Decision Trees and Ensemble Methods

Mingfei Sun

Foundations of Machine Learning
The University of Manchester

Outline

Nodes in a decision tree

- A decision tree consists of:
  1. a **root node** (or starting node),
  2. **interior nodes**
  3. and **leaf nodes** (or terminating nodes).
- Each of the non-leaf nodes (root and interior) in the tree specifies a test to be carried out on one of the query's descriptive features.
- Each of the leaf nodes specifies a predicted classification for the query.

An email spam prediction dataset

Table: An email spam prediction dataset.

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Two decision trees and a query instance

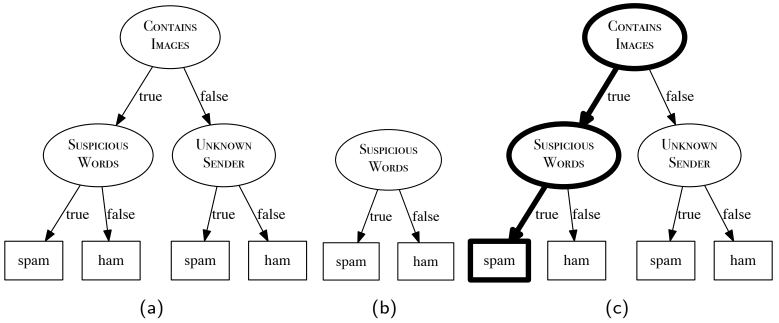


Figure: (a) and (b) show two decision trees that are consistent with the instances in the spam dataset. (c) shows the path taken through the tree shown in (a) to make a prediction for the query instance: SUSPICIOUS WORDS = *'true'*, UNKNOWN SENDER = *'true'*, CONTAINS IMAGES = *'true'*.

Which tree to use?

- Both of these trees will return identical predictions for all the examples in the dataset.
- So, which tree should we use?
- Occam's Razor: *"Among competing hypotheses, the one with the fewest assumptions should be selected"*.

Informative features

- A decision tree is a machine learning algorithm that tries to build predictive models **using the most informative features**.
- An informative feature is a feature whose values split the instances in the dataset into **homogeneous sets** with respect to the target value.
- So the problem is to figure out which features are the most informative ones to ask questions about.

How do we create shallow trees?

- The tree that tests SUSPICIOUS WORDS at the root is very shallow because the SUSPICIOUS WORDS feature perfectly splits the data into pure groups of *'spam'* and *'ham'*.
- Descriptive features that split the dataset into pure sets with respect to the target feature provide information about the target feature.
- So we can make shallow trees by testing the informative features early on in the tree.
- All we need to do that is a computational metric of the purity of a set: **entropy**

Entropy

- Claude Shannon's entropy model defines a computational measure of the impurity of the elements of a set.
- An easy way to understand the entropy of a set is to think in terms of the uncertainty associated with guessing the result if you were to make a random selection from the set.
- Entropy is related to the probability of an outcome.
  - High probability $\rightarrow$ Low entropy
  - Low probability $\rightarrow$ High entropy

Definition of entropy

The entropy $H_b(X)$ in base b of a discrete random variable X is defined as

$$H_b(X) = -\sum_{x \in \mathcal{X}} P(X = x) \log_b P(X = x),$$

where we use the convention that $0 \times \log_b(0) = 0$. For $b = 2$ we usually write $H(X)$ instead $H_2(X)$, and write $\log(q)$ instead $\log_2(q)$.

- The entropy only depends on the pmf of the random variable X, i.e., for two different random variables X and $\hat{X}$ with the same pmf, their entropies are the same.
- $H(X) = -E[\log P(X)]$ where $P(\cdot)$ is the pmf of X.
- The choice of base 2 for the logarithm is common (due to computers using two states) but not essential.
- The unit of entropy in base 2 is called a `bit`, in base e `nat`, in base 256 a `byte`.

Entropy examples: poker cards

- What is the entropy of a set of 52 different playing cards?

