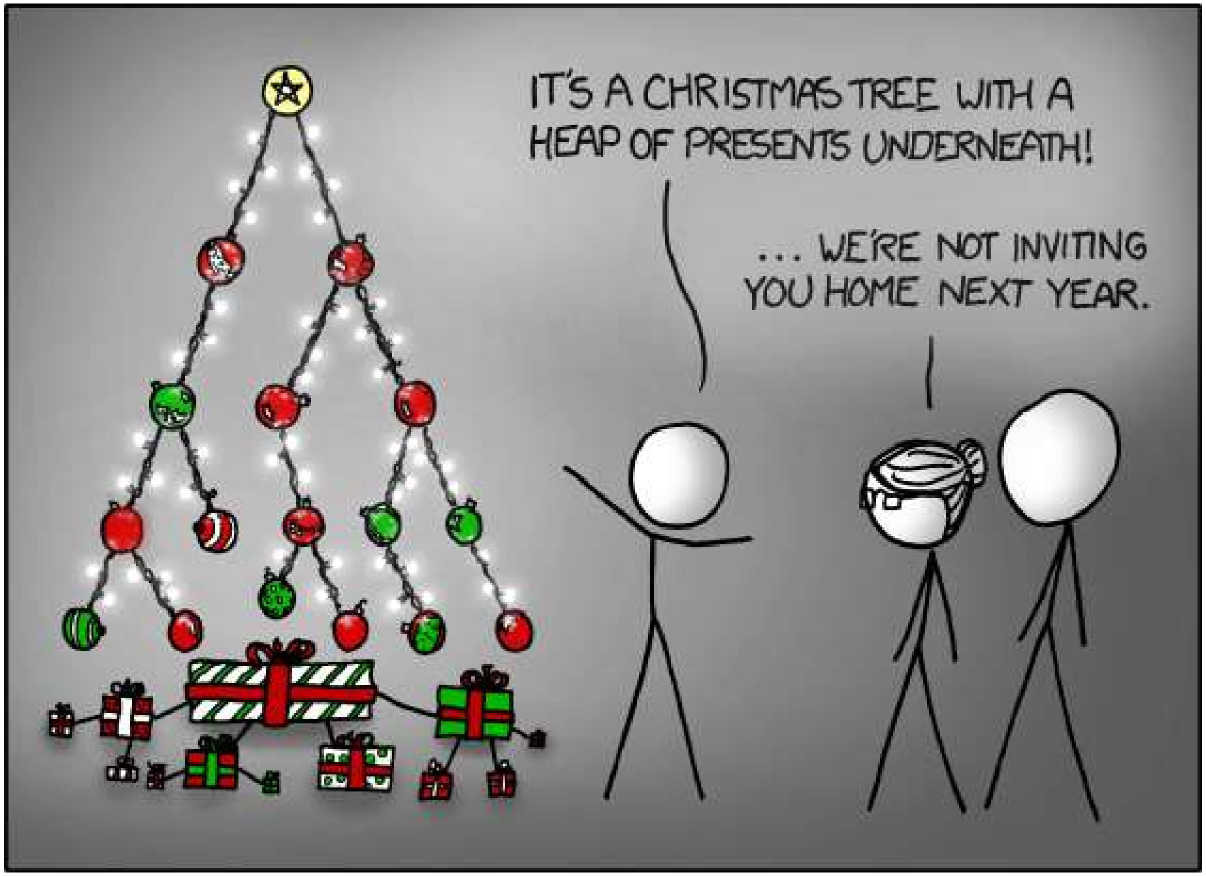
Machine Learning EN.601.475/675

DR. PHILIP GRAFF



Decision Trees

Decision Trees

- Have a long history in machine learning
  - The first popular algorithms dating back to 1979
- Popular in real-world settings
- Intuitive to understand or explain
- Easy to build

History

Elementary Perceiver and Memorizer (EPAM)

- Feigenbaum 1961
- Cognitive simulation model of human concept learning

CLS - Early algorithm for decision tree construction

- Hunt 1966

ID3 based on information theory

- Quinlan 1979

C4.5 improved over ID3

- Quinlan 1993

Also has history in statistics as CART (Classification and regression tree)

