Inductive Classification
Machine learning tasks

• Classification
• Problem Solving

– Classification/categorization: the set of categories is given (e.g. lion, frog)
– Classification/clustering: the set of categories is not known (we need to “cluster” instances by similarity)
– First case is TRAINED or SUPERVISED
– Second is UNSUPERVISED
Machine learning as a categorization task: definition

• Given:
  – A description of an instance, \( x \in X \), where \( X \) is the
    *instance language* or *instance space* (e.g. a way of
    *representing instances*).
  – A fixed (known) set of categories: \( C = \{ c_1, c_2, \ldots c_n \} \)

• Determine:
  – The category of \( x \): \( c(x) \in C \), where \( c(x) \) is a classification
    function whose domain is \( X \) and whose range is \( C \).
  – If \( c(x) \) is a binary function \( C = \{ 0,1 \} \) (\{true,false\},
    \{positive, negative\}) then it is called a *concept*. 
Learning for Categorization

• A training example is an instance $x \in X$, paired with its correct category $c(x)$: $<x, c(x)>$ for an unknown categorization function, $c$. $x$ is represented by its selected features.

• Given a set of training examples, $D$

• Find a hypothesized categorization function, $h(x)$, such that:

$$\forall <x, c(x)> \in D : h(x) = c(x)$$

Consistency
Sample Category Learning Problem

- **Instance language**: \(\langle\text{size, color, shape}\rangle\)
  - \(\text{size} \in \{\text{small, medium, large}\}\)
  - \(\text{color} \in \{\text{red, blue, green}\}\)
  - \(\text{shape} \in \{\text{square, circle, triangle}\}\)

- \(C = \{\text{positive, negative}\}\)

- \(D:\)
<table>
<thead>
<tr>
<th>Instances</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>C(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

This means that every instance is represented by a set of attributes, or features, each taking values in a finite set.
Hypothesis representation

- We can represent an hypothesis for $C(x)$ e.g. with a \textit{boolean expression}, or a rule, e.g.
- If $(\text{color}=\text{red}) \land (\text{shape}=\text{circle})$ THEN $C=\text{positive}$
- Or equivalently: $\text{red} \land \text{circle}$

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>
Hypothesis Selection

• Many hypotheses are usually consistent with the training data.
  – red & circle
  – (small & circle) or (large & red)
  – (small & red & circle) or (large & red & circle)
  – not [ ( red & triangle) or (blue & circle) ]
  – not [ ( small & red & triangle) or (large & blue & circle) ]

• Bias
  – Any criteria other than consistency with the training data that is used to select a hypothesis.
  – E.g: preferring a conjunctive form

Notice that the first is the minimized conjunctive form
Generalization

• Hypotheses must **generalize** to correctly classify instances **not in the training data**.

• *Simply memorizing training examples is a consistent hypothesis that does not generalize:*

  $((\text{small} \& \text{red} \& \text{circle}) \text{or} (\text{large} \& \text{red} \& \text{circle})) \& (\text{not}((\text{small} \& \text{red} \& \text{triangle}) \text{or} (\text{large} \& \text{blue} \& \text{circle}) \text{or} \ldots))$

• *Occam’s razor (later in this course):*

  – Finding a *simple* hypothesis helps to ensure generalization.
Hypothesis Space

- Learned functions a priori restrict to a given hypothesis space, $H$, of functions $h(x)$ that can be considered as definitions of $c(x)$.
- For example, we restrict to conjunctive functions like `small&red`.
- For learning concepts on instances described by $n$ discrete-valued features, consider the space of conjunctive hypotheses represented by a vector of $n$ features $<f_1, f_2, \ldots f_n>$ where each $f_i$ is either:
  - $?$, a wild card indicating no constraint on the $i$th feature
  - A specific value from the domain of the $i$th feature
  - $\emptyset$ indicating no value is acceptable
- Sample conjunctive hypotheses are
  - $<$big, red, ?> (equivalent to big&red, or size=big&color=red)
  - $<$?, ?, ?> (most general hypothesis, $\forall x, c(x) = 1$)
  - $<\emptyset, \emptyset, \emptyset>$ (most specific hypothesis, $\forall x, c(x) = 0$)
Inductive Learning Hypothesis

- Any function that is found to approximate the target concept well on a sufficiently large set of training examples will also approximate the target function well on unobserved examples.
- Assumes that the training and test examples are drawn independently from the same underlying distribution (IID).
- This is a fundamentally unprovable hypothesis unless additional assumptions are made about the target concept and the notion of “approximating the target function well on unobserved examples” is defined appropriately (cf. computational learning theory).
Evaluation of Classification Learning

• Classification accuracy (% of instances classified correctly).
  – Measured on an independent test data.

