When There is no Clear Answer; A Case Study of Classification Comparison

AMIN HOSSEINY MARANI

Additional Key Words and Phrases: classification, neural networks, support vector machine, SVM, deep learning, deep neural networks, boosting, XGboost, random forest

ACM Reference Format:
Amin Hosseiny Marani. 2018. When There is no Clear Answer; A Case Study of Classification Comparison. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2018), 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Classification performance differs from algorithm to algorithm and within different dataset; even parameter modification of a single classifier can lead to different results [8, 14]. Moreover, selecting data of a dataset to train, random initialization of models such as neural networks [15], and/or different order of data in training [4] can alter the final results.

Analyzing these modifications and differences on a single model needs a huge set of experiments. In this paper, we analyzed three different algorithms (Deep Neural Networks, Support Vector Machine, Boosting) to see how each algorithm perform with two similar dataset. Readers must be aware that dominance of an algorithm in using a dataset, or even in a one series of experiment can't be interpret as dominance in general or other specific applications of these algorithms.

To be fair, two different dataset has been applied to examine algorithms' performance. Both dataset have 10 classes (one dataset has another class of noise), but one has 13 features and other is a digit image set with more than 700 features which are pixels. The main reasons behind choosing these dataset are: 1-different dimensions bring different complexity 2-we are able to compare performance of different models on image dataset and continues-valued non-image dataset. 3-since one dataset has over 150 samples and other have over 6000, influence of different number of samples can be compared, too.

Rest of this paper is organized as follow: In next section a brief description of each model is provided. In section 3, datasets are introduced. In section 4 a comprehensive comparison is provided and different models are analyzed. A description of which model is the best as a clear answer is discussed in section 5. Finally a conclusion is provided at the end of this paper.

2 METHODS

2.1 Support Vector Machine

SVM[1] classifies samples by maximizing margin of different classes' distribution. SVM finds these maximum margins by fitting hyper planes on samples of each classes' border which is called support vectors. This margin

[1]Support Vector Machine

Author's address: Amin Hosseiny Marani, amh418@lehigh.edu.

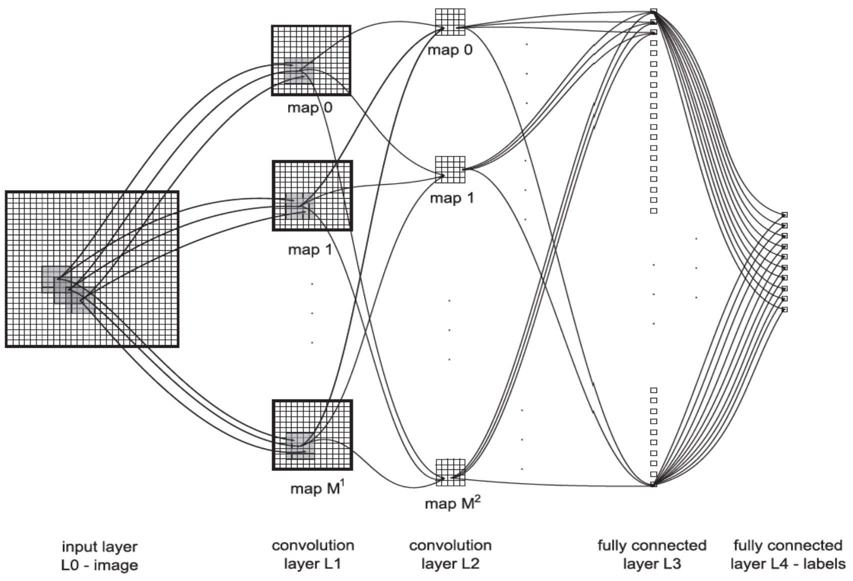


Fig. 1. A four-layer CNN.

can also be hard to exclude borders with small errors or soft to consider ϵ error in the training phase. This soft margin adds more flexibility to the model.

