ML Assignment 1

Nikita Mehrotra, PhD18013

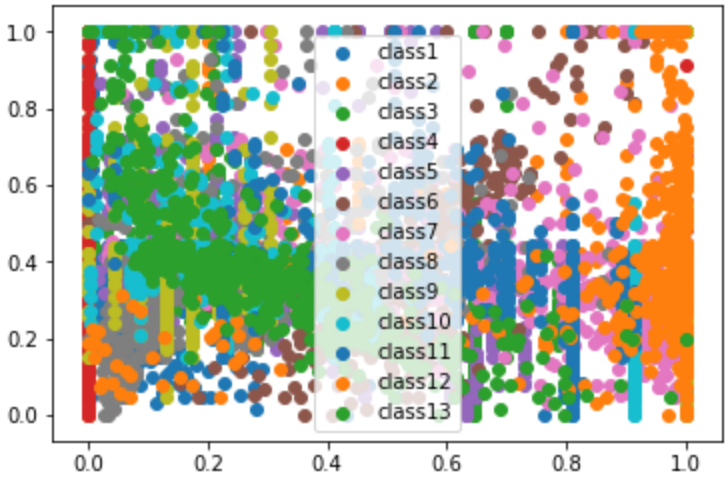
Solution 1: **Linear Regression**

(i) I have implemented Non Regularized Multivariate Linear Regression on Boston Housing Dataset. The dataset contains 13 features and we have to predict the housing prices using those features. For handling (manipulation and analysis of) the dataset I have used Pandas library and for numerical computation I have used numpy software library. In the dataset all the features are on different scale, so I have used min-max scaling to bring down all parameters value between 0 and 1. Formula for min-max scaling is:-

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

List of functions implemented:

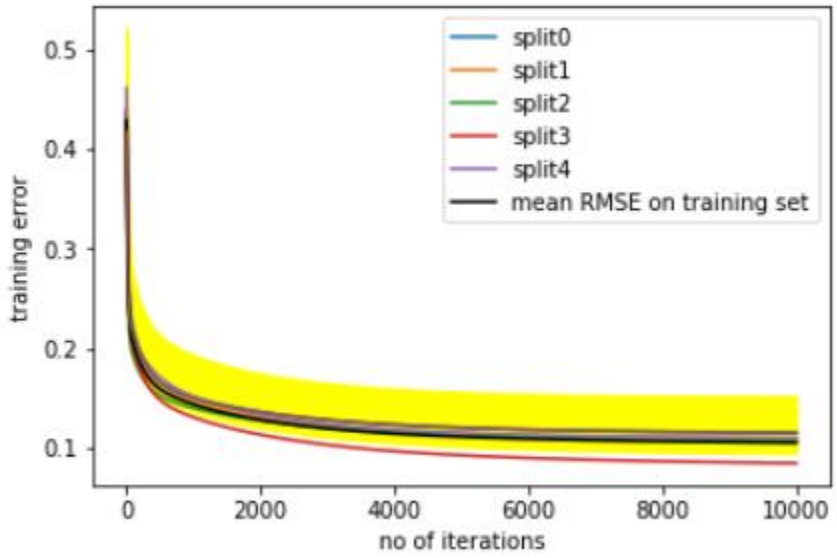
1. **hypothesis function**: This simply calculates our hypothesis i.e. predicts the prices of houses. The equation for hypothesis is :
$$y' = \theta^T x$$
2. **loss function**: This function calculates loss on our predicted regressor model. The equation for calculating loss is:
$$loss = \frac{\sum_{i=1}^{m}(y' - y)^2}{2m}$$
3. **grad_descent function**: This function implements batch gradient descent to update our parameters weight. The function is executed for 30000 iterations with a learning rate of 0.009. The iterations and learning rate are decided on the basis of trial and error. The update equation for gradient descent is:
$$\theta = \theta - \alpha(\nabla_\theta(loss))$$
4. **rmse function**: This function calculates root mean squared error on our predicted values. The formula for calculating root mean square error is :
$$rmse = \sqrt[2]{\frac{(y' - y)^2}{m}}$$
here m denotes the number of cases



