CSC 442 Project 4 - Learning

Name : Tushar Kumar
NetId : tusharku

December 13, 2018

Abstract

In this project I have implemented two learning algorithms, Decision Tree and Neural Network. I have tested my implementation on the AIMA-Restaurant exmaple and also on two datasets from UCI Repository. One is Iris data set and the other is Connect-4 dataset. I have validated the different hyper parameters like learning rate, batch size, network architecture for neural networks and depth of tree and splitting criteria for decision trees.

1 Introduction

Machine learning is the branch of Artificial Intelligence that incorporates algorithms to analyze data which is inputted, and via statistical analysis can make a prediction on an output, while incorporating new data as it becomes available, to update the predicted output.

Our goal is to build a program which given a dataset learns to predict the target attribute based on attribute values of the data. Post that we should be able to predict the target attribute on unseen data with decent accuracy.

2 Design

The basic design can be understood from the Component Diagram 1. I use the concept of inheritance to abstract out all common learning methodologies in the base class (BaseModels). Each of the child classes then only have behavior that is specific to that class only. I also built utilities for validating different hyper parameters for each of the models I implemented.

2.1 Key Design and High Level Implementation Details

a) **Implemented Decision Tree** and tested on all datasets.

b) **Implemented Neural network** and tested on all datasets.

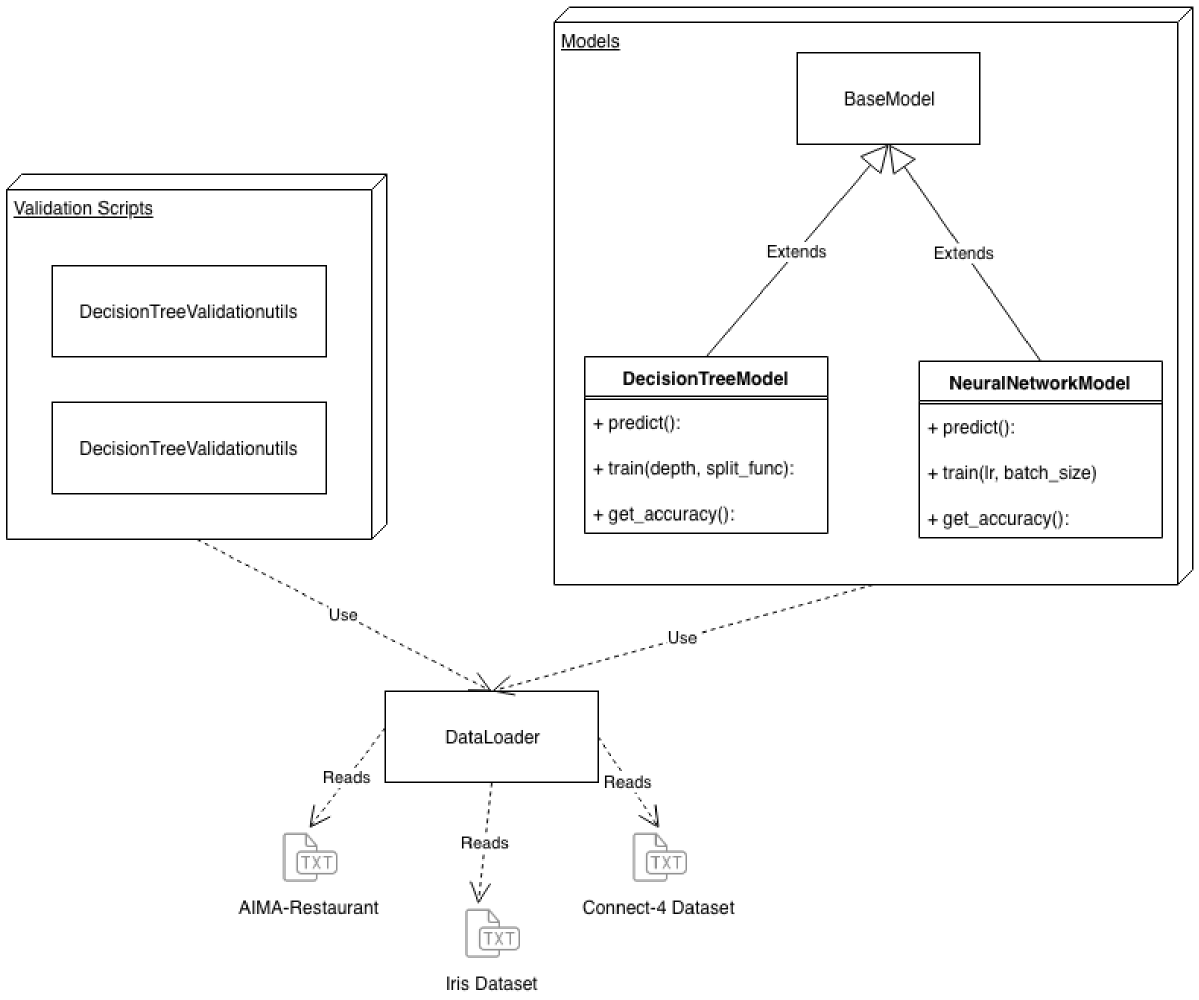


Figure 1: Component Diagram for Learning Project

c) **Converted discrete data to one hot encoding vectors** for neural networks.

d) **Analyzed the effect of learning rate, batch size, network architecture**for neural network on the datasets using validation.

e) **Analyzed the effect of splitting criteria and maximum allowed depth to grow** for decision trees on the datasets using validation.

f) **Graceful Error Handling** and providing descriptive errors to direct the user in the correct way of running the program.

3 Datasets

3.1 Iris Dataset

Iris dataset[1] is available in UCL ML repository. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Since I have only built algorithms based on the attributes being categorical, I use the version of dataset provided to us where the attribute values have been discretized to S/MS/L/ML.

3.2 Connect-4 Dataset

Connect-4 dataset[2] is available in UCL ML repository. This database contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. The outcome 'class' is the game theoretical value for the first player. I predict the class attribute for this dataset using both the learning algorithms. The dataset consists of 67557 rows and 42 attributes all of which are categorical.

