Christian McDaniel
clm121@uga.edu

Assignment 1

CSCI 8960
April 4, 2019

Part 1. Logistic Regression

Problem 1.

(a)

Algorithm 1: Non-Private Gradient Functions

```
def ComputeGradient(b_vec, xy_pairs):
    """
    Computes the gradient of the logistic
    loss based on the gradient of the
    equation for logistic loss.
    """
    n = len(xy_pairs)
    loss_grad = 1/n * \
        np.sum([np.multiply(sigma(pair[1]*\
        np.dot(np.transpose(b_vec),pair[0]))-1,
        np.multiply(pair[1],pair[0]))\
        for pair in xy_pairs],axis=0)
    return loss_grad

def ApproxGrad(b_vec, xy_pairs, h=10**-5):
    """
    Calculates the approximate gradient
    of b_vec by approximating the change
    in f(x) - here the logistic loss
    function - due to small perturbations
    along each attribute.
    """
    n = len(xy_pairs)
    approx_grad=[]
    for i in range(len(b_vec)):
        b_plus = copy.deepcopy(b_vec)
        b_plus[i] = b_plus[i]+h
        b_minus = copy.deepcopy(b_vec)
        b_minus[i] = b_minus[i]-h
        grad_i = (((-1/n) * \
            sum([np.log(sigma(pair[1] * \
            np.dot(np.transpose(b_plus),pair[0])))\
            for pair in xy_pairs]))\
            - ((-1/n) * sum([np.log(sigma(pair[1] *
            np.dot(np.transpose(b_minus),pair[0]))) \
            for pair in xy_pairs])))/(2*h)
        approx_grad.append(grad_i)
    return approx_grad
```

When running logistic regression, the calculated and approximate gradients of three different β vectors were returned as follows:

++++ Gradient Example 1; Training Iteration 1 ++++

MEAN DIFFERENCE: -2.7118319614394654e-11

++++ Gradient Example 2; Training Iteration 19 ++++

MEAN DIFFERENCE: -1.3525725248745623e-11

++++ Gradient Example 3; Training Iteration 199 ++++

MEAN DIFFERENCE: -2.102528302562362e-11

Please see **Appendix A** for the full β vectors, gradient calculations and element-wise differences.

(b) The Hessian of $\mathcal{L}(\beta)$ can be represented as follows:

$$H\mathcal{L}(\beta) = \frac{\partial^2 \mathcal{L}}{\partial \beta^2} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial \beta_1 \partial \beta_1} & \frac{\partial^2 \mathcal{L}}{\partial \beta_1 \partial \beta_2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \beta_1 \partial \beta_p} \\ \frac{\partial^2 \mathcal{L}}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 \mathcal{L}}{\partial \beta_2 \partial \beta_2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \beta_2 \partial \beta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{L}}{\partial \beta_p \partial \beta_1} & \frac{\partial^2 \mathcal{L}}{\partial \beta_p \partial \beta_2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial \beta_p \partial \beta_p} \end{bmatrix},$$

where each $\frac{\partial^2 \mathcal{L}}{\partial \beta_i \partial \beta_j}$ can be represented as

$$H\mathcal{L}(\beta)_{qj} = \frac{\partial^2 \mathcal{L}}{\partial \beta_q \partial \beta_j} = \frac{\partial}{\partial \beta_q} \cdot \frac{\partial \mathcal{L}}{\partial \beta_j} = \sigma(y_i \beta_j \cdot x_i)^T (\sigma(y_i \beta_j \cdot x_i) + 1),$$

where $\sigma(x) = 1/(1 + e^{-x})$, i.e., the sigmoid function, $y_i \in \{+1, -1\}$, and $x_i \in^{p+1}$.

(c) Algorithm 2: Sigmoid Function

```
def sigma(x):
    """
    Sigmoid function with clipping for
    overflow avoidance.

    overflow avoidance source:
    https://stackoverflow.com/questions/23128401/
    overflow-error-in-neural-networks-implementation
    """
    signal = np.clip(x, -500, 500 )
    sigmoid = 1/(1+np.exp(-signal))
    return sigmoid
```

(d) Algorithm 3: Loss, Backtrack and Prediction Functions

```
def ComputeLoss(b_vec, xy_pair):
    """
    Computes loss for a single example;
    Mean over all examples taken during main log. reg. algo.
```

```
        """
        l_p = np.log(sigma(xy_pair[1] * \
            np.dot(np.transpose(b_vec),xy_pair[0])))
        return l_p

    def backtrack(b_vector,xy_pairs,grad,s=0.9,a=0.1):
        """
        Runs the backtrack algorithm to determine the
        step size for gradient descent.
        """
        n=1
        while (-1/(len(xy_pairs)) * \
                sum([ComputeLoss(np.subtract(b_vector,
                np.multiply(n,grad)),\
                xy_pairs[i]) for i in range(len(xy_pairs))]))\
                > (-1/(len(xy_pairs)) * \
                sum([ComputeLoss(b_vector,xy_pairs[i])\
                - (a*n)*np.linalg.norm(grad)**2 \
                for i in range(len(xy_pairs))])):
            n *= s
        return n

    def Pred(x_test_i, beta):
        """
        Predicts y_hat for a given example from
        the testing dataset.
        """
        y_hat_i = round(1/(1+np.exp(-(np.dot(np.transpose(beta),
            x_test_i))))[0])
        if y_hat_i==0: y_hat_i=-1
        return y_hat_i
```

As can be seen from **Figure 1**, 249 iterations were required to reach the maximum accuracy for the testing set, and 401 iterations were required to reach the maximum accuracy on the training set.

See **Appendix B** for implementation of Logistic Regression function and (non-private) call to the function.

Output after training for 500 iterations (from 'logregperformance.txt' file, generated at end of training):

Final Loss: [[0.32480719]]

