Deep Learning Hwk 3

Cody Grogan
A02313514

October 2023

Problem 1

1)

The benefit of online learning is that learning occurs much faster in each epoch. This is because in a mini-batch of 20 data points only a single gradient descent step is taken. Where as online learning would have taken 20. The disadvantage of online learning is that the gradient may be noisy and gradient steps may be taken in a non-optimal direction. Where as the mini-batches are much more robust to fluctuations in the cost function.

2)

I used PyTorch for this section and got a Test accuracy of 97.89, learning rate 0.25, 200 hidden nodes, and mini batch of 20. I also used a weight decay of 0.0001.

Problem 2

1)

Starting with the original equation for δ^L,

$$\delta^L = \begin{bmatrix} \frac{\partial C}{\partial a_1^L} \\ \frac{\partial C}{\partial a_2^L} \\ \vdots \\ \frac{\partial C}{\partial a_n^L} \end{bmatrix} \odot \begin{bmatrix} \sigma'(z_1^L) \\ \sigma'(z_2^L) \\ \vdots \\ \sigma'(z_n^L) \end{bmatrix} = \begin{bmatrix} \frac{\partial C}{\partial a_1^L}\sigma'(z_1^L) \\ \frac{\partial C}{\partial a_2^L}\sigma'(z_2^L) \\ \vdots \\ \frac{\partial C}{\partial a_n^L}\sigma'(z_n^L) \end{bmatrix} \tag{1}$$

Now we know that any matrix multiplied by the identity matrix is itself because the identity matrix contains all of the basis vectors for the space. So it would go to say that if put our $\sigma'(z^L)$ on the diagonal we would see that the linear transformation is simply the basis vectors for the space multiplied by our corresponding activation.

$$\delta^L = \begin{bmatrix} \sigma'(z_1^L) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma'(z_n^L) \end{bmatrix} \begin{bmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_n^L} \end{bmatrix} = \begin{bmatrix} \frac{\partial C}{\partial a_1^L}\sigma'(z_1^L) \\ \vdots \\ \frac{\partial C}{\partial a_n^L}\sigma'(z_n^L) \end{bmatrix} \tag{2}$$

Thus showing the equivalence of the representations

2)

Following the same logic as the previous question we see,

$$\begin{aligned} \delta^l &= ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) && (3) \\ &= \left(\begin{bmatrix} w_{11}^{l+1} & w_{21}^{l+1} & \dots & w_{n1}^{l+1} \\ w_{12}^{l+1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ w_{1n}^{l+1} & \dots & \dots & w_{nn}^{l+1} \end{bmatrix} \begin{bmatrix} \delta_1^{l+1} \\ \delta_2^{l+1} \\ \vdots \\ \delta_n^{l+1} \end{bmatrix} \right) \odot \begin{bmatrix} \sigma'(z_1^l) \\ \sigma'(z_2^l) \\ \vdots \\ \sigma'(z_n^l) \end{bmatrix} && (4) \\ &= \begin{bmatrix} \sum_j^n (w_{j1}^{l+1}\delta_j^{l+1}) \\ \sum_j^n (w_{j2}^{l+1}\delta_j^{l+1}) \\ \vdots \\ \sum_j^n (w_{jn}^{l+1}\delta_j^{l+1}) \end{bmatrix} \odot \begin{bmatrix} \sigma'(z_1^l) \\ \sigma'(z_2^l) \\ \vdots \\ \sigma'(z_n^l) \end{bmatrix} = \begin{bmatrix} \sum_j^n (w_{j1}^{l+1}\delta_j^{l+1}) \cdot \sigma'(z_1^l) \\ \sum_j^n (w_{j2}^{l+1}\delta_j^{l+1}) \cdot \sigma'(z_2^l) \\ \vdots \\ \sum_j^n (w_{jn}^{l+1}\delta_j^{l+1}) \cdot \sigma'(z_n^l) \end{bmatrix} && (5) \end{aligned}$$

