ECEN 765 Programming Assignment2
Liuyi Jin 225009797

Project 2 Optimization & Linear Models

This assignment aims to provide different logistic regression models to accurately predict data labels with given raw data. I used python to implement different algorithms.

1. **Implement the perceptron algorithm for logistic regression**

Logistic regression is not a generative model, it is based on the linear combination of feature variables. Since we have our label domain as {1, -1}, so we generate the prediction model as

$$P(c = 1|\boldsymbol{x}) = \delta(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$$

Where $\boldsymbol{\beta}$ is the weight vector on each feature vector. Similarly, we also have

$$P(c = -1|\boldsymbol{x}) = \delta(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$$

Then we need to compute the sign of log-odds ratio

$$\log \frac{P(c = 1|\boldsymbol{x})}{P(c = -1|\boldsymbol{x})}$$

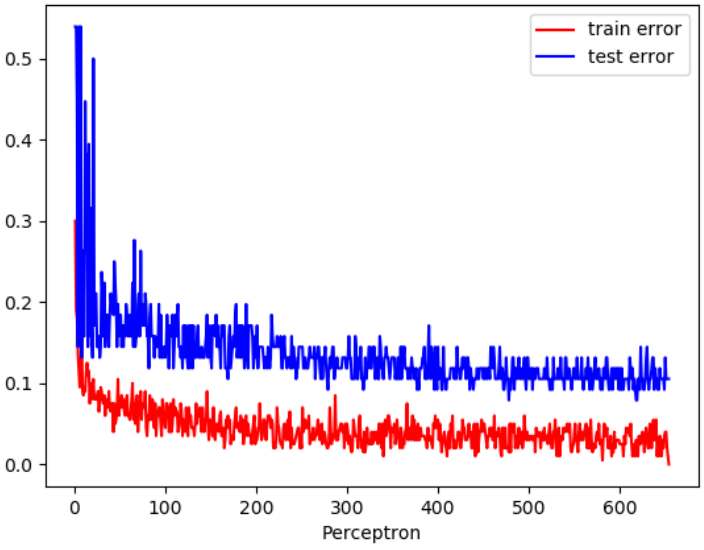
Which can further be reduced to compute the sign of $\boldsymbol{\beta}^T \boldsymbol{x}$. As we use perceptron algorithms to update the weight vector, we compute

$$\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^n + c^i \boldsymbol{x}^i \quad if \ \ sign\left(\boldsymbol{\beta}^{i^T} \boldsymbol{x}\right) \neq c^i$$

I gave results of my implementation in the following.

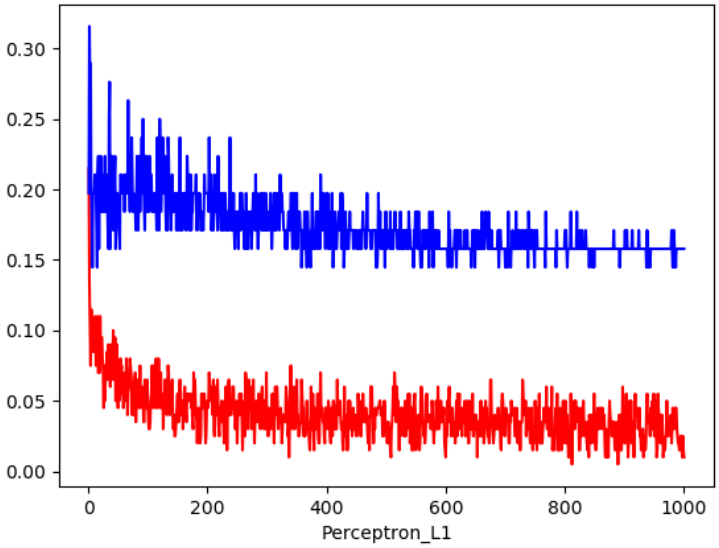
(1) use the raw data, unit L1 norm data, also L2 norm data to run the logistic regression

