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

Henry Chai & Zack Lipton

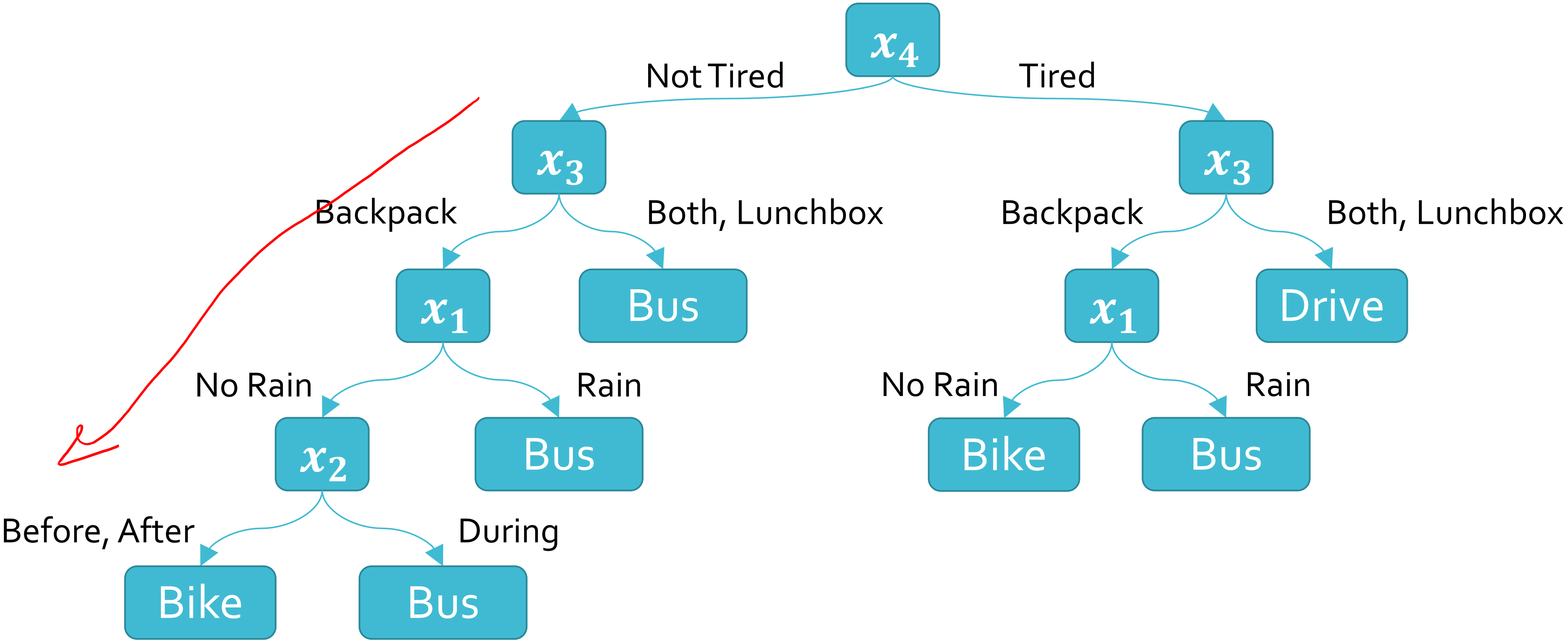
9/6/23

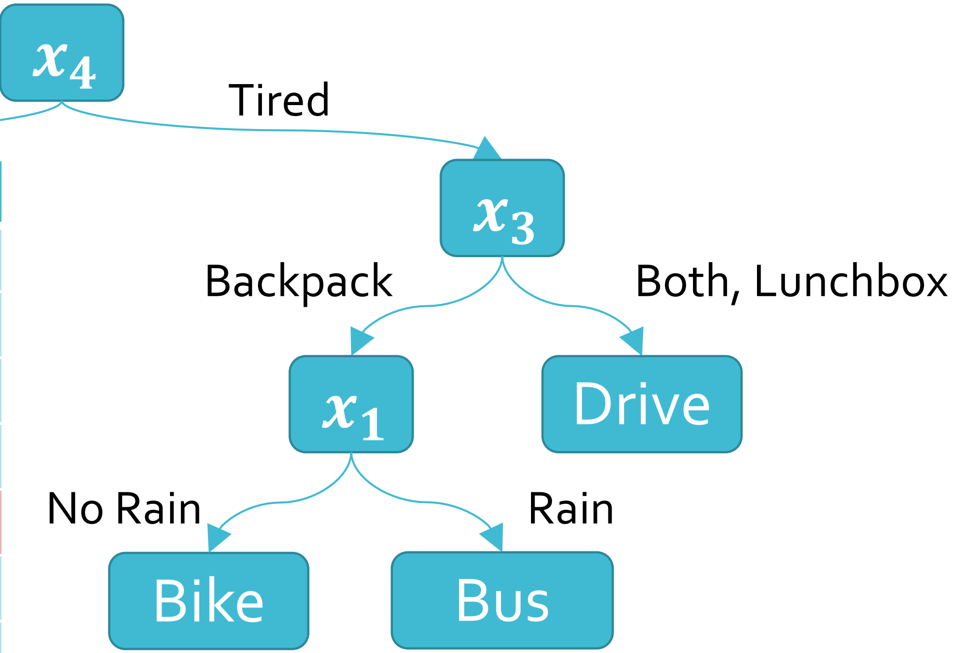
Front Matter

- Announcements:
  - HW1 released 9/6 (today!), due 9/20 at 11:59 PM
  - Recitation 1: Decision Trees and KNNs on 9/8
    - Same time and place as lecture
- Recommended Readings:
  - Mitchell, Section 8.1 – 8.2: k-Nearest Neighbor Learning
  - Daumé III, Chapter 3: Geometry and Nearest Neighbors

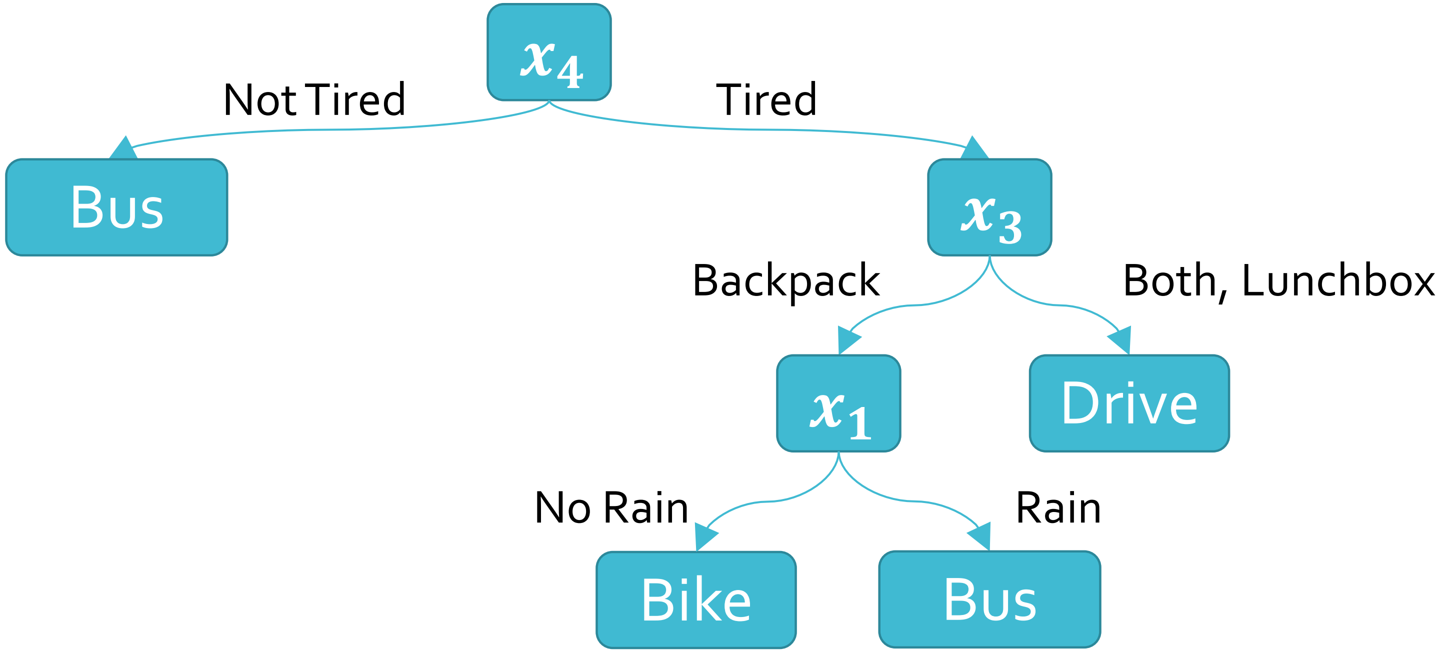
Recall: Decision Trees

- 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!