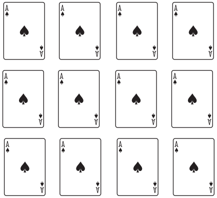
$$
\begin{aligned}
H(\text{card}) &= -\sum_{i=1}^{52} P(\text{card} = i) \cdot \log_2 P(\text{card} = i) \\
&= -\sum_{i=1}^{52} \frac{1}{52} \cdot \log_2 \frac{1}{52} \\
&\approx 5.700 \text{ bits}
\end{aligned}
$$

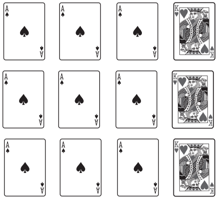
Entropy examples: poker cards

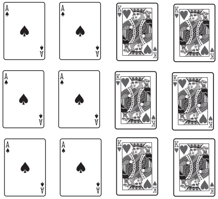
- What is the entropy of a set of 52 playing cards if we only distinguish between the cards based on their suit $\{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}$?

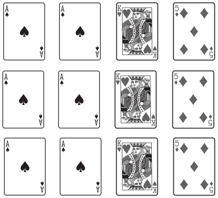
$$\begin{aligned}
H(\text{suit}) &= -\sum_{l \in \{\heartsuit,\clubsuit,\diamondsuit,\spadesuit\}} P(\text{suit} = l) \cdot \log_2 P(\text{suit} = l) \\
&= -\big[P(\heartsuit) \cdot \log_2 P(\heartsuit) + P(\clubsuit) \cdot \log_2 P(\clubsuit) + \\
&\quad P(\diamondsuit) \cdot \log_2 P(\diamondsuit) + P(\spadesuit) \cdot \log_2 P(\spadesuit)\big] \\
&= -4 \cdot \frac{13}{52} \cdot \log_2 \frac{13}{52} \\
&= 2 \text{ bits}
\end{aligned}$$

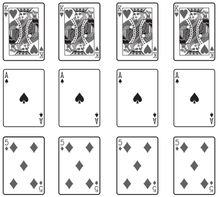
Different entropies


(a) $H(\text{card}) = 0.00$


(b) $H(\text{card}) = 0.81$


(c) $H(\text{card}) = 1.00$


(d) $H(\text{card}) = 1.50$


(e) $H(\text{card}) = 1.58$

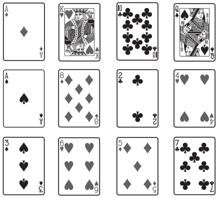

(f) $H(\text{card}) = 3.58$

Figure: The entropy of different sets of playing cards measured in bits.

Examples of entropy

- If $\mathcal{X} = \{H, T\}$ and $P(X = H) = p$, then
$$H(X) = -p\log(p) - (1-p)\log(1-p).$$
If $p \in \{0, 1\}$, then $H(X) = 0$. Differentiating in p shows that the entropy as a function of p increases on $(0, 0.5)$ and decreasing on $(0.5, 1)$. Hence, the entropy is maximised if $p = 0.5$ with $H(X) = \log(2) = 1$ bits.
- If X is a 2-dim vector in the form (X_1, X_2) with $X_i \in \mathcal{X}_i$ for $i = 1, 2$, then
$$H(X) = H(X_1, X_2) = -\sum_{x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2} P(x_1, x_2) \log P(x_1, x_2).$$
If additionally, X_1 and X_2 are independent, i.e., $P(x_1, x_2) = P(x_1)P(x_2)$, then
$$H(X) = H(X_1) + H(X_2).$$
If X_1 and X_2 are independent and identically distributed (i.i.d.), then
$$H(X) = 2H(X_1) = 2H(X_2).$$

Entropy of a message

Table: The relationship between the entropy of a message and the set it was selected from.

Entropy of a Message	**Properties of the Message Set**
High	A large set of equally likely messages.
Medium	A large set of messages, some more likely than others.
Medium	A small set of equally likely messages.
Low	A small set of messages with one very likely message.

The spam dataset again

Table: An email spam prediction dataset.

