CPT_S 570: Homework #4

Instructor: Jana Doppa

Sheng Guan

Problem 1

Answer:
In this paper, the key idea is – usually the heuristic algorithms such as gradient descent(for neural networks) and greedy search (for decision trees) can be viewed as suboptimal if its objective is to find the smallest model that fits the data, but it can be viewed as optimal if its objective is to find a model that minimizes some combination of model size plus a penalty term that corrects for the difference between the training data and the ultimate test data. So far, the support for this view is primarily empirical.
Learning algorithms essentially operate by searching some space of functions (hypothesis class) for a function that fits the given data. This search usually is formalized by defining an objective function. However, finding such a function that minimizes the objective function is NP-hard. What's worse is that if we work too hard to find the very best fit to the training data, there's a risk that we will fit the noise in the data. That's the reason we need to augment the objective function with various penalty terms(e.g., regularization methods, minimum-description-length methods, generalized cross-validation, etc.) By considering this, simple gradient descent or greedy search usually perform better on real test datasets.
Because the original optimization problems were intractable, but these suboptimal approaches usually can give us a polynomial-time algorithm that does the right thing. By "undercomputing" we avoid "overfitting."

Problem 2

Answer:
This paper reviews the methods including Bayesian averaging, error-correcting output coding, Bagging, and boosting, etc. and explains why ensembles can often perform better than any single classifier. Besides, this paper uncovers the reasons that Adaboost does not overfit rapidly.
A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse. Accurate means the classifier has an error rate of better than random guessing on a new testing example. Diverse means two classifiers make different errors on new data points. Based on the fact that the error rate of more hypotheses are simultaneously wrong is much less than the one of individual hypotheses, one key to successful ensemble methods is to construct individual classifiers with error rates below 0.5 whose errors are at least somewhat uncorrelated. The authors further illustrates that there are three fundamental reasons that make it often possible to construct very good ensembles.
(1) The statistical problem arises when the amount of training data available is too small compared to the size of the hypothesis space. Without sufficient data, the learning algorithm can find many different hypotheses in H that all give the same accuracy on the training data. By constructing an ensemble out of all of these accurate classifiers, the algorithm can "average" their votes and reduce the risk of choosing the wrong classifier.
(2) An ensemble constructed by running the local search from different starting points may provide a better approximation to the true unknown function than any of the individual classifiers.
(3) The true function f cannot be represented by any of the hypotheses in H. By forming weighted sums of hypotheses drawn from H, it may be possible to expand the space of representable functions.
The authors review general ensemble methods:
(1) Bayesian Voting: Enumerating the Hypotheses; The most idealized aspect of the Bayesian analysis is the prior belief $P(h)$. If this prior completely captures all of the knowledge that we have about f before we obtain S, then by definition we cannot do better. In practice, it is often difficult to construct a space H and assign a prior P(h) that captures our prior knowledge adequately. Often H and $P(h)$ are chosen for computational convenience.In such cases, the Bayesian committee is not optimal, and other ensemble methods may produce better results.

(2) Manipulating training set –Bagging with bootstrap aggregation, cross-validated committees, AdaBoost algorithm. This technique works especially well for unstable learning algorithms- algorithms whose output classifier undergoes major changes in response to small changes in the training data. Decision tree, neural network, and rule learning algorithms are unstable.
(3) Manipulating the Input features, this technique only works when the input features are highly redundant.
(4) Manipulating the Output targets, error-correcting output coding and this coding method has implementation simplicity but yields an excellent ensemble classification performance.
(5) Injecting randomness, the randomness often combines with other learning algorithm to achieve better performance.
In the empirical study, the authors compare with three algorithms: ADABOOST, Bagging, and Randomized trees. Bagging accomplishes this by manipulating the input data, and Randomization directly alters the choices of C4.5. Both Bagging and Randomization are sampling from the space of all possible hypotheses with a bias toward hypotheses that give good accuracy on the training data. In contrast, ADABOOST constructs each new decision tree to eliminate "residual" errors that have not been properly handled by the weighted vote of the previously-constructed trees. ADABOOST is directly trying to optimize the weighted vote. Hence, it is making a direct assault on the representational problem.
Another angle to see ADABOOST is that this algorithm can be viewed as a stage-wise algorithm for minimizing a particular error function.

