CS5590 – Foundation of Machine Learning
Assignment - 4

Tushar K Raysad (BM21MTECH14004)

1. <u>Non-Uniform Weights in Linear Regression:</u>

You are given a dataset in which the data points are denoted by (x_n, t_n), $n = 1, \cdots, N$. Each data point is associated with a non-negative weighting factor $g_n > 0$. The error function is thus modified to:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} g_n \left(t_n - \mathbf{w}^T \Phi(\mathbf{x}_n) \right)^2$$

where $\Phi(\cdot)$ is any representation of the data.

(a) Find an expression for the solution w^* that minimises the above error function.

(b) Give two alternative interpretations of the above weighted sum-of-squares error function in terms of: (i) data-dependent noise variance and (ii) replicated data points.

→ Given Error Function is,

$$E_D(W) = \frac{1}{2} \sum_{n=1}^{N} g_n \left(t_n - w^T \phi(x_n)\right)^2$$

By derivating the above equation with respect to w, we get

$$\nabla E_D(W) = \sum_{n=1}^{N} g_n \left(t_n - w^T \phi(x_n)\right) \phi(x_n)^T$$

Equating the above equation to 0,

$$\sum_{n=1}^{N} g_n t_n \phi(x_n)^T - w^T \left(\sum_{n=1}^{N} g_n \phi(x_n) \phi(x_n)^T\right) = 0$$

Let us take $\sqrt{g_n}\,\phi(x_n) = \phi_1(x_n)$ and

$$\sqrt{g_n}\, t_n = t_n'$$

Substituting these values, we get

$$\sum_{n=1}^{N} t_n' \phi_1(x_n)^T - w^T \left(\sum_{n=1}^{N} \phi_1(x_n)\, \phi_1(x_n)^T\right) = 0.$$

$$\sum_{n=1}^{N} t_n' \phi_1(x_n)^T = w^T \left(\sum_{n=1}^{N} \phi_1(x_n)\, \phi_1(x_n)^T\right) = 0$$

Solving for w^*, we get

$$w^* = (\phi^T \phi)^{-1} \phi^T t$$

where we have t as

$$t = \left(\sqrt{g_1}\, t_1, \sqrt{g_2}\, t_2, \ldots \ldots \sqrt{g_n}\, t_n\right)$$

and ϕ as a $N \times M$ matrix with element

$$\phi(i,j) = \sqrt{g_i}\, \phi_j(x_i)$$

$$\phi = \sqrt{g_n} \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & & & \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

i) Data - dependent noise variance :-

We have dataset where data points are denoted by (x_n, t_n), $n = 1, \ldots N$.

The likelihood function expression for the above data, we get

$$p\left(t / X, w, \underset{g_n}{\beta}\right) = \sum_{n=1}^{N} N\left(t_n / w^T \phi(x_n), \beta^{-1} \cdot g_n\right)$$

w and β are functions of adjustable parameters.

g_n is non negative weighting factor.

Taking logarithm of likelihood function,

$$\ln p(t/w, \beta, g_n) = \sum_{n=1}^{N} \ln \mathcal{N}\left(t_n / w^T \phi(x_n), g_n \beta^{-1}\right)$$

Taking sum of squared error function, we get,

$$E_D(w) = \frac{1}{2} \sum_{n=1}^{N} g_n \left(t_n - w^T \phi(x_n)\right)^2 .$$

ii) Replicated Data points!

For replicated data points g_n can be percieved as effective number of observation of dataset (x_n, t_n). In alternative way, we can treat the dataset (x_n, t_n) as the datapoints repeatedly occuring for g_n number of times.

2. <u>Bayes Optimal Classifier:</u>

Let there be 5 hypotheses h_1 through h_5 that could guide a robot to move either Forward(F) or Left(L) or Right(R):

$P(h_i\|D)$	$P(F\|h_i)$	$P(L\|h_i)$	$P(R\|h_i)$
0.4	1	0	0
0.2	0	1	0
0.1	0	0	1
0.1	0	1	0
0.2	0	1	0

Compute the MAP estimate and Bayes optimal estimate using the data provided in the table. Are they the same? Justify your answer.

