CS 760: Machine Learning
Supervised Learning II

Kirthi Kandasamy

University of Wisconsin-Madison

2/6/2023

Announcements

- Homeworks:
  - Homework 1 due on Wednesday 2/8.
  - Homework 2 due next Wednesday 2/15.

- Recordings for Lecture 2 and 3 are out
  - Will release every 2-3 weeks going forward to reduce overheads on my end.

Outline

- **Review from last time**
  - k-NN, variations, strengths and weaknesses, generalizations
- **Decision trees, part I**
  - Setup, splits, learning, information gain, pros and cons
- **Decision trees, part II**
  - Stopping criteria, accuracy, overfitting

Outline

- **Review from last time**
  - k-NN, variations, strengths and weaknesses, generalizations
- **Decision trees, part I**
  - Setup, splits, learning, information gain, pros and cons
- **Decision trees, part II**
  - Stopping criteria, accuracy, overfitting

k-Nearest Neighbors: Classification

Training/learning: given

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

Prediction: for x , find ***k*** most similar training points

Return plurality class

$$\hat{y} \leftarrow \arg\max_{v \in \mathcal{Y}} \sum_{i=1}^{k} \delta(v, y^{(i)})$$

- I.e., among the ***k*** points, output most popular class.

k-Nearest Neighbors: Regression

Training/learning: given

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

Prediction: for x , find ***k*** most similar training points

Return

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y^{(i)}$$

- I.e., among the ***k*** points, output mean label.

k-Nearest Neighbors: Distances

Discrete features: Hamming distance

$$d_H(x^{(i)}, x^{(j)}) = \sum_{a=1}^{d} 1\{x_a^{(i)} \neq x_a^{(j)}\}$$

Continuous features:

- Euclidean distance:

$$d(x^{(i)}, x^{(j)}) = \left(\sum_{a=1}^{d} (x_a^{(i)} - x_a^{(j)})^2 \right)^{\frac{1}{2}}$$

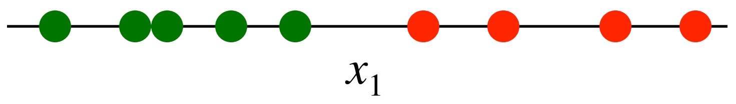
- L1 (Manhattan) dist.:

$$d(x^{(i)}, x^{(j)}) = \sum_{a=1}^{d} |x_a^{(i)} - x_a^{(j)}|$$

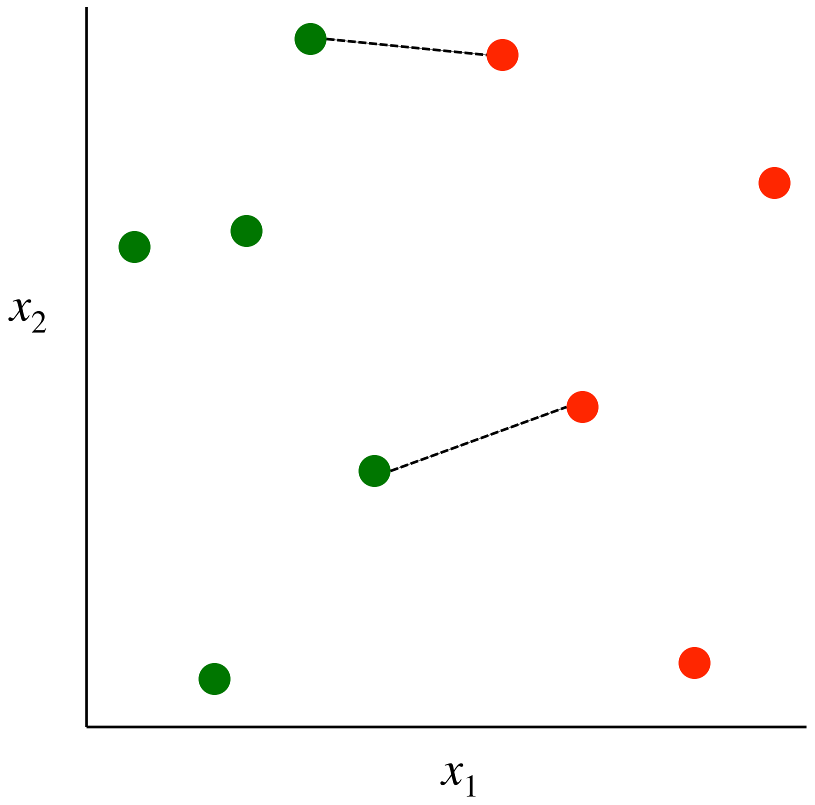
Dealing with **Irrelevant Features**

One relevant feature x_1

1-NN rule classifies each instance correctly



Effect of an irrelevant feature x_2 on distances and nearest neighbors



kNN: Strengths & Weaknesses

Strengths

- Easy to explain predictions
- Simple to implement and conceptualize.
- No training!
- Often good in practice, especially in low dimensions

Weaknesses

- Sensitive to irrelevant + correlated features
  - Can try to solve via variations. More later
- Prediction stage can be expensive
- No “model” to interpret

Inductive Bias

- ***Inductive bias***: assumptions a learner uses to predict y_i for a previously unseen instance $\boldsymbol{x}_i$
- Two components (mostly)
  - *hypothesis space bias*: determines the models that can be represented
  - *preference bias*: specifies a preference ordering within the space of models

learner	hypothesis space bias	preference bias
k-NN	Decomposition of space determined by nearest neighbors	instances in neighborhood belong to same class

Break & Quiz

Q1-1: Select the correct option.

