Approaches to Multi-Class classification using Support Vector Machines

Bbiswabasu Roy
Electronics & Electrical Communicaton
Indian Institute of Technology, Kharagpur
Kharagpur, India
19EC39007

Jothi Prakash
Electronics & Electrical Communicaton
Indian Institute of Technology, Kharagpur
Kharagpur, India
19EC39023

***Abstract*—Classification is an important task in the domain of Machine learning. Classification problems can be either binary classification or it can be multi-class classification. There are various approaches which are currently used to solve the classification problem, some of which includes Decision Trees, Bayes Classifier, Support Vector Machines, Neural Networks, etc. Support Vector Machines (SVMs) have been widely and successfully used for binary classification. However, it is not trivial to extend the concept of SVM to multi-class classification problems. In this paper, we propose various approaches to solve the above mentioned problem and compare their performances on our synthetic dataset as well as iris dataset through simulations.**

***Index Terms*—Machine Learning, Support Vector Machines, Multi-class classification, DAG-SVM**

Implementation of the classifiers in python

I. Introduction

Support Vector Machines (SVM) is a popular machine learning framework under the Supervised Learning algorithms to classify binary data. Real world data on the other hand are not always binary and majorly demand a multiclass classifier approach to classify data. In this paper we try to approach the multiclass classification problem using SVMs [1], while modifying it using different algorithmic techniques to test its performance. Subsequently in the paper we have used 3 main approaches namely (III-A) $One-against-all$, (III-B) $One-against-one$ classifier and (III-C) $DAG-SVM$ classifier [2]. We then compare the results (IV) of each classifier on the Iris Dataset to quantitatively measure the performance metrics.

The structure of paper will be, Section II Theory of the Binary SVM classifier along with the impact of margins and SVM kernels. Section III All the 3 multiclass SVM classifier approaches. Section IV Visualizations and performance ratios on the Iris Dataset. Section V Conclusion of our classifier approach.

II. Binary SVM classifier

The problem that Binary SVM classifier attempts to solve is as follows: Given a set of labeled training vectors with each vector in exactly one of the two classes, find the hyperplane that seperates the data points with maximum margin. The margin is defined as twice the distance of the hyperplane from the point in either class which is closest to the hyperplane. Depending on whether we want the separating hyperplane to completely separate the points without any overlap or we allow some overlap between the two classes, there can hard margin SVM and soft margin SVM respectively. We start with the solution of hard margin SVM and then modify the equations slightly to arrive at the soft margin SVM.

A. Hard margin classifier

Let the data points under consideration be $(\mathbf{x_i}, y_i)$, where $\mathbf{x_i} \in \mathbb{R}^{\mathrm{d}}$and $y_i \in \{-1, 1\}$, $i = 1, 2, ..., n$. Any hyperplane h in this space can be represented as $\mathbf{w^t x} + b = 0$. Distance of point $\mathbf{x_i}$ from h is given by

$$D(\mathbf{x_i}, h) = |\frac{\mathbf{w^t x_i} + b}{||\mathbf{w}||}| \tag{1}$$

Since we are considering hard margin boundary, $y_i(\mathbf{w^t x_i} + b) > 0, i = 1, 2, ..., n$. Now our objective becomes

$$\max_{\mathbf{w},b}[\min_i(y_i(\frac{\mathbf{w^t x_i} + b}{||\mathbf{w}||}))] \tag{2}$$

Without loss of generality, we may assume that $y_s(\mathbf{w^t x_s} + b) = 1$, where $(\mathbf{x_s}, y_s)$ belongs to the set of points closest to h. This implies $y_i(\mathbf{w^t x_i} + b) > 1$ for other points. Hence, our constrained objective function becomes

$$\begin{aligned} &\min_{\mathbf{w},b}(\frac{1}{2}||\mathbf{w}||^2) \\ &\text{subject to } y_i(\mathbf{w^t x_i} + b) \geq 1, i = 1, 2, ...n \end{aligned} \tag{3}$$

The Lagrangian is

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i(y_i(\mathbf{w^t x_i} + b) - 1) \tag{4}$$

The stationary conditions are

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i y_i \alpha_i \mathbf{x_i} = 0 \tag{5}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_i y_i \alpha_i = 0 \tag{6}$$

Substituting back into the Lagrangian gives the dual cost function and the optimization now becomes

