10-701: Introduction to Machine Learning

# Lecture 3 – KNNs

Hoda Heidari

9/3/2025

* Slides adopted from F24 offering of 10701 by Henry Chai.

# Decision Tree Learning: ID3 Algorithm

1. Start with the entire training dataset, $\mathcal{D}$.
2. For each attribute, $x_d$, calculate the information gain if the dataset were split using that attribute.
3. Select the attribute with the highest information gain as the splitting attribute for the current node.
4. Create a new node in the decision tree with this attribute.
5. For each possible value of the chosen attribute, create a new branch and a corresponding subset of the data.
6. Recursively apply steps 2-6 to each subset until a stopping criteria is met:
   - All examples in the subset have the same label. → predict that label
   - There are no more attributes to split on. → predict majority label
   - There are no more examples in the subset. → predict randomly.
   - …

# Correction Mutual Information

- Mutual information between two random variables $X$ and $Y$ describes how much clarity about the value of one variable is gained by observing the other

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

Where $H(Y|X) = \sum_x p(x)\, H(Y|X = x)$

$$= -\sum_x p(x) \sum_y \frac{p(x,y)}{p(x)} \log_2\left(\frac{p(x,y)}{p(x)}\right)$$

$$= -\sum_{x,y} p(x,y) \log_2\left(\frac{p(x,y)}{p(x)}\right)$$

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

# Mutual Information Symmetry

- Proof by showing that

$$H(Y) - H(Y|X) = \sum_{x,y} p(x,y) \log_2\left(\frac{p(x,y)}{p(x)p(y)}\right) = H(X) - H(X|Y)$$

$$-\sum_{y} p(y) \log(p(y)) + \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)}$$

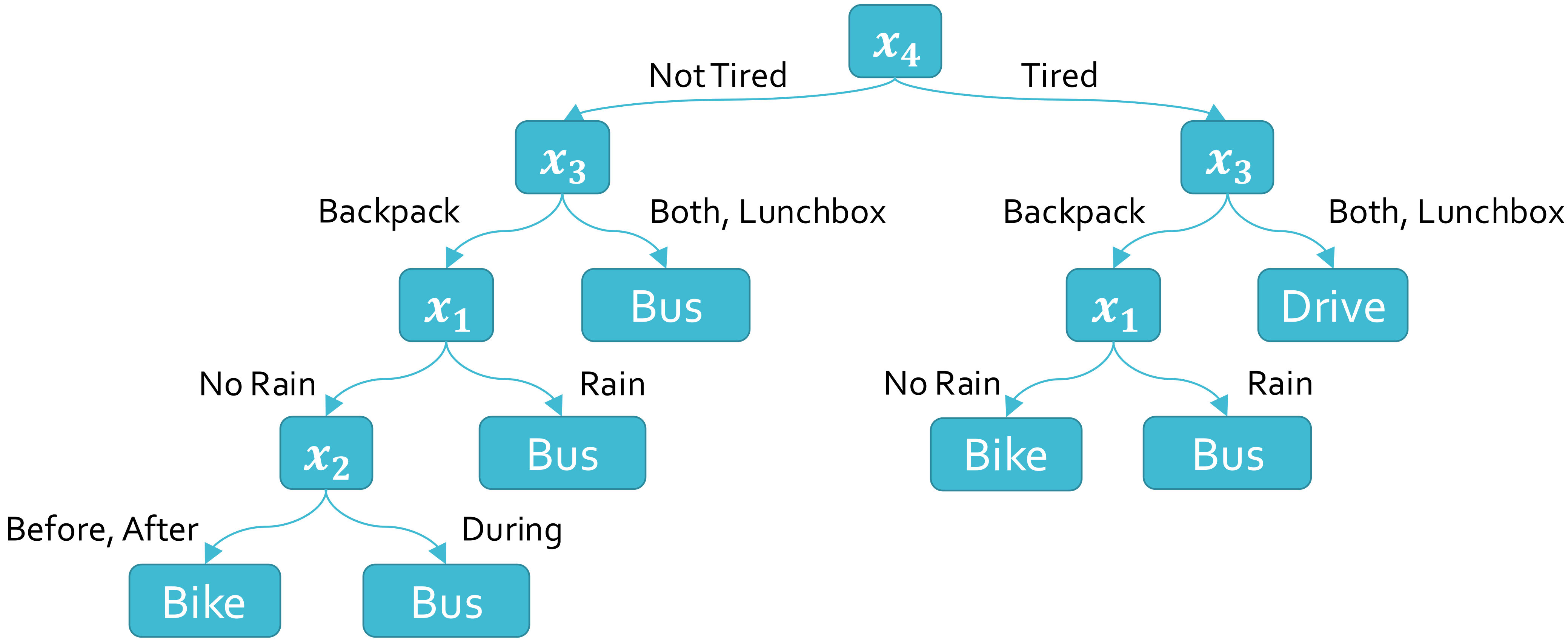
$$= -\sum_{y}\sum_{x} p(x,y) \log(p(y)) + \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)}$$

$$= -\sum_{x,y} p(x,y) \left[\log p(y) - \log \frac{p(x,y)}{p(x)}\right]$$

$$= -\sum_{x,y} p(x,y) \log \frac{p(y)\,p(x)}{p(x,y)}$$

# How is Hoda 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 |


$x_4$
Not Tired
Tired
$x_3$
$x_3$
Backpack
Both, Lunchbox
Backpack
Both, Lunchbox
$x_1$
Bus
$x_1$
Drive
No Rain
Rain
No Rain
Rain
$x_2$
Bus
Bike
Bus
Before, After
During
Bike
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 Smallest / shallowest tree that achieves lowest training error with highest mutual information features at the top

# 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 |

$\rightarrow$

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

$\leftarrow$ $x < 38.5$

# 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 |

$\rightarrow$

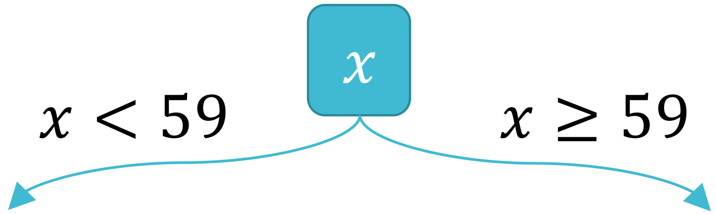
| $x$ | $y$ |
|---|---|
| 33 | Drive |
| 44 | Metro |
| 45 | Metro |
| 51 | Metro |
| 55 | Metro |
| 63 | Bike |
| 74 | Drive |
| 78 | Drive |
| 80 | Drive |
| 81 | Drive |

$\leftarrow$ $x < 44.5$

# 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 |

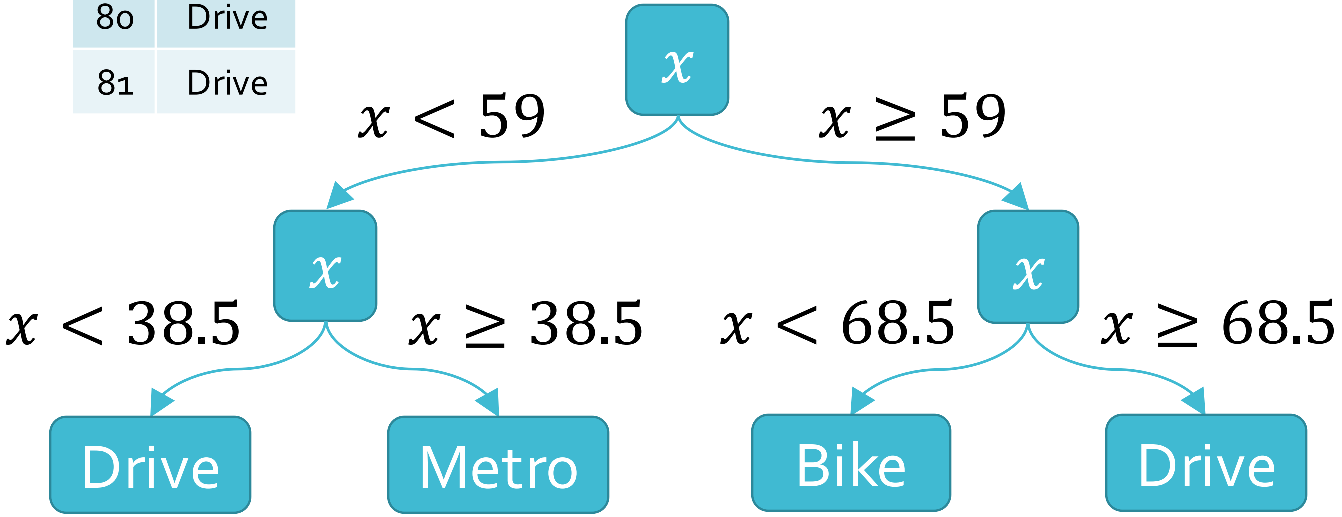
| $x$ | $y$ |
|---|---|
| 33 | Drive |
| 44 | Metro |
| 45 | Metro |
| 51 | Metro |
| 55 | Metro |
| 63 | Bike |
| 74 | Drive |
| 78 | Drive |
| 80 | Drive |
| 81 | Drive |



# 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!

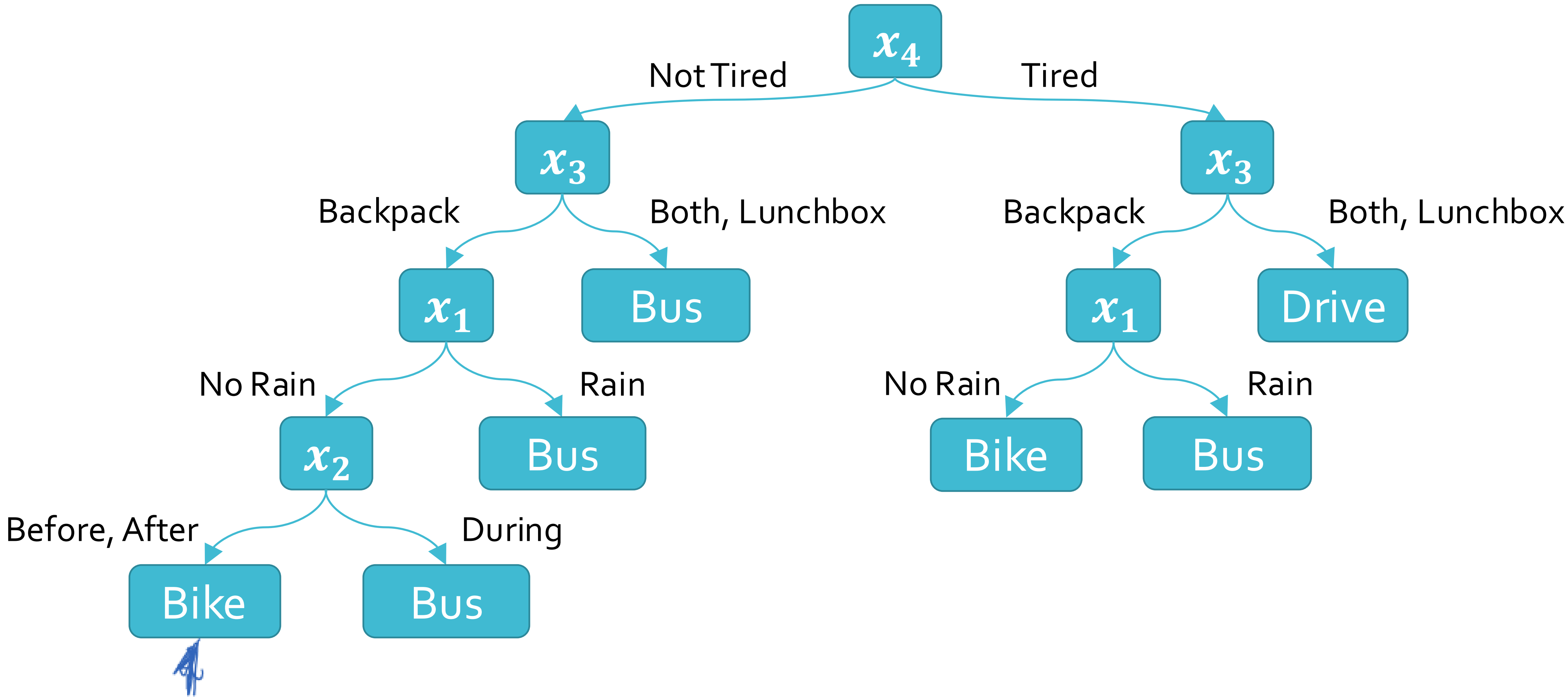
# Overfitting Underfitting

- 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)

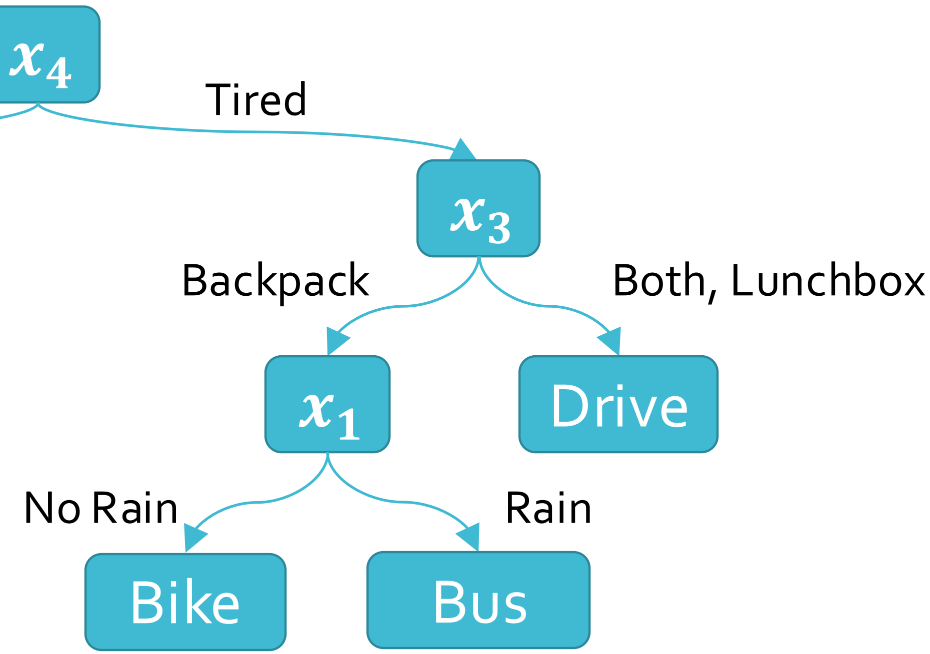
# Different Kinds of Error

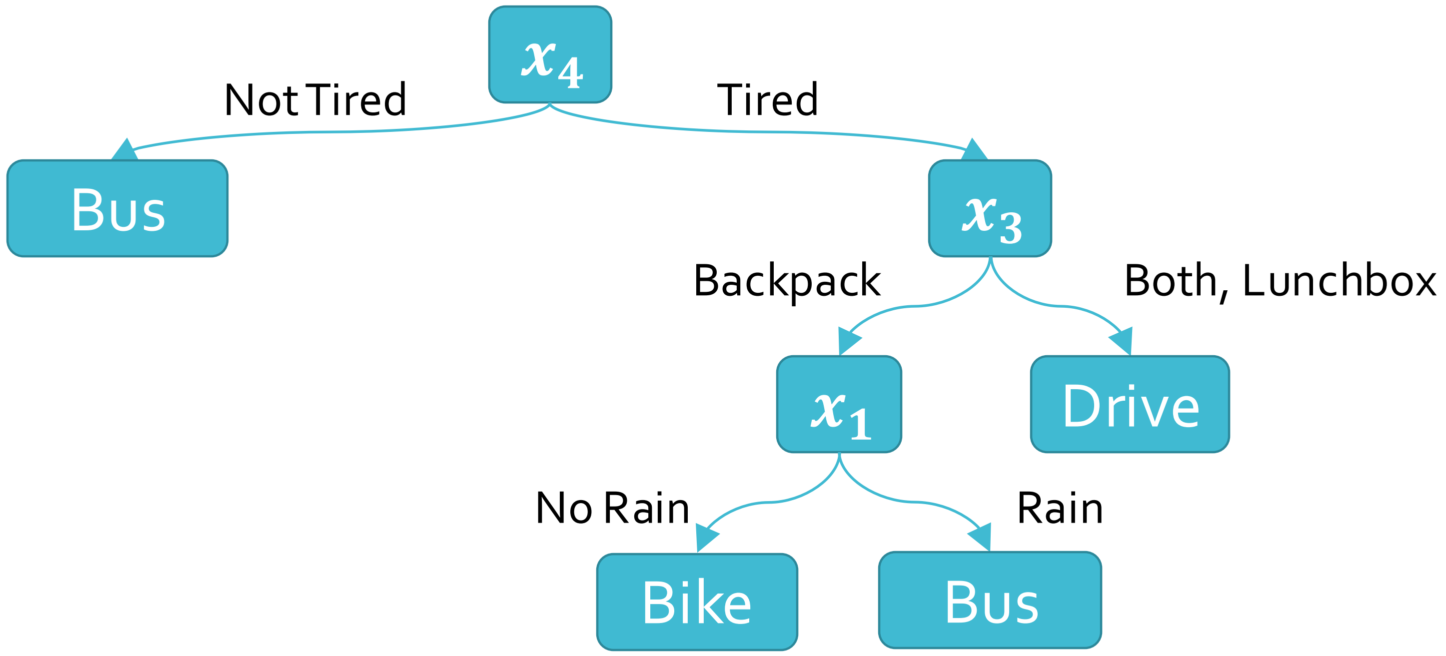
- Training error rate = $err(h, \mathcal{D}_{train})$
- Test error rate = $err(h, \mathcal{D}_{test})$
- True error rate = $err(h)$

