

Decision Trees

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Outline

1 Examples

2 Algorithm

3 Hyperparameters

Sample dataset: predict if game was played

- Class label denotes whether tennis game was played.
- Columns denote features/attributes/predictors and labels/targets/response
- Rows denote instance-label pairs (x_n, y_n) .

Predictors				Response
Outlook	Temperature	Humidity	Wind	Class
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

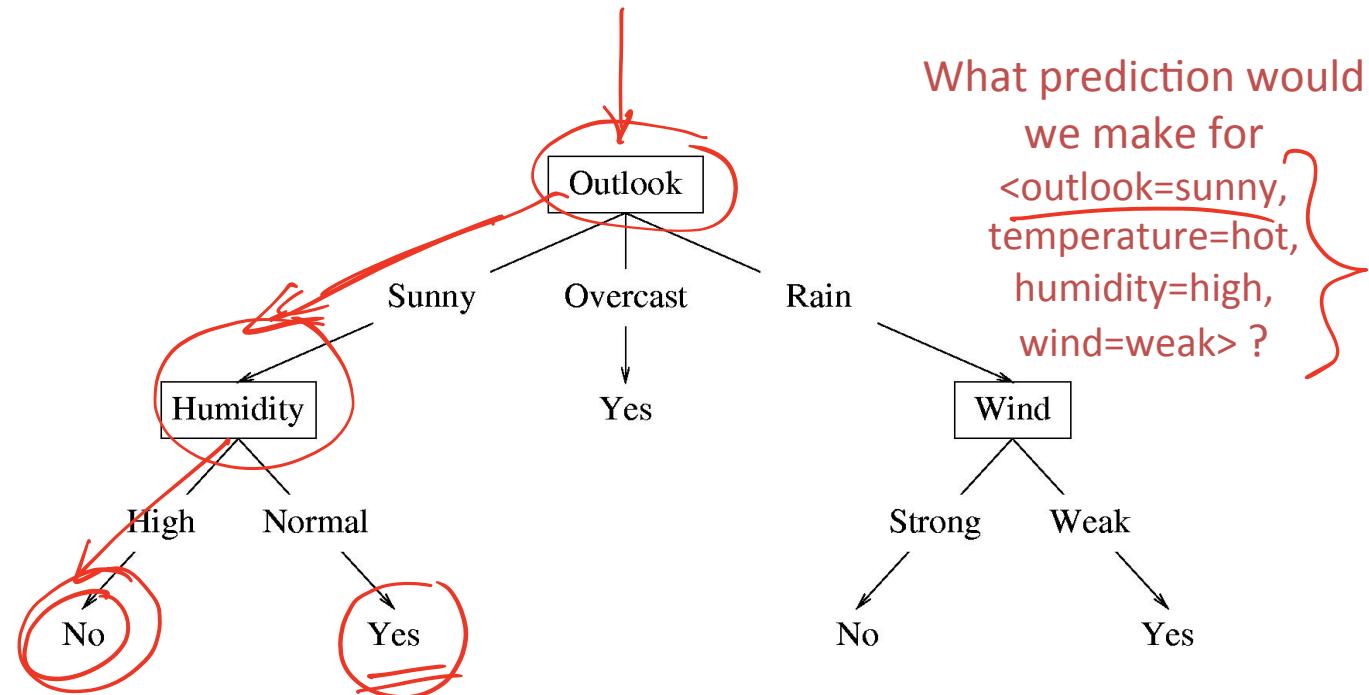
sunny, cool, normal,
weak?

overcast, cool, high,
weak?

Can you describe a
“model” that could be
used to make
decisions in general?

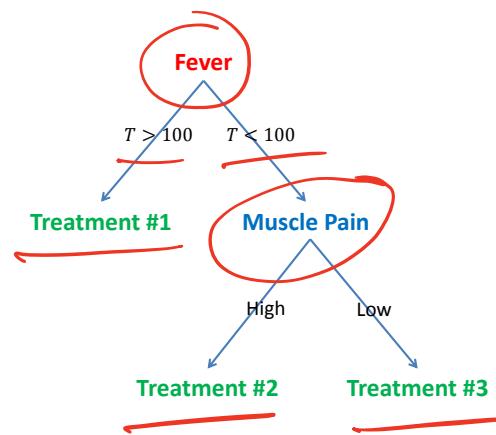
A possible decision tree

- Each internal node : test one feature
- Each edge : select one value for the feature
- Each leaf: predict class

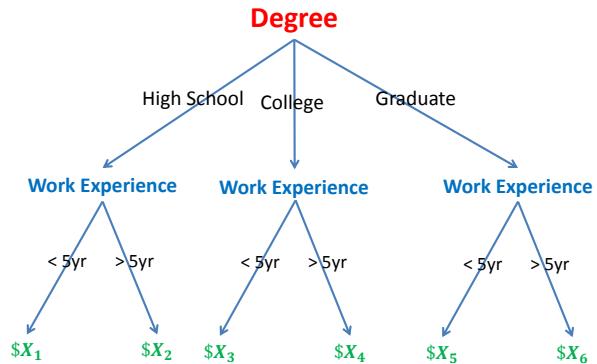


Many decisions are tree structures

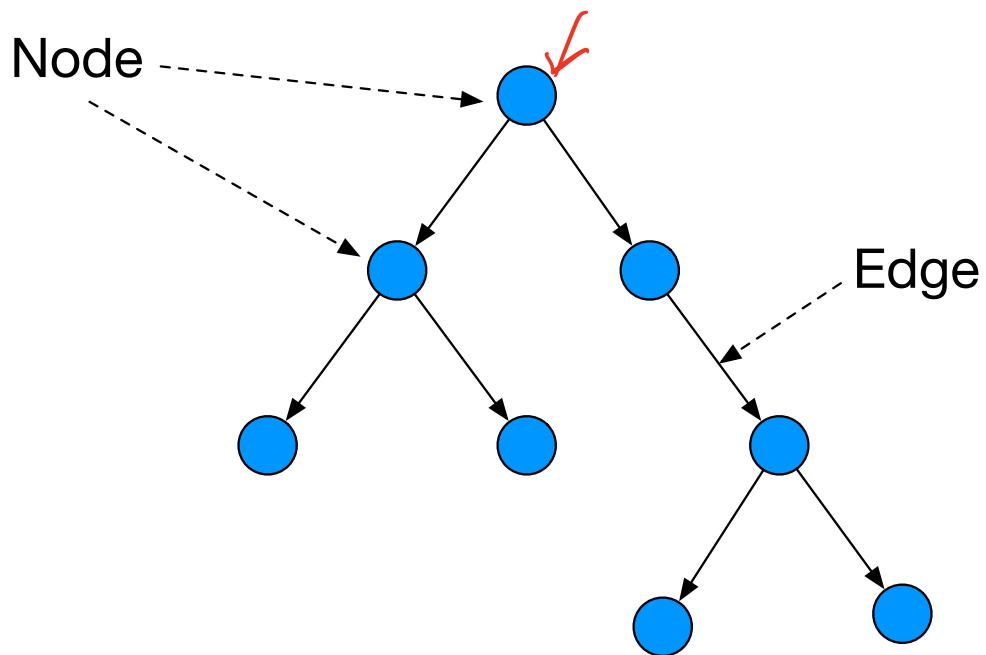
Medical treatment



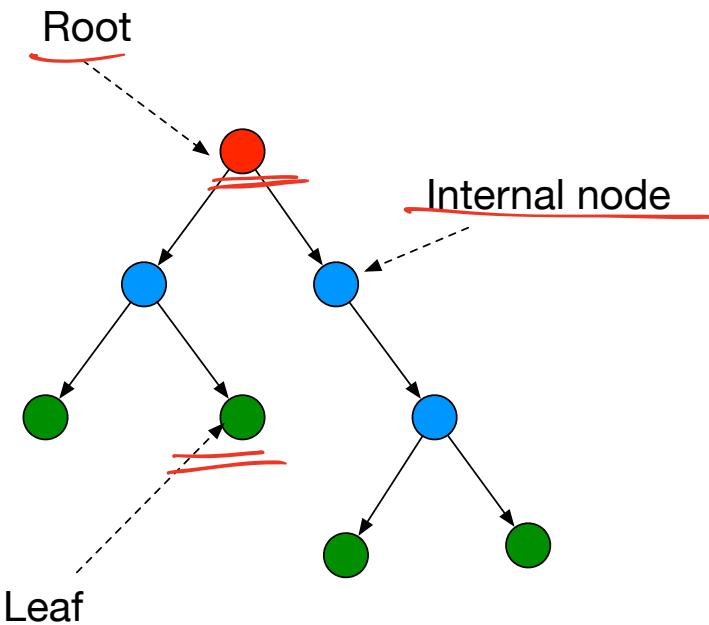
Salary in a company



What is a Tree?

