

10-701: Introduction to Machine Learning Lecture 3 –KNNs

Henry Chai

1/24/24

Front Matter

- Announcements:
 - HW1 released 1/24 (today!), due 2/2 at 11:59 PM
 - Recitations will be held on Fridays, at the same time and place as lecture
 - HW1 recitation this Friday (1/26)
 - Office hours will start 1/24 (today!)
- Recommended Readings:
 - Mitchell, [Section 8.1 – 8.2: \$k\$ -Nearest Neighbor Learning](#)
 - Daumé III, [Chapter 3: Geometry and Nearest Neighbors](#)

Recall: Decision Tree Prediction - Pseudocode

```
def predict( $x'$ ):
```

```
- walk from root node to a leaf node
```

```
while(true):
```

```
    if current node is internal (non-leaf):
```

```
        check the associated attribute,  $x_d$ 
```

```
        go down branch according to  $x'_d$ 
```

```
    if current node is a leaf node:
```

```
        return label stored at that leaf
```

Recall: Decision Tree Learning - Pseudocode

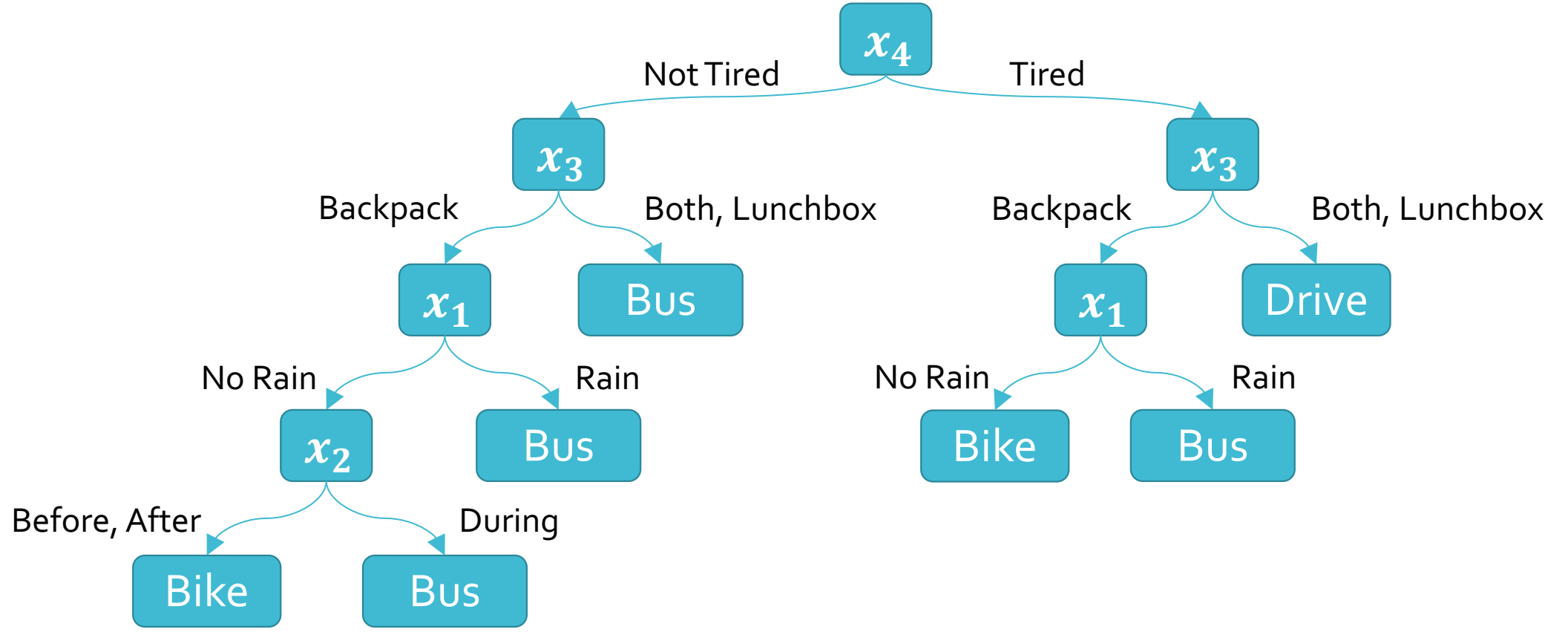
```
def train( $\mathcal{D}$ ):  
    store root = tree_recurse( $\mathcal{D}$ )  
def tree_recurse( $\mathcal{D}'$ ):  
    q = new node()  
    base case - if (SOME CONDITION):  
    recursion - else:  
        find best attribute to split on,  $x_d$   
        q.split =  $x_d$   
        for  $v$  in  $V(x_d)$ , all possible values of  $x_d$ :  
             $\mathcal{D}_v = \{(x^{(n)}, y^{(n)}) \in \mathcal{D}' \mid x_d^{(n)} = v\}$   
            q.children( $v$ ) = tree_recurse( $\mathcal{D}_v$ )  
    return q
```

Recall: Decision Tree Learning - Pseudocode

```
def train( $\mathcal{D}$ ):  
    store root = tree_recurse( $\mathcal{D}$ )  
def tree_recurse( $\mathcal{D}'$ ):  
    q = new node()  
    base case - if ( $\mathcal{D}'$  is empty OR  
        all labels in  $\mathcal{D}'$  are the same OR  
        all features in  $\mathcal{D}'$  are identical OR  
        some other stopping criterion):  
        q.label = majority_vote( $\mathcal{D}'$ )  
  
    recursion - else:  
        return q
```

How is Henry Getting to Work?

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



Decision Trees: Inductive Bias

- 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 i.e., decision tree learning with mutual information maximization as the splitting criterion?
 - Try to find the shortest tree that achieves zero training error with high mutual information features at the top
- Occam's razor: try to find the "simplest" (e.g., smallest decision tree) classifier that explains the training dataset

Decision Trees: Pros & Cons

- Pros
 - Interpretable
 - Efficient (computational cost and storage)
 - Can be used for classification and regression tasks
 - Compatible with categorical and real-valued features
- Cons

Real-Valued Features: Example - x = Outside Temperature (°F)

x	y
74	Drive
55	Metro
63	Bike
33	Drive
80	Drive
81	Drive
44	Metro
45	Metro
78	Drive
51	Metro



x	y
33	Drive
44	Metro
45	Metro
51	Metro
55	Metro
63	Bike
74	Drive
78	Drive
80	Drive
81	Drive

← $x < 38.5$

Real-Valued Features: Example - x = Outside Temperature (°F)

x	y
74	Drive
55	Metro
63	Bike
33	Drive
80	Drive
81	Drive
44	Metro
45	Metro
78	Drive
51	Metro



x	y
33	Drive
44	Metro
45	Metro
51	Metro
55	Metro
63	Bike
74	Drive
78	Drive
80	Drive
81	Drive

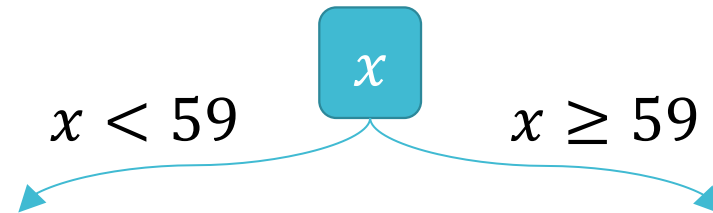
← $x < 44.5$

Real-Valued Features: Example - x = Outside Temperature (°F)

x	y
74	Drive
55	Metro
63	Bike
33	Drive
80	Drive
81	Drive
44	Metro
45	Metro
78	Drive
51	Metro



x	y
33	Drive
44	Metro
45	Metro
51	Metro
55	Metro
63	Bike
74	Drive
78	Drive
80	Drive
81	Drive

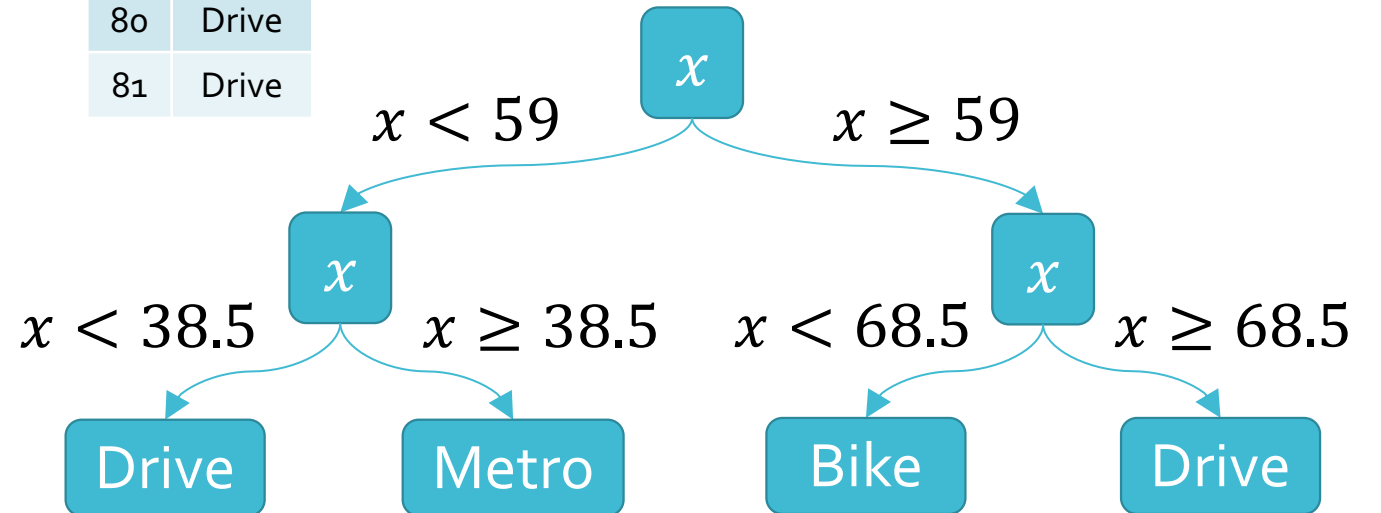


Real-Valued Features: Example - x = Outside Temperature (°F)

x	y
74	Drive
55	Metro
63	Bike
33	Drive
80	Drive
81	Drive
44	Metro
45	Metro
78	Drive
51	Metro



x	y
33	Drive
44	Metro
45	Metro
51	Metro
55	Metro
63	Bike
74	Drive
78	Drive
80	Drive
81	Drive



Decision Trees: Pros & Cons

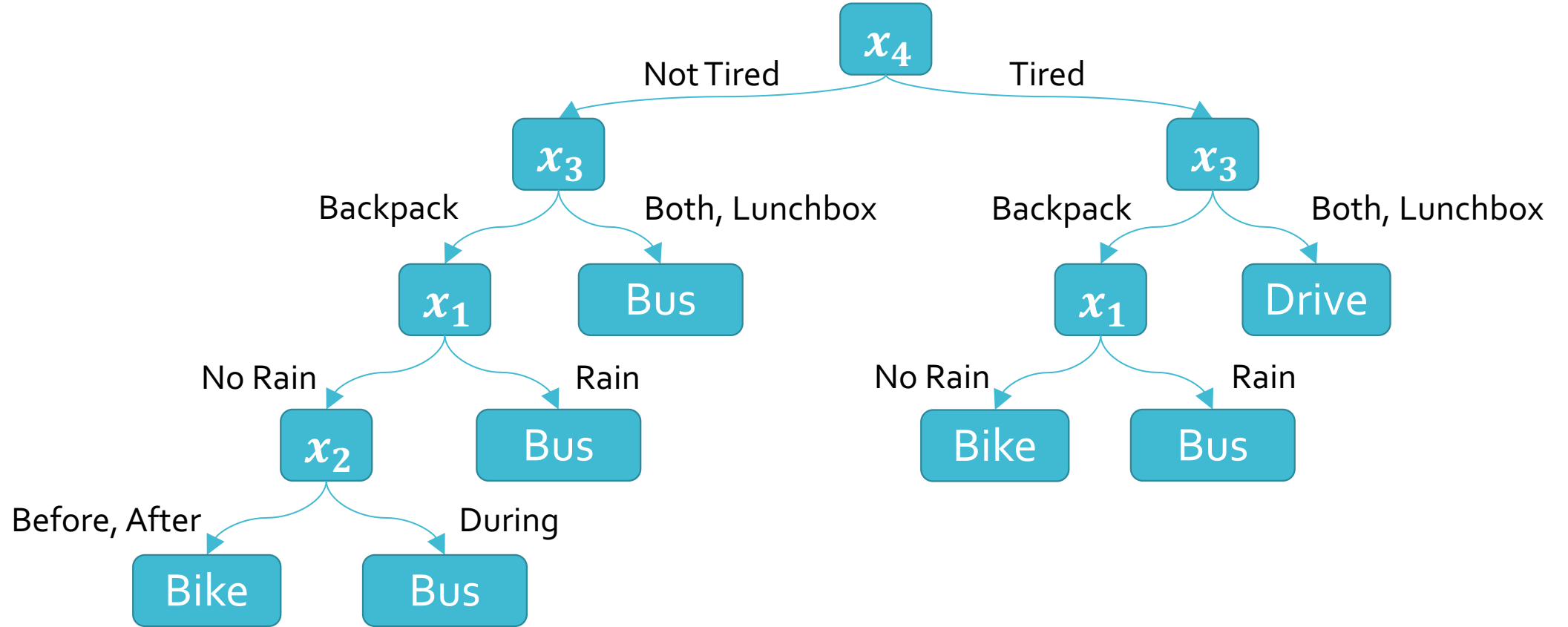
- Pros
 - Interpretable
 - Efficient (computational cost and storage)
 - Can be used for classification and regression tasks
 - Compatible with categorical and real-valued features
- Cons
 - Learned greedily: each split only considers the immediate impact on the splitting criterion
 - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
 - Liable to overfit!