With the ability to define different kernels, SVM is able to show high quality classification performance. However, selecting a kernel to map the data from its original space to kernel function space is another challenge. In both dataset we analyzed in this paper, there is no linear relation between class labels and data features. Thus a linear kernel would be a wrong choice for these experiments. On the other hand, using RBF[2] and polynomial kernel with different degrees (degrees>2) could be appropriate choices. RBF kernel, allows SVM to fit multiple function with a predefined radius on the trained data. Polynomial kernel fits a curvature to the training data using a polynomial degree function, as well [2].

2.2 Deep Neural Network

Neural networks are able to fit hyper planes on feature space in order to classify data. However, a multi-layer perceptron neural network or other non-deep models are usually not able to show high quality performance on complicated datasets, especially image datasets. A deep neural network model is a chain of neural networks designed to capture low-level to high-level feature components.

More specifically a convolutional neural network has three main parts: 1-series of convolutional layer that pass a small matrix (e.g. 3*3) and capture feature blocks of layer's input. 2-a series of dense layer that connect every single input to each unit and used to fit the model on the layer's input. 3-The last layer which is usually a softmax layer, maps units' output as final results. You can see a four layered CNN[3] on figure 1 on which two first layers are convolutional, third layer is a fully connected dense layer and final layer is a softmax function [13].

[2]Radial Basis Function
[3]Convolutional Neural Networks

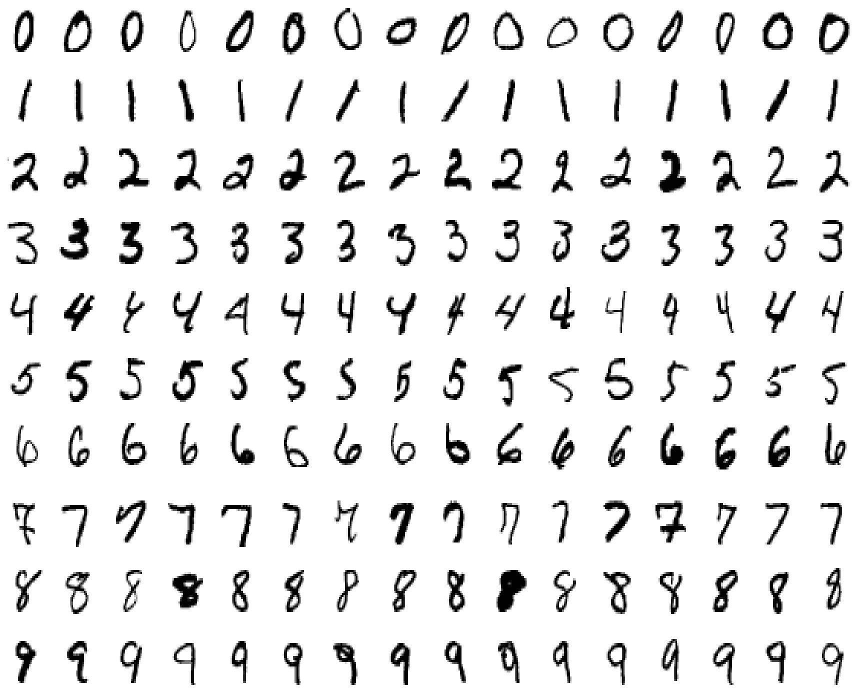


Fig. 2. Samples of MNIST hand-written digit dataset

2.3 XGboost

A single model is not pruned to mis-classification errors due to data-order, parameters selection, training procedure differences, etc. The idea of stacking multiple models together is called ensemble learning [7]. An ensemble is a collection of at least 3 machine learning models. An ensemble learning can be either a mixture of experts (high quality model with small errors) [3], or a mixture of naive models with a performance slightly better than random.