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Partitions in the spam dataset

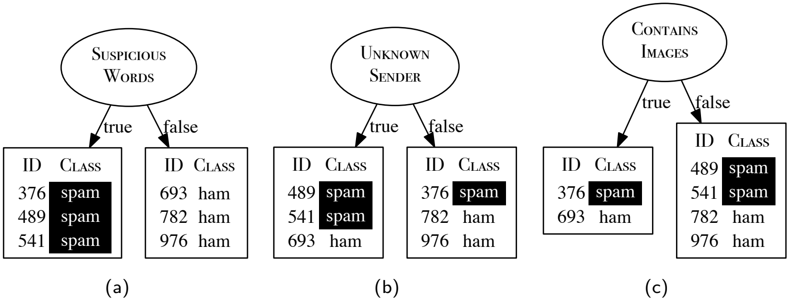


Figure: How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset table.

Pure subsets

- Our intuition is that the ideal discriminative feature will partition the data into **pure** subsets where all the instances in each subset have the same classification.
  - SUSPICIOUS WORDS perfect split.
  - UNKNOWN SENDER mixture but some information (when *'true'* most instances are *'spam'*).
  - CONTAINS IMAGES no information.
- One way to implement this idea is to use a metric called **information gain**.

Information Gain

The information gain of a descriptive feature can be understood as a measure of the reduction in the overall entropy of a prediction task by testing on that feature.

Computing the information gain

1. Compute the entropy of the original dataset with respect to the target feature. This gives us a measure of how much information is required in order to organize the dataset into pure sets.
2. For each descriptive feature, create the sets that result by partitioning the instances in the dataset using their feature values, and then sum the entropy scores of each of these sets. This gives a measure of the information that remains required to organize the instances into pure sets after we have split them using the descriptive feature.
3. Subtract the remaining entropy value (computed in step 2) from the original entropy value (computed in step 1) to give the information gain.

Computing the information gain

Computing information gain involves the following three equations:

$$H(X, \mathcal{D}) = - \sum_{l \in \text{levels}(X)} P(X = l) \cdot \log_2 P(X = l)$$

$$rem(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \cdot \underbrace{H(X, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

$$IG(d, \mathcal{D}) = H(X, \mathcal{D}) - rem(d, \mathcal{D})$$

Example with the spam dataset

- As an example we will calculate the information gain for each of the descriptive features in the spam email dataset.

Step 1: Entropy for the target feature

- Calculate the **entropy** for the target feature in the dataset.

$$H(X, \mathcal{D}) = -\sum_{l \in levels(X)} P(l) \cdot \log_2 P(l)$$

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

$$\begin{aligned}
H(X, \mathcal{D}) &= -\sum_{l \in \{\,'spam', 'ham'\}} P(l) \cdot \log_2 P(l) \\
&= -P(X = 'spam') \cdot \log_2 P(X = 'spam') - P(X = 'ham') \cdot \log_2 P(X = 'ham') \\
&= -\frac{3}{6} \cdot \log_2 \frac{3}{6} - \frac{3}{6} \cdot \log_2 \frac{3}{6} \\
&= 1 \; bit
\end{aligned}$$

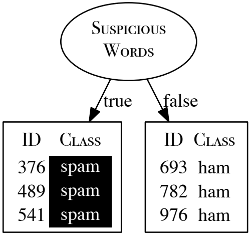
Step 2: Remainder for the SUSPICIOUS W. feature

- Calculate the **remainder** for the SUSPICIOUS WORDS feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \cdot \underbrace{H(X, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Remainder for the SUSPICIOUS W. feature (cont.)



(a)

Figure: Partitions for the SUSPICIOUS W. feature

$$
\begin{aligned}
rem\,(\text{WORDS}, \mathcal{D}) &= \left(\frac{|\mathcal{D}_{\text{WORDS}=T}|}{|\mathcal{D}|} \cdot H\,(X, \mathcal{D}_{\text{WORDS}=T})\right) + \left(\frac{|\mathcal{D}_{\text{WORDS}=F}|}{|\mathcal{D}|} \cdot H\,(X, \mathcal{D}_{\text{WORDS}=F})\right) \\
&= -\frac{3}{6} \cdot \sum_{l \in \{'spam', 'ham'\}} P(l) \cdot \log_2 P(l) - \frac{3}{6} \cdot \sum_{l \in \{'spam', 'ham'\}} P(l) \cdot \log_2 P(l) \\
&= -\frac{3}{6} \cdot \left(\frac{3}{3} \cdot \log_2 \frac{3}{3} + \frac{0}{3} \cdot \log_2 \frac{0}{3}\right) - \frac{3}{6} \cdot \left(\frac{0}{3} \cdot \log_2 \frac{0}{3} + \frac{3}{3} \cdot \log_2 \frac{3}{3}\right) \\
&= 0 \ bits
\end{aligned}
$$