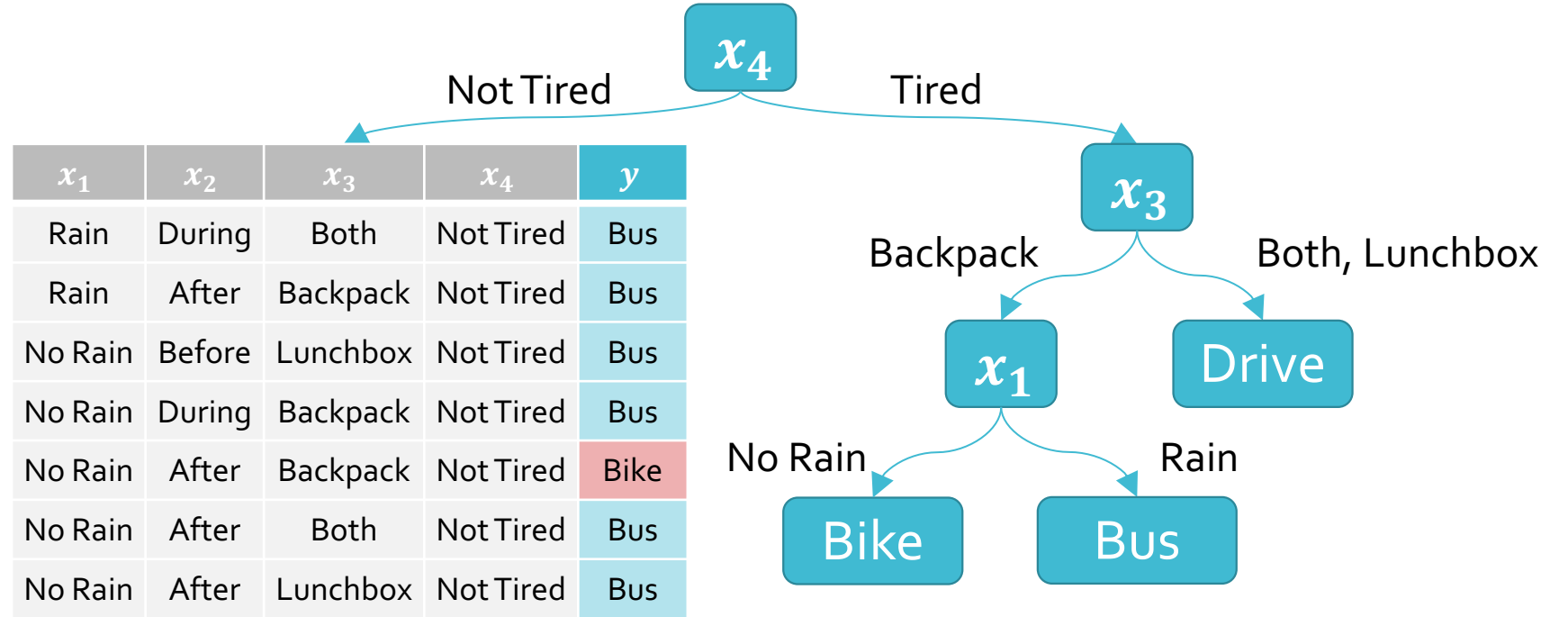
Overfitting

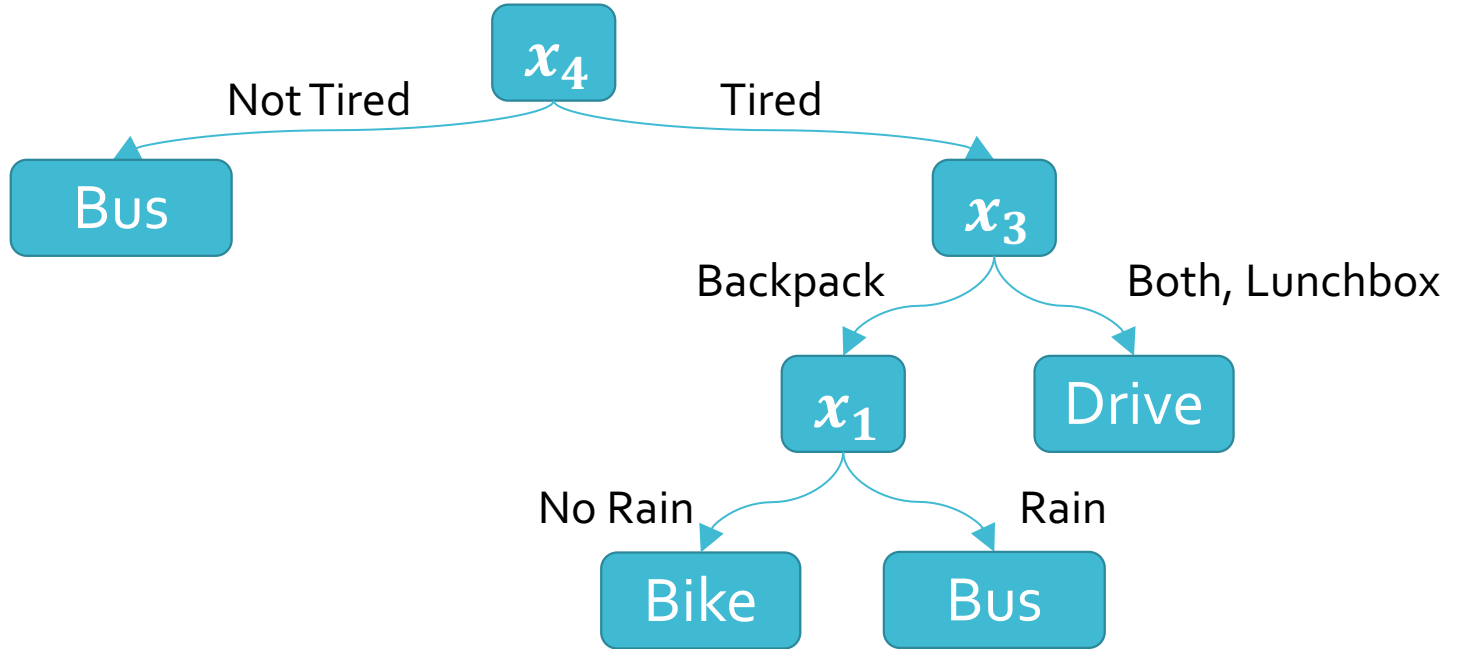
- Overfitting occurs when the classifier (or model)...
 - is too complex
 - fits noise or “outliers” in the training dataset as opposed to the actual pattern of interest
 - doesn’t have enough inductive bias pushing it to generalize (e.g., memorizer)
- Underfitting occurs when the classifier (or model)...
 - is too simple
 - can’t capture the actual pattern of interest in the training dataset
 - has too much inductive bias (e.g., majority vote)

Different Kinds of Error

- Training error rate = $err(h, \mathcal{D}_{train})$
- Test error rate = $err(h, \mathcal{D}_{test})$
- True error rate = $err(h)$
 - = the error rate of h on all possible examples
 - In machine learning, this is the quantity that we care about but, in most cases, it is unknowable.
- Overfitting occurs when $err(h) > err(h, \mathcal{D}_{train})$
 - $err(h) - err(h, \mathcal{D}_{train})$ can be thought of as a measure of overfitting

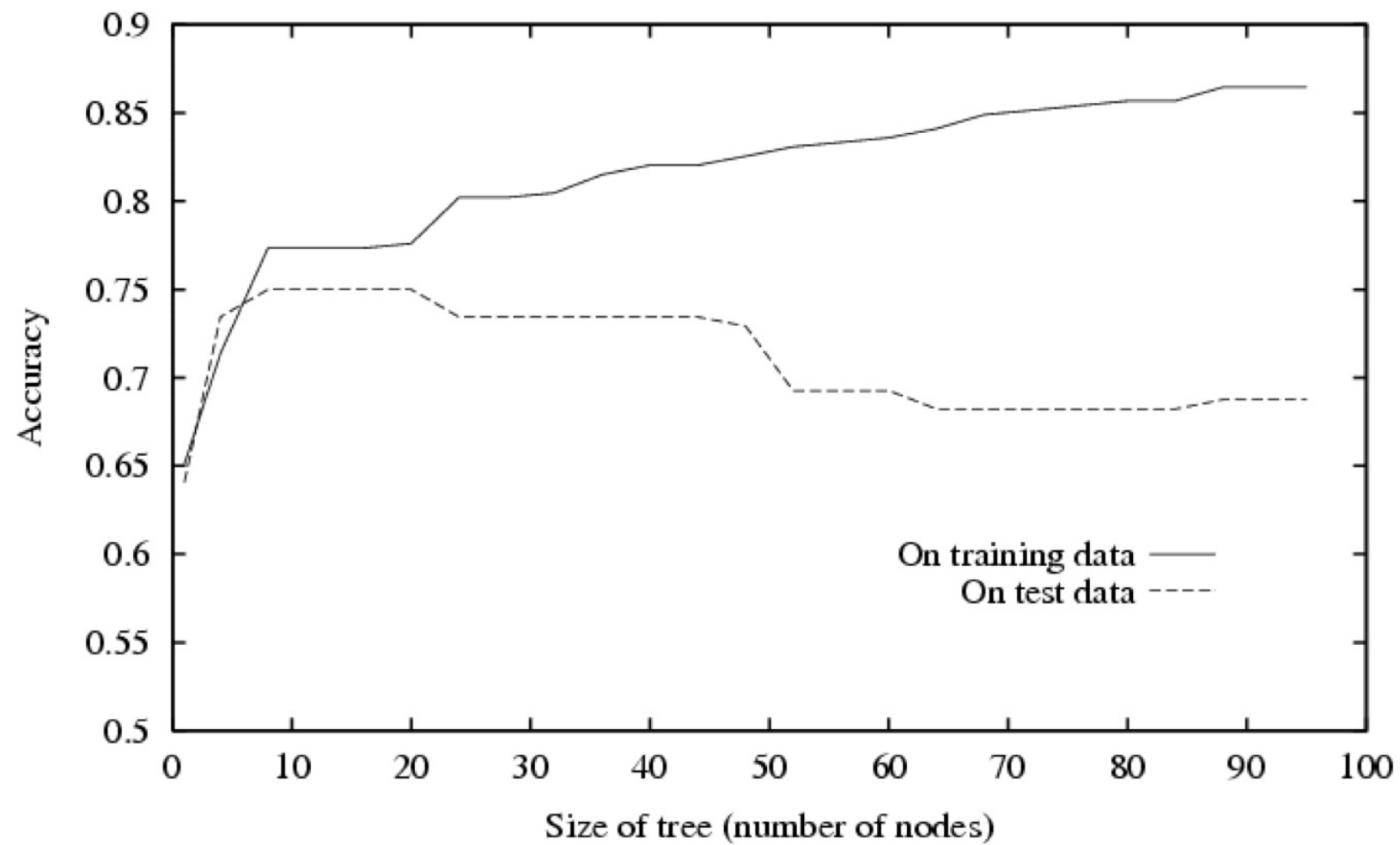






This tree only misclassifies one training data point!

Overfitting in Decision Trees

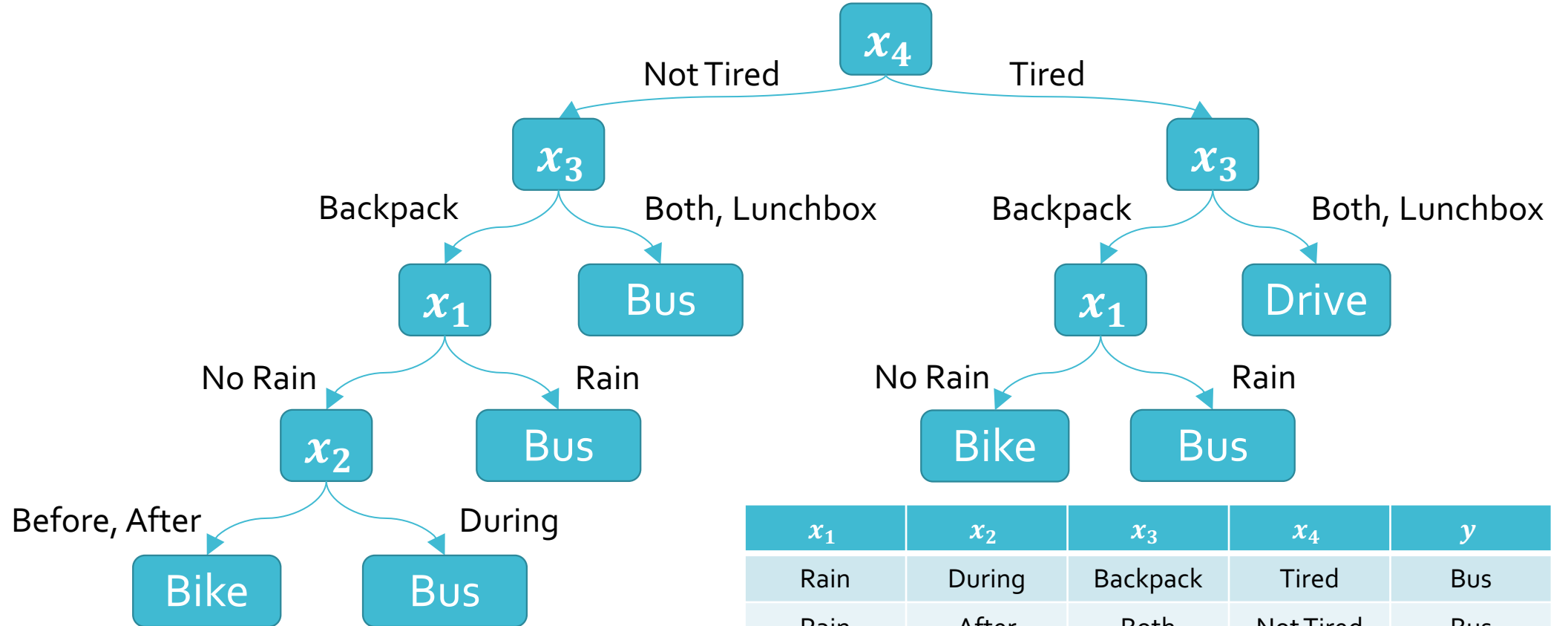


Combatting Overfitting in Decision Trees

- Intuition: deeper trees are “more complicated” and thus more liable to overfit
- Heuristics:
 - Do not split leaves past a fixed depth, δ
 - Do not split leaves with fewer than c data points
 - Do not split leaves where the maximal information gain is less than τ
- Take a majority vote in impure leaves

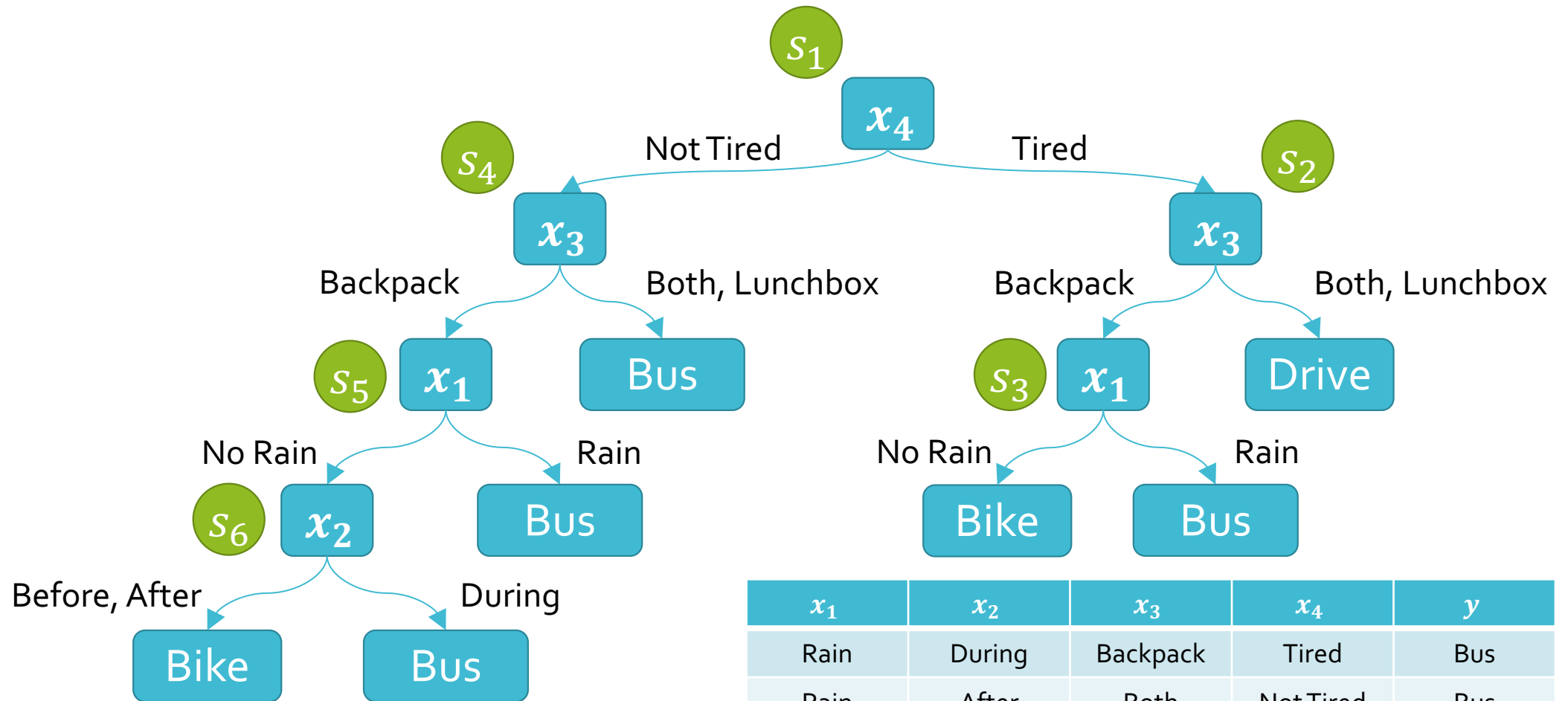
Combatting Overfitting in Decision Trees