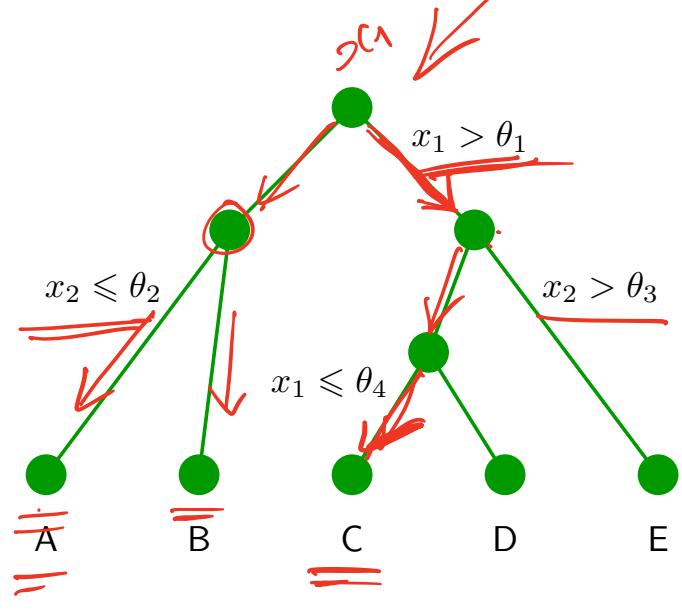
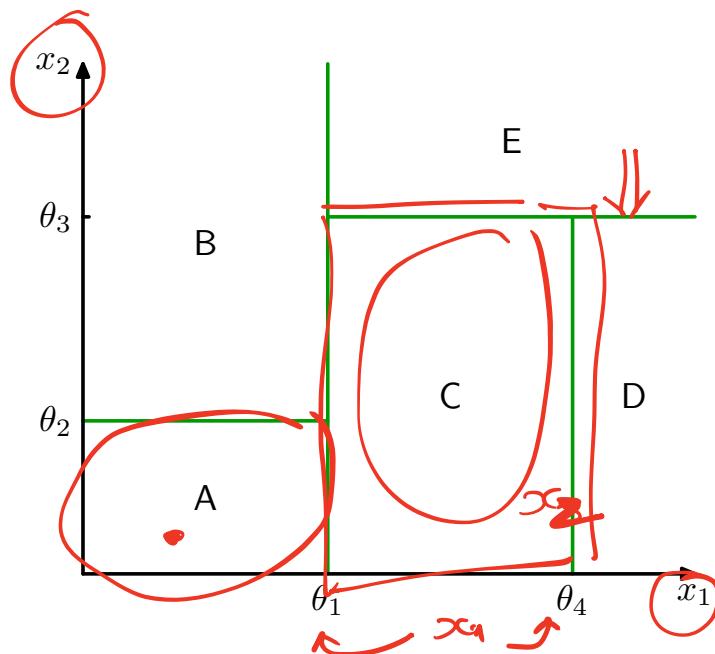


Special Names for Nodes in a Tree



Will sometimes drop the arrows on the edges

A tree partitions the feature space



$$\begin{array}{l} \underline{x_1 > \theta_1 \& x_2 \leq \theta_3} \\ \underline{\theta_1 < x_1 \leq \theta_4 \& x_2 \leq \theta_3} \end{array}$$

(x_1, x_2)

Decision tree learning

Setup

- Set of possible instances \mathbb{X}
 - ▶ Each instance $x \in \mathbb{X}$ is a feature vector
 - ▶ (humidity=low, wind=weak, outlook=rain, temp=hot)
- Set of possible labels \mathbb{Y}
 - ▶ \mathbb{Y} is discrete valued
- Unknown target function $f : \mathbb{X} \rightarrow \mathbb{Y}$
- Model/Hypotheses: $H = \{h | h : \mathbb{X} \rightarrow \mathbb{Y}\}$.
- Each hypothesis h is a decision tree

Goal: Train/induce/learn a function h that maps instance to label.

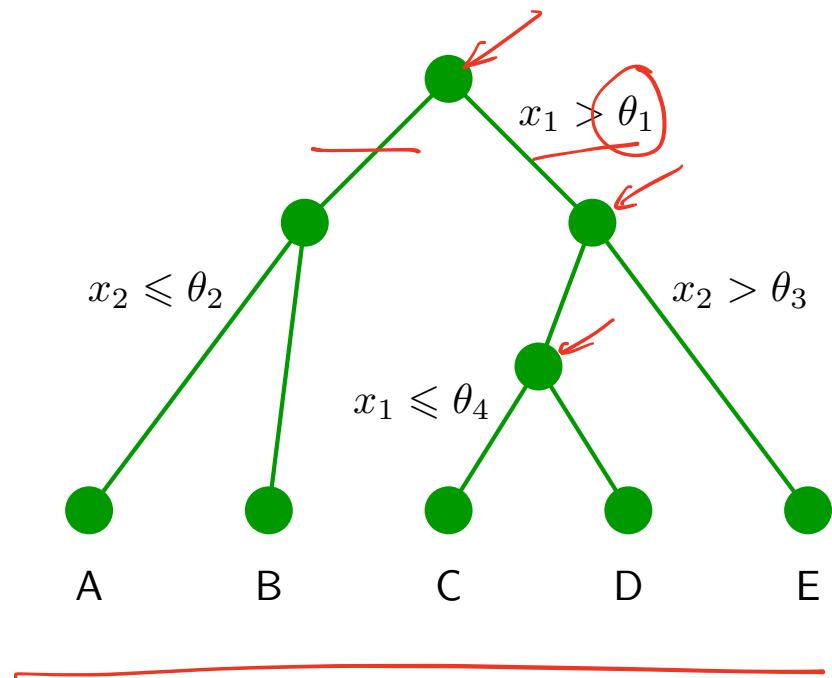
Learning a tree model

Three things to learn:

Learning a tree model

Three things to learn:

- ① The structure of the tree.
- ② The threshold values (θ_i).
- ③ The values for the leaves (A, B, \dots).



