Decision Trees

Lecture 10

Last Time

- ***Boosting*** is an algorithmic framework for extending a “base” hypothesis class into a more complex one

- ***AdaBoost*** (adaptive boosting) learns an ensemble of base hypotheses that vote to make predictions. Its complexity is only limited by the ensemble size.

- Textbook: chapter 10

This Class

- A new class of hypotheses: decision trees

Textbook: chapter 18

Motivation

Will a Cat Adoption Be Successful?

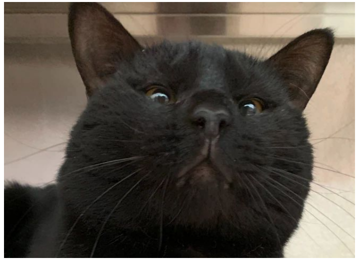
- Lots of attributes can affect matching cats with owners

- Attributes:
  - Owner home all day?
  - Cat is kitten?
  - Cat spayed/neutered?

Clyde

Pumpkin Spice

Johnny Bravo



Black Beard

https://rispca.com/

Example Data

Owner home?	**Kitten?**	**Spay/Neuter?**	**Success?**
True	True	False	***TRUE***
False	False	True	***TRUE***
True	True	True	***TRUE***
False	True	False	***FALSE***
True	False	True	***FALSE***
False	True	True	***FALSE***

Example Data

Best linear predictor?

If Owner Home = True
Then Success = True,

Else Success = False

(2/6 training error)

Owner home?	**Kitten?**	**Spay/Neuter?**	**Success?**
True	True	False	***TRUE***
False	False	True	***TRUE***
True	True	True	***TRUE***
False	True	False	***FALSE***
True	False	True	***FALSE***
False	True	True	***FALSE***

Example Data

What about more rules?

If Owner Home = True
  And Kitten = True
Then Success = True,

Or if Owner Home = False
  And Kitten = False
Then Success = True,

Else Success = False

(0/6 training error)

Owner home?	Kitten?	Spay/Neuter?	Success?
True	True	False	***TRUE***
False	False	True	***TRUE***
True	True	True	***TRUE***
False	True	False	***FALSE***
True	False	True	***FALSE***
False	True	True	***FALSE***

Advantages

- Precise rules
  - (Owner Home ***and*** Kitten) ***or*** (***not*** Owner Home ***and not*** Kitten)
  - Outputs True only if all the conditions in either set are met

- Interpretability
  - Do our rules make sense?

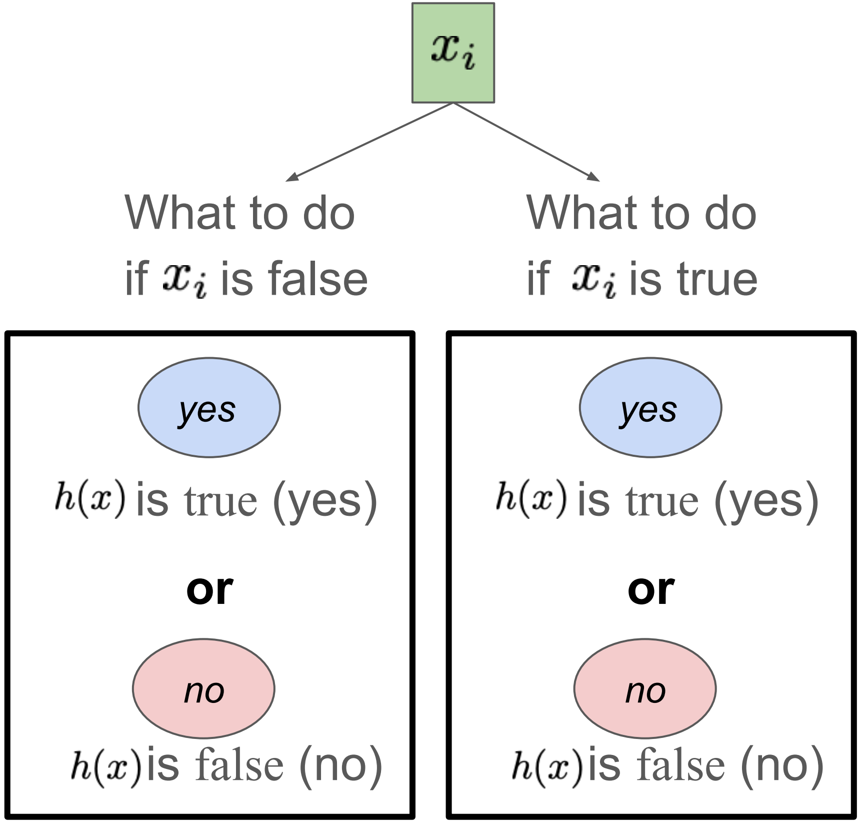
Challenges

- We might need a lot of rules to describe the training data
  - How do we organize the rules?

