IMPLEMENTING INDUCTIVE BIAS I

- Training data
- Inductive biases
- Search methods
- Hypotheses
• In order to learn something you have to know a fair bit *(perhaps a lot)* about it to begin with.

- Inductive bias is the name that we give to our prior knowledge regarding the set of possible hypotheses appropriate for a given learning problem.

- Without such a bias a learning algorithm will not be able to generalize beyond the training examples.

- In addition to governing the ability of a learning algorithm to generalize, an inductive bias incurs a computational cost and thereby induces a *trade-off*.

- In choosing an inductive bias, we often have to trade solution quality against computational tractability.
Hypothesis Spaces

- Logical formulas
  - Conjunctions and disjunctions
  - CNF and DNF formulas
  - $k$-DNF (each conjunction has $k$ terms)
  - $k$-term-DNF (the formula has $k$ conjuncts)

- Artificial neural networks
  - Linear threshold units (perceptrons)
  - 3NN functions (three-layer neural network)
  - $k$-3NN functions (3NN with $k$ hidden units)

- Decision trees
  - DT functions — functions encoded as decision trees
  - DL functions - decision lists encoding functions as
    \[(t_1, c_1), \ldots, (t_k, c_k)\]
    in which the tests $t_i$ are evaluated in order returning the $c_i$ paired with the first test that evaluates to true.
  - $k$-DL functions — decision lists with $k$ tests
    - Note that decision lists and decision trees are completely general in that both can represent any boolean formula; however, decision trees represent some boolean formulas more compactly than decision lists. $k$-DL formulas are not completely general.
Computational Problems

- Finding a consistent hypothesis
  - Easy problems (polynomial time)
    - Logical conjunctions
    - Linear threshold units
    - $k$-DNF formulas
    - $k$-DL formulas
  - Hard problems (NP-complete)
    - $k$-term-DNF formulas
    - $k$-3NN functions

- Finding a simple\(^1\) hypothesis
  - Easy problems (polynomial time)
    - Logical conjunctions
  - Hard problems (NP-complete)
    - Finite state machines
    - Unknown complexity
    - 3NN functions
    - Decision trees

---

1. Where in this case “simple” means that the size of the hypothesis, \textit{e.g.}, the number of conjuncts in a conjunction, is within a polynomial factor of the size of the smallest hypothesis.
**Decision Trees**

- Decision trees for learning a concept $g$

```
0 1

0 1 0 1

\text{discrimination at nodes is achieved by using attribute functions } \{f_i\} \text{ that capture relevant features of the data}
```

- Training data in the form of a table

<table>
<thead>
<tr>
<th>#</th>
<th>attribute 1</th>
<th>…</th>
<th>attribute n</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
<tr>
<td>2</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
<tr>
<td>3</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
<tr>
<td>4</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
<tr>
<td>5</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
<tr>
<td>6</td>
<td>value</td>
<td></td>
<td>value</td>
<td>class</td>
</tr>
</tbody>
</table>
Building Decision Trees

- Given a set of training examples, create a new (root) node $n$ which is associated with all of the training examples and invoke \texttt{BUILDTREE} on $n$

Procedure: \texttt{BUILDTREE}(n)

1. If all of the examples associated with $n$ fall the same class, then quit
2. If all of the examples do not fall in the same class, then
   a. Choose some attribute $f$ and set it to be the attribute for $n$
   b. Partition the examples into subsets according to the values for $f$
   c. Create new nodes for each nonempty subset of examples in the partition
   d. Set the list of new nodes to the children of $n$
   e. Apply \texttt{BUILDTREE} recursively to each of the new nodes

- How should we choose the attributes?
Preference for Small Decision Trees

- Decision trees are expressive enough to represent any boolean formula
  - Given this expressivity we use a bias that prefers smaller decision trees to encourage generalization.

- Is there any justification for this preference?
  - By a simple counting argument, there are far fewer simple decision trees than there are complex ones.
  - Thus the probability is small that a simple hypothesis consistent with the data will have a large error.
  - In the absence of information to the contrary, a simple hypothesis which is consistent with the data is more likely to be correct than a complex one.