Number of Iterations: 500

Final Test Accuracy: 85.148%

Max Test Accuracy: 85.204%

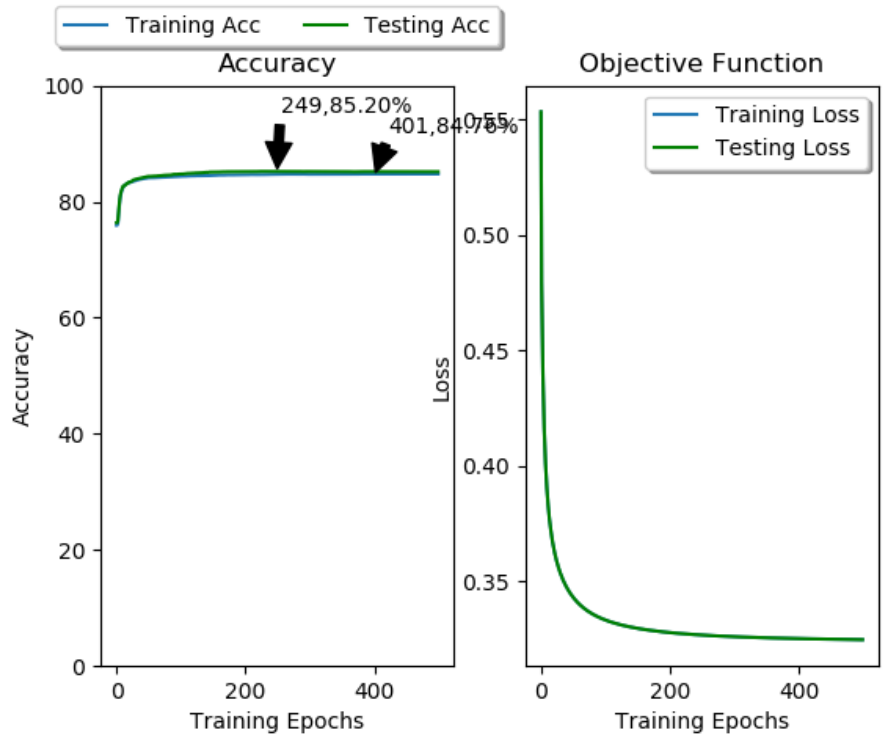


Figure 1: Test and Train (Validation) Accuracy and Loss During Training

Part 2. Differentially Private Gradient Descent

Problem 2.

Algorithm 4: Noisy Gradient Function

```
def NoisyGradient(b_vec, xy_pairs, sensitivity, ep,
                  delta=10**-5):
    """
    Computes the gradient in a differentially private
    manner. Computes the true gradient and adds
    a unique Gaussian noise term to each dimension.
    """
    # See below (part a) for Sensitivity calculation
    var = sensitivity/ep * math.sqrt(2*math.log(1.25/delta))
    Y = np.random.normal(0,
        var**2,xy_pairs[0][0].shape[0])
    grad_tilda = np.reshape(np.add(np.transpose\
        (ComputeGradient(b_vec,xy_pairs))[0],Y),
        (b_vec.shape[0],1))
    return grad_tilda
```

(a)

Algorithm 5: Sensitivity Calculation

```
def dp_sensitivity(b_vec, xy_pairs):
    """
    Computes the sensitivity of the algorithm, used for
    determining the variance parameter of the Gaussian
    noise.
    Here, the upper bound on the difference of norms
    between any two neighboring datasets is set to 1
    due to the use of normalized data.
    """
    max_grad_dj = np.linalg.norm(1)*1
    sens = np.max([max_grad_dj for dj in range(len(xy_pairs))])
    print(f"Sensitivity Calculation for Differential Privacy: {sens}")
    return sens
```

Due to our assumption that $\|x\|_2 \leq 1 \forall x$, the sensitivity is calculated as 1.

(b) See **Appendix B** for implementation of Logistic Regression function and call to the function so it runs in a differentially private way.

Output after training for 15 iterations on each value for epsilon (from 'logregperformance.txt' file, generated at end of training):

Diff. Private Run, epsilon=0.1

Final Loss: [[0.37486264]]

Number of Iterations: 15

Final Test Accuracy: 85.148

Diff. Private Run, epsilon=0.5

Final Loss: [[0.37486264]]

Number of Iterations: 15

Final Test Accuracy: 85.148

Diff. Private Run, epsilon=1.0

Final Loss: [[0.37486264]]

Number of Iterations: 15

Final Test Accuracy: 85.148

Diff. Private Run, epsilon=1.5

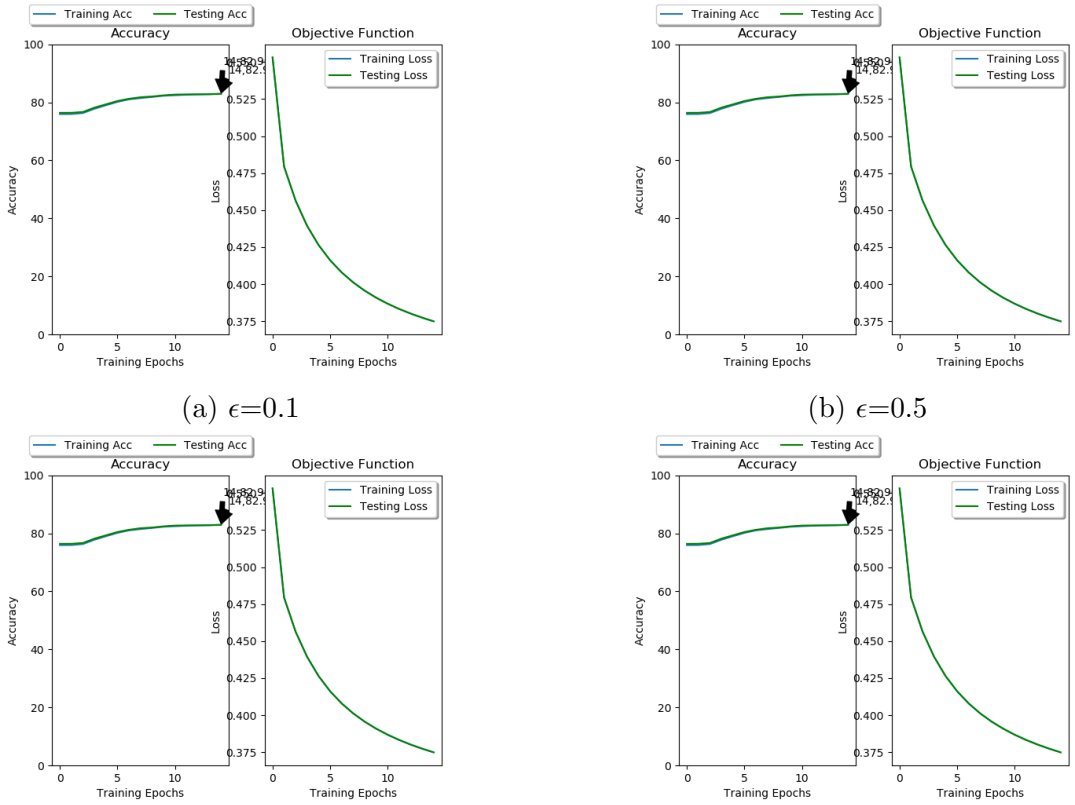
Final Loss: [[0.37486264]]

Number of Iterations: 15

Final Test Accuracy: 85.148

As can be seen, each differentially private model eas able to achieve an accuracy competitive with that of the non-private model.

(c) See **Figure 2** for plots from Differentially Private Training runs.



(a) ϵ=0.1

(b) ϵ=0.5

(c) ϵ=1.0

(d) ϵ=1.5

Figure 2: Training and Testing Accuracies and Loss for four different values of ϵ. Please see the outputs above for Testing Accuracies.

References

In addition to the url's listed as comments in some of the functions throughout this assignment, **Dr. Lee's CSCI8960 lecture notes** were used as guidance to write many of the functions.

