CSE 592: Data Mining
Instructor: Pedro Domingos

Today’s Agenda
- Inductive learning
- Decision trees

Inductive Learning

Supervised Learning
- Given: Training examples \((x, f(x))\) for some unknown function \(f\).
- Find: A good approximation to \(f\).

Example Applications
- Credit risk assessment
  - \(f(x)\): Proportion of customer and proposed purchase.
- Disease diagnosis
  - \(f(x)\): Disease (or maybe, recommended therapy).
- Face recognition
  - \(f(x)\): Blurry picture of person's face.
- Automatic steering
  - \(f(x)\): Blurry picture of road surface in front of car.

Appropriate Applications for Supervised Learning
- Situations where there is no human expert
  - Training examples \((x, f(x))\) for some unknown function \(f\).
- Situations where humans can perform the task but can’t describe how they do it.
  - Training examples \((x, f(x))\) for some unknown function \(f\).
- Situations where the desired function is changing frequently
  - Training examples \((x, f(x))\) for some unknown function \(f\).
- Situations where each user needs a customized function \(f\)
  - Training examples \((x, f(x))\) for some unknown function \(f\).

A Learning Problem
- Example: \(x_1, x_2, x_3, x_4, y\)
- \(x_1, x_2, x_3, x_4\) are training examples.
- \(y\) is the target function.

Example:
- \(x_1\):
  - 1 0 0 1 0 0
  - 2 0 1 0 0 0
  - 3 0 0 1 1 1
  - 4 1 0 1 1 1
  - 5 0 1 1 0 0
  - 6 1 1 0 0 0
  - 7 0 1 0 1 0
Hypothesis Spaces

- Complete Ignorance: There are \(2^n = 64\) possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 4 examples, we still have \(2^3 = 8\) possibilities.

```
0 0 0 0 0
0 0 0 1 0
0 0 1 0 0
0 0 1 1 1
0 1 0 0 0
0 1 0 1 0
0 1 1 0 0
0 1 1 1 1
1 0 0 0 0
1 0 0 1 0
1 0 1 0 0
1 0 1 1 1
1 1 0 0 0
1 1 0 1 0
1 1 1 0 0
1 1 1 1 1
```

Hypothesis Spaces (2)

- Simple Rules: There are only 56 simple conjunctive rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Counterexample</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x \wedge y)</td>
<td>1</td>
</tr>
<tr>
<td>(x \wedge \neg y)</td>
<td>1</td>
</tr>
<tr>
<td>(\neg x \wedge y)</td>
<td>1</td>
</tr>
<tr>
<td>(\neg x \wedge \neg y)</td>
<td>1</td>
</tr>
<tr>
<td>(x \wedge \neg x \wedge y)</td>
<td>1</td>
</tr>
<tr>
<td>(x \wedge \neg x \wedge \neg y)</td>
<td>1</td>
</tr>
<tr>
<td>(\neg x \wedge \neg x \wedge y)</td>
<td>1</td>
</tr>
<tr>
<td>(\neg x \wedge \neg x \wedge \neg y)</td>
<td>1</td>
</tr>
<tr>
<td>(x \wedge \neg x \wedge y \wedge \neg x)</td>
<td>1</td>
</tr>
<tr>
<td>(\neg x \wedge \neg x \wedge y \wedge \neg x)</td>
<td>1</td>
</tr>
</tbody>
</table>

No simple rule explains the data. The same is true for simple-clauses.

Hypothesis Space (3)

- \(n\)-off-n rules: There are \(2^n\) possible rules (include simple conjunctive and clauses).

<table>
<thead>
<tr>
<th>Counterexample</th>
<th>variable</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(z)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg x)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg z)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x \wedge y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x \wedge \neg y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg x \wedge y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg x \wedge \neg y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x \wedge \neg x \wedge y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(x \wedge \neg x \wedge \neg y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg x \wedge \neg x \wedge y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\neg x \wedge \neg x \wedge \neg y)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Two Views of Learning

- Learning is the removal of our remaining uncertainty. Suppose we knew that the unknown function was an \(x \wedge y\) boolean function, then we could use the training examples to find which function it is.

- Learning requires guessing a good, small hypothesis class. We can start with a very small class and enlarge it until it contains an hypothesis that fits the data.

We could be wrong!

- Our prior knowledge might be wrong
- Our guess of the hypothesis class could be wrong

Example: \(x \wedge \neg y\) or \(x \wedge y\) is also consistent with the training data.