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}}[\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \mathbf{x_i^t x_j}]$$
$$\text{such that} \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0 \qquad (7)$$

From the KKT optimality conditions, $\alpha_i(y_i(\mathbf{w^t x_i} + b) - 1) = 0$ which implies either $\alpha_i = 0$ or $y_i(\mathbf{w^t x_i} + b) - 1 = 0$. The latter inequality holds for the points which are at minimum distance from the hyperplane and are called *support vectors*. While for rest of the points $\alpha_i = 0$. Fig. 1 shows data points belonging to two different classes, the separating hyperplane obtained using SVM along with the support vectors. Finally, the classification function is given by $f(\mathbf{x}) = sign(\mathbf{w^t x} + b)$, where $\mathbf{x}$ is the test vector to be classified.

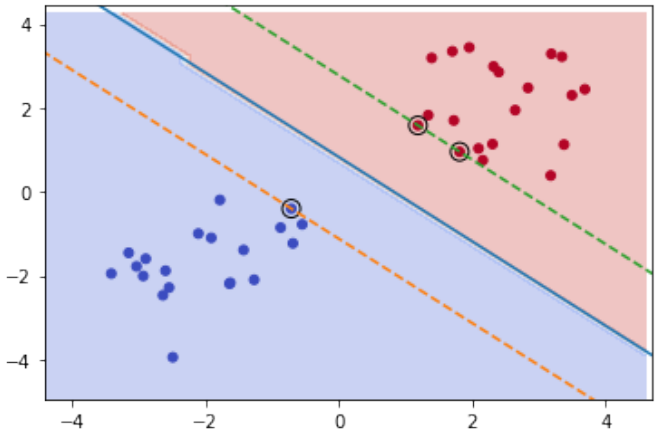

Fig. 1. Linear binary hard margin SVM classifier

B. Soft margin classifier

The equations of hard margin classifier can be modified using non-negative *slack variables* $\xi_i \geq 0$ to get soft margin classifier. The modified objective can be written as

$$\min_{\mathbf{w},b}(\frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i)$$
$$\text{subject to } y_i(\mathbf{w^t x_i} + b) \geq 1 - \xi_i, i = 1, 2, ...n \qquad (8)$$

C is a user-defined constant, which controls the amount of error that we allow on training set, by penalizing wrong classifications. Using the same method as eariler, the Lagrange dual problem becomes

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}}[\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \mathbf{x_i^t x_j}]$$
$$\text{such that} \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C \qquad (9)$$

As can be seen, the only difference for soft margin classifiers is the upper bound on α_i's. Fig. 2 shows the soft margin classifier with the boundary, when $C = 1$. It treats the two points as outliers, accommodates high error for them while maintaining good generalizability. On the other hand, in Fig. 3 when $C = 10^6$, the margin tries to become as close to hard margin classifier as possible as a result of which the generalizability is lost.

