Machine Learning Department
School of Computer Science
Carnegie Mellon University

Overfitting
+
k-Nearest Neighbors

Matt Gormley
Lecture 4
Jan. 27, 2025

Course Staff

Education Associate

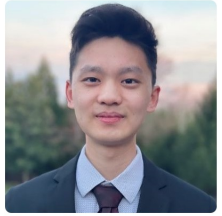

Brynn Edmunds

Instructors


Henry Chai


Matt Gormley

HW2/HW6


Sebastian Lu


Mihir Mathur

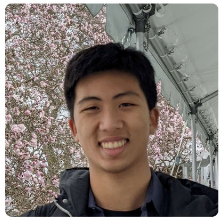

Joaquin Wang


Zhifei Li

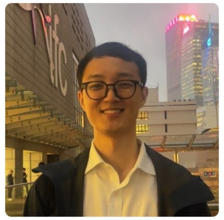

Rohini Banerjee

HW3/HW7


Bhargav Hadya

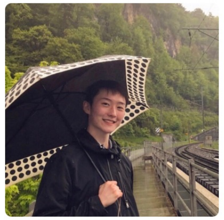

Maxwell Chien

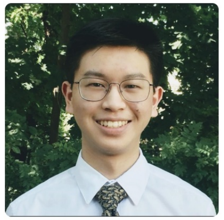

Zachary Gelman

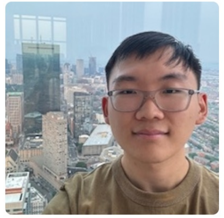

Changwook Shim


Santiago Arambulo

HW4/HW8


Andrew Wang


Max Tang


Albert Zhang


Andra Liu


Vianna Seifi

HW5/HW9


Shivi Jindal


Yuhan Gao


Siyan Chen


Jenny Yang


Zoe Xu

Reminders

- **Homework 2: Decision Trees**
  - Out: Wed, Jan. 22
  - Due: Mon, Feb. 3 at 11:59pm

Q&A

Q: Why don't my entropy calculations match those on the slides?

A: Remember that H(Y) is conventionally reported in "bits" and computed using log base 2.
e.g., $H(Y) = - P(Y=0) \log_2 P(Y=0) - P(Y=1) \log_2 P(Y=1)$

Q: When and how do we decide to stop growing trees? What if the set of values an attribute could take was really large or even infinite?

A: We'll address this question for discrete attributes today. If an attribute is real-valued, there's a clever trick that only considers O(L) splits where L = # of values the attribute takes in the training set. Can you guess what it does?

Q&A

Q: What does decision tree training do if a branch receives no data?

A: Then we hit the base case and create a leaf node. So the real question is what does majority vote do when there is no data? Of course, there is no majority label, so (if forced to) we could just return one randomly.

Q: What do we do at test time when we observe a value for a feature that we didn't see at training time.

A: This really just a variant of the first question. That said, a real DT implementation needs to elegantly handle this case. We could do so by either (a) assuming that all possible values will be seen at train time, so there should be a branch for all attributes even if the partition of the dataset doesn't include them all or (b) recognize the unseen value at test time and return some appropriate label in that case.

EMPIRICAL COMPARISON OF SPLITTING CRITERIA

Experiments: Splitting Criteria

Bluntine & Niblett (1992) compared 4 criteria (random, Gini, mutual information, Marshall) on 12 datasets

Medical Diagnosis Datasets: (4 of 12)

- **hypo**: data set of 3772 examples records expert opinion on possible hypo- thyroid conditions from 29 real and discrete attributes of the patient such as sex, age, taking of relevant drugs, and hormone readings taken from drug samples.
- **breast:** The classes are reoccurrence or non-reoccurrence of breast cancer sometime after an operation. There are nine attributes giving details about the original cancer nodes, position on the breast, and age, with multi-valued discrete and real values.
- **tumor:** examples of the location of a primary tumor
- **lymph**: from the lymphography domain in oncology. The classes are normal, metastases, malignant, and fibrosis, and there are nineteen attributes giving details about the lymphatics and lymph nodes

Table 1. Properties of the data sets

Data Set	Classes	Attr.s	Training Set	Test Set
hypo	4	29	1000	2772
breast	2	9	200	86
tumor	22	18	237	102
lymph	4	18	103	45
LED	10	7	200	1800
mush	2	22	200	7924
votes	2	17	200	235
votesl	2	16	200	235
iris	3	4	100	50
glass	7	9	100	114
xd6	2	10	200	400
pole	2	4	200	1647

Table from Bluntine & Niblett (1992)

Experiments: Splitting Criteria

Table 3. Error for different splitting rules (pruned trees).

	Splitting Rule			
Data Set	GINI	Info. Gain	Marsh.	Random
hypo	1.01 ± 0.29	0.95 ± 0.22	1.27 ± 0.47	7.44 ± 0.53
breast	28.66 ± 3.87	28.49 ± 4.28	27.15 ± 4.22	29.65 ± 4.97
tumor	60.88 ± 5.44	62.70 ± 3.89	61.62 ± 3.98	67.94 ± 5.68
lymph	24.44 ± 6.92	24.00 ± 6.87	24.33 ± 5.51	32.33 ± 11.25
LED	33.77 ± 3.06	32.89 ± 2.59	33.15 ± 4.02	38.18 ± 4.57
mush	1.44 ± 0.47	1.44 ± 0.47	7.31 ± 2.25	8.77 ± 4.65
votes	4.47 ± 0.95	4.57 ± 0.87	11.77 ± 3.95	12.40 ± 4.56
votes1	12.79 ± 1.48	13.04 ± 1.65	15.13 ± 2.89	15.62 ± 2.73
iris	5.00 ± 3.08	4.90 ± 3.08	5.50 ± 2.59	14.20 ± 6.77
glass	39.56 ± 6.20	50.57 ± 6.73	40.53 ± 6.41	53.20 ± 5.01
xd6	22.14 ± 3.23	22.17 ± 3.36	[illegible]2.06 ± 3.37	31.86 ± 3.62
pole	15.43 ± 1.51	15.47 ± 0.88	[illegible]	[illegible]

Key Takeaway: GINI gain and Mutual Information are statistically indistinguishable!

Info. Gain is another name for *mutual information*

Table from Bluntine & Niblett (1992)

Experiments: Splitting Criteria

Table 4. Difference and significance of error for GINI splitting rule versus others.

	Splitting Rule		
Data Set	Info. Gain	Marsh.	Random
hypo	−0.06 (0.82)	0.26 (0.99)	6.43 (1.00)
breast	−0.17 (0.23)	−1.51 (0.94)	0.99 (0.72)
tumor	1.81 (0.84)	0.74 (0.39)	7.06 (0.99)
lymph	−0.44 (0.83)	0.11 (0.05)	7.89 (0.99)
LED	0.12 (0.17)	[illegible]	
mush	0.00 (0.00)	5.86	
votes	0.11 (0.55)	7.30	
votes1	0.26 (0.47)	2.34	
iris	−0.10 (0.67)	0.50	
glass	1.01 (0.50)	0.96	
xd6	0.04 (0.11)	−0.07	
pole	0.03 (0.11)	−0.43	

Key Takeaway: GINI gain and Mutual Information are statistically indistinguishable!

Results are of the form A.AA (B.BB) where:

1. A.AA is the **average difference in errors** between the two methods
2. B.BB is the **significance** of the difference according to a two-tailed **paired t-test**

Table from Bluntine & Niblett (1992)

INDUCTIVE BIAS
(FOR DECISION TREES)

Decision Tree Learning Example

Dataset:

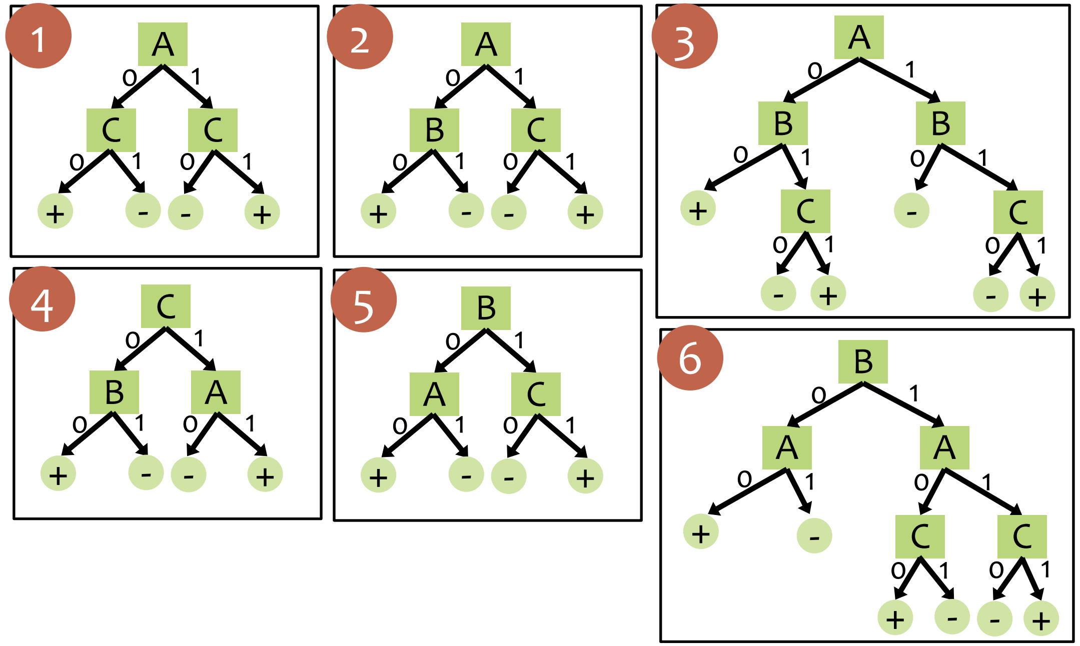
Output Y, Attributes A, B, C

Y	A	B	C
+	0	0	0
+	0	0	1
-	0	1	0
+	0	1	1
-	1	0	0
-	1	0	1
-	1	1	0
+	1	1	1

Poll Question 1

Which decision tree would you learn if you used error rate as the splitting criterion?

(Assume ties are broken alphabetically.)



Decision Tree Learning Example

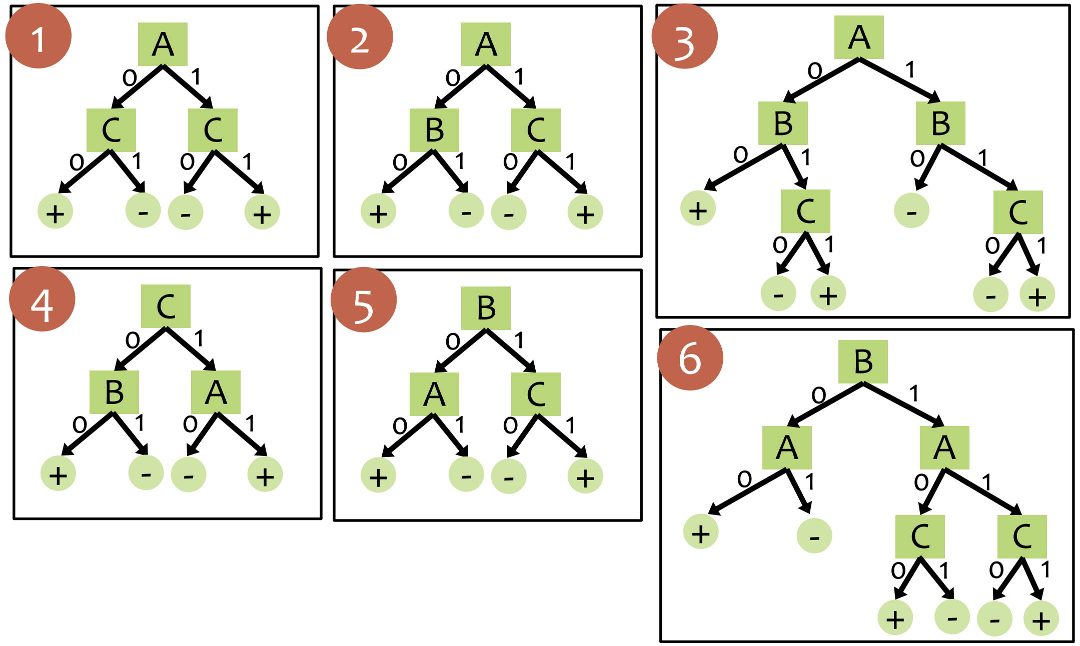
Dataset:

Output Y, Attributes A, B, C

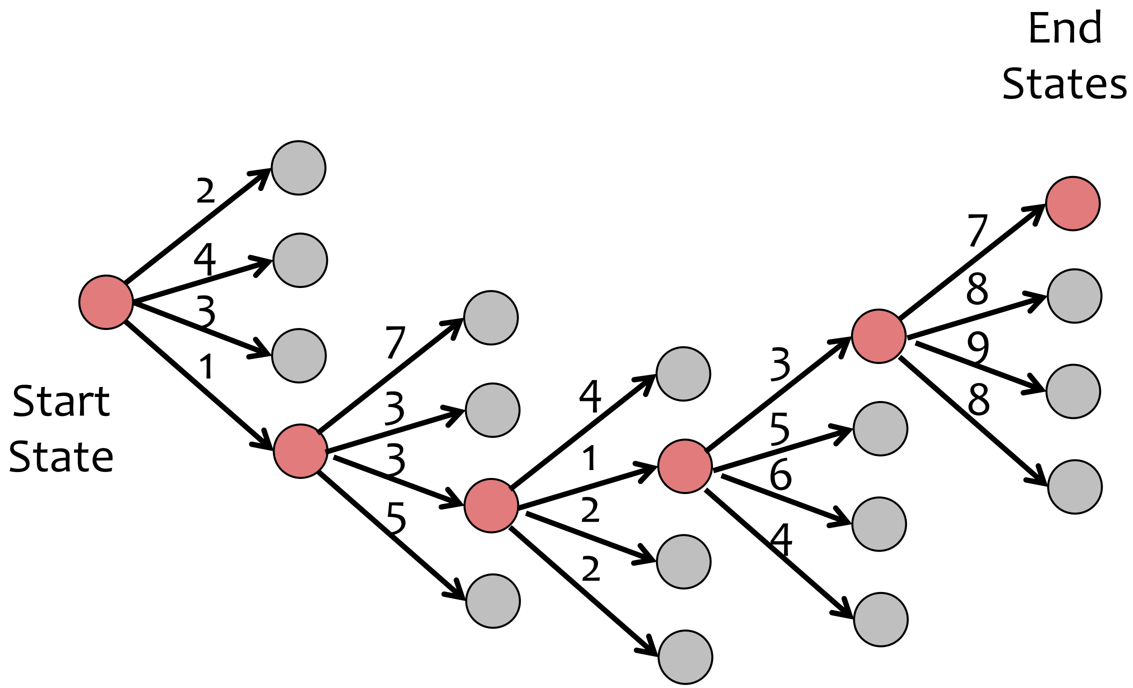
Y	A	B	C
+	0	0	0
+	0	0	1
-	0	1	0
+	0	1	1
-	1	0	0
-	1	0	1
-	1	1	0
+	1	1	1

Poll Question 2

Which decision tree is the smallest tree that achieves the lowest possible training error?



