CS 559: Machine Learning – Fundamentals & Applications

Assignment 3 Due: 3/4/2025 Tuesday 11:59 p.m.

- The assignment must be individual work and must not be copied or shared. Any tendency to cheat/copy evidence will lead to a 0 mark for the assignment.
- Students must only use Pandas, NumPy, Matplotlib, and Scipy if the problem does not specify libraries/packages.
- All codes will be tested in grading. Any codes with an error will be marked 0. Make sure to restart the kernel and run it all before submission. Delete any codes that should not be graded.
- Results must be displayed.
- All problems must be submitted in a single notebook file. Do not use a text editor to write codes.

1 Neural Networks [60 pts]

In this problem, you will implement the forward and back propagation algorithms in Python. All parts of this problem assume the activation function is logistic sigmoid, so you may hard code that aspect of the algorithms, but otherwise your code should handle any number of inputs, outputs, and any sizes of hidden layers.

a. [20 pts] Implement forward propagation as a function:

```
import numpy as np

# x: inputs
# y: outputs
# ws: weights
# bs: biases
# zs: the z values
# E: the total error
def forward_prop(x, y, ws, bs):
    ...
    return zs, E
```

Note that the function takes weights and biases as parameters rather than setting them randomly. This allows us to test your code on known values. For instance, running your code on the example from the lecture should produce results similar to the following.

```
x = np.array([0.7, 0.3, 0.5]).reshape(3, 1)
y = np.array([1]).reshape(1, 1)
weights = [np.array([[-1.0,  -2.3 ,  1.7],
    [-0.8 ,  0.3 , 1.4]]), np.array([[-2.0, -0.4 ]])]
biases = [np.array([[ 1.6], [-0.6]]), np.array([[-0.5]])]

zs, E = forward_prop(x, y, weights, biases)

print(f'zs = {zs}')
print(f'E = {E}')

# Output:
zs = [array([[0.7],
    [0.3],
    [0.5]]), array([[0.74269055],
    [0.40854102]]), array([[0.10444365]])]
E = 0.4010105918229552
```

b. [20 pts] Implement back propagation as a function. It must return the updated weights and biases.

```
def back_prop(y, zs, ws, bs, eta):
    ...
    return ws, bs
```

Note that the function requires as one of its inputs the z values that were returned from `forward_prop`. Running the function using the lecture example should produce results similar to the following.

```
eta = 3.0

weights, biases = back_prop(y, zs, weights, biases, eta)

print(f'weights = {weights}')
print(f'biases = {biases}')

# Output:
weights = [array([[-1.06095869, -2.32612515,  1.65645808],
    [-0.81263842,  0.29458354,  1.39097256]]), array([[-1.81336331, -0.29733444]])]
biases = [array([[ 1.51291616],
    [-0.61805488]]), array([[-0.24870195]])]
```

c. [20 pts] Run your code on the following example and report the output.

```
x = np.array([5.2, -2.3, -1.7, 8.3]).reshape(4, 1)
y = np.array([0, 1]).reshape(2, 1)
eta = 5.0
weights = [np.array([
    [0.09, -0.68 , -0.38, 0.93],
    [-0.37, 0.61, 0.45, 0.57],
    [-0.29, -0.76, 0.46, 0.01]]),
    np.array([[0.20, -0.20, 0.83],
    [0.35, -0.59, 0.36]])]
biases = [np.array([[-0.53], [0.84], [-0.34]]), np.array([[-0.31], [0.36]])]

zs, E = forward_prop(x, y, weights, biases)
print(f'zs = {zs}')
print(f'E = {E}')

weights, biases = back_prop(y, zs, weights, biases, eta)
print(f'weights = {weights}')
print(f'biases = {biases}')
```

2 Training Data Process [40 pts]

In this assignment, students will practice preparing the training data using the data **Hitters.csv**. There are no specific directions except the following.

a. Split the data into training and test sets by 8 to 2 using `sklearn.model_selection`.
b. [15 pts] Perform EDA using the training data and explain the findings with visualizations.
c. [15 pts] Perform feature extraction, engineering, and selection using the training data.
d. [10 pts] Repeat (c) on the test set so it has the same structure as the training set.
e. Save the data sets. They will be used in the next assignment.