Einführung in Web- und Data-Science

Ensemble Learning

Dr. Marcel Gehrke
Universität zu Lübeck
Institut für Informationssysteme
Recap: Decision Trees

\[ x = \begin{bmatrix} \text{age} \\ 1_{\text{gender=male}} \end{bmatrix} \quad y = \begin{cases} 1 & \text{height > 55"} \\ 0 & \text{height \leq 55"} \end{cases} \]
Ensembles of Classifiers

- None of the classifiers is perfect
- Idea
  - Combine the classifiers to improve performance
- Ensembles of classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting
Example: Weather Forecast

<table>
<thead>
<tr>
<th>Reality</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reality</td>
<td>Stormy</td>
<td>Sunny</td>
<td>Stormy</td>
<td>Sunny</td>
<td>Stormy</td>
<td>Sunny</td>
</tr>
<tr>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>4</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>5</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
Voting

- Linear combination of \( d_j \in \{-1, 1\} \)

\[
y = \sum_{j=1}^{L} w_j d_j
\]

\( w_j \geq 0 \) and \( \sum_{j=1}^{L} w_j = 1 \)

- Unweighted voting: \( w_j = 1/L \)

- Also possible \( d_j \in \mathbb{Z} \)
- High values for \(|y|\) means high "confidence"
- Possibly use \( \text{sign}(y) \in \{-1, 1\} \)
Outline

• Bias/Variance Tradeoff

• Ensemble methods that minimize variance
  – Bagging [Breiman 94]
  – Random Forests [Breiman 97]

• Ensemble methods that minimize bias
  – Boosting [Freund&Schapire 95, Friedman 98]
  – Ensemble Selection

Subsequent slides are based on a presentation by Yisong Yue
An Introduction to Ensemble Methods
Bagging, Boosting, Random Forests, and More
Generalization Error

- **“True” distribution:** \( P(x,y) \)
  - Unknown to us

- **Train:** \( h(x) = y \)
  - Using training data \( S = \{(x_1,y_1), \ldots, (x_n,y_n)\} \)
  - Sampled from \( P(x,y) \)

- **Generalization Error:**
  - \( \mathcal{L}(h) = E_{(x,y) \sim P(x,y)}[ f(h(x),y) ] \)
  - E.g., \( f(a,b) = (a-b)^2 \)
<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jessica</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Amy</td>
<td>12</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Xavier</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cathy</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Eugene</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rafael</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Peter</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Henry</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rose</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Iain</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Paulo</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Margaret</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jill</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Leon</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Sarah</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gena</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Patrick</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Generalization Error:**

$L(h) = \mathbb{E}_{(x,y) \sim P(x,y)}[ f(h(x), y) ]$
Bias/Variance Tradeoff

- Treat $h(x|S)$ as a random function
  - Depends on training data $S$

- $\mathcal{L} = \mathbb{E}_S[\mathbb{E}_{(x,y) \sim P(x,y)}[ f(h(x|S),y) ] ]$
  - Expected generalization error
  - Over the randomness of $S$
Bias/Variance Tradeoff

- Squared loss: \( f(a,b) = (a-b)^2 \)
- Consider one data point \((x,y)\)
- Notation:
  - \( Z = h(x|S) - y \)
  - \( \bar{z} = E_S[Z] \)
  - \( Z-\bar{z} = h(x|S) - E_S[h(x|S)] \)

\[
E_S[(Z-\bar{z})^2] = E_S[Z^2 - 2Z\bar{z} + \bar{z}^2] \\
= E_S[Z^2] - 2E_S[Z]\bar{z} + \bar{z}^2 \\
= E_S[Z^2] - \bar{z}^2
\]

\[
E_S[f(h(x|S),y)] = E_S[Z^2] \\
= E_S[(Z-\bar{z})^2] + \bar{z}^2
\]

