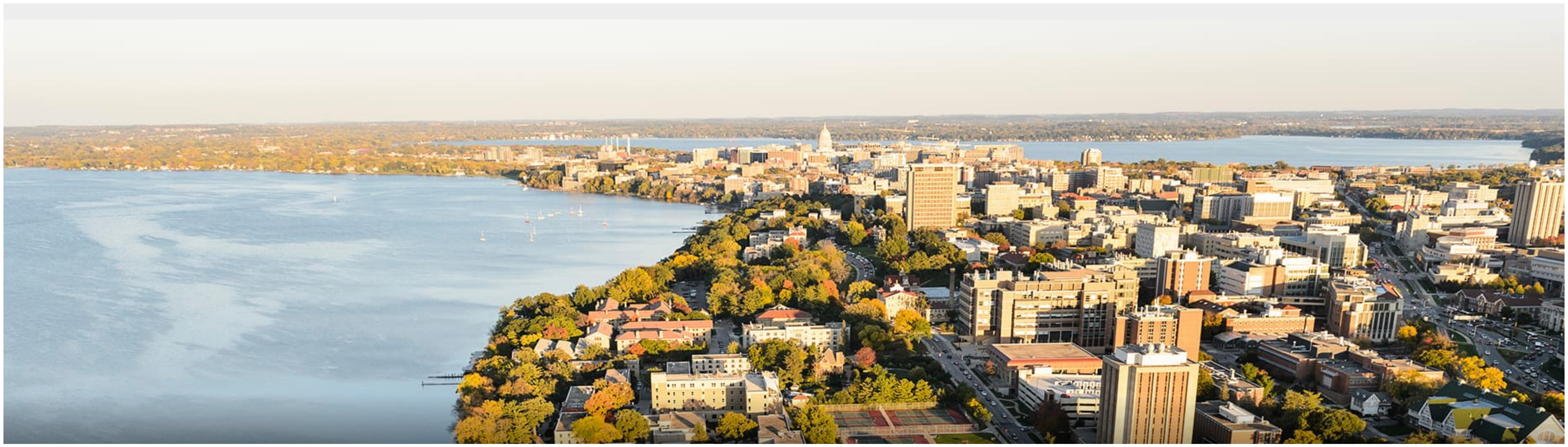


CS 760: Machine Learning
Supervised Learning II

Ilias Diakonikolas

University of Wisconsin-Madison

9/20/2022

Announcements

- **Announcement**:
  - HW 2 released next week
- Class roadmap:

Tuesday Sept. 20	Supervised Learning II
Thursday Sept. 22	Evaluation
Tuesday Sept. 27	Regression I
Thursday Sept. 29	Regression II
Tuesday, Oct. 4	Naive Bayes

All Supervised Learning

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

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

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: 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.:

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

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

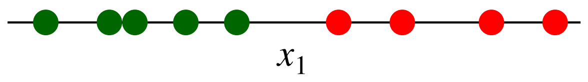
$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y^{(i)}$$