4 Decision Trees

I have implemented the ID3 algorithm for decision trees using entropy as the splitting criteria. I follow the pseudocode in AIMA Fig 18.5 3rd Ed. The procedure basically involves iterating through all the attributes one by one and finding the attribute that provides the maximum reduction in entropy value. A dataset which has all rows with same class value will have the minimum entropy value whereas a dataset which has equal number rows for each target attribute domain value will have maximum entropy. ONce we find the best attribute, we partition the dataset into based on the different values of that attribute and then recursively do the same for each partition. The base case happens when there are either no rows or no attributes to test or all the rows in the partition have the same target attribute value. In case there are no rows, we use the majority class value from the parent node to predict the label for this node. Every node of decision tree can either have a label value(in which case it will be a laef) or in the other scenario it will have a dictionary mapping of the attribute value and the branch that the tree takes if it finds that as the attribute value. I also provide a mechanism to print the tree. Apart from entropy, I also implemented Gini impurity as the splitting criteria(which was actually leads to better accuracy for more complex datasets). I will now provide the analysis of running decision trees on all the different datasets.

4.1 Aima-Restaurant

Aima Restaurant is the example mentioned in AIMA where the scenario is a bunch of friends are going to a restaurant and they have been asked by the manager to wait. Given the dif-

[1]https://archive.ics.uci.edu/ml/datasets/iris

[2]https://archive.ics.uci.edu/ml/datasets/Connect-4

ferent attribute of the scenario like whether its weekend or whether the restaurant is full and so on, we have a value of "WillWait:" target attribute which basically decides whether the friends will wait or not. The goal is to learn a decision tree from this dataset which automatically can be sued for future such scenarios.

I have attached the output for this particular example problem in Figure 2. One must note that this in order to print the tree programmatically, I had to output the tree in Depth first manner and hence the output will not look like a tree per say. All branches having same indent are on the same level. So first level is Patrons, second level is hungry check(and leaf nodes from patrons check, third level is type check(and leaf nodes from hungry check) and fourth level is Fri/Sat check(and leaf nodes from type check Last level is all the leaf nodes from Fri/Sat check. One one can see that the tree checks Patrons then Hungry and then Type. Only if type is Thai, we also check for Fri/Sat. This is exactly the same order of attributes and indeed the same tree found in AIMA Fig 18.6 3rd Ed.

Since this is just a dummy dataset, I have not run any analysis and experimentation on this dataset. I also did not see the need to train a neural network for this. I will be doing that for the Iris and Connect-4 dataset and providing the detailed analysis.

4.2 Iris

In this section I will discuss the analysis of applying decision tree learning algorithm on the Iris dataset. Since the iris dataset comes with only 150 samples, the decision tree learned does not necessarily have a large depth, but I still experiment with restricting the tree to different depths and seeing how much validation accuracy it really helps.

4.2.1 Experiment Setup

I use the following experiment setup for validating the different parameters for decision tree for iris dataset.

a) Divide the entire data into train, test, validate split of 60/20/20

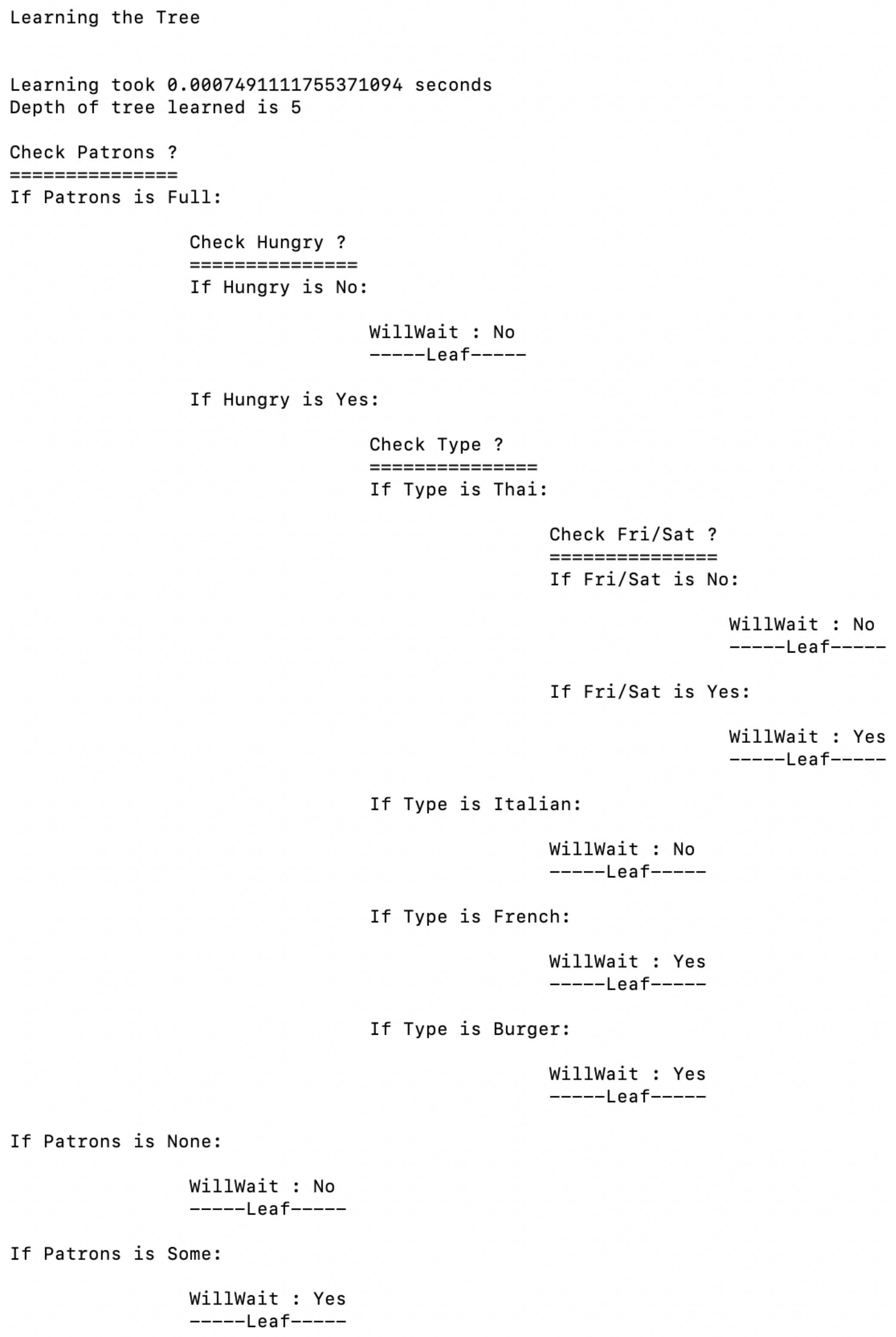
b) Train the decision tree on the 60% and validate the hyper parameter of depth and splitting criteria on the validate set.

c) Train the decision tree using the validated hyper parameters on the 80% (keeping test data untouched)

d) Test and report the accuracy of decision tree on the 20% of data.

