Neural-Network-on-MNIST

Figure 1 shows the plot of train and test set errors using batch learning with single layer neural network ($\eta = 10^{-5}$).

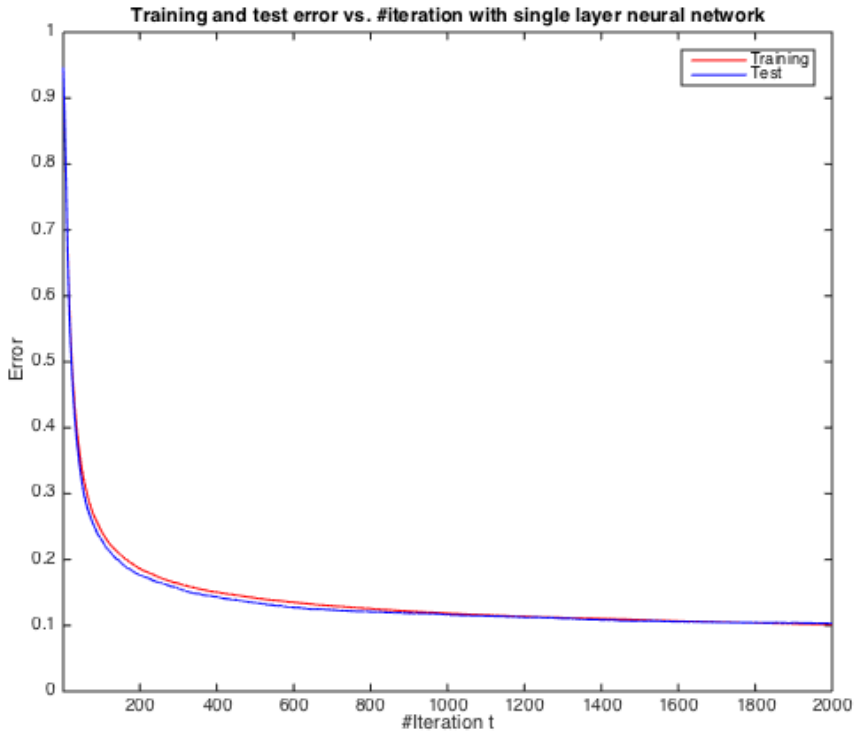

Fig. 1: Training and test errors vs. #iteration (single layer / batch)

The final training and test errors are shown below.

	Final training error	Final test error
Single layer / Batch	0.1016	0.1033

Figure 2 shows the plot of train and test set errors using batch learning with two layer neural network for the number of hidden neurons $H \in \{10, 20, 50\}$ ($\eta = 10^{-5}$).

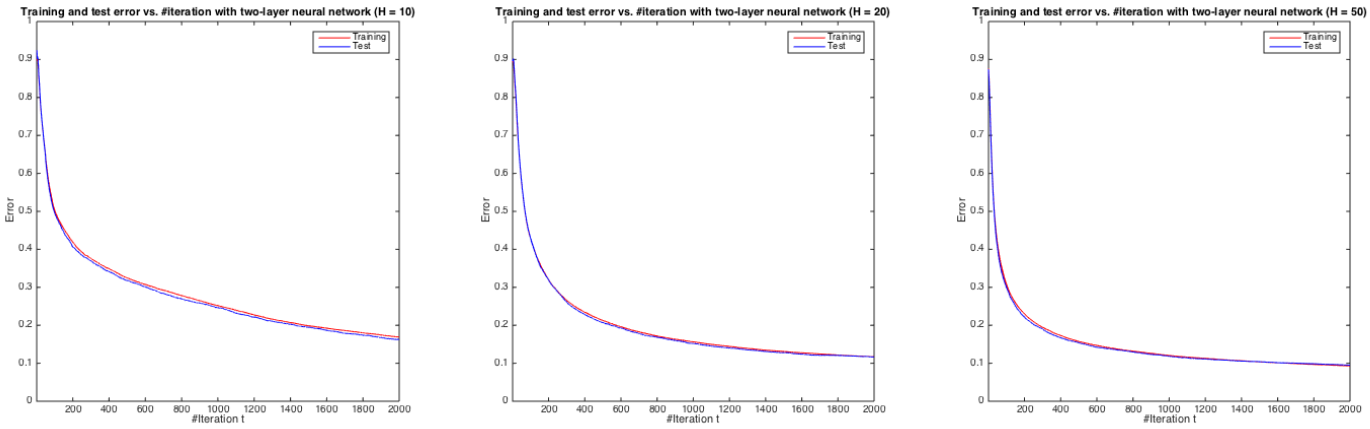

Fig. 2: Training and test errors vs. #iteration (two layer / batch)

The final training and test errors are shown below.