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

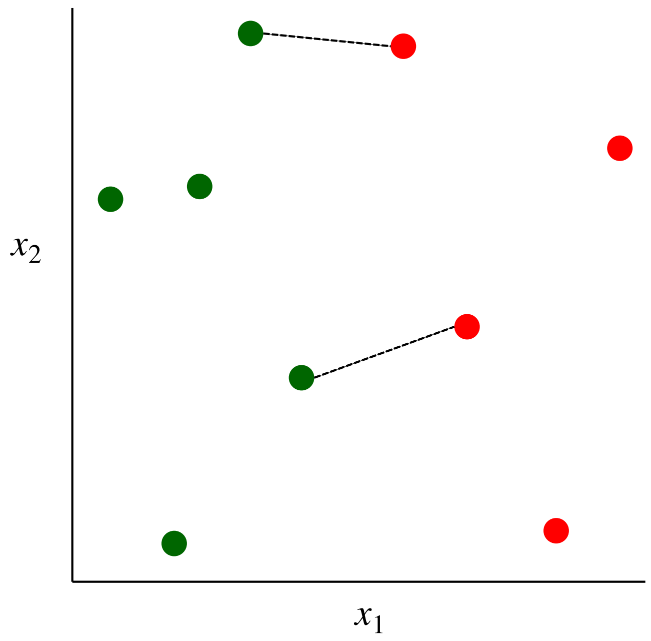
Dealing with **Irrelevant Features**

One relevant feature x_1

1-NN rule classifies each instance correctly

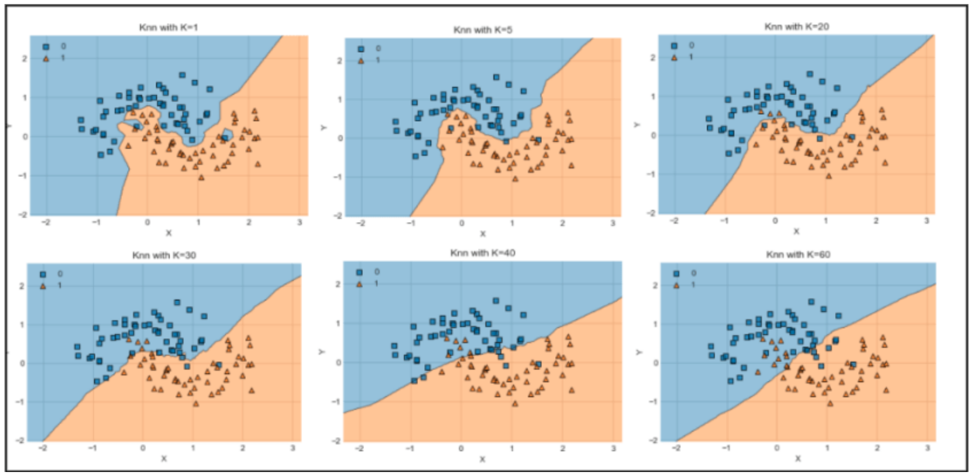


Effect of an irrelevant feature x_2 on distances and nearest neighbors



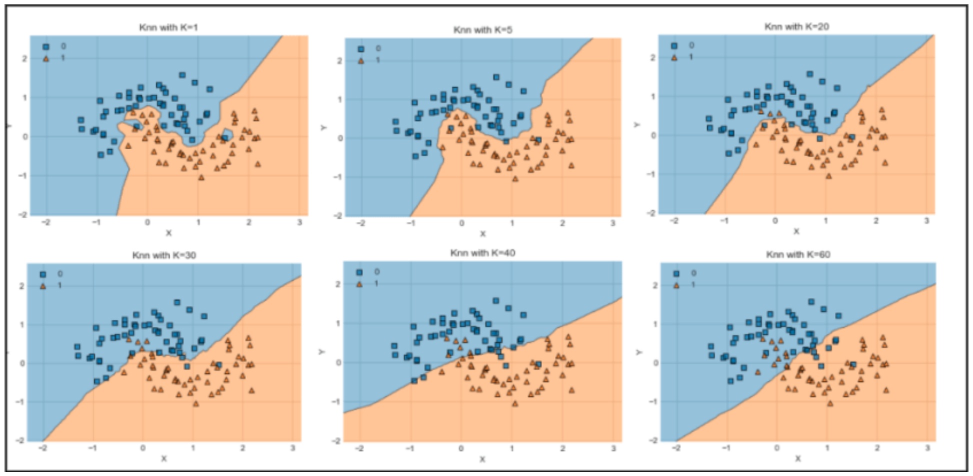
Instance-Based Learning: Strengths

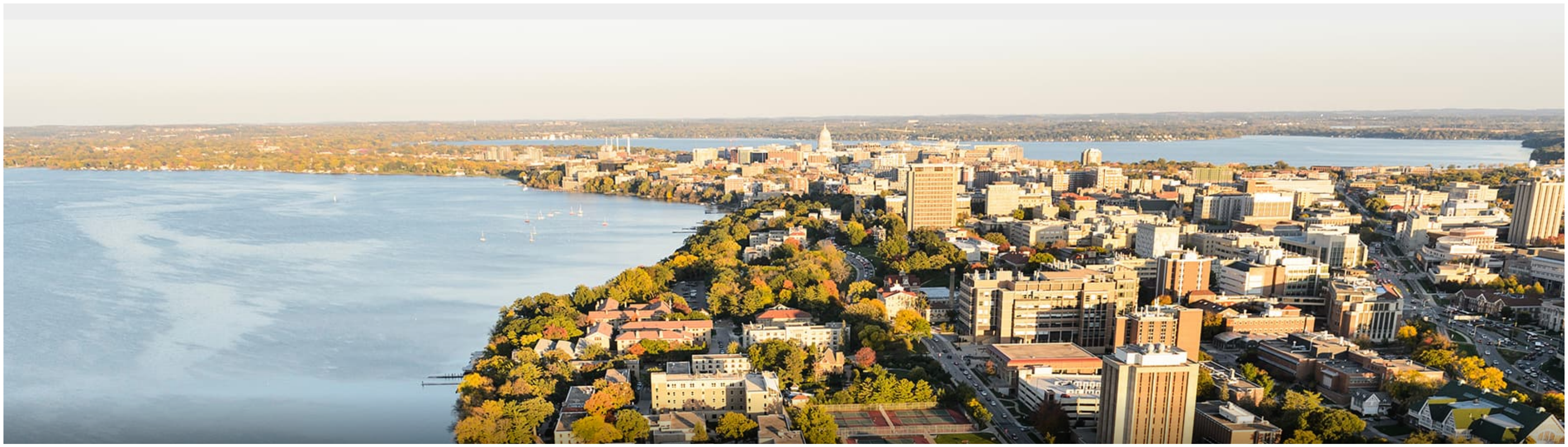
- Simple to implement
- No training!
- Easily done online
- Robust to noisy data (for enough samples)
- Often good in practice!



Instance-Based Learning: Weaknesses

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





Break & Quiz

Q1-1: Select the correct option.

A. Instance based learning 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.

Q1-1: Select the correct option.

A. Instance based learning is sensitive to range of feature values.

B. Training is very efficient.

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

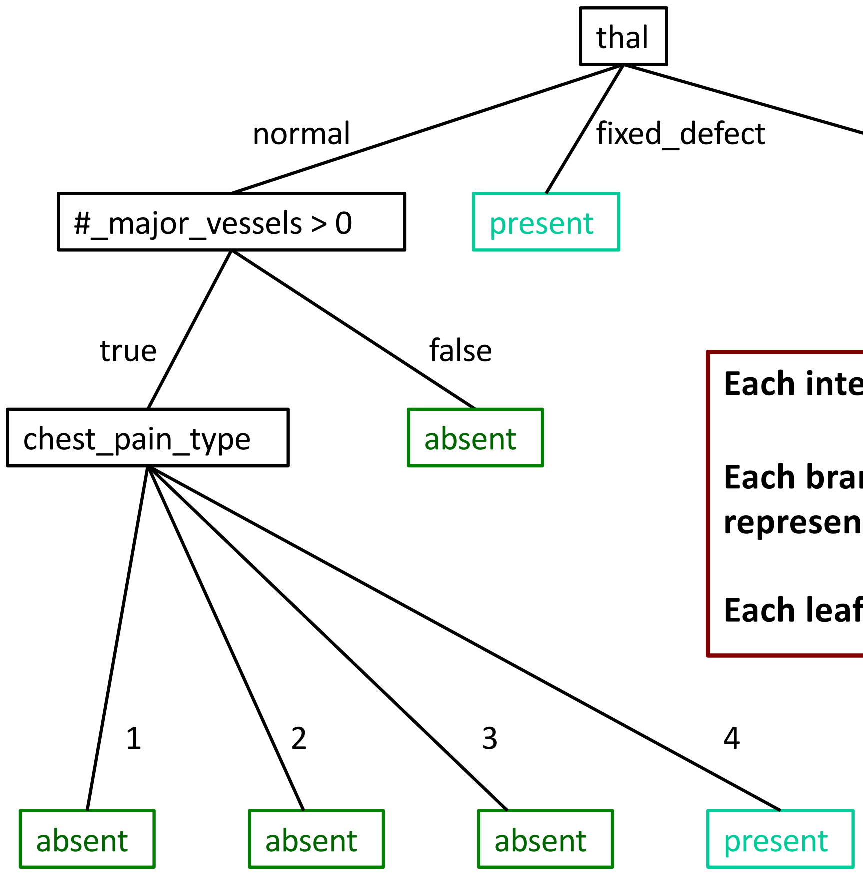
1. Statement A is true. Statement B, C are false.
2. [✓ Statement A, B are true. Statement C is false.]
3. Statement B, C are true. Statement A is false.
4. All Statements are true.