• Training time (efficiency of training algorithm).

• Testing time (efficiency of subsequent classification).
Category Learning as Search

- Category learning can be viewed as searching the hypothesis space for one (or more) hypotheses that are consistent with the training data.

- Consider an instance space consisting of $n$ binary features which therefore has $2^n$ instances.

- For **conjunctive hypotheses** of binary features, there are 4 choices for each feature’s value: Ø, T, F, ? (feature is always false, true, false, or does’t matter), so there are $4^n$ syntactically distinct hypotheses.

- However, all hypotheses with 1 or more Øs are equivalent (Ø means “always false”, and hypotheses have the form of a conjunction), so there are $3^n + 1$ semantically distinct hypotheses.

- Eg with two attributes a and b:

  \[
  \text{FALSE}, ab, \overline{ab}, \overline{ab}, \overline{a} \overline{b}, a, b, \overline{a}, \overline{b}, \text{TRUE}
  \]
Example hypothesis space for conjunctive functions (two binary features)
Category Learning as Search (2)

• Conjunctive hypotheses are a small subset of the space of possible functions
• Most real-world hypothesis spaces are intractably large or even infinite.
Learning by Enumeration

• For any finite or countably infinite hypothesis space, one can simply enumerate and test hypotheses one at a time until a consistent one is found.

  For each $h$ in $H$ do:
  
  If $h$ is consistent with the training data $D$,
  then terminate and return $h$.

• This algorithm is guaranteed to terminate with a consistent hypothesis if one exists; however, it is obviously computationally intractable for almost any practical problem.
Efficient Learning

- Is there a way to learn conjunctive concepts without enumerating them?
- How do human subjects learn conjunctive concepts?
- Is there a way to efficiently find an unconstrained boolean function consistent with a set of discrete-valued training instances?
- If so, is it a useful/practical algorithm?
Conjunctive Rule Learning

- Conjunctive descriptions are easily learned by finding all commonalities shared by all positive examples.

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

Learned rule: red & circle $\rightarrow$ positive

- Must check consistency with negative examples. If inconsistent, no conjunctive rule exists.
Limitations of Conjunctive Rules

- If a concept does not have a single set of necessary and sufficient conditions, conjunctive learning fails.

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
<tr>
<td>5</td>
<td>medium</td>
<td>red</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

Learned rule: red & circle → positive

Inconsistent with negative example #5!
Disjunctive Concepts

- Concept may be disjunctive (in this case a conjunctive hypothesis cannot be found!)

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
<tr>
<td>5</td>
<td>medium</td>
<td>red</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>

Learned rules: small & circle $\rightarrow$ positive
large & red $\rightarrow$ positive

$h(x) = (\text{small} \ & \ \text{circle}) \ or \ (\text{large} \ & \ \text{red})$
Using the Generality Structure

• By exploiting the structure imposed by the generality of hypotheses, an hypothesis space can be searched for consistent hypotheses without enumerating or explicitly exploring all hypotheses.
• An instance, \( x \in X \), is said to **satisfy** an hypothesis, \( h \), iff \( h(x) = 1 \) (positive)
• Given two hypotheses \( h_1 \) and \( h_2 \), \( h_1 \) is **more general than or equal to** \( h_2 \) (\( h_1 \geq h_2 \)) iff every instance that satisfies \( h_2 \) also satisfies \( h_1 \).
• Given two hypotheses \( h_1 \) and \( h_2 \), \( h_1 \) is (**strictly** more general than) \( h_2 \) (\( h_1 > h_2 \)) iff \( h_1 \geq h_2 \) and it is not the case that \( h_2 \geq h_1 \).
• **Generality defines a partial order on hypotheses.**
Examples of Generality

- **Conjunctive feature vectors**
  - `<?, red, ?>` is more general than `<?, red, circle>` (remember `?`="any value is ok")
  - Neither of `<?, red, ?>` and `<?, ?, circle>` is more general than the other.

- **Axis-parallel rectangles in 2-d space**

  - A is more general than B
  - Neither of A and C are more general than the other.
Digression: What are these rectangles?