$$\sum_i exp(-y_i \sum_l w_l h_l(x_i)) \tag{1}$$

which is the negative exponential of the margin of the weighted voted classifier.
In low-noise cases, ADABOOST gives good performance, because it is able to optimize the ensemble without overfitting. However, in high-noise cases, ADABOOST puts a large amount of weight on the mislabeled examples, and this leads it to overfit very badly. Bagging and Randomization do well in both the noisy and noise-free cases since they are focusing on the statistical problem. In very large datasets, Randomization can be expected to do better than Bagging.
The reason why ADABOOST doesn't overfit more often may lie in the "stage-wise" nature of ADABOOST. In each iteration, it reweights the training examples, constructs a new hypothesis, and chooses a weight w_l for that hypothesis. It never "back up" and modifies the previous choices of hypotheses or weights that it has made to compensate for this new hypothesis. The authors construct an aggressive version of ADABOOST which reconsiders the weights of all of the learned hypotheses after each new hypothesis is added. Then it reweights the training examples to reflect the revised hypothesis weights. The comparison shows that the exceptionally good performance of Standard ADABOOST is due to the stage-wise optimization process, which is slow to fit the data.

Problem 3

Answer:
This paper reviews five approximate statistical tests for determining whether one learning algorithm outperforms another on a particular learning task. These tests are compared experimentally to determine their probability of incorrectly detecting a difference when no difference exists (type I error). The paper also measures the power (ability to detect algorithm differences when they do exist) of these tests. To design and evaluate statistical tests, the first step is to identify the sources of variation that must be controlled by each test. For the case we are considering, there are four sources of variation.
(1) there is the random variation in the selection of the test data that is used to evaluate the learning algorithms.
(2) the selection of the training data.
(3) internal randomness in the learning algorithm.
(4) the test data points can be randomly mislabeled.

To account for test-data variation and the possibility of random classification error, the statistical procedure must consider the size of the test set and the consequences of changes in the test set. To account for training-data variation and internal randomness, the statistical procedure must execute the learning algorithm multiple times and measure the variation in accuracy of the resulting classifiers.
This paper describes five statistical tests – McNemar's test, a test for the difference of two proportions, the resampled t test, the cross-validated t test, and 5*2cv test. The simulation result shows that only McNemar's test, cross-validated t test and 5*2cv test have acceptable Type I error. Further a set of experiments using real learning algorithms on realistic data sets show that the cross-validated t test has consistently elevated Type I error. The difference-of-proportions test has acceptable type I error, but low power. Both of the remaining two tests have good Type I error and reasonable power. The 5*2cv test is slightly more powerful than McNemar's test, but more expensive to perform.
The paper concludes that the 5*2cv test is the test of choice for inexpensive learning algorithms, but that McNemar's test is better for more expensive algorithms.
(1) McNemar's test, McNemar's test is based on a χ^2 test for goodness-of-fit that compares the distribution of counts expected under the null hypothesis to the observed counts. However, this test has two shortcomings. First, it does not directly measure variability due to the choice of the training set or the internal randomness of the learning algorithm. Secondly, it does not directly compare the performance of the algorithms on training sets of size $|S|$, but only on sets of size $|R|$,which must be substantially smaller than $|S|$ to ensure a sufficiently large test set.
(2) A test for the difference of two proportions,

$$p_A = (n_{00} + n_{01})/n$$
$$p_B = (n_{00} + n_{10})/n$$

$$z = \frac{p_A - p_B}{\sqrt{2p(1-p)/n}} \tag{2}$$

There are several problems with this test. First, because p_A and p_B are each measured on the same test set T, they are not independent. Second, the test shares the drawbacks of McNemar's test.
(3) The resampled paired t test,

$$t = \frac{\overline{p} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(p^{(i)} - \overline{p})^2}{n-1}}},$$

