CS573 - Lab2 Report

Phuong Nguyen

March 2020

1 Introduction

The fully-connected feed-forward neural networks experiment is in $MLPs.py$ file and its result can be seen in $MLPs\_result.py$. The Convolutional Neural Networks is in the CNNs.py file.

In this lab, we use $RandomizedSearchCV$ method in $sklearn$ library to search for the best hyper-parameters of the MLPs. You can use $GridSearchCV$ but it is slower and gives the same results most of the time.

The hyper-parameters we tested for them are (excluding loss functions and activation hidden units):

- number of epochs
- batch size
- number of hidden layers
- number of hidden units in each layer
- learning rates
- momentum rates
- whether or not scaling input

The convolutional neural networks experiment is in the CNNs.py file and the hyper-parameters we tested for them are:

- number of epochs
- batch size
- number of Dense layers
- whether or not scaling input
- filter size

- kernel size
- pooling size
- strides
- dropout rate
- number of hidden units in Dense layer

2 Experiment with fully-connected feed-forward neural networks (MLPs)

(a) Sum-of-squares error (mean-squared-error) vs. cross-entropy error function:

The best hyper-parameters for MLPs with sum-of-squares error are: epochs: 350, batch size: 256, # of hidden layers: 1, # of hidden units in each layer: 30, learning rates: 0.01, momentum: 0.99 without input scaling. The results are: Training overall accuracy: 0.981, Testing overall accuracy: 0.946. The running time is: 2.54s.

The best hyper-parameters for MLPs with cross-entropy error are: epochs: 200, batch size: 256, # of hidden layers: 1, # of hidden units in each layer: 35, learning rates: 0.0001, momentum: 0.99 without input scaling. The results are: Training overall accuracy: 0.97, Testing overall accuracy: 0.94. The running time is: 3.5s.

The cross-entropy error function should in theory give better results than the sum-of-squares error. This is because in classification we work with very particular set of possible output values thus sum-of-squared is badly defined (as it does not have this kind of knowledge thus penalizes errors in incompatible way). But due to early stopping, sometime sum-of-squares error give better results.

(b) *tanh* vs. *ReLU* hidden units:

The best hyper-parameters for MLPs with *tanh* hidden unite are: epochs: 150, batch size: 32, # of hidden layers: 1, # of hidden units in each layer: 20, learning rates: 0.01, momentum: 0.9 without input scaling. The results are: Training overall accuracy: 0.978, Testing overall accuracy: 0.93. The running time is: 6.13s.

The *ReLU* hidden units give better results and run faster than the *tanh*. This is because in *tanh*, a *tanh* activated neuron may lead to saturation and cause vanishing gradient problem. Also, the derivative of *tanh* is compute-intensive because of e^x. Meanwhile, *ReLU* does not saturate for positive value of weighted sum of inputs and is easier to compute derivative.

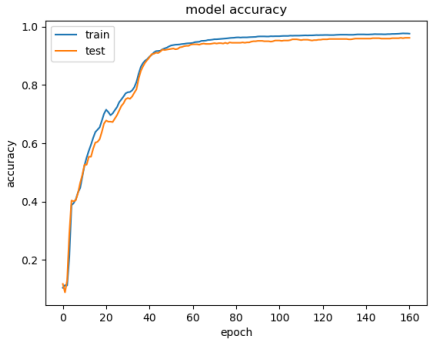

Figure 1: CNNs' Accuracy vs Epochs with cross-entropy.

3 Experiment with convolutional networks (CNNs)

The best hyper-parameters for CNNs are: epochs: 10, batch size: 32, # of Dense layers: 2, # of hidden units in each Dense layer: 128, 50, filter: 32, kernel size: (5,5), strides: (1,1), pool size:(2,2), dropout rate: 0.2, with input scaling. The results are: Training overall accuracy: 0.99, Testing overall accuracy: 0.97. The running time is: 3.1s

CNNs are basically MLPs with special structure. The training and testing accuracy of CNNs reache the high value very fast (around 10 epochs) and then saturate there. The best result of both network is pretty comparable with CNNs is slighter better.

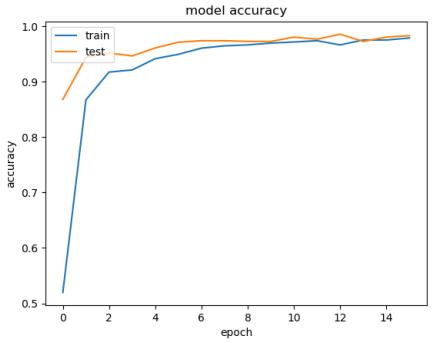

Figure 2: CNNs' Accuracy vs Epochs.