Suppose X axis is cholesterol and Y is age. Green points are positive for illness M and each rectangle represents the rule: if $A < x < B$ AND $C < y < D$ then M
Question

- Consider $h_1$: <big, red, circle> and $h_2$: <?, blue, ?>

- Is $h_2 > h_1$?

- Given two hypotheses $h_1$ and $h_2$, $h_1$ is more general than or equal to $h_2$ ($h_1 \geq h_2$) iff every instance that satisfies $h_2$ also satisfies $h_1$. 
Back to conjunctive hypotheses: Sample Generalization Lattice

Size: \{\text{small, big}\}  \quad \text{Color:} \ \{\text{red, blue}\}  \quad \text{Shape:} \ \{\text{circle, square}\}

\[
\begin{array}{c}
< ?, ?, ?> \\
< ?, ?, \text{circ} > < \text{big, ?, ?> < ?, \text{red, ?> < ?, \text{blue, ?> < ?, ?, \text{sqr} >
\end{array}
\]

\[
\begin{array}{c}
< ?, \text{red, circ} > < \text{big, ?, circ} > < \text{big, red, ?> < \text{big, blue, ?> < \text{sm, ?, circ} > < ?, \text{blue, circ} > < ?, \text{red, sqr} > < \text{sm, ?, sqr} > < \text{sm, red, ?> < \text{sm, blue, ?> < \text{big, ?, sqa} > < ?, \text{blue, sqa} >
\end{array}
\]

\[
\begin{array}{c}
< \text{big, red, circ} > < \text{sm, red, circ} > < \text{big, blue, circ} > < \text{sm, blue, circ} > < \text{big, red, sqr} > < \text{sm, red, sqa} > < \text{sm, blue, sqa} > < \text{big, blue, sqa} > < \text{sm, blue, sqa} >
\end{array}
\]

\[
< \emptyset, \emptyset, \emptyset >
\]

Number of hypotheses = $3^3 + 1 = 28$
Algorithm 1: Most Specific Learner (Find-S)

- Find the most-specific hypothesis (least-general generalization, LGG) that is consistent with the training data.
- Incrementally update hypothesis after every positive example, generalizing it just enough to satisfy the new example.
- For conjunctive feature vectors, this is easy:
  
  Initialize \( h = \langle \emptyset, \emptyset, \ldots \emptyset \rangle \)
  
  For each positive training instance \( x \) in \( D \)
    
    For each feature \( f_i \)
      
      If the constraint on \( f_i \) in \( h \) is not satisfied by \( x \)
        
        If \( f_i \) in \( h \) is \( \emptyset \)
          
          then set \( f_i \) in \( h \) to the value of \( f_i \) in \( x \)
        
        else set \( f_i \) in \( h \) to “?”
      
      If \( h \) is consistent with the negative training instances in \( D \)
        
        then return \( h \)
      
      else no consistent hypothesis exists

Time complexity: \( O(|D| \ n) \) if \( n \) is the number of features
(an odd) Example: learning a user profile

We assume there are no errors in $D$ (often not true!)

"Click" is the classification function defined in $(0,1)$ (will the user click on the page?)

<table>
<thead>
<tr>
<th>domain</th>
<th>platform</th>
<th>Browser</th>
<th>day</th>
<th>screen</th>
<th>country</th>
<th>Click?</th>
</tr>
</thead>
<tbody>
<tr>
<td>edu</td>
<td>Mac</td>
<td>Net3</td>
<td>Lu</td>
<td>XVGA</td>
<td>America</td>
<td>Si</td>
</tr>
<tr>
<td>com</td>
<td>Mac</td>
<td>NetCom</td>
<td>Lu</td>
<td>XVGA</td>
<td>America</td>
<td>Si</td>
</tr>
<tr>
<td>com</td>
<td>PC</td>
<td>IExpl</td>
<td>Sab</td>
<td>VGA</td>
<td>Asia</td>
<td>No</td>
</tr>
<tr>
<td>org</td>
<td>Unix</td>
<td>Net2</td>
<td>Gio</td>
<td>XVGA</td>
<td>Europa</td>
<td>Si</td>
</tr>
</tbody>
</table>

$C(x)$ values
Find-S

Training set $D$

$X1 = (<edu, mac, Net3, Lun, XVGA, America>, 1)$
$X2 = (<com, mac, Net3, Mar, XVGA, America>, 1)$
$X3 = (<com, PC, IE, Sab, VGA, Eur>, 0)$
$X4 = (<org, Unix, Net2, Mer, XVGA, America>, 1)$

Hypothesis space $H$

$h0 = <\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset>$
$h1 = <edu, mac, Net3, Lun, XVGA, America>$
$h2 = <?, mac, Net3, ?, XVGA, America>$
$h3 = <?, mac, Net3, ?, XVGA, America>$
$h4 = <?, ?, ?, ?, XVGA, America>$

generality
Properties of Find-S

• For conjunctive feature vectors, the most-specific hypothesis is unique and found by Find-S (if enough examples are provided).