So now substituting in our $\Sigma'(z)$ we get the same result,

$$\delta^l = \begin{bmatrix} \sigma'(z_1^l) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma'(z_n^l) \end{bmatrix} \begin{bmatrix} \sum_j^n (w_{j1}^{l+1}\delta_j^{l+1}) \\ \sum_j^n (w_{j2}^{l+1}\delta_j^{l+1}) \\ \vdots \\ \sum_j^n (w_{jn}^{l+1}\delta_j^{l+1}) \end{bmatrix} = \begin{bmatrix} \sum_j^n (w_{j1}^{l+1}\delta_j^{l+1})\sigma'(z_1^l) \\ \sum_j^n (w_{j2}^{l+1}\delta_j^{l+1})\sigma'(z_2^l) \\ \vdots \\ \sum_j^n (w_{jn}^{l+1}\delta_j^{l+1})\sigma'(z_n^l) \end{bmatrix} \tag{6}$$

3)

This is fairly trivial to solve starting with an arbitrary error at layer l,

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \delta^{l+1} \tag{7}$$

We then see we can substitute the same equation in for δ^{l+1},

$$\begin{aligned} \delta^l &= \Sigma'(z^l)(w^{l+1})^T \Sigma'(z^{l+1})(w^{l+2})^T \delta^{l+3} && (8) \\ &= \Sigma'(z^l)(w^{l+1})^T \Sigma'(z^{l+1})(w^{l+2})^T \quad ... \quad (w^L)^T \Sigma'(z^L) \nabla_a C && (9) \end{aligned}$$

4)

For the first fundamental equation we would see,

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{10}$$

$$= \sum_k \frac{\partial C}{\partial a_k^L}\frac{\partial z_k^L}{\partial z_j^L} \tag{11}$$

$$= \frac{\partial C}{\partial a_j^L} \tag{12}$$

$$\tag{13}$$

From this we see that the error at node j in the output layer is simply the partial of the cost with respect to the activation. For the second fundamental equation we would see the derivative of the activation will always be 1 so,

$$\delta^l = (w^{l+1})^T \delta^{l+1} \tag{14}$$

For the third fundamental equation we see,

$$\frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial a_k^l}\frac{\partial a_k^l}{\partial z_k^l}\frac{\partial z_k^l}{\partial b_j^l} \tag{15}$$

$$= \frac{\partial C}{\partial a_j^l}\frac{\partial a_j^l}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \tag{16}$$

$$= \frac{\partial C}{\partial a_j^l} \tag{17}$$

Now for the Fourth fundamental equation we get

$$\frac{\partial C}{\partial w_{kj}^l} = \sum_i \frac{\partial C}{\partial a_i^l}\frac{\partial a_i^l}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{kj}^l} \tag{18}$$

$$= \sum_i \frac{\partial C}{\partial a_i^l}\frac{\partial a_i^l}{\partial z_i^l}\frac{\partial}{\partial w_{kj}^l}(\sum_m w_{mi}^l a_m^{l-1} + b_m^l) \tag{19}$$

$$= \frac{\partial C}{\partial a_k^l} a_j^{l-1} \tag{20}$$

Now the back propagation algorithm would follow the form,

$$\delta^l = (w^{l+1})^T (w^{l+2})^T \quad ... \quad (w^L)^T \nabla_a C \tag{21}$$

Problem 3

Starting with the bad expression for cross entropy,

$$C = -(a\ln(y) + (1-a)\ln(1-y)) \tag{22}$$

When y is either 1 or 0 we get one term with the ln(0) which is uncomputable. However in the other expression the y's are the coefficients of the natural log which have computable results. In addition, our a term from the output of the network will never be 0 because the our z would have to be infinite. Thus making our natural log always computable.

2)

So this question we know the probability mass function for a Bernoulli distribution is,

$$P(y|x) = \begin{cases} x & y = 1 \\ (1-x) & y = 0 \end{cases} = p^y(1-p)^{1-y} \tag{23}$$

Deriving the binary cross entropy loss from equation 1 we see,

$$\begin{aligned} L(\theta) &= -\ln(E_{x,y\ P}\left[\hat{y}^{y_i}(1-\hat{y})^{1-y_i}\right]) && (24)\\ &= -\ln(\prod_{i=1}^{n}(\hat{y}_i^{yi}(1-\hat{y}_i)^{1-y_i}) && (25)\\ &= -\frac{1}{n}\sum_{i=1}^{n} y_i\ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i) && (26) \end{aligned}$$

Where n is the size of the minibatch and $\hat{y}$ is the output from our network that represents the probability of success.