- Reduced Error Pruning:
 1. Learn a decision tree
 2. Evaluate each split using a “validation” dataset by comparing the validation error rate with and without that split
 3. Greedily remove the split that most decreases the validation error rate
 - Break ties in favor of smaller trees
 4. Stop if no split is removed



$D_{val} =$

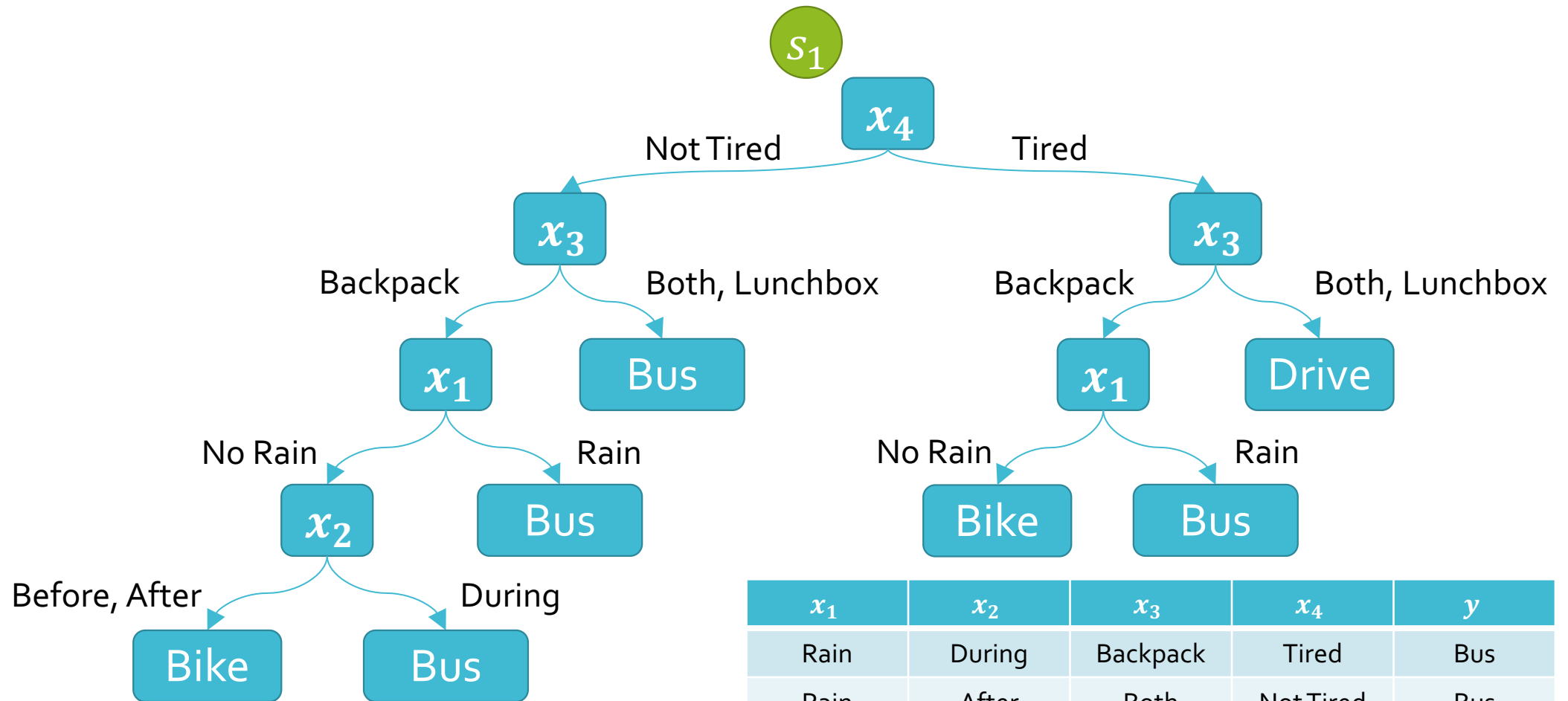
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



$D_{val} =$

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

$err(h, D_{val}) = 0.2$



$D_{val} =$

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

$err(h - s_1, D_{val})$



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

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

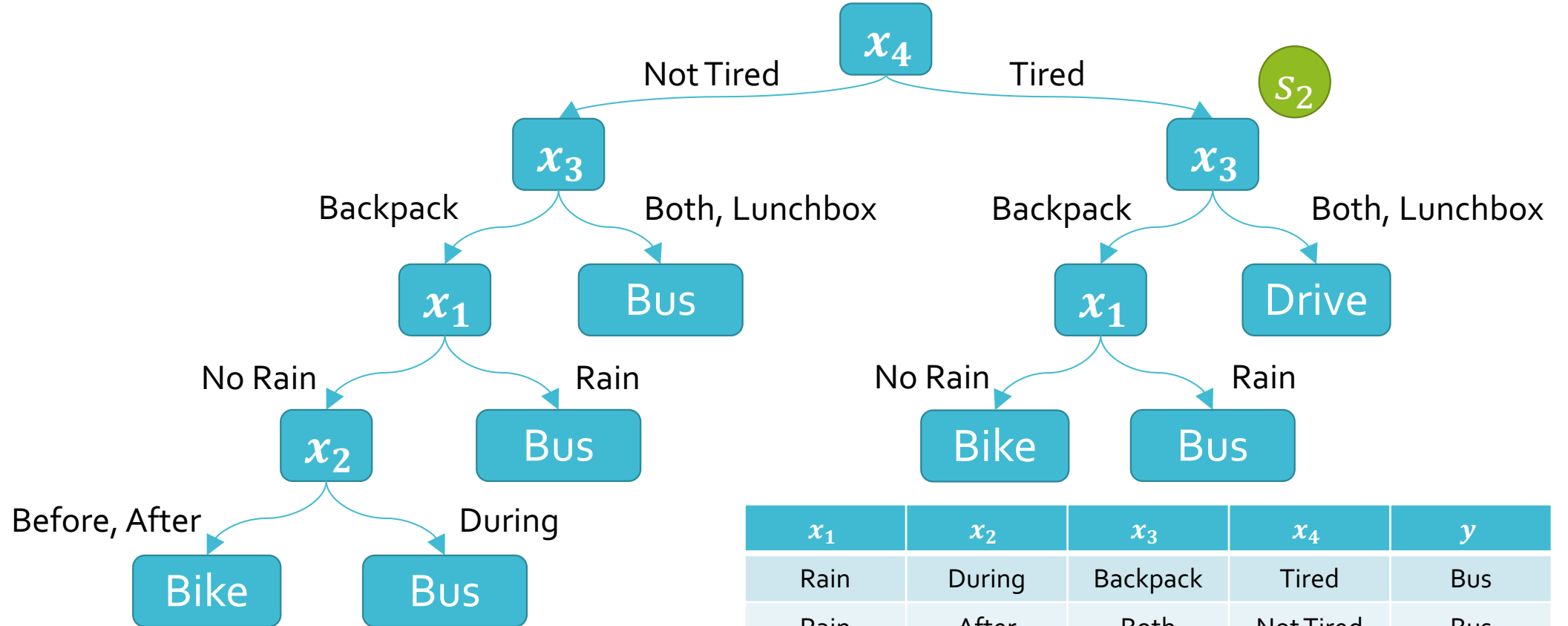
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



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

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

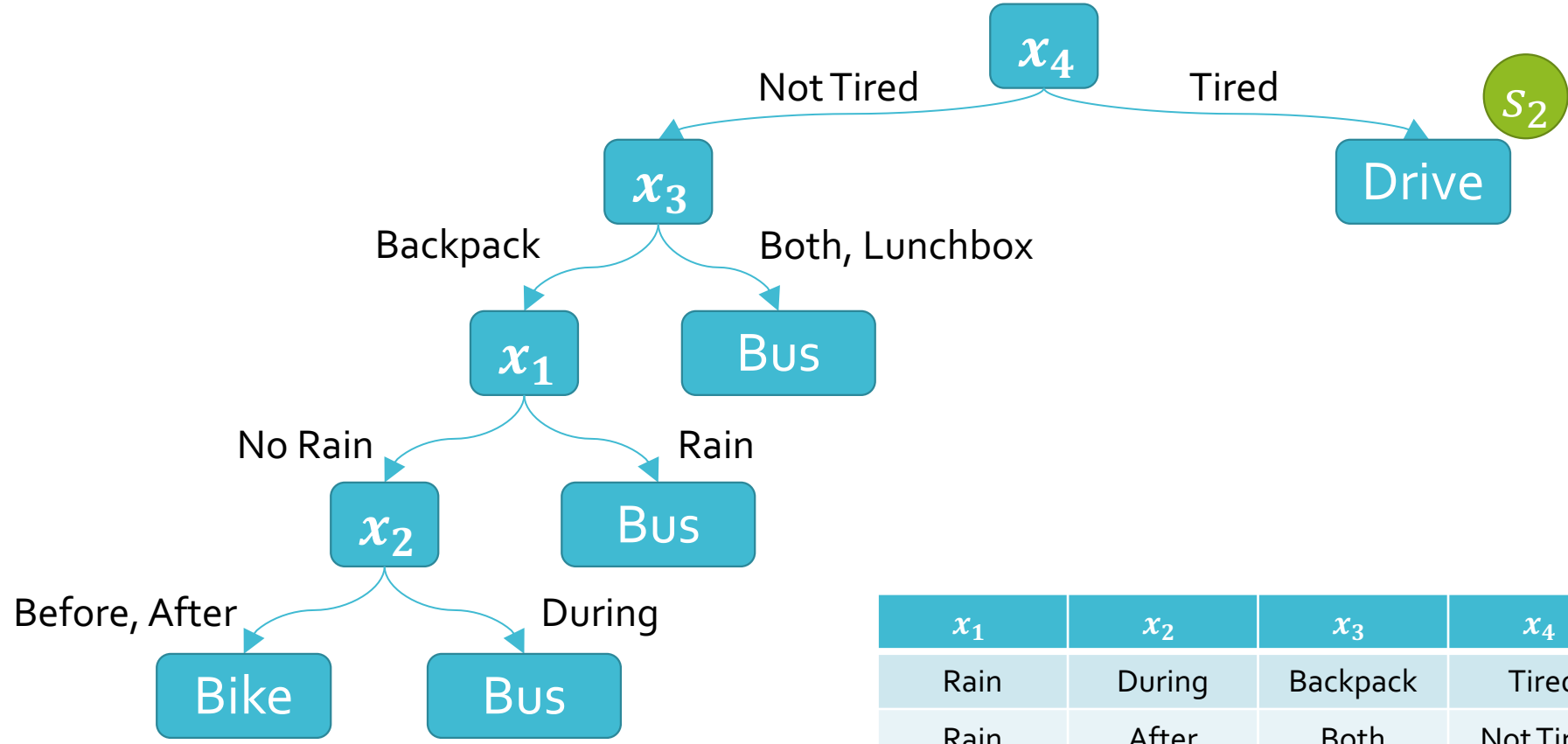
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



