Assignment 1: Supervised Learning

CS7641 Machine Learning, 2019 Fall

Hui Xia (hxia40)
903459648
Georgia Institute of Technology

1. Introduction

Supervised learning is a major category in machine learning. The target of this assignment is to gain understanding of commonly used algorithms under various circumstances. In this assignment, two datasets are analyzed using five algorithms: Decision trees (DT) with some form of pruning, Artificial Neural networks (ANN), Boosting, Support Vector Machines (SVM), and k-nearest neighbors (kNN), as per required. In this report, I will firstly describe the dataset used, then describe and discuss the learning curve analysis and model complexity analysis performed on these datasets, using the said algorithms.

2. Experiment Methods

2.1. Datasets

In this work, two datasets of similar size are analyzed. One dataset is a slice from Fashion-MNIST. Each row of this dataset describes a 28 x 28 grayscale pixel image (each pixel is represented with a value of 0-255), thus, the training and test datasets have (28 x 28) = 784 columns, i.e. X1-X784 [1]. Each row is also associate with a label y (value of 0-9) that respectively represents different 10 kinds of clothing. All of the y values have equal proportion in this dataset. For this assignment, the first 2000 rows in the train dataset, and the first 500 rows in the test dataset are used. The total size of the training and testing dataset is 5.5 MB.

Another dataset is the Epileptic Seizure Recognition Data Set (ESR) from UCI Machine Learning Repository [2]. This dataset is collected as the recording of brain activity via electric signals. Each row of the dataset describes 178 data points (i.e. X1-X178) happened within 1 second in a person's brain. Each row is associated with a label y that represent 5 different status (eyes open, eyes closed, healthy area, tumor area, and seizure) of the person tested. All of the five kinds of y status have equal proportion in this dataset. For this assignment, the first 6000 rows in the dataset are split into a 5000-row train and 1000-row test dataset. The total size of the training and testing dataset is 5.5 MB.

I found that there are several interesting points among these two datasets. First, as MNIST is an image-based database while ESR is not, it is interesting to find if more complex model, such as ANN, will perform better (compared with other algorithms) in MNIST compared with in ESR. Second, the ESR dataset is not only of my personal research interest (I hold a Ph.D. in biomedical engineering), but also a 'weak' classification dataset that is well-known as hard to gain good performance. It will be interesting to find out that if I can find out a way to build a high-performance algorithm. Third, as described above, both of the datasets are used for a classification problem, and I purposely sliced them into same storage size. This fact itself make the compare between the two datasets interesting - not only on algorithm performance, but also on running time.

2.2. Experiment design

The training set from both of the datasets are subjected to a classification study against their respective y values. In details, a learning curve study is firstly performed on the datasets using five algorithms (decision tree, boosting using decision tree, neural networks, support vector machines, and k-nearest neighbor) under default parameters. The classifier and the related default hyper-parameters are listed in **Table 1**. In the learning curve study, a cross-validation is performed using 10-section `ShuffleSplit`. The training score and cross-validation score is plotted against 50 different sample sizes (from 2% to 100% of the size of the training set). For the iterative algorithms (i.e. boosting, ANN, and SVM), the learning curve study is also performed as plotting the training score and cross-validation score against iterations. The iterations of boosting and SVM are controlled using their

respective hyperparameters `n_estimators` and `max_iter`, while the iterations of ANN is realized using its `partial_fit` method.

Table 1. Classifier and default hyperparameters adopted in first learning curve study.

Algorithm	Classifier	Hyperparameter 1		Hyperparameter 2		Iterator	
		name	default value	name	default value	name	default value
Decision tree	sklearn.tree. DecisionTreeClassifier	*max_depth*	*None*	*min_samples_leaf*	*25*		
Boosting using decision tree	sklearn.ensemble. AdaBoostClassifier (base_estimator= sklearn.tree. DecisionTreeClassifier)	*min_samples_leaf*	*25*	*learning_rate*	*1*	*n_estimators*	*5*
Artificial neural network	sklearn.neural_network. MLPClassifier	*hidden_layer_size*	*(5,)*	*alpha*	*0.0001*	*partial_fit*	*N/A*
k-nearest neighbors	sklearn.neighbors. KNeighborsClassifier	*n_neighbors*	*5*	*algorithm*	*'auto'*		
Support vector machines	sklearn.svm.SVC	*C*	*1*	*kernel*	*'rbf'*	*max_iter*	*-1*

Then, a model complexity analysis is performed to the two datasets using the said five algorithms. In the model complexity analysis, while keeping all the other hyperparameters at default, one of the hyperparameters listed in **Table 1** are changed, and the resulted classification performance score, for both the training set and cross-validation set, is plotted against the changing hyperparameter. For each hyperparameter, 50 different values are tested on the training and the validation set to find the optimal hyperparameter setting. The value of the hyperparameter that grant the maximum cross-validation score is adopted as optimal. Similar to the learning curve study, the cross-validation is performed for the model complexity analysis using 10-section shuffle-split.