3)

Using the definition of the joint probability mass function for a multinoulli distribution, where n is our number of classes, y is a nx1 vector with all values 0 except for the class and p is a nx1 vector of probabilities for each class

$$P(y|x,\theta) = \prod_{i=1}^{n} p_i^{y_i} \tag{27}$$

Because we are using the softmax of our output we know that our output $\hat{y}(x,\theta)$ is a nx1 vector containing our probabilities of success. Now plugging this into equation 1 we see,

$$L(\theta) = -\ln\left(\prod_{i=1}^{k}\prod_{j=1}^{n} p_{i,j}^{y_{i,j}}\right) \tag{28}$$

$$= -\sum_{i=1}^{k}\ln\left(\prod_{j=1}^{n} p_{i,j}^{y_{i,j}}\right) \tag{29}$$

Now we see that the for the inner product the only probability not equal to 1 is when $y_j = 1$ so the product will have the form,

$$P(y_l = 1|x) = 1 \cdot 1 \ldots p_l \ldots 1 \tag{30}$$

And in addition we know that ln(1) is zero so the negative log likelihood has the form,

$$L(\theta) = -\frac{1}{n}\sum_{i=1}^{k} y_l \ln\left(p_{i,l}\right) \tag{31}$$

Where l the index of the correct class for for the input and n is the size of the minibatch.

4)

Starting with taking the derivative of our cross entropy function of a with respect to the activation,

$$\frac{\partial C}{\partial a} = \frac{\partial}{\partial a} - \frac{1}{n}\sum_{i=1}^{n} y\ln(a) + (1-y)\ln(1-a) \tag{32}$$

$$= \frac{1}{n}\sum_{i=1}^{n} -\frac{y}{a} + \frac{1-y}{1-a} \tag{33}$$

Now just looking a single training example and setting the partial equal to 0,

$$0 = -\frac{y}{a} + \frac{1-y}{1-a} \tag{34}$$

$$\frac{y}{a} = \frac{1-y}{1-a} \tag{35}$$

$$y - ya = a - ay \tag{36}$$

$$y = a \tag{37}$$

From this we see that there is critical point where the output of the activation function is equal y. Now to check if the point is a maxima or minima we take the second derivative of the cost,

$$\frac{\partial^2 C}{\partial a^2} = \frac{\partial}{\partial a} - \frac{y}{a} + \frac{1-y}{1-a} \tag{38}$$

$$= \frac{y}{a^2} + \frac{1-y}{(1-a)^2} \tag{39}$$

Now if $y \in (0,1)$ and $a \in (0,1)$ then the result of the evaluation of the second derivative will always be positive showing the point y = a is a minima.

5)

The softmax gets its name because it is a differentiable argmax of the inputs.

6)

Using the first fundamental equation we see the following where $y_k = 1$,

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \tag{40}$$

$$= \frac{\partial C}{\partial a_j^L} \frac{\partial}{\partial z_j^L} \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \tag{41}$$

$$= \frac{\partial C}{\partial a_j^L} \frac{e^{z_j^L} \sum_k e^{z_k^L} - e^{2z_k^L}}{(\sum_k e^{z_k^L})^2} \tag{42}$$

$$= \frac{\partial C}{\partial a_j^L} \frac{e^{z_j^L}}{\sum_k e^{z_k^L}} \frac{\sum_k e^{z_k^L} - e^{z_k^L}}{\sum_k e^{z_k^L}} \tag{43}$$

$$= (\frac{\partial}{\partial a_j^L} - \ln(a_j^L))(a_j^L(1 - a_j^L) \tag{44}$$

$$= -\frac{1}{a_j^L}(a_j^L(1 - a_j^L)) \tag{45}$$

$$= a_j^L - 1 \tag{46}$$

Now because of the classification we know that the -1 would only be for the case where j is the correct classification so,

$$\delta_j^L = a_j^L - y_j \tag{47}$$

Now for the case where $y_m = 1$

$$\delta_j^L = \frac{\partial C}{\partial a_m^L} \frac{\partial a_j^m}{\partial z_j^L} \tag{48}$$

$$= \frac{\partial C}{\partial a_m^L} \frac{\partial}{\partial z_j^L} \frac{e^{z_m^L}}{\sum_k e^{z_k^L}} \tag{49}$$

$$= \frac{\partial C}{\partial a_m^L} \frac{-e^{z_j^L} e^{z_m^L}}{(\sum_k e^{z_k^L})^2} \tag{50}$$

$$= -\frac{1}{a_m^L} \frac{-e^{z_j^L}}{\sum_k e^{z_k^L}} a_m^L \tag{51}$$

$$= a_j^L \tag{52}$$

Then since our $y_j = 0$ we can add it to the final result,

$$\delta_j^L = a_j^L - y_j \tag{53}$$

This then shows that the equation holds true for the error of all output nodes.