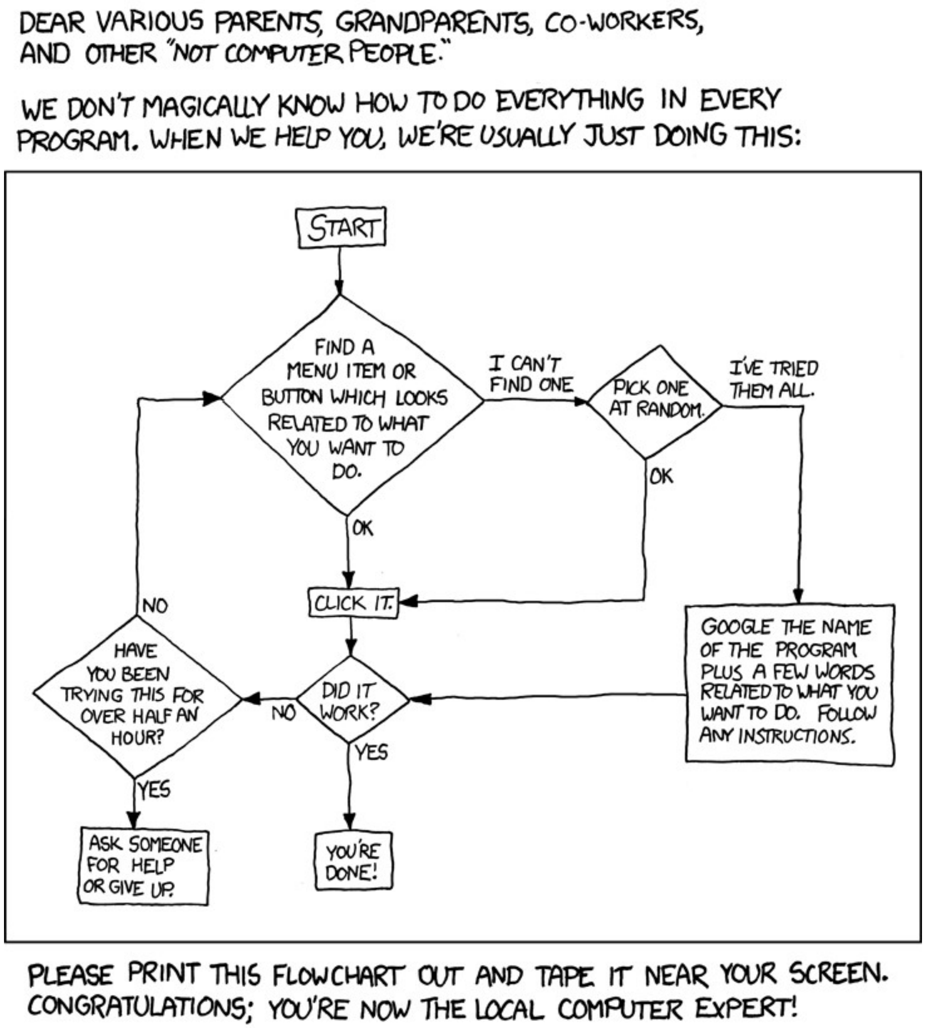
Motivation

How do people make decisions in real life?

- Consider a variety of factors
- Follow a logical path of checks and decisions

Should I eat at this restaurant?

