## CS6140 Assignment6 Guanglei “Garrett” Wu

### 1, Autoencoder Neural Network

1.1

One layer network with 3 neurons could produce this Neural Network. However, if have only w parameter without b, this network could hardly produce encoder-decoder mechanism(always have at least 1 error). With parameter b added, it could easily produce the network. Could also consider to use batch or momentum to accelerate and smooth the training process.

1.2

The algorithm is to encode an input with 8 classes into 3 binary digits, and decode into the original classes. Since 3 binary digits could represent 2 ^ 3 = 8 different values. The network could implement the encoder-decoder mechanism.

1.3

Hidden units = 1:

Could not converge, is not encoder/decoder.

Hidden units = 2:

Without parameter b: Could not converge, is not encoder/decoder.

With parameter b: Could converge, is encoder/decoder.

Hidden units = 3:

Without parameter b: Could almost converge, is encoder/decoder (with 1 or 2 errors).

With parameter b: Could converge, is encoder/decoder.

Hidden units = 4:

Could converge, is encoder/decoder.

Increase the size of hidden units certainly improves the flexibility of the model to produce an encoder/decoder. Adding interception parameter could also help a little.

1.4

For linear,

For forward propagation:

is replace by , oj =

For back propagation:

Since oj = , we only need to abandon term.

For a unit in output layer:

Since only the output is linear, below is not needed. If hidden layer is also linear, we could apply:

For a unit in the hidden layer:

### 2, Boosting

It seems that the formula in this question has some problems. So I used the formula in the video. Illustrated below:

Consider we have a dataset, first we set Dt to 1/n\_instances.

First round we got a weak classifier of 75% accuracy.

So we have gamma = 0.5

Then, alpha = ln( 1.5 /0.5 ) = 1.1

Then update Dt+1,

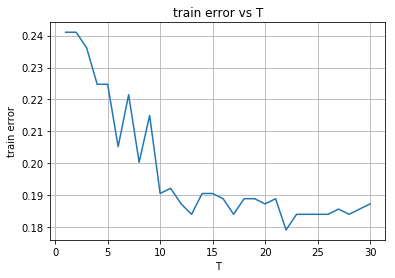
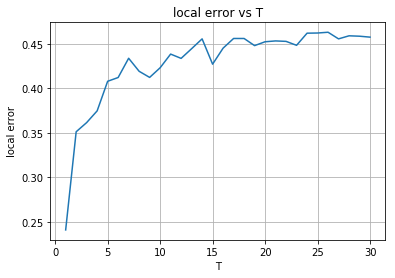
for correct ones we have Dt+1 = exp(-1.1) Dt / Z = 0.3 Dt / Z

for incorrect ones we have Dt+1 = exp(1.1) Dt / Z = 3 Dt / Z

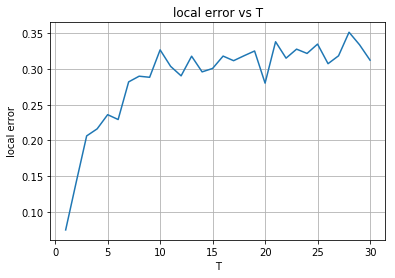
Which means next round we have 10 times weight for incorrect data than correct data. So large difference leads the algorithm to choose the reverse strategy of last round: which correctly classified the misclassified data last round, that with highest weights. However, this movement does not help at all, only lead to choosing the same feature and same threshold with reverse directions, make the outcome back and forth.

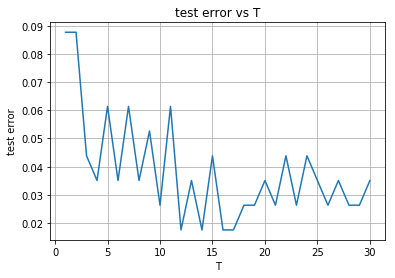
2.3.1

AdaBoost Optimal for: diabetes.csv

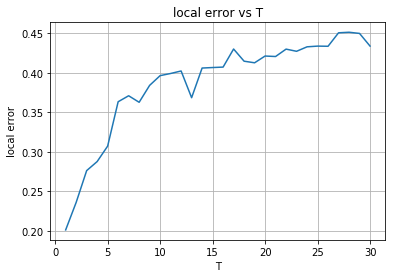


AdaBoost Optimal for: breastcancer.csv





AdaBoost Optimal for: spambase.csv

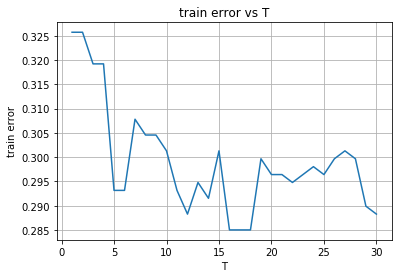
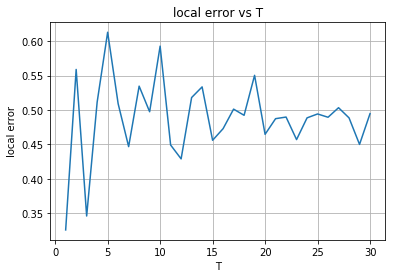




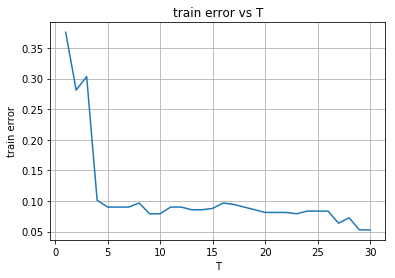
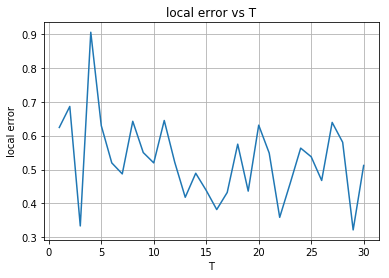


2.3.2

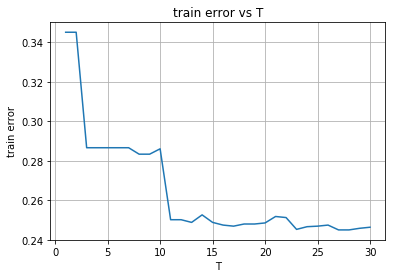
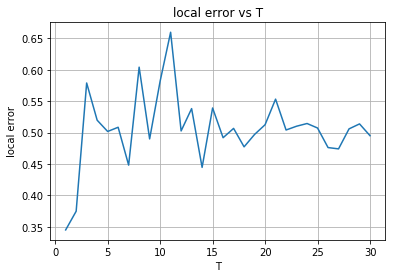
AdaBoost Random for: diabetes.csv



AdaBoost Random for: breastcancer.csv



AdaBoost Random for: spambase.csv



2.3.3

We determine boosting has converged by witnessing that the training error and testing error are flattened. And before testing error goes up.

For the optimal decision stumps, it produces better fit to the training data, means it is less vulnerable to underfitting, but more vulnerable to overfitting. So we tend to choose smaller T, before the testing error goes up.

For the random decision stumps, it fits not as good as optimal decision stumps to the training data. So it is more vulnerable to underfitting. We tend to choose larger T, to get better fitting to data.