Problem 4

1)

Starting with the regularized logistic regression equation,

$$l(w, w_0) = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \ln\left(\frac{1}{1+e^{-\hat{w}^T x_i}}\right) + (1-y_i) \ln\left(\frac{e^{-\hat{w}^T x_i}}{1+e^{-\hat{w}^T x_i}}\right) \right] + \lambda \|w\|^2 \tag{54}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \ln\left(\frac{1}{1+e^{-\hat{w}^T x_i}}\right) + (1-y_i) \ln\left(\frac{e^{-\hat{w}^T x_i}}{1+e^{-\hat{w}^T x_i}}\right) \right] + \lambda \hat{w}^T \hat{w} \tag{55}$$

$$\tag{56}$$

Now taking the gradient with respect to the weights where $\hat{w}^T x_i$ is h(x), $1 + e^{-h(x)}$ is g(x). For the first term we get,

$$= y_i \frac{\partial h(x)}{\partial w} \frac{\partial g(x)}{\partial h(x)} \frac{\partial f(x)}{\partial g(x)} \ln(\frac{1}{g(x)}) \tag{57}$$

$$= -y_i x_i (-e^{\hat{w}^T x_i}) \frac{1}{1 + e^{\hat{w}^T x_i}} \tag{58}$$

$$= y_i x_i \frac{e^{\hat{w}^T x_i}}{1 + e^{\hat{w}^T x_i}} \tag{59}$$

$$\tag{60}$$

For the second term we get where g(x) is $\frac{e^{-h(x)}}{1+e^{-h(x)}}$

$$= (1 - y_i) \frac{\partial h(x)}{\partial w} \frac{\partial g(x)}{\partial h(x)} \frac{\partial f(x)}{\partial g(x)} \ln(g(x)) \tag{61}$$

$$= (1 - y_i) x_i \frac{-e^{-h(x)}(1 + e^{-h(x)}) + e^{-h(x)} e^{-h(x)}}{(1 + e^{-h(x)})^2} \frac{(1 + e^{-h(x)})}{e^{-h(x)}} \tag{62}$$

$$= (1 - y_i) x_i \frac{-(1 + e^{-h(x)}) + e^{-h(x)}}{(1 + e^{-h(x)})} \tag{63}$$

$$= (1 - y_i) x_i \frac{-1}{(1 + e^{-\hat{w}^T x_i})} \tag{64}$$

$$\tag{65}$$

Now for the regularization term,

$$= 2\lambda w \tag{66}$$

Adding them all together we get,

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i x_i \frac{e^{\hat{w}^T x_i}}{1 + e^{\hat{w}^T x_i}} + (y_i - 1) x_i \frac{1}{(1 + e^{-\hat{w}^T x_i})} \right] + 2\lambda w \tag{67}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i x_i - x_i \frac{1}{1 + e^{-\hat{w}^T x_i}} \right] + 2\lambda w \tag{68}$$

$$\tag{69}$$

Now taking the gradient with respect to the bias we notice that the result should be the same but the only non 0 part of the x_i vector is the 1 added on to the end. And the gradient of the regularization term is simply the the bias so,

$$= -\frac{1}{n} \sum_{i=1}^{n} \left[y_i - \frac{1}{1 + e^{-\hat{w}^T x_i}} \right] \tag{70}$$