A. kNN is sensitive to range of feature values.

B. Training is very efficient.

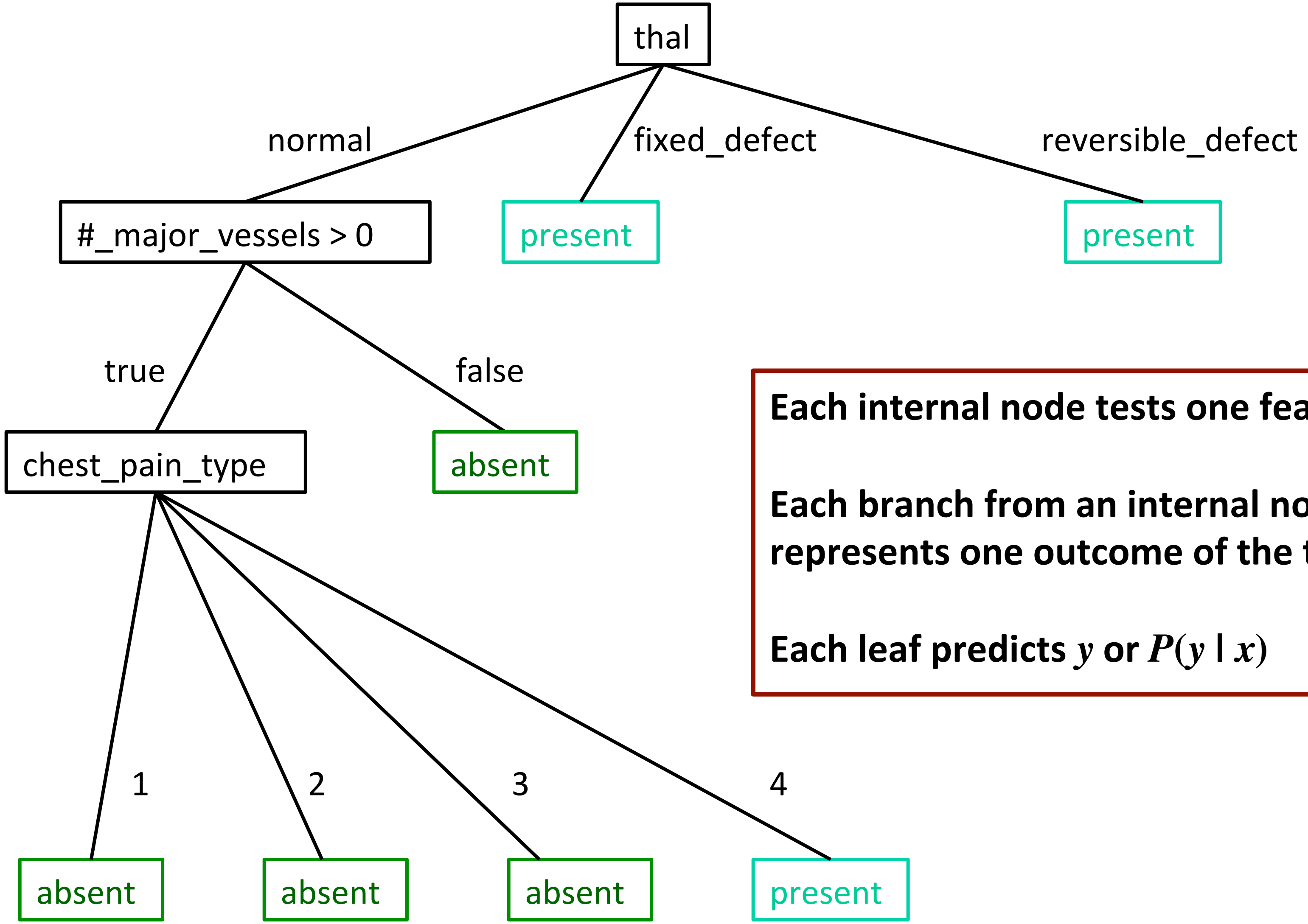
C. Occam's razor is an example of hypothesis space bias.

1. Statement A is true. Statement B, C are false.
2. Statement A, B are true. Statement C is false.
3. Statement B, C are true. Statement A is false.
4. All Statements are true.

Outline

- **Review from last time**
  - Instance-based learning, k-NN, variations, strengths and weaknesses, generalizations
- **Decision trees, part I**
  - Setup, splits, learning, information gain, pros and cons
- **Decision trees, part II**
  - Stopping criteria, accuracy, overfitting

Decision Trees: Heart Disease Example



Each internal node tests one feature x_i

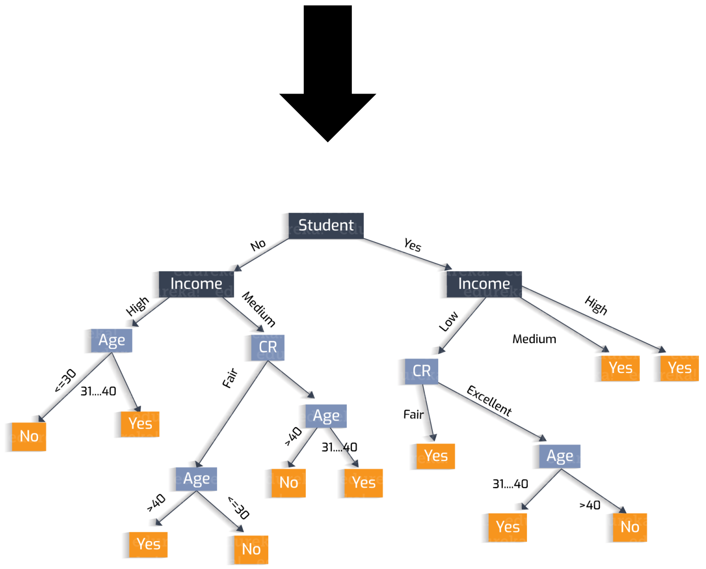
Each branch from an internal node represents one outcome of the test

Each leaf predicts y or $P(y \mid x)$

Decision Trees: Learning

- **Learning Algorithm:**

$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$



MakeSubtree(set of training instances D)

C = DetermineCandidateSplits(D)

if stopping criteria met

make a leaf node N

determine class label/probabilities for N

else

make an internal node N

S = FindBestSplit(D, C)

for each outcome k of S

D_k = subset of instances that have outcome k

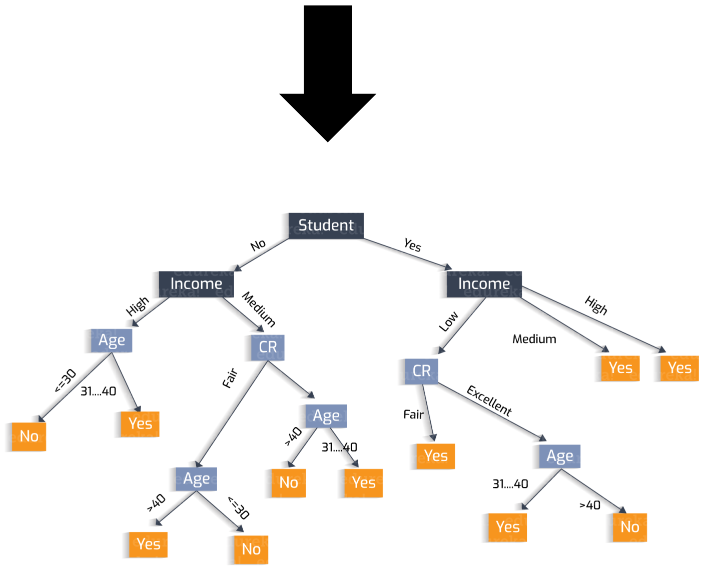
k^{th} child of N = MakeSubtree(D_k)

return subtree rooted at N

Decision Trees: Learning

- **Learning Algorithm:**

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$



MakeSubtree(set of training instances D)

C = **DetermineCandidateSplits**(D)

if **stopping criteria** is met

make a leaf node N

determine class label for N

else

make an internal node N

S = **FindBestSplit**(D, C)

for each group k of S

D_k = subset of training data in group k

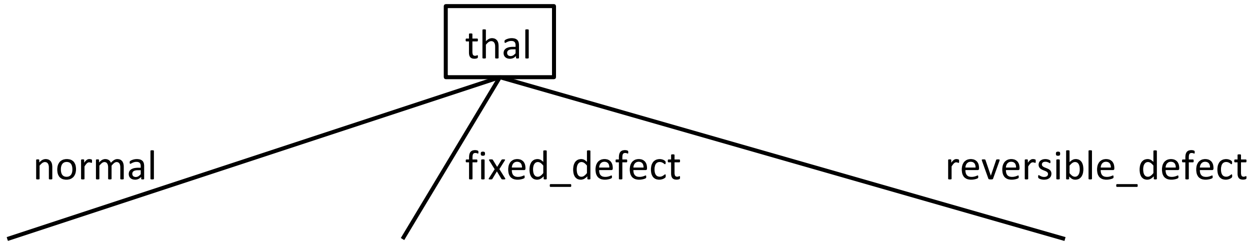
k^{th} child of N = MakeSubtree(D_k)

return subtree rooted at N

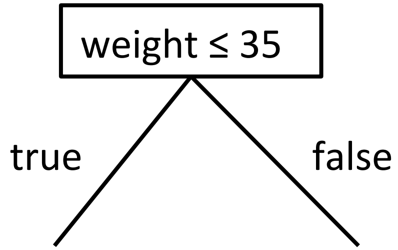
DT Learning: Candidate Splits

First, need to determine how to **split features**

- Splits on nominal features have one branch per value



- Splits on numeric features use a threshold/interval



ID3, C4.5

Numeric Feature Splits Algorithm

// Run this subroutine for each numeric feature at each node of DT induction

DetermineCandidateNumericSplits(set of training instances D, feature X_i)

