EECS663: Homework II

Vimig Socrates

March 2021

1 Problem 1

1.1 Ex. 1

In the 1D case, the loss landscape $C(v)$ is represented as a line shape, with the x-axis being the only variable v_1, and the y-axis being the loss. Therefore, in a simple case of a quadratic curve, our "rolling ball" would roll down to the "bottom" (minimum) of the quadratic.

1.2 Ex. 2

An advantage of online learning is the same as classic SGD, which allows the model the ability to jump out of saddle points and local minima quickly as further iterations of backprop are run. However, when performing backprop on only one sample at a time, this also works against the model. With noisy data, a particular example may adjust the weights incorrectly and throw the backprop algorithm wildly away from a potentially effective local minima. Therefore mini-batching over a subset of the entire data finds a balance between the advantage and disadvantage of SGD.

2 Problem 2

2.1 Ex. 1

If there is only a single neuron that uses a nonsigmoidal activation, then when we compute the backpropagated error for the layer that the nonsigmoidal neuron is on l: δ_j^l, we just need to use the correct derivative of the other activation function f'.

2.2 Ex. 2

In order to validate $\delta_j^L = a_j^L - y_i$, we must compute the two partials with respect to the weights and biases: $\frac{\partial C}{\partial b_j^L} = a_j^L - y_j$ and $\frac{\partial C}{\partial w_{jk}^L} = a_k^{L-1}(a_j^L - y_j)$. Starting with the biases:

$$\frac{\partial C}{\partial b_j^L} = \sum_k \frac{\partial C}{\partial z_k^L} \cdot \frac{\partial z_k^L}{\partial b_j^L} \tag{1}$$

Unlike in the original derivation, we are not able to only choose one of the elements in the summation. We find the derivative of the log likelihood cost function wrt the activation to get (we also use the derivative of the softmax function, which is $a_j^L(1 - a_j^L)$):

$$\delta_j^L = \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} = \frac{\partial - \ln a_y^L}{\partial a_k^L} = -\frac{1}{a_y^L} \cdot a_j^L(1 - a_j^L) \tag{2}$$

For the jth value of y y_j which is the same as when $j = y$:

$$\delta_j^L = -\frac{1}{a_y^L} \cdot a_j^L(1 - a_j^L) = a_y^L - 1_{j=y} = a_y^L - y_j \tag{3}$$

We can show a similar relationship for the weights term as following:

$$\frac{\partial C}{\partial w_{jk}^L} = \sum_i \frac{\partial C}{\partial z_i^l} \cdot \frac{\partial z_i^l}{\partial w_{jk}^L} = \sum_i \frac{\partial C}{\partial a_i^L} \frac{\partial a_i^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{jk}^L} \tag{4}$$

The first two partials are the same from above, and for we get:

$$\frac{\partial z_j^L}{\partial w_{jk}^L} = \frac{\partial \sum_i w_{ij}^L a_i^{L-1} + b_j^L}{\partial w_{jk}^L} \tag{5}$$

Since only the weight of the respective partial applies (i.e. partials where $i \neq j = 0$), we can simplify to:

$$((w^{l+1})^T \delta^{l+1}) = a_k^{L-1} \tag{6}$$

So put all together, we get what we expect:

$$\frac{\partial C}{\partial w_{jk}^L} \quad = \quad a_k^{L-1}(a_j^L - y_j) \tag{7}$$

Therefore, we've shown that we get the expected backprop of the weights and biases using $\delta_j^L = a_j^L - y_j$.

2.3 Ex. 3

In backprop with linear neurons, we can replace the $\sigma(z) = z$ and $\sigma'(z) = 1$. This makes the following questions:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) = \frac{\partial C}{\partial a_j^L}1 = \frac{\partial C}{\partial a_j^L} \tag{8}$$

This translates to:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) = \nabla_a C \tag{9}$$

Also, to backpropagate the rest of the errors:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) = ((w^{l+1})^T \delta^{l+1}) \tag{10}$$

3 Problem 3

3.1 Ex. 1

In the second equation, we run into significant issues if either $y = 0$ or $y = 1$ because either the left or the right term (respectively) is undefined, as $\ln 0 = undef$. This issue doesn't afflict the first equation, since the model will almost never predict with 100% confidence that a prediction is either 0 or 1. If it does, we would have the same issue, however this is unlikely to be the case. Moreover, we want only one of the sides of the + to hold if our class label is either 0 or 1. In the second equation, this would not be the case, both sides of the equation would impact the loss (incorrectly).