Appendix A. Gradient Differences

The following is output from the file 'gradientdiffs.txt' generated during Logistic Regression training:

———- Calculated/Approx. Gradient Differences ———-

++++ Gradient Example 1; Training Iteration 1 ++++

MEAN DIFFERENCE: -2.7118319614394654e-11

Actual Differences: [[[-2.33985956e-10] [1.21753503e-01] [9.74396165e-02] ... [1.02370447e-01] [1.02223353e-01] [1.02296266e-01]]

[[-1.21753503e-01] [-5.44575392e-11] [-2.43138864e-02] ... [-1.93830555e-02] [-1.95301499e-02] [-1.94572364e-02]]

[[-9.74396167e-02] [2.43138864e-02] [5.77349245e-11] ... [4.93083092e-03] [4.78373651e-03] [4.85664998e-03]]

...

[[-1.02370448e-01] [1.93830554e-02] [-4.93083087e-03] ... [-1.00876771e-11] [-1.47094425e-04] [-7.41809488e-05]]

[[-1.02223353e-01] [1.95301499e-02] [-4.78373645e-03] ... [1.47094415e-04] [1.15611120e-13] [7.29134768e-05]]

[[-1.02296267e-01] [1.94572364e-02] [-4.85664992e-03] ... [7.41809402e-05] [-7.29134751e-05] [1.49817670e-12]]]

b-vector: [[-2.59190443e-01 -9.49448727e-02 -6.13771076e-02 -4.24127023e-02 -2.44617794e-02 -3.59939805e-02 -1.96093486e-01 -1.67838826e-02 1.96554160e-03 -3.34756304e-03 -1.31906268e-02 -9.09062989e-03 -2.14981112e-04 -1.07490556e-04 -5.31156906e-02 -4.84782408e-02 -4.93228095e-02 -5.46052025e-02 -5.36684991e-02 -1.40198397e-02 -6.93621203e-02 -1.62003624e-02 -1.09809281e-01 4.14606431e-03 -8.24606124e-03 -1.01348239e-02 -7.06366512e-03 -8.69137926e-03 -5.63557630e-03 2.99437978e-03 -2.39550382e-03 -1.24228371e-02 3.05580295e-03 -4.62209392e-03 -7.83145481e-04 -5.78145634e-02 -1.09809281e-01 -6.93621203e-02 -1.83808851e-02 -3.82359264e-03 -2.44464236e-02 -5.40063266e-02 -1.48966555e-01 -1.37127238e-02 -1.26378182e-02 -5.37452781e-03 -4.60673812e-05 -5.55879733e-03 -3.44123338e-02 -4.63898529e-02 -2.58591567e-02 -1.99625319e-03 -6.48014496e-03 -1.83962409e-02 -2.30644022e-02 -4.23205676e-02 -1.17318264e-02 -1.46954946e-02 -2.25730168e-03 -3.48576518e-03 -1.07490556e-04 -1.19775191e-03 -7.57654863e-02 -2.08378121e-02 -1.01240748e-01 -1.39277049e-02 -4.62209392e-02 -2.08562391e-01 -7.47827155e-03 -3.67003470e-03 -3.39363042e-03 -3.60861153e-02 -1.29188293e-01 -1.30002150e-01 -2.69048862e-01 9.85841958e-03 -2.59605049e-01 4.14606431e-04 -7.38460121e-02 -2.26958631e-02 -1.33948589e-01 -1.13172200e-02 -1.73827585e-02 -2.27695710e-01 -7.67789687e-05 -4.60673812e-04 -1.38202144e-03 -6.60299131e-04 -7.52433893e-04 -2.14981112e-04 -3.07115875e-04 -2.14981112e-04 -1.99625319e-04 -7.37078100e-04 -5.37452781e-04 -6.91010718e-04 -1.07490556e-04 -1.68913731e-04 -1.16704032e-03 -3.53183256e-04 -5.52808575e-04 -9.36703418e-04 -8.75280243e-04 -8.86029299e-03 -4.45318018e-04 -2.14981112e-04 -7.67789687e-05 -1.01348239e-03 -2.14981112e-04 -3.07115875e-04 -1.68913731e-04 -5.52808575e-04 -8.44568656e-04 -1.07490556e-04 -8.90636037e-04 -4.60673812e-04 -9.21347624e-05 -1.84269525e-04 -6.14231750e-05 -1.35130985e-03 -2.30336906e-04 -4.14606431e-04 -1.22846350e-04 -1.53557937e-05]]

Calc. Grad.: [[-1.02291255e-01 1.94622477e-02 -4.85163863e-03 -3.26668725e-02 -4.42035486e-02 -4.00314433e-02 -6.23475381e-02 -9.67974648e-03 -1.31254948e-02 -6.72638721e-03 -8.68744853e-03 -4.56711865e-03 6.47145149e-05 2.96450676e-05 -1.90849207e-02 -2.34784686e-02 -2.30364091e-02 -1.79862116e-02 -1.87052454e-02 -4.27290090e-02 -1.32486307e-02 2.95686328e-03 -1.17104559e-02 -9.79682059e-03 -3.28368941e-03 -4.82438466e-03 1.45720890e-03 1.75507916e-03 8.05501181e-04 -2.05894738e-02 6.41061791e-04 2.02700230e-03 -7.11730944e-03 1.12230361e-03 2.43497897e-04 1.10085181e-02 -1.17104559e-02 -1.32486307e-02 -8.10807408e-03 -8.02326128e-02 -1.35470076e-01 4.10286762e-03 2.41935118e-02 2.41390135e-03 1.75598016e-03 9.20394731e-04 -2.07834939e-04 -4.74815590e-03 -1.19650984e-02 8.49421219e-03 -1.50354393e-02 -4.14329251e-02 -3.69885778e-02 2.54146311e-03 1.93982031e-04 -8.37468077e-04 8.66364965e-04 -3.24918580e-03 5.92868302e-04 -3.51016139e-03 9.10190526e-06 -1.53719831e-02 1.61765239e-02 -1.20194455e-01 6.21321070e-03 3.00106665e-03 7.88438105e-03 -1.03002677e-01 -2.60266714e-03 3.77278576e-04 6.23768564e-04 2.31304136e-03 7.62852491e-03 -1.09919780e-01 -6.58609092e-02 -3.64303462e-02 -8.67547939e-02 -1.55364615e-02 1.07342389e-

02 -1.15472569e-03 -3.83030846e-02 -1.65286627e-02 -5.70390213e-02 -9.94545813e-02 -1.07489134e-04 -4.63136526e-04 2.05414435e-04 -5.74366169e-04 -6.38390107e-04 6.80741129e-05 -5.87520345e-04 -3.80888139e-04 -8.64320604e-05 -2.02889909e-06 -1.47542680e-04 -2.88668787e-04 -3.08673102e-04 3.42594283e-05 -7.23663799e-04 -3.89785060e-04 -7.50182782e-05 1.11872473e-04 2.12846874e-04 2.02632050e-03 5.94736174e-05 -4.44359889e-05 -2.16818089e-04 2.87019522e-04 3.71015141e-05 1.68528028e-05 -2.88410164e-04 1.17707116e-04 2.22905650e-04 -2.33025414e-05 1.98294583e-04 1.02638945e-04 -3.42853299e-05 1.18024434e-05 -1.04939454e-04 2.25115518e-04 4.35877173e-05 7.91922316e-05 -6.79021838e-05 5.01129288e-06]]

