CS486/686: Introduction to Artificial Intelligence
Lecture 6b - Decision Trees and Training Strategies

Jesse Hoey & Victor Zhong

School of Computer Science, University of Waterloo

January 27, 2025

Readings: Poole & Mackworth Chap. 7.3-7.3.1, 7.4-7.4.1

Example: User Discussion Board Behaviors

- Consider an application that predicts if a user will **read** or **skip** a discussion board article
- User action depends on the following **attributes** or **features** of articles:
  - the **author** of the article is **known** or **unknown** to the user
  - the **thread** is **new** or a **follow up**
  - the article's **length** is **long** or **short**
  - the user reads the article at **home** or at **work**
- Try to predict, based only on your **prior knowledge** of threaded discussion boards, what the user's action will be (**read** or **skip**) for the following examples:

example	**author**	**thread**	**length**	**where read**	**user's action**
t1	unknown	new	long	work	
t2	known	new	short	home	
t3	unknown	follow up	short	work	
t4	unknown	follow up	long	home	
t5	known	follow up	short	home	

Dataset: User Discussion Board Behaviors

After seeing this dataset, Now what is your prediction?

example	**author**	**thread**	**length**	**where read**	**user's action**
e1	known	new	long	home	skips
e2	unknown	new	short	work	reads
e3	unknown	follow up	long	work	skips
e4	known	follow up	long	home	skips
e5	known	new	short	home	reads
e6	known	follow up	long	work	skips
e7	unknown	follow up	short	work	skips
e8	unknown	new	short	work	reads
e9	known	follow up	long	home	skips
e10	known	new	long	work	skips
e11	unknown	follow up	short	home	skips
e12	known	new	long	work	skips
e13	known	follow up	short	home	reads
e14	known	new	short	work	reads
e15	known	new	short	home	reads
e16	known	follow up	short	work	reads
e17	known	new	short	home	reads
e18	unknown	new	short	work	reads
t1	unknown	new	long	work	?
t2	known	new	short	home	?

Example: User Discussion Board Behaviors

It appears the user mostly skips long articles (yellow lines) with two exceptions (green lines)

example	**author**	**thread**	**length**	**where read**	**user's action**
e1	known	new	long	home	skips
e2	unknown	new	short	work	reads
e3	unknown	follow up	long	work	skips
e4	known	follow up	long	home	skips
e5	known	new	short	home	reads
e6	known	follow up	long	work	skips
e7	unknown	follow up	short	work	skips
e8	unknown	new	short	work	reads
e9	known	follow up	long	home	skips
e10	known	new	long	work	skips
e11	unknown	follow up	short	home	skips
e12	known	new	long	work	skips
e13	known	follow up	short	home	reads
e14	known	new	short	work	reads
e15	known	new	short	home	reads
e16	known	follow up	short	work	reads
e17	known	new	short	home	reads
e18	unknown	new	short	work	reads
t1	unknown	new	long	work	?
t2	known	new	short	home	?

Learning Decision Trees

Simple, successful technique for supervised learning from discrete data

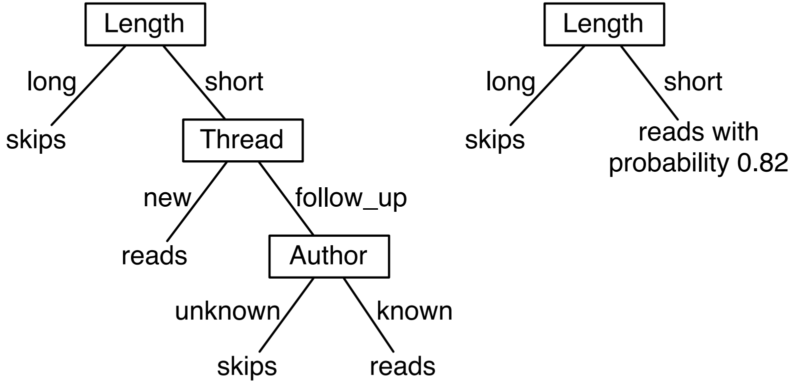
- **Representation** is a decision tree
- **Bias** is towards **simple** decision trees
- Search through the space of decision trees, from simple decision trees to more complex ones

Decision Trees

- **Nodes** are input attributes/features
- **Branches** are labeled with input feature value(s)
- **Leaves** are predictions for target features (**point estimates**)
- Can have many branches per node
- Branches can be labeled with **multiple feature values**

Example Decision Trees

Which decision tree is better for the discussion board example?