4.2.2 Validation of depth of tree

Even though this is a small dataset and hence the depth of tree would not be expected to grow to huge amounts, I went ahead and ran validation experiments for different depths of tree. I keep 20% of data as test data aside, and then ran validation experiments for depth by randomly choosing 75% of remaining data(this is equivalent to 60% of total data) and



```
Learning the Tree


Learning took 0.0007491111755371094 seconds
Depth of tree learned is 5

Check Patrons ?
===============
If Patrons is Full:

                Check Hungry ?
                ===============
                If Hungry is No:

                                WillWait : No
                                -----Leaf-----

                If Hungry is Yes:

                                Check Type ?
                                ===============
                                If Type is Thai:

                                                Check Fri/Sat ?
                                                ===============
                                                If Fri/Sat is No:

                                                                WillWait : No
                                                                -----Leaf-----

                                                If Fri/Sat is Yes:

                                                                WillWait : Yes
                                                                -----Leaf-----

                                If Type is Italian:

                                                WillWait : No
                                                -----Leaf-----

                                If Type is French:

                                                WillWait : Yes
                                                -----Leaf-----

                                If Type is Burger:

                                                WillWait : Yes
                                                -----Leaf-----

If Patrons is None:

                WillWait : No
                -----Leaf-----

If Patrons is Some:

                WillWait : Yes
                -----Leaf-----
```

Figure 2: Decision Tree Learned for Aima-Restaurant

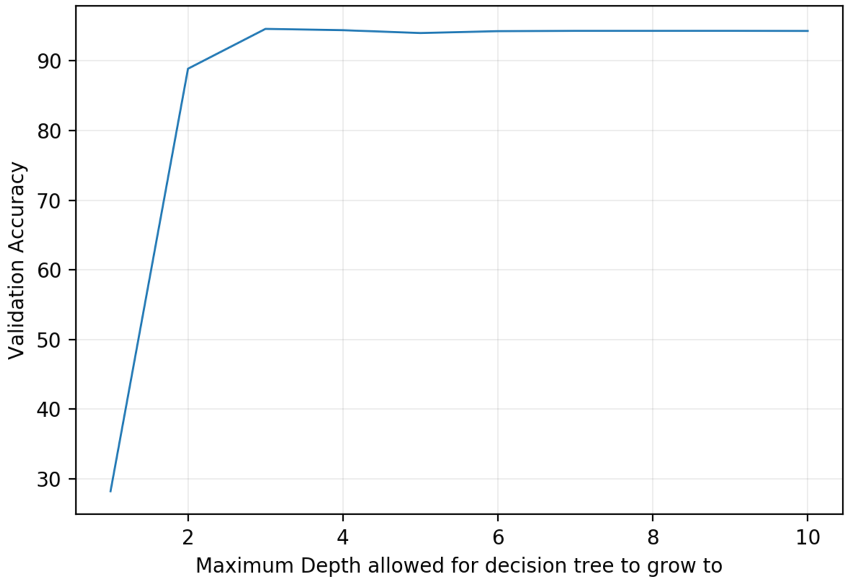


Figure 3: Validation Accuracy for different depths on Iris Dataset

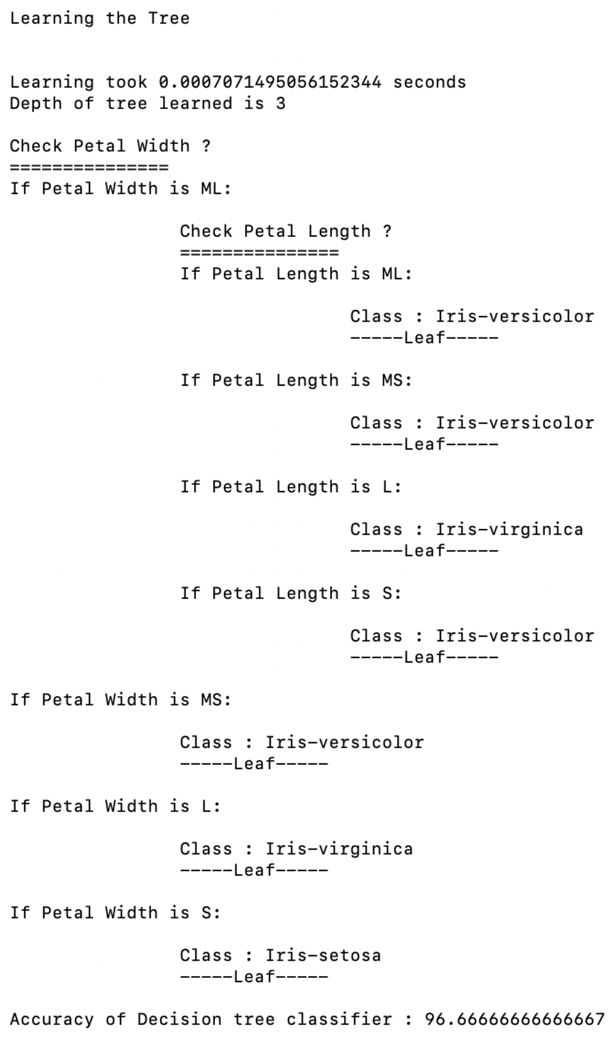
training on them and then validating on the other 25%(which is 20% of total data). I do the experiments on this 80% data which is set aside 100 times and report the average accuracy for each depth.

