CS189 — Homework 6

Alvin Wong, Wai Meng Lei, Chun Yin Yau

April 21, 2014

1. Single-Layer Neural Network

Derivation of gradient updates:

Let $y_k = \sigma(s_k)$, where $s_k = \sum_j W_{jk}x_j + b_k$.

1) Using the mean squared error: $J = \frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2$:

$$\begin{aligned}
\frac{dJ}{dW_{jk}} &= \frac{d}{dW_{jk}}(\frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2 \\
&= (y_k - t_k)\frac{d}{dW_{jk}}\sigma(s_k) \\
&= (y_k - t_k)\sigma(s_k)(1-\sigma(s_k))\frac{d}{dW_jk}(W_{jk}x_j + b_k) \\
&= (y_k - t_k)\sigma(s_k)(1-\sigma(s_k))x_j
\end{aligned}$$

$$\begin{aligned}
\frac{dJ}{db_j} &= \frac{d}{db_j}(\frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2 \\
&= (y_k - t_k)\frac{d}{db_j}\sigma(s_k) \\
&= (y_k - t_k)\sigma(s_k)(1-\sigma(s_k))\frac{d}{db_j}(W_{jk}x_j + b_k) \\
&= (y_k - t_k)\sigma(s_k)(1-\sigma(s_k))
\end{aligned}$$

In terms of matrices and vectors: Let $\vec{y} = \sigma(\vec{s})$, and $\vec{s} = W\vec{x} + \vec{b}$:

$$\boxed{\frac{dJ}{dW} = diag(\vec{y})(1 - diag(\vec{y}))[\vec{y} - \vec{t}]\vec{x}^T}$$

$$\boxed{\frac{dJ}{d\vec{b}} = diag(\vec{y})(1 - diag(\vec{y}))[\vec{y} - \vec{t}]}$$

2) Using the cross-entropy error: $J = -\sum_{k=1}^{n_{out}}[t_k \ln y_k + (1-t_k)\ln(1-y_k)]$:

The math works out the same, it's just the initial $y_k - t_k$ term becomes the derivative of the cross-entropy error term above:

$$\begin{aligned}
\frac{dJ}{dW_{jk}} &= [-\frac{t_k}{y_k} + \frac{1-t_k}{1-y_k}]\sigma(s_k)(1-\sigma(s_k))x_j \\
\frac{dJ}{db_j} &= [-\frac{t_k}{y_k} + \frac{1-t_k}{1-y_k}]\sigma(s_k)(1-\sigma(s_k))
\end{aligned}$$

In terms of matrices and vectors:

$$\boxed{\frac{dJ}{dW} = diag(\vec{y})(1 - diag(\vec{y}))[-\frac{\vec{t}}{\vec{y}} + \frac{1-\vec{t}}{1-\vec{y}}]\vec{x}^T}$$

$$\boxed{\frac{dJ}{d\vec{b}} = diag(\vec{y})(1 - diag(\vec{y}))[-\frac{\vec{t}}{\vec{y}} + \frac{1-\vec{t}}{1-\vec{y}}]}$$

Figure 1: Plot of the training loss (blue) and test set accuracy (red), 500 epochs, MSE error
The test accuracy converged to 92.25%. The total training loss converrged to about 4370. This took about a little over an hour to run.