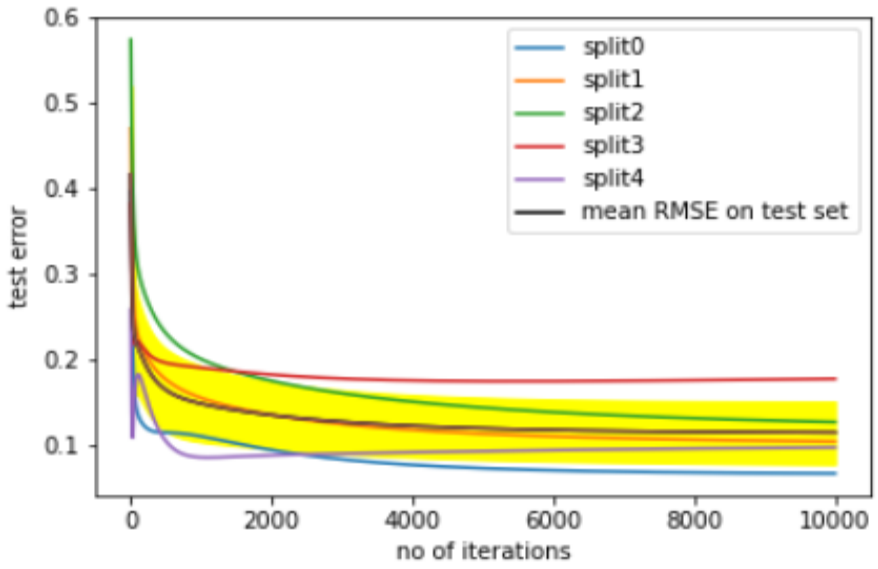
Visual representation of our data

Plots for non-regularized linear regression:

a. RMSE vs Gradient Descent iterations for training data:



b. RMSE vs Gradient Descent iterations for test data:



The above plots show RMSE error for each gradient descent iteration for training and test data. Mean RMSE is plotted in black while standard deviation is plotted with yellow showing the range in which our RMSE error can lie.

Table 1.Summary of RMSE values on test and train data for non-regularized linear regression

Split Number	RMSE of train set	μ±σ (training set)	RMSE of test set	μ±σ (test set)
0	0.114	0.13 ± 0.02	0.067	0.08 ± 0.02
1	0.110	0.12 ± 0.02	0.105	0.12 ± 0.02
2	0.107	0.12 ± 0.02	0.127	0.15 ± 0.03
3	0.084	0.10 ± 0.02	0.178	0.18 ± 0.01
4	0.110	0.12 ± 0.02	0.098	0.09 ± 0.01

(ii) **Regularized Linear Regression:** For regularized multivariate linear regression **grid search with 5 folds cross validation** is used to tune the hyper-parameter lambda, which penalizes our weights (or Θ) to reduce model complexity (or to prevent overfitting of the training data). Grid Search basically returns the best hyper-parameter value for the model from the grid of values.

<u>List of functions implemented:</u>

1. ridgegrd_search: This function implements grid search cv on L2 regularized regressor to tune our hyperparameter ƛ.
2. lassogrd_search: This is implementation of grid search cv on Lasso regression to find best ƛ.
3. l2grad_desc: This function implements Ridge regularization. The parameter update equation is:

$$\theta = \theta - \alpha(\nabla_\theta[(loss\ function) + \lambda\theta^2]$$

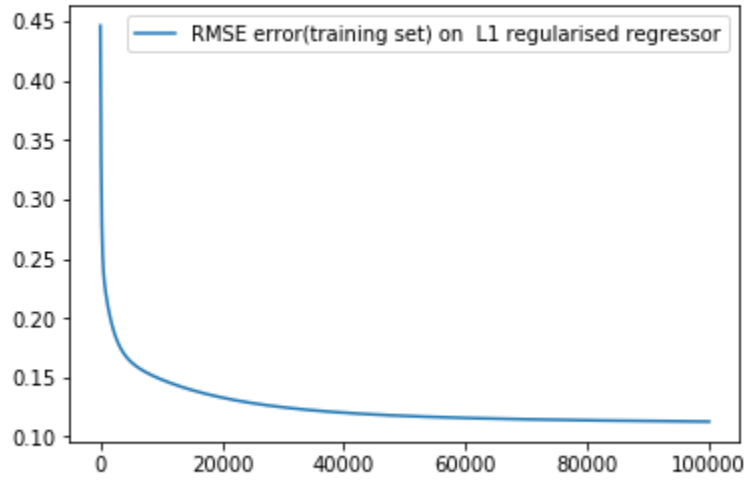
4. l1grad_desc This function implements Lasso regularization. The parameter update equation is :

$$\theta = \theta - \alpha(\nabla_\theta[(loss\ function) + \lambda\theta]$$

The plots for RMSE error in Lasso and Ridge regression are:

b. L1 regularized linear regression (Lasso)