• If the most specific hypothesis is not consistent with the negative examples, then there is no consistent function in the hypothesis space, since, by definition, it cannot be made more specific and retain consistency with the positive examples.

• Notice however that FIND S does not consider negative examples!

• For conjunctive feature vectors, if the most-specific hypothesis is inconsistent, then the target concept must be disjunctive.
Issues with Find-S

- Given sufficient training examples, does Find-S converge to a correct definition of the target concept (assuming it is in the hypothesis space)?
- How do we know when the hypothesis has converged to a correct definition?
- Why prefer the most-specific hypothesis? Are more general hypotheses consistent? What about the most-general hypothesis? What about the simplest hypothesis?
- If the least general generalization LGG is not unique
  - Which LGG should be chosen?
  - How can a single consistent LGG be efficiently computed or determined not to exist?
- What if there is noise in the training data and some training examples are incorrectly labeled?
Effect of Noise in Training Data

• Frequently realistic training data is corrupted by errors (noise) in the features or class values.

• Such noise can result in missing valid generalizations.
  - For example, imagine there are many positive examples like #1 and #2, but out of many negative examples, only one like #5 that actually resulted from a error in labeling.

<table>
<thead>
<tr>
<th>Example</th>
<th>Size</th>
<th>Color</th>
<th>Shape</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>small</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>2</td>
<td>large</td>
<td>red</td>
<td>circle</td>
<td>positive</td>
</tr>
<tr>
<td>3</td>
<td>small</td>
<td>red</td>
<td>triangle</td>
<td>negative</td>
</tr>
<tr>
<td>4</td>
<td>large</td>
<td>blue</td>
<td>circle</td>
<td>negative</td>
</tr>
<tr>
<td>5</td>
<td>medium</td>
<td>red</td>
<td>circle</td>
<td>negative</td>
</tr>
</tbody>
</table>
Algorithm 2: Version Space

• Given an hypothesis space, $H$, and training data, $D$, the *version space* is the *complete subset of $H$ that is consistent (compatible)* with $D$.

• The version space can be naively generated for any finite $H$ by enumerating all hypotheses and eliminating the inconsistent ones.

• Can one compute the version space more efficiently than using enumeration?
Version Space with S and G

- The version space can be represented more compactly by maintaining **two boundary sets** of hypotheses, $S$, the set of **most specific consistent hypotheses**, and $G$, the set of **most general consistent hypotheses**:

  \[
  S = \{ s \in H \mid \text{Consistent}(s, D) \land \neg \exists s' \in H[ s > s' \land \text{Consistent}(s', D)] \} \\
  G = \{ g \in H \mid \text{Consistent}(g, D) \land \neg \exists g' \in H[ g' > g \land \text{Consistent}(g', D)] \}
  \]

- $S$ and $G$ represent the entire version space via its boundaries in the generalization lattice:
Version Space Lattice

Size: \{sm, big\}  Color: \{red, blue\}  Shape: \{circ, squ\}

Color Code:

G
S
other VS

<<big, red, squ> positive>
<<sm, blue, circ> negative>
Candidate Elimination (Version Space) Algorithm

Initialize $G$ to the set of most-general hypotheses in $H$
Initialize $S$ to the set of most-specific hypotheses in $H$

For each training example, $d$, do:

If $d$ is a positive example then:
- Remove from $G$ any hypotheses that do not match $d$
- For each hypothesis $s$ in $S$ that does not match $d$
  - Remove $s$ from $S$
  - Add to $S$ all minimal generalizations, $h$, of $s$ such that:
    1) $h$ matches $d$
    2) some member of $G$ is more general than $h$
- Remove from $S$ any $h$ that is more general than another hypothesis in $S$

If $d$ is a negative example then:
- Remove from $S$ any hypotheses that match $d$
- For each hypothesis $g$ in $G$ that matches $d$
  - Remove $g$ from $G$
  - Add to $G$ all minimal specializations, $h$, of $g$ such that:
    1) $h$ does not match $d$
    2) some member of $S$ is more specific than $h$
- Remove from $G$ any $h$ that is more specific than another hypothesis in $G$
Sample VS Trace

\[ S = \{< \emptyset, \emptyset, \emptyset>\}; \quad G = \{<?, ?, ?>\} \]