Boosting is an ensemble learning model that consist of naive models (e.g. trees with low depth) to provide a robust classifier. In a boosting model, T models are trained on random distribution of samples. Final prediction of test samples, are based on weighted voting of these T trained models which is assigned based on their performance. XGboost is a specific version of boosting in which models are optimized using Gradient Decent algorithm (Gboost) and the whole process is optimized on both software and hardware(XGboost) [5].

3 DATA

3.1 MNIST, handwritten digits

MNIST dataset consists of two set of images for train and test. It includes over 8000 images as 28*28 matrices. Each cell of matrix has a valued between 0-1 (or 0-255) which represents a gray-scale pixel of the hand written digits [6]. Figure 2 shows a series of examples of MNIST dataset.

Number of classes	samples	Samples per class	Lowest number of sample	Highest number of samples
10 (11 with noise)	517	47.00 (SD=39.51)	145 (class 2)	7 (class 5)

Table 1. Details of Iyer dataset.

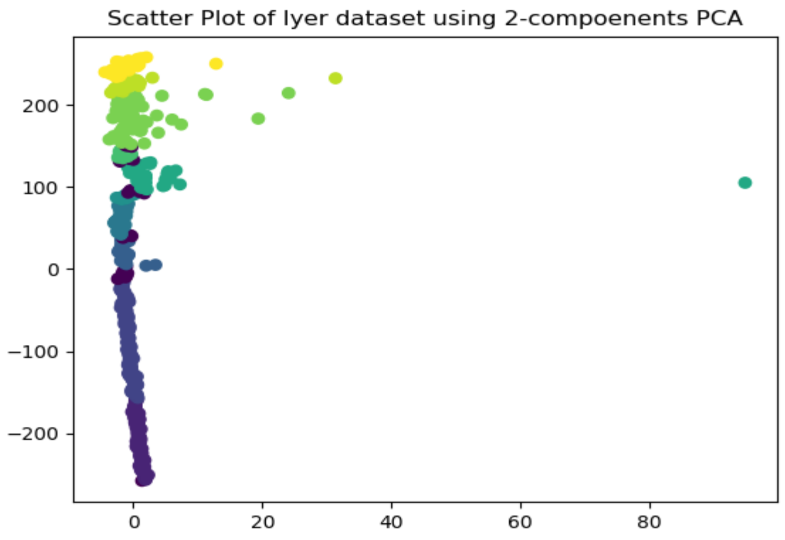


Fig. 3. Iyer dataset scatter plot in a 2D dimensional space. 13-feature space is mapped within a 2-components PCA. Each color represents a class and there are 10 classes of samples and one class of noise.

3.2 Iyer

Iyer dataset consists over 500 samples with 13 continues valued features and one label as samples' classes. Classes' labels ranges from 1 to 10 with an extra label as noise with value of -1. These noise samples can be seen either as non-classified samples or noise-classified samples based on the clustering/classification algorithm a researcher uses. Table 1 shows a brief statistical information about Iyer dataset.

Figure 3 also provides a 2D view of 11 classes (including noise) of samples using 2-components PCA of samples. As it is shown in the figure 3, some classes like the class with color yellow at the top, or purple at the bottom are easier to classify. However, data samples in between make this classification task harder for those classes. Data points that are far from main distribution (e.g. green point at the top-right) are noises.

4 EXPERIMENTAL RESULTS

In this section, we walked through different experiments to examine each model's performance to parameter selection. Later in this section multiple models are compared against each other on different datasets. Finally a dimensional reduction algorithm (PCA) is applied on each model to see, how dimension reduction can affect performance.

Dataset	Sigmoid	RBF	Polynomial (deg. 4)
MNIST	77.59	**97.92**	96.98
Iyer	47.43	76.28	**89.74**

Table 2. SVM classification accuracy over two datasets with different kernels.

4.1 SVM Results Analysis

Table 2 shows SVM results with different kernels and parameters within two datasets. There are two options one can choose for multi-class classification in the SVM model. One comparing each classifier with others (one-vs-others) and another compare each pair and select highest probable predicted class (one-vs-one). In our experiments, changing this setting did not change or slightly changed the results.

