Contents

• Decision trees
  – Entropy (Shannon/Binary)
  – Overfitting
  – Some real world issues

• Reinforcement learning
  – Passive/Active
  – Exploration/exploitation
Learning

“A program for performing a task has been acquired by learning if it has been acquired by any means other than explicit programming” [Valiant, 1984].

- Learning is about generalizing from instances to concepts
- The concept to be learnt can be more or less anything:
  - A logical formula
  - A function
  - A formal language (grammatical inference)
  - A setting for some parameters
  - A way of moving from place A to place B
  - Etc.
Learning types

- **Supervised learning**
  - Learning from example *input/output pairs*
  - Analyzes the training data and produces an inferred function

- **Unsupervised learning**
  - No explicit feedback provided
  - E.g. clustering

- **Reinforcement learning**
  - The agent *acts* on its environment and receives some evaluation of its action (*reinforcement*), but is not told of which action is the correct one to achieve its goal
  - Like playing a game with unknown rules. After 100 moves the opponent says “you lose”
Chapter 18.3.4 – 18.3.6

DECISION TREES
Entropy

• **Entropy** is a measure of the uncertainty of a random variable
  – Getting more information reduce the entropy
• E.g. flipping a coin gives 1 **bit** entropy
• No uncertainty is defined as zero entropy
Shannon entropy

Let $X$ be a discrete random variable that takes values in the domain $\{x_1, \ldots, x_n\}$. Then the **Shannon entropy** of $X$ is described by

$$H(X) = \sum_{i=1}^{n} -P(x_i) \cdot \log_2 P(x_i)$$

If, for instance, $n = 4$ and all $x_i$ are equally probable, we get:

$$H(X) = -4 \cdot \frac{1}{4} \cdot \log_2 \frac{1}{4} = -4 \cdot \frac{1}{4} \cdot -2 = 2$$

If $P(x_1) = 1/2$ and the other $x_i$ have probability 1/6 we get

$$H(X) = - \frac{1}{2} \cdot \log_2 \frac{1}{2} - 3 \cdot \frac{1}{6} \cdot \log_2 \frac{1}{6} \approx \frac{1}{2} + 3 \cdot 0.43 = 1.79$$
Binary entropy

We define $B(p)$ to be the entropy of a Boolean random variable that is true with probability $p$:

$$B(p) = -(p \cdot \log_2 p + (1-p) \cdot \log_2 (1-p)) = B(1-p)$$

For example, we have

$$B\left(\frac{1}{4}\right) = -\left(\frac{1}{4} \log_2 \frac{1}{4} + \frac{3}{4} \log_2 \frac{3}{4}\right) \approx 0.811$$

and

$$B\left(\frac{1}{2}\right) = -\left(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}\right) = 1$$

and

$$B(1) = -(1 \log_2 1 + 0 \log_2 0) = 0$$
The Movie Conundrum

Example

Amanda frequently asks her friend Bob to go with her to the movies.

The problem is that Bob only agrees about half the time, and Amanda is fed up with asking.

She has therefore collected some data about the last 12 times she asked Bob to come, and now wants to use the data to predict when it would make sense to ask Bob.

Amanda decides to use her data to construct a decision tree that will make sure that she only asks Bob to go to the movies when there is a better than average chance that he will accept.
### Example

**Attributes:**
- **Type:** Comedy (C), Horror (H), Sci-Fi (SF), Romance (R)
- **Length (Len):** Short (S), Medium (M), Long (L)
- **3D:** yes/no
- **New:** yes/no
- **Animated (Ani):** yes/no
- **Time:** Early (E), Late (L)
- **Extra leg space (LSp):** yes/no
- **Week day (WDay):** Monday (Mon), Tuesday (Tue), Wednesday (Wed), Thursday (Thu), Friday (Fri), Saturday (Sat), Sunday (Sun)
- **Tom Hanks (TH):** yes/no
### Example