There are many drawbacks of this approach. First, the individual differences $p^{(i)}$ will not have a normal distribution, because $p_A^{(i)}$ and $p_B^{(i)}$ are not independent. Second, the $p^{(i)}$'s are not independent, because the test sets in the trials overlap.
(4) The k-fold cross-validated paired t test, we randomly divide S into k disjoint sets of equal size, $T_1, ..., T_k$. We then conduct k trials. In each trial, the test set is T_i, and the training set is the union of all of the other T_j, $j \neq i$. The same t statistic is computed. The advantage of this approach is that each test set is independent of the others. However, this test still suffers from the problem that the training sets overlap. This overlap may prevent this statistical test from obtaining a good estimate of the amount of variation that would be observed if each training set were completely independent of previous training sets.
(5)The 5*2cv paired t test, first we employ the normal approximation to the binomial distribution. Second,

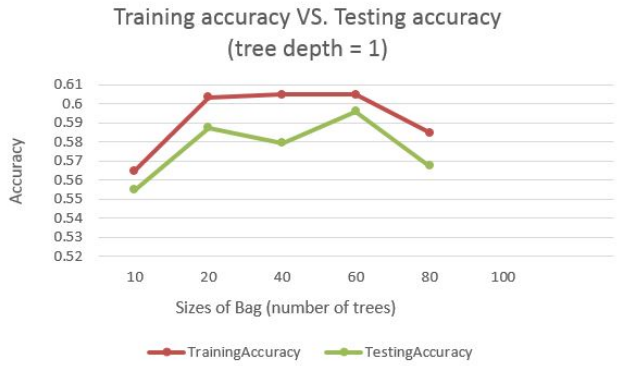
we assume pairwise independence of $p_i^{(1)}$ and $p_i^{(2)}$ for all i. Third, we assume independence between the s_i's. Finally, we assume independence between the numerator and denominator of the $\tilde{t}$ statistic. The 5*2cv test requires a large number of independence assumptions that are known to be violated.
Because of the poor behavior and high cost of (3)The resampled paired t test, we exclude it from further analysis.
Power Measurement, if one's goal is to detect whether there is a difference between two learning algorithms, then the power of the statistical test is important. The power of a test is the probability that it will reject the null hypothesis when the null hypothesis is false. The experiment results show that if the goal is to be confident that there' no difference between two algorithms, then the cross-validated t test is the test of choice even though its Type I error is unacceptable. Of the tests with acceptable Type I error, the 5*2cv paired t test is the most powerful.
The experiments lead us to recommend either the 5*2cv paired t test for situations in which the learning algorithms are efficient enough to run ten times or McNemar's test for situations where the learning algorithms can be run only once based on the power consideration. The resampled t test should never be employed. The cross-validated t test should be employed with caution since it has an elevated probability of Type I error.