Approx. Grad.: [array([[-0.10229126]]), array([[0.01946225]]), array([[-0.00485164]]), array([[-0.03266687]]), array([[-0.04420355]]), array([[-0.04003144]]), array([[-0.06234754]]), array([[-0.00967975]]), array([[-0.01312549]]), array([[-0.00672639]]), array([[-0.00868745]]), array([[-0.00456712]]), array([[6.47145171e-05]]), array([[2.96450697e-05]]), array([[-0.01908492]]), array([[-0.02347847]]), array([[-0.02303641]]), array([[-0.01798621]]), array([[-0.01870525]]), array([[-0.04272901]]), array([[-0.01324863]]), array([[0.00295686]]), array([[-0.01171046]]), array([[-0.00979682]]), array([[-0.00328369]]), array([[-0.00482438]]), array([[0.00145721]]), array([[0.00175508]]), array([[0.0008055]]), array([[-0.02058947]]), array([[0.00064106]]), array([[0.002027]]), array([[-0.00711731]]), array([[0.0011223]]), array([[0.0002435]]), array([[0.01100852]]), array([[-0.01171046]]), array([[-0.01324863]]), array([[-0.00810807]]), array([[-0.08023261]]), array([[-0.13547008]]), array([[0.00410287]]), array([[0.02419351]]), array([[0.0024139]]), array([[0.00175598]]), array([[0.00092039]]), array([[-0.00020783]]), array([[-0.00474816]]), array([[-0.0119651]]), array([[0.00849421]]), array([[-0.01503544]]), array([[-0.04143293]]), array([[-0.03698858]]), array([[0.00254146]]), array([[0.00019398]]), array([[-0.00083747]]), array([[0.00086636]]), array([[-0.00324919]]), array([[0.00059287]]), array([[-0.00351016]]), array([[9.10190812e-06]]), array([[-0.01537198]]), array([[0.01617652]]), array([[-0.12019445]]), array([[0.00621321]]), array([[0.00300107]]), array([[0.00788438]]), array([[-0.10300268]]), array([[-0.00260267]]), array([[0.00037728]]), array([[0.00062377]]), array([[0.00231304]]), array([[0.00762852]]), array([[-0.10991978]]), array([[-0.06586091]]), array([[-0.03643035]]), array([[-0.08675479]]), array([[-0.01553646]]), array([[0.01073424]]), array([[-0.00115473]]), array([[-0.03830308]]), array([[-0.01652866]]), array([[-0.05703902]]), array([[-0.09945458]]), array([[-0.00010749]]), array([[-0.00046314]]), array([[0.00020541]]), array([[-0.00057437]]), array([[-0.00063839]]), array([[6.8074113e-05]]), array([[-0.00058752]]), array([[-0.00038089]]), array([[-8.64320615e-05]]), array([[-2.02889372e-06]]), array([[-0.00014754]]), array([[-0.00028867]]), array([[-0.00030867]]), array([[3.42594231e-05]]), array([[-0.00072366]]), array([[-0.00038979]]), array([[-7.50182805e-05]]), array([[0.00011187]]), array([[0.00021285]]), array([[0.00202632]]), array([[5.9473626e-05]]), array([[-4.44359827e-05]]), array([[-0.00021682]]), array([[0.00028702]]), array([[3.71015163e-05]]), array([[1.68527969e-05]]), array([[-0.00028841]]), array([[0.00011771]]), array([[0.00022291]]), array([[-2.33025432e-05]]), array([[0.00019829]]), array([[0.00010264]]), array([[-3.42853246e-05]]), array([[1.18024368e-05]]), array([[-0.00010494]]), array([[0.00022512]]), array([[4.35877279e-05]]), array([[7.91922417e-05]]), array([[-6.79021839e-05]]), array([[5.01129138e-06]])]

++++ Gradient Example 2; Training Iteration 19 ++++

MEAN DIFFERENCE: -1.3525725248745623e-11

Actual Differences: [[[-1.10232843e-10] [6.69222365e-03] [2.66714193e-03] ... [-2.43447123e-03] [-2.51396078e-03] [-2.52078405e-03]]

[[-6.69222384e-03] [-7.31503261e-11] [-4.02508179e-03] ... [-9.12669495e-03] [-9.20618450e-03] [-9.21300778e-03]]

[[-2.66714214e-03] [4.02508162e-03] [-1.00902127e-10] ... [-5.10161326e-03] [-5.18110281e-03] [-5.18792608e-03]]

...

[[2.43447112e-03] [9.12669488e-03] [5.10161316e-03] ... [3.12855510e-12] [-7.94895441e-05] [-8.63128185e-05]]

[[2.51396067e-03] [9.20618443e-03] [5.18110271e-03] ... [7.94895514e-05] [4.14269799e-12] [-6.82327019e-06]]

[[2.52078394e-03] [9.21300770e-03] [5.18792598e-03] ... [8.63128211e-05] [6.82327387e-06] [-4.58352294e-13]]]

b-vector: [[-2.42995044e-01 -3.97734958e-01 -1.99737482e-01 5.59276474e-02 1.95101558e-01 1.03448191e-01 -1.65926906e-01 -7.09519785e-02 8.79684865e-02 4.79814754e-02 -7.88763845e-03 -1.40898914e-02 -1.86190583e-03 -5.27188293e-04 -8.56846979e-02 -2.01456640e-02 -2.03098514e-02 -6.27117465e-02 -5.41430839e-02 1.56451096e-01 -9.86615749e-02 -8.20741296e-02 -2.48253743e-