Learning a Decision Tree

- Incrementally **split** the training data
- **Recursively** solve sub-problems
- Hard part: **how to split** the data?
- **Criteria** for a good decision tree (bias):
  - small decision tree
  - good classification (low error on training data)
  - good generalisation (low error on test data)

Decision Tree Learning: Pseudocode

//X is input features, Y is output features,
//E is training examples
//output is a decision tree, which is either
// - a point estimate of Y, or
// - of the form $< X_i, T_1, \ldots, T_N >$ where
// X_i is an input feature and $T_1, \ldots, T_N$ are decision trees

procedure **DecisionTreeLearner**(X,Y,E)
  if stopping criteria is met **then**
    return pointEstimate(Y,E)
  else
    select feature $X_i \in X$
    for each value x_i of X_i **do**
      $E_i =$ all examples in E where $X_i = x_i$
      $T_i =$ DecisionTreeLearner($X \backslash \{X_i\}, Y, E_i$)
    end for
    return $< X_i, T_1, \ldots, T_N >$
end procedure

Decision Tree Classification/Inference: Pseudocode

//X is is input features, Y is output features,
//e is test example
//DT is a decision tree
//output is a prediction of Y for e

procedure **ClassifyExample**(e,X,Y,DT)
    $S \leftarrow DT$
    while S is internal node of the form $< X_i, T_1, \ldots, T_N >$ **do**
        $j \leftarrow X_i(e)$
        $S \leftarrow T_j$
    end while
    return S
end procedure

Remaining Issues

- **Stopping** criteria
- **Selection of features**
- **Point estimate** (final return value at leaf)
- Reducing number of branches (**partition** of domain for N-ary features)

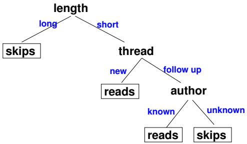
Stopping Criteria

- How do we decide to stop splitting?
- The stopping criteria is related to the final return value
- Depends on what we will need to do
- Possible stopping criteria:
  - **No more features**
  - Performance on training data is **good enough**

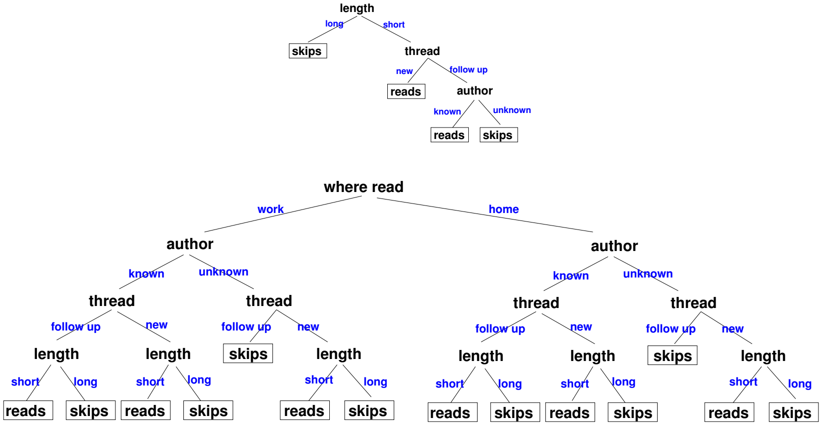
Feature Selection

- Ideal: choose sequence of features that result in **smallest tree**
- Actual: **myopically** split - as if only allowed one split, which feature would give best performance?
- **Heuristics** for best performing feature:
  - Most even split
  - Maximum information gain
  - GINI index
  - ... others domain dependent ...

