Fall 2008

Time: T-Th 5:00pm - 6:20pm
Location: GFS 118

Instructor: Sofus A. Macskassy (macskass@usc.edu)
Office: SAL 216
Office hours: by appointment

Teaching assistant: Cheol Han (cheolhan@usc.edu)
Office: SAL 229
Office hours: M 2-3pm, W 11-12

Class web page:
http://www-scf.usc.edu/~csci567/index.html
Lecture 6 Outline

• Off-the-shelf Classifiers
• Decision Trees, Part 1
Hypothesis Spaces, LTUs, etc.

This is representative and not comprehensive
Comparing Perceptron, Logistic Regression, and LDA

• How should we choose among these three algorithms?
• There is a big debate within the machine learning community!
Issues in the Debate

• **Statistical Efficiency.** If the generative model $P(x,y)$ is correct, then LDA usually gives the highest accuracy, particularly when the amount of training data is small. If the model is correct, LDA requires 30% less data than Logistic Regression in theory.

• **Computational Efficiency.** Generative models typically are the easiest to learn. In our example, LDA can be computed directly from the data without using gradient descent.
Issues in the Debate

- Robustness to changing loss functions. Both generative and conditional probability models allow the loss function to be changed at run time without re-learning. Perceptron requires re-training the classifier when the loss function changes.

- Robustness to model assumptions. The generative model usually performs poorly when the assumptions are violated. For example, if $P(\mathbf{x} \mid y)$ is very non-Gaussian, then LDA won’t work well. Logistic Regression is more robust to model assumptions, and Perceptron is even more robust.

- Robustness to missing values and noise. In many applications, some of the features $x_{ij}$ may be missing or corrupted in some of the training examples. Generative models typically provide better ways of handling this than non-generative models.
Off-The-Shelf Classifiers

• A method that can be applied directly to data without requiring a great deal of time-consuming data preprocessing or careful tuning of the learning procedure

• Let’s compare Perceptron, Logistic Regression, and LDA to ask which algorithms can serve as good off-the-shelf classifiers
Off-The-Shelf Criteria

- Natural handling of “mixed” data types
  - continuous, ordered-discrete, unordered-discrete
- Handling of missing values
- Robustness to outliers in input space
- Insensitive to monotone transformations of input features
- Computational scalability for large data sets
- Ability to deal with irrelevant inputs
- Ability to extract linear combinations of features
- Interpretability
- Predictive power
Handling Mixed Data Types with Numerical Classifiers

- **Indicator Variables**
  - sex: Convert to 0/1 variable
  - county-of-residence: Introduce a 0/1 variable for each county

- **Ordered-discrete variables**
  - example: {small, medium, large}
  - Treat as unordered
  - Treat as real-valued

Sometimes it is possible to measure the “distance” between discrete terms. For example, how often is one value mistaken for another? These distances can then be combined via multi-dimensional scaling to assign real values.
Missing Values

- Two basic causes of missing values
  - Missing at random: independent errors cause features to be missing. Examples:
    - clouds prevent satellite from seeing the ground.
    - data transmission (wireless network) is lost from time-to-time
  - Missing for cause:
    - Results of a medical test are missing because physician decided not to perform it.
    - Very large or very small values fail to be recorded
    - Human subjects refuse to answer personal questions
Dealing with Missing Values

• Missing at Random
  – P(x, y) methods can still learn a model of P(x), even when some features are not measured.
  – The EM algorithm can be applied to fill in the missing features with the most likely values for those features.
  – A simpler approach is to replace each missing value by its average value or its most likely value.
  – There are specialized methods for decision trees.

• Missing for cause
  – The “first principles” approach is to model the causes of the missing data as additional hidden variables and then try to fit the combined model to the available data.
  – Another approach is to treat “missing” as a separate value for the feature.
    • For discrete features, this is easy.
    • For continuous features, we typically introduce an indicator feature that is 1 if the associated real-valued feature was observed and 0 if not.
Robust to Outliers in the Input Space

- Perceptron: Outliers can cause the algorithm to loop forever
- Logistic Regression: Outliers far from the decision boundary have little impact – robust!
- LDA/QDA: Outliers have a strong impact on the models of $P(\mathbf{x}|y)$ – not robust!
Remaining Criteria

- **Monotone Scaling**: All linear classifiers are sensitive to non-linear transformations of the inputs, because this may make the data less linearly separable.

- **Computational Scaling**: All three methods scale well to large data sets.

- **Irrelevant Inputs**: In theory, all three methods will assign small weights to irrelevant inputs. In practice, LDA can crash because the $\Sigma$ matrix becomes singular and cannot be inverted. This can be solved through a technique known as regularization (later!)

- **Extract linear combinations of features**: All three algorithms learn LTUs, which are linear combinations!

- **Interpretability**: All three models are fairly easy to interpret.

