Deep Learning: Homework 4

Qi Luo
A02274095

3/6/2019

Problem 1

1. For the second expression,
   when $y = 1$, $-[a \ln y + (1-a)\ln(1-y)] = -(1-a)\ln(0)$
   when $y = 0$, $-[a \ln y + (1-a)\ln(1-y)] = -a\ln(0)$

   For the first expression,
   when $y = 1$, $-[y \ln a + (1-y)\ln(1-a)] = -\ln(a)$
   when $y = 0$, $-[y \ln a + (1-y)\ln(1-a)] = -\ln(1-a)$

   Since $\ln(0)$ is not well define, we can not get the exact value when $y = 0$ and $y = 1$ for the second expression. However, this could be a problem since $y = 0$ and $y = 1$ could be happen. For the first expression, this will not happen since we use sigmoid activation function the vaule of a should be between $[0, 1]$. All in all, I do not think expression two is afflict to expression one, since it should be fine if output is a certain value, but what we want is a certain network.

2. Since $C(a) = -[y \ln a + (1-y)\ln(1-a)]$, then $\frac{\partial C}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
   Set $\frac{\partial C}{\partial a} = 0$, $-\frac{y}{a} + \frac{1-y}{1-a} = 0$, then we can get $y = a$.
   $\frac{\partial^2 C}{\partial^2 a} = \frac{y}{a^2} + \frac{1-y}{(1-a)^2}$ Since $y \in (0,1)$ and $a \in (0,1)$, $\forall a \in (0,1)$, $\frac{\partial^2 C}{\partial^2 a} \geq 0$.
   Therefore, when $y = a$ is a minimized point for this function.

3. For this part, I used python to help me do the math part. All code in $prob1_3.py$.
   The derivatives of the cost with respect to the weights:

$$w1 = \begin{bmatrix} 0.00226835 & 0.00257727 \\ 0.0045367 & 0.00515454 \end{bmatrix}$$

$$w2 = \begin{bmatrix} 0.43982965 & -0.12878203 \\ 0.44250923 & -0.12956661 \end{bmatrix}$$

   Since the derivatives of the cost with respect to the biases is equal to δ for each layer: $\delta^2 = [0.74136507, -0.21707153]$ $\delta^1 = [0.04536701, 0.05154539]$

Problem 2

1. Assume the modified neuron is the i^{th} neuron in the l^{th} layer. For feed-forward step, everything should be the same value except for $a_i^l = f(Z_i^l)$.
   Case 1: When the modified neuron is in the output layer, then $\delta_i^L = \frac{\partial C}{\partial a_i^L} f'(Z_i^L)$ for the changed neuron. For others, $\delta_i^L = \frac{\partial C}{\partial a_i^L}\sigma'(Z_i^L)$. Also for other steps will not be changed.
   Case 2: When the modified neuron is not in the output layer, step 3 and 5 will not be changed. But changes happen on step 4, for the modified neuron $\delta_i^l = \sum_k w_{ki}^{l+1}\delta_k^{l+1} f'(z_j^l)$. For others, $\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l)$.

2. $\frac{\partial C}{\partial b_j^L} = \frac{\partial C}{\partial z_j^L}\frac{\partial z_j^L}{\partial b_j^L} = -\sum_k y_k \frac{1}{a_k^L}\frac{\partial a_k^L}{z_j^L} = -y_j \frac{1}{a_j^L}\frac{\partial a_j^L}{\partial z_j^L} - \sum_{k\neq j} y_k \frac{1}{a_k^L}\frac{a_k^L}{z_j^L} =$
   $-y_j \frac{1}{a_j^L}\cdot a_j^L(1-a_j^L) - \sum_{k\neq j} y_k \frac{1}{a_k^L}\cdot(-a_j^L a_k^L) = -y_j + y_j a_j^L + \sum_{k\neq j} y_k a_j^L$
   $= a_j^L - y_j$

3. Since $a_j^L = \frac{e^{cz_j^L}}{\sum_k e^{cz_k^L}}$ with $c > 0$.
   If $c \to +\infty$, $\lim_{c\to+\infty} \frac{e^{cz_j^L}}{\sum_k e^{cz_k^L}} = \frac{1}{\sum_k e^{c(z_k^L - z_j^L)}} = 1$.
   If $j \neq k$, $\lim_{c\to\infty} e^{c(z_k^L - z_j^L)} = 0$.
   If $j = k$, $z_j^L = z_k^L$, then $\lim_{c\to\infty} e^{c(z_k^L - z_j^L)} = 1$.
   Since $\sum a_j^L = \frac{\sum_j e^{cz_j^L}}{\sum_k e^{cz_k^L}} = 1$, if there are n maximums, and we assumed a_j^L is maximum, then we have each $a_j^L = \frac{1}{n}$, and others are zeros.
   Therefore, we called it softmax is becouse of the exponential function which allow the function slowly and gently close to the edge.

4. From problem 2.2, we can get $\delta_j^L = \frac{\partial C}{\partial z_j^L} = a_j^L - y_i$. Since, $z^L = \sum_k w^L a_k^{L-1} + b^L$.
   Therefore, $\frac{\partial C}{\partial w_{jk}^L} = \frac{\partial C}{\partial z_j^L}\cdot\frac{\partial z_j^L}{\partial w_{jk}^L} = a_k^{(L-1)}(a_j^L - y_i)$.

Problem 3

1

(a) The dropout rate is 0.1. Relu for activation function in each layer. Cross-entropy for cost function. Randomly initialize weights. When the accuracy is keep bounding for more then 20 epoches then stop.[early stop]

(b) Learning rate is 0.0005, regularization parameter is 0.00000045, mini-batch size is 32, and the number of nodes in each of the hidden layers is 100 and 80.
   First, my default setting is Learning rate is 0.001, regularization parameter is 0, mini-batch size is 32, and the number of nodes in each of the

hidden layers is 100 and 30, which means I tried to no add L2 regularization. For this setup, i can get about 95% accuracy. I modified all the paramers before I add regularication parameter into my network. So when I add L2 regularization with parameter 0.9, which decrease my validation accuracy to 10.17%. Then I decrece the parameter by one decimal each time, then I can get about 96.22% accuracy with 0.000009. Since it is very close to 97%, I decide to decrease half of the parameter instead of decrease one decimal number. For all of thses parameters, I only changed one parameter value for each run, in order to see what is the difference and how does this parameter affect the accuracy.

(c) My final validation error is 98.03%.

(d) My final test error is 98.03%.

2 For the beginning of training, I would say the the initialize with the weights obtained using L2 regularization is absolutly better than initialize randomly. But for the final result they are both very similar. For randomly initialize, I can get around 97.5% accuracy. For L2, I can get about 98.3% accuracy. So based on my result, I would say initialize with the weights obtained using L2 regularization worked the best. (I used the same regularization parameter as I used on L2 instead of the learning rate, since if I used the same value for the learning rate for regularization parameter, the accuracy is really low.) Since I compare the L1 and L2 with the same regularization parameter, I can get the highest accuracy on L1 is 98.10% which is only 0.7% higher than L2. Based on my result, L1 worked better than L2. But basicaly I think they both worked well.

Problem 3

1. Yes.