Good Feature Selection



Bad Feature Selection



Information Theory

- a **bit** is a binary digit: 0 or 1
- n bits can distinguish 2^n items
- can do better by taking **probabilities** into account

Example:

Distinguish $\{a, b, c, d\}$ with
$P(a) = 0.5, P(b) = 0.25, P(c) = P(d) = 0.125$

If we encode

a:00 b:01 c:10: d:11

It uses on average **2 bits**

Information Theory

- a **bit** is a binary digit: 0 or 1
- n bits can distinguish 2^n items
- can do better by taking **probabilities** into account

Example:

Distinguish $\{a, b, c, d\}$ with
$P(a) = 0.5, P(b) = 0.25, P(c) = P(d) = 0.125$

If we encode

a:00 b:01 c:10: d:11

It uses on average **2 bits**

But if we encode

a:0 b:10 c:110 d:111

Information Theory

- a **bit** is a binary digit: 0 or 1
- n bits can distinguish 2^n items
- can do better by taking **probabilities** into account

Example:

Distinguish $\{a, b, c, d\}$ with
$P(a) = 0.5, P(b) = 0.25, P(c) = P(d) = 0.125$

If we encode

a:00 b:01 c:10: d:11

It uses on average **2 bits**

But if we encode

a:0 b:10 c:110 d:111

It uses on average $P(a) \times 1 + P(b) \times 2 + P(c) \times 3 + P(d) \times 3 =$ **1.75 bits**

Information Theory

- In general, need $-\log_2 \mathbf{P}(\mathbf{x})$ bits to encode $\mathbf{x}$
- Each symbol requires on average

$$-\mathbf{P}(\mathbf{x}) \log_2 \mathbf{P}(\mathbf{x}) \qquad \text{bits}$$

- To transmit an entire sequence distributed according to $P(x)$, we need **on average**

$$\sum_{\mathbf{x}} -\mathbf{P}(\mathbf{x}) \log_2 \mathbf{P}(\mathbf{x}) \qquad \text{bits}$$

  of information per symbol we wish to transmit
- **Information content** or **entropy** of the sequence

Information Gain

Given a set E of N training examples, if the number of examples with output feature $Y = y_i$ is N_i, then

$$P(Y = y_i) = P(y_i) = \frac{N_i}{N}$$

(the point estimate)

Total information content for the set E is (**assume** $\log \equiv \log_2$):

$$I(E) = -\sum_{y_i \in Y} P(y_i) \log P(y_i)$$

So, after splitting E up into E_1 and E_2 (size N_1, N_2) based on input attribute X_i, the information content

$$I(E_{split}) = \frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2)$$

and we want the X_i that maximises the **information gain**:

$$I(E) - I(E_{split})$$

Information Gain

Information gain is always non-negative

- Intuitively, information gain is the reduction in uncertainty about the output feature Y given the value of a certain input feature X
- Mathematically,

$$
\begin{aligned}
IG(E, X) &= I(E) - I(E_{split}) \\
&= -\sum_{y \in Y} P(y) \log P(y) - \left(\frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2) \right)
\end{aligned}
$$

Information Gain (cont.)

$$\frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2) = -\frac{N_1}{N} \left(\sum_{y \in Y} P(y|x=1) \log P(y|x=1) \right) - \frac{N_2}{N} \left(\sum_{y \in Y} P(y|x=2) \log P(y|x=2) \right)$$

Information Gain (cont.)

$$
\begin{aligned}
\frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2) &= -\frac{N_1}{N}\left(\sum_{y \in Y} P(y|x=1)\log P(y|x=1)\right) \\
&\quad -\frac{N_2}{N}\left(\sum_{y \in Y} P(y|x=2)\log P(y|x=2)\right) \\
&= -\sum_{y \in Y} P(x=1)P(y|x=1)\log P(y|x=1) \\
&\quad -\sum_{y \in Y} P(x=2)P(y|x=2)\log P(y|x=2)
\end{aligned}
$$

Information Gain (cont.)

$$
\begin{aligned}
\frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2) &= -\frac{N_1}{N}\left(\sum_{y \in Y} P(y|x=1) \log P(y|x=1)\right) \\
&\quad - \frac{N_2}{N}\left(\sum_{y \in Y} P(y|x=2) \log P(y|x=2)\right) \\
&= -\sum_{y \in Y} P(x=1) P(y|x=1) \log P(y|x=1) \\
&\quad - \sum_{y \in Y} P(x=2) P(y|x=2) \log P(y|x=2) \\
&= -\sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)}
\end{aligned}
$$

Information Gain (cont.)

Jensen's inequality: for a convex function $f(x)$, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$
$-\log(\cdot)$ is a convex function, so

$$\begin{aligned} IG(E, X) &= I(E) - I(E_{split}) \\ &= -\sum_{y \in Y} P(y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \end{aligned}$$

Information Gain (cont.)