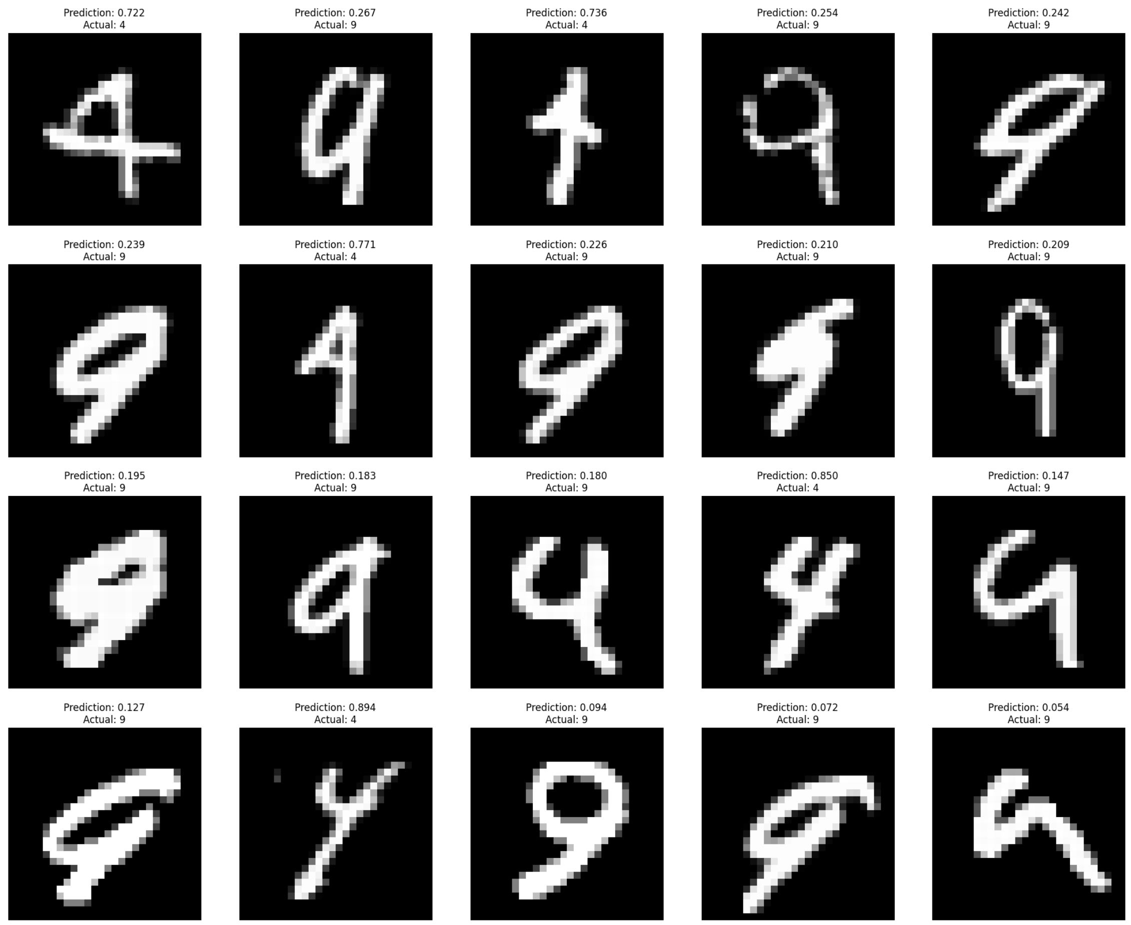
$$\tag{71}$$

2)

For $\lambda = .01$ I got a test accuracy of 95.1% with a learning rate of: 0.01, using xavier uniform initialization, and a train loss 0.112.
For $\lambda = 1$ I got a test accuracy of 87% with a learning rate of: 0.01, using xavier uniform initialization, and a train loss 0.477.

For both cases I used a stopping condition in which I stop if the test accuracy (this biases the tuning) doesn't improve over 30 epochs.

3)


Prediction: 0.722
Actual: 4
Prediction: 0.267
Actual: 9
Prediction: 0.736
Actual: 4
Prediction: 0.254
Actual: 9
Prediction: 0.242
Actual: 9
Prediction: 0.239
Actual: 9
Prediction: 0.771
Actual: 4
Prediction: 0.226
Actual: 9
Prediction: 0.210
Actual: 9
Prediction: 0.209
Actual: 9
Prediction: 0.195
Actual: 9
Prediction: 0.183
Actual: 9
Prediction: 0.180
Actual: 9
Prediction: 0.850
Actual: 4
Prediction: 0.147
Actual: 9
Prediction: 0.127
Actual: 9
Prediction: 0.894
Actual: 4
Prediction: 0.094
Actual: 9
Prediction: 0.072
Actual: 9
Prediction: 0.054
Actual: 9