Another parameter we can change is radial basis Gamma value to control inverse radius of the fitting function. As the value gets higher, fitting functions fits on less diverse function and radius becomes smaller [1]. Changing this value, also did not improve the results over the default value $(1/(number-of-features*features-variance))$. On the other hand, different degrees for polynomial kernels changed the final accuracy from 40% to almost 90%.

In running experiments over MNIST dataset, RBF kernel overperformed other kernels with less than 3% errors. However, polynomial kernel with degree 4 was also close to RBF kernel performance. On the other hand, across Iyer dataset, polynomial kernel with degree 4 dominated other settings.

4.2 DNN Results Analysis

Designing a deep neural model is a challenge of number of layers, number of units, filter size, kernel size, activation function, etc. In this set of experiments we applied a four layered deep model with first two convolutional layers, one fully connected dense layer (all three with Rectified Linear Activation Function) and one final dense layer with softmax activation to map outputs to class labels across MNIST dataset.

Rectified linear activation function maps any value bigger than 0 to itself and any value equal and lower to 0 to zero. With this ability of linear mapping and generating true zero value, it enables a deep model to: 1-to simplify the computations instead of using Tanh and Sigmoid activation function [9]. 2-allow sparse representation since there are real 0 values (while Tanh and Sigmoid make small values near zeros) [10]. 3-show a linear behavior instead of vanishing gradient behavior [9].

Another common layer is max pooling layer. After convolutional layer provides results as (n*n) matrices, a max pooling layer gets maximum of a passing array of size (m*m) in which m can be any size but smaller than input's layer.

These convolutional and max-pooling layers extract different features of input's matrix. However, a fully connected dense layer is usually added at the end of the model to extract non-linear relationship between class labels and DNN convolutional output.

Another setting that is common in the dense layer is dropout settings. This parameter helps the model to generalizes and avoids overfitting. The value is a number [0-1] that expressed the percentage of units that are ignored during weight updates. At each iteration when the model is updating weights, a pre-set drop-out ratios of units are ignored. In our experiments best value for drop-out was 20%.

Despite of the reasons described for choosing this architecture, number of units, activiation function and the rest in these experiments, we should be aware that exploring all possible architectures and settings is almost impossible. There are dozens of methods for hyper-parameter selection such as Evolutionary Algorithm to select best possible settings for a DNN [12].

Figure 4 shows number of units selection and its influence in the final results. First two numbers for each model in this figure, show convolution kernel size, and third number is the number of units of the dense layer.