01 8.88358479e-02 -2.69802333e-03 -3.94780807e-03 -4.76175175e-02 -6.81103106e-02 -2.50993170e-02 1.47899977e-01 -1.88801318e-02 -6.83276986e-02 6.88481481e-02 -3.53892565e-02 -5.97060175e-03 -3.51468963e-01 -2.48253743e-01 -9.86615749e-02 -6.64583140e-03 4.62035069e-01 5.88534277e-01 -1.78533536e-01 -5.21235889e-01 -6.15869713e-02 -4.70359981e-02 -2.61050438e-02 2.96811713e-03 3.05618005e-02 -7.12708143e-02 -2.07477509e-01 2.65615884e-02 2.90214500e-01 2.26027167e-01 -9.32937297e-02 -9.88241038e-02 -8.76034453e-02 -8.86352028e-02 -5.31427647e-02 -1.10973828e-02 1.38072619e-02 -5.95723176e-04 1.78892840e-01 -3.07650988e-01 4.22569768e-01 -2.77648120e-01 -6.74675743e-02 -1.91690971e-01 -6.14800314e-02 -2.00625368e-02 -1.89687822e-02 -2.07485722e-02 -1.21735121e-01 -3.01890251e-01 5.88952070e-02 -5.44353426e-01 3.01358382e-01 -3.64126565e-01 1.21131521e-01 -3.25773307e-01 -6.79115617e-02 -1.73456882e-01 7.51101506e-02 2.49036556e-01 -9.75087613e-02 9.23368796e-04 2.28049891e-03 -8.13045469e-03 1.59085456e-03 2.13196552e-03 -1.63663270e-03 -2.83175204e-04 1.82676910e-03 -1.49218725e-03 -5.28450536e-03 -3.82063404e-03 -8.23374890e-04 1.03760327e-03 -8.11797760e-04 7.67000533e-04 1.21247907e-03 -2.56821892e-03 -3.91896244e-03 -6.41916500e-03 -6.28186330e-02 -3.02543219e-03 -2.85572360e-04 1.67880365e-03 -6.79920256e-03 -1.35624834e-03 -1.80955186e-03 7.49054053e-04 -3.02663471e-03 -6.42897619e-03 -4.50928561e-04 -4.85549787e-03 -3.14327189e-03 -5.16408368e-05 -9.49792363e-04 3.50833181e-04 -7.68553233e-03 -1.34854000e-03 -2.45078822e-03 -2.12005966e-04 -8.24193819e-05]]

Calc. Grad.: [[2.52303040e-03 9.21525416e-03 5.19017244e-03 -2.36335043e-03 -6.01610548e-03 -3.50294029e-03 -2.47336026e-03 4.33631865e-03 -2.53674295e-03 -2.33976947e-03 6.85152858e-04 7.46961136e-04 8.28445064e-05 1.25912672e-05 2.43791052e-03 -8.96842482e-05 -2.89395636e-04 3.68517806e-04 9.56819550e-05 -1.16702470e-03 -3.43792995e-04 2.34951265e-03 2.34446394e-03 -3.04655765e-03 -3.57435272e-04 -2.18712423e-04 1.74804113e-03 2.78436066e-03 6.11362237e-04 -4.14373589e-03 7.38024501e-04 2.16318175e-03 -2.53702352e-03 1.37066024e-03 2.27705744e-04 1.19928489e-02 2.34446394e-03 -3.43792995e-04 -5.76147695e-04 -1.08943418e-02 -1.17215049e-02 3.01390564e-03 8.30802798e-03 1.54339269e-03 8.42032329e-04 7.17027633e-04 -1.79851011e-04 -1.94317793e-03 1.61451971e-03 5.43329011e-03 -1.96242458e-03 -9.46065762e-03 -6.76539471e-03 2.98734054e-03 3.09203413e-03 6.58834765e-05 4.03411163e-03 1.89649154e-03 3.02797676e-04 -8.15770207e-04 2.23607010e-05 -8.91632397e-03 6.45975660e-03 -3.27257960e-03 2.07918550e-03 1.78119206e-03 4.39179981e-03 -1.90016092e-03 7.95136221e-04 6.48250448e-04 7.47970473e-04 2.23183417e-03 8.64294899e-04 1.65873550e-03 1.21060425e-02 -9.58301215e-03 6.94812012e-03 -4.42508972e-03 9.04402144e-03 1.56446334e-03 3.01215820e-04 -2.88042410e-03 -5.50624611e-03 -3.12272540e-03 -6.12700621e-05 -1.26813891e-04 2.94706353e-04 -8.87606213e-05 -1.20798879e-04 6.52592260e-05 1.24994454e-04 -6.90720434e-05 1.02551786e-04 2.60157688e-04 2.21859278e-04 2.43670434e-06 -8.99359340e-06 2.16480282e-05 -9.66847355e-05 -6.96882536e-05 1.07872478e-04 9.77797920e-05 2.62005343e-04 2.35352358e-03 1.17190314e-04 -7.07661014e-06 -7.87615494e-05 2.41917222e-04 4.80482500e-05 7.27826066e-05 -1.23973640e-05 9.08966746e-05 2.70152227e-04 2.01367352e-05 1.41222520e-04 1.22501567e-04 -3.17684638e-06 3.55585472e-05 -1.00258171e-05 2.68069161e-04 4.42672716e-05 8.85592769e-05 9.06972967e-06 2.24645534e-06]]

Approx. Grad.: [array([[0.00252303]]), array([[0.00921525]]), array([[0.00519017]]), array([[-0.00236335]]), array([[-0.00601611]]), array([[-0.00350294]]), array([[-0.00247336]]), array([[0.00433632]]), array([[-0.00253674]]), array([[-0.00233977]]), array([[0.00068515]]), array([[0.00074696]]), array([[8.28444979e-05]]), array([[1.25912669e-05]]), array([[0.00243791]]), array([[-8.96842239e-05]]), array([[-0.0002894]]), array([[0.00036852]]), array([[9.56820123e-05]]), array([[-0.00116702]]), array([[-0.00034379]]), array([[0.00234951]]), array([[0.00234446]]), array([[-0.00304656]]), array([[-0.00035744]]), array([[-0.00021871]]), array([[0.00174804]]), array([[0.00278436]]), array([[0.00061136]]), array([[-0.00414374]]), array([[0.00073802]]), array([[0.00216318]]), array([[-0.00253702]]), array([[0.00137066]]), array([[0.00022771]]), array([[0.01199285]]), array([[0.00234446]]), array([[-0.00034379]]), array([[-0.00057615]]), array([[-0.01089434]]), array([[-0.0117215]]), array([[0.00301391]]), array([[0.00830803]]), array([[0.00154339]]), array([[0.00084203]]), array([[0.00071703]]), array([[-0.00017985]]), array([[-0.00194318]]), array([[0.00161452]]), array([[0.00543329]]), array([[-0.00196242]]), array([[-0.00946066]]), array([[-0.00676539]]), array([[0.00298734]]), array([[0.00309203]]), array([[6.58834903e-05]]), array([[0.00403411]]), array([[0.00189649]]), array([[0.0003028]]), array([[-0.00081577]]), array([[2.23607022e-05]]), array([[-0.00891632]]), array([[0.00645976]]), array([[-0.00327258]]), array([[0.00207919]]), array([[0.00178119]]), array([[0.0043918]]), array([[-0.00190016]]), array([[0.00079514]]), array([[0.00064825]]),

