Deep Learning Homework 3

Austin Gardiner

October 13, 2023

1 Stochastic Gradient Descent on MNIST

1.1 Incremental Learning

One disadvantage to online learning (batch size of 1) is that the model takes longer to train because each iteration only using one training input. This is especially true when using GPUs because the system is able to use parallel processing with larger mini-batch sizes.

One advantage of online learning is that the process of updating weights after each training input helps the program generalize better and avoid shallow local minima.

1.2 Neural Network with Two Hidden Layers

I used Pytorch and based my code off the starter code. My best performing model used 3136 for the first hidden layer size, 1568 for the second hidden layer size, a learning rate of 0.1, a batch size of 32, and trained for 6 epochs. This model had an accuracy of 98.0%. The code used is submitted along with this document.

2 Backpropagation

2.1 Show that $\delta^L = \nabla_a C \odot \sigma'(z^L) = \sum'(z^L)\nabla_a C$

1. set $\delta^L = \nabla_a C \odot \sigma'(z^L) = \sum'(z^L)\nabla_a C$

2. LHS

$$\nabla_a C \odot \sigma'(z^L) = \begin{bmatrix} \frac{\partial C}{\partial a_1^L} \\ \frac{\partial C}{\partial a_2^L} \\ \cdots \\ \frac{\partial C}{\partial a_n^L} \end{bmatrix} \odot \begin{bmatrix} \sigma'(z_1^L) \\ \sigma'(z_2^L) \\ \cdots \\ \sigma'(z_n^L) \end{bmatrix} = \begin{bmatrix} \frac{\partial C}{\partial a_1^L}\sigma'(z_1^L) \\ \frac{\partial C}{\partial a_2^L}\sigma'(z_2^L) \\ \cdots \\ \frac{\partial C}{\partial a_n^L}\sigma'(z_n^L) \end{bmatrix}$$

3. RHS

$$\sum'(z^L)\nabla_a C = \begin{bmatrix} \sigma'(z_1^L) & 0 & \cdots & 0 \\ 0 & \sigma'(z_2^L) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma'(z_n^L) \end{bmatrix} \begin{bmatrix} \frac{\partial C}{\partial a_1^L} \\ \frac{\partial C}{\partial a_2^L} \\ \cdots \\ \frac{\partial C}{\partial a_n^L} \end{bmatrix} = \begin{bmatrix} \sigma'(z_1^L)\frac{\partial C}{\partial a_1^L} \\ \sigma'(z_2^L)\frac{\partial C}{\partial a_2^L} \\ \cdots \\ \sigma'(z_n^L)\frac{\partial C}{\partial a_n^L} \end{bmatrix}$$

4. compare

because $\frac{\partial C}{\partial a_i^L}$ and $\sigma'(z_i^L)$ are both scalar values, $\frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) = \sigma'(z_j^L)\frac{\partial C}{\partial a_j^L}$

$$\therefore \begin{bmatrix} \frac{\partial C}{\partial a_1^L}\sigma'(z_1^L) \\ \frac{\partial C}{\partial a_2^L}\sigma'(z_2^L) \\ \cdots \\ \frac{\partial C}{\partial a_n^L}\sigma'(z_n^L) \end{bmatrix} = \begin{bmatrix} \sigma'(z_1^L)\frac{\partial C}{\partial a_1^L} \\ \sigma'(z_2^L)\frac{\partial C}{\partial a_2^L} \\ \cdots \\ \sigma'(z_n^L)\frac{\partial C}{\partial a_n^L} \end{bmatrix} \Rightarrow \nabla_a C \odot \sigma'(z^L) = \sum'(z^L)\nabla_a C$$

2.2 Show that $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) = \sum'(z^l)(w^{l+1})^T \delta^{l+1}$

$$\delta^l = \left((w^{l+1})^T \delta^{l+1}\right) \odot \sigma'(z^l) = \sum'(z^l)\,(w^{l+1})^T \delta^{l+1}$$

$$\left[(w^{l+1})^T \delta^{l+1}\right]_j = \sum_k w^{l+1}_{kj}\, \delta^{l+1}_k$$

$$\text{LHS: } (w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l) = \begin{bmatrix} \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k \\ \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k \\ \dots \\ \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k \end{bmatrix} \odot \begin{bmatrix} \sigma'(z^l_1) \\ \sigma'(z^l_2) \\ \dots \\ \sigma'(z^l_n) \end{bmatrix} = \begin{bmatrix} \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k\, \sigma'(z^l_1) \\ \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k\, \sigma'(z^l_2) \\ \dots \\ \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k\, \sigma'(z^l_n) \end{bmatrix}$$

$$\text{RHS: } \sum'(z^l)\,(w^{l+1})^T \delta^{l+1} = \begin{bmatrix} \sigma'(z^l_1) & 0 & \dots & 0 \\ 0 & \sigma'(z^l_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \sigma'(z^l_n) \end{bmatrix} \begin{bmatrix} \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k \\ \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k \\ \dots \\ \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k \end{bmatrix} = \begin{bmatrix} \sigma'(z^l_1) \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k \\ \sigma'(z^l_2) \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k \\ \dots \\ \sigma'(z^l_n) \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k \end{bmatrix}$$