- No wait → Yes
- Short wait and hungry → Yes
- Else → No



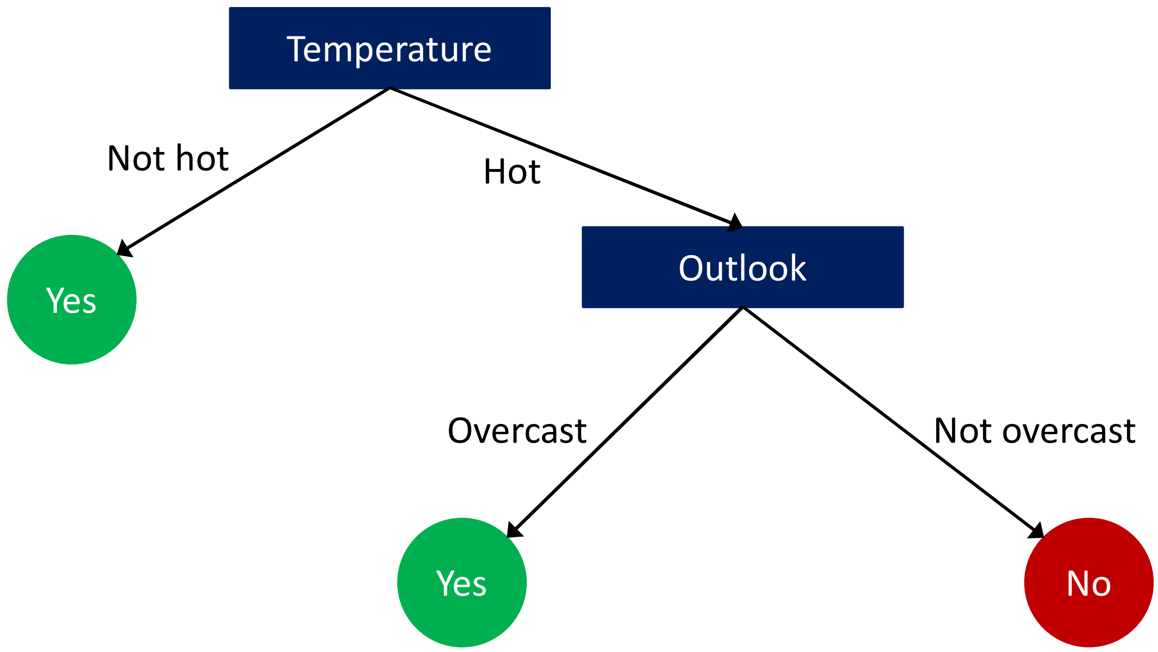
Example Decision Graph

Should we Play Tennis?

Outlook	Temperature	Humidity	Windy	Play Tennis
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cold	Normal	No	Yes

x y

Should we Play Tennis?

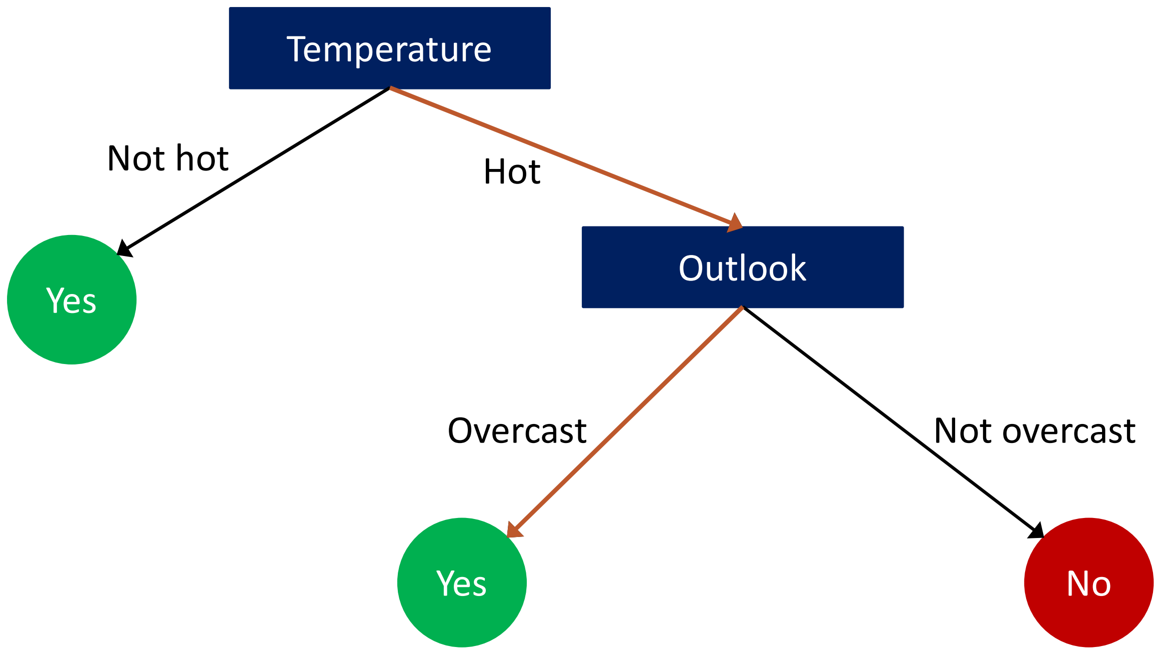


Should we Play Tennis?

How do we classify a new point?