  = 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 the measure of overfitting,
  - often estimated by $err_{test}(h) - err(h, \mathcal{D}_{train})$


$x_4$
Not Tired
Tired
$x_3$
$x_3$
Backpack
Both, Lunchbox
Backpack
Both, Lunchbox
$x_1$
Bus
$x_1$
Drive
No Rain
Rain
No Rain
Rain
$x_2$
Bus
Bike
Bus
Before, After
During
Bike
Bus
4

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






This tree only misclassifies one training data point!

# Overfitting in Decision Trees

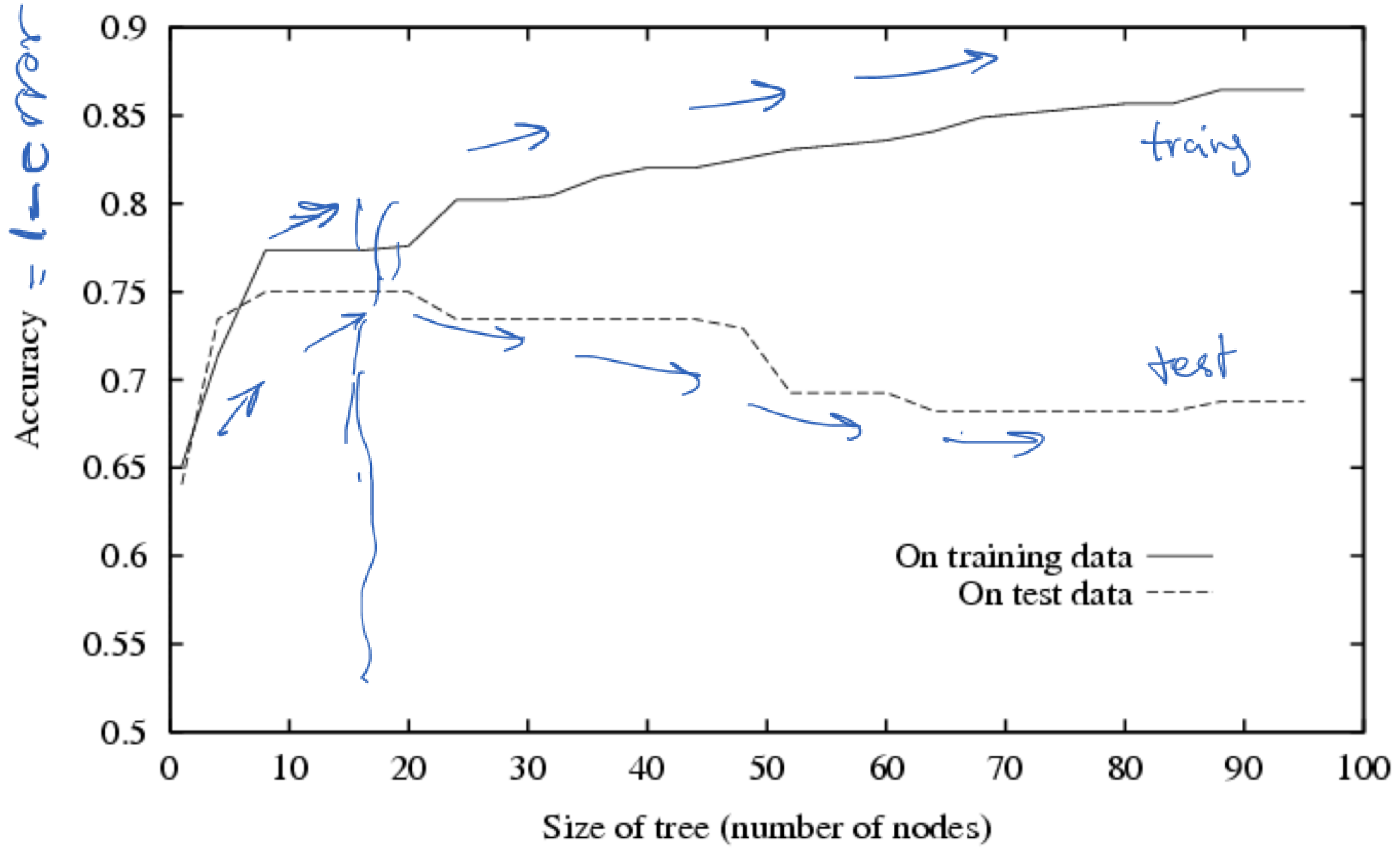




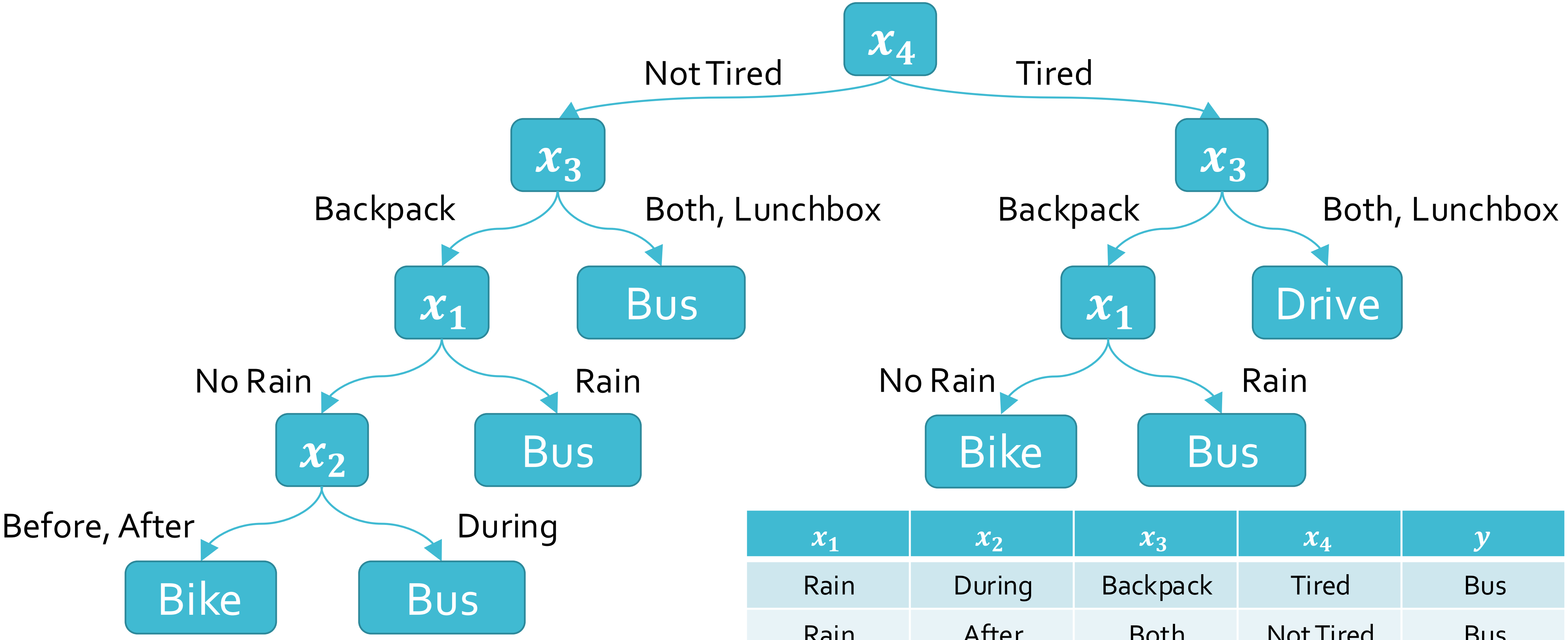

Figure courtesy of Tom Mitchell

# 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, $\delta$
  - Do not split leaves with fewer than $c$ data points
  - Do not split leaves where the maximal information gain is less than $\tau$
- Take a majority vote in impure leaves

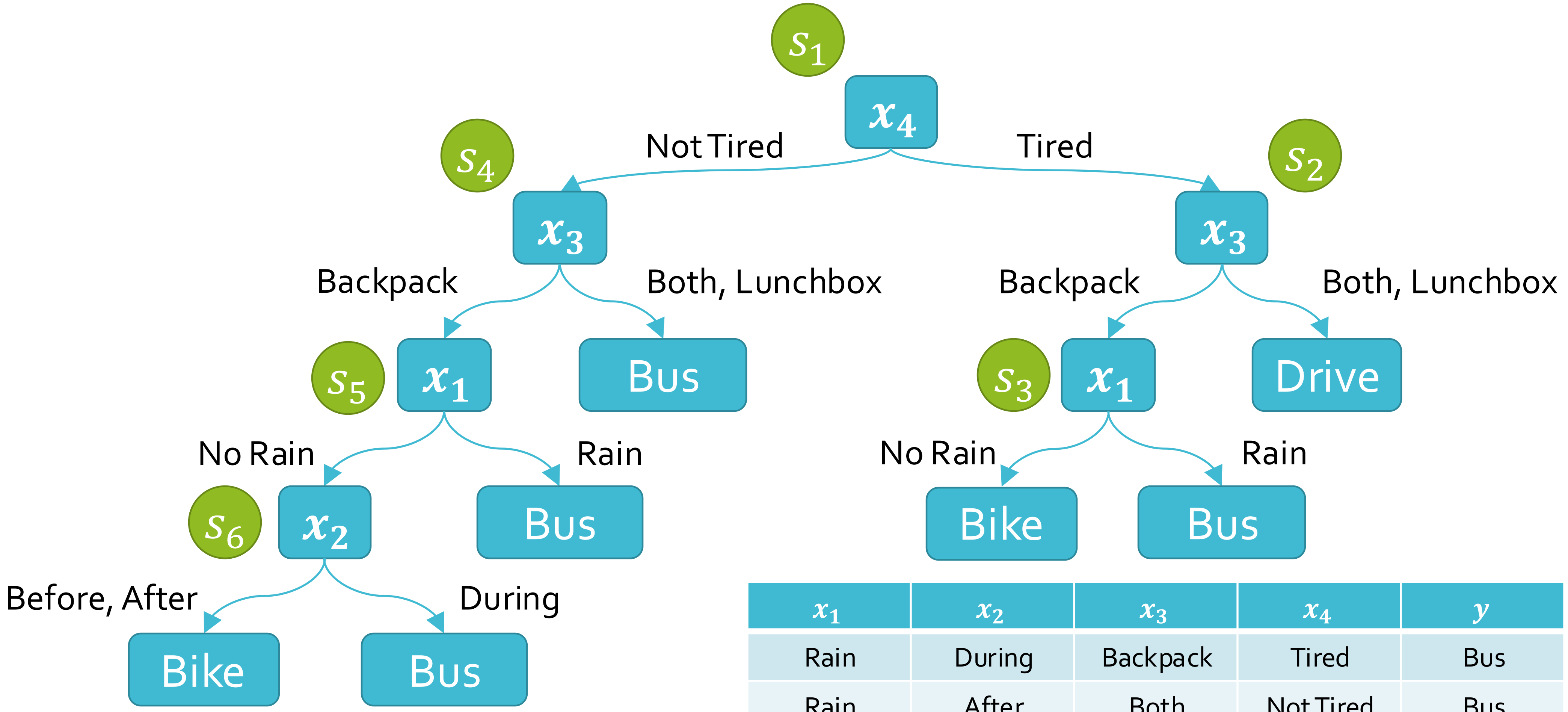
# Combatting Overfitting in Decision Trees

- 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




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

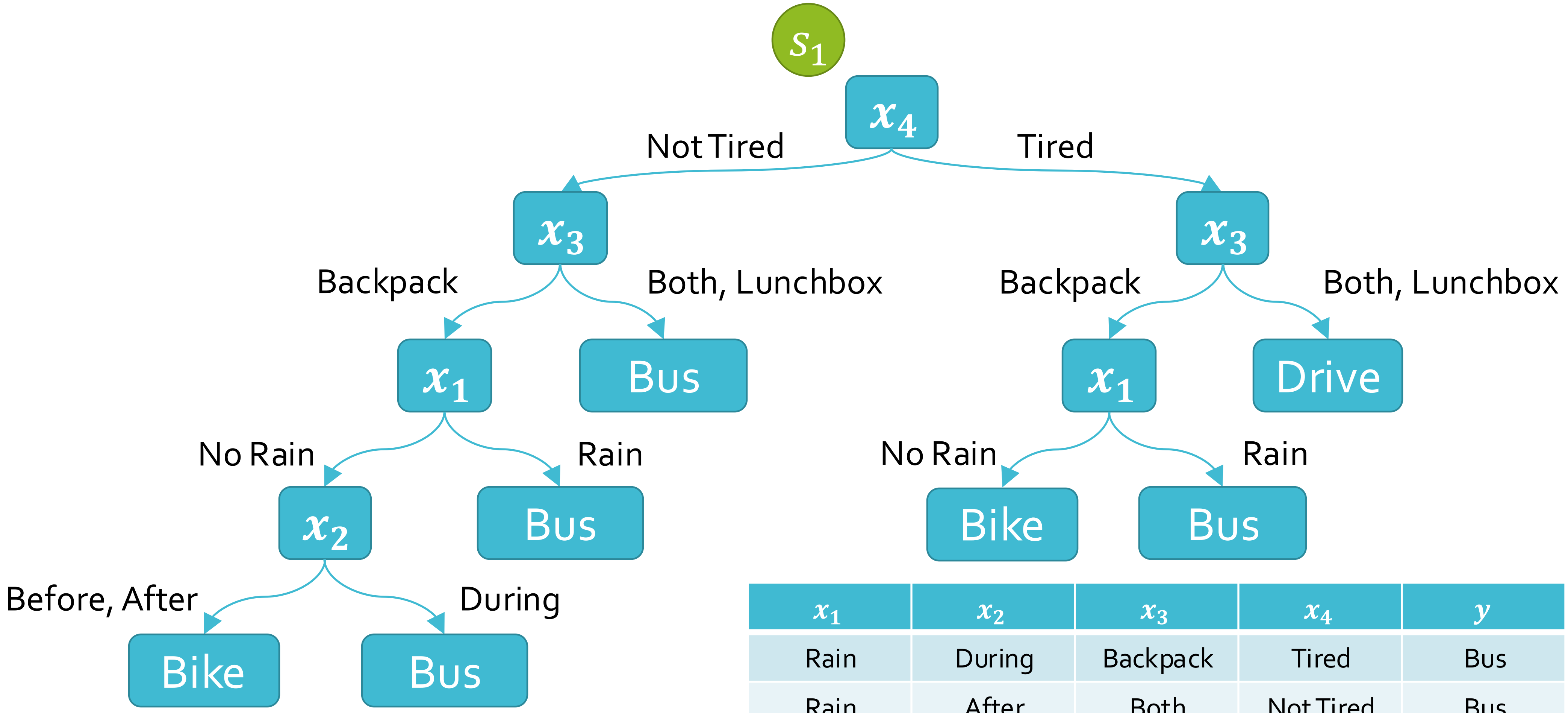
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| Rain | During | Backpack | Tired | Bus |
| Rain | After | Both | Not Tired | Bus |
| No Rain | Before | Backpack | Not Tired | Bus |
| No Rain | During | Lunchbox | Tired | Drive |
| No Rain | After | Lunchbox | Tired | Drive |




