Intro to Artificial Intelligence
CS440_Assignment_2 report
08/12/2018
Pr. Rahul Shome
Group Members: Zetao Yu, Chenjie Wu, Jinghui Gao, Shenghan Hu

1. **Building the model**

   We use softmax regression as a training model to achieve the accuracy over 90%. Generally, the softmax regression has two steps: first we add up the evidence of our input being in certain classes, and then we convert that evidence into probability.

   -Layers: There are three layers in the model, which are input layer, hidden layer and output layer.

   -Nonlinearities: As a nonlinear regression, softmax regression basically exponentiate its inputs and then normalize them. (Detailed function attached)

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

   -Input & output: In this model, the input would be a vector of int with length 784, in which each element represents the pixel in a 28*28 image (i.e. 0: uncolored, 1: colored). The output would a probability distribution over all result classes. In this case, the set of output classes would be {0,1,2,3,4,5,6,7,8,9}. By knowing the probability distribution, we can choose the class with highest probability as our estimation of the digit.

2. **Loss and accuracy function**

   We use “cross-entropy” as our loss function.

   -Loss function formula:

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

   In neural network, loss function tells us how far off our model is from our desired outcome. It quantifies the error. Although changing the choice of loss function cannot significantly increase the accuracy of the model given that both loss functions are valid, by improving the learning algorithm in neural network to minimize this loss, we can get the smaller error margin, thus the better model with higher accuracy.

   -Accuracy function formula:

   $Accuracy = \frac{\sum_N prediction}{N}$, where N is the number of test digits and variable prediction is either 0 or 1 for wrong prediction and correct prediction.

The accuracy function calculates the mean of prediction results to quantify how accurate the model can recognize digits. Its output is a probability between 0-1 which represents the rate of correctly predicting digit.

3. **Training Function**

-We use gradient descent algorithm with a learning rate of 0.5 as optimizer.

-With lower learning step, the slower we travel along the downward slope, which costs us more time to converge. But in this way, we also lower our probability to miss a local minimum, which is a good thing. Conversely, higher learning step costs us less time but increases the probability to miss a local minimum.

4. **Training Loop**

In each training loop, we train the model in "train, validate, test" three steps. Approximately, we feed 100 batches of data to train the model using optimizer, then 1 batch of data to validate and 1 batch of data to test and print the accuracy. We repeat this whole process for 5 times.

<u>-Definition of three sets:</u>

Training set: A set used for learning, which is to adjust the weights of parameters.

Validation set: A set used to validate the accuracy of our current model and tune the hyperparameters (in hidden layer of neural network) to improve its performance.

Test set: A set used only to assess the performance (generalization) of the final model.

<u>-Purpose and reasoning:</u>

By breaking out the dataset into three parts, we can use the data efficiently and avoid overfitting. That means, after do training with the training dataset, we adjust the weights of all variables and get a model which is mostly fit with current dataset. However, we have no idea how this model performs in other dataset in the future. Also, we cannot know the performance of hyper parameters in the neural network hidden layers. At this time, validation dataset provides us a set of unbiased data to check whether our model is good enough for general use. If not, it can also adjust the hyper parameters to improve the performance of model. After validation, testing dataset, which is also unbiased, calculates the accuracy of our model, which tells us whether to do more training again to adjust weights. In this step, the model is fixed and would not be adjusted with testing data.

