically, (w*, b*) = argmax (2/||w||).
Let, 'y_i' be the labels corresponding to the input vectors 'x'.
So for two-class classification, for the positive label, $y_i = +1$.
And for the negative label, $y_i = -1$.
If original label y_i is positive, then $y_i(w^T x_i+b)>1$.
Similarly, for the negative label, $y_i(w^T x_i+b)>1$.
For support vectors, the value of $y_i(w^T x_i+b) = 1$.
So, the optimization problem is the following:
w*, b* = argmax (2/||w||), such that, for all 'i',
$y_i(w^T x_i+b) > 1$.
The above-mentioned optimization problem formulation is 'hard-margin SVM' [6]. The hard-margin SVM works well for linearly separable data. Figure.6 explains the hard-margin SVM.

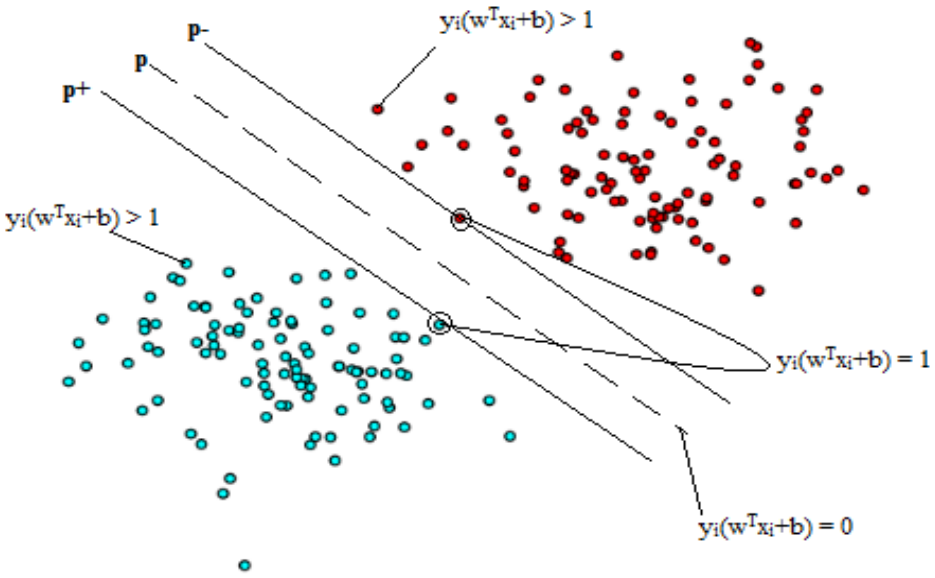


Figure.6: Hard-margin SVM.

If we modify the SVM, we'll get a soft-margin SVM that better generalizes the linearly as-well-as non-linearly separable data. The modification of hard-margin SVM is done by introducing the Zeita ('z_i') value.
We subtract the Zeita value from 1, so the final formulation for soft-margin SVM is as follows:
$y_i(w^T x_i+b) >= 1- z_i$.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

Zeita is the distance of the point away from the correct-label hyper-plane. So as the value of the sum of z_i increases, the model moves towards more misclassifications of points. To make the model more precise and accurate, we need to minimize the value of the summation of z_i also.
So the formulation of soft-SVM is as follows:
$(w^*, b^*) = \text{argmin } (||w||/2) + C * 1/n \Sigma z_i$.
'C' is the hyper-parameter to minimize the value of the sum of z_i.
The last term after C is the average distance of misclassified points from their correct hyperplanes.
To make the model fall between overfitting and underfitting, we add the regularization parameter to the optimization formula.
$(w^*, b^*) = \text{argmin } (||w||/2) + C * 1/n \Sigma z_i + \lambda^*(R)$.
Here, R is the regularization and λ is the hyper-parameter for regularization.
We are applying regularization to prevent our model from overfitting and underfitting. Let us define overfitting and underfitting.
Overfitting is the situation when the model produces good results while training but performs poorly in the testing phase on unseen data. The model hard fit to the training data and loses it generalization for unseen data. The overfitted model shows low bias and high variance.
Underfitting is the situation that occurs when the model performs poorly in both the training and testing phase. The under-fitted model produces randomized results and is also called a weak learner.