Using the optimal hyperparameters, another set of learning curve study is performed on the datasets using the five algorithms. The training score and cross-validation score is plotted against 50 different sample sizes (from 2% to 100% of the size of the training set), same as the first set of learning curve study. The learning curve plot generated before and after the model complexity analysis is compared, to find the effectiveness of the hyperparameter optimizing on the performance of the respective algorithm. Finally, the running time and the performance score of the five algorithms (using the optimal hyperparameters) on the testing dataset is measured to investigate inter-algorithm difference.

3. Results and Discussion

3.1. Learning curve using default hyperparameter

The learning curve plot using said algorithms on datasets MNIST and ESR against sample size and against iteration are shown in **Figure 1** and **Figure 2**, respectively**.** As both datasets are multilabel classification problems, accuracy score is plotted on the y-axis to describe the performance of the algorithms. It is notable that because the training and validation data are subjected to cross-validation and treated using `ShuffleSplit`, when the training size is very small, the over-fitting (which is due to the that model fit training set well while not generalizing to validation set) are avoided for the “simpler” algorithms (e.g. decision tree, kNN, and boosting over decision tree).

Using decision tree and the boosting-decision tree algorithms, the training and cross-validation score for both MNIST and ESR datasets plateaus alone the increase of sample size (**Fig. 1a-d)**, indicating that the addition of more datasets will not significantly improve their performance. As the training/cross-validation curve does not show the trait to converging over the growth of sample size, the model could be suffering from a high

variance problem. Also, as the training curve is far from 1.0, the models could be also suffering from a high bias problem. As it is premature to judge which side should we move in the bias-variance trade-off, it would be reasonable to tune the `min_samples_leaf` over a range between 1 and 200 to show if the pre-pruning of the decision tree by increasing or decreasing the complexity could improve their performance. Additionally, when the model takes more than 40 iterations (i.e. when `n_estimators` >= 40, shown in **Fig. 2a-b**), the large train/validation gap and good train set performance indicate that the boosting model suffers from a high-variance and low-bias problem. Decreasing model complexity (e.g. increase leaf size and/or decrease learning rate) could improve the performance thereof.

