Model Combination

Or, Ensemble Methods

Based on lecture by Parikshit Ram
# Numerous Possible Classifiers!

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Training time</th>
<th>Cross validation</th>
<th>Testing time</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN classifier</td>
<td>None</td>
<td>Can be slow</td>
<td>Slow</td>
<td>??</td>
</tr>
<tr>
<td>Decision trees</td>
<td>Slow</td>
<td>Very slow</td>
<td>Very fast</td>
<td>??</td>
</tr>
<tr>
<td>Naive Bayes classifier</td>
<td>Fast</td>
<td>None</td>
<td>Fast</td>
<td>??</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Which Classifier/Model to Choose?

Possible strategies:

• Go from simplest model to more complex model until you obtain desired accuracy

• Discover a new model if the existing ones do not work for you

• Combine all (simple) models
Common Strategy: Bagging

Consider the data set \( S = \{(x_i, y_i)\}_{i=1,\ldots,n} \)

- Pick a sample \( S^* \) with replacement of size \( n \) from \( S \)
- Train on this set \( S^* \) to get a classifier \( f^* \)
- Repeat above steps \( B \) times to get \( f_1, f_2, \ldots, f_B \)
- Final classifier \( f(x) = \text{majority}\{f_b(x)\}_{j=1,\ldots,B} \)
Common Strategy: Bagging

Why would bagging work?
• Combining multiple classifiers reduces the variance of the final classifier

When would this be useful?
• We have a classifier with low bias and high variance (any examples)
Bagging decision trees

Consider the data set \( S \)

- Pick a sample \( S^* \) with replacement of size \( n \) from \( S \)
- Grow a decision tree \( T_b \) greedily
- Repeat \( B \) times to get \( T_1, \ldots, T_B \)
- The final classifier will be

\[
f(x) = \text{majority}\{f_{T_b}(x)\}_{b=1,\ldots,B}
\]
Random Forests

Almost identical to bagging decision trees, except we introduce some randomness:

- Randomly pick any \( m \) of the \( d \) attributes available
- Grow the tree only using those \( m \) attributes available

That is, Bagged random decision trees

\[ = \text{ Random forests} \]
Points about random forests

Algorithm parameters

• Usual values for $m$: $\sqrt{d}, 1, 10$
• Usual value for $B$: keep increasing $B$ until the training error stabilizes
Bagging/Random forests

Consider the data set $S = \{(x_i, y_i)\}_{i=1,..,n}$

- Pick a sample $S^*$ with replacement of size $n$ from $S$
- Do the training on this set $S^*$ to get a classifier (e.g. random decision tree) $f^*$
- Repeat the above step $B$ times to get $f_1, f_2, \ldots, f_B$
- Final classifier $f(x) = \text{majority}\{f_b(x)\}_{j=1,\ldots,B}$
Final words

Advantages

- Efficient and simple training
- Allows you to work with simple classifiers
- Random-forests generally useful and accurate in practice (one of the best classifiers)
- Embarrassingly parallelizable

Caveats:

- Needs low-bias classifiers
- Can make a not-good-enough classifier worse
Final words

Reading material

• Bagging: ESL Chapter 8.7
• Random forests: ESL Chapter 15

Strategy 2: Boosting

Consider the data set $S = \{(x_i, y_i)\}_{i=1,\ldots,n}$

• Assign a weight $w_{(i,0)} = (1/n)$ to each $i$

• Repeat for $t = 1,\ldots,T$:
  o Train a classifier $f_t$ on $S$ that minimizes the weighted loss: $\sum_{i=1}^{n} w_{(i,t)} L(y_i, f_t(x_i))$
  o Obtain a weight $a_t$ for the classifier $f_t$
  o Update the weight for every point $i$ to $w_{(i, t+1)}$
    • Increase the weights for $i$:
    • Decrease the weights for $i$: $y_i \neq f_t(x_i)$

• Final:

$$f(x) = \text{sign} \left( \sum_{t=1}^{T} a_t f_t(x) \right)$$
Final words on boosting

Advantages
• Extremely useful in practice and has great theory as well
  o Better accuracy than random forests usually
• Can work with very simple classifiers

Caveats:
• Training is inherently sequential
  o Hard to parallelize

Reading material:
• ESL book, Chapter 10
• Le Song's slides: http://www.cc.gatech.edu/~lsong/teaching/CSE6704/lecture9.pdf
Visualizing Classification

Usual tools

• ROC curve / cost curves
  o True-positive rate vs. false-positive rate

• Confusion matrix

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>Cat</td>
<td>5</td>
</tr>
<tr>
<td>Dog</td>
<td>2</td>
</tr>
<tr>
<td>Rabbit</td>
<td>0</td>
</tr>
</tbody>
</table>

![ROC curves](image)
Visualizing Classification
Newer tool
• Visualize the data and class boundary with 2D projection (dimensionality reduction)
Weights in combined models

Bagging / Random forests
• Majority voting

Let people play with the weights?
EnsembleMatrix

Figure 1. Primary view in EnsembleMatrix. Confusion matrices of component classifiers are shown in thumbnails on the right. The matrix on the left shows the confusion matrix of the current ensemble classifier built by the user.

Understanding performance

Figure 2. Representations of confusion matrix for a handwritten digit classification task. (top) standard confusion matrix; (bottom) heat-map confusion matrix. It is much easier to identify underlying patterns in the visual representation; 3 and 8 are often misclassified as each other and 5 is misclassified as many different numbers.

Improving performance

Improving performance

- Adjust the weights of the individual classifiers
- Data partition to separate problem areas
  - Adjust weights just for these individual parts
- State-of-the-art performance, on one dataset

ReGroup - Naive Bayes at work

ReGroup

<table>
<thead>
<tr>
<th>Features to represent each friend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender, Age group</td>
</tr>
<tr>
<td>Family</td>
</tr>
<tr>
<td>Home city/state/country</td>
</tr>
<tr>
<td>Current city/state/country</td>
</tr>
<tr>
<td>High school/college/grad school</td>
</tr>
<tr>
<td>Workplace</td>
</tr>
<tr>
<td>Amount of correspondence</td>
</tr>
<tr>
<td>Recency of correspondence</td>
</tr>
<tr>
<td>Friendship duration</td>
</tr>
<tr>
<td># of mutual friends</td>
</tr>
<tr>
<td>Amount seen together</td>
</tr>
</tbody>
</table>

Y - In group?
X - Features of a friend

\[ P(Y = true|X) = ? \]

Compute \( P(X_d|Y = true) \) for each feature \( d \) using the current group members (how?)

ReGroup

Y - In group?
X - Features of a friend
\[ P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \]
\[ P(X|Y) = P(X_1|Y) \times ... \times P(X_d|Y) \]
Compute \( P(X_i|Y = \text{true}) \) for every feature \( d \) using the current group members

- Use simple counting

Not exactly classification!

- Reorder remaining friends with respect to \( P(X|Y=\text{true}) \)
- "Train" every time a new member is added to the group
Some additional reading

• Interactive machine learning
  o [link](http://research.microsoft.com/en-us/um/redmond/groups/cue/iml/)
  o [link](http://research.microsoft.com/en-us/um/people/samershi/pubs.html)
  o [link](http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/AAAI2012-PnP.pdf)
  o [link](http://research.microsoft.com/en-us/um/redmond/groups/cue/publications/AAAI2012-L2L.pdf)