$\sigma'(z^l_i)$ and $\sum_k w^{l+1}_{kj}\, \delta^{l+1}_k$ both become scalars so

$$\begin{bmatrix} \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k\, \sigma'(z^l_1) \\ \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k\, \sigma'(z^l_2) \\ \dots \\ \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k\, \sigma'(z^l_n) \end{bmatrix} = \begin{bmatrix} \sigma'(z^l_1) \sum_k w^{l+1}_{k1}\, \delta^{l+1}_k \\ \sigma'(z^l_2) \sum_k w^{l+1}_{k2}\, \delta^{l+1}_k \\ \dots \\ \sigma'(z^l_n) \sum_k w^{l+1}_{kn}\, \delta^{l+1}_k \end{bmatrix} \quad \therefore \quad \delta^l = \left((w^{l+1})^T \delta^{l+1}\right) \odot \sigma'(z^l) = \sum'(z^l)(w^{l+1})^T \delta^{l+1}$$

2.3 Combine the results from 2.1 and 2.2 to show $\delta^l = \sum'(z^l)(w^{l+1})^T \ldots \sum'(z^{L-1})(w^L)^T \sum'(z^L)\nabla_a C$

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \ldots \Sigma'(z^{L-1})(w^L)^T \Sigma'(z^L)\nabla_a C$$

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

$$\delta^L = \Sigma'(z^L)\nabla_a C \qquad (2)$$

derive

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \ldots \Sigma'(z^{L-1})(w^L)^T \Sigma'(z^L)\nabla_a C$$

by substituting eq. 1 back into δ^{l+1} until $l+n = L$

at which point you substitute in eq. 2 $\left(\delta^{l+1} = \Sigma'(z^{l+1})(w^{l+2})^T \delta^{l+2}\right)$

an example where $l = L-2$ is shown below

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \Sigma'(z^{l+1})(w^L)^T \Sigma'(z^L)\nabla_a C$$

a more expanded version of the equation would be

$$\delta^l = \Sigma'(z^l)(w^{l+1})^T \Sigma'(z^{l+1})(w^{l+2})^T \ldots \Sigma'(z^{L-1})(w^L)\Sigma'(z^L)\nabla_a C$$

2.4 Rewrite the four fundamental equations of backpropagation for a linear activation function

if $\sigma(z) = z$, $\sigma'(z) = 1$, $\Sigma'(z) = I$

Equations

1. $\delta^L = \Sigma'(z^L)\nabla_a C \longrightarrow \delta^L = I\nabla_a C = \nabla_a C$, $\delta_j^L = \frac{\partial C}{\partial a_j^L}$
2. $\delta^l = \Sigma'(z^l)\left((w^{l+1})^T \delta^{l+1}\right) \longrightarrow \delta^l = \left((w^{l+1})^T \delta^{l+1}\right)$, $\delta_i^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1}$
3. $\frac{\partial C}{\partial b_j^l} = \delta_j^l \longrightarrow \frac{\partial C}{\partial b_j^l} = \delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1}$
4. $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \longrightarrow \frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l = a_k^{l-1} \sum w_{kj}^{l+1} \delta_k^{l+1}$

3 Cross-Entropy

3.1 Switching y and a in cross-entropy: what happens at y=0 and y=1 for the incorrect formula?

Incorrect Formula

$-[a \ln y + (1-a)\ln(1-y)]$

eg 2: $y=0$: $-[a \ln 0 + (1-a)\ln(1\overset{\neq 1}{-0})] = -[\text{undef.} + (1-a)]$

$y=1$: $-[a \ln 1 + (1-a)\ln(1\overset{\neq 0}{-1})] = -[a + \text{undef.}]$

$\ln 0$ is undefined, the formula breaks

Correct Formula

$-[y \ln a + (1-y)\ln(1-a)]$

eg 1: $y=0$: $-[0 \ln a + (1-0)\ln(1-a)] = -[\ln(1-a)]$

$y=1$: $-[1 \ln a + (1-1)\ln(1-a)] = -[\ln(a)]$

the correct formula does not have the same issue because y is not in the $\ln$ term

