CS 7641 Machine Learning HW1

Hao-Jen Wang

Abstract

In this assignment, I choose the MNIST and CIFAR-10 datasets and conduct some analysis. I will first analyze their data distributions, split the data properly, evaluate performances of five models, draw some insights from results, and finally make conclusions.

1. Introduction

Nowadays machine learning succeeds in various kinds of computer vision applications. For instance, street number classification can improve the efficiency of modern map making, and the camera surveillance system helps us to detect suspicious people quickly. To discover how these models work in practice, I find the MNIST and CIFAR-10 datasets interesting, as they are suited for analysis. Both have full label information and are cited from various works. Compared to UCI toy datasets, classifying these two datasets is much more complex yet worthy and closer to real-world scenarios.

2. Dataset

MNIST handwritten dataset consists of a stack of 28x28 pixel images with 10 labels from digits 0 to 9. Compared to the UCI pen-based toy dataset which only contains thousands of data, MNIST has a much larger data size, with 60,000 training and 10,000 test data. Therefore, we can train more robust modes and can verify the performance on a test set which is closer to the real-world data distribution.

On the other hand, CIFAR-10 consists of a bunch of 32x32 color images in 10 classes, with 6,000 images per class. The 10 classes represent (0) airplane, (1) automobile, (2) bird, (3) cat, (4) deer, (5) dog, (6) frog, (7) horse, (8) ship, and (9) truck. It has 50,000 training and 10,000 test data. Unlike MNIST, images in CIFAR-10 have three channels. They are also more complicated since images are often being rotated, scaled, or inside different backgrounds. Even images in the same class night have distinct shape or color. Samples from the two datasets are shown in Figure 1.

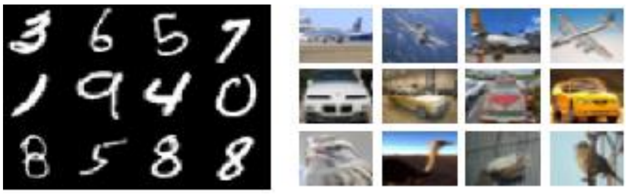


Figure 1. Samples of MNIST and CIFAR-10

3. Data Analysis

The evaluation metrics contain accuracy and running time on test set. Confusion matrices are also shown to better discover misclassifications.

To verify the generalization power for each model, an extra validation set is required so that we can pick up the best model checkpoint with an optimal validation accuracy. Though K-fold cross-validation is suggested, it is inefficient to train on the large-scale data in high dimensions. Due to a limited training capability, I split the original MNIST training set into 50,000 images for training and 10,000 for validation and split the original CIFAR-10 into 40,000 and 10,000. Both splits are random with fixed random seeds to ensure reproducibility.

Figure 2 shows the label distributions for the two datasets. They are both evenly distributed.

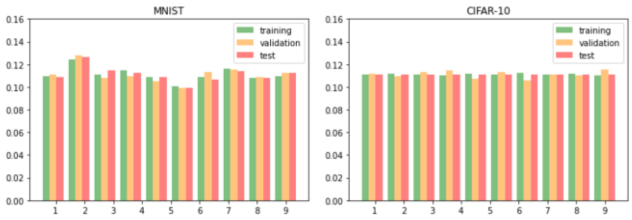


Figure 2. Label distribution

4. Experiments

The features of image are pixel values. Each image has 784 features in MNIST and 3,072 in CIFAR-10. To avoid scalar issues, I normalize the values within [0, 1] by dividing 255. After pre-processing, I can further evaluate models. The **total clock time** here is defined as the sum of training time and validation time.

A. Decision Tree – MNIST

In theory, shallow trees cause underfitting, while deep trees may cause overfitting. Within the tree depth [1, 20], Figure 3 shows how the tree depths affect performances.

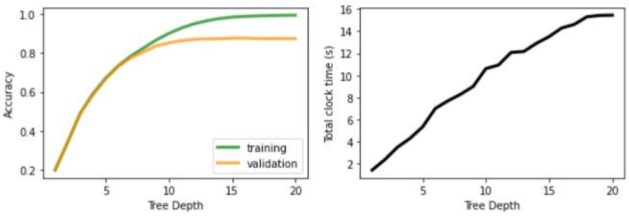
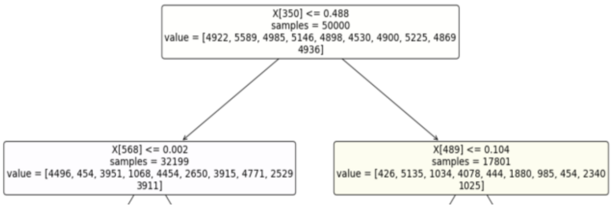


Figure 3. Tree depth curve (MNIST)

In Figure 3, both the training and validation curves converge after a depth of 12. The former one reaches a maximal accuracy of 99.56%, and the latter one reaches 87.76%. The graph on the right shows that the running time is linear to the tree depth. To reduce computations and model complexity, we can stop training earlier like a depth of 12. Though the training accuracy drops to 95.12%, the validation accuracy is still high, and 25.69% of the running time can be saved. Models with lower complexity can generalize better to unseen data, so tree with a depth of 12 should be an optimal model. For simplicity, I define such a tree as "DT_{12}".