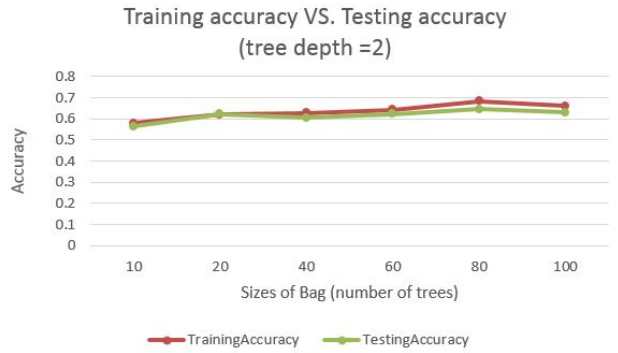
Problem 4

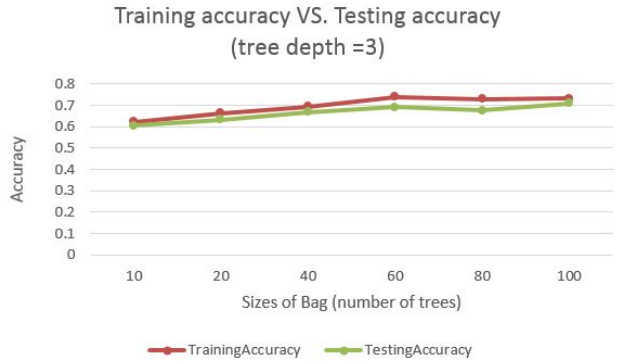
Answer:
For computational efficiency, we set different thresholds for the change in log-likelihood. If k=3 or 4, the threshold = 0.0001. Else k=5, the threshold = 0.001.
When value k =3:
$\mu_1 = 4.79483908421633, \sum_1 = 0.6187511739234698, \alpha_1 = 0.15262668324774836$
$\mu_2 = 6.088038404497007, \sum_2 = 0.5014641521524771, \alpha_2 = 0.17280783250509985$
$\mu_3 = 20.306133972128382, \sum_3 = 28.09685195031509, \alpha_3 = 0.674565484247151$
When value k=4:
$\mu_1 = 15.449160787715165, \sum_1 = 0.9671159494369864, \alpha_1 = 0.33333333334024706$
$\mu_2 = 4.615852032160412, \sum_2 = 0.5791801907493386, \alpha_2 = 0.10910223814121675$
$\mu_3 = 5.943986828937934, \sum_3 = 0.6723845205195613, \alpha_3 = 0.22423109518839193$
$\mu_4 = 25.486654429329228, \sum_4 = 0.9980966181791336, \alpha_4 = 0.33333333333014475$
When value k=5:
$\mu_1 = 4.5760055254447325, \sum_1 = 0.5606455469999461, \alpha_1 = 0.046729509067420585$
$\mu_2 = 5.016614620123633, \sum_2 = 0.6993211734352556, \alpha_2 = 0.10309077685028734$
$\mu_3 = 6.1600972513021945, \sum_3 = 0.47428401250483654, \alpha_3 = 0.14135694061716528$
$\mu_4 = 5.31421913927234, \sum_4 = 0.7173102767178843, \alpha_4 = 0.03424327683114846$
$\mu_5 = 20.305906572834463, \sum_5 = 28.098575440550015, \alpha_5 = 0.6745794966339766$

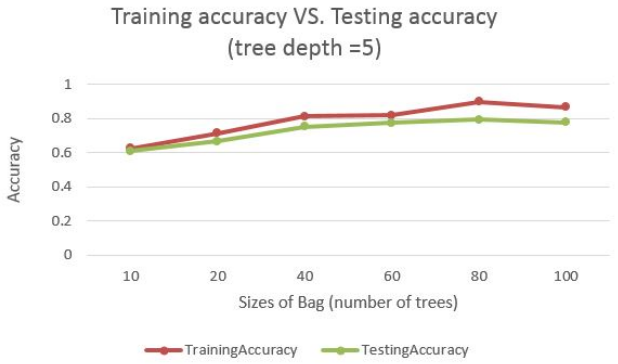
Problem 5

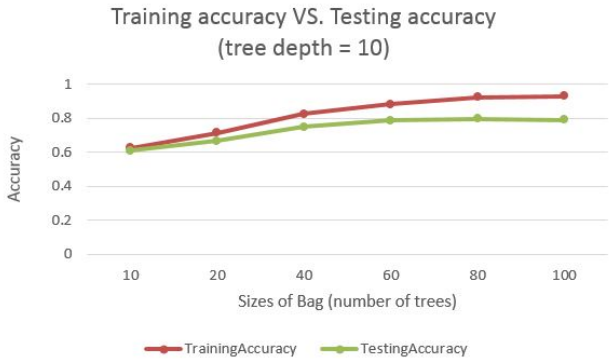
Answer: a.
The result is shown as follows.


