Submission Date: 2023/10/23

Fundamentals of AI : Assignment 2

Classification of Handwritten Digits

UMU CS ID : ens23cco, ens23hoa
NAME: Changhee Cho,Hinako Oshima

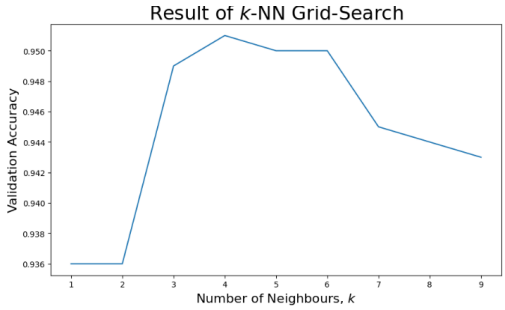
k – NN

Implementation

The KNeighborsClassifier is a class used for implementing the k-NN model, inheriting from the super class neighbors in the sklearn module. This class comprises three functions: __init__, fit, and predict. The __init__ function is responsible for setting the hyperparameter k. The fit function reads in the known data X (matrix data) and y (label data), while the predict function classifies the digit for unknown data.In this assignment, grid search has been used for selecting optimal hyperparameter k. In the beginning, the range of search was set as numbers 1-9.

In this study, different numbers of neighbors (k) were tested. The number of data used in the experiment was "Xtrain = 5000, ytrain = 5000, Xvalid = 1000, and yvalid = 1000". With k=1, the validation accuracy was 93.6%. When k was increased to 3, accuracy improved to 94.9%. Further increasing k to 4 resulted in a validation accuracy of 95.1%. However, setting k to 7 led to a slight decrease in accuracy to 94.5%. Finally, with k=9, the accuracy was 94.3%

The accuracy increased monotonically for every increment of k. However, it began to decrease monotonically after k=4 whose accuracy is 0.951. From this, it was predicted that hyperparameter values beyond k=9 would continue to lower the accuracy. Therefore, for the test data, the hyperparameter value was determined to be k=4.



pseudo-code

pseudo-code
function fit
1.put all the handwritten digit matrix data to X
2.put all the digits label data to y

function predict
for all element in X
1.find distance from point to other data
2.sort them
3.get k nearest labels
4.count the numbers of each nearest label
5.append the highest number of label in prediction list
return prediction list

Results

As a result, the Final validation accuracy for the hyperparameter k = 4 was 0.9617142857142857. The number

of data used in the experiment was "Xtrain = 20000, ytrain = 20000, Xvalid = 10000, and yvalid = 10000". The final test data accuracy result was 0.9649142857142857.

Discussion

Setting the hyperparameters in k-NN is predicted to have a significant impact on accuracy, and as a result, we observed a difference of approximately 0.3% in accuracy depending on the value of k. Furthermore, when we experimentally decreased the total number of data points to three digits, we found that it produced a different optimal value of k compared to when using validation score or test score, leading us to believe that the optimal value of k may also vary depending on the total number of data points in both X and y.

Artificial Neural Network (ANN)

What is hyper parameter?

While parameters are variables that the model finds through training, hyper parameters are variables that we set before training. Therefore, setting the hyper parameters well can significantly improve the performance of the model.

Experiment overview

The hyperparameters we set are as follows: numbers of layers, numbers of neurons, learning rate, number of epochs. To represent the experiments in code, we have separated the code sections for each hyper-parameter experiment. Thus, the code for problem 2 is divided into four main sections in our notebook.

To make the experiment more accurate, we kept the values of the other variables fixed and only transformed one hyperparameter. In order to reduce the time, the number of data used in the experiment was “Xtrain = 5000, Ytrain = 5000, Xvalid = 1000, and Yvalid = 1000”, which is 10 times less than the original data.

Implementation

(1) Number of layers: We compared the validation accuracy with the number of neurons unified at 100 and the number of hidden layers varied from 2 to 6.

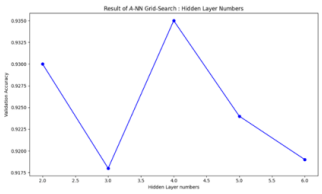
(2) Number of neurons: We varied the random number of neurons based 4 layers, as previous experiments showed the highest accuracy. We set the minimum and maximum number of neurons for each layer. In general, CNNs tend to get bigger from the input values and then get smaller as they move to the output layer, so we set them in a similar way. Then, we ran the for loop 10 times.