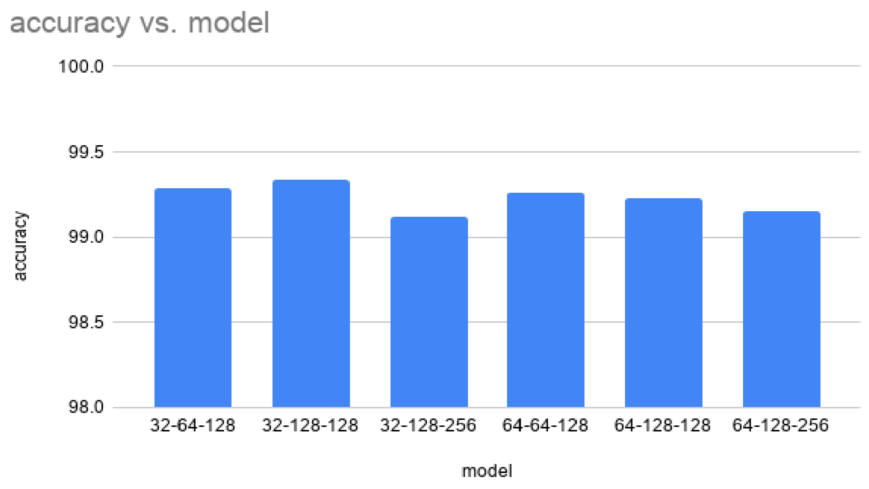
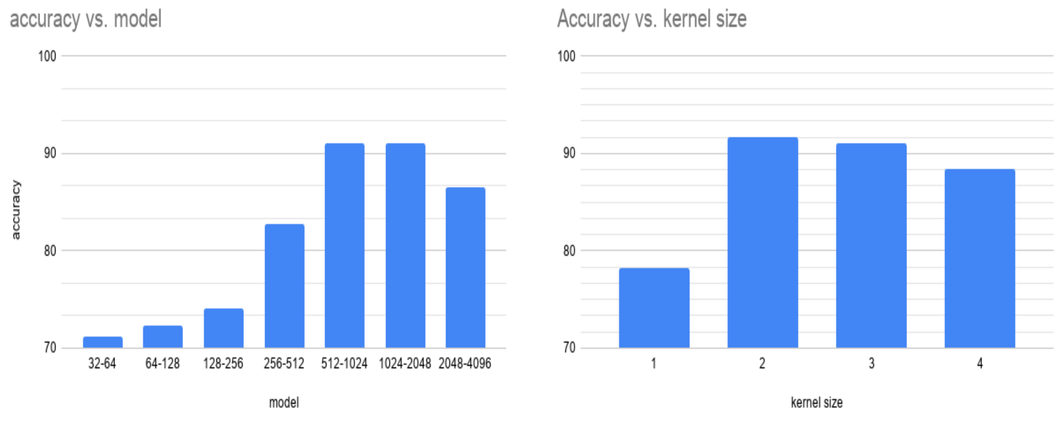


Fig. 4. Units' number and kernel size comparison of CNN across MNIST dataset.



(a) Accuracy of the CNN model with different units (b) Accuracy of the CNN model with different kernel size

Fig. 5. Accuracy comparison of different CNN models with different settings across Iyer dataset.

DNN model is able to classify MNIST dataset with error of less than 1%, and changing kernel size or number of units in the dense layer does not change the final result much.

Unlike MNIST dataset, Iyer dataset is a one dimension data. With that in mind, a 1D convolutional layer with ReLU[4] activation function (for MNIST 2D conv layer was applied) and a one dense layer with ReLU activation function and a final dense output layer (with softmax activation function) is applied. Figure5 shows different settings comparison of CNN model for the iyer dataset.

[4]Rectified Linear Unit

Dataset	Accuracy	AUC	Recall	Precision	F1
MNIST	96.15	97.53	95.45	94.11	94.08
Iyer	97.80	98.76	97.78	97.79	97.78

Table 3. XGBoost classification performance across two datasets

In the CNN models designed for Iyer dataset classification, number of units in both first layers are much more higher than MNIST dataset. And as it increase the accuracy of the model gets higher as well. However, using more than 1024 kernels in first layer and 2048 units in the dense layer brings less accuracy. Figure 5.b also shows kernel size of 2 or 3 are appropriate choices for first layer, and other value higher or lower than these values lead the model to more miss-classification error.

4.3 XGboost Results Analysis

Xgboost were applied to both datasets using Python XGboost library. There are multiple settings to set such as type of trees, max tree depth, weight regularization, etc. In our experiments changing default parameters did not affect classification performance. Table3 shows XGboost performance over different datasets. From this table, we can see XGBoost had a more balanced classification w.r.t recall, precision, and F1 metrics. Further comparisons of this model with two others are provided within next section.

4.4 Dimension Reduction and Ensemble Learning

As we discussed in section 2.3 about ensemble learning, we also used two methods of majority vote and weighted voting using CNN, XGBoost, and SVM outputs. Majority vote takes three models' outputs and label the sample with maximum vote. On the other hand, weighted voting places importance on each model based on their performance (i.e accuracy) and takes the voting. Figure 6 and 7 compared all three models with two ensemble models. These resutls are average of 5 runs for each model. Black line on each chart shows variance of the accuracy. As you can see, only deep neural network changes over different runs because of random initializing sensitivity. Because of that, Ensemble learning also has small variance as an influence of DNN outputs.