array([[0.00074797]]), array([[0.00223183]]), array([[0.0008643]]), array([[0.00165874]]), array([[0.01210604]]), array([[-0.00958301]]), array([[0.00694812]]), array([[-0.00442509]]), array([[0.00904402]]), array([[0.00156446]]), array([[0.00030122]]), array([[-0.00288042]]), array([[-0.00550625]]), array([[-0.00312273]]), array([[-6.12700668e-05]]), array([[-0.00012681]]), array([[0.00029471]]), array([[-8.87606322e-05]]), array([[-0.0001208]]), array([[6.5259223e-05]]), array([[0.00012499]]), array([[-6.9072037e-05]]), array([[0.00010255]]), array([[0.00026016]]), array([[0.00022186]]), array([[2.43669529e-06]]), array([[-8.99359476e-06]]), array([[2.16480334e-05]]), array([[-9.66847463e-05]]), array([[-6.96882746e-05]]), array([[0.00010787]]), array([[9.77797815e-05]]), array([[0.00026201]]), array([[0.00235352]]), array([[0.00011719]]), array([[-7.07660874e-06]]), array([[-7.87615556e-05]]), array([[0.00024192]]), array([[4.80482459e-05]]), array([[7.27826077e-05]]), array([[-1.23973637e-05]]), array([[9.08966735e-05]]), array([[0.00027015]]), array([[2.01367339e-05]]), array([[0.00014122]]), array([[0.0001225]]), array([[-3.17684767e-06]]), array([[3.55585478e-05]]), array([[-1.00258191e-05]]), array([[0.00026807]]), array([[4.42672704e-05]]), array([[8.85592738e-05]]), array([[9.06972553e-06]]), array([[2.2464558e-06]])]

++++ Gradient Example 3; Training Iteration 199 ++++

MEAN DIFFERENCE: -2.102528302562362e-11

Actual Differences: [[[-1.75501895e-10] [1.13458589e-03] [-3.75322278e-04] ... [-4.12944960e-04] [-4.65699235e-04] [-4.57397030e-04]]

[[-1.13458614e-03] [-6.70971221e-11] [-1.50990824e-03] ... [-1.54753092e-03] [-1.60028520e-03] [-1.59198299e-03]]

[[3.75322051e-04] [1.50990812e-03] [-5.18683845e-11] ... [-3.76227336e-05] [-9.03770086e-05] [-8.20748042e-05]]

...

[[4.12944783e-04] [1.54753085e-03] [3.76226808e-05] ... [-8.78857107e-13] [-5.27542759e-05] [-4.44520715e-05]]

[[4.65699059e-04] [1.60028513e-03] [9.03769568e-05] ... [5.27542751e-05] [3.26658256e-14] [8.30220443e-06]]

[[4.57396856e-04] [1.59198293e-03] [8.20747534e-05] ... [4.44520717e-05] [-8.30220334e-06] [1.06393457e-12]]]

b-vector: [[-4.02286578e-01 -9.97212112e-01 -4.00480145e-01 1.66900293e-01 4.39146356e-01 3.89359029e-01 5.06174898e-02 -3.28488692e-01 2.59293277e-01 3.23062223e-01 -9.57760743e-02 -1.06117951e-01 -1.40020476e-02 -1.71306187e-03 -2.29063089e-01 -5.14046993e-02 -4.60145962e-02 -4.98307163e-02 -2.59734780e-02 -8.64367808e-02 -3.69521919e-02 -1.69038823e-01 -2.21484934e-01 3.63181376e-01 3.93881380e-02 2.47987100e-02 -1.66825464e-01 -2.72679874e-01 -2.06520522e-02 2.15996458e-01 -7.48046347e-02 -1.61119687e-01 3.22884866e-01 -1.30487135e-01 -2.80545508e-02 -1.02366222e+00 -2.21484934e-01 -3.69521919e-02 6.41868480e-02 8.15625919e-01 1.02981761e+00 -3.68003917e-01 -7.23147406e-01 -1.98014556e-01 -8.98222455e-02 -9.01367245e-02 3.70206610e-02 3.13592551e-01 -1.17432150e-02 -5.66475400e-01 2.21324498e-01 7.12298193e-01 5.19235790e-01 -3.44631482e-01 -3.16844777e-01 -9.82049640e-03 -4.54374363e-01 -1.33520141e-01 -4.50661833e-02 2.08063563e-01 -3.45031295e-03 9.05125555e-01 -6.72489938e-01 1.81393131e-01 -1.45459856e-01 -2.09942988e-01 -4.60912483e-01 6.92365368e-03 -4.38156021e-02 -9.18836609e-02 -9.19726891e-02 -1.81538280e-01 -4.26663085e-01 2.43765061e-02 -1.01405382e+00 6.11767240e-01 -7.36410020e-01 3.34123441e-01 -8.98781363e-01 -1.70846728e-01 -5.90292302e-02 2.80589274e-01 4.45781469e-01 1.62497994e-01 1.59134120e-02 2.54924031e-02 -3.88278801e-02 2.65429776e-02 2.92748093e-02 -9.96579941e-03 -2.09856453e-02 1.44860126e-02 -2.11057965e-02 -3.98443850e-02 -3.41319325e-02 1.08017585e-02 1.63624464e-03 -3.05902466e-03 4.34808937e-02 2.82352168e-02 -1.22912380e-02 -8.21756104e-03 -3.66812188e-02 -2.37597822e-01 -1.13710693e-02 4.99298163e-03 1.40679078e-02 -3.13284544e-02 -5.51150454e-03 -9.13039519e-03 6.09739137e-03 -1.00703009e-02 -4.42880639e-02 -2.93037871e-03 -1.40659605e-02 -1.65616136e-02 -1.52829584e-04 -5.03991702e-03 4.67924042e-03 -3.54013396e-02 -5.41549417e-03 -1.27010431e-02 3.63503829e-04 -3.13549808e-04]]

Calc. Grad.: [[4.58329019e-04 1.59291509e-03 8.30069161e-05 -2.02851602e-04 -3.57253673e-04 -6.57487711e-04 -3.20839618e-04 2.03444986e-04 -4.30958586e-04 -9.72609544e-04 1.92705487e-04 3.17752000e-04 6.13229306e-05 5.91102199e-06 2.17208871e-04 1.16069282e-04 1.29264809e-04 1.17391911e-05 -1.59531346e-05 8.46334563e-04 1.24212193e-05 -7.02896874e-05 -6.42227626e-06 -9.25007813e-04 -5.34639100e-05 -1.77308407e-05 3.00271758e-04 5.50240203e-04 -2.03745207e-04 4.78554091e-04 1.65063208e-04 1.08470566e-05 -9.58360532e-04 2.41051050e-04 8.85661341e-05 1.08200452e-03 -6.42227626e-06 1.24212193e-05 -7.11947507e-