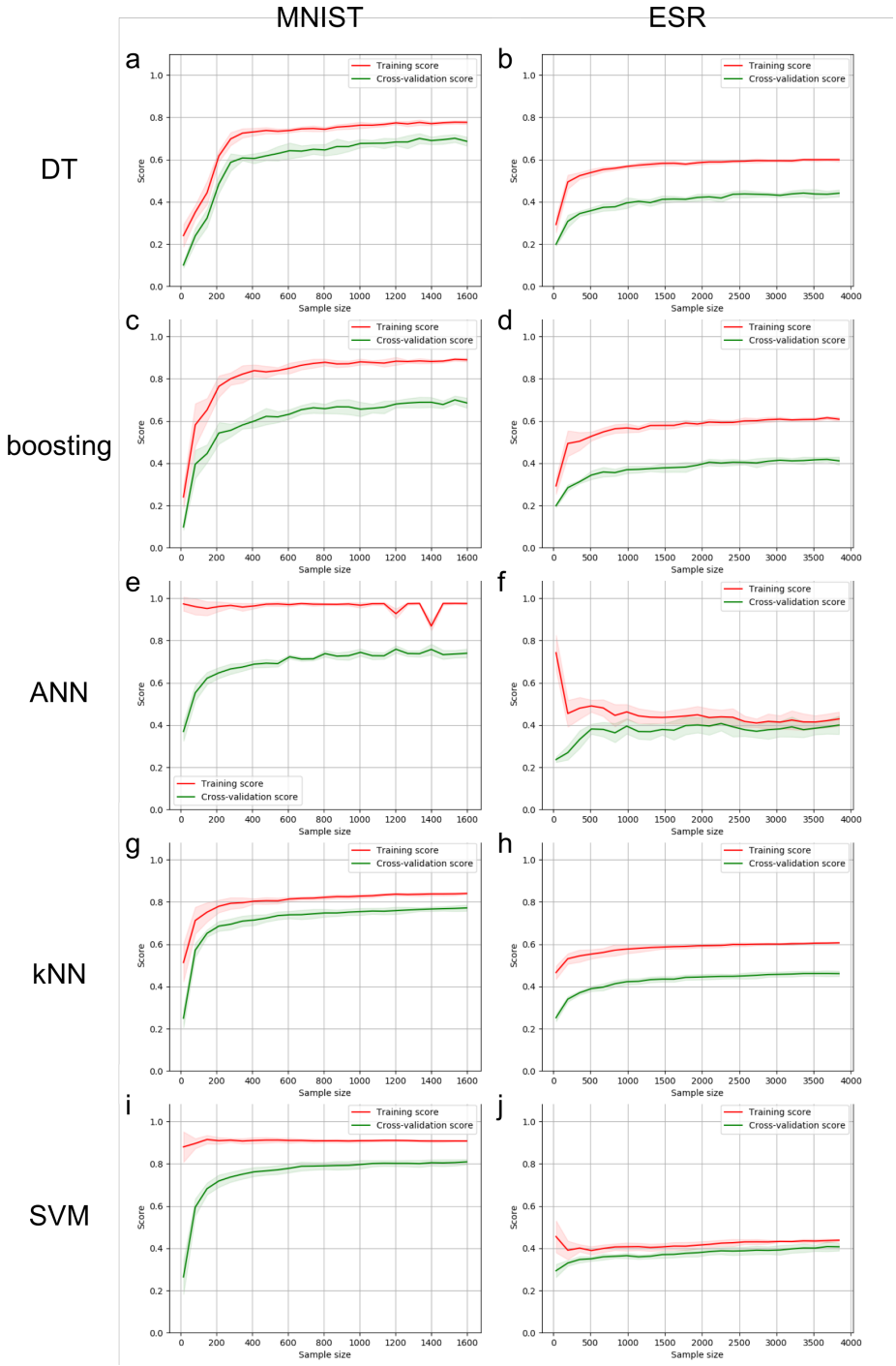

Figure 1: Learning curve over sample size using default hyperparameter

Interestingly, using ANN and SVM, the training and cross-validation score for MNIST and ESR performs differently. On MNIST, the ANN and the SVM algorithms show a typical low bias and high variance problem, features by the large gap between the train/validation curves and high performance on the training curve (**Fig. 1e and i**). Quite on the opposite side however, for ESR, the ANN and the SVM algorithms show a typical high bias and low variance problem, which is features by the small gap between the train/validation curves and poor performance on the training curve (**Fig. 1f and j**). Along the increase of sample size, the training and cross-validation score for both MNIST and ESR datasets plateaus, indicating that the addition of more datasets will not significantly improve their performance. Additionally, along the increase of iteration times, both ANN and SVM show low-bias and low-variance for MNIST, and high-bias and high-variance for ESR, indicating that the increase of iterations (when epochs >= 15 for ANN and `max_iter` >=60 for SVM) can only improve the performance on both datasets in a limited manner. Thus, I will target to bring the model complexity down for the MNIST dataset, and up for the ESR.

For both MNIST and ESR, the learning curve using kNN looks very similar to that using decision tree (**Fig. 1g-h**). That is, the training and cross-validation score for both datasets plateaus along the increase of sample size, and it is hard to decide if the model is suffering from high-bias or high-variance problem. Thus, I will tune the `n_neighbors` to investigate for a better performance on both directions from the default value of `n_neighbors = 5`.

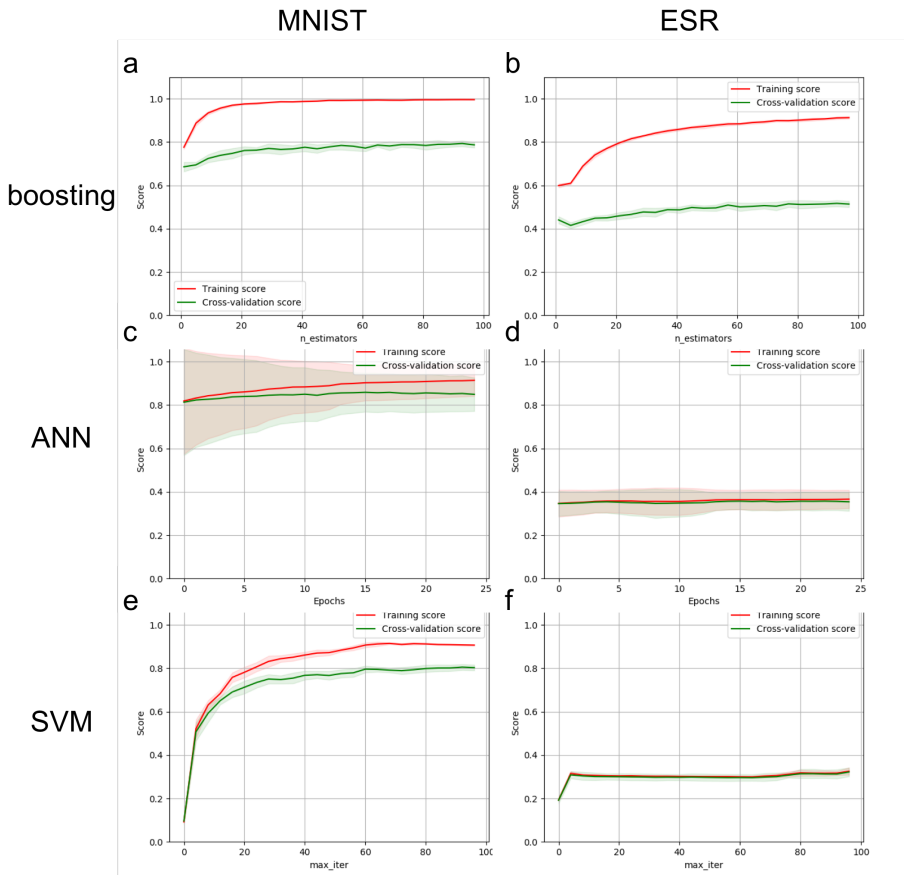

Figure 2: Learning curve over iteration using default hyperparameter

3.2. Model complexity analysis

As discussed in **Sections 2.2** and **3.1**, while keeping all the other hyperparameters (or iterators, if applicable) at default, one of the hyperparameters listed in **Table 1** is changed and plotted against the changing hyperparameter. **Figure 3** demonstrate the plotted validation curve for all the five algorithms using MNIST and ESR datasets.

