Problem 1. **Problem Statement**

Write down detailed formulas for the gradient of the loss function in the case of logistic regression, and write detailed pseudo code for training a LR model based on gradient descent. Count how many operations are done per each gradient descent iteration and explain how you computed your answer (use the following variables in your answer: n for the number of examples and d for the dimensionality)

derivative of formulas

Derive the gradient of the negative log-likelihood in terms of w for this setting.

$$y = -sign(\langle \theta, x \rangle)$$

calculate the maximum likelihood estimation:

$$\hat{\theta}_{MLE} = argmax_\theta \prod_{i=1}^{n} P_\theta(Y = y^{(i)} | X = x^{(i)})$$

$$= argmax_\theta \ln(\prod_{i=1}^{n} P_\theta(Y = y^{(i)} | X = x^{(i)}))$$

$$= argmax_\theta \sum_{i=1}^{n} ln(P_\theta(Y = y^{(i)} | X = x^{(i)}))$$

Since logistic function is a natural choice to represent the degree of certainty as probabilites. $f(x) = \frac{1}{1+exp(-x)}$. Then we define:

$$P_\theta(Y = y^{(i)} | X = x^{(i)}) = \frac{1}{1 + exp(y^{(i)} \langle \theta, x^{(i)} \rangle)}$$

after substitute our probability function into equation, we get:

$$\hat{\theta}_{MLE} = argmax_\theta \sum_{i=1}^{n} \frac{1}{1 + exp(y^{(i)} \langle \theta, x^{(i)} \rangle)}$$

since $ln(A^{(-1)}) = -ln(A)$, the above equation becomes:

$$\hat{\theta}_{MLE} = argmin_\theta \sum_{i=1}^{n} ln(1 + exp(y^{(i)} \langle \theta, x^{(i)} \rangle))$$

here we try to find θ that minimizes $\mathcal{L}(\theta) = \sum_i ln(1 + exp(y^{(i)} \langle \theta, x^{(i)} \rangle))$
to calculate the $\bigtriangledown \mathcal{L}$, we need to calculate the partial derivative:

$$\frac{\partial}{\partial \theta^{(i)}} = \frac{\partial}{\partial \theta^{(i)}} \sum_{j=1}^{n} ln(1 + exp(y^{(j)} \langle \theta, x^{(i)} \rangle))$$

$$= \sum_{j=1}^{n} \frac{\partial}{\partial \theta^{(i)}} ln(1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle))$$

since $\frac{d}{dx} ln(f(x)) = \frac{f'(x)}{f(x)}$, the above equation becomes:

$$= \sum_{j=1}^{n} \frac{\frac{\partial}{\partial \theta^{(i)}} (1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle))}{1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

$$= \sum_{j=1}^{n} \frac{\frac{\partial}{\partial \theta^{(i)}} exp(y^{(j)} \langle \theta, x^{(j)} \rangle)}{1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

Since $exp(y^{(i)} \langle \theta, x^{(i)} \rangle) = exp(y^{(j)} \sum_{k=1}^{d} \theta_k x_k^{(j)})$ and using the exponential derivative property, $\frac{d}{dx} exp(f(x)) = exp(f(x)) \cdot f'(x)$

We can obtain:

$$\frac{\partial}{\partial \theta^{(i)}} \mathcal{L}(\theta) = \sum_{j=1}^{n} \frac{exp(y^{(j)} \langle \theta, x^{(j)} \rangle) \cdot \frac{\partial}{\partial \theta^{(i)}} (y^{(j)} \sum_{k=1}^{d} \theta_k x_x^{(j)})}{1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

the only non-zero term in the derivative part of the expression occurs when k = i, so our expression becomes:

$$\frac{\partial \mathcal{L}}{\partial \theta_i}(\theta) = \sum_{j=1}^{n} \frac{exp(y^{(j)} \langle \theta, x^{(j)} \rangle) \cdot \frac{\partial}{\partial \theta^{(i)}} (y^{(j)} \theta_i x_i^{(j)})}{1 + exp(y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

$$= - \sum_{j=1}^{n} \frac{exp(y^{(j)} \langle \theta, x^{(j)} \rangle) \cdot y^{(j)} x_i^{(j)}}{1 + exp(y^{(j)} \langle \theta, x_i^{(j)} \rangle)}$$

it can be further simplified to:

$$= - \sum_{j=1}^{n} \frac{y^{(j)} x_i^{(j)}}{1 + exp(-y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

The gradient has the above as its components:

$$\bigtriangledown \mathcal{L}(\theta) = (\frac{\partial \mathcal{L}}{\partial \theta_1}(\theta), ..., \frac{\partial \mathcal{L}}{\partial \theta_d}(\theta))$$

Therefore θ can be updated as :

$$\theta_i^t = \theta_i^{(t-1)} - \eta \cdot \sum_{j=1}^{n} \frac{y^{(j)} x_i^{(j)}}{1 + exp(-y^{(j)} \langle \theta, x^{(j)} \rangle)}$$

η is learning rate

i $\in$ {1,2...d} (represents each feature)

j $\in$ {1, 2, ... n} (represents each sample)

Pseudo code for training LR model

Data: a n by d numeric matrix and a n by 1 matrix.
Result: a n by 1 numeric matrix
Initialization of a n by 1 numeric matrix θ;
while θ *greater than tolerance* **do**
| update theta according to the gradient descent rule;
end

Algorithm 1: Pseudocode for logistic regression

Count the operations for each gradient descent iteration

Calculate the operation for each gradient descent iteration:

the operation is calculated based on my code (in question 3). For each iteration:

Step 1, calculate the operations for z:

matrix X_train times θ:

dimension of X_train: n by (d + 1)

dimension of θ: (d + 1) by 1

for each sample out of n samples, there are (d + 1) multiplications and d additions. Therefore there are n * (2d + 1) operations for all n samples.

Step 2, calculate operations for H:

Since Y * Z operates on n samples, it counts as n operations.

Exponential (Y * Z) also operates on n samples, counts as n operations.

1 + exponential (Y * Z) operates on n samples, also counts as n operatons.

Y / (1 + exponentila(Y * z)) operates on n samples, counts as n oerations as well.

Therefore there are 4n operations.

Step 3, calculate the operation for er_in(in sample error):

Since X_train transpose is not involved in any computation, therefore I did not count it as operation.

Then calculate the operations for matrix of X_train transpose times vector H,

dimension of X_train transpose: (d + 1) by n

dimension of H: n by 1

for each dimension of (d + 1) dimensions, there are n multiplications and n - 1 additions, therefore there are (2n - 1)(d + 1) total operatoins.

then one division, one multiplication and one subtraction on (d + 1) dimension, so 3(d + 1) operations.

therefore there are (2n - 1)(d + 1) + 3(d + 1) operations.

In conclusion, total operations involved in each iteration are : n (2d + 1) + 4n + (2n - 1) (d + 1) + 3(d + 1)

= 2dn + n + 4n + 2dn -d + 2n - 1 + 3d + 3

= 4dn + 2d + 7n