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



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

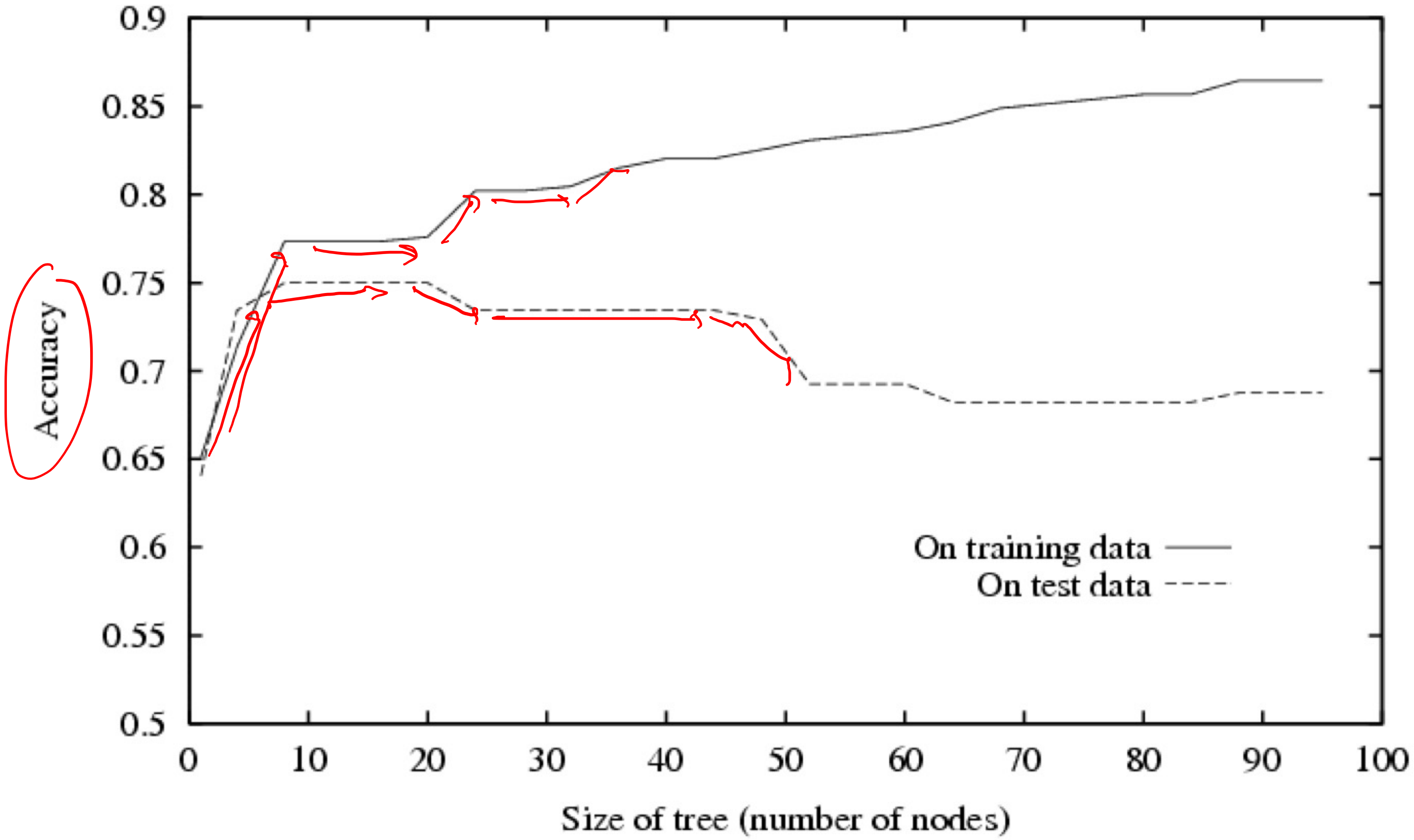


Figure courtesy of Tom Mitchell

Combatting Overfitting in Decision Trees

- 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

- Pruning:
  1. First, learn a decision tree
  2. Then, 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

