18-661 Introduction to Machine Learning

Decision Trees

Spring 2025

ECE – Carnegie Mellon University

Announcements

- Homework 3 has been released. It is due on March 28.
- Midterm coversheet with detailed instructions and a question outline will be released by tomorrow night. You may bring a one page, double-sided, letter or A4 size, handwritten, physical cheat sheet to the exam.
- Midterm exam will be in lecture for 110 minutes.
- No recitation this Friday.

Concepts You Should Know

This is a quick overview of important concepts/methods/models.

- MLE/MAP: how to find the likelihood of one or more observations, how to use a prior distribution, how to optimize the likelihood
- Linear regression: formulation, how it relates to MLE/MAP, feature scaling, ridge regression, gradient descent, nonlinear basis functions
- Bias-variance trade-off: bias and variance, overfitting, cross-validation
- Naive Bayes: naive classification rule, why it is naive, Laplacian smoothing
- Logistic regression: generative vs. discriminative models, formulation, how it relates to MLE, comparison to naive Bayes, sigmoid function, cross-entropy function, nonlinear boundaries
- Multi-class: one-vs-all and one-vs-one approaches, softmax function/multinomial logistic regression
- SVMs: hinge loss formulation, max-margin formulation, dual of the SVM problem, kernel functions
- Nearest neighbors: parametric vs. nonparametric models, k-nearest neighbors, tuning hyperparameters, feature scaling

Outline

1. Review: Nearest Neighbors
2. Decision Trees: Motivation
3. Learning A Decision Tree
4. Practical Considerations for Decision Trees

Review: Nearest Neighbors

K-Nearest Neighbor (KNN) Classification

- 1st-nearest neighbor: $\text{nn}_1(\boldsymbol{x}) = \text{argmin}_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 2nd-nearest neighbor: $\text{nn}_2(\boldsymbol{x}) = \text{argmin}_{n \in [\text{N}] - \text{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 3rd-nearest neighbor: $\text{nn}_3(\boldsymbol{x}) = \text{argmin}_{n \in [\text{N}] - \text{nn}_1(\boldsymbol{x}) - \text{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

The set of K-nearest neighbors

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

How to Classify with K Neighbors?

- Aggregate everyone's vote

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

- Label with the majority, breaking ties arbitrarily

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

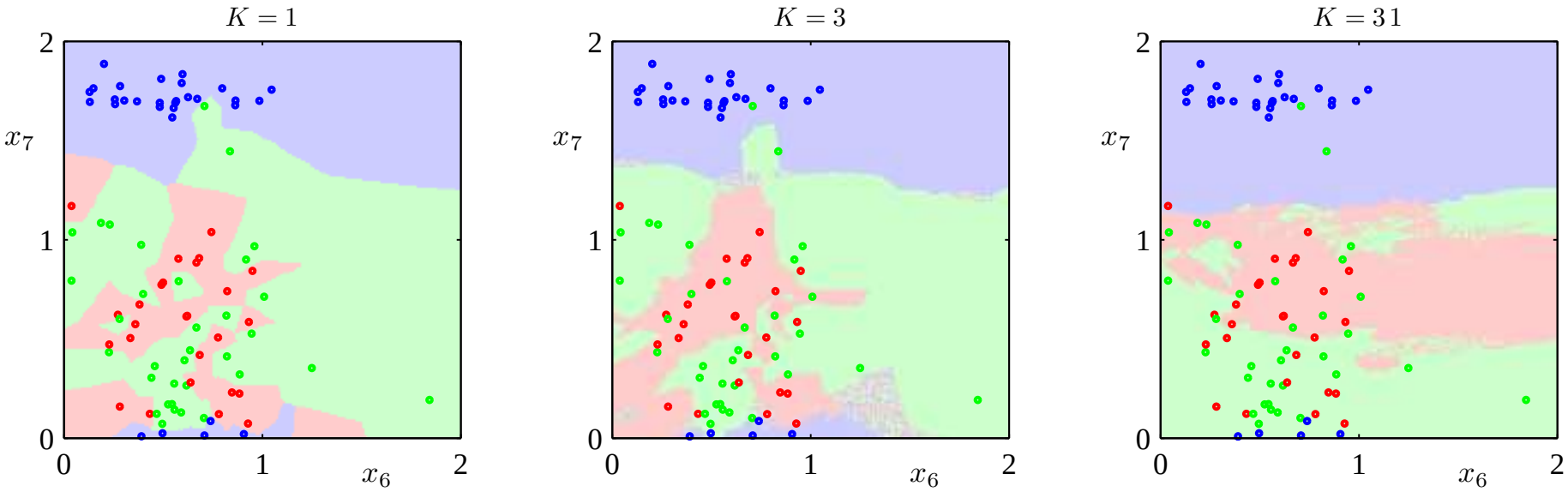
Hyperparameters in NN

Two crucial choices for NN

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure: Euclidean distance, L_1 or L_p distance, feature scaling, ...

In practice, these are hyper parameters that need to be tuned by a validation dataset or cross validation.

How to Choose an Optimal K?



- When K increases, the decision boundary becomes smoother and less susceptible to outliers.
- However, if K is too large, it can also lead to misclassification as we are taking votes from faraway training points.

Why Use Nearest Neighbors?

Advantages of NNC

- Simple and easy to implement – just compute distances, no optimization required
- Can learn complex decision boundaries

Disadvantages of NNC

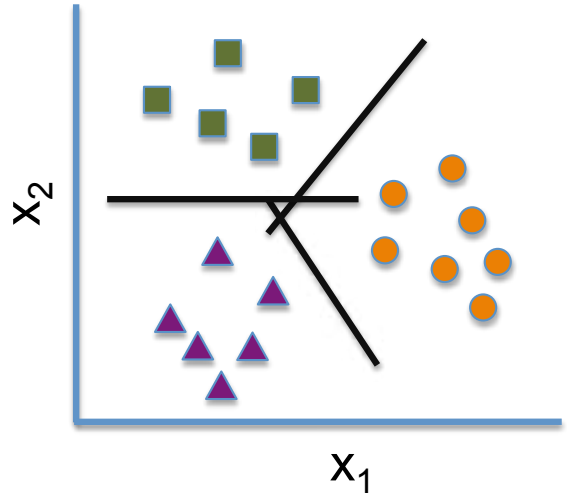
- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a data point.
- We need to "carry" the training data around. Without it, we cannot do classification.
- Choosing the right distance measure and K can be difficult.
- Relies on the existence of training data points "close" to test points.
- Can break down if the classes are unbalanced.

Outline

1. Review: Nearest Neighbors
2. Decision Trees: Motivation
3. Learning A Decision Tree
4. Practical Considerations for Decision Trees

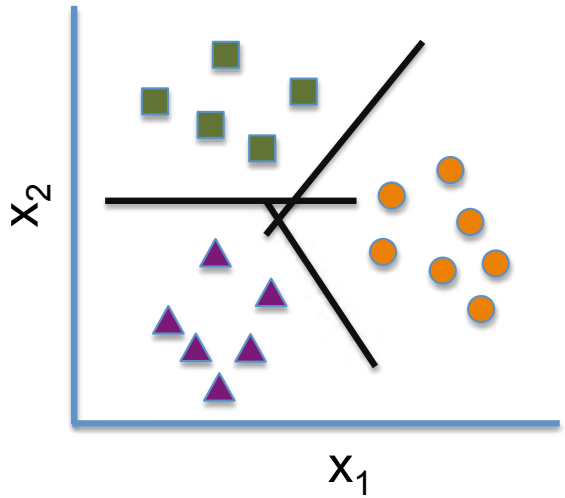
Decision Trees: Motivation

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.