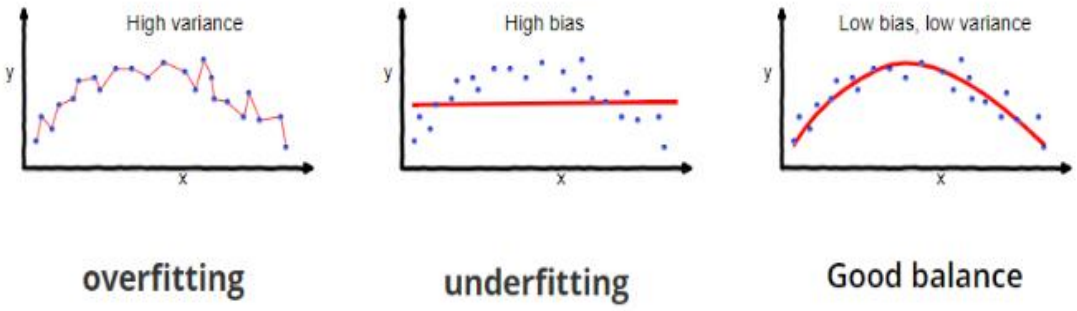


Figure.7: Overfitting VS Underfitting.

Linear models like Logistic regression and Linear regression uses the concept of the loss function. Linear regression uses linear loss function and regularization to produce a linear model.
Logistic regression uses logistic loss and regularization to produce a linear classification model.
Similarly, SVM uses Hinge loss and regularization to produce a linear SVM classification model. The Hinge loss is shown in the form of a graph in Figure.8.

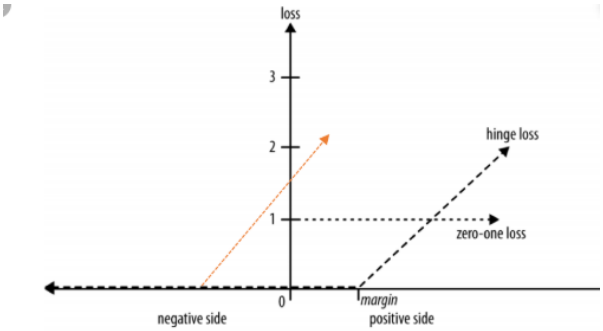


Figure.8: Hinge-loss.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

If $f_i = y_i(w^T x_i+b)$.
Then, for $f_i >=1$, Hinge loss = 0
And, for $f_i<1$, Hinge-loss=$1-z_i$.

V. DUAL FORM OF SVM

The optimization problem of the Support Vector Machine has been derived in section 4. However, The optimization formula derived above is found to be less convenient. The question arises is whether we have a convenient optimization formula for SVM. Due to the curse of dimensionality, optimization of an algorithm became a very powerful tool for solving machine learning problems. To make the SVM algorithm more consistent with programming, the SVM formulation is converted into a quadratic function that can be solved efficiently using a quadratic programming algorithm.
In the primal form of SVM, we have soft margin SVM formulation.
$(w^*, b^*) = \text{argmin } (||w||/2) + C * 1/n \Sigma z_i$. (From primal form)
Here, $z_i>=0$, and $y_i(w^T x_i+b)>=1-z_i$.
In the dual form of SVM [3], the dependencies of x_i is minimized and x_i is converted into some function of alpha (α_i).
The occurrence of x_i exists only in the form of $x_i^T x_j$.
The query input x_q exists in the form of function $f(x_q)$.
$f(x_q) = \Sigma \alpha_i y_i x_i^T x_q + b$.
$\alpha_i > 0$ only for support vectors.
The dual form is shown below:

$$\text{maximize } f(c_1 \dots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j,$$

$$\text{subject to } \sum_{i-1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

In the dual form of SVM, there is a dot product between x_i and x_j. This dot product is nothing but a similarity function between them. Here, we have a cosine similarity between the two of them.
A kernel can be thought of as a similarity function between the input vectors. If K is a kernel function, then the above formulation can be changed by replacing the dot products of x_i and x_j with the kernel function $K(x_i,x_j)$.
So, $f(x_q) = \Sigma \alpha_i y_i K(x_i,x_q) + b$.

VI. KERNEL TRICKS IN SVM

The kernel is the most important idea in SVM. SVM without a Kernel is called Linear SVM [4]. Kernelization is the world-changing idea that makes the SVM different from other linear algorithms like logistic regression.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

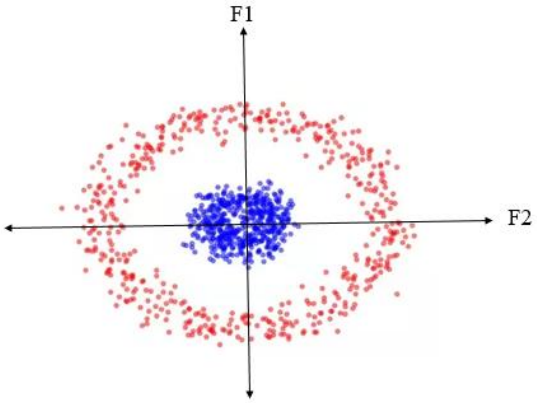


Figure.9: Concentric Circle Dataset.