Outline

1 Examples

2 Algorithm

3 Hyperparameters

A tree model for deciding where to eat

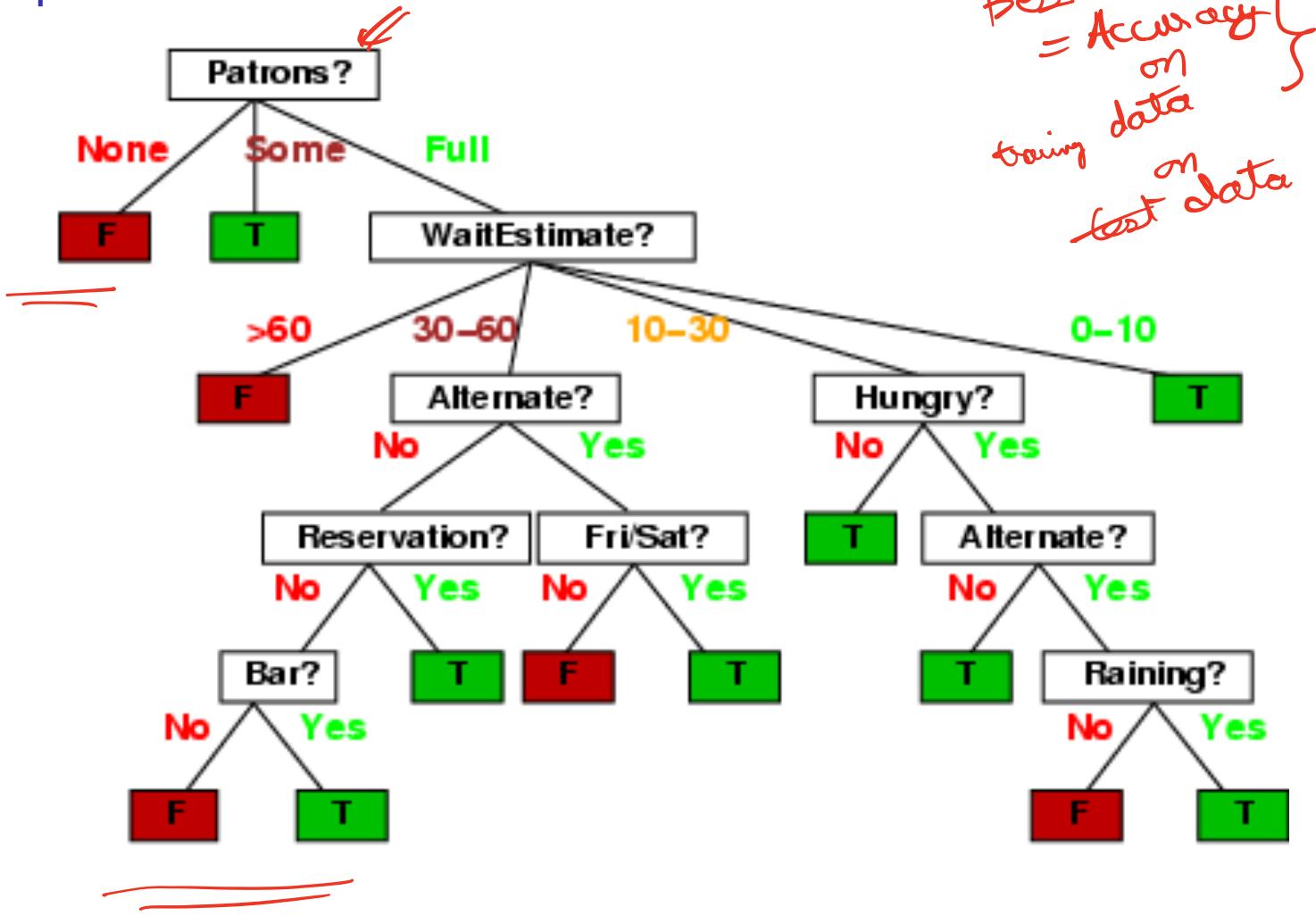
Choosing a restaurant

(Example from Russell & Norvig, AIMA)

Example	Attributes										Target WillWait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X ₁	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X ₂	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X ₃	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X ₄	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X ₅	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X ₆	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X ₇	F	T	F	F	None	\$	T	F	Burger	0–10	F
X ₈	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X ₉	F	T	T	F	Full	\$	T	F	Burger	>60	F
X ₁₀	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X ₁₁	F	F	F	F	None	\$	F	F	Thai	0–10	F
X ₁₂	T	T	T	T	Full	\$	F	F	Burger	30–60	T

Classification of examples is positive (T) or negative (F)

A possible decision tree



Is this the best decision tree?

Ockham's razor

The simplest consistent explanation is best

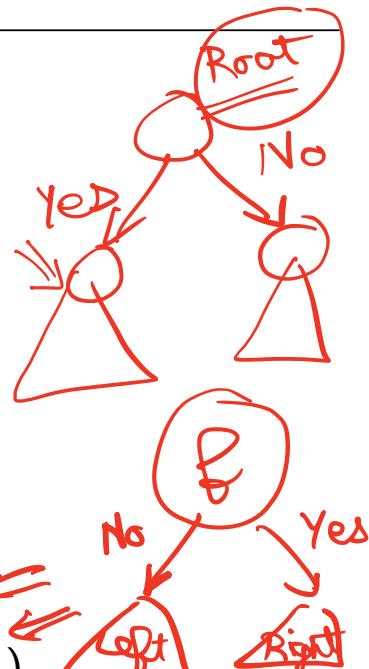
- A form of **inductive bias**.
 - Find smallest decision tree that correctly classifies all training examples
 - NP-hard
 - Instead, greedily construct a decision tree that is pretty small.
-

Decision tree training/learning

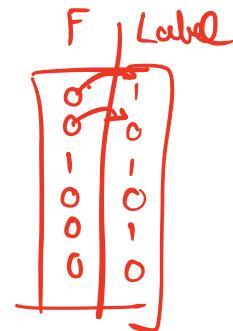
For simplicity assume each feature is binary (takes values YES/NO)

Algorithm 1 DecisionTreeTrain (*data*, *features*)

```
1: guess  $\leftarrow$  the most frequent label in data
2: if all labels in data are the same then
3:   return LEAF (guess)
4: else if features is empty then
5:   return LEAF (guess)
6: else
7:   f  $\leftarrow$  the “best” feature  $\in$  features
8:   NO  $\leftarrow$  the subset of data on which f = NO
9:   YES  $\leftarrow$  the subset of data on which f = YES
10:  left  $\leftarrow$  DecisionTreeTrain (NO, features - {f})
11:  right  $\leftarrow$  DecisionTreeTrain (YES, features - {f})
12:  return NODE(f, left, right)
13: end if
```



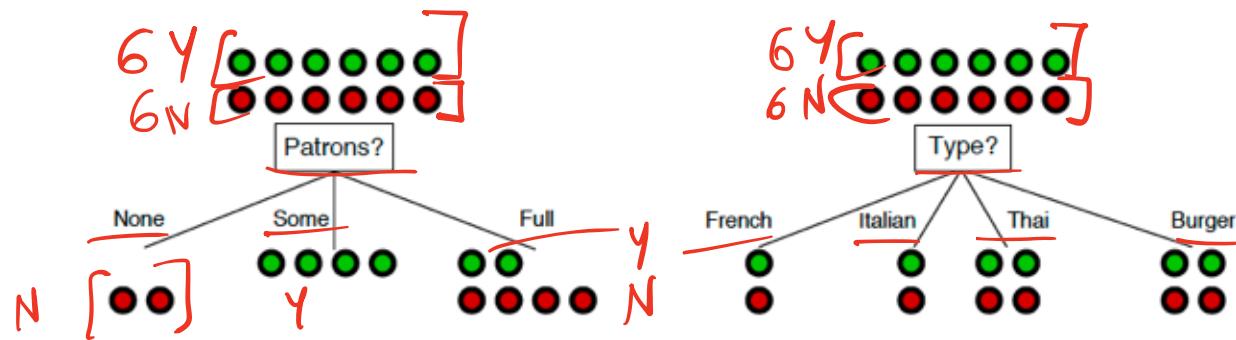
Choosing the best feature



- Possibilities
 - ▶ Random: select a feature at random
 - ▶ Highest accuracy: select feature with largest accuracy
 - ▶ Max-gain: select feature with largest information gain (to be explained shortly).
- ID3 algorithm: One algorithm for decision tree learning
 - ▶ Select the feature with largest information gain.

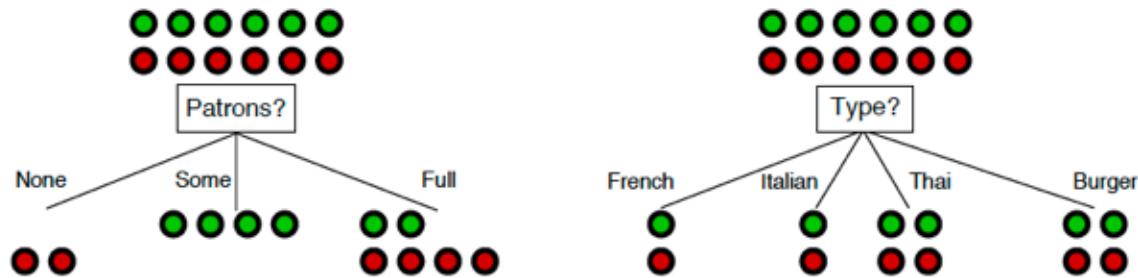
First decision: at the root of the tree

Which attribute to split?



First decision: at the root of the tree

Which attribute to split?



Patrons? is a better choice—gives **information** about the classification

Idea: use **information gain** to choose
which attribute to split

How to measure information gain?

Idea:

Gaining information reduces uncertainty

the base can be 2 ,
though it is not essential
(if the base is 2, the unit
of the entropy is called
“bit”)

Use to entropy to measure uncertainty

If a random variable X has K different values, a_1, a_2, \dots, a_K , its entropy is given by

$$H[X] = - \sum_{k=1}^K P(X = a_k) \log P(X = a_k)$$

$P(X = a_1)$
 $P(X = a_k)$

Entropy

- Measures the amount of uncertainty of a random variable with a specific probability distribution. Higher it is, less confident we are in its outcome.

