CS 615 - Deep Learning

Assignment 1 - Artificial Neurons

John Obuch

Q1: Theory

1. Given a class probability distribution of {0.1, 0.8, 0.1}, what is the cross entropy if the correct class is the first class? (3pts)

   Let the distribution defined above be assumed to be obtained from taking the softmax. Recall that the softmax is defined as follows:

$$P(y = j) = \hat{y}_j = \frac{e^{x\theta_{:,j}}}{\sum_{k=1}^{K} e^{x\theta_{:,k}}}$$

   Moving forward under this assumption, we can compute the cross-entropy if the true class is the first class. To do this, we will leverage the following equation:

$$H = H(a, b) = \sum_{k=1}^{K} -a_k \ln(b_k)$$

   Since we are proceeding under the assumption that the class probability distribution provided are outputs from the softmax activation function, we can compute the cross-entropy loss objective function which we will write as follows:

$$J = J(y, \hat{y}) = -\sum_{k=1}^{K} y_k \ln(\hat{y_k})$$

   Here, we will define our y distribution to be one-hot encoded assigning the the value 1 to that of the first class since we are operating under the assumption that the correct class is the first class. Given this, the equation above reduces to the following:

$$-\ln(\hat{y_a})$$

   Thus, we will show this processes by proceeding as follows:

   Let $y = \{1, 0, 0\}$ and let $\hat{y} = \{0.1, 0.8, 0.1\}$. Following the equations above we compute the following:

$$J = J(y, \hat{y}) = -(1\ln(0.1) + 0\ln(0.8) + 0\ln(0.1)) = -\ln(0.1) = 2.3026$$

Therefore, the cross-entropy log loss assuming the correct class is the first class from the probability distribution provided above is $J = 2.3026$.

■

2. If we're using cross-entropy as our objective function, are we attempting to minimize it or maximize it (2pts)?

   We desire to minimize the cross-entropy logarithmic loss to ensure a higher predicted probability. The value of the cross-entropy determines how "far away" from the true value we are. Furthermore, we want to minimize the randomness in our system. Additionally, we can actually minimize or maximize the cross-entropy loss. To explain this further, when we seek to optimize an objective function $f(x)$, we can either minimize $f(x)$ or maximize $-f(x)$. in our case, we elect to minimize the cross-entropy log loss.

■

3. Given the confusion table in Figure 1:

		True Class			
		1	2	3	4
Predicted Class	1	5	2	3	4
	2	8	12	30	4
	3	0	8	45	4
	4	10	0	5	80

Figure 1: Confusion Matrix

Figure 1: Confusion Table

- What are the class priors? (3pts)

  To compute the class priors, we will leverage the column vectors of the *True Class* values in the confusion matrix provided above. We will notate the class priors as p_{c_i} for $i = 1, \ldots, 4$ where $p_{c_i} = \frac{\sum \vec{x_{:,i}}}{N}$. Note that here, we notate $\vec{x_{:,i}}$ as the column vector associated to class i (i.e., we sum the elements of the column vector associated with class i). Additionally, we observe the value of $N = 220$ to be the total number of records in the confusion matrix. What follows are the associated class priors:

$$p_{c_1} = \frac{5+8+0+10}{220} = \frac{23}{220} = 0.10\overline{45}$$
$$p_{c_2} = \frac{2+12+8+0}{220} = \frac{22}{220} = 0.10$$
$$p_{c_3} = \frac{3+30+45+5}{220} = \frac{83}{220} = 0.37\overline{27}$$
$$p_{c_3} = \frac{4+4+4+80}{220} = \frac{92}{220} = 0.4\overline{181}$$

Check to confirm the probabilities sum to 1:

$$p_{c_1} + p_{c_2} + p_{c_3} + p_{c_4} = 0.10\overline{45} + 0.10 + 0.37\overline{27} + 0.4\overline{181} = 1$$

■

- What is the overall accuracy of the system? (2pts)

  To compute the overall accuracy of the system, we will take the trace of the matrix. Recall that the accuracy of a multi-classification model is computed as follows:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}(Y_i == \hat{Y_i})$$

  To make this easier on ourselves, we will convert the above equation into matrix notation. Let $C = Y^T\hat{Y}$ be a (k x k) square confusion matrix of the system and let $\vec{e}$ be a (k x 1) column vector with entries of all ones. It is important to observe that the cases where $Y_i == \hat{Y_i}$ fall along the main diagonal of the confusion matrix C. Finally, note that the computation $\vec{e}^T C\vec{e}$ results in a scalar s which can be thought of as the sum of all elements in the matrix C. From here, we can compute the accuracy of the system as follows:

$$\text{Acc} = \frac{\text{Tr}(C)}{\vec{e}^T C\vec{e}} = \frac{142}{220} = 0.64\overline{54}$$