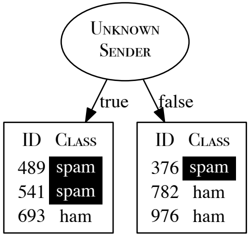
Step 2: Remainder for the UNKNOWN SENDER feature

- Calculate the **remainder** for the UNKNOWN SENDER feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \cdot \underbrace{H(X, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Remainder for the Unknown Sender feature (cont.)



(a)

Figure: Partitions for the Unknown Sender feature

$$
\begin{aligned}
rem\,(\text{SENDER}, \mathcal{D}) &= \left(\frac{|\mathcal{D}_{\text{SENDER}=T}|}{|\mathcal{D}|} \cdot H\,(X, \mathcal{D}_{\text{SENDER}=T})\right) + \left(\frac{|\mathcal{D}_{\text{SENDER}=F}|}{|\mathcal{D}|} \cdot H\,(X, \mathcal{D}_{\text{SENDER}=F})\right) \\
&= -\frac{3}{6} \cdot \sum_{l \in \{\,'spam', 'ham'\}} P(l) \cdot \log_2 P(l) - \frac{3}{6} \cdot \sum_{l \in \{\,'spam', 'ham'\}} P(l) \cdot \log_2 P(l) \\
&= -\frac{3}{6} \cdot \left(\frac{2}{3} \cdot \log_2 \frac{2}{3} + \frac{1}{3} \cdot \log_2 \frac{1}{3}\right) - \frac{3}{6} \cdot \left(\frac{1}{3} \cdot \log_2 \frac{1}{3} + \frac{2}{3} \cdot \log_2 \frac{2}{3}\right) \\
&= 0.9183\ bits
\end{aligned}
$$

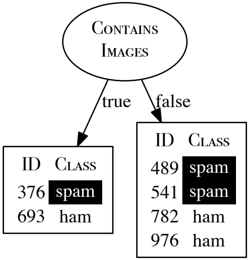
Step 2: Remainder for the CONTAINS IMAGES feature

- Calculate the **remainder** for the CONTAINS IMAGES feature in the dataset.

$$rem\left(d, \mathcal{D}\right) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \cdot \underbrace{H\left(X, \mathcal{D}_{d=l}\right)}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Step 2: Remainder for the CONTAINS IMAGES feature (cont.)



(a)

Figure: Partitions for the CONTAINS IMAGES feature

$$
\begin{aligned}
rem\left(\text{IMAGES}, \mathcal{D}\right) &= \left(\frac{|\mathcal{D}_{\text{IMAGES}=T}|}{|\mathcal{D}|} \cdot H\left(X, \mathcal{D}_{\text{IMAGES}=T}\right)\right) + \left(\frac{|\mathcal{D}_{\text{IMAGES}=F}|}{|\mathcal{D}|} \cdot H\left(X, \mathcal{D}_{\text{IMAGES}=F}\right)\right) \\
&= -\frac{2}{6} \cdot \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(l) \cdot \log_2 P(l) - \frac{4}{6} \cdot \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(l) \cdot \log_2 P(l) \\
&= -\frac{2}{6} \cdot \left(\frac{1}{2} \cdot \log_2 \frac{1}{2} + \frac{1}{2} \cdot \log_2 \frac{1}{2}\right) - \frac{4}{6} \cdot \left(\frac{2}{4} \cdot \log_2 \frac{2}{4} + \frac{2}{4} \cdot \log_2 \frac{2}{4}\right) \\
&= 1 \ bit
\end{aligned}
$$