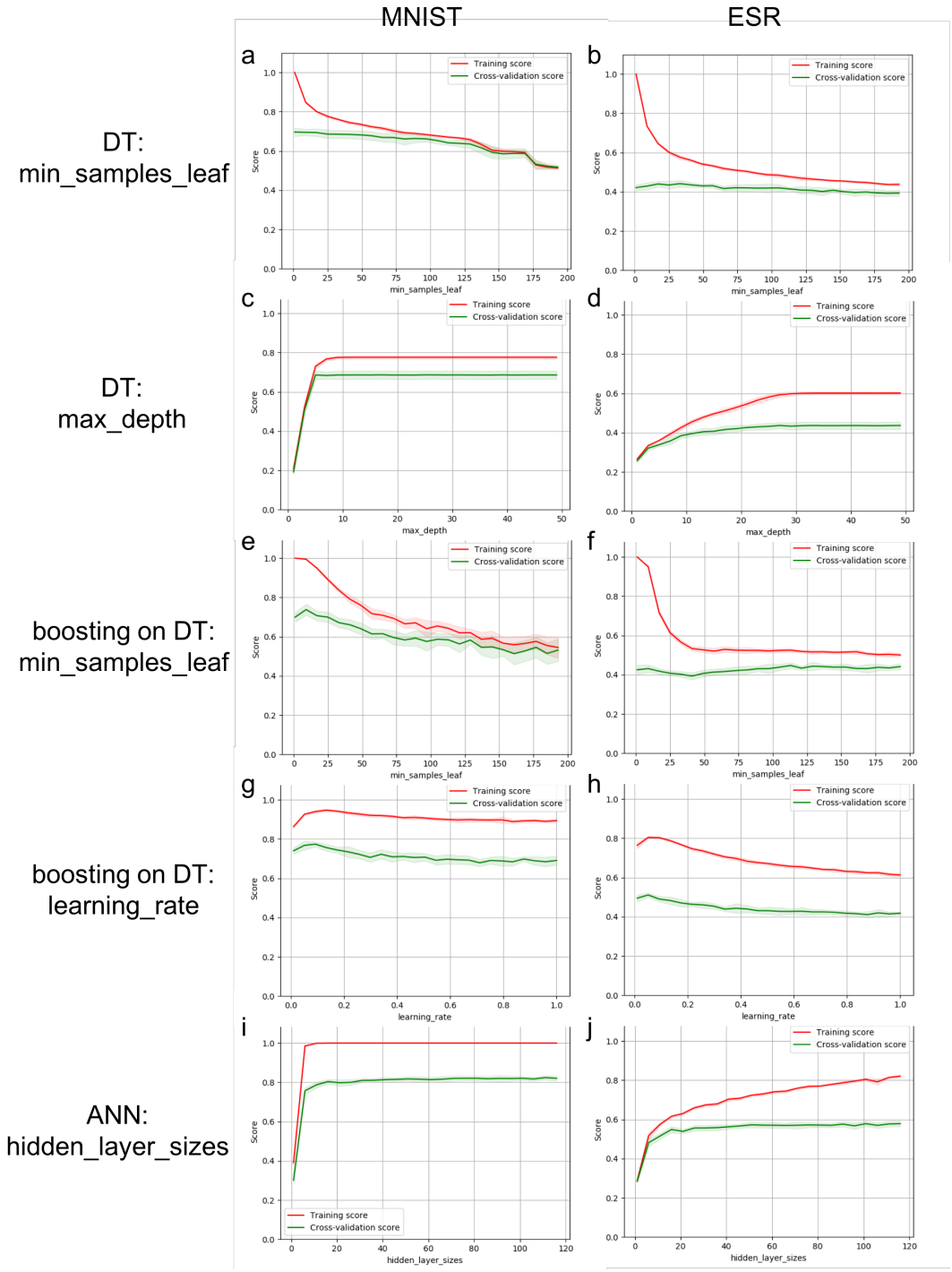

Figure 3: Model complexity analysis

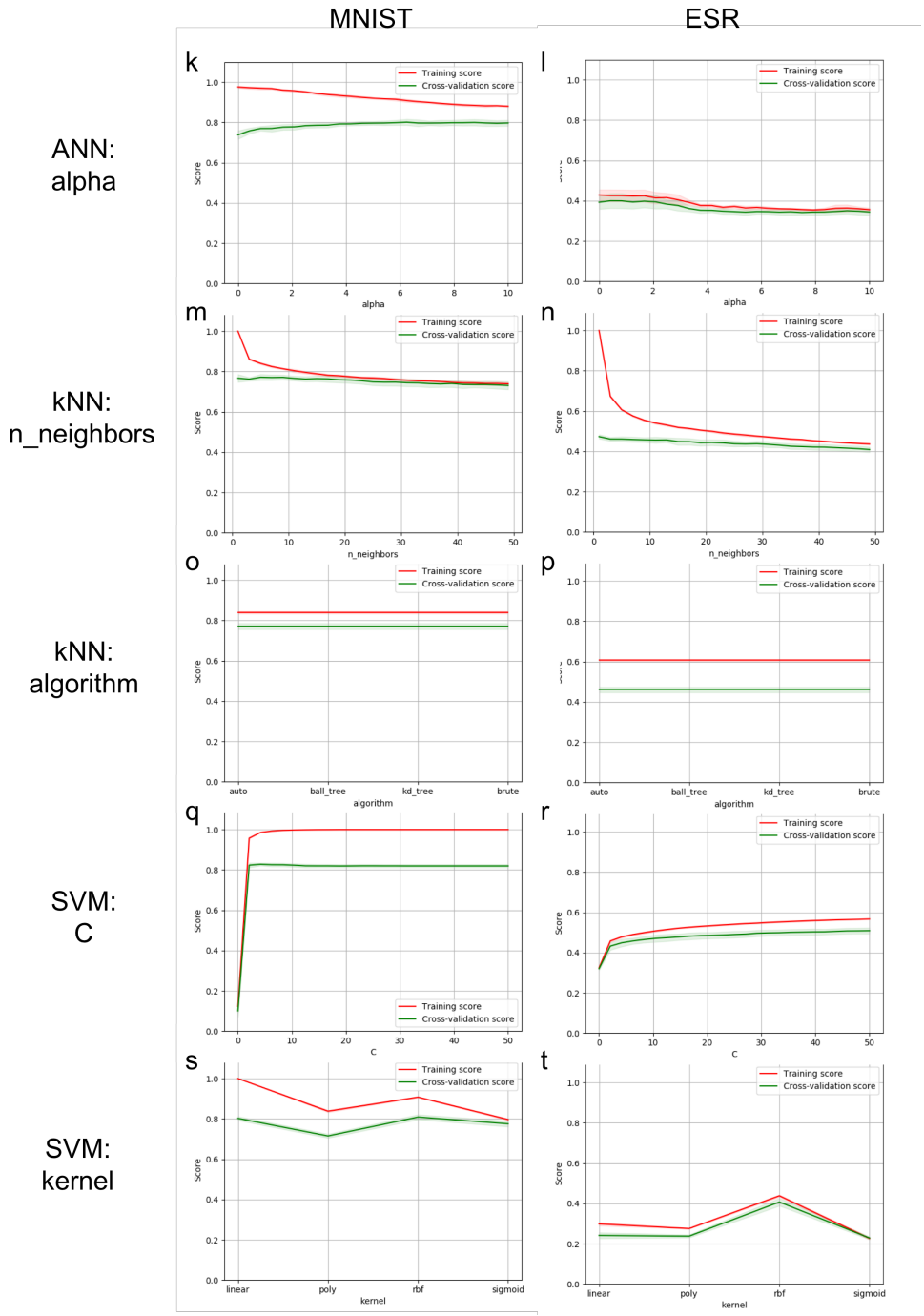

Figure 3 (continued): Model complexity analysis

As shown in **Figure 3a-b**, pre-pruning of decision tree using `min_samples_leaf` on both of the datasets, increasing `min_samples_leaf` decrease the decision tree model's variance and increase the bias thereof. This is reasonable, as increasing the size of leaf will make the model less complex, thus making the model to take a more generalized point of view (i.e. to be more biased). Additionally, the performance of the model on MNIST decreases alone the increase of leaf size, but the performance of the model on ESR peaks when `min_samples_leaf` = 33. On the other hand, the pre-pruning of decision tree using `max_depth` show that higher value of `max_depth` will lead to better model performance (**Fig. 3c-d**). Thus, I will take `min_samples_leaf` = 1 and `max_depth` = None for MNIST, and `min_samples_leaf` = 33 and `max_depth` = None for ESR as the optimal.

Boosting using decision tree show a similar trend when adjusting on `min_samples_leaf` on both of the datasets (**Fig. 3e-f**). Again, increasing leaf size decrease the decision tree model's variance and increase the bias. It is also interesting to find that the ESR dataset always performs better using larger leaf size compared with the MNIST dataset, using both decision tree model and the boosted decision tree model. This is quite counterintuitive, as we would consider the MNIST, as an image-recognizing dataset, would favor a more "general", or more biased model, just like we consider human use their 'feelings' on certain features to recognize photos. However, we see that the image-based MNIST dataset require a more complex model to perform well.

Additionally, the performance of the model on MNIST and on ESR peaks when their `min_samples_leaf` = 9 and = 113, respectively. Adjusting on learning_rate shows a trend similar to adjusting the `min_samples_leaf`. Alone the increase of learning_rate, the performance of the model firstly briefly increases, then decreases (**Fig. 3 g-h**). As a result, the optimal parameter `learning_rate` can be found at 0.0925 and 0.05125, for the MNIST and the ESR datasets, respectively. When the number of iterations of the boosting model stays constant, a small learning rate will make the model to underfit, or has higher bias [3]. The high performance using low learning rate indicate that the boosting model on both datasets prefer to bring up the bias, which agrees with our hypothesis in **Section 3.1**.