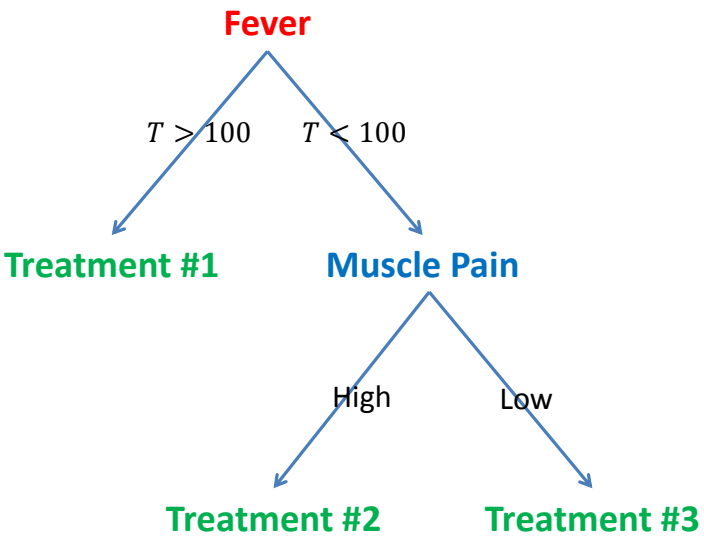
Need Interpretable Decision Boundaries



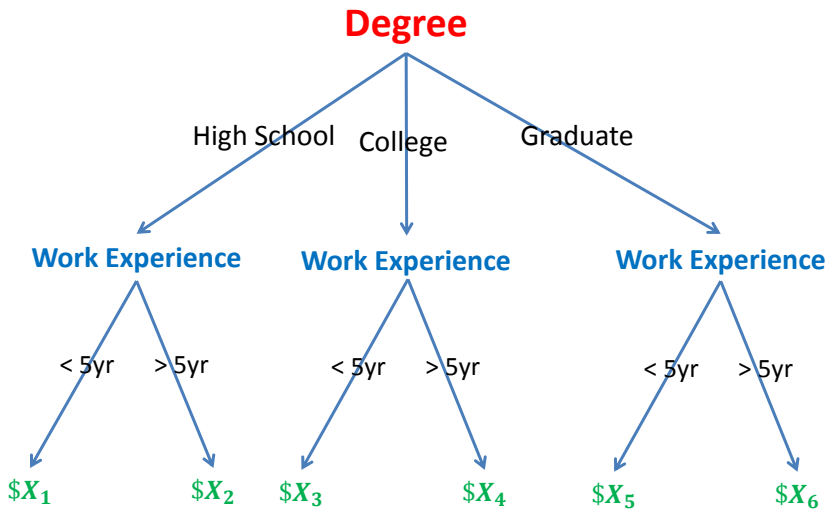
- Should be able to explain the reasoning in clear terms, e.g., "I always recommend treatment 1 when a patient has fever $\geq 100F$"
- The rules that you use to make decisions can be easily used by a lay-person without performing complex computations
- Decision trees can provide such simple decision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