4.2.3 Result of analysis for depth of tree on Iris dataset

Figure 3 shows the validation accuracies plot against the depth of tree. As one can see, post depths of three, there isn't any significant improvement. Infact the maximum accuracy is at depth 3. If we don't set any depth then the tree grows till depths of 5. But since our validation accuracy tells that a depth of 3 is better, we will use that as the maximum allowed depth for iris dataset. On running the test for depths of 4 and 5 we observe that both give the exact same accuracy. **Moreover a decision tree of depth 3 only uses 2 attributes (Petal Width, Petal Length). This means the information about sepal width and sepal length does not provide any additional benefit to accurately predicting the iris class.** Figure 4 shows tree output of program and how only using 2 of the attributes by restricting to a maximum depth of 3, we get a more simpler tree and achieve the same accuracy having only 7 leaf nodes(If the tree does not have any restriction on max depth it grows with 16 leaf nodes, a rather complicated tree and overfitted tree for such a small dataset)

4.2.4 Validation of Splitting criteria for tree

I try with two different splitting criteria, Entropy and gini impurity. The same validation setup is used for splitting criteria as for depth. Table 1 provides the different accuracy for each of the splitting criteria. Both are very much the same but gini impurity does slightly



```
Learning the Tree


Learning took 0.0007071495056152344 seconds
Depth of tree learned is 3

Check Petal Width ?
===============
If Petal Width is ML:

                Check Petal Length ?
                ===============
                If Petal Length is ML:

                                Class : Iris-versicolor
                                -----Leaf-----

                If Petal Length is MS:

                                Class : Iris-versicolor
                                -----Leaf-----

                If Petal Length is L:

                                Class : Iris-virginica
                                -----Leaf-----

                If Petal Length is S:

                                Class : Iris-versicolor
                                -----Leaf-----

If Petal Width is MS:

                Class : Iris-versicolor
                -----Leaf-----

If Petal Width is L:

                Class : Iris-virginica
                -----Leaf-----

If Petal Width is S:

                Class : Iris-setosa
                -----Leaf-----

Accuracy of Decision tree classifier : 96.66666666666667
```

Figure 4: Decision tree output for Iris with maxdepth restricted to 3

better.

Splitting Criteria	Accuracy
Entropy	94.41
Gini Impurity	**94.42**

Table 1: Validation Accuracy for Splitting Criteria

4.2.5 Results Decision tree learning algorithm on Iris

a) Achieved a **testing accuracy of 96.67%**

b) Validated the depth of tree and found that **a depth of 3**(which **only uses two of the four attributes** of data is best).

c) **If no max depth is provided the tree grows to depth 5 and uses all four attributes but the accuracy is same as the one with depth 3**

d) **Entropy and Gini Impurity both perform equally** well with gini impurity performing a tad bit better.

e) **Time taken for learning is very small because of the small size of dataset - 0.7 ms**

4.3 Connect-4

We will now talk about the analysis of applying the decision tree learning algorithm on connect-4 dataset. Since connect-4 data set consist of a whole lot more attributes , we will need to use validation again here to ensure that we do not let the tree grow to a depth where it starts overfitting the data and hence loses the capability to generalize well. The experiment setup is exactly the same wherein we use a 60/20/20 split for train/validation/test. Since the dataset has about 60K rows this augurs well for our training more complex trees.

4.3.1 Result of analysis for depth of tree on Connect-4 dataset

Figure 5 shows the validation accuracy plot against depth on the Connect-4 dataset. Because for larger depths the tree takes a long time to learn we only repeat the experiment for 5 runs and provide average accuracy. Given that from the plot post depth 11 accuracy drops, this means that after that whilst the tree is indeed getting more complex this additional complexity provides no benefit in terms of accuracy or generalization and on the contrary decreases it. Hence it would be best if we restrict the tree to grow to only a depth of 11.

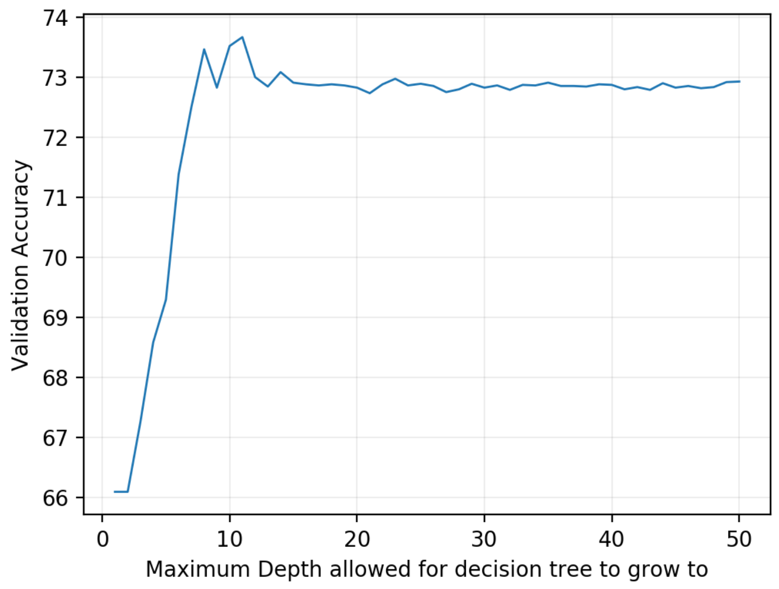


Figure 5: Validation Accuracy for different depths on Connect-4 Dataset

4.3.2 Validation of Splitting criteria for tree

Same as Iris, I try with two different splitting criteria, Entropy and gini impurity. The same validation setup is used for splitting criteria as for depth. Table 2 provides the different accuracy for each of the splitting criteria. These datapoints are gathered based on average accuracies for 10 runs. As one can see that gini impurity performs better in terms of splitting criteria for this dataset on average.