- How do we select small decision trees?
  - The general problem of finding the smallest decision tree consistent with the training data is intractable.
  - Instead we use BUILDTREE along with a heuristic for choosing attributes that tends to find small trees.
  - The resulting algorithm is not guaranteed to find the smallest decision tree but it works well in practice.
Choosing Attributes

• Consider the consequences of choosing a particular attribute in building a decision tree

Let $f$ be a binary attribute that we want to evaluate and $E$ be the set of examples associated with a node $n$

Let $E^c_v$ be the subset of $E$ such that $f(x) = v \in \{0, 1\}$ and $g(x) = c \in \{+, -\}$

The result of splitting $n$ using the attribute $f$ can be viewed in terms of creating two partitions:

$\{E^+, E^-\}$

0 1

$\{E_0^+, E_0^-\} \cup \{E_1^+, E_1^-\}$

where $E^c = E_0^c \cup E_1^c$ for $c \in \{+, -\}$

• How do we evaluate an attribute $f$ for a node $n$ on the basis of the result of splitting $n$ using $f$?
Choosing Attributes (continued)

• Here are some examples of bad splits

\[
\begin{align*}
|E| & \\
0 & 1
\end{align*}
\]

\[
\begin{align*}
|E_0^-| &= \frac{|E|}{4} & |E_1^-| &= \frac{|E|}{4} & |E_0^-| &= 0 & |E_1^-| &= \frac{|E|}{2} \\
|E_0^+| &= \frac{|E|}{4} & |E_1^+| &= \frac{|E|}{4} & |E_0^+| &= 0 & |E_1^+| &= \frac{|E|}{2}
\end{align*}
\]

• Here is an example of a good split

\[
\begin{align*}
|E| & \\
0 & 1
\end{align*}
\]

\[
\begin{align*}
|E_0^-| &= 0 & |E_1^-| &= \frac{|E|}{2} \\
|E_0^+| &= \frac{|E|}{2} & |E_1^+| &= 0
\end{align*}
\]
Choosing Attributes (continued)

- We want an evaluation function that has
  - a maximum when the examples are equally divided among the classes, and
  - a minimum when all of the examples belong to the same class.

- The following evaluation function satisfies the above requirements

\[
\frac{|E^+|}{|E|} \log |E^+| - \left\{ \frac{|E_0^+|}{|E|} \log |E_0^+| + \frac{|E_0^-|}{|E|} \log |E_0^-| \right\} + \frac{|E^-|}{|E|} \log |E^-| - \left\{ \frac{|E_1^+|}{|E|} \log |E_1^+| + \frac{|E_1^-|}{|E|} \log |E_1^-| \right\}
\]

where \( E_v = E_v^+ \cup E_v^- \) for \( v \in \{0, 1\} \)

- The above function provides a measure which is proportional to the change in information content which results from using an attribute.
  - We are interested in maximizing the information gain.
Choosing Attributes (continued)

- Maximizing information gain

\[
I(E^+, E^-) = \frac{|E^+|}{|E|} \log |E^+| \cdot \frac{|E^-|}{|E|} \log |E^-| - \frac{|E_0^+|}{|E_0|} \log |E_0^+| - \frac{|E_0^-|}{|E_0|} \log |E_0^-|
\]

Gain in information

\[
I(E^+, E^-) - \frac{|E_0^+|}{|E_0|} I(E_0^+, E_0^-) - \frac{|E_1^+|}{|E_1|} I(E_1^+, E_1^-)
\]
Choosing Attributes (continued)