Many Decisions Are Tree-Structured

Medical treatment



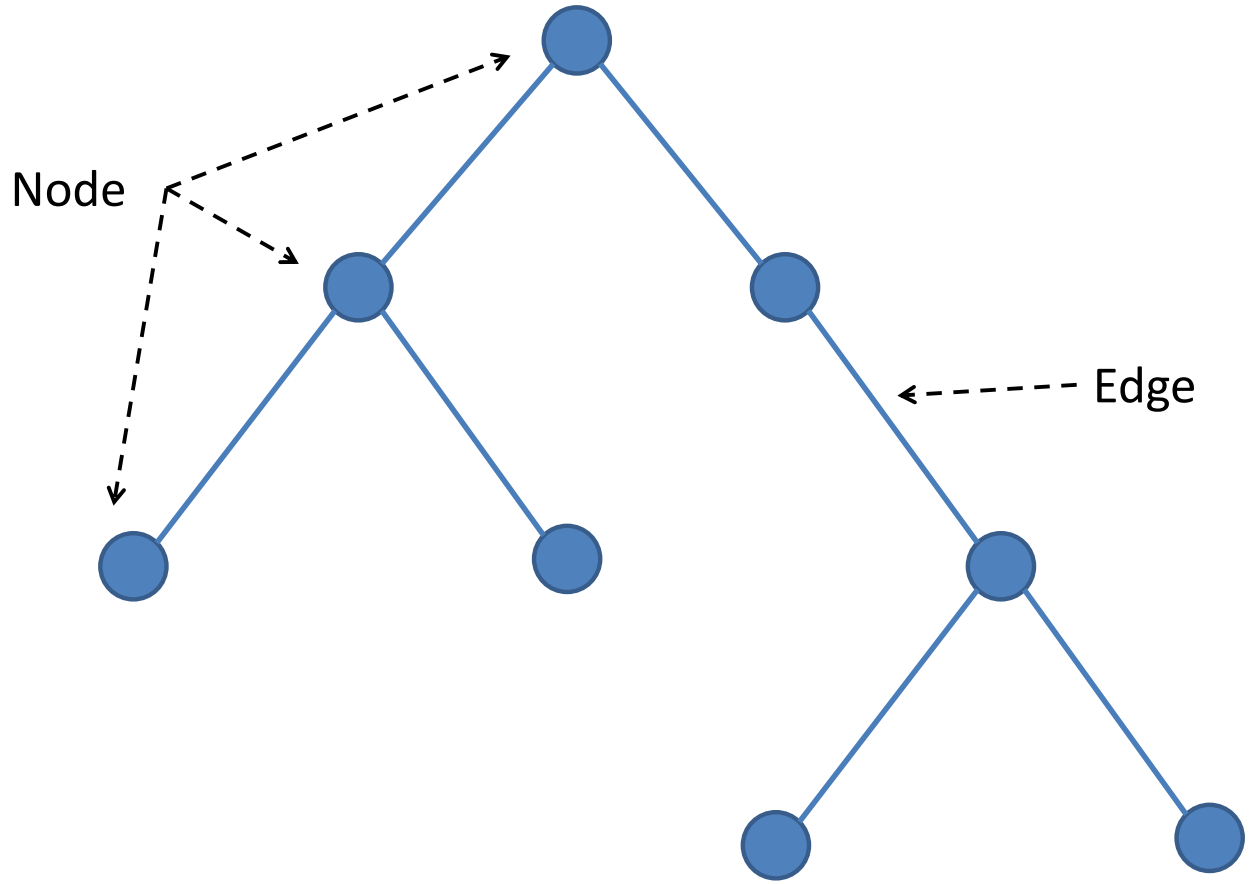
Salary in a company



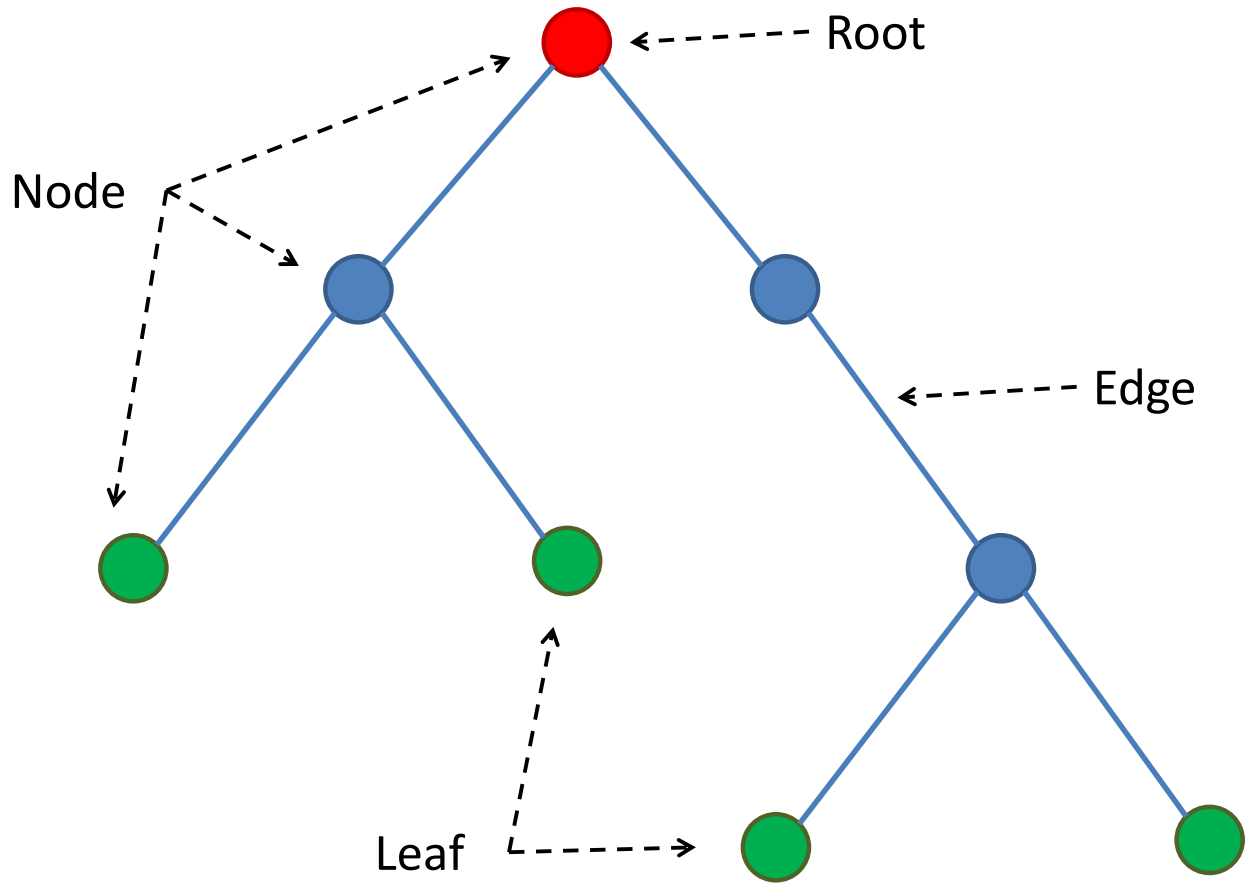
Other examples: fault detection in manufacturing systems, student admissions decisions, jail/parole decisions

Note: here we have a mix of numerical features (temperature) and categorical features (degree) - both are allowed in decision trees.

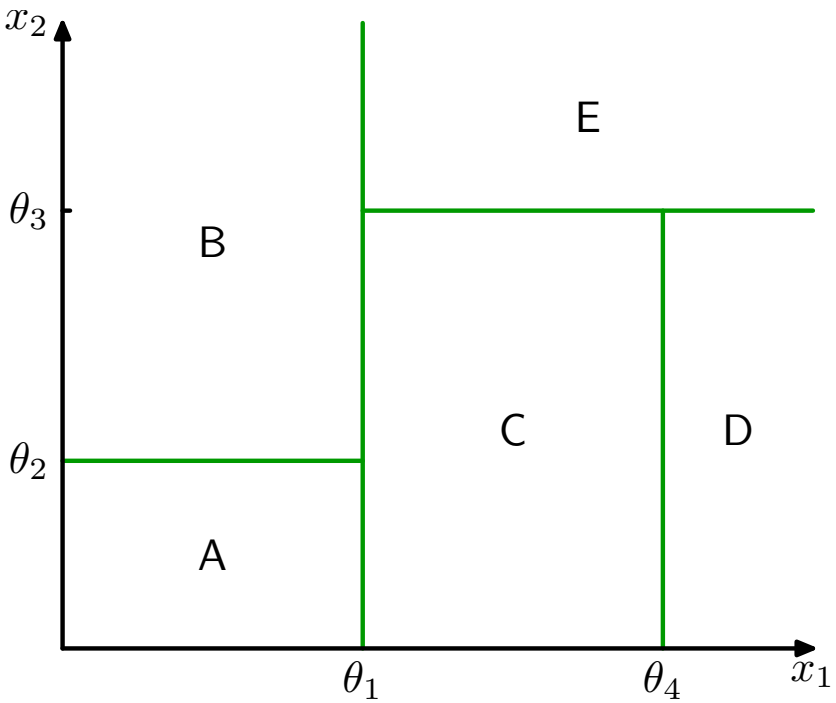
What Is a Tree?