**Positive: X: <big, red, circle>**
Nothing to remove from \( G \) (\( X \) is compatible with \( G \), \( G \) would “accept” \( X \))
Minimal generalization of only \( S \) element is <big, red, circle> which is more specific than \( G \).
\( S = \{<\text{big, red, circle}>\}; \quad G = \{<?, ?, ?>\} \)

**Negative: Y: <small, red, triangle>**
Nothing to remove from \( S \). (\( Y \) is compatible with \( S = \{<\text{big, red, circle}>\}, \ S \) would reject \( Y \))
Minimal specializations of <?, ?, ?> that would reject the negative example are:  <medium, ?, ?>, <big, ?, ?>, <?, blue, ?>, <?, green, ?>, <?, ?, circle>, <?, ?, square> but most are not more general than some element of \( S \) hence the final set \( G \) is  <big, ?, ?>, <?, ?, circle>
\( S = \{<\text{big, red, circle}>\}; \quad G = \{<\text{big, ?, ?>, <?, ?, circle}>\)
Sample VS Trace (cont)

\(S=\{<\text{big, red, circle}>\}; G=\{<\text{big, ?, ?>, <?, ?, circle}>\}\)

Positive: \(Z\): \(<\text{small, red, circle}>\)
Remove \(<\text{big, ?, ?>\) from \(G\)
Minimal generalization of \(<\text{big, red, circle}>\) that would accept the positive example is \(<\text{?, red, circle}>\)
\(S=\{<\text{?, red, circle}>\}; G=\{<\text{?, ?, circle}>\}\)

Negative: \(N\): \(<\text{big, blue, circle}>\)
Nothing to remove from \(S\) (\(S\) would correctly reject the example)
Minimal specializations of \(<\text{?, ?, circle}>\) that would reject the example are:
\(<\text{small, ?, circle}>\), \(<\text{medium, ?, circle}>\), \(<\text{?, red, circle}>\), \(<\text{?, green, circle}>\) but most are not more general than some element of \(S\).
\(S=\{<\text{?, red, circle}>\}; G=\{<\text{?, red, circle}>\}\)

\(S=G\); Converged!
Example 2 (a generic 3-monomial conjunctive function)
Properties of VS Algorithm

- $S$ summarizes the relevant information in the positive examples (relative to $H$) so that positive examples do not need to be retained.
- $G$ summarizes the relevant information in the negative examples, so that negative examples do not need to be retained.
- Result is not affected by the order in which examples are processes but computational efficiency may.
- Positive examples move the $S$ boundary up; Negative examples move the $G$ boundary down.
- If $S$ and $G$ converge to the same hypothesis, then it is the only one in $H$ that is consistent with the data.
- If $S$ and $G$ become empty (if one does the other must also) then there is no hypothesis in $H$ consistent with the data.
Correctness of Learning

- Since the entire version space is maintained, given a continuous stream of noise-free training examples, the VS algorithm will eventually converge to the correct target concept if it is in the hypothesis space, $H$, or eventually correctly determine that it is not in $H$.
- Convergence is correctly indicated when $S=G$. 
Computational Complexity of VS

- Computing the $S$ set for conjunctive feature vectors is linear in the number of features and the number of training examples.
- Computing the $G$ set for conjunctive feature vectors is exponential in the number of training examples in the worst case.
- In more expressive languages (than conjunctive rules), both $S$ and $G$ can grow exponentially.
- The order in which examples are processed can significantly affect computational complexity.
Before we start presenting new (and more practical) ML algorithms..
A number of relevant issues that apply to any ML problem/algorithm

1. Feature selection and object representation
2. Best order to present training examples
3. Multiple categories
4. Inductive Bias: specific assumptions might improve learning
Feature Selection

• Many factors affect the success of machine learning on a given task.
• The representation and quality of the example data is first and foremost.
• Theoretically, having more features should result in more discriminating power.
• However, practical experience with machine learning algorithms has shown that this is not always the case.
The importance of features (attributes) selection

- Reduce the cost of learning by reducing the number of attributes.
- Provide better learning performance compared to using full attribute set.
Feature selection is task dependent

To classify lions and frogs in the appropriate category, a simple color histogram could perform very well: $f=\text{color}$
Feature selection is task dependent

To classify horses and lions, more features are needed. Some would be useless, e.g. Color, Number-of-legs...since they would not help to differentiate the 2 categories.
Feature selection methods

There are two approach for attribute selection.