3.2 Deriving BCELoss using $L(\theta) = -E_{x,y\hat{P}_{data}} log(P_{model}(y|x,\theta)$

Bernoulli distribution: $P_r(X=1) = p = 1 - P_r(x=0) = 1-q$ $\therefore p+q=1$

$f(k;p) = p^k(1-p)^{1-k}$ for $k \in \{0,1\}$ or $f(k;p) = pk + (1-p)(1-k)$ for $k \in \{0,1\}$, $\log(p^k(1-p)^{1-k}) = k\log p + (1-k)\log(1-p)$

$a^L = \hat{y}(x,\theta)$ → output depends on both x and θ , $\hat{y} = P(y=1|x,\theta)$

$L(\theta) = -E_{x,y\sim\hat{p}_{data}} \log(P_{model}(y|x,\theta))$

BCE Loss $= -\frac{1}{N}\sum_{i=1}^{N} y_i \log(P(y_i)) + (1-y)\log(1-P(y_i)) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log a_i + (1-y_i)\log(1-a_i)$

1. Using the Bernoulli distribution formula $f(k|p) = p^k(1-p)^{1-k}$ and substituting into $P_{model}(y|x,\theta)$ and $\hat{y} = a = f(x,\theta)$

$f(k|P) = P_{model}(y|x,\theta) = \hat{y}^y(1-\hat{y})^{1-y}$

$\log(P_{model}(y|x,\theta)) = \log(\hat{y}^y(1-\hat{y})^{1-y}) = y\log(\hat{y}) + (1-y)\log(1-\hat{y}) = y\log(a) + (1-y)\log(1-a)$

2. Account for $-E_{x,y\sim\hat{p}_{data}}$, $-E_{x,y\sim\hat{p}_{data}}$ is the expectation where $E(y) = \frac{1}{N}\sum_{i=1}^{N} y_i$

$-E(\log(P_{model}(y|x,\theta))) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(a_i) + (1-y_i)\log(1-a_i)$

3. Compare to Binary Cross Entropy Loss formula:

$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(a_i) + (1-y_i)\log(1-a_i)$

BCE Loss $= -\frac{1}{N}\sum_{i=1}^{N} y_i \log(a_i) + (1-y_i)\log(1-a_i)$

$L(\theta)$ = BCE Loss

3.3 Deriving multi-class cross entropy with the same method

Multinoulli Distribution: $f(x|P) = \prod_{i=1}^{K} P_i^{[x=i]}$, $[x=i] = 1$ if $x=i$, else $[x=i]=0$

$\log(f(x|P)) = \log(P_1^{[x=1]}) + \log(P_2^{[x=2]}) + \ldots + \log(P_k^{[x=k]}) = \sum_{i=1}^{K} \log(P_i^{[x=k]})$

1. Substitute for $y, \hat{y}$: $\log(P(y|\hat{y})) = \sum_{j=1}^{K} y_j \log(\hat{y}_j)$

2. Substitute back into eq. 1: $L(\theta) = -E_{x,y\sim\hat{p}_{data}} \log(P_{model}(y|x,\theta)) = -\sum_{i=1}^{N}\sum_{j=1}^{K}(y_j \log(\hat{y}_j))_i$

$$\text{CE Loss} = -\sum_{i=1}^{N}\sum_{j=1}^{K}(y_j \log(\hat{y}_j))_i$$

3.4 Show that CELoss is minimized when $\sigma(z) = y$

using $C = -\frac{1}{N}\sum_x [y \ln y + (1-y)\ln(1-y)]$

testing a single value with $y_i = 1$: $C_i = 1 \cdot \ln 1 + (1-1)\ln(1-1) = 1 \cdot 0 + 0 \ln 0 = 0$

$y_i = 0$: $C_i = 0 \cdot \ln 0 + 1 \cdot \ln 1 = 0 \ln 0 + 1 \cdot 0 = 0$

- avoids the issue shown in problem 4.1

Check derivatives: $\frac{\partial C}{\partial w_j} = -\frac{1}{N}\sum_x x_j(\sigma(z) - y) = -\frac{1}{N}\sum_x x_j(y-y) = 0$

$\frac{\partial C}{\partial b} = -\frac{1}{N}\sum_x (\sigma(z) - y) = -\frac{1}{N}\sum_x (y-y) = 0$

3.5 Where does the softmax name come from?

The softmax function is a softened form of the max function. This is important because it transforms it into a differentiable function that can be used in backpropagation. The outputs of the softmax function form a probability distribution because as one output increases, all others decrease. The sum of all these outputs is always 1.

