18-661 Introduction to Machine Learning

Decision Trees

Spring 2020

ECE – Carnegie Mellon University

Course logistics

- HW 4 will be released on Friday, due March 18.
- Midterm exam will be graded by next week.
- Mid-semester grades (including HWs 1 to 3 and the midterm exam) are due to the registrar by March 9. These do not go on your transcript and are only meant as an indicator of how you are doing so far. There is plenty of time to make up for low grades!

What's next

- Decision trees, boosting
- Spring break!
- Neural networks
- Unsupervised learning (clustering, PCA)
- Reinforcement learning

Outline

1. Recap: Nearest Neighbors
2. Decision Trees: Motivation
3. Learning A Decision Tree

Recap: Nearest Neighbors

Parametric vs. nonparametric models

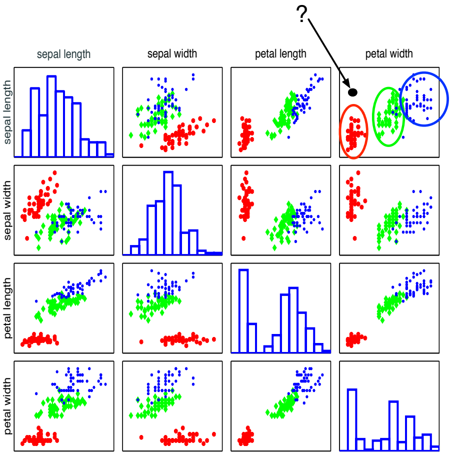
Key difference:

- Parametric models assume that the data can be characterized via some fixed set of parameters θ corresponding to a fixed set of features. Given these parameters, our future predictions are independent of the data $\mathcal{D}$, i.e., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.
  - Often simpler and faster to learn, but can sometimes be a poor fit
- Nonparametric models instead assume that the model features depend on the data $\mathcal{D}$. The number of parameters tends to grow with the size of the dataset.
  - More complex and expensive, but can learn more flexible patterns

Recognizing flowers

Types of Iris: setosa, versicolor, and virginica

Labeling an unknown flower type



Closer to red cluster: so labeling it as setosa

Nearest neighbor classification (NNC)

The nearest neighbor of a point x is

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

$\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$, i.e., it is the index of a training instance.

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n \in [\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

Classification rule

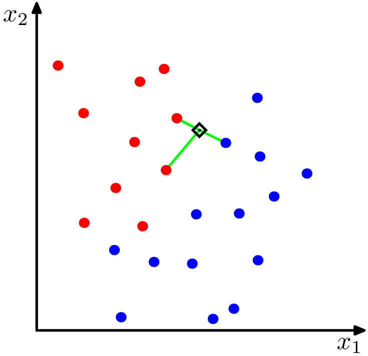
$$y = f(\boldsymbol{x}) = y_{\mathrm{nn}(\boldsymbol{x})}$$

Example: if $\mathrm{nn}(\boldsymbol{x}) = 2$, then $y_{\mathrm{nn}(\boldsymbol{x})} = y_2$, which is the label of the 2nd data point.

Intuitively, we find the training instance that most resembles $\boldsymbol{x}$ (is its nearest neighbor) and apply its label.

Visual example

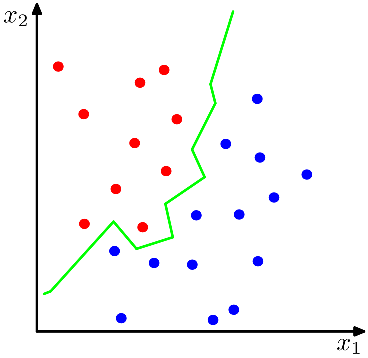
In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance, thus, $\boldsymbol{x}$ will be labeled as red.



(a)

Decision boundary

For every point in the space, we can determine its label using the NNC rule. This gives rise to a decision boundary that partitions the space into different regions.