Background: Greedy Search



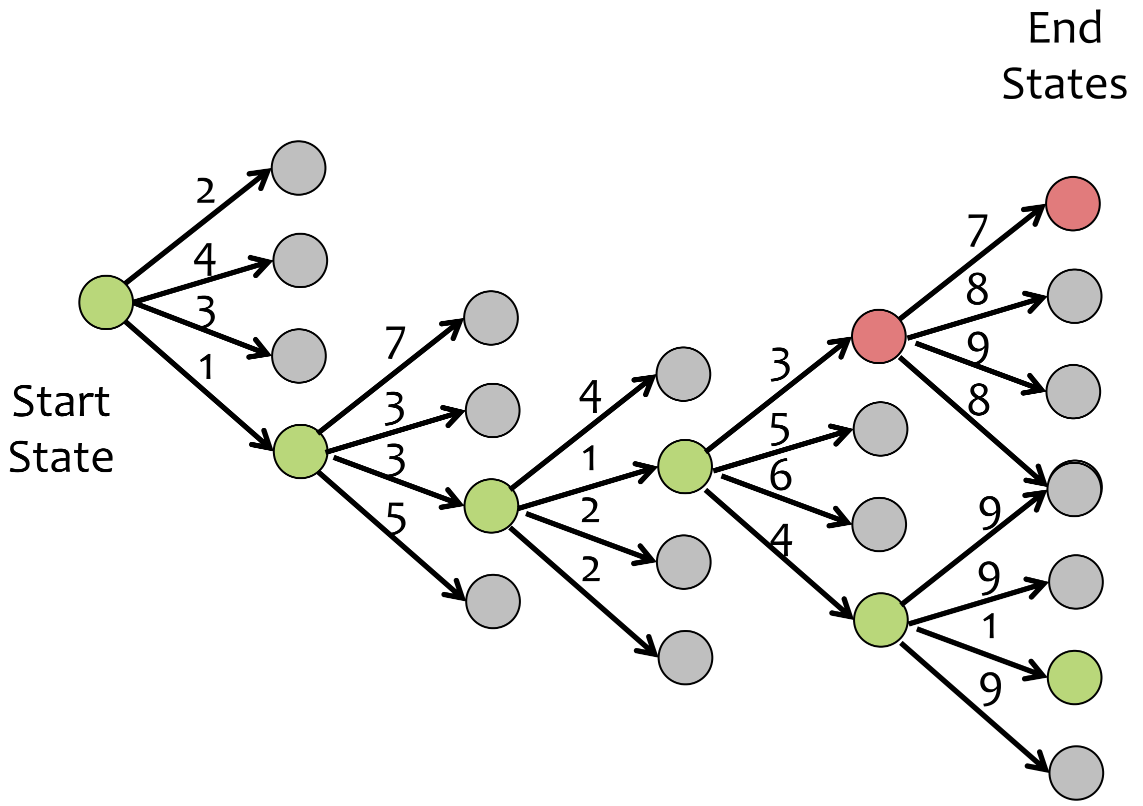
Goal:

- Search space consists of nodes and weighted edges
- Goal is to find the lowest (total) weight path from root to a leaf

Greedy Search:

- At each node, selects the edge with lowest (immediate) weight
- **Heuristic** method of search (i.e. does *not* necessarily find the best path)
- Computation time: **linear** in max path length

Background: Greedy Search



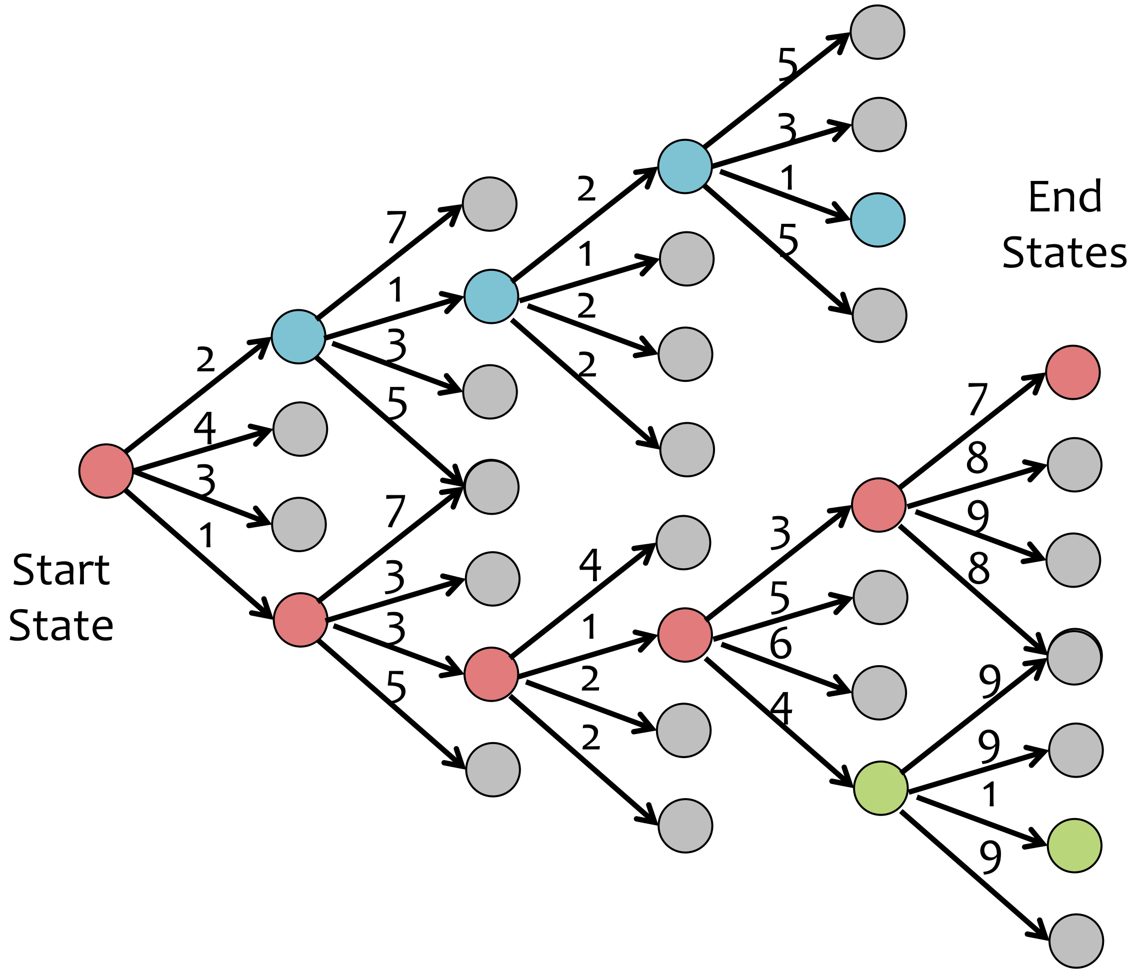
Goal:

- Search space consists of nodes and weighted edges
- Goal is to find the lowest (total) weight path from root to a leaf

Greedy Search:

- At each node, selects the edge with lowest (immediate) weight
- **Heuristic** method of search (i.e. does *not* necessarily find the best path)
- Computation time: **linear** in max path length

Background: Greedy Search



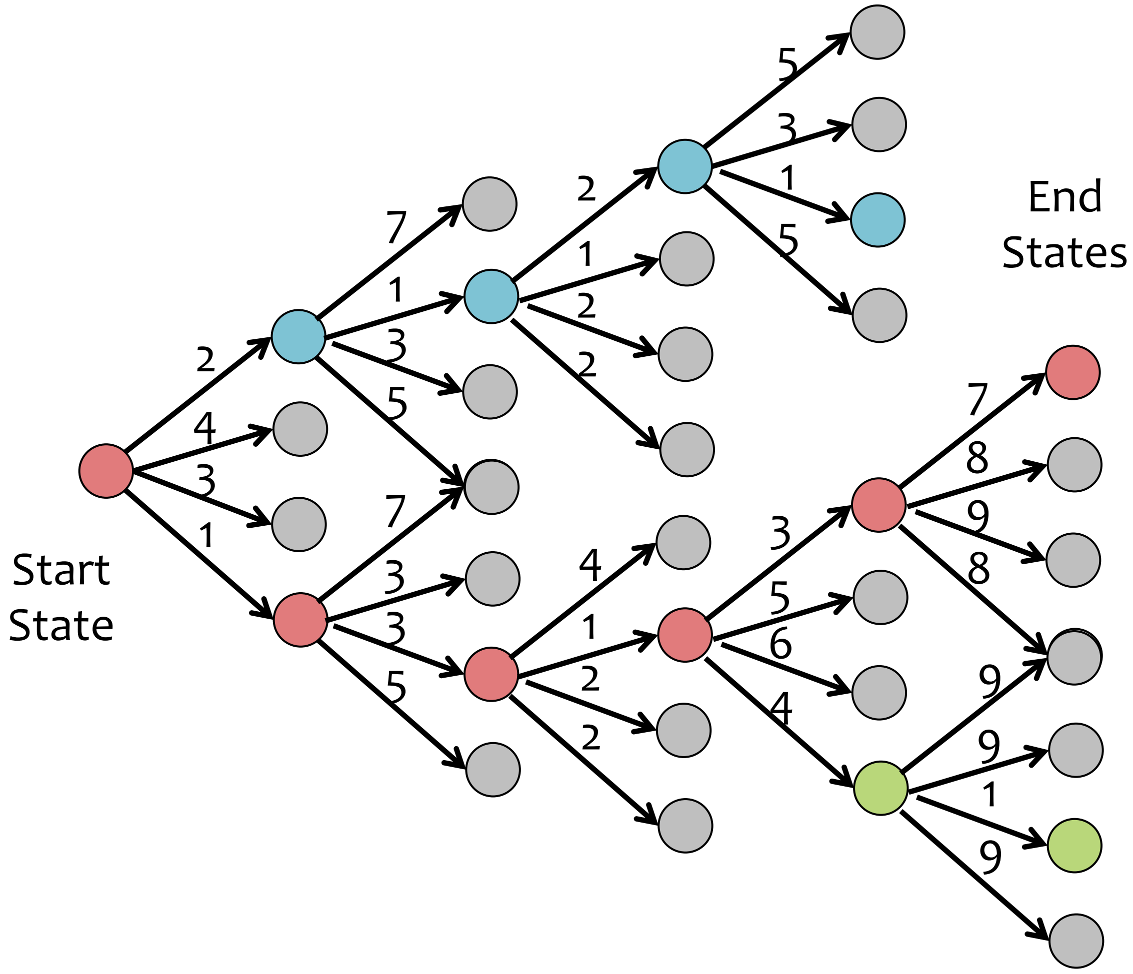
Goal:

- Search space consists of nodes and weighted edges
- Goal is to find the lowest (total) weight path from root to a leaf

Greedy Search:

- At each node, selects the edge with lowest (immediate) weight
- **Heuristic** method of search (i.e. does *not* necessarily find the best path)
- Computation time: **linear** in max path length

Background: Global Search



Goal:

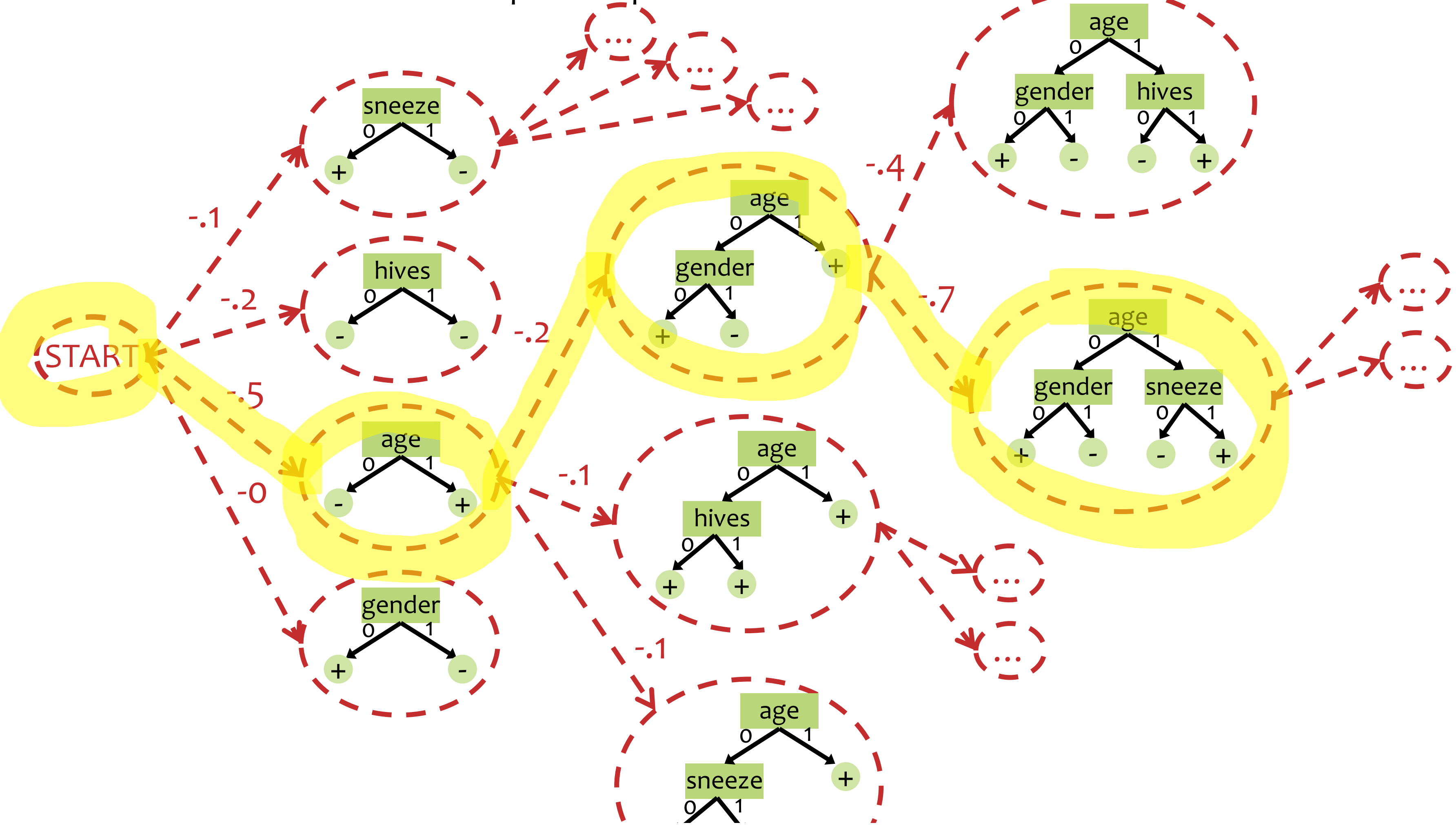
- Search space consists of nodes and weighted edges
- Goal is to find the lowest (total) weight path from root to a leaf

Global Search:

- Compute the weight of the path to **every** leaf
- **Exact** method of search (i.e. gauranteed to find the best path)
- Computation time: **exponential** in max path length

Decision Tree Learning as Search

1. **search space:** all possible decision trees
2. **node:** single decision tree
3. **edge:** connects one full tree to another, where child has one more split than parent
4. **edge weight:** (negative) splitting criterion
5. **DT learning:** greedy search, maximizing our splitting criterion at each step



Big Question:

How is it that your ML algorithm can generalize to unseen examples?