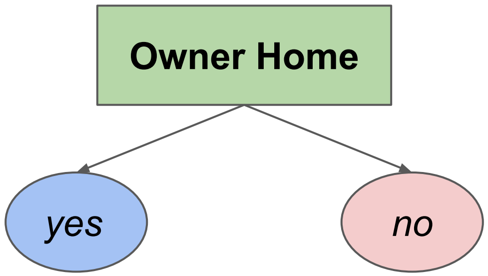
- It might be hard to find exactly those precise rules
  - How do we learn when our hypothesis class is discrete (no gradients)?

Decision Trees

Decision Stump



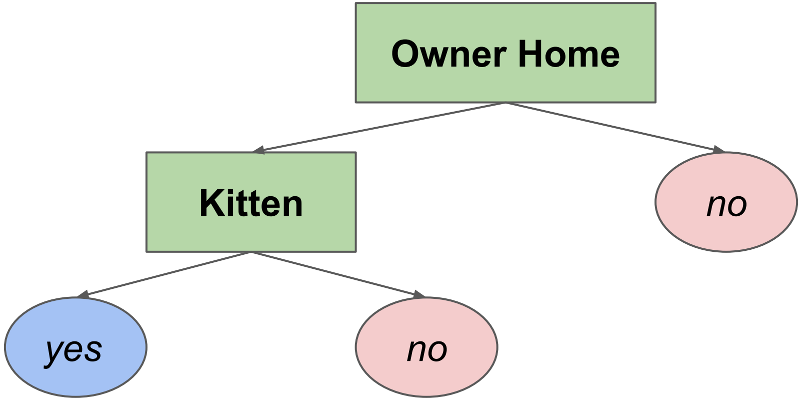
Decision Stump



Try this one. What rule is this?

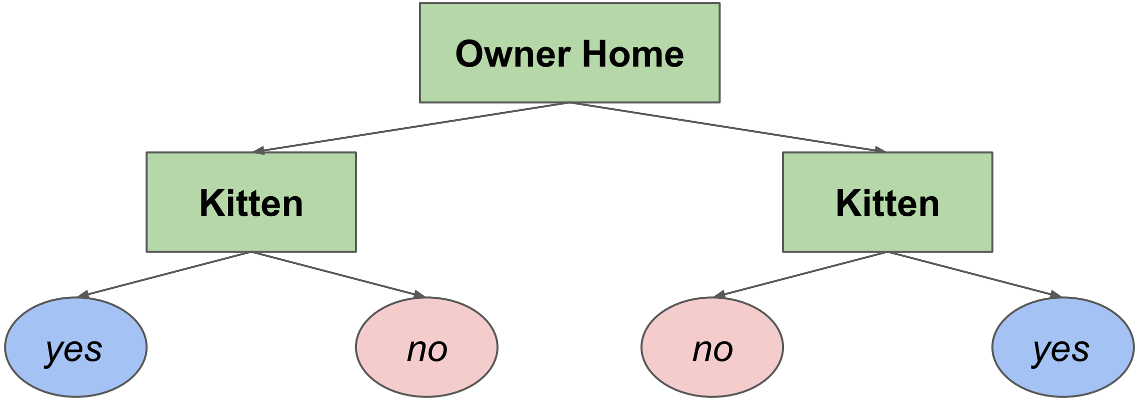
not Owner Home

Decision Tree for 1 Rule



not Owner Home ***and not*** kitten

Decision Tree for 2 Rules



(Owner Home ***and*** Kitten) ***or*** (***not*** Owner Home ***and not*** Kitten)