Occam's razor is an example of **preference bias,** i.e – Prefer one hypothesis over another even though they have similar training accuracy. For example, we prefer smaller trees in the hypothesis space of decision trees

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

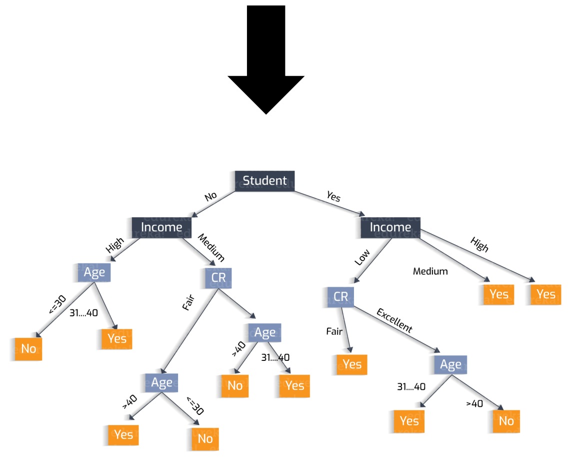
Each branch from an internal node represents one outcome of the test

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

Decision Trees: Learning

- **Learning Algorithm**:

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

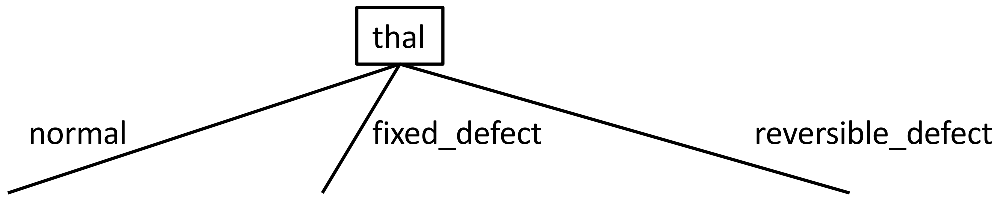


```
MakeSubtree(set of training instances $D$)
    $C$ = DetermineCandidateSplits($D$)
    if stopping criteria met
        make a leaf node $N$
        determine class label/probabilities for $N$
    else
        make an internal node $N$
        $S$ = FindBestSplit($D$, $C$)
        for each outcome $k$ of $S$
            $D_k$ = subset of instances that have outcome $k$
            $k^{th}$ child of $N$ = MakeSubtree($D_k$)
    return subtree rooted at $N$
```

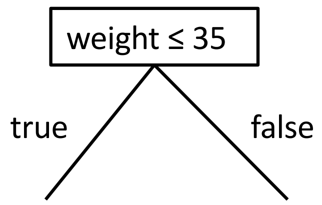
DT Learning: Candidate Splits

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

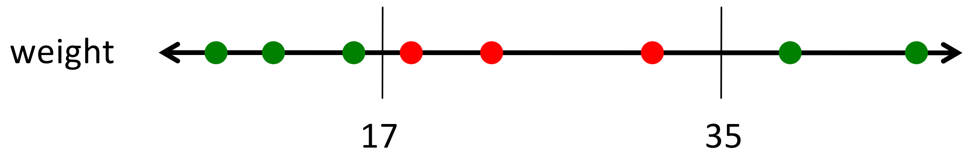


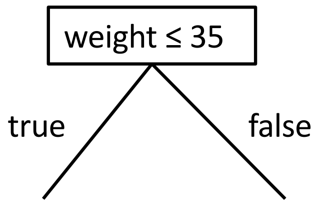
ID3, C4.5

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



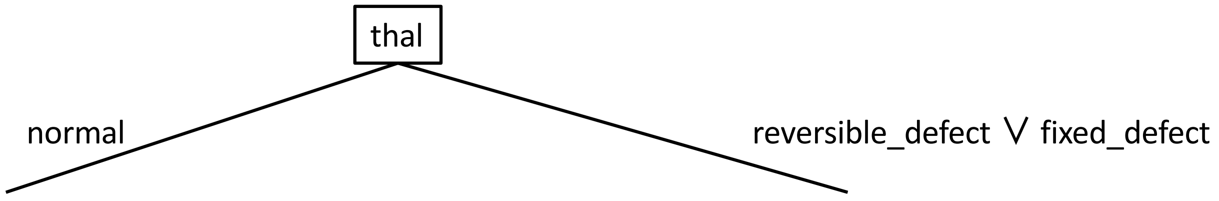


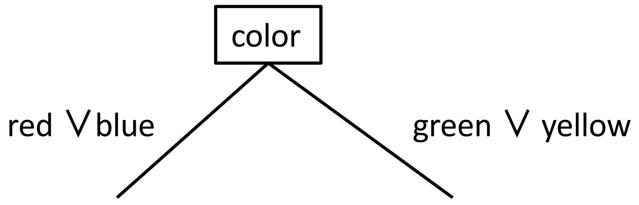
Numeric Feature Splits Algorithm

```
// 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 $X_i$
    $S$ = partition instances in $D$ into sets $s_1 \dots s_V$ where the instances in each set have the same value for $X_i$
    let $v_j$ denote the value of $X_i$ for set $s_j$
    sort the sets in $S$ using $v_j$ as the key for each $s_j$
    for each pair of adjacent sets $s_j$, $s_{j+1}$ in sorted $S$
        if $s_j$ and $s_{j+1}$ contain a pair of instances with different class labels
            // assume we're using midpoints for splits
            add candidate split $X_i \leq (v_j + v_{j+1})/2$ to $C$
    return $C$
```

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)





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

- "Nunquam ponenda est pluralitis sin necesitate"
- "Entities should not be multiplied beyond necessity"
- "when you have two competing theories that make the same predictions, the simpler one is the better"

DT Learning: Finding the Best Splits

Occam's razor

- "Nunquam ponenda est pluralitis sin necesitate"
- "Entities should not be multiplied beyond necessity"
- "when you have two competing theories that make the same predictions, the simpler one is the better"

- **Ptolemy** (~1000 years earlier)
- "We consider it a good principle to explain the phenomena by the simplest hypothesis possible."