• **Filter approach** attempt to assess the merits of attributes from the data, ignoring learning algorithm.

• **Wrapper approach** the attributes subset selection is done using the learning algorithm as a black box.
Filtering
Filtering

- A feature $f_i$ is said to be strongly relevant to the target concept(s) if the probability distribution of the class values, given the full feature set, changes when $f_i$ is removed.
- A feature $f_i$ is said to be weakly relevant if it is not strongly relevant and the probability distribution of the class values, given some subset $S$ (containing $f_i$) of the full feature set, does not change when $f_i$ is removed.
Example: whant to learn chair(x) (chair(x)=true if x=chair)

\[ x=(\text{color, has-back, 4-legs}) \]

Prior probability of instances in D
\[ P(\text{chair})=\frac{8}{16}=0.5 \]
\[ P(\text{table})=P(\text{not(chair)})=\frac{8}{16}=0.5 \]

Consider the feature:
4-legs (boolean)
Example

• If we group instances according to this feature, we have:

Grouping instances according to 4-legs does not vary the probability distribution of the 2 categories: it remains 0.5!!
```
Conclusion

• “4-legs” is not a good feature to correctly classify tables and chairs!!
• Instead “has-back” would be enough to perfectly separate the 2 categories (given the training set of instances of the example)
• In fact has-back=true would group only chairs, has-back=false would group only tables)
• But..
```
Measuring the “probability distribution”:
Entropy filtering

Ranking according to entropy gain of attributes.
Entropy for given set of data with 2 classes can be defined as

\[
Entropy = - \sum_{1}^{2} p(j) \log_2 p(j)
\]

After classification that use one attribute we can calculate gain

\[
GAIN = Entropy - \left( \sum_{1}^{2} \frac{n_i}{n} Entropy(i) \right)
\]

Larger value of gain better attribute.
Wrt previous example

• $E = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$

• If we classify according to 4-legs:
  – $E(4\text{-legs=true}) = 1$ and we have 6 chairs and 6 tables
  – $E(4\text{-legs=false}) = 1$ and we have 2 chairs and 2 tables
  – $\text{GAIN} = 1 - ((6/12) \times 1 + (3/6) \times 1) = 1 - 1 = 0$

• If we classify according to has-back:
  – $E(\text{has-back=true}) = 0$ (they are all chairs)
  – $E(\text{has-back=false}) = 0$ (they are all tables)
  – Hence $\text{GAIN} = 1 - 0 = 1$
In previous example we had two extremes: perfectly useless (4-legs) and perfectly useful (has-back) feature.

Common case is that entropy changes, but not from 1 to 0!!

Ex. black instances positive, white are negative.

Partition the data set in two groups according to \( f_1 = 1 \): positive and negative.

At each step, choose the feature that “reduces entropy” most. Work towards “node purity”.

Pure nodes are those including instances with a unique classification (e.g. all tables).
Wrappers

- Employs the target learning algorithm to evaluate feature sets
- Uses an induction algorithm along with a statistical re-sampling technique such as cross-validation to estimate the final accuracy of feature subsets
Wrappers
Say we have features A, B, C and classifier M. We want to predict $C(X)$ given the smallest possible subset of \{A,B,C\}, while achieving maximal performance (accuracy).

<table>
<thead>
<tr>
<th>FEATURE SET</th>
<th>CLASSIFIER</th>
<th>PERFORMANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A,B,C}</td>
<td>$M$</td>
<td>98%</td>
</tr>
<tr>
<td>{A,B}</td>
<td>$M$</td>
<td>98%</td>
</tr>
<tr>
<td>{A,C}</td>
<td>$M$</td>
<td>77%</td>
</tr>
<tr>
<td>{B,C}</td>
<td>$M$</td>
<td>56%</td>
</tr>
<tr>
<td>{A}</td>
<td>$M$</td>
<td>89%</td>
</tr>
<tr>
<td>{B}</td>
<td>$M$</td>
<td>90%</td>
</tr>
<tr>
<td>{C}</td>
<td>$M$</td>
<td>91%</td>
</tr>
<tr>
<td>{}</td>
<td>$M$</td>
<td>85%</td>
</tr>
</tbody>
</table>
Filters vs Wrappers: Wrappers