Example: \(x \wedge \neg x \wedge y\) or \(x \wedge \neg x \wedge \neg y\) is also consistent with the training data.

If either of these is the unknown function, then we will make errors when we are given new a values.

Two Strategies for Machine Learning

- Develop Languages for Expressing Prior Knowledge: Rule grammars and stochastic models.

In either case:

- Develop Algorithms for Finding an Hypothesis that Fits the Data

Terminology

- Training example: An example of the form \((x, f(x))\).
- Target function (target concept): The true function \(f\).
- Hypothesis: A proposed function \(h\) to be similar to \(f\).
- Concept: A boolean function. Examples for which \(f(x) = 1\) are called positive examples or positive instances of the concept. Examples for which \(f(x) = 0\) are called negative examples or negative instances.
- Classifier: A learning function. The possible values \(f(x) \in \{1, \ldots, K\}\) are called the classes or class labels.
- Hypothesis Space: The space of all hypothesis that can, in principle, be output by a learning algorithm.
- Version Space: The space of all hypotheses in the hypothesis space that have yet been ruled out by a training example.
Key Issues in Machine Learning
- What are good hypothesis spaces?
- Which spaces have been useful in practical applications and why?
- What algorithms can work with these spaces?
- Are there general design principles for machine learning algorithms?
- How can we optimize accuracy on future data points?
- This is sometimes called the "pruning problem".
- How can we have confidence in the results?
- How much training data is required to find accurate hypotheses? (the statistical question)
- Are some learning problems computationally intractable?
- (the computational question)
- How can we formulate application problems as machine learning problems? (the engineering question)

A Framework for Hypothesis Spaces
- Size: Does the hypothesis space have a fixed size or variable size?
  - Fixed-size spaces are easier to understand, but variable-size spaces are generally more useful. Variable-size spaces introduce the problem of contracting.
- Randomness: Is each hypothesis deterministic or stochastic?
  - This affects how we evaluate hypotheses. With a deterministic hypothesis, a training example is either unassigned (correctly predicted) or misassigned (incorrectly predicted).
  - With a stochastic hypothesis, a training example is more likely or less likely.
- Parameterization: Is each hypothesis described by a set of symbolic (discrete) values or is it described by a set of continuous parameters?
  - Both are required, we say the hypothesis space has a mixed parameterization.
  - Discrete parameters must be found by combinatorial search methods; continuous parameters can be found by commercial search methods.

A Framework for Learning Algorithms
- Search Procedure:
  - Direction Computation: solve for the hypothesis directly.
  - Local Search: start with an initial hypothesis, make small improvements until a local optimum.
  - Convergent Search: start with an empty hypothesis, gradually add structure to it until local optimum.
- Timing
  - Eager: Analyze the training data and construct an explicit hypothesis.
  - Lazy: Store the training data and wait until a test data point is presented, then construct an all-encompassing hypothesis to classify that one data point.
- Online vs. Batch (for eager algorithms)
  - Online: Analyze each training example as it is presented.
  - Batch: Collect training examples, analyze them, output one hypothesis.

Decision Trees

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 $k(x)$.

Suppose the features are Outlook ($x_1$), Temperature ($x_2$), Humidity ($x_3$), and Wind ($x_4$). Then the feature vector $x = (\text{Sunny}, \text{Hot}, \text{High}, \text{Strong})$ will be classified as No. The Temperature feature is irrelevant.
Decision Tree Hypothesis Space

If the features are continuous, internal nodes may test the value of a feature against a threshold.

Decision Tree Decision Boundaries

Decision trees divide the feature space into sub-parallel rectangles, and label each rectangle with one of the K classes.

Decision Trees Can Represent Any Boolean Function

The tree will in the worst case require exponentially many nodes; however.

Decision Trees Provide Variable-Size 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, some boolean functions involving three features (e.g., \((x_1 \land x_2) \lor \neg x_3\))
- etc.