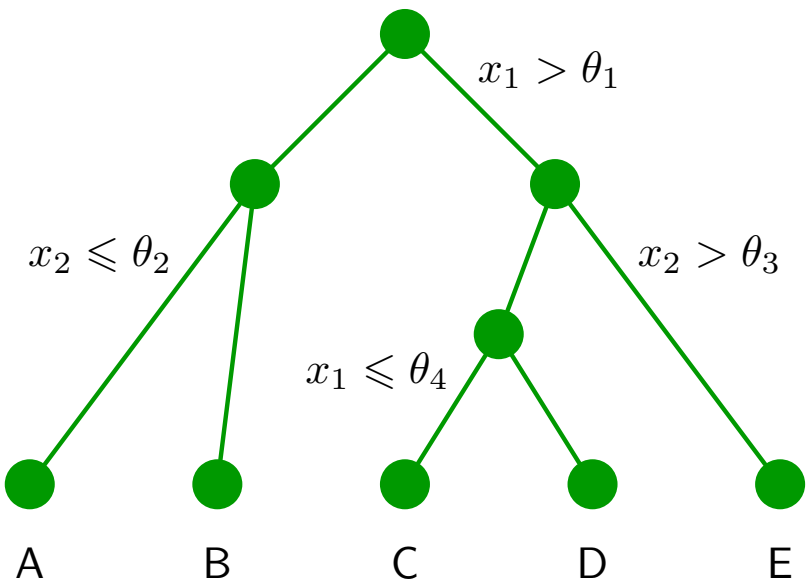


Special Names for Nodes in a Tree



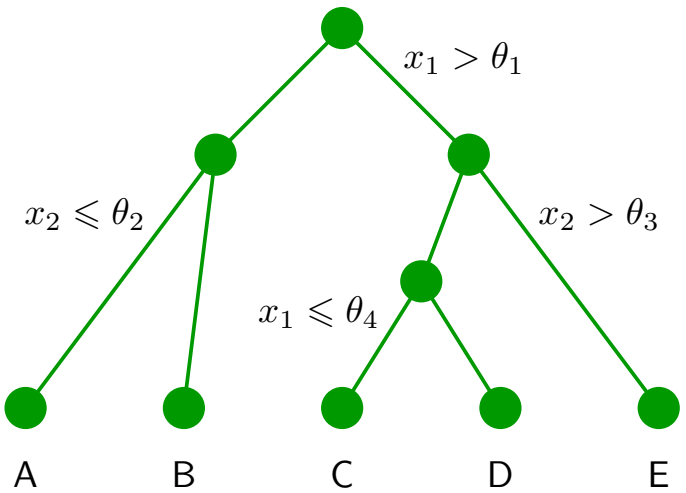
A Tree Partitions the Feature Space





Learning A Decision Tree

Learning a Tree Model



Things to learn:

1. The structure of the tree.
2. The "split" rule at each node.
   - Which feature to consider?
   - What is the threshold θ_i?
   - What about categorical features?
3. The predicted values for the leaves $(A, B, \ldots)$.

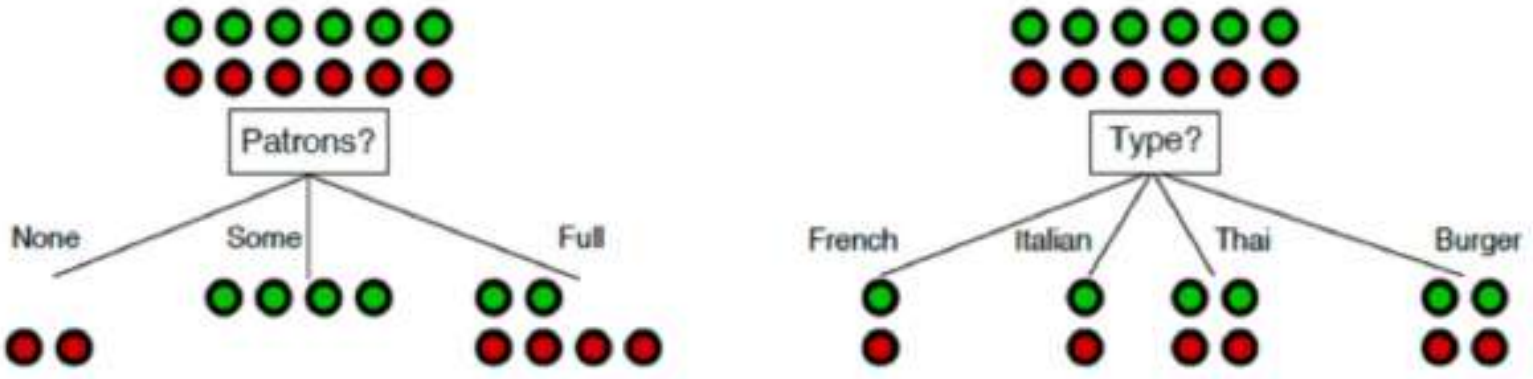
Up-next: an example where all the features are categorical. Will talk about numerical features later.

Example: Choosing Whether 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 $I(X; Y)$?

Gaining information is equivalent to reducing our uncertainty.

We use entropy $H(Y)$ to measure uncertainty in Y.

Definition (Entropy)
If a random variable Y takes K different values, a_1, $a_2 \ldots a_K$, 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 $p = 0.5$

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

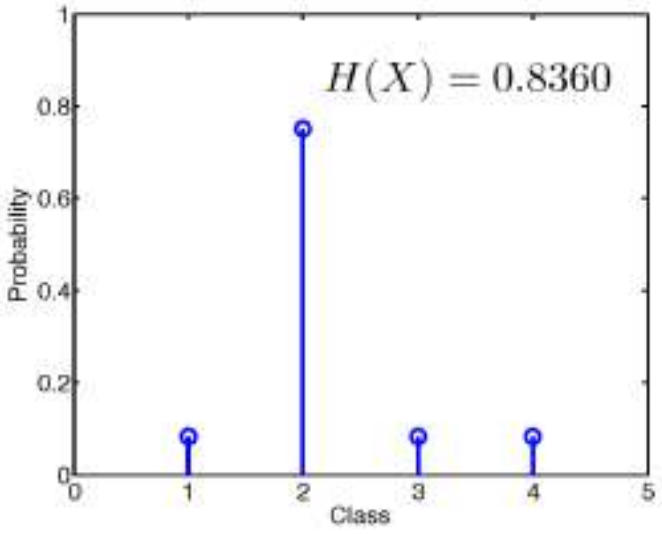
- For $p = 0.25$

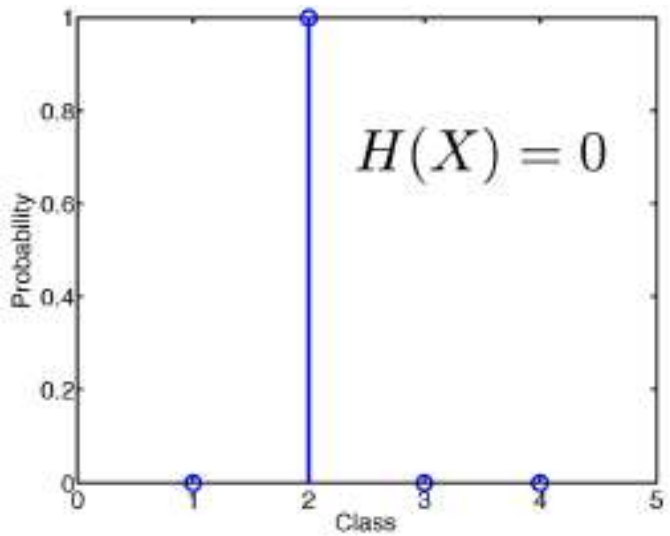
  $$H(Y) = -(0.25 \log 0.25 + 0.75 \log 0.75) = 2 \log 2 - 0.75 \log 3 \approx 0.81 \text{ bits}$$