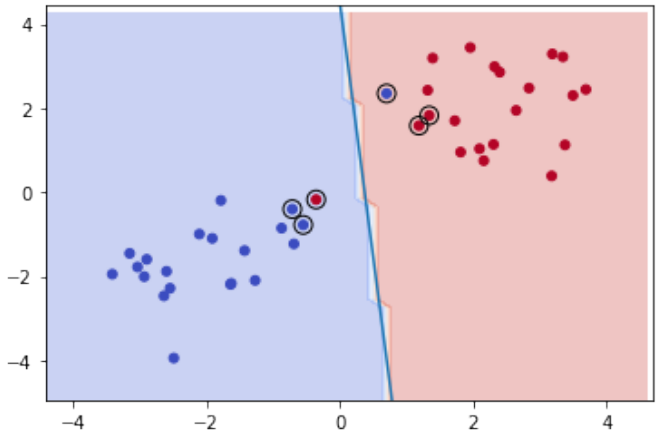

Fig. 2. Linear binary soft margin SVM classifier with $C = 1$

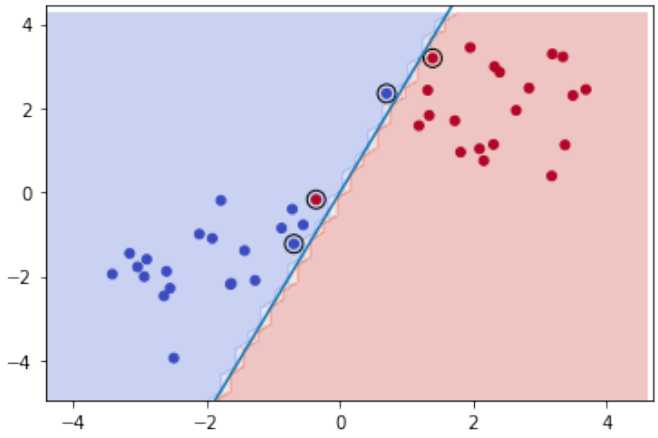

Fig. 3. Linear binary soft margin SVM classifier with $C = 10^6$

C. Kernel Trick

The SVM models discussed till now are only able to draw the hyperplane if the data is linearly separable or at most by considering some points as outliers, it can find a good enough hyperplane. The basic idea is that even if the data points are not linear separable in its original d-dimensional, they can be linearly separable in some higher dimensional space. Consider a function $\boldsymbol{\phi} : \mathbb{R}^{\mathrm{d}} \rightarrow \mathbb{R}^{\hat{\mathrm{d}}}$, $\hat{d} > d$, which maps the given set of points to higher dimensional space. Since the final optimization function involves only inner product of feature vectors, we can construct a kernel function $k : \mathbb{R}^{\mathrm{d}} \times \mathbb{R}^{\hat{\mathrm{d}}} \rightarrow \mathbb{R}$

which is defined as $k(\mathbf{x_1}, \mathbf{x_2}) = \boldsymbol{\phi}(\mathbf{x_1})^t \boldsymbol{\phi}(\mathbf{x_2})$. Now our most generalized optimization function for SVM becomes

$$\hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} [\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(\mathbf{x_i}, \mathbf{x_j})]$$
$$\text{such that} \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C \qquad (10)$$

Fig. 4 shows a set of data points which are not linearly separable. However, using Radial Basis Function (RBF) kernel, the SVM algorithm could find a non-linear separating boundary.

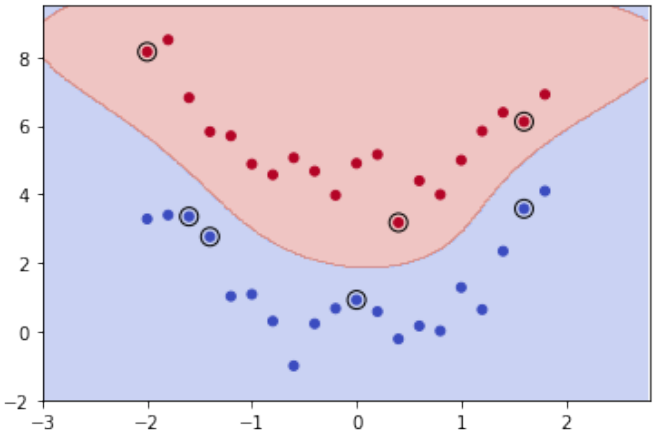

Fig. 4. Binary soft margin SVM classifier with kernel

III. Multi-class SVM classifier

A. One-against-all classifier

One of the earliest and simplest technique used to extend binary SVMs to do multi-class classification was using one-against-all method. Let the data points under consideration be $(\mathbf{x_i}, y_i)$, where $\mathbf{x_i} \in \mathbb{R}^d$and $y_i \in \{1, 2, ..., k\}$, $i = 1, 2, ..., n$. Here k is the number of classes. In this approach, we build k different SVMs where the i^{th} SVM is trained by treating points in i^{th} class to have label $y = 1$ while rest all points have label $y = -1$. Hence, the problem is now reduced to training k different binary SVM classifiers independently and the hyperplane parameters $\mathbf{w}^i$ and $b^i, i = 1, 2, ..., k$ can be obtained using Eq. (10). Fig. 5 shows the hyperplanes obtained using one-against-all method with the color of the line denoting the class which was considered against others. As can be seen, there are regions in the feature space which belongs to none of the classes as well as there are regions where multiple classes overlap. In order to avoid such ambiguities, the classifier function for a test point $\mathbf{x}$ is given by

$$f(\mathbf{x}) = \arg\max_i ((\mathbf{w}^i)^t \boldsymbol{\phi}(\mathbf{x}) + b^i) \qquad (11)$$

Fig. 6 shows the classification regions obtained using the classifier function (11). Although all points in the space are mapped to some class, it can be observed that the classification of some points do not look like having maximum margin classification.