Next, we may have a glimpse of how DT_{12} works in Figure 4.



X[350] <= 0.488
samples = 50000
value = [4922, 5589, 4985, 5146, 4898, 4530, 4900, 5225, 4869
4936]

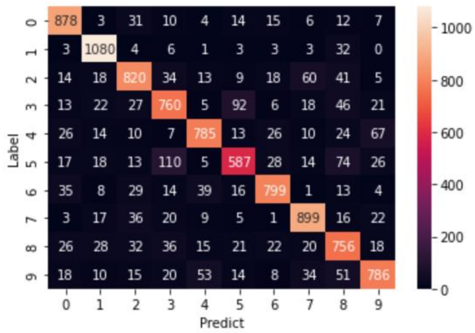
X[568] <= 0.002
samples = 32199
value = [4496, 454, 3951, 1068, 4454, 2650, 3915, 4771, 2529
3911]

X[489] <= 0.104
samples = 17801
value = [426, 5135, 1034, 4078, 444, 1880, 985, 454, 2340
1025]

Figure 4. First decision from DT_{12} (MNIST)

It shows that all decisions are made by some threshold values based on some pixel locations. In the DT_{12} example, the index 350 represents a point near the image center, and the threshold value 0.488 represents a gray color. We know digits like 0 and 7 do not have bright dots on the center pixel, so it is reasonable to distinguish them according to the center pixels. After the first decision in DT_{12}, the number of digit-0 is indeed reduced from 4,922 to 426, and digit-7 is reduced from 5,225 to 454.

As for the unseen test set, DT_{12} yields a test accuracy 81.50%. Figure 5 shows the confusion matrix of the test set from DT_{12}.



	0	1	2	3	4	5	6	7	8	9
0	878	3	31	10	4	14	15	6	12	7
1	3	1080	4	6	1	3	3	3	32	0
2	14	18	820	34	13	9	18	60	41	5
3	13	22	27	760	5	92	6	18	46	21
4	26	14	10	7	785	13	26	10	24	67
5	17	18	13	110	5	587	28	14	74	26
6	35	8	29	14	39	16	799	1	13	4
7	3	17	36	20	9	5	1	899	16	22
8	26	28	32	36	15	21	22	20	756	18
9	18	10	15	20	53	14	8	34	51	786

Figure 5. DT_{12} confusion matrix (MNIST)

In Figure 5, most errors happen to classify 3 as 5, and vice versa. It may be due to irregular handwriting styles in some cases. As we can see in Figure 1, the upper-left digit is 3, but it may be too messy for DT_{12} to distinguish.

Now we focus on the learning curve given by different sample sizes. Note that the training accuracy should be computed in an individual training set. To be fair, a smaller data should be a subset of a larger data. Based to these rules, the result is plotted in Figure 6.

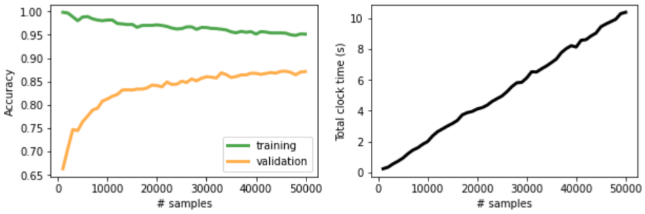


Figure 6. DT_{12} sample size curve (MNIST)

Given a fixed tree depth, models trained on smaller datasets seem to be too optimistic while training, yet overfit. As the size increases, the training curve slightly drops, but the validation curves rise significantly. Indeed, more data can yield a better generalization power for a model.

B. Decision Tree – CIFAR-10

Now let us verify whether decision trees can also yield decent performances here. Since the feature dimension is high, I expand the range of tree depth to [1,30]. The result is in Figure 7.

