CS 559 - Fall 2021
Homework #2

Krishna Garg
kgarg8@uic.edu

1 Results & Discussion

Reference Logs for N = 100:

**********N = 100**********

Optimal weights: [-0.06247144] [0.28060924] [0.90003316]

Initial weights: [-0.4059218] [-0.85026442] [0.05252539]

*******Training for eta = 1*******

Final weights: [-2.4059218] [12.52376611] [38.1803964]

Epochs taken: 6

*******Training for eta = 10*******

Final weights: [-20.4059218] [132.8900409] [381.33123541]

Epochs taken: 6

*******Training for eta = 0.1*******

Final weights: [-0.2059218] [1.09033421] [3.13865467]

Epochs taken: 14

(h)(vii) How do the final weights compare to the "optimal" weights?

Ans. We can see that the final weights after perceptron training get the same sign as the optimal weights even though they may be initialized with a different sign.

(l) Comment on how the changes in η affect the number of epochs needed until convergence?

Ans. From figure 2, we can observe that for N=100, the training converges the faster for higher values of η i.e. $\eta = 1, 10$ in 6 epochs whereas the it takes 14 epochs for $\eta = 0.1$ because that takes smaller update steps.

For N=1000 in figure 4, the misclassifications vs. epochs is almost similar. All η values training converges in 7-8 epochs. η=10 is the slowest.

Overall, we can observe that too large or too small η - both can end up taking a lot of time to converge.

(m) Comment on whether we would get the exact same results (in terms of the effects of η on training performance) if we had started with different weights, S?

Ans. For seed=110 & N=100, training converge takes 10, 10, 14 epochs.

Whereas for seed=112 & N=100, training converge takes 6, 6, 14 epochs.

So, we get different performance with different initializations of the weights.

Reference Logs for seed = 110:

**********seed = 110**********

**********N = 100**********

Optimal weights: [-0.19197057] [0.31691076] [-0.24882203]

Initial weights: [-0.59052897] [0.80012516] [-0.39797393]

*******Training for eta = 1*******

Final weights: [-27.59052897] [38.0039438] [-39.26596519]

Epochs taken: 10

*******Training for eta = 10*******

Final weights: [-270.59052897] [375.2754584] [-387.64665621]

Epochs taken: 10

*******Training for eta = 0.1*******

Final weights: [-2.19052897] [3.08330057] [-2.86037273]

Epochs taken: 14

(n) Do the same experiments with n = 1000 samples. Comment on the differences compared to n = 100.

Ans. Convergence for N = 1000 takes place in 7, 8, 7 epochs for η values = 1, 10, 0.1. The highest η value takes the maximum time to converge. This is in contrast to convergence for N = 100 where the same η values take 6, 6, 14 epochs i.e. the lowest η value takes the maximum time to converge.

Also, we can observe that the final weights have the same sign as the optimal weights. This is consistent with the observation for N = 100.

Also, the percentage misclassifications for the randomly initialized weights for N = 100, N = 1000 are 54%, 47.3% respectively.

Reference logs for N = 1000:

**********N = 1000**********

Optimal weights: [0.01650566] [0.29094265] [-0.39739492]

Initial weights: [-0.25057569] [-0.11014926] [-0.2647716]

*******Training for eta = 1*******

Final weights: [15.74942431] [270.42237947] [-368.27460577]

Epochs taken: 7

*******Training for eta = 10*******

Final weights: [159.74942431] [2703.30002186] [-3681.9660768]

Epochs taken: 8

*******Training for eta = 0.1*******

Final weights: [1.54942431] [27.06371949] [-36.97726246]