(b)

K-nearest neighbor (KNN) classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\mathrm{nn}_1(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 2nd-nearest neighbor: $\mathrm{nn}_2(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 3rd-nearest neighbor: $\mathrm{nn}_2(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x}) - \mathrm{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

The set of K-nearest neighbors

$$\mathrm{knn}(\boldsymbol{x}) = \{\mathrm{nn}_1(\boldsymbol{x}), \mathrm{nn}_2(\boldsymbol{x}), \cdots, \mathrm{nn}_K(\boldsymbol{x})\}$$

Let $\boldsymbol{x}(k) = \boldsymbol{x}_{\mathrm{nn}_k(\boldsymbol{x})}$, then

$$\|\boldsymbol{x} - \boldsymbol{x}(1)\|_2^2 \leq \|\boldsymbol{x} - \boldsymbol{x}(2)\|_2^2 \cdots \leq \|\boldsymbol{x} - \boldsymbol{x}(K)\|_2^2$$

How to classify with K neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
  - vote for c is 1
  - vote for $c' \neq c$ is 0

  We use the indicator function $\mathbb{I}(y_n == c)$ to represent the votes.
- Aggregate everyone's vote

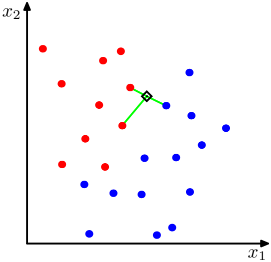
$$v_c = \sum_{n \in \text{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\mathrm{C}]$$

- Choose the label with the most votes

$$y = f(\boldsymbol{x}) = \arg\max_{c \in [\mathrm{C}]} v_c$$

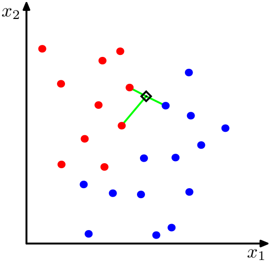
Example

K=1, Label: red



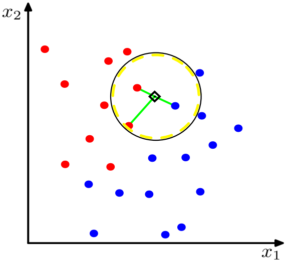
(a)

K=3, Label: red



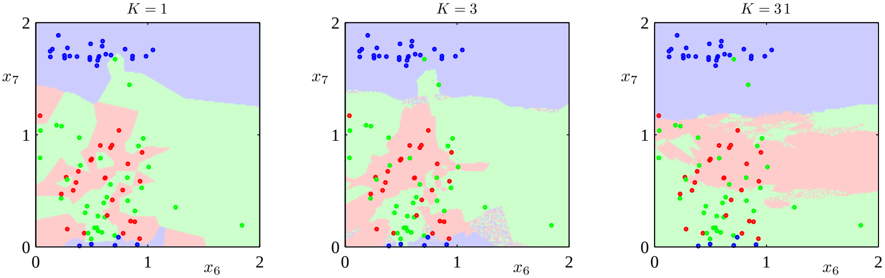
(a)

K=5, Label: blue



(a)

How to choose an optimal K?



When K increases, the decision boundary becomes smooth.

Hyperparameters in NN

Two crucial choices for NN

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance), for example, from the following generalized distance measure

$$\|\boldsymbol{x} - \boldsymbol{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$

for $p \geq 1$.

These are not specified by the algorithm itself — resolving them requires empirical studies and are task/dataset-specific.

Tuning by using a validation dataset

Training data

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathrm{N}, y_\mathrm{N})\}$
- They are used for learning $f(\cdot)$

Test data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathrm{M}, y_\mathrm{M})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Validation data

- L samples/instances: $\mathcal{D}^{\text{VAL}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathrm{L}, y_\mathrm{L})\}$
- They are used to optimize hyperparameter(s).

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

Preprocess data

Normalize data to have zero mean and unit standard deviation in each dimension

- Compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{\mathrm{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Many other ways of normalizing data — you would need/want to try different ones and pick among them using (cross) validation.

Mini-summary

Advantages of NNC

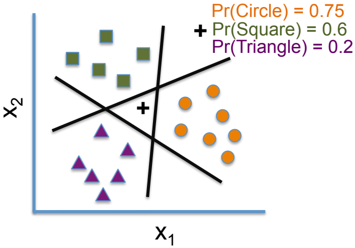
- Computationally, simple and easy to implement – just compute distances
- Can learn complex decision boundaries

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a data point
- We need to "carry" the training data around. Without it, we cannot do classification. This type of method is called nonparametric.
- Choosing the right distance measure and K can be difficult.