From the learning curve study, we mentioned that the ANN and the SVM algorithms for MNIST show a low bias and high variance problem, while for ESR, a high bias and low variance problem. Thus, here we strive to bring the bias up for the MNIST dataset, and to bring the bias down for the ESR. This can be achieved by adjusting the hyperparameter `alpha` from ANN, and `C` from SVM. Lower `alpha` values tend to overfit, or be low on bias but high on variance, and *vice versa* [4]. On the opposite, lower `C` value tend to underfit, or be low on variance buy high on bias, and *vice versa* [5], which is similar to hyperparameter `learning_rate` used in the boosting model. As shown in **Figure 3k-l and q-r**, MNIST prefer higher alpha in ANN but lower C in SVM, while ESR prefer lower alpha in ANN buy higher C in SVM. Again, this observation agrees with our hypothesis (that we need to bring the bias up for MNIST, and down for ESR) here. While in the measurable range (up to `C` = 1000), the performance of SVM on ESR monotonically increase with higher `C` (Fig. 3r). However, choosing higher `C` will also bring up the calculation time. Thus, for ESR, `C` = 50 is used here as optimal. Similarly, for both datasets on ANN, higher `hidden_layer_size` will result in better performance but longer time consumption (**Fig. 3i-j**). Thus, `hidden_layer_size` = (50,) is used.

Finally, on KNN model, the hyperparameter `algorithm` does not affect the model's performance. On the other hand, `n_neighbors` peaks on the value of 5 and 1, respectively, for the datasets MNIST and ESR. This indicate that overfitting (i.e. a high variance and low bias) strategy performs better for the kNN model for both of the datasets. In summary, the optimal hyperparameters that will be used for the MNIST and ESR datasets are listed in **Table 2**.

Table 2. Optimal hyperparameters adopted in second learning curve study.

Algorithm	Hyperparameter 1		Hyperparameter 2		Iterator	
	name	default value	name	default value	name	default value
MNIST						
Decision tree	*max_depth*	*None*	*min_samples_leaf*	*1*		
Boosting using decision tree	*min_samples_leaf*	*9*	*learning_rate*	*0.0925*	*n_estimators*	*40*
Artificial neural network	*hidden_layer_size*	*(50,)*	*alpha*	*6.25*	*partial_fit*	*N/A*
k-nearest neighbors	*n_neighbors*	*5*	*algorithm*	*'auto'*		
Support vector machines	*C*	*0.418*	*kernel*	*'rbf'*	*max_iter*	*-1*
ESR						
Decision tree	*max_depth*	*None*	*min_samples_leaf*	*33*		
Boosting using decision tree	*min_samples_leaf*	*113*	*learning_rate*	*0.5125*	*n_estimators*	*40*
Artificial neural network	*hidden_layer_size*	*(50,)*	*alpha*	*0.417*	*partial_fit*	*N/A*
k-nearest neighbors	*n_neighbors*	*1*	*algorithm*	*'auto'*		
Support vector machines	*C*	*50*	*kernel*	*'rbf'*	*max_iter*	*-1*

3.3. Learning curve using optimal hyperparameter

Using the hyperparameters listed in **Table 2**, another set of learning curve study is performed to examine the effectiveness of the hyperparameters tuning described in last section. The learning curve plot using said algorithms on datasets MNIST and ESR against sample size and against iteration are shown in **Figure 4** and **Figure 5**, respectively**.** Compared with the learning curves using default hyperparameters, significant improvement can be observed on the boosting, SVM, and ANN model on both of the datasets (**Fig. 4c-f and i-j**), indicating that our hypothesis was reasonable, and the associated hyperparameter tuning was effective. However, for the rest of the model-dataset combination, the performance on the validation curve stay rather unchanged after hyperparameter tuning. In this section I will discuss why the expected performance improvement did not happen.

I adjusted the leaf size from the default of 25 to 1 for MNIST, and to 33 for ESR, which means I tried to overfit and underfit these datasets, respectively. However, neither of these actions improved the performance of the validation curve significantly. The only significant difference compared **Figure 4a** with **Figure 1a** is that, the training curve in **Figure 4a** become a straight line of y = 1, indicating sever overfitting. Similar scenario also happens on the adjustment of `n_neighbors` in the kNN model. Adjusting the k value toward neither (higher bias or higher variance) direction could significantly improve the model performance. For this time, after adjusting `n_neighbors` to the value of 1,the training curve in **Figure 4h** become a straight line of y = 1, indicating sever overfitting. As both increasing or decreasing model complexity did not improve the model performance, I would conclude that irreducible error plays a major role that bring the performance of the decision tree and the kNN model down.

It is also interesting to observe that that the standard deviation of the ANN plots (showed area along the training and validation curves in **Fig. 2c-d**) decreased significantly after hyperparameter optimization (**Fig. 5c-d**). I reason this observation is due to that the number of neurons in the hidden layer from the default (which equals to 5) was too small – the hidden neuron of 50 in the optimal value fits both of the datasets better, as it has been

reported that the number of neurons in that layer is the mean of the neurons in the input and output layers [6][7]. The number of neurons to the output layer equal to 1 for both of the datasets, as they are classifier. The number of neurons to the input layer d=equal to the number of features, which are 784 and 178 respectively for MNIST and ESR. Thus, 50 will be a more proper hidden neuron size compared with 5.