Question

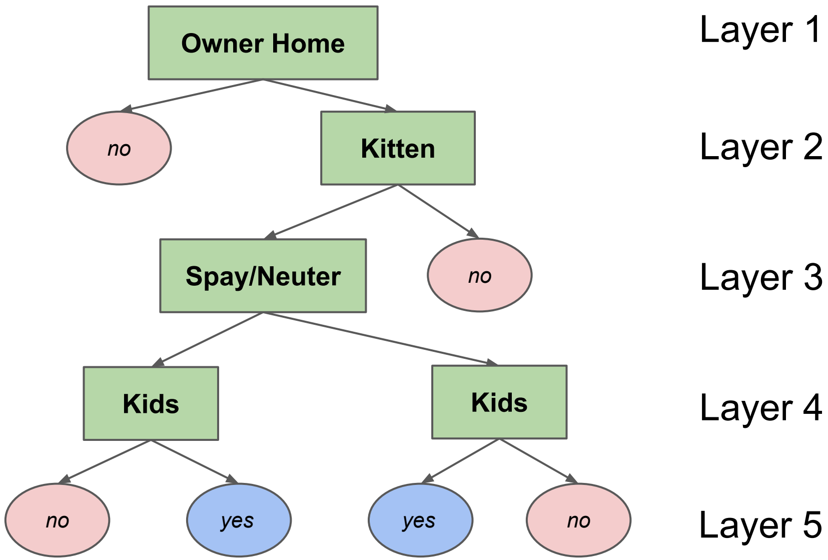
We’ll Have to Wait for the Question

Decision Trees of Depth T

$$\mathcal{X} = \{0,1\}^d$$

$$\mathcal{Y} = \{0,1\}$$

$$\mathcal{H} = \{h : h \text{ is a decision tree with layers } \leq T\}$$



Implications

1. At most 2^{T-1} leaves

2. Complexity grows with T

Learning Decision Trees

All Tree Search?

Let $\mathcal{H}$ be the set of all possible decision trees:

- How to compute ERM?

- What happens to generalization error?

All Tree Search?

Let $\mathcal{H}$ be the set of all possible decision trees:

- How to compute ERM?

  List all trees and pick best. Too expensive.

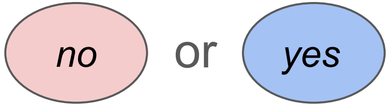
- What happens to generalization error?

  Poor. Overfits -- Low empirical risk, zero if no conflicting labels for same data

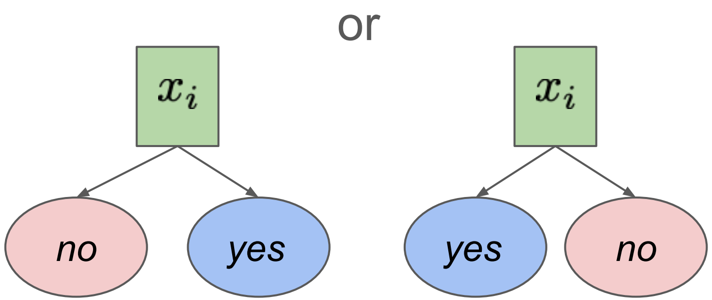
Greedy Search

- Idea: Build layer by layer, reducing empirical risk as fast and large as possible

- Start with the best possible 1-node tree



- On each iteration, we examine the effect of splitting a single leaf into one of two trees for each possible attribute x_i



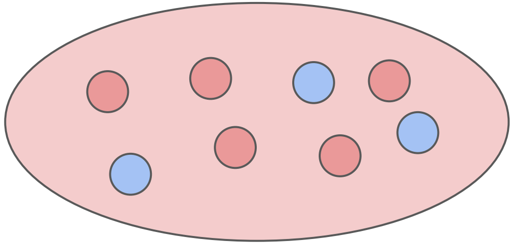
- Note that you can skip checking attributes that were split by a parent node

Scoring a Split

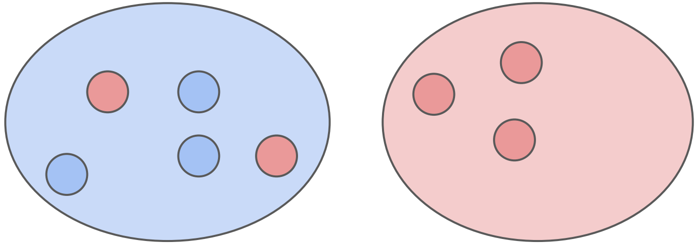
- A decision tree partitions each (x, y) in S into the leaves. Let l_s be the subset of S that reaches leaf l.

- Splitting a leaf doesn't change the score in other leaves. If we replace l with a split of x_i we get two new leaves:
  $l_0 = \{(x, y) \in l_s \ s.t. \ x_i = 0\}$
  and
  $l_1 = \{(x, y) \in l_s \ s.t. \ x_i = 1\}$

Before



After



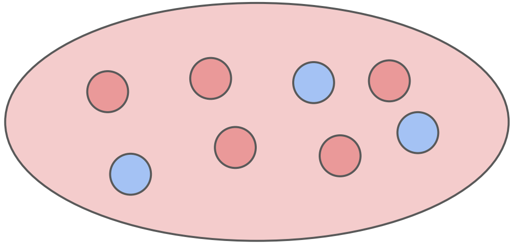
Scoring a Split

- Score of splitting a node l on x_i is defined as the Gain:

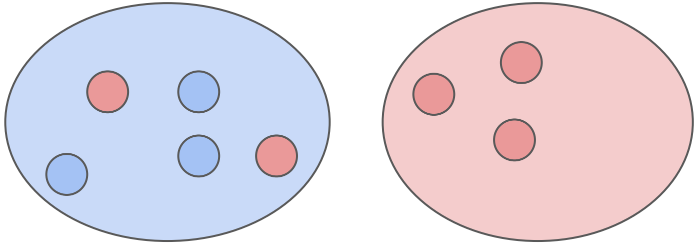
$$\begin{aligned}\mathrm{Gain}(l, i, S) = C(P_{\mathbf{x},y\sim l_S}[y=1]) \\ - P_{\mathbf{x},y\sim l_S}[x_i = 1] \cdot C(P_{\mathbf{x},y\sim l_S}[y = 1 | x_i = 1]) \\ - P_{\mathbf{x},y\sim l_S}[x_i = 0] \cdot C(P_{\mathbf{x},y\sim l_S}[y = 0 | x_i = 0])\end{aligned}$$

- If $C(a) = min(a, 1-a)$, then Gain is improvement in training error

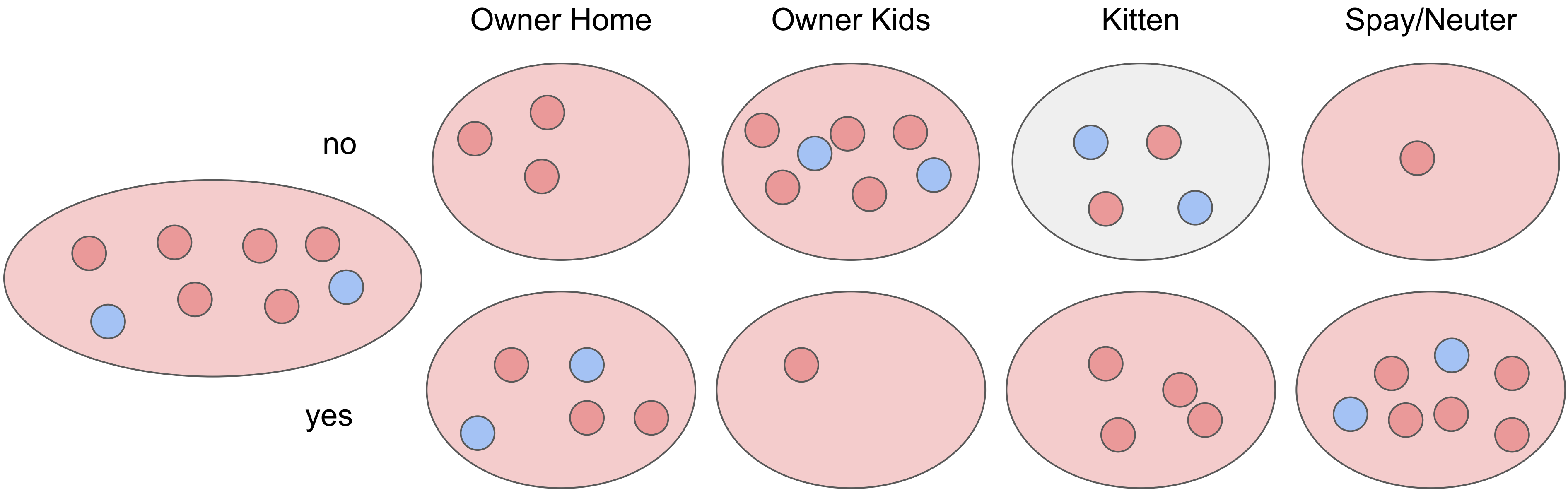
Before



After



Splits Visualized



Training error: 2/8 2/8 2/8 2/8 2/8

Desiderata: homogeneity, balance

Alternative Splitting Rules

Training error:

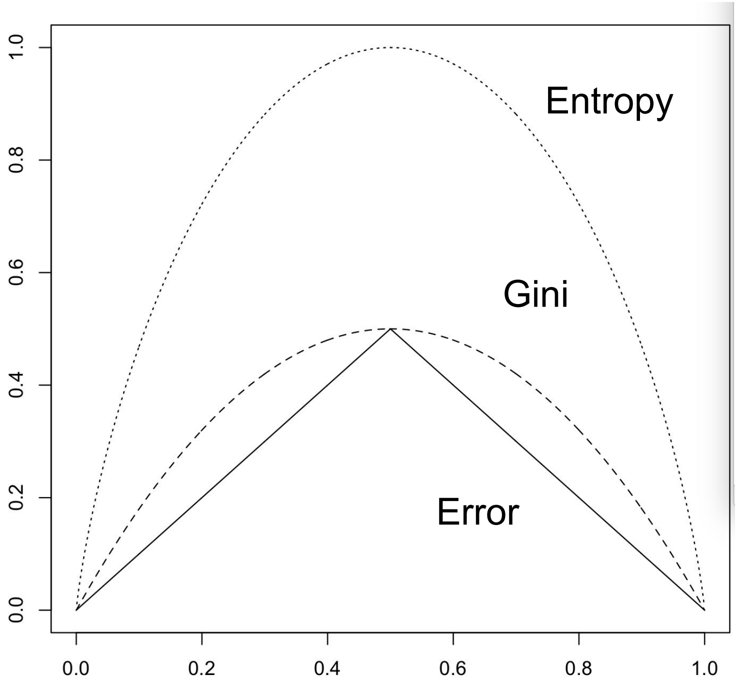
$$C(a) = min(a, 1 - a)$$

Entropy:

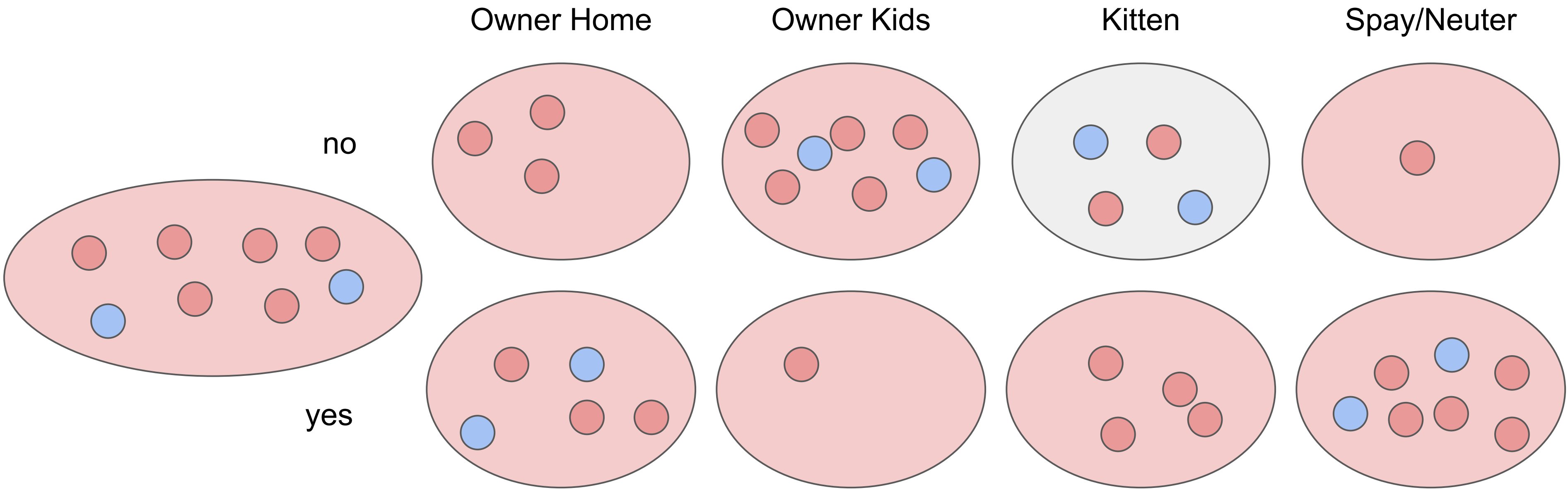
$$C(a) = -alog(a) - (1 - a)log(1 - a)$$

Gini impurity:

$$C(a) = 2a(1 - a)$$



Splits Visualized

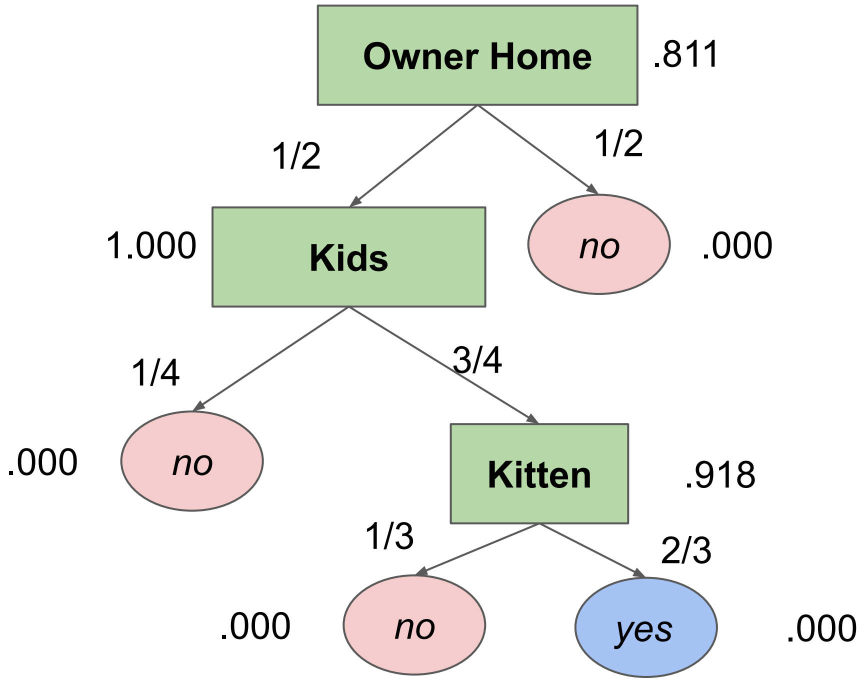


		Owner Home	Owner Kids	Kitten	Spay/Neuter
Training error:	2/8	2/8	2/8	2/8	2/8
Entropy:	.811	.607	.755	**.500**	.755
Gini:	.375	.300	.357	**.250**	.357

Splitting and Pruning

Learning the Tree Top Down

- Keep splitting leaves using the rule until leaf is completely homogeneous or all attributes have been split on that branch.
- Here (using entropy), we can get perfect classification.
- Should you split when the splitting rule shows no improvement?



Stopping Early Can Get Stuck

- Initial error: 1/2
- Splitting on **a**: 1/2
- Splitting on **b**: 1/2
- BUT, splitting on both **a** and **b** gets us to zero training error.

a	**b**	**label**
yes	no	yes
yes	yes	no
no	no	no
no	yes	yes

Not Stopping Overfits

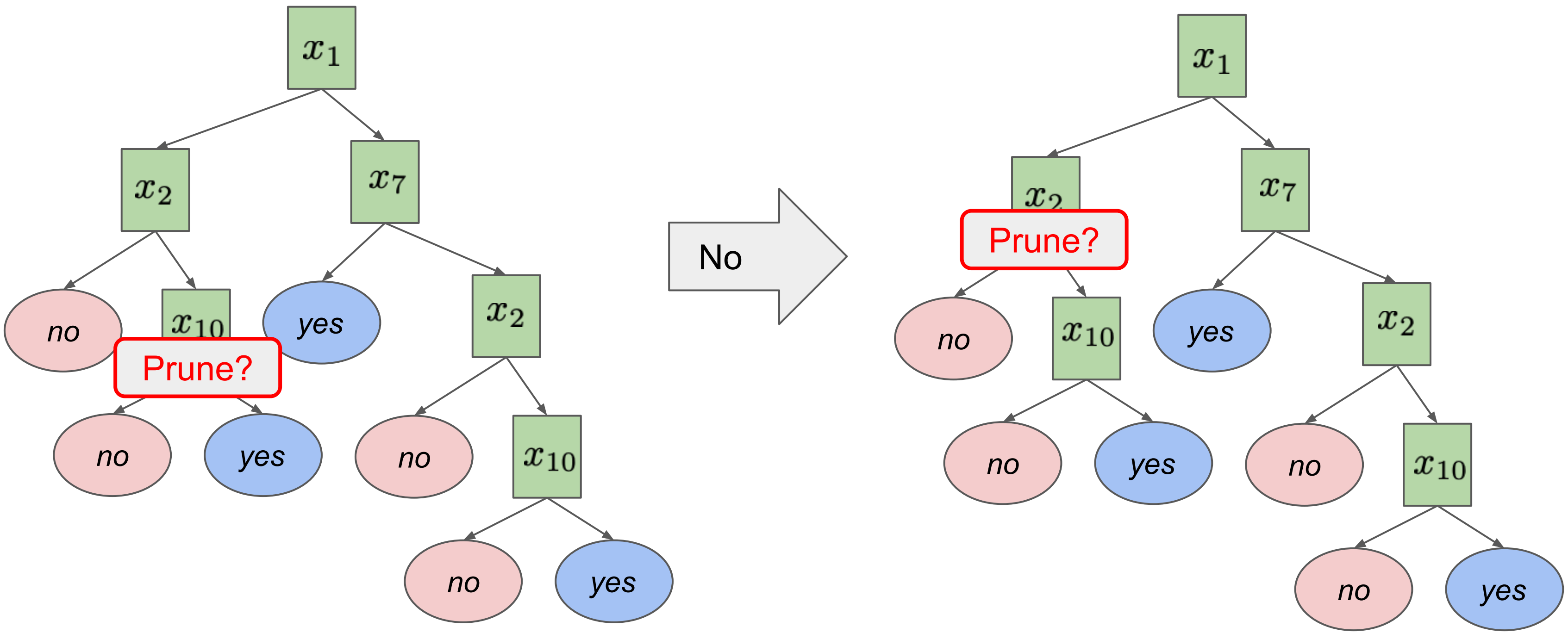
- The tree will grow until training error is zero (or all splits have been made)