Decision Trees: Motivation

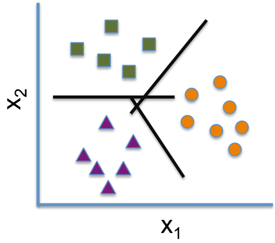
Recall: Multi-class classification



We combined binary decision boundaries to partition the feature space

- One-versus-all approach

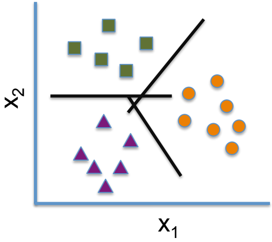
Recall: Multi-class classification



We combined binary decision boundaries to partition the feature space

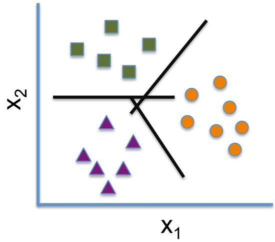
- One-versus-one approach

Recall: Multi-class classification



- Suppose the 3 classes are 3 possible treatments for an illness and you recommend treatment 1.
- The patient sues you and your lawyer needs to explain the reasoning behind the decision in court. What would she say?
  - "$\mathbf{w}_{(1)}^\top \mathbf{x} > 0$ and $\mathbf{w}_{(2)}^\top \mathbf{x} < 0$"? This might not convince the judge.
  - "Treatment 1 worked for similar patients"? This ignores the structure of your data.

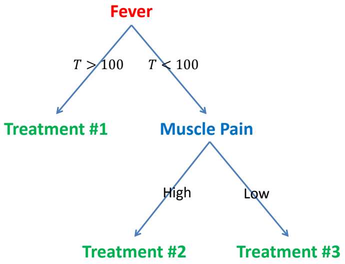
Need interpretable decision boundaries



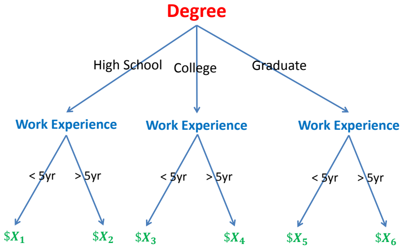
- Should be able to explain the reasoning in clear terms, e.g., "I always recommend treatment 1 when a patient has fever $\geq 100F$"
- The rules that you use to make decisions can be easily used by a lay-person without performing complex computations
- Decision trees can provide such simple decision rules

Many decisions are tree structured

Medical treatment

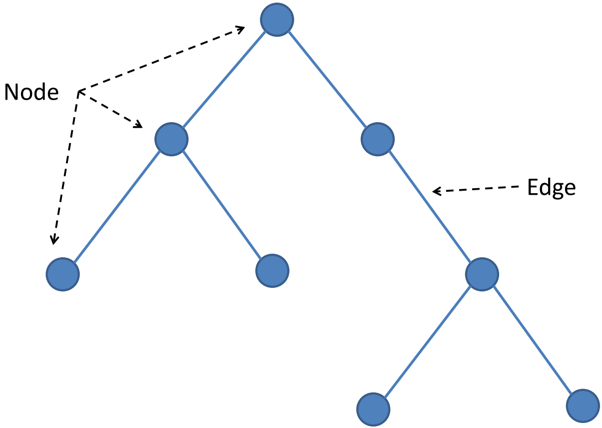


Salary in a company

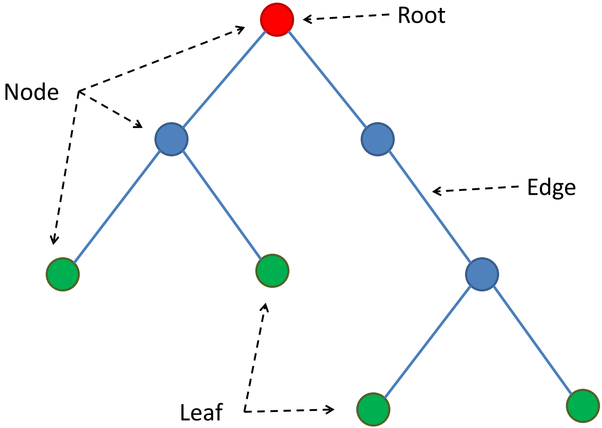


Other examples: fault detection in manufacturing systems, student admissions decisions, jail/parole decisions