- Behavior of the evaluation function

\[
\frac{|E_0|}{|E|} = 1
\]

\[
\frac{|E_1^+|}{|E_1|} = 1
\]

\[
\frac{|E_0|}{|E_0|} = 0
\]

\[
\frac{|E_1^+|}{|E_0|} = \frac{1}{2}
\]

\[
\frac{|E_0|}{|E|} = \frac{1}{2}
\]

\[
\frac{|E_1^+|}{|E_1|} = \frac{1}{2}
\]

where \( \frac{|E_1|}{|E|} = 1 - \frac{|E_0|}{|E|} \) and \( \frac{|E_1^-|}{|E_1|} = 1 - \frac{|E_1^+|}{|E_1|} \) and so on
Choosing Attributes (continued)

- There are problems with greedy selection
  - Suppose that the concept we are trying to learn is the following boolean function of two attributes
    \[ (\neg \text{attribute 1} \land \text{attribute 2}) \lor (\text{attribute 1} \land \neg \text{attribute 2}) \]
  - Now consider the following training data

<table>
<thead>
<tr>
<th>#</th>
<th>attribute 1</th>
<th>attribute 2</th>
<th>attribute 3</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

- All of the attributes have the same value according to our greedy heuristic for selecting attributes.
- This example can be modified to foil any limited look-ahead evaluation function where the cost of evaluation is exponential in the look ahead.
Decision Trees (continued)

• Greedy methods work well in practice
  - Using this evaluation function doesn’t guarantee that we will find the smallest decision tree but it has been shown to work quite well in practice.
  - A learning algorithm based on decision trees learned to fly a Cessna (using a flight simulator) better than the human experts that supplied it with training data.
  - British Petroleum uses an expert system generated by a decision-tree learning algorithm for controlling processes on their offshore oil platforms.

• Complications in using decision trees
  - **Missing data** — examples with missing attributes. What if you lose the patient’s blood test or the patient is never given a blood test in the first place?
  - **Highly specific attributes** — attributes such a person’s name maximize information gain but they do a poor job in terms of generalization. A person’s phone number makes a lousy attribute but it is commonly available.
  - **Continuous valued attributes** — discretization is one approach to dealing with continuous parameters. Do you discretize before or during the construction of the decision tree?
  - **Noisy data** — mistakes due to data entry, error-prone sensors, and stochastic phenomena abound in real-world learning problems.
### Decision Trees (continued)

- **Coping with noisy training data**
  - Consider the following training examples:

<table>
<thead>
<tr>
<th>#</th>
<th>attribute 1</th>
<th>attribute 2</th>
<th>attribute 3</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

- Suppose that $f$ splits a node into $\{E_0, E_1\}$.
- The expected numbers of + and - examples in $E_v$ for $v \in \{0, 1\}$ assuming that $f$ is irrelevant to $g$ are:
  $$\left|E^+\right|\left|\frac{E_v}{|E|}\right| \quad \text{and} \quad \left|E^-\right|\left|\frac{E_v}{|E|}\right|$$

- Statistically the observed numbers can deviate from the expectations — they’re just expectations after all.

- The $\chi^2$ (read “chi-square”) test enables us to quantify our confidence in the hypothesis that $f$ is irrelevant, based on how observation deviates from expectation.

- Using the $\chi^2$ test we pick a threshold confidence — typically 95% — to avoid selecting spurious attributes.
Evaluating Learning Algorithms

• How do you evaluate a given learning algorithm when faced with a particular problem to solve?

  - Simple partitioning — divide the training data $T$ into two sets, one used for training and the other used for evaluation.

  - Cross validation — partition the data $T$ into $M$ subsets $T_1, T_2, \ldots, T_M$ such that $T = T_1 \cup \ldots \cup T_M$ and the subsets are of roughly equal size.

    - For each $1 \leq i \leq M$, train the learning algorithm on the set $T - T_i$ and let $V_i$ be the performance of the hypothesis generated by program using $T_i$ for evaluation purposes.

    - The $M$-fold cross-validation estimate for the learning algorithm is defined as
      \[
      \frac{1}{M} \sum_{i=1}^{M} V_i
      \]

• Could you use cross validation to avoid the need for specifying a particular bias?

  - Using cross validation to select from among a set of learning algorithms is another way of implementing a learning algorithm and hence an inductive bias.