Pruning Decision Trees

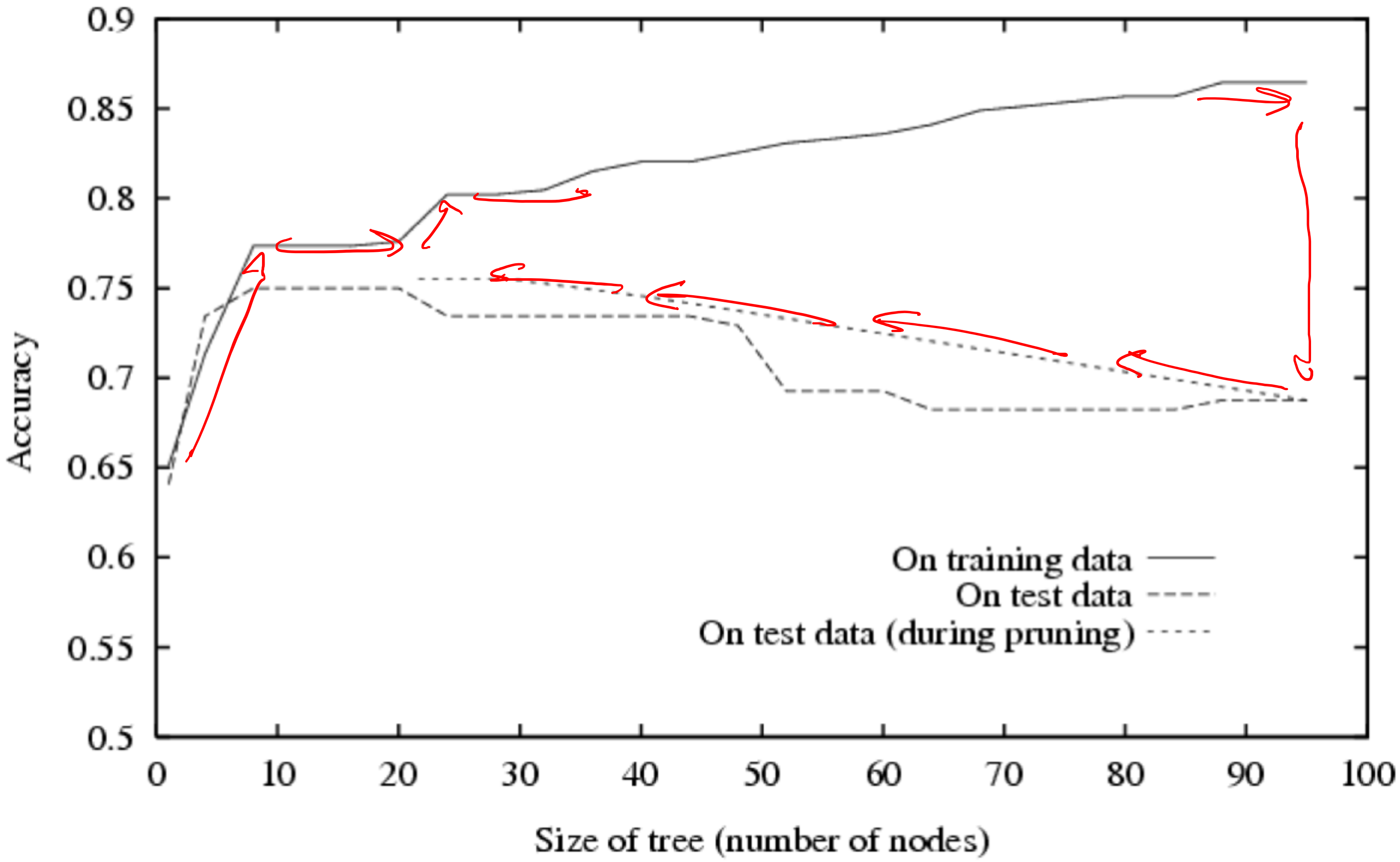
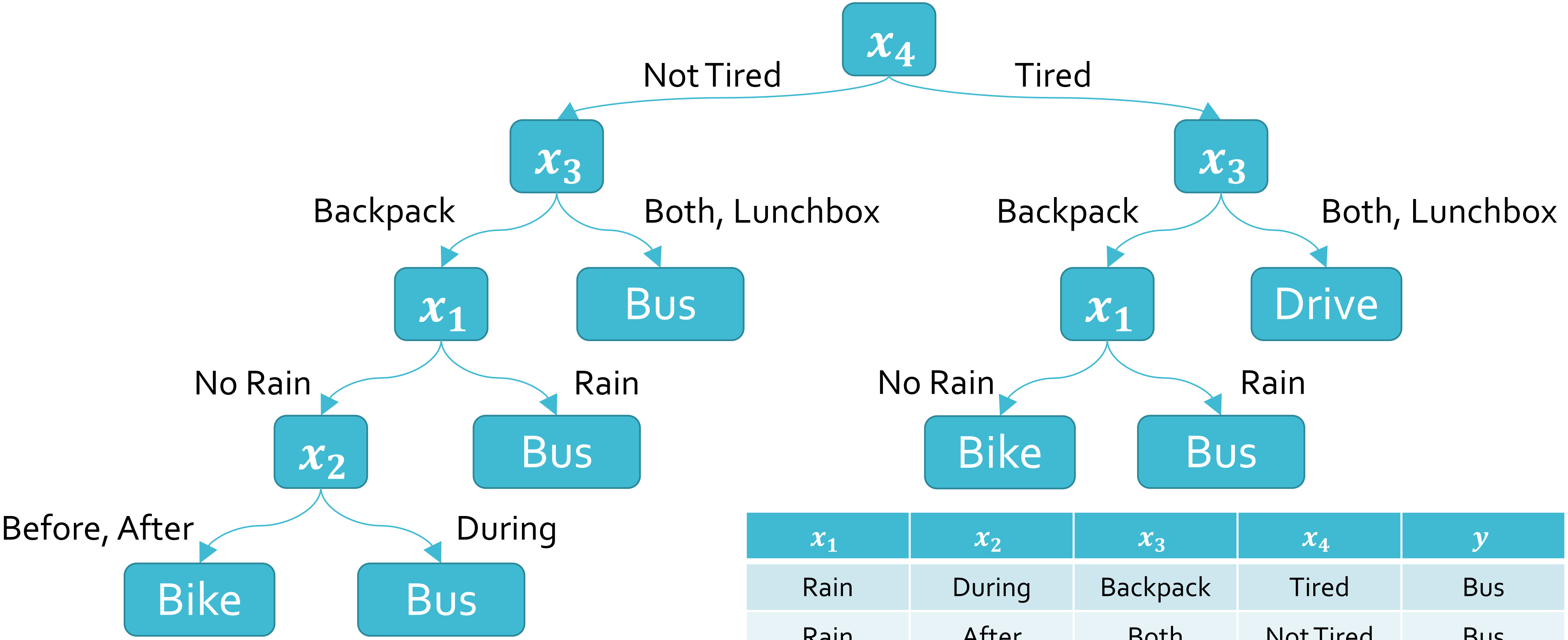
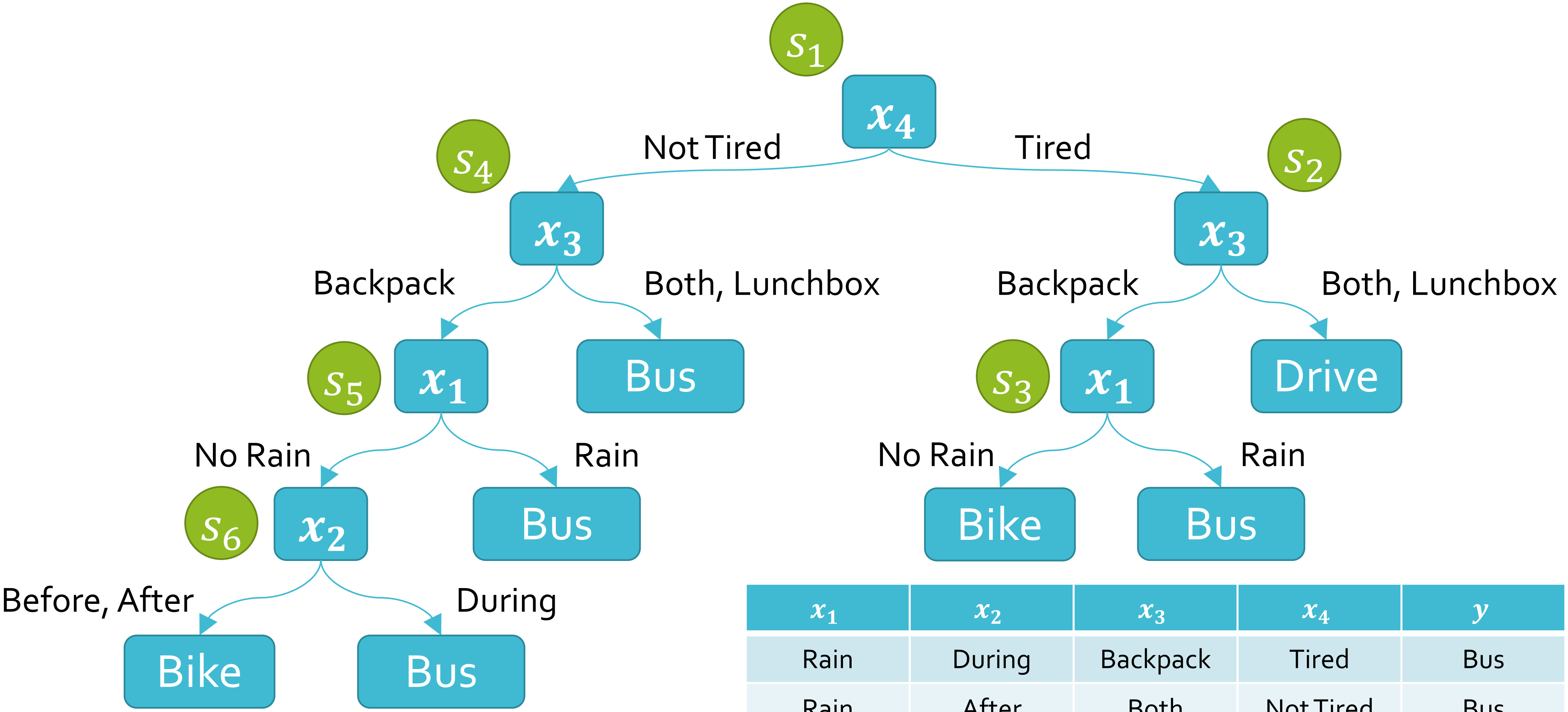