Step 3: Compute the Information Gain

- Calculate the **information gain** for the three descriptive feature in the dataset.

$$IG\left(d,\mathcal{D}\right)=H\left(X,\mathcal{D}\right)-rem\left(d,\mathcal{D}\right)$$

$$\begin{aligned}
IG\left(\textrm{Suspicious Words},\mathcal{D}\right) &= H\left(\textrm{Class},\mathcal{D}\right)-rem\left(\textrm{Suspicious Words},\mathcal{D}\right)\\
&= 1-0=1\ bit\\
IG\left(\textrm{Unknown Sender},\mathcal{D}\right) &= H\left(\textrm{Class},\mathcal{D}\right)-rem\left(\textrm{Unknown Sender},\mathcal{D}\right)\\
&= 1-0.9183=0.0817\ bits\\
IG\left(\textrm{Contains Images},\mathcal{D}\right) &= H\left(\textrm{Class},\mathcal{D}\right)-rem\left(\textrm{Contains Images},\mathcal{D}\right)\\
&= 1-1=0\ bits
\end{aligned}$$

- The results of these calculations match our intuitions.

Outline

Turn continuous features into boolean features

- The easiest way to handle continuous valued descriptive features is to turn them into boolean features by defining a threshold.
- The threshold is used to partition the instances based on their value of the continuous descriptive feature.
- How do we set the threshold?

Sorting the instances

1. The instances in the dataset are sorted according to the continuous feature values.
2. The adjacent instances in the ordering that have different classifications are then selected as possible threshold points.
3. The optimal threshold is found by computing the information gain for each of these classification transition boundaries and selecting the boundary with the highest information gain as the threshold.

Treat the feature as a categorial feature

- Once a threshold has been set the dynamically created new boolean feature can compete with the other categorical features for selection as the splitting feature at that node.
- This process can be repeated at each node as the tree grows.

Vegetation classification dataset

(a) chaparral veg.

(b) riparian veg.

(c) conifer veg.

Table: Dataset for predicting the vegetation in an area with a continuous ELEVATION feature (measured in feet).

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	3 900	chapparal
2	true	moderate	300	riparian
3	true	steep	1 500	riparian
4	false	steep	1 200	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer
7	true	steep	3 000	chapparal

Sorted instances

Table: Dataset for predicting the vegetation in an area sorted by the continuous ELEVATION feature.

ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

- Calculate the **entropy** for the target feature in the dataset.

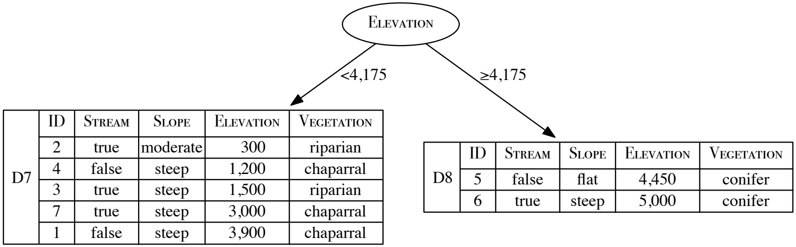
$$H(X, \mathcal{D}) = - \sum_{l \in \text{VEGETATION}(X)} P(l) \cdot \log_2 P(l) \approx 1.5567 \text{bits}$$

Thresholds and partitions

Table: Partition sets (Part.), entropy, remainder (Rem.), and information gain (Info. Gain) for the candidate ELEVATION thresholds: $\geq$750, $\geq$1 350, $\geq$2 250 and $\geq$4 175.