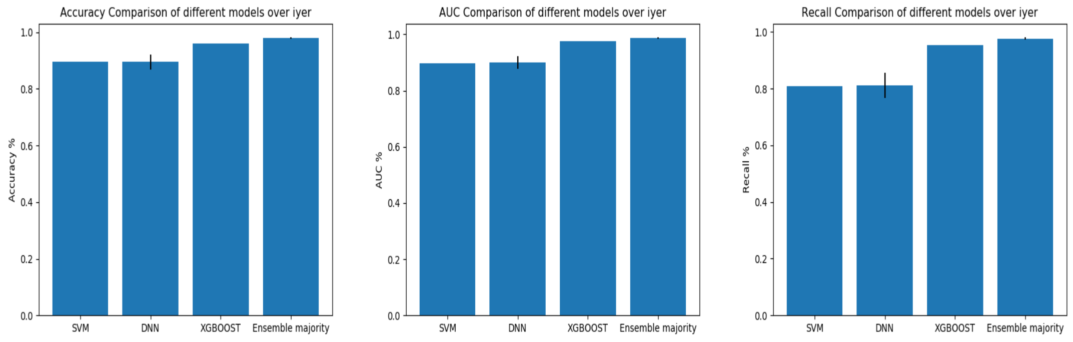
Ensemble learning with majority vote in figure 6 shows highest performance and close to XGBosst. On the other hand deep neural network performs slightly better than other models across MNIST dataset as shown in figure 7. Perhaps, since SVM and XGBoost have different outputs than DNN on the miss-classified ones, the ensemble learning are not able to outperform DNN.

Principal Component Analysis[5] is a reduction/combination dimension algorithm that maximizes variance or correlation on new feature set [16]. It has been shown PCA not only can reduce number of features, but also can increase classification performance by providing more valuable features [11]. In MNIST or any other image datasets with hundreds, thousands or even millions of features, reducing the dimensional and increasing performance or avoiding to loose performance can save time and cost.

We applied PCA with different number of components on both Iyer and MNIST dataset and re-trained three models to examine dimension reduction on performance. Since, there are 12 different values for Iyer as PCA components and more than 700 possible components for MNIST, we don't go over comparing different number and instead compared the best settings which is provided in figure 8 and 9. Each model showed its highest performance with different number of components and no single value is obtained for one dataset across different models.

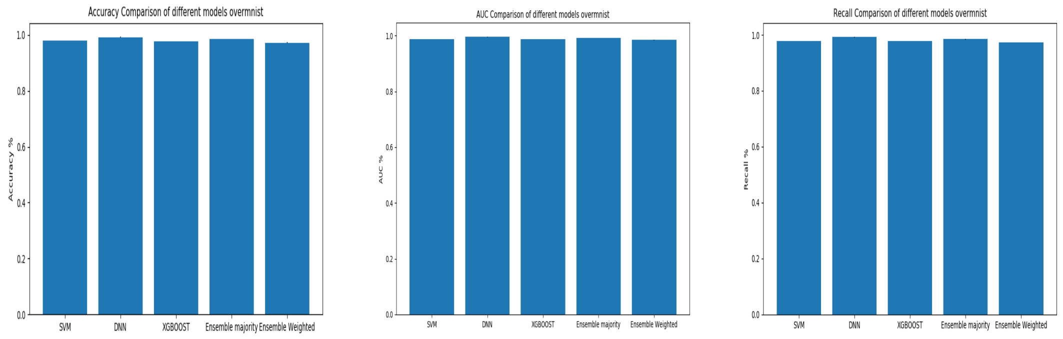
XGBoost performed much better than two other algorithms and also majority vote (figure 8) across Iyer dataset. It has less than 1% miss-classification error. With PCA applied to the dataset DNN performance also increased,

[5]PCA



(a) Accuracy comparison (b) Area Under Curve Comparison (c) Recall Comparison

Fig. 6. Comparison of different algorithms on Iyer dataset. Black line over bar charts shows variance of models' performance.



