CSC311 Assignment 1: Proofs and Explanation

Name: Kannika Kabilar

Student#: 1004019199

Utorid: kabilark

Question 2

Question 2 (c):

Execution time of matrix_poly for a 100x100 matrix is: 1.6911461353302002

Execution time of vectorized code for a 100x100 matrix is: 0.004945993423461914

Magnitude of the difference matrix for a 100x100 matrix is: 1.4535030042104203e-11

Execution time of matrix_poly for a 300x300 matrix is: 46.78812623023987

Execution time of vectorized code for a 300x300 matrix is: 0.0039975643157958984

Magnitude of the difference matrix for a 300x300 matrix is: 1.7369472572107675e-09

Execution time of matrix_poly for a 1000x1000 matrix is: 1727.2512502670288

Execution time of vectorized code for a 1000x1000 matrix is: 0.08295273780822754

Magnitude of the difference matrix for a 1000x1000 matrix is: 8.023592818062752e-05

Above is the printed output, when **timing(100)**, **timing(300)**, and **timing(1000)** are executed for reference. We know that based on our implementation, matrix multiplication is performed twice, so we will multiply the total number of floating-point multiplications for a matrix multiplication by 2. First, we will calculate the amount of floating-point multiplication performed for each element. This will be N, because each element in the row of the of the first matrix are multiplied with each of the element in the column of the second matrix. Now, each of the element in the new matrix are calculated from N floating point multiplications, and there are NxN elements in the new matrix, so from each matrix multiplication, N^3 floating point multiplications are performed, so **matrix_poly(A)** performs $\mathbf{2(N^3)}$ floating point multiplications.

Therefore,

timing(100) performs 2 000 000 total floating point multiplications

timing(300) performs 54 000 000 total floating point multiplications

timing(1000) performs 2 000 000 000 total floating point multiplications

Our implementation of **matrix_poly(A)** consists of three nested for-loops for matrix multiplication, each iterating N times (with a total of N^3 times) and two nested for-loops for matrix addition, each iteration N time (with a total of N^2 times) but smaller polynomial terms are ignored, so this results with a big-O time of $O(N^3)$. This can be observed from our execution, where timing(100) executes matrix_poly in about 1.69 seconds and when N is increased by a factor of 3, timing(300) executes matrix_poly in about 46.78 seconds which is equivalent to about $3^3 = 27$ so, $27*1.69 = 45.63 \approx 46.78$ and similarly for timing(1000), $10^3 = 1000$ so, $1000*1.69 = 1690 \approx 1727$

Question 4

Question 4(e):

We will start by proving the hint:

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{N} \left(y^{(i)} - t^{(i)}\right) x_j^{(i)} \left(\frac{1}{N}\right) \Leftrightarrow \left[\frac{\partial J}{\partial w}\right]_j = \left[X^T (y-t)\left(\frac{1}{N}\right)\right]_j$$

By definition of Gradient

$$\frac{\partial J}{\partial w_j} = \left[\frac{\partial J}{\partial w}\right]_j$$

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \sum_{i=1}^{N} \left([y]_i - [t]_i\right) x_j^{(i)} \left(\frac{1}{N}\right)$$

Both represent the ith element of their vector

$y^{(i)} = [y]_i$ and $t^{(i)} = [t]_i$

and similary, $x_j^{(i)} = x_{ij}$

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \sum_{i=1}^{N} \left([y-t]_i\right) x_j^{(i)} \left(\frac{1}{N}\right)$$

By definition of y and t and since they are same length vector so, $[y]_i - [t]_i = [y-t]_i$

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \sum_{i=1}^{N} \left([y-t]_i\right) x_{ij} \left(\frac{1}{N}\right)$$

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \sum_{i=1}^{N} \left([y-t]_i\right) \left[X^T\right]_{ji} \left(\frac{1}{N}\right)$$

By definition of matrix transpose, $[X^T]_{ji} = x_{ij}$

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \sum_{i=1}^{N} \left[X^T\right]_{ji} \left([y-t]_i\right) \left(\frac{1}{N}\right)$$

Rearrange terms

$$\Leftrightarrow \quad \left[\frac{\partial J}{\partial w}\right]_j = \left[X^T (y-t)\right]_j \left(\frac{1}{N}\right)$$

By definition of matrix multiplication

We have used equation 5 to prove that equation 7 also holds true using if and only if.