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

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




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

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

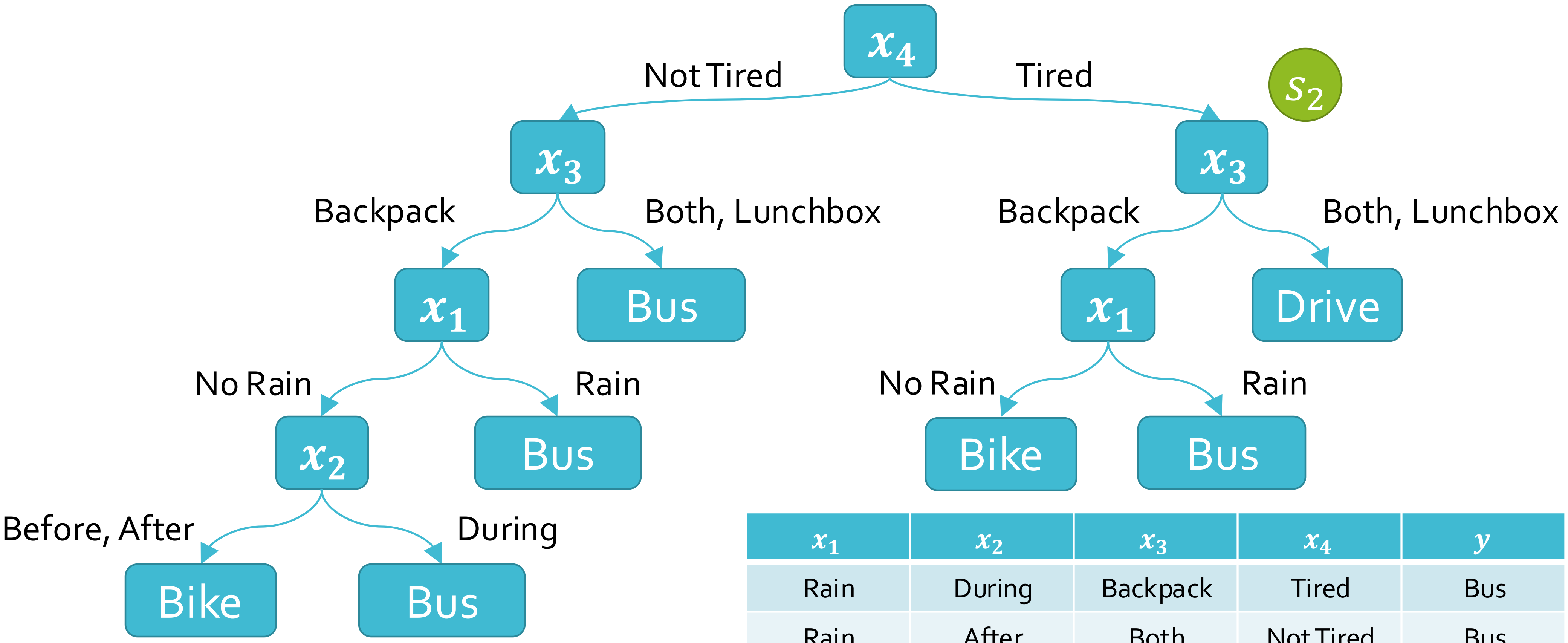
$s_1$

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 |

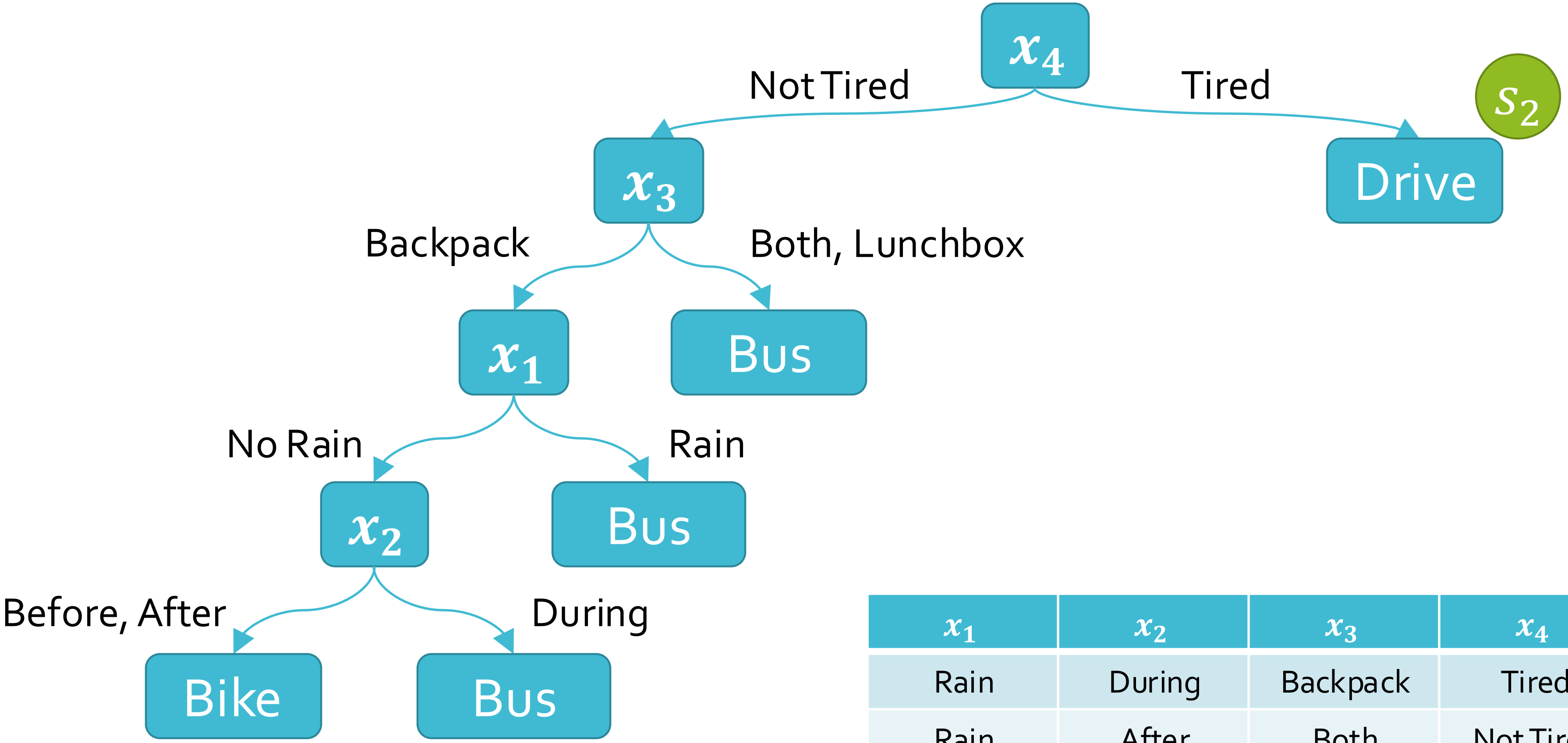
$$err(\underbrace{h - s_1}_{\text{decision tree } h \text{ minus split at } s_1}, \mathcal{D}_{val}) = 0.4$$