Figure courtesy of Tom Mitchell



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

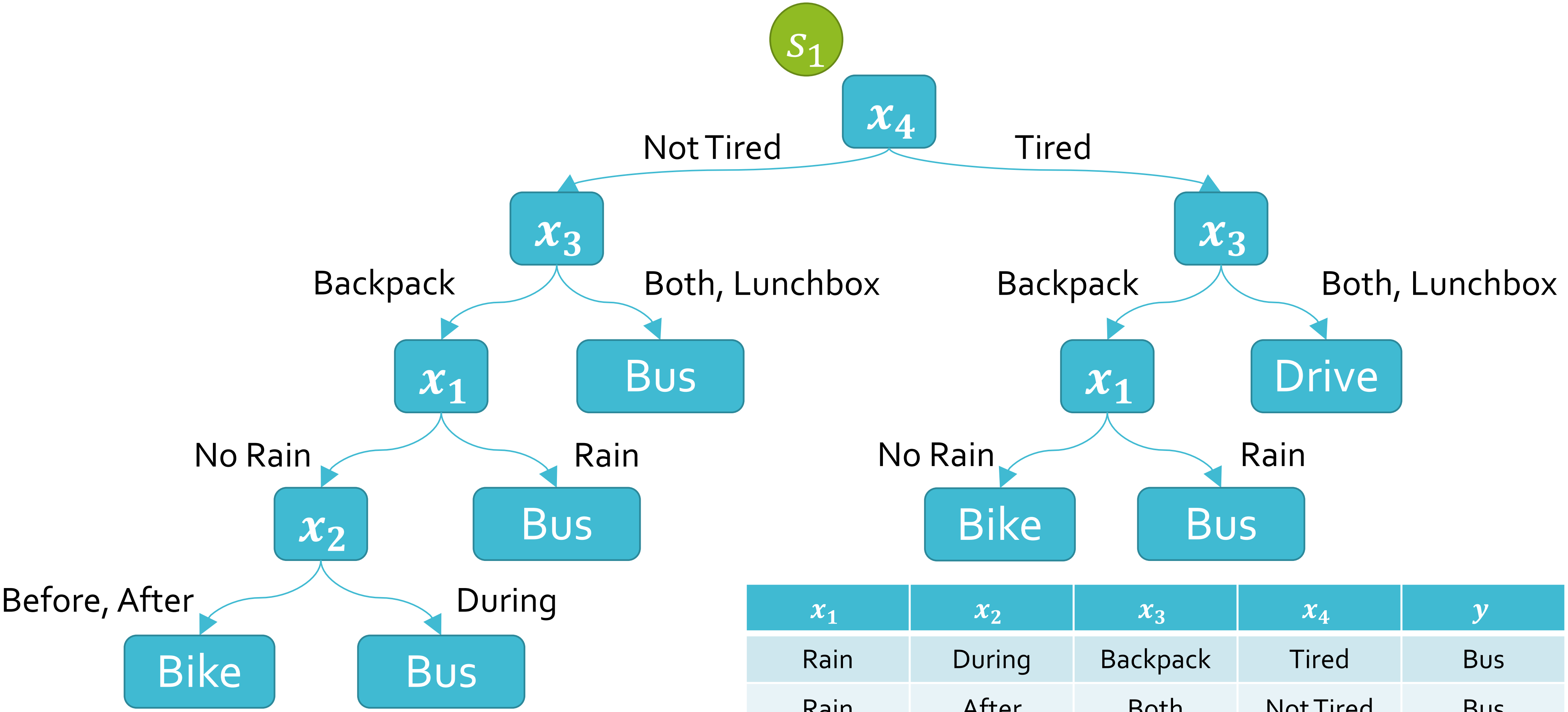
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



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

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

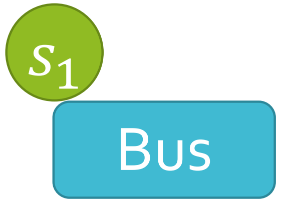
$$err(h, \mathcal{D}_{val}) = 0.2$$



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

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

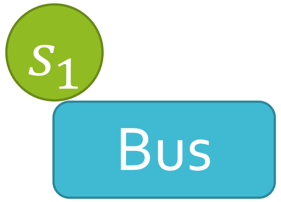
$$err(h - s_1, \mathcal{D}_{val})$$



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

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

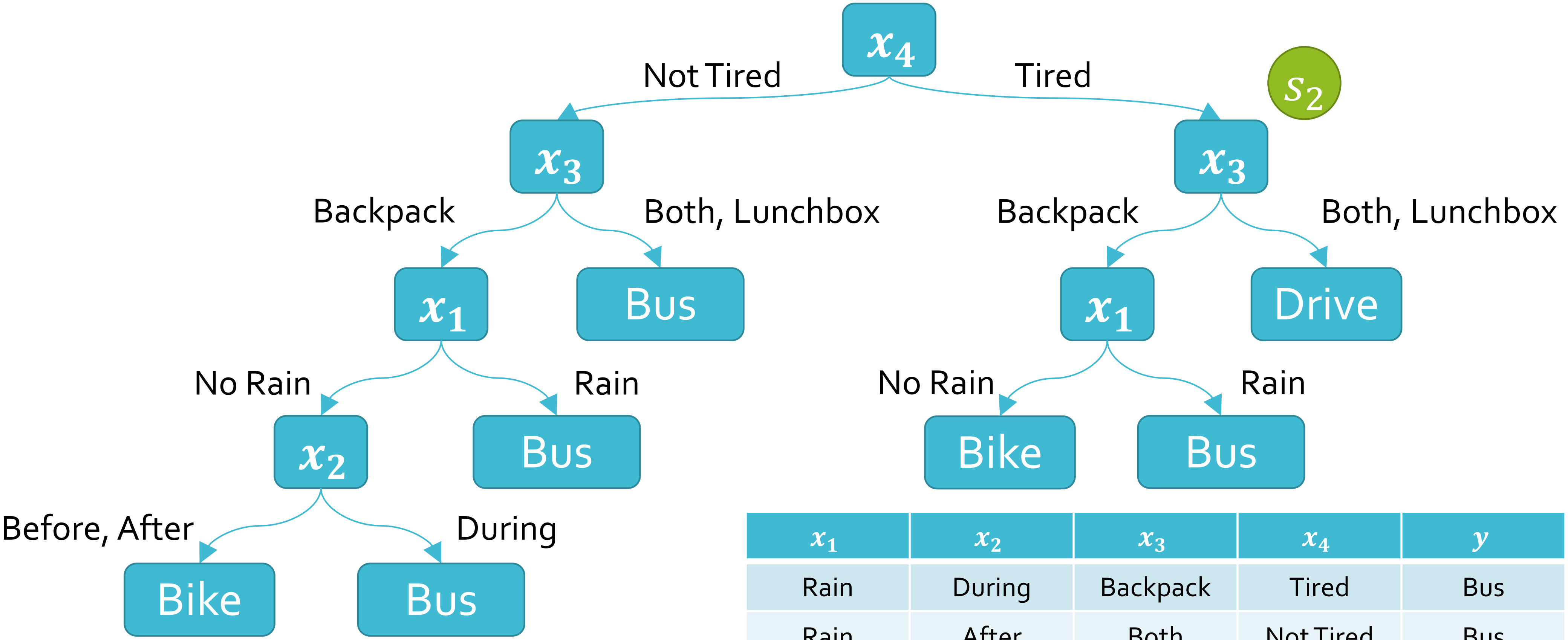
$err(h - s_1, \mathcal{D}_{val})$



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

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

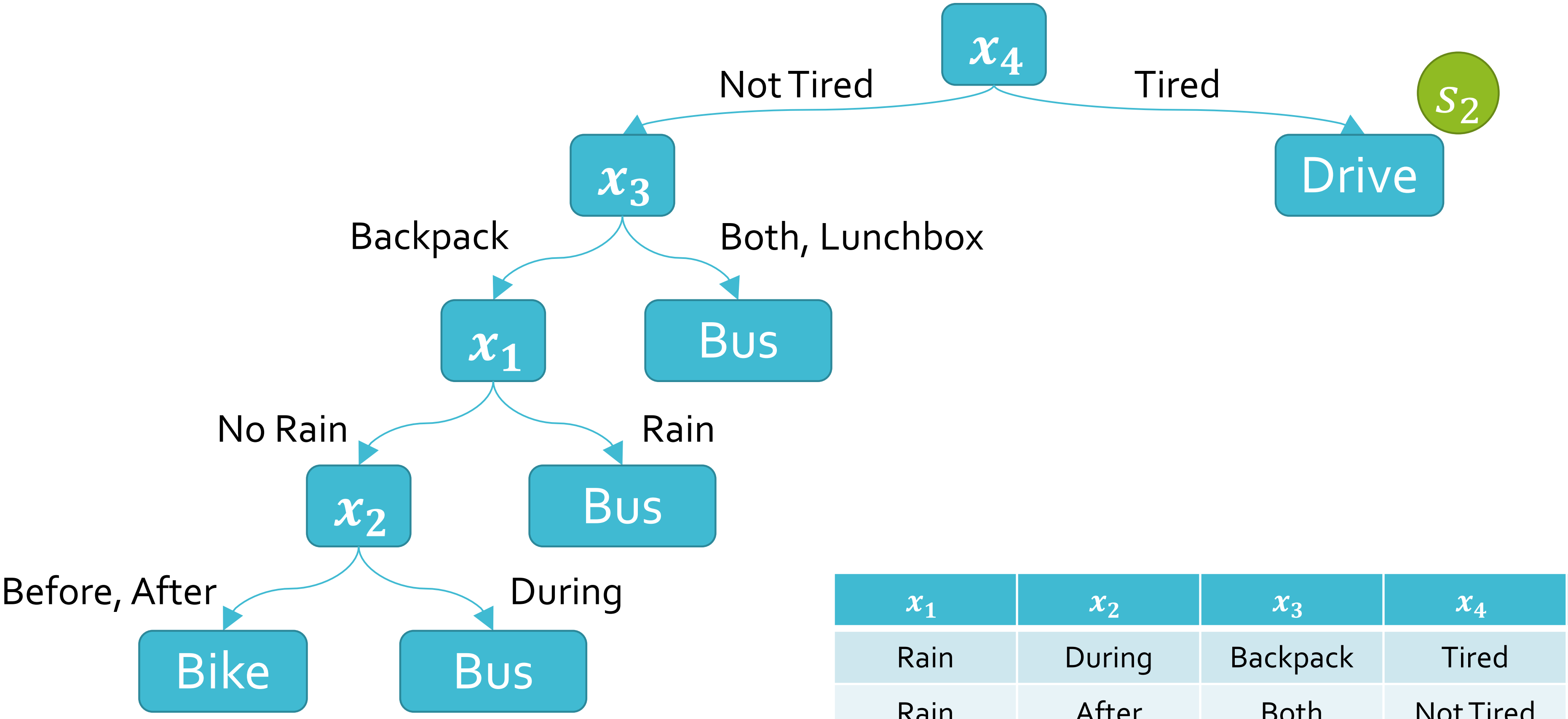
$$err(h - s_1, \mathcal{D}_{val}) = 0.4$$