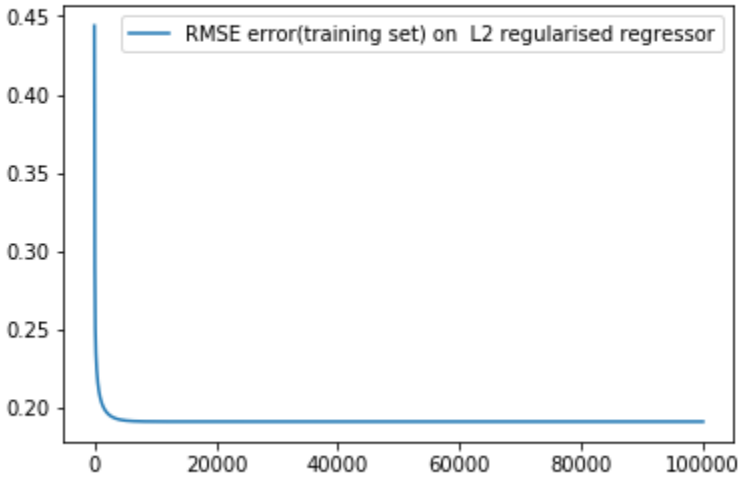
```
Mean RMSE on training set 0.1258768197747016
STD on training set 0.02219555950393447
```




```
RMSE error on Test Set: 0.06940336974050677
```

c. L2 regularized linear regression (Ridge)

```
Mean RMSE on training set 0.19149221812935077
STD on training set 0.005911095918420557
```

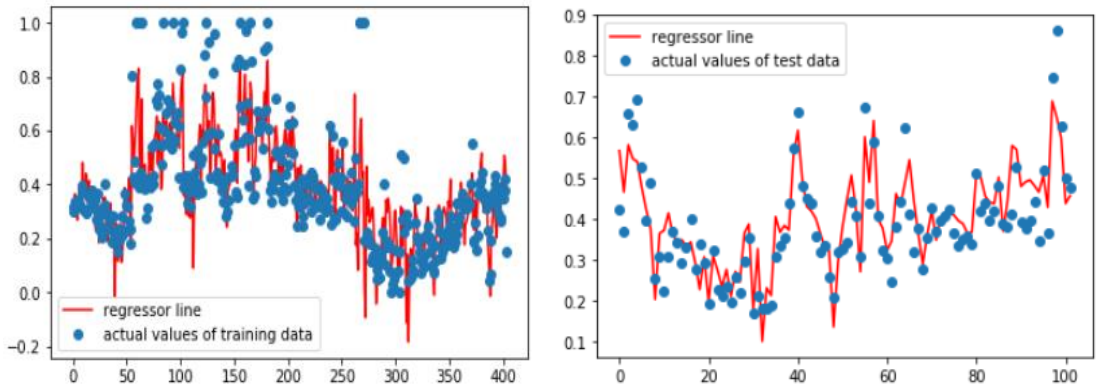



```
RMSE error on Test Set: 0.12214266978217964
```

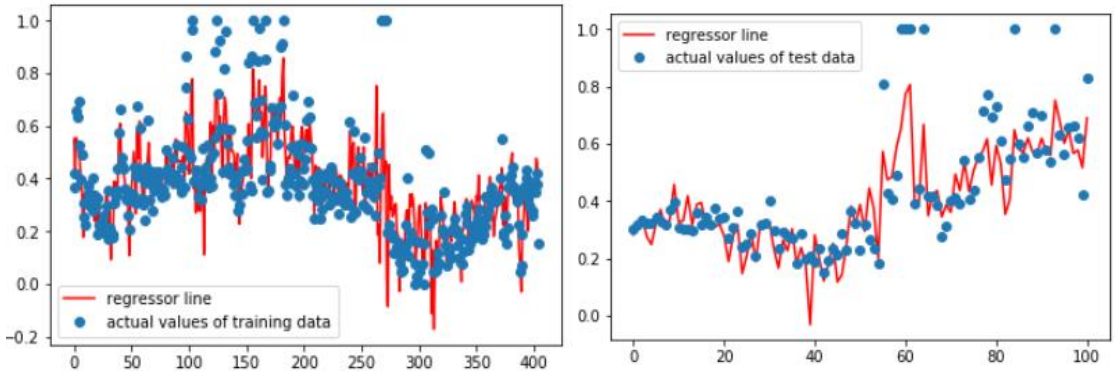
(iii) **Analysis:**

Regressor plot for non regularized linear regression:

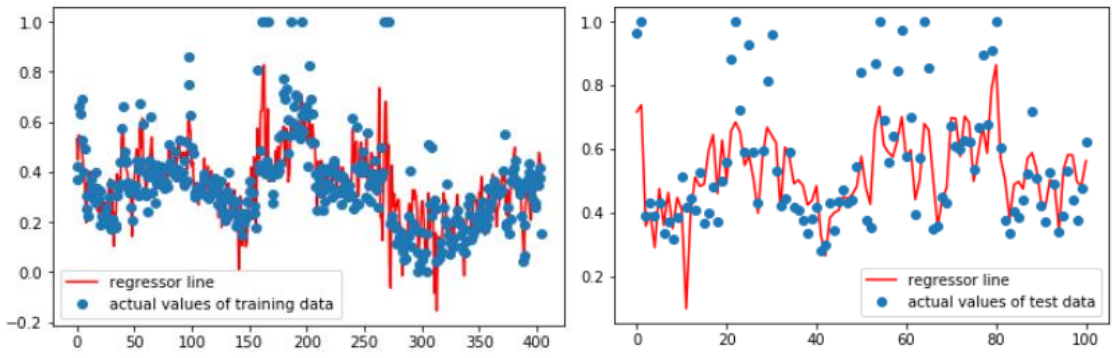
Split 1:



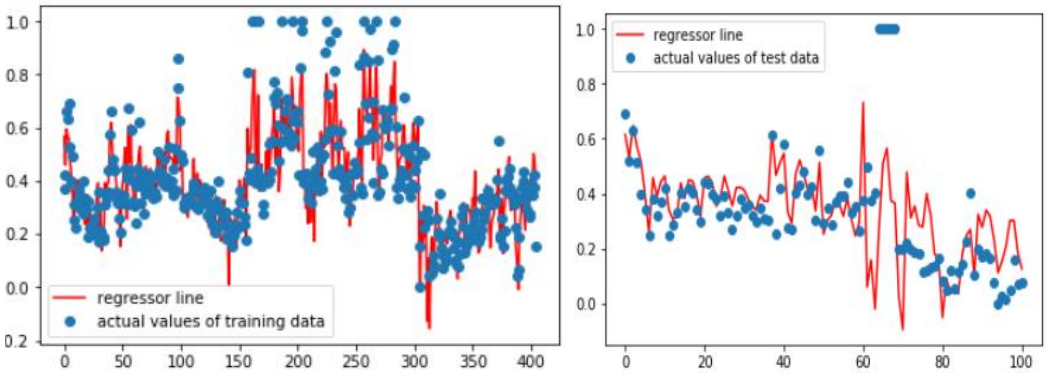
Split 2:



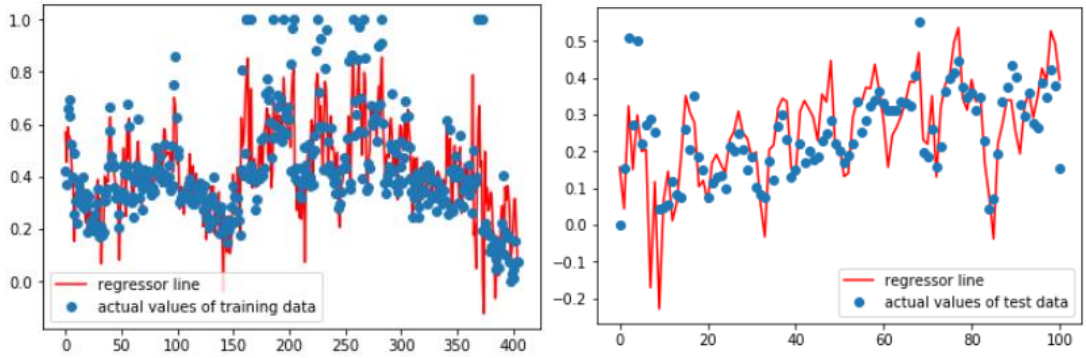
Split 3:



Split 4:

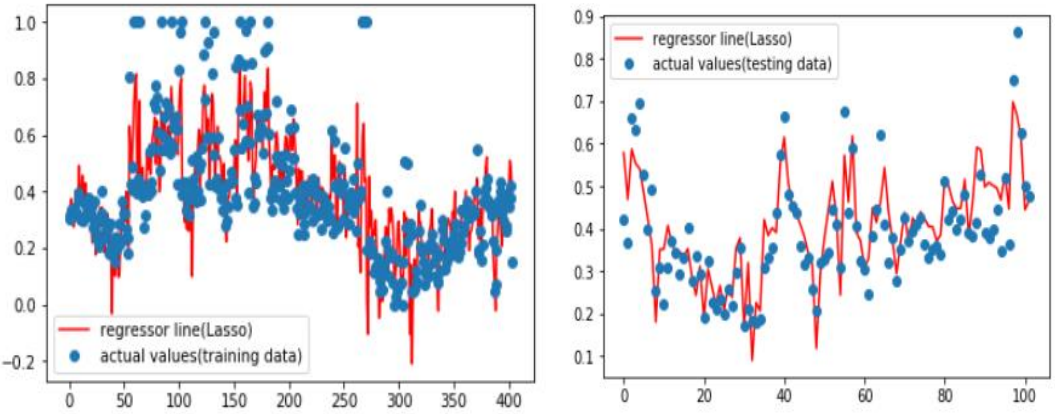


Split 5:



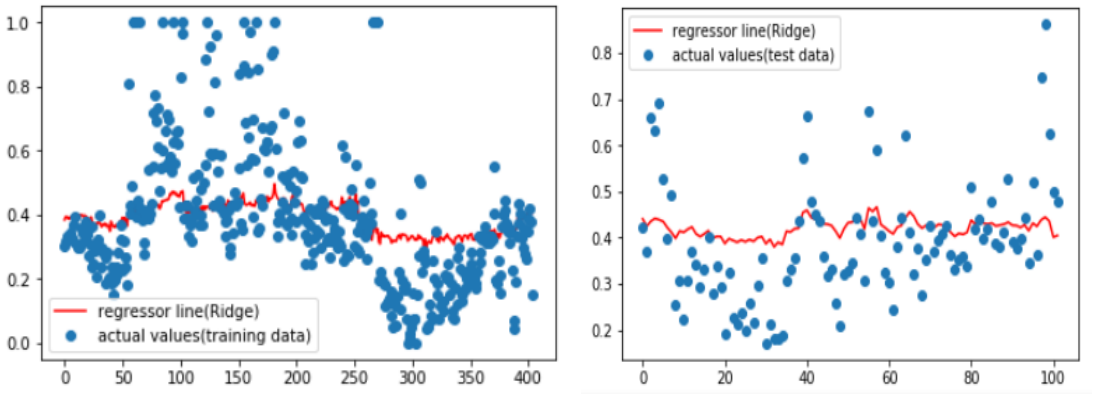
From the above plots of non–regularized linear regressor vs actual value, it can be seen that, the model is predicting our data accurately, as most of the points of actual value are lying on the regressor, so we can say that the model is a good fit for our data.

Regressor plot for L1 regularized linear regression:



In case of L1 regularized linear regression also, the regressor is predicting data accurately, and shape of model is nearly equal to non-regularized linear regression model. So we can say that all our features are required to accurately predict the data (none of them are correlated or contributing in overfitting of the data). Hence we can conclude that non regularized model will be a best fit for our data because there is no sense in applying regularization as the results are same.

Regressor plot for L2 regularized linear regression:



In case of L2 regularization, the model is under fitting our data due to high penalization of features.

Solution 2: **Logistic Regression**

Logistic regression is basically used to solve classification problem. Here we have used MNIST data set, in which we have to classify the handwritten digits. For classification we have used one versus rest approach to classify the digits in one of the 9 classes ([0, 1, 2, 3,...,9]).

List of functions:

1. l2_logisticregression: It is the implementation of L2 regularized logistic regression on MNIST data set.
2. l1_logisticregression: It is the implementation of L1 regularized logistic regression on MNIST data set.

The classification accuracy for each of the classes is shown below:-

(i) L1 Regularized:

Table 2. Accuracy score for training data

Class	**Accuracy Score**
0	97.82204963700826 %
1	97.77514090774251 %
2	90.75193017791206 %
3	89.92007829065405 %
4	93.6665525504964 %
5	86.95812580704667 %

6	96.43460628590739 %
7	93.71109337589785 %
8	88.32678174670995 %
9	89.86384266263238 %

Table 3. Accuracy score for test data

Class	Accuracy Score
0	98.06122448979592 %
1	98.23788546255507 %
2	87.6937984496124 %
3	90.99009900990099 %
4	92.56619144602851 %
5	84.97757847533633 %
6	94.88517745302714 %
7	92.21789883268482 %
8	87.88501026694045 %
9	89.49454905847374 %

(ii) L2 Regularized:

Table 4 Accuracy score for training data

Class	Accuracy Score
0	97.82204963700826 %
1	97.76030851379413 %
2	90.76871433366902 %
3	89.95269939650954 %
4	93.64943512495721 %
5	86.97657258808337 %
6	96.43460628590739 %
7	93.7270550678372 %
8	88.32678174670995 %
9	89.88065221045554 %

Table 5 Accuracy score for test data

Class	Accuracy Score
0	98.06122448979592 %
1	98.23788546255507 %
2	87.79069767441861 %
3	90.99009900990099 %
4	92.4643584521385 %
5	84.97757847533633 %
6	94.88517745302714 %
7	92.21789883268482 %

8	87.88501026694045 %
9	89.49454905847374 %

(iii) **Analysis:** By analyzing the accuracy score of above L1/L2 logistic classifier for both train and test data we can say that both model are good fit for our data.

Theory Questions

Page No.
Date : / /

3 logistic regression is a classification function defined as:

$$\sigma(f(x_i)) = \frac{1}{1+e^{-f(x_i)}}$$

where $f(x) = w^T x + b$ is the set of predicted values.

The sigmoid function squashes the o/p to be in between 0 & 1

$$\sigma(f(x_i)) = \begin{cases} \geq 0.5 & y_i(\text{class}) = 1 \\ < 0.5 & y_i(\text{class}) = 0 \end{cases}$$

Logistic regressor uses linear regression by adding some non linearity using sigmoid function. In this regression there are one or more independent input variables that determine outcome (or class label) with dichotomous variable.

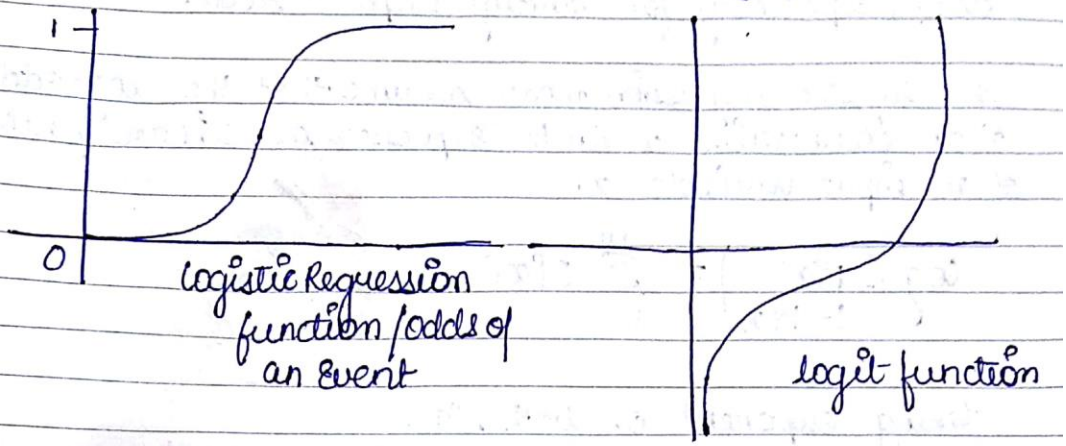
Since sigmoid is a non linear function with ~~to~~ use it linear seperation to model logistic regression we use odds of an event, which is defined as

$$\text{Odd of an Event} = \frac{\text{Probability of success}}{1 - \text{Probability of success}}$$

$$= \frac{p}{1-p}$$

Page No.
Date : / /

Since odds of an event predicts probability in the range $(0,1)$ & linear regressor models in the range $(-\infty, \infty)$, so we use log(odds) to model our problem in terms of linear regression.



So we can say

$$\text{logit}\left(\frac{p}{1-p}\right) = w^T x$$

$$= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \text{------}$$

$$\frac{p}{1-p} = e^{w^T x} \Rightarrow p = \frac{1}{1+e^{-w^T x}} = \frac{1}{1+e^{-z}}$$

$$(\text{let } z = w^T x)$$

hence, sigmoid function computes real valued scores b/w $(-\infty, \infty)$ & then squashes it to $(0,1)$ in order to model the probability of class label b/w 0 & 1. And to minimize mis-classification rate we predict $Y=1$ when $p \geq 0.5$ & $Y=0$ when $p \leq 0.5$. This means guessing $w^T x$ as non negative