- **Predictive power**: For small data sets, LDA and QDA often perform best. All three methods give good results.
Sensitivity to monotone transformations

linearly separable

After $y = y^4$ transform not linearly separable
## Summary So Far
*(we will add to this later)*

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Perc</th>
<th>Logistic</th>
<th>LDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed data</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Missing values</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Outliers</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Monotone transformations</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Scalability</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Irrelevant inputs</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Linear combinations</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Interpretable</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Accurate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
The Top Five Algorithms

- Decision trees (C4.5)
- Nearest Neighbor Method
- Neural networks (backpropagation)
- Probabilistic networks (Naïve Bayes; Mixture models)
- Support Vector Machines (SVMs)
Learning Decision Trees

- Decision trees provide a very popular and efficient hypothesis space
  - Variable size: any boolean function can be represented
  - Deterministic
  - Discrete and Continuous Parameters

- Learning algorithms for decision trees can be described as
  - Constructive Search: The tree is built by adding nodes
  - Eager
  - Batch (although online algorithms do exist)
Decision Tree Hypothesis Space

- Internal nodes: test the value of particular features $x_j$ and branch according to the results of the test.
- Leaf nodes: specify the class $f(x)$.

Features: Outlook ($x_1$), Temperature ($x_2$), Humidity ($x_3$), and Wind ($x_4$).

$x = (\text{sunny}, \text{hot}, \text{high}, \text{strong})$ will be classified as No.
• If the features are continuous, internal nodes may test the value of a feature against a threshold.
Decision Trees divide the feature space into axis-parallel rectangles and label each rectangle with one of the K classes.
Decision Trees Can Represent Any Boolean Function

- In the worst case, exponentially many nodes will be needed, however
Decision Trees Provide Variable-Sized Hypothesis Space

- As the number of nodes (or depth) of tree increases, the hypothesis space grows
  - Depth 1 ("decision stump") can represent any boolean function of one feature
  - Depth 2: Any boolean function of two features and some boolean functions involving three features:
    - \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_2)\)
Objective Function

- Let \( h \) be a decision tree
- Define our objective function to be the number of misclassification errors on the training data:
  \[
  J(h) = | \{ (x,y) \in S : h(x) \neq y \} |
  \]
  (what is the name of this loss function?)
- Find \( h \) that minimizes \( J(h) \)
  - Solution: Just create a decision tree with one path from root to leaf for each training example
  - Bug: Such a tree would just memorize the training data. It would not generalize to new data points
  - Solution 2: Find the smallest tree \( h \) that minimizes \( J(h) \).
  - Bug 2: This is NP-Hard
  - Solution 3: Use a greedy approximation
Learning Algorithm for Decision Trees

\[ \text{GrowTree}(S) \]

for \( c \in \{0, 1\} \)

\[ \text{if } (y = c \text{ for all } \langle x, y \rangle \in S) \text{ return new leaf}(c); \]

choose best attribute \( x_j \);

for \( c \in \{0, 1\} \)

\[ S_c := \text{ all } \langle x, y \rangle \in S \text{ with } x_j = c; \]

\[ \text{if } (S_c = \emptyset) \text{ return new leaf}(\text{majority}(S)); \]

return new node(\( x_j \), GrowTree(\( S_0 \)), GrowTree(\( S_1 \)));
Choosing the Best Attribute (Method 1)

- Perform 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data

ChooseBestAttribute(S)

choose $j$ to minimize $J_j$, computed as follows:

for $c \in \{0, 1\}$

$S_c := \text{all } \langle x, y \rangle \in S \text{ with } x_j = c;$

$y_c := \text{the most common value of } y \text{ in } S_c;$

$J_c := \text{number of examples } \langle x, y \rangle \in S_c \text{ with } y \neq y_c;$

$J_j := J_0 + J_1; \text{ (total errors if we split on this feature)}$

return $j;$
Choosing the Best Attribute
An Example

Training Examples

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
x_1 \quad x_2 \quad x_3 \quad y
\]

\[
0 \quad 0 \quad 0 \quad 1
0 \quad 0 \quad 1 \quad 0
0 \quad 1 \quad 0 \quad 1
0 \quad 1 \quad 1 \quad 1
1 \quad 0 \quad 0 \quad 0
1 \quad 0 \quad 1 \quad 1
1 \quad 1 \quad 0 \quad 0
1 \quad 1 \quad 1 \quad 0
\]
Choosing the Best Attribute (3)

- Unfortunately, this measure does not always work well, because it does not detect cases where we are making “progress” toward a good tree.
A Better Heuristic from Information Theory

- Let $V$ be a random variable with the following probability distribution:

<table>
<thead>
<tr>
<th>$P(V = 0)$</th>
<th>$P(V = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

- The surprise $S(V=v)$ of each value of $V$ is defined to be:

$$S(V=v) = - \log_2 P(V = v)$$

- An event with probability 1 has zero surprise
- An event with probability 0 has infinite surprise
- The surprise is equal to the asymptotic number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results. Hence, this is also called the description length of $V$. 
Entropy

- The entropy if $V$, denoted $H(V)$, is defined as

\[ H(V) = \sum_{v=0}^{1} -P(V = v) \log_2 P(V = v) \]

- This is the average surprise describing the result of one trial of $V$ (one coin toss). It can be viewed as a measure of uncertainty.
Entrophy and Information Gain