$D_{val} =$

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

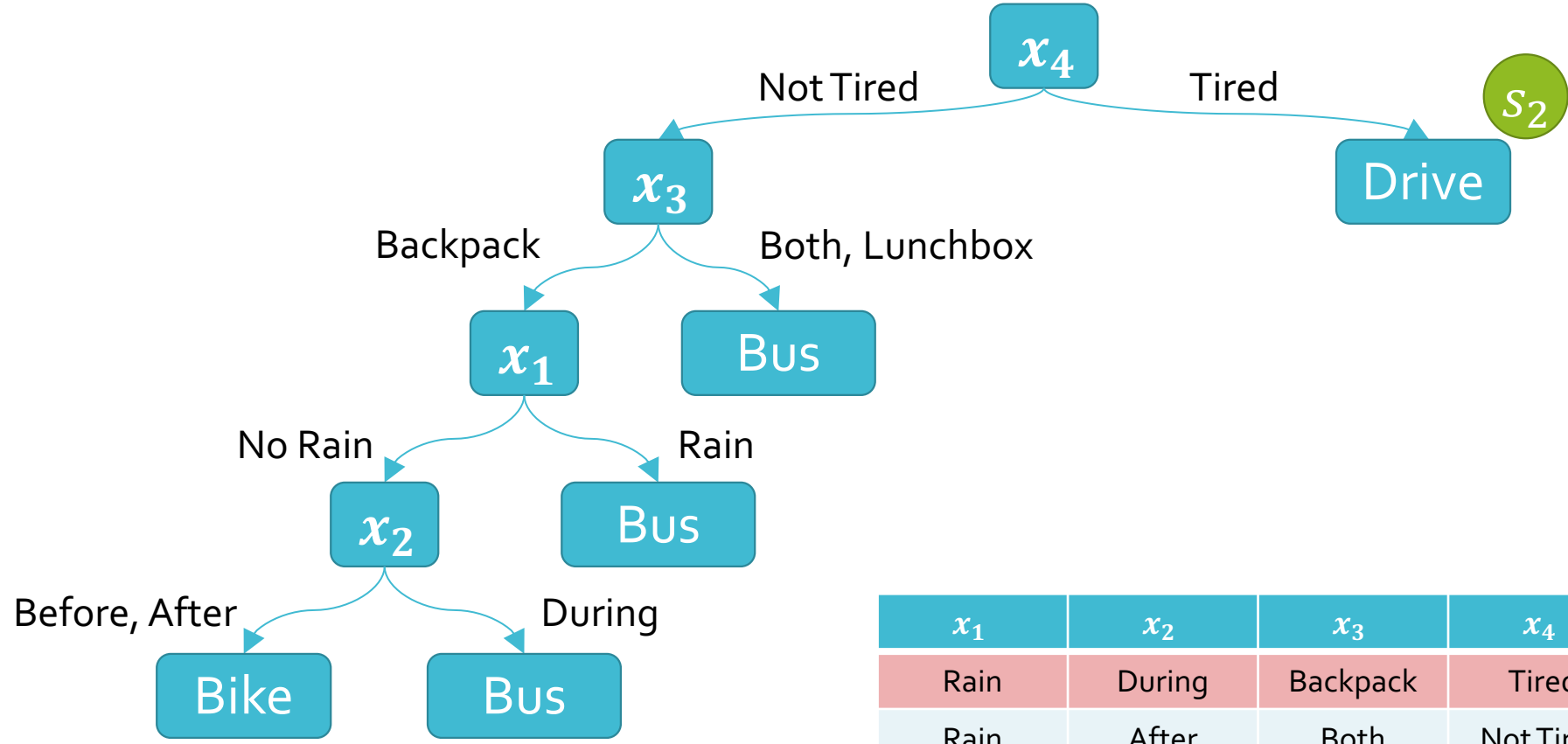
$err(h - s_2, D_{val})$



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

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

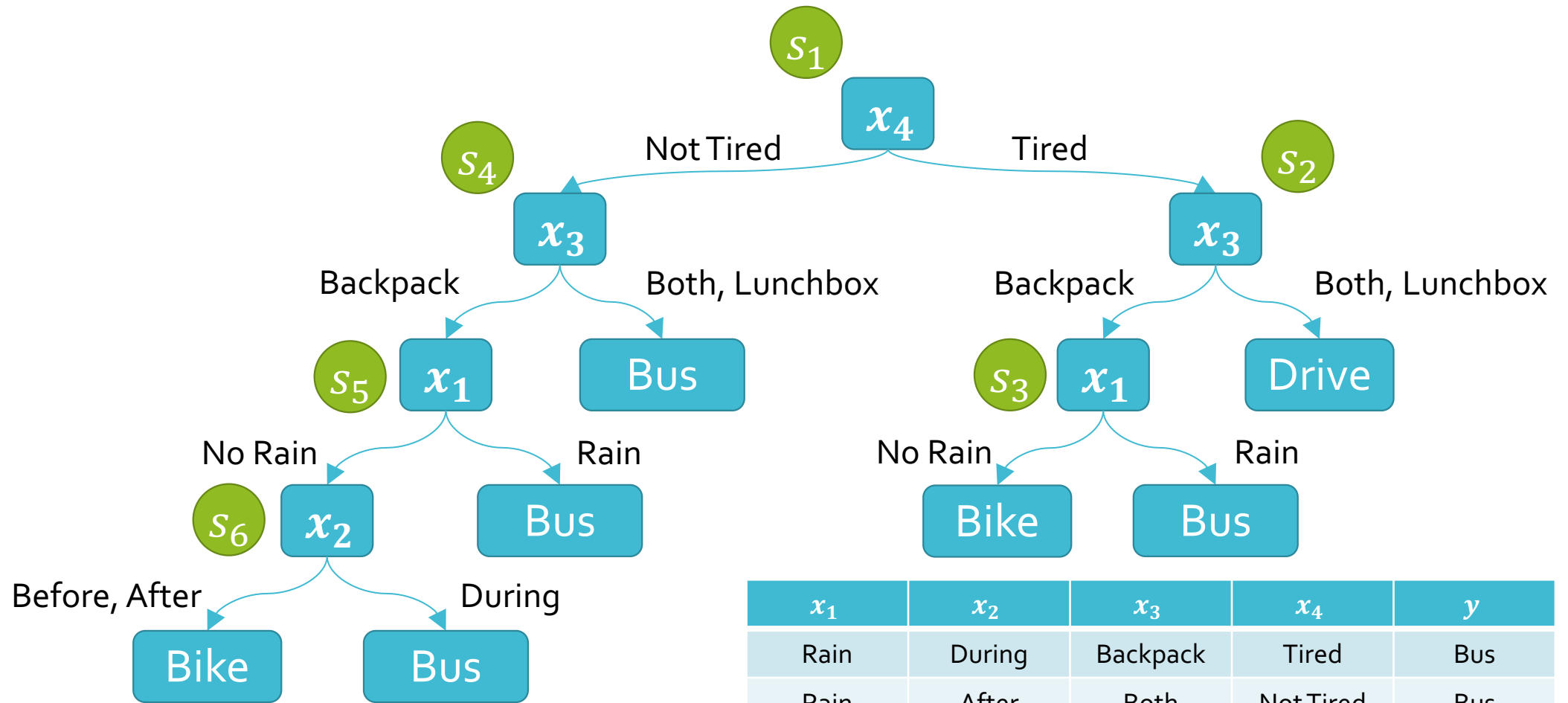
$err(h - s_2, \mathcal{D}_{val})$



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

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

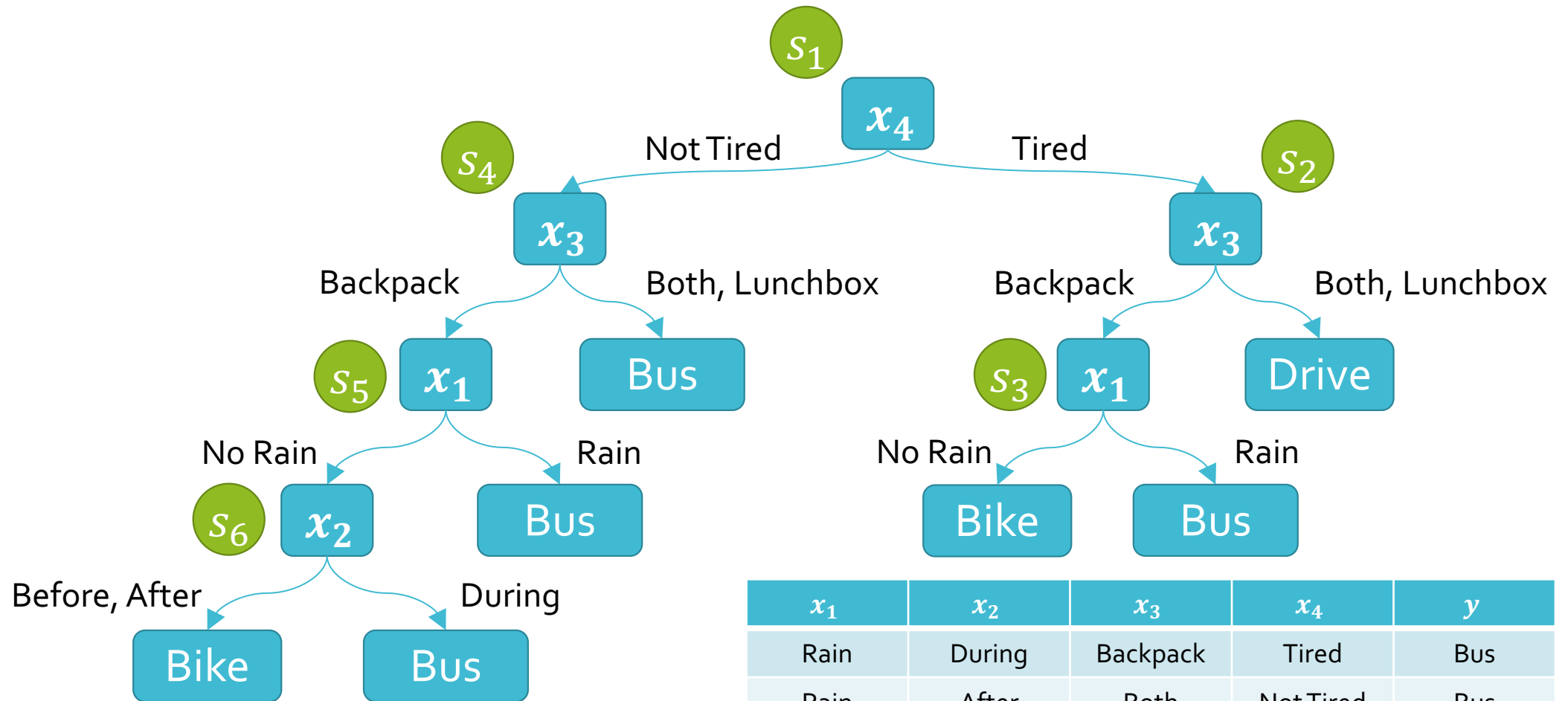
$err(h - s_2, \mathcal{D}_{val}) = 0.4$



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

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

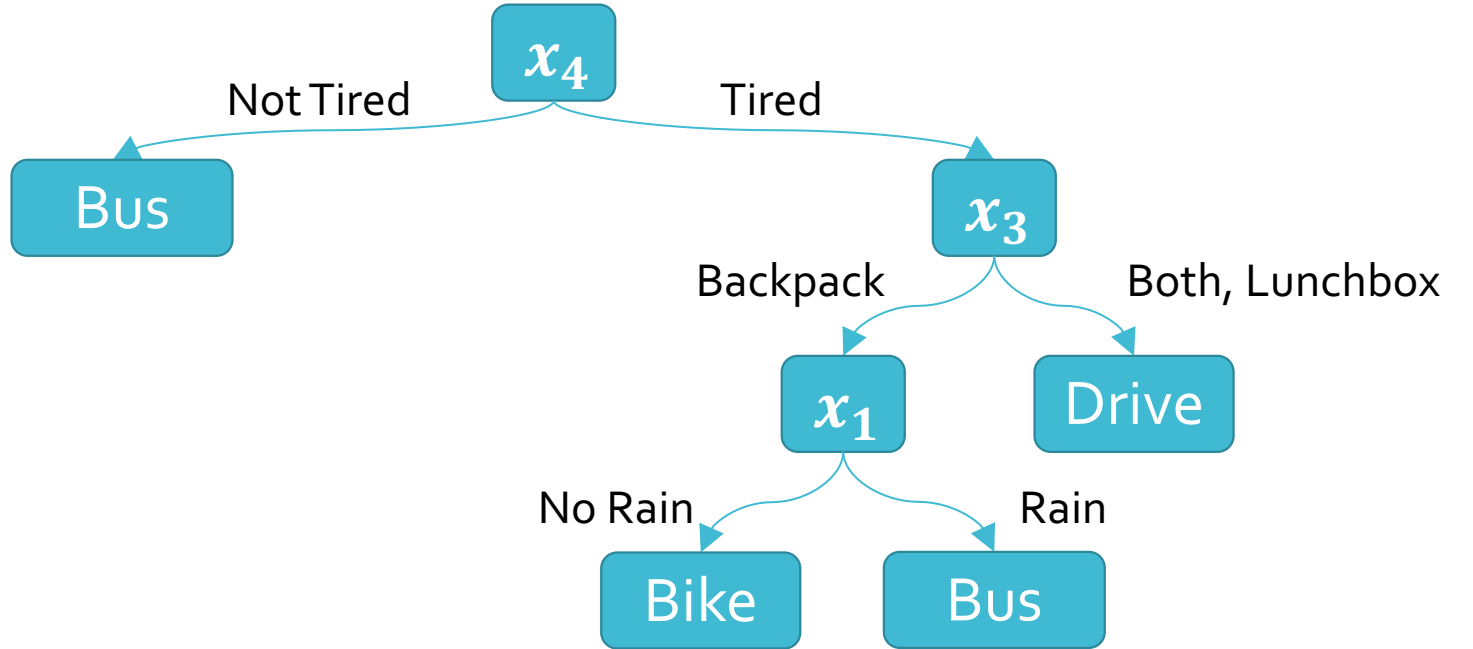
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



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

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

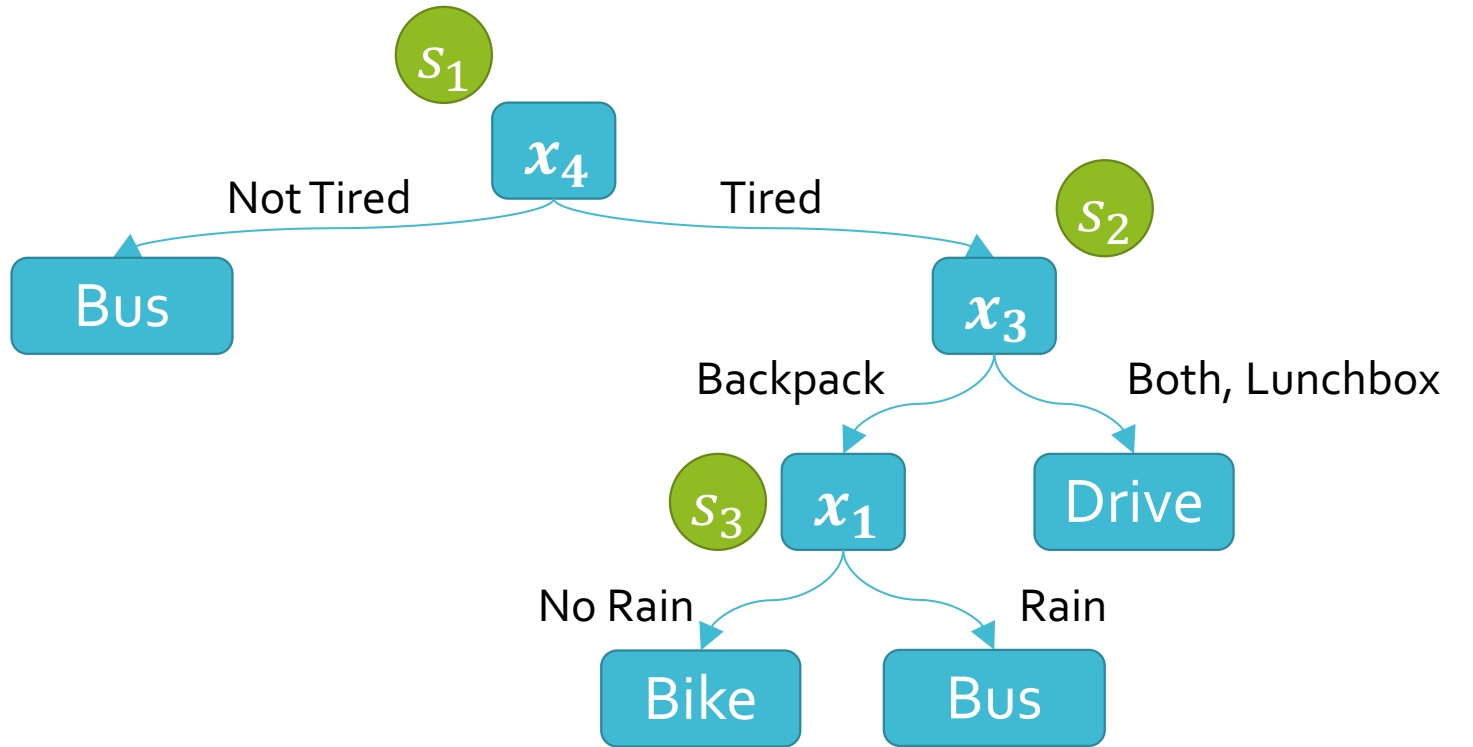
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