Splitting Criteria	Accuracy
Entropy	73.47
Gini Impurity	**73.88**

Table 2: Validation Accuracy for Splitting Criteria

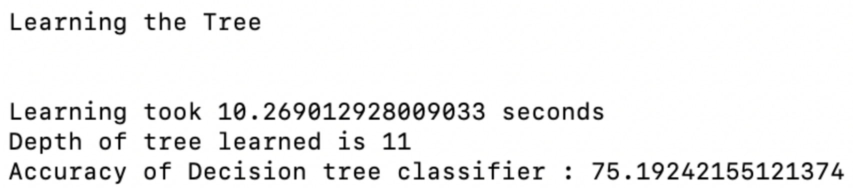
4.3.3 Results Decision tree learning algorithm on Connect-4

Figure 6 shows the output of running decision tree learning algorithm with accuracy being reported on the held out test set. I do not print the tree here because its too huge. One can obviously use the command options provided in ReadMe to see the tree of themselves.

a) Achieved an **accuracy of 75.19%**

b) Validated the depth of tree and found that **a depth of 11** is best using validation.

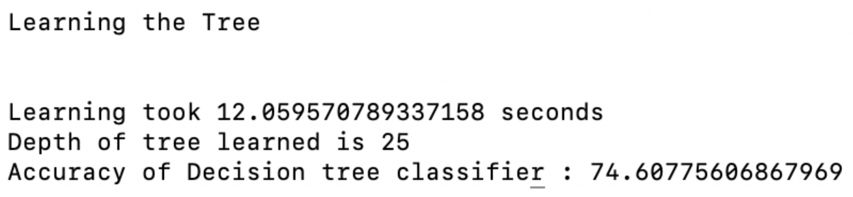
c) **If no max depth is provided the tree grows till depth of 25 but accuracy on test set drops to 74.6**



```
Learning the Tree


Learning took 10.269012928009033 seconds
Depth of tree learned is 11
Accuracy of Decision tree classifier : 75.19242155121374
```

Figure 6: Decision tree output for Connect-4 with maxdepth restricted to 11



```
Learning the Tree


Learning took 12.059570789337158 seconds
Depth of tree learned is 25
Accuracy of Decision tree classifier : 74.60775606867969
```

Figure 7: Decision tree output for Connect-4 with no maxdepth restrictions

d) **Entropy and Gini Impurity both perform equally** well with gini impurity performing a tad bit better.

e) **Time taken for learning is 10 seconds because of the large size of dataset**

f) **Time taken for learning if no depth restriction is provided is 12 seconds**

5 Neural Networks

Neural networks are a set of algorithms, modeled loosely after the functioning of human brain through neurons, that are designed to recognize patterns. Theoretically it has been proven that for any function there exists a neural net with certain architecture than can approximate that function to a very small amount of error. There are a lot of different elements of a neural network learning algorithm but the most important ones can be listed below :

a) Number of layers

b) Number of neurons in each layer

c) Activation function for layers - This is the element which introduces non-linearity in the architecture (in the absence of this , neural network would just be another linear model). Popular ones are sigmoid, tanh, ReLU, LeakyReLU.

d) Loss function - This function provides a measure to the neural net as to how good/bad its doing on the task as of now.

e) learning rate - Rate at which neural net is supposed to alter the weights by a change in the direction opposite to the gradient of loss(in effort to minimize the loss)

f) Regularization - Also called weight decay. In one version of regularization a function of weight is added to the cost in order to ensure that the neural network does not learn weights that are very huge. Other techniques are also present like dropout which randomly switch off neurons to ensure that networks do not overfit and also ensure that all neurons learn something from the data and the entire neural network does not just end up relying on few collection of neurons(and switch off the other ones) to learn the task,

My implementation of neural network has the following specifications:

a) **Converting data to one hot encoding** : Since I only model the network to deal with categorical data, I convert data to oone hot encoding schemes. This ensures that categorical data are converted to their vector forms. For example Iris dataset has petal length as S/MS/L/ML. I convert these to vectors of length 4 and represent S as [1 , 0 , 0, 0] , MS as [0, 1, 0, 0] and so on.

b) **Sigmoid function as activation** : My model requires to learn classification tasks and hence having the activations as sigmoid also gives me the cushion of not using the softmax layer and using the last layer activations as probability values itself(Since sigmoid ranges between 0 to 1).

c) **Cross Entropy Loss :** For evaluating how good my model is , I calculate the classification loss using cross entropy loss.

d) **Mini Batch SGD** I use mini batch stochiastic gradient descent for optimization procedure. I validate the batch size for each dataset.

5.1 Aima-Restaurant

Since AIMA-Restaurant is a dummy dataset with only 12 rows, I did not apply the Neural network learning algorithm on this dataset.

5.2 Iris

In this section I will discuss the analysis of applying neural network learning algorithm on the Iris dataset. Since the iris dataset comes with only 150 samples, we must be very careful on not allowing neural network to overfit disabling it from generalizing well to unseen data. Hence, I validate all the different aspects of the nerural network learning algorithm to optimize them.

5.2.1 Experiment Setup

I use the following experiment setup for validating the different parameters for neural networks on iris dataset.

a) Divide the entire data into train, test, validate split of 60/20/20

b) Train the decision tree on the 60% and validate the hyper parameter of depth and splitting criteria on the validate set.

c) Train the decision tree using the validated hyper parameters on the 80% (keeping test data untouched)

d) Test and report the accuracy of decision tree on the 20% of data.

Following are the parameters that I validate and optimize using validation set.

a) Learning Rate

b) Number of Layers

c) Number of Neurons in each layer

d) Batch Size

5.2.2 Validation of Learning Rate

Though the dataset is small for Iris only having 150 rows, I went ahead and ran validation experiments for different learning rates. I keep 20% of data as test data aside, and then ran validation experiments for depth by randomly choosing 75% of remaining data(this is equivalent to 60% of total data) and training on them and then validating on the other 25%(which is 20% of total data). I do the experiments on this 80% data and report the validation accuracy for learning rate..

5.2.3 Result of analysis for Learning rate on Iris dataset

Table 3 shows the validation accuracies against the learning rate. The architecture for all learning rate was kept at having 1 hidden layer with 100 neurons and batch size of 50 for 50 epochs. With slower learning rates, it takes much longer to achieve the same generalization and in some cases it never converges at all. I chose the learning rate of 0.1 since that was giving me the best validation accuracy.

