Foundations of Data-Science Assignment-1

Pramit Dutta, NUIM

1. Introduction to the Assignment

This assignment is written for the partial fulfilment of the FDS-1 coursework. The assignment is divided into 5 sections. Section 2 provides an introduction and understanding of the Support Vector Machine for classification problem with a brief outlay of the underlying theory and practical considerations. The example dataset of radar observations of ionosphere for radio signals and the evaluation of predictions using skill-score is discussed in Section 3. Section 4 gives a comparative analysis of using various classification techniques with the help of the skill-score discussed in Section 3. Section 5 summarizes the report.

2. Introduction to Support Vector Classifier

Machine-learning (ML) problems are categorically defined as regression or classification problems. 'Supervised Classification' problems is one of the most frequently carried out task in ML domain [1]. These problems deal with categorizing an input feature vector (x_i) into a particular class. The number of class outputs can be *binary* or *multi-class*. The term supervised learning refers to the fact that the training algorithm uses a set of feature-input and class-label pairs for defining the model weights and bias. Due to the wide application of such algorithms, number of classification algorithms have been developed and researched over the past few decades. Off these, Support Vector Machines (SVM), originally proposed in [2], is a binary classification algorithm that can handle inputs with high dimensionality and continuous features [3].

2.1 SVM Classification Problem and Theory

The SVM algorithm can be understood using Fig. 1. As shown, a set of input represented by x_i is distributed in the feature space into two separate classes, namely A and B. The problem pertains to defining an optimal mathematical solution or 'Hyperplane' that can clearly differentiate the class boundaries for linearly separable patterns. Non-linear separable pattern is discussed further in Section 2.2. As shown, a number of planes can perform the classification. Lines L_a, L_b and H, all provide a demarcation of the classes. However, an optimal hyperplane is defined as the linear decision function that provides the maximum margin (*d,* in Fig.1) between the vectors of the two classes [2], [4], represented by the line H in Fig. 1. The lines H1 and H2 represent the decision boundaries of the two classes. SVM algorithm relies on the tips of the *support vectors* that just touch the H1 and H2 boundaries. Fig. 1 shows 2 such points on either side of the decision function.

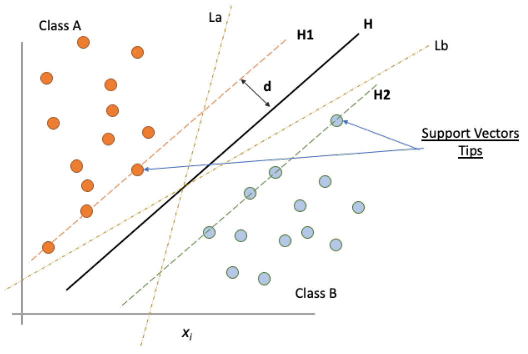


Fig. 1. Graphical Representation of SVC

For a given set of labels dataset in the form of

$$(y_1,x_1),(y_2,x_2),\ldots.(y_n,x_n)\,, \qquad y_i\epsilon\{-1,1\}$$

The classes are linearly separable if there exists a weight vector w and a scalar bias b such that

$$H_1 : \; w.x_i + b \geq 1 \;\; if \;\; y_i = 1$$

And, $$H_2 : w.x_i + b \;\leq -1 \;\; if \;\; y_i = -1$$

Combining the Eq. 1 & 2 we can write:

$$y_i\,(w.x_i + b) \geq 1, \qquad i = 1,\ldots,n$$

The optimal hyperplane, the median, is then defined by

$$w_o.x + b_o = 0$$

The optimization algorithm to generate the weights proceeds in such a way that only the support vectors determine the weights and thus the boundary. The distance between the optimal plane H and H1 or H2 can be written from the distance of a point from line which can be simplified as $1/||w||$, thus the margin between the boundaries given by H1 and H2 is $2/||w||$. In order to maximize the margin, the optimization problem is a quadratic programming problem to minimize $||w||$. The solution to this problem using Lagrange Multipliers is discussed in detail in references [2],

[4]–[6], and not presented further within the scope of this assignment.