Examples of computing entropy

$$X \in \{0, 1\}$$

Bernoulli
random
variables

$$P(X=0) = \underline{(-P)}$$

$$P(X=1) = \underline{\underline{P}}$$

$$\begin{aligned} H[X] &= - \left[P(X=0) \log P(X=0) + P(X=1) \log P(X=1) \right] \\ &= - \left[(-P) \log (-P) + P \log P \right] \\ \phi = 1 \quad H[X] &= - \left[\cancel{0 \log 0} + 1 \log 1 \right] \\ &= 0 \end{aligned}$$

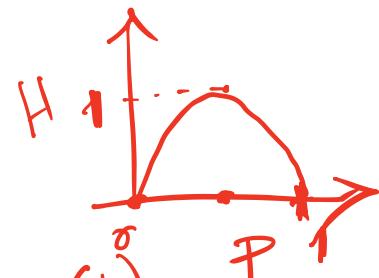
Examples of computing entropy

$$p=0$$

$$H[X] = 0$$

$$p=\frac{1}{2}$$

$$H[X] = - \left[\frac{1}{2} \log_2\left(\frac{1}{2}\right) + \frac{1}{2} \log_2\left(\frac{1}{2}\right) \right]$$

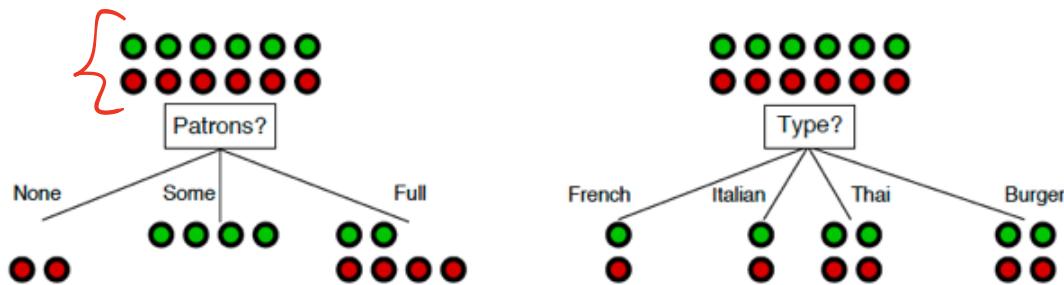


$$\log_2\left(\frac{1}{2}\right) = -\log_2(2) = -1$$

$$H[X] = \left[\frac{1}{2} \times 1 + \frac{1}{2} \times 1 \right]$$

$$-\left[\frac{1}{3} \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \log_2\left(\frac{1}{3}\right) \right] = 1$$

Which attribute to split?



Patrons? is a better choice—gives **information** about the classification

Patron vs. Type?

By choosing Patron, we end up with a partition (3 branches) with smaller entropy, ie, smaller uncertainty (0.45 bit)

By choosing Type, we end up with uncertainty of 1 bit.

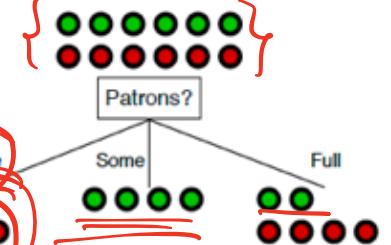
Thus, we choose Patron over Type.



Uncertainty if we go with “Patron”

For “None” branch

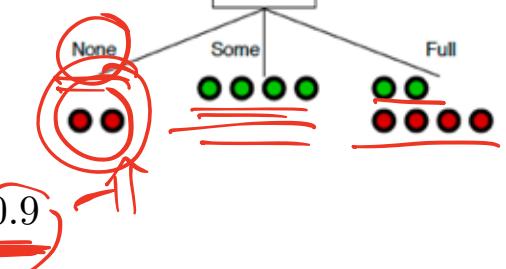
$$-\left(\frac{0}{0+2} \log \frac{0}{0+2} + \frac{2}{0+2} \log \frac{2}{0+2} \right) = 0$$

$$-\left[\frac{3}{2} \log \left(\frac{2}{2} \right) + \frac{1}{2} \log \left(\frac{0}{2} \right) \right]$$


For “Some” branch

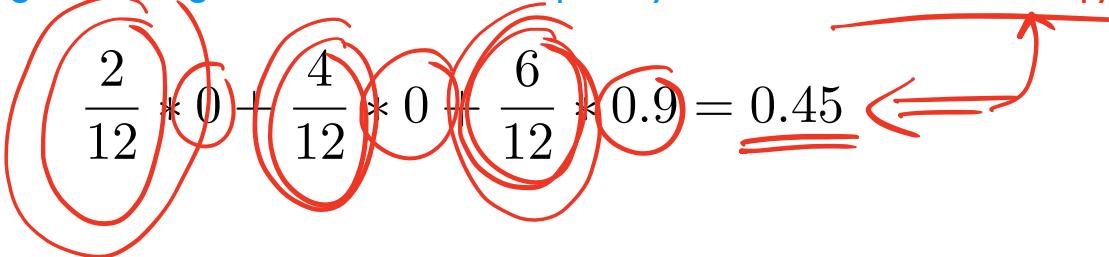
$$-\left(\frac{4}{4+0} \log \frac{4}{4+0} + \frac{0}{4+0} \log \frac{0}{4+0} \right) = 0$$

For “Full” branch

$$-\left(\frac{2}{2+4} \log \frac{2}{2+4} + \frac{4}{2+4} \log \frac{4}{2+4} \right) \approx 0.9$$


For choosing “Patrons”

weighted average of each branch: this quantity is called conditional entropy

$$\frac{2}{12} * 0 + \frac{4}{12} * 0 + \frac{6}{12} * 0.9 = 0.45$$


Conditional entropy

$$H[Y]$$

Definition. Given two random variables X and Y

$$H[Y|X] = \sum_k P(X = a_k) H[Y|X = a_k]$$

In our example

X: the attribute to be split

Y: Wait or not

When $H[Y]$ is fixed, we need only to compare conditional entropy

Relation to information gain

$$\text{GAIN} = H[Y] - H[Y|X]$$

Information gain in Decision trees

- The expected reduction in entropy of the target variable Y due to sorting on the feature X .
- Also called the **mutual information** between Y and X .

Conditional entropy for Type

For “French” branch

$$-\left(\frac{1}{1+1} \log \frac{1}{1+1} + \frac{1}{1+1} \log \frac{1}{1+1}\right) = 1$$

For “Italian” branch

$$-\left(\frac{1}{1+1} \log \frac{1}{1+1} + \frac{1}{1+1} \log \frac{1}{1+1}\right) = 1$$

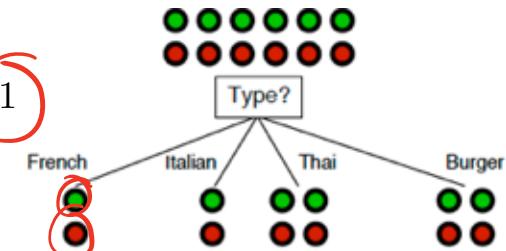
For “Thai” and “Burger” branches

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

For choosing “Type”

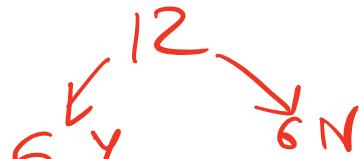
weighted average of each branch:

$$\frac{2}{12} * 1 + \frac{2}{12} * 1 + \frac{4}{12} * 1 + \frac{4}{12} * 1 = 1$$



Comparing the features/attributes

$$H[Y] = 1$$



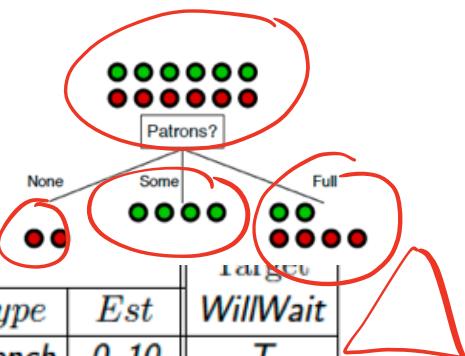
$$P(Y) = \frac{6}{12} = \frac{1}{2}$$

$$\begin{aligned} GAIN[\underline{Y}, \text{Patrons}] &= H[\underline{Y}] - H[\underline{Y} | \text{Patrons}] \\ &= 1 - 0.45 \\ &= 0.55 \end{aligned}$$

$$\begin{aligned} GAIN[\underline{Y}, \text{Type}] &= H[\underline{Y}] - H[\underline{Y} | \text{Type}] \\ &= 1 - 1 \\ &= 0 \end{aligned}$$

next split?

