Previously...

• We’ve learned how to build and apply single models
  – Nearest neighbor
  – Logistic regression
  – Linear regression
  – Trees
Ensemble Models

- An ensemble averages or sums predictions from multiple models

- Remember “Who Wants to be a Millionaire”?  
  - “Poll the audience” vs “Call a friend”

- Averaging multiple “weak” predictions is often more accurate than any single predictor  
  - e.g. audience success rate is 92% vs 66% for the friend

- Models can be constructed independently by sampling, or by incrementally training model to fix previous model’s mistakes  
  - Averaging independent predictions reduces variance  
  - Incrementally fixing mistakes reduces bias
Bias-Variance Trade-off

\[
E_{x,y,D} \left[ (h_D(x) - y)^2 \right] = E_{x,D} \left[ (h_D(x) - \bar{h}(x))^2 \right] + E_{x,y} \left[ (\bar{y}(x) - y)^2 \right] + E_x \left[ (\bar{h}(x) - \bar{y}(x))^2 \right]
\]

**Variance**: due to limited data
Different training samples will give different models that vary in predictions for the same test sample

“**Noise**”: irreducible error due to data/problem

**Bias**: expected error when optimal model is learned from infinite data

Above is for regression.
But same “expected error = variance + noise + bias^2“ holds for classification error and logistic regression.

See [this](#) for derivation
Bias-Variance Trade-off

\[
E_{x,y,D} \left[ (h_D(x) - y)^2 \right] = E_{x,D} \left[ (h_D(x) - \bar{h}(x))^2 \right] + E_{x,y} \left[ (\bar{y}(x) - y)^2 \right] + E_x \left[ (\bar{h}(x) - \bar{y}(x))^2 \right]
\]

- **Expected Test Error**
- **Variance**
- **Noise**
- **Bias^2**

![Bias-Variance Diagram](image)

See [this](#) for derivation

Fig Sources
Let’s see how ensembles battle bias and variance

- Bootstrapping
- Bagging
- Boosting (Schapire 1989)
- Adaboost (Schapire 1995)
Bootstrap Estimation

• Repeatedly draw $n$ samples from $D$

• For each set of samples, estimate a statistic

• The bootstrap estimate is the mean of the individual estimates

• Used to estimate a statistic (parameter) and its variance
Bagging - Aggregate Bootstrapping

• For $i = 1 .. M$
  – Draw $n^* < n$ samples from $D$ with replacement
  – Learn classifier $C_i$

• Final classifier is a vote of $C_1 .. C_M$

• Increases classifier stability / reduces variance
Random Forests

**Train** a collection of trees (e.g. 100 trees).

For each:

1. Randomly sample some fraction of data (e.g. 90%)
2. Randomly sample some number of features
   - For regression: suggest $(\# \text{ features}) / 3$
   - For classification: suggest $\sqrt{\# \text{ features}}$ or $\log_2(\# \text{ features})$
3. Train a tree
4. (Optional: can get validation error on held out data)

**Predict:** Average the predictions of all trees

Breiman 2001 [pdf]
Adaboost Terms

• Learner = Hypothesis = Classifier

• Weak Learner: classifier that can achieve < 50% training error over any training distribution

• Strong Learner: makes prediction by combining weak learner predictions
Boosting (Schapire 1989)

- Randomly select $n_1 < n$ samples from $D$ without replacement to obtain $D_1$
  - Train weak learner $C_1$

- Select $n_2 < n$ samples from $D$ with half of the samples misclassified by $C_1$ to obtain $D_2$
  - Train weak learner $C_2$

- Select all samples from $D$ that $C_1$ and $C_2$ disagree on
  - Train weak learner $C_3$

- Final strong learner is vote of weak learners

---

Boosting Terminology

- **Learner** = **Hypothesis** = **Classifier**
- **Weak Learner**: classifier that can achieve $< 50\%$ training error over any training distribution
- **Strong Learner**: makes prediction by combining weak learner predictions
Adaboost - Adaptive Boosting

• Instead of sampling, re-weight
  – Previous weak learner has only 50% accuracy over new distribution

• Learn “weak classifiers” on the re-weighted samples

• Final classification based on weighted vote of weak classifiers

https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf (Freund Schapire ‘99)
What does it mean to “weight” your training samples?

- Some examples count more than others toward parameter estimation or learning objective
- E.g., suppose you want to estimate $P(x=0 \mid y=0)$ for Naïve Bayes

Unweighted

$$
\theta_{\{x = 0\mid y = 0\}} = \frac{\sum_{x_n, y_n \in D} \delta(x_n = 0 \text{ and } y_n = 0)}{\sum_{x, y \in D} \delta(y_n = 0)}
$$

Weighted

$$
\theta_{w,\{x = 0\mid y = 0\}} = \frac{\sum_{x_n, y_n \in D} w_n \delta(x_n = 0 \text{ and } y_n = 0)}{\sum_{x_n, y_n \in D} w_n \delta(y_n = 0)}
$$
What does it mean to “weight” your training samples?