2.2 Practical Considerations in Implementing SVM

A number of libraries today provide the basic skeleton framework to implement the SVM algorithms for both classification and regression problems. Scikit-learn library most commonly used for implementation in Python. Other libraries such as 'TensorFlow' and 'TF-lite' build upon the Scikit-learn [7], [8].

Within this section we briefly discuss some of the practical aspects when implementing SVM classification using the Scikit learn library. The dimensionality reduction and class imbalance presented here can be used with other algorithms as well.

Hard and Soft Margin

Hard-margin is defined when we can define a clear linear separable boundary between the 2 classes as shown in Fig.1. In most practical cases, however, such clearly defined datasets are seldom available. Often outliers cross over to the other side of the boundary and cause issues in such boundary demarcation. As shown in [2], the Soft-margin hyperplane algorithm incorporates an additional hyperparameter C that allows for some amount of margin violations. A smaller C leads to a wider boundary allowing more outliers and generalizing the classification [8].

Non-Linear Classification

As in the case of Soft-margins another common problem with practical datasets is the existence of a non-linear boundary between the two classes. This means that the boundary is not a straight line in the feature space but rather a polynomial of some higher degree. In such case two common tricks are used during the training. The first method is to introduce '*polynomial features'* to the dataset using certain transformation so that in the new feature space the decision function is linear. Usually an input-pipeline is created to read the input data-frame and convert and add the polynomial feature. The second method uses the *'kernel-trick'* which implements similar to method one but does not actually transform the dataset to a new feature space and is thus faster [6]. The hyper-parameter *'degree'* can be used implement the same.

Dimensionality Reduction

Most often the input vectors are multi-dimensional, i.e., each training vector x_i has a number of features. This makes the training slow and harder to find a good hyperplane. Various techniques such as projections, manifold learning and Principal Component Analysis (PCA) are used to select the features of input instances that provide maximum variance and can be effectively used for training. Such technique also provides improved visualization.

PCA identifies the axis that accounts for the largest amount of variance in the training set [6]. Using PCA, feature axes can be chosen which contribute the most in defining the boundaries between the classes. PCA inherently uses Singular Value Decomposition (SVD) to decompose the training input matrix X intro the dot product $\mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T$, where $\mathbf{V}^T$, gives the orthogonal principal components. Further, *'explained variance ratio'* can be used to determine the contribution of each principal component axes towards the dataset variance.

Class Imbalance

The final practical consideration is the evaluation of number of input vectors available for each of the classes. The dataset is imbalanced if the classes are not approximately equally represented. A practical example is in case of fraud detection where 1 in 100 cases is a potential fraud. The SMOTE (Synthetic Minority Oversampling Technique) algorithm is a very commonly used method to reduce such imbalance. Within this, for certain N% of minority class random inputs, k nearest neighbors are selected and synthetic data features are generated [9]. For cases involving large datasets, a combination of under-sampling of the majority class and oversampling of the minority class using SMOTE is used to produce a balance training dataset [10].

The techniques discussed in this section can be used along with other optimizations of hyper-parameters such as batch size, number of epochs, learning rate, etc. to achieve an appropriate ML model.

3. Ionosphere Dataset and Skill Score Matrix

This section showcases the understanding of the given dataset, in brief, and the skill score methods for understanding the weather predictions.

The dataset [11] is comprised of 350 instances of radar data that was collected by the Goose-bay laboratory. The radar signals bounced off the free electrons in the ionosphere are categorized as good returns where as those that move past the ionosphere. The dataset is in the form of 350x35 array wherein the first 34 columns of a row provide an instance of the captured radar data. The 35th column gives the good/ bad ('g'/ 'b') label for the instance [12]. During the training of the ML algorithm the dataset is split into a 66.6/33.3 ratio for training and testing.

