Building a Neural Network from Scratch

Andrew Dudley and Sailik Sengupta

November 10, 2017

1 Problem Statement

1.1 Building a Neural Network from Scratch

Deep Neural Networks (DNN) are powerful models that are able to approximate any function given enough data in order to perform classification and regression tasks. They use one or more hidden layers of "neurons" to learn the features of the given training data by utilizing backpropogation, which leverages the differentiability of a cost function in parameter space in an attempt to minimize the overall loss.

Many optimized frameworks exist that enable rapid development of deep neural networks with fully configurable architectures. To better understand the fundamentals of neural networks, however, it is best to first implement them from scratch. As such, this project seeks to develop a simple, configurable Python framework to build DNN models and then train those models using the canonical MNIST data set. An architecture specification has been provided which states that the model must trained using two hidden sigmoid layers with 256 units per layer, a softmax output layer, and negative log probability as the loss function to be minimized. Dropout - a method of disabling random neurons during the training process to prevent overfitting - should also be employed.

1.2 Investigation of Bounded Misclassification for Operational Security

While these deep models have been used with great success in recent years, they are also known to be highly vulnerable to adversarial attacks, in which the model can be made to intentionally misclassify an input by an imperceivable modification to that input. In order to provide security to safety-critical applications that deploy DNN, it is vital that procedures be developed that can increase the robustness of the neural networks to these attacks.

In this project, we aim to investigate the applicability of using cost-based loss functions as a means to reduce the *degree of misclassification* of both legitimate and adversarial inputs by using domain-specific distance metrics.

2 Current Progress

2.1 Deep Neural Network Framework

We have developed a neural network framework in Python that supports the following features:

- Specification of network architecture (input size, number of hidden layers, and type of layers)

- Dynamic building of network based on specified architecture.
- Layer-by-layer feed-forward computations using numpy matrices.
- Layer-by-layer back-propogation using stochastic gradient descent algorithm with dynamic programming to efficiently compute analytic gradients.
- Optional dropout parameter to enable stochastic dropout of neurons in the hidden layers.
- Adaptive learning rate to improve the convergence properties while training the model.
- Saving and loading of trained models

Thus far, DNN models trained on the MNIST data set using our framework have reached 98% testing accuracy.

2.2 Bounded Misclassification

We have compiled results from preliminary tests for using weighted loss functions for bounded misclassification.

3 Remaining Tasks and Timeline

Our neural network framework is almost complete. Majority of remaining work involves implementing the features necessary to complete our investigation of bounded misclassification using our own framework. Preliminary results generated using Tensorflow.

November 13	Add command-line argument functionality.
November 17	Implement graphing of loss function across iterations.
November 17	Implement custom parsing and sampling of MNIST data (currently utilizes tensorflow function for data loading.)
November 24	Complete implementation of weighted loss function feature within our own neural network framework.
November 27	Compile all performance results from different hyperparameters.
November 27	Design poster to present the results of our neural network models and our findings on misclassification bounds.

4 Acknowledgement

We hereby ackowledge that we have read, understood, and will abide by the Academic Integrity Policy as noted in the course syllabus and project requirements.