DT: Remarks

ID3 = Decision Tree Learning with Mutual Information as the splitting criterion

Question: Which tree does ID3 find?

Definition:

We say that the **inductive bias** of a machine learning algorithm is the principal by which it generalizes to unseen examples

What is the inductive bias of the ID3 algorithm?

OVERFITTING (FOR DECISION TREES)

Overfitting and Underfitting

Underfitting

- The model…
  - is too simple
  - is unable captures the trends in the data
  - exhibits too much bias
- *Example*: majority-vote classifier (i.e. depth-zero decision tree)
- *Example*: a toddler (that has **not** attended medical school) attempting to carry out medical diagnosis

Overfitting

- The model…
  - is too complex
  - is fitting the noise in the data or fitting "outliers"
  - does not have enough bias
- *Example*: our "memorizer" algorithm responding to an irrelevant attribute
- *Example*: medical student who simply memorizes patient case studies, but does not understand how to apply knowledge to new patients

Overfitting

- Given a hypothesis *h*, its…

  …error rate over all training data: $\text{error}(h, D_{train})$

  …error rate over all test data: $\text{error}(h, D_{test})$

  …true error over all data: $\text{error}_{true}(h)$

- We say h overfits the training data if…

$$\text{error}_{true}(h) > \text{error}(h, D_{train})$$

- Amount of overfitting =

$$\text{error}_{true}(h) - \text{error}(h, D_{train})$$

In practice, $\text{error}_{true}(h)$ is **unknown**

Slide adapted from Tom Mitchell

Overfitting in Decision Tree Learning

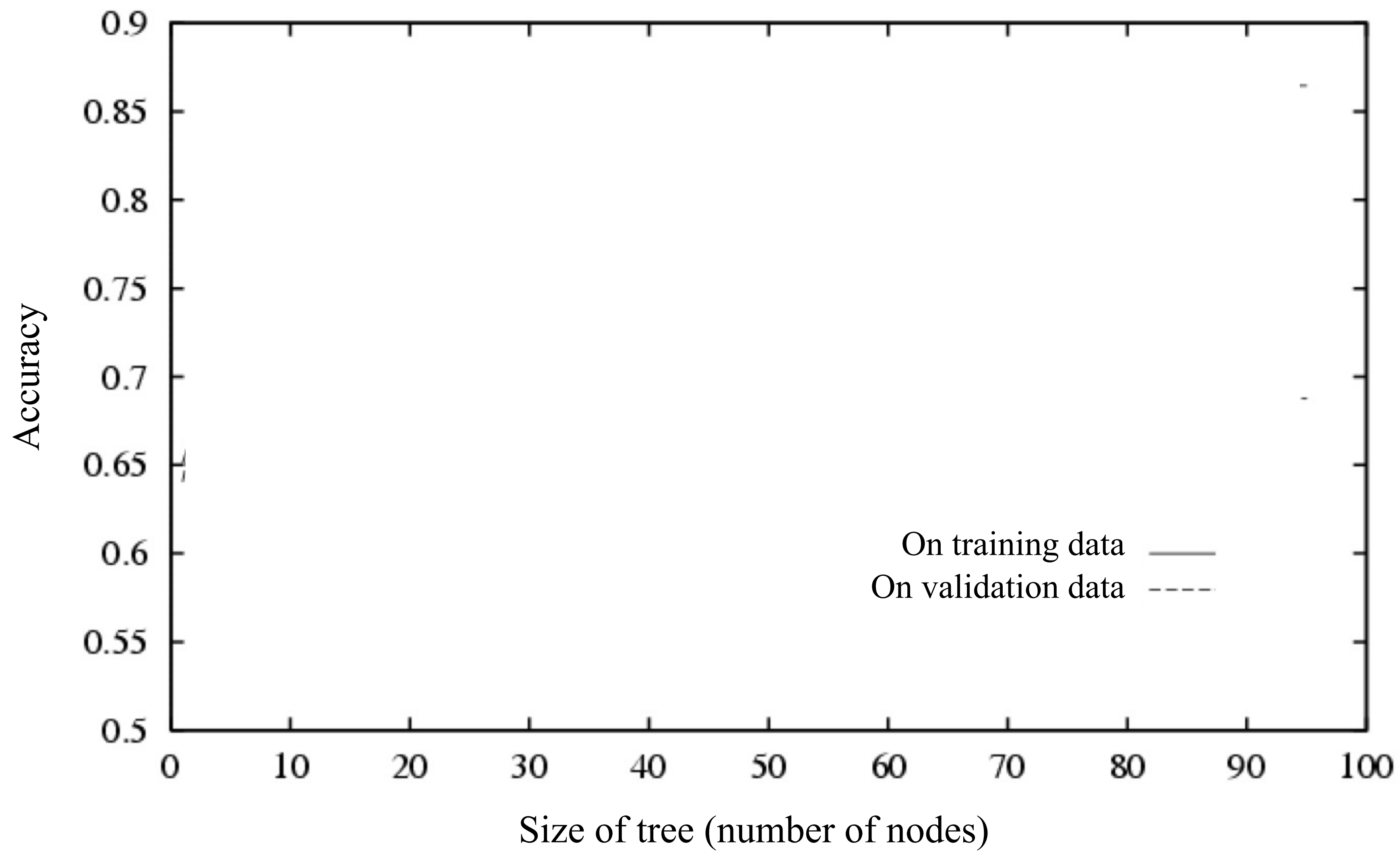


Figure from Tom Mitchell

Overfitting in Decision Tree Learning

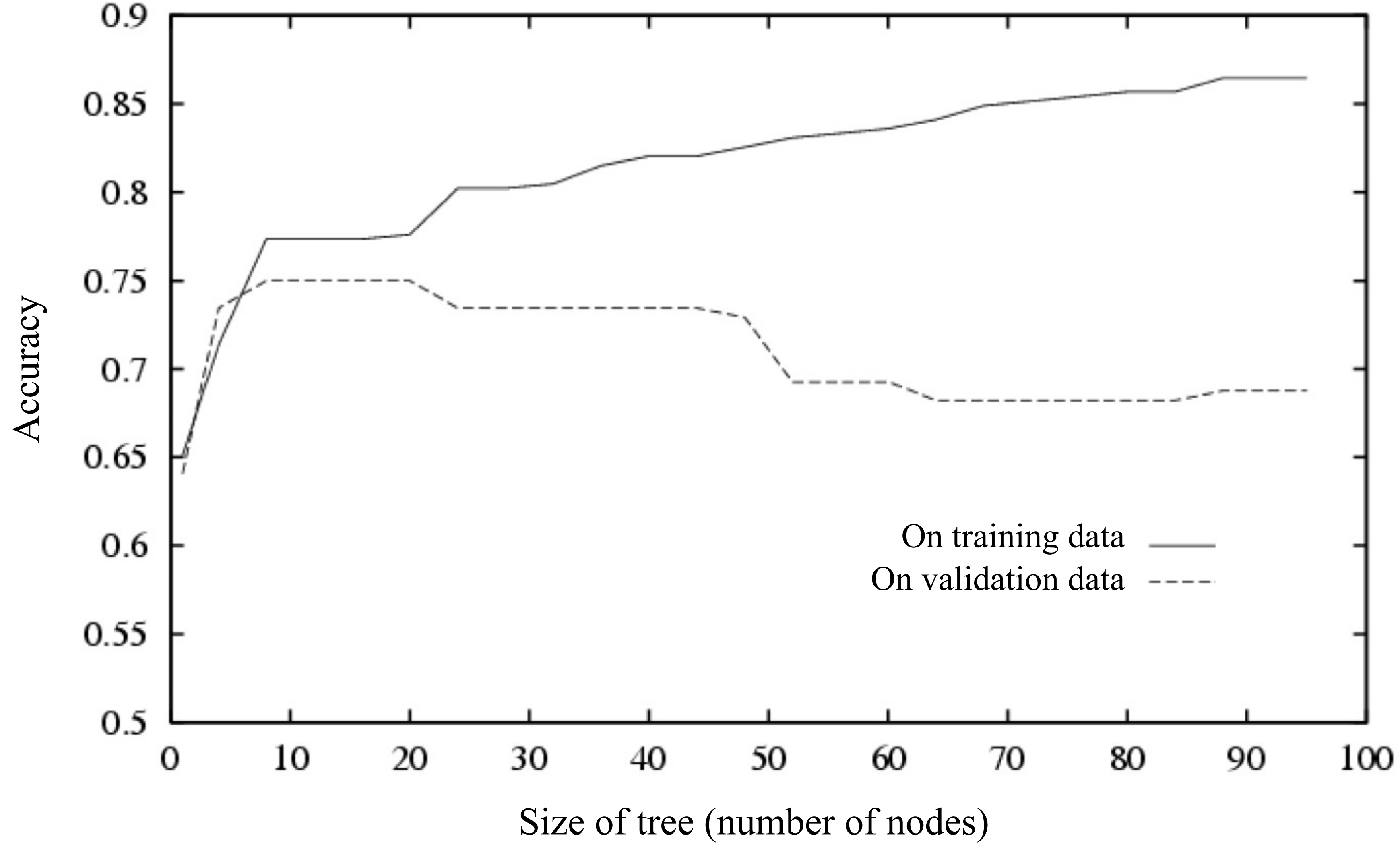


Figure from Tom Mitchell

How to Avoid Overfitting?

For Decision Trees…

1. Do not grow tree beyond some **maximum depth**
2. Do not split if splitting criterion (e.g. mutual information) is **below some threshold**
3. Stop growing when the split is **not statistically significant**
4. Grow the entire tree, then **prune** (e.g. Reduced Error Pruning)

Reduced Error Pruning

1. Split data in two: *training* dataset and *validation* dataset
2. Grow the full tree using the *training* dataset
3. Repeatedly prune the tree:
   - Evaluate each split using a *validation* dataset by comparing the validation error rate **with and without** that split
   - (Greedily) remove the split that most decreases the validation error rate
   - Stop if no split improves validation error, otherwise repeat

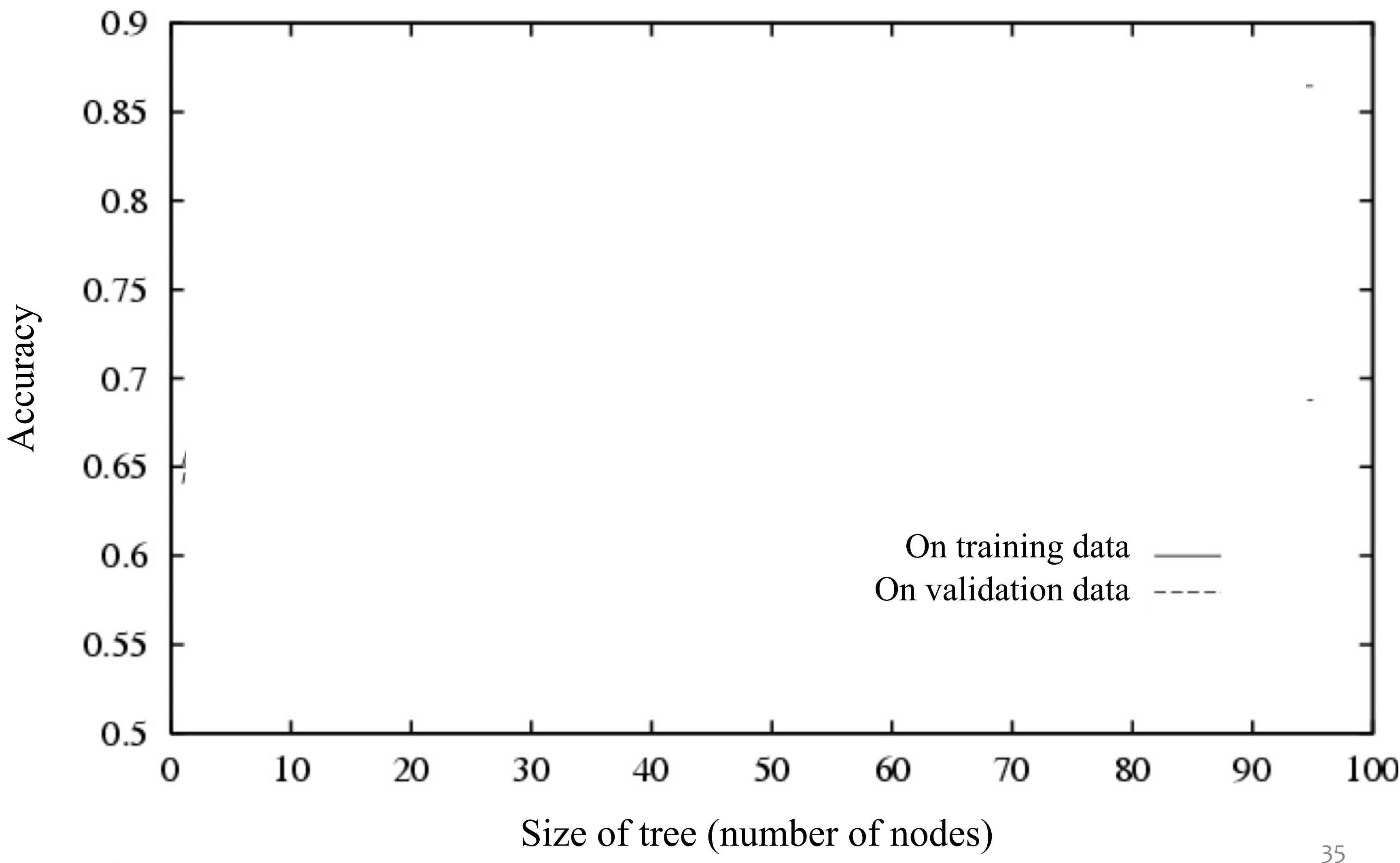


Figure from Tom Mitchell

Reduced Error Pruning

1. Split data in two: *training* dataset and *validation* dataset
2. Grow the full tree using the *training* dataset
3. Repeatedly prune the tree:
   - Evaluate each split using a *validation* dataset by comparing the validation error rate **with and without** that split
   - (Greedily) remove the split that most decreases the validation error rate
   - Stop if no split improves validation error, otherwise repeat

