Department of Electronic & Telecommunication Engineering

University of Moratuwa

EN3150 – Pattern Recognition

Learning from data and related challenges and Classification

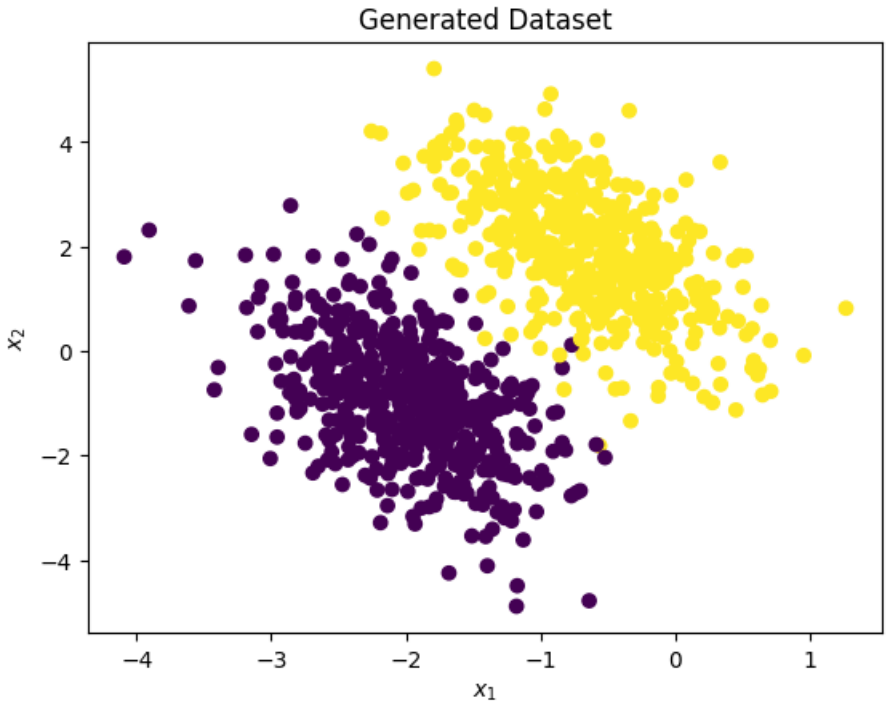
(Assignment 01 - Report)

Pattern-Recognition/Classification at main · kavindukalinga/Pattern-Recognition (github.com)

200087A Chandrasiri Y.U.K.K.

1. Logistic Regression Weight Update Process

The code given in listing 1 is used to generate below dataset.



The data set consists of two features x_1 , x_2 and two targets. The model is in the form of;

$$p(y = 1) = sigmoid(w_0 + w_1x_1 + w_2x_2) = \frac{1}{1 + e^{-(w_0+w_1x_1+w_2x_2)}}$$

The two targets are 1(True) and 0(False).

Gradient Descent

Initializing weights as zeros, gradient descent-based weight update is performed for the given data. Here, binary cross entropy is used as a loss function. Further, it's used learning rate as α = 0.1 and number of iterations as t = 10.