$C = \{\}$ // initialize set of candidate splits for feature X_i

let v_j denote the value of X_i for the j^{th} data point

sort the dataset using v_j as the key for each data point

for each pair of adjacent v_j, v_{j+1} in the sorted order

if the corresponding class labels are different

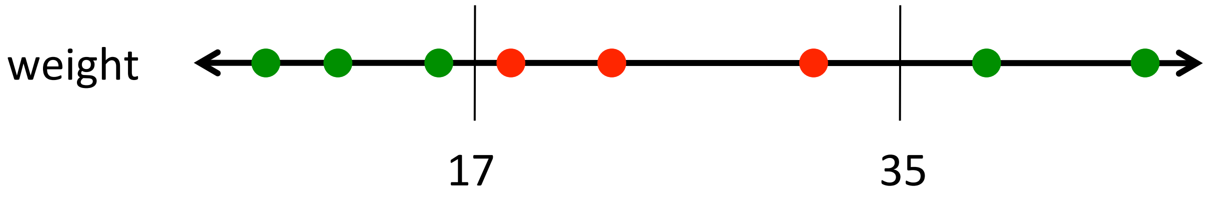
add candidate split $X_i \leq (v_j + v_{j+1})/2$ to C

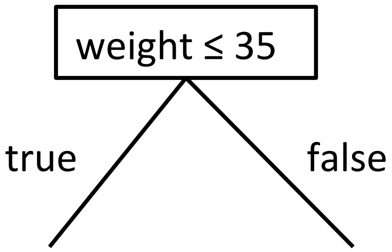
return C

DT Learning: Numeric Feature Splits

Given a set of training instances D and a specific feature X_i

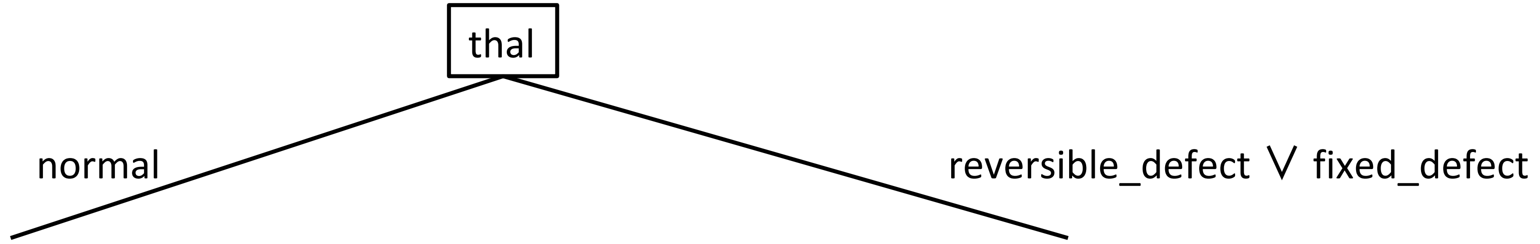
- Sort the values of X_i in D
- Evaluate split thresholds in intervals between instances of different classes

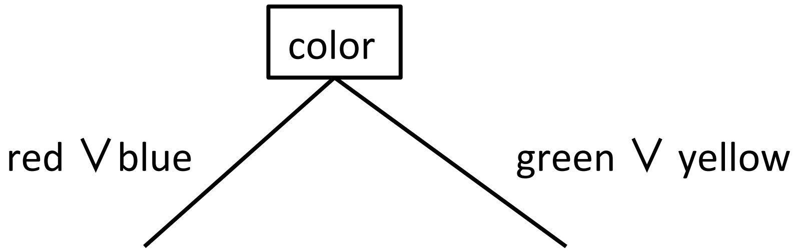




DT: Splits on Nominal Features

Instead of using k-way splits for k-valued features, could require binary splits on all nominal features (CART does this)

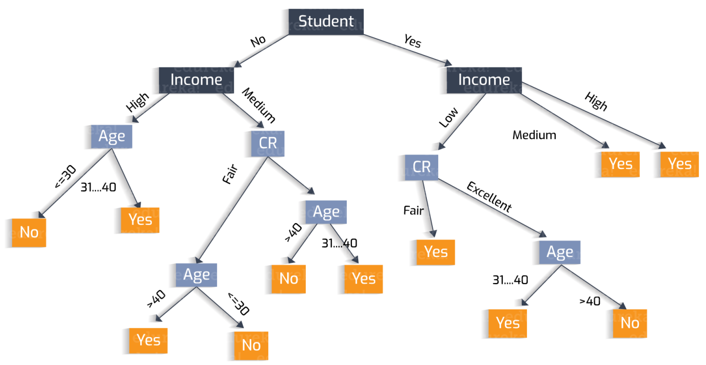




Decision Trees: Learning

- **Learning Algorithm:**

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$



MakeSubtree(set of training instances D)

C = **DeterminECandidateSplits**(D)

if **stopping criteria** is met

make a leaf node N

determine class label for N

else

make an internal node N

S = **FindBestSplit**(D, C)

for each group k of S

D_k = subset of training data in group k

k^{th} child of N = MakeSubtree(D_k)

return subtree rooted at N

DT Learning: Finding the Best Splits

How to we select the best feature to split on at each step?

- **Hypothesis**: simplest tree that classifies the training instances accurately will generalize

Occam's razor

- "when you have two competing theories that make the same predictions, the simpler one is the better"

DT Learning: Finding the Best Splits

How to we select the best feature to split on at each step?

- **Hypothesis**: simplest tree that classifies the training instances accurately will generalize

Why is Occam's razor a **reasonable heuristic?**

- There are fewer short models (i.e. small trees) than long ones
- A short model is unlikely to fit the training data well by chance
- A long model is more likely to fit the training data well coincidentally

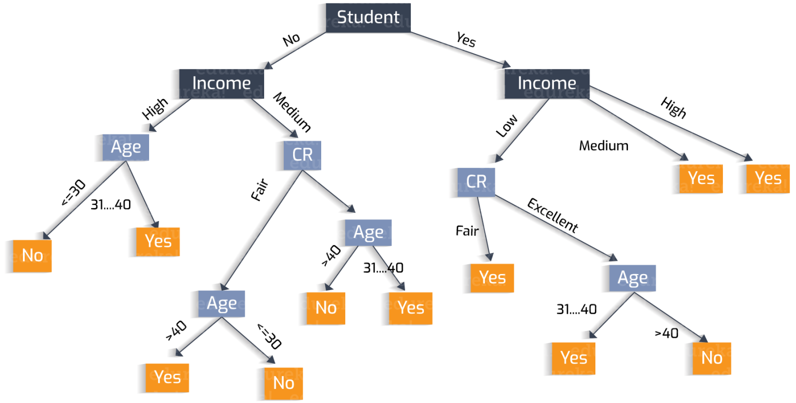
DT Learning: Finding Optimal Splits?