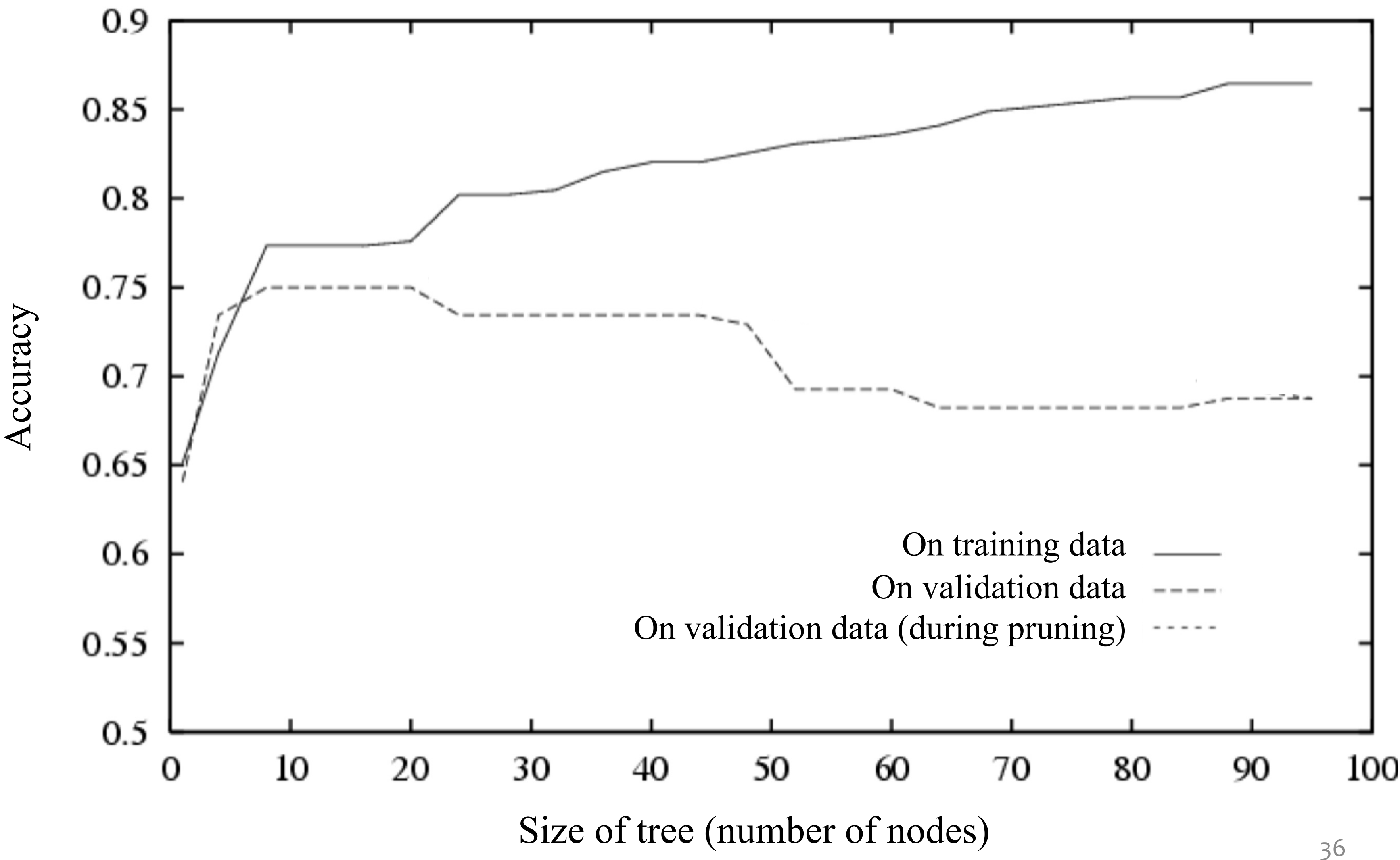
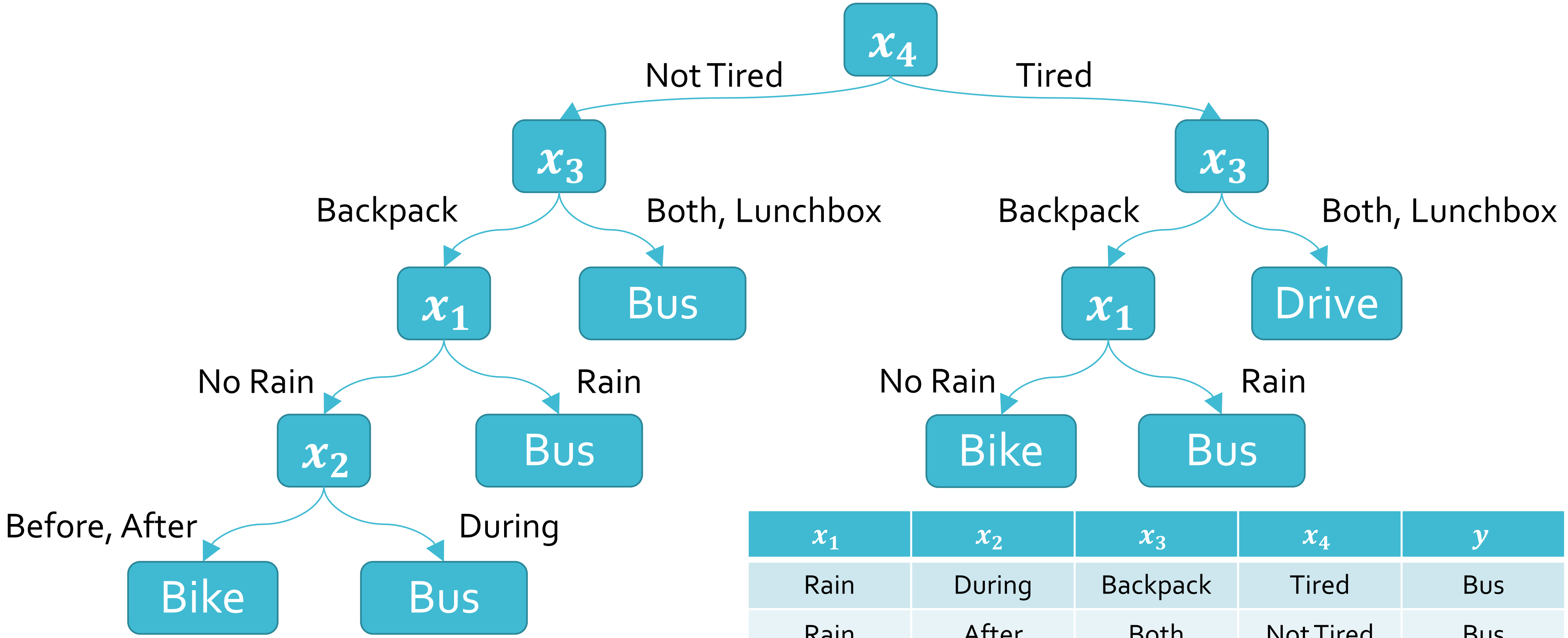


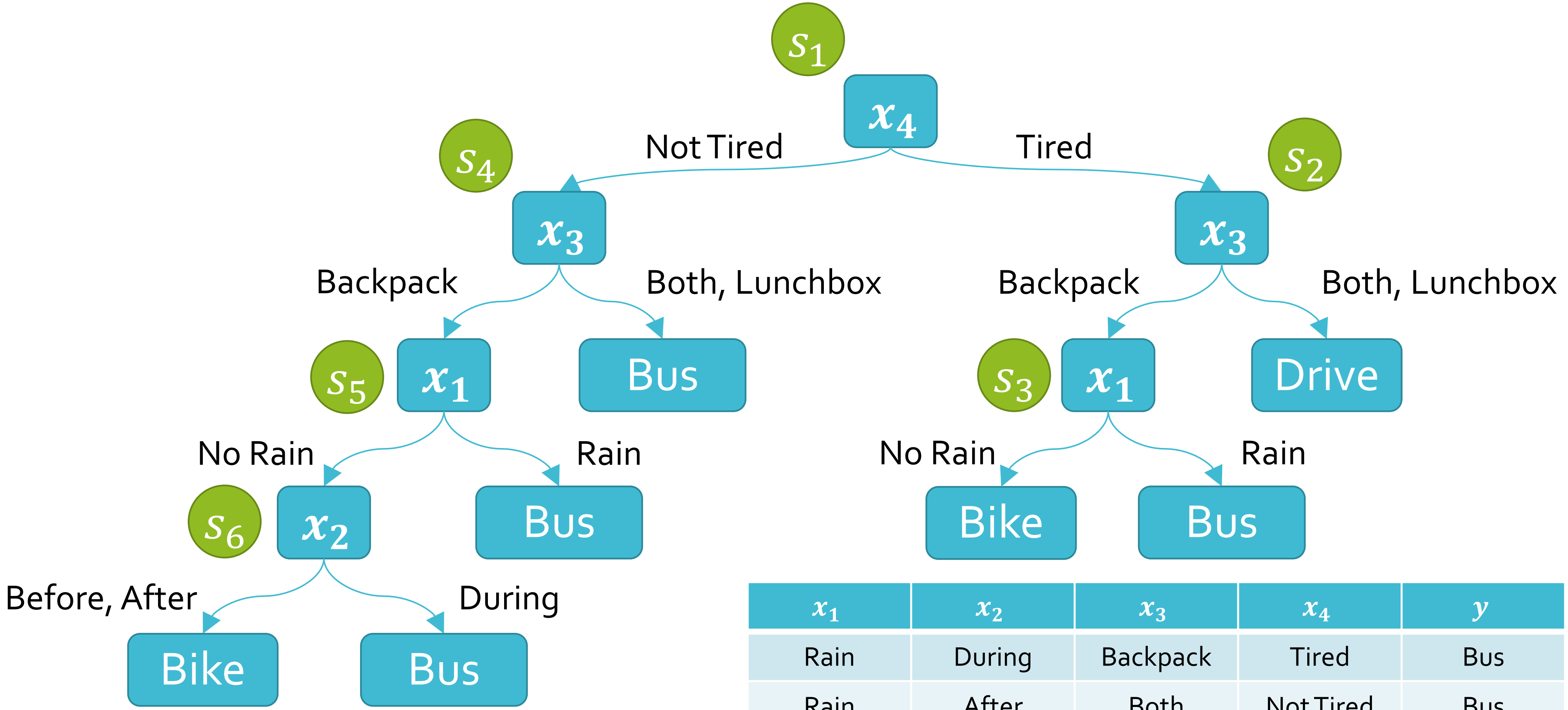
Figure from Tom Mitchell



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

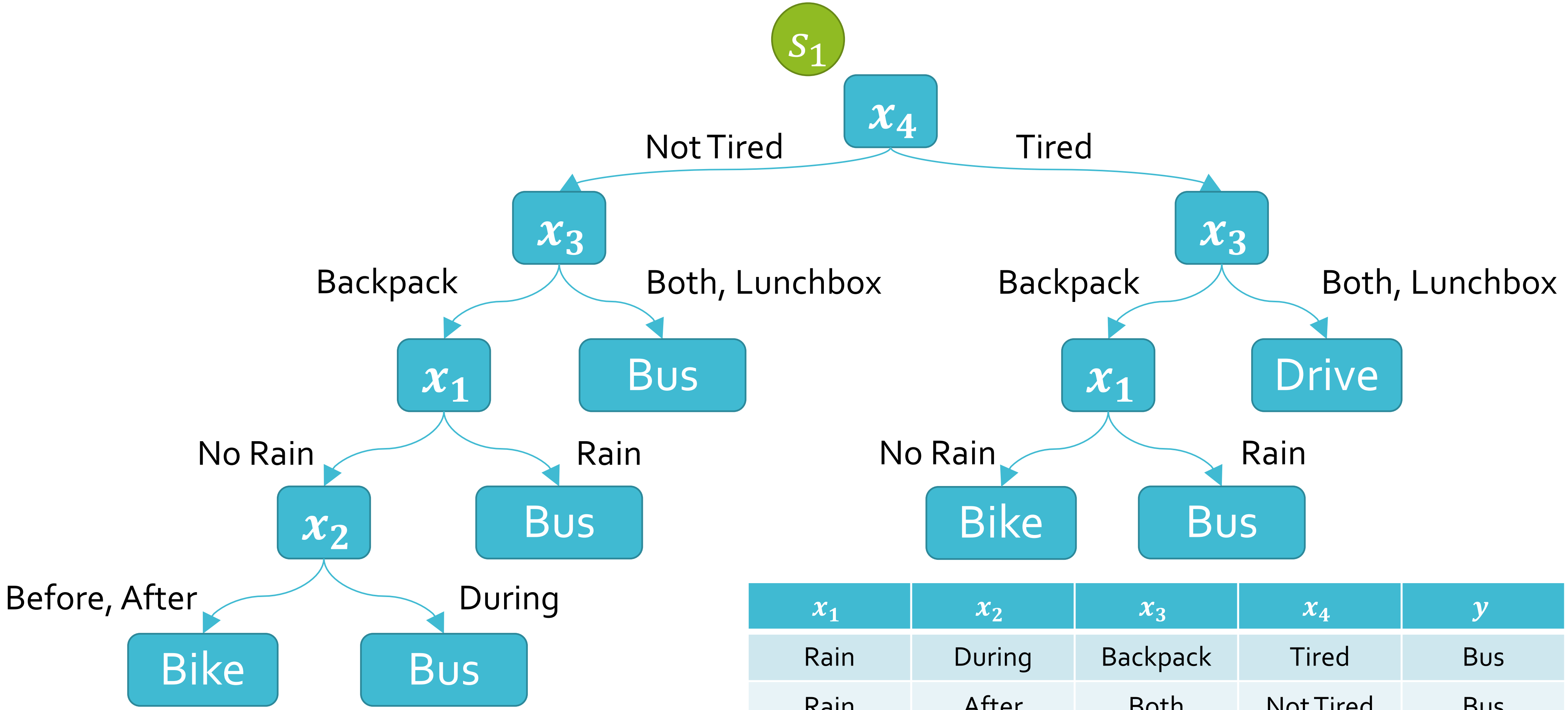
Slide from Henry Chai



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$err(h, \mathcal{D}_{val}) = 0.2$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$err(h - s_1, \mathcal{D}_{val})$