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

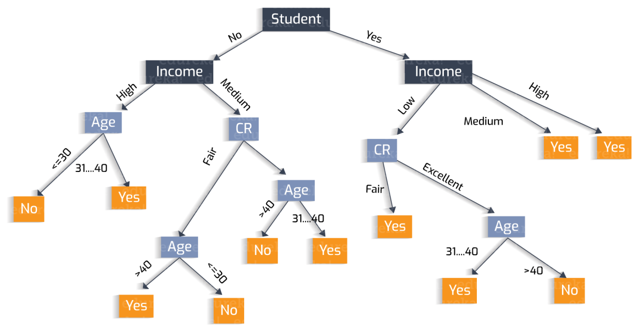
DT Learning: Finding Optimal Splits?

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

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

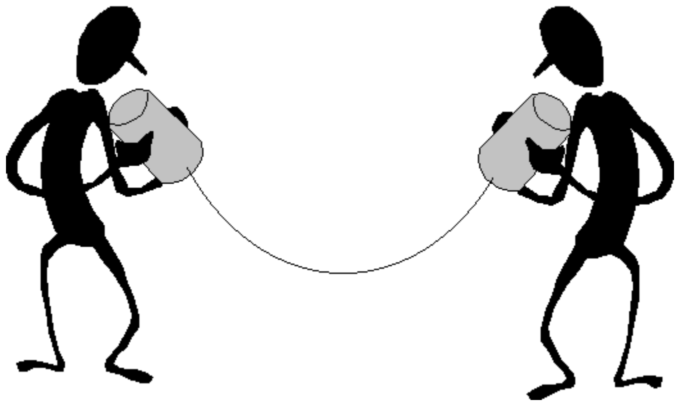
  [Hyafil & Rivest, *Information Processing Letters, 1976*]

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



Information Theory: Super-Quick Intro

- **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**

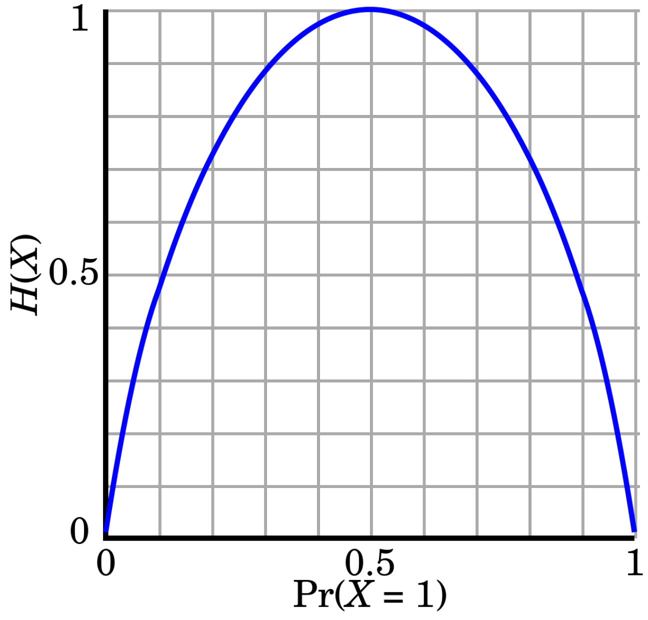
$$-\sum_{y\in\mathcal{Y}} P(y)\log_2 P(y)$$

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

$$H(Y|X = black) = -0.5\times \log 0.5 - 0.25\times \log 0.25 - 0.25\times \log 0.25 - 0 = 1.5$$
$$H(Y|X = white) = -0.5\times \log 0.5 - 0.25\times \log 0.25 - 0 - 0.25\times \log 0.25 = 1.5$$
$$H(Y|X) = 0.5\times H(Y|X = black) + 0.5\times H(Y|white) = 1.5$$

Information Theory: Mutual Information

- Similar comparison between R.V.s:

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

- How much uncertainty of Y that X can reduce.

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

$$I(Y;X) = H(Y) - H(Y|X) = 1.75 - 1.5 = 0.25$$

DT Learning: Back to Splits

Want to choose split S that maximizes

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

ie, mutual information.

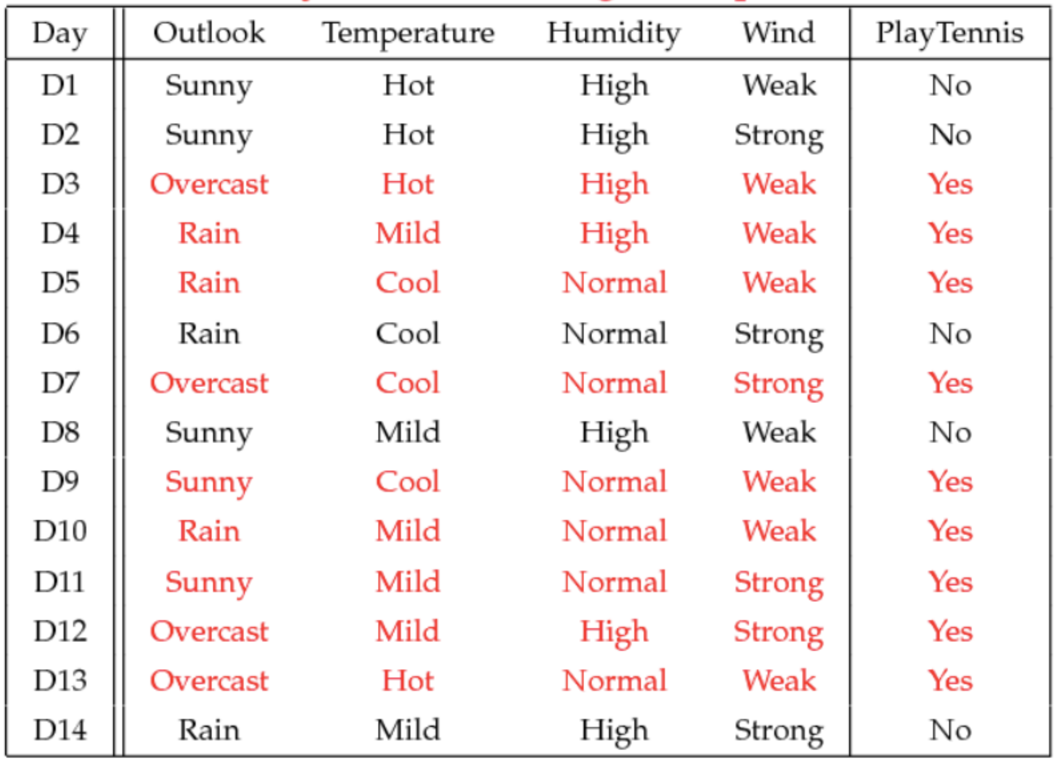
- Note: D denotes that this is the **empirical** entropy
  - We don't know the real distribution of *Y*, just have our dataset