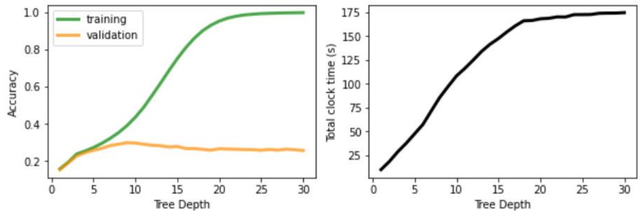


Figure 7. Tree depth curve (CIFAR-10)

The training accuracy grows when the depth is deeper. Nevertheless, the validation accuracy never exceeds 30%. It is useless to further deepen the tree since the trend has stop growing. All validation scores are relatively low, so I decide to choose the deepest tree. The tree depth 30 model (DT_{30}) yields a 99.84% training and a 25.63% validation accuracy. It is obvious that it overfits on the training data. On the test set, it yields only 26.89% accuracy, and the confusion matrix is shown in Figure 8.

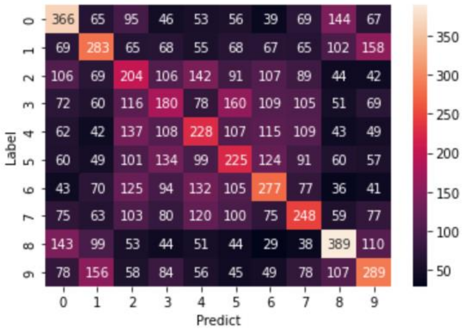


Figure 8. DT_{30} confusion matrix (CIFAR-10)

Note that a maximal value for each column still lies in the diagonal. The two hardest classes for DT_{30} to classify is birds and cats (labeled as 2 and 3). To better figure out the reason, we can visualize some examples in Figure 9.

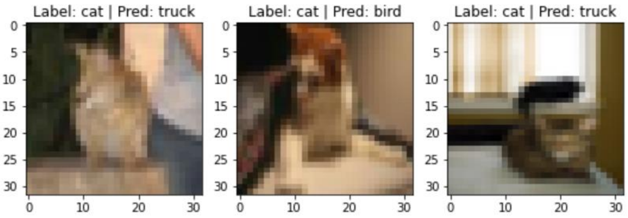


Figure 9. DT_{30} misclassifications (CIFAR-10)

In general, images with trucks tend to have huge rectangular objects, whereas birds tend to have a relatively tall and thin object with a tail on endpoints and sometimes wings on two sides.

Based on observations, the center image has a tall and thin cat with a straight tail which seems to contain features of birds, and the rightmost image has an explicit window occupying a large portion of the area. With all kinds of variation, it is hard to make good decisions.

Now that decision tree overfits seriously in CIFAR-10. Does it matter regarding the data size? Figure 10 shows the data size curve.

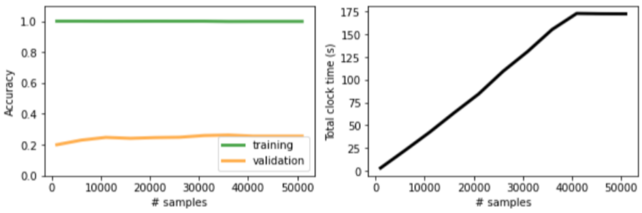


Figure 10. DT_{30} sample size curve (CIFAR-10)

Unlike MNIST, increasing the size seems to be irrelevant to the performance. The standard deviation of the validation curve is only 0.017, so difference is just due to the randomness from experiment. Such a fact tells that most data have no contribution for making the decisions!

Perhaps decision tree models cannot classify problems successfully when data lies in an over complex feature space. To simplify the problem, I subsample the data that contains birds and cats only. These two classes are difficult for decision trees as Figure 7 described. With a size of 7,594 for training, 2,046 for validation and 2,000 for testing, the result is shown in Figure 11.

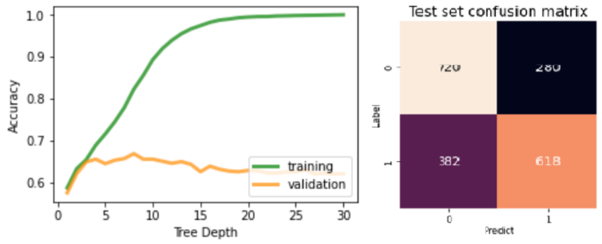


Figure 11. Simplified evaluations (CIFAR-10)

In Figure 11, the graph on the left shows the training and validation curves, and the right one shows the confusion matrix on the test set. The validation has 66.86% accuracy when the depth is 8, with a 66.90% test accuracy. Compared to the previous result, we know that decision trees cannot shatter the whole 10 classes, so reducing the difficulty can boost the accuracy.

C. *k-NN – MNIST*

k-NN is a lazy method that make inferences from saved training data. The inference time is super slow since it needs to compute distances from data points in a high dimensional feature space. With a limited time, I ignore the training accuracy and set the range of *k* in [1, 5]. The result is in Figure 12.