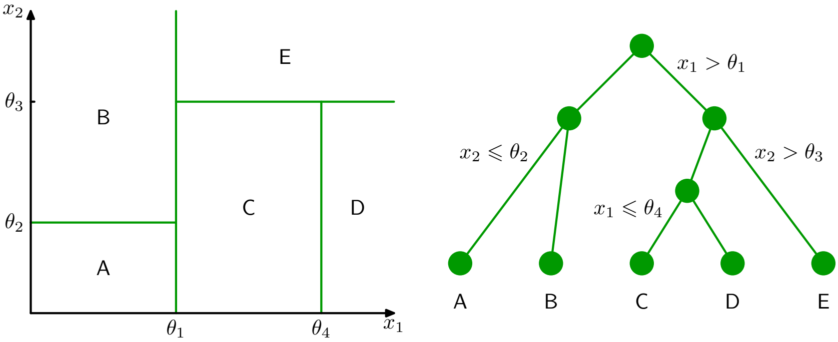
What is a tree?



Special names for nodes in a tree

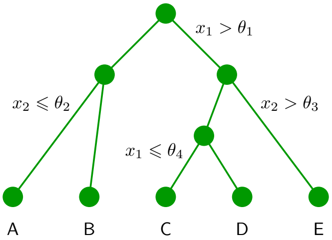


A tree partitions the feature space



Learning A Decision Tree

Learning a tree model



Three things to learn:

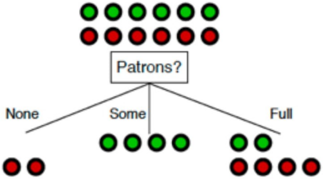
1. The structure of the tree.
2. The threshold values (θ_i).
3. The values for the leaves ($A, B, \ldots$).

Example: Choosing whether you want to wait at a restaurant

Attributes										Target
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
T	F	F	T	Some	$$$	F	T	French	0–10	T
T	F	F	T	Full	$	F	F	Thai	30–60	F
F	T	F	F	Some	$	F	F	Burger	0–10	T
T	F	T	T	Full	$	F	F	Thai	10–30	T
T	F	T	F	Full	$$$	F	T	French	>60	F
F	T	F	T	Some	$$	T	T	Italian	0–10	T
F	T	F	F	None	$	T	F	Burger	0–10	F
F	F	F	T	Some	$$	T	T	Thai	0–10	T
F	T	T	F	Full	$	T	F	Burger	>60	F
T	T	T	T	Full	$$$	F	T	Italian	10–30	F
F	F	F	F	None	$	F	F	Thai	0–10	F
T	T	T	T	Full	$	F	F	Burger	30–60	T

Use the attributes to decide whether to wait (T) or not wait (F)

Which attribute to split first?



- Patron is a better choice – gives more information to help distinguish between the labels
- Intuition: Like playing 20 questions and choosing carefully which question to ask first
- More formally: use information gain to choose which attribute to split

How to measure information gain $I(X;Y)$?

Idea: Gaining information is equivalent to reducing our uncertainty.

- Use entropy $H(Y)$ to measure uncertainty in Y.
- We define $H(Y)$ and $H(Y|X)$ next

Definition (Entropy)

If a random variable Y takes K different values, a_1, $a_2 ... a_K$, then its entropy is

$$H[Y] = -\sum_{i=1}^{K} \Pr(Y = a_i) \log \Pr(Y = a_i)$$

Convention: $0 \log 0$ is considered as 0

Example: Entropy of a Bernoulli random variable

What is the entropy $H(Y)$ of Y, which is 1 with probability p and 0 otherwise?

Find the entropy $H(Y)$ for $p = 0.5$, $p = 0.25$, $p = 0$.