<table>
<thead>
<tr>
<th>Film</th>
<th>Type</th>
<th>Len</th>
<th>3D</th>
<th>New</th>
<th>Ani</th>
<th>Time</th>
<th>LSp</th>
<th>WDay</th>
<th>TH</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>C</td>
<td>S</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>E</td>
<td>n</td>
<td>Thu</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>$x_2$</td>
<td>H</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>L</td>
<td>n</td>
<td>Sun</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_3$</td>
<td>C</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
<td>Sat</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>$x_4$</td>
<td>H</td>
<td>M</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>y</td>
<td>Fri</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>$x_5$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
<td>Mon</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_6$</td>
<td>R</td>
<td>L</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>n</td>
<td>Mon</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_7$</td>
<td>R</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>L</td>
<td>y</td>
<td>Tue</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>$x_8$</td>
<td>C</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>L</td>
<td>y</td>
<td>Thu</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_9$</td>
<td>H</td>
<td>S</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>E</td>
<td>y</td>
<td>Fri</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>L</td>
<td>n</td>
<td>Sat</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>SF</td>
<td>S</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>L</td>
<td>n</td>
<td>Tue</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>H</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
<td>Sun</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>
Let’s test on 3D movie

Example

• There are 5 films in 3D and 7 in 2D.
  ➢ Of the 5 films in 3D, Bill went to 2.
  ➢ Of the 7 films in 2D, Bill went to 4.

• The remaining entropy in the yes-branch is:
  \[ B\left(\frac{2}{5}\right) = -\left(\frac{2}{5} \cdot \log_2 \frac{2}{5} + \frac{3}{5} \cdot \log_2 \frac{3}{5}\right) \approx 0.971 \]

• The remaining entropy in the no-branch is
  \[ B\left(\frac{4}{7}\right) = -\left(\frac{4}{7} \cdot \log_2 \frac{4}{7} + \frac{3}{7} \cdot \log_2 \frac{3}{7}\right) \approx 0.985 \]

• The total remaining entropy is
  \[ \text{Remainder}(3D) = \frac{5}{12} B\left(\frac{2}{5}\right) + \frac{7}{12} B\left(\frac{4}{7}\right) \approx 0.979 \]
Let’s test on Tom Hanks

Example

• There are 3 films with Tom Hanks. Bill went to none of them.
  ➢ Of the remaining 9 films, Bill went to 6.

• The remaining entropy in the yes-branch: $B(0) = 0$

• The remaining entropy in the no-branch is
  
  $$B(6/9) = B(1/3) = -\left(\frac{1}{3} \cdot \log_2 \frac{1}{3} + \frac{2}{3} \cdot \log_2 \frac{2}{3}\right) \approx 0.919$$

• The total remaining entropy is
  
  $$Remainder(TH) = \frac{3}{12} B(0) + \frac{9}{12} B(1/3) \approx 0.689$$

Much better!
Let’s test on film type

Example

Let’s look at the film types:

• Bill went to
  ➢ 1/3 comedies
  ➢ 2/4 horror films
  ➢ 2/3 sci-fi films
  ➢ 1/2 romantic films

\[
\text{Remainder}(Type) = \frac{3}{12} B(1/3) + \frac{4}{12} B(2/4) + \frac{3}{12} B(2/3) + \frac{2}{12} B(1/2) \approx \\
\frac{3}{12} \cdot 0.919 + \frac{4}{12} \cdot 1 + \frac{3}{12} \cdot 0.919 + \frac{2}{12} \cdot 1 = 0.9595
\]
# Remaining entropies

<table>
<thead>
<tr>
<th>Example</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remainder(Type)</td>
<td>0.9595</td>
</tr>
<tr>
<td>Remainder(Len)</td>
<td>0.905</td>
</tr>
<tr>
<td>Remainder(3D)</td>
<td>0.979</td>
</tr>
<tr>
<td>Remainder(New)</td>
<td>1</td>
</tr>
<tr>
<td>Remainder(Ani)</td>
<td>0.906</td>
</tr>
<tr>
<td>Remainder(Time)</td>
<td>0.979</td>
</tr>
<tr>
<td>Remainder(LSp)</td>
<td>0.906</td>
</tr>
<tr>
<td>Remainder(WDay)</td>
<td>0.333</td>
</tr>
<tr>
<td>Remainder(TH)</td>
<td>0.689</td>
</tr>
</tbody>
</table>
A typical case of overfitting!
Overfitting

- Occurs when we learn from specific attributes of the training data that are not relevant for the generalization.
- Occurs in all types of learners
- More likely when the hypothesis space is large
- More likely when there are many attributes
- Less likely with more training data
- $\chi^2$ pruning can help
- In Amanda’s case, the weekday attribute seems to have too many possible values compared to the training set.
## Remaining entropies