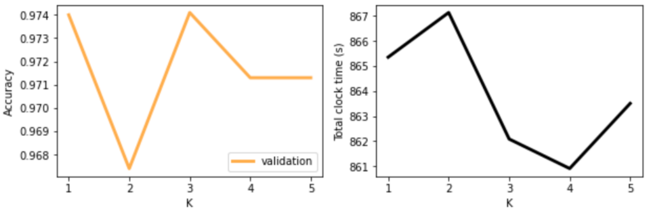


Figure 12. *k*-NN learning curve (MNIST)

Figure 12 shows that the validation accuracy is high despite *k.* 3-NN has the best validation score 97.41%, with a test score 96.94%. Given a fixed *k* =3, the relationship between the size and the performances is shown in Figure 13.

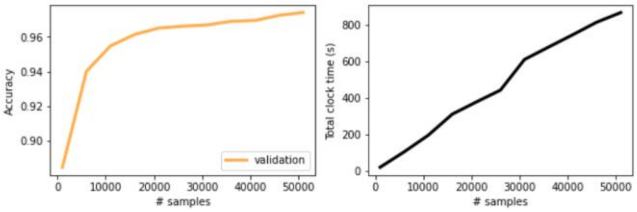


Figure 13. 3-NN sample size curve (MNIST)

Just as the result in decision trees, more data yields a better test accuracy.

D. *k-NN – CIFAR-10*

Now let’s see if the result is still good here. After running more than 5 hours, the curve for *k* in range [1, 5] is plotted in Figure 14.

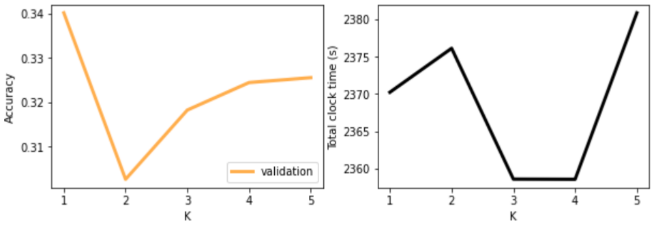


Figure 14. *k*-NN learning curve (CIFAR-10)

The optimal validation accuracy is 34.01% when *k*=1, and its test accuracy is 34.00%. Note that we cannot assert such a model is worse than a naïve guess, since this is a multi-class problem, and data in CIFAR-10 is hard for models to tell without any prior domain knowledge.

From another standpoint, it is not surprising that *k*-NN performs poor on CIFAR-10. Since *k*-NN measures the closest points, it implies that similar images shall have the same distribution of pixels. However, some images in the same class have totally different pixel distribution in CIFAR-10. Thus, computing the differences is not a proper method in this task.

We can further analyze the confusion matrix for 1-NN in Figure 15. The diagonal values are still relatively high. Compared to the result from the decision tree, 1-NN can better classify birds and cats. However, 1-NN performs poorer on classifying automobiles and trucks. Both classes are often misclassified as ships. Perhaps, objects for transportation have closed pixel distributions.

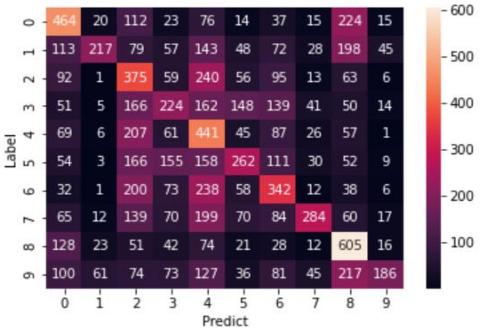


Figure 15. 1-NN confusion matrix (CIFAR-10)

So far, *k*-NN outperforms the decision tree in accuracy yet worse in running time. Note that both datasets do not require a high *k*. It implies that **only a few references are sufficient** to measure the similarity.

E. *Neural Network – MNIST*

In theory, a deeper neural network (NN) can capture more features. In this experiment, I use Adam optimizer and ReLU as default. First, I evaluate the relationship between size of hidden nodes and performances in Figure 16. Since too many variables should be considered, I train the model in a single layer and will ensure that it is converged. As the default learning rate (0.001) causes convergence warning, I multiply it by 10.

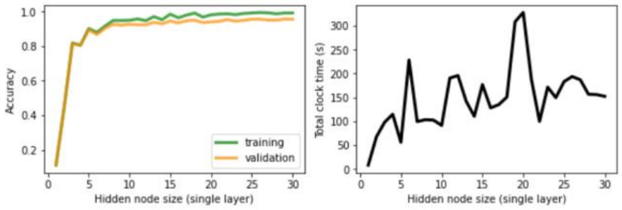


Figure 16. NN node size curve (MNIST)

Both the training and validation curve rise very fast when the hidden node size is greater than 5. The best validation checkpoint, with a node size of 29, has test accuracy of 95.54%. It is interesting that the number of nodes does not in time proportional to the running time. Thus, the computational overhead **does not depend on the size of hidden nodes**.