Bias = systematic error resulting from the effect that the expected value of estimation results differs from the true underlying quantitative parameter being estimated.
Example
$h(x|S)$
h(x|S)
$h(x|S)$
\[ E[S(h(x|S) - y)^2] = E[S(Z - \bar{z})^2] + \bar{z}^2 \]

Variance

Bias

Expected Error
Outline

- Bias/Variance Tradeoff

- Ensemble methods that minimize variance
  - Bagging
  - Random Forests

- Ensemble methods that minimize bias
  - Boosting
  - Ensemble Selection
**Bagging**

- **Goal:** reduce variance

- **Ideal setting:** many training sets $S'$
  - Train model using each $S'$
  - Average predictions

\[
E_S[(h(x|S) - y)^2] = E_S[(Z - \hat{z})^2] + \hat{z}^2
\]

- Variance reduces linearly
- Bias unchanged

"Bagging Predictors" [Leo Breiman, 1994]

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Bagging

Sampling schemes may be without replacement (‘WOR’—no element can be selected more than once in the same sample) or with replacement (‘WR’—an element may appear multiple times in the one sample). [Wikipedia]

- **Goal**: reduce variance
- **In practice**: resample $S'$ with replacement
  - Train model using each $S'$
  - Average predictions

$$E_S[(h(x|S) - y)^2] = E_S[(Z - \bar{Z})^2] + \bar{Z}^2$$

Expected Error  \uparrow \quad Variance \quad \uparrow \quad Bias

“Bagging Predictors” [Leo Breiman, 1994]

Bagging = Bootstrap Aggregation

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Bagging

Training dataset $D = \{<x_i, y_i>\}_{i=1,\ldots,50}$

Sampling

Bootstrap datasets

$D_1$, $D_2$, $\ldots$, $D_B$

Estimation 1, Estimation 2, $\ldots$, Estimation $B$

Estimation_{Bagging} = \frac{1}{B} \sum_{b=1}^{B} \text{Estimation}_{Bootstrap}$

Majority voting
Figure 5. The bias-variance decomposition for MC4 and three versions of Bagging. In most cases, the reduction in error is due to a reduction in variance (e.g., waveform, letter, satimage, shuttle), but there are also examples of bias reduction when pruning is disabled (as in mushroom and letter).

“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139, 1999
Random Forests

• **Goal**: reduce variance
  – Bagging can only do so much
  – Resampling training data converges asymptotically to minimum reachable error

• **Random Forests**: sample data & features!
  – Sample $S'$
  – Train DT
    - At each node, sample feature subset
  – Average predictions

http://oz.berkeley.edu/~breiman/random-forests.pdf
The Random Forest Algorithm

Given a training set \( S \)

For \( i := 1 \) to \( k \) do:

Build subset \( S_i \) by sampling with replacement from \( S \)

Learn tree \( T_i \) from \( S_i \)

At each node:

Choose best split from random subset of \( F \) features

Each tree grows to the largest extent, and no pruning

Make predictions according to majority vote of the set of \( k \) trees.
Outline

• Bias/Variance Tradeoff

• Ensemble methods that minimize variance
  – Bagging
  – Random Forests

• Ensemble methods that minimize bias
  – Boosting
  – Ensemble Selection

Yoav Freund and Robert Schapire who won the Gödel Prize in 2003

Selection of a Series of Classifiers

Original training set

Data set 1

Data set 2

... ...

Data set T

Classifier 1

Classifier 2

... ...

Classifier T

Training instances that are wrongly predicted by Classifier 1 motivate the selection of the best classifier from a pool able to deal with previously erroneously classified instances.

Next set of training instance is determined by weighted sampling.

