Deep Learning: Homework 5

Qi Luo
A02274095

3/27/2019

Problem 1

1. According to cost function, $C = C_0 + \frac{\lambda}{2n}\sum_w w^2$. For λ, we want to minimize C by changing λ. Therefore, λ will decrease until it reach to 0. However, we do not want the learning rate down to 0. For η, we can tell by the cost function that the cost function does not depend on η at all. Therefore, that is an obstacles to using standard gradient desent to determine η.

2. (a) According to function $w \to (1 - \frac{\eta\lambda}{n})w - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial w}$, weights will change on every epoch since $(1 - \frac{\eta\lambda}{n})$ on the weights. At the first epoch, weights were all randomly chose. Therefore, the weights at the first epoch of training will depends on the second term of this equation, which is the weight decay part.

   (b) For each epoch, $(1 - \frac{\eta\lambda}{n})$ will be multipled $\frac{n}{m}$ times. Then after each epoch, the weight will decay by a factor $(1 - \frac{\eta\lambda}{n})^{\frac{n}{m}}$. Therefore, when $n \to \infty$, $(1 - \frac{\eta\lambda}{n})^{\frac{n}{m}} = [(1 - \frac{\eta\lambda}{n})^{-\frac{n}{\eta\lambda}}]^{-\frac{\eta\lambda}{m}} = exp(-\frac{\eta\lambda}{m})$ since $\eta\lambda \ll n$, $(1 - \frac{\eta\lambda}{n})^{-\frac{n}{\eta\lambda}} \approx e$.

   (c) When λ is not too large, according to $C = C_0 + \frac{\lambda}{2n}\sum_w w^2$ we can ignore C_0, then $C = \frac{\lambda}{2n}\sum_w w^2$. So, $w = \sqrt{\frac{2nC}{\lambda}}\frac{1}{\sqrt{n}}$. Therefore, the weight decay will tail off when the weights are down to a size around $\frac{1}{\sqrt{n}}$ when λ is not too large.

Problem 2

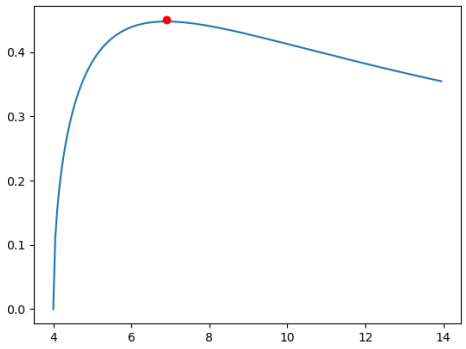
1. Usually, the activation function whose derivative is close to 1, then the learning rate for each layer should be close despite the proeduct of many

terms. But it will not solve the unstable gradient problem. Therefore, use a different activation function, one whose derivative is be much larger will probably help this problem.

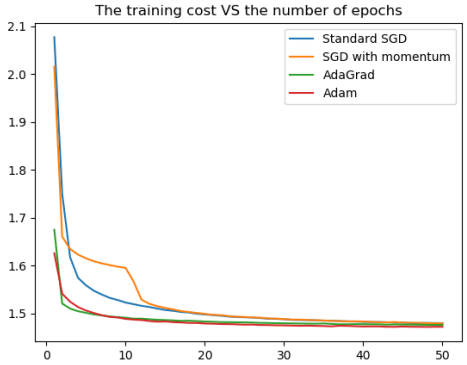
2. (a) $|w\sigma'(wa+b)| \leq |w||\sigma'(wa+b)| \leq |w| * \frac{1}{4}$
(according to above question, $|\sigma'(z)| < \frac{1}{4}$)
Since $|w\sigma'(wa+b)| \geq 1$, $1 \leq |w| * \frac{1}{4}$, therefore, this can only ever occur if $|w| \geq 4$.

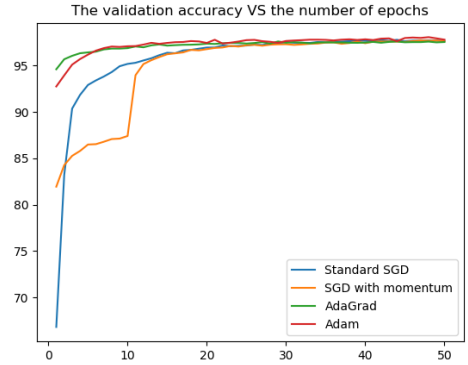
   (b) $|w\sigma'(wa+b)| \geq 1 \Leftrightarrow \sigma'(wa) \geq \frac{1}{|w|} \Leftrightarrow \frac{e^{-wa}}{(1+e^{-wa})^2} \geq \frac{1}{|w|}$. Let $x = e^{-wa}$, then we can get $(1+x)^2 - |w|x = x^2 + (2-|w|)x + 1 \leq 0$. In order to slove this equation, $\Delta = (2-|w|)^2 - 4 = |w|(|w|-4)$. Since $|w| > 4$, $\Delta > 0$ which means this equation has two different roots, x_1 and x_2. Then, $x_1 \leq e^{-wa} \leq x_2$. When $w > 0$, $-\frac{1}{w}lnx_2 \leq a \leq -\frac{1}{w}lnx_1$. So, the width of interval is $\frac{1}{|w|}ln\frac{x_2}{x_1}$. When $w < 0$, $-\frac{1}{w}lnx_1 \leq a \leq -\frac{1}{w}lnx_2$. So, the width of interval is $\frac{1}{|w|}ln\frac{x_2}{x_1}$. Since the equation for x_1 and x_2 is quadratic equation, $x_1x_2 = 1$ and $x_2 = \frac{|w|(1+\sqrt{1-\frac{4}{|w|}})}{2} - 1$. Then the interval no greaterin width than $\frac{2}{|w|}ln(\frac{|w|(1+\sqrt{1-\frac{4}{|w|}})}{2} - 1)$

