CMSC726 Spring 2006: Bias-Variance Tradeoff and Ensemble Methods

readings: handed out in class
material from:
Tom Dietterich, http://www.cs.orst.edu/~tgd
Rich Maclin

Outline
- Bias-Variance Decomposition for Regression
- Bias-Variance Analysis of Learning Algorithms
- Ensemble Methods
- Effect of Bagging on Bias and Variance
- Effect of Boosting on Bias and Variance
- Summary and Conclusion

Intuition 1
- The goal in learning is not to learn an exact representation of the training data itself, but to build a statistical model of the process which generates the data. This is important if the algorithm is to have good generalization performance
- We saw that
  - models with too few parameters can perform poorly
  - models with too many parameters can perform poorly
- Need to optimize the complexity of the model to achieve the best performance
- One way to get insight into this tradeoff is the decomposition of generalization error into bias + variance
  - a model which is too simple, or too inflexible, will have a large bias
  - a model which has too much flexibility will have high variance

Intuition
- bias:
  - measures the accuracy or quality of the algorithm
  - high bias means a poor match
- variance:
  - measures the precision or specificity of the match
  - a high variance means a weak match
- We would like to minimize each of these
- Unfortunately, we can't do this independently, there is a trade-off

Bias-Variance Analysis in Regression
- True function is $y = f(x) + \varepsilon$
  - where $\varepsilon$ is normally distributed with zero mean and standard deviation $\sigma$
- Given a set of training examples, $\{(x_i, y_i)\}$, we fit an hypothesis $h(x) = w \cdot x + b$ to the data to minimize the squared error
  $\sum (y_i - h(x_i))^2$

Example: 20 points
$y = x + 2 \sin(1.5x) + N(0,0.2)$
Now, given a new data point \( x^* \) (with observed value \( y^* = f(x^*) + \epsilon \)), we would like to understand the expected prediction error:

\[
E[(y^* - h(x^*))^2]
\]

Classical Statistical Analysis

Imagine that our particular training sample \( S \) is drawn from some population of possible training samples according to \( P(S) \).
- Compute \( E_P[(y^* - h(x^*))^2] \)
- Decompose this into “bias”, “variance”, and “noise”

Bias, Variance, and Noise

- Variance: \( E[(h(x^*) - \bar{h}(x^*))^2] \)
  - Describes how much \( h(x^*) \) varies from one training set \( S \) to another
- Bias: \( [h(x^*) - f(x^*)] \)
  - Describes the average error of \( h(x^*) \).
- Noise: \( E[(y^* - f(x^*))^2] = E[\epsilon^2] = \sigma^2 \)
  - Describes how much \( y^* \) varies from \( f(x^*) \)

Bias-Variance-Noise Decomposition

\[
E[(h(x^*) - y^*)^2] = E[h(x^*)^2] - 2E[h(x^*)y^*] + E[y^*^2]
\]

\[
= E[h(x^*)^2] - 2E[h(x^*)E[y^*]] + E[y^*^2]
\]

\[
= E[h(x^*)^2] - 2E[h(x^*)f(x^*)] + f(x^*)^2
\]

\[
= \left( E[h(x^*)^2] - f(x^*)^2 \right) - 2\left( E[h(x^*)] - f(x^*) \right)^2
\]

Expected prediction error = Variance + Bias^2 + Noise^2
Measuring Bias and Variance

- In practice (unlike in theory), we have only ONE training set S.

- We can simulate multiple training sets by bootstrap replicates:
  - \( S' = \{x \mid x \text{ is drawn at random with replacement from } S\} \) and \( |S'| = |S| \).

- Procedure for Measuring Bias and Variance:
  - Construct B bootstrap replicates of S (e.g., B = 200): \( S_1, \ldots, S_B \)
  - Apply learning algorithm to each replicate \( S_b \) to obtain hypothesis \( h_b \)
  - Let \( T_b = S \setminus S_b \) be the data points that do not appear in \( S_b \) (out of bag points)
  - Compute predicted value \( h_b(x) \) for each \( x \) in \( T_b \).
Estimating Bias and Variance (continued)

- For each data point $x$, we will now have the observed corresponding value $y$ and several predictions $y_1, \ldots, y_K$.
- Compute the average prediction $h$.
- Estimate bias as $(h - y)$.
- Estimate variance as $\sum_k (y_k - h)^2/(K - 1)$.
- Assume noise is 0.

Approximations in this Procedure

- Bootstrap replicates are not real data.
- We ignore the noise:
  - If we have multiple data points with the same $x$ value, then we can estimate the noise.
  - We can also estimate noise by pooling $y$ values from nearby $x$ values.

Ensemble Learning

- What is an ensemble?
- Why use an ensemble?
- Selecting component classifiers.
- Selecting combining mechanism.
- Some results.

