Neural Network Initialization

Vivek Rathod, Guifan Li

ABSTRACT

This is a report describing the experimental results on the performance of a shallow neural network on MNIST digits dataset with different weight initialization schemes.

NETWORK STRUCTURE

We built a shallow feed forward neural network with one hidden layer with *tanh* activation units as recommended in (LeCun et al. 2012)

$$1.7159 * \tanh\left(\frac{2}{3}x\right)$$

and an output layer with *softmax* activation units to model the problem as a multinomial classification problem:

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

The input layer has 784 units and the output layer has 10 units corresponding to the ten target classes.

TRAINING METHOD

The neural network is trained using mini-batch stochastic gradient descent (SGD) and back propagation with a batch size of 32. A constant learning rate of 0.01 is used. No regularization is applied. SGD is run for a fixed number of iterations and the model that performs best on the validation set at any time during training is used as the final model.

WEIGHT INITIALIZATION

Neural Networks are commonly initialized with weights sampled randomly from a uniform distribution such that the network begins operating in its linear regime. This ensures that the gradients are large enough when the training begins. In addition, it also helps the network to learn the linear features first and harder

non-linear features later. Weights at each layer are initialized at each layer using the following heuristic:

$$W_{ij} = U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$

where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and n is the size of the previous layer (Erhan et al. 2009). The biases are set to zero.

EXPERIMENTS

The mnist training data set is split into two: one set of 50000 examples is used for training and the set of other 10000 examples is used for the validation. The standard mnist test set is used for testing the performance of the neural network. The dataset is standardized to have a mean of zero and variance of one in all features.

Shallow Neural Network

We set up a shallow neural network as described in section 1 with 300 hidden units. The weights were initialized as described in section 3. The network was trained with mini-batch SGD back propagation. The network converges to zero error after 30 passes through the dataset and achieves an error of 3.6% on the test set. The Training and Validation errors over training epochs are show in figure 1. We also plot the distribution of weights and biases to monitor the change over training epochs in figure 2. We observe that the magnitude of the weights and biases mapping the hidden layer to output layer are larger than those mapping the inputs to hidden layer. Also, the gradients of the cost function calculated at each layer decrease in the direction from output to input layer. Therefore if you use a very small learning rate like 0.001, the weights and biases of the layer mapping inputs to the hidden layer change negligibly. Finally figure 4 shows the distribution of activations in hidden layer output. The mean activations in the hidden layer moves close toward the saturation value of 1.7159 as the training proceeds.

