UNIVERSITY OF MORATUWA, SRI LANKA
Faculty of Engineering
Department of Electronic and Telecommunication Engineering
Semester 5 (Intake 2020)

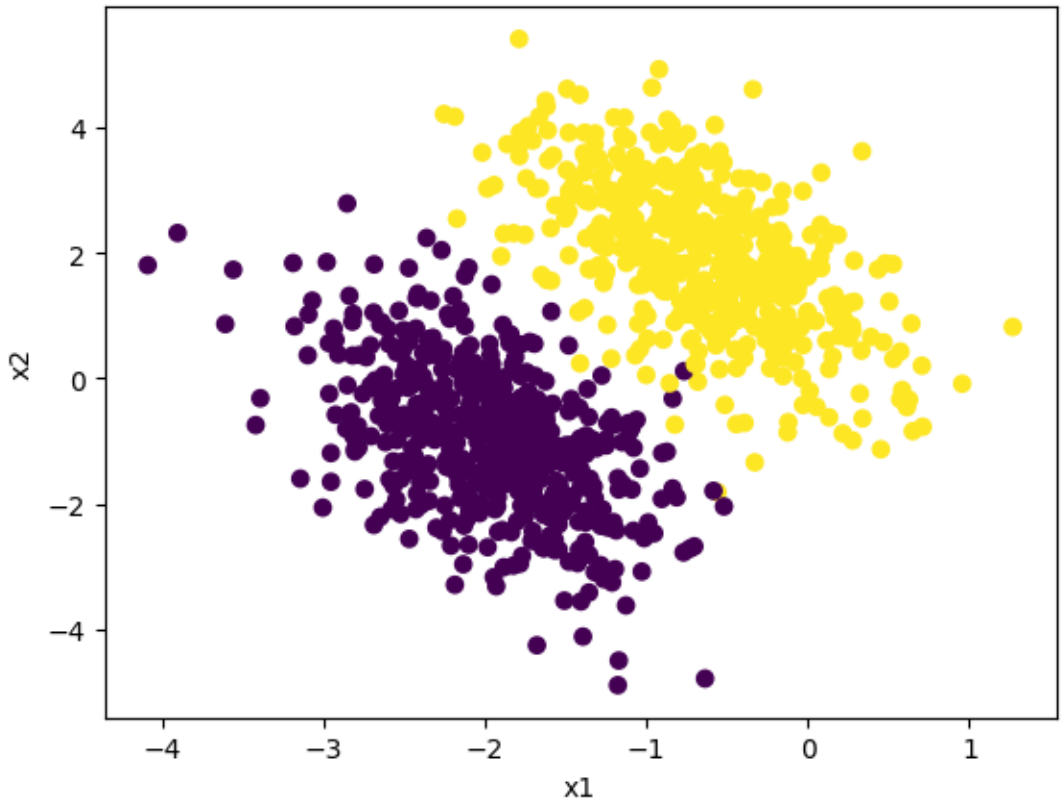
EN3150 – Pattern recognition

Assignment 02
Learning from data and related challenges and classification

AMARASEKARA A. T. P.
200023C

1. Logistic regression weight update process

The data was generated and visualized and the following plot is obtained.



The data set contains two features and the target value is binary (0 or 1). Hence the, sigmoid function is used to predict the output and the binary cross-entropy function is used to calculate the loss.

$$P(y = 1) = sigmoid(w_0 + w_1 x_1 + w_2 x_2) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

$$Loss = -y_{true} \log(y_{pred}) - (1 - y_{true}) \log(1 - y_{pred})$$

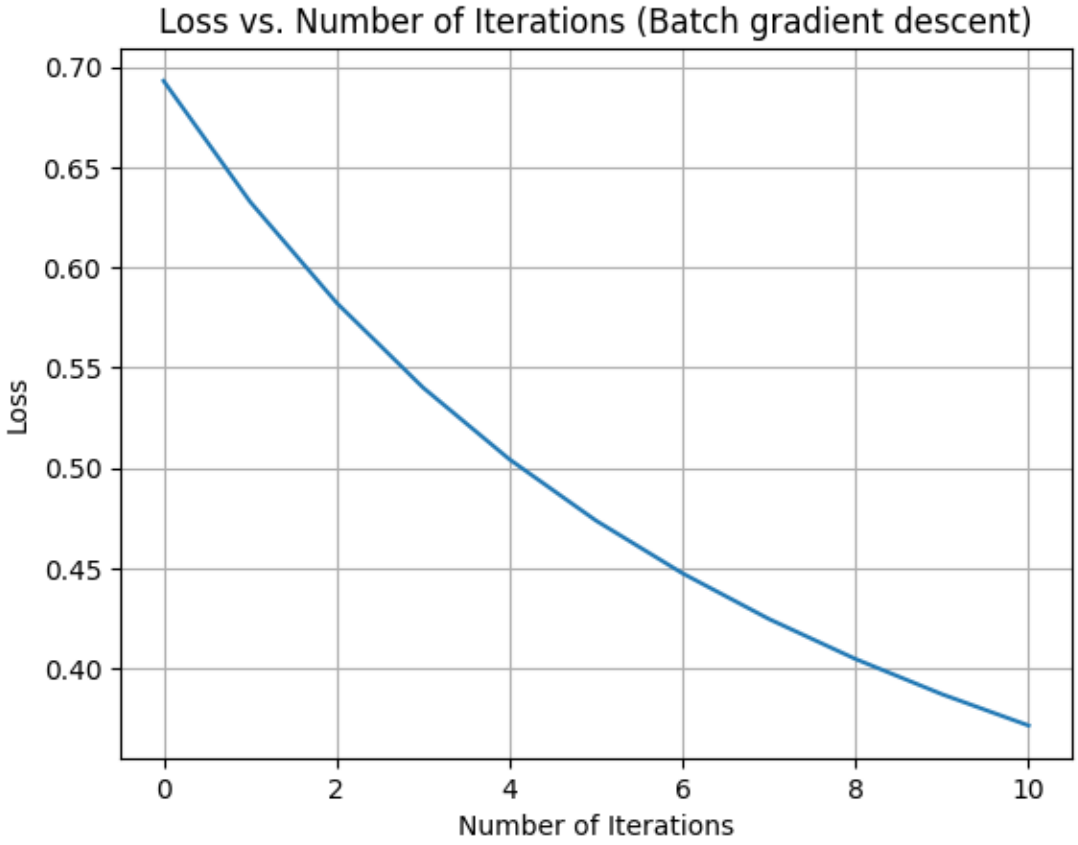
Initializing the weights as zeroes, the weights were updated using Batch Gradient Descent method and Newton's method.

Weight update using Batch Gradient Descent method

Using the data provided, we will use the batch gradient descent method over t=10 iterations with learning rate set to 0.1, to identify the model parameters w_0, w_1, w_2 that minimize the binary cross-entropy cost function.

$$\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)} - \alpha \frac{1}{N} \left(\mathbf{1}_N^T \, \mathrm{diag}(\mathrm{sigm}(\boldsymbol{w}_{(t)}^T \boldsymbol{x}_i) - \boldsymbol{y}_i) \boldsymbol{X} \right)^T$$

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



After 10 iterations, the updated weights are as follows.

$$w = \begin{bmatrix} 0.00903176 \\ 0.26230116 \\ 0.49949384 \end{bmatrix}$$

The error of the predictions decreases as we iterate through the Batch Gradient Descent method. However, the plot exhibits a slow convergence.

Weight update using Newton's method

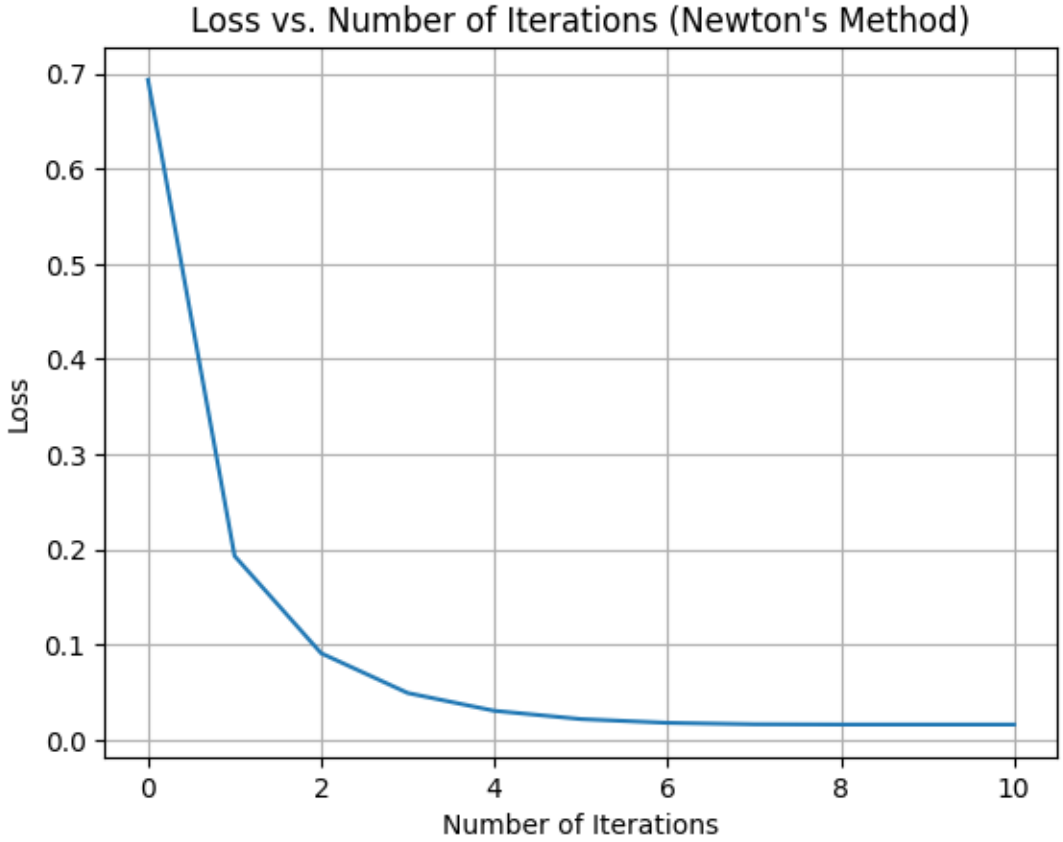
Using the data provided, we will use the Newton's method over t=10 iterations with learning rate set to 0.1, to identify the model parameters w_0, w_1, w_2 that minimize the binary cross-entropy cost function.

$$\boldsymbol{w}_{(t+1)} \leftarrow \boldsymbol{w}_{(t)} - \left(\frac{1}{N}\,\boldsymbol{X}^T\boldsymbol{S}\boldsymbol{X}\right)^{-1}\left(\frac{1}{N}\,\left(\mathbf{1}_N^T\,\mathrm{diag}(\mathrm{sigm}(\boldsymbol{w}_{(t)}^T\boldsymbol{x}_i) - \boldsymbol{y}_i)\boldsymbol{X}\right)^T\right)$$

and is S given by

$$\boldsymbol{S} = \mathrm{diag}(s_1,\, s_2, \ldots, s_N),$$

$$\boldsymbol{s_i} = \left(\mathrm{sigm}(\boldsymbol{w}_{(t)}^T\boldsymbol{x}_i) - \boldsymbol{y}_i)\right)\left(1 - \mathrm{sigm}(\boldsymbol{w}_{(t)}^T\boldsymbol{x}_i) - \boldsymbol{y}_i)\right)$$



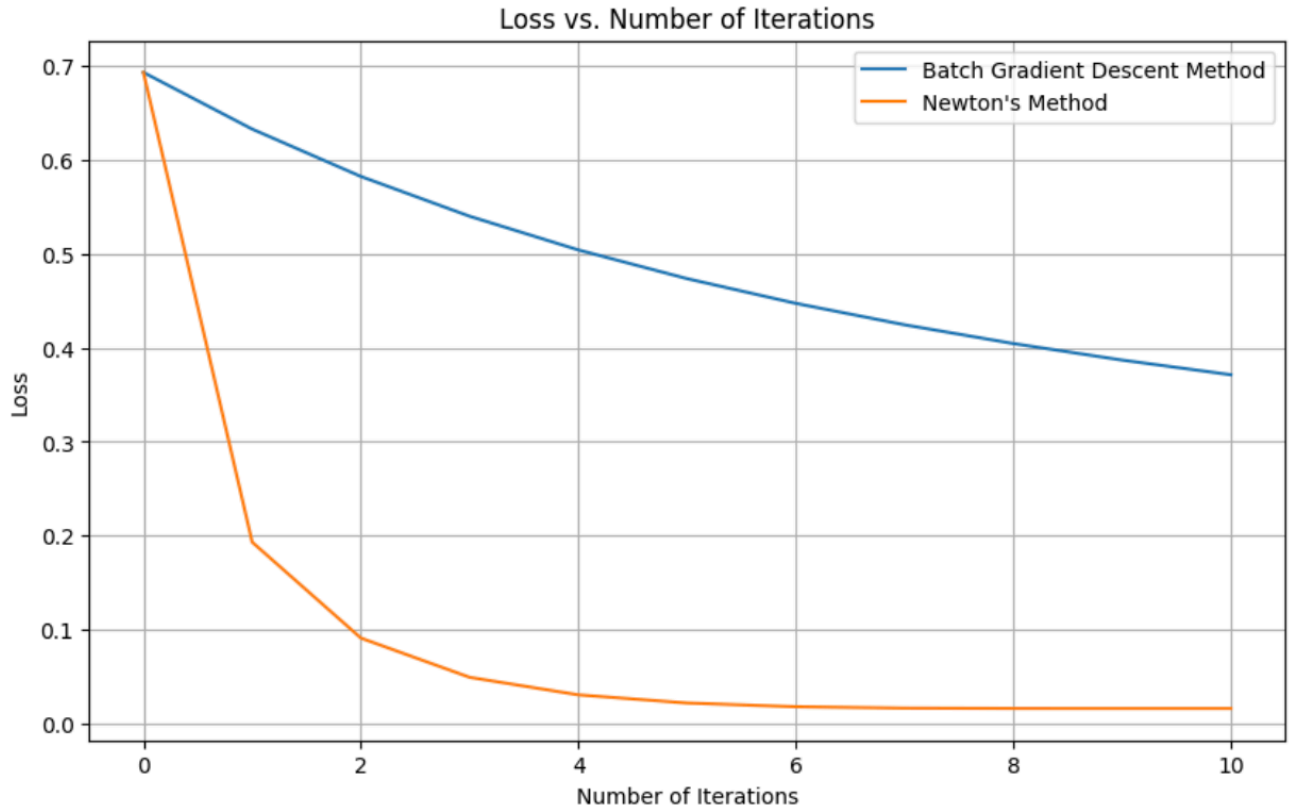
After 10 iterations, the updated weights are as follows.

$$w = \begin{bmatrix} 11.71643583 \\ 10.20983751 \\ 4.43019025 \end{bmatrix}$$

The Newton's method for updating the weights shows a fast convergence and reaches the stability level within 6 iterations.

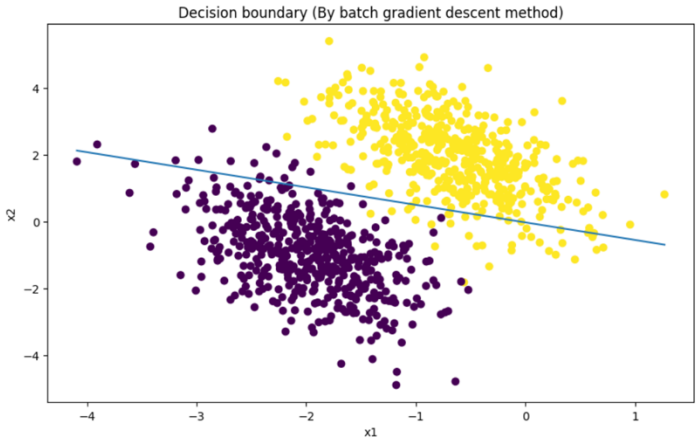
Comparison of Gradient descent and Newton ‘s method

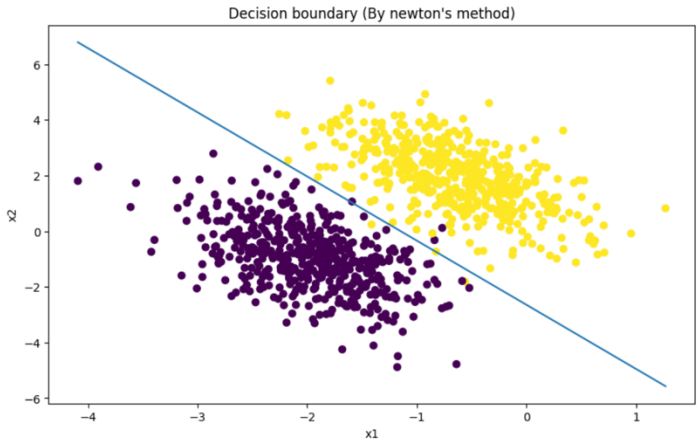
The following plot compare the loss with respect to the number of iterations for the two methods.



Comparing the two methods for updating the weights of the model, we can observe that the Newton’s method for updating weight shows a much faster convergence than the Gradient descent method.

The following plots compares the decision boundaries obtained by the two methods after 10 iterations.





By the above plots as well observe that the Newton’s method provides a much better decision boundary for the classification of the two classes in comparison with the Gradient descent method.

However, a limitation of Newton’s method for being used for weight update process of regressions is the high computational complexity of the method as it requires computations of more terms in addition to the gradient which is required by the Gradient descent method.

2. Grid search for hyper-parameter tuning

In the code, permutation outputs an array of randomly shuffled indices. Hence, X[permutation] and y[permutation] is used to reorder the data in a random order. This randomization helps ensure that the model does not learn any patterns related to the order of the data, which could lead to more robust and unbiased model training and evaluation.

Defining a Pipeline with scaler and Lasso regression

Pipeline is a structured and automated workflow that combines multiple data processing and machine learning steps into a single sequence. It mainly includes, data preprocessing, model training, hyperparameter tuning, and cross-validation.

In the code, the pipeline includes data scaling with StandardScaler and Lasso Logistic Regression to estimate the best fit model for our data.

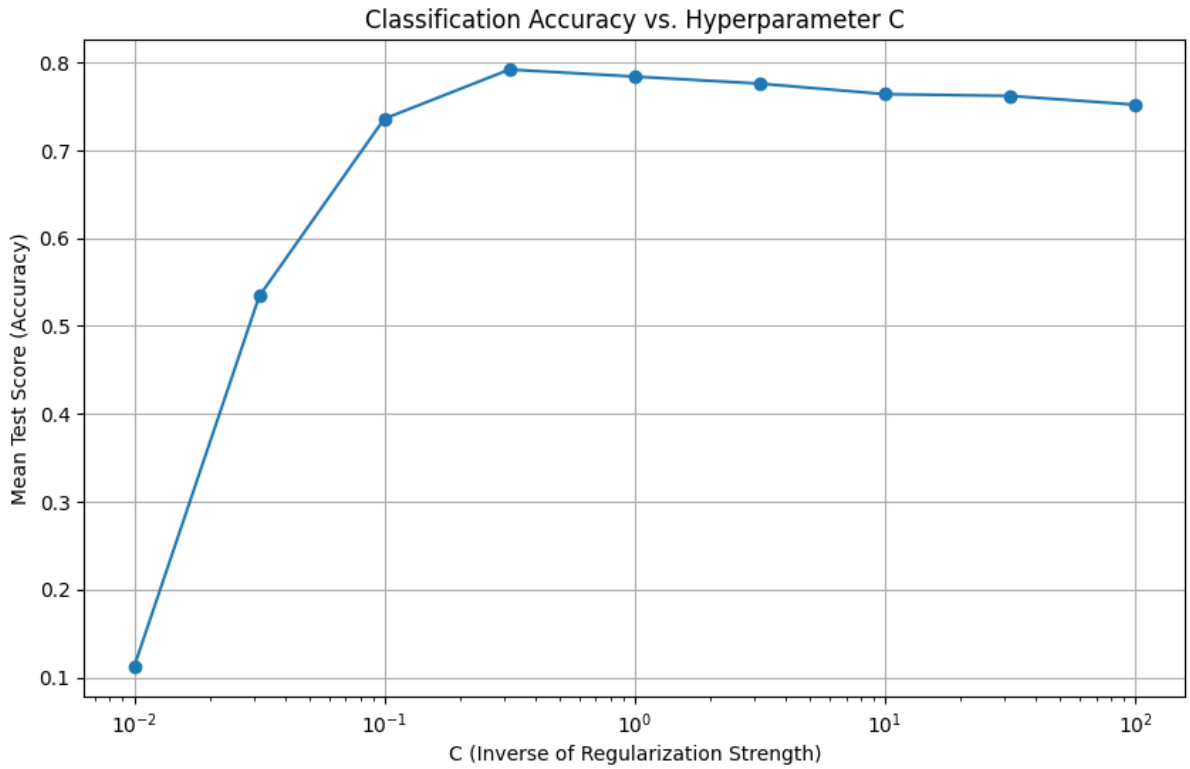
```
# Define a Standard Scaler to normalize inputs
scaler = StandardScaler()

# Define the logistic regression estimator
logistic = LogisticRegression(penalty='l1', solver='liblinear',
multi_class='auto')

# Create a pipeline with scaling and the estimator
pipeline = Pipeline([("scaler", scaler), ("logistic", logistic)])

# Parameters of pipelines
param_grid = {
    "logistic__C": np.logspace(-2, 2, 9),
}
```

Performing Grid Search

After defining the pipeline, in order to obtain the hyperparameter involved with the regularization term of Lasso Logistic regression, grid search is performed.

```
# Perform grid search
grid_search = GridSearchCV(pipeline, param_grid, cv=5,  n_jobs=-1)

# Fit the model to the training data
grid_search.fit(X_train, y_train)

# Extract the results of the grid search
results = grid_search.cv_results_

# Get the values of hyperparameter C from the parameter grid
C_values = param_grid['logistic__C']

best_params = grid_search.best_params_
print('Best parameter:', best_params)
```

After performing the grid search, the following plot of classification accuracy vs. C values was obtained.



According to the graph, the best performing hyperparameter that is the hyperparameter resulting in the highest classification accuracy is,
Inverse of Regularization Strength = 0.3162

Additionally, the following confusion matrix, precision, recall and F1-score was obtained.

Confusion Matrix:

```
[[12  0  0  0  0  0  0  0  0  0]
 [ 0  9  0  0  0  0  0  0  0  0]
 [ 0  1  8  0  0  0  0  0  0  0]
 [ 0  1  1 12  0  0  0  0  0  0]
 [ 0  0  0  0 10  0  0  0  0  1]
 [ 2  1  0  1  0  3  0  0  2  0]
 [ 0  1  0  0  0  0  9  0  0  0]
 [ 0  0  0  0  0  0  0  8  0  1]
 [ 0  0  0  0  0  0  0  0 11  0]
 [ 0  0  0  0  1  0  0  1  0  4]]
```

Precision: 0.8775
Recall: 0.8600
F1-Score: 0.8494

By the above values it is observable that precision, recall and F1 score are very high valued.

A higher precision of 0.8775 indicates a lower rate of false positives, which means the model is making fewer incorrect positive predictions.
A higher recall of 0.8600 indicates a lower rate of false negatives, meaning the model is correctly identifying more positive instances.
A higher F1 score of 0.8494 indicate a better balance between precision and recall.
Hence, it can be concluded that model's performance is of high satisfactory level.

3. Logistic regression

In logistic regression we use the sigmoid function to model the probability of the prediction. The sigmoid function is defined as,

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

Where,

P(y=1) is the probability of receiving an A+
w_0, w_1, and w_2 are the estimated coefficients
x_1 is the number of hours studied
x_2 is the undergraduate GPA
e is the base of the natural logarithm

For the above scenario, $w_0 = -6$, $w_1 = 0.05$, and $w_2 = 1$

To estimate the 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, substitute 40 and 3.5 for x_1 and x_2 respectively.

$$P(y=1) = \frac{1}{1+e^{(-6+(0.05)(40)+(1)(3.5))}}$$

$$P(y=1) = 0.3775$$

Therefore, the above-mentioned student has a **37.75%** probability of receiving an A+ in the class.

To obtain how many hours the above-mentioned student has to study to achieve a 50% probability of receiving an A+ in the class, substitute 3.5 for x_2 and 0.5 for P(y=1).

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

$$x_1 = 50$$

Therefore, the above-mentioned student has to study for **50 hours** to achieve a 50% probability of receiving an A+ in the class.