Training accuracy VS. Testing accuracy
(tree depth = 1)
Accuracy
0.61
0.6
0.59
0.58
0.57
0.56
0.55
0.54
0.53
0.52
10 20 40 60 80 100
Sizes of Bag (number of trees)
TrainingAccuracy
TestingAccuracy

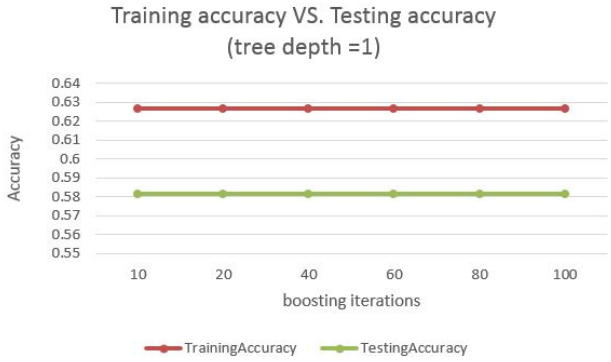

Training accuracy VS. Testing accuracy
(tree depth =2)
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10 20 40 60 80 100
Sizes of Bag (number of trees)
TrainingAccuracy
TestingAccuracy

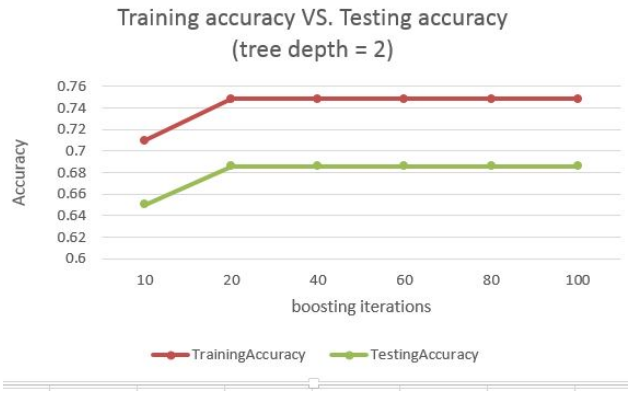

Training accuracy VS. Testing accuracy
(tree depth =3)
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10 20 40 60 80 100
Sizes of Bag (number of trees)
TrainingAccuracy
TestingAccuracy

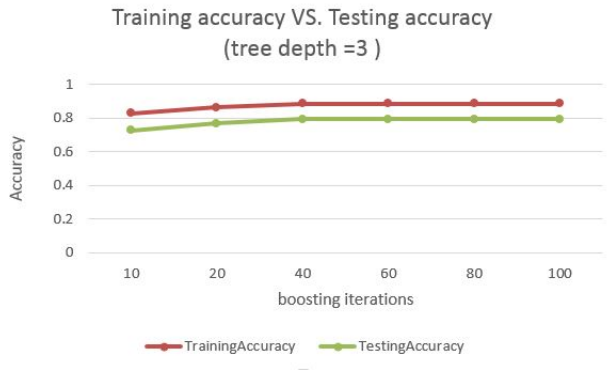

Training accuracy VS. Testing accuracy
(tree depth =5)
Accuracy
1
0.8
0.6
0.4
0.2
0
10 20 40 60 80 100
Sizes of Bag (number of trees)
TrainingAccuracy
TestingAccuracy



There are several observations:
(1) For this dataset, it seems that when increasing the tree depth, it is helpful to increase the testing accuracy since the best testing accuracy obtained in depth =10;
(2) When fixing the depth and the number of trees increases, it doesn't monotonically get a higher testing accuracy, which indicates we need to locate the optimal number of trees value within certain range to optimize this parameter;
(3) The testing accuracy in general is lower than training accuracy for this dataset when the tree depth is fixed.
b.
The result is shown as follows.







There are several observations:
(1) For this dataset, it seems that when increasing the tree depth, it is helpful to increase the testing accuracy since the best testing accuracy obtained in depth =3. However, since we only test to depth =3, this conclusion is very weak;
(2) When fixing the depth, in general the number of iterations is not much helpful to improve the testing accuracy;
(3) The testing accuracy is lower than training accuracy for this dataset when the tree depth is fixed.