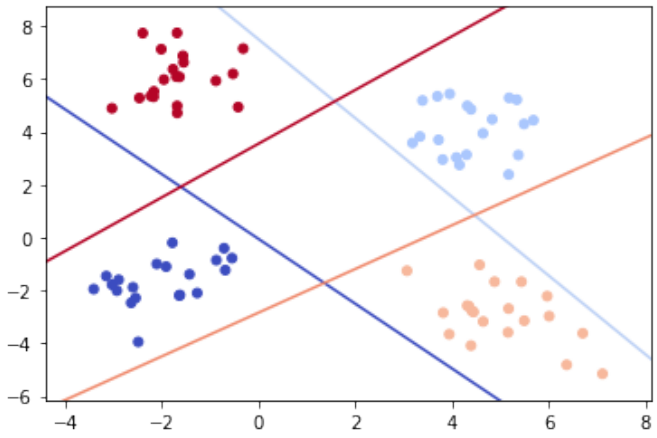

Fig. 5. Individual hyperplanes obtained using one-against-all method

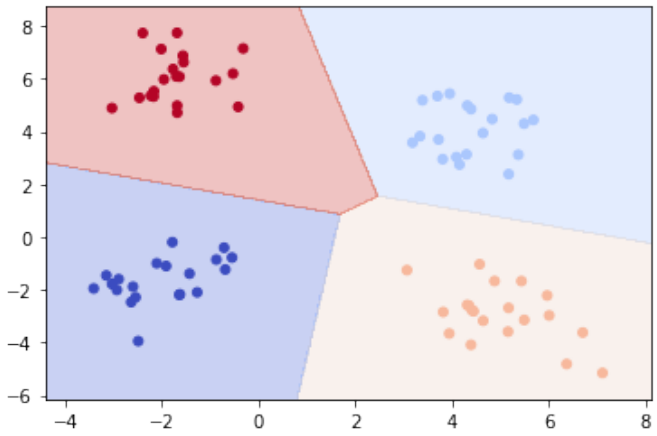

Fig. 6. Classifier obtained using one-against-all method

B. One-against-one classifier

The next method is one-against-one classifier in which we take two classes (say i & j) at a time and get SVM classifier separating those two classes using Eq. 10. In this way, we will get $k(k-1)/2$ classifiers having parameters $\mathbf{w}^{ij}$ and $b^{ij}, i = 1, 2, ..., k, i < j$. In order to classify a test point, we take each pair of classes i, j and check which class the point belongs to using the classifier function for binary SVM. If it is classified as class i, we add a vote for class i, else we add a vote for class j. After doing this for all pairs of classes, we get a distribution of votes for each class and the test point is classified as belonging to the class having highest number of votes. Fig. 7 shows the hyperplanes obtained using this method with the two colors on the lines denoting the two classes which were considered to construct the classifier.

The classification regions were obtained using majority voting strategy and is shown in Fig. 8. It can be seen that classification regions obtained using one-against-one classifier is much more balanced as compared to that obtained using one-against-all classifier.