Learning Rate	Validation Accuracy after 50 epochs
0.1	**93.33**
0.05	93.33
0.01	90
0.005	56.66
0.001	26.66
0.0005	20.0
0.0001	36.66

Table 3: Validation Accuracy for different learning rate

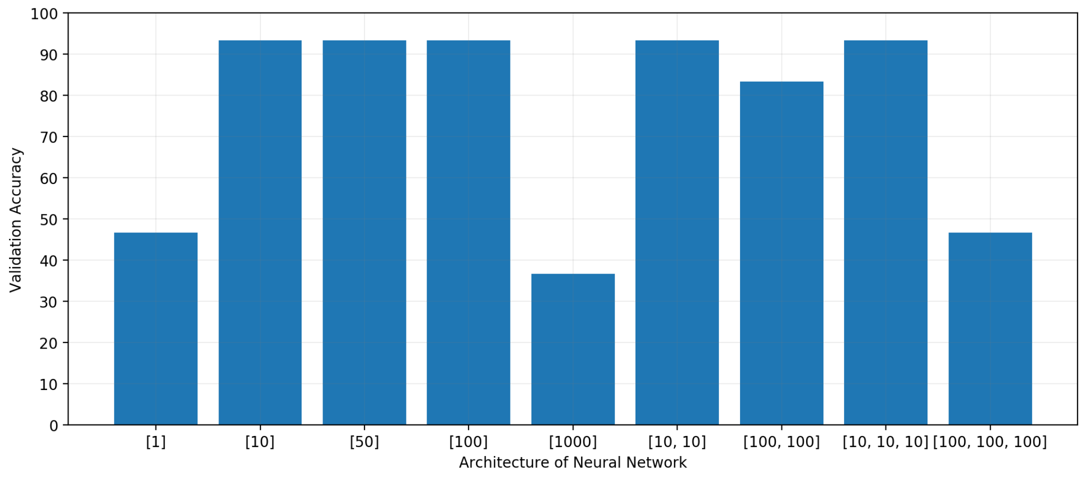


Figure 8: Validation Accuracy for different architecture of hidden layers

5.2.4 Validation of number of layers and number of neurons

I use the same setup as before for validation. I try with following different architectures.

a) 1 layer with 1 hidden neuron

b) 1 layer with 10 hidden neurons

c) 1 layer with 50 hidden neurons

d) 1 layer with 100 hidden neurons

e) 1 layer with 1000 hidden neurons

f) 2 layer with 10 hidden neurons each

g) 2 layer with 100 hidden neurons each

h) 3 layer with 10 hidden neurons each

i) 3 layer with 100 hidden neurons each

Figure 8 provides the different accuracy for each of the architectures. We can see that the dataset isn't large enough to make learning with large neuron sizes or huge layers achievable. **The best accuracy that we get on validation set is for 1 layer with 10 neurons each**. That network itself is good enough to get the same accuracy as any other architecture.

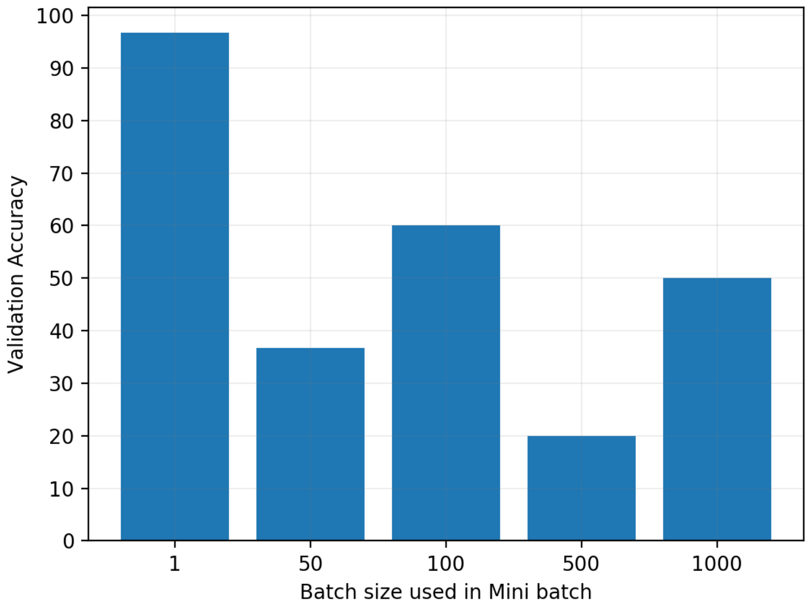


Figure 9: Validation Accuracy for different batch sizes

5.2.5 Validation of batch size

I use the mini batch algorithm for gradient descent and an important aspect of that is batch size. How many rows will I collect the gradient for before making an update to my weights and bias values. I tun my experiments for 75 epochs with architecture of one layer with 10 neurons and learning rate of 0.1 Figure 9 shows the plot of accuracy and batch sizes I tries. One can see that the best batch size is actually 1, which makes it equivalent to stochiastic gradient descent.

5.2.6 Results of applying Neural network on Iris dataset

Figure 10 shows the snapshot of output when the neural net is ran on Iris dataset

a) Achieved an **accuracy of 96.67%**

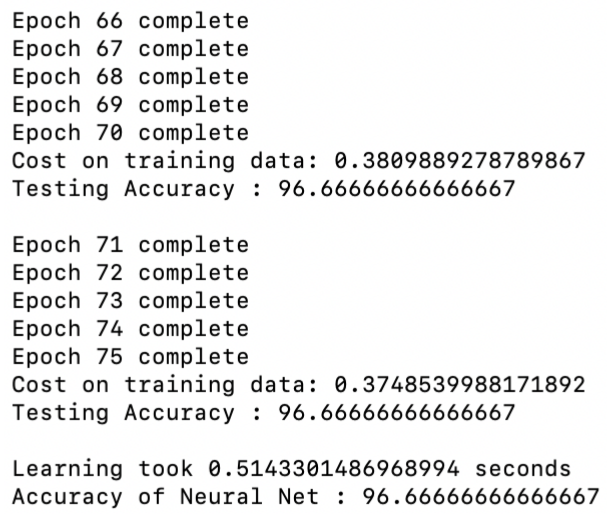
b) Validated the learning rate and found that **learning rate of 0.1 is best**

c) Validated the batch size and architecture and found that **batch size of 1 with 1 layer neural net and 10 hidden neurons is sufficient for this dataset**

d) **Time taken for learning is small because of small dataset size - 0.5 seconds**