Can we find and return the smallest possible decision tree that accurately classifies the training set?

- **NO! This is an NP-hard problem**

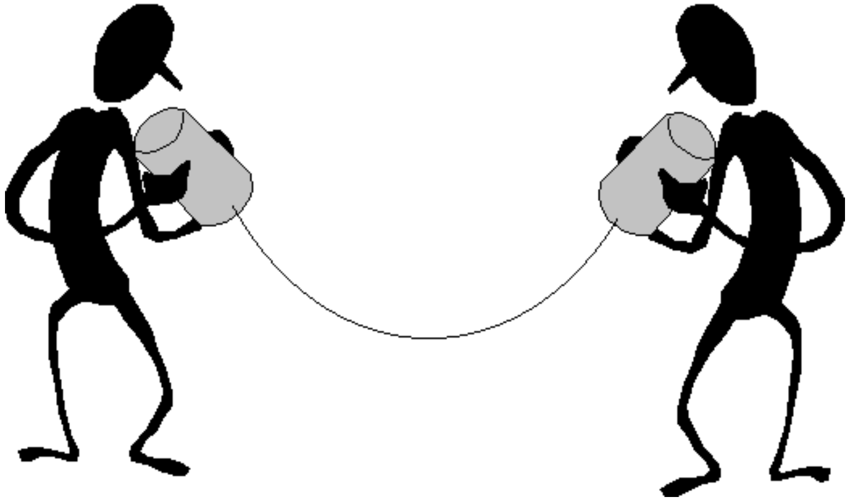
  [Hyafil & Rivest, *Information Processing Letters, 1976*]

- Instead, we'll use an information-theoretic heuristic to greedily choose splits



Digression: Information Theory

- **Goal**: communicate information to a receiver
- Ex: as bikes go past, communicate the maker of each bike



Information Theory: Encoding

- Could yell out the names of the manufacturers…
  - Suppose there are 4: **Trek**, **Specialized**, **Cervelo**, **Serrota**

- Inefficient… since there's just 4, we could **encode** them
  - # of bits: 2 per communication

type	code
Trek	11
Specialized	10
Cervelo	01
Serrota	00

Information Theory: Encoding

- Now, some bikes are rarer than others…
  - **Cervelo** is a rarer specialty bike.
  - We could **save some bits**… make more popular messages fewer bits, rarer ones more bits
  - Note: this is **on average**

- Expected # bits: **1.75**

$$-\sum_{y \in \mathcal{Y}} P(y) \log_2 P(y)$$

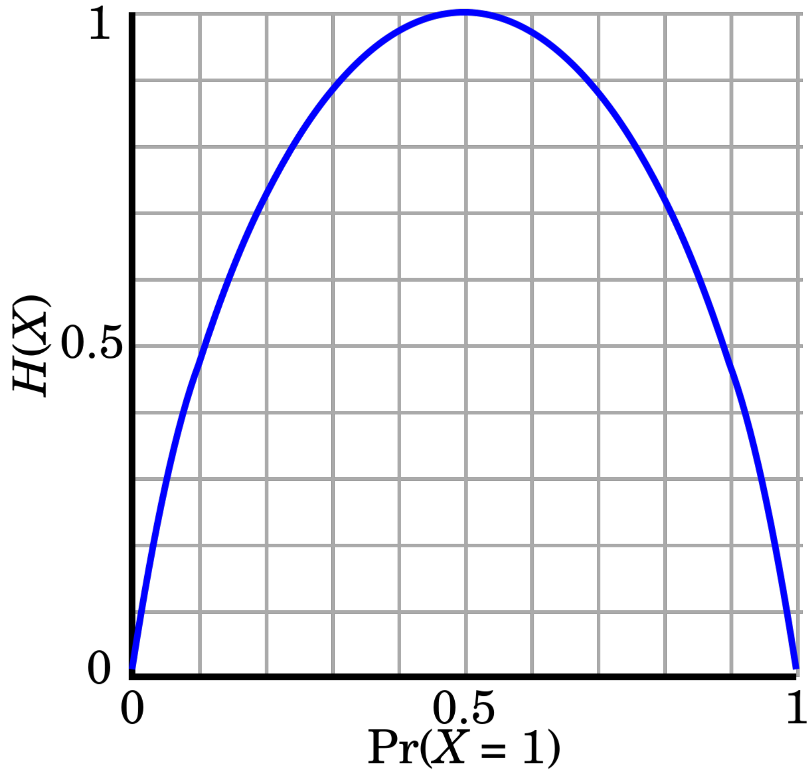
Type/probability	# bits	code
$P(\text{Trek}) = 0.5$	1	1
$P(\text{Specialized}) = 0.25$	2	01
$P(\text{Cervelo}) = 0.125$	3	001
$P(\text{Serrota}) = 0.125$	3	000

Information Theory: Entropy

- Measure of uncertainty for random variables/distributions

- **Expected number of bits** required to communicate the value of the variable

$$H(Y) = -\sum_{y \in \mathcal{Y}} P(y) \log_2 P(y)$$



Information Theory: Conditional Entropy

- Suppose we know *X*. **CE**: how much uncertainty left in *Y*?

$$H(Y|X) = -\sum_{x \in \mathcal{X}} P(X = x) H(Y|X = x)$$

- Here,

$$H(Y|X = x) = -\sum_{y \in \mathcal{Y}} P(Y = y|X = x) \log_2 P(Y = y|X = x)$$

- What is it if Y=X?
- What if Y is **independent** of X?

Information Theory: Conditional Entropy

- Example. *Y* is still the bike maker, *X* is color.

Y=Type/X=Color	Black	White
Trek	0.25	0.25
Specialized	0.125	0.125
Cervelo	0.125	0
Serrota	0	0.125

$H(Y|X=\text{black}) = -0.5 \log(0.5) - 0.25 \log(0.25) - 0.25 \log(0.25) - 0 = 1.5$

$H(Y|X=\text{white}) = -0.5 \log(0.5) - 0.25 \log(0.25) - 0 - 0.25 \log(0.25) = 1.5$

$H(Y|X) = 0.5 * H(Y|X=\text{black}) + 0.5 * H(Y|X=\text{white}) = 1.5$

Information Theory: Mutual Information

- Similar comparison between R.V.s:

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

Interpretation:

- How much uncertainty of Y that X can reduce.
- Or, how much information about Y can you glean by knowing X?

Y=Type/X=Color	Black	White
Trek	0.25	0.25
Specialized	0.125	0.125
Cervelo	0.125	0
Serrota	0	0.125

I(Y:X) = H(Y) − H(Y|X) = 1.75 − 1.5 = 0.25

Decision Tree Learning: Back to Splits

Want to choose split S that maximizes

$$\text{InfoGain}(D, S) = H_D(Y) - H_D(Y|S)$$

ie, mutual information.