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

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

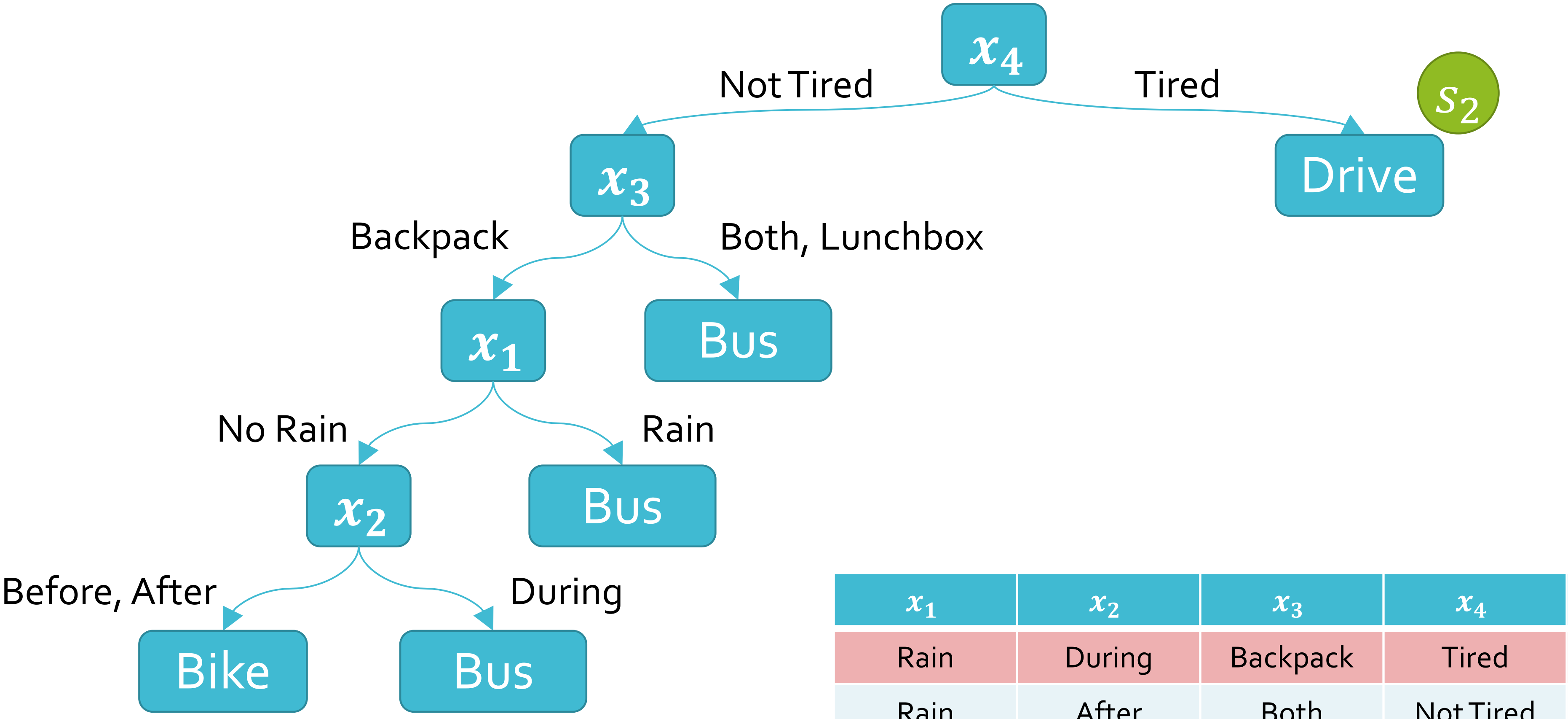
$err(h - s_2, \mathcal{D}_{val})$



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

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

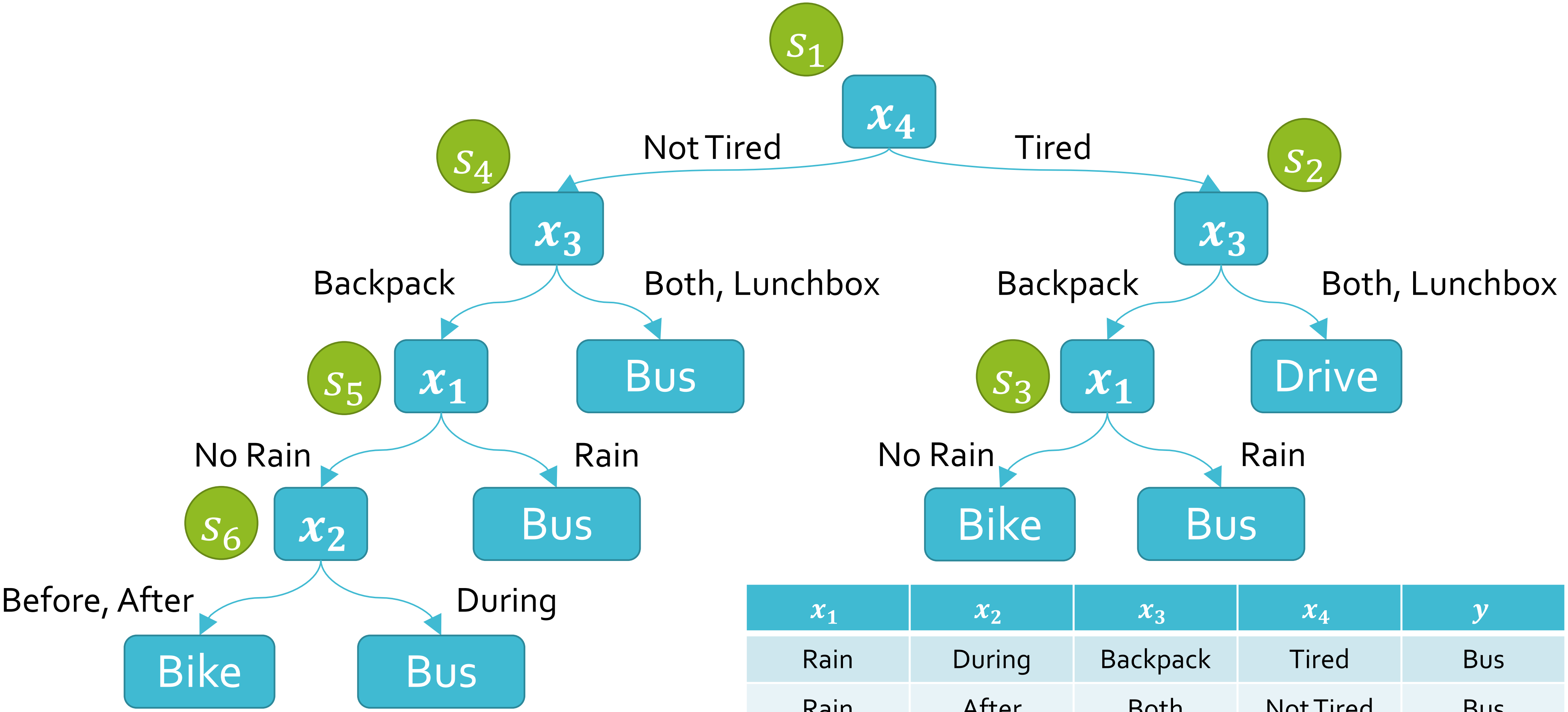
$err(h - s_2, \mathcal{D}_{val})$