s_1

Bus

$$\mathcal{D}_{val} =$$

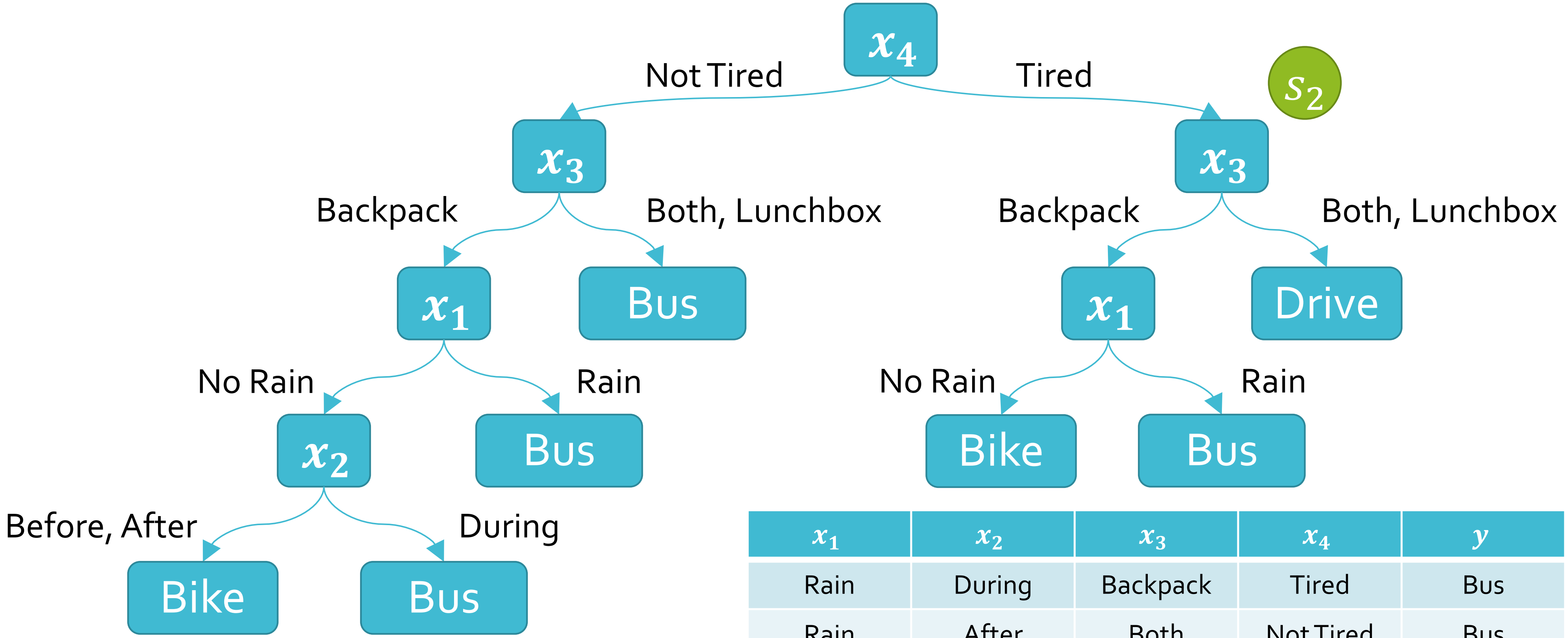
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

$$err(h - s_1, \mathcal{D}_{val})$$

$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

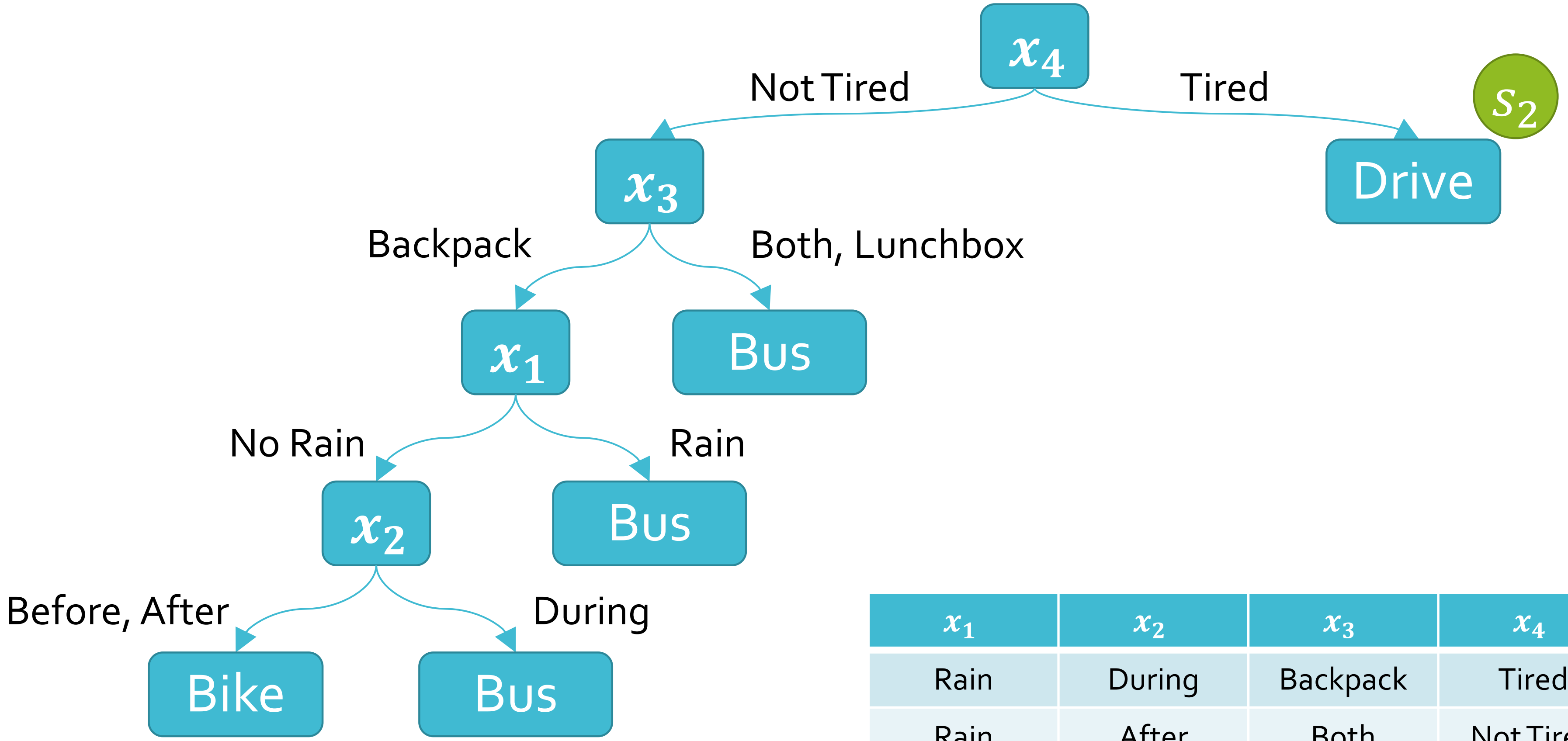
$$err(h - s_1, \mathcal{D}_{val}) = 0.4$$



$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

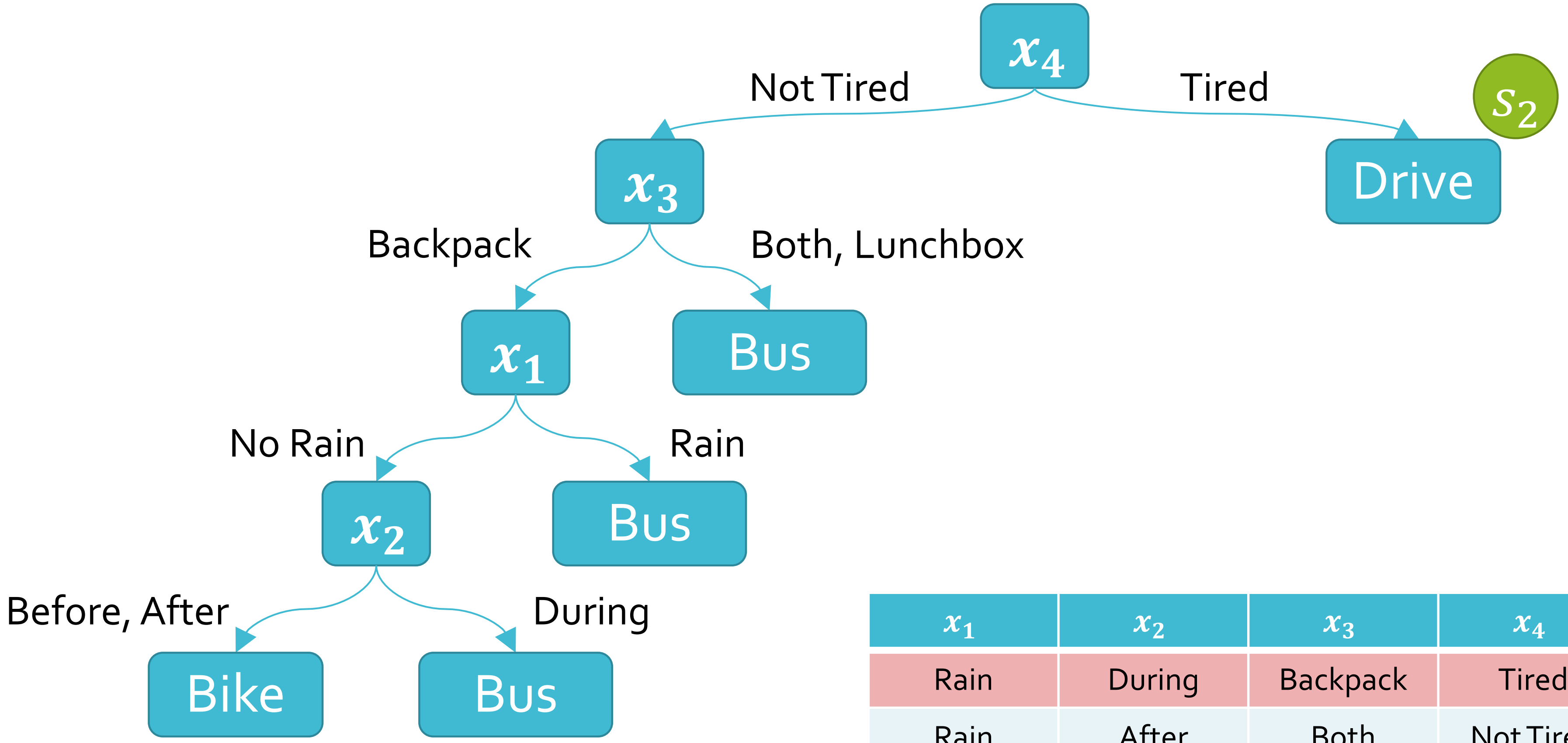
$$err(h - s_2, \mathcal{D}_{val})$$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

