Logistic Regression on Cleveland Dataset

Descriptions:

Learn a logistic regression model on the data in `clevelandtrain.csv`. Note that the labels in the data set are $0/1$ so you will need to convert those to $-1/+1$ in order to ensure that everything we've done in class is still valid. Use a learning rate of $\eta = 10^{-5}$ and automatically terminate if the magnitude of every element of the gradient is less than 10^{-3}. Initialize the weight vector to a vector of all zeros. Train the model three times with a different bound on the maximum number of iterations each time: 10^4, 10^5, and 10^6. (Note that the termination condition based on the magnitude of the gradient still applies). Use each model to classify the data using a cutoff probability of 0.5.

a For each of the maximum number of iterations, report E_{in} (the cross-entropy error), the binary classification error on both the training and test data sets, and how long the training process took (in seconds). What can you say about the generalization properties of the logistic regression model? How does increasing the maximum number of iterations affect the model's performance?

b Now train and test a logistic regression model using the inbuilt matlab function `glmfit` (learn about and use the "binomial" option, and check the label format). Report E_{in} (the cross-entropy error), the binary classification error on both the training and test data sets, and how long the training process took (in seconds). (Note that `glmfit` returns a weight vector, so you might need to calculate E_{in} yourself.)

c Now get back to your own implementation. Scale each feature by subtracting the mean and dividing by the standard deviation for each of the features in advance of calling the learning algorithm (hint: `zscore` function might be useful). Experiment with the learning rate $\eta = 0.01, 0.1, 1, 4, 6, 7$. Eliminate the iterations-based termination criteria and only terminate when the magnitude of every element of the gradient is less than 10^{-6}. For each learning rate, report E_{in}, the binary classification error on the test data, **the number of iterations required to terminate**, and the time required to train the model. Did normalizing the data affect the performance of the model? What is the effect of changing learning rate η?

(a)

Maximum # of iterations	E_{in} (cross-entropy error)	Binary Classification Error on training data	Binary Classification Error on testing data	Time that training process took (sec)
10^4	0.5847	0.3092	0.3172	0.129904
10^5	0.4937	0.2237	0.2069	1.011837
10^6	0.4354	0.1513	0.1310	9.306209

As we can see from the data above, the binary classification error on testing data is basically the same as the binary classification error on training data. Therefore, the generalization error is pretty small. That is because the model complexity is not very high, and we have enough number of data points.

Also, as the data above shown, when we increase the maximum number of iterations, although we spend more time in training, all of the cross-entropy error, the binary classification error on training data, and the binary classification error on testing data decrease. Namely, increasing the maximum number of iterations will increase the performance of the model.

(b)

	E_{in} (cross-entropy error)	Binary Classification Error on training data	Binary Classification Error on testing data	Time that training process took (sec)
glmfit	0.4074	0.1711	0.1103	0.066556

(c)

eta	E_{in} (cross-entropy error)	Binary Classification Error on testing data	Number of iterations required to terminate	Time that training process took (sec)
0.01	0.4074	0.1103	23368	0.255172
0.1	0.4074	0.1103	2333	0.032722
1	0.4074	0.1103	230	0.007798
4	0.4074	0.1103	54	0.005308
6	0.4074	0.1103	32	0.004477
7	0.4074	0.1103	43	0.004854

From the data above, we can see that normalizing the data will positively affect the performance of the model a little bit. Both the cross-entropy error and the binary classification error on testing data decrease a little bit.

When we increase the learning rate eta, the number of iterations decreases, and we spend less time in training. i.e. we find the optimal solution faster because each step we move becomes larger. However, we cannot increase eta too much. As illustrated in the last two rows, when we increase eta from 6 to 7, the number of iterations actually increases a little bit. Moreover, the cross-entropy error and the binary classification error on testing data remain the same as we increase eta.