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

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

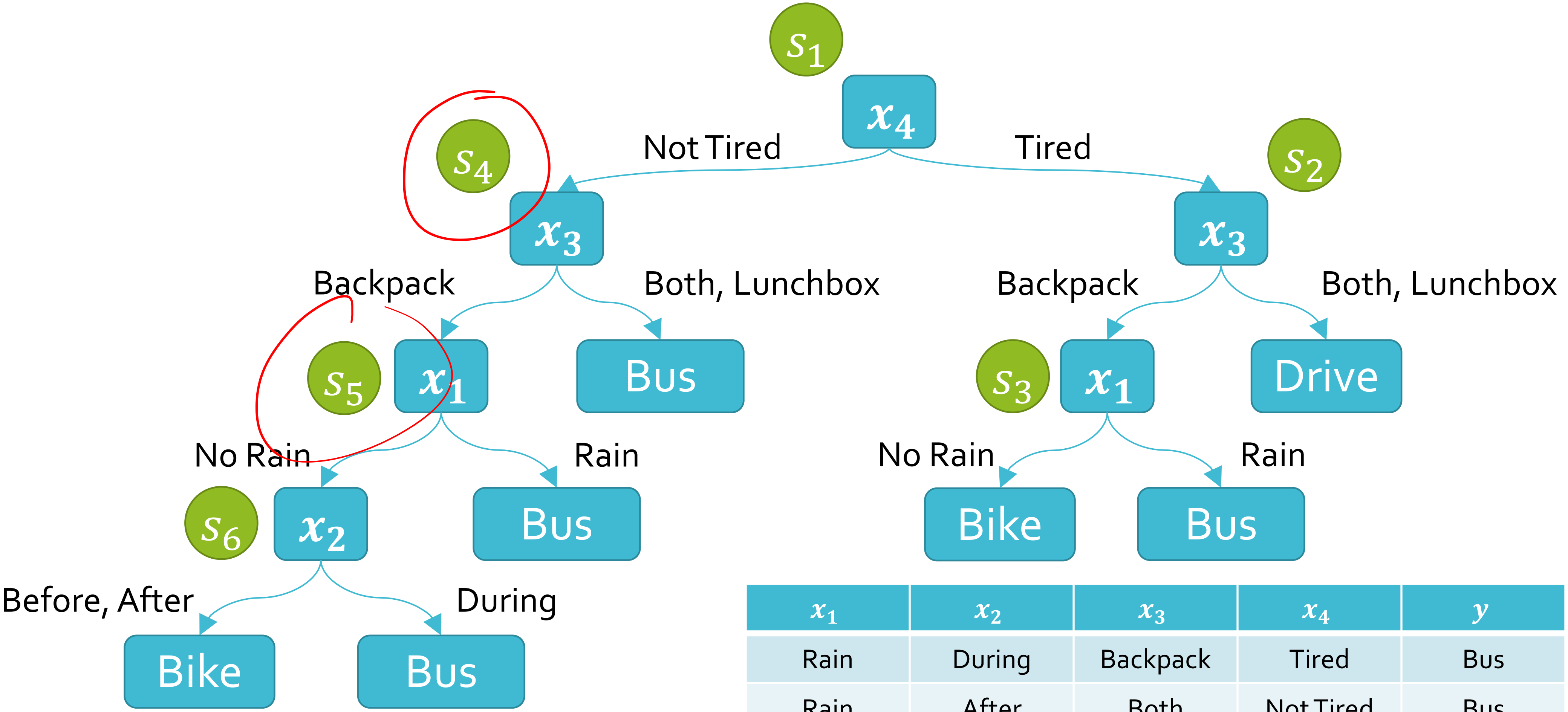
$$err(h - s_2, \mathcal{D}_{val}) = 0.4$$



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

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

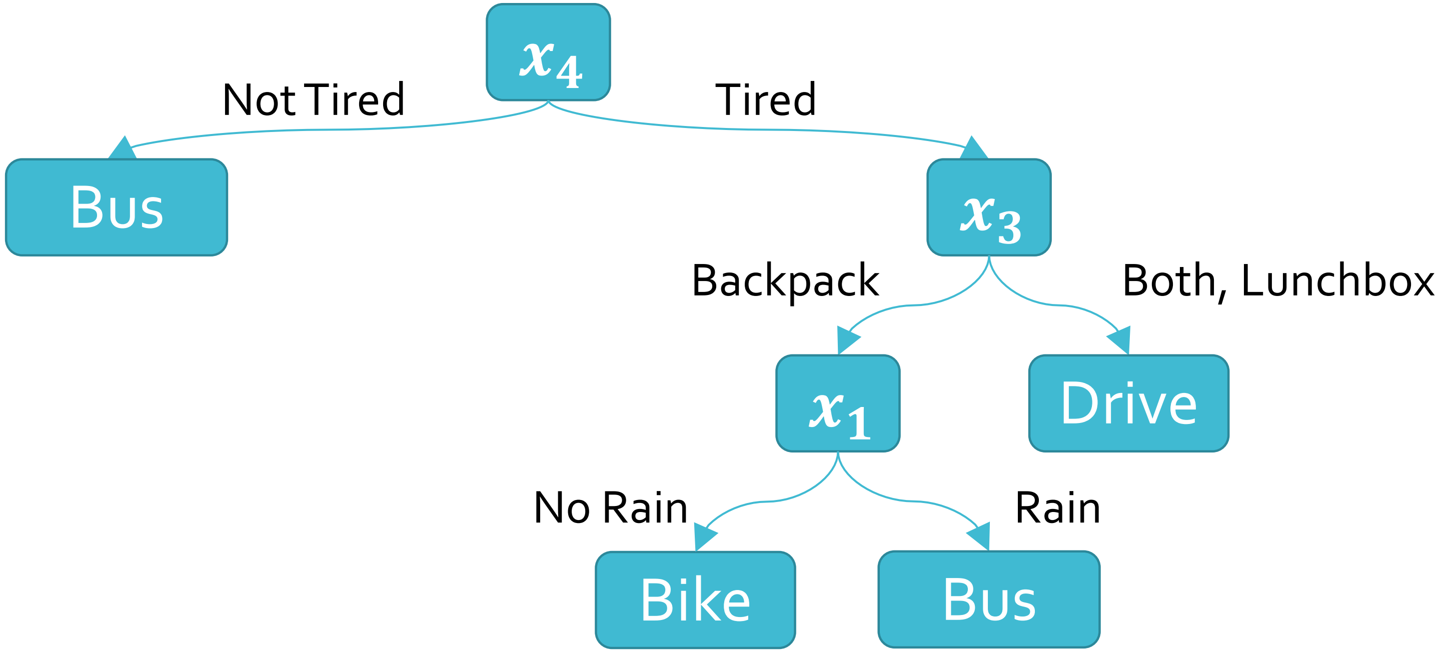
s	s_1	s_2	s_3	s_4	s_5	s_6
$err(h - s, \mathcal{D}_{val})$	0.4	0.4	0.4	0	0	0.2