Outlook: Overcast
Temp.: Hot
Humidity: Normal
Windy: No

Play: Yes

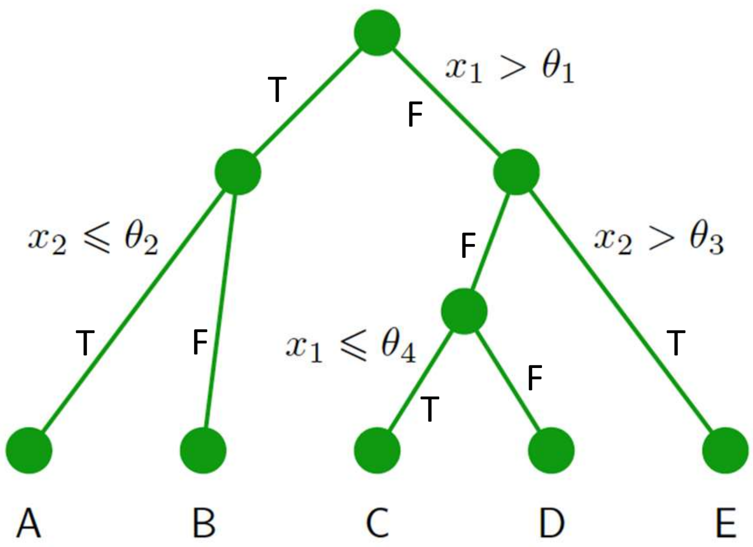


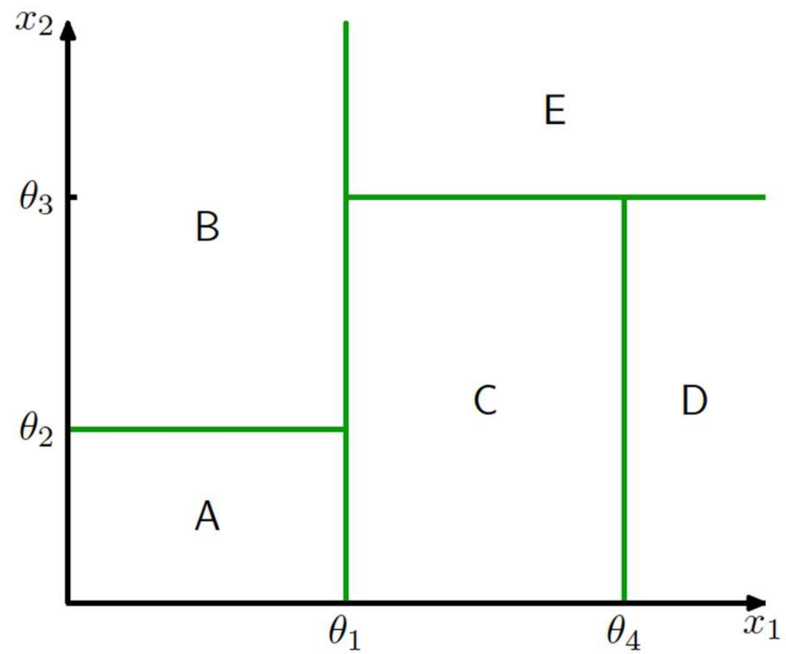
Decision Tree Anatomy

A decision tree is formed of

- Nodes
  - Attribute tests
- Branches
  - Results of attribute tests
- Leaves
  - Classifications

Decision Tree Example





Hypothesis Class

What functions can decision trees model?

Non-linear: very powerful hypothesis class

A decision tree can encode *any Boolean function*

- Given a truth table for the function
- Construct a path in the tree for each row
- Given a row, follow its path in the tree to the desired leaf
- Problem: exponentially large trees!

Y	X_1	X_2	X_3
1	0	0	0
0	0	0	1
1	0	1	0

Smaller Trees

Can we produce smaller decision trees for functions?

Most of the time: YES

- Counter examples: Parity function, Majority function

Decision trees are good for some functions but bad for others

Recall: tradeoff between hypothesis class expressiveness and learnability

Decision Trees

Fitting a function to data

Fitting: ???

Function: Any Boolean function

Data: Batch, construct a tree using all the data

Building Decision Trees

What Makes a Good Tree?

Small

- Ockham's razor → simpler is better
- Avoids over-fitting (we'll discuss more later)

A decision tree may be human readable but not use human logic

- The decision tree you would write for a problem may be different from what the computer devises

Small Trees

How do we build small trees that accurately capture the data?

Problem! Optimal decision tree learning is NP-complete*

- We can't guarantee that we'll find the optimal tree

* Constructing Optimal Binary Decision Trees is NP-complete. Laurent Hyafil, RL Rivest. Information Processing Letters, Vol. 5, No. 1. (1976), pp. 15-17.

Greedy Algorithms

Like for many NP-complete problems we can get pretty good solutions

Most decision tree learning uses greedy algorithms
- Adjustments usually to fix greedy selection problems

Top-down decision tree learning
- Recursive algorithms

ID3 Algorithm

```
function buildDecisionTree(data, labels):
    if all labels the same:
        return leaf node for that label
    else:
        let f be best feature for splitting (needs to be computed)
        left = buildDecisionTree(data with f=0, labels with f=0)
        right = buildDecisionTree(data with f=1, labels with f=1)
        return Tree(f, left, right)
```

Does this always terminate?