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

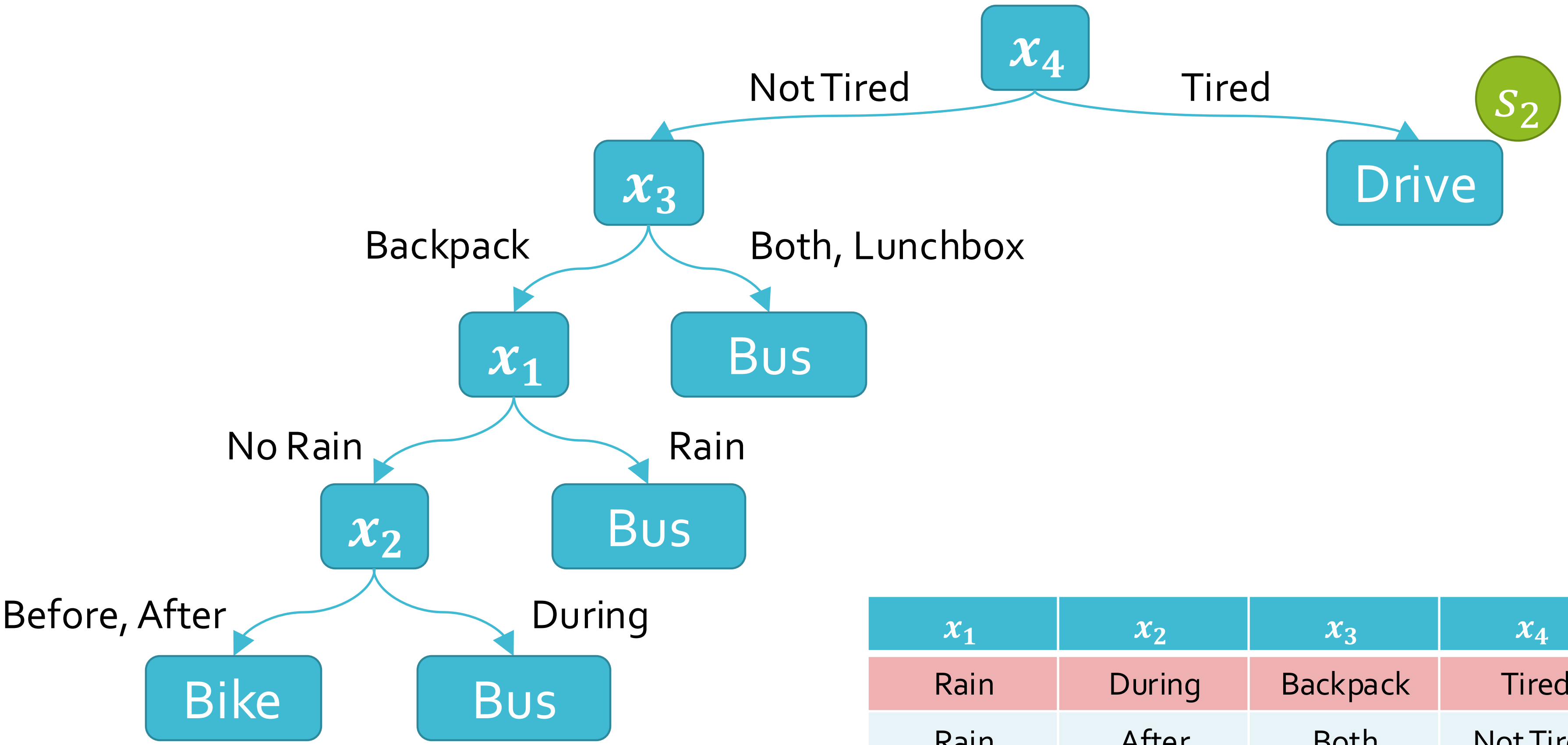
$err(h - s_2, \mathcal{D}_{val})$




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

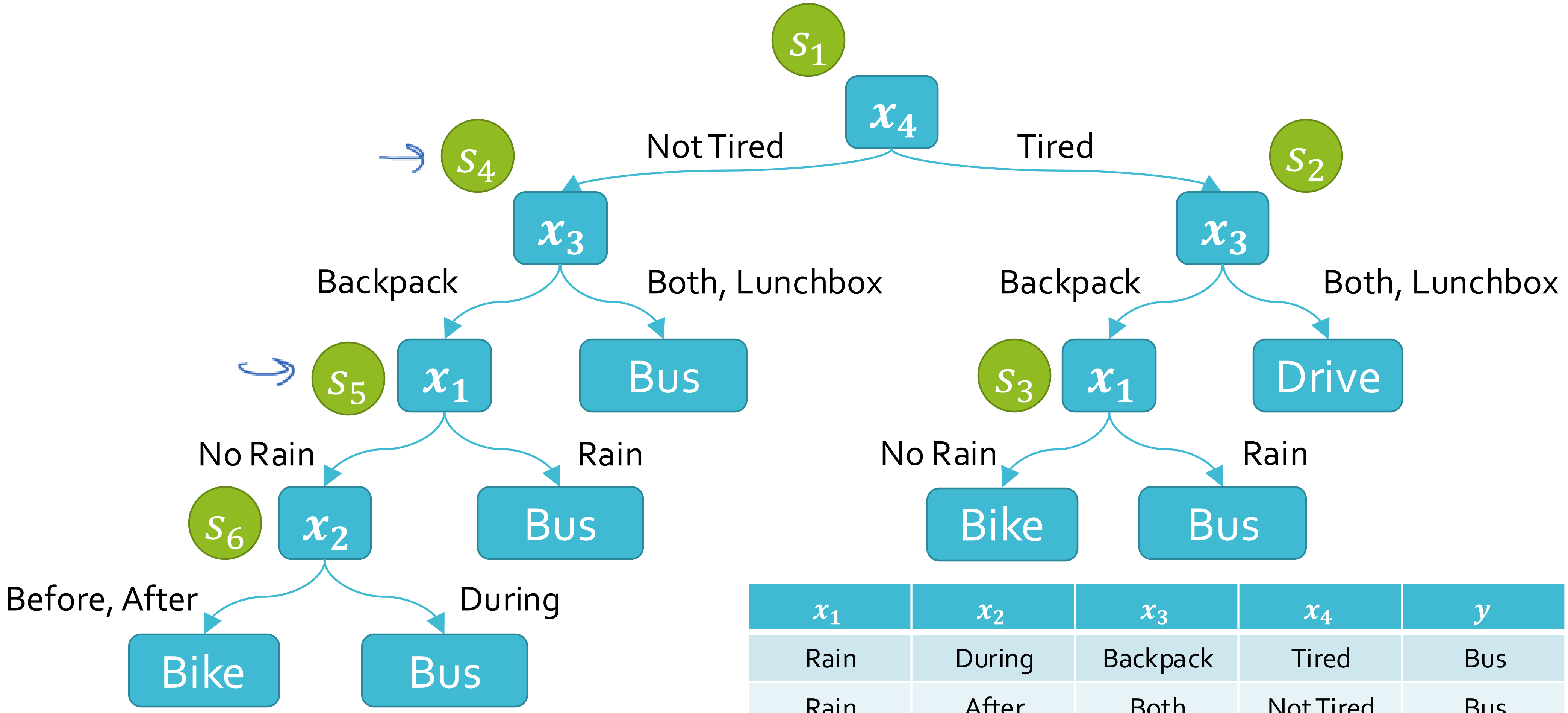
$err(h - s_2, \mathcal{D}_{val})$




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

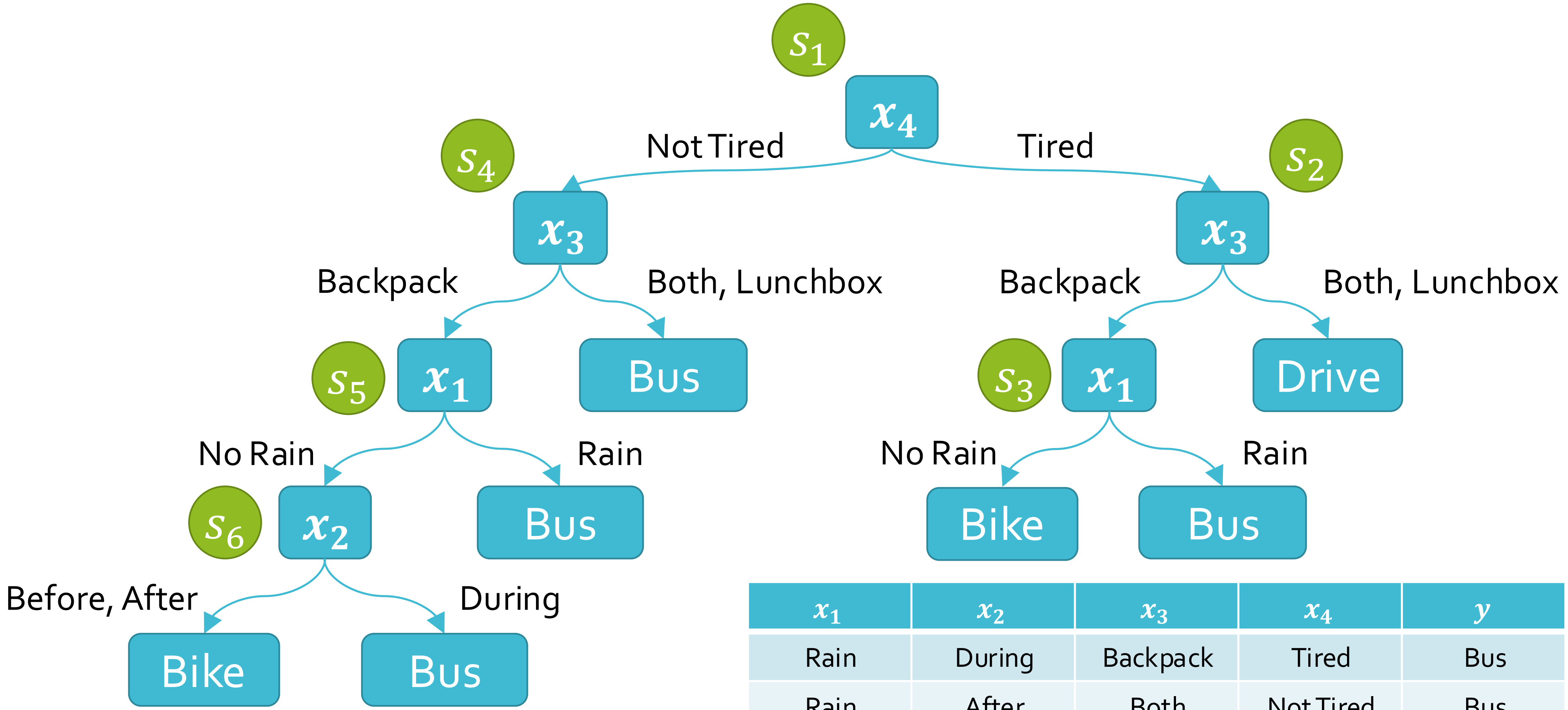
$$err(h - s_2, \mathcal{D}_{val}) = 0.4$$




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

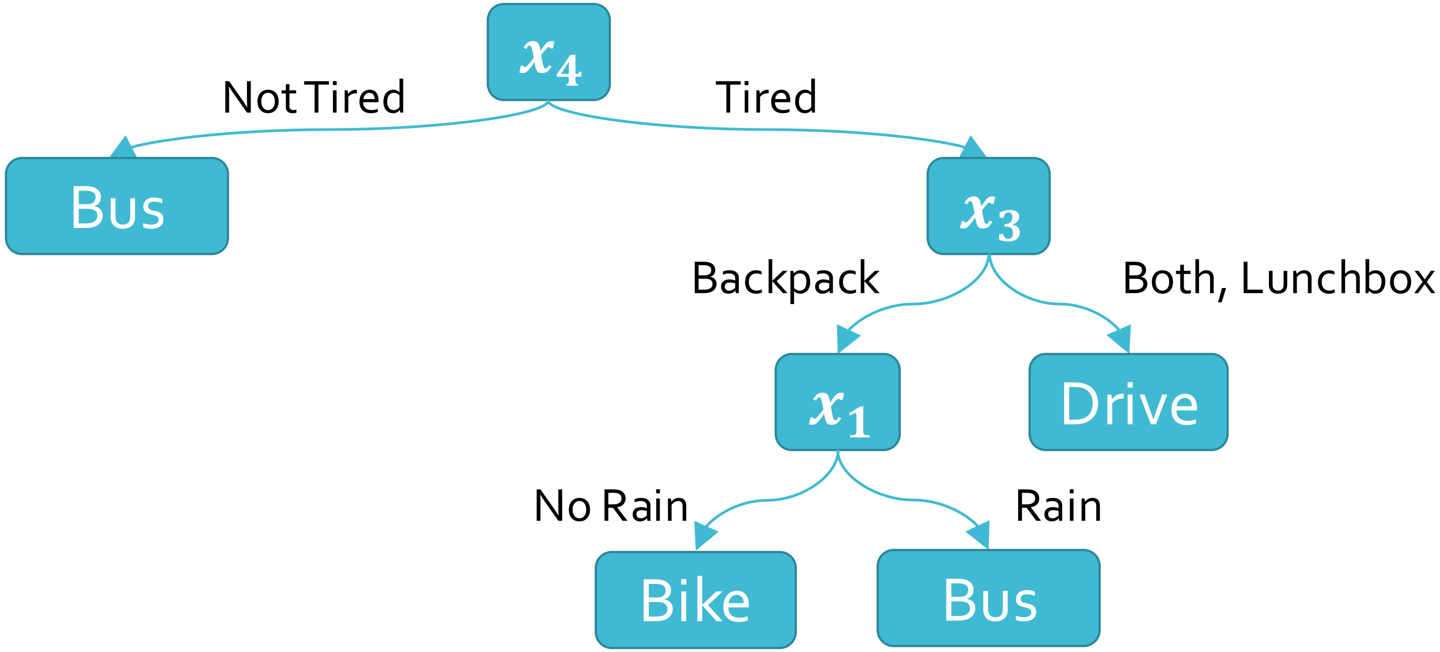
| $s$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
|---|---|---|---|---|---|---|
| $err(h - s, \mathcal{D}_{val})$ | 0.4 | 0.4 | 0.4 | 0 | 0 | 0.2 |




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

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

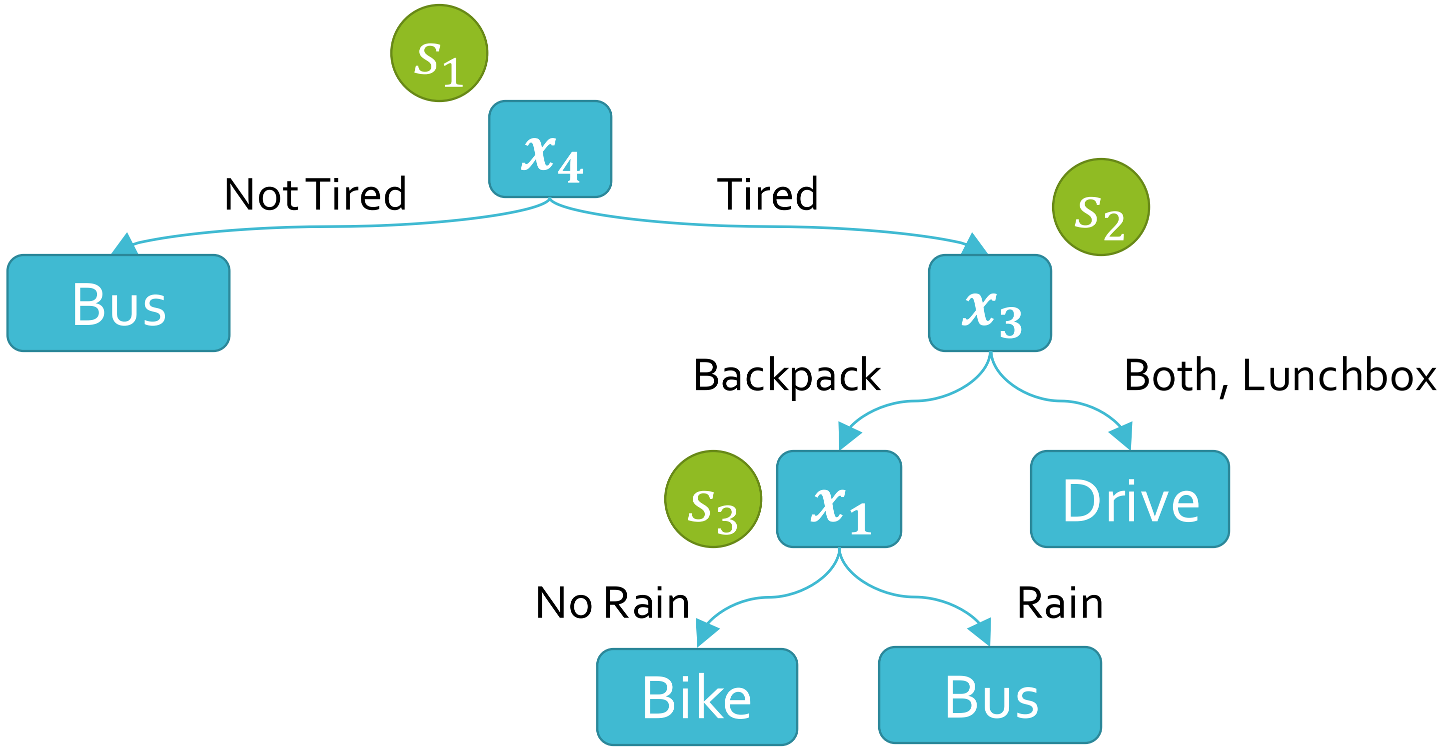
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| Rain | During | Backpack | Tired | Bus |
| Rain | After | Both | Not Tired | Bus |
| No Rain | Before | Backpack | Not Tired | Bus |
| No Rain | During | Lunchbox | Tired | Drive |
| No Rain | After | Lunchbox | Tired | Drive |




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

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 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 |