5.3 Connect-4

I will now discuss my analysis of applying the neural network learning algorithm on connect-4 dataset. The experiment setup is exactly the same wherein we use a 60/20/20 split for



```
Epoch 66 complete
Epoch 67 complete
Epoch 68 complete
Epoch 69 complete
Epoch 70 complete
Cost on training data: 0.3809889278789867
Testing Accuracy : 96.66666666666667

Epoch 71 complete
Epoch 72 complete
Epoch 73 complete
Epoch 74 complete
Epoch 75 complete
Cost on training data: 0.3748539988171892
Testing Accuracy : 96.66666666666667

Learning took 0.5143301486968994 seconds
Accuracy of Neural Net : 96.66666666666667
```

Figure 10: Snapshot of output of Neural net on Iris dataset

train/validation/test.

5.3.1 Result of analysis for Learning rate on Connect-4 dataset

Table 4 shows the validation accuracies against the learning rate. The architecture for all learning rate was kept at having 1 hidden layer with 100 neurons and batch size of 50 for 75 epochs. I chose the learning rate of 0.05 since that was giving me the best validation accuracy.

Learning Rate	Validation Accuracy after 75 epochs
0.1	82.85
0.05	**83.42**
0.01	81.87
0.005	80.74
0.001	76.03
0.0005	74.25
0.0001	65.92

Table 4: Validation Accuracy for different learning rate

5.3.2 Validation of number of layers and number of neurons

I use the same setup as before for validation, though I try with a smaller set of architectures because of large time it takes for learning. Figure 11 provides the different accuracy for each of the architectures. We can see that since the datset is large adding more layers does not detoriate the accuracy as much as in the case of smaller dataset like iris. However, **The best accuracy that we get on validation set is for 1 layer with 100 neurons each** rather than having more layers with larger neurons.

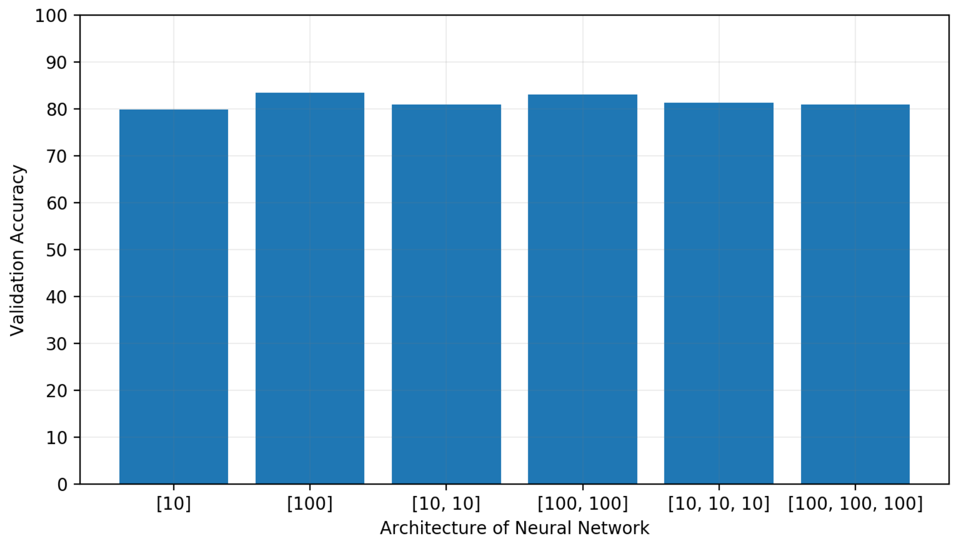


Figure 11: Validation Accuracy for different architecture of hidden layers

5.3.3 Validation of batch size

For validating batch sizes, I run my experiments for 75 epochs with architecture of one layer with 100 neurons and learning rate of 0.05 Figure 12 shows the plot of accuracy and batch sizes I try. One can see that the best batch size is again 1, same as Iris dataset.

5.3.4 Results of applying Neural network on Connect-4 dataset

Figure 13 shows the snapshot of output when the neural net is ran on Connect dataset. I also trace the decrease in loss and increase in accuracy as more and more epochs happen. Figure 14 provides the plot of training loss with iterations(where each iterations means neural net has sweeps over 15000 examples) . Figure 15 provides the plot of testing accuracy with iterations. The loss at first decreases significantly but after a few iterations the rate of decrease is slow but it still continues to decrease. The results were obtained using the following setting :

a) Sigmoid activations and cross entropy loss

b) Batch size of 1

c) Learning rate of 0.05

d) 1 hidden layer with 100 neurons in it

e) Trained for 75 epochs

f) Weights were initialized using random initialization mechanism of numpy with a seed value of 11

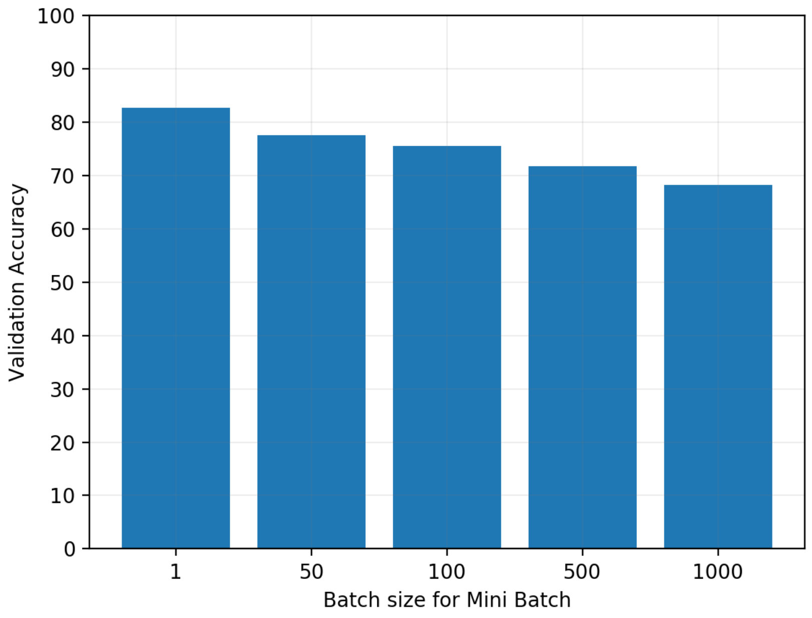


Figure 12: Validation Accuracy for different batch sizes

g) Training vs test split of 80/20 was used.