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

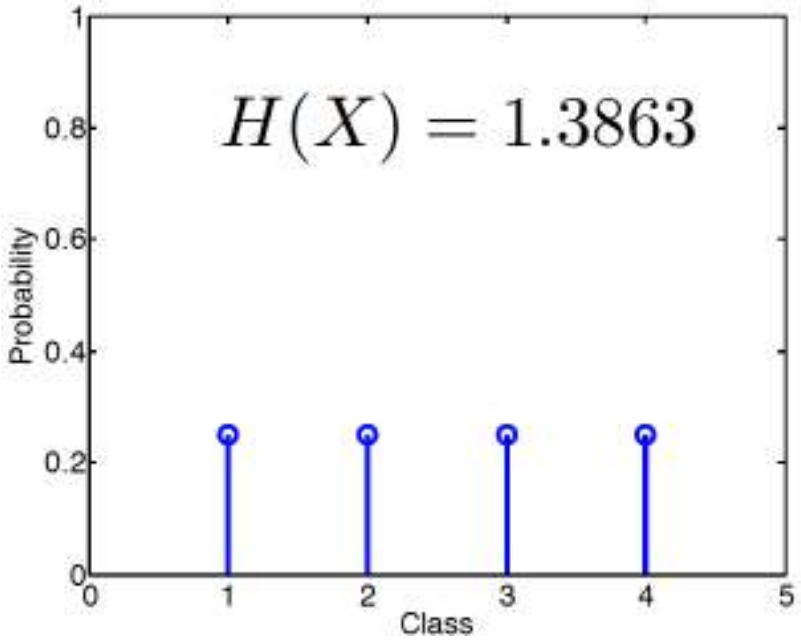
With more uncertainty ($p = 0.5$), we have a larger entropy.

Illustrating Entropy





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



Conditional Entropy

In our restaurant example:

- Y: wait or not (the labels)
- X: patron (none/some/full)

We defined $H(Y)$, the uncertainty in Y. We also wanted to characterize "if know X, the uncertainty in Y is reduced."

Definition (Conditional Entropy)
Given two random variables X and Y

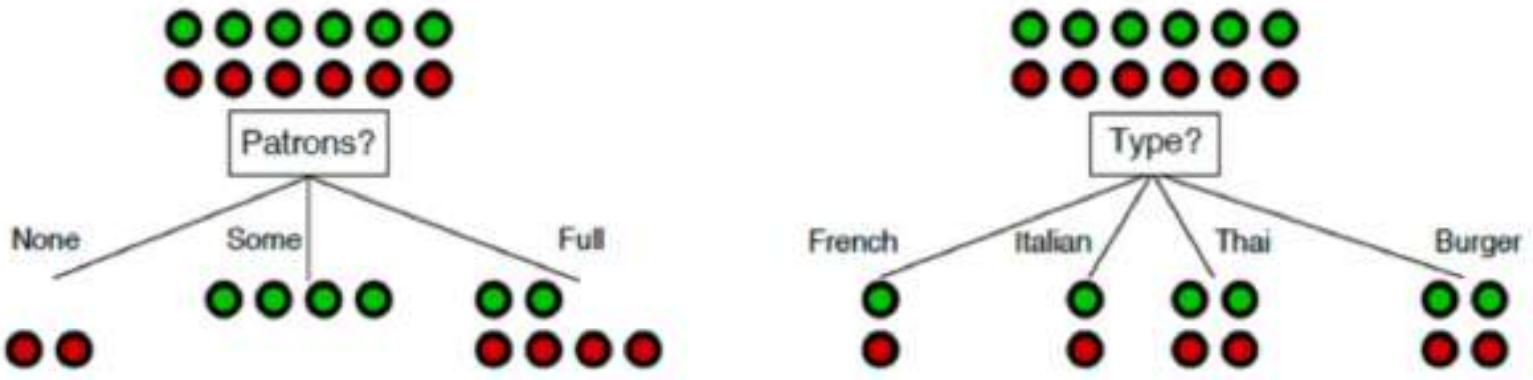
$$H(Y|X) = \sum_k P(X = a_k) H(Y|X = a_k)$$

Definition (Information Gain)

$$I(X; Y) = H(Y) - H(Y|X)$$

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

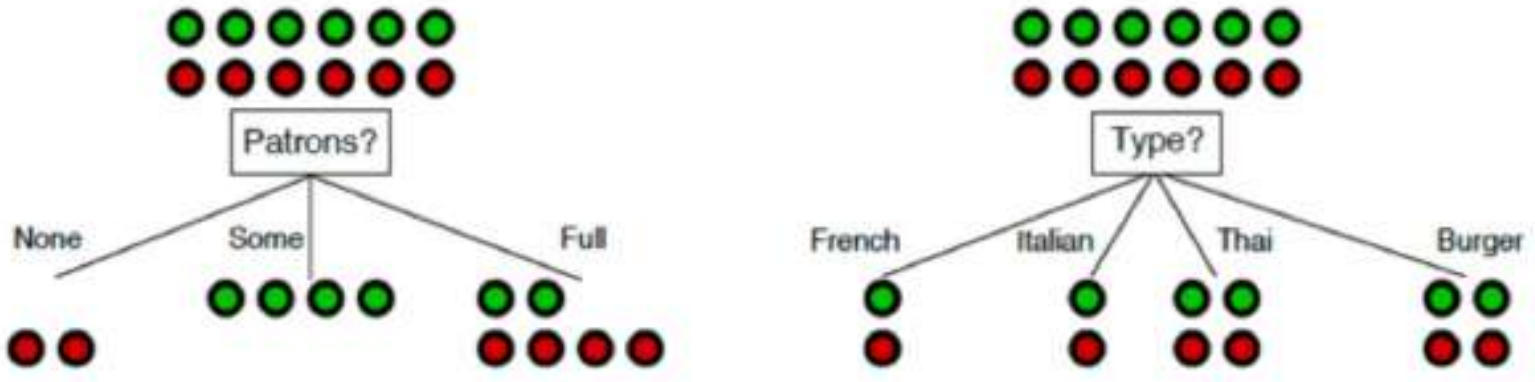
Which Attribute to Split?



Patron vs. Type?

- Let us compute the information gain $I(X; Y) = H(Y) - H(Y|X)$ for Patron and Type

Information Gain if We Split "Patron"

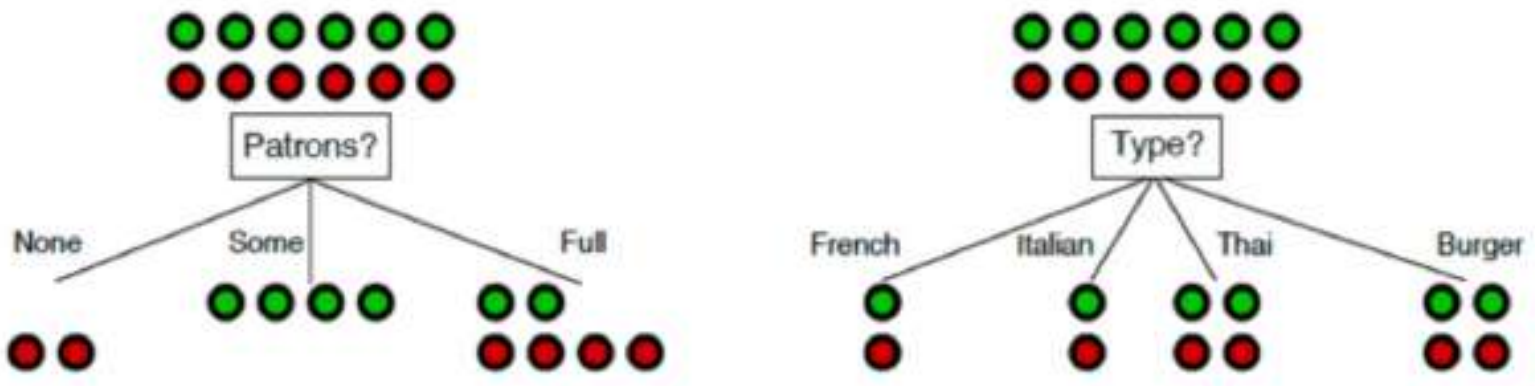


- $H(Y) = -\frac{6}{12}\log\frac{6}{12} - \frac{6}{12}\log\frac{6}{12} = 1$ bit
- $H(Y|X = none) = 0$
- $H(Y|X = some) = 0$
- $H(Y|X = full) = -\left(\frac{2}{2+4}\log\frac{2}{2+4} + \frac{4}{2+4}\log\frac{4}{2+4}\right) \approx 0.9$ bits
- Thus the conditional entropy is