- Equivalent to maximally reducing conditional entropy of Y

DT Learning: InfoGain Example

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

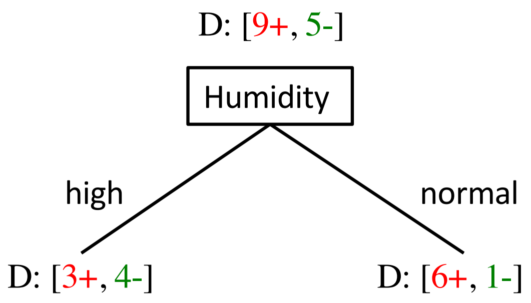
PlayTennis: training examples



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?



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

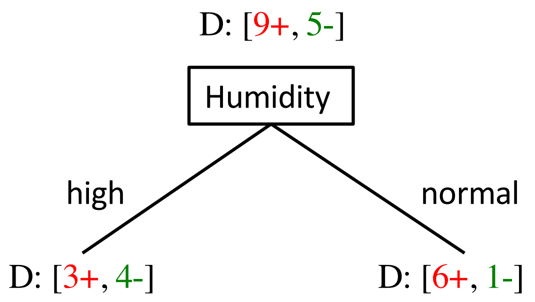
$$H_D(Y \mid \text{high}) = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$
$$= 0.985$$

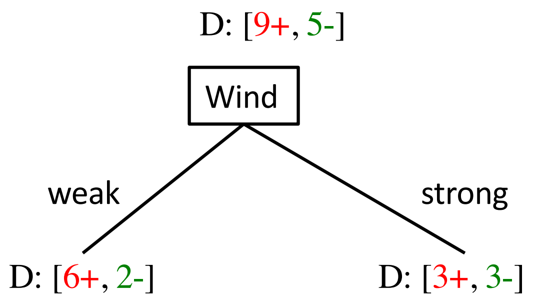
$$H_D(Y \mid \text{normal}) = -\frac{6}{7}\log_2\left(\frac{6}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)$$
$$= 0.592$$

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

DT Learning: Comparing Split InfoGains

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





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

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

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

DT Learning: InfoGain Limitations

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

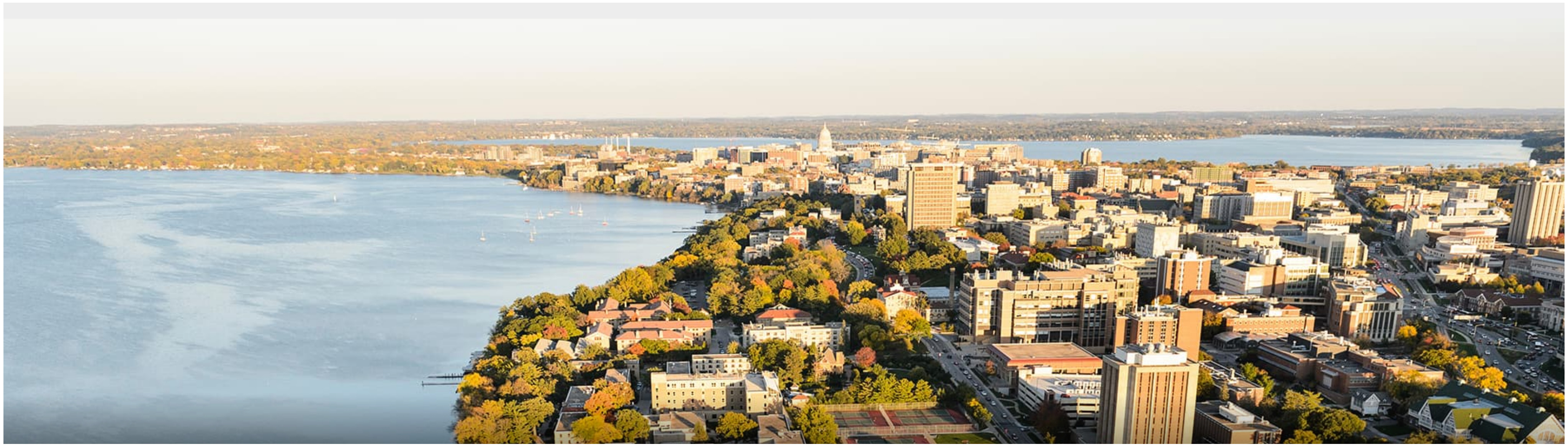
- Use **GainRatio**: normalize information gain by entropy

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

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
ID3 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



Break & Quiz

Q2-1: How many distinct (binary classification) decision trees are possible with *4* Boolean attributes? Here distinct means representing different functions.

1. 2^4
2. 2^8
3. 2^{16}
4. 2^{32}

Q2-1: How many distinct (binary classification) decision trees are possible with *4* Boolean attributes? Here distinct means representing different functions.

1. 2^4
2. 2^8
3. [✓ 2^{16}]
4. 2^{32}

#distinct decision trees
= #distinct Boolean functions
= #functions of 2^4 = 16 inputs, binary label for each input
= $\mathbf{2^{16}}$

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.