Now we know the optimal node size is 29, and let’s consider the relationship between layer sizes and the performances. By fixing the node number as 29, we can get a result in Figure 17.

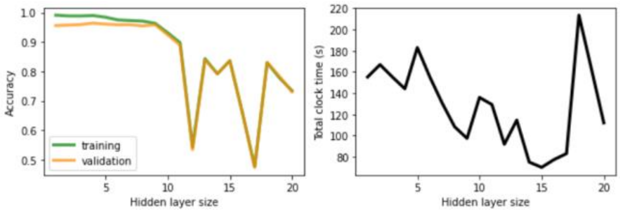


Figure 17. NN layer size curve (MNIST)

Surprisingly, **larger layer sizes do not yield better performances!** It becomes even worse when the layer is too deep. The best validation is 96.45% accuracy when the size is 4, and its test accuracy is 96.35%. It merely boosts 0.91% test accuracy by adding two extra layers!

Note that it does not imply overfitting as the training curve also drops. Ablation studies show no matter how small the number of nodes (5 or 29) is, or how large the L2 penalty term (0.001 or 0.0001) is, both the two curves drop when adding more layers (see Figure 18).

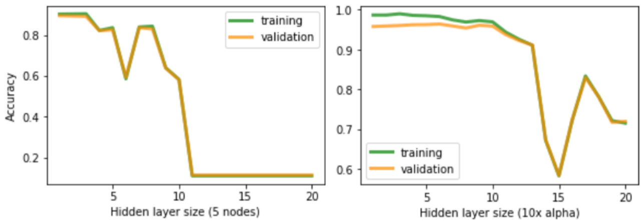


Figure 18. Ablation studies of NN (MNIST)

To better know the reason behind it, we can check the training errors. In the original settings (4 layers and 29 nodes each), Figure 19 includes all the training curves in different size of layers.

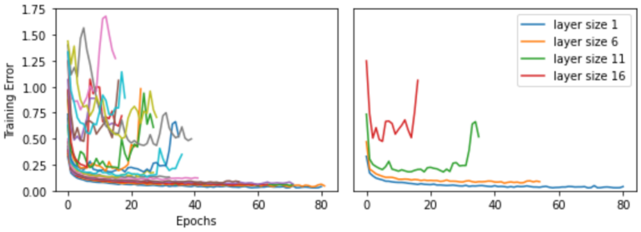


Figure 19. NN training error curve (MNIST)

The right figure samples some of the error curves on the left and is simply used for a better visualization. Given a fixed number of nodes in each layer, models with deeper layers prone to yield larger errors and are harder to converge. In theory, a deeper neural network has a **higher model complexity**, which makes the hypothesis space larger. With so many candidates, it may be difficult to find out the optimal solution.

To sum up, more layers do not yield more accurate models in recognizing digital images. Experimental results show that the NN with 4 hidden layers and 29 nodes for each layer has an optimal test accuracy 96.35%. Finally, we can evaluate how the data size influence the model performances in Figure 20.

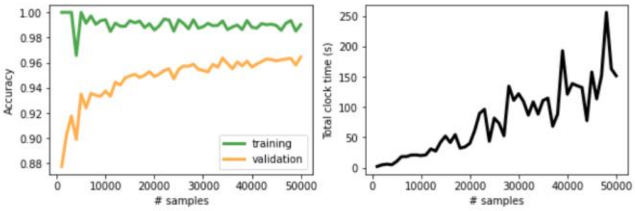


Figure 20. NN sample size curve (MNIST)

This trend is similar to the result from *k*-NN and the decision tree. The more data we have, a better generalization ability a model can get. So far for MNIST, 3-NN has the highest test score (96.94%), and the neural network with 20 node size and 4 layers is the second (96.35%).

F. *Neural Network – CIFAR-10*

In expecting that more nodes within a layer can boost the performance, let’s verify the result from CIFAR-10 shown in Figure 21.

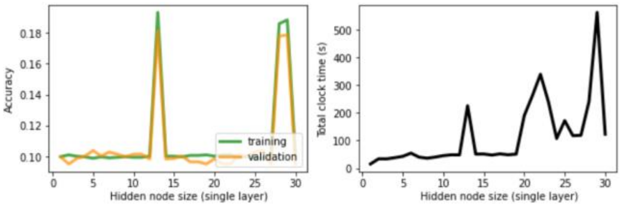


Figure 21. NN node size curve (CIFAR-10)

Unfortunately, adding extra nodes does not always increase the accuracy, and most of the time the performance is very bad. The optimal result here is a node size of 13, yielding 19.14% test accuracy. Maybe networks which is shallow cannot capture complex features in CIFAR-10. To verify it, we can check whether increasing the hypothesis space (i.e., the number of layers) can improve the performance in Figure 22.