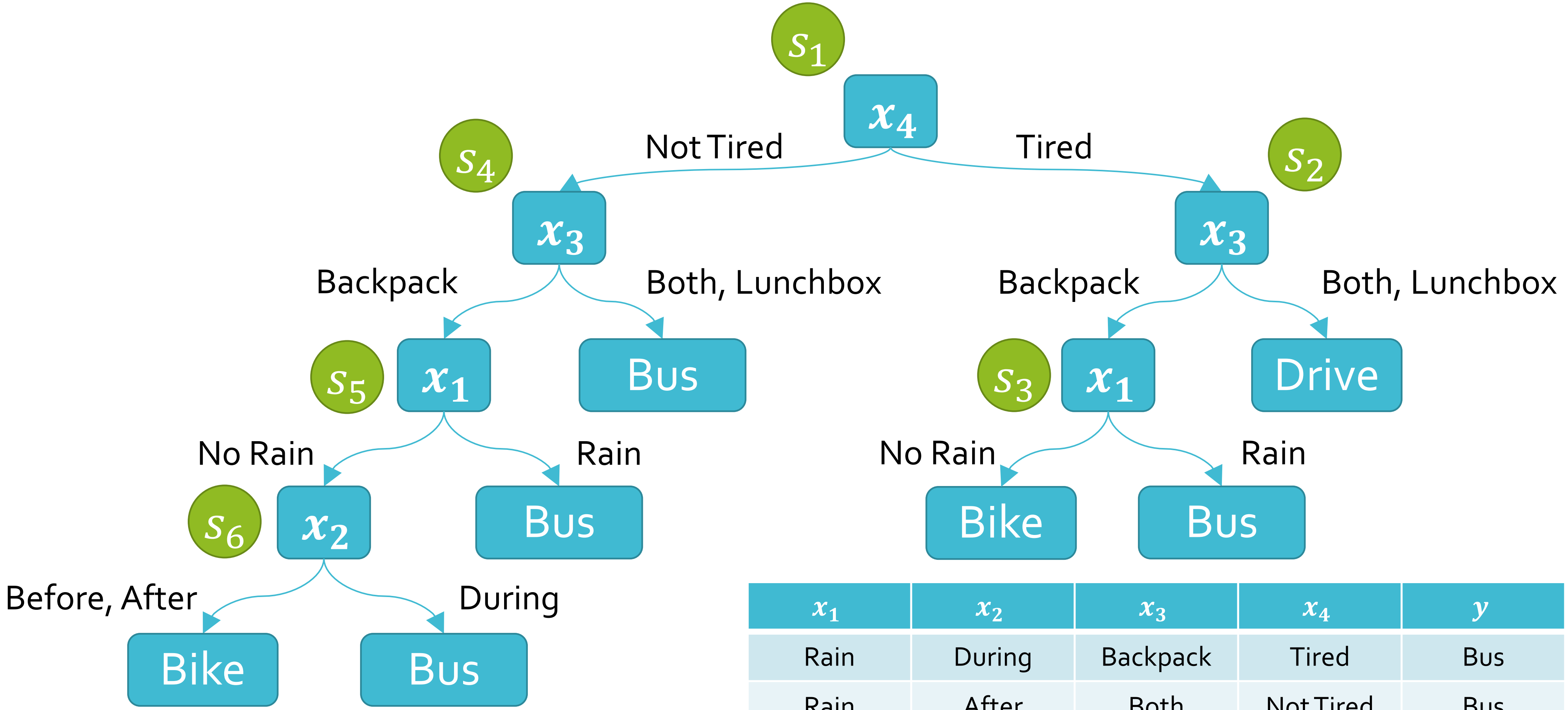
$err(h - s_2, \mathcal{D}_{val})$



$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

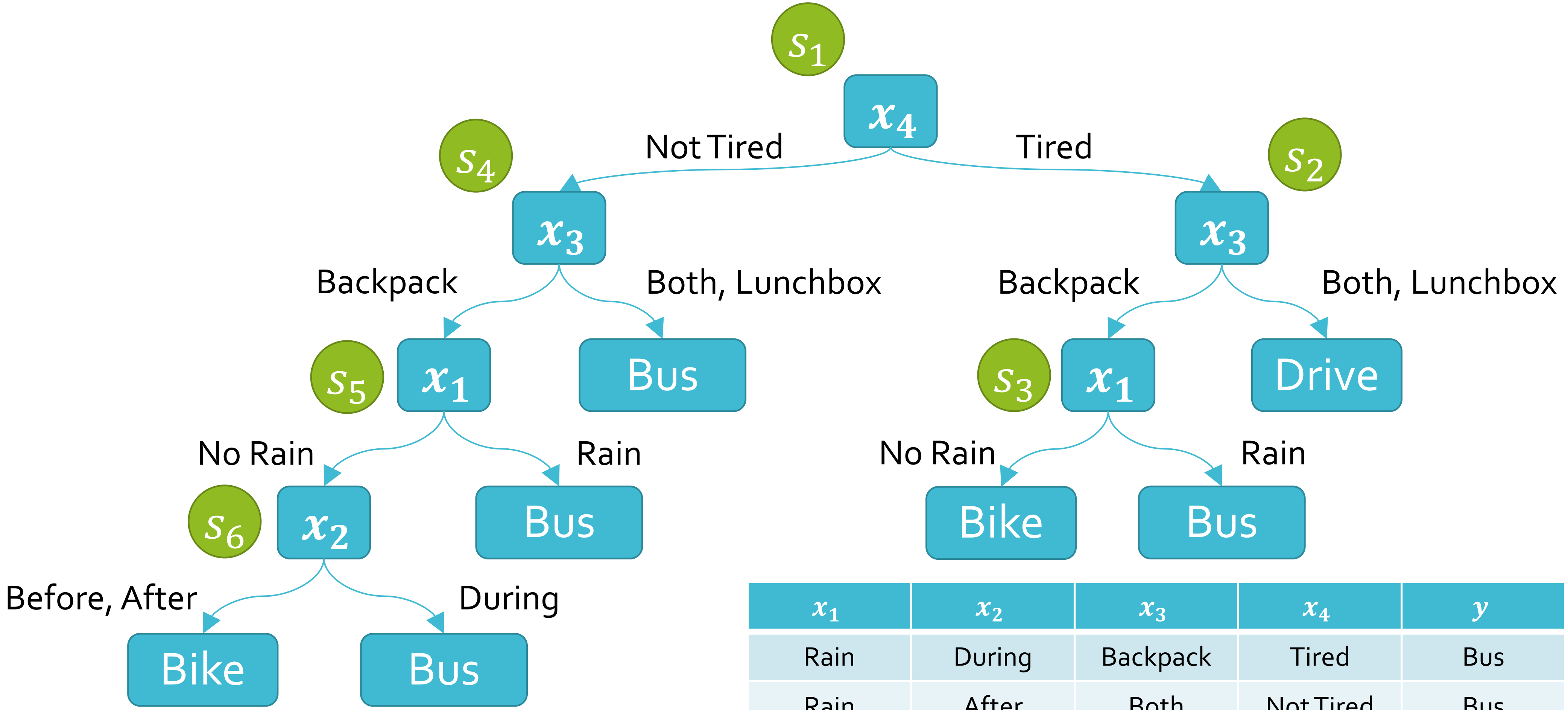
$$err(h - s_2, \mathcal{D}_{val}) = 0.4$$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

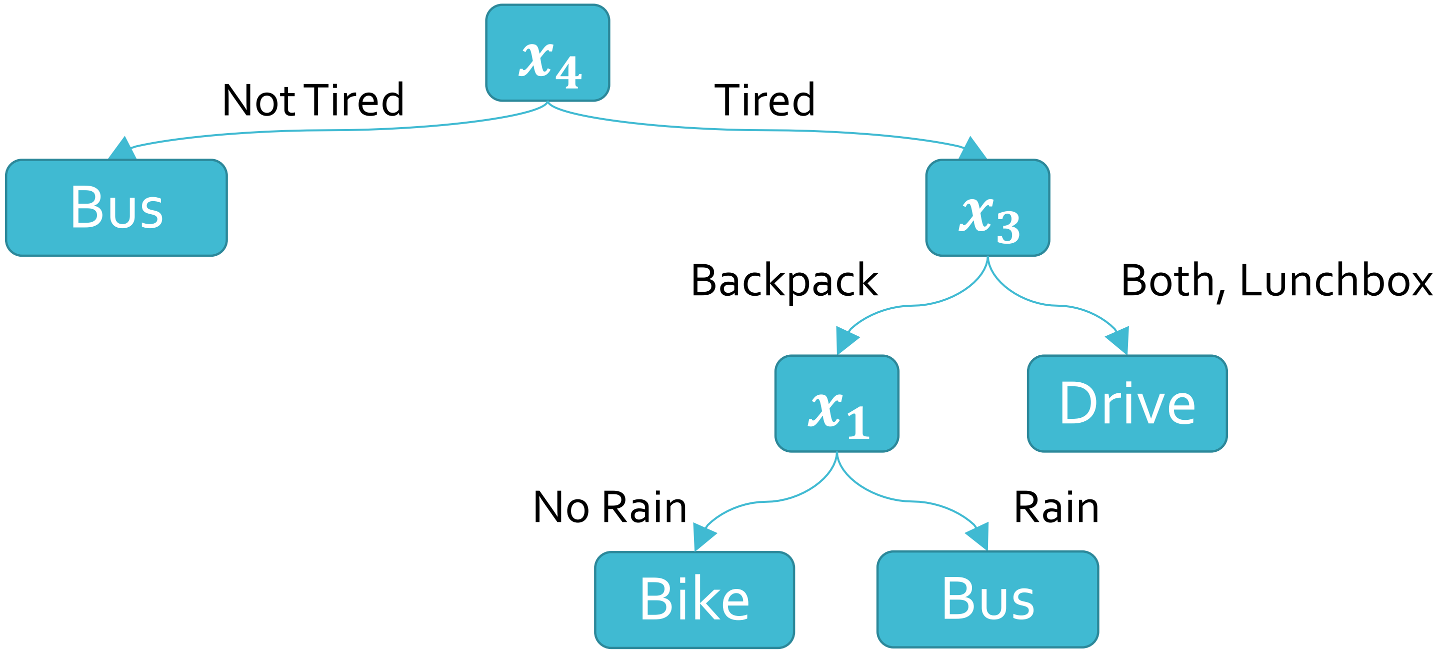
s	s_1	s_2	s_3	s_4	s_5	s_6
$err(h - s, \mathcal{D}_{val})$	0.4	0.4	0.4	0	0	0.2



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

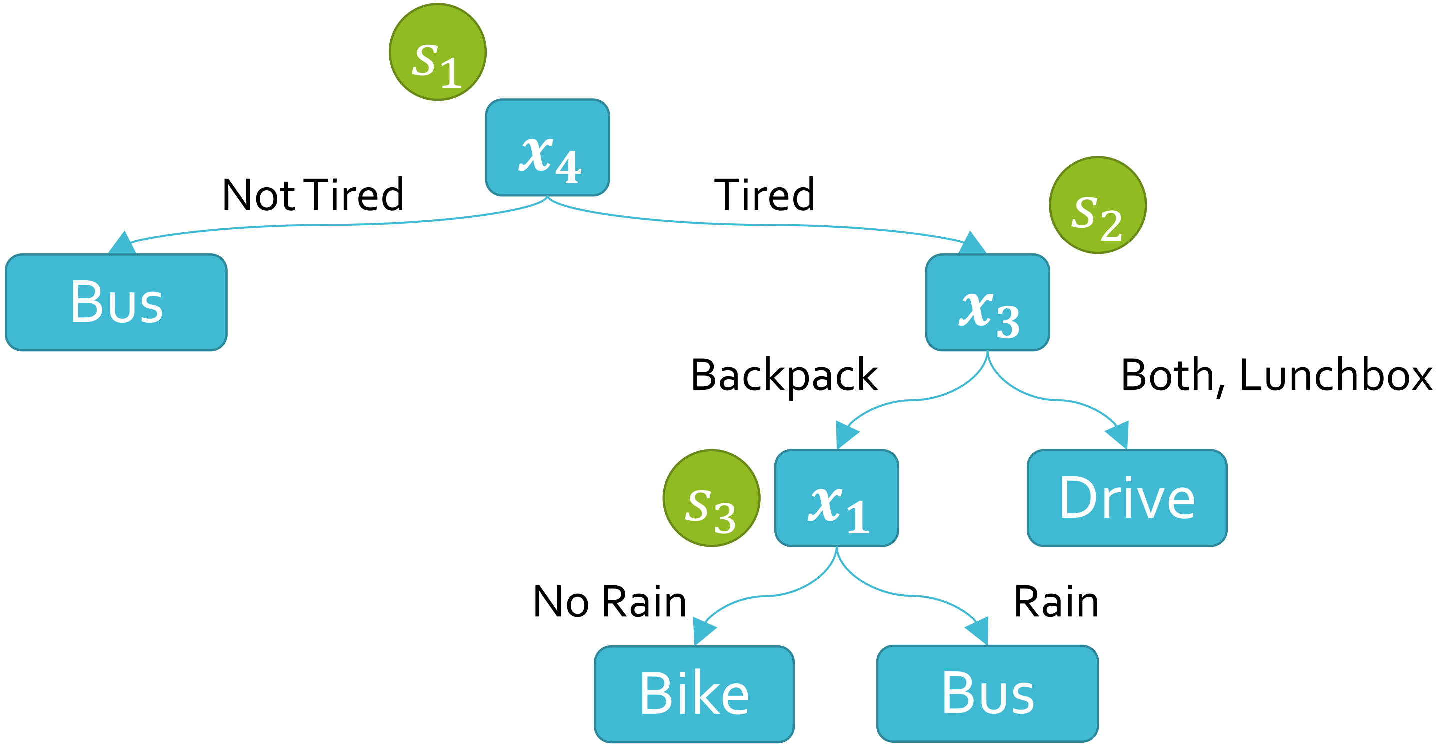
s	s_1	s_2	s_3	s_4	s_5	s_6
$err(h - s, \mathcal{D}_{val})$	0.4	0.4	0.4	0	0	0.2