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

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

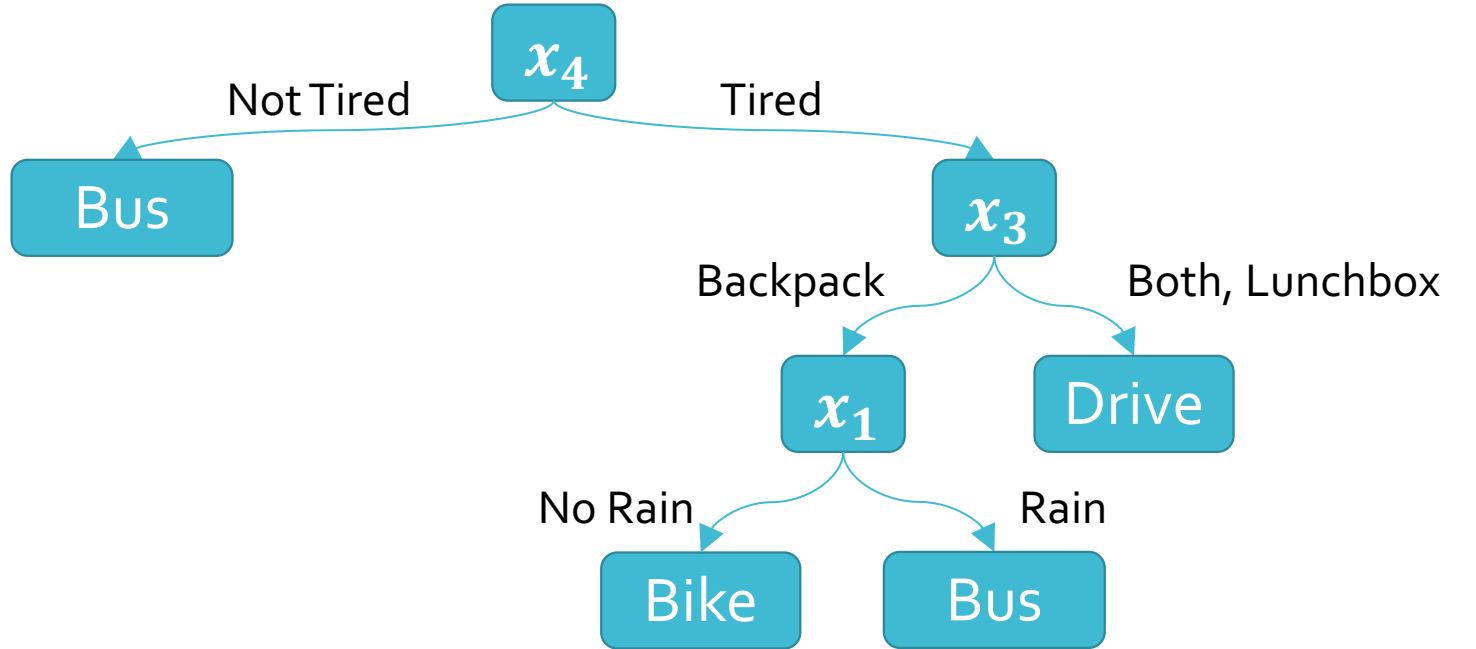
$err(h, \mathcal{D}_{val}) = 0$



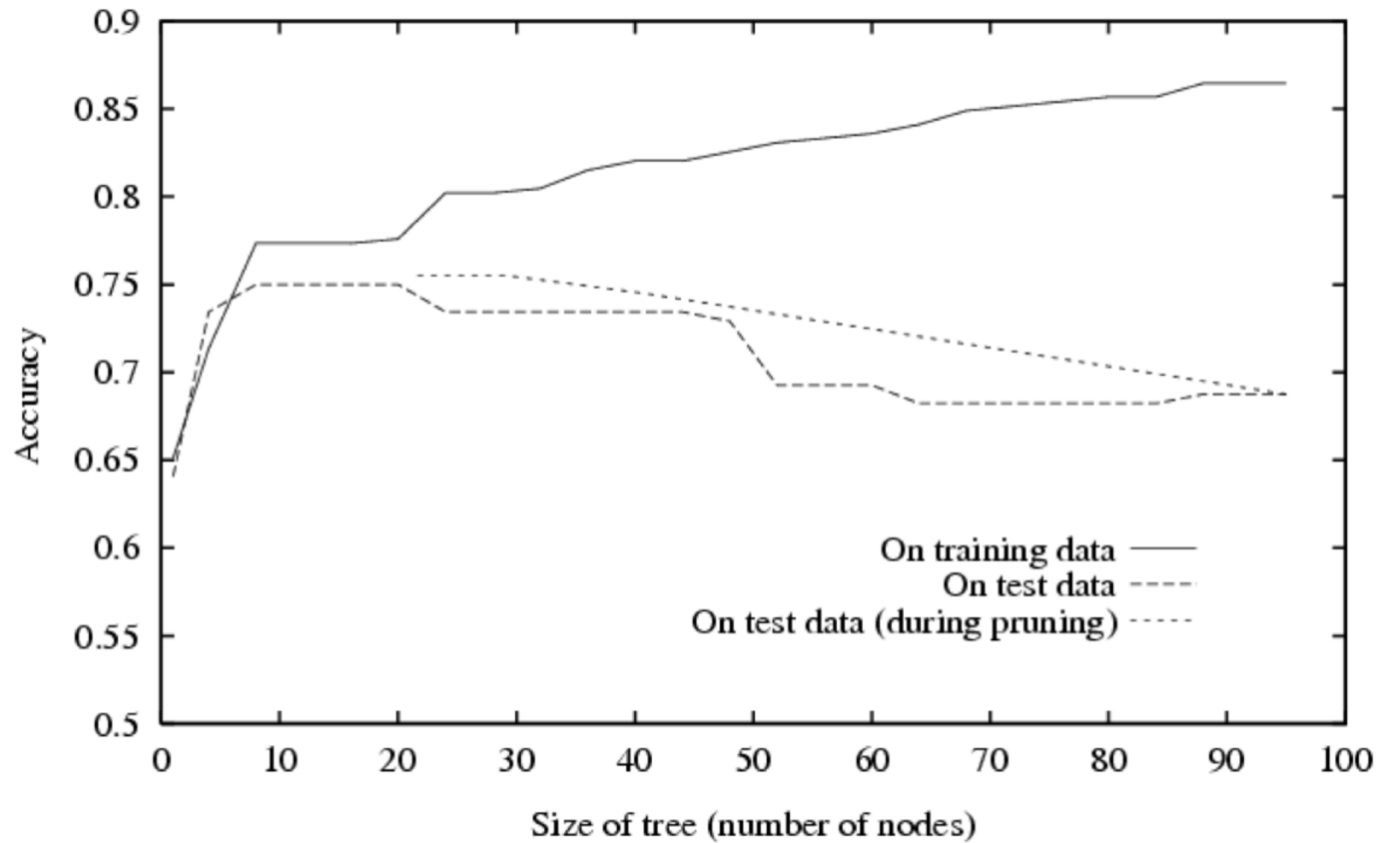
s	s_1	s_2	s_3
$err(h - s, \mathcal{D}_{val})$	0.4	0.2	0.2

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

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



Pruning Decision Trees



Key Takeaways

- Decision tree prediction algorithm
- Decision tree learning algorithm via recursion
- Inductive bias of decision trees
- Overfitting vs. Underfitting
- How to combat overfitting in decision trees

Real-valued Features



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)

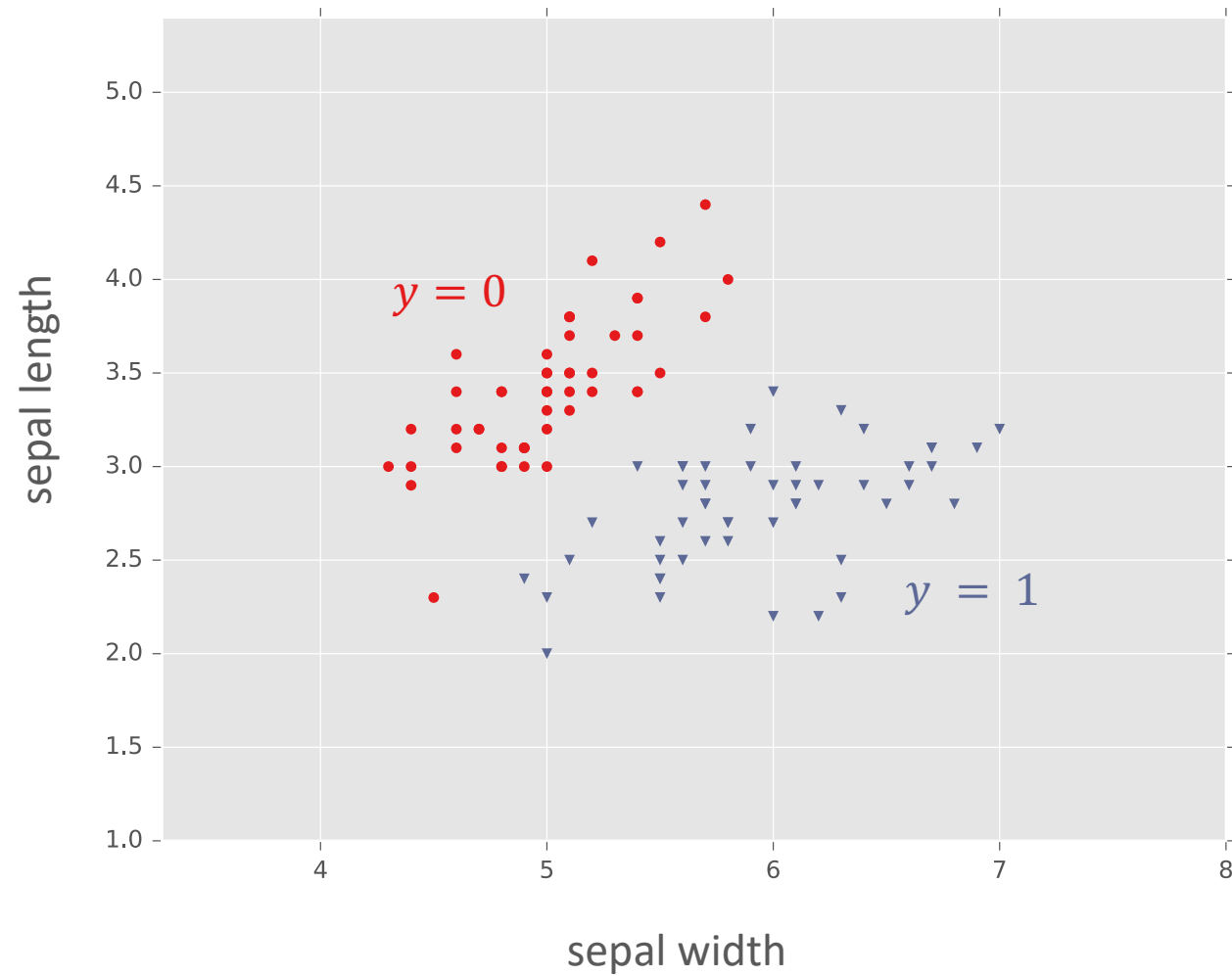
Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

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

Fisher Iris Dataset





WIKIPEDIA
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

Article

[Talk](#)

Duck test

From Wikipedia, the free encyclopedia

For the use of "the duck test" within the Wikipedia community, see [Wikipedia:DUCK](#).

The **duck test** is a form of [abductive reasoning](#). This is its usual expression:

If it looks like a duck, swims like a duck, and quacks like a duck, then it probably *is* a duck.

The Duck Test

The Duck Test for Machine Learning

- Classify a point as the label of the “most similar” training point
- Idea: given real-valued features, we can use a distance metric to determine how similar two data points are
- A common choice is Euclidean distance:

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2 = \sqrt{\sum_{d=1}^D (x_d - x'_d)^2}$$

- An alternative is the Manhattan distance:

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{d=1}^D |x_d - x'_d|$$

Nearest Neighbor Model

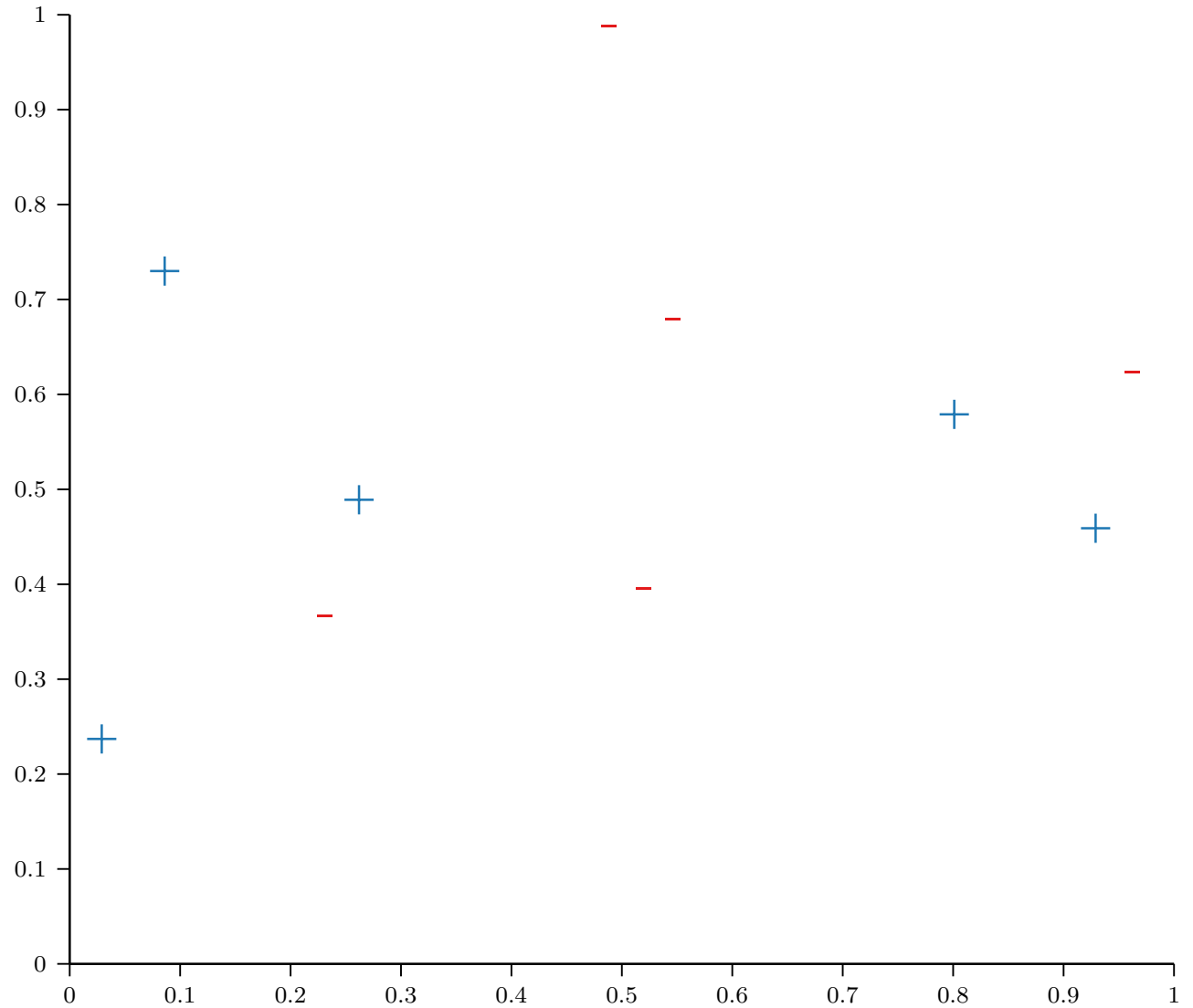
- Classify a point as the label of the “most similar” training point
- Given a training dataset $\mathcal{D}_{train} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$

$$\text{Let } \hat{i}(\mathbf{x}') = \underset{i \in \{1, \dots, N\}}{\operatorname{argmin}} d(\mathbf{x}^{(i)}, \mathbf{x}')$$