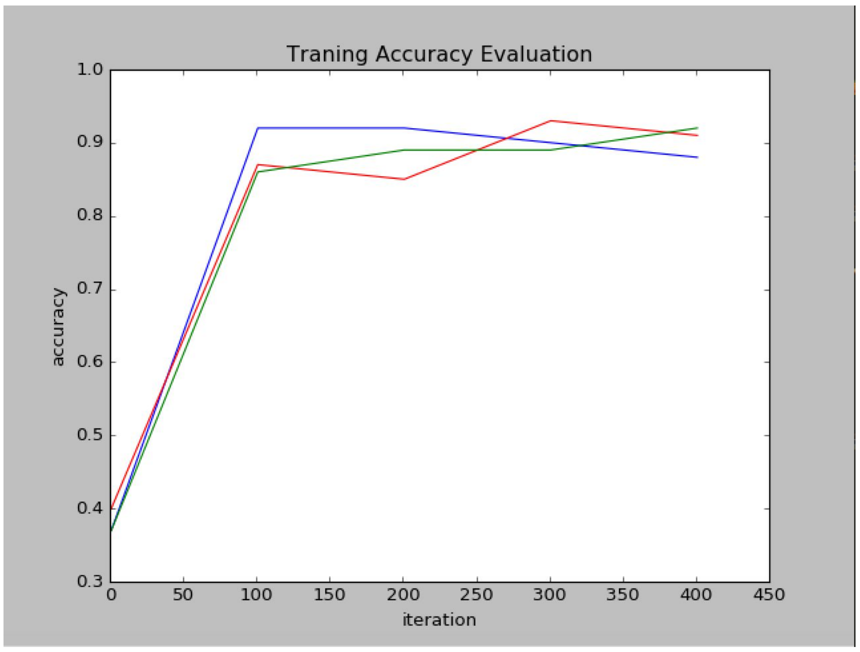
The downside of such split is that, looping with three steps takes more time to build a model, which might not be time efficient. Moreover, the proportion of whole dataset for each part can be hard to determine. A balance point should be found to let each step efficient.

With both pros and cons, under the current circumstance of the world that huge information stored and shared in Internet, such mode of learning can ensure the accuracy of the model. Although it might sacrifice time efficiency and the choice of test data might have high stochasticity, it is worth exchanging a more accurate model for future use.

5. Plot

<u>100 training data, 1 validation data and 1 test data:</u>

*unit: 1 batch = 100 sets; blue line: training, red line: validation, green line: testing



<u>-Observation:</u>

Three lines are approximately same after converging to local maximum accuracy. But if zoom in, we can observe that, with more dataset feeding, the training accuracy slightly decreases while the validation and testing accuracy slightly increases.

<u>-Reasoning:</u>

It is because after training with 100 batches of training data, the model is designed for this 100 batches of data, in other words, a little overfitting for those data, which leads to a unreal higher accuracy towards training data while a relatively lower accuracy towards validation and testing data (i.e. the points on curves with 100 of x-axis). However, with more loops executing containing validation and testing, such overfitting is being corrected and causes the increasing and decreasing of the curves.

In theory, I would expect the three lines finally be at the same accuracy, in which case this model covers and considers all types of cases by identifying the entire set of characteristics among three datasets and other case outside training data.

6. **Sensing**

-The accuracy in experiment is lower than the one recorded in the accuracy plot

-One possible reason is Accuracy Paradox, saying that higher accuracy may not lead to better prediction. In this case, when the base rate of certain digit in training data is way lower than others (i.e. < 2%), even if the prediction of this digit is all wrong in the testing step, the entire accuracy can still be above 90% because the test case of this digit has very little ratio in the whole dataset which makes little effect. But in the future, every time we meet this digit, we cannot predict it correctly.

Another possible reason is that there are more variations outside training data not being covered, which might contain more undiscovered characteristics.

7. **Extra Credits**

We can achieve around 97% accuracy by using expanded model. Basically we add two more convolutional max pooling layers, densely connected layer and readout layer to increase the complexity of model. We also change the rule for initializing weights to achieve less noise. Among all the changes and addition, adding more layers to the neural network significantly improves the accuracy of our model. However, the rate of the improvement is not linear, it drastically increases at first and slow down its step in the halfway. The rate keeps decreasing upon reaching the convergence.

(Code implementation is included in zip file, called tf_train_model_ASSIGNMENT_FILE_extra.py. One can test it by simply changing the 'isExtra' boolean in train_launch.py to True and run train_launch.py to train the data)