ECE4179 S2 2020 - Homework 1

Jia Lin Gao (jlgao2@student.monash.edu) - Monash University

August 29, 2020

Question 1

Restricting the bias term to be fixed would cause the separating hyperplane/decision boundary to necessarily intersect with the origin or a preset bias. Then as the weight updates during the training process would only be alter the "slope" of the separating hyperplane.

The implication of this is that a perceptron model will still train and converge if a true classification boundary intersects the origin/the set bias, however the algorithm will not converge if the training set can no longer be linearly separated.

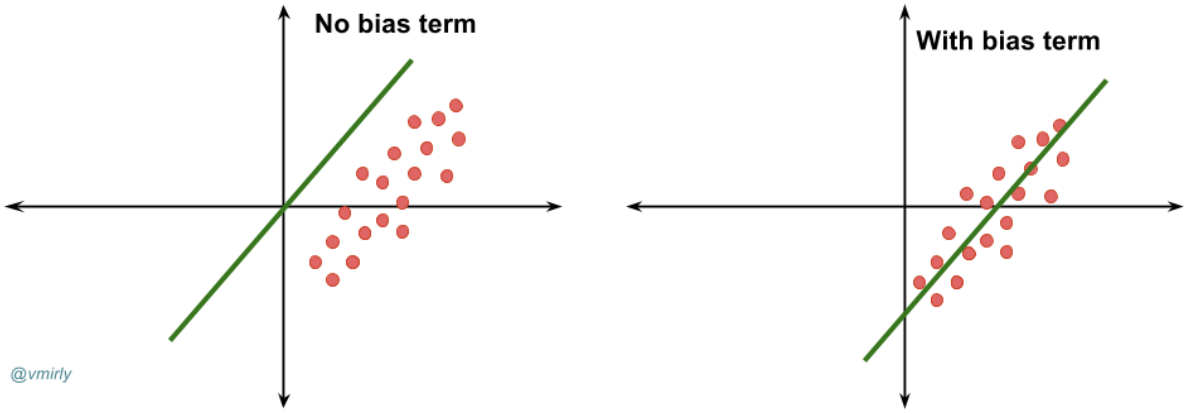

Figure 1: Decision boundary with and without bias term

Question 2

A function to compute the sigmoid

```
def sigmoid(x):
    # this function should compute the sigmoid of x
    return 1/(1+np.exp(-x))
```

A function to compute the gradient and the cost of logistic regression

```
def compute_grad_loss(X, y, theta):
    """
    this function will get X, a set of samples (each sample is a row in
↪   X), the corresponding labels in the array y and the current parameter
↪   of the logistic model theta then use the sigmoid function to compute
↪   the loss and the gradient of samples with respect to theta
    """
    y_hat = predict(X, theta)
    """
    when computing the loss value, pay extra attention to the log
↪   function. log(0) can cause problems so you need to handle it
    """
    try:
        loss =
        ↪  np.mean(np.matmul(-y.T,np.log(y_hat))-np.matmul((1-y).T,np.log(1-y_hat)))
    except:
        print("did you try and take the log of zero you madman?")
    grad_vec = np.matmul((y_hat-y).T,X)

    return loss, grad_vec
```

Gradient descent for loop

```
for epoch in range(max_epoch):
    #call the compute_grad_loss that you have implemented above to measure
    ↪  the loss and the gradient
    loss[epoch], grad_vec = compute_grad_loss(X_train, y_train, theta)
    #update the theta parameter according to the GD
    theta = theta - lr*grad_vec.T
```

Predict function

```
def predict(X, theta):
    """
    this function should get X, an array of samples, and theta, the
↪   parameters
    of the logistic model and generate 0 or 1 as the label of each sample
↪   in X
    #the rule is that, if the sigmoid of x >= 0.5, we predict the label of
↪   x to be 1
    otherwise the label is 0
    """
    return np.round(sigmoid(np.matmul(X , theta)))
```

Question 3

The learning rate parameter

The learning rate parameter in the gradient descent algorithm is a scalar multiplier of the gradient which affects the rate at which they weights will update per iteration.

A higher learning rate will mean that the learning algorithm will be able to converge to the loss minima more quickly, however a higher learning rate may also have the effect of not being able to converge on the absolute minima.

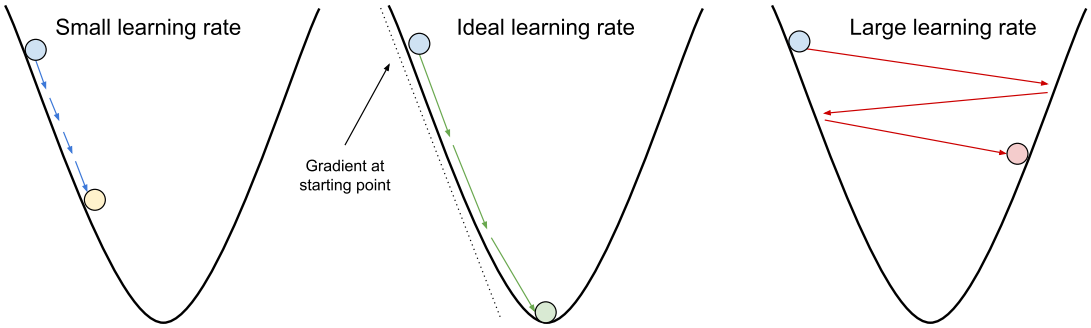

Figure 2: Learning Rate effect on Gradient Descent
Source: Adapted from nuric @ ICL

Conversely, smaller learning rates may cause the weights to be converge to a local but not global minima. The process of finding optimum learning rates is a focal point for emerging ML research.

Model accuracy

The model accuracy can be evaluated on the test data with the following code:

```
# first call the predict function on your test data with the parameters
↪  obtained by DG
y_test_hat = predict(X_test,theta)
# make sure that the predictions are either 0 or 1 and the shape of
↪  y_test_hat
# matches that ofy_test
y_test_hat
# the script below, if the dimensionality of the arrays is set correctly,
# will measure how many samples are correctly classified by your model
score = float(sum(y_test_hat == y_test))/ float(len(y_test))
print(score)
```

Table 1: Linear regression results with various hyperparameters

Epochs **Learning Rate**	0	10	20	30	40	50	60	70	80	90	100
0.005	0.64	0.77	0.77	0.81	0.83	0.76	0.73	0.78	0.82	0.78	0.77
0.010	0.85	0.84	0.85	0.75	0.76	0.84	0.77	0.83	0.83	0.85	0.78
0.015	0.73	0.83	0.71	0.85	0.74	0.79	0.77	0.74	0.81	0.81	0.85
0.020	0.26	0.79	0.82	0.85	0.85	0.79	0.82	0.82	0.85	0.70	0.85
0.025	0.83	0.77	0.82	0.75	0.83	0.82	0.73	0.80	0.81	0.83	0.76
0.030	0.31	0.80	0.78	0.85	0.74	0.75	0.76	0.82	0.85	0.75	0.78
0.035	0.71	0.85	0.81	0.85	0.85	0.78	0.84	0.76	0.83	0.73	0.75
0.040	0.31	0.82	0.74	0.77	0.68	0.76	0.81	0.77	0.71	0.82	0.81
0.045	0.27	0.84	0.81	0.75	0.77	0.82	0.71	0.81	0.82	0.82	0.85
0.050	0.69	0.83	0.81	0.76	0.86	0.73	0.81	0.70	0.85	0.78	0.85

A grid search for the learning rate and number of epochs parameters yielded the results in 1. The maximum accuracy of 0.8575 obtained at a learning rate of 0.05 with 40.0 epochs

Solution with and without bias term

By augmenting the X data matrices with a column of ones, we add a bias term to the linear regression which ca be done with the following code.

```
X_train_aug = np.c_[np.ones(X_train.shape[0]),X_train]
X_test_aug = np.c_[np.ones(X_test.shape[0]),X_test]
```

The maximum accuracy obtained with a bias term is 0.855 at a learning rate of 0.03 after 100.0 epochs. Which is not an improvement over the accuracy reported by the best model sans bias term.

Table 2: Linear regression results with bias term

Epochs **Learning Rate**	0	10	20	30	40	50	60	70	80	90	100
0.00	0.79	0.29	0.53	0.23	0.47	0.31	0.42	0.79	0.25	0.53	0.55
0.01	0.52	0.83	0.83	0.81	0.81	0.82	0.82	0.82	0.82	0.82	0.81
0.02	0.28	0.76	0.81	0.82	0.81	0.81	0.82	0.80	0.83	0.82	0.81
0.03	0.53	0.81	0.80	0.81	0.81	0.82	0.82	0.78	0.82	0.81	0.82
0.04	0.44	0.85	0.80	0.82	0.81	0.82	0.82	0.83	0.83	0.81	0.82
0.05	0.53	0.83	0.79	0.83	0.80	0.82	0.78	0.82	0.82	0.80	0.85
0.06	0.74	0.83	0.82	0.80	0.82	0.81	0.82	0.82	0.81	0.83	0.81
0.07	0.60	0.72	0.75	0.81	0.81	0.80	0.81	0.81	0.81	0.82	0.82
0.08	0.47	0.80	0.82	0.80	0.82	0.80	0.82	0.82	0.82	0.81	0.82
0.09	0.19	0.85	0.85	0.82	0.83	0.83	0.83	0.83	0.82	0.82	0.82
0.10	0.55	0.83	0.78	0.82	0.83	0.82	0.81	0.82	0.82	0.80	0.82

Visualising the decision boundary

The decision boundaries for the best linear regression models with and without bias are plotted below in figure 2. Since the best linear separating hyper-plane is very likely rest on or near the origin, the addition of the bias term did not improve classification performance in this instance.

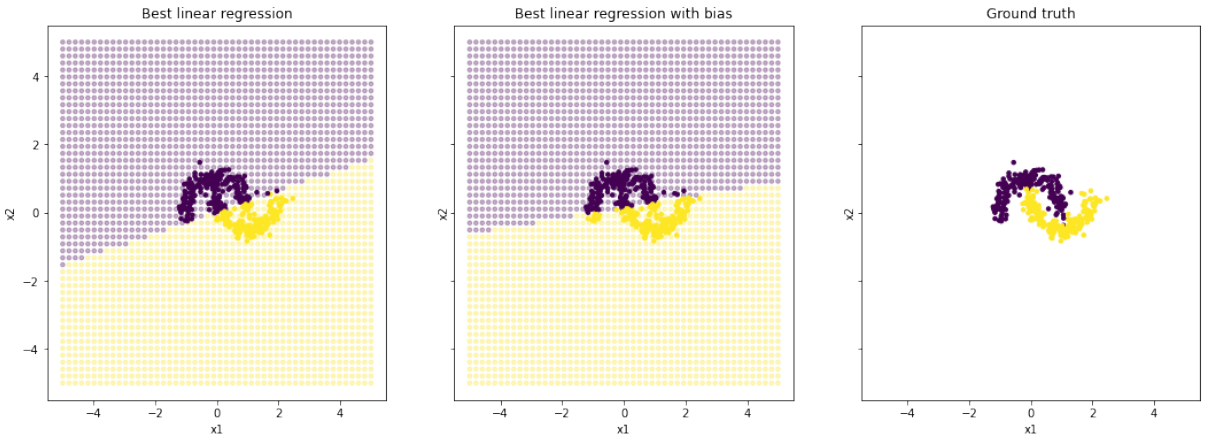

Figure 3: Linear Regression Results Visualised