But since we know that all elements are equivalent to their corresponding terms in their index, we can conclude that

$$\left[\frac{\partial J}{\partial w}\right]_j = \left[X^T(y-t)\right]_j\left(\frac{1}{N}\right) \quad \Leftrightarrow \quad \frac{\partial J}{\partial w} = X^T(y-t)\left(\frac{1}{N}\right)$$

Therefore, we have proven that equation 5 is true if and only if equation 6 is true.

Question 4(f):

We know the definition of the logistic cross entropy is $\mathcal{L}_{\mathrm{CE}} = -t \log y - (1-t)\log(1-y)$

And the definition of the logistic function is $\sigma(z) = \frac{1}{1+e^{-z}}$

$$\mathcal{L}_{\mathrm{LCE}}(z,t) = \mathcal{L}_{\mathrm{CE}}(\sigma(z),t)$$

Applying the definition of logistic cross entropy and the logistic function

$$= (-t)log\left(\frac{1}{1+e^{-z}}\right) - (1-t)log\left(1-\left(\frac{1}{1+e^{-z}}\right)\right)$$

By definition $(1/a) = a^{-1}$

$$= (-t)log\left(\left(1+e^{-z}\right)^{-1}\right) - (1-t)log\left(1-\frac{1}{1+e^{-z}}\right)$$

By definition, $\log(x^a)=a\log(x)$

$$= (-t)(-1)log\left(1+e^{-z}\right) - (1-t)log\left(\frac{\left(1+e^{-z}\right)}{\left(1+e^{-z}\right)} - \frac{1}{1+e^{-z}}\right)$$

$$= (t)log\left(1+e^{-z}\right) - (1-t)log\left(\frac{\left(e^{-z}\right)}{\left(1+e^{-z}\right)}\right)$$

Common denominators, allow us to simplify terms in numerators

$$= (t)log\left(1+e^{-z}\right) - (1-t)log\left(\left(e^{z}\right)^{-1}\left(1+e^{-z}\right)^{-1}\right)$$

By the exponential identity $a^x b^x = (ab)^x$

$$= (t)log\left(1+e^{-z}\right) - (1-t)log\left(\left(\left(e^{z}\right)\left(1+e^{-z}\right)\right)^{-1}\right)$$

$$= (t)log\left(1 + e^{-z}\right) - (1-t)(-1)log\left(\left(e^{z}\right)\left(1 + e^{-z}\right)\right)$$

Distribute terms

$$= (t)log\left(1 + e^{-z}\right) + (1-t)log\left(e^{z} + \left(\frac{e^{z}}{e^{z}}\right)\right)$$

$$= (t)log\left(1 + e^{-z}\right) + (1-t)log\left(e^{z} + 1\right)$$

$$= (t)log\left(1 + e^{-z}\right) + (1-t)log\left(1 + e^{z}\right)$$

Therefore, we have proven the equation for the logistic cross entropy.

Question 6

Question 6(e):

A possible reasons that the validation accuracy is higher is because there was a larger amount of training data, validation data and test data for the numbers ‘4’ and ‘7’ while the numbers ‘5’ and ‘6’ had fewer training, validation and test data. Another possible reasoning why the validation accuracy is significantly lower than the training accuracy is because of overfitting in part c) and underfitting in part d).

Question 6(f):

We only consider odd values for K in part c) because we’re performing a binary classification so an odd value of K prevents ties where both labels achieve the same score. Since an odd number of points cannot be evenly divided among the two classifications, one of the labels will contain a clear lead over the other.

Question 6(g):

The KNN produces high accuracies on the MINST data because the MINST data contains significantly large amounts of data for KNN to train. Furthermore, each data characteristic has a weight, this helps KNN realize correct decisions and since KNN is used in a binary classification, it is able to have a higher correct prediction rate. Additionally, the MINST data only contains numbers written on a plain white background, which makes it easier for KNN to classify it correctly resulting in high accuracies.