Learning Algorithm for Decision Trees

The same basic learning algorithm has been discovered by many people independently:

\[
\text{GrowTree}(S) =
\begin{cases}
\text{leaf} & \text{if} \left( y = \text{false for all} \ (x, y) \in S \right) \ \text{return} \ \text{leaf} \ v \ \text{false} \\
\text{leaf} & \text{if} \left( y = \text{true for all} \ (x, y) \in S \right) \ \text{return} \ \text{leaf} \ v \ \text{true} \\
\text{node} & \text{else} \\
\left( \text{choose best attribute } a_i \right) \\
S_1 = \left\{ (x, y) \in S \right\} \text{ with } x_i = 0 \\
S_2 = \left\{ (x, y) \in S \right\} \text{ with } x_i = 1 \\
\text{return} \ \text{node} \ v \ \text{GrowTree}(S_1) \ v \ \text{GrowTree}(S_2)
\end{cases}
\]

Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step binomial search and choose the attribute that gives the lowest error rate on the training data.

\[
\text{ChooseBestAttrib}(S) =
\begin{cases}
\text{choose } j \text{ to minimize } J_j \text{ computed as follows:} \\
S_0 = \left\{ (x, y) \in S \right\} \text{ with } x_j = 0 \\
S_1 = \left\{ (x, y) \in S \right\} \text{ with } x_j = 1 \\
p_0 = \text{the most common value of } y \in S_0 \\
p_1 = \text{the most common value of } y \in S_1 \\
\beta_0 = \text{number of examples} \ (x, y) \in S_0 \text{ with } y \neq p_0 \\
\beta_1 = \text{number of examples} \ (x, y) \in S_1 \text{ with } y \neq p_1 \\
J_j = \beta_0 + \beta_1 \text{ (total error if we split on this feature)} \\
\text{return } j
\end{cases}
\]
Choosing the Best Attribute—An Example

Choosing the Best Attribute (3)

A Better Heuristic From Information Theory

Entropy

Mutual Information

Visualizing Heuristics
Non-Boolean Features

- Features with multiple discrete values
  - Construct a multway split?
  - Test for one value versus all of the others?
  - Group the values into two disjoint subsets?
- Real-valued features
  - Consider a threshold split using each observed value of the feature.

Whichever method is used, the mutual information can be computed to choose the best split.

Attributes with Many Values

Problem:
- If attribute has many values, Gain will select it
- Imagine using Date = Jan.3,1996 as attribute

One approach: use GainRatio instead

\[
\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}
\]

\[
\text{SplitInformation}(S, A) = - \sum_{v=1}^{c} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}
\]

where \(S_v\) is subset of \(S\) for which \(A\) has value \(v\).

Overfitting in Decision Trees

Consider adding a noisy training example:
- Sunny, Hot, Normal, Strong, PlayTennis=True
  - What effect on tree?

Learning Parity with Noise

When learning exclusive (XOR parity), all split look equally good. If extra random boolean features are included, they also look equally good. Hence, decision tree algorithms cannot distinguish random noisy features from parity features.

Unknown Attribute Values

What if some examples are missing values of \(A\)?

Use training example anyway, sort through tree
- If node \(n\) tests \(A\), assign most common value of \(A\) among other examples sorted to node \(n\)
- Assign most common value of \(A\) among other examples with same target value
- Assign probability \(p_i\) to each possible value \(v_i\) of \(A\)
  - Assign fraction \(p_i\) of example to each descendant in tree
Classify new examples in same fashion.

Overfitting

Consider error of hypothesis \(h\) over:
- Training data: \(\text{error}_{\text{train}}(h)\)
- Entire distribution \(D\) of data: \(\text{error}_D(h)\)

Hypothesis \(h \in H\) overfits training data if there is an alternative hypothesis \(h' \in H\) such that

\[
\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')
\]

and

\[
\text{error}_D(h) > \text{error}_D(h')
\]
Overfitting in Decision Tree Learning

Avoiding Overfitting
How can we avoid overfitting?
- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select “best” tree:
- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

Reduced-Error Pruning
Split data into training and validation set
Do until further pruning is harmful:
1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy

Effect of Reduced-Error Pruning

Rule Post-Pruning
1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use
   Perhaps most frequently used method (e.g., C4.5)

Converting A Tree to Rules
Summary

- Inductive learning
- Decision trees
  - Representation
  - Tree growth
  - Heuristics
  - Overfitting and pruning
  - Scaling up

Scaling Up

- ID3, C4.5, etc. assume data fits in main memory (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data (OK for up to millions of examples)
- VFDT: at most one sequential scan (OK for up to billions of examples)

IF (Outlook = Sunny) AND (Humidity = High) THEN PlayTennis = No

IF (Outlook = Sunny) AND (Humidity = Normal) THEN PlayTennis = Yes

...