Base Cases (Terminating Recursion)

All data has the same label:

- Return that label

No examples:

- Return majority label from all data

No further splits possible:

- Return majority label of passed data

Selecting Features

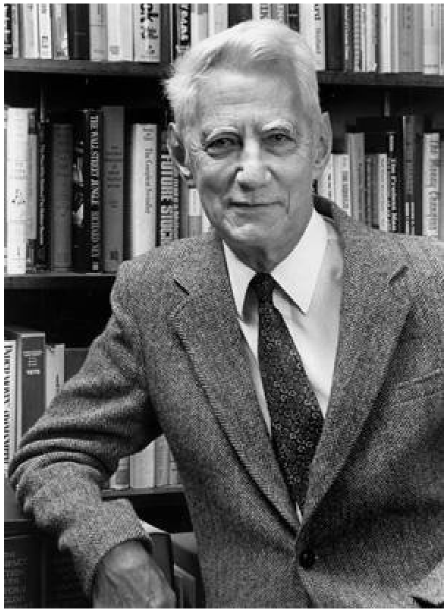
- How do we find the best feature for splitting?
- The most *informative* feature about the labels
- Must use *information theory*

Information Theory

The quantification of information

Founded by Claude Shannon

- Landmark paper in 1948
- Noisy channel theorem



Information Theory

Entropy: $$H(X) = -\sum_{x \in X} p(x) \log(p(x))$$

Conditional Entropy: $$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|X = x) \log(p(y|X = x)) \end{aligned}$$

Information Gain:

$$IG(Y|X) = H(Y) - H(Y|X)$$

Selecting Features

Can select the feature which maximizes the information gain

Measure relative to label distribution

- X = feature of choice
- Y = output label

Equivalent to minimizing the conditional entropy for each leaf

Notes for Decision Trees

1. We are comparing $H(Y|X)$ across different choices for X
   - $H(Y)$ is constant
   - We can omit it for comparisons

2. The base of the log doesn't matter as long as it is consistent

Should we Play Tennis?

Outlook	Temperature	Humidity	Windy	Play Tennis
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cold	Normal	No	Yes

$$H(\text{Tennis}) = -3/5 \log_2(3/5) - 2/5 \log_2(2/5) = 0.97$$

Should we Play Tennis?

Outlook	Temperature	Humidity	Windy	Play Tennis
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cold	Normal	No	Yes

$H(\text{Tennis} \mid \text{Outlook} = \text{Sunny}) = -2/2 \log_2(2/2) – 0/2 \log_2(0/2) = 0$

$H(\text{Tennis} \mid \text{Outlook} = \text{Overcast}) = -0/1 \log_2(0/1) – 1/1 \log_2(1/1) = 0$

$H(\text{Tennis} \mid \text{Outlook} = \text{Rainy}) = -0/2 \log_2(0/2) – 2/2 \log_2(2/2) = 0$

$H(\text{Tennis} \mid \text{Outlook}) = 2/5 * 0 + 1/5 * 0 + 2/5 * 0 = 0$

Should we Play Tennis?

Outlook	Temperature	Humidity	Windy	Play Tennis
Sunny	Hot	High	No	No
Sunny	Hot	High	Yes	No
Overcast	Hot	High	No	Yes
Rainy	Mild	High	No	Yes
Rainy	Cold	Normal	No	Yes

IG(Tennis | Outlook) = 0.97 – 0 = 0.97
If we know the Outlook we can perfectly predict Tennis!
Outlook is a great feature to pick for our decision tree

Base Cases

All data has the same label:

- Return that label

No examples:

- Return majority label from all data

No further splits possible:

- Return majority label of passed data

If max IG=0?

IG=0 as a Base Case

Consider the following:

Y	X_1	X_2
0	0	0
1	0	1
1	1	0
0	1	1

Both features give IG=0

Once we divide the data, we have perfect classification!

Training vs. Test Accuracy

Consider a similar tree built for similar tennis data:

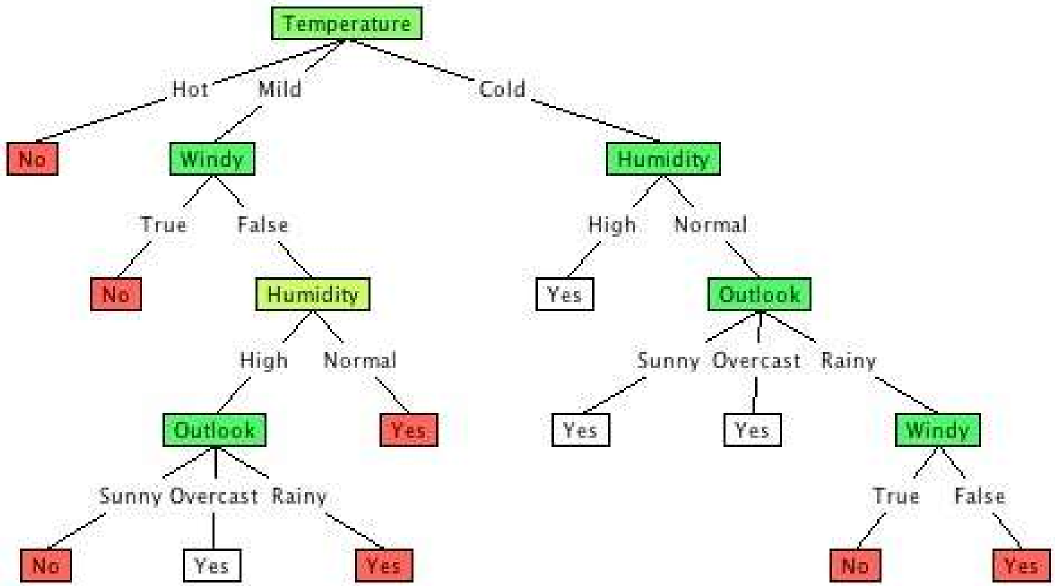
- Non-binary branches

100% training accuracy

30% testing accuracy

Why?

→ *Over-fitting!*



Over-Fitting

X_5 perfectly predicts Y

Let's randomly flip Y with probability ¼

X_5 will be the first split

- But tree will keep going

Duplicate training data into test

- Then flip Y again with probability ¼
- Train accuracy will be 100%
- Test accuracy will be 62.5% (5/8)
  - 1/16 examples are doubly corrupted
  - 9/16 are uncorrupted
  - 6/16 will be bad
- Single node test accuracy: 75%

Y	X_1	X_2	X_3	X_4	X_5
0	0	0	0	0	0
1	0	0	0	0	1
0	0	0	0	1	0
1	0	0	0	1	1
0	0	0	1	0	0
1	0	0	1	0	1
0	0	0	1	1	0
1	0	0	1	1	1
…	…	…	…	…	…

32 examples total

Bias/Variance Trade-off

Complete trees have no bias

- Can over-fit badly!
- Lots of variance from data samples

0-depth trees (return most likely label) have no variance

- Completely biased towards majority label

A good tree *balances* between these two

- How do we learn balanced trees?

Pruning: New Base Cases

1. Stop when too few examples in the branch
2. Stop when max depth reached
3. Stop when my classification error is not much more than the average of my children
   - Requires first computing and then removing
4. χ^2-pruning: stop when remainder is no more likely than chance

✓ Common practice is to build a large tree with stopping conditions 1 & 2, then remove leaf nodes based on criteria from 3 & 4

Parameters

All of these are parameters

How do you select parameters?

- Train data?
- Test data?
- Development data!

Decision Trees

Fitting a function to data

Fitting: greedy algorithm to find a good tree

- Extra heuristics to help with over-fitting
- Optimal decision tree learning: NP-complete

Function: Any Boolean function

Data: Batch, construct a tree using all the data

Extensions

Non-binary attributes

- Categorical
- Continuous (real-valued)
  - Handle by thresholding on splitting the range of values
  - Regression trees

Missing attributes

- Use weighted avg of all branches

Alternatives to information gain

- Gini index
- Misclassification rate

Non-greedy algorithms?

Continuous Parameters

How do we handle continuous inputs?

- We make them categorical!
- But how?

Thresholding!

Alternatives to Information Gain

Misclassification rate

- Label by most probable class in training data at each the leaf
- Error rate is fraction of test cases with wrong predicted label
- Simple to evaluate

Gini index

- Expected error rate based on distribution of classes at each leaf

$$\pi_c = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{1}(y_i = c)$$

$$G = \sum_{c \in C} \pi_c (1 - \pi_c) = \sum_c \pi_c - \sum_c \pi_c^2 = 1 - \sum_c \pi_c^2$$

Find feature *j** and threshold *t** that satisfies the following condition of minimizing the cost.
Continuous values on top, discrete values on bottom.
T_j is the set of possible thresholds/values for feature *j*.

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \left[\text{cost}\left(\{\mathbf{x}_i, y_i : x_{ij} \leq t\}\right) + \text{cost}\left(\{\mathbf{x}_i, y_i : x_{ij} > t\}\right)\right]$$

$$(j^*, t^*) = \arg \min_{j \in \{1,\ldots,D\}} \min_{t \in \mathcal{T}_j} \left[\text{cost}\left(\{\mathbf{x}_i, y_i : x_{ij} = t\}\right) + \text{cost}\left(\{\mathbf{x}_i, y_i : x_{ij} \neq t\}\right)\right]$$