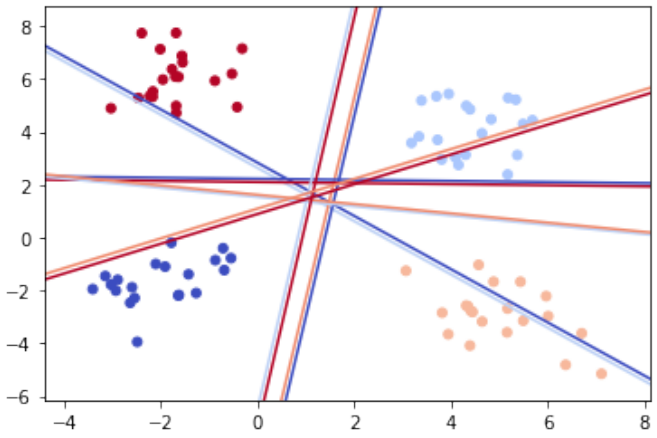

Fig. 7. Individual hyperplanes obtained using one-against-one method

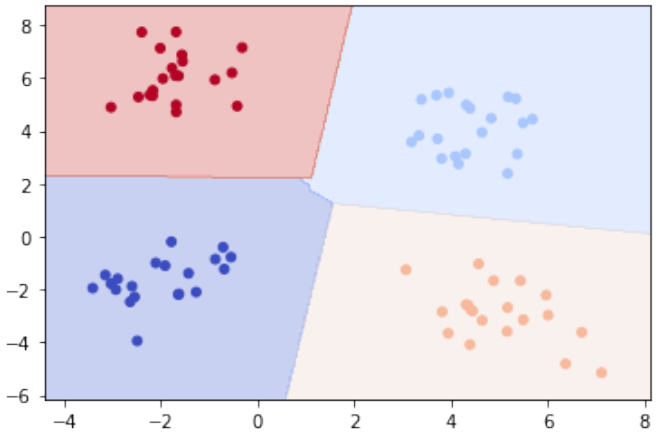

Fig. 8. Classifier obtained using one-against-one method

C. DAG SVM classifier

A modification over the one-against-one SVM classifier leads us to another approach known as Directed Acyclic Graph Support Vector Machine(DAG SVM) [3]. Its training phase is same as that of one-against-one SVM but during the test phase, it performs prediction by traversing a DAG whose construction is discussed subsequently. This way the classifier is guided to a single bucket and when no further comparison can be made, we assign that class to the current sample.

Let us consider a k class problem where the data points $(\mathbf{x_i}, y_i)$, $\mathbf{x_i} \in \mathbb{R}^{\mathrm{d}}$and $y_i \in \{1, 2, ..., k\}$, $i = 1, 2, ..., n$. Initially we train $k(k-1)/2$ SVM classifiers using each pair of distinct classes i and j to create a DAG structure. Each node in the graph is a binary SVM classifier trained for the pair of class i and j (call it $node_{ij}$). $node_{ij}$ (except sink nodes) has two outgoing edges, one of which represents the given point not belonging to class i while the other representing. The left child of $node_{ij}$ is $node_{i+1j}$ while the right child is $node_{ij-1}$. The root node is $node_{1k}$. Refer Fig 9 for the node structure. This process of training and storing the classifier is done recursively at each node till we reach the sink nodes, where $i = j$. It can

$node_{ij}$
{
$Tuple$ index(i,j),
$SVM - Classifier$ clf,
$node_{i,j-1}$ leftChild,
$node_{i+1,j}$ rightChild
}

Fig. 9. Data structure of $node_{ij}$

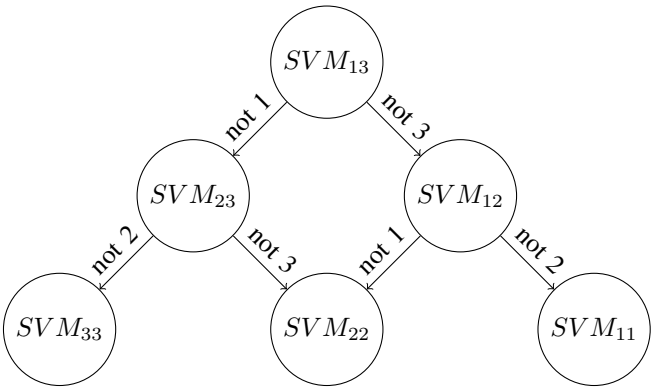

Fig. 10. Graph structure of DAG SVM - 3 classes

be observed that the number of nodes in the graph will be $k(k+1)/2$ out of which k are sink nodes (one for each class) and the height of the graph will be k since at each stage we reject one of the classes. Fig. 10 shows a DAG SVM structure with 3 classes.

The class prediction for a test point x_{test} starts from the root node, where we make a prediction using the $node_{1k}$ on whether x_{test} does not belong to class 1 or it does not belong to class k and accordingly choose the edge to traverse. If we predict that x_{test} does not belong to class 1, then we move to the $node_{2k}$, else we move to node $node_{1(k-1)}$. We repeat this recursively at each node till we reach the sink node $node_{ij}$, where $i = j$. Predicted class label for x_{test} will be i.

IV. Experiments and Results

In this section, we describe and discuss experiments that we have performed on the Iris Dataset from the UCI Machine Learning Repository. The Iris Dataset is split into 70% train and 30% test. We then trained our 3 different Multi class SVM classifiers namely - $One - against - all$ classifier, $One - against - one$ classifier and the $DAGSVM$ classifier and measured the performance on the train and test dataset.

