Variance Reduction and Ensemble Methods

Nicholas Ruozzi
University of Texas at Dallas

Based on the slides of Vibhav Gogate and David Sontag
Last Time

- PAC learning
- Bias/variance tradeoff
  - small hypothesis spaces (not enough flexibility) can have high bias
  - rich hypothesis spaces (too much flexibility) can have high variance
- Today: more on this phenomenon and how to get around it
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
  - 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
• True function is $y = f(x) + \epsilon$

• Where noise, $\epsilon$, is normally distributed with zero mean and standard deviation $\sigma$

• Given a set of training examples, $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$, we fit a hypothesis $g(x) = w^T x + b$ to the data to minimize the squared error

$$\sum_{i} [y^{(i)} - g(x^{(i)})]^2$$
Sample 20 points from \( f(x) = x + 2 \sin(1.5x) + N(0,0.2) \)
2-D Example

50 fits (20 examples each)

![Graph showing 50 fits with 20 examples each]
Bias-Variance Analysis

• Given a new data point $x'$ with observed value $y' = f(x') + \epsilon$, want to understand the expected prediction error.

• Suppose that training samples are drawn independently from a distribution $p(S)$, want to compute the expected error of the estimator

$$E[ (y' - g_S(x'))^2 ]$$
• Variance of a random variable, $Z$

$$Var(Z) = E[(Z - E[Z])^2]$$

$$= E[Z^2 - 2ZE[Z] + E[Z]^2]$$

$$= E[Z^2] - E[Z]^2$$

• Properties of $Var(Z)$

$$Var(aZ) = E[a^2Z^2] - E[aZ]^2 = a^2Var(Z)$$
Bias-Variance-Noise Decomposition

\[ E \left[ (y' - g_s(x'))^2 \right] = E[g_s(x')^2 - 2g_s(x')y' + y'^2] \]

\[ = E[g_s(x')^2] - 2E[g_s(x')]E[y'] + E[y'^2] \]

\[ = Var(g_s(x')) + E[g_s(x')^2] - 2E[g_s(x')]f(x') + Var(y') + f(x')^2 \]

\[ = Var(g_s(x')) + (E[g_s(x')] - f(x'))^2 + Var(\epsilon) \]

\[ = Var(g_s(x')) + (E[g_s(x')] - f(x'))^2 + \sigma^2 \]
Bias-Variance-Noise Decomposition

\[
E \left[ (y' - g_S(x'))^2 \right] = E[g_S(x')^2 - 2g_S(x')y' + y'^2]
\]

\[
= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2]
\]

\[
= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \\
+ \text{Var}(y') + f(x')^2
\]

\[
= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon)
\]

\[
= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2
\]

The samples $S$ and the noise $\epsilon$ are independent.
Bias-Variance-Noise Decomposition

\[ E \left[ (y' - g_S(x'))^2 \right] = E[g_S(x')^2 - 2g_S(x')y' + y'^2] \]

\[ = E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2] \]

\[ = \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') \]
\[ + \text{Var}(y') + f(x')^2 \]

\[ = \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon) \]

\[ = \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2 \]
Bias-Variance-Noise Decomposition

\[
E \left[ (y' - g_S(x'))^2 \right] = E[g_S(x')^2 - 2g_S(x')y' + y'^2]
\]
\[
= E[g_S(x')^2] - 2E[g_S(x')]E[y'] + E[y'^2]
\]
\[
= \text{Var}(g_S(x')) + E[g_S(x')]^2 - 2E[g_S(x')]f(x') + \text{Var}(y') + f(x')^2
\]
\[
= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \text{Var}(\epsilon)
\]
\[
= \text{Var}(g_S(x')) + (E[g_S(x')] - f(x'))^2 + \sigma^2
\]
Bias-Variance-Noise Decomposition

\[
E \left[ (y' - g_S(x'))^2 \right] = E\left[ g_S(x')^2 - 2g_S(x')y' + y'^2 \right]
\]

\[
= E\left[ g_S(x')^2 \right] - 2E\left[ g_S(x') \right]E\left[ y' \right] + E\left[ y'^2 \right]
\]

\[
= Var (g_S(x')) + E\left[ g_S(x') \right]^2 - 2E\left[ g_S(x') \right]f(x')
+ Var (y') + f(x')^2
\]

\[
= Var (g_S(x')) + (E\left[ g_S(x') \right] - f(x'))^2 + Var (\epsilon)
\]

\[
= Var (g_S(x')) + (E\left[ g_S(x') \right] - f(x'))^2 + \sigma^2
\]