The set of all subsets is the power set and its size is $2^{\mid V\mid}$. Hence for large $V$ we cannot do this procedure exhaustively; instead we rely on heuristic search of the space of all possible feature subsets.

\[ \{A\} 89 \rightarrow \{A,C\} 77 \rightarrow \{A,B\} 98 \]
\[ \{B\} 90 \rightarrow \{B,C\} 56 \rightarrow \{A,B\} 98 \]
\[ \{C\} 91 \rightarrow \{A,C\} 77 \rightarrow \{B,C\} 56 \]
\[ \{A,B,C\} 98 \]

this is prior probability of $C(X)$
Hill climbing

A common example of heuristic search is hill climbing: keep adding features one at a time until no further improvement can be achieved.

1. Let $v \leftarrow$ initial state of attributes.
2. Expand $v$: find $v$’s children.
3. Apply the evaluation function $f$ to each child $w$ of $v$.
4. Let $v' =$ the child $w$ with highest evaluation $f(w)$
5. If $f(v') > f(v)$ then $v \leftarrow v'$; go to 2
6. Return $v$
Hill Climbing

Step 1: with first expansion $f(v') > f(v)$ for all nodes
Step 2: C is expanded first, but condition is not met;
Step 3: B is expanded and nodes A,B and B,C are generated; A,B is the best
Step 4: A,B is expanded but condition is not met; OUTPUT is \{A,B\}
A number of relevant issues that apply to any ML problem/algorithm

1. Feature selection and object representation
2. Best order to present training examples
3. Multiple categories
4. Inductive Bias: specific assumptions might improve learning
Active Learning

• In *active learning*, the system is responsible for selecting good training examples and asking a teacher (oracle) to provide a class label.

• Goal is to **minimize the number of examples required to learn an accurate concept description**.

• The general idea is to select the training examples in a “smart” way
Toy Example: 1D classifier

Unlabeled data: labels are all 0 then all 1 (left to right)

Classifier (threshold function): $h_w(x) = 1$ if $x > w$ (0 otherwise)

Goal: find transition between 0 and 1 labels in minimum steps

Naïve method: choose points to label at random on line
  • Requires $O(n)$ training data to find underlying classifier

Better method: binary search for transition between 0 and 1
  • Requires $O(\log n)$ training data to find underlying classifier
  • Exponential reduction in training data size!
Active learning

• Why active learning?
  – Learning classifiers using labeled examples
  – But obtaining labels is
    • Time consuming, e.g., document classification
    • Expensive, e.g., medical decision (need doctors)
    • Sometimes dangerous, e.g., landmine detection
Active learning: choose labeled examples

Raw unlabeled data

$X_1, X_2, X_3, \ldots$

Learner requests labels for selected data

$(X_1, ?)$

$(X_1, Y_1)$

$(X_3, ?)$

$(X_3, Y_3)$

active learner

expert/oracle analyzes/experiments to determine labels

automatic classifier
Active learning example: pedestrian detection [Freund et al 03]
Active learning strategies

- **Uncertainty sampling**: label those points for which the current model is least certain as to what the correct output should be (if model is probabilistic)
- **Query by committee**: a variety of models are trained on the current labeled data, and vote on the output for unlabeled data; label those points for which the "committee" disagrees the most
- **Expected model change**: label those points that would most change the current model
- **Expected error reduction**: label those points that would most reduce the model's generalization error
- **Variance reduction**: label those points that would minimize output variance, which is one of the components of error
A common method is: take the example that “mostly differs” from those seen so far.
Typical heuristics for active learning

- Start with a pool of unlabeled data
- Pick a few points at random and get their labels
- Repeat
  - Query the unlabeled point that is closest to the boundary
  - (or most uncertain, or most likely to decrease overall uncertainty,...)

Knowing the label of the first two instances is more helpful at determining the positive/negative boundary than the third instance.

