Learning

R&N: ch 19, ch 20

based on material from
Ray Mooney, Daphne
Koller, Kevin Murphy,
Marie desJardins

Types of Learning

- Supervised Learning - classification, prediction
- Unsupervised Learning – clustering, segmentation, pattern discovery
- Reinforcement Learning – learning MDPs, online learning

Supervised Learning

- Someone gives you a bunch of examples, telling you what each one is
- Eventually, you figure out the mapping from properties (features) of the examples to their type (classification)

Supervised Learning

- Logic-based approaches:
  - learn a function $f(X) \to \{\text{true}, \text{false}\}$
- Statistical/Probabilistic approaches
  - Learn a probability distribution $p(Y|X)$
Outline

- Logic-based Approaches
  - Decision Trees (last time)
  - Version Spaces (today)
  - PAC learning (we won’t cover, ☹)
- Instance-based approaches
- Statistical Approaches

Predicate-Learning Methods

- Version space

Version Spaces

- The “version space” is the set of all hypotheses that are consistent with the training instances processed so far.
- An algorithm:
  - \( V := H \); the version space \( V \) is ALL hypotheses \( H \)
  - For each example \( e \):
    - Eliminate any member of \( V \) that disagrees with \( e \)
    - If \( V \) is empty, FAIL
  - Return \( V \) as the set of consistent hypotheses

Version Spaces: The Problem

- **PROBLEM:** \( V \) is huge!!
- Suppose you have \( N \) attributes, each with \( k \) possible values
- Suppose you allow a hypothesis to be any disjunction of instances
- There are \( k^N \) possible instances \( \rightarrow |H| = 2^{k^N} \)
- If \( N=5 \) and \( k=2 \), \( |H| = 2^{32}!! \)
**Version Spaces: The Tricks**

First Trick: Don’t allow arbitrary disjunctions
- Organize the feature values into a hierarchy of allowed disjunctions, e.g.
  - any-color
    - yellow
    - white
    - blue
    - black
- Now there are only 7 “abstract values” instead of 16 disjunctive combinations (e.g., “black or white” isn’t allowed)

Second Trick: Define a partial ordering on H (“general to specific”) and only keep track of the upper bound and lower bound of the version space

RESULT: An incremental, possibly efficient algorithm!

**Rewarded Card Example**

\[(r=1) \lor \ldots \lor (r=10) \lor (r=J) \lor (r=Q) \lor (r=K) \leftrightarrow \text{ANY-RANK}(r)\]
\[(r=1) \lor \ldots \lor (r=10) \leftrightarrow \text{NUM}(r)\]
\[(r=J) \lor (r=Q) \lor (r=K) \leftrightarrow \text{FACE}(r)\]
\[(s=\heartsuit) \lor (s=\clubsuit) \lor (s=\diamondsuit) \lor (s=\spadesuit) \leftrightarrow \text{ANY-SUIT}(s)\]
\[(s=\heartsuit) \lor (s=\clubsuit) \leftrightarrow \text{BLACK}(s)\]
\[(s=\diamondsuit) \lor (s=\spadesuit) \leftrightarrow \text{RED}(s)\]

A hypothesis is any sentence of the form:
\[R(r) \land S(s) \leftrightarrow \text{IN-CLASS}([r,s])\]

where:
- \(R(r)\) is ANY-RANK\((r)\), NUM\((r)\), FACE\((r)\), or \((r=j)\)
- \(S(s)\) is ANY-SUIT\((s)\), BLACK\((s)\), RED\((s)\), or \((s=k)\)

**Simplified Representation**

For simplicity, we represent a concept by \(rs\), with:
- \(r\) \(\in\) \{a, n, f, 1, …, 10, j, q, k\}
- \(s\) \(\in\) \{a, b, r, ♥, ♦, ♠, ♣\}

For example:
- \(n♥\) represents:
  \[\text{NUM}(r) \land (s=♥) \leftrightarrow \text{IN-CLASS}([r,s])\]
- \(aa\) represents:
  \[\text{ANY-RANK}(r) \land \text{ANY-SUIT}(s) \leftrightarrow \text{IN-CLASS}([r,s])\]

**Extension of a Hypothesis**

The *extension* of a hypothesis \(h\) is the set of objects that satisfies \(h\)

Examples:
- The extension of \(f♥\) is: \{j♥, q♥, k♥\}
- The extension of \(aa\) is the set of all cards
More General/Specific Relation

Let \( h_1 \) and \( h_2 \) be two hypotheses in \( H \)
\( h_1 \) is more general than \( h_2 \) iff the extension of \( h_1 \) is a proper superset of the extension of \( h_2 \)

Examples:
- \( aa \) is more general than \( f \)
- \( f \) is more general than \( q \)
- \( fr \) and \( nr \) are not comparable

More General/Specific Relation

Let \( h_1 \) and \( h_2 \) be two hypotheses in \( H \)
\( h_1 \) is more general than \( h_2 \) iff the extension of \( h_1 \) is a proper superset of the extension of \( h_2 \)
The inverse of the “more general” relation is the “more specific” relation
The “more general” relation defines a partial ordering on the hypotheses in \( H \)

Example: Subset of Partial Order

G-Boundary / S-Boundary of V

A hypothesis in \( V \) is most general iff no hypothesis in \( V \) is more general
\( G \)-boundary \( G \) of \( V \): Set of most general hypotheses in \( V \)
**G-Boundary / S-Boundary of V**

- A hypothesis in V is **most general** iff no hypothesis in V is more general
- **G-boundary** G of V: Set of most general hypotheses in V
- A hypothesis in V is **most specific** iff no hypothesis in V is more specific
- **S-boundary** S of V: Set of most specific hypotheses in V

**Example: G-/S-Boundaries of V**

Here, both G and S have size 1. This is not the case in general!