- Purpose of decision-tree split is to have each subset be “cleaner” than the original data set.
Entropy and Information Gain

- Purpose of decision-tree split is to have each subset be “cleaner” than the original data set.
  - Entropy measures ‘cleanliness’ of a set of points with respect to a particular attribute (e.g. target label)
  - Take ‘outlook’ attribute. It has 3 possible values:

\[
\begin{align*}
p_{\text{sunny}} &= 0.1 \\
p_{\text{rain}} &= 0.2 \\
p_{\text{overcast}} &= 0.7 \\
H(\text{outlook}) &= -[p_{\text{sunny}} \cdot \log_2(p_{\text{sunny}}) + \\
& \quad p_{\text{rain}} \cdot \log_2(p_{\text{rain}}) + \\
& \quad p_{\text{overcast}} \cdot \log_2(p_{\text{overcast}})] \\
&= -[0.1 \cdot \log_2(0.1) + 0.2 \cdot \log_2(0.2) + 0.7 \cdot \log_2(0.7)] \\
&= 1.157
\end{align*}
\]
Entrophy and Information Gain

- Purpose of decision-tree split is to have each subset be “cleaner” than the original data set.
  - Entropy measures ‘cleanliness’ of a set of points with respect to a particular attribute (e.g. target label).
  - Maximum entropy is when all values are equally likely (boring uniform distribution).
  - Minimum entropy is when only one value is present.

- Entropy measures ‘cleanliness’ of a set of points with respect to a particular attribute (e.g. target label).
- Maximum entropy is when all values are equally likely (boring uniform distribution).
- Minimum entropy is when only one value is present.
Entropy and Information Gain

- Purpose of decision-tree split is to have each subset be “cleaner” than the original data set.
- If we split on ‘outlook’, we can measure the specific conditional entropy of the target label with respect to each possible value of outlook:

\[
\begin{align*}
H(\text{rain}|\text{outlook} = \text{sunny}) \\
H(\text{rain}|\text{outlook} = \text{rain}) \\
H(\text{rain}|\text{outlook} = \text{overcast})
\end{align*}
\]
Entropy and Information Gain

- **Purpose of decision-tree split** is to have each subset be “cleaner” than the original data set.

- **Conditional Entropy** is the average specific conditional entropy of splitting on an attribute:

  \[
  H(\text{target}|A) = \sum_{a} P(A = a) \cdot H(\text{target}|A = a)
  \]

| $v$   | $P(v)$ | $P(\text{rain}|\text{outlook}=v)$ | $H(\text{rain}|\text{outlook}=v)$ |
|-------|--------|----------------------------------|-----------------------------------|
| sunny | 0.1    | 0.1                              | 0.469                             |
| rain  | 0.2    | 0.8                              | 0.722                             |
| overcast | 0.7    | 0.5                            | 1.000                             |

\[
H(\text{rain}|\text{outlook}) = 0.1 \cdot 0.469 + 0.2 \cdot 0.722 + 0.7 \cdot 1 = 0.893
\]
Entropy and Information Gain

- Purpose of decision-tree split is to have each subset be “cleaner” than the original data set.
- *Information Gain* is how much we improve, on average, the “cleanliness” of the data by splitting on an attribute.
  - i.e., how much do we decrease the entropy

\[
I(\text{target}|A) = H(\text{target}) - H(\text{target}|A)
\]
\[
I(\text{rain}|\text{outlook}) = H(\text{rain}) - H(\text{rain}|\text{outlook})
\]

\[
\begin{align*}
P(\text{rain}) & = 0.52 \\
H(\text{rain}) & = 0.999 \\
I(\text{rain}|\text{outlook}) & = 0.999 - 0.893 = 0.106
\end{align*}
\]
Mutual Information

• Consider two random variables $A$ and $B$ that are not necessarily independent. The mutual information between $A$ and $B$ is the amount of information we learn about $B$ by knowing the value of $A$ (and vice versa – it is symmetric). It is computed as follows:

$$I(A; B) = H(B) - \sum_a P(A = a) \cdot H(B|A = a)$$

• Consider the class $y$ of each training example and the value of feature $x_1$ to be random variables. The mutual information quantifies how much $x_1$ tells us about $y$.

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(Y)$</td>
<td>0.9183</td>
<td></td>
</tr>
</tbody>
</table>

- $P(x_1=0) = 0.6667$
- $P(x_1=1) = 0.3333$
- $H(Y|x_1=0) = 0.9710$
- $H(Y|x_1=1) = 0.7219$
- $I(Y;x_1) = 0.0304$
Choosing the Best Attribute (Method 2)

- Choose the attribute $x_j$ that has the highest mutual information with $y$.

$$\arg\max_j I(x_j; y) = H(y) - \sum_v P(x_j = v)H(y|x_j = v)$$

$$= \arg\min_j \sum_v P(x_j = v)H(y|x_j = v)$$

- Define $\tilde{J}(j)$ to be the expected remaining uncertainty about $y$ after testing $x_j$

$$\tilde{J}(j) = \sum_v P(x_j = v)H(y|x_j = v)$$