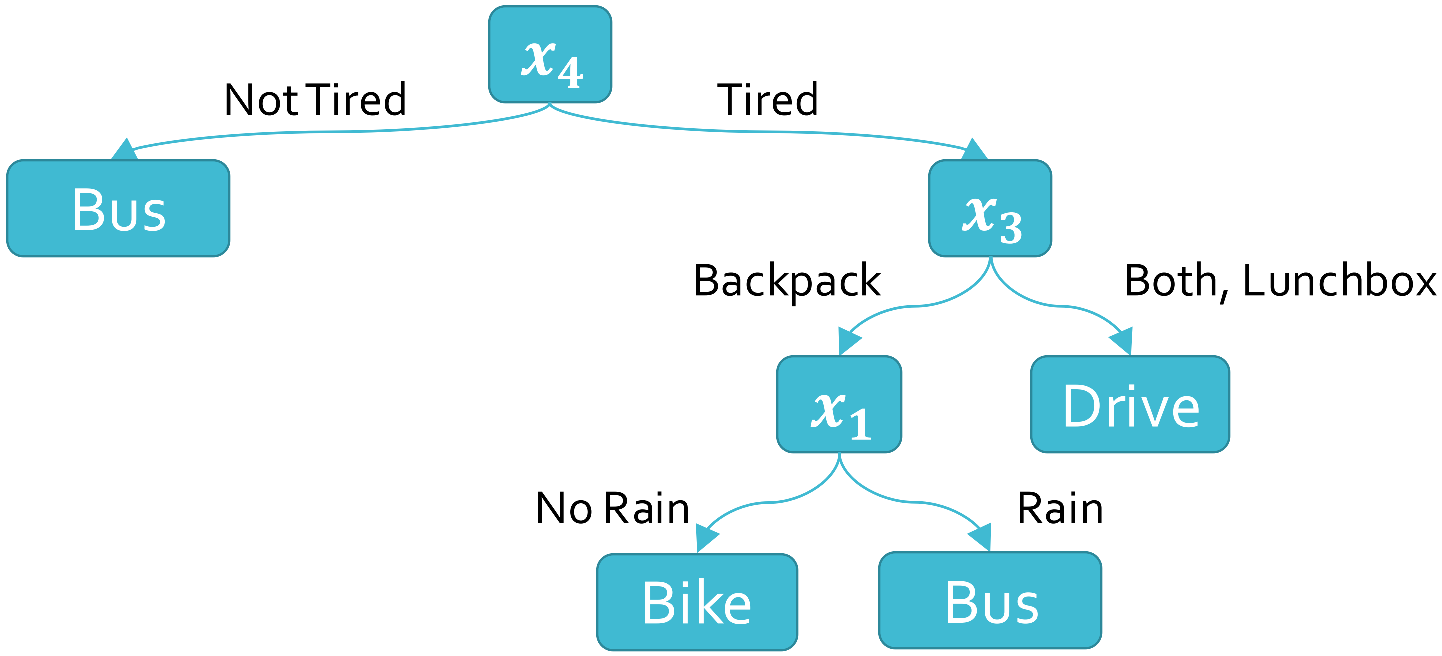
$err(h, \mathcal{D}_{val}) = 0$




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


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

# Pruning Decision Trees

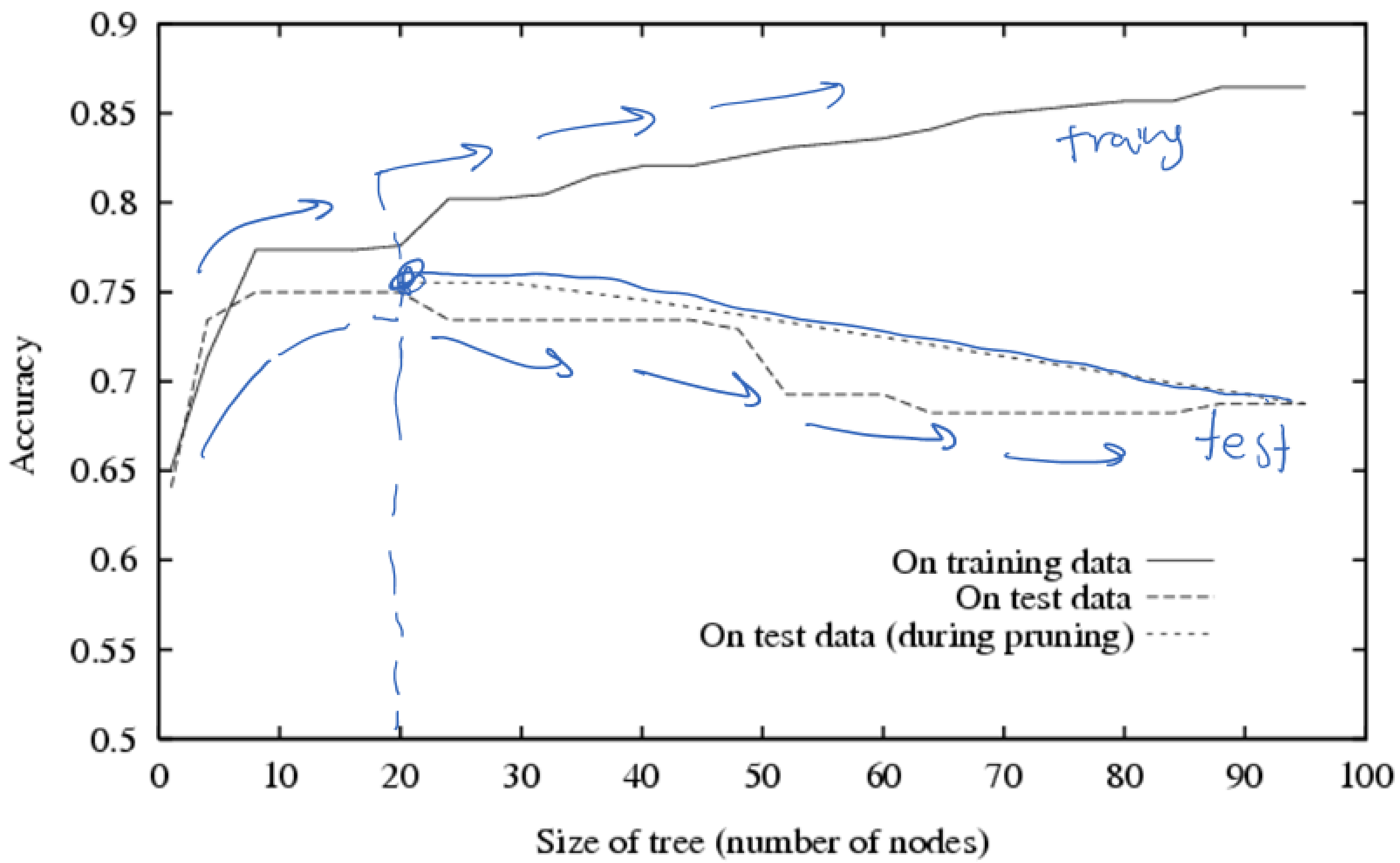
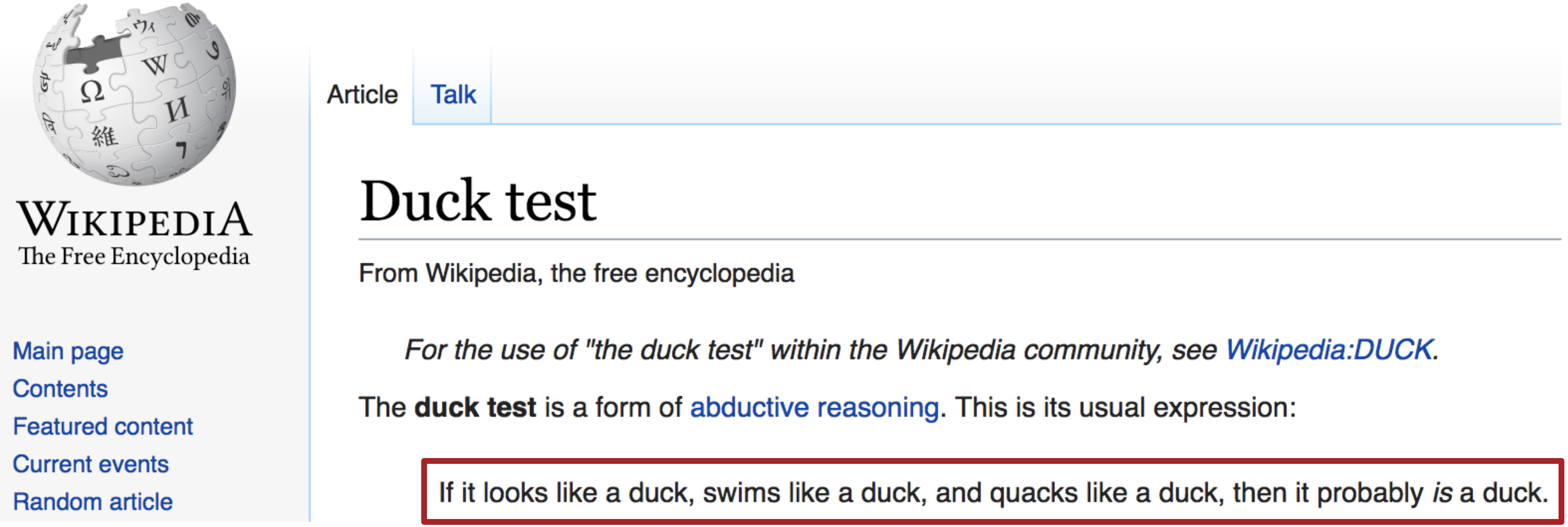



Figure courtesy of Tom Mitchell

# 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

# Class Activity


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

# 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 |

Source: 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 |

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

# Fisher Iris Dataset

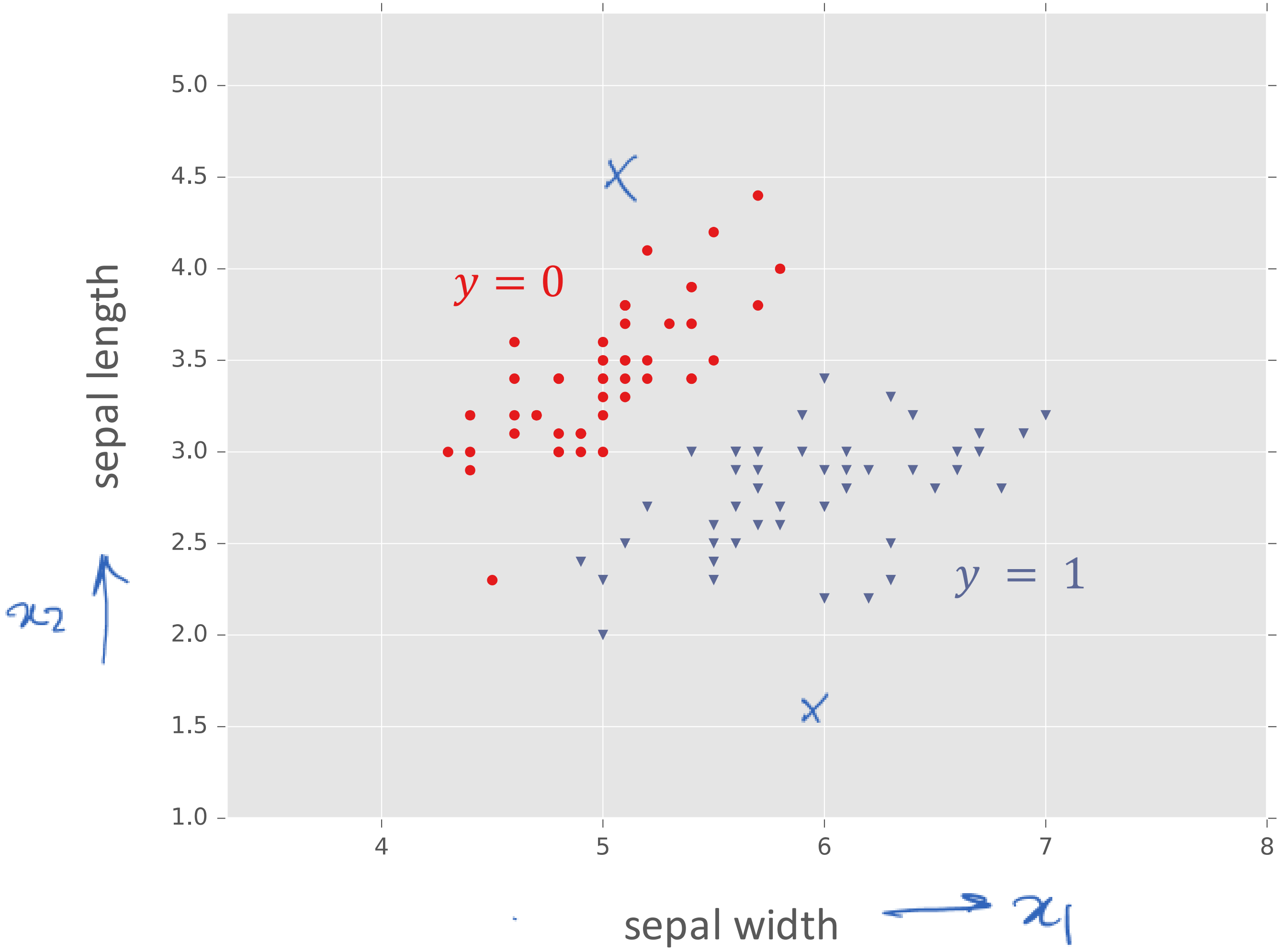





Figure courtesy of Matt Gormley

# 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:

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

- An alternative is the Manhattan distance:

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

# Nearest Neighbor Model

- Classify a point as the label of the "most similar" training point

- Given a training dataset $\mathcal{D}_{train} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$

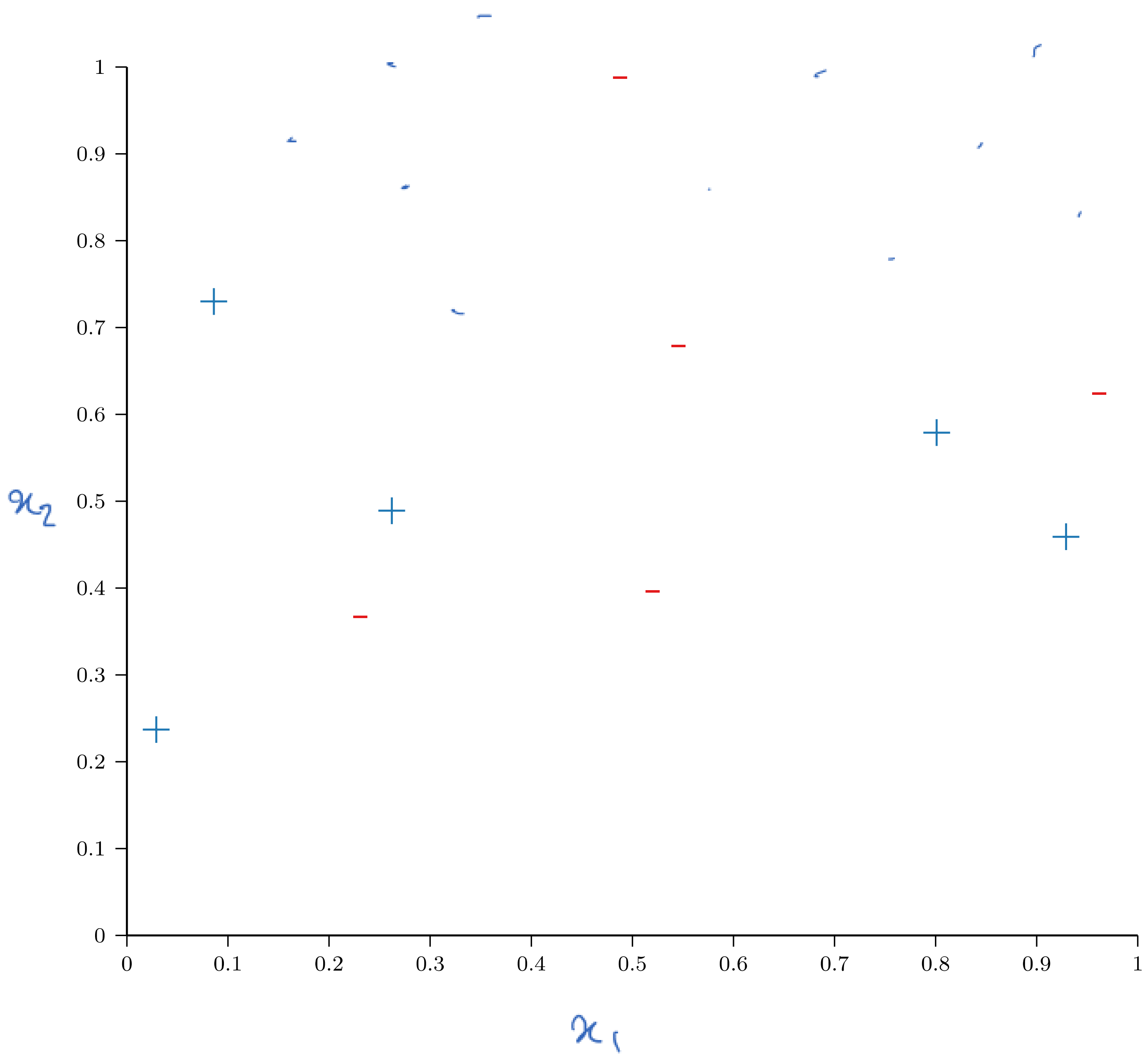
$$\text{Let } \hat{\imath}(\boldsymbol{x}') = \underset{i \in \{1,\ldots,N\}}{\operatorname{argmin}} \; d\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}'\right)$$

new instance

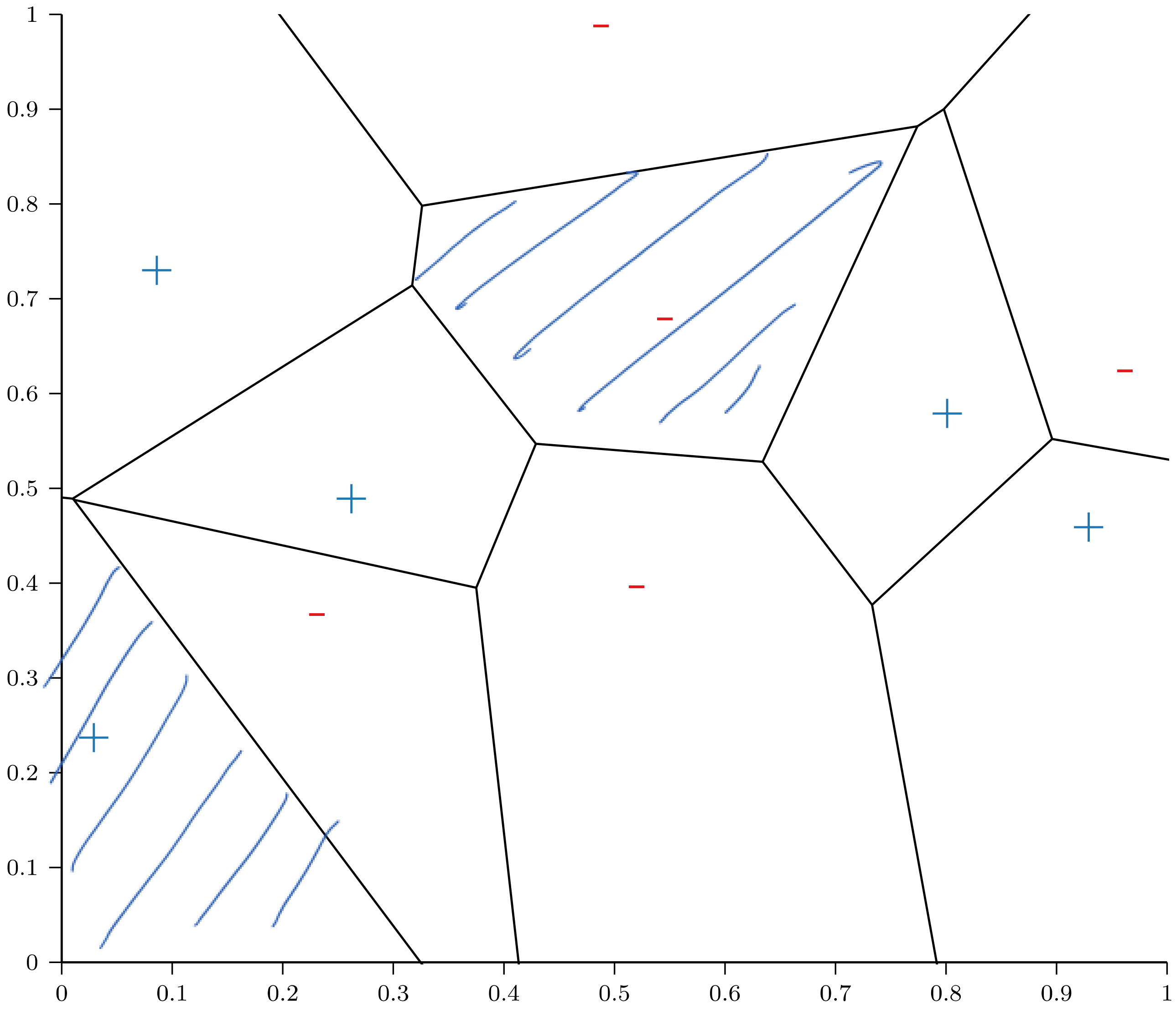
- Then the nearest neighbor classifier can be written as