Estimate $P(x=0 \mid y=0)$:

<table>
<thead>
<tr>
<th>$w$</th>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
What does it mean to “weight” your training samples?

Estimate $P(x=0 \mid y=0)$:

<table>
<thead>
<tr>
<th>w</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Unweighted: $P(x = 0 \mid y = 0) = \frac{1 + 1 + 1}{1 + 1 + 1 + 1 + 1} = \frac{3}{5}$

Weighted: $P(x = 0 \mid y = 0) = \frac{0.1 + 0.1 + 0.1}{0.1 + 0.1 + 0.2 + 0.1 + 0.2} = \frac{3}{7}$
Adaboost with Confidence Weighted Predictions (RealAB)

---

**Real AdaBoost**

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   
   (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights $w_i$ on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$. $y_i \in \{-1, 1\}$
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$.

---

Boosted decision trees

**Train**

1. Initialize sample weights to uniform
2. For each tree (e.g. 10-100), based on weighted samples:
   a. Train small tree (e.g. depth = 2-4 typically)
   b. Estimate logit prediction at each leaf node
   c. Reweight samples

**Predict**: sum logit predictions from all trees
ML Method Comparison by Caruana (2006)

Table 3. Normalized scores of each learning algorithm by problem (averaged over eight metrics)

<table>
<thead>
<tr>
<th>MODEL</th>
<th>CAL</th>
<th>COVT</th>
<th>ADULT</th>
<th>LTR.P1</th>
<th>LTR.P2</th>
<th>MEDIS</th>
<th>SLAC</th>
<th>HS</th>
<th>MG</th>
<th>CALHIOUS</th>
<th>COD</th>
<th>BACT</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST-DT</td>
<td>PLT</td>
<td>.938</td>
<td>.857</td>
<td>.959</td>
<td>.976</td>
<td>.700</td>
<td>.869</td>
<td>.933</td>
<td>.855</td>
<td>.974</td>
<td>.915</td>
<td>.878*</td>
<td>.896*</td>
</tr>
<tr>
<td>RF</td>
<td>PLT</td>
<td>.876</td>
<td>.940</td>
<td>.901</td>
<td>.969</td>
<td>.692*</td>
<td>.878</td>
<td>.927</td>
<td>.845</td>
<td>.965</td>
<td>.912</td>
<td>.861</td>
<td>.887*</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>ISO</td>
<td>.922*</td>
<td>.865</td>
<td>.901*</td>
<td>.969</td>
<td>.692*</td>
<td>.878</td>
<td>.927</td>
<td>.845</td>
<td>.965</td>
<td>.912</td>
<td>.861</td>
<td>.887*</td>
</tr>
<tr>
<td>RF</td>
<td>ISO</td>
<td>.865</td>
<td>.934</td>
<td>.851</td>
<td>.965</td>
<td>.776*</td>
<td>.920</td>
<td>.877</td>
<td>.933</td>
<td>.933</td>
<td>.897</td>
<td>.821</td>
<td>.880</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>ISO</td>
<td>.867</td>
<td>.933</td>
<td>.840</td>
<td>.915</td>
<td>.749</td>
<td>.897</td>
<td>.956</td>
<td>.940</td>
<td>.859</td>
<td>.907*</td>
<td>.877</td>
<td>.877</td>
</tr>
<tr>
<td>SVM</td>
<td>PLT</td>
<td>.765</td>
<td>.886</td>
<td>.936</td>
<td>.962</td>
<td>.733</td>
<td>.866</td>
<td>.913*</td>
<td>.816</td>
<td>.897</td>
<td>.900*</td>
<td>.807</td>
<td>.862</td>
</tr>
<tr>
<td>ANN</td>
<td>RTL</td>
<td>.766</td>
<td>.884</td>
<td>.913</td>
<td>.901</td>
<td>.791*</td>
<td>.881</td>
<td>.932</td>
<td>.850</td>
<td>.923</td>
<td>.667</td>
<td>.882</td>
<td>.854</td>
</tr>
<tr>
<td>SVM</td>
<td>ISL</td>
<td>.758</td>
<td>.882</td>
<td>.899</td>
<td>.954</td>
<td>.693*</td>
<td>.878</td>
<td>.907</td>
<td>.827</td>
<td>.900</td>
<td>.775</td>
<td>.852</td>
<td>.852</td>
</tr>
<tr>
<td>ANN</td>
<td>ISO</td>
<td>.766</td>
<td>.872</td>
<td>.898</td>
<td>.894</td>
<td>.775</td>
<td>.871</td>
<td>.929*</td>
<td>.846</td>
<td>.919</td>
<td>.665</td>