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.]

1. We can still have false correlation that leads to overfitting.
2. Overfitting would undermine the generalization ability.
3. Generalization error would first decrease and then increase as the model capacity increases.
4. Increasing training data size would help better approximate the true distribution.

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

DT Learning: Stopping Criteria

Form a leaf when

- All of the given subset of instances are same class
- We’ve exhausted all of the candidate splits

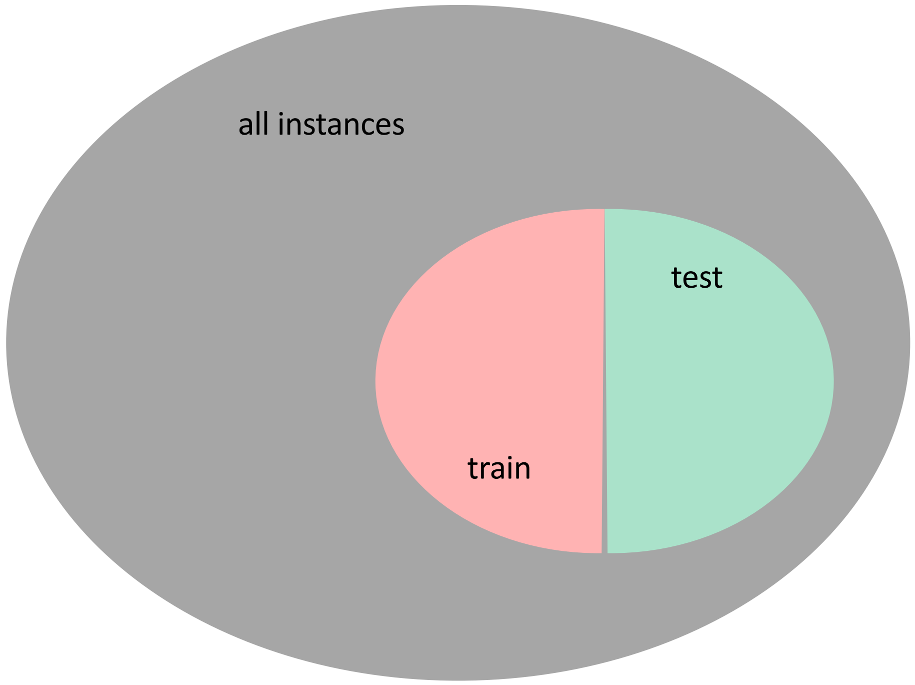
Evaluation: Accuracy

- 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 a learned decision tree be on previously unseen instances?
  - How accurate would it be on its training set?

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**



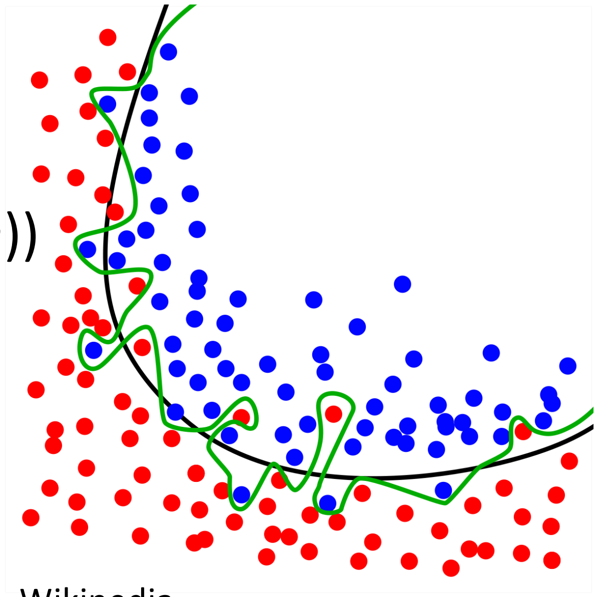
Overfitting

Notation: error of model *h* over

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

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

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



Wikipedia

Overfitting Example: Noisy Data

Target function is $Y = X_1 \wedge X_2$

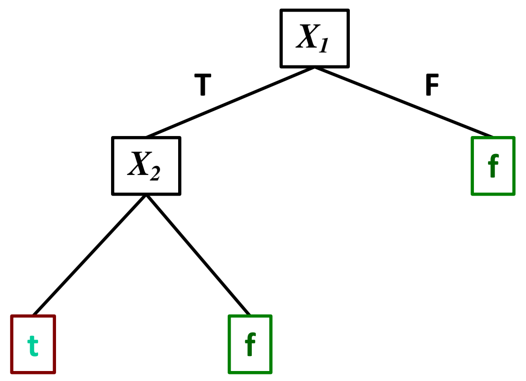
- There is noise in some feature values
- Training set

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

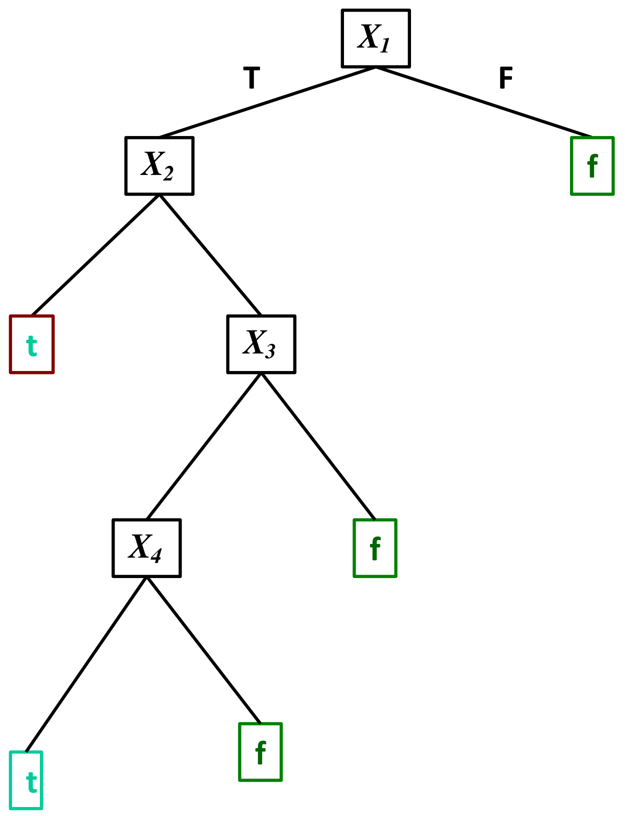
noisy value

Overfitting Example: Noisy Data

Correct tree



Tree that fits noisy training data



Overfitting Example: Noise-Free Data

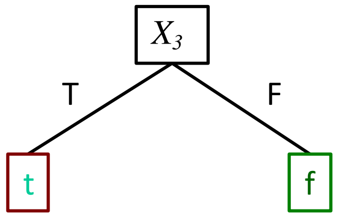
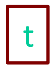
Target function is $Y = X_1 \wedge X_2$