We performed Principal Component Analysis (PCA) on the Iris Dataset and took 2 principal components to perform visualizations of the different decision boundaries and predictions on the test data. Fig 14 describes the training data after doing PCA. Each classifier was trained with linear

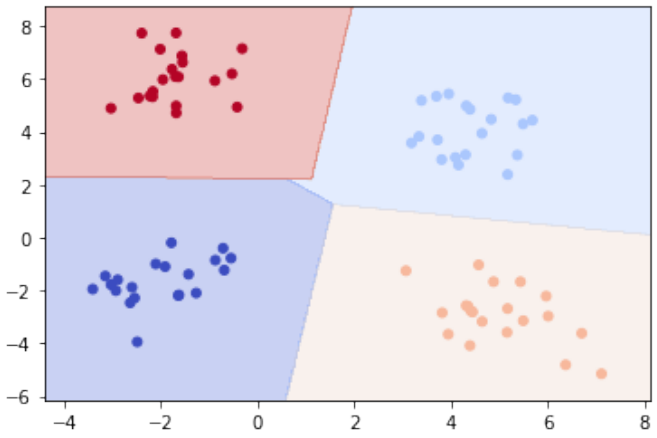

Fig. 11. DAG SVM one-against-one classifier

Feature	Type
Sepal Length	Continuous
Sepal Width	Continuous
Petal Length	Continuous
Petal Width	Continuous
Variety	Discrete

Fig. 12. Iris Data Set Description

kernel on the training dataset containing 105 samples chosen randomly and the test dataset contained 45 samples. Fig 13 shows the accuracy of each classifier on the training as well as the test set. It can be seen how $one - against - one$ and $DAGSVM$ classifiers outperformed $one - against - all$ classifier. Fig 15 shows the classification regions learnt by the $One - against - all$ classifier using the training set and classification of each point in the test set. Fig 16 shows the same for $One - against - one$ and Fig 17 does it for $DAGSVM$. As expected, we can see that there are many misclassified points in case of $one - against - all$ classifier while for the other two classifiers, there is only one misclassified point. Also, the cause of such high number of misclassifications can be observed from the decision boundaries of $One - against - all$ classifier as it is highly biased towards the class present on the left.

V. Conclusion

We discussed three different approaches for extending Support Vector Machines to Multi Class classification problem which were originally binary in nature. It is important to note that the only difference between $One - against - one$ and

Classifier	train	test
one-against-all SVM	90.47%	93.33%
one-against-one SVM	95.23%	97.77%
DAG-SVM	95.23%	97.77%