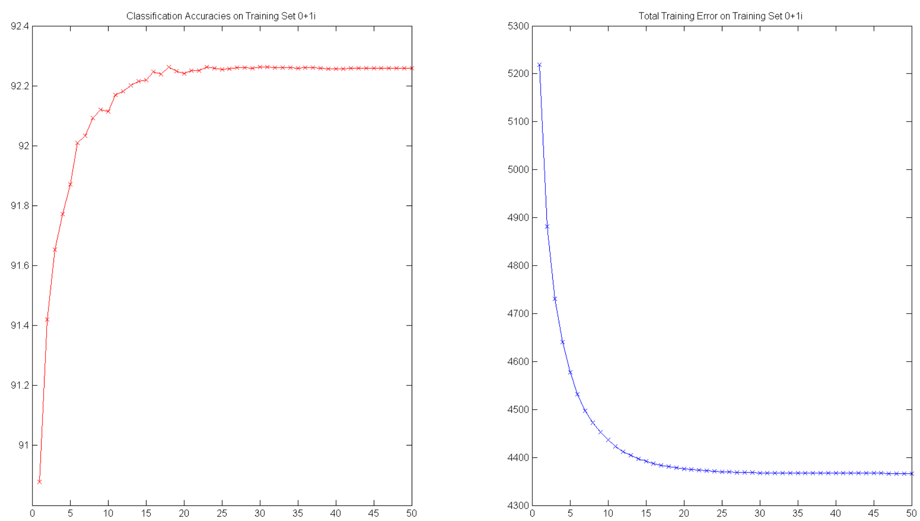
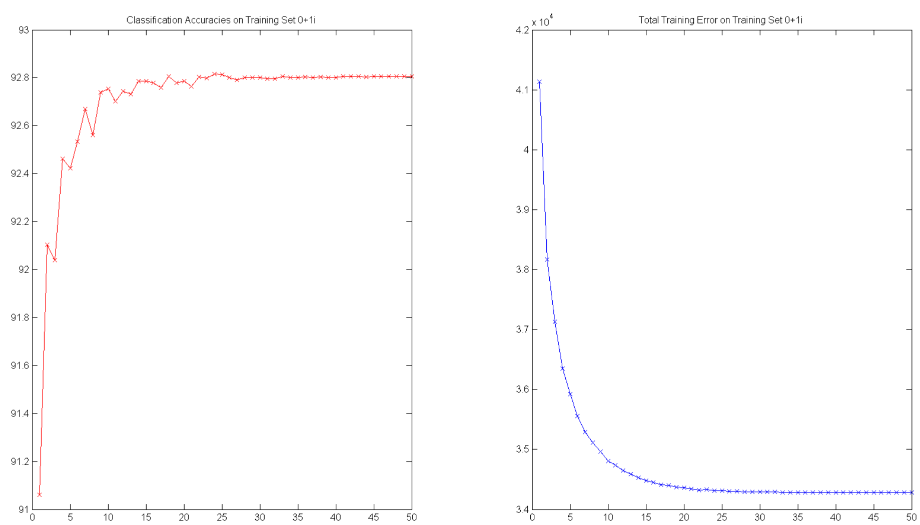


Figure 2: Plot of the training loss (blue) and test set accuracy (red), 500 epochs, CE error
The test accuracy converged to 92.8%. The total training loss converged to about 34300. This took about a little over an hour to run.



Note: don't mind the titles – they are wrong. Left graph is on test accuracy. Right graph is on training set loss.

2. Multi-Layer Hidden Neural Network

Derivation of gradient updates:

We'll be making use of the previous question's derivations to help us here:

1) Using the mean squared error: $J = \frac{1}{2}\sum_{k=1}^{n_{out}}(t_k - y_k)^2$:

Note: might have flipped the dimensionalities of the matrices, but nevertheless it's just a transpose difference.

Let:
$\vec{s_1} = W_1\vec{x} + \vec{b_1}$
$\vec{s_2} = W_2 tanh(\vec{s_1}) + \vec{b_2}$
$\vec{s_{out}} = W_{out} tanh(\vec{s_2}) + \vec{b_{out}}$
$\vec{y} = \sigma(\vec{s_{out}})$

Then the backprop algorithm works as follows:
$\vec{\delta_{out}} = diag(\vec{y})diag(1-\vec{y})[\vec{y} - \vec{t}]$
$W_{grad,out} \leftarrow W_{grad,out} + \vec{\delta_{out}} tanh(\vec{s_2})^T$
$b_{grad,out} \leftarrow b_{grad,out} + \vec{\delta_{out}}$
$\vec{\delta_2} = (1 - tanh(\vec{s_2})^2). * W_{out}^T \vec{\delta_{out}}$
$W_{grad,2} \leftarrow W_{grad,2} + \vec{\delta_2} tanh(\vec{s_1})^T$
$b_{grad,2} \leftarrow b_{grad,2} + \vec{\delta_2}$
$\vec{\delta_1} = (1 - tanh(\vec{s_1})^2). * W_2^T \vec{\delta_2}$
$W_{grad,1} \leftarrow W_{grad,1} + \vec{\delta_1} tanh(\vec{x})^T$
$b_{grad,1} \leftarrow b_{grad,1} + \vec{\delta_1}$

We'll then apply SGD with a time varying α rate to update the parameters by the gradients.