Raw data: training error 0.0, test error 0.115

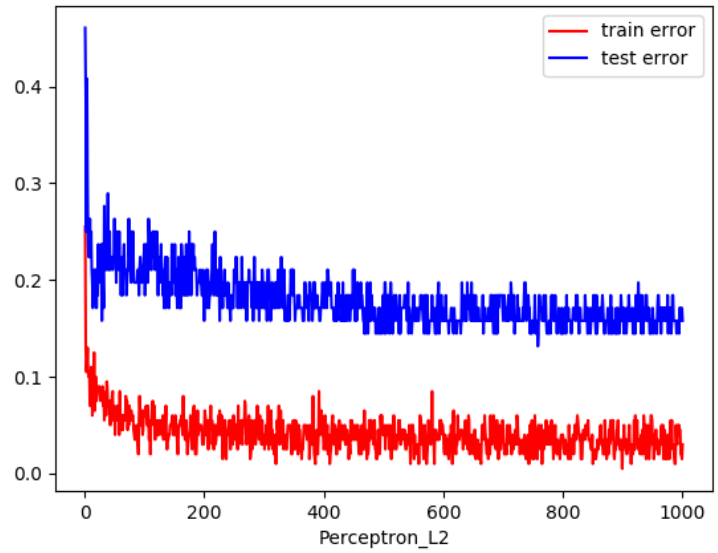


L1 norm data: training error 0.01, test error 0.168

ECEN 765 Programming Assignment2
Liuyi Jin 225009797



L2 norm data: training error 0.03, test error 0.168



(2) modify the perceptron algorithms by using averaged weights

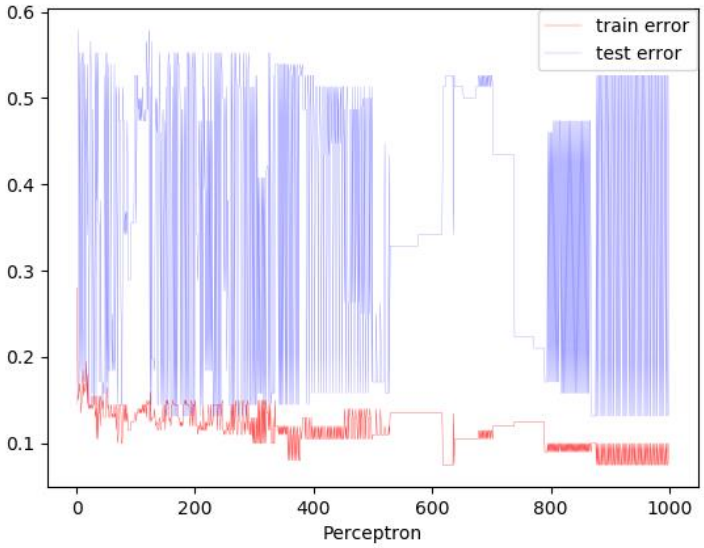
In this section, we used the averaged weights over the number of iterations for the next iteration.

$$\boldsymbol{\beta}^{n+1} = \frac{1}{N}\sum_{iter}^{N} \boldsymbol{\beta}^{iter}$$

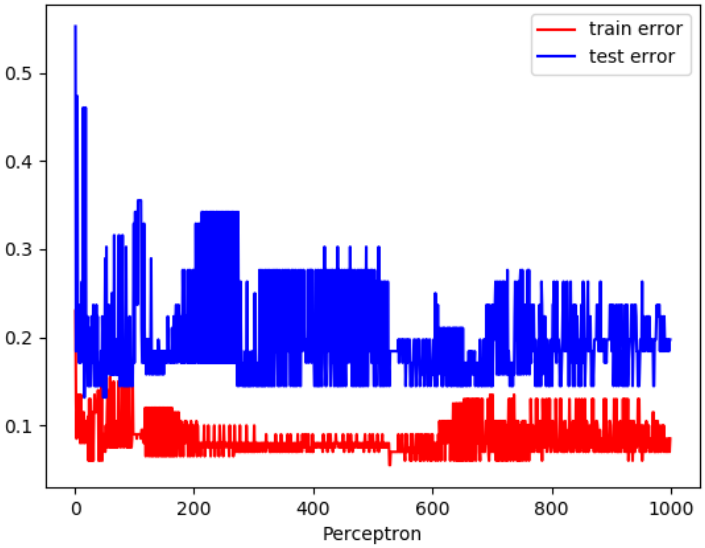
In this case, we also list the results as you see in the following graphs