- For $p = 0.5$

$$H(Y) = -(0.5 \log 0.5 + 0.5 \log 0.5) = \log 2 = 1 \text{ bit (log is base 2)}$$

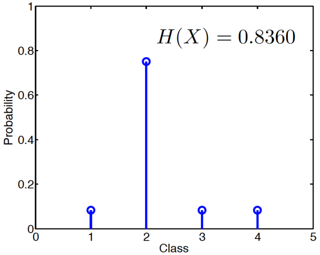
- For $p = 0.25$

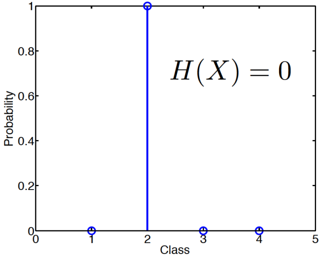
$$H(Y) = -(0.25 \log 0.25 + 0.75 \log 0.75) = 2 \log 2 - 0.75 \log 3 = 0.81 \text{ bits}$$

- For $p = 0$, $H(Y) = 0$

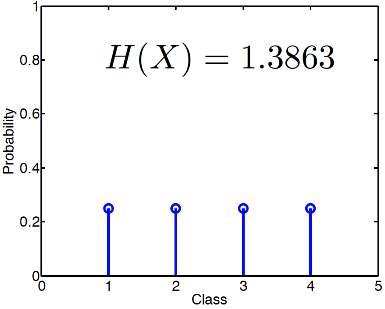
With more uncertainty ($p = 0.5$), we have a larger entropy.

Illustrating Entropy





Given a range of possible values, entropy is maximized with a uniform distribution.



Conditional entropy

Definition (Conditional Entropy)

Given two random variables X and Y

$$H[Y|X] = \sum_k P(X = a_k)H[Y|X = a_k] \tag{1}$$

In our restaurant example:

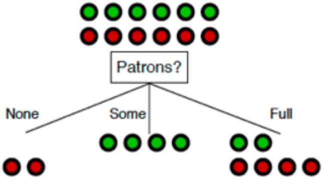
- X: the attribute to be split
- Y: wait or not (the labels)

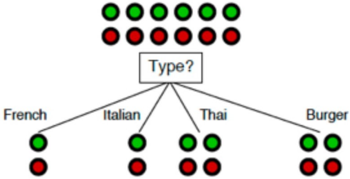
Definition (Information Gain)

$$I(X;Y) = H[Y] - H[Y|X] \tag{2}$$

Measures the reduction in entropy (i.e., the reduction of uncertainty in Y) when we also consider X.

Which attribute to split?

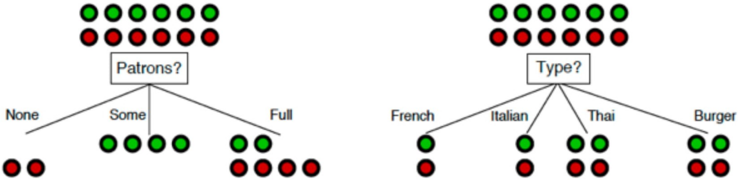




Patron vs. Type?

- Let us compute the information gain $I(X;Y) = H[Y] - H[Y|X]$ for Patron and Type
- When $H[Y]$ is fixed, we need only to compare conditional entropies

Information Gain if we split "Patron"

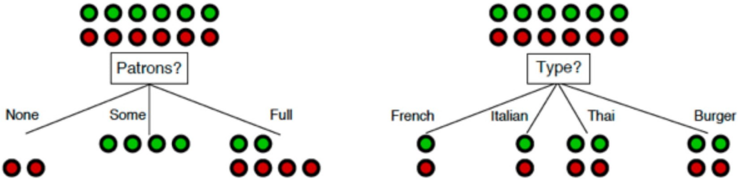


- $H(Y) = -\frac{6}{12}\log\frac{6}{12} - \frac{6}{12}\log\frac{6}{12} = 1$ bit
- $H(Y|X = none) = 0$
- $H(Y|X = some) = 0$
- $H(Y|X = full) = -\left(\frac{2}{2+4}\log\frac{2}{2+4} + \frac{4}{2+4}\log\frac{4}{2+4}\right) \approx 0.9$ bits
- Thus the conditional entropy is

$$H(Y|X) = \left(\frac{2}{12} \times 0 + \frac{4}{12} \times 0 + \frac{6}{12} \times 0.9\right) = 0.45 \text{ bits}$$

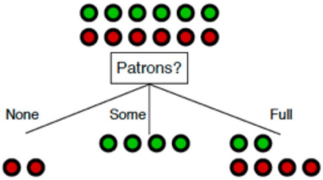
- Information Gain $I(X;Y) = 1 - 0.45 = 0.55$ bits