$$h(\boldsymbol{x}') = y^{\left(\hat{\imath}(\boldsymbol{x}')\right)}$$

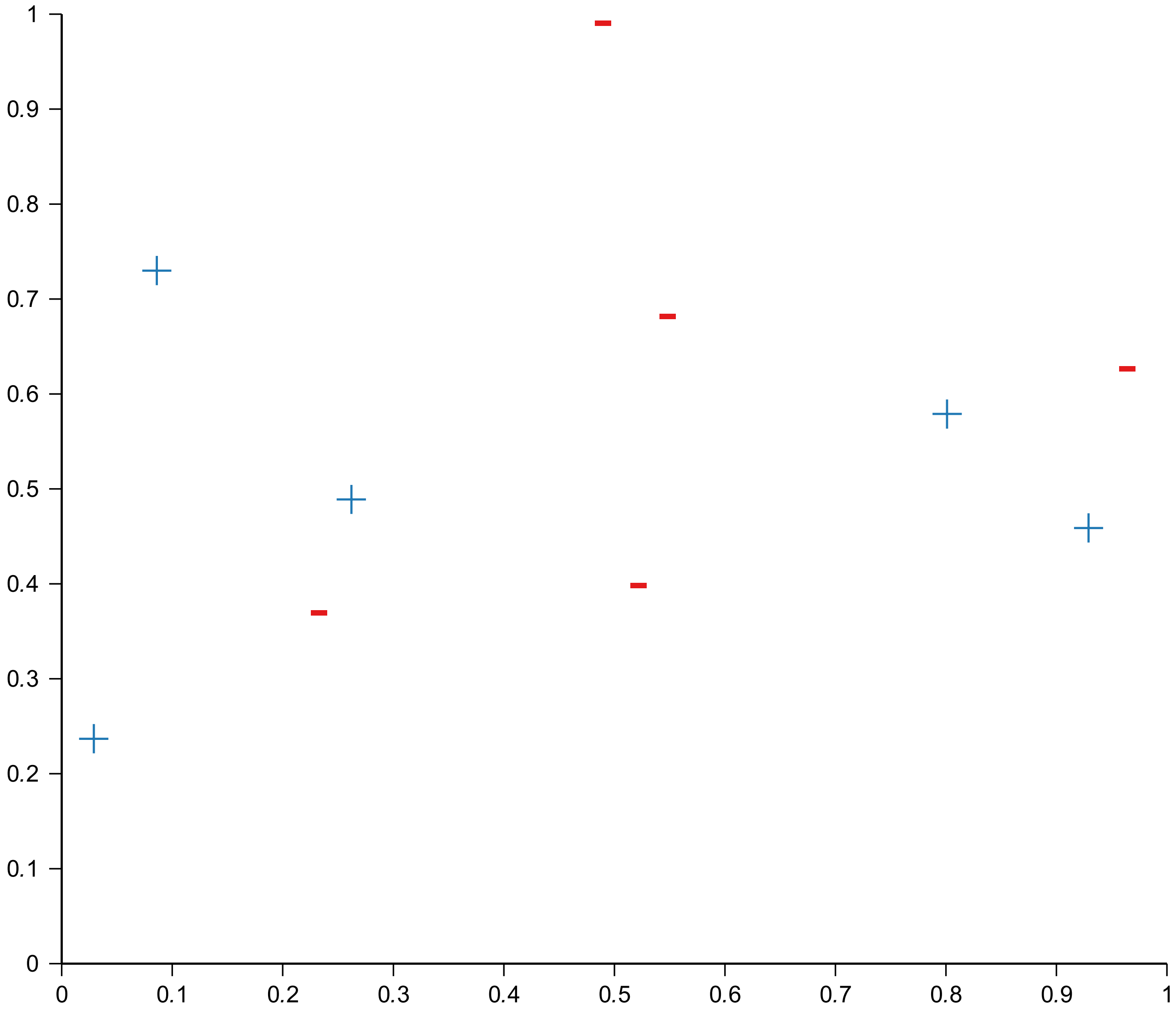
# Nearest Neighbor: Example



# 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…

# Bayes Optimal Classifier

- Assume a binary classification problem: $\mathcal{Y} = \{1, 0\}$
- Assume data points are drawn *independently* from some probability distribution $P$ defined over $\mathcal{X} \times \mathcal{Y}$ $P(x, y)$
- Assume labels are *stochastic*: let $\pi(\boldsymbol{x}) = P\{y = 1|\boldsymbol{x}\}$.

  $\pi(\vec{x}) = 0.6$
- What is the optimal prediction for $\boldsymbol{x}$ knowing $\pi(\boldsymbol{x})$?

  $\pi(x) = 0.65 \rightarrow \hat{y} = 1$

  $\pi(x') = 0.2 \longrightarrow \hat{y} = 0$
- Assume $\pi(\boldsymbol{x})$ is continuous.
- As $N \to \infty$, $\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)} \to \boldsymbol{x}' \Longrightarrow \pi\left(\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)}\right) \to \pi(\boldsymbol{x}')$

# 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 $h$ model $\leq 2 \ * $ the Bayes error rate (the optimal classifier)
- Proof: $\lim_{N \to \infty} err(h) = \lim_{N \to \infty} \mathbb{E}_{x' \sim P}\left[\mathbb{1}(h(x') \neq y)\right]$

$$= \mathbb{E}_{x' \sim P}\left[(h(x')=1 \wedge y'=0) \vee (h(x')=0 \wedge y'=1)\right]$$

$$= \mathbb{P}_{x' \sim P}\left[(h(x')=1 \wedge y'=0)\right] + \mathbb{P}\left[(h(x')=0 \wedge y'=1)\right]$$

$$\pi(x^{i(x')}) \leftarrow y^{i(x')} = 1$$

$$\pi(x^{i(x')})\,(1-\pi(x')) \qquad (1-\pi(x^{i(x')}))\,\pi(x')$$

$$= \pi(x')(1-\pi(x')) + \pi(x')(1-\pi(x')) = 2\pi(x')(1-\pi(x'))$$

$$\leq 2\min(\pi(x'), 1-\pi(x'))$$

error of BoC



Source: [https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964)

# Generalization of Nearest Neighbor (Cover and Hart, 1967)

- Claim: under certain conditions, as $n \to \infty$, with high probability, the true error rate of the nearest neighbor model $\leq 2 \ * $ the Bayes error rate (the optimal classifier)
- Proof (cont.):
- $err(h) = \mathbb{E}_{x'}[\mathbb{1}(h(\boldsymbol{x}') \neq y')] = P\{h(\boldsymbol{x}') \neq y'\}$

$$= P\{h(\boldsymbol{x}') = 1, y' = 0\} + P\{h(\boldsymbol{x}') = 0, y' = 1\}$$

$$= \pi\left(\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)}\right)\left(1 - \pi(\boldsymbol{x}')\right) + \left(1 - \pi\left(\boldsymbol{x}^{\left(\hat{\imath}(\boldsymbol{x}')\right)}\right)\right)\pi(\boldsymbol{x}')$$

$$\to \pi(\boldsymbol{x}')\left(1 - \pi(\boldsymbol{x}')\right) + \left(1 - \pi(\boldsymbol{x}')\right)\pi(\boldsymbol{x}')$$

$$= 2\pi(\boldsymbol{x}')\left(1 - \pi(\boldsymbol{x}')\right)$$

$$\leq 2\min\left(\pi(\boldsymbol{x}'), \left(1 - \pi(\boldsymbol{x}')\right)\right) = 2err(h^*) \ \blacksquare$$



Source: [https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964)

# $k$-Nearest Neighbors ($k$NN)

- Why limit ourselves to just one neighbor?
- 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