- P(X_3 = t) = 0.5 for both classes
- P(Y = t) = 0.67
- 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

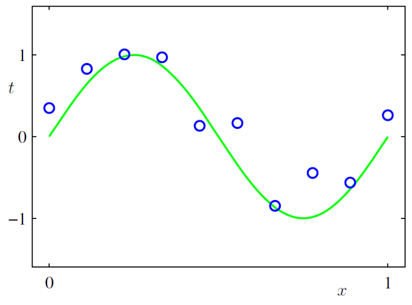
Overfitting Example: Noise-Free Data

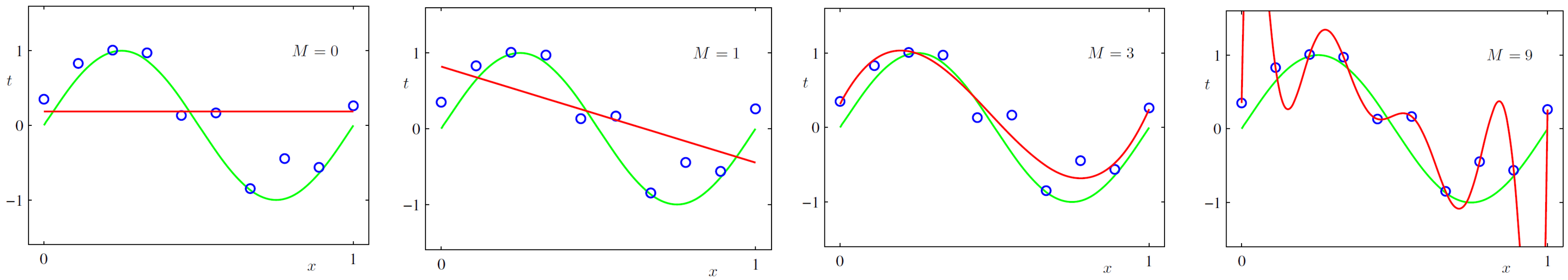
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance

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

Overfitting Example: Polynomial Regression

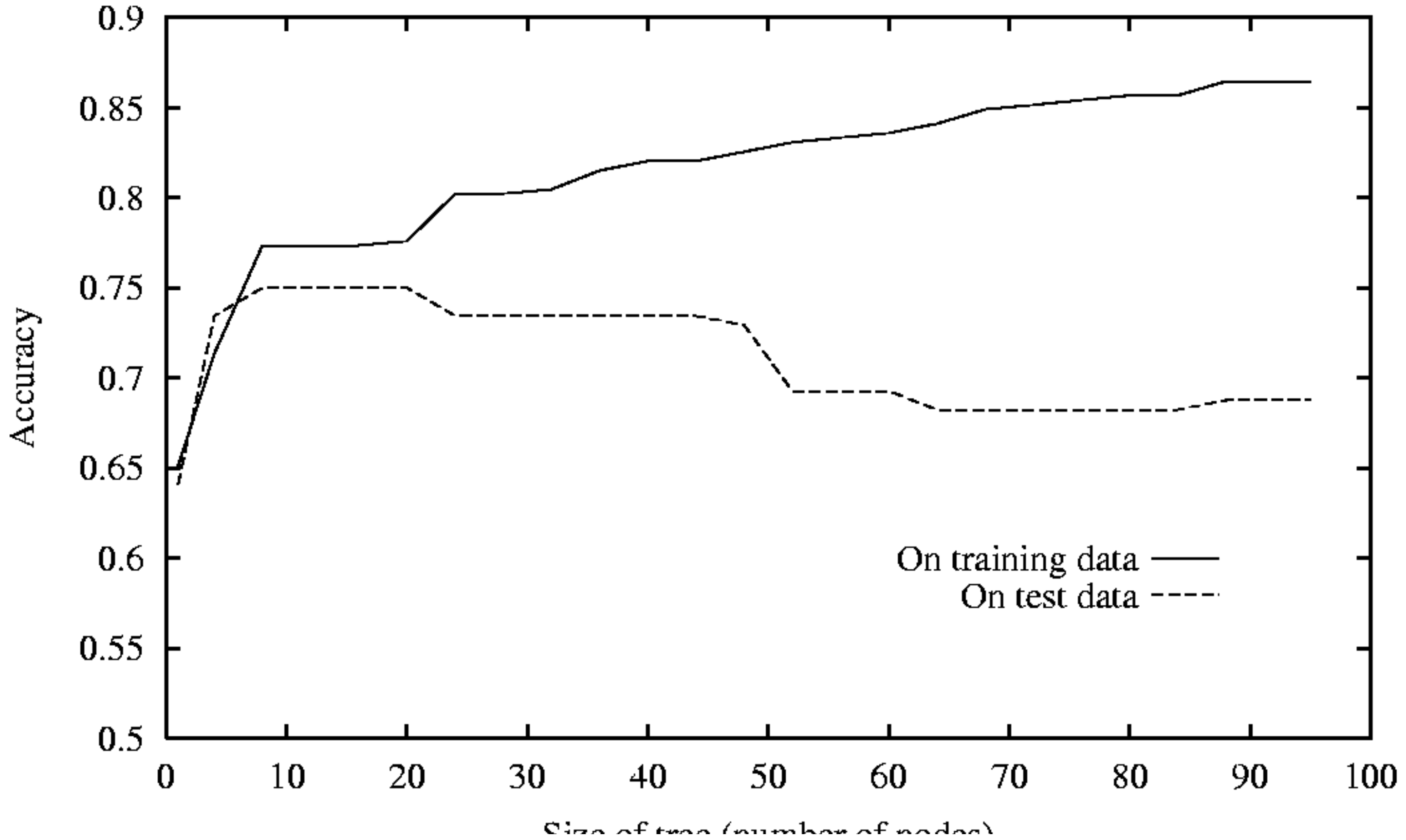
- Training set is a **limited sample.** Might be (combinations of) features that are correlated with the target concept by chance