Information Gain if we split "Type"



- $H(Y) = -\frac{6}{12}\log\frac{6}{12} - \frac{6}{12}\log\frac{6}{12} = 1$ bit
- $H(Y|X = french) = \log 2 = 1$ bit
- $H(Y|X = italian) = \log 2 = 1$ bit
- $H(Y|X = thai) = \log 2 = 1$ bit
- $H(Y|X = burger) = \log 2 = 1$ bit
- Thus the conditional entropy is
  $H(Y|X) = \frac{2}{12} \times 1 + \frac{2}{12} \times 1 + \frac{4}{12} \times 1 + \frac{4}{12} \times 1 = 1$ bit
- Information Gain $I(X;Y) = 1 - 1 = 0$ bits

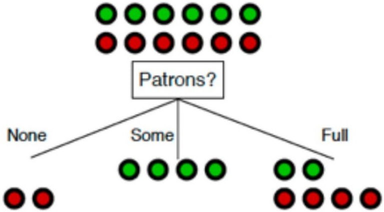
Splitting on “Patron” or “Type”?



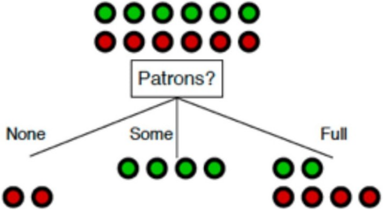
- Information gain from “Patron” is 0.55 bits.
- Information gain from “Type” is 0 bits.

Thus, we should split on “Patron” and not “Type” (higher information gain). This is consistent with our intuition.

Next split?



Do we split on "None" or "Some"?



- No, we do not
- The decision is deterministic, as seen from the training data.

Next split

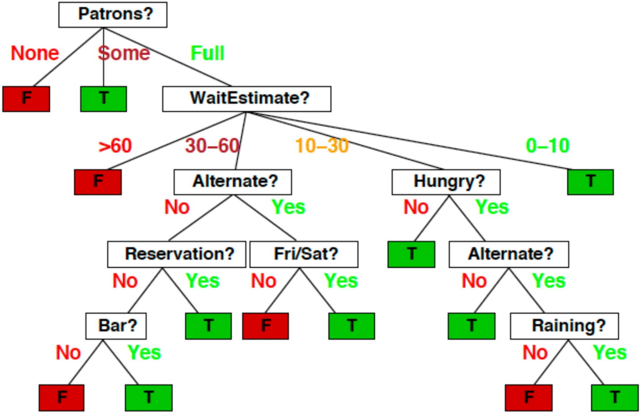
Next split?

Example	Attributes										Target
	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)

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

Greedily we build the tree and get this



What is the optimal Tree Depth?

- What happens if we pick the wrong depth?
  - If the tree is too deep, we can overfit
  - If the tree is too shallow, we underfit
- Max depth is a hyperparameter that should be tuned by the data
- Alternative strategy is to create a very deep tree, and then to prune it (see Section 9.2.2 in ESL for details)

Cost Complexity Pruning

Pruning means collapsing non-terminal nodes to eliminate a split.

Cost complexity criterion

$$C_\alpha(T) = \sum_{m=1}^{|T|} \text{error}_m(T) + \alpha|T|$$

- Find the tree T that minimizes the cost $C_\alpha(T)$, where $m = 1, 2, \ldots, |T|$ indexes the leaf nodes.
- Measure error of the training data at each leaf node as before (misclassification rate, squared error for linear regression).
- Choose α as a hyperparameter (similar to regularization).

To find the tree that minimizes C_α, greedily collapse the node in the full tree that increases the error rate the least.

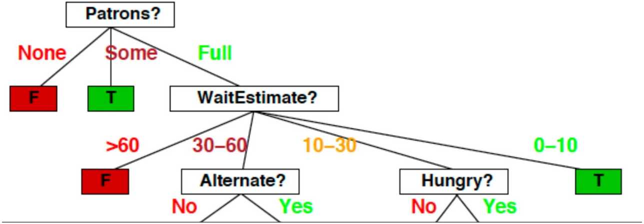
Overfitting in Decision Trees

- Including irrelevant attributes can result in overfitting the training example data.
- If we have too little training data, even a reasonable hypothesis space will overfit.