→ Bayes Optimal Classifier:-

To find Bayes Optimal Estimate:-

For Forward turn:-

$$\sum_{h_i \in H} P(F/h_i)\, P(h_i/D) = 0.4$$

For Left turn:-

$$\sum_{h_i \in H} P(L/h_i)\, P(h_i/D) = 0.2 + 0.1 + 0.2 = 0.5$$

For Right turn:-

$$\sum_{h_i \in H} P(R/h_i)\, P(h_i/D) = 0.1$$

As Bayes optimal is maximum for left turn the MAP estimate recommends the Robot turn left.

MAP estimate:-

Forward:-

$\text{argmax}\ (0.4 \times 1) = 0.4$

Left:-

$\text{argmax}\ (0.2 \times 1,\ 0.1 \times 1,\ 0.2 \times 1) = 0.2$

Right:-

$\text{argmax}\ (0.1 \times 1) = 0.1$

MAP hypothesis suggests the Robot moves forward.

Therefore both hypothesis are not same.

3. <u>VC-Dimension:</u>

Consider a data setup of one-dimensional data $\in R^1$, where the hypothesis space H is parametrized by {p,q} where x is classified as 1 iff $p < x < q$. Find the VC-dimension of H.

→ Consider the two element set

$M = \{3.5, 4.5\}$.

The dichotomies of the set M is shown in the table below:

X	3.5	4.5		X	3.5	4.5
label	0 (i)	0		label	0 (ii)	1
X	3.5	4.5		X	3.5	4.5
label	1 (iii)	0		label	1 (iv)	1.

It can be inferred that hypothesis $h_{5,6}$ is consistent with (i), $h_{4,5}$ is with (ii), $h_{3,4}$ with (iii) and $h_{3,5}$ with (iv) respectively. Therefore the set M is shattered by H.

Now let us consider a 3 element subset $G = (x_1, x_2, x_3)$. And let us assume that $x_1 < x_2 < x_3$. Hypothesis H can't be shattered by G because the dichotomy represented by set (x_1, x_3) can't be represented by a hypothesis H as any interval containing both x_1 and x_3 will also have x_2.

Therefore the size of the largest subset x shattered by H is 2.

Therefore $VC(H) = 2$.

4. <u>Regularizer:</u>

Given D-dimensional data $x = [x_1, x_2, \cdots, x_D]$, consider a linear model of the form:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{k=1}^{D} w_k x_k$$

Now, for N such data samples with their corresponding labels (x_i, t_i), $i = 1, 2, \cdots, N$, the

sum-of-squares error (or mean-squared-error) function is given by:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \Big(y(\mathbf{x}_i, \mathbf{w}) - t_i \Big)^2$$

Now, suppose that Gaussian noise $\varepsilon_k \sim N(0, \sigma^2)$ (i.e. zero mean and variance σ^2) is added independently to each of the input variables x_k. Find a relation between: minimizing the above sum-of-squares error averaged over the noisy data, and minimizing the standard sum- of-squares error (averaged over noise-free input data) with a L_2 weight-decay regularization term, in which the bias parameter w_0 is omitted from the regularizer.

→ Given Error Function :-

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} \left(y(x_i, w) - t_i \right)^2$$

and $y(x, w) = w_0 + \sum_{k=1}^{D} w_k x_k$

Substituting $y(x, w)$ in error function,

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} \left(w_0 + \sum_{k=1}^{D} w_k x_k - t_i \right)^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left[\left(w_0 + \sum_{k=1}^{D} w_k x_k \right) - t_i + \sum_{k=1}^{D} w_k L_{2k} \right]^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left[y(x_i, w) - t_i + \sum_{k=1}^{D} w_k L_{2k} \right]^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left[\left(y(x_i, w) - t_i \right)^2 + 2 \left(y(x_i, w) - t_i \right) \left(\sum_{k=1}^{D} w_k L_{2k} \right) + \left(\sum_{k=1}^{D} w_k L_{2k} \right)^2 \right] \quad \text{——— (1)}$$

Let us take the third term of (1) and calculate expected value of the term with respect to L_2,

$$E_{L_2}\left[\sum_{k=1}^{D} w_k L_{2k}\right]^2 = E_{L_2}\left[\sum_{k=1}^{D}\sum_{j=1}^{D} w_k w_j L_{2k} L_{2j}\right]$$

$$= \sum_{k=1}^{D}\sum_{j=1}^{D} w_k w_j E_{L_2}(L_{2k} L_{2j})$$

$$= \sigma^2 \sum_{k=1}^{D}\sum_{j=1}^{D} w_k w_j \delta_{kj}$$

$$= \sigma^2 \sum_{k=1}^{D} w_k^2.$$

Now Let us take the second term and calculate expected value of the term w.r.t L_2.