(a) Accuracy comparison (b) Area Under Curve Comparison (c) Recall Comparison

Fig. 7. Comparison of different algorithms on MNIST dataset. Black line over bar charts shows variance of models' performance.

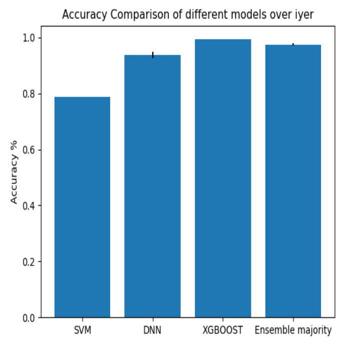
but interestingly SVM performance decreased. Perhaps because, SVM cannot classify data with a polynomial or RBF kernel on a more complicated feature set.

In classyfiying MNIST dataset, SVM performed almost similar to DNN (best performance) across MNIST with PCA dimension reduction. While 50 components works better for both XGBoost and DNN, 70 components gets highest accuracy of SVM. Interestingly, XGBoost performs the worst unlike its performance across Iyer dataset.

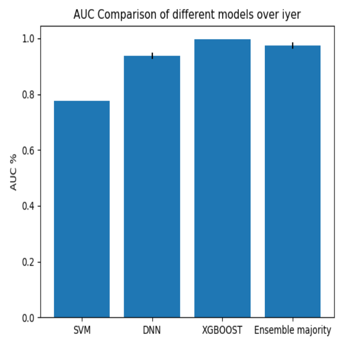
5 DISCUSSION AND CONCLUSION

5.1 Data Features

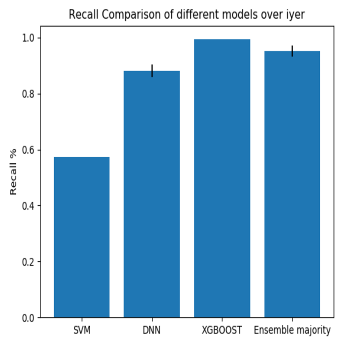
As we analyzed and compared models with and without PCA, we can now say dimension reduction is a boost to classification. However, in lots of cases using PCA won't improve classification. For example in image classification with RGB images, combining features of different color plates perhaps won't produce meaningful and correlated feature set. Hopefully it was not the case in our experiment.



(a) Accuracy comparison

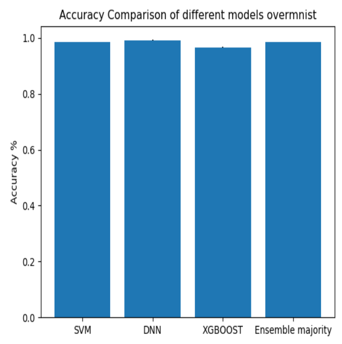


(b) Area Under Curve Comparison

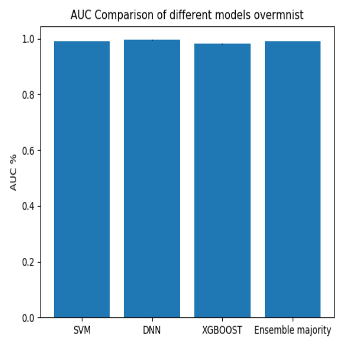


(c) Recall Comparison

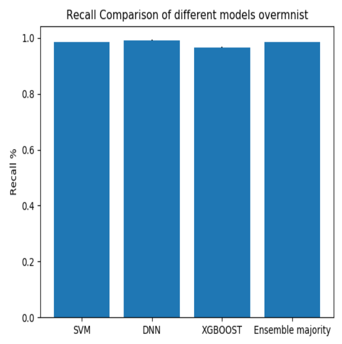
Fig. 8. Comparison of applying PCA on different algorithms across Iyer dataset.