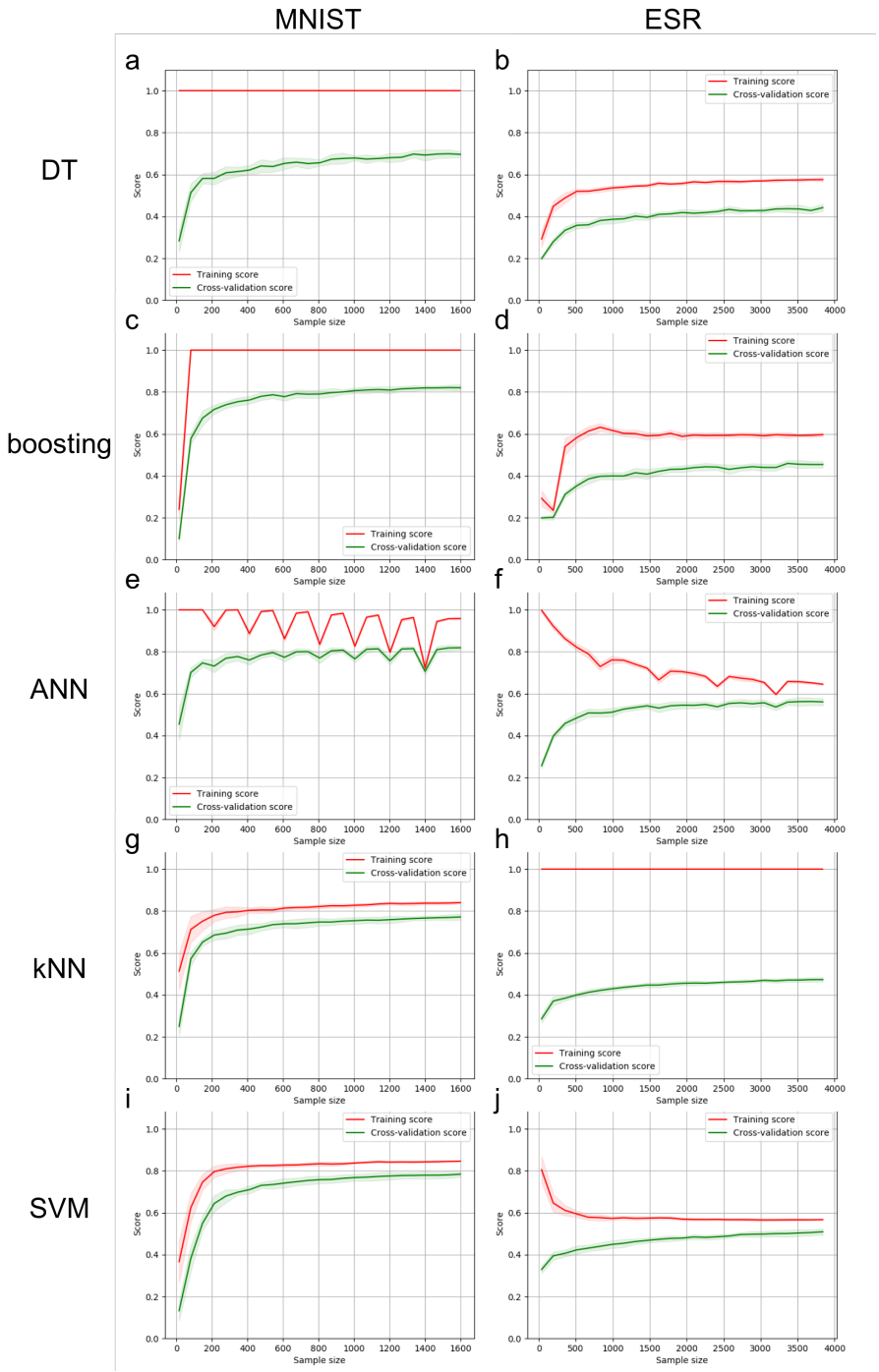

Figure 4: Learning curve over sample size using optimal hyperparameter

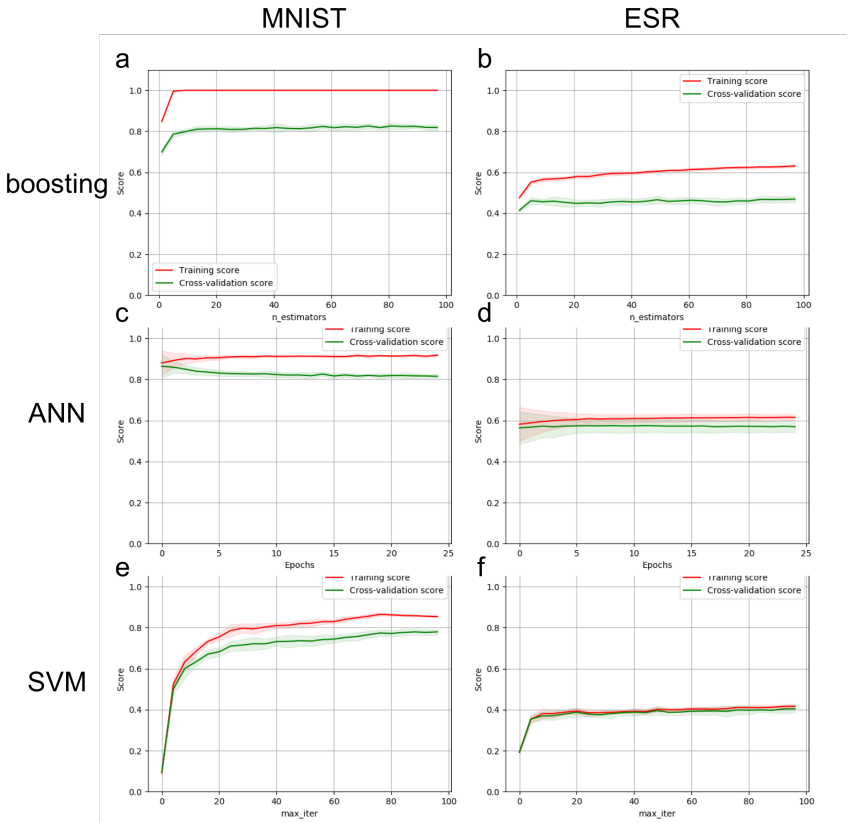

Figure 5: Learning curve over iteration using optimal hyperparameter

3.3. Inter-model comparison

Using the optimal hyperparameter, the trained models are tested using the testing dataset. The resulted performance score and training/testing time is shown in **Table 3**. **Table 3** also includes the training/testing time of the five algorithms using the default hyperparameters, for a better comparison between the two datasets. Each of the data point shown is the average of 10 trials. For both of the datasets, ANN has obtained significantly better performance compared with other models. The performance superiority of ANN is more pronounced when predicting the ESR dataset, where the performance of all models is in general worse. The training/testing time for both datasets using default hyperparameter is in similar range.

I reason that this is due to the superior complexity of ANN. As we have discussed in **Sections 3.1** and **3.2**, based on the learning curve analysis using default hyperparameter, we tried to bring bias up for both datasets under the boosting model, and to bring the bias up for MNIST and down for ESR, under the ANN and SVM models. In **Section 3.3**, the learning curve analysis showed that all the above-said bias adjustment successfully improved the model performance. This observation indicate that compared with MNIST, ESR needs a more complex (at least ANN and SVM) model to perform well. Thus, as the most complex algorithm, ANN will have a more pronounced superiority when handling ESR. Similarly, the simple algorithms (i.e. decision tree and kNN) performs worse when predicting the model complex dataset ESR, compared with their “acceptable” performance when predicting MNIST.

Another interesting observation is, compared with other iterative algorithms, the boosting performs worse when predicting ESR. I reason that this is due to that the boosting algorithm requires what it is boosting need to be at least a ‘weak learner’, i.e. always have a

more than 50% prediction accuracy. While it is easy to find such 'weak learner' when predicting MNIST, as the score performance of decision tree on MNIST(0.691) is in general higher than 0.5, it is probably hard to find 'weak learner' when predicting ESR. This observation suggests there is a Matthew effect on the boosting algorithm: it only works when the best-algorithm performs well and beyond.