Random Features Network

We performed an experiment to determine the performance of neural network with random (linear and non linear) features. This is done by keeping the weights that map the input layer to the hidden layer (IHL) fixed, while changing only the weights that map the hidden layer to the output layer (HOL) using Back Propagation. The weights and biases in HOL are initialized as mentioned in section 3 while the weights and biases in IHL are sampled randomly from a uniform distribution $U[-a, a]$ where a is varied from 0.01 to 3.0. The hidden layer is set to have 300 hidden units. The network is trained using back propagation for 10 epochs with a fixed learning rate and batch size as mentioned in section 2. The error obtained on the test set is recorded in table 1. The number of epochs was fixed to 10 because the training error did not drop further after the first epoch. We also did one experiment by choosing the $a = 0.6$ such that the mean of all activations in the hidden layer lie between the linear region of the activation
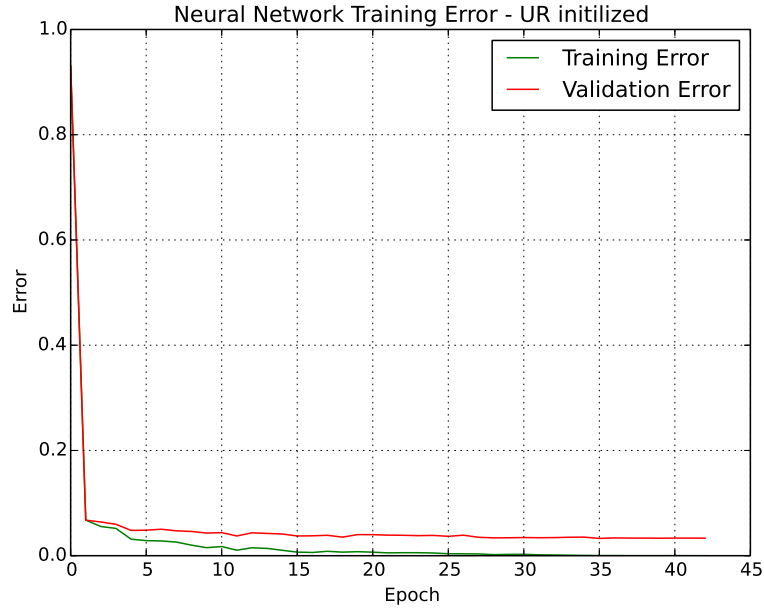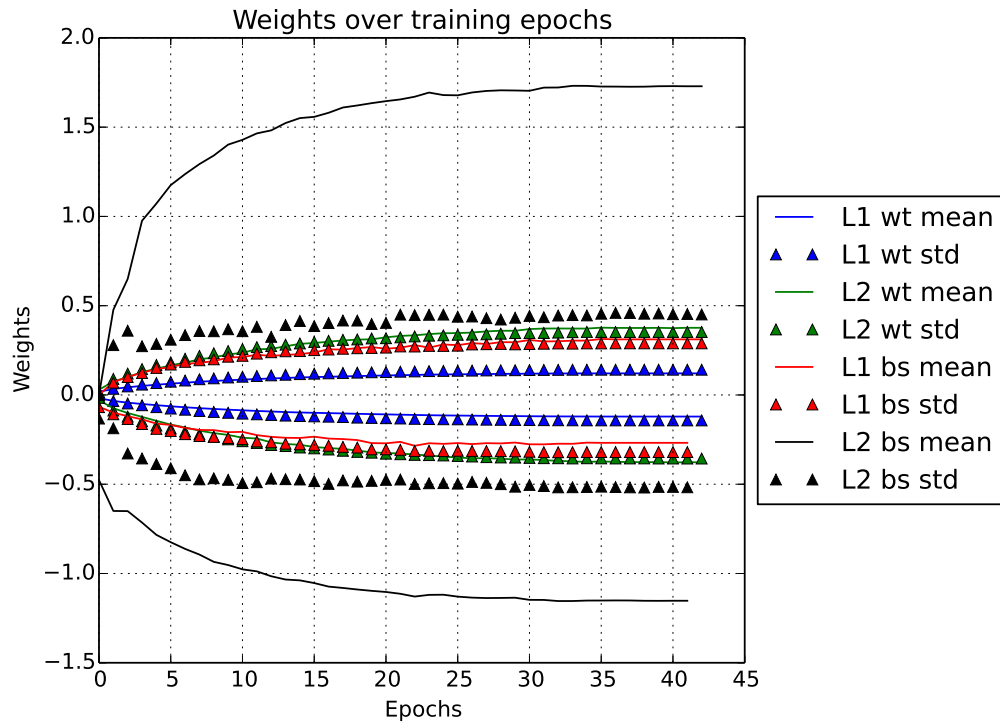
FIG. 1. Training and Validation Error

Neural Network Training Error - UR initilized

L1 maps inputs to hidden layer and L2 maps hidden to output layer.

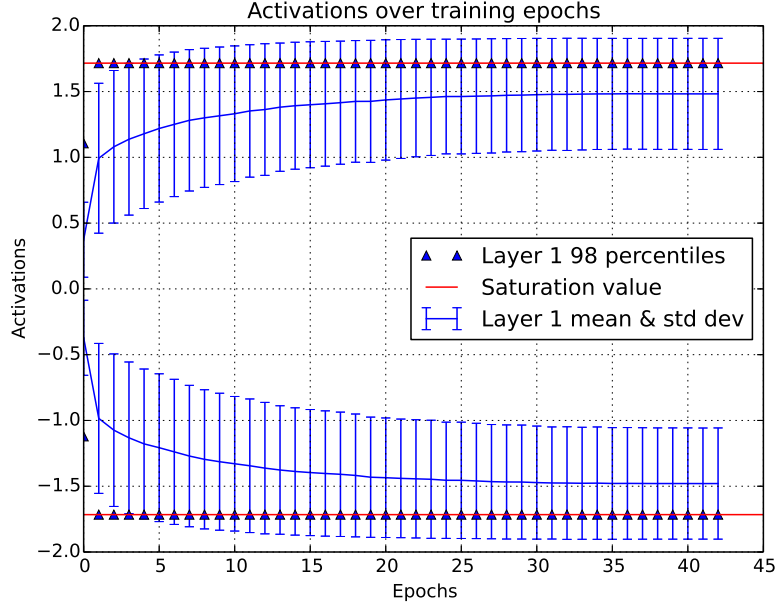## FIG. 2. Weight Distribution

Weights over training epochs

FIG. 3. Layer 1 Activations

function and its saturation value, see figure 5. We also increased the number of hidden units to 900. We observed that although the training error dropped below 10% slowly over 50 epochs, the validation error did not drop below 10%; see figure 4.