Split by Threshold	Part.	Instances	Partition Entropy	Rem.	Info. Gain
$\geq$750	$\mathcal{D}_1$	$\mathbf{d}_2$	0.0	1.2507	0.3060
	$\mathcal{D}_2$	$\mathbf{d}_4, \mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.4591		
$\geq$1 350	$\mathcal{D}_3$	$\mathbf{d}_2, \mathbf{d}_4$	1.0	1.3728	0.1839
	$\mathcal{D}_4$	$\mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.5219		
$\geq$2 250	$\mathcal{D}_5$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_3$	0.9183	0.9650	0.5917
	$\mathcal{D}_6$	$\mathbf{d}_7, \mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_6$	1.0		
$\geq$4 175	$\mathcal{D}_7$	$\mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_3, \mathbf{d}_7, \mathbf{d}_1$	0.9710	0.6935	0.8631
	$\mathcal{D}_8$	$\mathbf{d}_5, \mathbf{d}_6$	0.0		

Using Elevation as root node



D7

ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1,200	chaparral
3	true	steep	1,500	riparian
7	true	steep	3,000	chaparral
1	false	steep	3,900	chaparral

D8

ID	STREAM	SLOPE	ELEVATION	VEGETATION
5	false	flat	4,450	conifer
6	true	steep	5,000	conifer

Figure: The vegetation classification decision tree after the dataset has been split using ELEVATION $\geq$ 4 175.

Outline

The ID3 Algorithm

- ID3 Algorithm (Iterative Dichotomizer 3)
- Attempts to create the shallowest tree that is consistent with the data that it is given.
- The ID3 algorithm builds the tree in a recursive, depth-first manner, beginning at the root node and working down to the leaf nodes.

How the different nodes are treated?

1. The algorithm begins by choosing the best descriptive feature to test (i.e., the best question to ask first) using **information gain**.
2. A root node is then added to the tree and labelled with the selected test feature.
3. The training dataset is then partitioned using the test.
4. For each partition a branch is grown from the node.
5. The process is then repeated for each of these branches using the relevant partition of the training set in place of the full training set and with the selected test feature excluded from further testing.

The CART Algorithm

- CART (Classification and Regression Trees).
- Constructs binary trees.
- At each node it looks for the feature and threshold in that feature that provides the largest information gain at that node.
- This is the one implemented in scikit-learn.

Greedy algorithms

- Both ID3 and CART are examples of *greedy* algorithms.
- They find the “best tree” by greedily finding the best partitions at each level of the tree.
- They don’t check whether the best splits done at the higher levels of the tree will lead to the lowest possible impurity several levels down.

Outline

Regression trees

- Regression trees are constructed so as to reduce the variance in the set of training examples at each of the leaf nodes in the tree
- We can do this by adapting the ID3 algorithm to use a measure of variance rather than a measure of classification impurity (entropy) when selecting the best attribute
- The impurity (variance) at a node can be calculated using the following equation:

$$var\left(t, \mathcal{D}\right) = \frac{\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2}{n-1}$$

- We select the feature that minimizes the weighted variance across the resulting partitions:

$$\mathbf{d}[best] = \operatorname*{arg\,min}_{d \in \mathbf{d}} \sum_{l \in levels(d)} \frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|} \times var(t, \mathcal{D}_{d=l})$$

A dataset listing the number of bike rentals per day

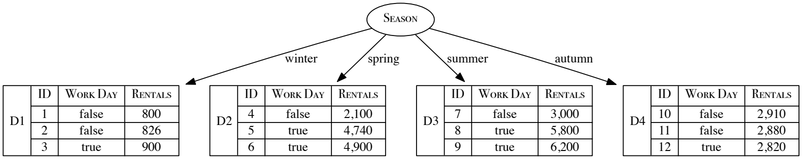
ID	SEASON	WORK DAY	RENTALS
1	winter	false	800
2	winter	false	826
3	winter	true	900
4	spring	false	2 100
5	spring	true	4 740
6	spring	true	4 900
7	summer	false	3 000
8	summer	true	5 800
9	summer	true	6 200
10	autumn	false	2 910
11	autumn	false	2 880
12	autumn	true	2 820

Weighted variance according to the partitions

Table: Partitioning of the dataset for bike rentals based on SEASON and WORK DAY features and the computation of the weighted variance for each partitioning.