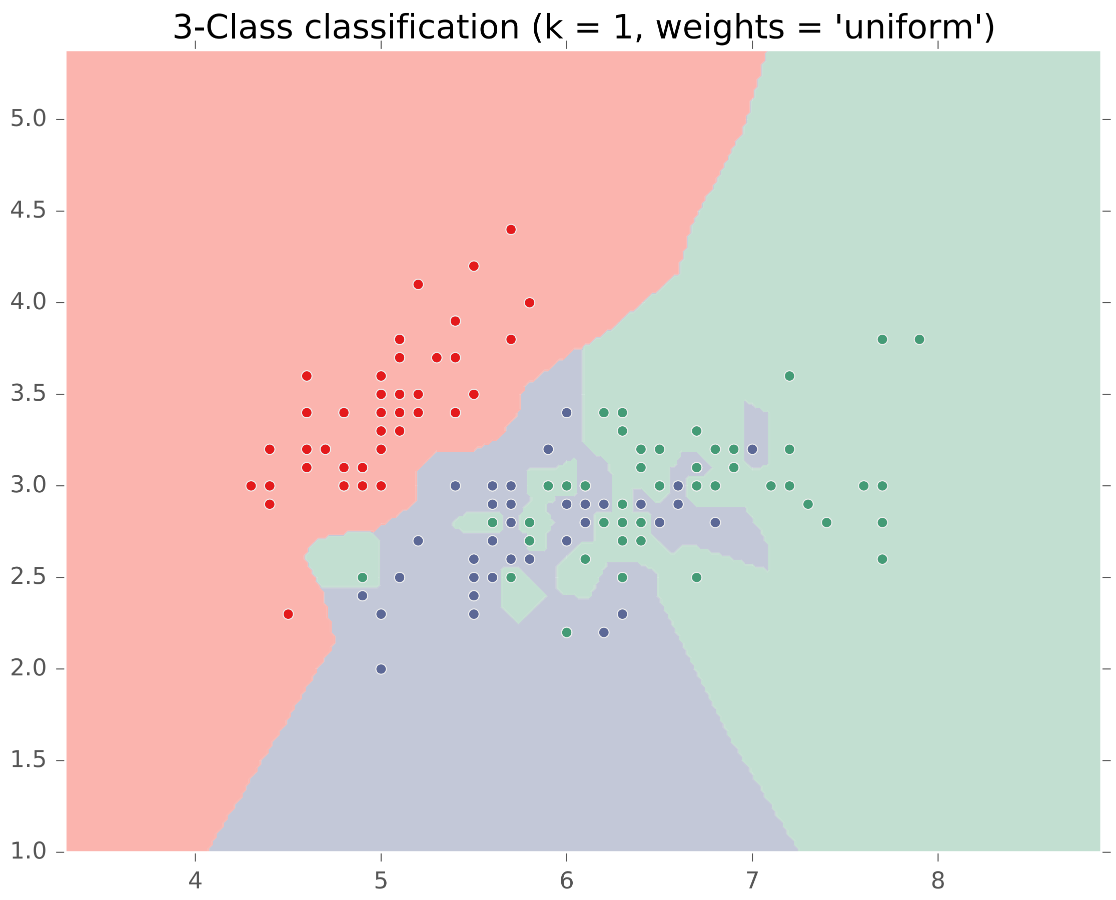



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

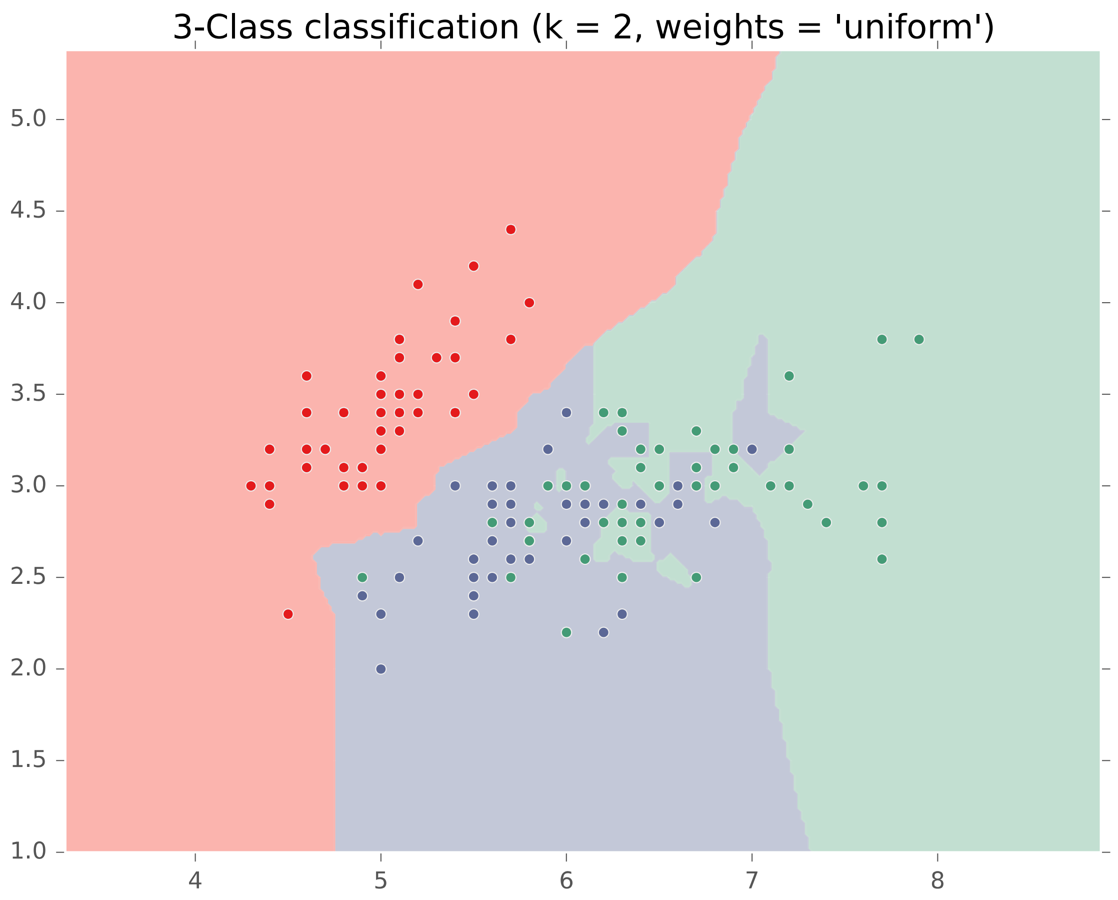



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

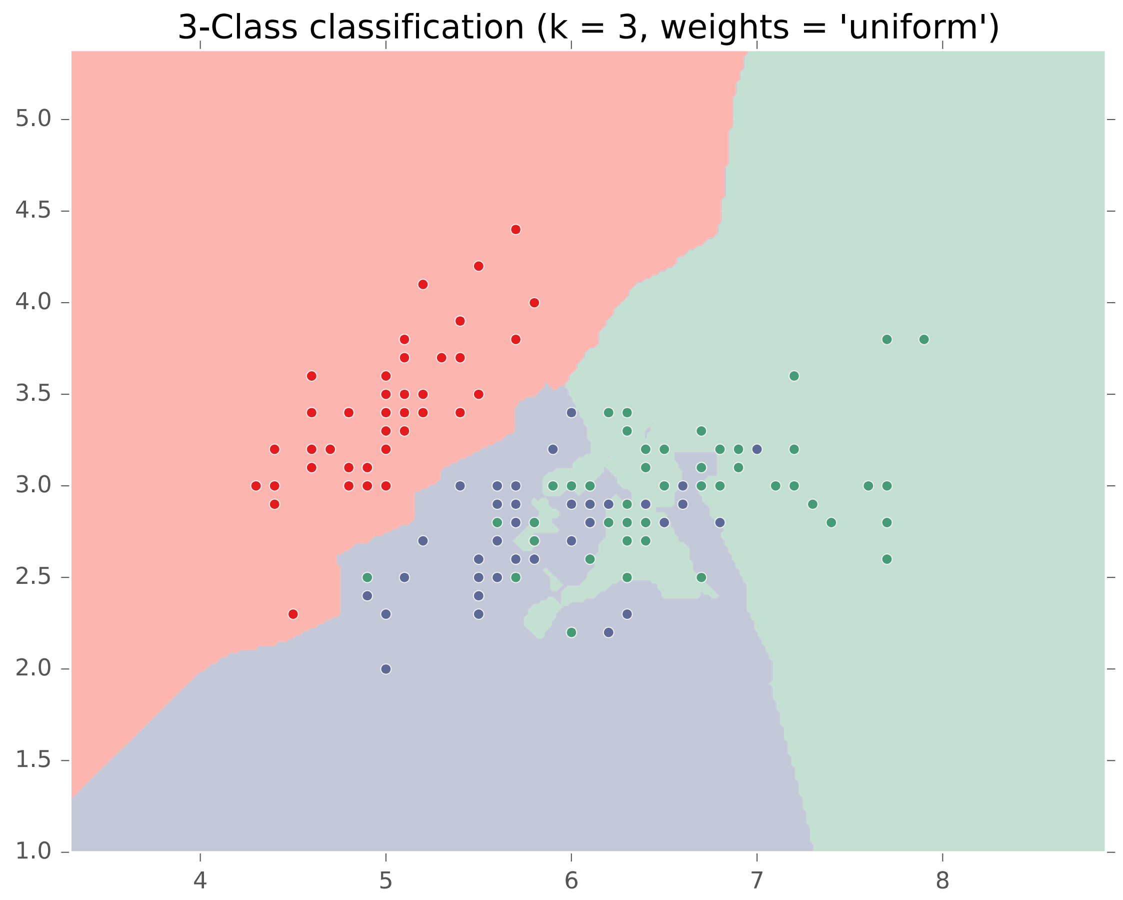





Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

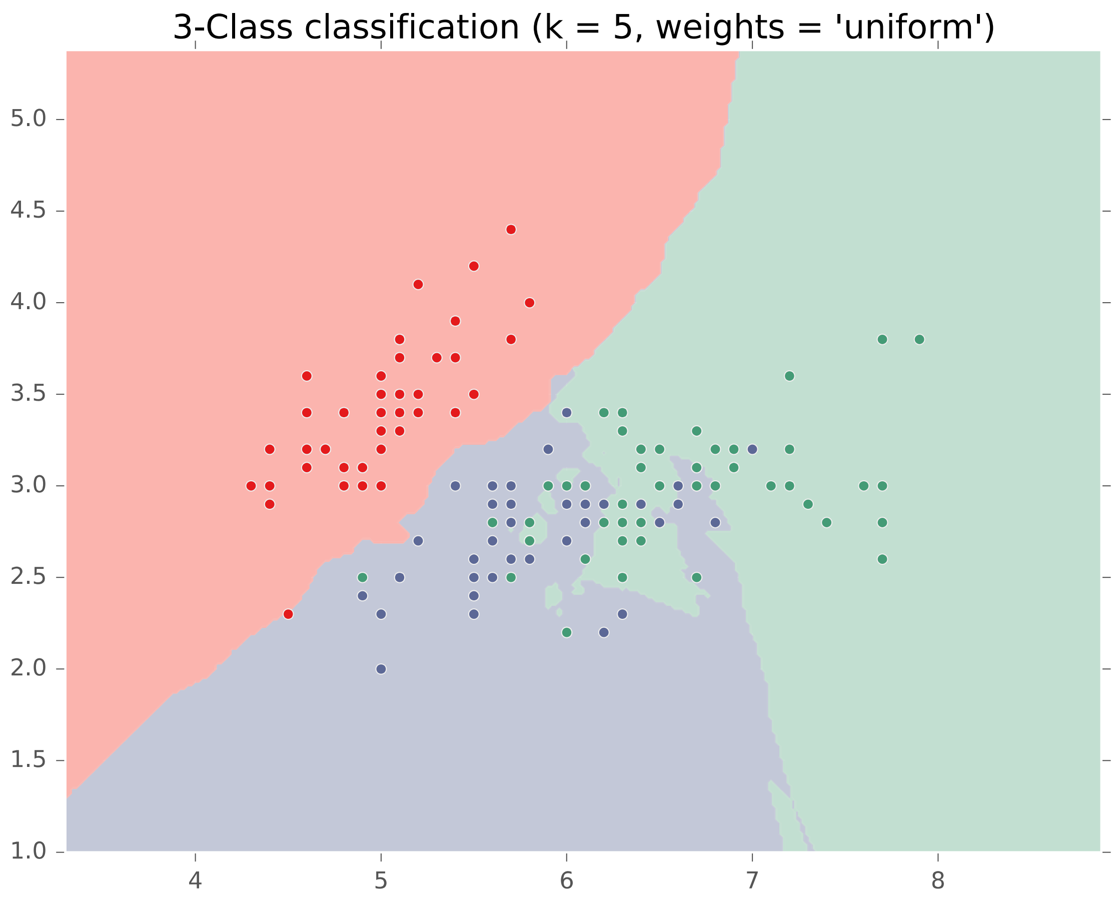



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

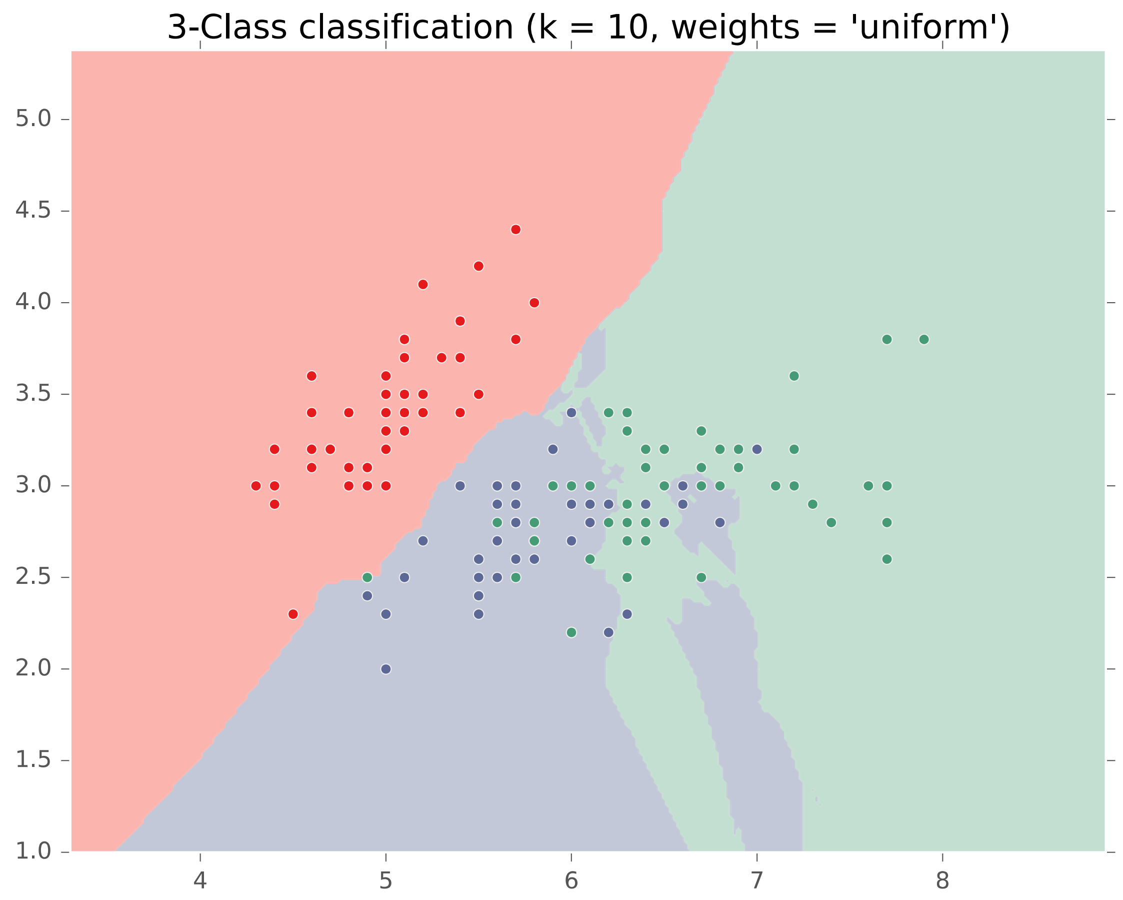



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

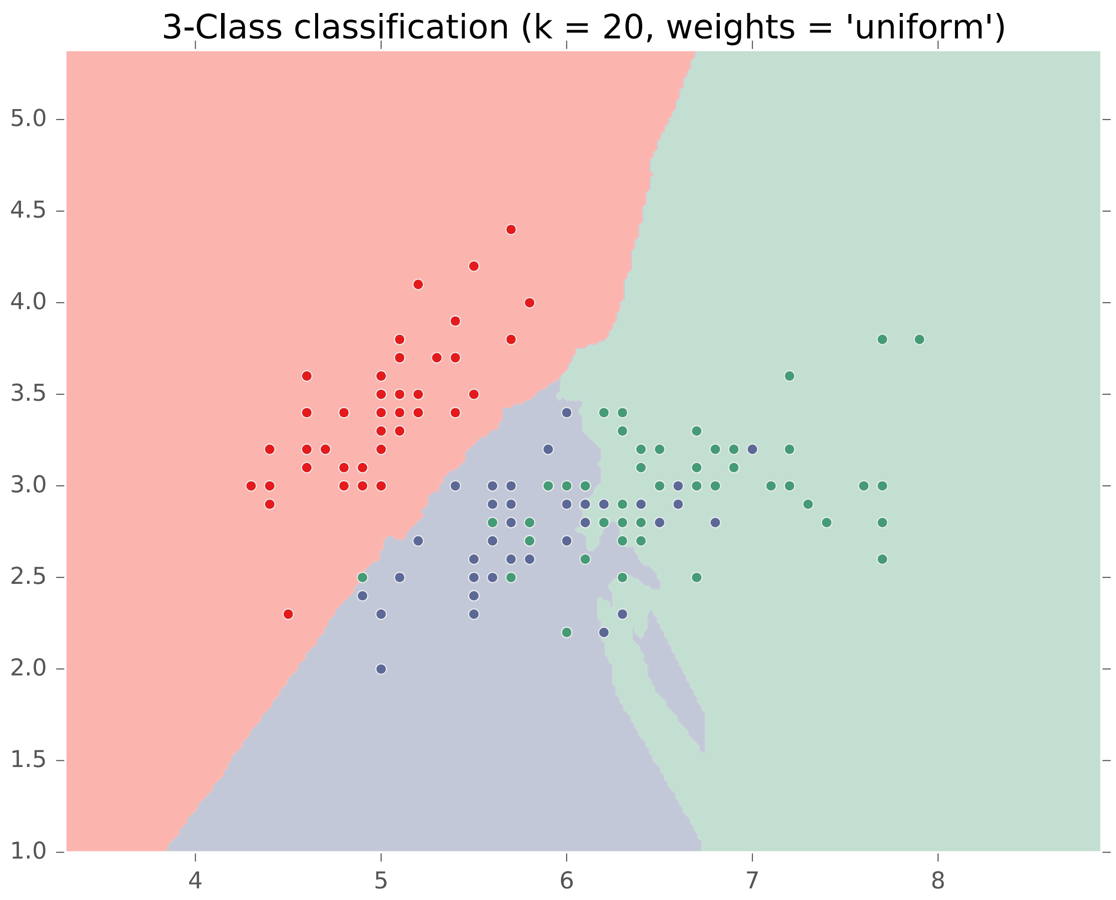





Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

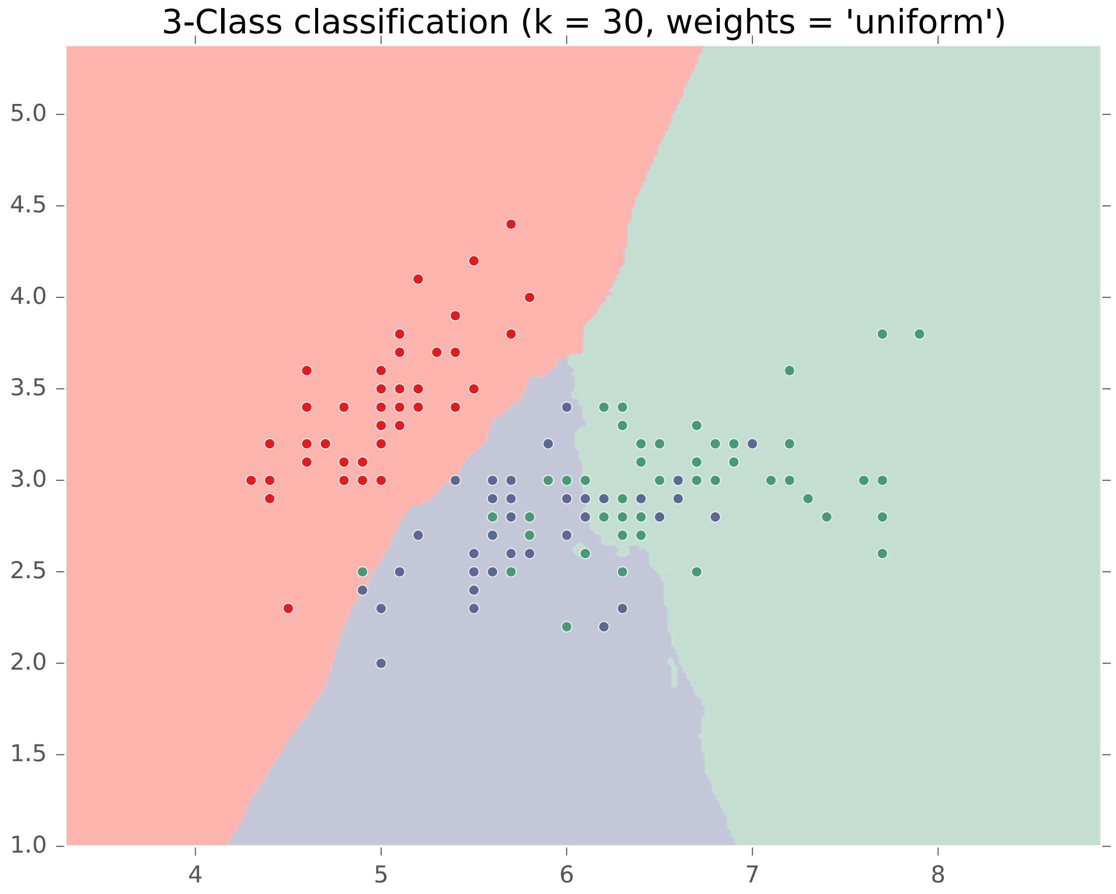



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