(3) Learning rate: The learning rate is the stage at which machine learning becomes learning. So setting the right learning rate is important for learning speed and accuracy. MLPClassifier automatically gradually decreases learning rate from initial value as it approaches the minimum. We compared the results by decreasing the initial value by an exponential factor from 0.1 to 0.0001.

(4) Number of epochs: The number of epochs is also an important hyper-parameter that determines accuracy. Beyond a certain number of epochs, overfitting can occur. For our experiment, we used the partial_fit function to force the training to the desired number of epochs.

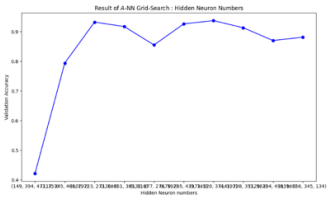
Results & interpret

Number of layers



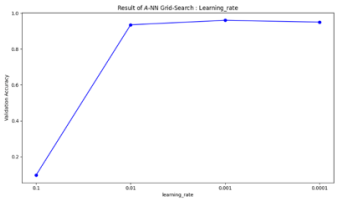
Before the experiment, we expected that the more layers there are, the higher the accuracy will be. However, the maximum value was 0.935 when the number of layers was 4 and then it falls. There are several reasons why the number of layers decreased after that. It could be due to overfitting, or the inability to fit other hyper-parameters, or it could be because the number of data is not large enough.

Number of neurons



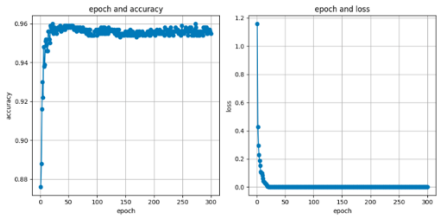
The results were quite irregular. We didn't see any correlation between the neuron size setting and accuracy, and when we ran the code again with the same neuron size, the accuracy was very different. We concluded that neuron size is a hyper-parameter that doesn't have a significant impact on accuracy. The best accuracy was achieved with [173, 220, 374, 197].

Learning rate



When we set it to 0.001, we saw a significant improvement in accuracy to 0.96. This shows that the learning rate is an important hyper-parameter for machine learning. The reason is that this task doesn't have that many parameters, so reducing the learning rate didn't slow down the learning speed much.

Number of epoch



After about 30-40 epochs, the loss no longer decreases, but stays the same. We can see that iterating is time-consuming and pointless.

```
ANN model training data accuracy   : 0.9808761904761905
ANN model validation data accuracy : 0.9712
ANN model test data accuracy       : 0.9706285714285714
```

By setting the hyperparameters determined by the experiment, the final ANN model was calculated as 15000 train sets and 3000 valid sets. The KNN final model was trained with 20000 data sets, and we found about 10% better performance than KNN.

Discussion

1. As we experimented with different hyper-parameters, we realized that there is a prioritization of hyper-parameters. In our experiments, the learning rate setting had the biggest impact on accuracy.
2. Among the four hyper-parameter experiments, the number of neurons was the most irregular. We need to find out how to set the number of neurons more effectively.
3. ANN showed about 10% higher performance despite using less data. However, it took less time when running KNN. Here, the time-accuracy trade-off relationship between deep learning and machine learning could be found.

Problems and issues of our work

When running experiments with the number of epochs, it was difficult to run experiments because even if we changed the value of max_iter, it would automatically stop training at 30-40. In the end, we solved this problem by manually running the training once.

How did we divide the work

Cho Changhee: Coding KNN, conducting ANN experiments, and writing reports ANN part. In the case of KNN, we wrote pseudocode together to understand the logic, then wrote our own code for a complete understanding. Then, we complemented each other. In the case of ANN, we explored the types of hyper parameters together and I conducted experiments for each hyper parameter. Then, I wrote a report on ANN.
Hinako Oshima: Responsible for conducting experiments and creating the KNN section of the report. Under the guidance of partner Changhee Cho, gained a deeper understanding of machine learning with regards to ANN.