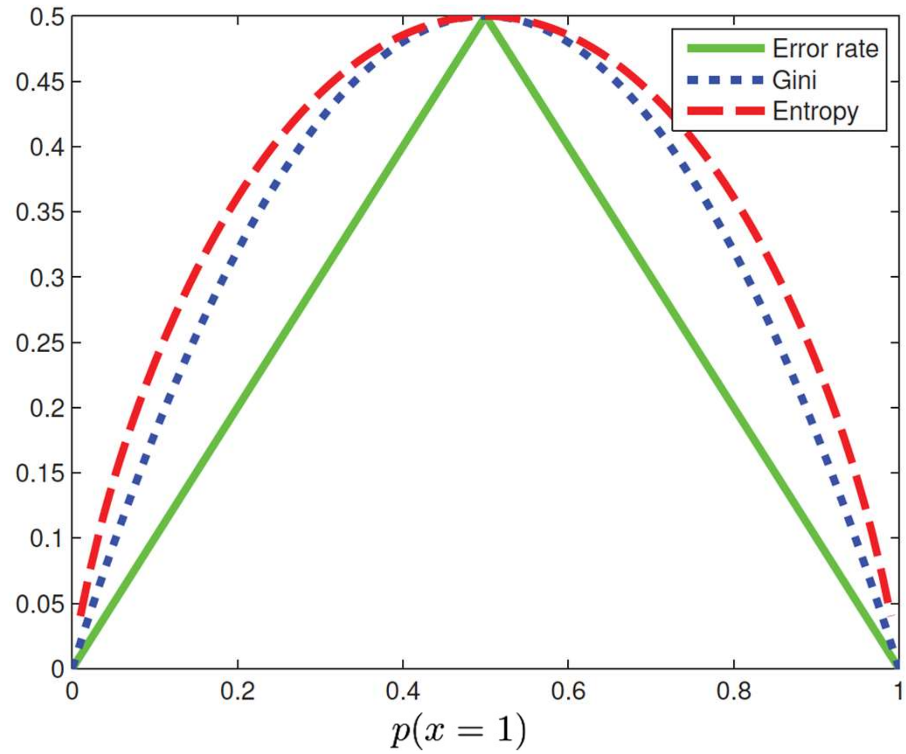
Measure the quality of a split by evaluating the reduction in
the cost, weighting by the sizes of the splits

$$\Delta \doteq \text{cost}(\mathcal{D}) - \left(\frac{|\mathcal{D}_L|}{|\mathcal{D}|}\text{cost}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{|\mathcal{D}|}\text{cost}(\mathcal{D}_R)\right)$$

Comparison of Alternatives

Binary classification case

- We can turn any categorical or continuous feature into binary options



Regression Trees

We can perform regression with trees as well

At each leaf, predict either:

1. Mean of response variable for data points at the leaf
2. Fit a linear model for the data points at the leaf

#1 is faster, but #2 is more precise

Cost function is typical least-squares-error

Constructs piecewise linear function

Pros and Cons of Decision Trees

Pros

Easy to interpret

Easily handle mixed continuous and discrete data types

Insensitive to monotonic transformations of inputs

Perform variable selection automatically

Scalable

Can handle missing inputs

Cons

Not as accurate as other models, partly due to greedy training

Unstable

- Small changes in training data can have large effects on tree structure
- Errors at the top propagate down due to hierarchical nature