Variance
Bias
Noise
Bias, Variance, and Noise

- **Variance:** $E[ (g_S(x') - E[g_S(x')])^2 ]$
  - Describes how much $g_S(x')$ varies from one training set $S$ to another

- **Bias:** $E[g_S(x')] - f(x')$
  - Describes the average error of $g_S(x')$

- **Noise:** $E \left[ (y' - f(x'))^2 \right] = E[\epsilon^2] = \sigma^2$
  - Describes how much $y'$ varies from $f(x')$
2-D Example

50 fits (20 examples each)
Bias
Variance
Noise
Bias

- Low bias
  - ?

- High bias
  - ?
Bias

- Low bias
  - Linear regression applied to linear data
  - 2nd degree polynomial applied to quadratic data

- High bias
  - Constant function
  - Linear regression applied to highly non-linear data
Variance

- Low variance
  - ?

- High variance
  - ?
Variance

• Low variance
  • Constant function
  • Model independent of training data

• High variance
  • High degree polynomial
Bias/Variance Tradeoff

• $(bias^2 + \text{variance})$ is what counts for prediction

• As we saw in PAC learning, we often have
  • Low bias $\Rightarrow$ high variance
  • Low variance $\Rightarrow$ high bias
  • Is this a firm rule?
Reduce Variance Without Increasing Bias

- Averaging reduces variance: let $Z_1, ..., Z_N$ be i.i.d random variables

$$Var\left(\frac{1}{N} \sum_{i} Z_i\right) = \frac{1}{N} Var(Z_i)$$

- Idea: average models to reduce model variance

- The problem
  - Only one training set
  - Where do multiple models come from?
Bagging: Bootstrap Aggregation

• Take repeated bootstrap samples from training set $D$ (Breiman, 1994)

• **Bootstrap sampling**: Given set $D$ containing $N$ training examples, create $D'$ by drawing $N$ examples at random with replacement from $D$

• **Bagging**:
  
  • Create $k$ bootstrap samples $D_1, \ldots, D_k$
  
  • Train distinct classifier on each $D_i$
  
  • Classify new instance by majority vote / average
Bagging: Bootstrap Aggregation

Step 1: Create Multiple Data Sets

Step 2: Build Multiple Classifiers

Step 3: Combine Classifiers

Original Training data

[image from the slides of David Sontag]
Bagging

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS 1</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>BS 2</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>BS 3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- Build a classifier from each bootstrap sample
- In each bootstrap sample, each data point has probability \((1 - \frac{1}{N})^N\) of not being selected
  - Expected number of distinct data points in each sample is then
    \[
    N \cdot \left(1 - \left(1 - \frac{1}{N}\right)^N\right) \approx N \cdot (1 - \exp(-1)) = .632 \cdot N
    \]
Bagging

<table>
<thead>
<tr>
<th>Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS 1</td>
<td>7</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>BS 2</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>BS 3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

- Build a classifier from each bootstrap sample
- In each bootstrap sample, each data point has probability \( \left(1 - \frac{1}{N}\right)^N \) of not being selected
  - If we have 1 TB of data, each bootstrap sample will be ~ 632GB (this can present computational challenges)
Decision Tree Bagging

[Image from the slides of David Sontag]
Decision Tree Bagging (100 Bagged Trees)

[Image from the slides of David Sontag]
Bagging Experiments

i) The data set is randomly divided into a test set $\mathcal{T}$ and a learning set $\mathcal{L}$. In the real data sets $\mathcal{T}$ is 10% of the data. In the simulated waveform data, 1800 samples are generated. $\mathcal{L}$ consists of 300 of these, and $\mathcal{T}$ the remainder.