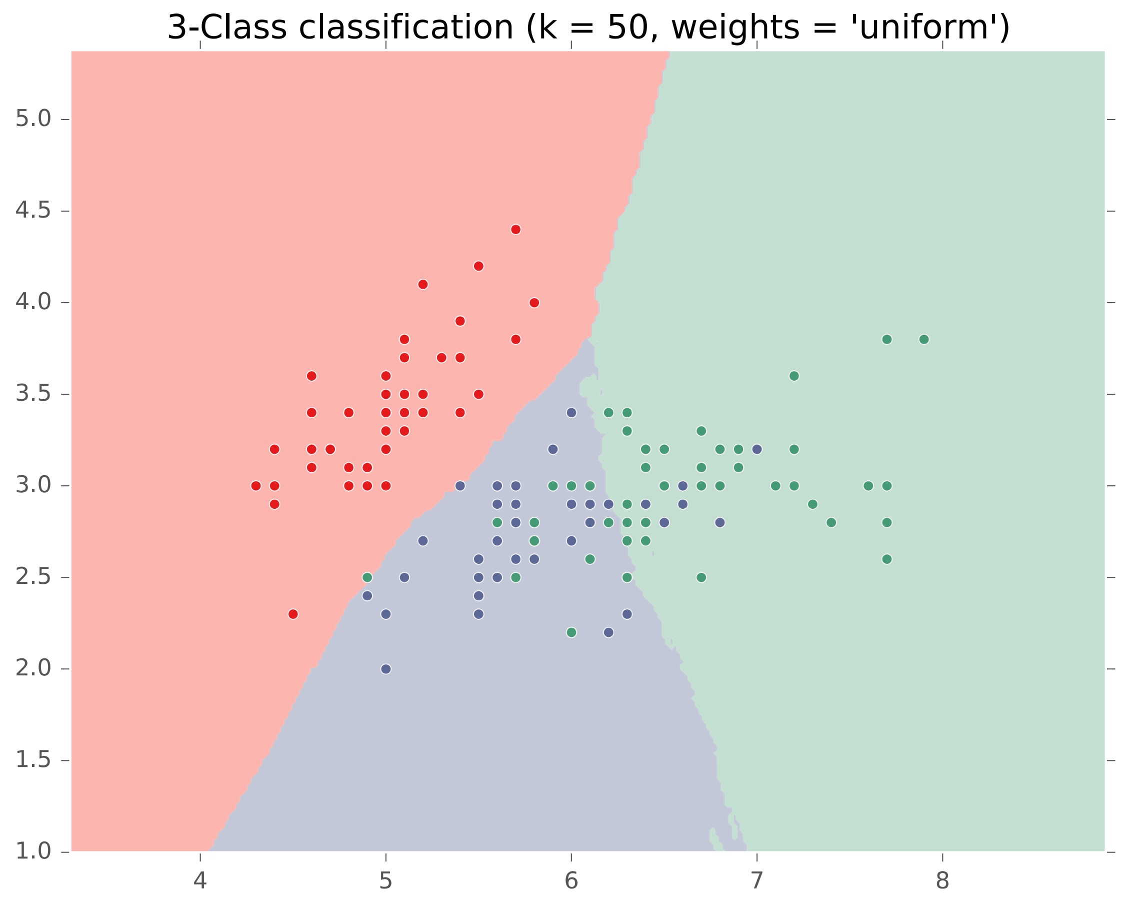





Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

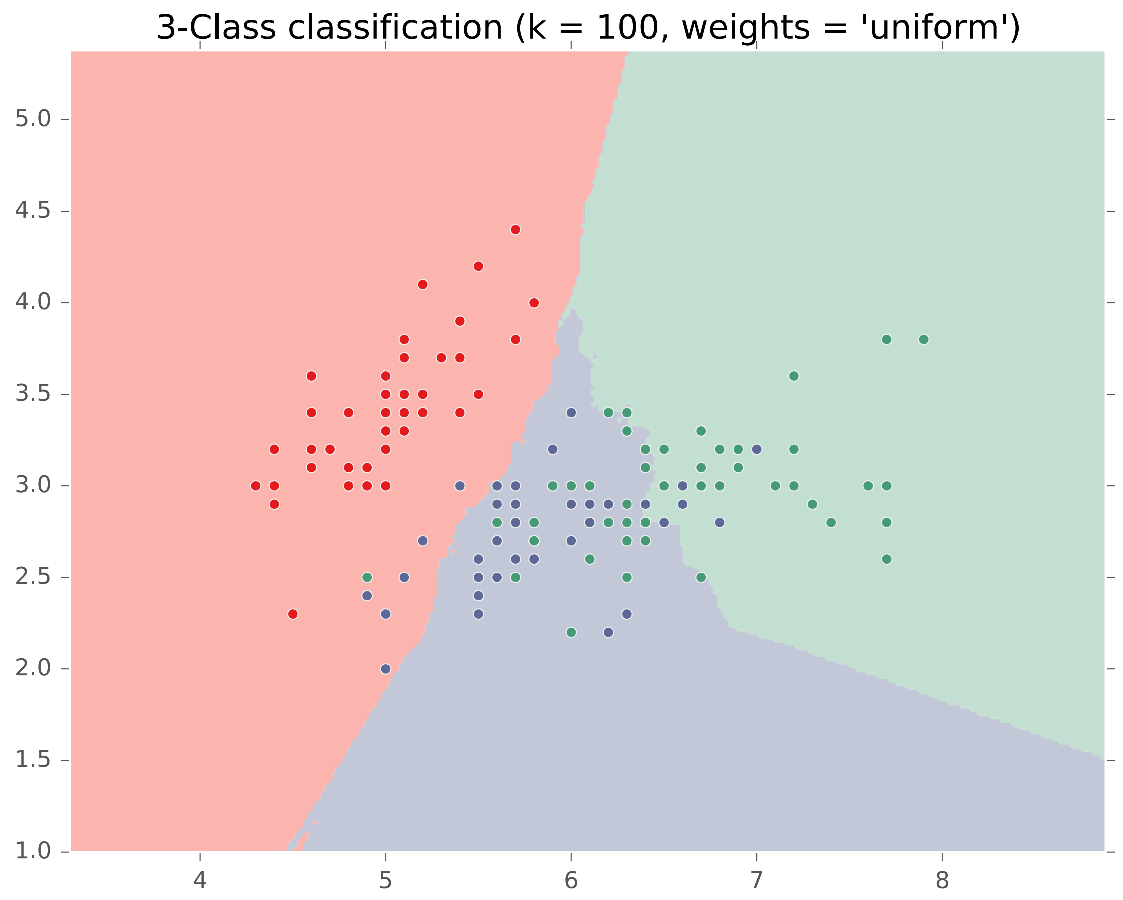





Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

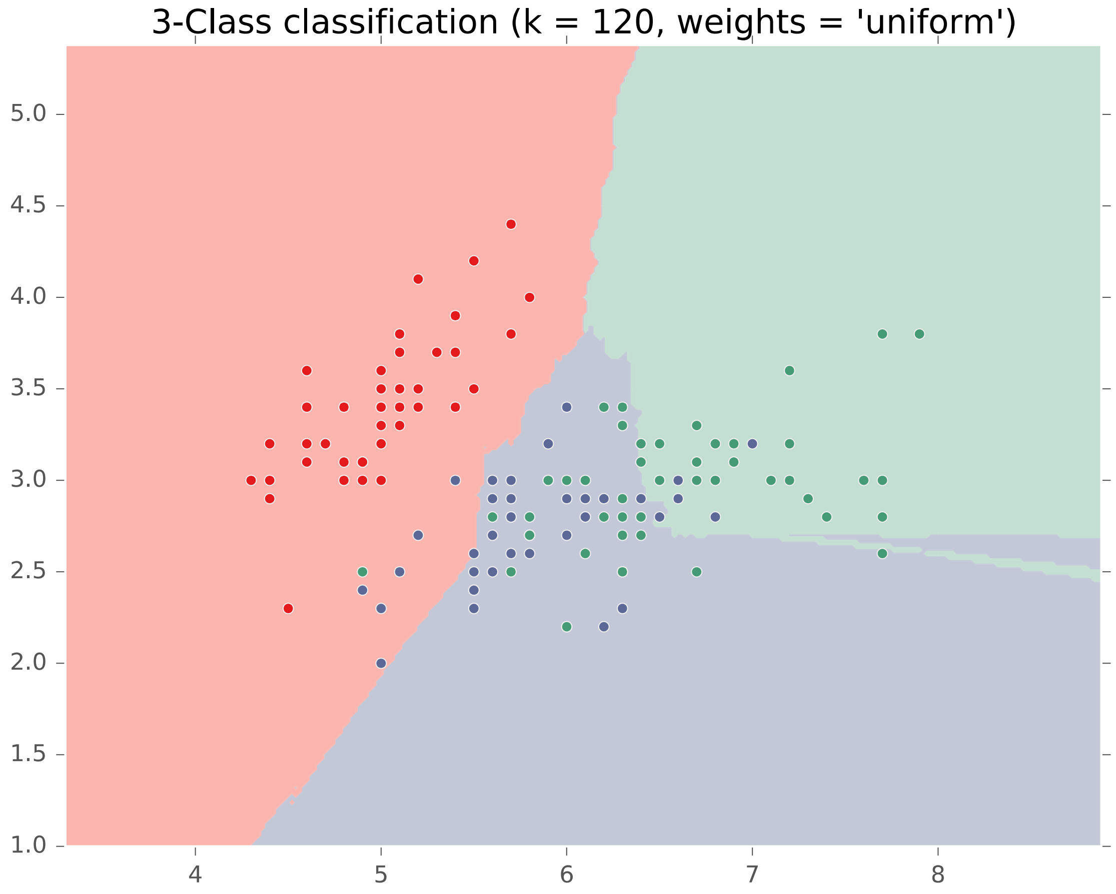



Figure courtesy of Matt Gormley

# $k$NN on Fisher Iris Data

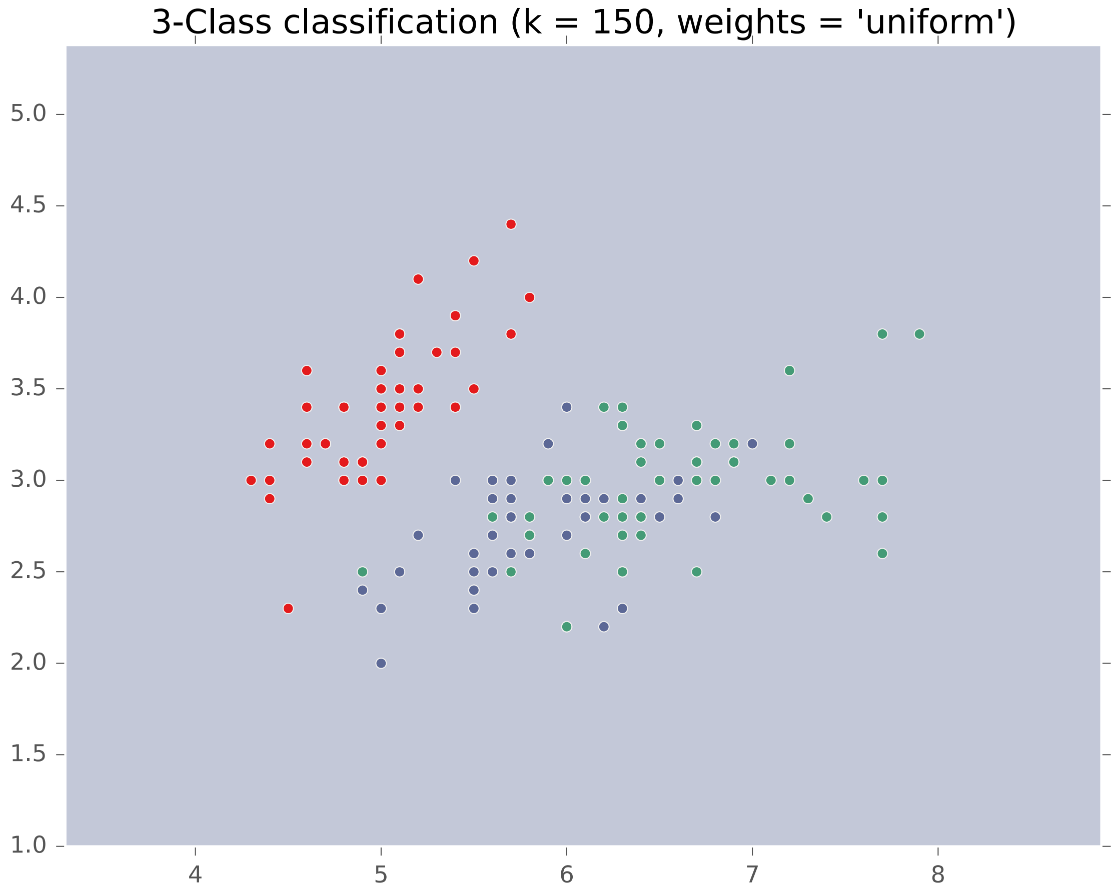
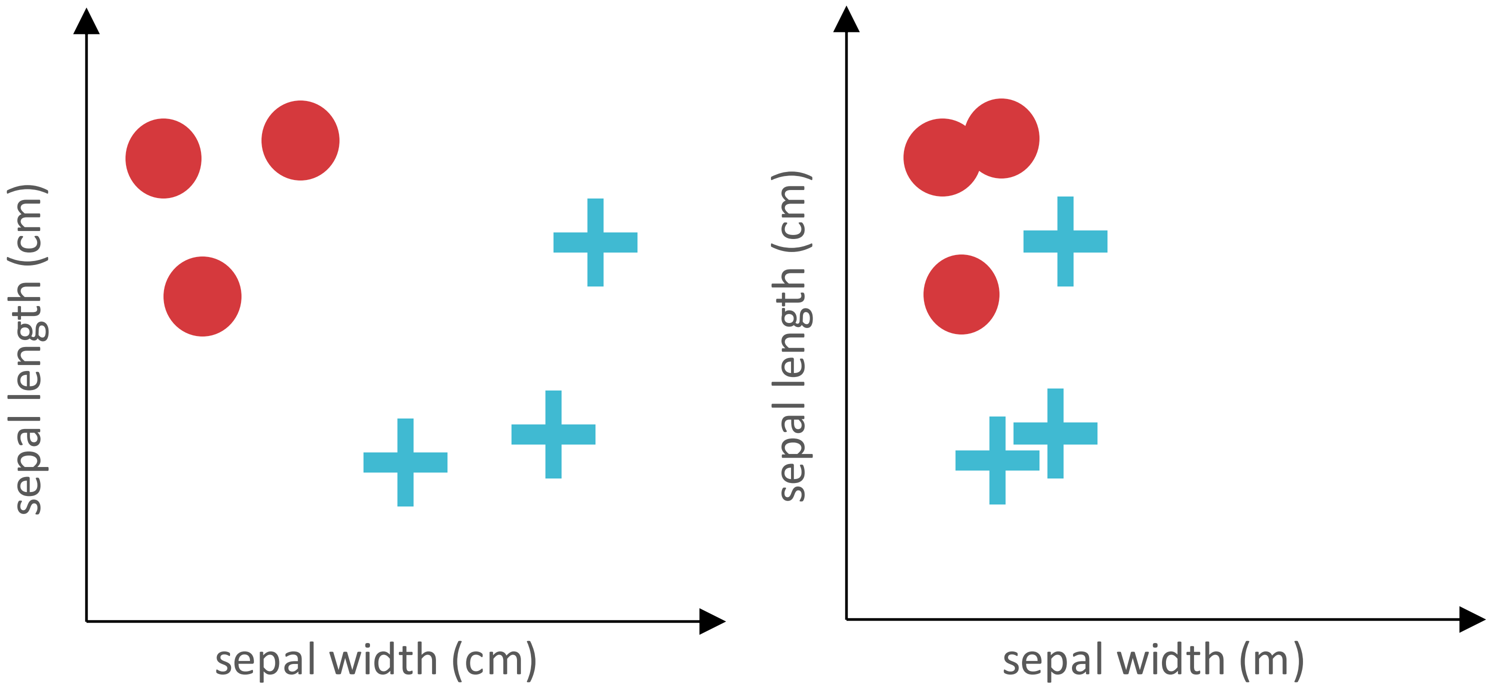



Figure courtesy of Matt Gormley

# $k$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!



Figure courtesy of Matt Gormley

# 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 $\Longrightarrow$ $k$ affects how well the learned hypothesis will generalize