$$E_{L_2}\left[2\left(\sum_{k=1}^{D} w_k L_{2k}\right)(y(x_i,w)-t_i)\right]$$

$$= 2\left(y(x_i,w)-t_i\right) E_{L_2}\left(\sum_{k=1}^{D} w_k L_{2k}\right)$$

$$= 0.$$

Therefore upon calculating expectation of error function $E(w)$ w.r.t L_2, we get

$$E_{L_2}(E(w)) = \frac{1}{2}\sum_{i=1}^{N}\left(y(x_i,w)-t_i\right)^2 + \frac{\sigma^2}{2}\sum_{k=1}^{D} w_k^2.$$

5. ***Logistic Regression:***

(a) Implement your own code for a logistic regression classifier, which is trained using gradient descent and cross-entropy error as the error function.

Index	x_1	x_2	y
1	0.346	0.780	0
2	0.303	0.439	0
3	0.358	0.729	0
4	0.602	0.863	1
5	0.790	0.753	1
6	0.611	0.965	1

Table 1: Train Set

Index	x_1	x_2	y
1	0.959	0.382	0
2	0.750	0.306	0
3	0.395	0.760	0
4	0.823	0.764	1
5	0.761	0.874	1
6	0.844	0.435	1

Table 2: Test Set

(b) Consider the training set and test set given in Tables 1 and 2. We use the linear model $f_\theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ and the logistic regression function $\sigma(f_\theta(x_1, x_2)) = \frac{1}{1+\exp^{-f_\theta(x_1,x_2)}}$. Consider the initial weights as $\theta_0 = -1$, $\theta_1 = 1.5$, $\theta_2 = 0.5$, and learning rate as 0.1 (for gradient descent).

i. ***(1 mark)*** What is the logistic model $P(\hat{y} = 1|x_1, x_2)$ and its cross-entropy error function?

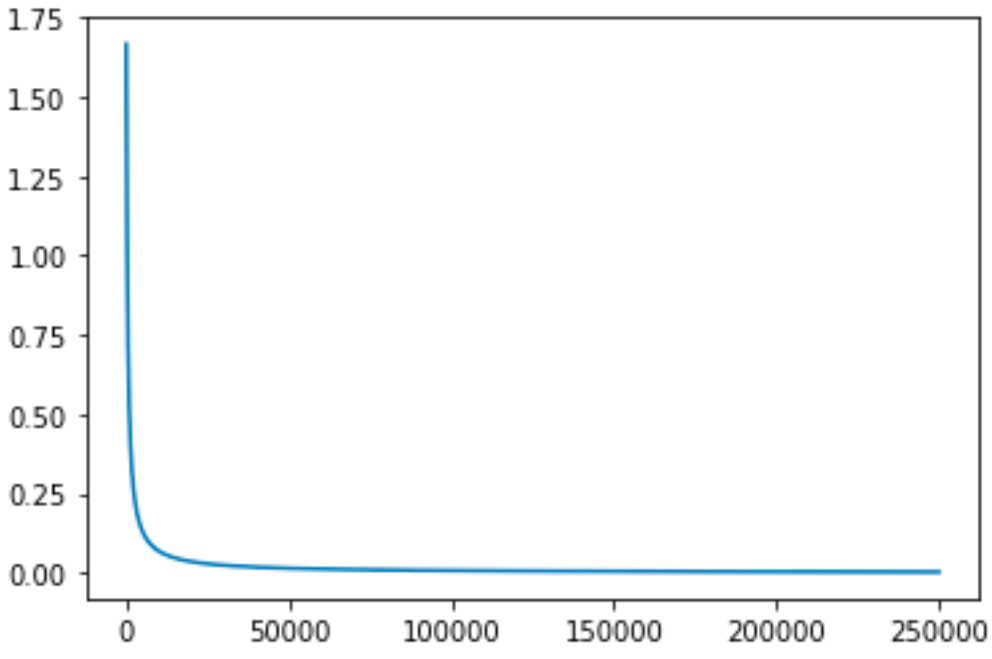
ii. ***(1 mark)*** Use gradient descent to update θ_0, θ_1, θ_2 for one iteration. Write down the updated logistic regression model.

iii. ***(2 mark)*** At convergence of gradient descent, use the model to make predictions for all the samples in the test dataset. Calculate and report the accuracy, precision and recall to evaluate this model.