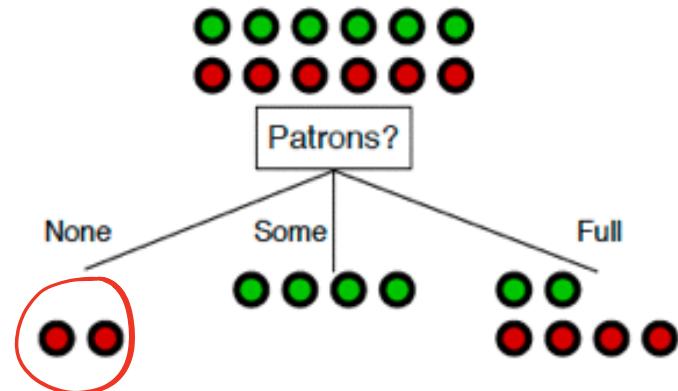
We will look only at the 6 instances with
Patrons == Full



Example	Attributes												
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait		
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T		
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F		
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T		
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T		
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F		
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T		
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F		
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T		
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F		
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F		
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F		
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T		

Classification of examples is positive (T) or negative (F)

Do we split on “Non” or “Some”?



No, we do not

The decision is deterministic, as seen from the training data

Decision tree training/learning

For simplicity assume each feature is binary (takes values YES/NO)

Algorithm 2 DecisionTreeTrain (*data, features*)

```
1: guess  $\leftarrow$  the most frequent label in data
2: if all labels in data are the same then
3:   return LEAF (guess)
4: else if features is empty then
5:   return LEAF (guess)
6: else
7:   f  $\leftarrow$  the “best” feature  $\in$  features
8:   NO  $\leftarrow$  the subset of data on which f = NO
9:   YES  $\leftarrow$  the subset of data on which f = YES
10:  left  $\leftarrow$  DecisionTreeTrain (NO, features  $- \{f\})
11:  right  $\leftarrow$  DecisionTreeTrain (YES, features  $- \{f\})
12:  return NODE(f, left, right)
13: end if$$ 
```

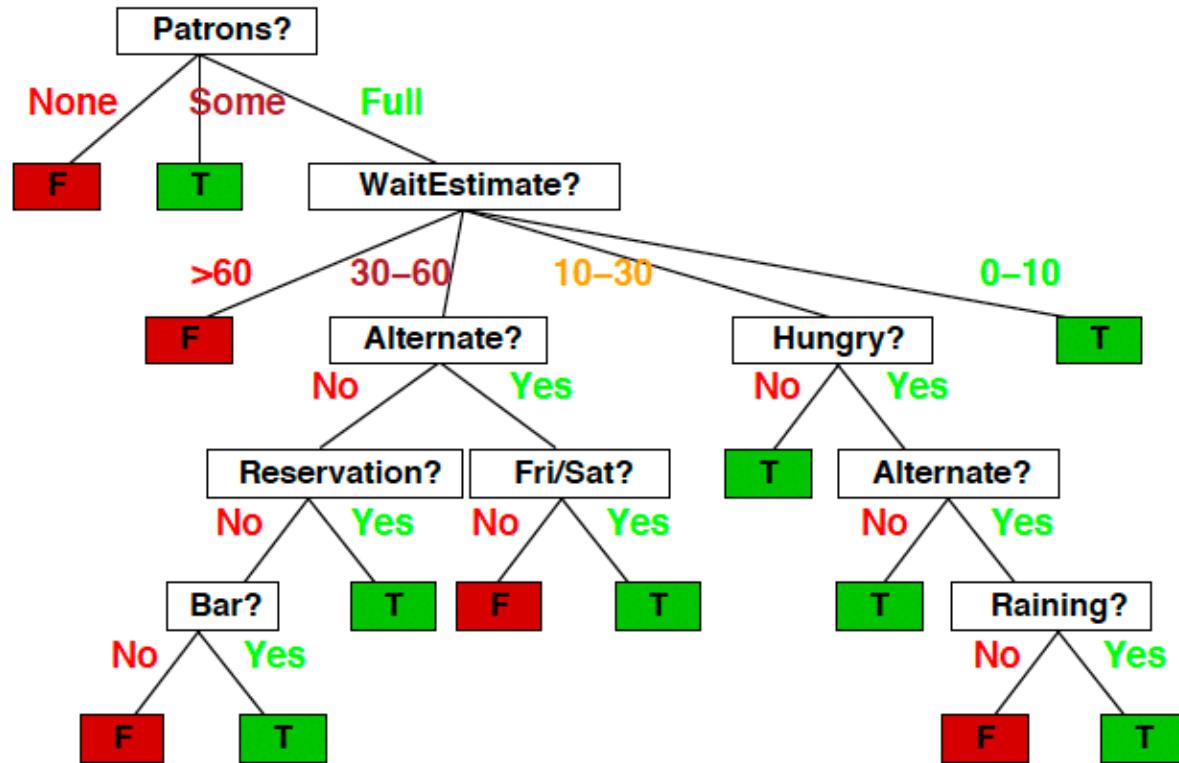
Decision tree training/learning

For simplicity assume each feature is binary (takes values YES/NO)

Algorithm 3 DecisionTreePredict (tree, instance)

```
1: if tree has form LEAF (guess) then
2:   return guess
3: else if tree has form NODE( $f, left, right$  ) then
4:   if  $f = NO$  in instance then
5:     return DecisionTreeTest(left, instance)
6:   else
7:     return DecisionTreeTest(right, instance)
8:   end if
9: end if
```

Greedily we build the tree and get this



Require the tree is not too deep

Prefering shallow trees: a form of inductive bias

Require the tree is not too deep

Prefering shallow trees: a form of inductive bias

- We need to be careful to pick an appropriate tree depth
- If the tree is too deep, we can overfit (memorize the training data).
- If the tree is too shallow, we underfit (not learn enough).
- Max depth is a **hyperparameter** that should be tuned by the data
 - ▶ A parameter that controls the other parameters of the tree.
 - ▶ Max depth of 0: underfitting
 - ▶ Max depth of ∞ : overfitting.