ii) A classification tree is constructed from $\mathcal{L}$ using 10-fold cross-validation. Running the test set $\mathcal{T}$ down this tree gives the misclassification rate $e_S(\mathcal{L}, \mathcal{T})$.

iii) A bootstrap sample $\mathcal{L}_B$ is selected from $\mathcal{L}$, and a tree grown using $\mathcal{L}_B$. The original learning set $\mathcal{L}$ is used as test set to select the best pruned subtree (see Section 4.3). This is repeated 50 times giving tree classifiers $\phi_1(x), \ldots, \phi_{50}(x)$.

iv) If $(j_n, x_n) \in \mathcal{T}$, then the estimated class of $x_n$ is that class having the plurality in $\phi_1(x_n), \ldots, \phi_{50}(x_n)$. If there is a tie, the estimated class is the one with the lowest class label. The proportion of times the estimated class differs from the true class is the bagging misclassification rate $e_B(\mathcal{L}, \mathcal{T})$.

v) The random division of the data into $\mathcal{L}$ and $\mathcal{T}$ is repeated 100 times and the reported $\bar{e}_S$, $\bar{e}_B$ are the averages over the 100 iterations. For the waveform data, 1800 new cases are generated at each iteration. Standard errors of $\bar{e}_S$ and $\bar{e}_B$ over the 100 iterations are also computed.
## Bagging Results

<table>
<thead>
<tr>
<th>Data Set</th>
<th>$\bar{e}_S$</th>
<th>$\bar{e}_B$</th>
<th>Decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>waveform</td>
<td>29.1</td>
<td>19.3</td>
<td>34%</td>
</tr>
<tr>
<td>heart</td>
<td>4.9</td>
<td>2.8</td>
<td>43%</td>
</tr>
<tr>
<td>breast cancer</td>
<td>5.9</td>
<td>3.7</td>
<td>37%</td>
</tr>
<tr>
<td>ionosphere</td>
<td>11.2</td>
<td>7.9</td>
<td>29%</td>
</tr>
<tr>
<td>diabetes</td>
<td>25.3</td>
<td>23.9</td>
<td>6%</td>
</tr>
<tr>
<td>glass</td>
<td>30.4</td>
<td>23.6</td>
<td>22%</td>
</tr>
<tr>
<td>soybean</td>
<td>8.6</td>
<td>6.8</td>
<td>21%</td>
</tr>
</tbody>
</table>

Breiman “Bagging Predictors” Berkeley Statistics Department TR#421, 1994
Random Forests

Step 1: Create random vectors

Step 2: Use random vector to build multiple decision trees

Step 3: Combine decision trees
Random Forests

- Ensemble method specifically designed for decision tree classifiers

- Introduce two sources of randomness: “bagging” and “random input vectors”
  - Bagging method: each tree is grown using a bootstrap sample of training data
  - Random vector method: best split at each node is chosen from a random sample of $m$ attributes instead of all attributes
Random Forest Algorithm

- For $b = 1$ to $B$
  - Draw a bootstrap sample of size $N$ from the data
  - Grow a tree $T_b$ using the bootstrap sample as follows
    - Choose $m$ attributes uniformly at random from the data
    - Choose the best attribute among the $m$ to split on
    - Split on the best attribute and recurse (until partitions have fewer than $s_{min}$ number of nodes)

- Prediction for a new data point $x$
  - Regression: $\frac{1}{B} \sum_b T_b(x)$
  - Classification: choose the majority class label among $T_1(x), ..., T_B(x)$
A demo of random forests implemented in JavaScript
When Will Bagging Improve Accuracy?

• Depends on the stability of the base-level classifiers

• A learner is **unstable** if a small change to the training set causes a large change in the output hypothesis
  
  • If small changes in $D$ cause large changes in the output, then there will likely be an improvement in performance with bagging

• Bagging can help unstable procedures, but could hurt the performance of stable procedures
  
  • Decision trees are unstable
  
  • $k$-nearest neighbor is stable