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

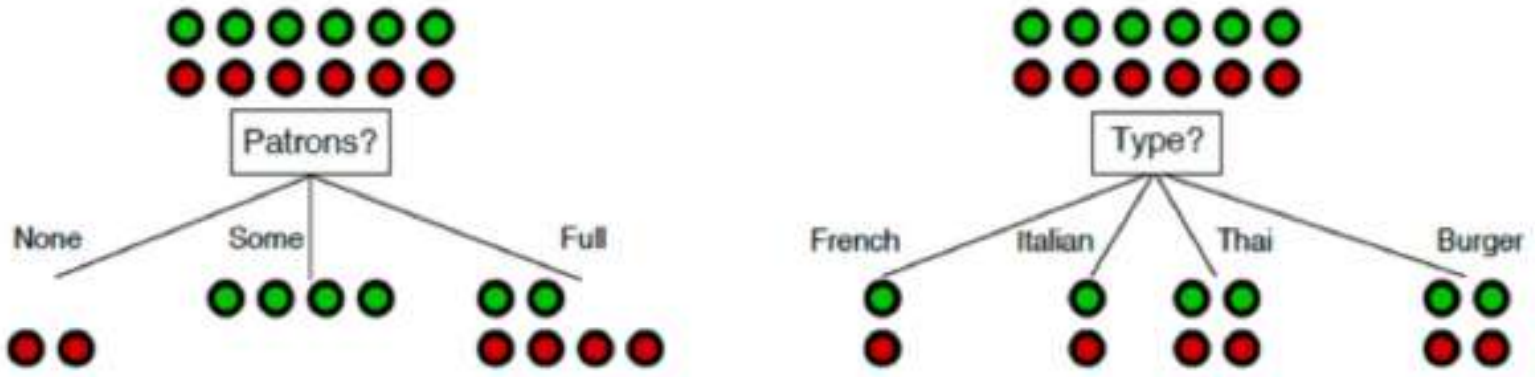
- Information Gain $I(X; Y) = 1 - 0.45 = 0.55$ bits

Information Gain if We Split "Type"



- $H(Y) = -\frac{6}{12}\log\frac{6}{12} - \frac{6}{12}\log\frac{6}{12} = 1$ bit
- $H(Y|X = \textit{french}) = \log 2 = 1$ bit
- $H(Y|X = \textit{italian}) = \log 2 = 1$ bit
- $H(Y|X = \textit{thai}) = \log 2 = 1$ bit
- $H(Y|X = \textit{burger}) = \log 2 = 1$ 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

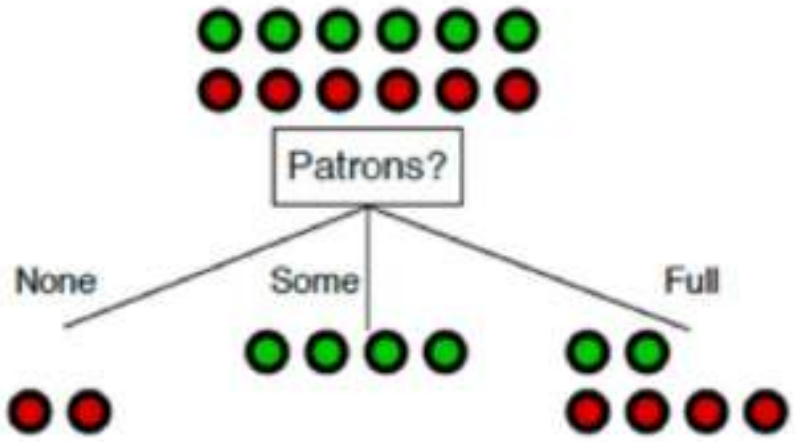
Splitting on "Patron" or "Type"?



- 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.

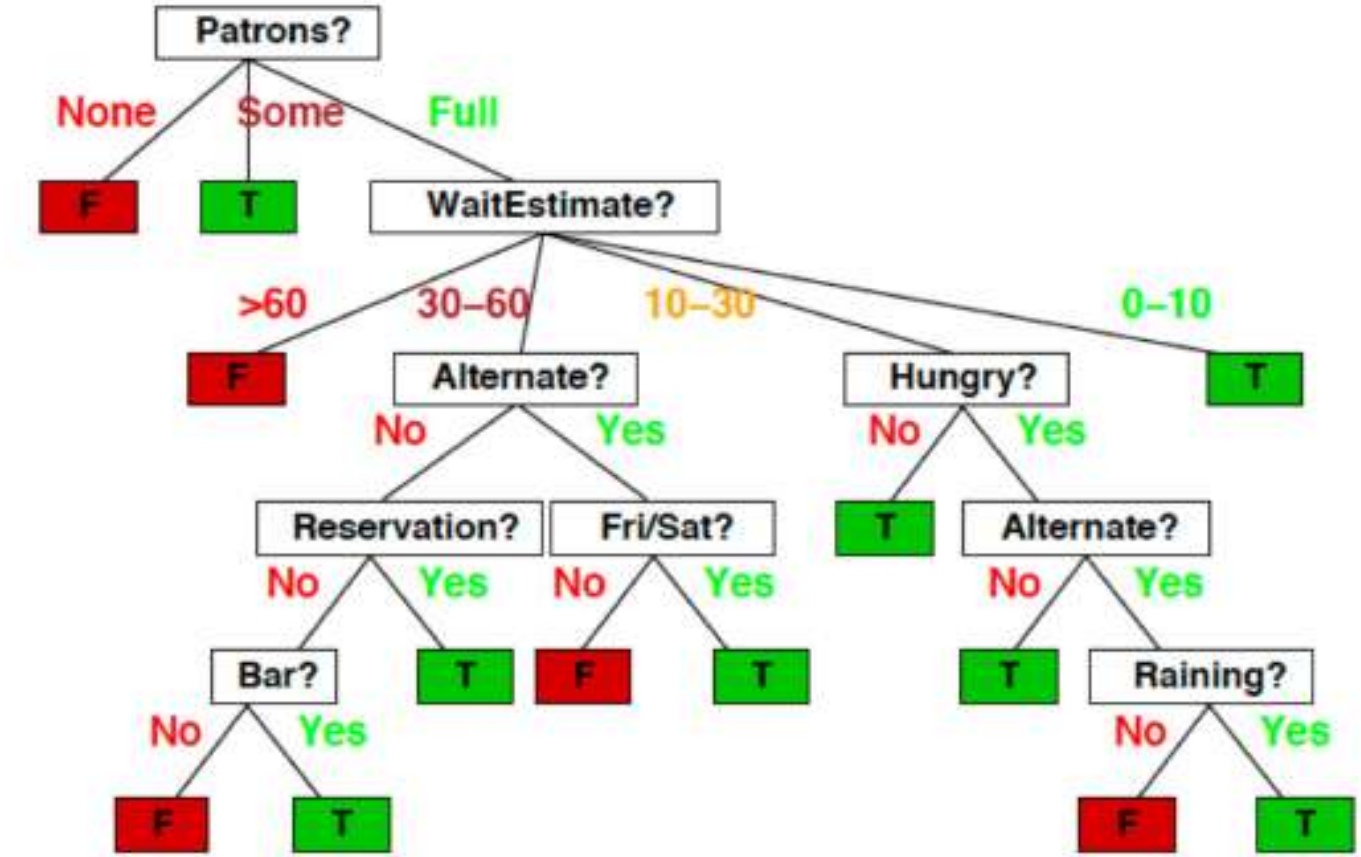
Next Split?