Raw data: training error 0.1, test error 0.14

ECEN 765 Programming Assignment2
Liuyi Jin 225009797

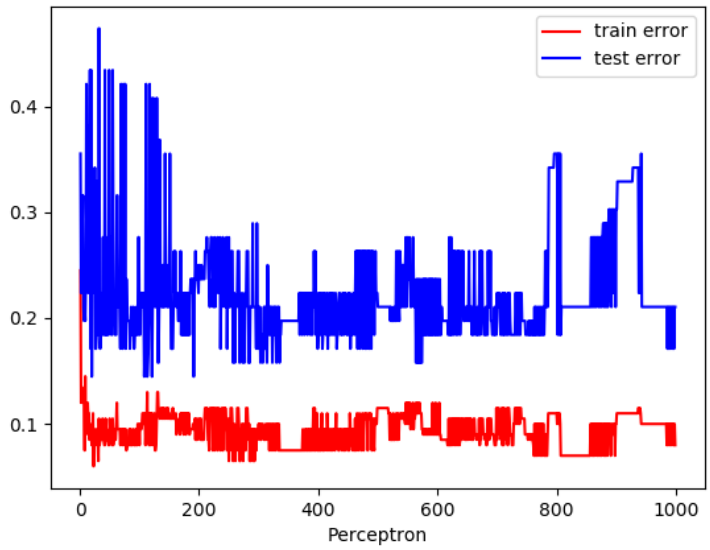


L1 norm data: training error 0.085, test error 0.2



L2 norm data: training error 0.08, test error 0.18

ECEN 765 Programming Assignment2
Liuyi Jin 225009797



(3) compare two versions of perceptron algorithms

(a) visually, the perceptron algorithms with averaged algorithms displayed more variations during the iterations. This means that in the learning process, the error rate does not keep decreasing. Instead, the error rate can be extremely higher for certain cases. I'd say that the averaged weights processes take the previous weights into considerations, which might have contradictory effect on achieving the lower error rates. In certain iterations, the adverse effect may larger than the favorable one. This should be the reason why the perceptron algorithm with average weights give much unstable results

(b) In terms of the final results, the perceptron algorithms without averaged weights are more accurate since the training error rates and test error rates are lower. This is easier to understand. In the perceptron learning process, once the weight has been updated in certain iteration, then this weight should be more suitable to be used to predict our labels. However, the averaged weights always took the previously generated less suitable weights into calculations. This may explain why the prediction accuracy of perceptron algorithms without averaged weights are always higher

2. **Implement the perceptron algorithm for logistic regression**

The fundamental update algorithm is

$$\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^{n} - \varepsilon \nabla_{\boldsymbol{\beta}} f(\boldsymbol{X})$$

Since we are using the logistic regression, we set our training and test label domain as {0,1}, and compute the first derivative of likelihood function as

$$\nabla_{\boldsymbol{\beta}} f(\boldsymbol{X^i}) = \sum_{i=1}^{N} \left(y^i - \sigma(\boldsymbol{\beta}^{n^T} \boldsymbol{X^i}) \right) \boldsymbol{X^i}$$

ECEN 765 Programming Assignment2
Liuyi Jin 225009797

With the update scheme specified, we then use the raw data, L1 norm data and L2 norm data. In L1 preprocess scheme, for each element $\boldsymbol{X}_j^i$ of each input feature $\boldsymbol{X}^i$,

$$\frac{1}{N}\sum_{i=1}^{N}\left|\boldsymbol{X}_j^i-\mu\right|=\sigma$$

$$\boldsymbol{X}_j^i|_{L1}=\frac{\boldsymbol{X}_j^i-\mu}{\sigma}$$

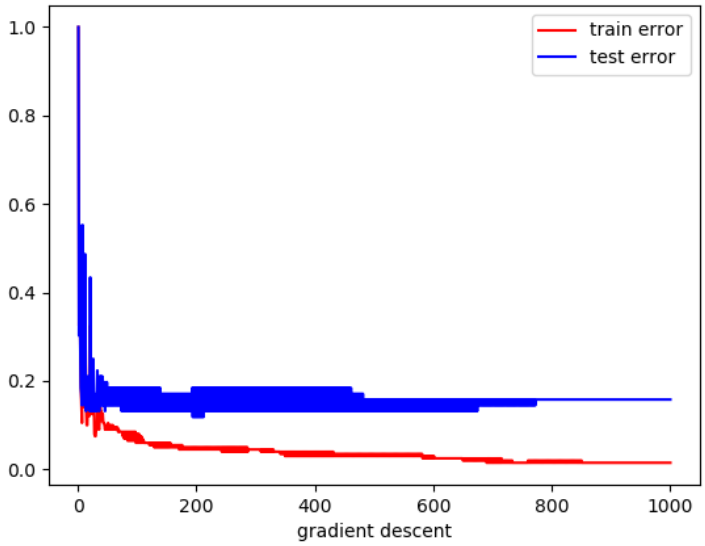
Where N is the number of inputs, $\boldsymbol{X}_j^i$ refers to the j-th element of i-th input feature. Similarly, we preprocess using L2 norm, for each element $\boldsymbol{X}_j^i$ of each input feature $\boldsymbol{X}^i$,