1. The logistic model P(y=1|x1,x2) is: 0.5936978427941283

Cross Entropy Error Function after 0 iteration is: 1.6669074269981987
Cross Entropy Error Function after 1 iteration is: 1.6630660083987943
Cross Entropy Error Function after 25000 iteration is: 0.028677517117260964
Cross Entropy Error Function after 50000 iteration is: 0.014598707352406442
Cross Entropy Error Function after 75000 iteration is: 0.009805373925452006
Cross Entropy Error Function after 100000 iteration is: 0.007386167210650619
Cross Entropy Error Function after 125000 iteration is: 0.005926471111481545
Cross Entropy Error Function after 150000 iteration is: 0.00494957689906003
Cross Entropy Error Function after 175000 iteration is: 0.004249788296695725
Cross Entropy Error Function after 200000 iteration is: 0.003723764160927053
Cross Entropy Error Function after 225000 iteration is: 0.0033138849426556415

Number of iterations vs Cost :



2. The updated logistic regression model after 1 iteration:

W = [1.53612616 , 0.51739964]

B = -1.0117732408457443

3. The accuracy, precision and recall at convergence of gradient descent:

Accuracy of the model is : 0.667

	precision	recall
0	1.00	0.33
1	0.60	1.00

6. ***Kaggle - Taxi Fare Prediction:***

The next task of this assignment is to work on a (completed) Kaggle challenge on taxi fare prediction. As part of this task, please visit https://www.kaggle.com/c/new-york-city-taxi-fare-prediction to know more about this problem, and download the data. (You now know how to download data from Kaggle.)

You are allowed to use any machine learning library of your choice: scikitlearn, pandas, Weka (we recommend scikitlearn), and any regression method too. Use train.csv to train your classifier. Predict the fares on the data in test.csv, and report your best 2 scores in your report. (We will also upload your codes randomly to confirm the scores.)

– The models I used in this Kaggle Challenge are:

1. ***Linear Regression***: For this Regression method I got the Root Mean Square Error (RMSE) of **7.57483647462791**. After uploading the CSV file to Kaggle I got the private score and public score as **7.39761**.

2. ***XG Boost Regressor***: For this Regression method I used hyper parameters such as " objective='reg:squarederror', n_jobs=-1, random_state=42, n_estimators=500, max_depth=5, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8 " and got the RMSE of **3.7789418711790015**. And the private score and public score for this model on kaggle is **3.31088**. This model is one among the best achieved models that I have used for this challenge.

3. ***Light GBM*** : For this Regression method I used hyper parameters such as " 'learning_rate':0.05, 'application':'regression', 'max_depth':7, 'num_leaves':200, 'verbosity':-1, 'metric':'RMSE' " and got the RMSE of **3.7491920085977934**. And the private score and public score for this model on kaggle is **3.36108**. This model is also one among the best 2 achieved models that I have used for this challenge.

4. ***Gradient Boosting Regressor:*** For this Regression method I got the Root Mean Square Error (RMSE) of **3.909091383336689**. After uploading the CSV file to Kaggle I got the private score and public score as **3.59530**.

5. ***Random Forest Regressor***: For this Regression method I used hyper parameters such as " max_depth=10, n_jobs=-1, random_state=7, n_estimators=50 " and got the RMSE of **3.887173648487748**. And the private score and public score for this model on kaggle is **3.57386**.

- **LightGBM is a boosting tree based algorithm. The LightGBM algorithm grows leaf wise instead of level wise. And the algorithm uses the node which will result in maximum delta loss to split. And also LightGBM is very fast, takes less RAM to run and focuses on the accuracy of the result.**

- **XG Boost also uses boosting to train the dataset. It uses a few parameters that dramatically affect the accuracy and training speed. The key difference in reduced speed of XGBoost compared to LightGBM is because it splits the tree nodes one level at a time while LightGBM does that one node at a time.**