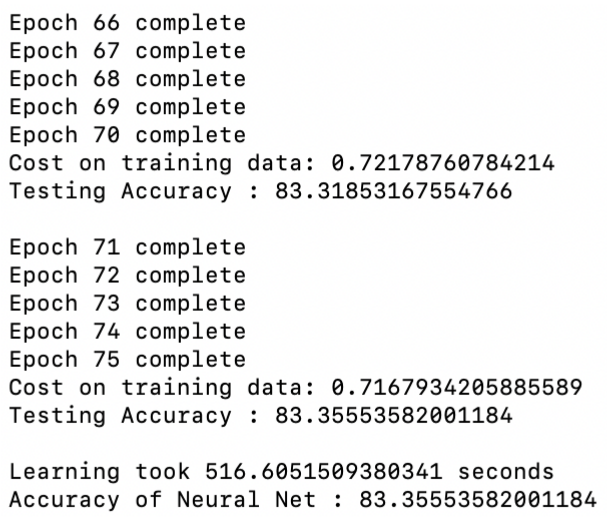
Here are the details of the analysis of applying neural network on Connect-4 dataset:

a) Achieved a **testing accuracy of 83.55%**

b) Validated the learning rate and found that **learning rate of 0.05 is best**

c) Validated the batch size and architecture and found that **batch size of 1 with 1 layer neural net and 100 hidden neurons is sufficient for this dataset**

d) **Time taken for learning is larger than other because of size of dataset - 516 seconds for 75 epochs**



```
Epoch 66 complete
Epoch 67 complete
Epoch 68 complete
Epoch 69 complete
Epoch 70 complete
Cost on training data: 0.72178760784214
Testing Accuracy : 83.31853167554766

Epoch 71 complete
Epoch 72 complete
Epoch 73 complete
Epoch 74 complete
Epoch 75 complete
Cost on training data: 0.7167934205885589
Testing Accuracy : 83.35553582001184

Learning took 516.6051509380341 seconds
Accuracy of Neural Net : 83.35553582001184
```

Figure 13: Snapshot of output of Neural net on Connect-4 dataset

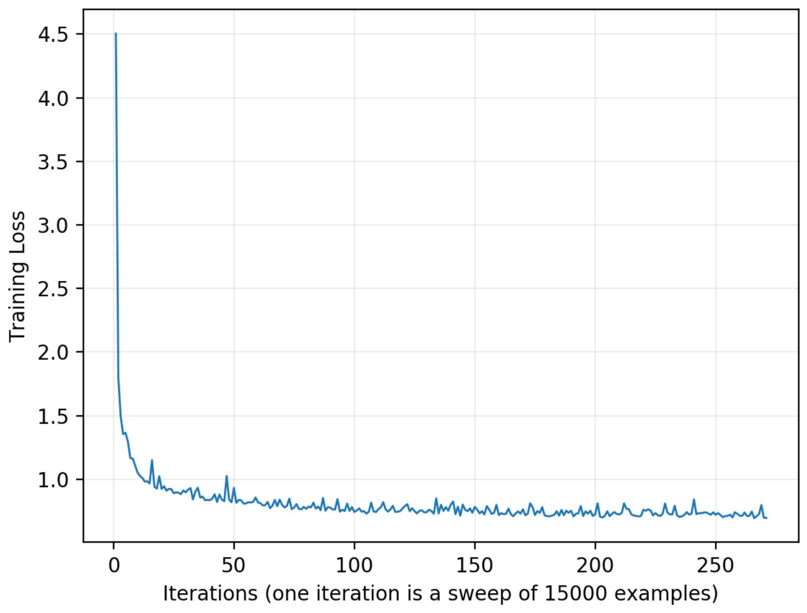


Figure 14: Plot of training loss with iterations

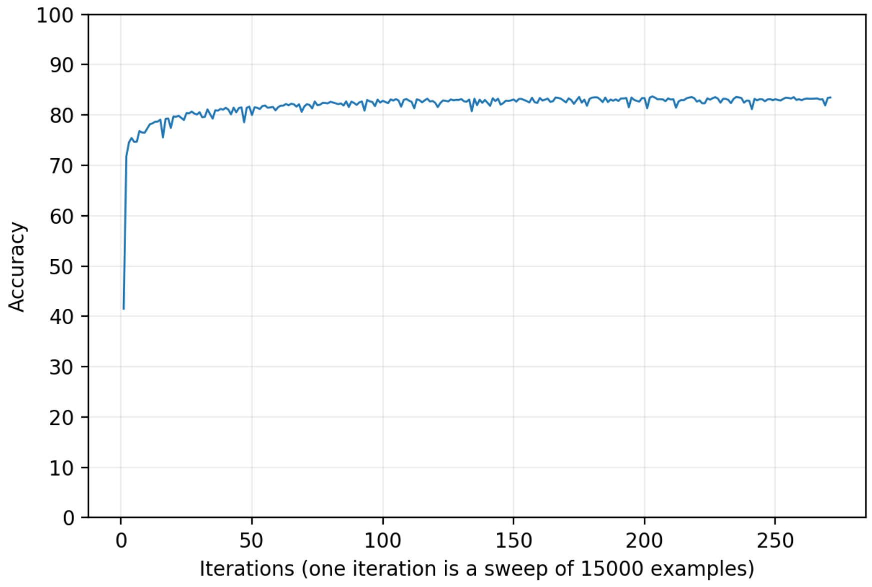


Figure 15: Plot of testing accuracy with iterations