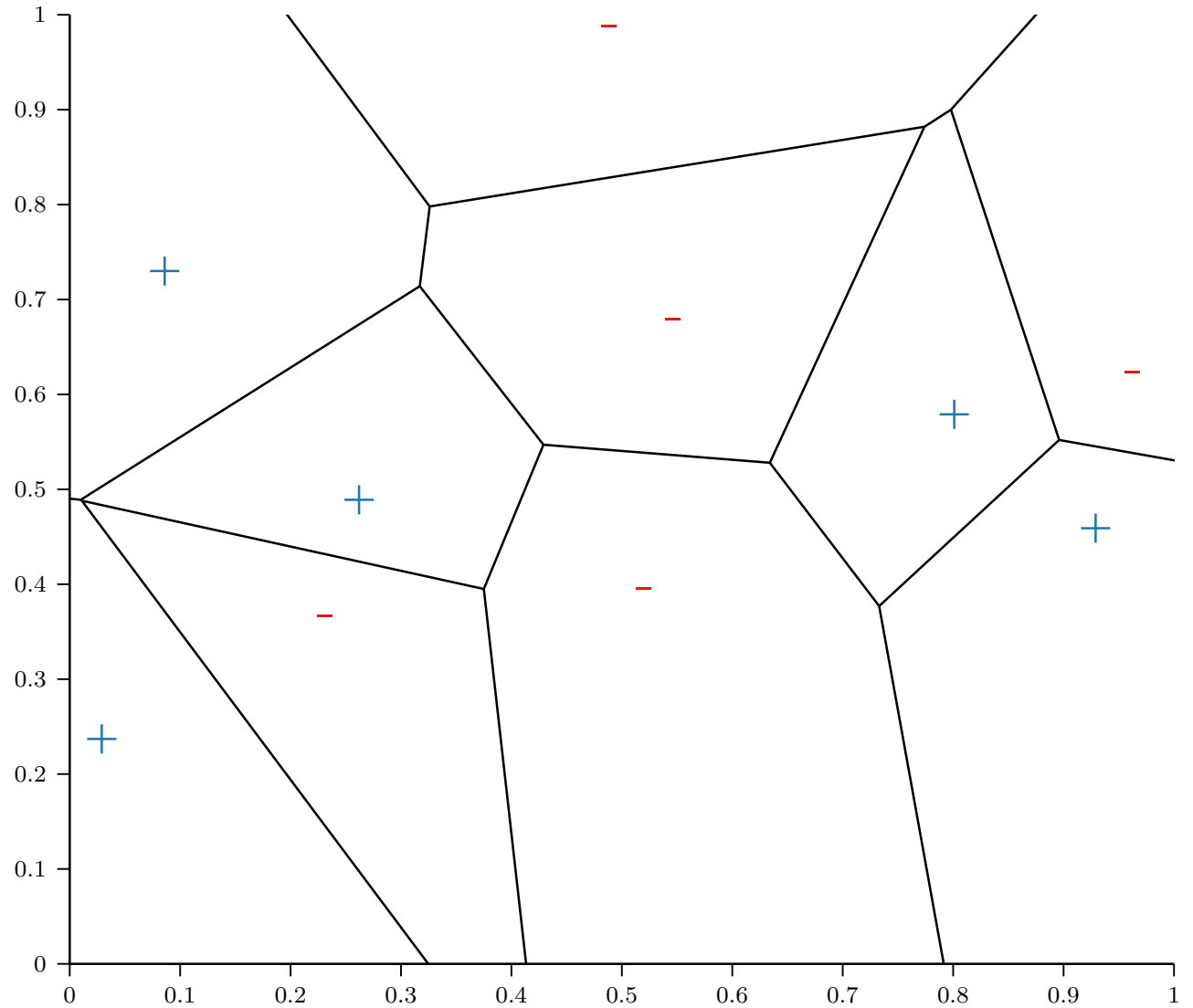
- Then the nearest neighbor classifier can be written as

$$h(\mathbf{x}') = y^{(\hat{i}(\mathbf{x}'))}$$

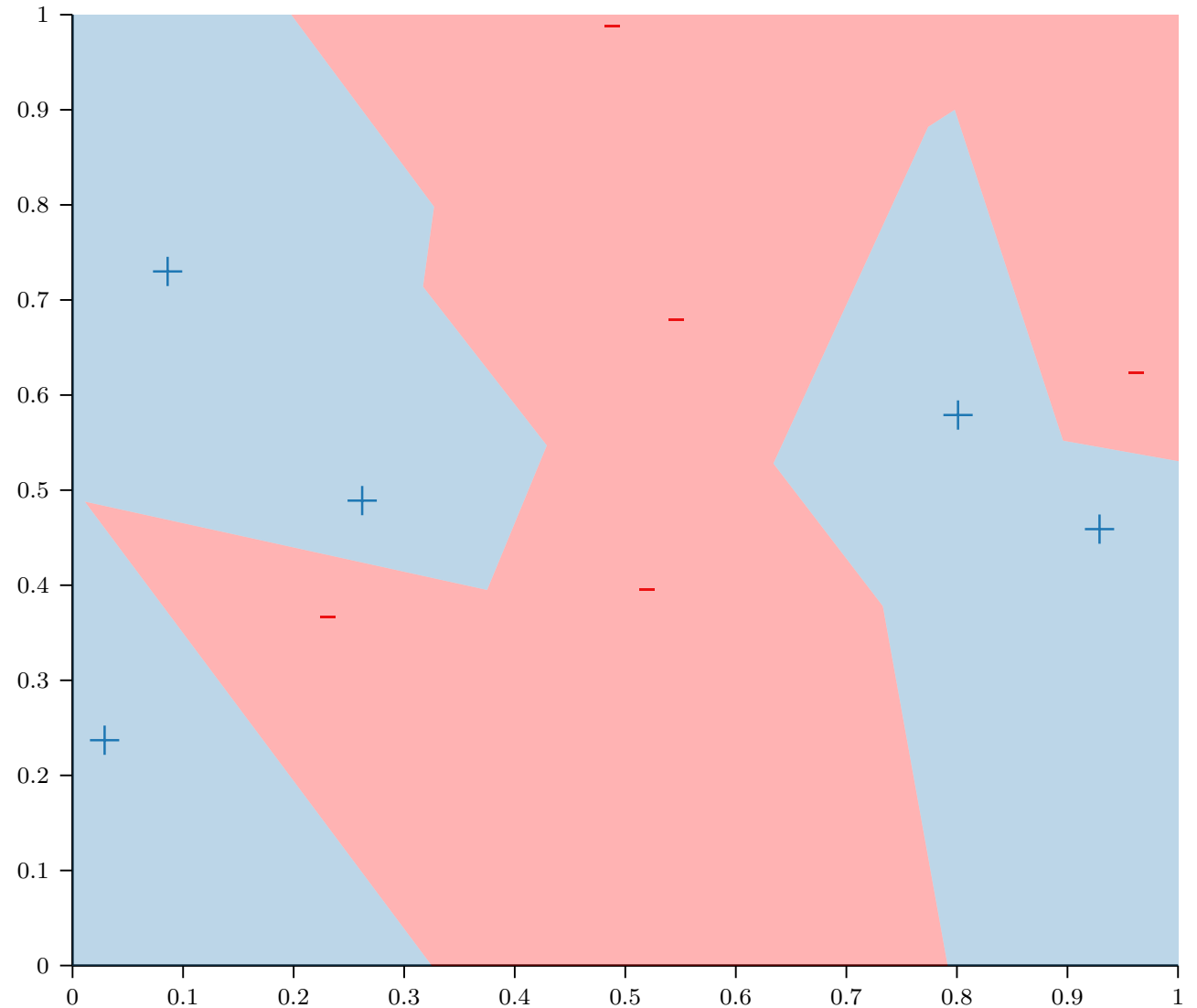
Nearest Neighbor: Example



Nearest Neighbor: Example



Nearest Neighbor: Example



The Nearest Neighbor Model

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

⋮

- Always has zero training error...

Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 * \text{the Bayes error rate (the optimal classifier)}$
- Proof:
 - Assume a binary classification problem: $\mathcal{Y} = \{1, 0\}$
 - Assume data points are drawn *independently* from some probability distribution
 - Assume labels are *stochastic*: let $\pi(\mathbf{x}) = P\{y = 1|\mathbf{x}\}$
 - Assume $\pi(\mathbf{x})$ is continuous
 - As $N \rightarrow \infty$, $\mathbf{x}^{(\hat{i}(\mathbf{x}'))} \rightarrow \mathbf{x}' \implies \pi(\mathbf{x}^{(\hat{i}(\mathbf{x}'))}) \rightarrow \pi(\mathbf{x}')$

Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 * \text{the Bayes error rate (the optimal classifier)}$
- Proof (cont.):
- $err(h) = \mathbb{E}_{\mathbf{x}'}[\mathbb{1}(h(\mathbf{x}') \neq y')] = P\{h(\mathbf{x}') \neq y'\}$
 $= P\{h(\mathbf{x}') = 1, y' = 0\} + P\{h(\mathbf{x}') = 0, y' = 1\}$
 $= \pi(\mathbf{x}^{(\hat{i}(\mathbf{x}'))})(1 - \pi(\mathbf{x}')) + (1 - \pi(\mathbf{x}^{(\hat{i}(\mathbf{x}'))}))\pi(\mathbf{x}')$
 $\rightarrow \pi(\mathbf{x}')(1 - \pi(\mathbf{x}')) + (1 - \pi(\mathbf{x}'))\pi(\mathbf{x}')$
 $= 2\pi(\mathbf{x}')(1 - \pi(\mathbf{x}'))$
 $\leq 2 \min(\pi(\mathbf{x}'), (1 - \pi(\mathbf{x}'))) = 2err(h^*) \blacksquare$

Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 * \text{the Bayes error rate (the optimal classifier)}$
- Interpretation: “In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor.”

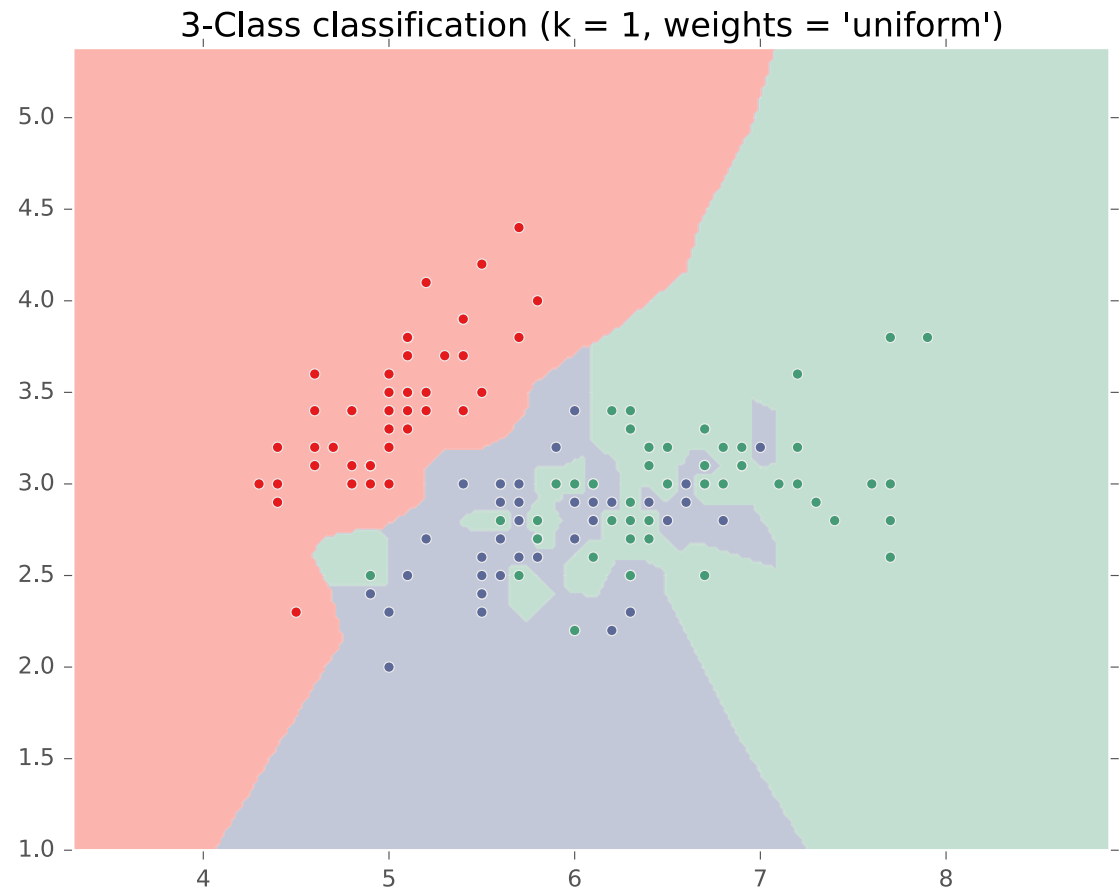
But why limit ourselves to just one neighbor?

- Claim: under certain conditions, as $n \rightarrow \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 * \text{the Bayes error rate (the optimal classifier)}$
- Interpretation: “In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor.”

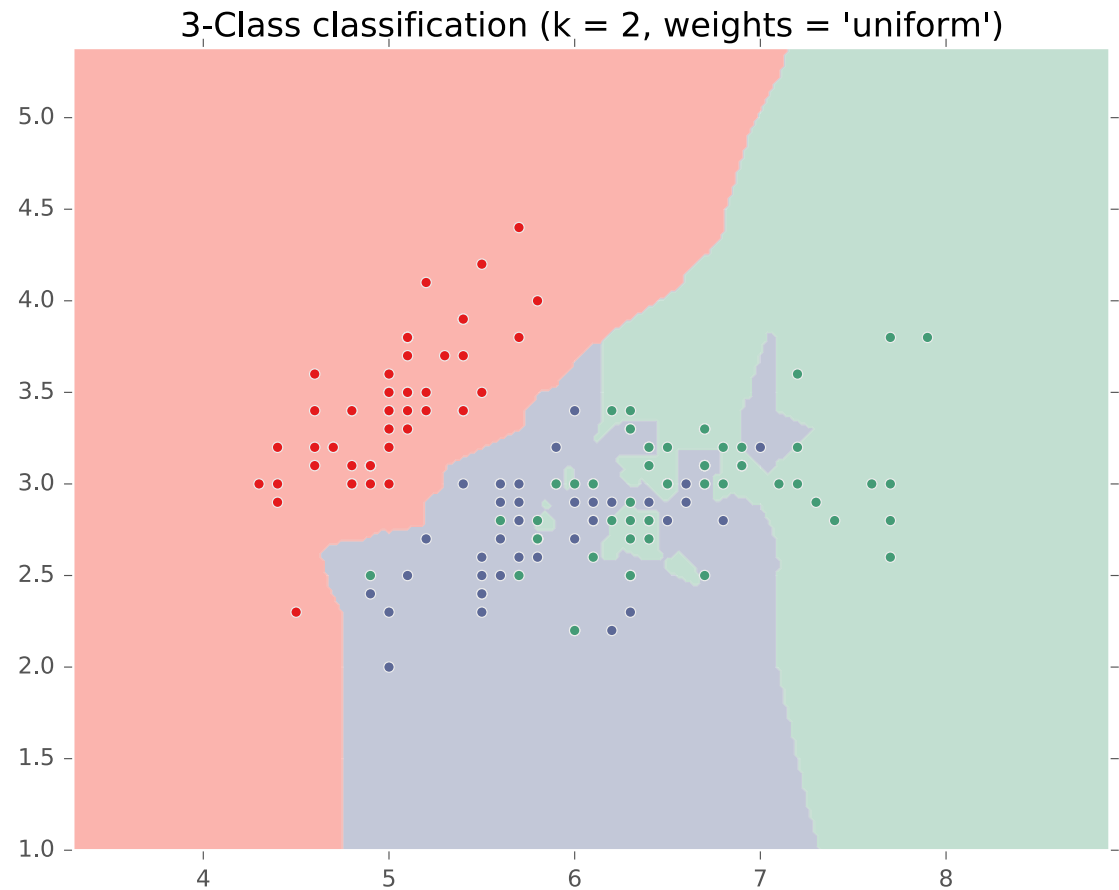
k -Nearest Neighbors (k NN)

- Classify a point as the most common label among the labels of the k nearest training points
- Tie-breaking (in case of even k and/or more than 2 classes)
 - Weight votes by distance
 - Remove furthest neighbor
 - Add next closest neighbor
 - Use a different distance metric