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

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

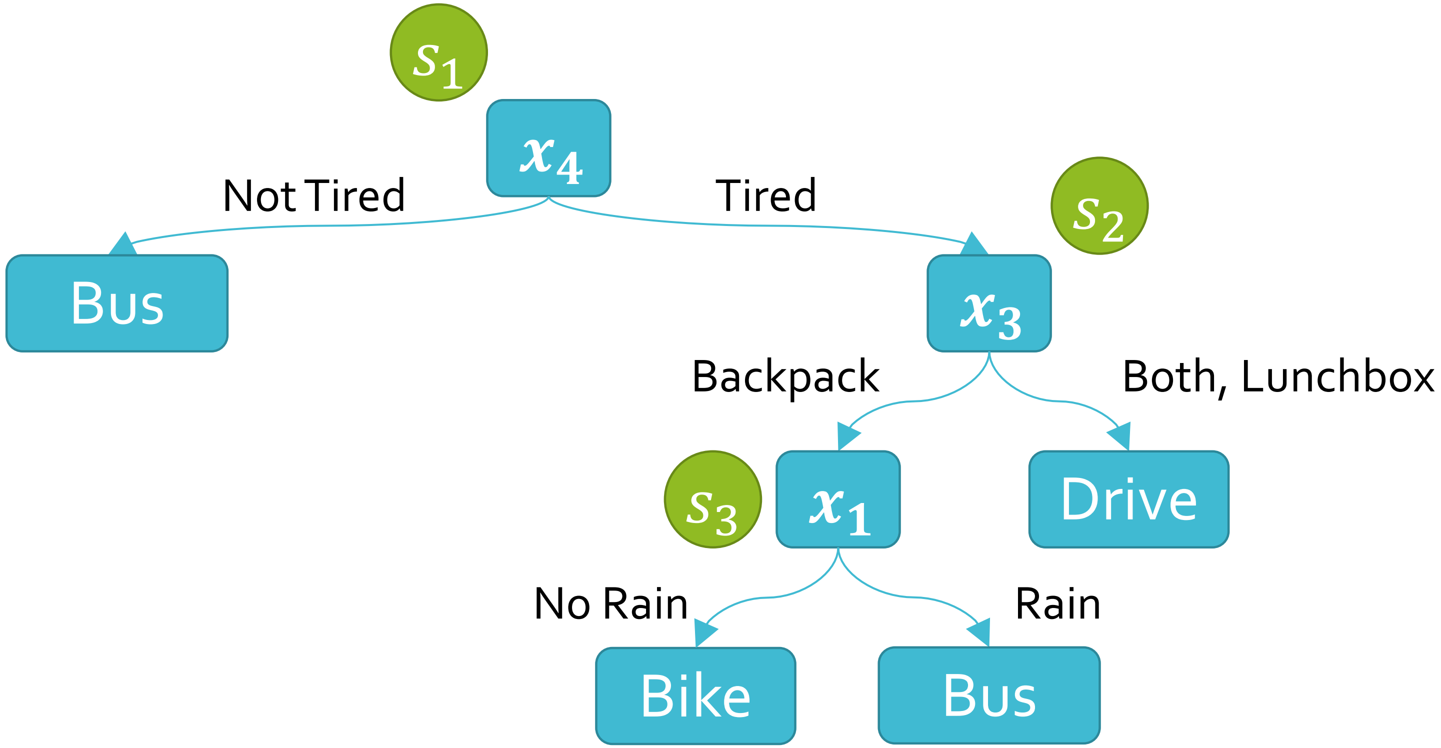
x_1	x_2	x_3	x_4	y
Rain	During	Backpack	Tired	Bus
Rain	After	Both	Not Tired	Bus
No Rain	Before	Backpack	Not Tired	Bus
No Rain	During	Lunchbox	Tired	Drive
No Rain	After	Lunchbox	Tired	Drive