- Note: D denotes that this is the **empirical** entropy
  - We don't know the real distribution of *Y*, just have our dataset
- Equivalent to maximally reducing conditional entropy of Y

DT Learning: InfoGain Example

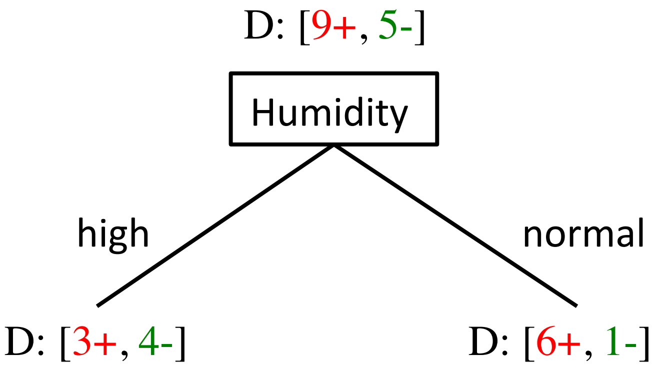
Simple binary classification (**play tennis**?) with 4 features.

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

DT Learning: InfoGain For One Split

- What's the information gain of splitting on Humidity?



$$H_D(Y) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

$$H_D(Y \mid \text{high}) = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$
$$= 0.985$$

$$H_D(Y \mid \text{normal}) = -\frac{6}{7}\log_2\left(\frac{6}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)$$
$$= 0.592$$

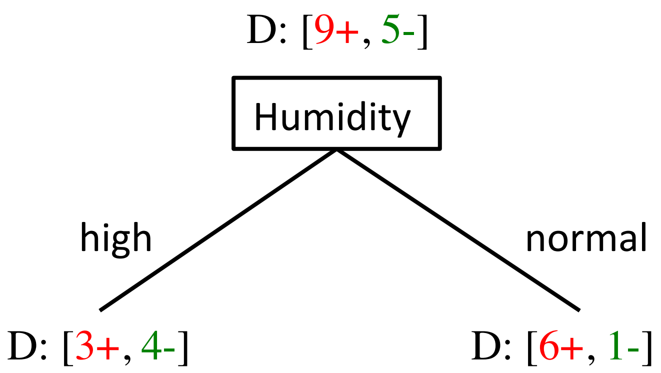
$$\text{InfoGain}(D, \text{Humidity}) = H_D(Y) - H_D(Y \mid \text{Humidity})$$
$$= 0.940 - \left[\frac{7}{14}(0.985) + \frac{7}{14}(0.592)\right]$$
$$= 0.151$$

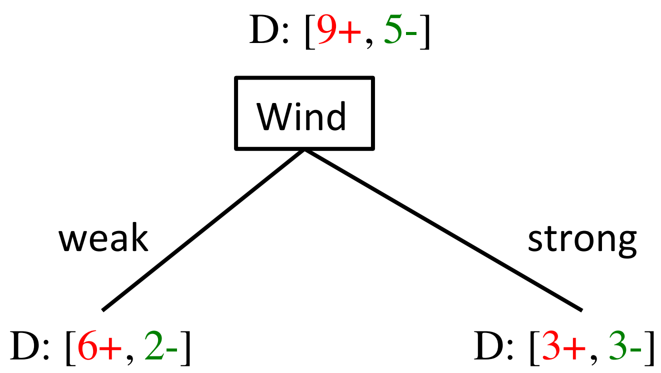
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

DT Learning: Comparing Split InfoGains

- Is it better to split on **Humidity** or **Wind**?





$$H_D(Y \mid \text{weak}) = 0.811 \qquad H_D(Y \mid \text{strong}) = 1.0$$

✔
$$\text{InfoGain}(D, \text{Humidity}) = 0.940 - \left[\frac{7}{14}(0.985) + \frac{7}{14}(0.592)\right]$$
$$= 0.151$$

$$\text{InfoGain}(D, \text{Wind}) = 0.940 - \left[\frac{8}{14}(0.811) + \frac{6}{14}(1.0)\right]$$
$$= 0.048$$

DT Learning: InfoGain Limitations

- InfoGain is biased towards tests with many outcomes
  - Splitting on it results in many branches, each of which is "pure" (has instances of only one class)
  - In the extreme: A feature that uniquely identifies each instance
  - **Maximal** information gain!

- Use **GainRatio**: normalize information gain by entropy

$$\mathrm{GainRatio}(D, S) = \frac{\mathrm{InfoGain}(D,S)}{H_D(S)} = \frac{H_D(Y) - H_D(Y|S)}{H_D(S)}$$

Break & Quiz

Q2-2: Which of the following statements is TRUE?

1. If there is no noise, then there is no overfitting.
2. Overfitting may improve the generalization ability of a model.
3. Generalization error is monotone with respect to the capacity/ complexity of a model.
4. More training data may help preventing overfitting.

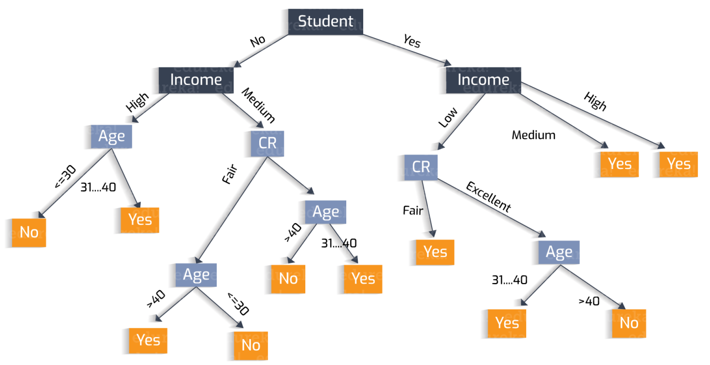
Outline

- **Review from last time**
  - Instance-based learning, k-NN, variations, strengths and weaknesses, generalizations
- **Decision trees, part I**
  - Setup, splits, learning, information gain, pros and cons
- **Decision trees, part II**
  - Stopping criteria, accuracy, overfitting

Decision Trees: Learning

- **Learning Algorithm:**

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$



MakeSubtree(set of training instances D)

C = **DetermineCandidateSplits**(D)

if **stopping criteria** is met

make a leaf node N

determine class label for N

else

make an internal node N

S = **FindBestSplit**(D, C)

for each group k of S

D_k = subset of training data in group k

k^{th} child of N = MakeSubtree(D_k)

return subtree rooted at N

Decision tree learning: Stopping Criteria

Some ideas

- Stop when you reach a single data point?
- Stop when the subset of instances are all in the same class?
- Stop when we a large fraction of the instances are all in the same class?
- We have exhausted all of the candidate splits

What about regression?

Inductive Bias

- Recall: ***Inductive bias***: assumptions a learner uses to predict y_i for a previously unseen instance $\boldsymbol{x}_i$
- Two components
  - *hypothesis space bias*: determines the models that can be represented
  - *preference bias*: specifies a preference ordering within the space of models

learner	hypothesis space bias	preference bias
Decision tree	trees with single-feature, axis-parallel splits	small trees identified by greedy search
k-NN	Voronoi decomposition determined by nearest neighbors	instances in neighborhood belong to same class

Quiz: Which of the following statements are True?

1. In a decision tree, once you split using one feature, you cannot split again using the same feature.
2. We should split along all features to create a decision tree.
3. We should keep splitting the tree until there is only one data point left at each leaf node.