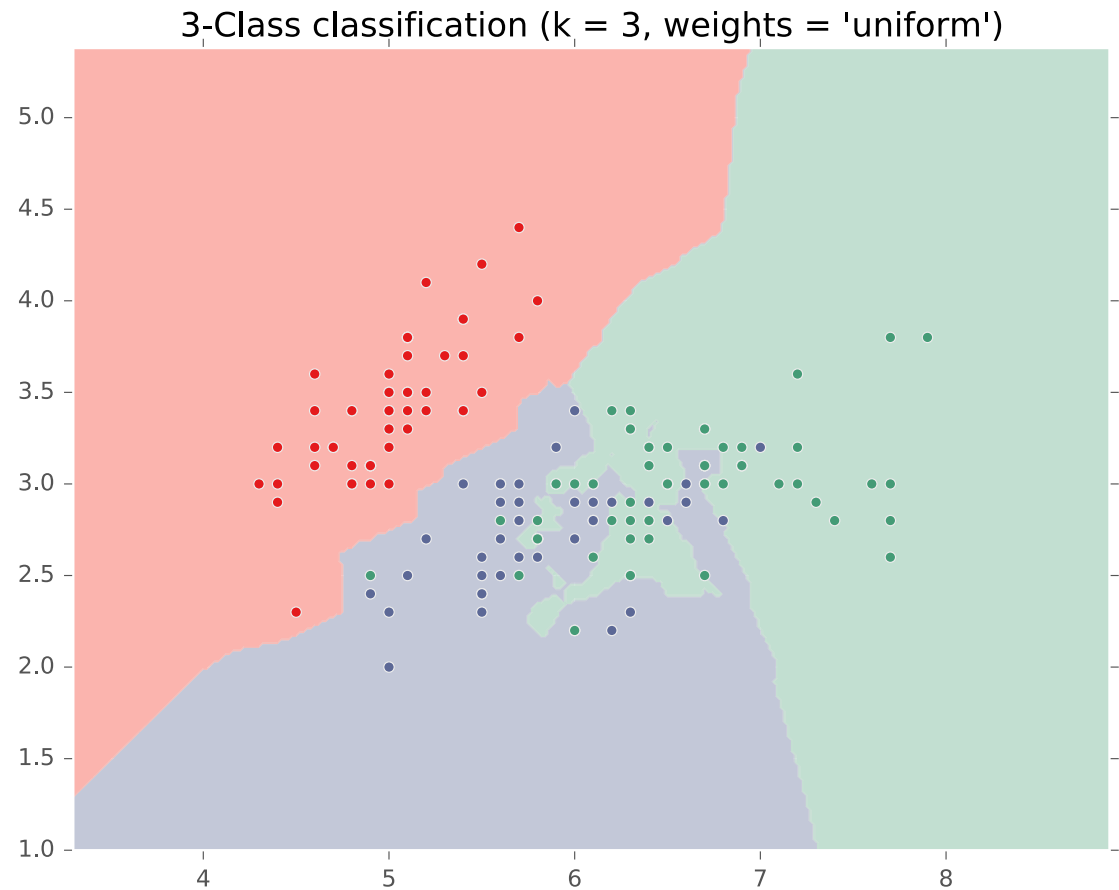
k NN on Fisher Iris Data



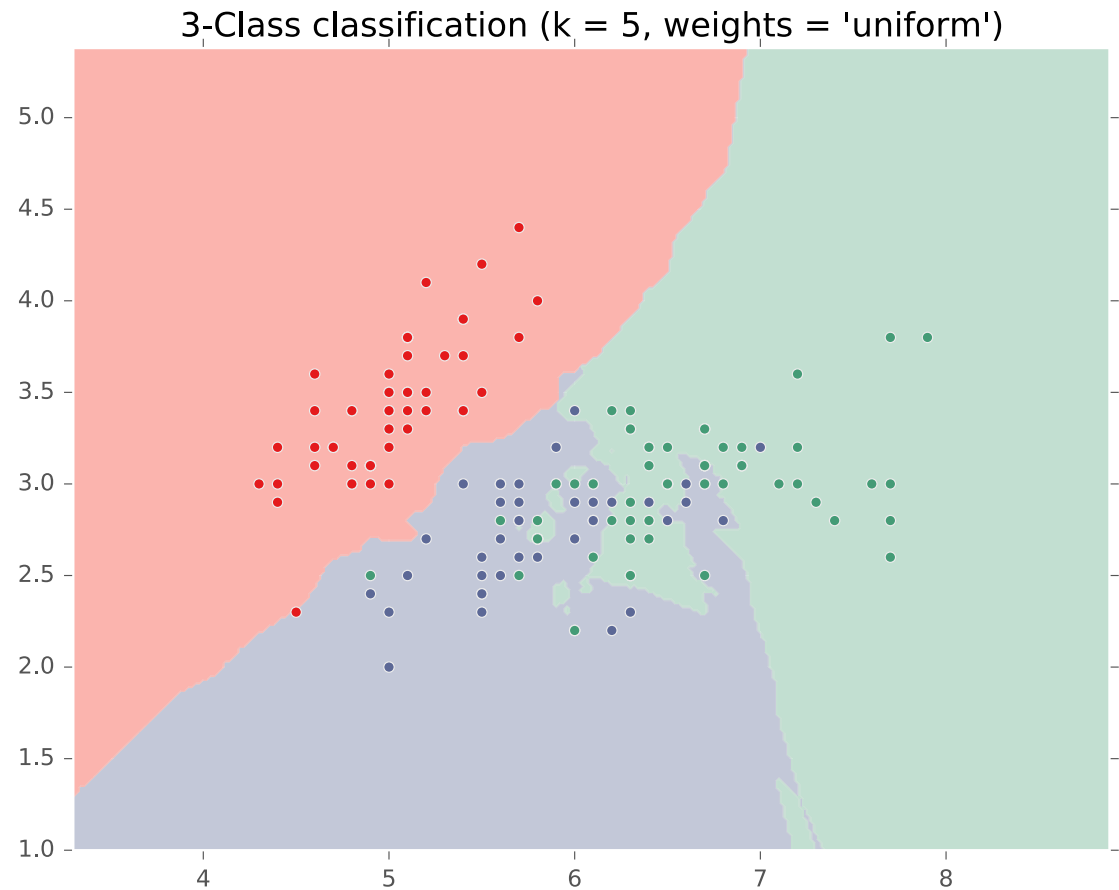
k NN on Fisher Iris Data



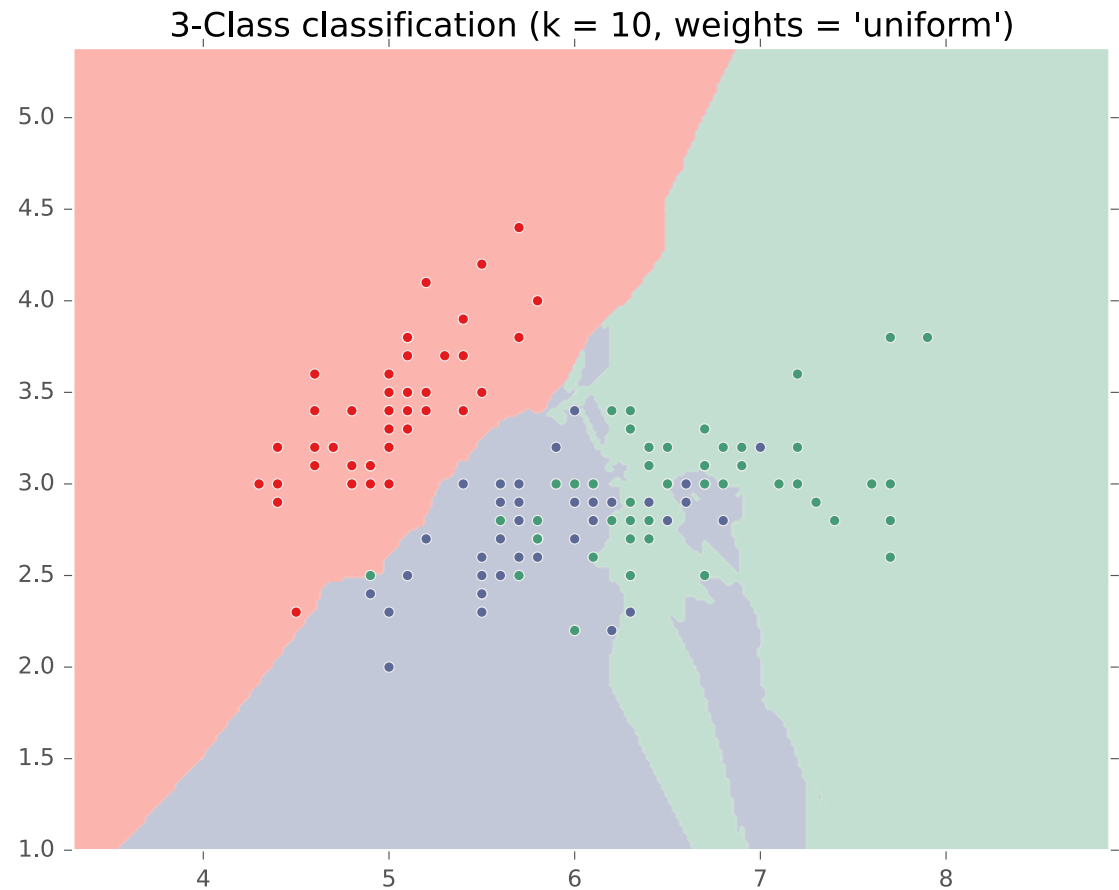
k NN on Fisher Iris Data



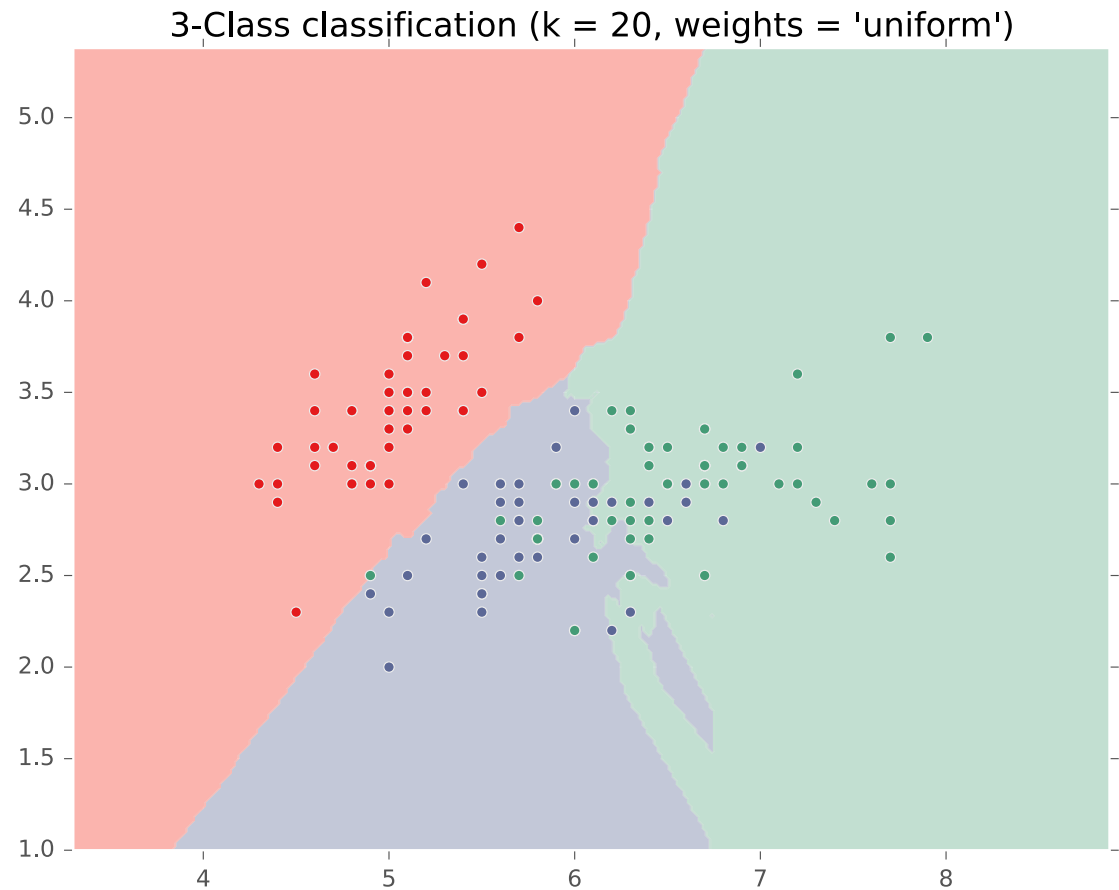
k NN on Fisher Iris Data



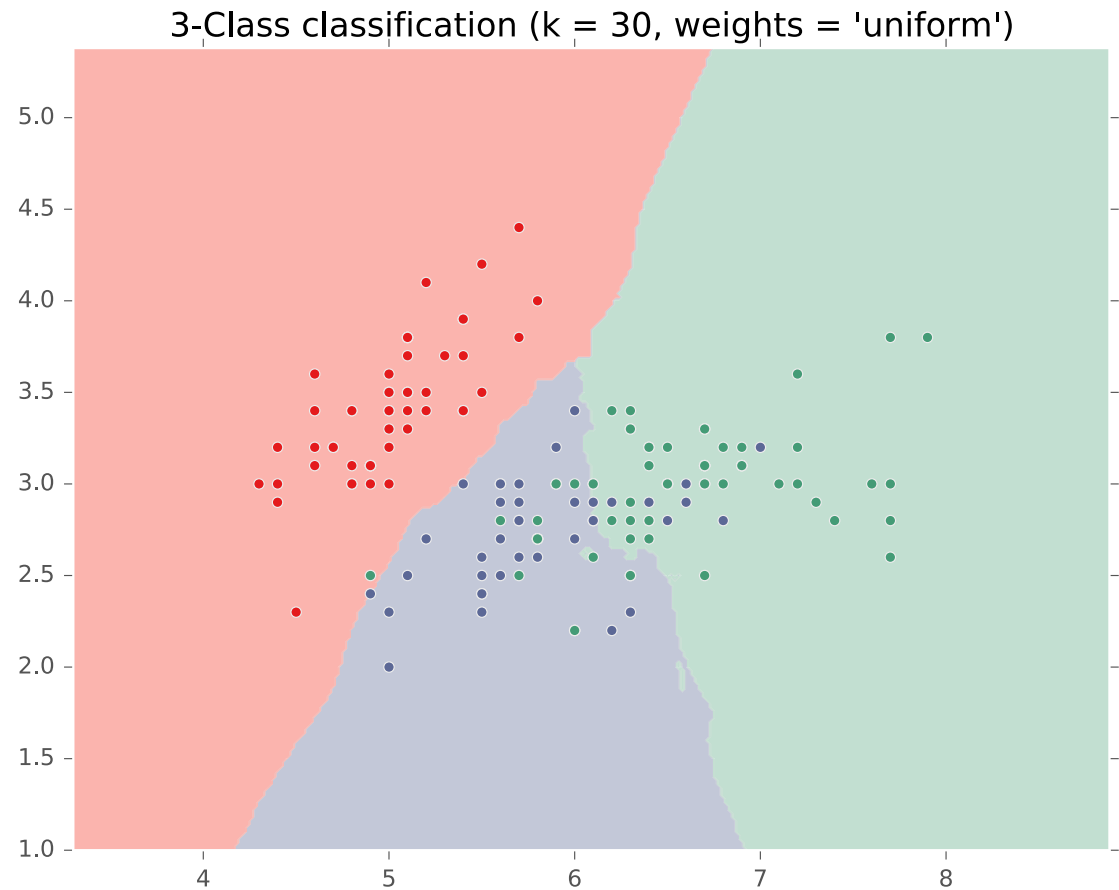
k NN on Fisher Iris Data



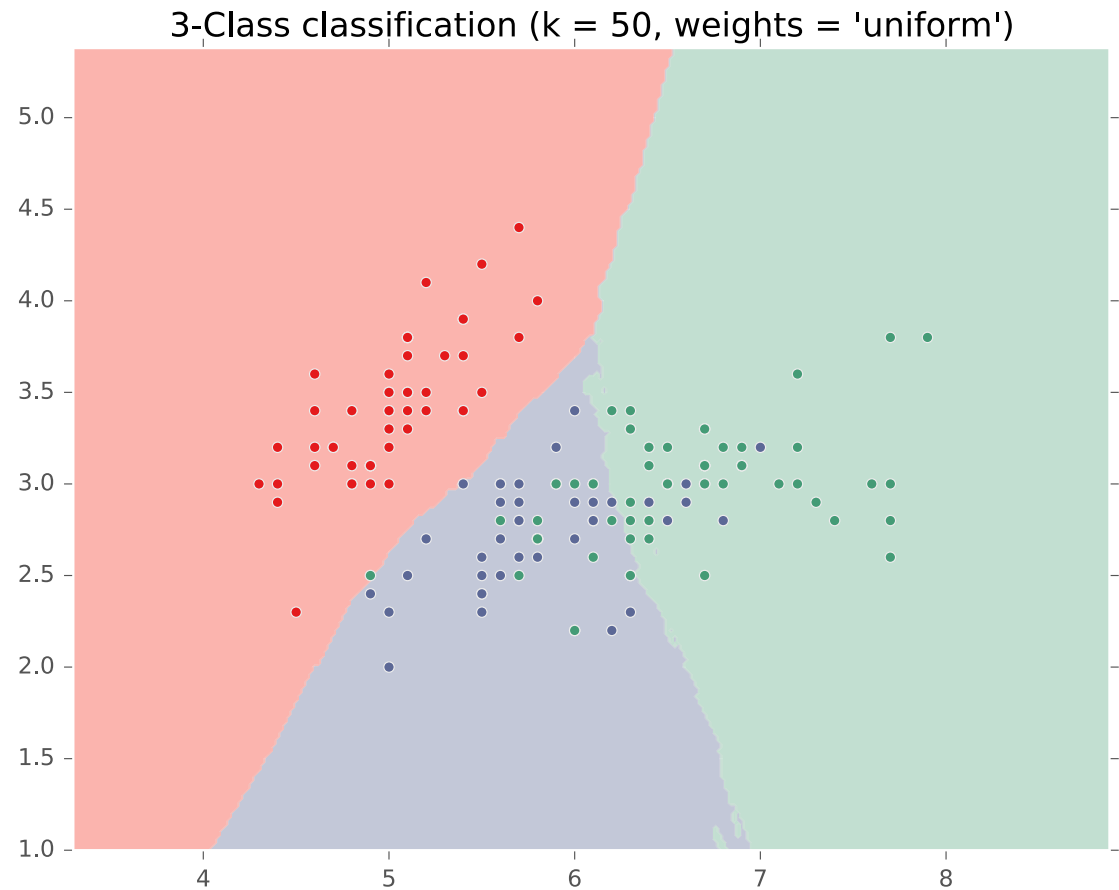
k NN on Fisher Iris Data



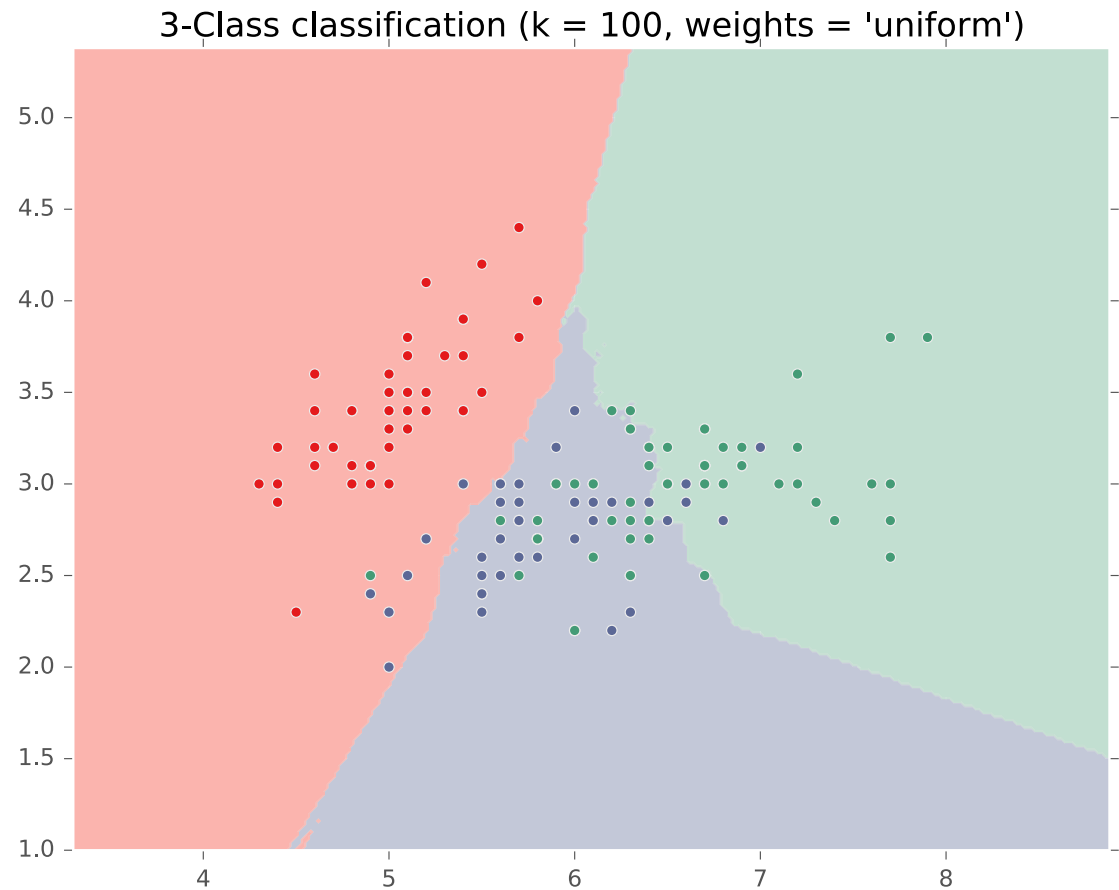
k NN on Fisher Iris Data



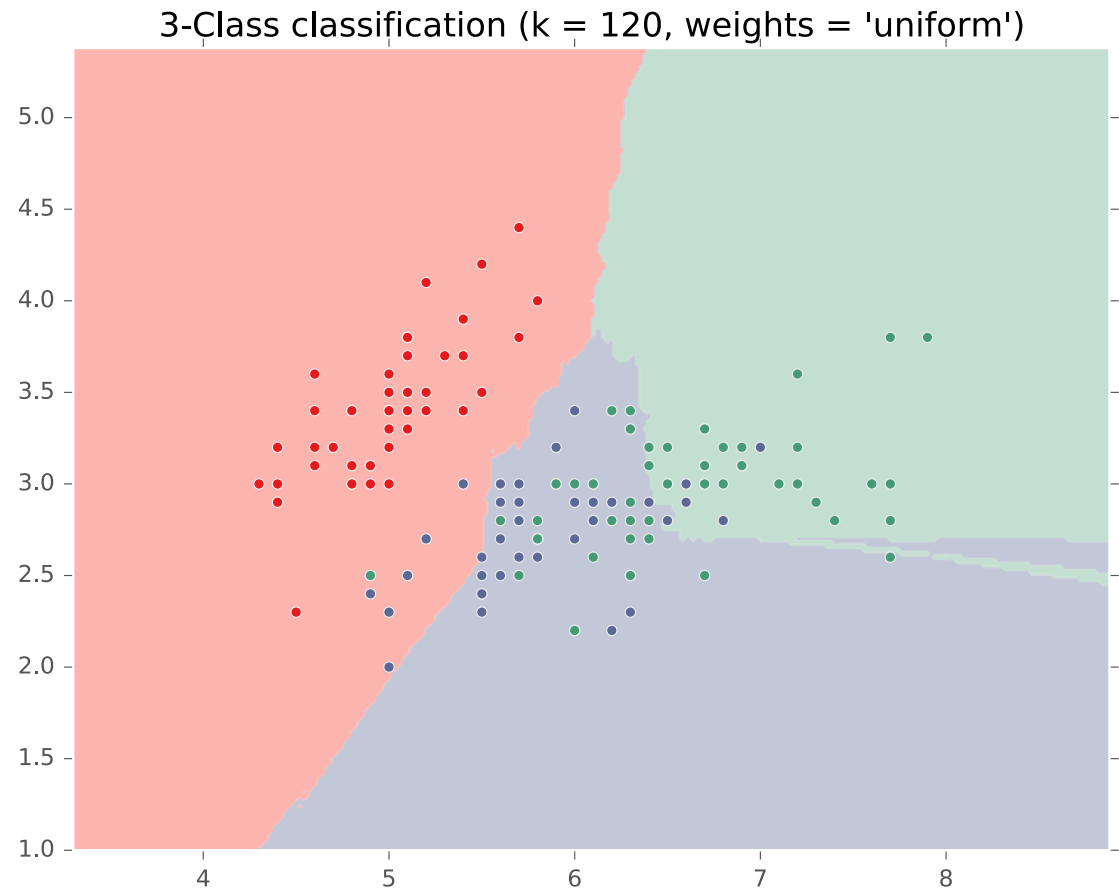
k NN on Fisher Iris Data



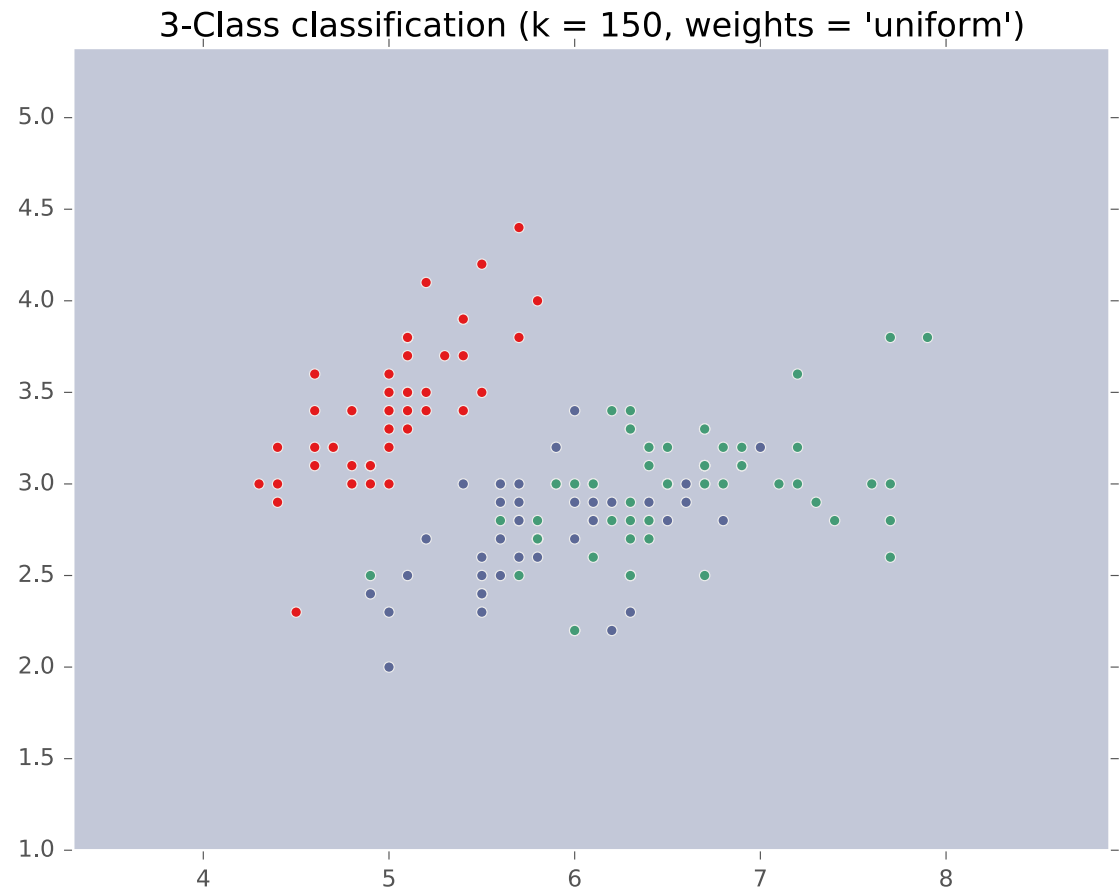
k NN on Fisher Iris Data



k NN on Fisher Iris Data