The verification of the classification can be done using a number of metrics. The simplest metrics to use is mean accuracy of the predictions that the trained model makes on the testing dataset. However, in case of existence of a class imbalance, a high prediction accuracy might be misrepresenting as the model may predict sufficiently well for the majority class rather than the minority class. In such cases a confusion matrix as shown below is used.

Confusion Matrix	Observed (Or Actual) Outcome	
Predicted Outcome	Negative	Positive
Negative	True Negative (Correct Negative)	False Negative (Misses)
Positive	False Positive (False Alarm)	True Positive (Hits)

Table 1. Confusion Matrix Definition

Skill score metrics for predictions can then be defined as various probabilities based on the above confusion matrix. Out of the various skill score metrics provided in references [8], the assignment uses the following two for comparing the ML algorithms implemented in Section 4.

Peirces's Skill Score (True Skill Score)

The True Skill Score (TSS) is defined as follows:

$$TSS = \frac{TP}{TP + FN} - \frac{FP}{FP + TN}$$

For binary classification problems, this score provides a realistic insight into capability of the algorithm to distinguish between the yes and on event. The perfect score for this is 1.

Heidke Skill Score

Measures the fraction of correct forecasts after eliminating those forecasts which would be correct due purely to random chance. The HSS measures the fractional improvement of the prediction over the standard forecast. It is defined as follows:

$$HSS = \frac{2 * (TP.TN - FP.FN)}{(TP + FN)(FN + TN) + (TP + FP)(FP + TN)}$$

It is normalized by the total range of possible improvement over the standard, which means HSS can be used on various datasets.

4. Classification of Ionosphere dataset using various algorithms

In this section we perform the binary classification of ionosphere data using various algorithms. The general pseudo-code for all implementations is as shown below:

ALGORITHM:

```
1. DF (351x35) ← Read input csv file
2. Convert: DF [34] from 'g'/ 'b' to 1 / 0
3. Labels ← DF [34]
4. Input ← DF[0:350,0:33]
5. Split: Train_Set (Inp-Out), Test_Set (Inp-Out)
6. SMOTE: Resample (Train_Set)
7. PCA: Train and Test Inputs
8. model ← ML-Algorithm (SVC / NN / RFC)
9. Fit-model ← Train_Set (Input-Out)
10. Predict-model ← Test_Set (Input)
11. Confusion_Mat_model ←[Test_Set(Out), Predict-model]
12. Find_Heidkescore ← Confusion_Mat_model
13. Find_TrueSkillScore ← Confusion_Mat_model
```

** End of Pseudo-Code **

As a part of the assignment various ML-Algorithm, in *line 8*, have been used on the same dataset and compared based on the Heidke score and True Skill score as explained in section 3. As the basic implementation of the code is same, in each of the following subsections only the hyper-parameters have been discussed for each classification algorithm. *The complete code of the assignment is available in the git-hub page here* [13].

Support Vector Classifier

The SVC algorithm is implemented using the SVM class of scikit-learn classifier. To evaluate the effect of soft classification the soft-margin coefficient hyper-parameter is varied between 1 and 20 to find an optimum value based on the TSS and HSS scores. The result of the same is shown in Fig. 2. As seen, a 'C' value of 3 gives the highest scores.

Random Forest Classifier

Similar to SVC, RFC is implemented with a hyper-parameter optimization for the depth of the decision tree. The depth is varied between the range of 1 to 20. The result for the same is shown in Fig. 3. As seen, a depth of 5 provides the best results for HSS and TSS.

Neural Network Classifier

The neural network is also implemented for binary classification.