■

Q2: Visualizing Gradient Descent

What follows are the resulting plots after convergence criteria for θ is met with respect to the objective function $J = (\theta_1 + \theta_2 - 2)^2$. Computing the partial derivatives with respect to θ_1 and θ_2, we yield the following:

$$\frac{\partial J}{\partial \theta_1} = 2(\theta_1 + \theta_2 - 2) = 2\theta_1 + 2\theta_2 - 4$$

$$\frac{\partial J}{\partial \theta_2} = 2(\theta_1 + \theta_2 - 2) = 2\theta_1 + 2\theta_2 - 4$$

Thus, we see that $\frac{\partial J}{\partial \theta_1} = \frac{\partial J}{\partial \theta_2}$. Initializing $\theta = \vec{0}$ and performing gradient decent, we achieve the following 3D convergence visualization with respect to the coordinates $(x, y, z) = (\theta_1, \theta_2, J)$:

Python:

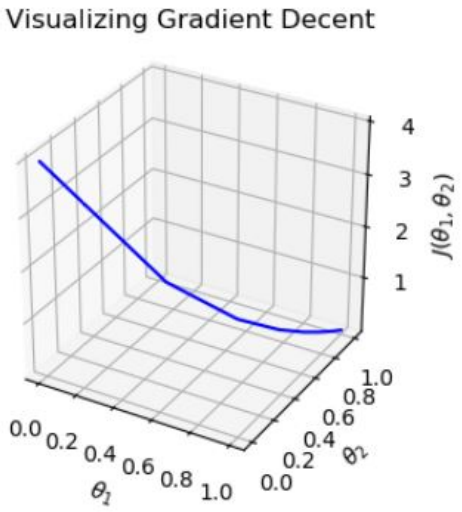

Figure 2: Gradient Decent

MATLAB:

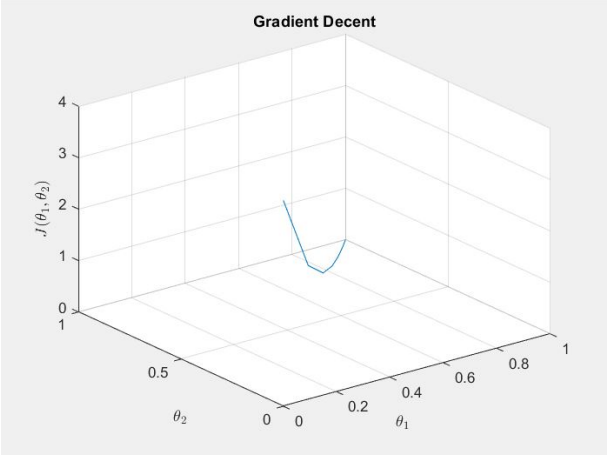

Figure 3: Gradient Decent

See source code for more details.

■

Q3: Gradient Descent Logistic Regression

Chosen Hyper Parameters:

1. Initial Parameters: θ where the column vectors of θ were initialized with uniformly random values between $[-1, 1]$ resulting in an $(N,\ K)$ matrix where N is the number of rows with respect to the training data set, and K is the number of class labels.

2. Learning Rate: $\eta = 0.01$

3. L2 Regularization amount: $\lambda = 5$ where $0 \leq \lambda \leq \infty$ is our blending factor. Note that $L2 = \frac{1}{2}\theta^T\theta$ and we regularize our cost function by performing the operation $\lambda L2$ and subtracting it from the logarithmic cost function. Similarly, we penalize the gradient by subtracting off $\lambda\theta$.

4. Change in Cost Threshold: log_thresh $= 2e - 23$

5. Iteration Threshold: iter_thresh $= 5000$

6. Random seed 42

   Note: Training occurs until convergence criteria is met. Specifically, the algorithm will converge/terminate when the change in the log cost is less than or equal to log_thresh, and the number of iterations is greater than or equal to the iter_thresh (i.e., While (chg_in_log $>$ log_thresh) (iter $<$ iter_thresh), train the data).