05 -5.58479690e-04 -6.73093947e-04 4.84597920e-04 -2.28311732e-05 5.29830272e-04 6.74707749e-05 2.62409235e-04 -1.90054063e-04 -1.20275000e-03 -4.16877291e-04 8.63102064e-04 -5.49803898e-04 -4.77659698e-04 -3.77472614e-04 8.13311332e-04 4.76072867e-04 -2.91059341e-04 1.12768221e-03 -2.45774141e-05 1.55359316e-04 -1.05969707e-03 1.51871861e-05 -1.94917101e-03 8.80721615e-04 1.12155316e-03 -4.66367840e-04 5.09449037e-04 3.62144056e-04 1.25452958e-05 -1.11744127e-04 3.27561669e-04 2.73224744e-04 -4.32585624e-05 8.84813210e-04 -4.26484191e-04 4.01564068e-04 5.67649504e-05 4.85852206e-04 -2.75231878e-05 1.14003854e-03 2.03866151e-04 -2.09070149e-04 -4.49491008e-04 -2.27014515e-04 -5.54496669e-04 -9.01331472e-05 -1.33391482e-04 1.25174942e-04 -1.63784120e-04 -1.75604144e-04 3.97776544e-05 1.00545872e-04 -7.87574930e-05 1.05696682e-04 1.61706135e-04 1.40700964e-04 -9.35530614e-05 -1.08459632e-05 1.00560072e-05 -2.87119987e-04 -1.85603656e-04 2.51081594e-05 2.38343772e-06 1.32942954e-04 4.86975262e-04 2.14047462e-05 -3.90112339e-05 -6.64282298e-05 1.03709096e-04 1.61634955e-05 2.76566231e-05 -3.88232418e-05 2.33773225e-05 1.90295617e-04 9.38190927e-06 2.52687117e-05 5.66448569e-05 2.09300345e-06 1.71723543e-05 -3.44743194e-05 1.15280987e-04 1.83457020e-05 4.53842344e-05 -7.37004066e-06 9.32163744e-07]]

Approx. Grad.: [array([[0.00045833]]), array([[0.00159292]]), array([[8.3006968e-05]]), array([[-0.00020285]]), array([[-0.00035725]]), array([[-0.00065749]]), array([[-0.00032084]]), array([[0.00020344]]), array([[-0.00043096]]), array([[-0.00097261]]), array([[0.00019271]]), array([[0.00031775]]), array([[6.13229245e-05]]), array([[5.9110189e-06]]), array([[0.00021721]]), array([[0.00011607]]), array([[0.00012926]]), array([[1.17392374e-05]]), array([[-1.59531111e-05]]), array([[0.00084633]]), array([[1.2421314e-05]]), array([[-7.02896907e-05]]), array([[-6.42237652e-06]]), array([[-0.00092501]]), array([[-5.34638583e-05]]), array([[-1.77308113e-05]]), array([[0.00030027]]), array([[0.00055024]]), array([[-0.00020375]]), array([[0.00047855]]), array([[0.00016506]]), array([[1.08470594e-05]]), array([[-0.00095836]]), array([[0.00024105]]), array([[8.85661378e-05]]), array([[0.001082]]), array([[-6.4223793e-06]]), array([[1.2421314e-05]]), array([[-7.11946668e-05]]), array([[-0.00055848]]), array([[-0.00067309]]), array([[0.0004846]]), array([[-2.28310315e-05]]), array([[0.00052983]]), array([[6.74707984e-05]]), array([[0.00026241]]), array([[-0.00019005]]), array([[-0.00120275]]), array([[-0.00041688]]), array([[0.0008631]]), array([[-0.0005498]]), array([[-0.00047766]]), array([[-0.00037747]]), array([[0.00081331]]), array([[0.00047607]]), array([[-0.00029106]]), array([[0.00112768]]), array([[-2.45774012e-05]]), array([[0.00015536]]), array([[-0.0010597]]), array([[1.51871876e-05]]), array([[-0.00194917]]), array([[0.00088072]]), array([[0.00112155]]), array([[-0.00046637]]), array([[0.00050945]]), array([[0.00036214]]), array([[1.25454619e-05]]), array([[-0.00011174]]), array([[0.00032756]]), array([[0.00027322]]), array([[-4.32585856e-05]]), array([[0.00088481]]), array([[-0.00042648]]), array([[0.00040156]]), array([[5.67649483e-05]]), array([[0.00048585]]), array([[-2.75231421e-05]]), array([[0.00114004]]), array([[0.00020387]]), array([[-0.00020907]]), array([[-0.00044949]]), array([[-0.00022701]]), array([[-0.0005545]]), array([[-9.01331454e-05]]), array([[-0.00013339]]), array([[0.00012517]]), array([[-0.00016378]]), array([[-0.0001756]]), array([[3.97776562e-05]]), array([[0.00010055]]), array([[-7.87574894e-05]]), array([[0.0001057]]), array([[0.00016171]]), array([[0.0001407]]), array([[-9.35530681e-05]]), array([[-1.0845963e-05]]), array([[1.00560144e-05]]), array([[-0.00028712]]), array([[-0.0001856]]), array([[2.510816e-05]]), array([[2.38344344e-06]]), array([[0.00013294]]), array([[0.00048698]]), array([[2.14047557e-05]]), array([[-3.90112359e-05]]), array([[-6.64282379e-05]]), array([[0.00010371]]), array([[1.61634955e-05]]), array([[2.76566214e-05]]), array([[-3.88232363e-05]]), array([[2.33773306e-05]]), array([[0.0001903]]), array([[9.38190636e-06]]), array([[2.52687093e-05]]), array([[5.66448582e-05]]), array([[2.09300077e-06]]), array([[1.71723552e-05]]), array([[-3.44743206e-05]]), array([[0.00011528]]), array([[1.83457027e-05]]), array([[4.53842353e-05]]), array([[-7.37004069e-06]]), array([[9.3216268e-07]])]