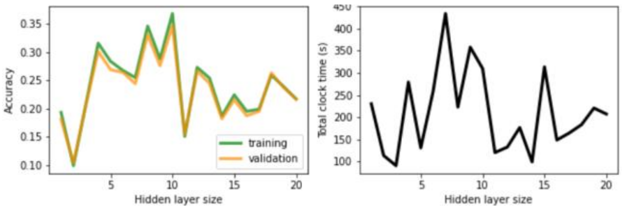
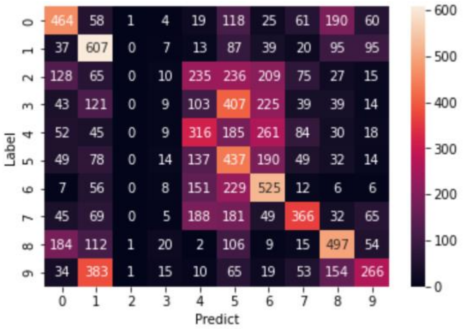


Figure 22. NN layer size curve (CIFAR-10)

Given a fixed number of nodes 13 in each layer, the model with a layer size of 10 yields an optimal test accuracy 34.87%. Despite it is still low, we can observe that adding extra layers can improve the performance. It seems inconsistent with the previous result, and I believe the reason may be due to the complex features in CIFAR-10. Deeper network might be required to enrich the hypothesis space for such a task.

Now *k*-NN slightly does a better job than the neural network. We can see more details from the confusion matrix in Figure 23.



	0	1	2	3	4	5	6	7	8	9
0	464	58	1	4	19	118	25	61	190	60
1	37	607	0	7	13	87	39	20	95	95
2	128	65	0	10	235	236	209	75	27	15
3	43	121	0	9	103	407	225	39	39	14
4	52	45	0	9	316	185	261	84	30	18
5	49	78	0	14	137	437	190	49	32	14
6	7	56	0	8	151	229	525	12	6	6
7	45	69	0	5	188	181	49	366	32	65
8	184	112	1	20	2	106	9	15	497	54
9	34	383	1	15	10	65	19	53	154	266

Figure 23. NN confusion matrix (CIFAR-10)

Notice that the neural network can perfectly detect automobiles, yet it is totally unable to detect birds and cats, even more extreme than decision trees! The underlying reasons is hard to analyze since we have a balanced dataset, yet we can observe that the neural network still has a higher accuracy. Perhaps optimizers focus on minimizing the overall error instead of each individual error. With a limit learning capacity, it is better to get rid of some hard examples to yield a better overall performance in some cases.

G. *SVM – MNIST*

SVM refers a subset of data points to draw a unique decision with a maximal margin. In such a scenario, validation data is useless since there are no multiple checkpoints. In this experiment, I use the linear and RBF kernels. **I am unable to plot the learning curve here**, since *sklearn* package only support the learning curve with K-fold cross-validation, whereas my experiment has a fix validation set across all models.

After training on 50,000 data, linear SVM yields 93.71% test accuracy, and RBF SVM has 97.83% test accuracy. The linear kernel is worse since the original data is not linear separable. So far in MNIST, the top two accurate models are RBF SVM (97.83%) and 3-NN (96.94%).

Using the RBF kernel, we can see how the data size affects the performance in Figure 24.

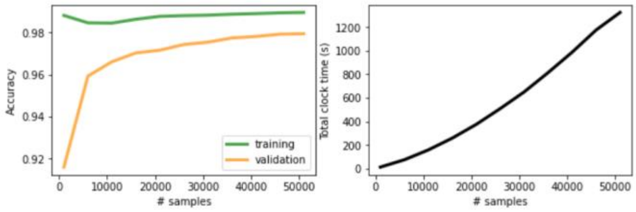
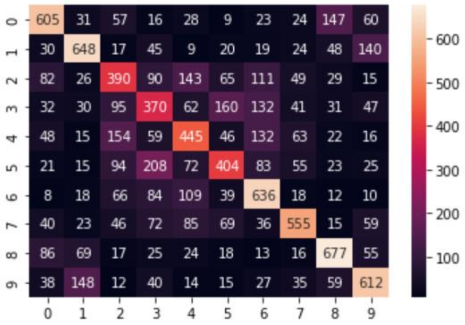


Figure 24. SVM sample size curve (MNIST)

We know SVM ignores datapoints that are not belong to the support vectors. In Figure 24, the range of the validation score is very narrow, meaning that smaller data size is already enough to train a decent SVM model. Therefore, if the running time is limited, it may be fine to reduce the data size properly.