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{X}_j^i-\mu\right)^2}=\sigma$$

$$\boldsymbol{X}_j^i|_{L1}=\frac{\boldsymbol{X}_j^i-\mu}{\sigma}$$

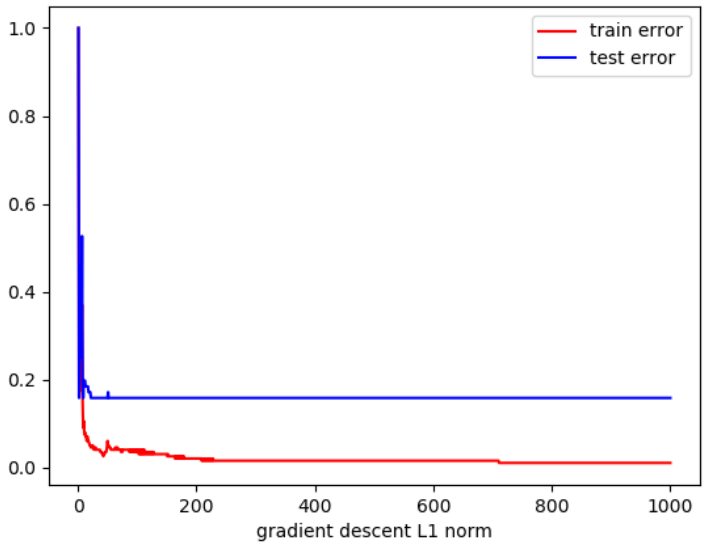
(1) Plot the results: the following graphs are the error rates versus iterations.

Raw data: training error 0.01, test error 0.158

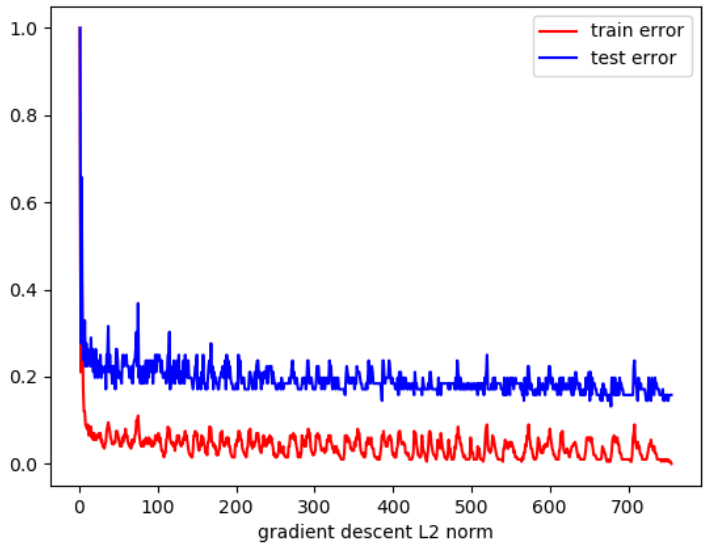


L1 norm data: training 0.01, test error 0.158

ECEN 765 Programming Assignment2
Liuyi Jin 225009797



L2 norm data: training 0.0, test error 0.158



(2) Compare the results with perceptron algorithm
   (a) With the gradient descent algorithm adopted, it's appreciable to see that we have faster convergence speed. From the graphs in the gradient descent section, the iteration converges to a close optimal solution in less than 50 steps. This is due to the differences between the optimization principles of gradient descent and perceptron. In perceptron algorithm, the hyperplane may rotate in the feature space with respect to any incoming inputs, the update does not guarantee any monotonic decrease in the error rates. However, in the gradient descent, the algorithm was developed based on minimizing the error rates, so it can guarantee decreasing error rates over iterations.
   (b) The variations of perceptron algorithms are much larger than that in gradient descent algorithm. This can also trace back to the fundamental principles of the differences between perceptron and gradient descent.