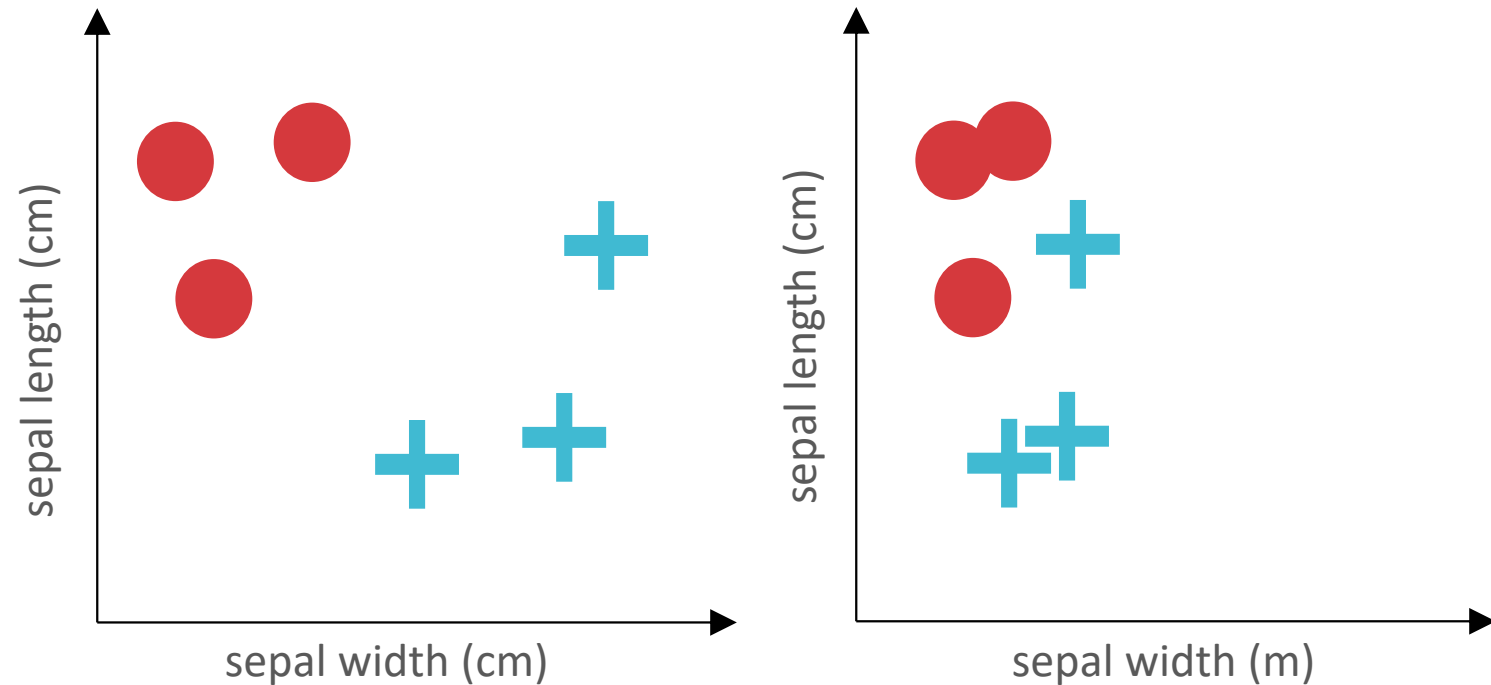


k NN on Fisher Iris Data



k NN: Inductive Bias

- What is the inductive bias of a k NN model that uses the Euclidean distance metric?
- Similar points should have similar labels and *all features are equivalently important for determining similarity*



- Feature scale can dramatically influence results!

Setting k

- When $k = 1$:
 - many, complicated decision boundaries
 - may *overfit*
- When $k = N$:
 - no decision boundaries; always predicts the most common label in the training data
 - may *underfit*
- k controls the complexity of the hypothesis set $\implies k$ affects how well the learned hypothesis will generalize