Consider Figure.9, which shows the concentric circle dataset. A normal linear model like logistic regression is unable to separate the positive points from the negative one. But if we use the feature transformation along with logistic regression, we might be able to separate the positive data from the negative data. Similarly, if we apply the SVM model with the required kernel function, we can successfully separate the two-classes.

Suppose, we want to apply feature transformation, then in the X and Y-axis of the data, we can apply the feature transformation by squaring the features. F1 and F2 are the features in either axis. If we apply to feature transformation on F1 and F2, we get $F1^2$ and $F2^2$, as the 2 new features that lie in the first quadrant with separated data. We get the result as shown in Figure.10. We can separate the classes using the linear model now.

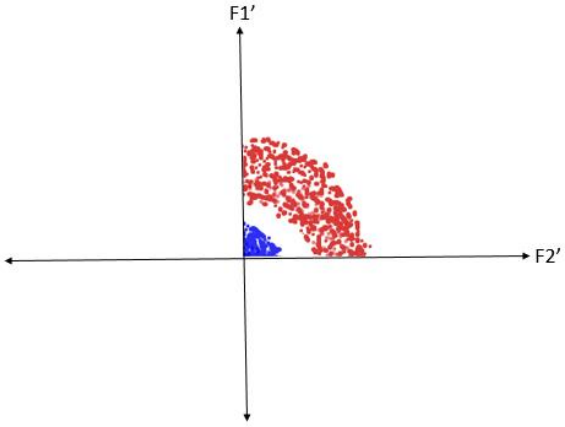


Figure.10: Feature transformation.

Similarly, we have many kernel functions for SVM, that is able to separate the non-linearly separable data.

A. Polynomial Kernel

The kernel function with a degree greater than 1 is called the polynomial kernel. If we generalize the polynomial kernel, then we have,

$K(x1,x2)=(x_1^T x_2 + c)^d$.

Here, d is the degree of the polynomial. If the value of d is 2, then it is a quadratic kernel function.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

If we go deep inside the kernel function, it converts the x1 and x2 into new dimensional feature input x1' and x2'.
So the d dimension of the input feature is changed into some new dimensional feature space d'.

B. Gaussian Kernel

It is a general-purpose kernel. If we don't have an idea of data, and couldn't figure out the right kernel to use, we can go for Gaussian kernel.

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Equation.1: Equation of Gaussian Kernel.

Here, sigma is hyper-parameter that is tuned while fitting the model.

C. Gaussian RBF Kernel

The full form of RBF is Radial Basis Function. This is also a general-purpose kernel that can be used if we don't have prior knowledge of data.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma\|\mathbf{x_i} - \mathbf{x_j}\|^2)$$

Equation.2: RBF Kernel Function.

Here, gamma is hyper-parameter that is tuned at the time of model fitting.

D. Domain-Specific Kernel

We have different types of kernels that can be used according to the domain knowledge. If we are dealing with a text classification problem, we can use the string kernel function.
Similarly, if we have problem statements for the prediction of gene traits prediction, then we can use the Genom Kernel [9].
For graph and tree-related problems, we can use graph kernels.
Kernels have made the problem of machine learning simple. Previously, we used to apply feature transformation and then perform the linear separation of data. Now we only need to find the right kernel function according to the problem statement.

VII. TRAIN AND RUN-TIME COMPLEXITIES

In this section, we will learn about using the model for prediction. Till now, we have learned about mathematical expressions and derived the optimization function for Support Vector Machine. Before moving to the prediction task, the very first step is to train the model. For training the SVM, we can use Stochastic Gradient Descent commonly known as the SGD algorithm. For the dual form of SVM, we have an in-built library of SVM known as the SMO. SMO is Sequential Minimal Optimization which is available in the open-source library called lib SVM.
SVM can also be implemented through open source Sk-Learn library via SVC i.e. Support Vector Classifier.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