Jensen's inequality: for a convex function $f(x)$, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$
$-\log(\cdot)$ is a convex function, so

$$
\begin{aligned}
IG(E, X) &= I(E) - I(E_{split}) \\
&= -\sum_{y \in Y} P(y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \\
&= -\sum_{x \in X, y \in Y} P(x, y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)}
\end{aligned}
$$

Information Gain (cont.)

Jensen's inequality: for a convex function $f(x)$, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$
$-\log(\cdot)$ is a convex function, so

$$
\begin{aligned}
IG(E, X) &= I(E) - I(E_{split}) \\
&= -\sum_{y \in Y} P(y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \\
&= -\sum_{x \in X, y \in Y} P(x, y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \\
&= -\sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x)P(y)}{P(x, y)}
\end{aligned}
$$

Information Gain (cont.)

Jensen's inequality: for a convex function $f(x)$, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$
$-\log(\cdot)$ is a convex function, so

$$
\begin{aligned}
IG(E, X) &= I(E) - I(E_{split}) \\
&= -\sum_{y \in Y} P(y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \\
&= -\sum_{x \in X, y \in Y} P(x, y) \log P(y) + \sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)} \\
&= -\sum_{x \in X, y \in Y} P(x, y) \log \frac{P(x)P(y)}{P(x, y)} \\
&\geq -\log \left(\sum_{x \in X, y \in Y} P(x, y) \frac{P(x)P(y)}{P(x, y)} \right) = 0
\end{aligned}
$$

Example: User Discussion Board Behaviors

Build a decision tree for this dataset, using information gain to split, then make predictions

example	**author**	**thread**	**length**	**where read**	**user's action**
e1	known	new	long	home	skips
e2	unknown	new	short	work	reads
e3	unknown	follow up	long	work	skips
e4	known	follow up	long	home	skips
e5	known	new	short	home	reads
e6	known	follow up	long	work	skips
e7	unknown	follow up	short	work	skips
e8	unknown	new	short	work	reads
e9	known	follow up	long	home	skips
e10	known	new	long	work	skips
e11	unknown	follow up	short	home	skips
e12	known	new	long	work	skips
e13	known	follow up	short	home	reads
e14	known	new	short	work	reads
e15	known	new	short	home	reads
e16	known	follow up	short	work	reads
e17	known	new	short	home	reads
e18	unknown	new	short	work	reads

(see lect06-handout-dtexample.pdf)

Final Return Value

- **Point estimate** of Y (output features) over all examples
- Point estimate is just a **prediction of target** features
  - mean value,
  - median value,
  - most likely classification,
  - etc.

  e.g.

$$P(Y = y_i) = \frac{N_i}{N}$$

  where
  - N_i is the number of training samples at the leaf with $Y = Y_i$
  - N is the total number of training samples at the leaf.

Using a Priority Queue to Learn the DT

- The "vanilla" version we saw grows all branches for a node
- But there might be some branches that are more worthwhile to expand
- Idea: sort the leaves using a **priority queue** ranked by how much information can be gained with the best feature at that leaf
- Always expand the leaf at the top of the queue

Priority Queue (PQ) Decision Tree: Pseudocode V1

procedure **DecisionTreeLearner**(X,Y,E)
Start *PQ* **with a single node** (index 0) with
- whole data set $E_0 \equiv E$,
- the point estimate for E_0, y_0,
- the best next feature to split E_0 on, X_0 and
- the amount of information gain ΔI_0 if E_0 split on X_0.
- **add node 0 to PQ**

Repeat until a stopping criteria is reached:
- find leaf (index i) with highest information gain (**head of PQ**)
  $\rightarrow$ leaf i is the next split to do.
- Split the data at that leaf (E_i) according to the Best-Feature X_i
  $\rightarrow$ two datasets E_i+ and E_i-
- Add 2 children to node i, one with E_i+ and one with E_i-
- for each new child: compute and store in the child nodes:
  - point estimate,
  - best next feature to split on (of all the remaining features), and
  - information gain for that split
- **add child nodes to PQ by information gain**