**Example: G-/S-Boundaries of V**

We replace every hypothesis in S whose extension does not contain 4♣ by its generalization set

**Example: G-/S-Boundaries of V**

The **generalization set** of an hypothesis h is the set of the hypotheses that are immediately more general than h

**Example: G-/S-Boundaries of V**

Let 7♣ be the next (positive) example

Generalization set of 4♣
Example: G-/S-Boundaries of V

Let 7♣ be the next (positive) example

Example: G-/S-Boundaries of V

Specialization set of aa

Let 5♥ be the next (negative) example

Example: G-/S-Boundaries of V

G and S, and all hypotheses in between form exactly the version space

Example: G-/S-Boundaries of V

At this stage ...

Do 8♣, 6♦, j♠ satisfy CONCEPT?
Example: G-/S-Boundaries of V

Let 2♣ be the next (positive) example

Example: G-/S-Boundaries of V

Let j♠ be the next (negative) example

Example: G-/S-Boundaries of V

Let us return to the version space ...
... and let 8♣ be the next (negative) example

Example: G-/S-Boundaries of V

The only most specific hypothesis disagrees with this example, so no hypothesis in H agrees with all examples
Example: G-/S-Boundaries of V

Let us return to the version space ...
... and let $j$ be the next (positive) example.

The only most general hypothesis disagrees with this example, so no hypothesis in $H$ agrees with all examples.

Version Space Update

1. $x \leftarrow$ new example
2. If $x$ is positive then
   \[(G,S) \leftarrow \text{POSITIVE-UPDATE}(G,S,x)\]
3. Else
   \[(G,S) \leftarrow \text{NEGATIVE-UPDATE}(G,S,x)\]
4. If $G$ or $S$ is empty then return failure

**POSITIVE-UPDATE(G,S,x)**

1. Eliminate all hypotheses in $G$ that do not agree with $x$
2. Minimally generalize all hypotheses in $S$ until they are consistent with $x$

Using the generalization sets of the hypotheses
**POSITIVE-UPDATE(G,S,x)**

1. Eliminate all hypotheses in G that do not agree with x
2. Minimally generalize all hypotheses in S until they are consistent with x
3. Remove from S every hypothesis that is neither more specific than nor equal to a hypothesis in G
   
   *(This step was not needed in the card example)*

**NEGATIVE-UPDATE(G,S,x)**

1. Eliminate all hypotheses in S that do not agree with x
2. Minimally specialize all hypotheses in G until they are consistent with x
3. Remove from G every hypothesis that is neither more general than nor equal to a hypothesis in S
4. Remove from G every hypothesis that is more specific than another hypothesis in G
5. Return (G,S)

**Example-Selection Strategy (aka Active Learning)**

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses
- Then a single hypothesis will be isolated in \(O(\log |H|)\) steps
Example-Selection Strategy

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example.
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses.
- Then a single hypothesis will be isolated in $O(\log |H|)$ steps.
- But picking the object that eliminates half the version space may be expensive.

Noise

- If some examples are misclassified, the version space may collapse.
- Possible solution: Maintain several $G$- and $S$-boundaries, e.g., consistent with all examples, all examples but one, etc...

Version Spaces

- Useful for understanding logical (consistency-based) approaches to learning.
- Practical if strong hypothesis bias (concept hierarchies, conjunctive hypothesis).
- Don’t handle noise well.
Statistical Approaches

- Instance-based Learning (20.4)
- Statistical Learning (20.1)
- Neural Networks (20.5) on Dec. 2

Issues

- Distance measure
  - Most common: euclidean
  - Better distance measures: normalize each variable by standard deviation
- Choosing k
  - Increasing k reduces variance, increases bias
- For high-dimensional space, problem that the nearest neighbor may not be very close at all!
- Memory-based technique. Must make a pass through the data for each classification. This can be prohibitive for large data sets.
- Indexing the data can help; for example KD trees

Nearest Neighbor Methods

To classify a new input vector $x$, examine the k-closest training data points to $x$ and assign the object to the most frequently occurring class

Bayesian Learning
Example: Candy Bags

- Candy comes in two flavors: cherry (☺) and lime (⁄)
- Candy is wrapped, can’t tell which flavor until opened
- There are 5 kinds of bags of candy:
  - \( H_1 \) = all cherry
  - \( H_2 \) = 75% cherry, 25% lime
  - \( H_3 \) = 50% cherry, 50% lime
  - \( H_4 \) = 25% cherry, 75% lime
  - \( H_5 \) = 100% lime
- Given a new bag of candy, predict \( H \)
- Observations: \( D_1, D_2, D_3, \ldots \)

Bayesian Learning

- Calculate the probability of each hypothesis, given the data, and make prediction weighted by this probability (i.e. use all the hypothesis, not just the single best)
  \[
P(h_i | d) = \frac{P(d | h_i)P(h_i)}{\sum_i P(d | h_i)P(h_i)} = \alpha P(d | h_i)P(h_i)
\]
- Now, if we want to predict some unknown quantity \( X \)
  \[
P(X | d) = \sum_i P(X | h_i)P(h_i | d)
\]

Bayesian Learning cont.

- Calculating \( P(h|d) \)
  \[
P(h_i | d) \propto P(d | h_i)P(h_i)
\]
- Assume the observations are i.i.d.—independent and identically distributed
  \[
P(d | h_i) = \prod_j P(d_j | h_i)
\]

Example:

- Hypothesis Prior over \( h_1, \ldots, h_5 \) is \( \{0.1,0.2,0.4,0.2,0.1\} \)
- Data:
  - Q1: After seeing \( d_1 \), what is \( P(h_1|d_1) \)?
  - Q2: After seeing \( d_1 \), what is \( P(d_2 = \bullet|d_1) \)?