Epochs taken: 7

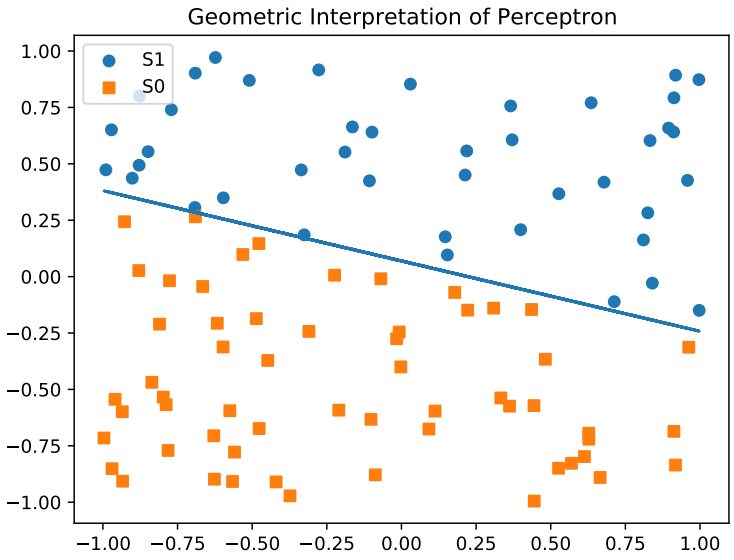
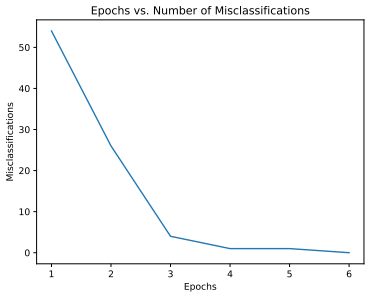
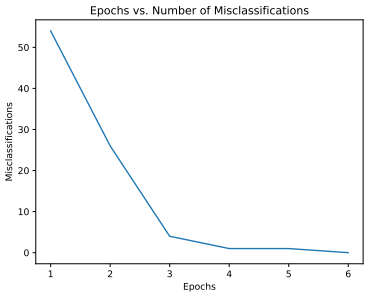


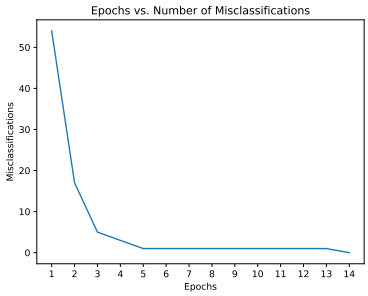
Figure 1: Geometric Interpretation of Perceptron for N = 100



(a) $\eta = 1$, N = 100



(b) $\eta = 10$, N = 100



(c) $\eta = 0.1$, N = 100

Figure 2: Graphs for Epochs vs. Number of Misclassifications for different η values, N = 100

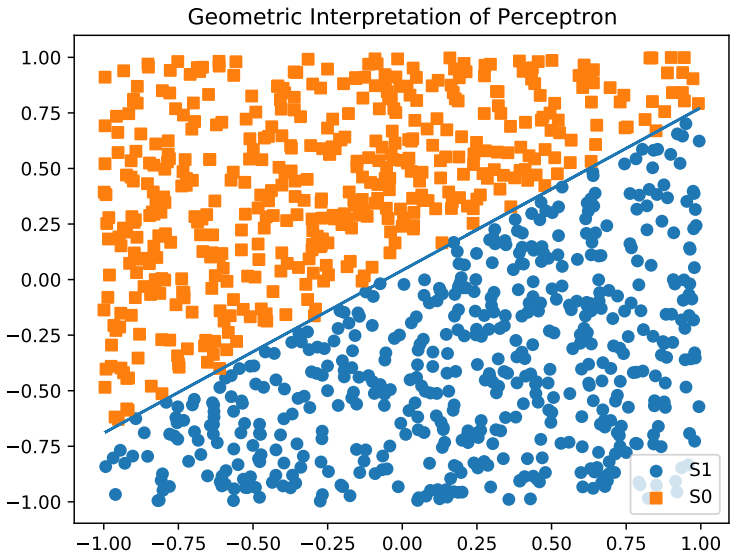
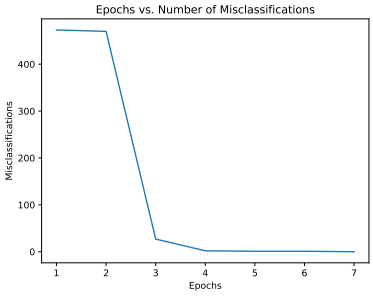
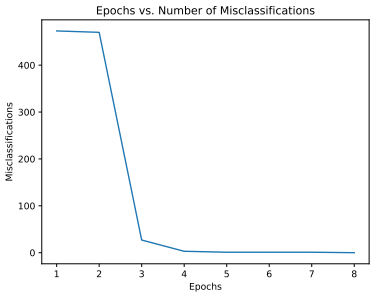