Do We Split on "None" or "Some"?

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

Greedily We Build the Tree and Get...

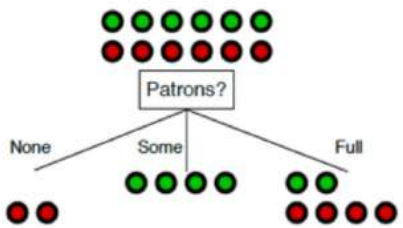


- We do not further split a node if
  - All the samples under the node have (almost) the same category, OR
  - The maximum depth has been reached.
- We finish when no more nodes can be split.

Learning a Tree Model

Three things to learn:

1. The structure of the tree.
   - Keep splitting until no more nodes can be split.
   - Here, a node cannot be split if the samples are (almost) the same label, OR maximum depth reached. How to determine max depth?
2. The split rule at each node.

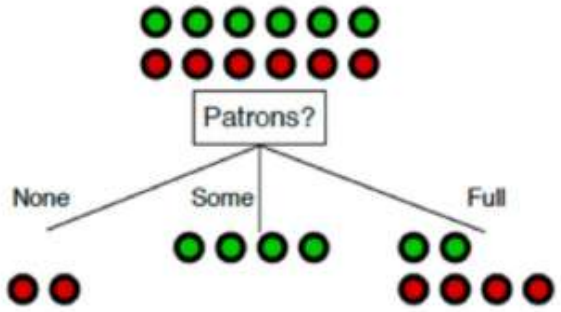


   - Pick the feature with highest information gain.
   - Create a branch for each category. What if too many categories?
   - What if features are continuous?
3. The values for the leaves $(A, B, \ldots)$.
   - Obvious if all training samples in the leaf hav the same label.
   - Otherwise? Majority vote based on the samples in the leaf.

Practical Considerations for Decision Trees

Often, We Use Binary Decision Trees

In our example, we split each node among all categories.

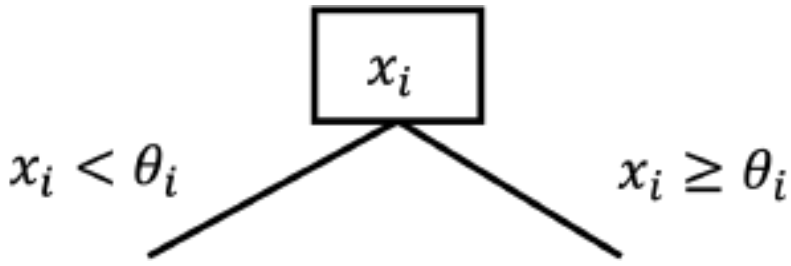


This may overfit with a large number of categories!

- Many implementations only allow two outcomes for each split.
- Binary decision trees can capture multiple splits by adding an additional level of splits. (e.g., first split on "none" and "some + full", then split the "some + full" node on "some" and "full').

Under the binary tree setting, how to determine the splitting rule for numerical and categorical variables?

Splitting Numerical Features



Recall when splitting, we select a feature with the highest information gain.

For a numerical feature x_1, how to calculate the information gain, and how to determine the splitting threshold θ_1 if x_1 is selected?

Splitting Numerical Features

For a numerical feature x_1, how to calculate the information gain, and how to determine the splitting threshold θ_1 if x_1 is selected?

- Suppose there are n training samples under this node, sorted in increasing order. We only need to consider $n-1$ possible thresholds.

$$x_1^{(1)} \quad x_1^{(2)} \quad x_1^{(3)} \;\Big|^{\theta_1}\; x_1^{(4)} \quad \dots \quad x_1^{(n)}$$

- For each possible threshold θ_1, we can calculate information gain:

$$H(Y) - H(Y|\text{whether } X_1 \text{ is greater than } \theta_1 \text{ or not})$$

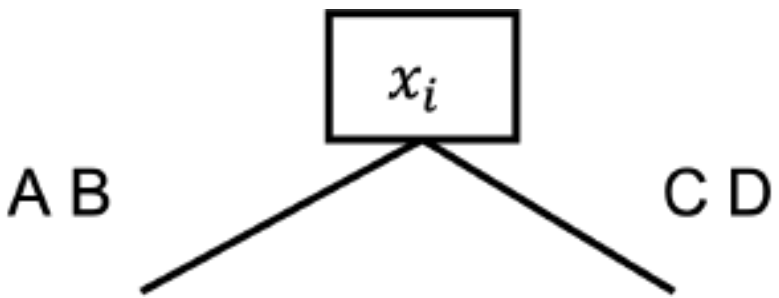
- Find the θ_1^* that achieves the highest information gain among above. This will also be the information gain of feature x_1.
- If x_1 is selected, then the threshold will be θ_1^*.
- This takes $O(dn \log n)$ time, where d is the number of features

Splitting Categorical Features

Assuming q distinct categories, there are $\frac{1}{2}\left(2^q - 2\right) = 2^{q-1} - 1$ possible partitions!

Example: 4 categories A, B, C, D, the possible ways of partition into two groups: A vs BCD, AB vs CD, AC vs BD, AD vs BC, ABC vs D, ...

