Maseeh College of Engineering
and Computer Science
PORTLAND STATE UNIVERSITY

Revised Backpropagation Algorithm of Artificial Neural Network

-- By Mark Musil, Jian Meng, Yuhao Zhou, Peilong Ning

Slideshow Structure

- Background
- Introduction and Problem Statement
- Related Works
- Algorithm 1
- Algorithm 2
- Results
- Conclusion

Background

Theory :

- Forward propagation
- Calculate Error
- Back propagation
- Weight Update

Method :

- Gradient Descent

Introduction and Problem Statement

- Modern neural networks can propagate error through successive layers, decreasing accuracy
- In addition, backpropagation via gradient descent is computationally expensive so we want to improve this as well
    - Faster Convergence (fewer epochs)
    - Fewer Operations

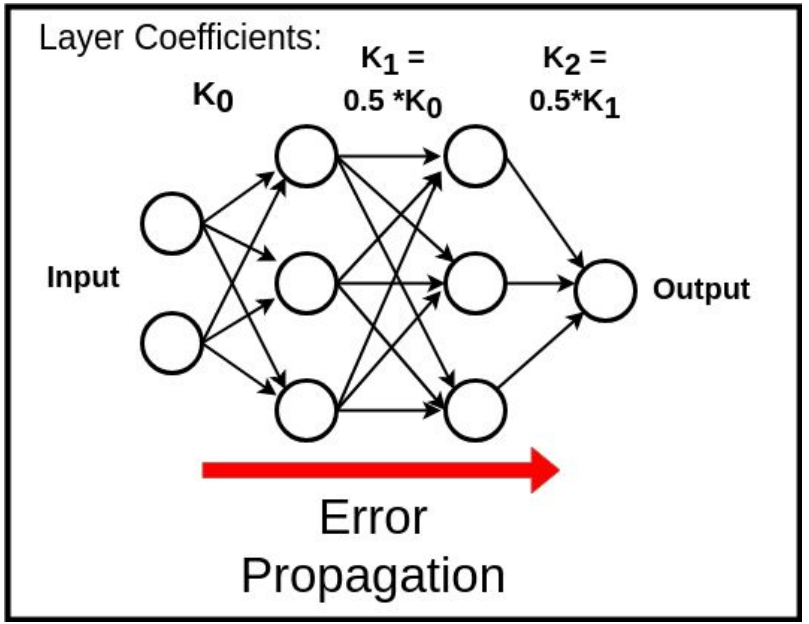
Related Works

- We began by considering the importance of the activation function as is addressed in [1], [2], [3]
- We also explored gradient-free methods such as the directed random search [4].
- RPROP Method[5] uses changing sign of $\delta C/\delta w_{jk}$ to avoid inefficient moves.

[1] Hahn-Ming Lee, Chih-Ming Chen, and Tzong-Ching Huang. Learning efficiency improvement of back-propagation algorithm by error saturation prevention method. Neurocomputing, 41(1-4):125–143, 2001.
[2] Youngjik Lee, Sang-Hoon Oh, and Myung Won Kim. An analysis of premature saturation in back propagation learning. Neural Networks, 6(5):719 – 728, 1993.
[3] X.G. Wang, Z. Tang, H. Tamura, M. Ishii, and W.D. Sun. An improved backpropagation algorithm to avoid the local minima problem. Neurocomputing, 56:455 – 460, 2004.
[4] U. Seiffert and B. Michaelis. Directed random search for multiple layer perceptron training. 2001.
[5] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rpop algorithm. 1993.

Algorithm 1 Error-by-layer method

- κ^l chosen for each layer as a hyperparameter
- $w^l_{jk} \mathrel{-}= \eta(a^{l-1}_k \delta^l_j)\kappa^l, \quad b^l_j \mathrel{-}= \eta\delta^l_j\kappa^l$
- This method works by training certain layers faster and preventing error propagation



Algorithm 2 - Error-by-neuron

- Applying SVD on sensitive matrix for each layer: $\delta = U\Sigma V^T$
- Pick the first ***k*** singular values and vectors to do reconstruction: $\delta_k = U_k \Sigma_k V_k^T$
- Then, use the new neuron-wise modified sensitive matrix to update gradient
- This method aims at reducing the generalization error and normalizing the direction of gradient in mini-batch.

Results

- The objectives of the experiments is to verify the feasibility and the efficiency of the proposed algorithms. In general, all the experiments are examining the algorithms in the following two perspectives:
  - The speed of convergence
  - The accuracy of two methods in both training and test set.

- To evaluate the difference between the revised methods and the traditional stochastic gradient descent.
  - ***IRIS dataset*** *- 150 samples*
  - ***MNIST dataset*** *- 60,000 samples*
  - ***Covertype dataset*** *- 581,012 samples*
  - ***Olivetti faces dataset*** *- 400 samples*

Results: Continue

Datasets	Training loss	Test loss
SGD - Iris	0.23	0.24
SGD - MNIST	0.22	0.25
Algorithm #1 - Iris	0.23	0.23
Algorithm #1 - MNIST	0.24	0.22
Algorithm #2 - Iris	0.24	0.26
Algorithm #2 - MNIST	0.13	0.13