   (c)



3. If $\mu > 1$, we may increase momentum even if $\bigtriangledown C$ has always been 0. And maybe, when we get velocity as a larger number, we will need to times by a number which is greater than 1 for every step, which is a bad idea than it may be hard to control and overshoot the minimum. If $\mu < 0$, the velocity will be between positive and negative value.





Problem 3

1. According to plots, Adam works better then others. Firsrt, the accuracy climbs faster than others, even though others can reach the same accuracy after 50 epochs training. Also, it can get the best accuracy and less traning cost than other optimizations.
   1) Standard SGD: $lr = 0.05, weight_decay = 0.000000045$ $Acc = 97.77\%$
   2) SGD with momentum: $lr = 0.05, weight_decay = 0.00000005$ $Acc = 97.67\%$
   3) Adagrad: $lr = 0.05, weight_decay = 0.00000005$ $Acc = 97.57\%$
   4) Adam: $lr = 0.0005, weight_decay = 0.000000045$ $Acc = 98.05\%$

2. According to my network, I used Adam to apply batch normalization to each of the hidden layers with $lr = 0.0001, weight_decay = 0.000000045$. The batch normalization seem to help a litte bit in here because even the first epoch it can get about 94% on validation accuracy. Also the accuracy can stay above 98% more than other models. The best accuracy I can get in here is: 98.06%

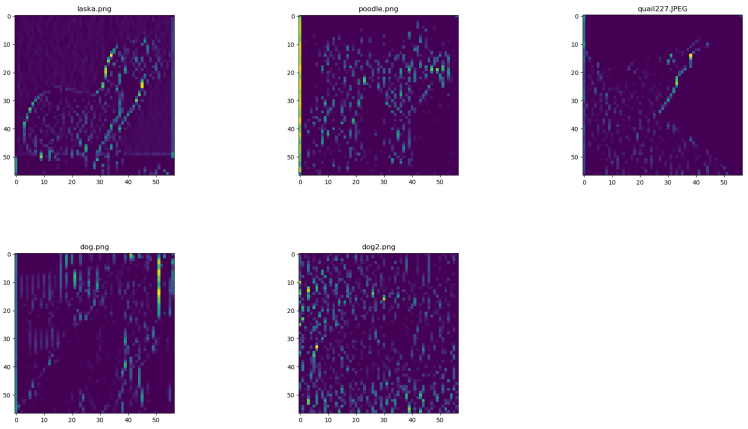
Problem 4

1. Download the AlexNet

2. Output shapes: (1, 57, 57, 96)

3. Dimension of the final layer: (1, 1000)

4. Using test.png and the classification seems correct.
   Labrador retriever 0.690323



Weimaraner 0.24764444
Chesapeake Bay retriever 0.046632566
flat-coated retriever 0.0040035252
curly-coated retriever 0.0028921247
The test image is two Labrador retrievers.

Problem 5

1. Backpropagation in a convolutional network: [the full process is way too long, so only the final answer in here]
   1. does not change.
   2.

$$\delta^1_{j,k} = \begin{cases} 0 & \text{if } a^1_{j,k} \neq max(a^1_{2j',2k'}, a^1_{2j',2k'+1}, a^1_{2j'+1,2k'}, a^1_{2j'+1,2k'+1}) \\ \sum_{l=0}^{9} \delta^3_l w_{l;j',k'} \sigma'(z^1_{j,k}) & \text{if } a^1_{j,k} = max(a^1_{2j',2k'}, a^1_{2j',2k'+1}, a^1_{2j'+1,2k'}, a^1_{2j'+1,2k'+1}) \end{cases} \tag{1}$$

   3. $\frac{\partial C}{\partial b^1} = \sum_{0 \leq j,k \leq 23} \delta^1_{j,k}$ or $\frac{\partial C}{\partial b^1} = \delta^3_l$
   4. $\frac{\partial C}{\partial w^3_{l;j,k}} = a^2_{j,k}\delta^3_l$ or $\frac{\partial C}{\partial w^1_{l,m}} = \sum_{0 \leq j,k \leq 23} \delta^1_{j,k} a^0_{j+l,k+m}$

2. The final accuracy I can get is about 99.4%. At first, applying the original code from the website with the learning rate is set to 0.001 where I can get 98.33% accuracy. After changing the learning rate to 0.05 the accuracy gets to 98.45%. Second, I changed kernal from [2, 2] [2, 2] to [4, 4] [1, 1] the accuracy is about 99.16%. Last, I incease the steps from 1000 to 20000 to get 99.4% accuracy. Therefore, based on the result I had, CNN seem to do better than other models.