Ensemble Learning Methods

- Given training sample $S$.
- Generate multiple hypotheses, $h_1, h_2, \ldots, h_L$.
- Optionally: determining corresponding weights $w_1, w_2, \ldots, w_L$.
- Classify new points according to $\sum w_i h_i > \theta$.

A Classifier Ensemble

![Diagram of a classifier ensemble]

Key Ensemble Questions

- Which components to combine?
  - Different learning algorithms.
  - Same learning algorithm trained in different ways.
  - Same learning algorithm trained the same way.
- How to combine classifications?
  - Majority vote.
  - Weighted (confidence of classifier) vote.
  - Weighted (confidence in classifier) vote.
  - Learned combiner.
- What makes a good (accurate) ensemble?
What Makes a Good Ensemble?

Krogh and Vedelsby, 1995
Can show that the accuracy of an ensemble is mathematically related:
\[ \hat{E} = E - D \]
\( \hat{E} \) is the error of the entire ensemble
\( E \) is the average error of the component classifiers
\( D \) is a term measuring the diversity of the components
Effective ensembles have accurate and diverse components

Ensemble Mechanisms - Components

- Separate learning methods
  - not often used
  - very effective in certain problems (e.g., protein folding, Rost and Sander, Zhang)
- Same learning method
  - generally still need to vary something externally
    - exception, some good results with neural networks
  - most often, data set used for training varied:
    - Bagging (Bootstrap Aggregating), Breiman
    - Boosting, Freund & Schapire
    - Ada, Freund & Schapire
    - Arcing, Breiman

Ensemble Mechanisms - Combiners

- Voting
- Averaging (if predictions not 0,1)
- Weighted Averaging
  - base weights on confidence in component
- Learning combiner
  - Stacking, Wolpert
  - region combiner
  - RegionBoosting, Maclin
    - piecewise combiner

Bagging

Varies data set
Each training set a bootstrap sample
bootstrap sample - select set of examples (with replacement) from original sample
Algorithm:
for \( k = 1 \) to \# of samples
  \( \text{train}^k = \text{bootstrap sample of train set} \)
  create classifier using \text{train}^k as training set
combine classifications using simple voting

Bagging: Bootstrap Aggregating

- For \( b = 1, \ldots, B \) do
  - \( S_b = \text{bootstrap replicate of } S \)
  - Apply learning algorithm to \( S_b \) to learn \( h_b \)
- Classify new points by unweighted vote:
  - \( \sum_{b=1}^{B} h_b(x) / B > 0 \)

- Bagging makes predictions according to
  \( y = \sum_{b=1}^{B} h_b(x) / B \)
- Hence, bagging’s predictions are \( h(x) \)
Estimated Bias and Variance of Bagging

- If we estimate bias and variance using the same B bootstrap samples, we will have:
  - Bias = (h – y) [same as before]
  - Variance = \( \Sigma_i (y – h)^2/(K – 1) = 0 \)
- Hence, according to this approximate way of estimating variance, bagging removes the variance while leaving bias unchanged.
- In reality, bagging only reduces variance and tends to slightly increase bias.

Bagging Decision Trees
(Freund & Schapire)

Bias/Variance Heuristics

- Models that fit the data poorly have high bias: “inflexible models” such as linear regression, regression stumps
- Models that can fit the data very well have low bias but high variance: “flexible” models such as nearest neighbor regression, regression trees
- This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

Weak Learning

Schapire showed that a set of weak learners (learners with > 50% accuracy, but not much greater) could be combined into a strong learner

Idea: weight the data set based on how well we have predicted data points so far
- data points predicted accurately - low weight
- data points mispredicted - high weight

Result: focuses components on portion of data space not previously well predicted

Boosting - Ada

Varies weights on training data

Algorithm:

- for each data points: weight \( w_i \) to 1..#datapoints
- for \( k = 1 \) to #classifiers
  - generate classifier \( k \) with current weighted train set
  - \( e_k = \) weighted sum of \( w_i \)'s of misclassified points
  - \( \beta_k = (1 - e_k)/e_k \)
  - multiply weights of all misclassified points by \( \beta_k \)
  - normalize weights to sum to 1
- combine: weighted vote, weight for classifier \( k \) is \( \log(\beta_k^{-1}) \)

Boosting

Input: a set \( S \) of \( m \) labeled examples \( S = \{ (x_i, y_i), i = 1..m \} \)
- labels \( y_i \in Y = \{1,...,K\} \)
- Learn (a learning algorithm) a constant \( \epsilon \)