ECEN 765 Programming Assignment2
Liuyi Jin 225009797

3. Locally weighted logistic regression

The updates are implemented using Newton's method with the following scheme

$$\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^{n} - \varepsilon \boldsymbol{H}^{-1} \nabla_{\boldsymbol{\beta}} f(\boldsymbol{X})$$

(1) Compute the 1st derivative and the 2nd derivative of the likelihood function of logistic regression.

$$\nabla_{\beta} l(\beta) = \sum_{i=1}^{N} w^{i} \{\sigma(\boldsymbol{\beta}^{T} \boldsymbol{x}^{i}) - y^{i}\} \boldsymbol{x}^{i} - 2\lambda \boldsymbol{\beta}$$

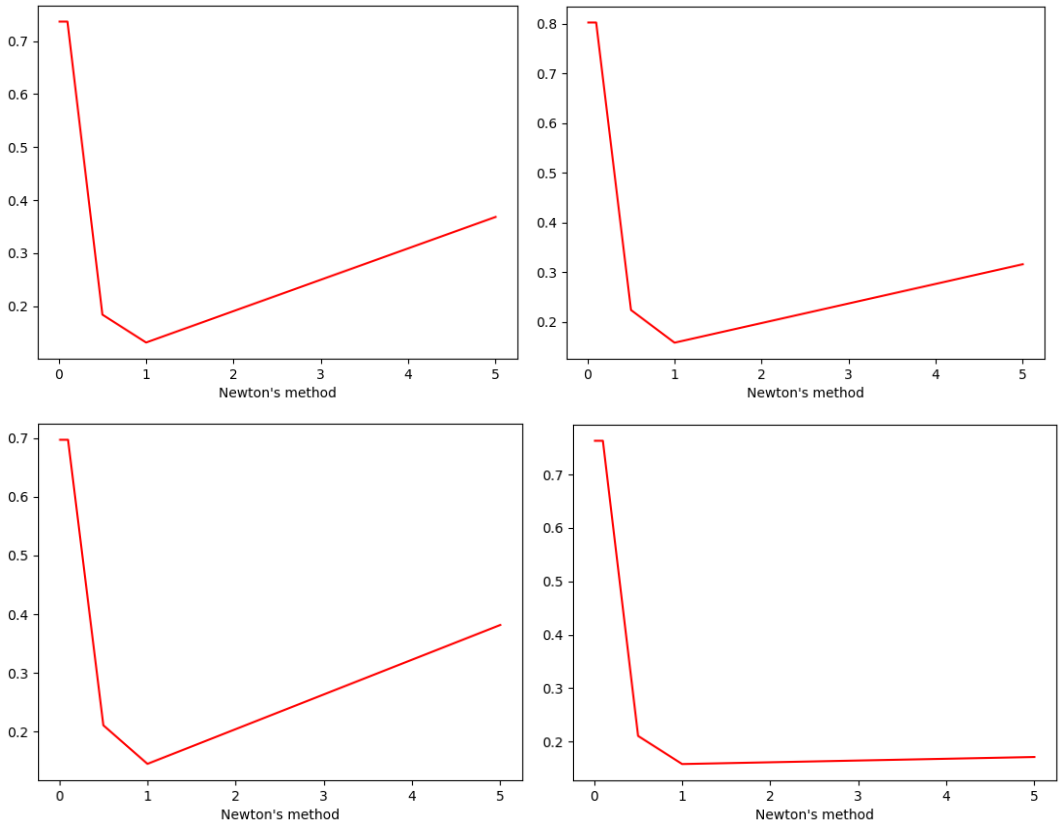
$$H = \nabla_{\beta} [\nabla_{\beta} l(\beta)] = \sum_{i=1}^{N} w^{i} \left\{\sigma(\boldsymbol{\beta}^{T} \boldsymbol{x}^{i}) [1 - \sigma(\boldsymbol{\beta}^{T} \boldsymbol{x}^{i})] \boldsymbol{x}^{i} \boldsymbol{x}^{i^{T}}\right\} - 2\lambda \boldsymbol{I}$$

(2) Compute the w^is for each test sample. Here for each test sample, we compute its weight with respect to each training sample using

$$w^{i} = \exp\left(-\frac{\|x - x^{i}\|_{l2}^{2}}{2\tau^{2}}\right)$$

(3) Plot the error rates with respect to τ, and compare them with the ones obtained in **1**

In the process of comparing the effect of τ on the accuracy, I noticed that the initialization of β affected the accuracy. However, the normal shape of error rates versus τ should decrease until somewhere τ = 0.5-1.0, and then the error rates increase as the τ keeps increasing. I had tested four initializations of β.



As shown in the four graphs above, we see that $\tau = 1$ can achieve the best performance, which has similar error rates level with that of perceptron test error rates.