The training time for the SVM algorithm depends on many attributes. Generally, Kernel SVMs take training time complexity in the order of $O(n^2)$ [7]. In 2007, an optimization to SVM leads to improvement in the training time which is in the order of $O(n*d^2)$. Here, d is the dimensionality of data. As the dimension of data increses, the time complexity of SVM also increases.
So, for the large datasets, we don't use SVM to get rid of large training time complexity. Talking about Run-time complexity in SVM, it depends on the number of Support Vectors. If k is the number of support vectors, then the run-time complexity of SVM is of the order of O(k*d) where d is the dimensionality of data. On training the SVM model on Amazon Fine-food review data, we got the good results. In the case of Linear SVM, we have reached the training AUC of 0.98 and RBF kernel performed fairly well as we have used less data as compared to that used in Linear SVM. Figure.13 shows the training and the testing area under the curve results in the form of an error plot for the Amazon Fine Food review dataset. Here, we have predicted the sentiments of the review using Linear and Kernel SVM [8]. Figure.14 shows the error plot for kernel SVM. We have used fewer data to train Kernel SVM because it has high training time complexity. Still, we have got an acceptable result for Kernel SVM.
Moreover, SVMs have good Interpretability and feature importance. In real-world cases, it is important for the model to be interpretable. Interpretability provides logical reasoning for the predicted output. In the real world, people cannot blindly follow the model. They should be provided with a valid reason for the prediction. Thus, interpretability helps in projecting the important features along with the prediction so that the user has access to proper reasoning of the predicted output. SVMs are less impacted to Outliers as it completely depends on Support Vectors. Outliers are the noice points that lie away from the dense region of the data, but RBF kernel SVM, with small sigma value, can be impacted to outliers.

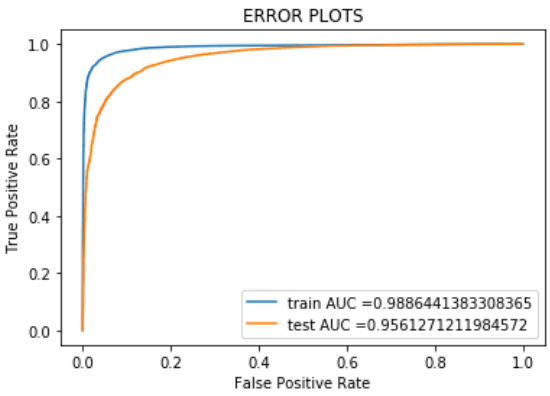


Figure. 13: AUC of Linear SVM.

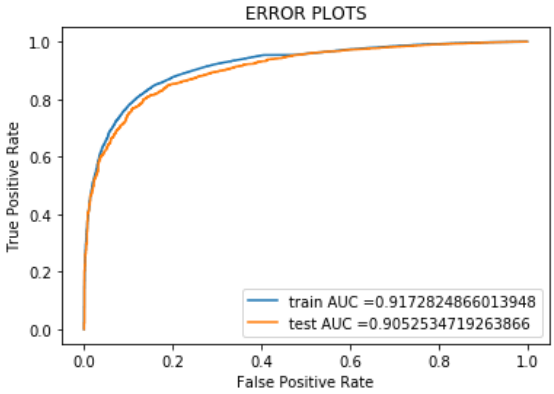


Figure.14: AUC of RBF Kernel SVM

VIII. CONCLUSION

This paper projects the importance of a linear prediction model- Support Vector Machine. We have derived the SVMs and explained their types. We have shown different cases of SVM along with the optimization function, different forms of SVM and their formulation, and how SVMs are different from other linear models. The introduction to kernel tricks associated with SVM is explained and found to better than feature transformation tricks. We have shown the results of Linear SVM vs Kernel SVM. even for small data, kernel SVM produces good results which proves that RBF SVM is better than Linear SVM for sentimental analysis on the raw set of data.

REFERENCES

[1] Corinna Cortes, Vladimir Vapnik., AT&T Bell Labs., Holmdel, NJ 07733, USA, Kluwer Academic Publishers, Boston, 1995.

Copyright © 2019Authors

ISSN: 0971-1260
Vol-22-Issue-16-August-2019

[2] Yujun Yang; Jianping Li; Yimei Yang, 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), IEEE, 2015 .

[3] Yingjie Tian, Yong Shi, Xiaohui Liu, Recent Advances On Support Vector Machines Research, Technological and Economic Development of Economy, March 2012, ISSN.

[4] Alex J. Smola and Bernhard Scholkopf ¨ RSISE, Australian National University, Canberra 0200, Australia, November 2004, 2004 Kluwer Academic Publishers.
https://alex.smola.org/papers/2004/SmoSch04.pdf

[5] Shawe-Taylor, John, and Nello Cristianini. "Support vector machines." An Introduction to Support Vector Machines and Other Kernel-based Learning Methods (2000): 93-112.

[6] Scholkopf, Bernhard, and Alexander J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001.

[7] Hearst, Marti A., et al. "Support vector machines." IEEE Intelligent Systems and their applications 13.4 (1998): 18-28.

[8] Liu, Tie-Yan, et al. "Support vector machines classification with a very large-scale taxonomy." Acm Sigkdd Explorations Newsletter 7.1 (2005): 36-43.

[9] Gunn, Steve R. "Support vector machines for classification and regression." ISIS technical report 14.1 (1998): 5-16.

Copyright © 2019Authors