Table 3. Inter-model comparison on performance accuracy score and running time, as average of 10 trials.

Algorithm	MNIST			ESR		
	score	time (s)	time-default (s)	score	time (s)	time-default (s)
Decision tree	0.691	6.5	4.4	0.464	2.5	7.4
Boosting using decision tree	0.817	278.3	22.8	0.518	60.9	34.1
Artificial neural network	0.846	36.8	57.8	0.619	67.4	40.4
k-nearest neighbors	0.792	13.6	15.3	0.483	18.4	19.4
Support vector machines	0.796	26.6	31.7	0.564	55.4	77.8

4. Conclusion

This report is my first time to systematically investigate how the hyperparameters affect machine learning model. I have learned that there is a trade-off between choosing low bias/high variance (overfitting) and low variance/high bias (overfitting), and choosing more/less data, more/less features, and tuning the hyperparameters can alter the trade-off toward favorable direction for the improvement of performance. In this report, the performance of all iterative models has been significant improved. I have to point out that individually 'optimizing' parameters, while keeping others at default, usually cannot find "the best solution" (i.e. a set of hyperparameters that yield highest model performance possible). This is especially true when there are many hyperparameters, e.g. the ANN model. The narrow hyperparameter search in this report, the lack of grid search on hyperparameters, and the lack of comparison with current gold standard method in research, will have to be improved in the future works.

5. Reference

[1] Fashion MNIST: An MNIST-like dataset of 70,000 28x28 labeled fashion images (2017, August 25). Retrieved from https://www.kaggle.com/zalando-research/fashionmnist
[2] Epileptic Seizure Recognition Data Set (2017, May 24). Retrieved from https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
[3]. Boosting: why is the learning rate called a regularization parameter? (2015, September 6). Retrieved from https://stats.stackexchange.com/questions/168666/boosting-why-is-the-learning-rate-called-a-regularization-parameter
[4] Neural network models (supervised) (2019, September 8) Retrieved from https://scikit-learn.org/stable/modules/neural_networks_supervised.html
[5] How can I choose the parameter C for SVM? (2017, June 6) Retrieved from https://www.quora.com/How-can-I-choose-the-parameter-C-for-SVM
[6] Heaton, Jeff. Introduction to neural networks with Java. Heaton Research, Inc., 2008.
[7] How to choose the number in a feedforward neural network? (2018, July, 22) Retrieved from https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw