COMP5329 - Deep Learning Assignment 1 Report

Zhecheng Zhong(SID:490319299)

Semester1 2020

1 Introduction

1.1 Aim of study

Given an input dataset with features and labels (here examples provide training data with 128 features and 10 labels), accomplish a multi-class classification task via building a multi-layer ANN. This classification combines activation, optimizations, weight decay, dropout, initialization, cross-entropy loss, mini-batch training and normalization. In the test section, the classification should succeed in predicting the testing data with a decent accuracy. The report content starts with method design, then illustrating on experiments, following with discussion and conclusion

1.2 Importance of the study

Importance of this study can be emphasized in many ways: firstly, it is highly valuable for consolidating the concept by combining with real experiments. Second, considering the huge variety of problem domains MLP is applied in, it is intriguing to suspect that specific problems call for single or a set of specific methods[1]. Knowing how to build best-fit model by matching functions with specific data is significant in deep learning aspects.

2 Methods

This section gives detail about data pre-processing, project structure design, and method principles. The method principles sub-section is affiliated to the structure design as a detailed explanation.

2.1 Data pre-processing

A one-hot is a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0)[2]. The training label is transferred into a one-hot form as a way to handle discrete data. Data from input is split into train, validation and test parts.Common ratios used are: 0.8 train, 0.1 val, 0.1 test. The data is transpose as each column representing an sample with length of column features. Data normalization is implemented as the last step of data pre-processing. As a result, both features and label data are normalized and column based.

2.2 Design

Project structures are defined based on the principle of multi-layer neural network. In layer creating process, layers record initialization and activation. In forward and back propagation, modules are activation, batch normalization, dropout, regularizer and cost.

2.2.1 Multi-layer neural network

Figure 1 shows the multi-layer structure of the project. They are input layer, three hidden layer with decreasing neurons(128-96-64-32-output), and one output layer. Experiments suggest to choose a three hidden layers structure, and this will illustrated in latter part.

Within each layer, it starts with initializing the weight following batch normalization function and activation function. Hidden layers utilize He as initialization method and relu as activation method. The last layers applies Xavier as weight initialization and

softmax as final activation method. During the transmit between each layer, dropout method is used with set droupout rate to randomly abandon some neurons. Cost, regularization and gradient are part during forward and backward.

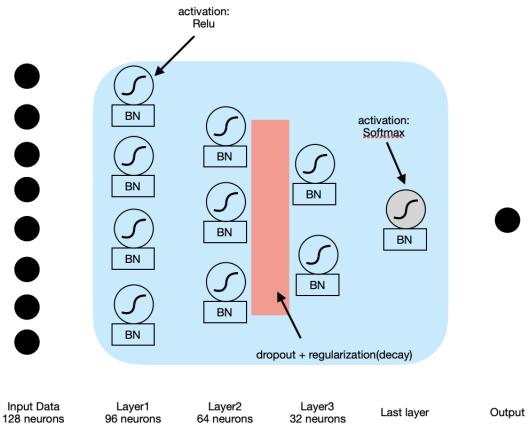

Figure 1: Multi-layer structure of the project

2.2.2 Create model and layer

Since this is multi-classifier problem, the cost function should be cross entropy. The first step is to specify regularization and the way of calculating gradient descendent —- or we say the optimization. Based on the multi-layer structure of the project, layer is created with two main parameters: way of initializing weight and activation method. Figure 2 draws the

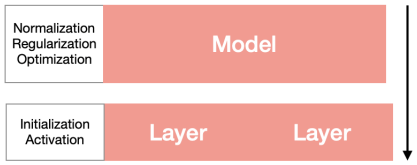

Figure 2: Model and layer creating

concept of model and layer creating. Figure 3 shows the logical path in project code.

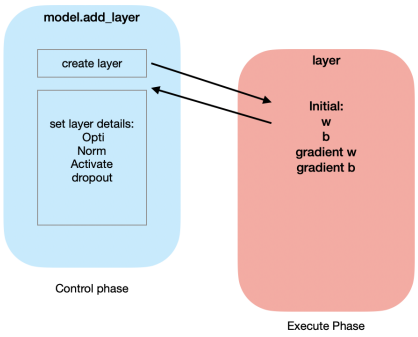

Figure 3: layer creating logical path

2.2.3 Forward propagation design

The input data is fed in the forward direction through the multi-layer network. Each hidden layer accepts the input data, processes it as per the normalization and activation function and passes to the successive layer. For each epoch, the forward is executed by each batch with assigned batch size (Figure 4).

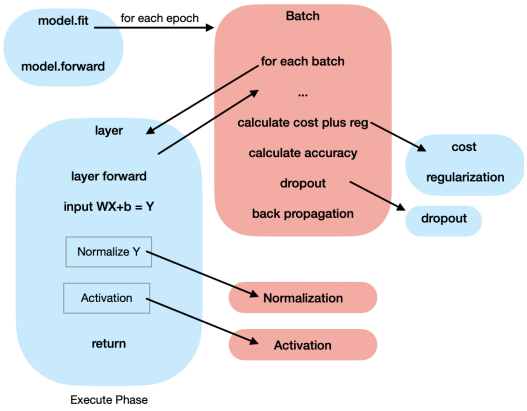

Figure 4: forward propagation

2.2.4 Back propagation design

Back propagation is about understanding how changing the weights and biases in a network changes the cost function. In this assignment, cross entropy loss function is written in the code to become the role of

calculating cost, and applying momentum method to update the new gradient of weight W and b. Details of implementation are drawn in Figure 5.

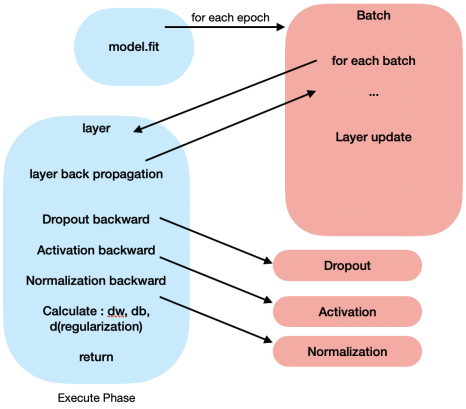

Figure 5: back propagation

2.2.5 Predict design

After building the model, test data would goes through the forward process without training and it is saved to /output path. Since the value inside indicates the maximum probability that the predicted object represents the current class, argmax function would be used to retrieve label value.

2.2.6 Method Principles

Weight initialization. Xavier's derivation process is based on several assumptions: the activation function should be linear and activation value is symmetric about 0, which do not apply to nonlinear activation functions such as ReLU and sigmoid. So the hidden layers pair with He initialization. The last layer applies Xavier with softmax activation function. The momentum optimization method is implemented to obtain the gradient of weight.

The last hidden layer produces output values forming a vector X. The output neuronal layer is meant to classify categories with a SoftMax activation function assigning conditional probabilities accompanying with cross entropy loss function. However, it takes time to calculate the derivation of softmax with cross entropy, instead, a simplified equation is used to ease the computing process.

3 Instruction on how to run the program

Required environments: python 3.7.4, matplotlib 3.2.1, numpy 1.18.3 and h5py 2.10.0

Three folders are in the project, including Algorithm folder (containing python codes), input folder (putting input data, label data, and test data), output folder(the prediction output of test data).

TO run the program, please ensure these three folders are in the same directory. Put trainunderscore128.h5, trainunderscorelabel.h5, testunderscore128.h5 data with certain name into the input folder. At the terminal, run "python main.py", and the result file will be created in output file. It is acceptable to change the directory of input directory if goes to the /Code/Algorithm/parameter.py directory.

4 Experiments and results

4.1 Accuracy

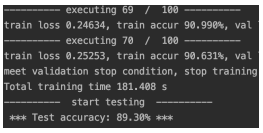

Figure 6: result

Having all the methods settled in previous section, define the validation stop point as validation accuracy reaching 89.8 %, we get the final test accuracy being 89.30% within 182 seconds.(shown in Figure 6) The training has an early stop in 70th out of 100 epoch training cycle. Figure 7 depicts the change of accuracy and loss between training and validation.

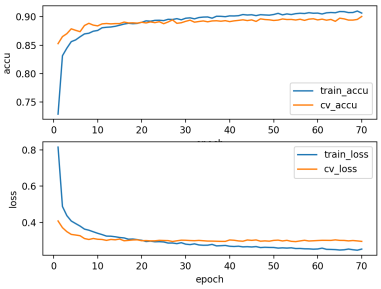

Figure 7: accuracy and loss between training and validation

Initialization	Activation	Learning rate	Decay	Dropout	Test accuracy
He	Relu	0.01	0.0005	0	87.6%
He	Relu	0.01	0.0005	0.05	89.20%
He	Relu	0.01	0.0005	0.1	89.33%
He	Relu	0.01	0.0005	0.15	88.98%
He	Relu	0.01	0.0005	0.2	88.78%
He	Relu	0.01	0.0001	0.1	88.87%
He	Relu	0.01	0.001	0.1	89.20%
He	Relu	0.01	0.005	0.1	87.68%
He	Relu	0.005	0.0005	0.1	89.02%
He	Relu	0.02	0.0005	0.1	88.92%
He	Relu	0.05	0.0005	0.1	88.48%
He	Relu	0.1	0.0005	0.1	87.74%

Figure 8: hyper parameter selection

4.2 Extensive analysis (Comparison)

Some methods of experiment models are defined and assigned with static values, while other hyper parameters and methods such as initialization and activation being able to do parameter adjustment. By controlling variates, in this section, some hyper parameters are adjusted to find the best fit model(figure 8).

5 Discussion

During the processing of implementing the project, some blurred concepts are implemented roughly and they are remain in discussion. Also, we have defined some latter improvement method to work on.

5.1 Validation stop point

It is vague to determine how to stop the training circulating using validation data set. We propose three choices: stop when validation accuracy >x; stop when train accuracy >validation accuracy and train loss <validation loss; stop when (train accuracy - validation accuracy) >x. So far the project embeds the first choice as implementation. On the other hand, Keras has function EarlyStopping to evaluating validation loss and accuracy. However, finding other better stop method should be in later study.

5.2 Use GridSearchCV to choose parameters

In section 4.2, finding optimize hyper parameter is time consuming. Not to mention the bias when choosing the best value — it is highly likely to obtain a local best option. Using GridSearchCV by building a parameter matrix, the function is able to find the best parameter for experiment.

6 Conclusion

In conclusion, based on training data, we built a three-hidden layer neural network model with Kaiming initialization and RuLU activation, having momentum optimization for weight gradient decay, applying batch normalization and regularization both in forward and backward propagation. 10 % of the training data is each split as validation data set and test data set, and as a result, the test data have a 89.33% matching rate using the designed model. The predict data is stored in /output file.

7 Other (LATEX)

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. This report is written and compiled using online LATEX application.

References

[1] Bekir. *Performance analysis of various activation functions in generalized MLP architectures of neural networks.* International Journal of Artificial Intelligence and Expert Systems, 2011.

[2] Harris. *Digital design and computer architecture (2nd ed.).* Calif.: Morgan Kaufmann, 2012.