Split by Feature	Level	Part.	Instances	$\frac{\lvert\mathcal{D}_{d=l}\rvert}{\lvert\mathcal{D}\rvert}$	$var(t, \mathcal{D})$	Weighted Variance
SEASON	*'winter'*	$\mathcal{D}_1$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.25	2 692	$1\,379\,331\frac{1}{3}$
	'spring'	$\mathcal{D}_2$	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.25	$2\,472\,533\frac{1}{3}$	
	'summer'	$\mathcal{D}_3$	$\mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9$	0.25	3 040 000	
	'autumn'	$\mathcal{D}_4$	$\mathbf{d}_{10}, \mathbf{d}_{11}, \mathbf{d}_{12}$	0.25	2 100	
WORK DAY	*'true'*	$\mathcal{D}_5$	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{12}$	0.50	$4\,026\,346\frac{1}{3}$	$2\,551\,813\frac{1}{3}$
	'false'	$\mathcal{D}_6$	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_7, \mathbf{d}_{10}, \mathbf{d}_{11}$	0.50	1 077 280	

Resulting decision tree



D1

ID	WORK DAY	RENTALS
1	false	800
2	false	826
3	true	900

D2

ID	WORK DAY	RENTALS
4	false	2,100
5	true	4,740
6	true	4,900

D3

ID	WORK DAY	RENTALS
7	false	3,000
8	true	5,800
9	true	6,200

D4

ID	WORK DAY	RENTALS
10	false	2,910
11	false	2,880
12	true	2,820

Figure: The decision tree resulting from splitting the data using the feature SEASON.

Final decision tree

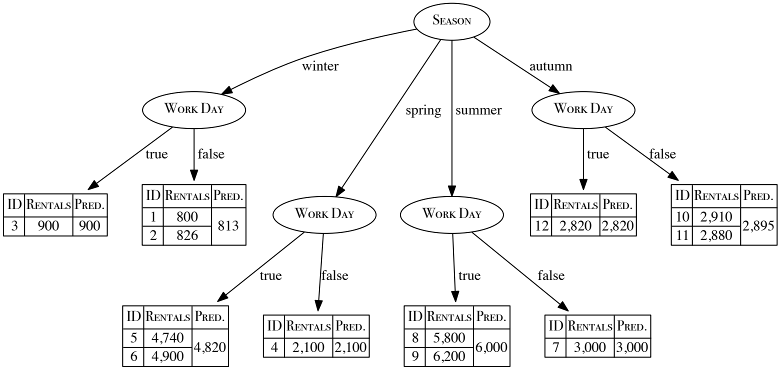


Figure: The final decision tree induced from the dataset. To illustrate how the tree generates predictions, this tree lists the instances that ended up at each leaf node and the prediction (PRED.) made by each leaf node.

Outline

Alternative impurity measures: Gini index

- Another commonly used measure of impurity is the **Gini index**:

$$\text{Gini}\,(X, \mathcal{D}) = 1 - \sum_{l \in \text{levels}(X)} P(l)^2$$

- The Gini index can be thought of as calculating how often you would misclassify an instance in the dataset if you classified it based on the probability distribution of the target feature in the dataset.
- Information gain can be calculated using the Gini index by replacing the entropy measure with the Gini index.

Alternative impurity measures: Information gain ratio

- Entropy based information gain has a preference for features with many values.
- One way of addressing this issue is to use **information gain ratio** which is computed by dividing the information gain of a feature by the amount of information used to determine the value of the feature:

$$GR\left(d,\mathcal{D}\right) = \frac{IG\left(d,\mathcal{D}\right)}{-\sum\limits_{l \in \text{levels}(X)} P(l) \cdot log_2 P(l)}$$

Overfitting and Tree Pruning

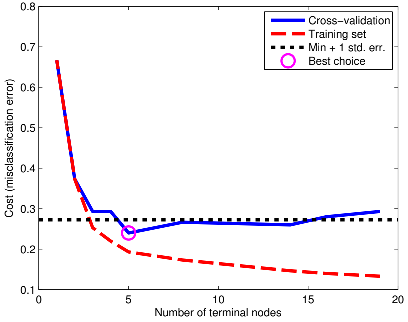
The likelihood of over-fitting occurring increases as a tree gets deeper because the resulting classifications are based on smaller and smaller subsets as the dataset is partitioned after each feature test in the path.

