CSE 592: Data Mining

Instructor: Pedro Domingos

Today's Agenda

- Inductive learning
- Decision trees

Inductive Learning

Supervised Learning

- **Given:** Training examples $\langle \mathbf{x}, f(\mathbf{x}) \rangle$ for some unknown function f.
- **Find:** A good approximation to f.

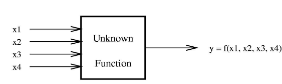
Example Applications

- **Credit risk assessment**
  $\mathbf{x}$: Properties of customer and proposed purchase.
  $f(\mathbf{x})$: Approve purchase or not.
- **Disease diagnosis**
  $\mathbf{x}$: Properties of patient (symptoms, lab tests)
  $f(\mathbf{x})$: Disease (or maybe, recommended therapy)
- **Face recognition**
  $\mathbf{x}$: Bitmap picture of person's face
  $f(\mathbf{x})$: Name of the person.
- **Automatic Steering**
  $\mathbf{x}$: Bitmap picture of road surface in front of car.
  $f(\mathbf{x})$: Degrees to turn the steering wheel.

Appropriate Applications for Supervised Learning

- **Situations where there is no human expert**
  $\mathbf{x}$: Bond graph for a new molecule.
  $f(\mathbf{x})$: Predicted binding strength to AIDS protease molecule.
- **Situations where humans can perform the task but can't describe how they do it.**
  $\mathbf{x}$: Bitmap picture of hand-written character
  $f(\mathbf{x})$: Ascii code of the character
- **Situations where the desired function is changing frequently**
  $\mathbf{x}$: Description of stock prices and trades for last 10 days.
  $f(\mathbf{x})$: Recommended stock transactions
- **Situations where each user needs a customized function f**
  $\mathbf{x}$: Incoming email message.
  $f(\mathbf{x})$: Importance score for presenting to user (or deleting without presenting).

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	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 $2^{16} = 65536$ 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 7 examples, we still have 2^9 possibilities.

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
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	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Hypothesis Spaces (2)

- **Simple Rules.** There are only 16 simple conjunctive rules.

Rule	Counterexample
$\Rightarrow y$	1
$x_1 \Rightarrow y$	3
$x_2 \Rightarrow y$	2
$x_3 \Rightarrow y$	1
$x_4 \Rightarrow y$	7
$x_1 \wedge x_2 \Rightarrow y$	3
$x_1 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \Rightarrow y$	3
$x_2 \wedge x_4 \Rightarrow y$	3
$x_3 \wedge x_4 \Rightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3

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

Hypothesis Space (3)

- **m-of-n rules**. There are 32 possible rules (includes simple conjunctions and clauses).

	Counterexample			
variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	3	3	–	–
$\{x_1, x_3\}$	4	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_3\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	***	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Two Views of Learning

- **Learning is the removal of our remaining uncertainty**. Suppose we *knew* that the unknown function was an m-of-n boolean function, then we could use the training examples to infer 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**
  The smaller the hypothesis class, the more likely we are wrong.

Example: $x_4 \wedge Oneof\{x_1, x_3\} \Rightarrow y$ is also consistent with the training data.

Example: $x_4 \wedge \neg x_2 \Rightarrow y$ 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 x values.

Two Strategies for Machine Learning

- **Develop Languages for Expressing Prior Knowledge:** Rule grammars and stochastic models.
- **Develop Flexible Hypothesis Spaces:** Nested collections of hypotheses.
  Decision trees, rules, neural networks, cases.

In either case:

- **Develop Algorithms for Finding an Hypothesis that Fits the Data**

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.
- **Target function (target concept).** The true function f.
- **Hypothesis**. A proposed function h believed to be similar to f.
- **Concept**. A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances.**
- **Classifier**. A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \ldots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space**. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space**. The space of all hypotheses in the hypothesis space that have not 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 "problem of overfitting".
- **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 overfitting.
- **Randomness**. Is each hypothesis **deterministic** or **stochastic**?
  This affects how we evaluate hypotheses. With a deterministic hypothesis, a training example is either *consistent* (correctly predicted) or *inconsistent* (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) choices or is it described by a set of **continuous** parameters? If 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 numerical 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.
  Constructive 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 ad hoc 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 an hypothesis.

Decision Trees

Learning Decision Trees

Decision trees provide a very popular and efficient hypothesis space.

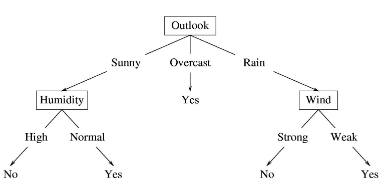
- **Variable Size**. Any boolean function can be represented.
- **Deterministic**.
- **Discrete and Continuous Parameters**.

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

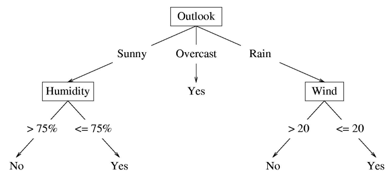
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



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

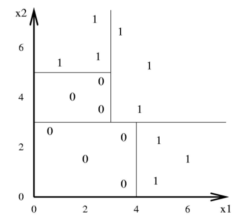
Decision Tree Hypothesis Space

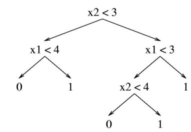
If the features are continuous, internal nodes may test the value of a feature against a threshold.