Page No.
Date : / /

and 0 otherwise. So logistic regression gives a linear classifier

4 Logistic Regression is used to fit models for categorical data, especially for binary response data.

The logistic regression model assume that the log-odds of an observation y can be expressed as linear function of m input variable x

$$\log\left(\frac{P(x)}{1-P(x)}\right) = \sum_{i=0}^{m} \theta_i x_i$$

Taking exponent on both sides

$$\frac{P(x)}{1-P(x)} = \exp\left(\sum_{i=0}^{m} \theta_i x_i\right)$$

$$= \prod_{i=0}^{m} \left(\exp(\theta_i x_i)\right) = \prod_{i=0}^{m} \left(\exp(z)\right) \quad \left\{\text{let } z = \sum_{i=1}^{m} \theta_i x_i\right\}$$

$$P(x) = \frac{\exp(z)}{1+\exp(z)}$$

This function maps value from $(-\infty, \infty)$ to $(0,1)$ & is approximately linear near the origin.

The soln to logistic regression is a set of parameters θ that maximizes the likelihood of the data which is expressed as:

$$L(X|P) = \prod_{i=1}^{N} P(x_i)^{y_i=1} \prod_{i=1}^{N} (1-P(x_i))^{y_i=0}$$

Page No.
Date : / /

Since log is a monotonic function so we can take log of the likelihood function in order to simplify the computation :

$$\log L(X|P) = \sum_{i=1\ y_i=1}^{N} \log P(x_i) + \sum_{i=1\ y_i=0}^{N} \log (1 - P(x_i))$$

maximizing the log likelihood will maximize our likelihood.

To maximize log-likelihood we take the gradient of log-likelihood wrt θ.

$$\nabla_\theta L = \sum_{\substack{i=0 \\ y_i=1}}^{N} \frac{P'(x_i)}{P(x_i)} * x_i - \sum_{\substack{i=0 \\ y_i=0}}^{N} \frac{P_i'}{1-P_i} x_i \quad \text{---(1)}$$

Since $P'(x_i)$ is equivalent to :

$$P'(x_i) = P(x_i)(1 - P(x_i))$$

$$P(z) = \frac{e^z}{1+e^z} = e^z(1+e^z)^{-1}$$

$$P'(z) = e^z(1+e^z)^{-1} - \frac{(e^z)^2}{(1+e^z)^2}$$

$$= \frac{e^z}{1+e^z} - \frac{(e^z)^2}{(1+e^z)^2} = \frac{e^z(1+e^z) - e^{2z}}{(1+e^z)^2}$$

$$= \frac{e^z}{(1+e^z)^2} = \frac{e^z}{(1+e^z)} \cdot \frac{1}{(1+e^z)}$$

$$P'(z) = P(z)(1 - P(z))$$

Substituting this in (1)

$$\nabla_\theta l = \sum_{\substack{i=0 \\ y_i=1}}^{N} \frac{P(x_i)(1-P(x_i))}{P(x_i)} x_i - \sum_{\substack{i=0 \\ y_i=0}}^{N} \frac{P(x_i)(1-P(x_i))}{(1-P(x_i))} x_i$$

$$= \sum_{\substack{i=0 \\ y_i=1}}^{N} (1-P(x_i)) x_i - \sum_{\substack{i=1 \\ y_i=0}}^{N} P(x_i) x_i$$

$$= \sum_{i=1}^{N} \left[(1-P(x_i)) y_i - (1-y_i) P(x_i) \right] x_i$$

for maximizing the log-likelihood the gradient descent must be zero

$$\nabla_\theta l = 0$$

$$\sum_{i=1}^{N} \left[y_i - P(x_i) \right] x_i = 0$$

$$\sum_{i=1}^{N} y_i x_i - P(x_i) x_i = 0 \Rightarrow \sum_{i=1}^{N} y_i = \sum_{i=1}^{N} P(x_i)$$

Now here we can see that the Equation is in terms of probabilities P not in term of parameter θ. This means logistic regression are coordinate free, meaning the probability will remain same, even if we shift, rescale or combine our variables.

For solving for the coef.(θ) we can use Newton's method or Fisher scoring method.