Strategies to avoid overfitting

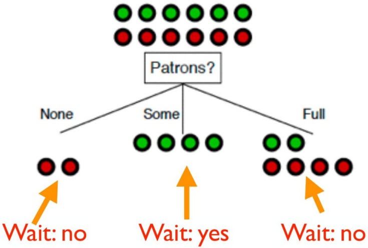
- Stop growing when data split is not statistically significant.
- Acquire more training data.
- Remove irrelevant attributes (manual process — not always possible).
- Grow full tree, then post-prune (e.g., cost complexity)

How to classify with a pruned decision tree?



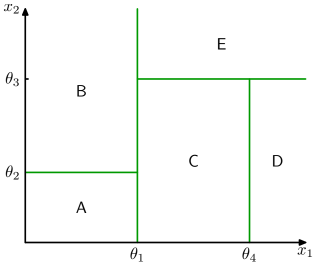
- If we stop here, not all training samples would be classified correctly
- More importantly, how do we classify a new instance?
- We label the leaves of this smaller tree with the majority of training sample's labels.

Example



Computational Considerations: Numerical Features

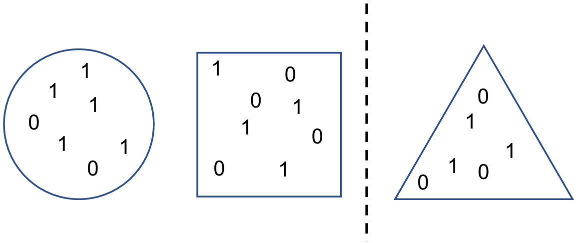
- How should we decide the threshold to use in splitting the feature?
- Can we do this efficiently?
    - Yes – for a given feature we only need to consider the n values in the training data!
    - If we sort each feature by these n values, we can quickly compute and maximize the information gain along each possible threshold.
    - This takes $O(dn \log n)$ time, where d is the number of features



$$x_1^{(1)} \quad x_1^{(2)} \quad x_1^{(3)} \;\Big|^{\theta_5}\; x_1^{(4)} \quad \ldots \quad x_1^{(n)}$$

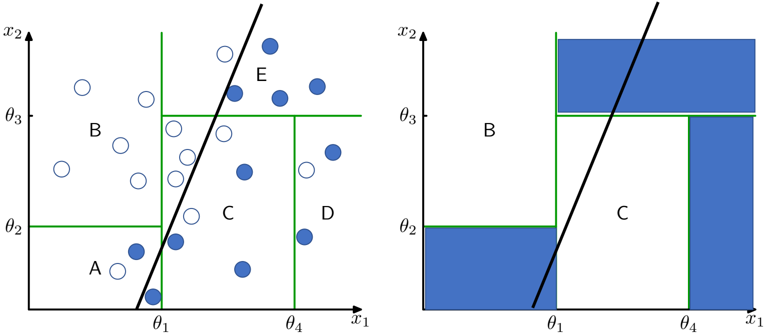
Computational Considerations: Categorical Features

- Assuming q distinct categories, there are $\frac{1}{2}(2^q - 2) = 2^{q-1} - 1$ possible partitions
- Things simplify in the case of binary classification or regression
  - Can sort the features by the fraction of labels falling in class 1
  - Suffices to consider only $q - 1$ possible splits (see Section 9.2.4 in ESL)
- Example: suppose we have two labels (0 or 1) and the feature is "shape," which has three categories (circle, square, or triangle).



Disadvantages of Decision Trees

- Binary decision trees find it hard to learn linear boundaries.
- Decision trees can have high variance due to dependence on the training data.
- We use heuristic training techniques: finding the optimal partition is NP-hard.



Advantages of Decision Trees

- Can be interpreted by humans (as long as the tree is not too big)
- Computationally efficient (for shallow trees)
- Handles both numerical and categorical data
- Can be used for both classification and regression
- Compact representation: unlike Nearest Neighbors we don't need training data at test time
- But, like NN, decision trees are nonparametric because the number of parameters depends on the data

Summary of Decision Trees

You should know:

- Motivation for considering decision trees
- How to construct a decision tree
- Techniques for ensuring the tree does not overfit
- Disadvantages of decision tree methods

Decision trees are a common building block for various ensemble methods (more on this next lecture).