Overfitting: Tree Size vs. Accuracy

- Tree size vs accuracy



General Phenomenon

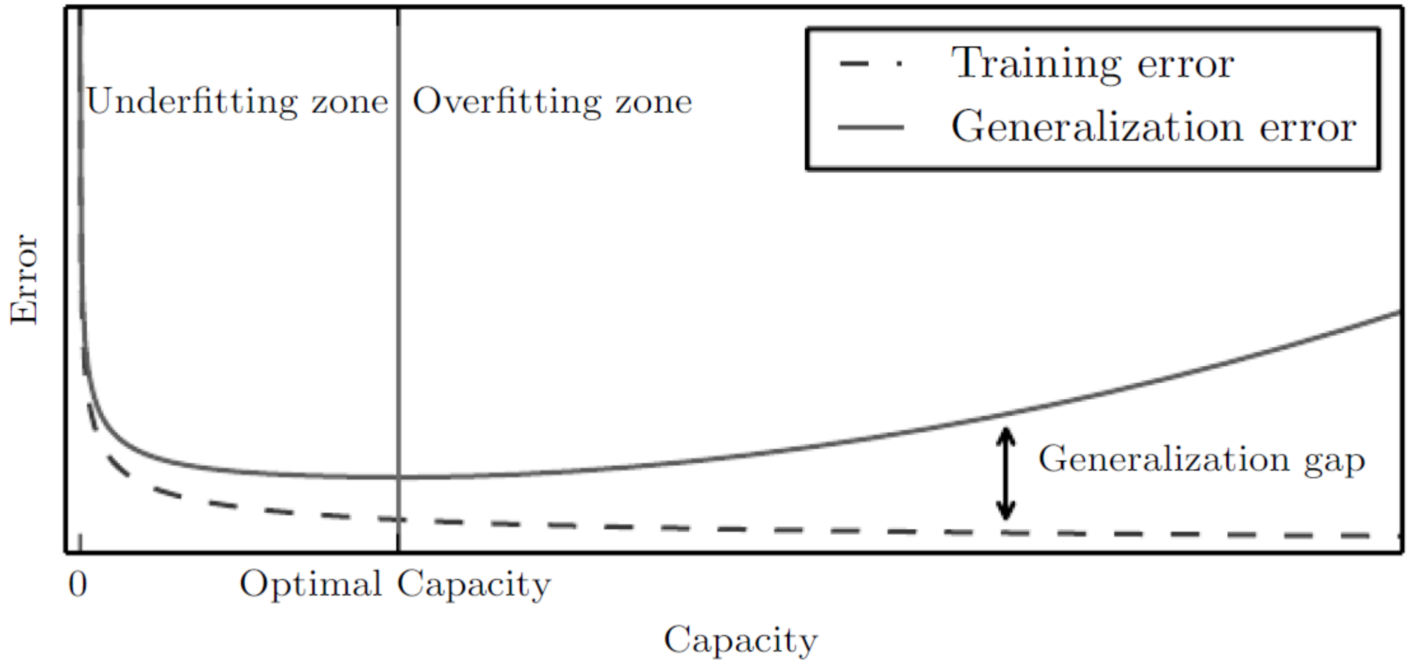
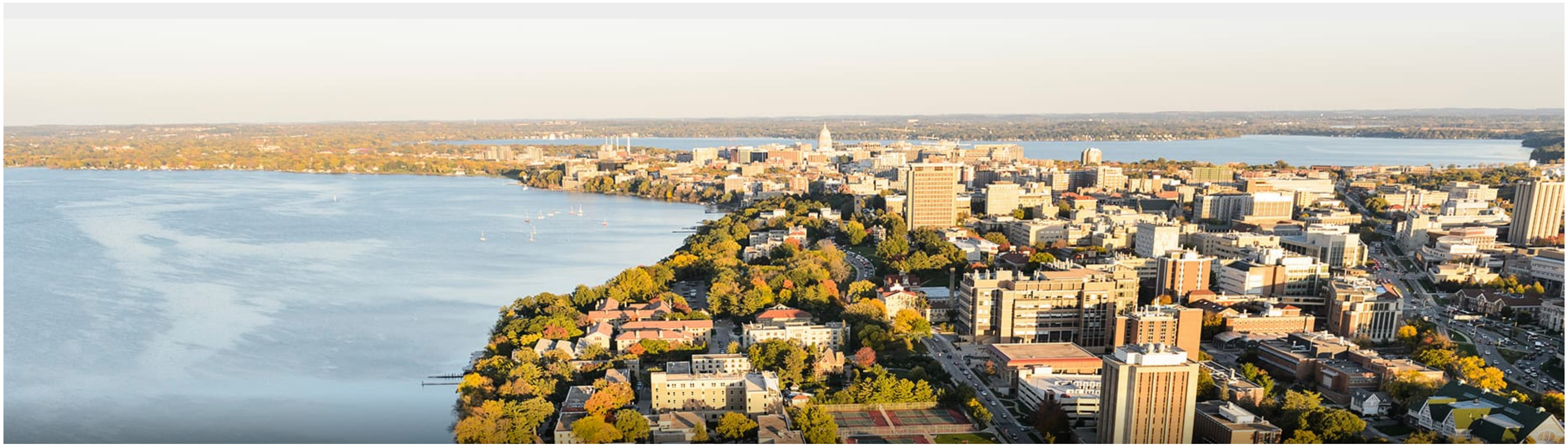


Figure from *Deep Learning*, Goodfellow, Bengio and Courville



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, Fred Sala