a	$TestError(\%)$
0.01	8.85
0.035	8.53
0.1	10.72
0.3	12.9
0.9	14.88
3.0	15.54

TABLE 1. Random Feature Network with 300 Hidden Units

FIG. 4. Training and Validation Error

LL Neural Network Training Error - UR initialized

FIG. 5. Layer 1 Activations

Activations over training epochs
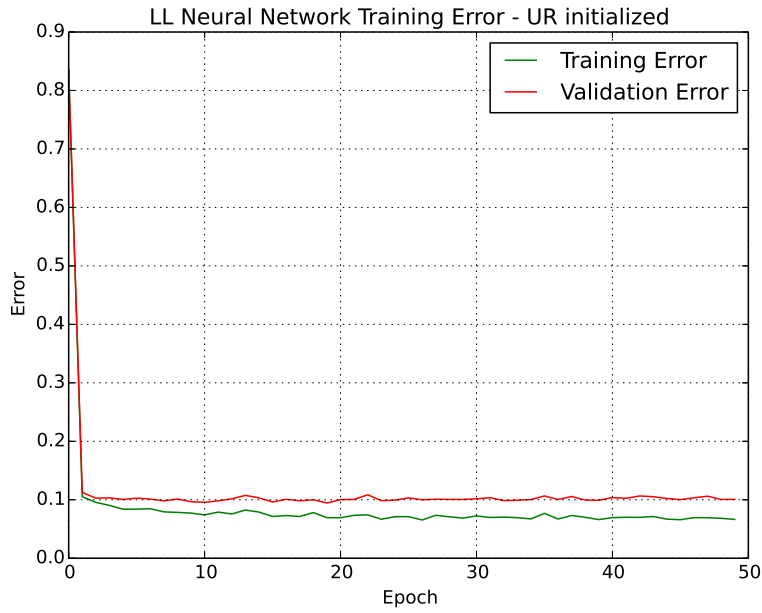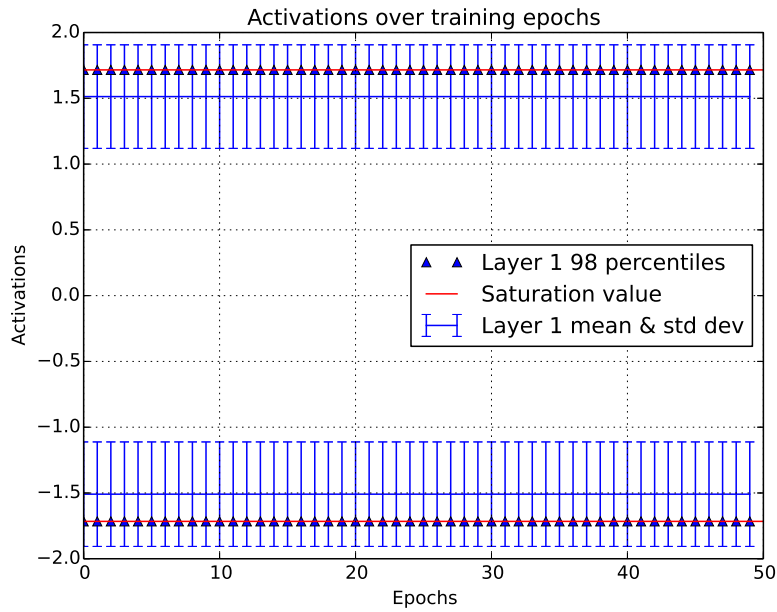
REFERENCES

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009).
   "The difficulty of training deep architectures and the effect of unsupervised

pre-training." *International Conference on Artificial Intelligence and Statistics*, 153–160.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). "Efficient backprop." *Neural networks: Tricks of the trade*, Springer, 9–48.

References