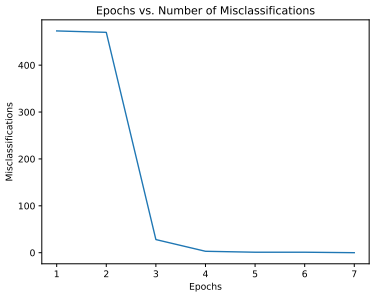
Figure 3: Geometric Interpretation of Perceptron for N = 1000



(a) $\eta = 1$, N = 1000



(b) $\eta = 10$, N = 1000



(c) $\eta = 0.1$, N = 1000

Figure 4: Graphs for Epochs vs. Number of Misclassifications for different η values, N = 1000

```
import numpy as np, matplotlib.pyplot as plt

seed = 112
np.random.seed(seed)
N_values = [100, 1000]
```

```
print('****************************seed =
    {}**************************'.format(seed))
for N in N_values:
   print('*********************N = {}*********************'.format(N))

   # (a) Geometric Interpretation of Perceptron

   # Consider following as Optimal Weights
   w0 = np.random.uniform(-0.25,0.25,1)
   w1 = np.random.uniform(-1,1,1)
   w2 = np.random.uniform(-1,1,1)

   # Create sets S (containing 100 vectors), S1 (x.w^T >= 0), S0 (x.W^T
       < 0)
   S, S1, S0, W = [], [], [], []
   for i in range(N):
      vec_x = np.random.uniform(-1, 1, 2)
      S.append(vec_x.tolist())
      w = w0 + w1*vec_x[0] + w2*vec_x[1]
      W.append(w)
      if w >= 0:
         S1.append(vec_x)
      else:
         S0.append(vec_x)

   # Plot S1
   x1 = [item[0] for item in S1]
   x2 = [item[1] for item in S1]
   plt.figure()
   plt.scatter(x1, x2, marker='o', label='S1')

   # Plot S0
   x1 = [item[0] for item in S0]
   x2 = [item[1] for item in S0]
   plt.scatter(x1, x2, marker='s', label='S0')

   # Plot w_0 + w_1x_1 + w_2x_2 = 0, the separator line
   x = [item[0] for item in S]
   y = [(-w0 - w1*item[0])/w2 for item in S]
   plt.plot(x, y)

   plt.legend()
   plt.title('Geometric Interpretation of Perceptron')
   plt.savefig('Geometric Interpretation of
       Perceptron_N_{}.pdf'.format(N))

   print('Optimal weights: {} {} {}'.format(w0, w1, w2))

   # (b) Perceptron Training algorithm

   # Initial Weights - same for all eta values
   w0_i = np.random.uniform(-1,1,1)
   w1_i = np.random.uniform(-1,1,1)
   w2_i = np.random.uniform(-1,1,1)

   eta_values = [1, 10, 0.1]
   for eta in eta_values:
      print('*******Training for eta = {}*******'.format(eta))
      w0_p = w0_i; w1_p = w1_i; w2_p = w2_i

      print('Initial weights: {} {} {}'.format(w0_p, w1_p, w2_p))

      epochs = 0
      misclassified_arr = []
```

```
while(True):
    misclassified = 0
    w0_pp = w0_p; w1_pp = w1_p; w2_pp = w2_p
    for i in range(N):
        w = w0_p + w1_p * S[i][0] + w2_p * S[i][1]

        if w>=0 and W[i]<0:         # Case1 misclassification
            misclassified += 1
            w0_pp = w0_pp - eta*1
            w1_pp = w1_pp - eta*S[i][0]
            w2_pp = w2_pp - eta*S[i][1]

        elif w<0 and W[i]>=0:       # Case2 misclassification
            misclassified += 1
            w0_pp = w0_pp + eta*1
            w1_pp = w1_pp + eta*S[i][0]
            w2_pp = w2_pp + eta*S[i][1]

    epochs += 1
    w0_p = w0_pp; w1_p = w1_pp; w2_p = w2_pp # Update weights only
        after the epoch
    misclassified_arr.append(misclassified)
    if misclassified == 0:          # Convergence condition
        break

print('Final weights: {} {} {}'.format(w0_p, w1_p, w2_p))
print('Epochs taken:' , epochs)

# Plot epochs vs. misclassifications
epoch_arr = list(range(1, epochs+1))
plt.figure()
plt.xlabel('Epochs')
plt.ylabel('Misclassifications')
plt.xticks(epoch_arr)
plt.plot(epoch_arr, misclassified_arr)
plt.title('Epochs vs. Number of Misclassifications')
plt.savefig('eta_{}_N_{}.pdf'.format(eta, N))
```