$$\mathcal{D}_{val} =$$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

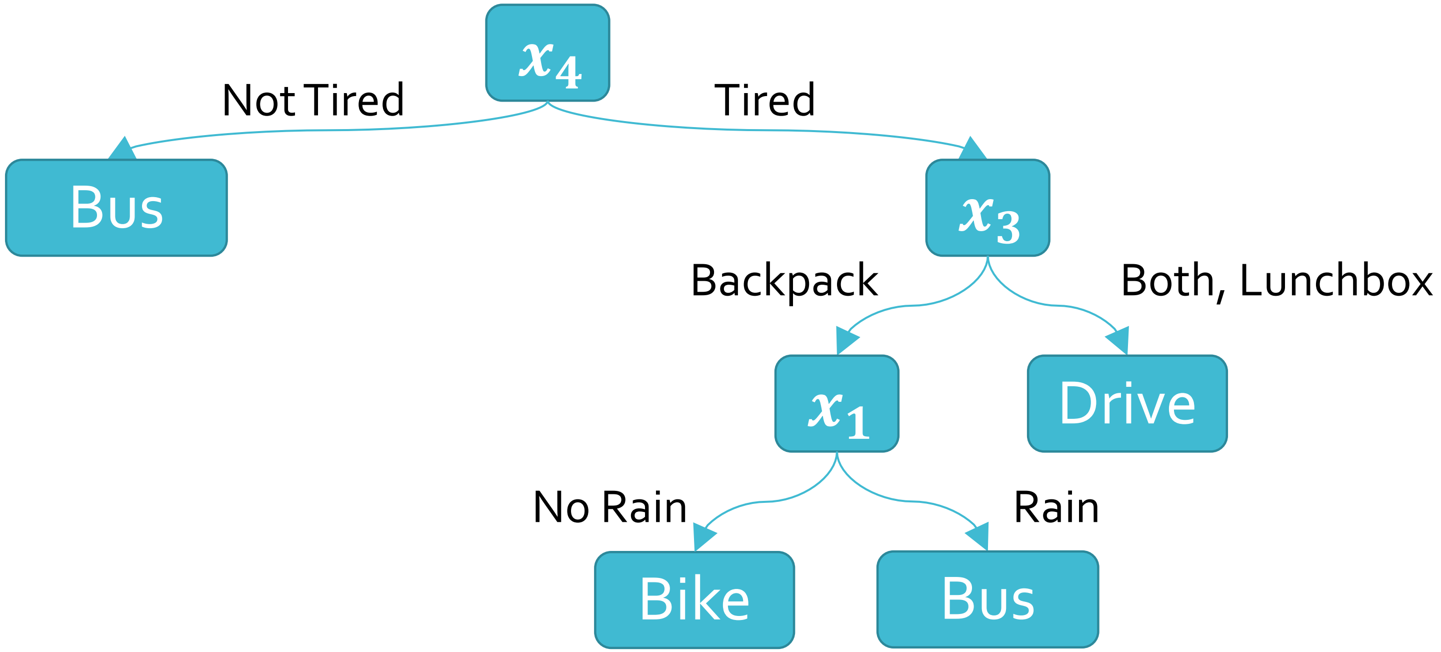
$$err(h, \mathcal{D}_{val}) = 0$$



$\mathcal{D}_{val} =$

x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive

s	s_1	s_2	s_3
$err(h - s, \mathcal{D}_{val})$	0.4	0.2	0.2

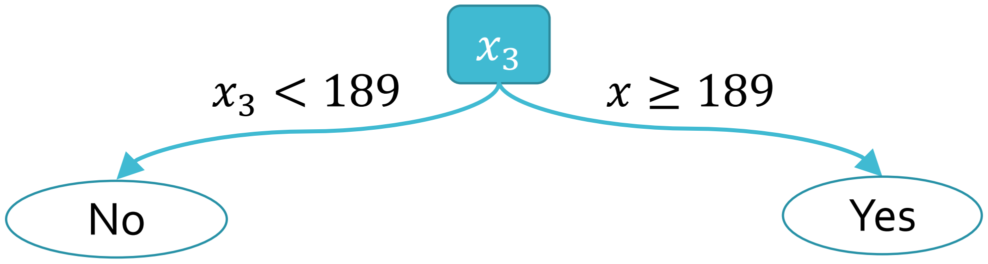

x_4
Not Tired
Tired
Bus
x_3
Backpack
Both, Lunchbox
x_1
Drive
No Rain
Rain
Bike
Bus

Real-Valued Features: Example

x_3 Cholesterol	y Heart Disease?
174	No
155	No
163	Yes
233	Yes
197	Yes
181	No
244	Yes
245	No
178	No
221	Yes

x_3 Cholesterol	y Heart Disease?
155	No
163	Yes
174	No
178	No
181	No
197	Yes
221	Yes
233	Yes
244	Yes
245	No

$x_3 < 189$



Real-Valued Features: Example

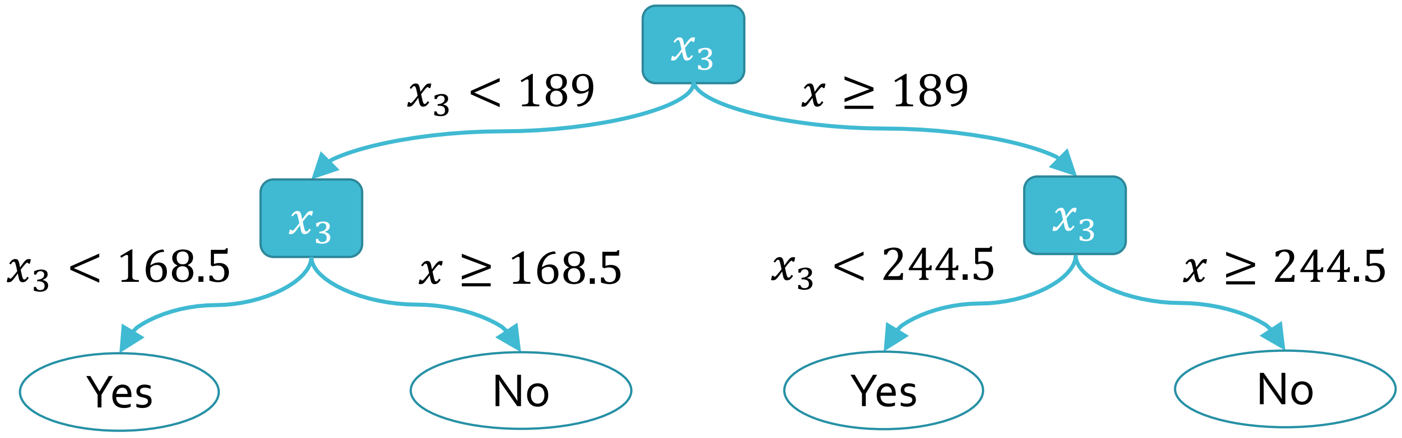
x_3 Cholesterol	y Heart Disease?
174	No
155	No
163	Yes
233	Yes
197	Yes
181	No
244	Yes
245	No
178	No
221	Yes

x_3 Cholesterol	y Heart Disease?
155	No
163	Yes
174	No
178	No
181	No
197	Yes
221	Yes
233	Yes
244	Yes
245	No

$x_3 < 168.5$

$x_3 < 189$

$x_3 < 244.5$



Decision Trees (DTs) in the Wild

- DTs are one of the most popular classification methods for practical applications
  - Reason #1: The learned representation is **easy to explain** a non-ML person
  - Reason #2: They are **efficient** in both computation and memory
- DTs can be applied to a wide variety of problems including **classification, regression, density estimation**, etc.
- **Applications of DTs** include…
  - medicine, molecular biology, text classification, manufacturing, astronomy, agriculture, and many others
- **Decision Forests** learn many DTs from random subsets of features; the result is a very powerful example of an **ensemble method** (discussed later in the course)

DT Learning Objectives

You should be able to…

1. Implement Decision Tree training and prediction
2. Use effective splitting criteria for Decision Trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in Decision Tree learning

REAL VALUED ATTRIBUTES


sepal
petal

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

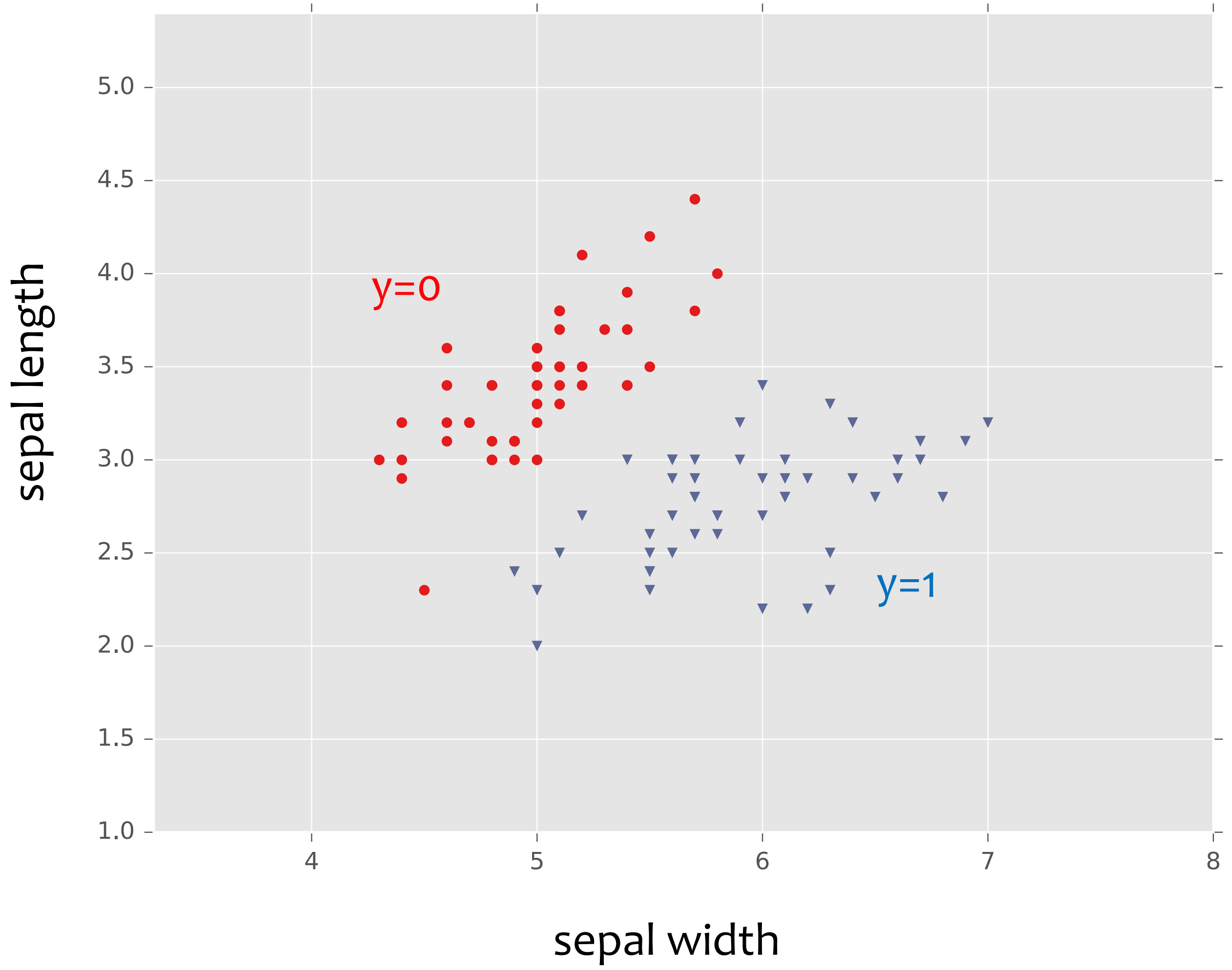
Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Deleted two of the four features, so that input space is 2D

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset



Classification & Real-Valued Features

Def: Classification

$$D = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$$

$\forall i, \ \mathbf{x}^{(i)} \in \mathbb{R}^M$ (features/instance)

$\forall i, \ y^{(i)} \in \{1, 2, \ldots, L\}$ (label/class)

M = # of features (dimensionality of $\mathbf{x}$)

N = # of training examples = $|D|$

Def: Binary Classification

Classification where $|\mathcal{Y}| = 2$

$\forall i, \ y^{(i)} \in \{+, -\}$

$\in$ {red, blue}

$\in$ {cat, dog}

Classification & Real-Valued Features

Def: Hypothesis (aka. Decision Rule) for Binary Classification

$$h : \mathbb{R}^M \rightarrow \{+, -\}$$

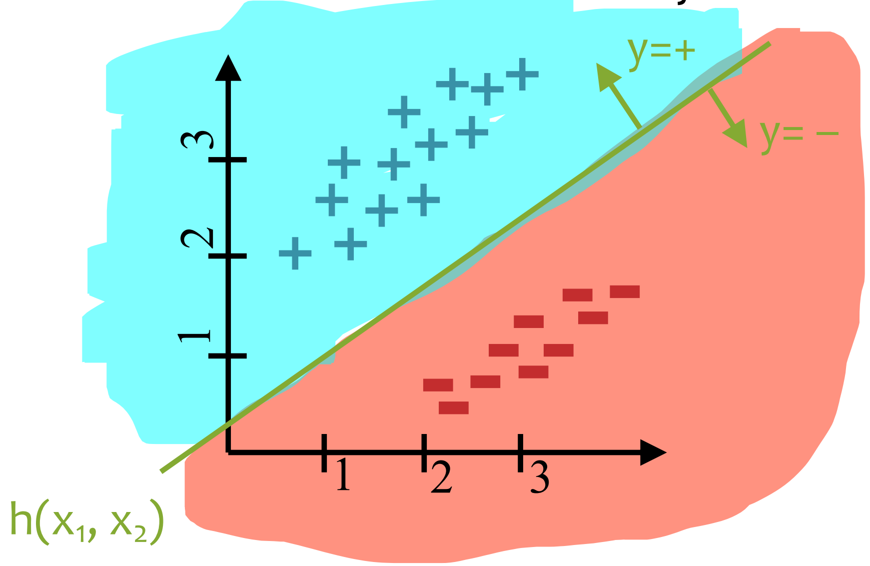
Train time: Learn h

Test time: Given $\hat{\mathbf{x}}$, predict $\hat{y} = h(\hat{\mathbf{x}})$

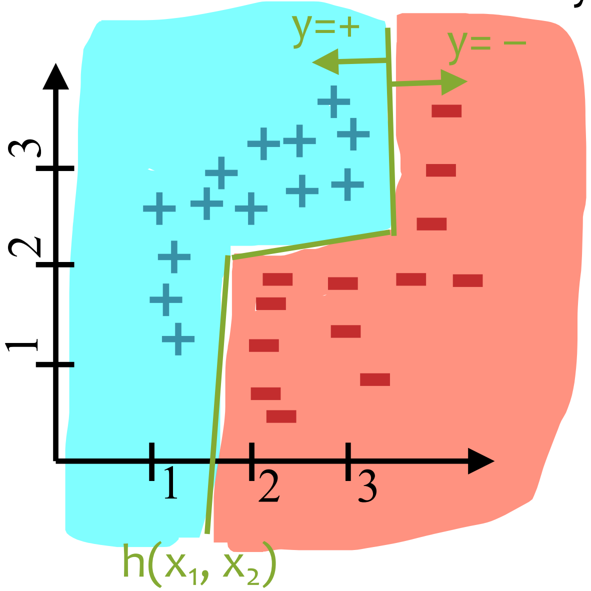
Evaluate h

Ex: Decision Boundaries (2D Binary Classification)