Which of the following instances would be an interesting example?
Active Learning with VS

- An ideal training example would eliminate half of the hypotheses in the current version space regardless of its label ($2^n \rightarrow 2^{n-1}$).
- If a training example matches half of the hypotheses in the version space, then the matching half is eliminated if the example is negative, and the other (non-matching) half is eliminated if the example is positive.
- Example:
  - Assume training set
    - Positive: <big, red, circle>
    - Negative: <small, red, triangle>
  - Current version space (after 2 examples):
    - {<big, red, circle>, <big, red, ?>, <big, ?, circle>, <?, red, circle> <?, ?, circle> <big, ?, ?>}
  - An optimal query: <big, blue, circle> (if +, then cuts 1,2 and 4, if -, it cuts 3,4,5)
- Given a ceiling of $\log_2 |VS|$ such examples will result in convergence. This is the best possible guarantee in general.
Using an Unconverged VS

- If the VS has not converged (or in general, if a learning algorithm L leaves you with many hypotheses), how does it classify a novel test instance?
  - If all elements of $S$ match an instance, then the entire version space match (since it is more general) and it can be confidently classified as positive (assuming target concept is in $H$).
  - If no element of $G$ matches an instance, then the entire version space must not (since it is more specific) and it can be confidently classified as negative (assuming target concept is in $H$).
  - Otherwise, one could vote all of the hypotheses in the VS (or just the $G$ and $S$ sets to avoid enumerating the VS) to give a classification with an associated confidence value.
  - Voting the entire VS is probabilistically optimal assuming the target concept is in $H$ and all hypotheses in $H$ are equally likely a priori.
A number of relevant issues that apply to any ML problem/algorithm

1. Feature selection and object representation
2. Best order to present training examples
3. Multiple categories
4. Inductive Bias: specific assumptions might improve learning
Learning for Multiple Categories

- What if the classification problem is not concept learning and involves more than two categories (e.g. select among several possible illnesses, given the symptoms)?
- Can treat as a series of concept learning problems (e.g. we use $n$ independent classifiers), where for each classifier $C_i$, all
  \[ x \in D, \text{s.t. } c_i(x) = 1 \]
  are treated as positive and all other instances in categories $C_j, j \neq i$ are treated as negative (one-versus-all).
- This will assign a unique category to each training instance but may assign a novel instance to zero or multiple categories.
- If the binary classifier produces confidence estimates (e.g. based on voting), then a novel instance can be assigned to the category with the highest confidence.
Example

- Classifier 1: red or not-red
- Classifier 2: blue or not-blue
- Classifier 3: yellow or not-yellow
A number of relevant issues that apply to any ML problem/algorithm

1. Feature selection and object representation
2. Best order to present training examples
3. Multiple categories
4. Inductive Bias: specific assumptions might improve learning
Inductive Bias

• A hypothesis space that does not include all possible classification functions on the instance space is said to incorporates a bias in the type of classifiers it can learn (e.g. restricting to conjunctive functions is a bias in boolean concept learning).

• Any means that a learning system uses to choose between two functions that are both consistent with the training data is called inductive bias.

• Inductive bias can take two forms:
  
  – **Language bias**: The language for representing concepts defines a hypothesis space that does not include all possible functions (e.g. disjunctive descriptions).
  
  – **Search bias**: The language is expressive enough to represent all possible functions (e.g. disjunctive normal form) but the search algorithm embodies a preference for certain consistent functions over others (e.g. conjunctive functions) This is called syntactic simplicity.
More in general, bias is a criterion for preferring a set of hypotheses over another.

- **a** and **b** belong to different languages (linear vrs. non-linear functions), **b** and **c** have a different search bias, since **c** relaxes consistency in favor of simplicity.
Unbiased Learning

- For instances described by $n$ features each with $m$ values, there are $m^n$ possible instances. If these are to be classified into $c$ categories, then there are $c^{m^n}$ possible classification functions (all possible ways of assigning $m^n$ instances to $c$ categories).
  - For $n=10$, $m=c=2$, there are approx. $3.4 \times 10^{38}$ possible boolean functions, of which only 59,049 can be represented as conjunctions (an incredibly small percentage!)
- However, unbiased learning is futile since if we consider all possible functions then simply memorizing the data without any real generalization is as good an option as any.
- Without bias, the version-space is always trivial. The unique most-specific hypothesis is the disjunction of the positive instances and the unique most general hypothesis is the negation of the disjunction of the negative instances:

$$S = \{(p_1 \lor p_2 \lor ... \lor p_k)\}$$
$$G = \{\neg(n_1 \lor n_2 \lor ... \lor n_j)\}$$
Ockham (Occam)’s Razor

• William of Ockham (1295-1349) was a Franciscan friar who applied the criteria to theology:
  – “Entities should not be multiplied beyond necessity” (Classical version but not an actual quote, which is: entia non sunt multiplicanda praeter necessitatem)
  – “The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.” (Einstein)

• Requires a precise definition of “simplicity”.
• Acts as a bias which assumes that nature itself is simple.
• Role of Occam’s razor in machine learning remains controversial (more on CLT course).