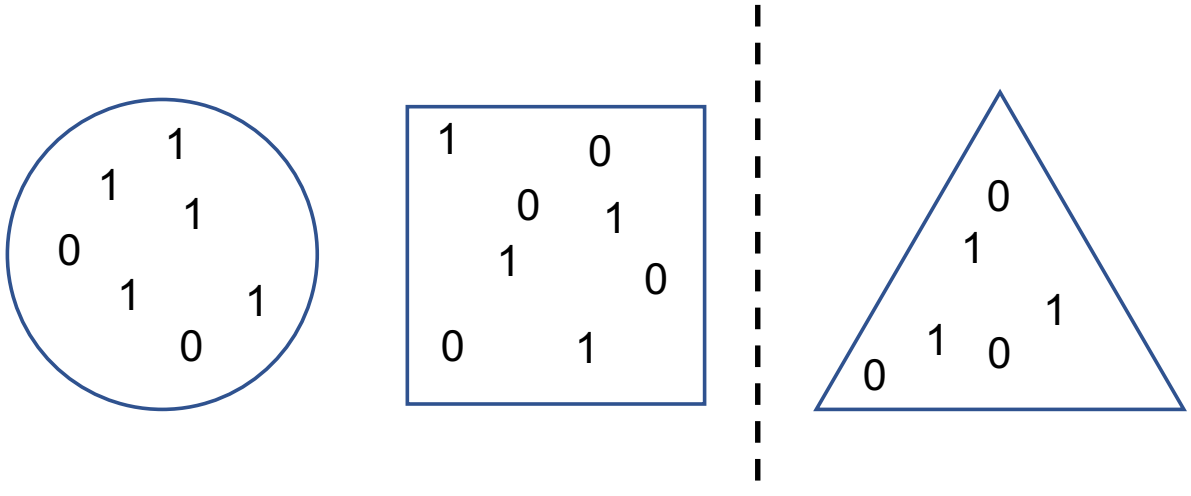
How to calculate information gain and determine the partition for a categorical variable?



Splitting Categorical Features

For binary classification, we can use the following heuristics.

- Can sort the categories by the fraction of labels falling in class 1
- Example: Suppose we have two labels (0 or 1) and the feature is "shape," which has three categories (circle, square, or triangle). We arrange the three by the fraction of "1" samples.



Suffices to consider only $q - 1$ possible partitions. Calculate information gain and choose a partition just like with a numerical feature (see Sec 9.2.4 in ESL).

Learning a Tree Model

1. The structure of the tree.
   - Keep splitting until no more nodes can be split.
   - Here, a node cannot be split if the samples are (almost) the same label, OR maximum depth reached. How to determine max depth?
2. The split rule at each node for *binary tree* setting.
   - Pick the feature with highest information gain.
   - *Numerical feature:* maximize information gain over $n - 1$ possible thresholds and d possible features. This process will determine both the information gain for this feature, and the associated threshold.
   - *Categorical feature:* Arrange the categories in an "order", and similar procedures apply.
3. The prediction for the leaves.
   - Obvious if all training samples in the leaf has the same label.
   - Otherwise? Majority vote based on the samples in the leaf.

What Is the Optimal Tree Depth?

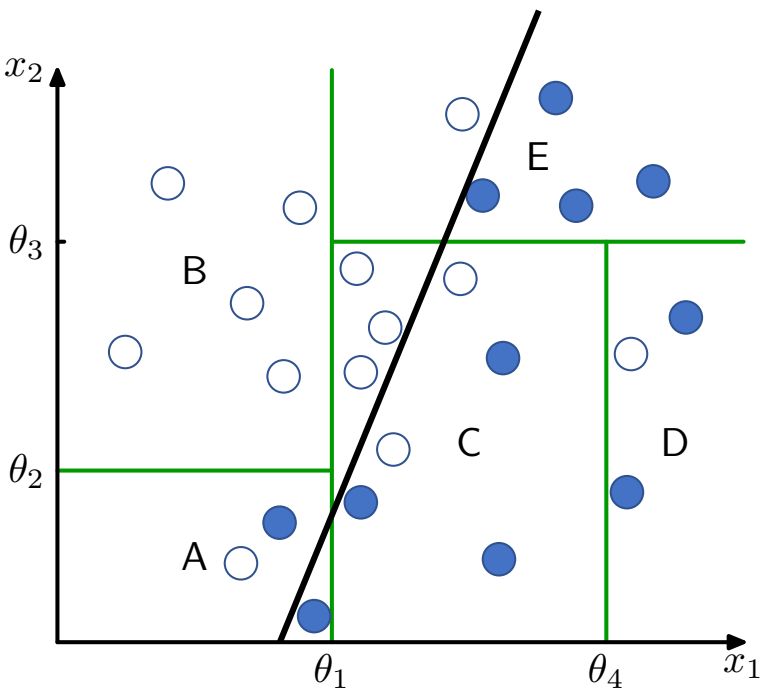
- What happens if we pick the wrong depth?
  - If the tree is too deep, we can overfit
  - If the tree is too shallow, we underfit
- Max depth is a hyperparameter that should be tuned by the data
- Alternative strategy is to create a very deep tree, and then to prune it (see Section 9.2.2 in ESL for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

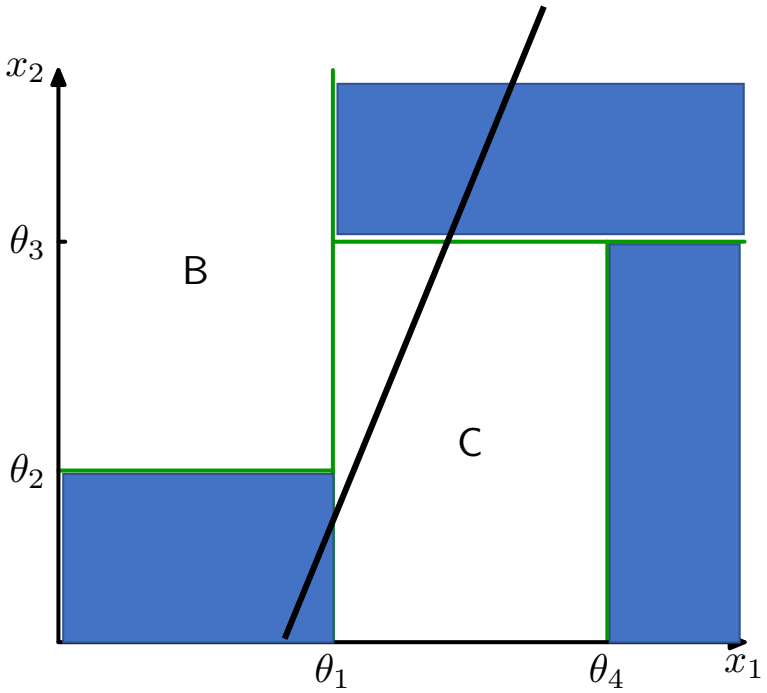
Summary: Advantages of Decision Trees

- Can be interpreted by humans (as long as the tree is not too big)
- Computationally efficient (for shallow trees)
- Handles both numerical and categorical features.
- Can be used for both classification and regression
- Like Nearest Neighbors, decision trees are nonparametric because we do not try to learn a fixed parameter vector. The number of parameters (which define the structure of the tree) depends on the training data.
- Unlike Nearest Neighbors we don't need training data when making predictions.

Summary: Disadvantages of Decision Trees

- Binary decision trees find it hard to learn linear boundaries.
- Decision trees are prone to overfitting!





Overfitting in Decision Trees

Recall: overfitting means model too complex, fits into "unnecessary noise" of the training data; small training error, high test error.

- A deep tree (thus high model complexity) is prone to overfitting.
- Including irrelevant attributes can result in overfitting.
- If we have too little training data, even a shallow tree 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.
- Use tree ensembles (random forest, boosting, to be covered next lecture).

You Should Know

- How to construct a decision tree
- How to use information gain to decide a decision tree split
- Advantages and disadvantages of decision tree models