<td>.871</td>
<td>.846</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>ISL</td>
<td>.767</td>
<td>.882</td>
<td>.821</td>
<td>.891</td>
<td>.755*</td>
<td>.895</td>
<td>.926*</td>
<td>.841</td>
<td>.915</td>
<td>.672</td>
<td>.862</td>
<td>.842</td>
</tr>
<tr>
<td>SVM</td>
<td>PLT</td>
<td>.874</td>
<td>.842</td>
<td>.875</td>
<td>.913</td>
<td>.523</td>
<td>.807</td>
<td>.860</td>
<td>.785</td>
<td>.933</td>
<td>.835</td>
<td>.858</td>
<td>.828</td>
</tr>
<tr>
<td>ANN</td>
<td>ISO</td>
<td>.819</td>
<td>.785</td>
<td>.920</td>
<td>.937</td>
<td>.626</td>
<td>.777</td>
<td>.803</td>
<td>.844</td>
<td>.827</td>
<td>.744</td>
<td>.855</td>
<td>.815</td>
</tr>
<tr>
<td>KNN</td>
<td>PLT</td>
<td>.807</td>
<td>.780</td>
<td>.912</td>
<td>.936</td>
<td>.598</td>
<td>.800</td>
<td>.801</td>
<td>.853</td>
<td>.827</td>
<td>.748</td>
<td>.852</td>
<td>.810</td>
</tr>
<tr>
<td>KNN</td>
<td>ISO</td>
<td>.814</td>
<td>.784</td>
<td>.879</td>
<td>.935</td>
<td>.633</td>
<td>.791</td>
<td>.794</td>
<td>.832</td>
<td>.824</td>
<td>.777</td>
<td>.833</td>
<td>.809</td>
</tr>
<tr>
<td>BST-DTM</td>
<td>PLT</td>
<td>.644</td>
<td>.940</td>
<td>.767</td>
<td>.688</td>
<td>.723</td>
<td>.806</td>
<td>.800</td>
<td>.862</td>
<td>.923</td>
<td>.622</td>
<td>.915*</td>
<td>.791</td>
</tr>
<tr>
<td>SVM</td>
<td>RTL</td>
<td>.696</td>
<td>.819</td>
<td>.731</td>
<td>.860</td>
<td>.600</td>
<td>.859</td>
<td>.788</td>
<td>.776</td>
<td>.853</td>
<td>.864</td>
<td>.763</td>
<td>.781</td>
</tr>
<tr>
<td>BST-DTM</td>
<td>ISL</td>
<td>.689</td>
<td>.941</td>
<td>.700</td>
<td>.681</td>
<td>.711</td>
<td>.807</td>
<td>.793</td>
<td>.862</td>
<td>.912</td>
<td>.632</td>
<td>.902*</td>
<td>.780</td>
</tr>
<tr>
<td>SVM</td>
<td>PLT</td>
<td>.605</td>
<td>.865</td>
<td>.540</td>
<td>.615</td>
<td>.624</td>
<td>.779</td>
<td>.683</td>
<td>.799</td>
<td>.817</td>
<td>.581</td>
<td>.906*</td>
<td>.710</td>
</tr>
<tr>
<td>KNN</td>
<td>RTL</td>
<td>.652</td>
<td>.872</td>
<td>.723</td>
<td>.760</td>
<td>.449</td>
<td>.706</td>
<td>.600</td>
<td>.820</td>
<td>.851</td>
<td>.829</td>
<td>.809*</td>
<td>.708</td>
</tr>
<tr>
<td>LR</td>
<td>PLT</td>
<td>.661</td>
<td>.863</td>
<td>.734</td>
<td>.756</td>
<td>.416</td>
<td>.779</td>
<td>.607</td>
<td>.822</td>
<td>.826</td>
<td>.497</td>
<td>.890*</td>
<td>.706</td>
</tr>
<tr>
<td>LR</td>
<td>ISO</td>
<td>.625</td>
<td>.886</td>
<td>.195</td>
<td>.448</td>
<td>.777*</td>
<td>.852</td>
<td>.675</td>
<td>.849</td>
<td>.848</td>
<td>.647</td>
<td>.905*</td>
<td>.700</td>
</tr>
<tr>
<td>LR</td>
<td>PLT</td>
<td>.610</td>
<td>.870</td>
<td>.185</td>
<td>.446</td>
<td>.738</td>
<td>.835</td>
<td>.667</td>
<td>.823</td>
<td>.892</td>
<td>.633</td>
<td>.895</td>
<td>.685</td>
</tr>
<tr>
<td>NB</td>
<td>PLT</td>
<td>.574</td>
<td>.904</td>
<td>.674</td>
<td>.557</td>
<td>.709</td>
<td>.724</td>
<td>.205</td>
<td>.658</td>
<td>.758</td>
<td>.633</td>
<td>.770</td>
<td>.654</td>
</tr>
<tr>
<td>NB</td>
<td>RTL</td>
<td>.572</td>
<td>.892</td>
<td>.648</td>
<td>.561</td>
<td>.694</td>
<td>.732</td>
<td>.213</td>
<td>.690</td>
<td>.755</td>
<td>.632</td>
<td>.756</td>
<td>.650</td>
</tr>
<tr>
<td>NB</td>
<td>RTL</td>
<td>.552</td>
<td>.843</td>
<td>.534</td>
<td>.566</td>
<td>.011</td>
<td>.714</td>
<td>.654</td>
<td>.655</td>
<td>.759</td>
<td>.636</td>
<td>.688</td>
<td>.650</td>
</tr>
</tbody>
</table>