### Example

<table>
<thead>
<tr>
<th>Entropy Type</th>
<th>Entropy Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remainder(Type)</td>
<td>0.9595</td>
</tr>
<tr>
<td>Remainder(Len)</td>
<td>0.905</td>
</tr>
<tr>
<td>Remainder(3D)</td>
<td>0.979</td>
</tr>
<tr>
<td>Remainder(New)</td>
<td>1</td>
</tr>
<tr>
<td>Remainder(Ani)</td>
<td>0.906</td>
</tr>
<tr>
<td>Remainder(Time)</td>
<td>0.979</td>
</tr>
<tr>
<td>Remainder(LSp)</td>
<td>0.906</td>
</tr>
<tr>
<td>Remainder(WDay)</td>
<td>0.333</td>
</tr>
<tr>
<td>Remainder(TH)</td>
<td>0.689</td>
</tr>
</tbody>
</table>
Split on Tom Hanks

```
\begin{array}{c}
\text{TH} \\
y \quad n \\
\text{?} \quad ? \\
\hline
x_1 \quad x_3 \quad x_7 \\
x_2 \quad x_5 \quad x_6 \\
x_8 \quad x_{11} \quad x_{12} \\
x_4 \quad x_9 \quad x_{10}
\end{array}
```

Artificial Intelligence: Methods and applications
Ola Ringdahl, Umeå University
Split on Tom Hanks

[Diagram]

Artificial Intelligence: Methods and applications
Ola Ringdahl, Umeå University
Residual training set

Example

We are left with the following residual problem:

<table>
<thead>
<tr>
<th>Film</th>
<th>Type</th>
<th>Len</th>
<th>3D</th>
<th>New</th>
<th>Ani</th>
<th>Time</th>
<th>LSp</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>H</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>L</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_4$</td>
<td>H</td>
<td>M</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>$x_5$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_6$</td>
<td>R</td>
<td>L</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_8$</td>
<td>C</td>
<td>M</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>L</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>$x_9$</td>
<td>H</td>
<td>S</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>E</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>L</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>SF</td>
<td>S</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>L</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>H</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>
Remaining entropies

Example

Computing the new residual entropies, we get the following:

- Remainder(Type) = 0.750
- Remainder(Len) = 0.666
- Remainder(3D) = 0.846
- Remainder(New) = 0.846
- Remainder(Ani) = 0.666
- Remainder(Time) = 0.918
- Remainder(LSp) = 0.739

Let’s use Ani as the next attribute to split on.
Current graph

TH

y

No

n

?

x_2  x_5  x_6

x_8  x_{11}  x_{12}

x_4  x_9  x_{10}
Split on Animation

- **TH**
  - yes (No)
  - no (Ani)
  - y (x2, x8, x11)
  - n (x5, x6, x12)
  - < ? (x4, x9, x10)
Split on Animation

- **TH**: Yes -> **Ani**
  - **Ani**: Yes -> **Yes**
  - **Ani**: No -> **X_5**, **X_6**, **X_{12}

- **TH**: No -> **No**
  - **No**: Yes -> **Yes**
  - **No**: No -> **X_4**, **X_9**, **X_{10}**
Residual training set

Example

Again, we are left with a residual problem:

<table>
<thead>
<tr>
<th>Film \ Type</th>
<th>Len</th>
<th>3D</th>
<th>New</th>
<th>Ani</th>
<th>Time</th>
<th>LSp</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_4$</td>
<td>H</td>
<td>M</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>y</td>
</tr>
<tr>
<td>$x_5$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
</tr>
<tr>
<td>$x_6$</td>
<td>R</td>
<td>L</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>L</td>
<td>n</td>
</tr>
<tr>
<td>$x_9$</td>
<td>H</td>
<td>S</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>E</td>
<td>y</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>SF</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>L</td>
<td>n</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>H</td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>E</td>
<td>n</td>
</tr>
</tbody>
</table>
Example

This gives us the following residual entropies:

- Remainder(Type) = 0.906
- Remainder(Len) = 0.541
- Remainder(3D) = 1
- Remainder(New) = 0.918
- Remainder(Time) = 0.918
- Remainder(LSp) = 0.541

Let’s use Len as the next attribute to split on.
Current graph

- TH: y → No, n → Ani
- Ani: y → Yes, n → ?