Page No.
Date : / /

5 $H[x] = \frac{1}{2}\ln|\Sigma| + \frac{D}{2}(1+\ln(2\pi))$

where x is $x \sim N(\mu, \Sigma)$
D is Dimensionality of x.

$$H(x) = -\int p(x)\ln p(x)\,dx$$

$$= -\int N(x|\mu,\Sigma)\ln N(x|\mu,\Sigma)\,dx$$

$$= -\int N(x|\mu,\Sigma)\ln\left[\frac{1}{(2\pi|\Sigma|)^{D/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)\right]dx$$

$$= -\int N(x|\mu,\Sigma)\left(-\frac{D}{2}\ln(2\pi|\Sigma|)\right)dx - \int N(x|\mu,\Sigma)\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)$$

$$= \frac{D}{2}\ln(2\pi|\Sigma|) + \int \mathrm{Trace}\left[\Sigma^{-1}(x-\mu)(x-\mu)^T\right]N(x|\mu,\Sigma|\,dx$$

$$= \frac{D}{2}\ln(2\pi|\Sigma|) + \mathrm{Trace}\left(\Sigma^{-1}\int(x-\mu)(x-\mu)^T N(x|\mu,\Sigma)\,dx\right)$$

$$= \frac{D}{2}\ln(2\pi|\Sigma|) + \mathrm{Trace}\left[\Sigma^{-1}\Sigma\right]$$

$$= \frac{D}{2}\ln(2\pi|\Sigma|) + \frac{D}{2} = \frac{D}{2}\left(1+\ln(2\pi|\Sigma|)\right)$$

Page No.
Date : / /

6 $y = \left(\frac{1}{d}\right) Rx + B$ where $x \subseteq R^2$ & $y \subseteq R^2$, $B = [a, b]^T$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\hat{y}_i = \frac{1}{d} \begin{bmatrix} x_i^1 \cos\theta - x_i^2 \sin\theta \\ x_i^1 \sin\theta + x_i^2 \cos\theta \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\text{loss} = \frac{1}{2n} \sum_{i=1}^{n} \left\| \hat{y}_i - y \right\|_2^2$$

for minimising the loss we have to minize the value of θ, d, a, b.
So applying gradient descent on above loss :-

loop, $\hat{}$ until convergence

$$\theta = \theta - \alpha * \frac{\nabla(\text{loss})}{\nabla\theta} = \theta - \alpha * \frac{\nabla}{\nabla\theta} \frac{\left(\sum \sqrt{\hat{y}_i - y_i}\right)^2}{2n}$$

$$a = a - \alpha * \frac{\nabla(\text{loss})}{\nabla a} = a - \alpha * \frac{\nabla}{\nabla a} \frac{\left(\sum \sqrt{\hat{y}_i - y_i}\right)^2}{2n}$$

$$b = b - \alpha * \frac{\nabla(\text{loss})}{\nabla b} = b - \alpha * \frac{\nabla}{\nabla b} \frac{\left(\sum \sqrt{\hat{y}_i - y_i}\right)^2}{2n}$$

$$d = d - \alpha * \frac{\nabla(\text{loss})}{\nabla b} = d - \alpha * \frac{\nabla}{\nabla d} \frac{\left(\sum \sqrt{\hat{y}_i - y_i}\right)^2}{2n}$$

}

for simplification I have substituted B in place of a, b & R in place of $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$

Page No.
Date : / /

loop until convergence {

$$\theta = \theta - \alpha * \left[\frac{2}{2n}\left(\frac{-1}{2(\sqrt{\hat{y}-y})}\right)\cdot \frac{\partial x}{d}\right]$$

$$B = B - \alpha\left[\frac{2}{2n}\left(\frac{-1}{2\sqrt{\hat{y}-y}}\right)\cdot\right]$$

$$d = d - \alpha\left[\frac{2}{2n}\left(\frac{-1}{2\sqrt{\hat{y}-y}}\right) Rx.\left(\frac{-1}{2d^2}\right)\right]$$

}

For finding min. value of R, a, b, d

$$R = 0$$

$$R - \alpha\left[\frac{2}{2n}\left(\frac{-1}{2(\sqrt{\hat{y}-y}}\right)\frac{\partial x}{d}\right] = 0$$

ignoring all constants

$$R = \frac{-x}{2\sqrt{\hat{y}-y}}$$

$$B = 0$$

$$B = \alpha\left[\frac{-1}{2\sqrt{\hat{y}-y}}\right]$$

ignoring constants

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{-1}{2\sqrt{\hat{y}-y}} \qquad a = \frac{-1}{2\sqrt{\hat{y}-y}}$$

Page No.
Date : / /

$$b = \frac{-1}{2\sqrt{\hat{y}-y}}$$

$$d = 0$$

$$d - \alpha\left[\frac{-1}{2\sqrt{\hat{y}-y}} \; Rx\left(\frac{-1}{2d^2}\right)\right] = 0$$

Ignoring constants

$$d = \frac{x}{4(\sqrt{\hat{y}-y})(d^2)}$$

$$d = \sqrt[3]{\frac{x}{4\sqrt{\hat{y}-y}}}$$