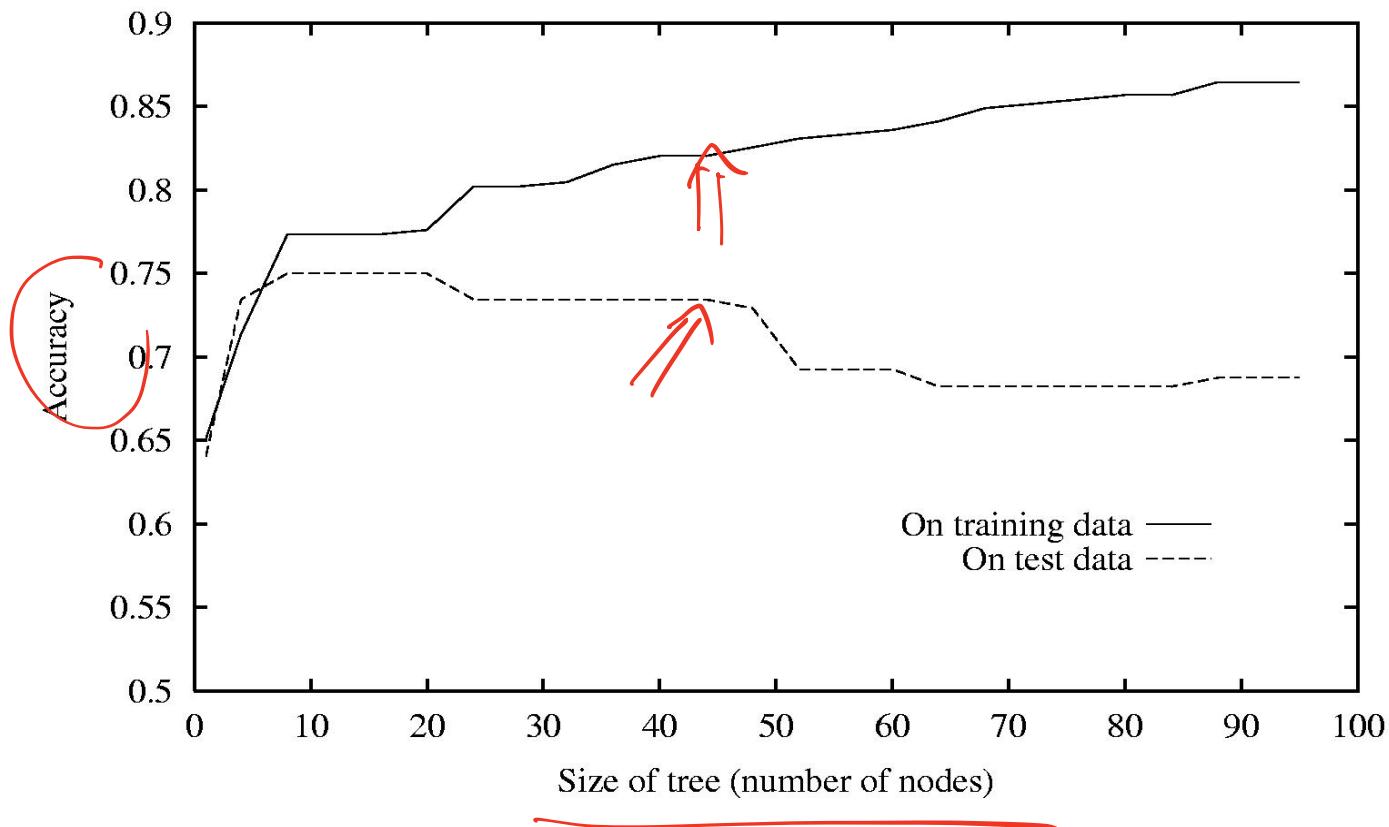
Overfitting

Reasons for overfitting

- Noisy data
 - ▶ Two instances have the same feature values but different class labels
 - ▶ Some of the feature or label values are incorrect
- Some features are irrelevant to classification.
- Target variable is non-deterministic in the features.
 - ▶ In general, we cannot measure all the variables needed to predict. So target variable is not uniquely determined by the input feature values.

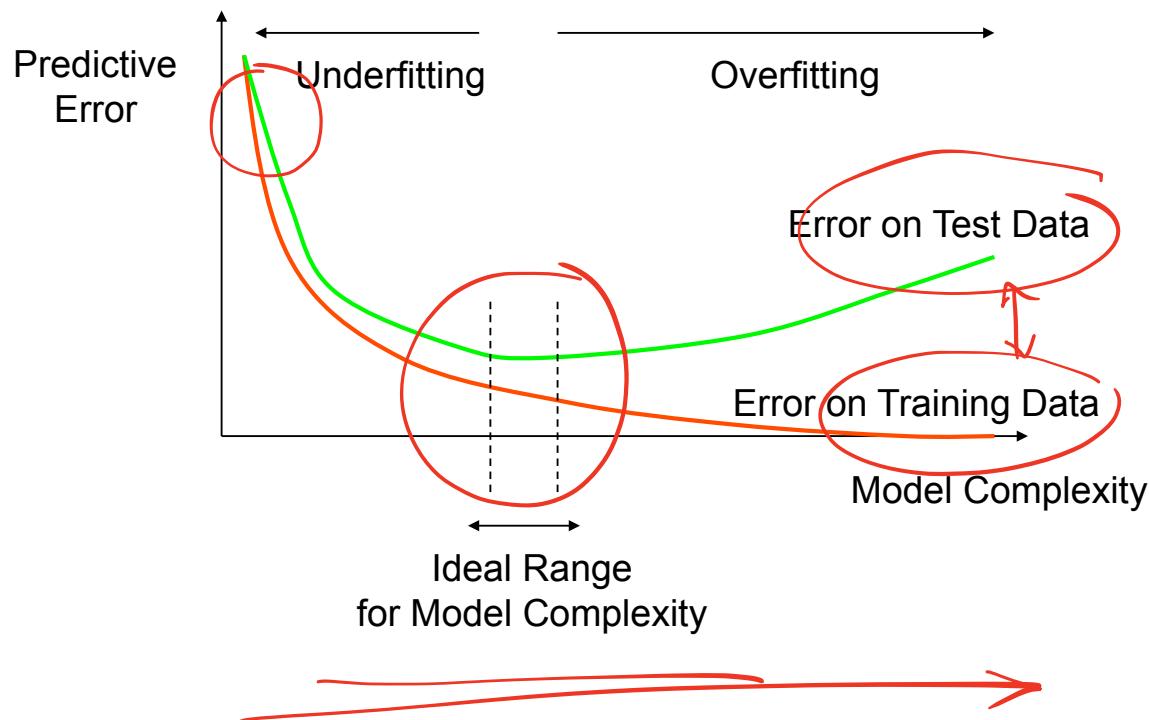
⇒ training error is not guaranteed to be zero

Overfitting in decision trees



Our decision tree learning procedure always increases training set accuracy. Though training accuracy is increasing, test accuracy can decrease.

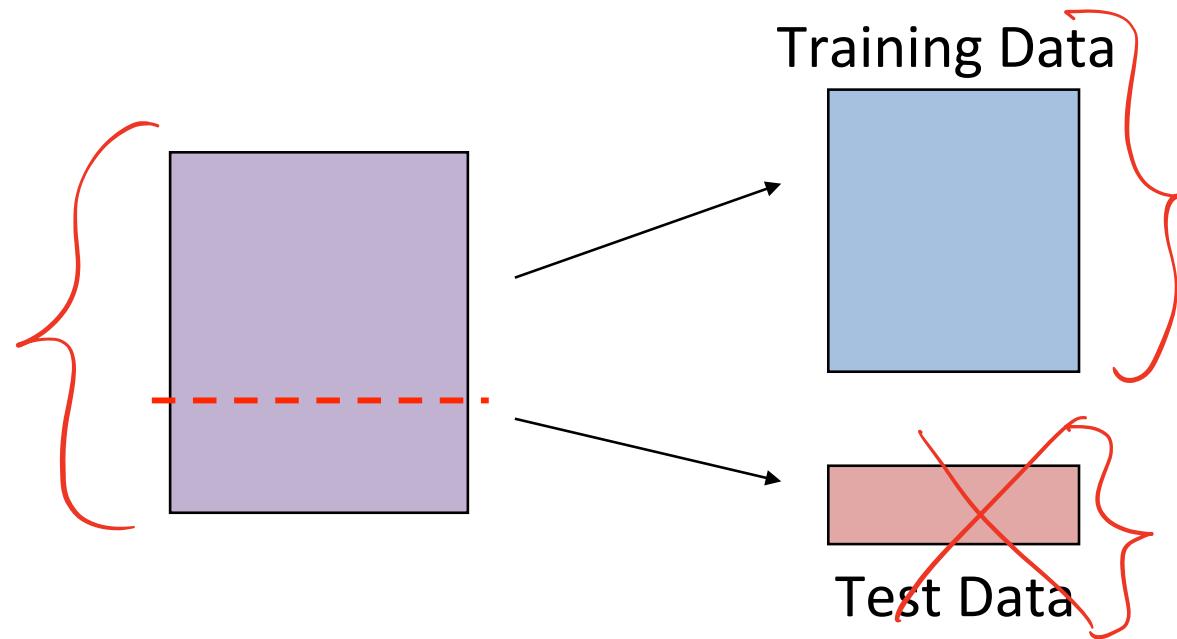
Overfitting



How to detect overfitting?

More generally, how to get a realistic estimate of accuracy

Train and test data

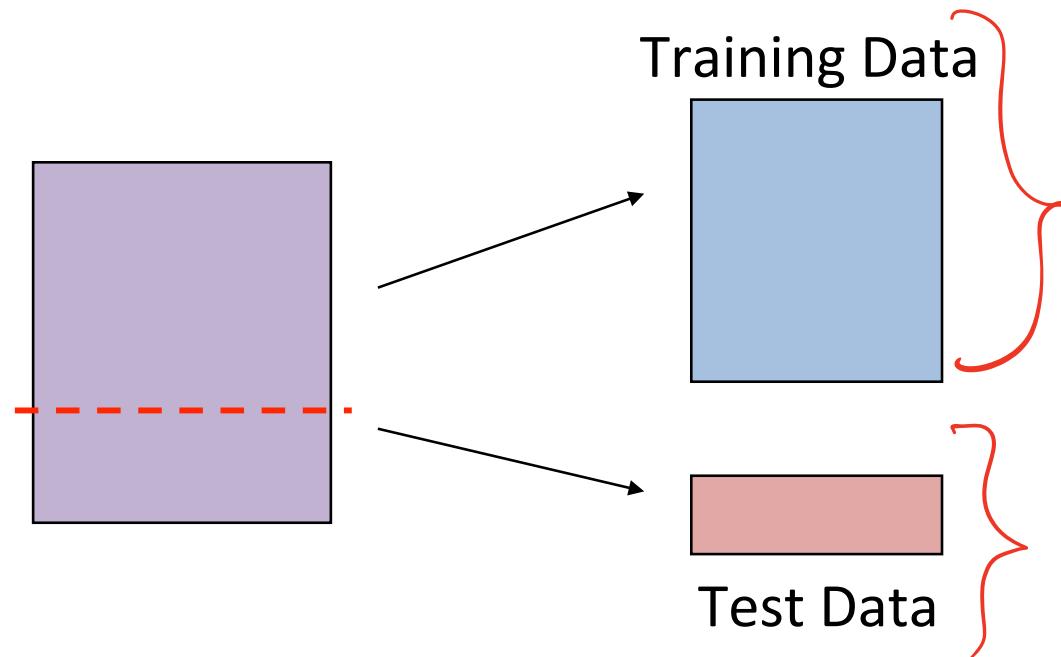


- Train model on training data
- Compute accuracy on test data

How to detect overfitting?

