**Homework #6**

1. **Bayesian Universe**
   1. Because is fixed, . Derive as follows:

By the procedure (c), substitute those parameters in equation (5.1.1),

Combine (5.1.1) and (5.1.2) we get the likelihood

We look at the maximum likelihood function. Therefore, our goal function should satisfy

Because those parameter do not correlated to can view as constant, the formula equal to

That is same as *linear regression* that we did on Problem 2.3-(1).

* 1. By the Bayesian theorem, we can derive the posteriori as follow:

Where, the likelihood here derive as 5.1(1), that is

Therefore, posterior is

To maximum it, our goal function should satisfy

That is equal to

Consider the negation, rewrite the goal function as

It is same as the *regularized linear regression,* and the relation between is

* 1. We derive the likelihood as follow:

The Objective function max should satisfy the equation

The *Logistic Regression* is equivalently gives the maximum likelihood estimated of .

1. **Power of Adaptive Boosting**
   1. In the first iteration we get
   2. Proof by Induction:
      1. Base: When t = 1, denote , by definition
      2. Inductive: Suppose that

We have

* 1. Express , denote that

|  |  |  |
| --- | --- | --- |
| The sign of s |  |  |
| The sign of v | Zero - Nonzero |  |
| The sign of v |  | Zero – Nonzero |

In each cases we get the result that .

* 1. Derive the step as follow:

By the definition of

* 1. Consider the function . In the region , we have
  2. Consider . In the case , we have , than we consider in case , thus, .
  3. Use the equation above:

* 1. Use the fact above,

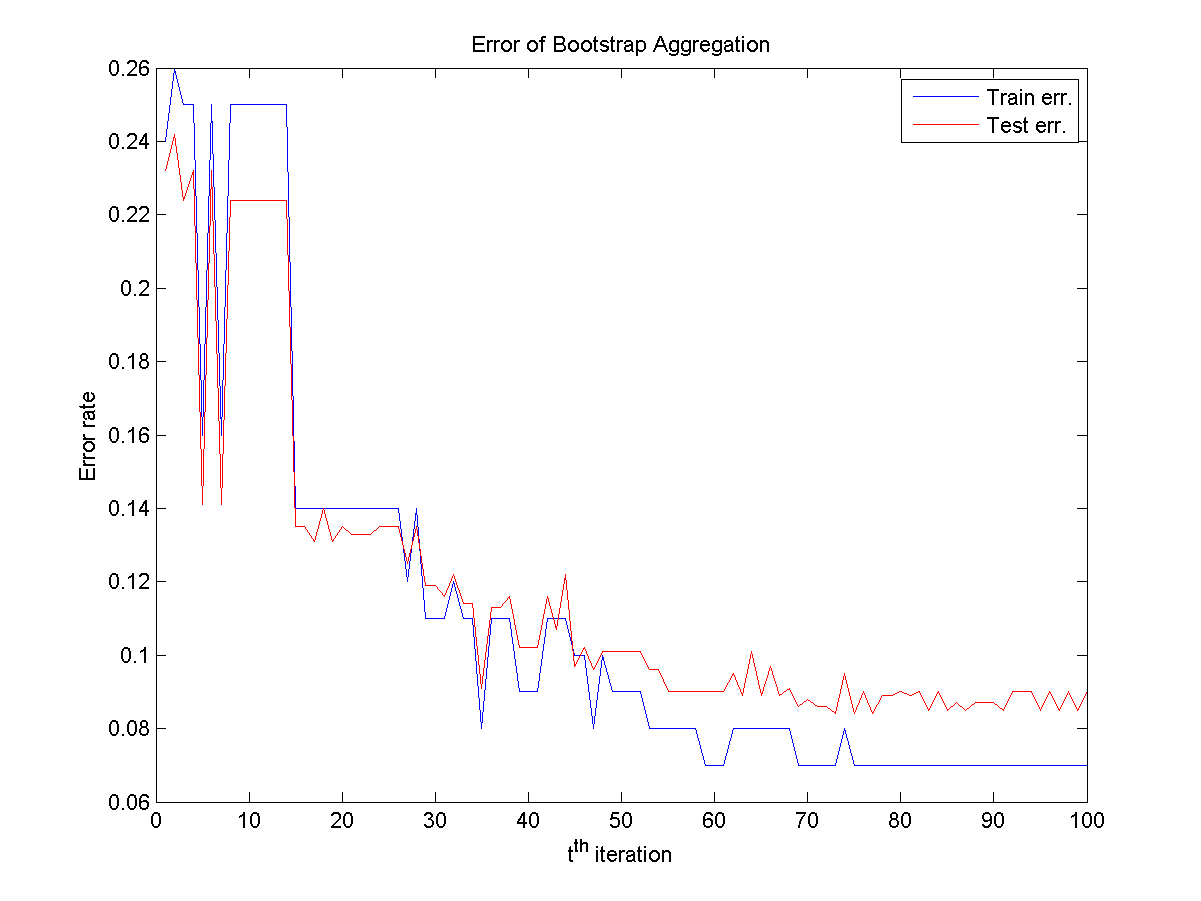
When T grow up, . And we also know

Find a T such that

Then we have

Therefore,

1. **Experiments with Bootstrap Aggregation**
   1. The training error is 0.23, and the test error is 0.255. Our brief finding:
      1. The error always is a fixed value, unless we change .
      2. One thing should be careful, the method that we find should cover [-1, 1], for a training data set range in [0, 1].
   2. The figure we found as follow:



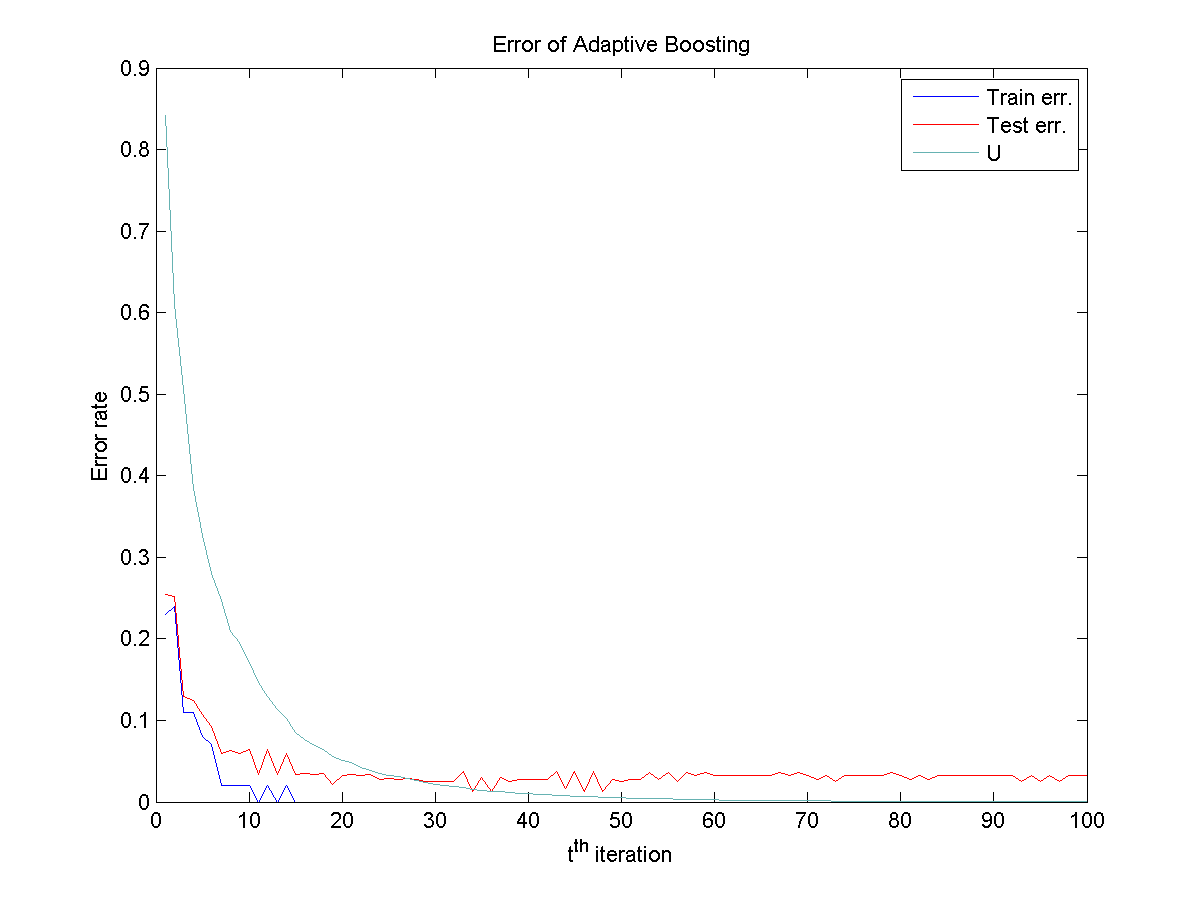
Our brief finding:

* + 1. Training error and testing error have strong correlated. Even a small pick on training error would reflect on testing error.
    2. The performance seems better than expected.
  1. Pseudocode:
     + 1. *D* = Sort data points increasingly in time
       2. = [0, Aggregating the positive value from left to right] in time
       3. = [Aggregating the negative value from right to left, 0] in time
       4. = [Aggregating the positive value from right to left, 0] in time
       5. = [0, Aggregating the negative value from left to right] in time
       6. = column weighted sum on [*L+*; *R-*] in time
       7. = column weighted sum on [*R+*; *L-*] in time
       8. = ( + ) = The argmin value column in in time
       9. = ( + ) = The argmin value column in in time
       10. = min( + ) in time

Our object function is

Therefore, we use the pseudocode above and get minimum a good result in time .

1. **Experiments with Adaptive Boosting**
   1. The figure show as follow:



Our brief finding:

* + 1. Training error `seems’ always less than testing error, and it would converge to 0.
    2. U `seems’ has an inverse proportion with t.
  1. Brief finding:
     1. Adaptive Boosting algorithm (AdaBoost) has a better result than Bootstrap Aggregation algorithm (Bagging).
     2. AdaBoost has a zero training error performance, but Bagging does not have that.
     3. Both of algorithms start with a not bad error rate, Bagging would go up but AdaBoost won’t.