Batch Gradient descent weight update

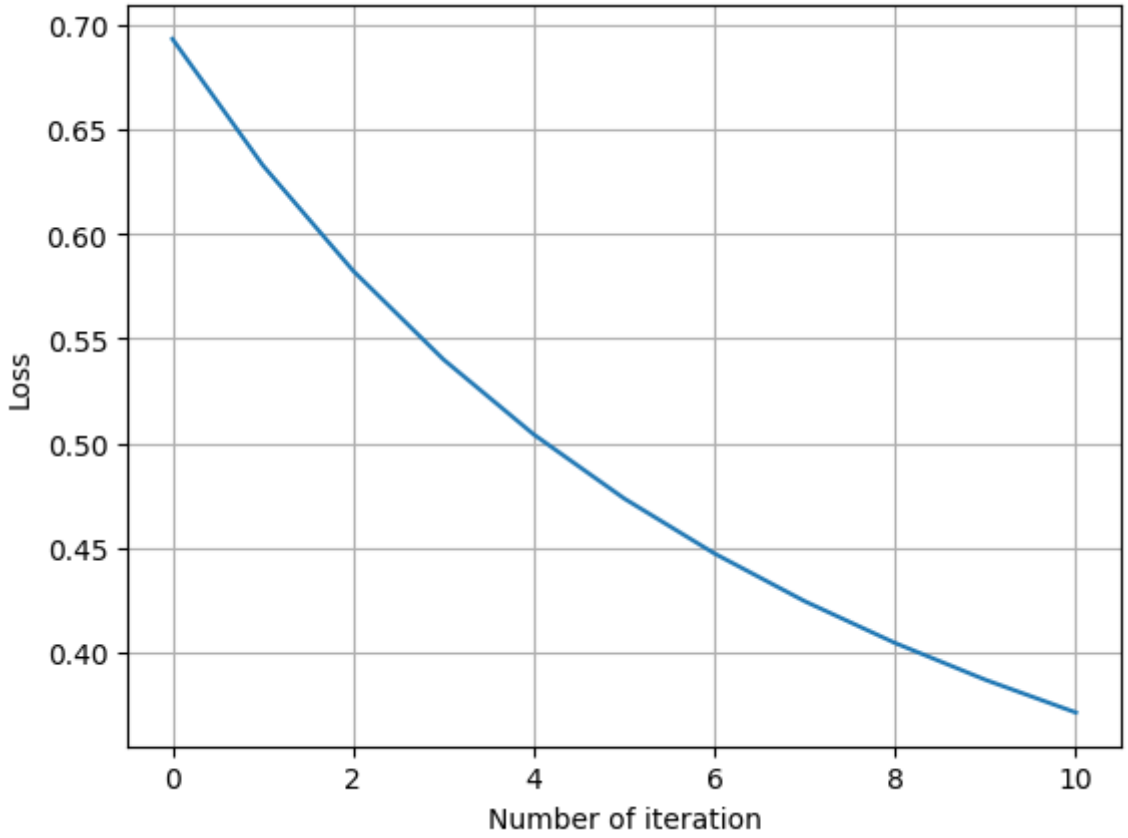
$$w_{(t+1)} \leftarrow w_{(t)} - \alpha \frac{1}{N} \left(1_N^T \, diag\left(sigm\left(w_{(t)}^T x_i\right) - y_i\right) X\right)^T$$

Here, X is data matrix of dimension of N×(D+1). Here, N is total number of data samples and D is number of features. Now, X is given by,

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{1,1} & x_{2,1} & \cdots & x_{D,1} \\ 1 & x_{1,2} & x_{2,2} & \cdots & x_{D,2} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{1,i} & x_{2,i} & \cdots & x_{D,i} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_{1,N} & x_{2,N} & \cdots & x_{D,N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_i^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}.$$

The batch gradient descent is performed over 10 iterations and plotted the binary cross entropy loss.

Figure: The loss with respect to number of iterations.



Newton's Method

Initializing weights as zeros, Newton's method weight update is performed for the given data. Here also, binary cross entropy is used as a loss function. Further, number of iterations is set as t = 10.

Batch Newton's method weight update

$$w_{(t+1)} \leftarrow w_{(t)} - \left(\frac{1}{N} X^T S X\right)^{-1} \left(\frac{1}{N}\left(1_N^T \, diag\left(sigm\left(w_{(t)}^T x_i\right) - y_i\right) X\right)^T\right)$$

