The approach to learning covered in this segment will be quite a bit different from that of neural networks. The output of the learning procedure will be a symbolic expression that is interpretable and can be combined with other knowledge in symbolic form.

On the other hand the approaches to be covered here are very sensitive to noisy data. The neural network approach is relatively robust in the presence of noise.

The topics to be considered are as follows:

1) Decision Trees
2) Learning Logical Descriptions
   (a) Current best hypothesis
   (b) Version Spaces

Consider the following data set from Russell and Norvig.

It consists of twelve restaurant visits. They are characterized by 10 different attributes. The goal to be learned is whether or not the people involved in these visits will wait at the restaurant for a table or go elsewhere.
Inductive Learning

In any inductive learning learning problem we have a set of input output pairs. The output $Y$ is the categorization that we want to learn.

$\langle X, Y \rangle \ X = \text{input} \ Y = \text{output} \ y = f(x)$

From a collection of examples, we want a function $h$ that approximates $f$ as closely as possible.

The aim of the learning problem is to construct $h$

Restaurant Example Continued

Here is $f$ for the example above. It is the decision tree actually used by the customers to generate the behavior found in the restaurant visits given above.

The Algorithm

Here is the first step in constructing $h$ for the data above. The $h$ should give the same behavior as $f$ on the data set – but there is no reason why it should be identical. We have not guarantee that it will give the same results on other examples.
The Algorithm

Note that the decision tree algorithm chooses as its next attribute the one that yields the largest number of definitive answers for the examples in the answer set. You should see why **Patrons** is a better attribute to pick than **Type**.

Applications

Induction of Rules for: Chemical Process Control
What control settings yield substances of high quality—in the production of pellets of uranium dioxide. (Leech 1986)

Credit Analysis for Loans
American Express UK (Michie 1989)

Diagnosis of Mechanical Devices

Classification of Celestial Objects

Logic and Generalization

Logic as a representation language yields a notion of learning and generalization.

If hypothesis H1 with definition C1 is a generalization of hypothesis H2 with definition C2 then we must have:

$$\forall x C_2(x) \rightarrow C_1(x)$$

The following are ways to generalize a logical expression in accordance with this definition.
Generalization (cont)

1. Replacing Constants with Variables
   Color(Obj1, red)
   Generalizes to
   Color(X, red)
   Color(obj, red) => Color(Obj, X)

2) Adding a disjunct to an expression

   Shape(X, round) \ Size(X, small)
   \ Color(X, red)
   generalizes to
   Shape(X, round) \ Size(X, small) \ (Color(X, red) \ Color(x, blue))

3. Dropping conditions from a conjunctive expression

   Color(x, red) \ Shape(X, round)
   \ Size(x, small)
   generalizes to
   Color(x, red) \ Shape(x, round)

4) Replacing a property with its parent in a class hierarchy.
   If we know that primary-Color is a superclass of Color(x,red)
   generalizes to
   Color(x, primary-color)

Current Best Hypothesis Search

The idea here is to just keep the best hypothesis. So, one begins with a positive example and constructs an expression that covers the positive example and as little else as possible.

Then when a negative example is considered, the hypothesis is specialized to exclude the negative example — but yet to still include all positive examples considered so far.

When a positive example is considered, the hypothesis is generalized to cover this positive example, but yet to still not cover any of the negative examples considered so far.

Examples

Example X1 -- Positive

H1: \ x WillWait(x) <-> Alternate

Example X2 -- Negative

H2: \ x WillWait(x) <-> Alternate(x) \ Patrons(x, s)
Examples (cont)

Example X3 -- Positive

H3: \( \forall x \text{ WillWait}(x) \rightarrow \text{Patrons}(x, \text{Some}) \)

Example X4 -- Positive

H4: \( \forall x \text{ WillWait}(x) \leftrightarrow \text{Patrons}(x, \text{some}) \)

\( \land \text{(Patrons}(x, \text{Full}) \land \text{Fri/Sat}(x)) \)

Version Spaces

The current-best hypothesis search approach only considers one hypothesis at a time – even though at each point there may be multiple hypotheses that can be entertained. Thus backtracking is necessary.

The version space algorithm represents the space of all possible hypotheses that are consistent with the data seen so far. This is the set of hypotheses between \( G \) (most general boundary) and \( S \) (most specific boundary).

Version Spaces (cont)

G – General Boundary Set

S – Most Specific Boundary Set

The current version space is the set of hypotheses consistent with the examples so far.

Every member of the \( S \) – Set is consistent with obs so far, and there are no consistent hypothesis that are more specific.

Every member of the \( G \) – Set is Consistent with obs so far, and there are no consistent hypotheses that are more general.

Converging boundaries of the \( G \) and \( S \) Sets in the candidate elimination algorithm.
The set of hypotheses is viewed as a lattice with the most general on the top and the most specific at the bottom. This is illustrated below for a simple language characterizing objects as for size, color, and shape. (example taken from Luger and Stubblefield).

G: True (most general Hypotheses)
S: False (empty)

Initially we set G to be the most general hypothesis and S to be the most specific hypothesis.

Example (cont)

First consider separately half of the version space algorithm. This half considers only positive examples, begins with the most specific S and generalizes to cover each example in turn.

G: {obj(X, Y, Z)}
Negative: obj(small, red, brick)

G: {obj(large, Y, Z), obj(X, white, Z), obj(X, blue, Z), obj(X, Y, ball), obj(X, Y, cube)}

G: {obj(large, Y, Z), obj(X, white, Z), obj(X, Y, ball)}
Negative: obj(large, blue, cube)

G: {obj(X, white, Z), obj(X, Y, ball)}

G: {obj(Y, X, ball)}

Specific to general search of the version space learning the concept "ball"
Candidate Elimination Algorithm

Now we put both halves of the algorithm together and modify both S and G as examples are considered.

Initialize G to contain one element, the null description (all variables)
Initialize S to contain one element: the first positive example.

i) For each new positive instance p:
   Delete all members of G that fail to match P.
   For every s in S, if s does not match P, replace s with its most specific generalizations that match P.
   Delete from S any hypothesis more general than some other hypothesis in S.
   Delete from S any hypothesis more general than some hypothesis in G.

ii) For each new negative instance n:
   Delete all members of s that match n.
   For each g in G that matches n, replace g with its most general specializations that do not match n.
   Delete from G any hypothesis more specific than some other hypothesis in G.
   Delete from G any hypothesis more specific than some hypothesis in S.

iii) If G = S and both are singletons
     Concept found
     If G, S become empty,
     Fail

Final Illustration

Now here is the algorithm put together on the same example.

G: \{obj(X, Y, Z)\}
S: {}
Positive: obj(small, red, ball)

G: \{obj(X, Y, Z)\}
S: \{obj(small, red, ball)\}
Negative: obj(small, blue, brick)

G: \{obj(X, red, Z), obj(X, Y, ball)\}
S: \{obj(small, red, ball)\}
Positive: obj(large, red, ball)

G: \{obj(X, red, Z), obj(X, Y, ball)\}
S: \{obj(X, red, ball)\}
Negative: obj(large, red, cube)

G: \{obj(X, Y, ball)\}
S: \{obj(X, red, ball)\}
The candidate elimination algorithm learning the concept "red ball."