H. SVM – CIFAR-10

From the previous experiment we know that RBF kernel is better than linear. Due to limited computational resources, I just run the RBF for CIFAR-10. After training 1.76 hours and testing 24.63 minutes, the test accuracy is 53.42%! The confusion matrix is also shown in Figure 25.



	0	1	2	3	4	5	6	7	8	9
0	605	31	57	16	28	9	23	24	147	60
1	30	648	17	45	9	20	19	24	48	140
2	82	26	390	90	143	65	111	49	29	15
3	32	30	95	370	62	160	132	41	31	47
4	48	15	154	59	445	46	132	63	22	16
5	21	15	94	208	72	404	83	55	23	25
6	8	18	66	84	109	39	636	18	12	10
7	40	23	46	72	85	69	36	555	15	59
8	86	69	17	25	24	18	13	16	677	55
9	38	148	12	40	14	15	27	35	59	612

Figure 25. SVM confusion matrix (CIFAR-10)

Note that the diagonal values are maximum in each column, meaning that the prediction for every class has a non-trivial true positive rate. The hardest classes are still birds and cats, but the true positive rate is higher compared to the previous models.

I. Boosting – MNIST

Boosting ensembles multiple weak learners into a stronger model. The training time is super slow since each weak learner decides based on previous results iteratively. In this experiment, I use AdaBoost and the weak learners as decision trees with depth of 1. In the first trial, I set the number of learners ranged from 25 to 200. The learning curve is shown in Figure 26.

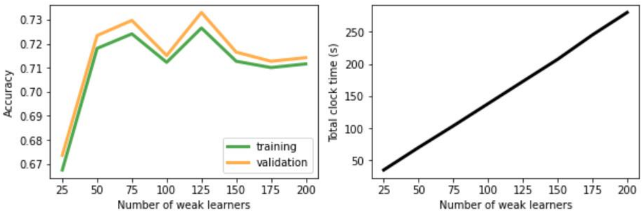


Figure 26. AdaBoost learning curve (MNIST)

The running time is in time proportional to the number of weak learners. The accuracies are not ideal, and the optimal validation accuracy is 73.30% when the number of learners is 125, and its corresponded test score 71.49%. Notice that the training curve is worse than the validation curve, which means the model is underfitting.

The performance is no ideal, and maybe the number of learners is not enough. In theory, a maximal of 200 iterations can only run through at most 200 features when the weak learner is a depth-1 decision tree, forcing other features to be ignored. Based on reasoning, the number of weak learners should be at least 784 in MNIST.

Now, the range is set from 1,000 to 5,000, and let's see if the accuracy is boosted in Figure 27.

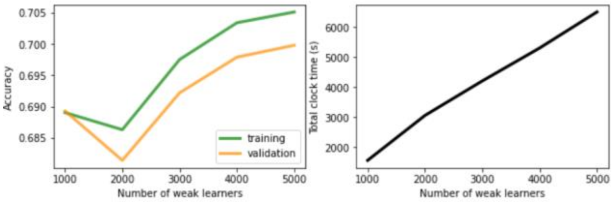


Figure 27. AdaBoost + more learners (MNIST)

Unfortunately, the result shows that adding more extra learners does not guarantee a better performance! After training 5.73 hours here, the optimal validation becomes 69.98%, which is even worse than a decision tree model.

Another hyper-parameter that I can tune is the power of weak learners. Thus, I try to add more tree depth for the weak learners (from 1 to 10) and see if the performance will be boosted. The result is shown in Figure 28.

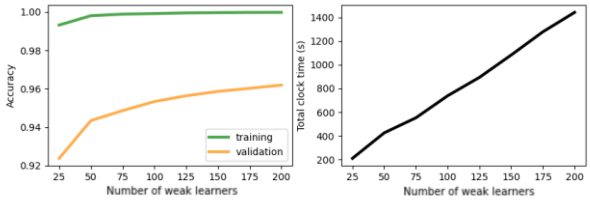


Figure 28. AdaBoost + refined learners (MNIST)

It turns out the performance is significantly boost! The optimal validation score is 96.19% when the number of learners is 200, and its test accuracy is 95.95%. This result contradicts the lecture suggestion: adding extra learners does not harm the performance, while refining weak learners may cause the model to overfit. At least for MNIST, **it is required** to have some refined base learners. Perhaps decision trees with depth of 1 is too hard to find a proper hyperplane that separates the data in a high dimensional space.

As for the running time, both the number of learners and the maximal depth of learners are in time proportional to the running time. Based on the previous results, The time has a slope of 1 regarding the number of learners but has a less slope of 0.4 regarding the depth of learners. As a result, refining the trees can make the training more effective and more efficient.

Finally, we may again check the relationship between the data size and performances. With a fixed tree depth, Figure 29 shows the result.