- We’re optimizing over the space of all trees, so a very complex class

Pruning: Best of Both Worlds

- Build the whole tree, top down
- Chop off pieces, bottom up, that don’t seem to helping much
- How do we measure “helping?” Hold out some data for a validation set!
- Prune nodes (bottom up) whenever doing so improves 0-1 loss on a held out validation set

Pruning: Best of Both Worlds

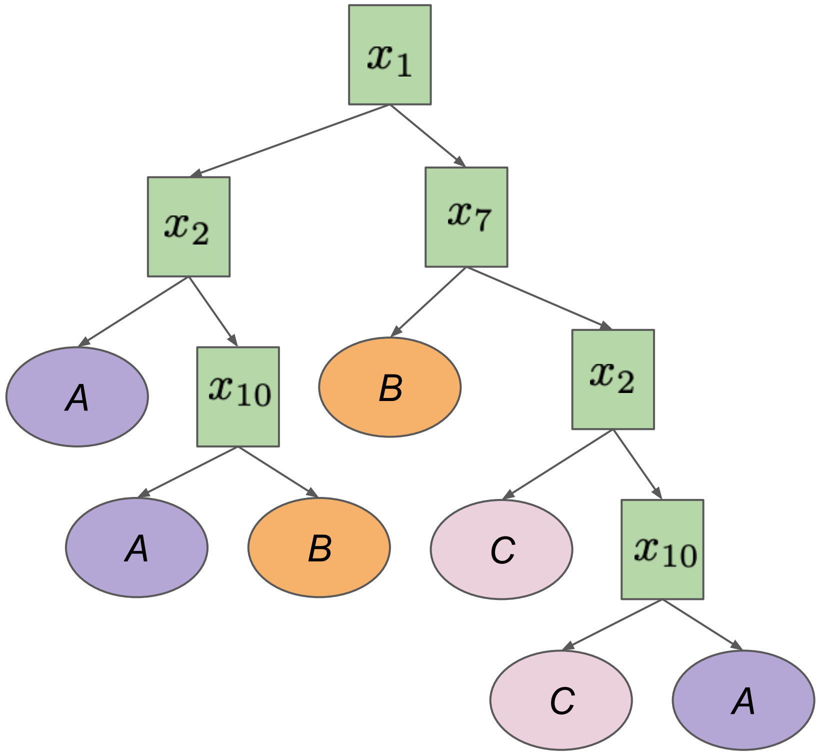


And so on...

Other Types of Decision Trees

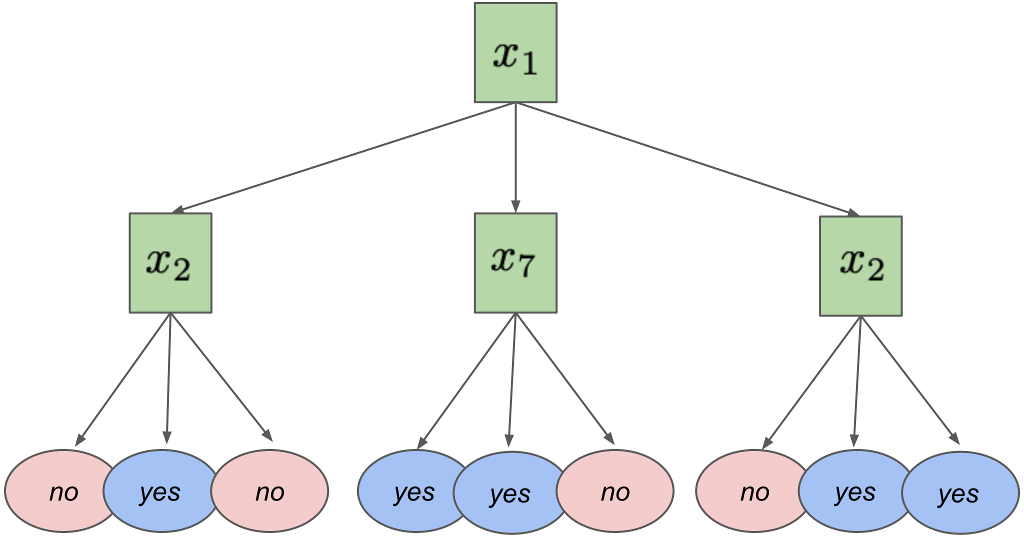
Multi-Class Classification

$$\mathcal{X} = \{0,1\}^{10}$$
$$\mathcal{Y} = \{A, B, C\}$$



Higher Arity Attributes (more than 2 values)