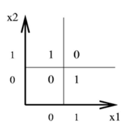
Decision Tree Decision Boundaries

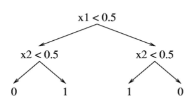
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





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., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$)
- **etc.**

Learning Algorithm for Decision Trees

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

GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y\rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y\rangle \in S$) **return** new leaf(1)
else
  choose best attribute x_j
  $S_0 =$ all $\langle \mathbf{x}, y\rangle \in S$ with $x_j = 0$;
  $S_1 =$ all $\langle \mathbf{x}, y\rangle \in S$ with $x_j = 1$;
  return new node(x_j, GrowTree(S_0), GrowTree(S_1))

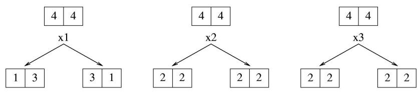
Choosing the Best Attribute

One way to choose the best attribute is to perform a 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:
  $S_0 =$ all $\langle \mathbf{x}, y\rangle \in S$ with $x_j = 0$;
  $S_1 =$ all $\langle \mathbf{x}, y\rangle \in S$ with $x_j = 1$;
  $y_0 =$ the most common value of y in S_0
  $y_1 =$ the most common value of y in S_1
  $J_0 =$ number of examples $\langle \mathbf{x}, y\rangle \in S_0$ with $y \neq y_0$
  $J_1 =$ number of examples $\langle \mathbf{x}, y\rangle \in S_1$ with $y \neq y_1$
  $J_j = J_0 + J_1$ (total errors if we split on this feature)
return j

Choosing the Best Attribute—An Example

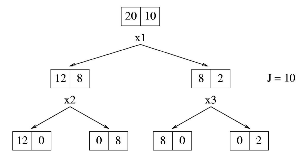
x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



J=2 J=4 J=4

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:

$P(V=0)$	$P(V=1)$
0.2	0.8

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

$$S(V=v) = -\lg P(V=v).$$

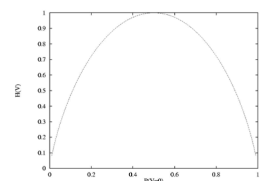
An event with probability 1 gives us zero surprise.
An event with probability 0 gives us infinite surprise!
It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.
This is also called the ***description length*** of $V=v$.
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

Entropy

The ***entropy*** of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H=v) \lg P(H=v).$$

This is the average surprise of describing the result of one "trial" of V (one coin toss).



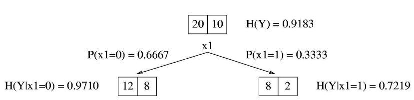
Entropy can be viewed as a measure of uncertainty.

Mutual Information

Now 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 knowning the value of A (and vice versa—it is symmetric). It is computed as follows:

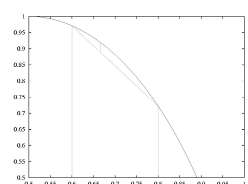
$$I(A;B) = H(B) - \sum_b P(B=b) \cdot H(A|B=b)$$

In particular, consider the class Y of each training example and the value of feature x_1 to be random variables. Then the mutual information quantifies how much x_1 tells us about the value of the class Y.

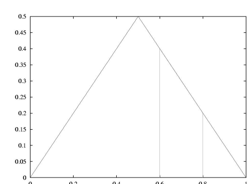


I(Y;x1) = 0.0304

Visualizing Heuristics



Entropy



Absolute Error

Mutual information works because it is a convex measure.

Non-Boolean Features

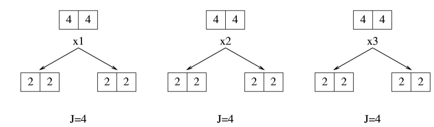
- **Features with multiple discrete values**
  Construct a multiway split?
  Test for one value versus all of the others?
  Group the values into two disjoint subsets?
- **Real-valued features**
  Consider a threshold split using each observed value of the feature.

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

Learning Parity with Noise

When learning exclusive-or (2-bit parity), all splits 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.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



Attributes with Many Values

Problem:

- If attribute has many values, *Gain* will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use *GainRatio* instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Unknown Attribute Values

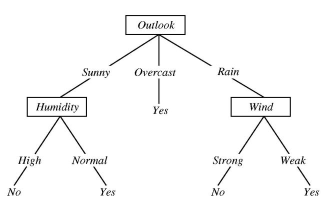
What if some examples are missing values of A?

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 in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

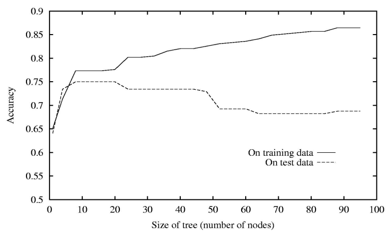
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

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

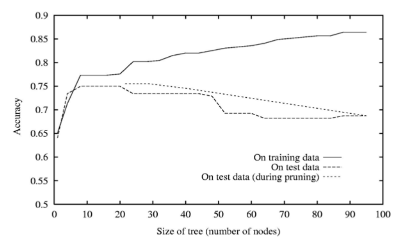
Reduced-Error Pruning

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

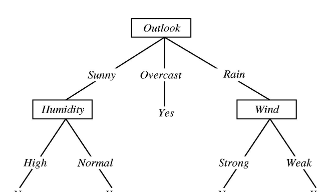


Rule Post-Pruning

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



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

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

...

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)

Summary

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