Testing Accuracy:

Tux: Acc $= 0.92857$

What follows are the resulting convergence graphs and confusion matrix:

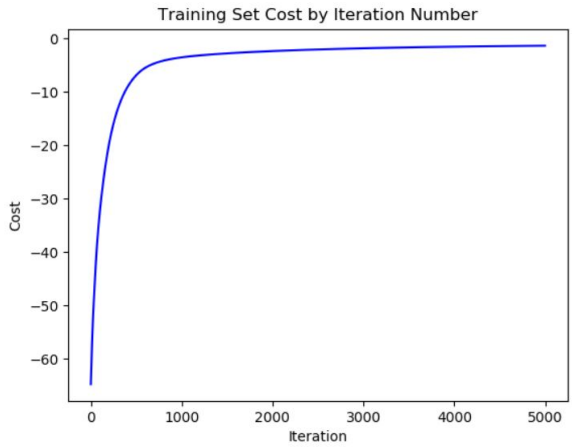

Figure 4: Average log Likelihood VS Iteration Number (Train)

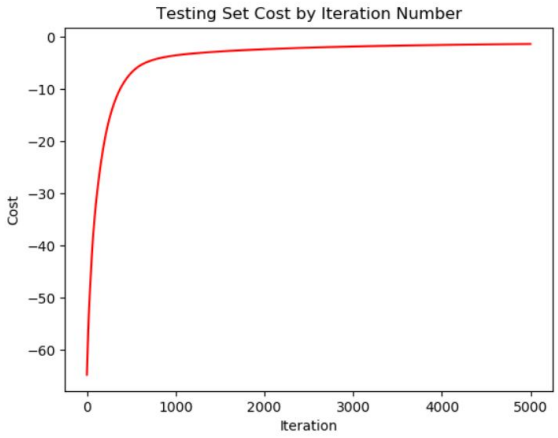

Figure 5: Average log Likelihood VS Iteration Number (Test)

```
TEST: CONFUSION MATRIX

[[3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 2.]]
```

Figure 6: Confusion Matrix (Test) - Columns and Rows represent True Class and Predicted Class respectively.

See source code for more details.

■

Q4: Gradient Descent w/ Softmax and Cross-Entropy

Chosen Hyper Parameters:

1. Initial Parameters: θ where the column vectors of θ were initialized with uniformly random values between $[-1, 1]$ resulting in an $(N,\ K)$ matrix where N is the number of rows with respect to the training data set, and K is the number of class labels.

2. Learning Rate: $\eta = 0.1$

3. L2 Regularization amount: $\lambda = 5$ where $0 \leq \lambda \leq \infty$ is our blending factor. Note that $L2 = \frac{1}{2}\theta^T\theta$ and we regularize our cost function by performing the operation $\lambda L2$ and adding the logarithmic cost function. Similarly, we penalize the gradient by adding on $\lambda\theta$.

4. Change in Cost Threshold: log_thresh $= 2e - 23$

5. Iteration Threshold: iter_thresh $= 50,000$

6. Random seed 42

   Note: Training occurs until convergence criteria is met. Specifically, the algorithm will converge/terminate when the change in the average cross entropy cost is less than or equal to log_thresh, and the number of iterations is greater than or equal to the iter_thresh (i.e., While (chg_in_cross_entropy > log_thresh) (iter < iter_thresh), train the data).

Testing Accuracy:

Tux: Acc $= 0.9047619$

What follows are the resulting convergence graphs and confusion matrix:

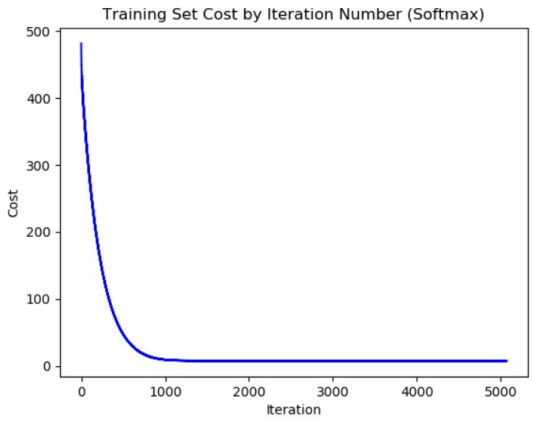

Figure 7: Average Cross Entropy VS Iteration Number (Train)

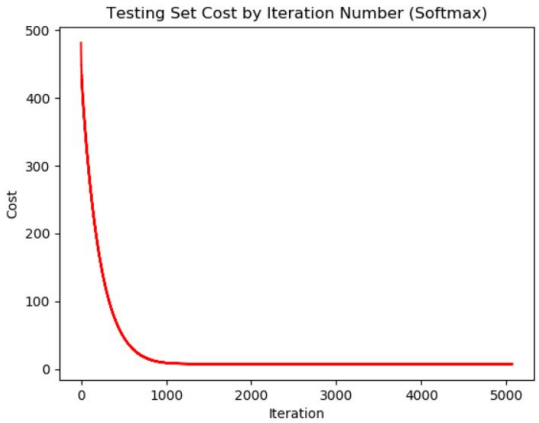

Figure 8: Average Cross Entropy VS Iteration Number (Test)

```
TEST: CONFUSION MATRIX

[[3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 2. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 2.]]
```

Figure 9: Confusion Matrix (Test) - Columns and Rows represent True Class and Predicted Class respectively.

See source code for more details.

■