linear decision boundary



nonlinear decision boundary



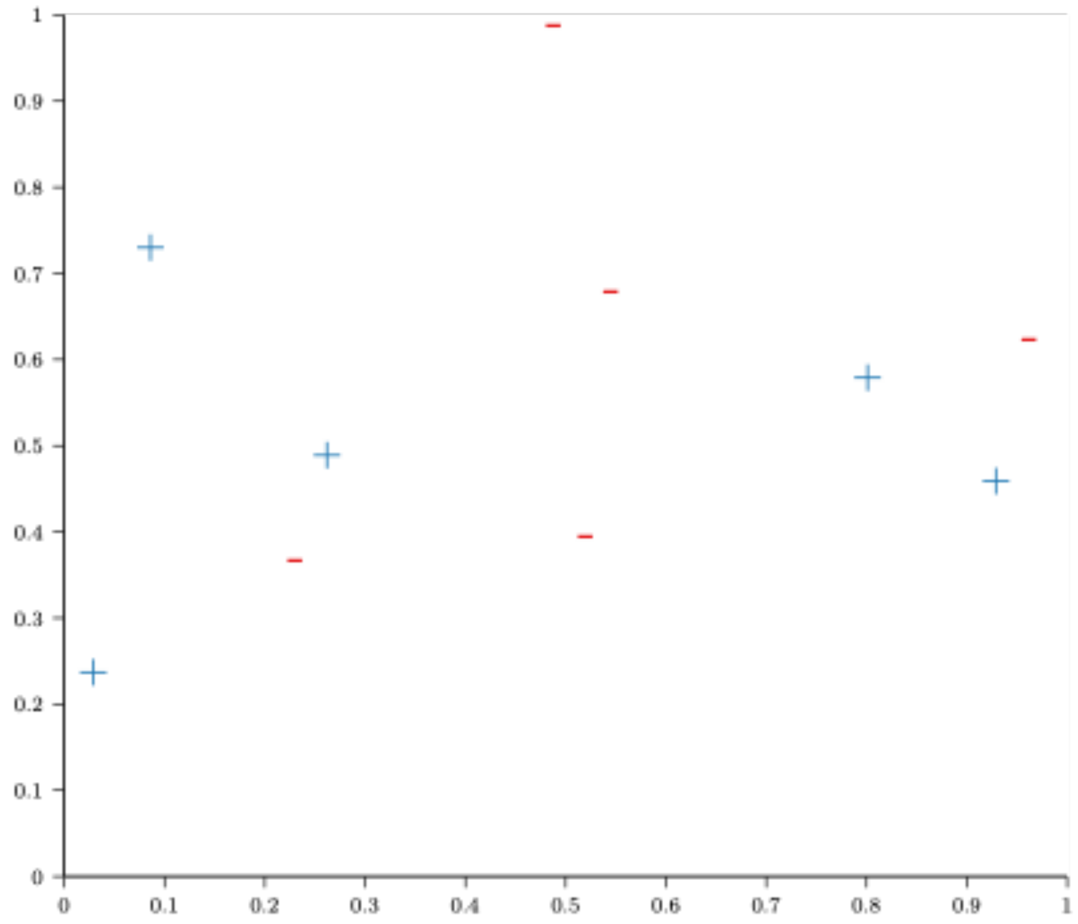
K-NEAREST NEIGHBORS

Nearest Neighbor: Algorithm

def train($\mathcal{D}$):
    Store $\mathcal{D}$

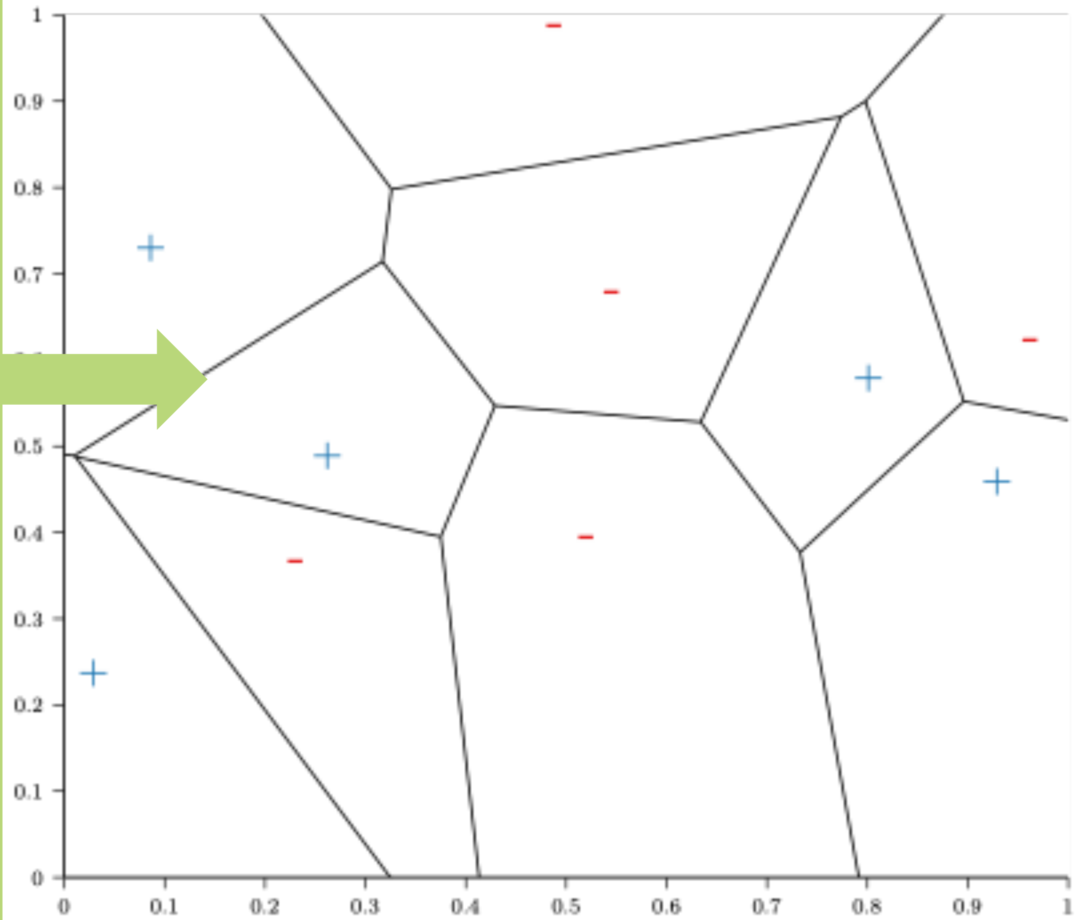
def h($\boldsymbol{x}'$):
    Let $\boldsymbol{x}^{(i)}$ = the point in $\mathcal{D}$ that is nearest to $\boldsymbol{x}'$
    return $y^{(i)}$

Nearest Neighbor: Example

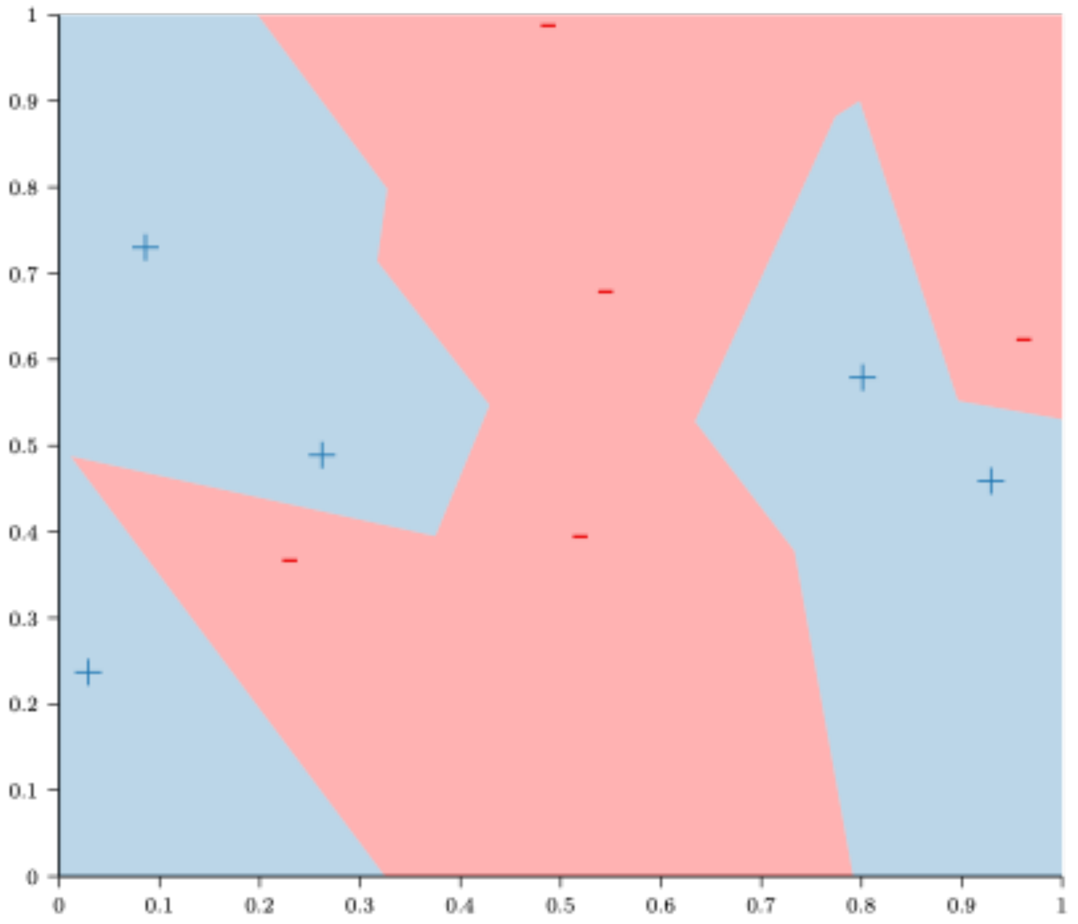


Nearest Neighbor: Example

- This is a **Voronoi diagram**
- Each **cell** contain one of our training examples
- **All points within a cell** are **closer** to that training example, than to any other training example
- **Points on the Voronoi line segments** are **equidistant** to one or more training examples



Nearest Neighbor: Example



The Nearest Neighbor Model

- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***

k-Nearest Neighbors: Algorithm

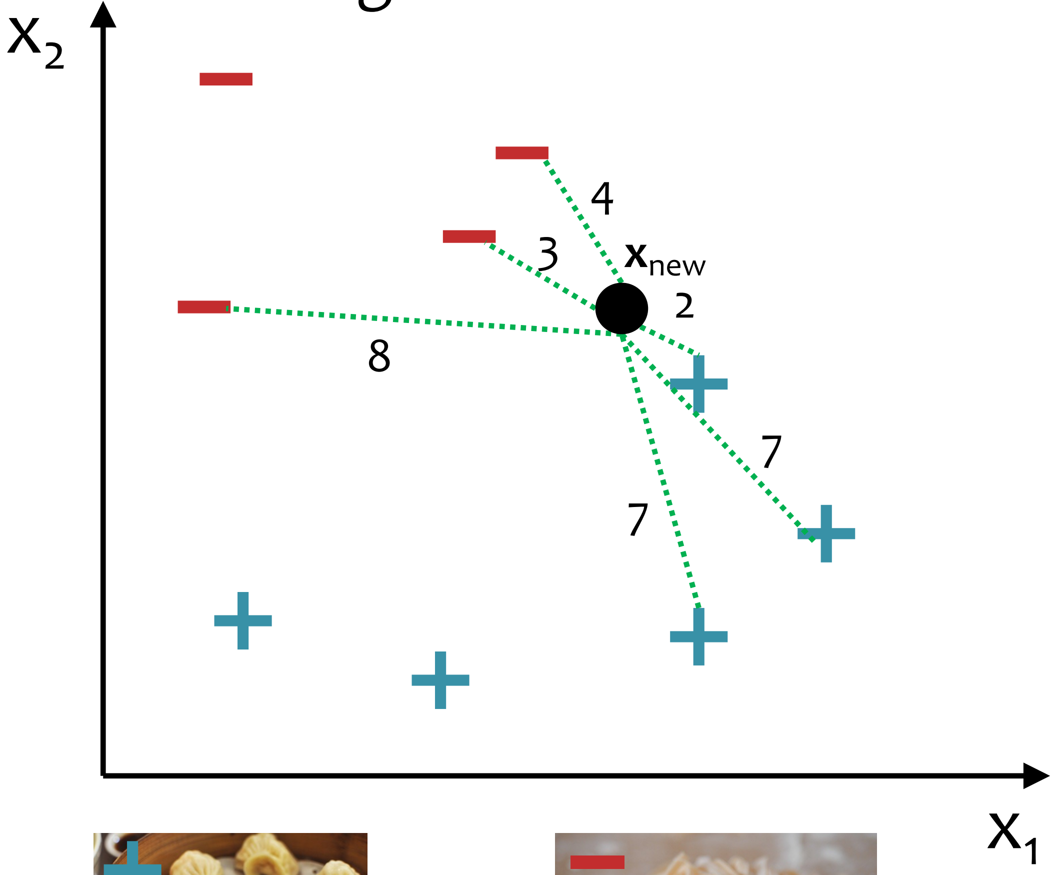
```
def set_hyperparameters(k, d):
        Store k
        Store d(·, ·)

def train(𝒟):
        Store 𝒟

def h(x′):
        Let S = the set of k points in 𝒟 nearest to x′
                according to distance function
                d(u, v)
        Let v = majority_vote(S)
        return v
```

k-Nearest Neighbors

Suppose we have the training dataset below.



How should we label the new point?

It depends on k:

if k=1, $h(\mathbf{x}_{new}) = +1$

if k=3, $h(\mathbf{x}_{new}) = -1$

if k=5, $h(\mathbf{x}_{new}) = +1$

KNN: Remarks

Distance Functions:

- KNN requires a **distance function**

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

- But there are other choices (e.g. **Manhattan distance**)

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{m=1}^{M} |u_m - v_m|$$

KNN: Computational Efficiency

- Suppose we have N training examples and each one has M features
- Computational complexity when k=1:

Task	Naive	k-d Tree
Train	O(1)	~ O(M N log N)
Predict (one test example)	O(MN)	~ O(2^M log N) on average

Problem: Very fast for small M, but very slow for large M

In practice: use stochastic approximations (very fast, and empirically often as good)

KNN: Theoretical Guarantees

Cover & Hart (1967)

Let h(x) be a Nearest Neighbor (k=1) binary classifier. As the number of training examples N goes to infinity…

$$\text{error}_{\text{true}}(h) < 2 \text{ x Bayes Error Rate}$$

"In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

very informally, Bayes Error Rate can be thought of as: *'the best you could possibly do'*