3.2 Ex. 2

In order to determine if C is minimized, we have to take the partial derivative with respect to the output z like so:

$$C = -\frac{1}{n}\sum_x [y \ln \hat{y} + (1-y) ln(1-\hat{y})] \tag{11}$$

We assume that the $\hat{y} = \sigma(z)$:

$$\frac{\partial C}{\partial z} = -y\frac{1}{\sigma(z)}\sigma'(z) + (1-y)\frac{1}{1-\sigma(z)}\sigma'(z) \tag{12}$$

$$\sigma'(z) = \frac{-y}{\sigma(z)} + \frac{(1-y)}{1-\sigma(z)} \tag{13}$$

Set it equal to 0 to find the extrema:

$$\sigma'(z) = \frac{\partial C}{\partial z} = \frac{-y}{\sigma(z)} + \frac{(1-y)}{1-\sigma(z)} \tag{14}$$

$$0 = \frac{-y}{\sigma(z)} + \frac{(1-y)}{1-\sigma(z)} \tag{15}$$

$$\frac{y}{\sigma(z)} = \frac{(1-y)}{1-\sigma(z)} \tag{16}$$

$$y(1-\sigma(z)) = \sigma(z)(1-y) \tag{17}$$

$$y = \sigma(z) \tag{18}$$

Then we must take the second derivative to make sure that we have a minima:

$$\frac{\partial^2 C}{\partial z^2} = \sigma''(z) = \frac{y}{\sigma(z)^2} + \frac{(1-y)}{(1-\sigma(z))^2} \tag{19}$$

$$\tag{20}$$

Since we know that both $1 > \sigma(z) > 0$ and $1 > z > 0$, we know that the above equation is ≥ 1 so our extrema is a minima.

3.3 Ex. 3

We must first perform the forward pass for both the hidden neurons:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 = \tag{21}$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 = 0.3775 \tag{22}$$

$$net_{h1} = 0.2 * 0.05 + 0.3 * 0.1 + 0.35 = 0.39 \tag{23}$$

$$\tag{24}$$

Squash them both with the sigmoid:

$$out_{h1} = \frac{1}{1-e^{-0.3775}} = 0.59327 \tag{25}$$

$$out_{h2} = \frac{1}{1-e^{-0.39}} = 0.596282 \tag{26}$$

We repeat this with the inputs from this past section for the two output nodes.

$$out_{o1} = \frac{1}{1+e^{-1.105905967}} = 0.75137 \tag{27}$$

$$out_{o2} = \frac{1}{1-e^{-0.39}} = 0.77287 \tag{28}$$

Now we try and calculate the error based on cross-entropy:

$$E_{o1} = 0.01 * \ln(0.75137) + (1 - 0.01) * \ln(1 - 0.75137) * 1.38073 \tag{29}$$
$$= 1.905327 \tag{30}$$
$$E_{o2} = 0.99 * \ln(0.77287) + (1 - 0.99) * \ln(1 - 0.77287) * 1.38073 \tag{31}$$
$$= 0.2755336 \tag{32}$$
$$E_{tot} = 0.2755336 + 1.905327 = 2.1808636 \tag{33}$$

In order to backprop, we can use the equations for the partial derivatives of the cross-entropy loss function wrt the weight and bias terms found in the Nielsen book:

$$\frac{\partial C}{\partial w_j} = x_j(\sigma(z) - y) \tag{34}$$
$$\frac{\partial C}{\partial b} = (\sigma(z) - y) \tag{35}$$

Using the numbers we have, we get:

$$\frac{\partial C}{\partial w_5} = 0.59327 * (0.75137 - 0.01) = 0.4398 \tag{36}$$
$$\frac{\partial C}{\partial w_6} = 0.596282 * (0.75137 - 0.01) = 0.44207 \tag{37}$$
$$\frac{\partial C}{\partial w_7} = 0.59327 * (0.77287 - 0.99) = -0.1288 \tag{38}$$
$$\frac{\partial C}{\partial w_8} = 0.596282 * (0.77287 - 0.99) = -0.1295 \tag{39}$$
$$\frac{\partial C}{\partial b2_{o1}} = (0.75137 - 0.01) = 0.74137 \tag{40}$$
$$\frac{\partial C}{\partial b2_{o2}} = (0.77287 - 0.99) = -0.21713 \tag{41}$$
$$\tag{42}$$