	Final training error	Final test error
H = 10	0.1689	0.1627
H = 20	0.1171	0.1165
H = 50	0.0930	0.0951

The results suggest that with the increasing number of hidden neurons, the training / test errors converge faster and the network achieves better performance.

Figure 3 shows the plot of train and test set errors using batch learning with two layer neural network for the number of hidden neurons $H \in \{10, 20, 50\}$, sigmoid / ReLU non-linearities, and weight decay ($\lambda = 0.001$, $\eta = 2 \times 10^{-6}$).

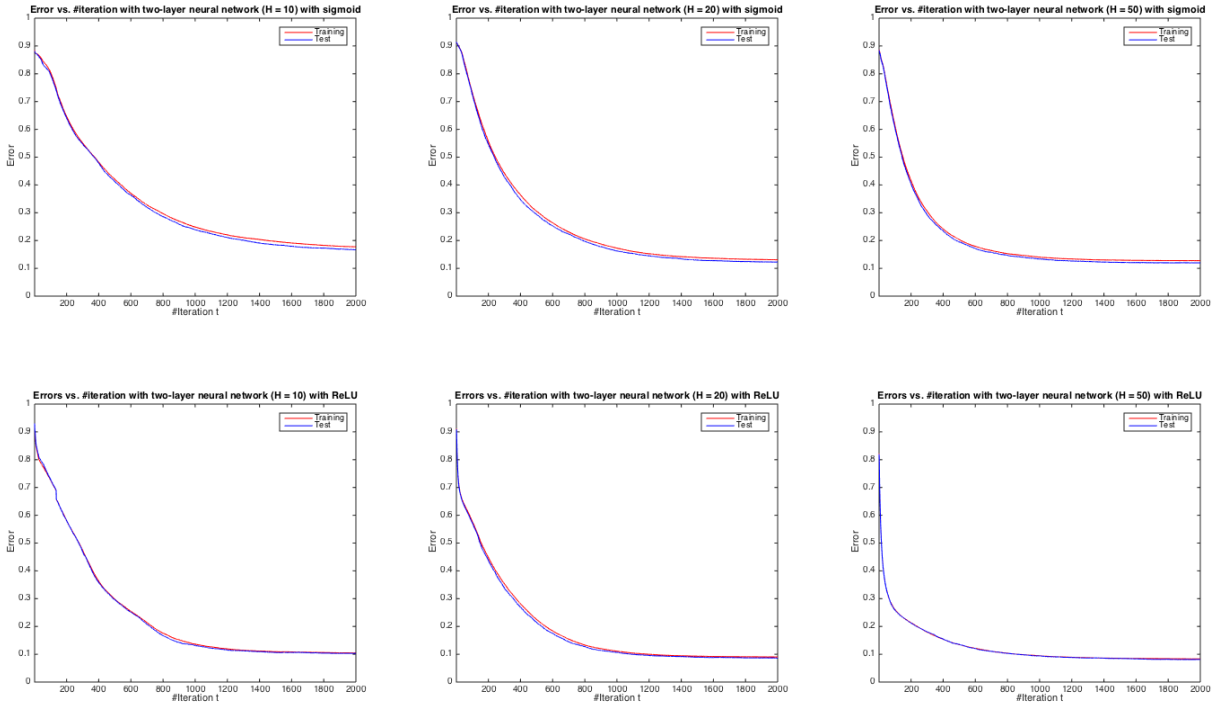

Fig. 3: Training and test errors vs. #iteration (two layer / batch / regularization-1)

The final training and test errors are shown below.

	Final training error (sigmoid)	Final training error (ReLU)	Final test error (sigmoid)	Final test error (ReLU)
H = 10	0.1773	0.1047	0.1667	0.1032
H = 20	0.1309	0.0904	0.1227	0.0862
H = 50	0.1276	0.0832	0.1194	0.0813

The results suggest that ReLU non-linearity achieves better performance and faster network convergence than sigmoid in batch learning case.

Figure 4 shows the plot of train and test set errors using batch learning with two layer neural network for the number of hidden neurons H $\in$ {10, 20, 50}, sigmoid / ReLU non-linearities, and weight decay ($\lambda = 0.0001$, $\eta = 2 \times 10^{-6}$).