Weighted combination

Pool of Classifiers
<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jessica</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Amy</td>
<td>12</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Xavier</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cathy</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Eugene</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rafael</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Peter</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Henry</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rose</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Iain</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Paulo</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Margaret</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jill</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Leon</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Sarah</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gena</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Patrick</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

How to implement weighted sampling?
Example of a Good Classifier: Bias minimal

How can we automatically construct such a classifier?
AdaBoost (Adaptive Boosting)

• Wanted: Two-class classifier for pattern recognition problem
• Given: Pool of 11 classifiers (experts)
• For a given pattern $x_i$ each expert $k_j$ can emit an opinion $k_j(x_i) \in \{-1, 1\}$
• Final decision: $\text{sign}(C(x))$ where $C(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \cdots + \alpha_{11} k_{11}(x_i)$
• $k_1, k_2, \ldots, k_{11}$ denote the eleven experts
• $\alpha_1, \alpha_2, \ldots, \alpha_{11}$ are the weights we assign to the opinion of each expert
• Problem: How to derive $\alpha_j$ (and $k_j$)?

AdaBoost: Constructing the Ensemble

- Derive expert ensemble iteratively
- Let us assume we have already $m-1$ experts
  - $C_{m-1}(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \cdots + \alpha_{m-1} k_{m-1}(x_i)$
- For the next one, classifier $m$, it holds that
  - $C_m(x_i) = C_{m-1}(x_i) + \alpha_m k_m(x_i)$ with $C_{m-1} = 0$ for $m = 1$
- Let us define an error function for the ensemble
  - If $y_i$ and $C_m(x_i)$ coincide, the error for $x_i$ should be small (in particular when $C_m(x_i)$ is large), if not, error should be large
  - $E(x) = \sum_{i=1}^{N} e^{-y_i(C_{m-1}(x_i) + \alpha_m k_m(x_i))}$ where $\alpha_m$ and $k_m$ are to be determined in an optimal way
AdaBoost (cntd.)

- \( E(x) = \sum_{i=1}^{N} w^{(m)}_i \cdot e^{-y_i \alpha_m k_m(x_i)} \)
  
   with \( w^{(m)}_i = e^{-y_i (c_{m-1}(x_i))} \) for \( i \in \{1..N\} \) and \( w^{(1)}_i = 1 \)

- \( E(x) = \sum_{y_i=k_m(x_i)} w^{(m)}_i \cdot e^{-\alpha_m} + \sum_{y_i \neq k_m(x_i)} w^{(m)}_i \cdot e^{\alpha_m} \)

- \( e^{\alpha_m} E(x) = W_c + W_e \cdot e^{2\alpha_m} \quad e^{2\alpha_m} > 1 \)

  constant in each iteration, call it \( W \)

- Pick classifier \( k_m \) with lowest weighted error to minimize right-hand side of equation

- Select \( k_m \)'s weight \( \alpha_m : \) Solve \( \arg\min_{\alpha_m} E(x) \)
AdaBoost (cntd.)

\[ \delta E / \delta \alpha_m = - W_c e^{-\alpha_m} + W_e e^{\alpha_m} \]

- Find minimum

- \(- W_c e^{-\alpha_m} + W_e e^{\alpha_m} = 0\)

- \(- W_c + W_e e^{2\alpha_m} = 0\)

- \(\alpha_m = \frac{1}{2} \ln \left( \frac{W_c}{W_e} \right)\)

- \(\alpha_m = \frac{1}{2} \ln \left( \frac{W - W_e}{W_e} \right)\)

- \(\alpha_m = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_m}{\varepsilon_m} \right)\)

with \(\varepsilon_m = \frac{W_e}{W}\) being the percentage rate of error given the weights of the data points
AdaBoost