Results: iris dataset - large size structure

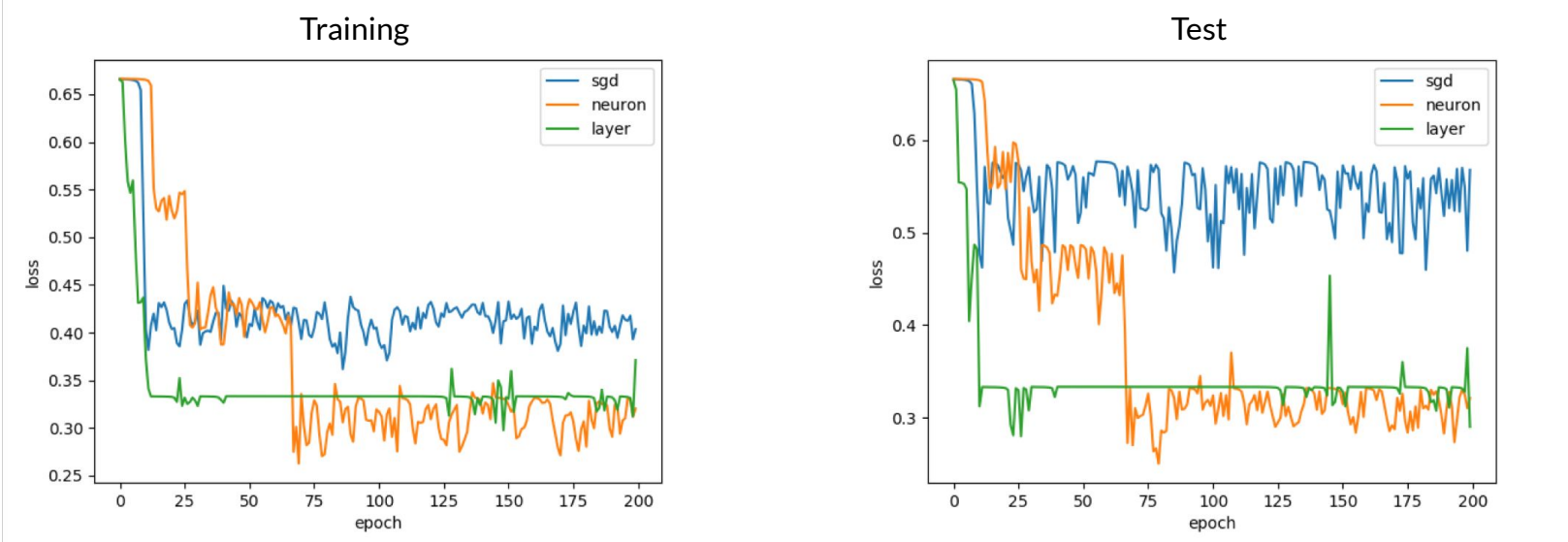


Figure 1: Test on IRIS dataset. Hidden Dims: [60 30 15]; Learning Rate 0.05

Results: MNIST dataset - large size structure

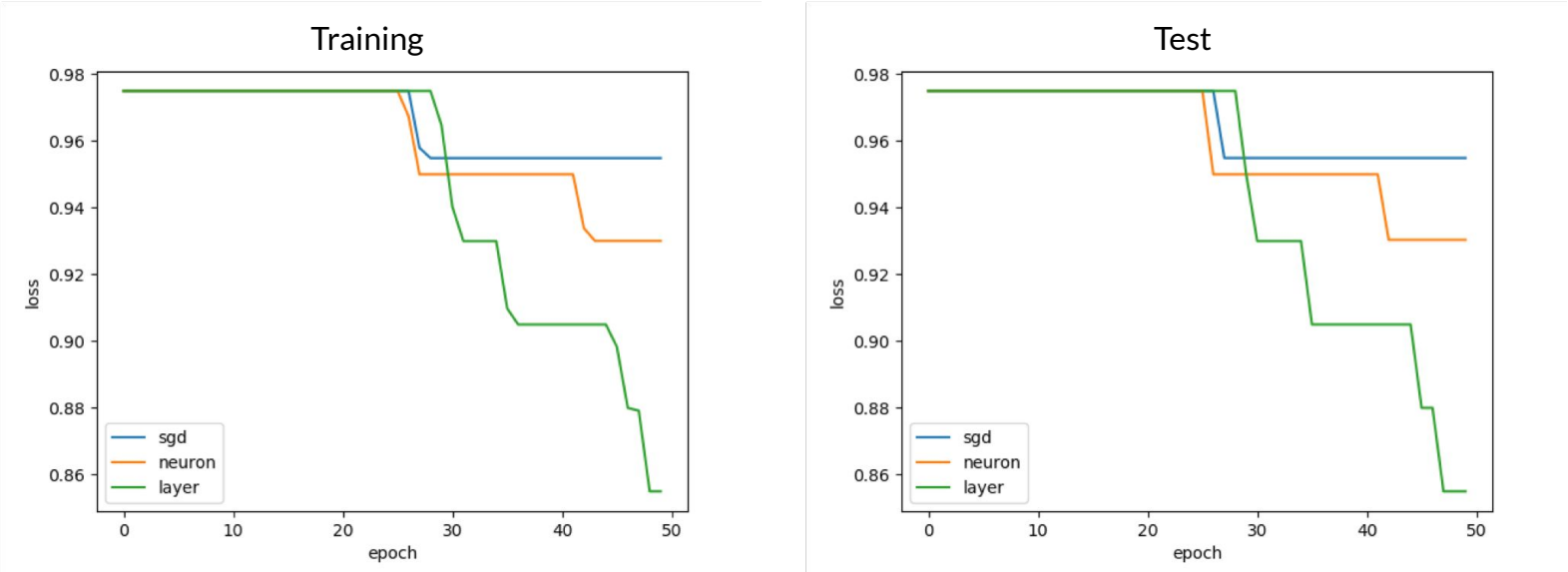


Figure 4: Test on MNIST dataset. Hidden Dims: [200 80 40]; Learning Rate 0.1

Conclusion

- The proposed methods are competitive with stochastic gradient descent
- In order to determine their usefulness, more testing should be done
  - Want to make sure networks didn't "memorize" the data
- We suggest testing with highly constrained networks where number of trainable parameters ~= number of training samples