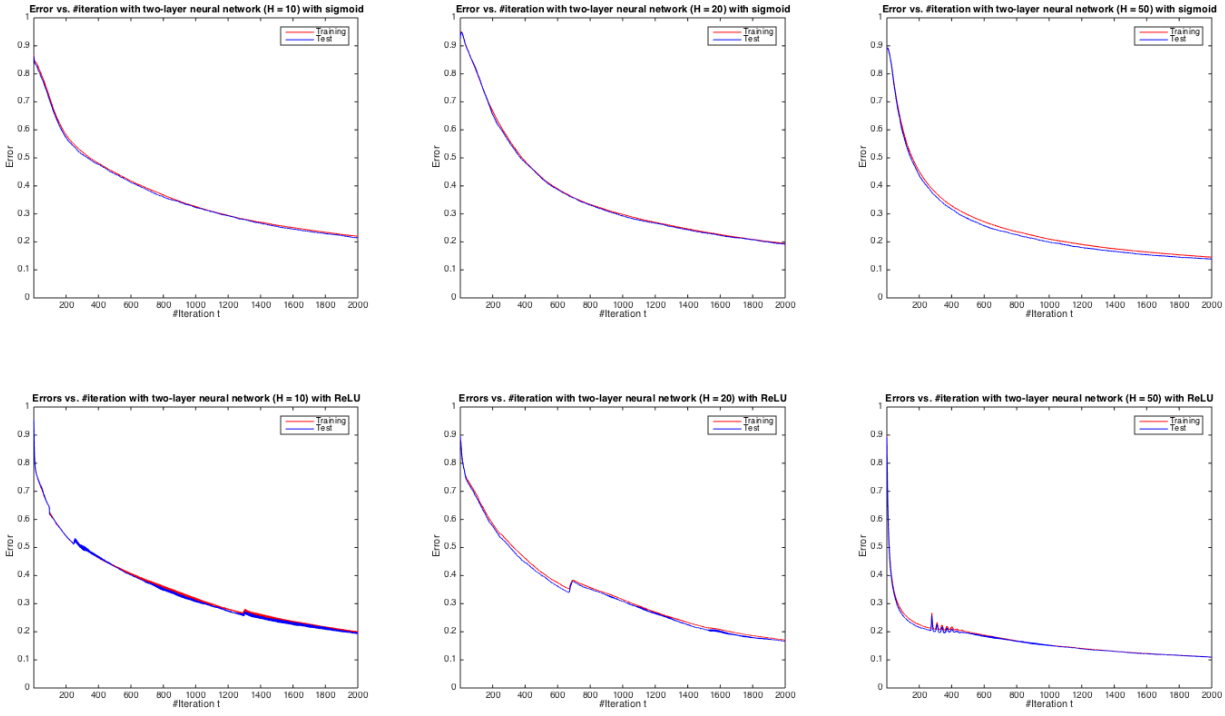

Fig. 4: Training and test errors vs. #iteration (two layer / batch / regularization-2)

The final training and test errors are shown below.

	Final training error (sigmoid)	Final training error (ReLU)	Final test error (sigmoid)	Final test error (ReLU)
H = 10	0.2200	0.1972	0.2140	0.1923
H = 20	0.1940	0.1705	0.1922	0.1666
H = 50	0.1456	0.1098	0.1387	0.1096

The results suggest that when using smaller λ, the large weights did not get enough penalization for ReLU as we see in the above plots (bump up). The final errors are also higher than that using larger λ.

Figure 5 shows the plot of train and test set errors using stochastic gradient descent learning with single layer neural network ($\eta = 10^{-5}$).

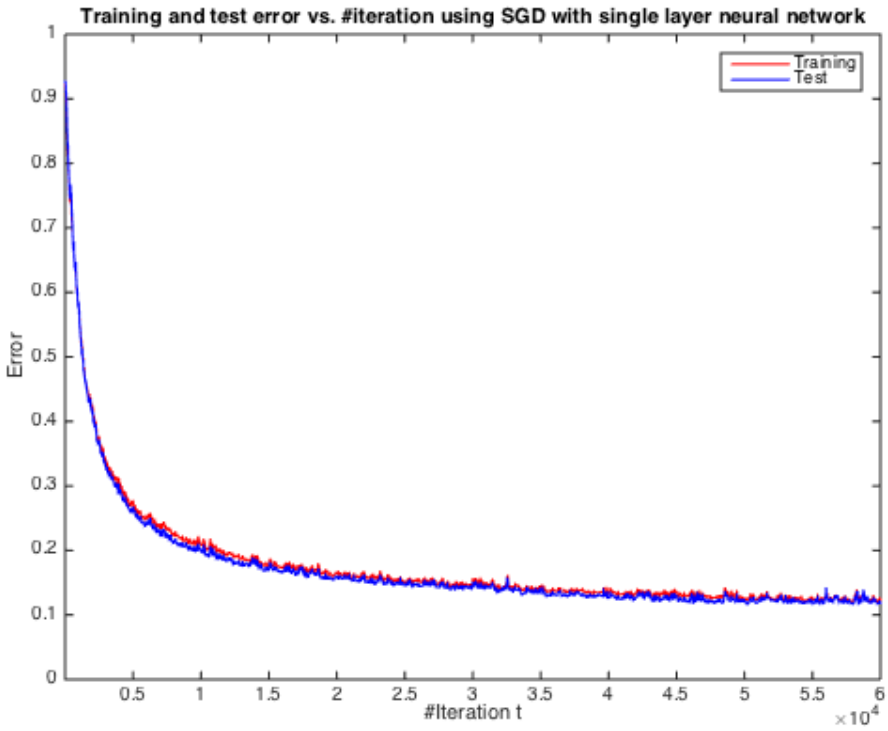

Fig. 5: Training and test errors vs. #iteration (single layer / SGD)

The final training and test errors are shown below.