For $m = 1$ to $M$

1. Select and extract from the pool of classifiers the classifier $k_m$ which minimizes

$$W_e = \sum_{y_i \neq k_m(x_i)} w_i^{(m)}$$

2. Set the weight $\alpha_m$ of the classifier to

$$\alpha_m = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_m}{\varepsilon_m} \right)$$

where $\varepsilon_m = W_e/W$

3. Update the weights of the data points for the next iteration. If $k_m(x_i)$ is a miss, set

$$w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m} = w_i^{(m)} \sqrt{\frac{1 - \varepsilon_m}{\varepsilon_m}}$$

otherwise

$$w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m} = w_i^{(m)} \sqrt{\frac{\varepsilon_m}{1 - \varepsilon_m}}$$
Round 1 of 3

\[
h_1 \quad \varepsilon_1 = 0.300 \\
\alpha_1 = 0.424
\]

\[D_2\]
Round 2 of 3

\[ \varepsilon_2 = 0.196 \]

\[ h_2 \]

\[ \alpha_2 = 0.704 \]

\[ D_2 \]
Round 3 of 3

$\varepsilon_3 = 0.344$

$\alpha_2 = 0.323$

STOP
Final Hypothesis

\[ H_{\text{final}} = \text{sign}[0.42(h1 ? 1|1-1) + 0.70(h2 ? 1|1-1) + 0.32(h3 ? 1|1-1)] \]
AdaBoost with Decision Trees

\[ h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_n h_n(x) \]

- \( S' = \{(x,y,w_1)\} \)
- \( S' = \{(x,y,w_2)\} \)
- \( S' = \{(x,y,w_n)\} \)

\( w \) – weighting on data points
\( \alpha \) – weight of linear combination

Stop when validation performance plateaus

Boosting often uses weak models
E.g., “shallow” decision trees
Weak models have lower variance

“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139, 1999
Bagging vs Boosting

- **Bagging**: the construction of complementary base-learners is left to chance and to the unstability of the learning methods
- **Boosting**: actively seek to generate complementary base-learners--- training the next base-learner based on the mistakes of the previous learners
Ensemble Selection

Maintain ensemble model as combination of $H$:

$$h(x) = h_1(x) + h_2(x) + \ldots + h_n(x) + h_{n+1}(x)$$

Add model from $H$ that maximizes performance on $V'$

Repeat

$H = \{2000$ models trained using $S'\}$

“Ensemble Selection from Libraries of Models”
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004
<table>
<thead>
<tr>
<th>Method</th>
<th>Minimize Bias?</th>
<th>Minimize Variance?</th>
<th>Other Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging</td>
<td>Complex model class. (Deep DTs)</td>
<td>Bootstrap aggregation (resampling training data)</td>
<td>Does not work for simple models.</td>
</tr>
<tr>
<td>Random Forests</td>
<td>Complex model class. (Deep DTs)</td>
<td>Bootstrap aggregation + bootstrapping features</td>
<td>Only for decision trees.</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>Optimize training performance.</td>
<td>Simple model class. (Shallow DTs)</td>
<td>Determines which model to add at run-time.</td>
</tr>
<tr>
<td>(AdaBoost)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble Selection</td>
<td>Optimize validation performance</td>
<td>Optimize validation performance</td>
<td>Pre-specified dictionary of models learned on training set.</td>
</tr>
</tbody>
</table>

...and many other ensemble methods as well.

- State-of-the-art prediction performance
  - Won Netflix Challenge
  - Won numerous KDD Cups
  - Industry standard

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences. **2009**

Although the data sets were constructed to preserve customer privacy, the Prize has been criticized by privacy advocates. In 2007 two researchers from the University of Texas were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database.
Average performance over many datasets
Random Forests perform the best

“An Empirical Evaluation of Supervised Learning in High Dimensions”
Caruana, Karampatziakis & Yessenalina, ICML 2008
Mixture of Experts: Gating

- Voting where weights are input-dependent (gating)
- Different input regions covered by different learners (Jacobs et al., 1991)

\[ y = \sum_{j=1}^{L} w_j d_j \]

- Gating decides which expert to use
- Need to learn the individual experts as well as the gating functions \( w_i(x) \):

\[ \sum w_j(x) = 1, \text{ for all } x \]
Mixture of Experts: Stacking

- Combiner $f()$ is another learner (Wolpert, 1992)
Mixture of Experts: Cascading

Use $d_j$ only if preceding ones are not confident

Cascade learners in order of complexity