2) Using the cross-entropy error: $J = -\sum_{k=1}^{n_{out}}[t_k \ln y_k + (1-t_k)\ln(1-y_k)]$:

The math works out the same, it's just the initial $\vec{y} - \vec{t}$ term becomes the derivative of the cross-entropy error term above:

Then the backprop algorithm works as follows:
$\vec{\delta_{out}} = diag(\vec{y})diag(1-\vec{y})[-\frac{\vec{t}}{\vec{y}} + \frac{1-\vec{t}}{1-\vec{y}}]$
$W_{grad,out} \leftarrow W_{grad,out} + \vec{\delta_{out}} tanh(\vec{s_2})^T$
$b_{grad,out} \leftarrow b_{grad,out} + \vec{\delta_{out}}$
$\vec{\delta_2} = (1 - tanh(\vec{s_2})^2). * W_{out}^T \vec{\delta_{out}}$
$W_{grad,2} \leftarrow W_{grad,2} + \vec{\delta_2} tanh(\vec{s_1})^T$
$b_{grad,2} \leftarrow b_{grad,2} + \vec{\delta_2}$
$\vec{\delta_1} = (1 - tanh(\vec{s_1})^2). * W_2^T \vec{\delta_2}$
$W_{grad,1} \leftarrow W_{grad,1} + \vec{\delta_1} tanh(\vec{x})^T$
$b_{grad,1} \leftarrow b_{grad,1} + \vec{\delta_1}$

Figure 1: Plot of the training loss (blue) and test set accuracy (red), 150 epochs, MSE error

The test accuracy converged to 92.25%. The total training loss converrged to about 4370. This took about a little over an hour to run.

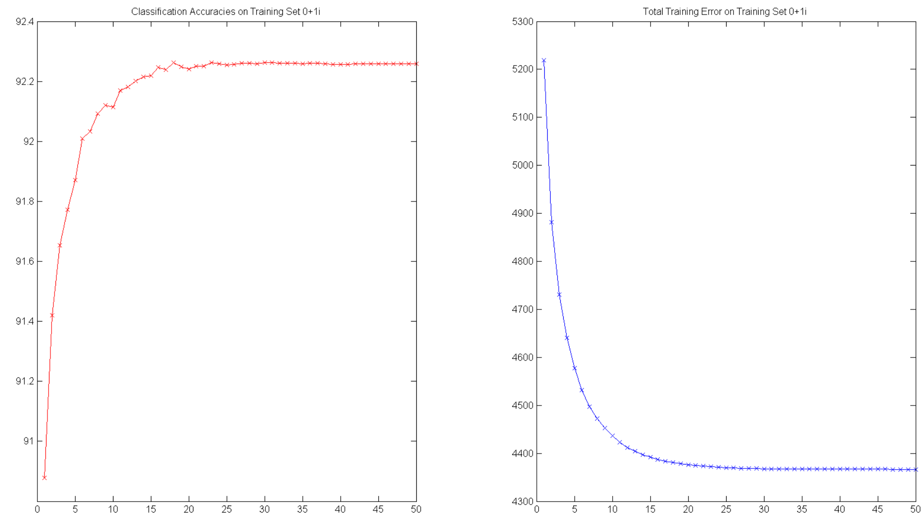
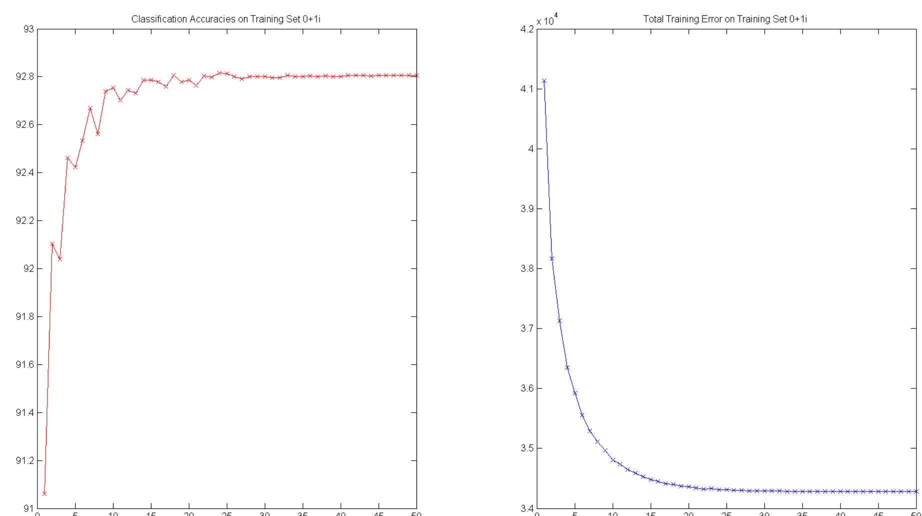


Figure 2: Plot of the training loss (blue) and test set accuracy (red), 150 epochs, CE error

The test accuracy converged to 92.8%. The total training loss converged to about 34300. This took about a little over an hour to run.



Note: don't mind the titles – they are wrong. Left graph is on test accuracy. Right graph is on training set loss.