**BST-DT**: Boosted Decision Tree  
**RF**: Random Forest  
**ANN**: Neural net  
**KNN**: SVM  
**NB**: Naïve Bayes  
**LR**: Logistic Regression

**Bold**: best  
*: not significantly worse than best

Calibration methods:  
**PLT**: Platt Calibration  
**ISO**: Isotonic Regression  
- None used
Caruana et al. 2008: comparison on high dimensional data

- Boosted Decision Trees FTW again!
- RF second again!
- But note that Adaboost underperforms in the very high dimensional datasets, where RF excels
Boosted Trees and Random Forests work for different reasons

• Boosted trees
  – Use small trees (high bias, low variance) to iteratively refine the prediction
  – Combining prediction from many trees reduces bias
  – Overfitting is a danger (i.e. too many / too large trees eliminates train error but increases test error)

• Random forest
  – Use large trees (low bias, high variance)
  – Average of many tree predictions reduces variance
  – Hard to break – just train a whole bunch of trees
Other ensembles

• Can average predictions of any classifiers / regressors
  – But they should not be duplicates, so e.g. averaging multiple linear regressors trained on all features/data has no point
  – Averaging multiple deep networks (even when trained on all data) reduces error and improves confidence estimates

• Cascades: early classifiers make decisions on easy examples; later ones deal only with hard examples

Wang et al. ICML 2022 [pdf]
Answer these questions

- Think about: Suppose you had an infinite sized audience to poll for a multiple choice question.
  - $y=\{A, B, C, D\}$, where A is correct answer
  - A randomly sampled audience member will report an answer with probability $P(y)$
- What condition guarantees a correct answer?
- If your friend is a random member of the audience, what is the probability that his or her answer is correct?
- After that we’ll do a detailed example with pose estimation
Example in detail: Depth from Kinect with RFs

- IR Projector
- IR Sensor
- Projected Light Pattern
- Stereo Algorithm
- Depth Image
- Segmentation, Part Prediction
- Body Pose
Goal: estimate pose from depth image

Real-Time Human Pose Recognition in Parts from a Single Depth Image
Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake
CVPR 2011
Goal: estimate pose from depth image

Challenges

• Lots of variation in bodies, orientation, poses
• Needs to be very fast (their algorithm runs at 200 FPS on the Xbox 360 GPU)
Extract body pixels by thresholding depth
Basic learning approach

• Very simple features

• Lots of data

• Flexible classifier
Features

• Difference of depth at two offsets
  – Offset is scaled by depth at center
Get lots of training data

• Capture and sample 500K mocap frames of people kicking, driving, dancing, etc.
• Get 3D models for 15 bodies with a variety of weight, height, etc.
• Synthesize mocap data for all 15 body types
Body models
Part prediction with random forests

- Randomized decision forests: collection of independently trained trees
- Each tree is a classifier that predicts the likelihood of a pixel belonging to each part
  - Node corresponds to a thresholded feature
  - The leaf node that an example falls into corresponds to a conjunction of several features
  - In training, at each node, a subset of features is chosen randomly, and the most discriminative is selected
Joint estimation

- Joints are estimated using mean-shift (a fast mode-finding algorithm)

- Observed part center is offset by pre-estimated value
Results

Ground Truth
More results

<table>
<thead>
<tr>
<th>Input depth image</th>
<th>Inferred body parts</th>
<th>Inferred joint proposals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>front side top</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Accuracy vs. Number of Training Examples

- Synthetic test set
- Real test set
- Silhouette (scale)
- Silhouette (no scale)
HW 4 (due April 1)

https://docs.google.com/document/d/1_9ZUFL7gi7Mq0-isQOcwDxhhmlDKVgdZg9mDaKokHEA/edit
Things to Remember

• Ensembles improve accuracy and confidence estimates by reducing bias and/or variance

• Boosted trees minimize bias by fixing previous mistakes

• Random forests minimize variance by averaging over multiple different trees

• Random forests and boosted trees are powerful classifiers and useful for a wide variety of problems
Thursday

• Stochastic Gradient Descent