Evaluating models

Evaluation: Accuracy

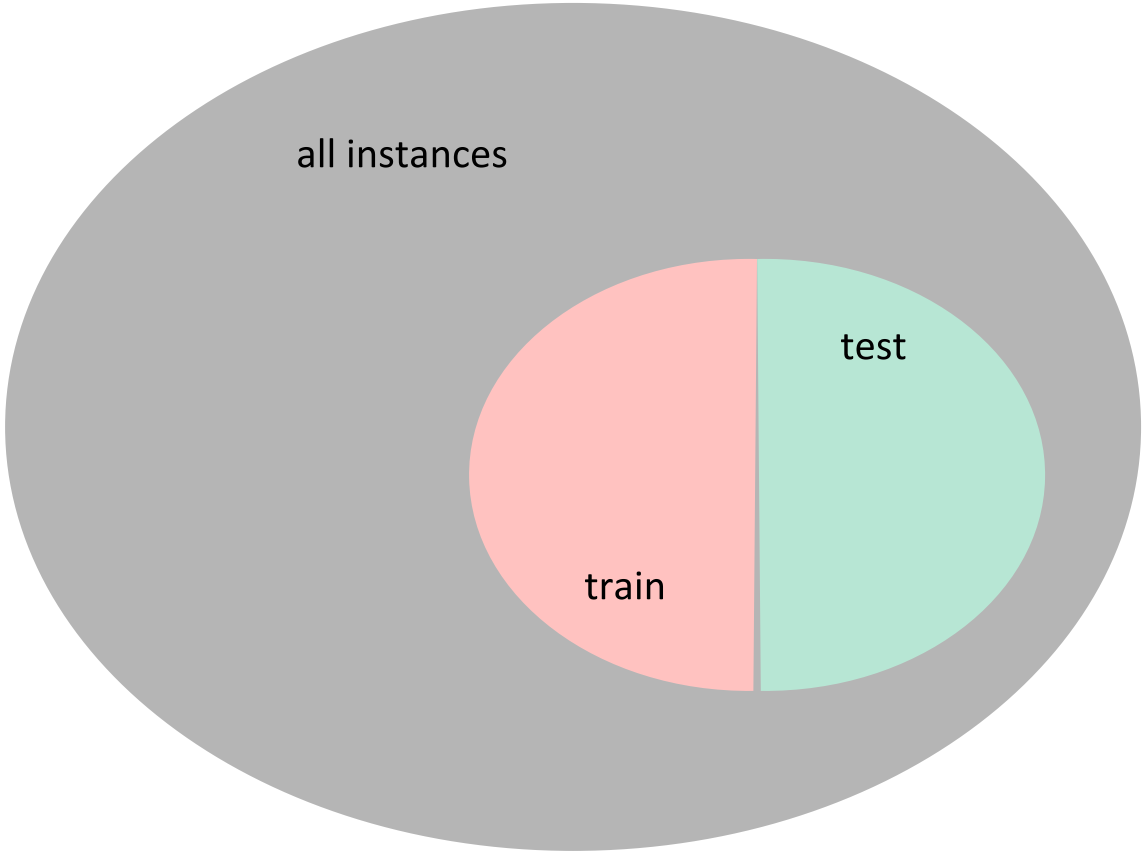
- Can we just calculate the fraction of training instances that are correctly classified?
- Consider a problem domain in which instances are assigned labels at random with $P(Y = 1) = 0.5$
  - How accurate would it be on its training set, if you stop when all instances are in the same class?
  - How accurate would a learned decision tree be on previously unseen instances?

- Recall: our goal is to do well on *future data*.

Evaluation: Accuracy

To get unbiased estimate of model accuracy, we must use a set of instances that are **held-aside** during learning

- This is called a **test set**



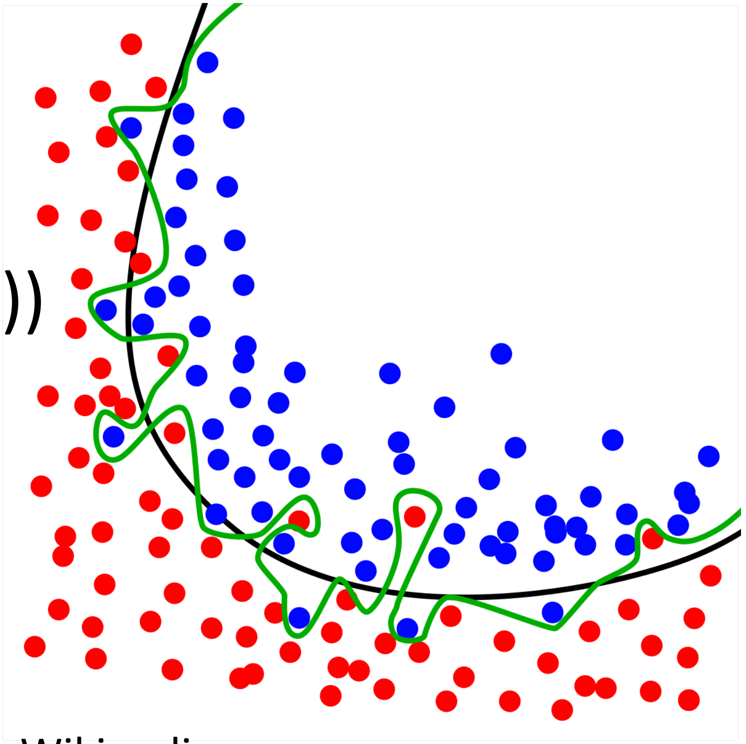
Overfitting

Notation: error of model *h* over

- training data: $\text{error}_{\mathrm{D}}(h)$
- entire distribution of data: $\text{error}_{\mathcal{D}}(h)$

Model *h* **overfits** training data if it has

- a low error on the training data (low $\text{error}_{\mathrm{D}}(h)$)
- high error on the entire distribution (high $\text{error}_{\mathcal{D}}(h)$)



Wikipedia

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

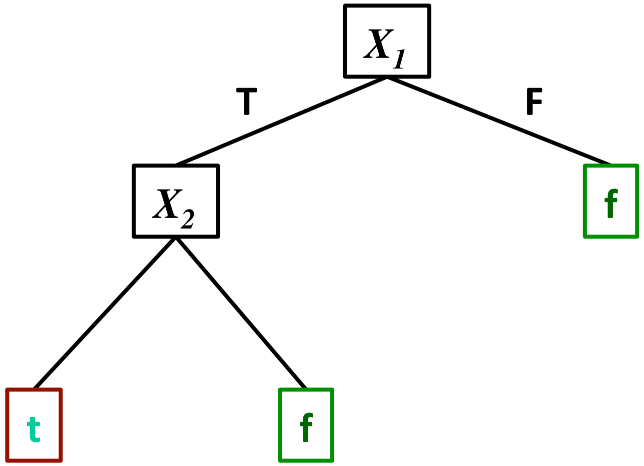
- There is noise in some feature values
- Training set

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	f	f	t	…	t
t	f	t	t	f	…	t
t	f	f	t	f	…	f
t	f	t	f	f	…	f
f	t	t	f	t	…	f