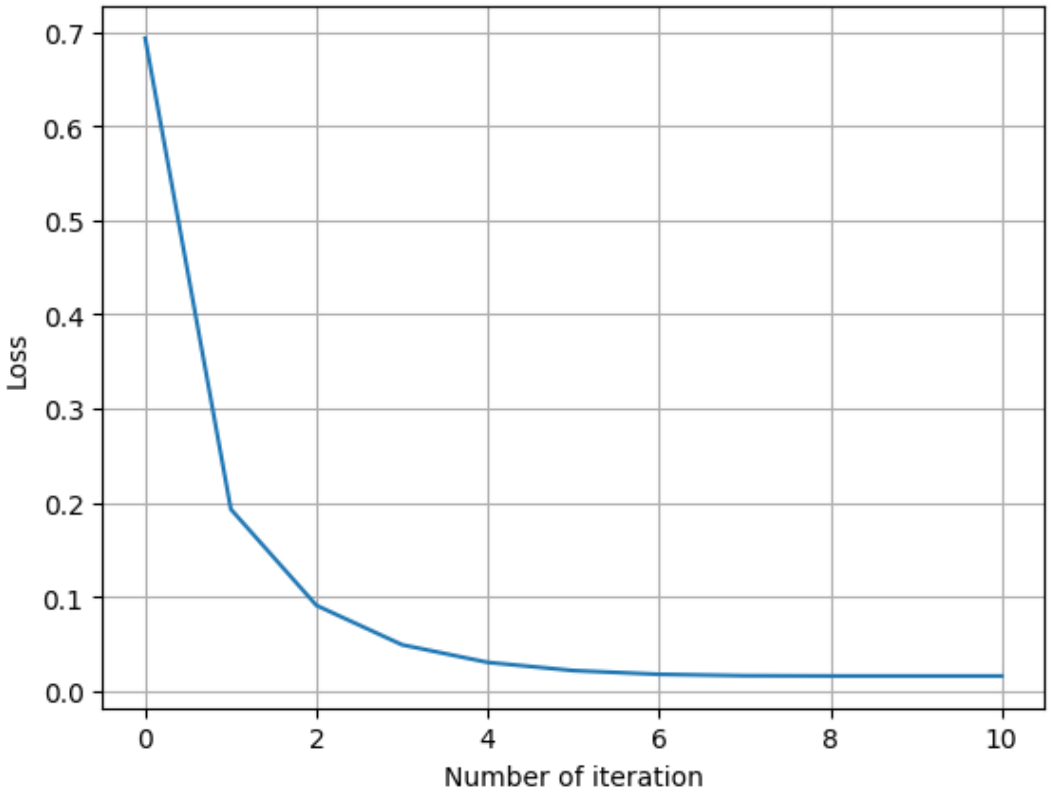
S is given by,

$$S = diag(s_1, s_2, \ldots., s_N)$$

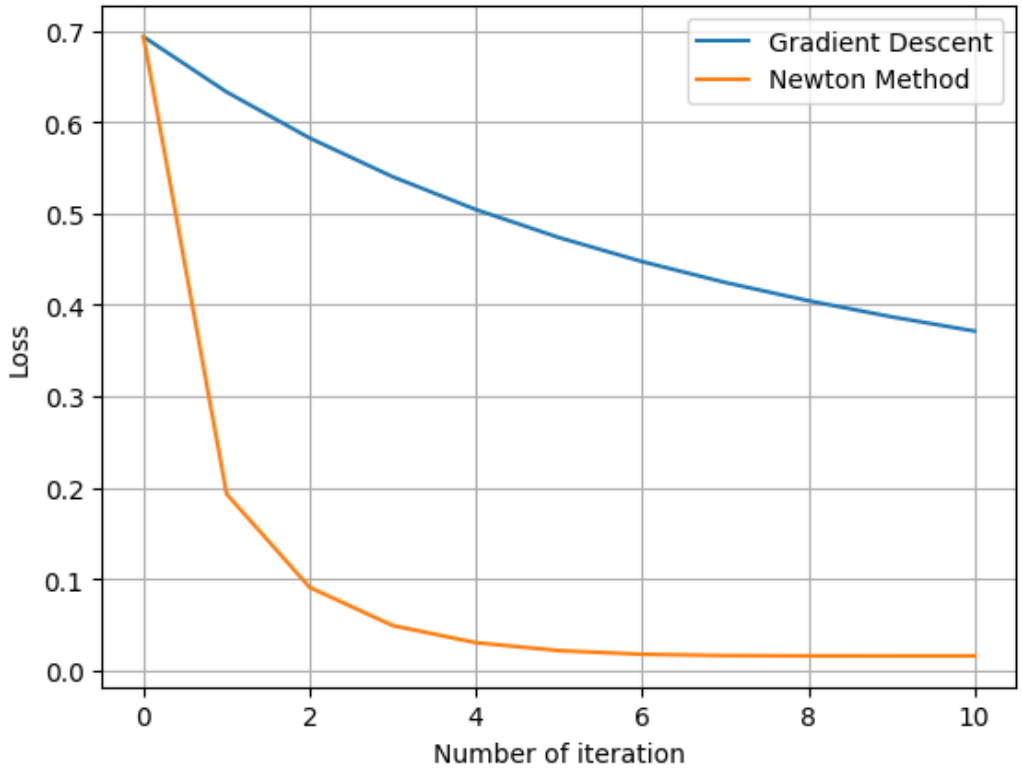
$$s_i = (sigm\left(w_{(i)}^T x_i\right) - y_i)(1 - sigm\left(w_{(i)}^T x_i\right) - y_i)$$

The Newton's Method is used over 10 iterations and plotted the binary cross entropy loss.

Figure: The loss with respect to number of iterations.



Comparison



The loss with respect to number of iterations for both Gradient descent and Newton's method is plotted in the above figure.

According to the plot, we can observe that the Newton's method is much faster than gradient descent and converges earlier with the number of iterations. But the Newton's method is computationally expensive as in the function, it consists of additional $\left(\frac{1}{N}X^TSX\right)^{-1}$ term.