(a) Accuracy comparison



(b) Area Under Curve Comparison



(c) Recall Comparison

Fig. 9. Comparison of applying PCA on different algorithms across MNIST dataset.

As much as PCA can improve a model's performance, it can rise new challenges for best number of components. For example, in genetic data with thousands or even millions of features, we can't re-run the experiments thousands of times, just to select the best number of components. On the other hand, if PCA improve classification, why not just pick a number without exploring all possible numbers.

Feature selection, factor analysis, etc. are other methods to select a subset of features or generate a new set of features in order to improve classification performance. After all, we can't tell if PCA or any other method can improve any problem and any model (e.g. in our experiments, SVM lost performance after applying PCA). So what is the best way to represent features? Perhaps the best answer is there is no such way. And the only way we can figour this out for each model is understanding the data distribution, model's weaknesses, and strengths.

5.2 Different Datasets

Comparing different algorithm's performance on different datasets in section 4 reveals that no single model is dominant in classification. One model may perform much better on a dataset, while it gets the highest miss-classification error across other datasets. Based on data distribution and data attributes a model should be selected. For example, with the ability to extract low to high level features a CNN can be among best options as we saw in MNIST classification. However, for a model with less matrix location importance such as Iyer, especially when the data are not sparse, perhaps a deep model is among the last choices. Understanding data set and feature representation is another key to select the best model for each different task.

5.3 Accuracy is not Everything

Despite accuracy, speed is another factor one should take into account. For example in MNIST datset with PCA, SVM performance is close to DNN while the training phase on the SVM is more than 10 times faster than DNN and XGBoost. What if instead of 6000 samples in MNIST, we have six millions? What if the images are HD and therefore we have millions of features to train? in each case, we should consider the importance of accuracy vs. speed, or even computational facilities we have.

Generality is another key for selecting a model over other algorithms. Once the training is finished, is the model able to generalize the output to new incoming data. Although, we consider test and evaluation set to avoid overfitting, but we can never avoid future miss-classification due to unpredictability of new samples. One should be aware of model's generality over different datasets, before making a decision.

Last but not least, ease of use for a model is crucial to the users. Not every single user is an expert of machine learning. They also want to apply their data to different models and see which one performs better. With just calling a function like SVM or XGBoost, or requirement to know at least about deep models' layers, one may choose simpler models.

REFERENCES

[1] Rami Albatal and Suzanne Little. 2014. Empirical exploration of extreme SVM-RBF parameter values for visual object classification. In *International Conference on Multimedia Modeling*. Springer, 299–306.

[2] Shun-ichi Amari and Si Wu. 1999. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks* 12, 6 (1999), 783–789.

[3] Ran Avnimelech and Nathan Intrator. 1999. Boosted mixture of experts: an ensemble learning scheme. *Neural computation* 11, 2 (1999), 483–497.

[4] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 15–26.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[6] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[7] Thomas G Dietterich et al. 2002. Ensemble learning. *The handbook of brain theory and neural networks* 2 (2002), 110–125.

[8] Holger Frohlich and Andreas Zell. 2005. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 3. IEEE, 1431–1436.

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[11] Kuang Liu, Mingmin Zhang, and Zhigeng Pan. 2016. Facial expression recognition with CNN ensemble. In *2016 international conference on cyberworlds (CW)*. IEEE, 163–166.

[12] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. 2017. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *Proceedings of the genetic and evolutionary computation conference*. 481–488.

[13] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*. Springer, 52–59.

[14] Rhonda D Phillips, Layne T Watson, and Randolph H Wynne. 2007. Hybrid image classification and parameter selection using a shared memory parallel algorithm. *Computers & Geosciences* 33, 7 (2007), 875–897.

[15] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. 1139–1147.

[16] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.