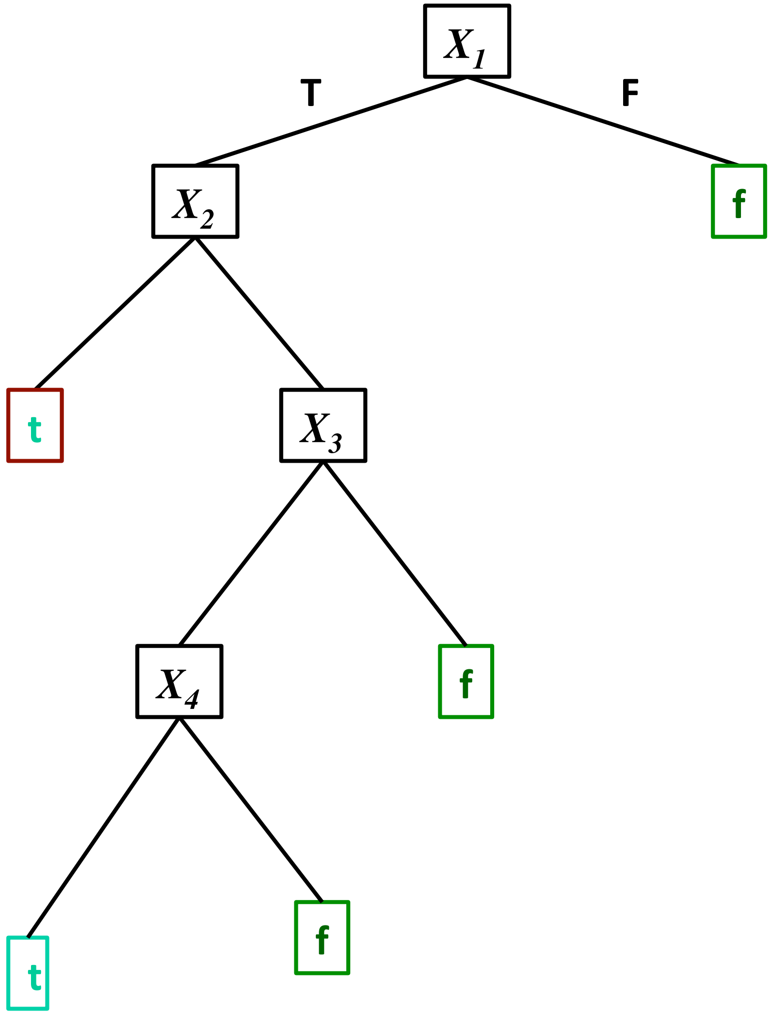
noisy value

Overfitting Example: Noisy Data

Correct tree



Tree that fits noisy training data



Overfitting Example: Noise-Free Data

Target function is $Y = X_1 \wedge X_2$

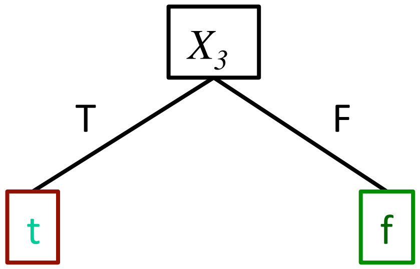
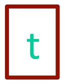
- What about irrelevant features?
- Training set:

X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

Overfitting Example: Noise-Free Data

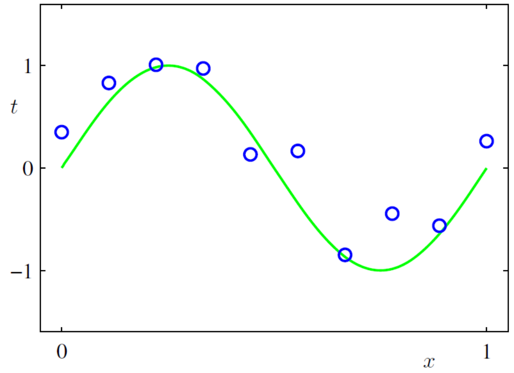
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance
- Assume, P(X_3 = t) = 0.5 for both classes and P(Y = t) = 0.67

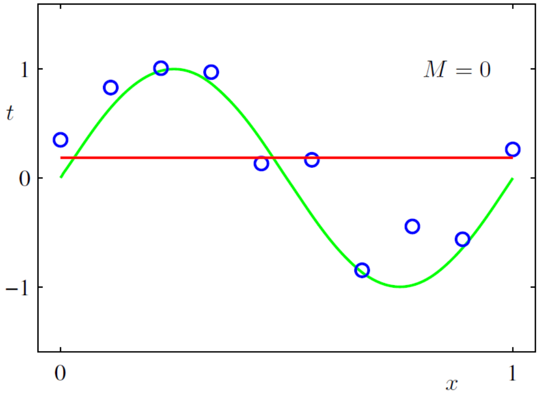
X_1	X_2	X_3	X_4	X_5	…	Y
t	t	t	t	t	…	t
t	t	t	f	t	…	t
t	t	t	t	f	…	t
t	f	f	t	f	…	f
f	t	f	f	t	…	f

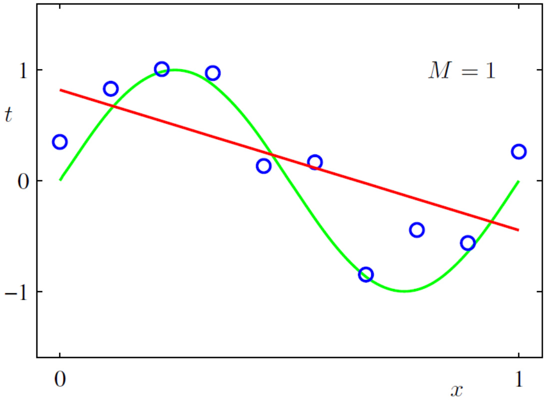
	Training set accuracy	**Test set accuracy**
	100%	50%
	66%	66%

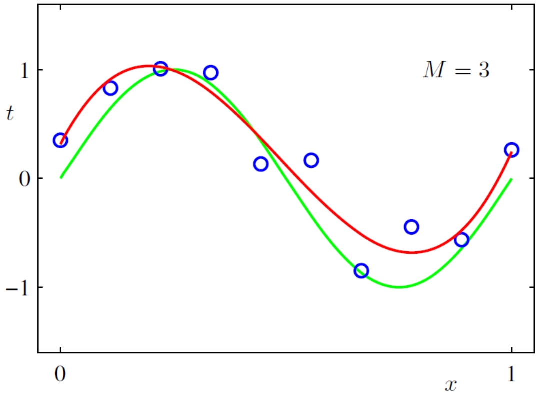
Overfitting Example: Polynomial Regression

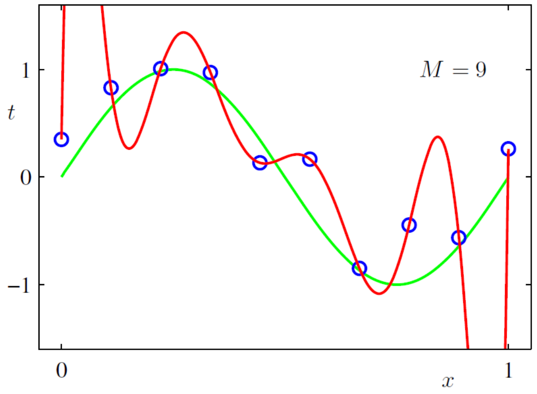
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance





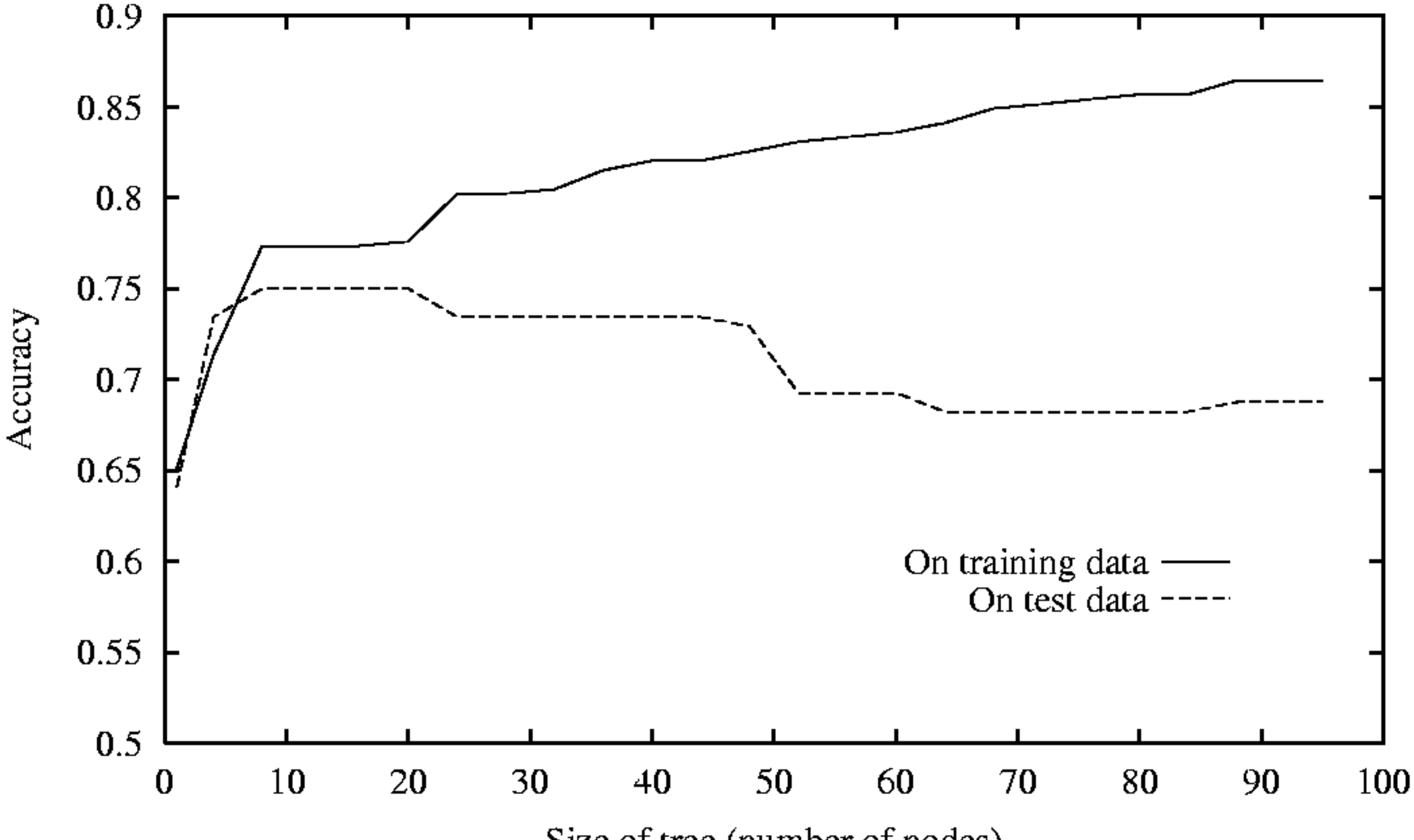






Overfitting: Tree Size vs. Accuracy

- Tree size vs accuracy



General Phenomenon

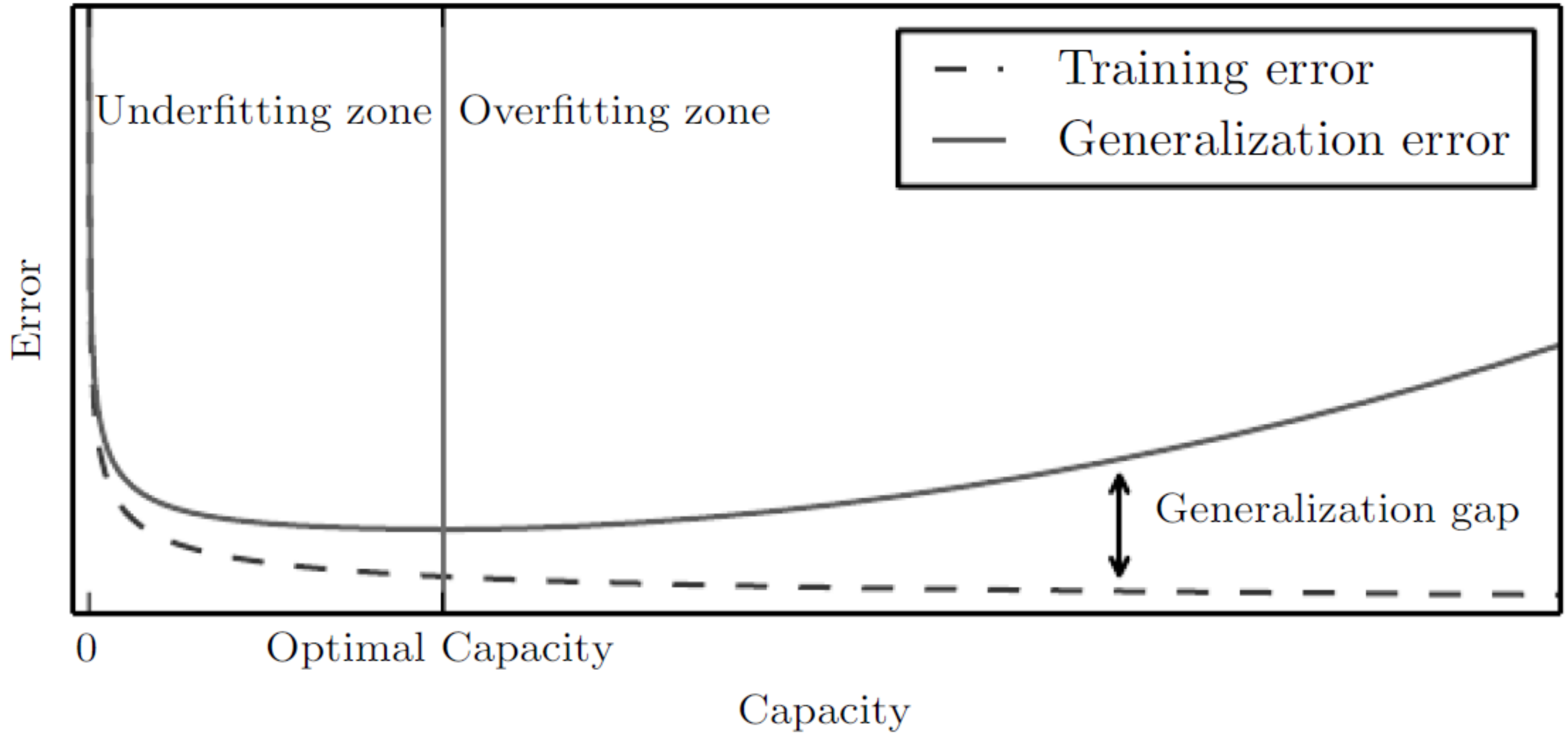


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, and Fred Sala