The below figures show the dataset and the model after 10 iterations.

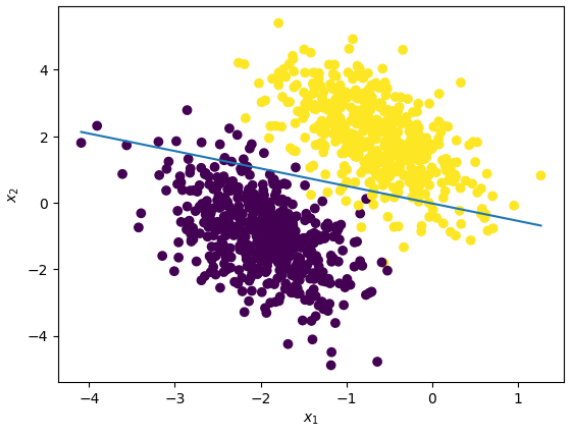


Figure: Model using Batch Gradient Descent

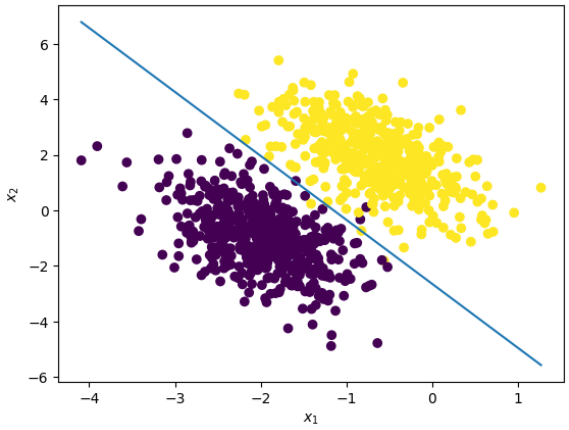


Figure: Model using Newton's Method

According to these graphs as well, we can observe that in 10 iterations, Newton's method has converged faster than the Batch Gradient Descent.

2. Perform grid search for hyper-parameter tuning

The code given in listing 2 is used to load data.

Purpose of $X = X[permutation]$ and $y = y[permutation]$

The purpose of these two functions is to shuffle the data samples and to make sure that they are ordered randomly and not in any particular order.

Lasso logistic regression for image classification

Lasso logistic regression for image classification is used as
$LogisticRegression(penalty = 'l1', solver = 'liblinear', multi\_class = 'auto')$

Next, created a pipeline that includes the scaling, the Lasso logistic regression estimator, and a parameter grid for hyperparameter tuning (C value).

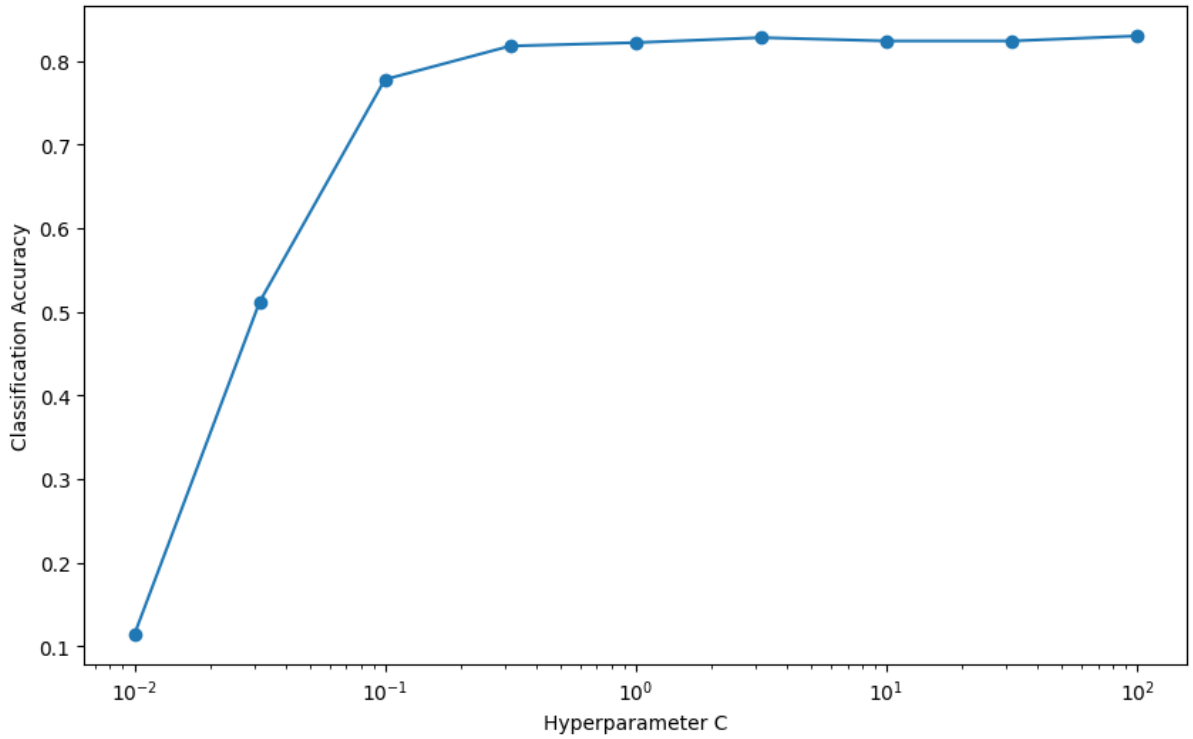
$GridSearchCV$ is used to perform a grid search over the range [ex: $np.logspace(-2, 2, 9)$] to find optimal value of hyperparameter C.

In this scenario, 5-fold cross validation is used (cv=5)

```python
pipe = Pipeline ([
    ('scaler' , StandardScaler ()),
    ('classifier' , LogisticRegression ( penalty ='l1' , solver ='liblinear' , multi_class= 'auto' ))
])

param_grid = [
    {'classifier__C' : np . logspace (-2 , 2 , 9) }
]

grid_search = GridSearchCV ( pipe , param_grid = param_grid , cv =5)

grid_search.fit(X_train , y_train)

best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

```

Python

Figure: The classification accuracy with respect to hyperparameter C

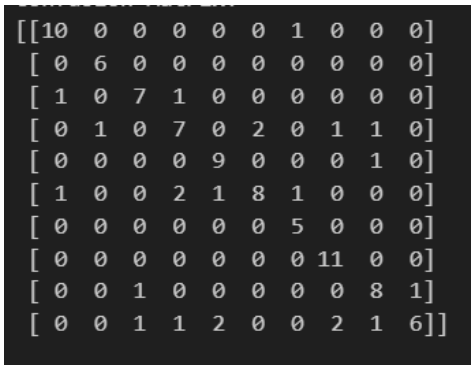


$$best\_params = \{'classifier\_\_C': 3.1622776601683795\}$$

When hyperparameter C=3.16227766, the classification accuracy becomes highest. Therefore it is the best value for C.

Confusion matrix, precision, recall and F1-score

Confusion matrix:



```
[[10  0  0  0  0  0  1  0  0  0]
 [ 0  6  0  0  0  0  0  0  0  0]
 [ 1  0  7  1  0  0  0  0  0  0]
 [ 0  1  0  7  0  2  0  1  1  0]
 [ 0  0  0  0  9  0  0  0  1  0]
 [ 1  0  0  2  1  8  1  0  0  0]
 [ 0  0  0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  0  0 11  0  0]
 [ 0  0  1  0  0  0  0  0  8  1]
 [ 0  0  1  1  2  0  0  2  1  6]]
```

Precision = 0.7747575757575756

Recall = 0.77

F1-Score = 0.7589903748425487

The model has a precision value of 0.77, a significantly high value shows that there can be low false positives. That means the predicted values has a higher probability to be correct,

And the recall value of 0.77 is also a significantly higher value. It shows that there can be low false negatives. That means the predicted false values will most probably false.

Higher F1 score means the balance between precision and recall in a good way. Therefore, this model works overall well.

3. Logistic Regression

$x_1 = number\ of\ hours\ studied$

$x_2 = undergraduate\ GPA$

$y = bool(student\ received\ A+)$

$$p(y = 1) = \frac{1}{1 + e^{-(-6+0.05x_1+x_2)}}$$

The estimated probability that a student, who has studied for 40 hours and has an undergraduate GPA of 3.5, will receive an A+ in the class.

$\frac{1}{1+e^{-(-6+0.05\times40+3.5)}}$ =0.3775

There is 0.3775(37.75%) of probability.

To achieve a 50% chance of receiving an A+ in the class, how many hours of study does a student like the one in part (1a) need to complete.

$$\frac{1}{1 + e^{-(-6+0.05x_1+3.5)}} = 0.5 \rightarrow x_1 = 50$$

Need to study 50 hours.