Appendix B. Logistic Regression Main Functions

Algorithm 6: Logistic Regression Training Function

```
def logistic_reg_train(x,y,x_tst,y_tst,iters=100,epsilon=10**-10,
                       thresh=10**-6,glob_sens=None,dp=False,dp_ep=0.5):
    if not dp:
```

```
        # initialize file for recording calc'd v. approx
        # gradients
        with open('./gradient_diffs.txt','w') as gd:
            gd.write('\n---------- Calculated/Approx. Gradient Differences ----------\n

    # get data
    xy = [[np.reshape(x[i],(x.shape[1],1)),y[i]] \
        for i in range(x.shape[0])]
    xy_test = [[np.reshape(x_tst[i],(x_tst.shape[1],1)),
        y_tst[i]] for i in range(x_tst.shape[0])]

    # initialize b-vector
    beta_t = np.reshape(np.zeros(x.shape[1]),(x.shape[1],1))

    if not glob_sens:
        # calc. sensitivity if not already provided
        glob_sens = dp_sensitivity(beta_t,xy)

    # compute the noisy or non-private gradient
    if dp: g_t = NoisyGradient(beta_t,xy,glob_sens,dp_ep)
    else: g_t = ComputeGradient(beta_t,xy)

    # compute the step size
    alpha_t = backtrack(beta_t,xy,g_t)

    # compute the initial loss
    l_new = -1/(len(xy))*sum([ComputeLoss(beta_t,xy[i])\
            for i in range(x.shape[0])])

    tr_ax=[]
    te_ax=[]
    objectives=[]
    obj_test=[]
    ex=0

    # run iterative training
    for k in range(iters):
        #gradient
        if dp: g_t = NoisyGradient(beta_t,xy,glob_sens,dp_ep)
        else: g_t = ComputeGradient(beta_t,xy)

        if k+1 in [2,20,200] and not dp:
            # calculate and record difference between calc'd
            # and approx gradients for select iterations
            ex+=1
            approx_grad = ApproxGrad(beta_t,xy)
            grad_diffs = np.mean(np.subtract(g_t,approx_grad))
            print(f"Gradient Example {ex} - On iteration {k}, the average difference be
            with open('./gradient_diffs.txt','a') as gd:
                gd.write(f'++++ Gradient Example {ex}; Training Iteration {k} ++++\nMEA
```

```
        if np.linalg.norm(g_t) < epsilon:
            # training checkpoint
            return {'beta':beta_t,'loss':l_new,
                    'iters':k,'step':alpha_t,
                    'exit':'1','train_acc':tr_ax,
                    'test_acc':te_ax,
                    'train_obj':objectives,
                    'test_obj':obj_test,
                    'sensitivity':glob_sens}

        # loss
        l_t = -1/(len(xy))*sum([ComputeLoss(beta_t,xy[i])\
            for i in range(x.shape[0])])

        # backtrack
        alpha_t = backtrack(beta_t,xy,g_t)
        print(f'step size on iter {k+1}: {alpha_t}')

        # update b-vector
        beta_new = np.subtract(beta_t,np.multiply(alpha_t,g_t))

        # calc new loss
        l_new = -1/(len(xy))*sum([ComputeLoss(beta_new,xy[i])\
            for i in range(x.shape[0])])
        if not dp:
            # training checkpoint 2
            if (l_t - l_new) <= thresh:
                return {'beta':beta_t,'loss':l_new,
                        'iters':k,'step':alpha_t,'exit':'2',
                        'train_acc':tr_ax,'test_acc':te_ax,
                        'train_obj':objectives,
                        'test_obj':obj_test,
                        'sensitivity':glob_sens}

        beta_t = beta_new

        # evaluate model re current iteration
        tr_acc = sum([1 for predxn in range(x.shape[0])\
                if Pred(x[predxn],beta_t) == y[predxn]])\
                / x.shape[0] * 100
        te_acc = sum([1 for predxn in range(x_tst.shape[0])\
                if Pred(x_tst[predxn],beta_t) == y_tst[predxn]])\
                 / x_tst.shape[0] * 100
        l_tst = -1/(len(xy))*sum([ComputeLoss(beta_new,xy[i])\
            for i in range(x.shape[0])])
        tr_ax.append(tr_acc)
        te_ax.append(te_acc)
        objectives.append(l_new[0][0])
        obj_test.append(l_tst[0][0])
    return {'beta':beta_t,'loss':l_new,'iters':k,
```

```
        'step':alpha_t,'exit':'3',
        'train_acc':tr_ax,'test_acc':te_ax,
        'train_obj':objectives,
        'test_obj':obj_test,'sensitivity':glob_sens}
```

Appendix C. Other Functions

Algorithm 7: Imported Packages and Libraries

```
from joblib import Memory
from sklearn.datasets import load_svmlight_file
from scipy.sparse import hstack
import matplotlib.pyplot as plt
import math
import numpy as np
import copy
from sklearn.linear_model import LogisticRegression
```

Algorithm 8: Function to Load the Data

```
@mem.cache
def get_data():
    #source: https://scikit-learn.org/stable/modules/generated/
    sklearn.datasets.load_svmlight_file.html
    tr_data = load_svmlight_file("adult_train.txt")
    te_data = load_svmlight_file("adult_test.t")

    # add a dummy column to calculate the bias b0
    tr_dummy = np.ones((tr_data[0].shape[0],1))
    te_dummy = np.ones((te_data[0].shape[0],1))

    trn_data = hstack((tr_dummy,tr_data[0])).toarray()
    tst_data = hstack((te_dummy,te_data[0],te_dummy)).toarray()

    return trn_data, tr_data[1], tst_data, te_data[1]
```

Algorithm 9: Function to Plot the graphs

```
def plot_training(trn_acc, tst_acc, obj_vals, obj_tv,
        dp=False,ep=0.5):
    tr_maxiter = trn_acc.index(np.max(trn_acc))
    te_maxiter = tst_acc.index(np.max(tst_acc))
    f,axrow = plt.subplots(1,2)
    axrow[0].plot(range(len(trn_acc)),trn_acc,
        label="Training Acc")
    axrow[0].plot(range(len(tst_acc)),tst_acc,
        label="Testing Acc", color='g')
    axrow[0].set(xlabel='Training Epochs',ylabel='Accuracy')
```

```
    axrow[0].set_title("Accuracy")
    axrow[0].set_ylim(0,100)
    axrow[0].annotate(f'{tr_maxiter},{np.max(trn_acc):.2f}%',
        xy=(tr_maxiter,np.max(trn_acc)),
        xytext=(tr_maxiter+0.04*len(trn_acc),
        np.max(trn_acc)+7),arrowprops=dict(facecolor='black', shrink=0.05),
        )
    axrow[0].annotate(f'{te_maxiter},{np.max(tst_acc):.2f}%',
        xy=(te_maxiter,np.max(tst_acc)),
        xytext=(te_maxiter+0.01*len(tst_acc),
        np.max(tst_acc)+10),arrowprops=dict(facecolor='black', shrink=0.05),
        )
    #annotation source:
    https://matplotlib.org/users/annotations_intro.html
    axrow[0].legend(loc='lower center',
        bbox_to_anchor=(0.5, 1.05),
        ncol=2, fancybox=True, shadow=True)
    axrow[1].plot(range(len(obj_vals)),obj_vals,
        label="Training Loss")
    axrow[1].plot(range(len(obj_tv)),obj_tv,
        label="Testing Loss",color='g')
    axrow[1].set(xlabel='Training Epochs',ylabel="Loss")
    axrow[1].set_title('Objective Function')
    axrow[1].legend(loc='upper right', bbox_to_anchor=(1, 1),
          ncol=1, fancybox=True, shadow=True)
    #legend source: https://pythonspot.com/matplotlib-legend/
    if dp: plt.savefig(f'./dp_plot_{ep}ep.png')
    else: plt.savefig('./nonpriv_plot.png')
    return
```

Algorithm 10: Function to Compare to SciKit-Learn Logistic Regression

```
def verify_performance(X_trn,y_trn,X_tst,y_tst):
    logreg = LogisticRegression().fit(X_trn[:,1:], y_trn)
    logr_pred = logreg.predict(X_tst[:,1:])
    beta=logreg.coef_
    b0 = logreg.intercept_
    acc = 100 - (sum([1 for predxn in range(len(logr_pred))\
        if logr_pred[predxn]!= y_tst[predxn]])/len(X_tst)*100)
    return acc
```