However, for the **hidden neurons**, we will need an additional application of the chain rule to take into consideration the errors from both outputs in the final layer that a particular weight in the hidden layer has of impacting. Therefore:

$$\frac{\partial C}{\partial w_1^1} = \frac{\partial C}{\partial z_1^1}\frac{\partial z_1^1}{\partial w_1^1} \tag{43}$$

$$\frac{\partial C}{\partial w_2^1} = \frac{\partial C}{\partial z_1^1}\frac{\partial z_1^1}{\partial w_2^1} \tag{44}$$

$$\frac{\partial C}{\partial w_3^1} = \frac{\partial C}{\partial z_2^1}\frac{\partial z_2^1}{\partial w_3^1} \tag{45}$$

$$\frac{\partial C}{\partial w_4^1} = \frac{\partial C}{\partial z_2^1}\frac{\partial z_2^1}{\partial w_4^1} \tag{46}$$

$$\tag{47}$$

In the above, z_j^1 is the weighted input sum at hidden unit j. In order to get this partial, we have to sum over both the output errors, like so:

$$\frac{\partial C}{\partial z_j^1} = \sum_{i=1}^{2} \frac{\partial C}{\partial z_i}\frac{\partial z_i}{\partial a_j}\frac{\partial a_j}{\partial z_j^1} \tag{48}$$

$$= \sum_{i=1}^{2} (\sigma(z) - y)(w_{ji})(a_j(1 - a_j)) \tag{49}$$

This gets us:

$$\frac{\partial C}{\partial w_j^1} = \sum_{i=1}^{2} (\sigma(z) - y)(w_{ji})(a_j(1 - a_j))(i_k) \tag{50}$$

where a_j is the activation from the respective hidden neuron, w_{ji} is the edge leading to the respective output node from the particular hidden node, and i_k is the respective input example value.

Plugging in numbers, we have:

$$\frac{\partial C}{\partial w_1^1} = \Big((0.75137 - 0.01) * (0.4) * (0.59327 * (1 - 0.59327)) * (0.05)\Big) + \tag{51}$$

$$\Big((0.77287 - 0.99) * (0.45) * (0.59327 * (1 - 0.59327))(0.05)\Big) = 0.002399$$

$$\frac{\partial C}{\partial w_2^1} = \Big((0.75137 - 0.01) * (0.5) * (0.596282 * (1 - 0.596282)) * (0.05)\Big) + \tag{52}$$

$$\Big((0.77287 - 0.99) * (0.55) * (0.596282 * (1 - 0.596282))(0.05)\Big) = 0.003024$$

$$\frac{\partial C}{\partial w_3^1} = \Big((0.75137 - 0.01) * (0.4) * (0.59327 * (1 - 0.59327)) * (0.1)\Big) + \tag{53}$$

$$\Big((0.77287 - 0.99) * (0.45) * (0.59327 * (1 - 0.59327))(0.1)\Big) = 0.004798$$

$$\frac{\partial C}{\partial w_4^1} = \Big((0.75137 - 0.01) * (0.5) * (0.596282 * (1 - 0.596282)) * (0.1)\Big) + \tag{54}$$

$$\Big((0.77287 - 0.99) * (0.55) * (0.596282 * (1 - 0.596282))(0.1)\Big) = 0.00605 \tag{55}$$

Biases are similar:

$$\frac{\partial C}{\partial b_1^1} = \frac{\partial C}{\partial z_1^1}\frac{\partial z_1^1}{\partial b_1^1} \tag{56}$$

$$\frac{\partial C}{\partial b_2^1} = \frac{\partial C}{\partial z_2^1}\frac{\partial z_2^1}{\partial b_2^1} \tag{57}$$

$$\tag{58}$$

Which leads to:

$$\frac{\partial C}{\partial b_1^1} = \Big((0.75137 - 0.01) * (0.4) * (0.59327 * (1 - 0.59327))\Big) + \tag{59}$$

$$\Big((0.77287 - 0.99) * (0.45) * (0.59327 * (1 - 0.59327))\Big) = 0.04798$$

$$\frac{\partial C}{\partial b_2^1} = \Big((0.75137 - 0.01) * (0.5) * (0.596282 * (1 - 0.596282))\Big) + \tag{60}$$

$$\Big((0.77287 - 0.99) * (0.55) * (0.596282 * (1 - 0.596282))\Big) = 0.06049 \tag{61}$$

4 Bonus

If we consider the following definition of softmax:

$$a_j^L = \frac{e^{cz_j^L}}{\sum_k e^{cz_k^L}} \tag{62}$$

where, conventionally, $c = 1$, but instead make $c \to \infty$, we approach argmax. The max of a_j^L is 1, and everything else is 0. Therefore, with $c = 1$, this is a "soft" version of the argmax function, where other values in the vector are still influential, instead of hard binary values.