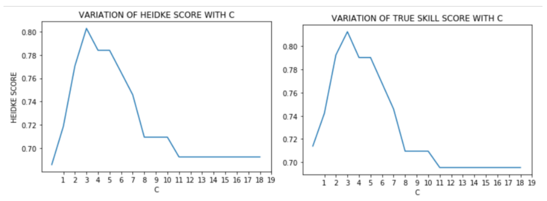


Fig 2: Variation of HSS and TSS with 'C'

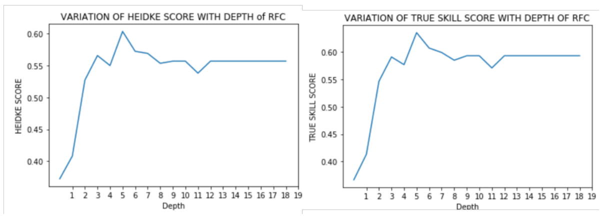
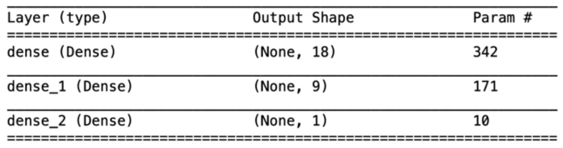


Fig. 3 Variation of HSS and TSS with 'Depth'



```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 18)                342
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 171
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 10
=================================================================
```

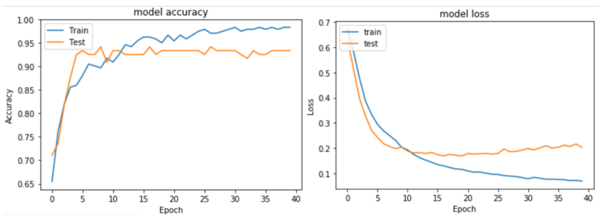


Fig. 4 NN Layers and Training-Testing Accuracy and Loss

Discussion and Comparison of Results

The final results with the optimum models for all the above classifications is shown in the table below.

The support vector classifier performs better than the random forest classifier. The RFC provides a number of false negatives on the validation set. The neural network model performs best in terms of train-test accuracy and validation accuracy. However, SVC algorithm is much easier to implement and faster.

	TSS	*HSS*	*Confusion Matrix*	
SVC with C = 3	0.813	0.803	TN: 41	FP: 4
			FN: 7	TP: 64
RFC With Depth = 5	0.635	0.603	TN: 40	FP: 5
			FN: 18	TP: 53
NN with Configuration (18,9,1)	0.821	0.848	TN: 23	FP: 5
			FN: 0	TP: 43

Table 2. Comparison of Classification Results

5. Conclusion

All the three parts of the assignment are presented in this report. The SVC algorithm theory and practical aspects have been presented. The given dataset has been classified for good and bad signals using three different algorithms – SVC, RFC and Neural Network. The True Skill score and Heidke Skill score show that the NN performs a better classification than the RFC and SVC.

Reference:

[1] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Inform.*, vol. 31, no. 3, pp. 249–268, 2007.

[2] U. Cortes, Corinna (AT&TBellLabs., Hohndel, NJ07733 and U. Vladimir, Vapnik (AT&TBellLabs., Hohndel, NJ07733, "Support-Vector Networks," *Mach. Learn.*, vol. 297, no. 20, pp. 273–297, 1995.

[3] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: A review of classification and combining techniques," *Artif. Intell. Rev.*, vol. 26, no. 3, pp. 159–190, 2006.

[4] S. Tong and D. (Stanford U. Koller, "Support Vector Machine Active Learning with Applications to Text Classification," *J. Mach. Learn. Res.*, pp. 45–66, 2001.

[5] W. Khan, A. Daud, J. A. Nasir, and T. Amjad, "A survey on the state-of-the-art machine learning models in the context of NLP," *Kuwait J. Sci.*, vol. 43, no. 4, pp. 95–113, 2016.

[6] A. Géron, *Hands-On Machine Learning with Scikit-Learn and Tensorflow*, First Edit. O'Reilly, 2017.

[7] L. Buitinck *et al.*, "{API} design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[8] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique Nitesh," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[10] J. Brownlee, "SMOTE for Imbalanced Classification with Python." 2020.

[11] D. Dua and C. Graff, "UCI Machine Learning Repository." 2017.

[12] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Tech. Dig. (Applied Phys. Lab.*, 1989.

[13] P. Dutta, "pramitd-git-hub." Sep-2020. https://github.com/pramitd/FDS1_Assignment/blob/master/ionosphere_scratch.ipynb