Pruning strategies

- **Pre-pruning**: stop the recursive partitioning early. Pre-pruning is also known as **forward pruning**.
- We can stop growing the tree if the decrease in the error is not sufficient to justify the extra complexity of adding an extra subtree.
- **Post-pruning**: allow the algorithm to grow the tree as much as it likes and then prune the tree of the branches that cause over-fitting.

Common **Post**-pruning Approach

- Using the validation set evaluate the prediction accuracy achieved by both the fully grown tree and the pruned copy of the tree.
- If the pruned copy of the tree performs no worse than the fully grown tree the node is a candidate for pruning.



Outline

Model ensembles

- Rather than creating a single model, we can generate a set of models and then make predictions by aggregating the outputs of these models.
- A prediction model that is composed of a set of models is called a **model ensemble**.
- In order for this approach to work the models that are in the ensemble must be different from each other.

Approaches

There are two standard approaches to creating ensembles:

1. **boosting**
2. **bagging**.

Boosting: How does it work?

- Boosting works by iteratively creating models and adding them to the ensemble.
- The iteration stops when a predefined number of models have been added.
- When we use **boosting** each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified.
- This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset**

Boosting: Weighted Dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $\frac{1}{n}$ where n is the number of instances in the dataset.
- After each model is added to the ensemble it is tested on the **training data** and the weights of the instances the model gets correct are decreased and the weights of the instances the model gets incorrect are increased.
- These weights are used as a distribution over which the dataset is sampled to created a **replicated training set**, where the replication of an instance is proportional to its weight.

Predictions

- Once the set of models have been created the ensemble makes **predictions** using a weighted aggregate of the predictions made by the individual models.
- The weights used in this aggregation are simply the confidence factors associated with each model.

Bagging: How does it work?

- When we use **bagging** (or **bootstrap aggregating**) each model in the ensemble is trained on a random sample of the dataset known as **bootstrap samples**.
- Each random sample is the same size as the dataset and **sampling with replacement** is used.
- Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different

Bagging (Bootstrap Aggregation)

See paper by Leo Breiman (1994)

- Given dataset $\mathcal{D} = \langle (x_i, y_i) \rangle_{i=1}^{N}$, sample $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ of size N from $\mathcal{D}$ with replacement
- Train classifiers $f_1, \ldots, f_k$ on $\mathcal{D}_1, \ldots, \mathcal{D}_k$
- When predicting use majority (or average if using regression)
- Clearly this approach is only practical for deep networks when training on a large cluster

Example of using bagging

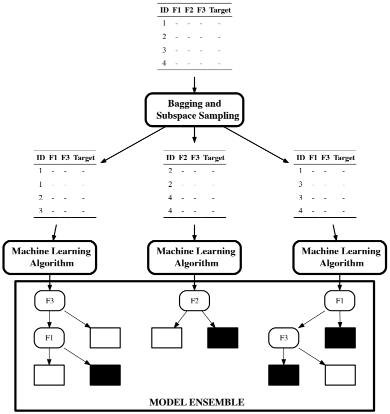
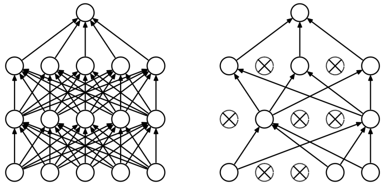
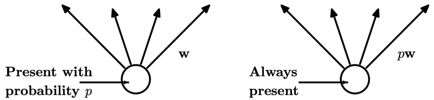


Figure: The process of creating a model ensemble using bagging and subspace sampling.

Bagging: Dropout


(a) Standard Neural Net (b) After applying dropout.



(a) At training time (b) At test time

- For input x drop each hidden unit with probability $1/2$ independently
- Every input will have a potentially different mask
- Potentially exponentially different models, but have "same weights"
- After training whole network is used by having all the weights

Bagging: Random forest

- When bagging is used with decision trees each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This is known as **subspace sampling**.
- The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model.

Recap

Decision trees

Handling continuous features

Growing a tree

Regression trees

Beyond basic decision trees

Model Ensembles

END LECTURE