More generally, how to get a realistic estimate of accuracy

Train and test data



- Train model on training data
- Compute accuracy on test data
- **Downside:** Throwing out data.

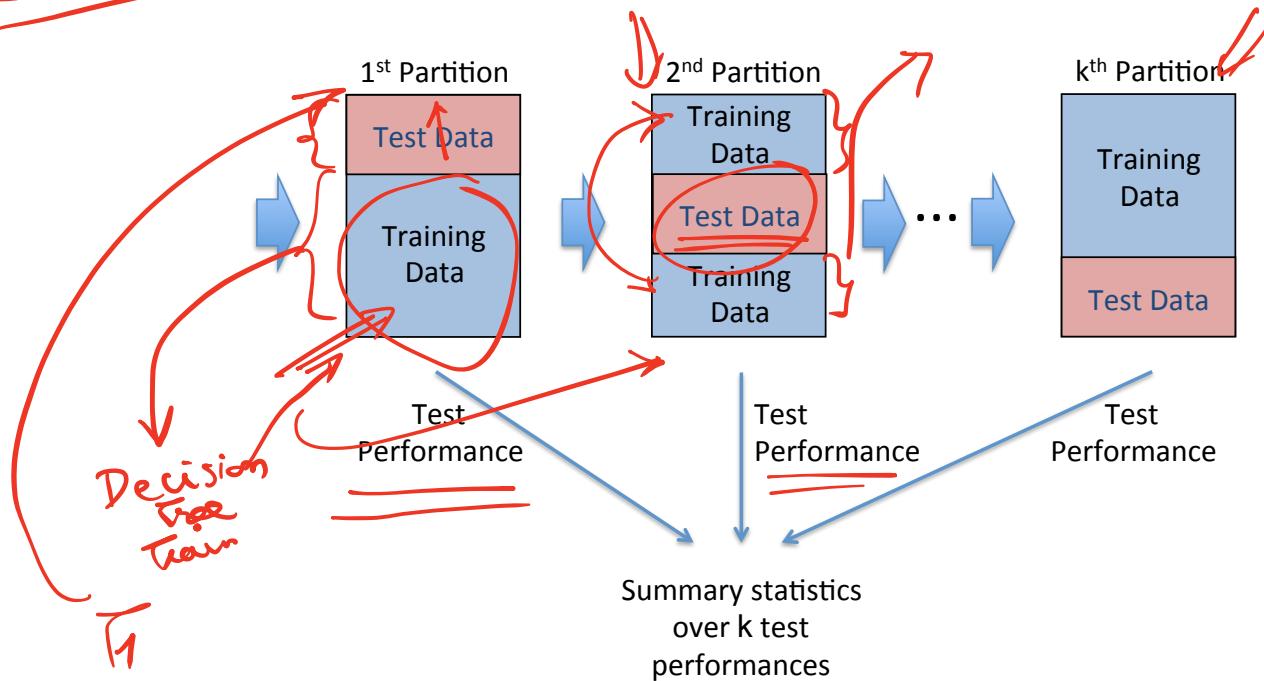
Cross-validation (CV)

Why just choose one particular split of the data?

- In principle, we should do this multiple times since performance may be different for each split
- K-Fold Cross-Validation (e.g., $K = 10$)
 - ▶ Randomly partition full data set of N instances into K disjoint subsets (each roughly of size $\frac{N}{K}$)
 - ▶ Choose each fold in turn as the test set; train model on the other folds and evaluate.
 - ▶ Compute average statistics over K test performances. Can also do leave-one-out CV where $K = N$.

Cross-validation CV

3-fold CV

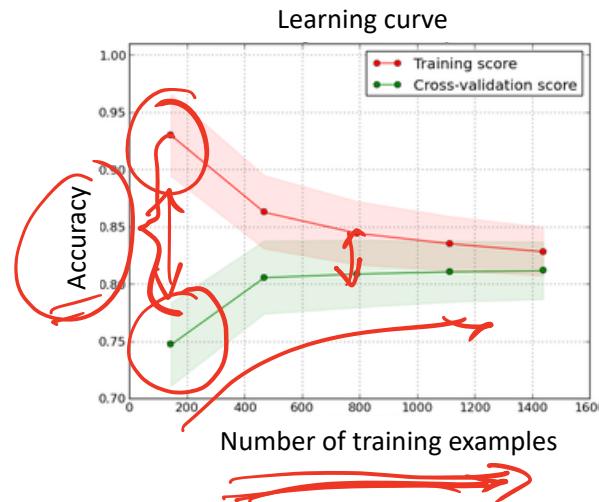


Recipe for K-fold CV

- Split the training data into K equal parts. Denote each part as $\underline{\mathcal{D}_k^{\text{TRAIN}}}$
- for every $\underline{k} \in [1, K]$
 - ▶ Train a model using $\underline{\mathcal{D}_{\setminus k}^{\text{TRAIN}}} = \mathcal{D}^{\text{TRAIN}} - \underline{\mathcal{D}_k^{\text{TRAIN}}}$
 - ▶ Evaluate the performance of the model on $\underline{\mathcal{D}_k^{\text{TRAIN}}}$
- Average the K performance metrics

Visualizing overfitting: Learning curves

- Shows performance versus number of training instances.
 - Compute over a single training/test split.
 - Average over multiple trials of CV