	Final training error	Final test error
Single layer / SGD	0.1214	0.1174

Stochastic gradient descent takes much more iterations to converge while running much faster than batch gradient descent, because every step it only learn a single sample. The error curves are attenuated wiggly for the same reason.

Figure 6 shows the plot of train and test set errors using batch learning with two layer neural network for the number of hidden neurons $H \in \{10, 20, 50\}$, sigmoid ($\eta = 10^{-2}$) / ReLU non-linearities ($\eta = 2 \times 10^{-3}$).

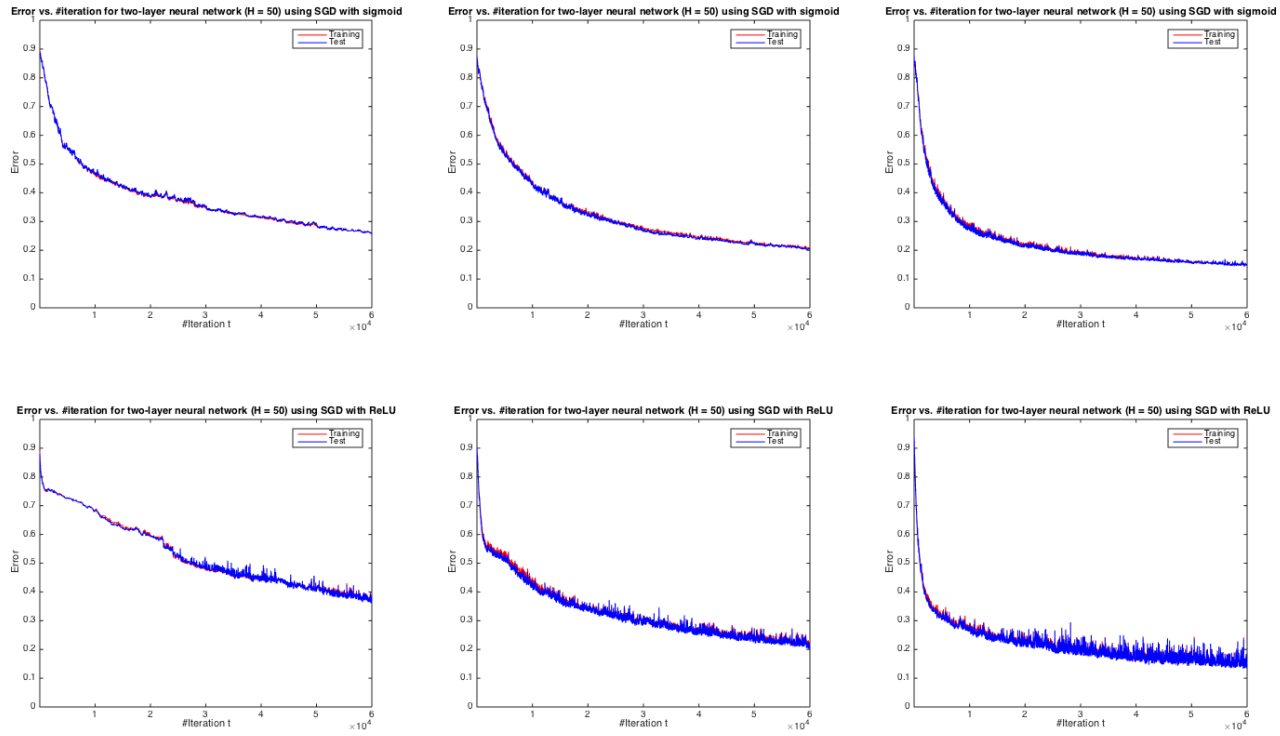

Fig. 6: Training and test errors vs. #iteration (two layer / SGD)

The final training and test errors are shown below.

	Final training error (sigmoid)	Final training error (ReLU)	Final test error (sigmoid)	Final test error (ReLU)
H = 10	0.2680	0.5827	0.2547	0.5727
H = 20	0.1727	0.3080	0.1704	0.3088
H = 50	0.1469	0.1608	0.1465	0.1569

From the plots above, stochastic gradient descent works better on sigmoid non-linearity than ReLU, probably because we are using shallow networks. ReLU is known to be popular in deep neural networks.