1. Initialize for all \( i \): \( w_i(1) = 1/m \)
2. for \( t = 1 \) to \( T \) do
3. for all \( i : p_t(x_i) = w_i(t)/(\sum_i w_i(t)) \)
4. \( h_t = \text{Learn}(w) \)
5. \( \epsilon_t = \sum_i w_i(t) \cdot [1 - \text{Label}(x_i) = y_i] \)
6. \( L_t = \epsilon_t \cdot (1 - \epsilon_t) \)
7. if \( L_t > 1/2 \) then
8. \( t = t + 1 \)
9. else
10. \( \Delta = \epsilon_t/(1 - \epsilon_t) \)
11. for all \( i : w_i(t+1) = w_i(t) \cdot e^{-\Delta \cdot p_t(x_i)/m} \)
12. end

Output: \( h_T(x) = \text{sign} \left( \sum_{t=1}^{T} \log(1/\beta_t) \cdot [h_t(x) = y] \right) \)
Boosting vs Bagging
(Freund & Schapire)

**Sample data set (like Bagging), but probability of data point being chosen weighted (like Boosting)**

\[
\text{probability of selecting point } i = \frac{1 + m_i}{\sum_{j=0}^{N-1} 1 + m_j}
\]

Value 4 chosen empirically

Combine using voting

---

Some Results - BP, C4.5 Components

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C4.5</th>
<th>BP</th>
<th>BagC4</th>
<th>BagBP</th>
<th>AdaC4</th>
<th>AdaBP</th>
<th>ArcC4</th>
<th>ArcBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter</td>
<td>14.0</td>
<td>18.0</td>
<td>7.0</td>
<td>10.5</td>
<td>4.1</td>
<td>5.7</td>
<td>3.9</td>
<td>4.6</td>
</tr>
<tr>
<td>segment</td>
<td>3.7</td>
<td>6.6</td>
<td>3.0</td>
<td>5.4</td>
<td>1.7</td>
<td>3.5</td>
<td>1.5</td>
<td>3.3</td>
</tr>
<tr>
<td>promoter</td>
<td>12.8</td>
<td>5.3</td>
<td>10.6</td>
<td>4.0</td>
<td>6.8</td>
<td>4.5</td>
<td>6.4</td>
<td>4.6</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.6</td>
<td>2.3</td>
<td>0.6</td>
<td>0.8</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>splice</td>
<td>5.9</td>
<td>4.7</td>
<td>5.4</td>
<td>3.9</td>
<td>5.1</td>
<td>4.0</td>
<td>5.3</td>
<td>4.2</td>
</tr>
<tr>
<td>breastc</td>
<td>5.0</td>
<td>3.4</td>
<td>3.7</td>
<td>3.4</td>
<td>3.3</td>
<td>3.5</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>housev</td>
<td>3.6</td>
<td>4.9</td>
<td>3.6</td>
<td>4.1</td>
<td>5.0</td>
<td>5.1</td>
<td>4.8</td>
<td>5.3</td>
</tr>
</tbody>
</table>

---

Some Theories on Bagging/Boosting

Error = noise error + Bias + Variance

Theories:
- Bagging can reduce variance part of error
- Boosting can reduce variance AND bias part of error
- Bagging will hardly ever increase error
- Boosting may increase error
- Boosting susceptible to noise
- Boosting increases margins

---

Combiner - Stacking

Idea:
- generate component (level 0) classifiers with part of the data (half, three quarters)
- train combiner (level 1) classifier to combine predictions of components using remaining data
- retrain component classifiers with all of training data
- In practice, often equivalent to voting

---

Combiner - RegionBoost

- Train “weight” classifier for each component classifier
- “weight” classifier predicts how likely point will be predicted correctly
- “weight” classifiers: k-Nearest Neighbor, Backprop
- Combiner, generate component classifier prediction and weight using corresponding “weight” classifier
- Small gains in accuracy
Sources of Bias and Variance

- Bias arises when the classifier cannot represent the true function— that is, the classifier underfits the data.
- Variance arises when the classifier overfits the data.
- There is often a tradeoff between bias and variance.

Effect of Algorithm Parameters on Bias and Variance

- k-nearest neighbor: increasing $k$ typically increases bias and reduces variance.
- Decision trees of depth $D$: increasing $D$ typically increases variance and reduces bias.
- RBF SVM with parameter $\sigma$: increasing $\sigma$ increases bias and reduces variance.

Effect of Bagging

- If the bootstrap replicate approximation were correct, then bagging would reduce variance without changing bias.
- In practice, bagging can reduce both bias and variance:
  - For high-bias classifiers, it can reduce bias.
  - For high-variance classifiers, it can reduce variance.

Effect of Boosting

- In the early iterations, boosting is primarily a bias-reducing method.
- In later iterations, it appears to be primarily a variance-reducing method.

Other Approaches

- Error Correcting Output Codes
- Mixture of Experts
- Cascading Classifiers
- Many others…