Variables:
- $x_5$, $x_6$, $x_{12}$
- $x_4$, $x_9$, $x_{10}$
Splitting on Len

Diagram:
- TH
  - y: Ani
    - n: Len
      - L: Yes
        - x_5, x_6, x_{12}, x_{10}
      - S, M: No
    - n: Len
      - S, M: No
  - n: No

Yes, No, Ani, Len, S, M
And so on....
Decision tree learning: issues to deal with

- **Missing data**: What to do if all attribute values are not known for all training examples?
  - How should we classify an example with missing attributes?
  - How to compute the residual entropies when constructing the tree?

- **Multivalued attribute**: If an attribute has too many possible values, the information gain computation doesn’t work properly (as with the weekday attribute).
  - Use gain ratio
  - Convert to **Boolean tests**: WDay = Mon?
Decision tree learning: issues to deal with (2)

- **Numerical input** (infinite possible values)
  - Split into intervals (e.g. split *Time* in *Early* and *Late*)
  - There are different methods for finding good split points
  - Most expensive part of tree learning

- **Predicting a numerical output**
  - Use *regression trees*: in the leafs, keep linear functions on some of the attributes instead of single values
Chapter 21.1-21.3
REINFORCEMENT LEARNING
Reinforcement learning (RL)

- In reinforcement learning, the agent learns from feedback.
  - Good decisions are encouraged
  - Bad decisions are discouraged
- Can be passive or active
Passive RL

- The agent’s policy $\pi$ is fixed: always execute action $\pi(s)$ in state $s$.
- Goal: to learn the utility function $U^\pi(s)$
- Similar to policy iteration algorithm from last lecture (Ch. 17.3)
  - Main difference is that we do not know the transition model $P(s'|s,a)$ nor the reward function $R(s)$
Passive RL cont.

- The agent executes a number of trials using policy $\pi$ and perceive state and reward. e.g:

$$(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$$

$$(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,1)_{-0.04}$$

$$(2,1)_{-0.04} \rightarrow (3,1)_{-0.04} \rightarrow (4,1)_{-0.04} \rightarrow (4,2)_{-1}$$

- Key idea: use info about rewards to learn an expected utility $U^\pi(s)$

A policy $\pi$ (optimal for $R(s) = -0.04$)

Utilities of the states given policy $\pi$
Direct utility estimation

- Each trial provides a *sample* of the expected total reward (called *reward-to-go*)
  - Converges to the correct values in the limit of infinitely many trials
- Is actually a kind of supervised learning
- Misses that the utilities of states are not independent
  - Utility(s) = R(s) + Utility of its successor states, i.e. they obey the Bellman equation:
    \[
    U^\pi(s) = R(s) + \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s')
    \]
  - As a result, it converges rather slowly
Adaptive dynamic programming

To be able to use the Bellman equation, the transition model, i.e., the probabilities \( P(s'|s, \pi(s)) \) have to be known.

The idea of adaptive dynamic programming is to use the trials to learn the transition model.

Once this has been achieved, the agent only has to solve a system of linear equations to compute the utilities.
Active RL

In active reinforcement learning, the agent has the freedom to choose actions.

This means that it doesn’t simply have to learn the utility of a certain policy

\[ U^\pi(s) = R(s) + \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s') \]

Rather, it should learn the utility of the optimal policy:

\[ U(s) = R(s) + \max_a \sum_{s'} P(s' \mid s, a) U(s') \]

Can be calculated by using value iteration or policy iteration algorithms from last lecture
Active RL cont.

This would suggest an algorithm along the following lines:

1. Use random actions in a first exploration to build a first transition model $P(s' | s, a)$
2. Compute the optimal policy $\pi_P^*$ for $P$
3. Explore again, using $\pi_P^*$
4. Update $P$ accordingly
5. Repeat from 2

**Problem:** The agent may get stuck with a suboptimal policy and never explore the whole state set sufficiently to find the correct transition model.
Active RL: Exploration

- We have to make a tradeoff between Exploration and Exploitation.
- Easy solution: Take a random action now and then to explore more.
- Better solution: Use an exploration function $f(u,n)$ where greed (high values of $u$) is traded off for actions with low $n$ (has not been tried often):

$$U^+(s) = R(s) + \max_a \left( \sum_{s'} P(s' \mid s, a)U^+(s'), N(s, a) \right)$$

Compare with original:

$$U(s) = R(s) + \max_a \sum_{s'} P(s' \mid s, a)U(s')$$