3.6 Show that $\delta_j^L = a_j^L - y_j$

$\delta_j^L = \frac{\partial C}{\partial z_j^L} = a_j^L - y_j$

$\frac{\partial C}{\partial z_j^L} = \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial z} = \frac{\sigma'(z)}{\sigma(z)(1-\sigma(z))}(\sigma(z) - y)$, $\sigma'(z) = \sigma(z)(1-\sigma(z))$

$\frac{\partial C}{\partial z_j^L} = \sigma(z_j^L) - y_j$, set $\sigma(z_j^L) = a_j^L$

$\frac{\partial C}{\partial z_j^L} = a_j^L - y_j$

4 Regularized Logistic Regression on MNIST

4.1 Gradient of the loss function

$$\ell(w, w_0) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+e^{-z_i}}\right) + (1-y_i)\log\left(\frac{e^{-z_i}}{1+e^{-z_i}}\right)\right] + \lambda\|w\|^2$$

convert to vector form

$$\ell(w, w_0) = -\frac{1}{n}\left(y \log\left(\frac{1}{1+e^{-\tilde{w}^T x}}\right) + (1-y)\log\left(\frac{e^{-\tilde{w}^T x}}{1+e^{-\tilde{w}^T x}}\right)\right) + \lambda\|w\|^2$$

substitute: $\sigma(z) = \frac{1}{1+e^{-z}}$, $z = \tilde{w}^T x$

$$\ell(w, w_0) = -\frac{1}{n}\left(y \log(\sigma(z)) + (1-y)\left(\log(e^{-z}) + \log(\sigma(z))\right)\right) + \lambda\, w^T w$$

take partial derivative wrt $\tilde{w}$

$$\frac{\partial \ell}{\partial \tilde{w}} = -\frac{1}{n}\left(y \frac{1}{\sigma(z)}\frac{\partial \sigma(z)}{\partial \tilde{w}} + (1-y)\left(\frac{1}{e^{-z}}\frac{\partial e^{-z}}{\partial \tilde{w}} + \frac{1}{\sigma(z)}\frac{\partial \sigma(z)}{\partial \tilde{w}}\right)\right) + 2\lambda w$$

$$\frac{\partial \ell}{\partial \tilde{w}} = \frac{1}{n}\left(y \frac{1}{\sigma(z)}\sigma'(z)x + (1-y)\left(\frac{1}{e^{-z}}(-e^{-z}x) + \frac{1}{\sigma(z)}\sigma' x\right)\right) + 2\lambda w$$

use $\sigma'(z) = \sigma(z)(1-\sigma(z))$

$$\frac{\partial \ell}{\partial \tilde{w}} = \frac{1}{n}\left(y \frac{1}{\sigma(z)}\sigma(z)(1-\sigma(z))x + (1-y)\left(-x + \frac{1}{\sigma(z)}\sigma(z)(1-\sigma(z))x\right)\right) + 2\lambda w$$

$$\frac{\partial \ell}{\partial \tilde{w}} = \frac{1}{n}\left(y(1-\sigma(z))x + (1-y)(-x + x - \sigma(z)x)\right) + 2\lambda w$$

$$\frac{\partial \ell}{\partial \tilde{w}} = \frac{1}{n}\left(y(1-\sigma(z))x + (1-y)(-\sigma(z)x)\right) + 2\lambda w$$

4.2 Implementing and Fitting a Logistic Regression Model

To find the best parameters for this model, I performed a grid sweep with a variety of values for lambda, learning rate, and threshold. After seeing that 0.5 performed best as the threshold value, I set that as a constant. In my code I left the grid sweep for lambda and learning rate, and I save the best performing model as a .ckpt file and then reread the parameters from this model for use in the later sections.

For lambda=0.01 and lr values of 0.1, 0.5, and 1 I get the model correctly predicting 972, 972, and 974 of 1000 test images. For lambda=1 and the same lr values, I get 970, 972, and 884 correct test images out of 1000. Most parameters for the model perform very similarly, with lambda=1 and lr=1 being the worst performer. My best performing models got 974 correct and this happened with both lambda=0.01/lr=1 and lambda=0.5/lr=1. My code chooses to use

lambda=0.01/lr=1 as the best performing model.

I trained these models for 1000 epochs and used torch.optim.SGD() to initialize the weights and biases. I use a single layer with a sigmoid activation function to get predictions. I use nn.BCELoss() as the criterion for error calculation, with an added term to account for regularization done in each epoch. Adding the regularization offset greatly improved the performance of the model. At the final step on the best performing model, the loss function value is 0.04337800666689873.