Decision Trees Summary

Use logical splits to divide data into more pure subsets

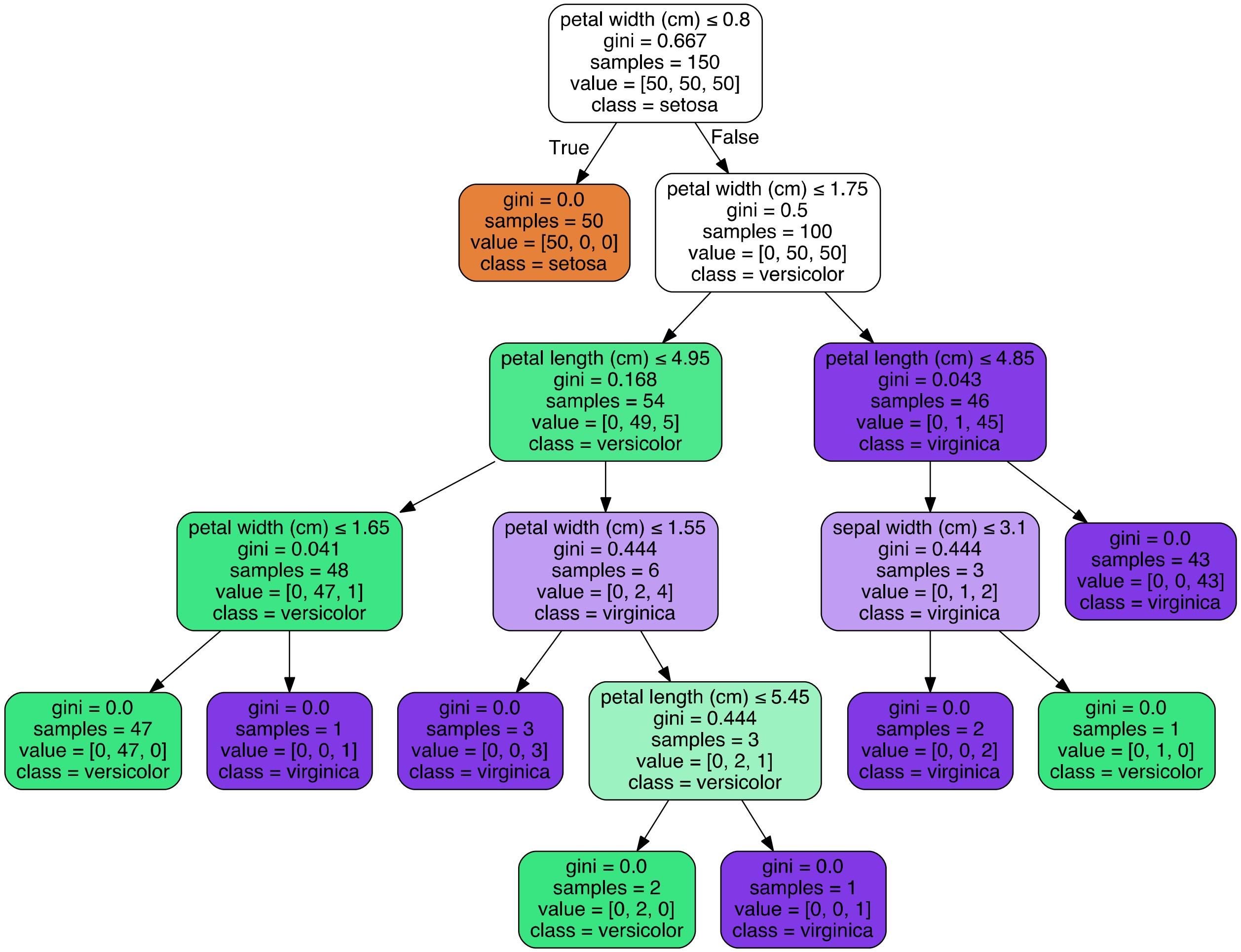
Make choices based on maximizing information gain

- Or minimizing similar metric: misclassification, Gini

Threshold continuous variables to make splits

Use pruning techniques to prevent over-fitting

Can perform regression with LSE and locally-linear models



Decision Tree Example: Iris Data

Ensemble Models

Ensemble Methods

Combine the outputs of classifiers to get something better

Build complex models from simpler models

Train many models and combine their predictions

If they are sufficiently different, they can account for each other's shortcomings and make better predictions overall

- Wisdom of the crowd

Learning

Weak learner

- learns a predictor slightly better than random guessing

Strong learner

- learns a predictor with arbitrary accuracy

Combine many weak learners to produce a strong learner

Ensemble Example

Consider an auto mechanic: You want to learn how to fix your car

Approach 1: Ask the mechanic to teach you how to diagnose car problems

- Problem: very complicated, unlikely to get a good answer

Approach 2: Show the mechanic a car with a problem

- Ask how to fix this problem
- Problem: you'll get a good answer, but only for that problem
- Solution: repeat until you get enough answers to cover all scenarios

Random Forests

Developed originally by Tin Kam Ho in 1995

Correct for variance of a single decision tree by averaging together many

- Classifiers vote on prediction
- Regression trees average predicted values

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x)$$

But won't we just generate the same tree *M* times?

Data Sampling

Bootstrap aggregating (Breiman 1996)

With a dataset of size *N*, create *M* different samples of size *N* by sampling **with replacement**

- Probability of an example being selected is $63.2\% = 1 - {}^{1}/_{e}$

Datasets for each tree are now different

- Still highly correlated
- Trees will be very similar

Feature Sampling

How can we further differentiate our trees?

Already randomly sample data points → randomly sample <u>features</u>

Pick random subset of features to consider for splitting

- At each split choose another random subset

Forces variation in tree structures built

- Some trees must account for using sub-optimal features

Random Forests Summary

Build an ensemble of decision trees
- Vote (classification) or average results (regression) to get total model prediction

Introduce variety in trees
- Data sampling – bagging
- Feature sampling
- Force under-fitting for each tree (small max depth, large min sample size)

Strong learner can outperform a single tree
- Better accuracy
- More robust to over-fitting

Next time: Boosting

EVEN BETTER ENSEMBLES!