Due Date: February 12th, 2018

Problem 1

In this problem, we will build a Multilayer Perceptron (MLP) and train it on the MNIST handwritten digit dataset [1].

Building the Model Consider an MLP with two hidden layers where the number of hidden units as h^1 and h^2. For the MNIST dataset, the number of features of the input data h^0 is 784. The output of the neural network is parameterized by a softmax of $h^3 = 10$ classes.

1. Build an MLP andd choose the values of h^1 and h^2 such that the total number of parameters (including biases) falls within the range of [0.5M, 1.0M].
2. Train the MLP using the probability loss (*cross entropy*) as training criterion. We minimize this criterion to optimize the model parameters using *stochastic gradient descent*.

In the following sub-questions, please specify the *model architecture* (number of hidden units per layer, and the total number of parameters), the *nonlinearity* chosen as neuron activation, *learning rate*, *mini-batch size*.

Initialization In this sub-question, we consider different initial values for the weight parameters. Set the biases to be zeros, and consider the following settings for the weight parameters:

- **Zero**: all weight parameters are initialized to be zeros (like biases).
- **Normal**: sample the initial weight values from a standard Normal distribution; $w_{i,j} \sim \mathcal{N}(w_{i,j}; 0, 1)$.
- **Glorot**: sample the initial weight values from a uniform distribution; $w^l_{i,j} \sim \mathcal{U}(w^l_{i,j}; -d^l, d^l)$ where $d^l = \sqrt{\frac{6}{h^{l-1}+h^l}}$.

1. Train the model for 10 epochs [2] using the initialization methods above and record the average loss measured on the training data at the end of each epoch (10 values for each setup).
2. Compare the three setups by plotting the losses against the training time (epoch) and comment on the result.

[1] Use the standard train/valid/test splits such as the one provided here.
[2] One epoch is one pass through the whole training set.

Learning Curves From now on, use the Glorot initialization method. In this subquestion and the next one, we consider different scenarios and model assumptions to explore the concept of generalization [3].

1. Find out a combination of hyper-parameters (model architecture, learning rate, nonlinearity, etc.) such that the average accuracy rate on the validation set ($r^{(valid)}$) is at least 97%.
2. Plot a figure of learning curves with the x-axis being the training time (epochs) and the y-axis being the accuracy. The plot should include the accuracy measured on both training and validation sets at the end of each epoch.
3. Train the model for 100 epochs.
4. Train the model for 100 epochs, this time double model capacity (in terms of number of parameters).
5. Comment on the result. (Hint: explain it using the bias-variance trade-off.)

Training Set Size, Generalization Gap, and Standard Error. Choose the best model and training specification you have designed so far in terms of validation accuracy.

1. Randomly draw subsets of the training data with $N_a = aN$ samples, where $N = 50000$ is the number of samples of the full training set of MNIST, for $a \in \{0.01, 0.02, 0.05, 0.1, 1.0\}$.

   For each subset, train a model for 100 epochs.

   (a) Keep track of accuracy measured on the i. training set, ii. validation set, iii. test set at the end of each epoch

   (b) Report the generalization gap defined as the difference between the training accuracy and the test accuracy ($G_a = r^{(train)} - r^{(test)}$) of the model with best validation performance [4].
2. Repeat this procedure 5 times and you should end up with a Table of size 5×5 (5 trials and 5 training set sizes).
3. Report the average generalization gap for each subset size and its corresponding standard error. Recall that the standard error of a statistic is the standard deviation of its sampling distribution.
4. Comment on the result.

[3] page 43 of slide00.

[4] Here we consider the snapshot at the end of each epoch throughout training as one model. Hence we are doing model selection (choosing the best out of 100 models) by seeing time as a hyper-parameter.

Problem 2

In this problem, we will build a Multi-layer Perceptron (MLP) and train it on the 20 Newsgroups [5]. 20 Newsgroups is a collection of roughly 20,000 documents taken from 20 different newsgroups (comp.graphics, sci.electronics, etc). The task is to predict which group the document comes from. Process the data as follows:

1. Each document is represented as a bag of words. Each document x_i is represented as a vector the size of the vocabulary (around 50k in our case). Each position x_{ij} is then the count of how many times the word w_j appears in document x_i.

Building the model For the following question, use an MLP with a single hidden layer of size 100, initialized using the Glorot initialization. A momentum [6] of 0.9 should also be used during the training. We will look at the effect that preprocessing can have on the results. Consider the three following preprocessing procedures:

- **No preprocessing**: The input of the model is the count of the different words.
- **tf-idf**: All the words in the documents are processed according to their tf-idf(Standard way to preprocess text document data). You can implement your own version of tf-idf if you prefer, but feel free to use the implementation from here. The focus is the importance of preprocessing.
- **Standardization**: All word counts in the document in the training and test set are normalized so that the mean is 0 and variance is 1: $d_j^i = \frac{d_j^i - u_{j_{train}}}{\sigma_{j_{train}} + \epsilon}$, where $\epsilon = 1e - 5$.

For the three processing procedures do a quick hyper-parameter search for the learning rate. Plot the accuracy on the training and test set for 20 epochs.

1. Briefly discuss the results,
2. and answer the following points in a few lines each:
   (a) Could the same learning rate could be used for all of the models? Why, or why not?
   (b) What might have happened if $\epsilon = 0$ for the Standardization preprocessing? Why? Besides modifying ϵ, how else could we deal with this problem?
   (c) What advantage does tf-idf has over basic word count approach that could help the model learn better?

[5] Use the standard train/test splits such as the one provided here. Partition 20% of the training set as your validation set.

[6] page 38 of slide02

Variance in training Using the same MLP as the previous question and the tf-idf preprocessing, do the following: Using **only** vanilla SGD, and a minibatch size of 1, record the loss on the training set for the first 5000 updates (not epoch!). Do the same thing with a minibatch size of 100 and plot them. The same learning rate should be used in both settings. It should also be high enough that the loss is clearly going down (≥ 0.1 is a good starting point).

Answer the following questions:

1. Given that the two models had updates of the same magnitude, how do you explain the variance and the difference in the loss?
2. If we could only have a minibatch of size 1, what technique could we use to reduce the variance during training? Explain the intuition behind it.

magnitude幅度