$$\mathcal{X} = \{1, 2, 3\}^{10}$$
$$\mathcal{Y} = \{0, 1\}$$



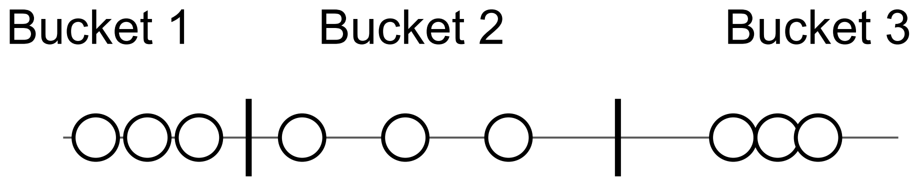
Continuous Attributes

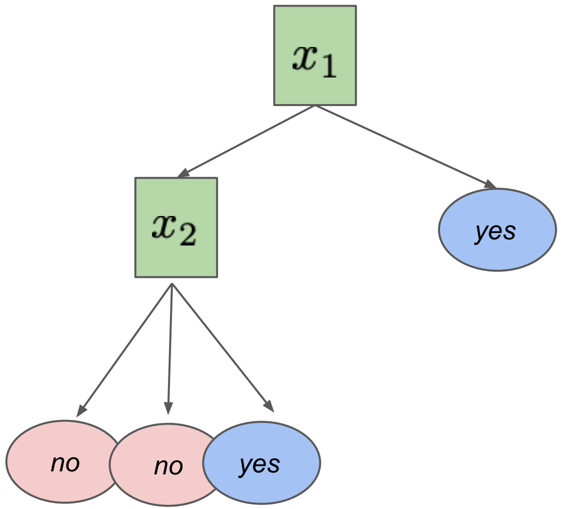
$$\mathcal{X} = \{0,1\} \times \mathbb{R}$$
$$\mathcal{Y} = \{0,1\}$$

Option #1: Discretization

Split attribute into k even buckets (quantiles)

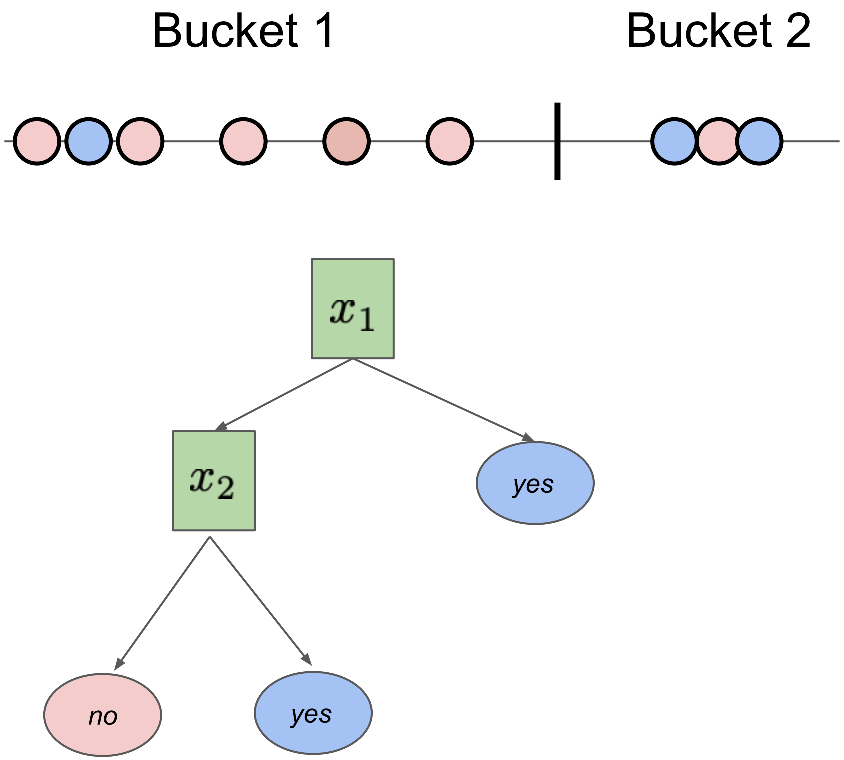
k is a hyperparameter





Continuous Attributes

$$\mathcal{X} = \{0,1\} \times \mathbb{R}$$
$$\mathcal{Y} = \{0,1\}$$



Option #2: Greedy Split

When considering whether to split on a continuous attribute, consider the best split point according to C(a)

The Most Important Things

- ***Decision trees*** encode a set of rules for making predictions
- We have different learning challenges due to discrete hypothesis class
- Greedy search with pruning is usually the preferred strategy

Textbook: chapter 18

Next Time

- Our next round of learning theory!
- What can we prove about the unrealizable case?
- Textbook: chapter 4