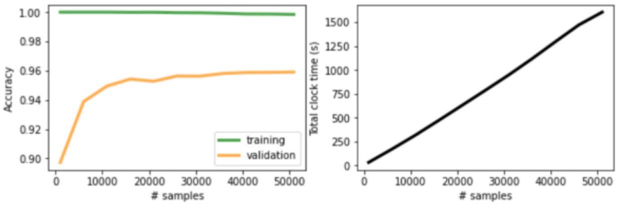


Figure 29. AdaBoost sample size curve (MNIST)

Without surprising, more data yields a better generalization. Notice that AdaBoost and SVM converge after the number exceeds 10,000 and make little improvement afterwards.

J. Boosting – CIFAR-10

Based on the previous result, the complexity of base learners is a more important factor than the number of learners. However, it takes about 2 hours to run the AdaBoost with merely 25 number of learners in depths of 10. Due to the limited time, I plot the curve with a base learner with depth 1 in Figure 30.

When the number of weak learners is 200, it yields an optimal validation score 32.57% and a test accuracy 32.81%. Note that the curve seems to be growing, but I don't have enough time to run the rest of the points.

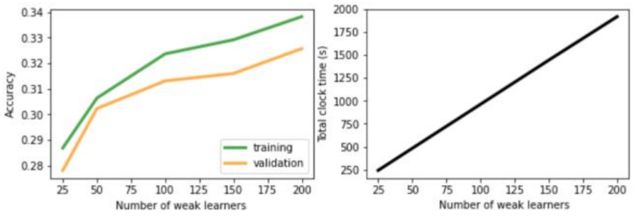


Figure 30. AdaBoost learning curve (CIFAR-10)

5. Discussion

In the previous experiments, I run different ML models for image classification. Datasets include the well-known MNIST and CIFAR-10, and their classes are evenly distribution. They have very high volume of data and dimension of feature space, and the latter one is even more complex regarding the variations within images. The main challenge here is the running time.

Overall, model performances are decent in MNIST yet unideal in CIFAR-10. Based on the validation checkpoints, the comparison of the test accuracy is shown in Table 1. It shows that RBF SVM has the highest score among all.

Table 1. Test accuracy comparison

Dataset Model	**MNIST**	**CIFAR-10**
Decision Tree	81.50%	26.89%
k-NN	96.94%	34.00%
Neural Network	96.35%	34.87%
SVM (RBF)	**97.83%**	**53.42%**
AdaBoost	95.95%	32.81% (*)

(*) Due to limited of times, it is not optimal

The main contribution in the experiments is to draw some useful insights based on limited observations. It is worth to reveal that many results are counterintuitive and even contradict some ML theories. For example, more data or more complex models do not guarantee a better model performance, since some data might be useless while training, and the optimal solution may be too difficult to be found in a complex hypothesis space. Other insights draw from the experiments include:

Data

- More data can improve the model accuracy as long as the target function is solvable.
- The hardest classes to classify in CIFAR-10 are birds and cats.

Decision Tree

- It is most interpretable yet worse in accuracy.
- Decisions are made based on pixel values, so it is reasonable to judge the center pixel value in the beginning for MNIST.
- It is unsuitable for classifying large-scale datasets due to computational overheads.
- It is unsuitable for classifying datasets with complex backgrounds, such as CIFAR-10.

k-NN

- Small *k* is sufficient for image classification.
- It is unsuitable for classifying datasets with distinct pixel distributions within the same class, such as CIFAR-10.

Neural Network

- Increasing the number of nodes can increase the performance in general.
- Increasing the number of layers can increase the performance when features are complex.
- In multi-class problems, it may sacrifice the accuracy for some hard examples to yield an optimal error rate.

SVM

- It is most accurate and fast to converge.
- It does not require too much training data.
- RBF kernel is better than linear kernel for non-linear separable datasets.

Boosting

- It does not require too much training data.
- It is unsuitable for classifying large-scale datasets due to computational overheads.
- Increasing the number of learners does not guarantee a better accuracy.
- Increasing the complexity of learners tends to yield a better accuracy.

6. Conclusion

Image classification is a challenging task for traditional machine learning methods. Basically, SVM with RBF kernel yields the best accuracy, while Decision Tree has a poorest performance yet a better interpretability and a faster running time. Increasing the size of data can boost the performance, whereas increasing the complexity of models does no guarantee so due to a larger hypothesis space.

Since both MNIST and CIFAR-10 have high volume of data and high dimensions of features, we might require techniques such as PCA or subsampling to reduce the training overhead. By doing so, I believe it is possible to further boost the model accuracy and training efficiency.

Reference

[1] https://scikit-learn.org/stable

[2] https://www.tensorflow.org/datasets

[3] http://yann.lecun.com/exdb/mnist/

[4] https://www.cs.toronto.edu/~kriz/cifar.html