4.3 Most confident misclassified images

Knowing that the confidence of the model's prediction is based on how close to 1 or 0 the prediction is, I sorted the misclassified data points by confidence. I then calculated the confidence value displayed on the chart by taking the difference of the prediction value with 0.5 (the threshold value) and then added 0.5 and converted to a percentage. This means that all of these images will have a prediction confidence of 50% or higher. This makes sense because if the model is less than 50% confident that an image is a 9, it will be over 50% confident that the image is a 4 and would therefore be classified as a 4.

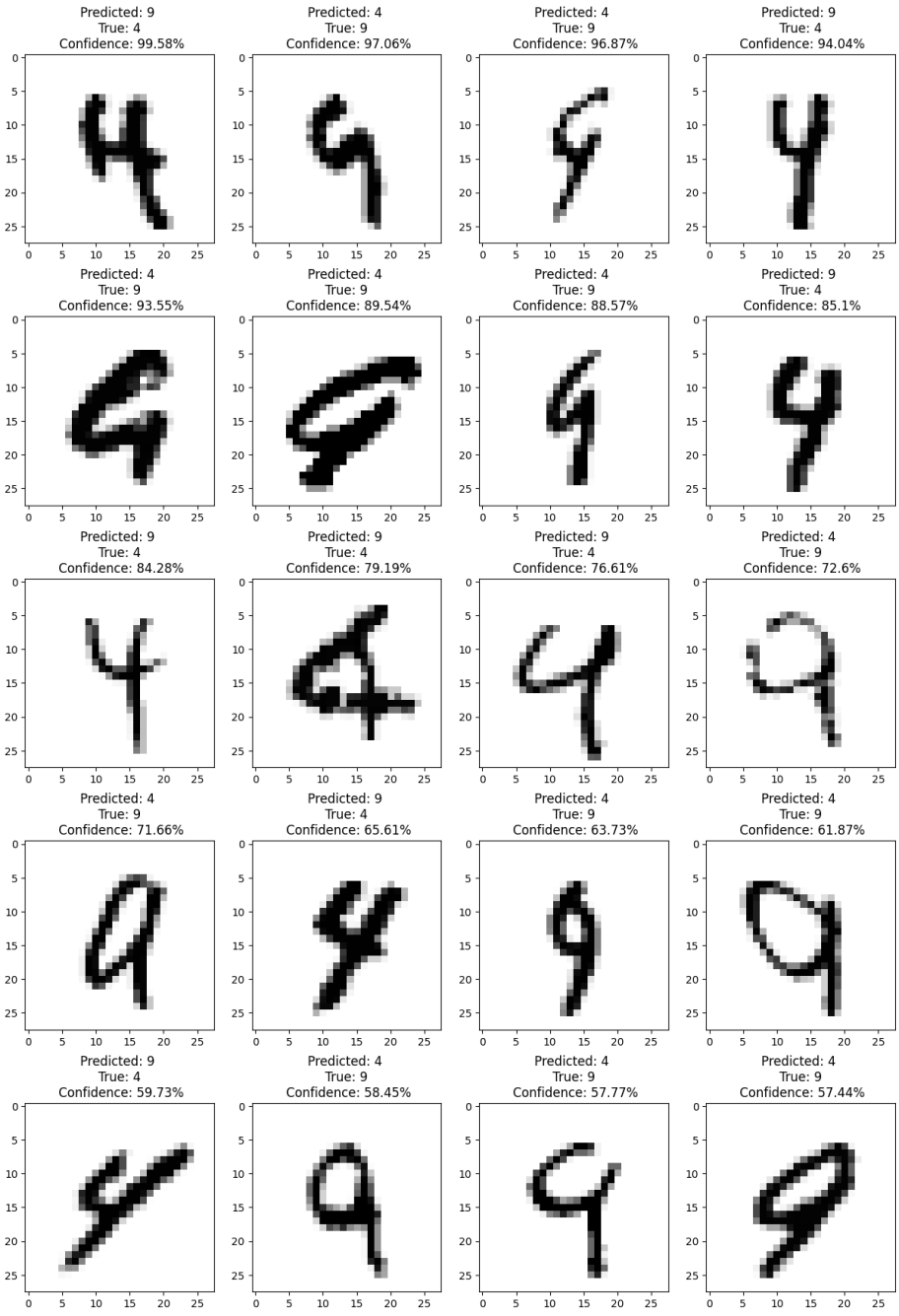

Figure 1: 4x5 Grid of the most confidently misclassified images