Fig. 13. Classification Accuracy for the Multiclass Classifiers

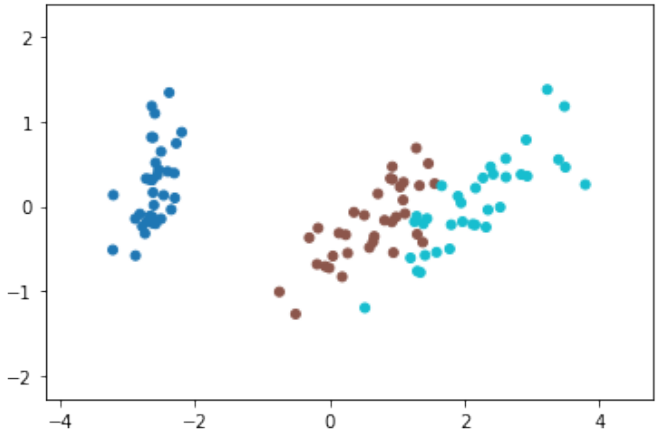

Fig. 14. Iris Training Dataset with two principal components

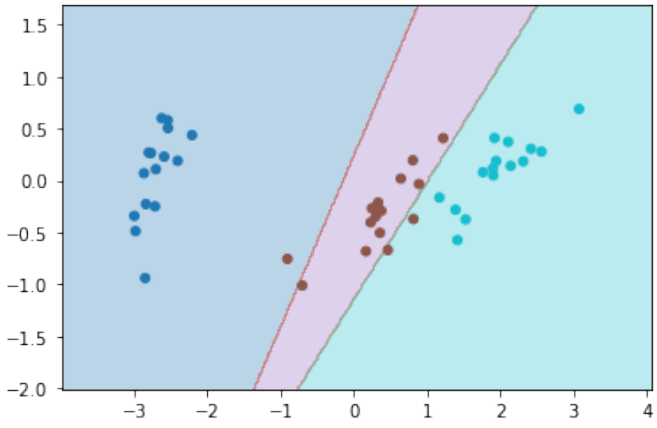

Fig. 15. One-against-all classifier on Iris test

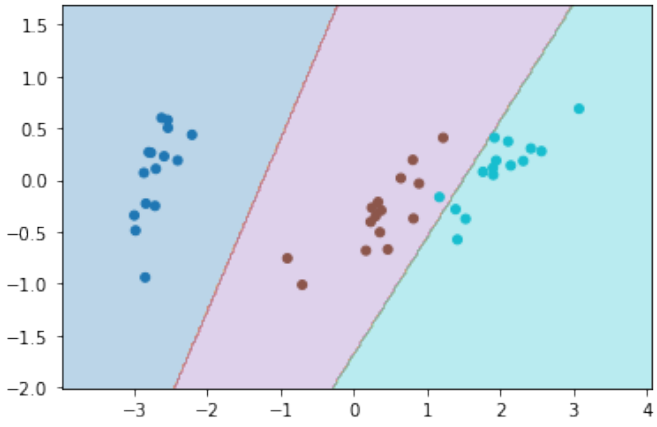

Fig. 16. One-against-one classifier on Iris test

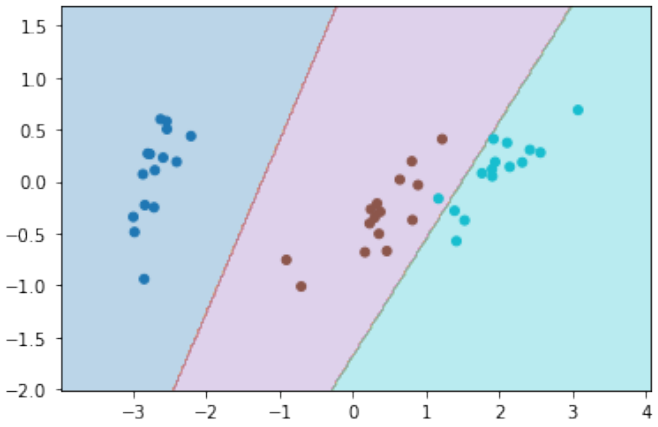

Fig. 17. DAG SVM classifier on Iris test

$DAGSVM$ classifiers is their classification function. While $One-against-one$ classifier uses majority voting to decide on the class of the test point, $DAGSVM$ uses a DAG to do the same. From our experimental results we can conclude that the $One-against-one$ and $DAGSVM$ performed significantly better than the $One-against-all$ classifier both on the training data and the test data.

$DAGSVM$ algorithm is indeed superior to other SVM multiclass algorithms in terms of both training time as well as test accuracy. Empirically, it is observed that SVM training scales super-linearly with the size of training dataset (say n) according to the power law $T = cn^{\gamma}$, where c is some proportionality constant and γ depends on the algorithm being used to train. For the $One-against-all$ algorithm, we use the entire training set to train each of the k classifiers and hence we obtain

$$T_{One-against-all} = ckn^{\gamma} \tag{12}$$

Assuming all classes have same number of examples, each training considers $2n/k$ samples and there will be $k(k-1)/2$ which gives

$$\begin{aligned} T_{One-against-one} &= c\frac{k(k-1)}{2}\left(\frac{2n}{k}\right)^{\gamma} \\ &\approx 2^{\gamma-1}ck^{2-\gamma}n^{\gamma} \end{aligned} \tag{13}$$

For a typical case, we have $\gamma = 2$, which yields $T_{One-against-all} = ckn^2$ and $T_{One-against-one} = 2cn^2$. So the time taken for the later is independent of the number of classes and is only twice the time taken to train each classifier in the former.

REFERENCES

[1] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[2] P. Chen and S. Liu, "An improved dag-svm for multi-class classification," in *2009 Fifth International Conference on Natural Computation*, vol. 1, pp. 460–462, 2009.

[3] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in Neural Information Processing Systems* (S. Solla, T. Leen, and K. Müller, eds.), vol. 12, MIT Press, 1999.