[ML20] Assignment 12

Your Name

Due: May 6 (by 8:45am)

Implement logistic regression using two optimization methods: (i) Newton's method and (ii) gradient descent. Keep in mind we aim to minimize $G_n(\beta)$ and its 1st-order and 2nd-order derivatives are in the lecture note.

Please present your results in a figure and a table. Below are detailed instructions.

[1] In Figure 1, show two curves. Each curve represents the training error of one optimization method versus its number of updates. (So x-axis is number of updates and y-axis is classification error on training data.) In the figure, choose the maximum number of iterations yourself so both curves can be seen converged. Label the curves and coordinates properly.

In experiment, let both methods start from the same random initial model, i.e., the two curves should start at the same point. For gradient descent, choose the learning rate yourself so that this method could converge as fast as possible but still converge to a similar result to Newton's method. Finally, split the data yourself (e.g., 75%/25% for training/testing, or 20%/80% for training/testing) and fix it for both methods.

Fig. 1. Training Error versus Number of Updates

[2] After both methods converge, you will obtain two (different) optimal models. Show their training error and testing error in Table 1. (Keep in mind these are classification errors, not MSE.)

Table 1. Training Error and Testing Error

Optimizer	Training Error	Testing Error
Newton	?	?
Gradient	?	?

[3] (Bonus) We plan to train a logistic regression model from a set of instances, and release its prediction to the public in a way that anyone can query $p(y = 1 \mid x)$ from the model for any provided instance x.

Suppose there is an adversary, who aims to infer the presence of an arbitrary instance in our training set, by continually $p(y = 1 \mid x)$ of different instances from our model. Our training data is at risk!

Please protect ϵ-differential privacy of our training data using the Laplacian mechanism. We could do so by adding a noise to the query as $\tilde{p}(y = 1 \mid x) = p(y = 1 \mid x) + \eta$ where η is drawn from a Laplacian distribution.

(3.a) Calculate the sensitivity of our query function $p(y = 1 \mid x)$.[1]

(3.b) Show us the exact distribution that generates η, i.e., the values of μ and b in $Lap(\mu, b)$.[2]

(3.c) Implement your mechanism and draw a curve in Figure 2. This curve shows your testing error versus the privacy parameter ϵ. (Thus x-axis is ϵ and y-axis is testing error.[3]) Choose five values of ϵ yourself so that your curve can tell a comprehensive story of our model behavior under different ϵ.

Note: in this task, you can run logistic regression using either your own code or any Python library.

Fig. 2. Training Error versus Number of Updates

[1] Its input is x and its output is a probability that $y = 1$.

[2] Your distribution will still be characterized by the privacy parameter ϵ.

[3] Keep in mind that classification is now done based on $\tilde{p}(y = 1 \mid x)$ instead of $p(y = 1 \mid x)$.

[4] (Bonus). When implementing logistic regression using Newton's method, we may face a problem that the Hessian matrix is singular. One solution is to use the following alternative update rule:

$$\beta_{new} \leftarrow \beta_{old} + (G''_n(\beta_{old}) + 2\lambda I_0)^{-1}(G'_n(\beta_{old}) + 2\lambda I_0 \beta), \tag{1}$$

where λ is a regularization coefficient and I_0 is same as the one in ridge regression[4].

Well, formula (1) is not a heuristic.

(4.a) Please show (1) is solution to the regularized logistic regression problem, which aims to minimize

$$\tilde{G}_n = G_n + \lambda\beta^T I_0 \beta = \sum_{i=1}^{n}(1 - y_i)(x_i^T\beta) - \log(1 + \exp(x_i^T\beta)) + \lambda\beta^T I_0 \beta. \tag{2}$$

(4.b) Please further prove that minimizing (2) is based on the MAP estimate of β with a proper prior of β. (Unlike the standard update rule which is based on the MLE estimate of β.)

[4] Here, we assume β also contains a bias term and thus I_0 contains a zero diagonal element.