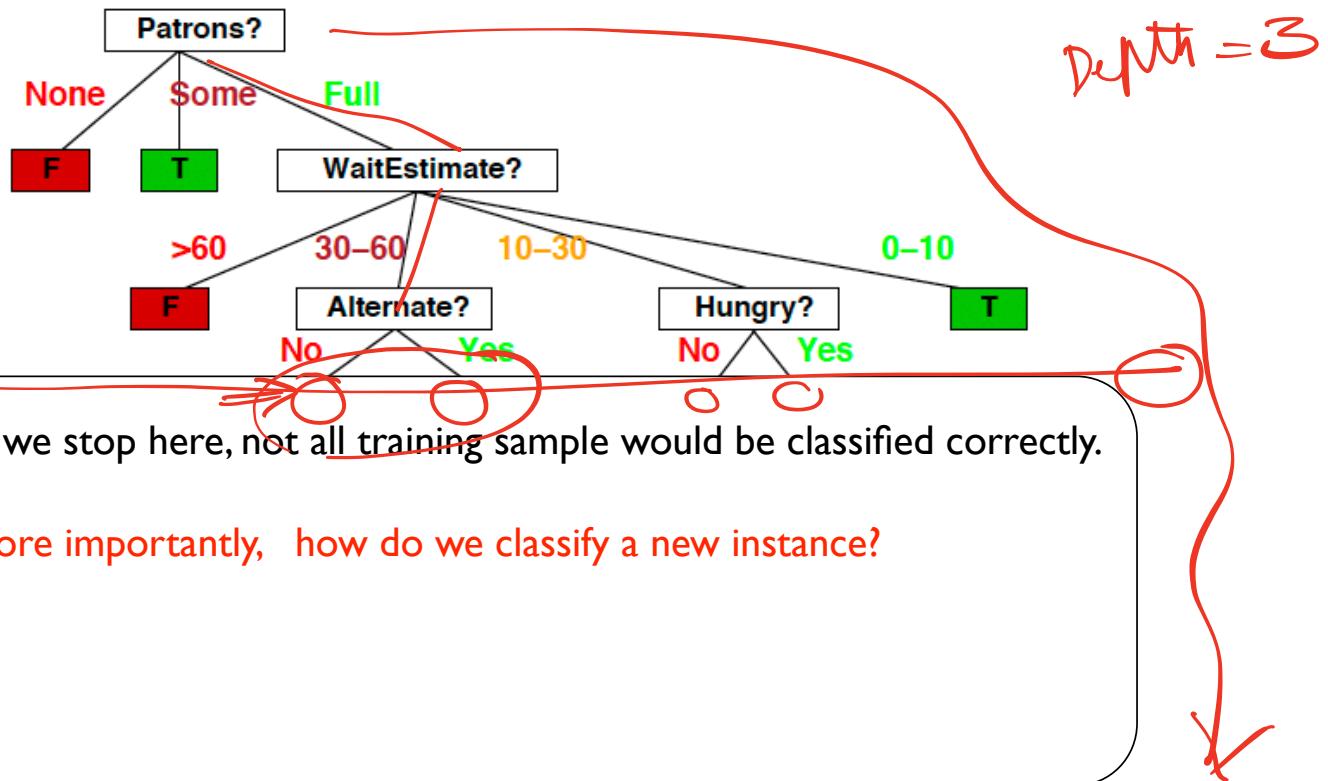


Avoiding overfitting

- Get more training data
- Remove irrelevant features
- Force the decision tree to be simple : Decision tree pruning
 - ▶ Prune while building tree (early stopping)
 - ▶ Prune after building tree (post-pruning)

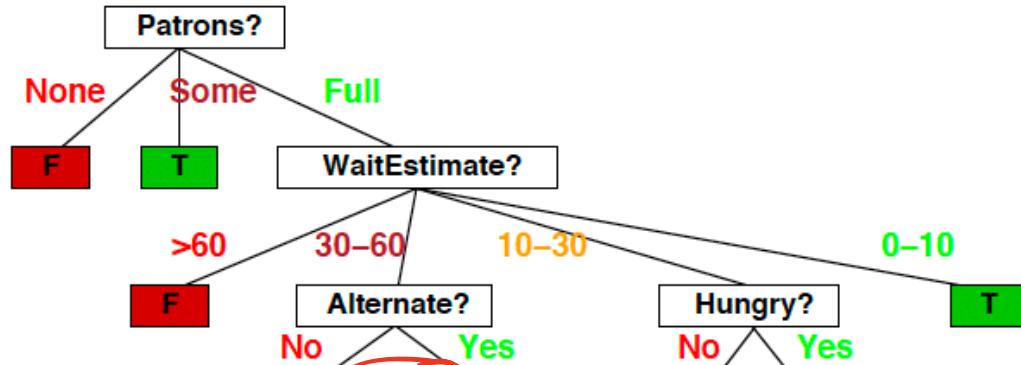
Pruning: Controlling the size of the tree

We would prune to have a smaller one



Control the size of the tree

We would prune to have a smaller one



If we stop here, not all training sample would be classified correctly.

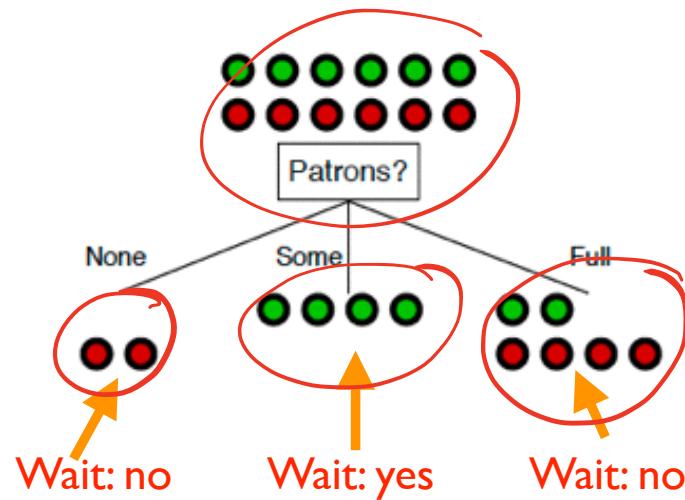
More importantly, how do we classify a new instance?

We label the leaves of this smaller tree with the majority of training samples' labels

Example

Example

We stop after the root (first node)



Outline

1 Examples

2 Algorithm

3 Hyperparameters

Hyperparameters in a Decision tree

Maximum depth of the tree

- Critical to choose max depth carefully for good generalization.
- Choosing max depth (as well as any other hyperparameters) based on empirical performance.

Hyperparameters in a Decision tree

Maximum depth of the tree

- Critical to choose max depth carefully for good generalization.
- Choosing max depth (as well as any other hyperparameters) based on empirical performance.

How do we choose max depth?

$$\begin{array}{ccccc} Y & & 100 & H[Y] \\ & \text{y=0} & 30 & P(Y=0) = \frac{30}{100} \\ & \text{y=1} & 70 & P(Y=1) = \frac{70}{100} \end{array}$$

Hyperparameter selection (tuning) by using a validation dataset

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- They are used for learning $h(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$
- They are used for assessing how well $h(\cdot)$ will do in predicting an unseen $x \notin \mathcal{D}^{\text{TRAIN}}$

Validation (or development) data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(x_1, y_1), (x_2, y_2), \dots, (x_L, y_L)\}$
- They are used to optimize hyperparameter(s).

Training data, validation and test data should *not* overlap!

Recipe for hyperparameter selection

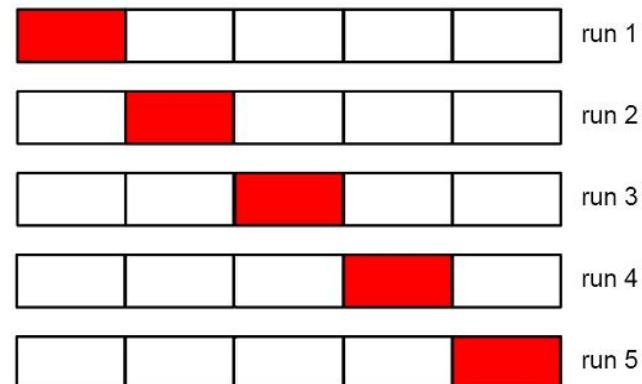
- For each possible value of the hyperparameter (say $\text{Max depth} = \underline{1, 3, \dots}$)
 - ▶ Train a model using $\mathcal{D}^{\text{TRAIN}}$
 - ▶ Evaluate the performance of the model on \mathcal{D}^{DEV}
- Choose the model with the best performance on \mathcal{D}^{DEV}
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Cross-validation (CV)

What if we do not have validation data?

- We split the training data into K equal parts (termed **folds or splits**).
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that ***on average***, the model performing the best

$K = 5$: 5-fold cross validation



Special case: when $K = N$, this will be leave-one-out (LOO).

Recipe

- Split the training data into K equal parts. Denote each part as $\mathcal{D}_k^{\text{TRAIN}}$
- For each possible value of the hyperparameter (say Max depth = 1, 3, \dots)
 - ▶ for every $k \in [1, K]$
 - ★ Train a model using $\mathcal{D}_{\setminus k}^{\text{TRAIN}} = \mathcal{D}^{\text{TRAIN}} - \underline{\mathcal{D}_k^{\text{TRAIN}}}$
 - ★ Evaluate the performance of the model on $\underline{\mathcal{D}_k^{\text{TRAIN}}}$
 - ▶ Average the K performance metrics
- Choose the hyperparameter corresponding to the best averaged performance
- Use the best hyperparameter to train on a model using all $\underline{\mathcal{D}^{\text{TRAIN}}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Summary

- Model/hypotheses: Trees
 - Parameters: Features at each internal node, feature values along edges and labels at leaves.
 - Hyperparameters: max depth

Summary

Advantages of using trees

- Easily interpretable by humans (as long as the tree is not too big)
- Computationally efficient
- Handles both numerical and categorical data

Disadvantages

- Heuristic training techniques
 - ▶ Finding a tree that minimizes empirical error is NP-hard
 - ▶ Resort to greedy approaches.

Summary

- You should now be able to use decision trees to do machine learning.
- Given data, use training, development and test splits (or cross-validation).
- Use training and development to tune for Max depth that trades off overfitting and underfitting.
- Use test to get an estimate of generalization or accuracy on unseen data.