Decision tree learning: pseudocode V2

procedure DecisionTreeLearner(X,Y,E)
$DT = pointEstimate(Y, E)$ = initial decision tree
$\{X', \Delta I\} \leftarrow$ best feature and Information Gain value for E
$PQ \leftarrow \{DT, E, X', \Delta I\}$ = priority queue of leaves ranked by ΔI
while stopping criteria is not met **do:**
    $\{S_\ell, E_\ell, X_\ell, \Delta I_\ell\} \leftarrow$ leaf at the head of PQ
    for each value x_i of X_ℓ **do**
        E_i = all examples in E_ℓ where $X_\ell = x_i$
        $\{X_j, \Delta I_j\}$ = best feature and value for E_i
        $T_i \leftarrow pointEstimate(Y, E_i)$
        insert $\{T_i, E_i, X_j, \Delta I_j\}$ into PQ according to ΔI_j
    end for
    $S_\ell \leftarrow < X_\ell, T_1, \ldots, T_N >$
end while
return DT
end procedure

Overfitting

Sometimes the decision tree is **too good** at classifying the training data, and will **not generalize** very well

This often occurs when there is **not much data**

Attributes: bad weather (W), I burnt my toast (T), my train is late (L)

Training data:

W , T , L ;

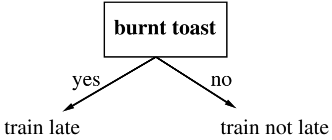
true, true, true;

false, false, false;

false, false, false;

true, false, false;

best decision tree (info gain):



Decision tree predicts the train based on burnt toast

Overfitting

Sometimes the decision tree is **too good** at classifying the training data, and will **not generalize** very well

This often occurs when there is **not much data**

Attributes: bad weather (W), I burnt my toast (T), my train is late (L)

Training data:

W , T , L ;

true, true, true;

false, false, false;

false, false, false;

true, false, false;

false, true, false;

true, false, true;

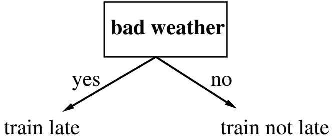
false,false, false;

true, true, true;

true, false, true;

false, true, false;

best decision tree (info gain):

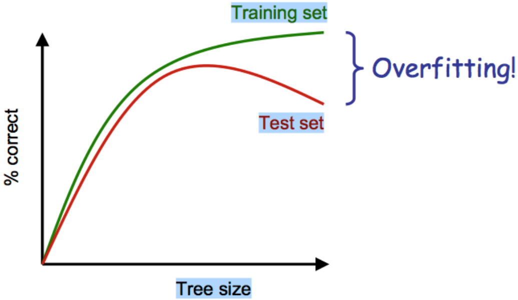


Decision tree predicts the train based on the weather

Overfitting

Some methods to avoid overfitting

- **Regularization**: e.g. Prefer small decision trees over big ones, so add a 'complexity' penalty to the stopping criteria - stop early
- **Pseudocounts**: add some data based on prior knowledge
- **Cross validation**



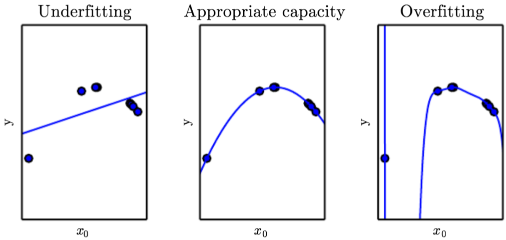
Overfitting

Test set errors caused by:

- **Bias**: the error due to the algorithm finding an imperfect model
  - **representation bias**: model is too simple
  - **search bias**: not enough search
- **Variance**: the error due to lack of data
- **Noise**: the error due to the data depending on features not modeled or because the process generating the data is inherently stochastic.
- **Bias-variance trade-off**:
  - Complicated model, not enough data (low bias, high variance)
  - Simple model, lots of data (high bias, low variance)
- see handout `lect06-handout-biasvariance.pdf`

Overfitting

- **Capacity** of a model is its ability to fit a wide variety of functions
- Capacity is like the inverse of bias - a high capacity model has low bias and vice-versa



Cross Validation

Cross Validation

- Split **training** data into a training and a **validation** set
- Use the validation set as a **“pretend” test set**
- **Optimise the decision maker** to perform well on the validation set, not the training set
- Can do this **multiple times** with different validation sets

Next

- Uncertainty (Poole & Mackworth Chapter 9)