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

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

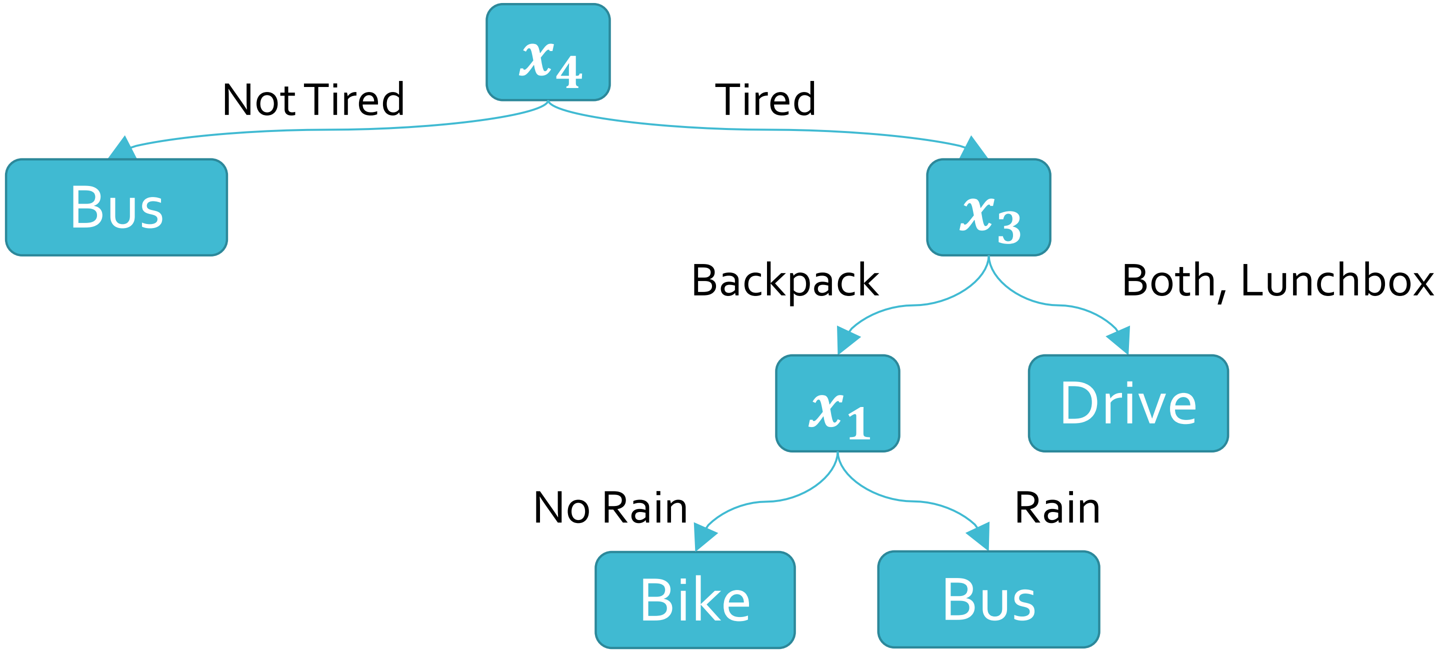
$err(h, \mathcal{D}_{val}) = 0$



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

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

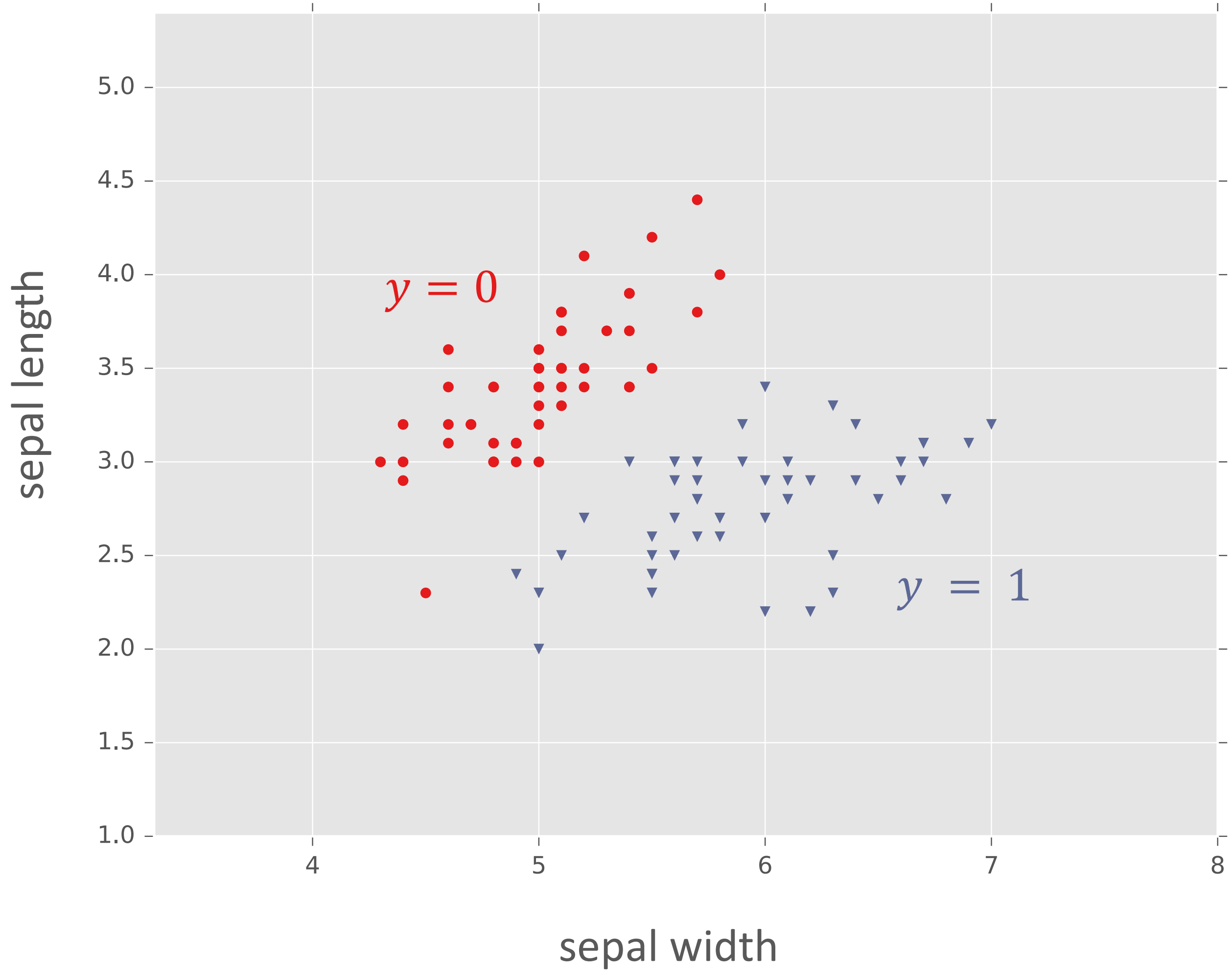
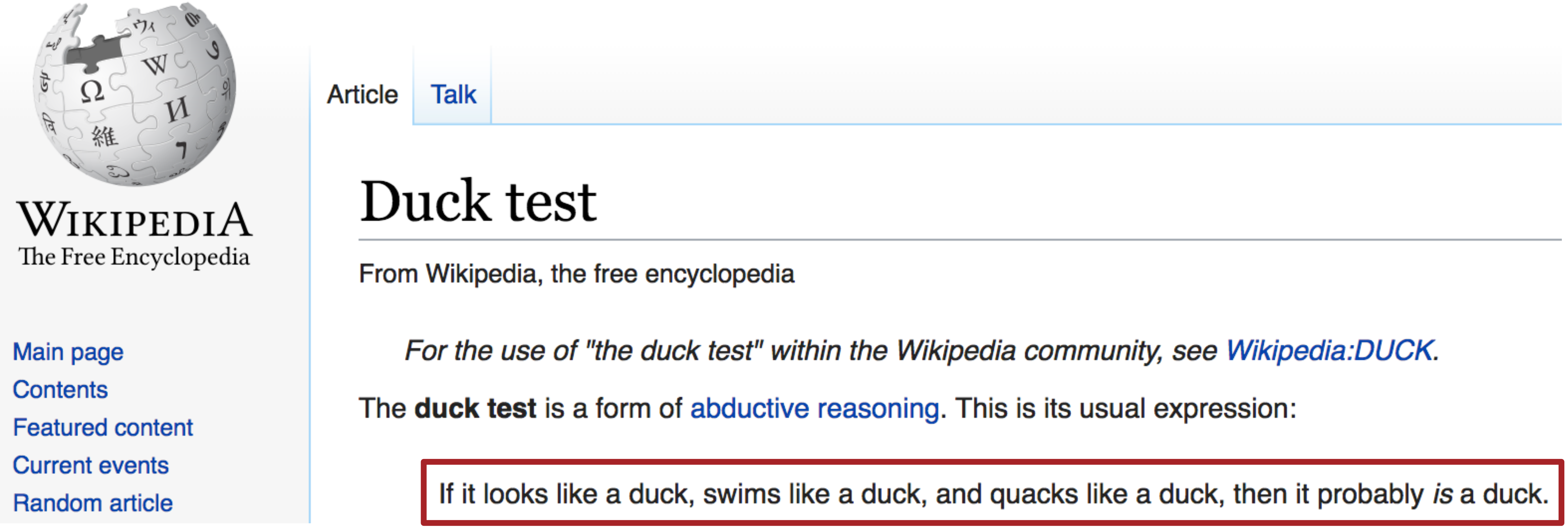


Figure courtesy of Matt Gormley



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

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

The Duck Test for Machine Learning

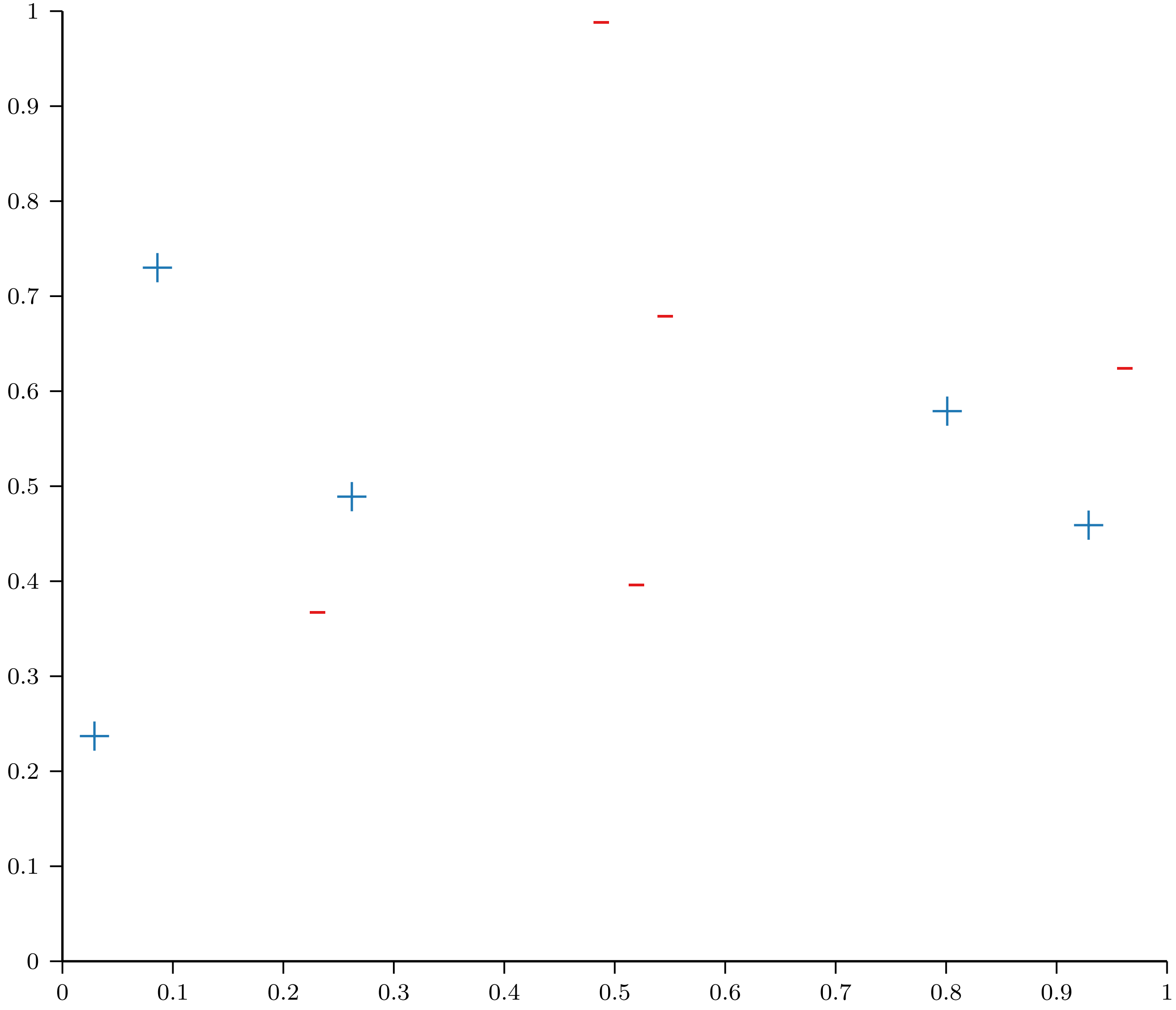
- Classify a point as the label of the "most similar" training point
- Idea: given real-valued features, we can use a distance metric to determine how similar two data points are
- A common choice is Euclidean distance:

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

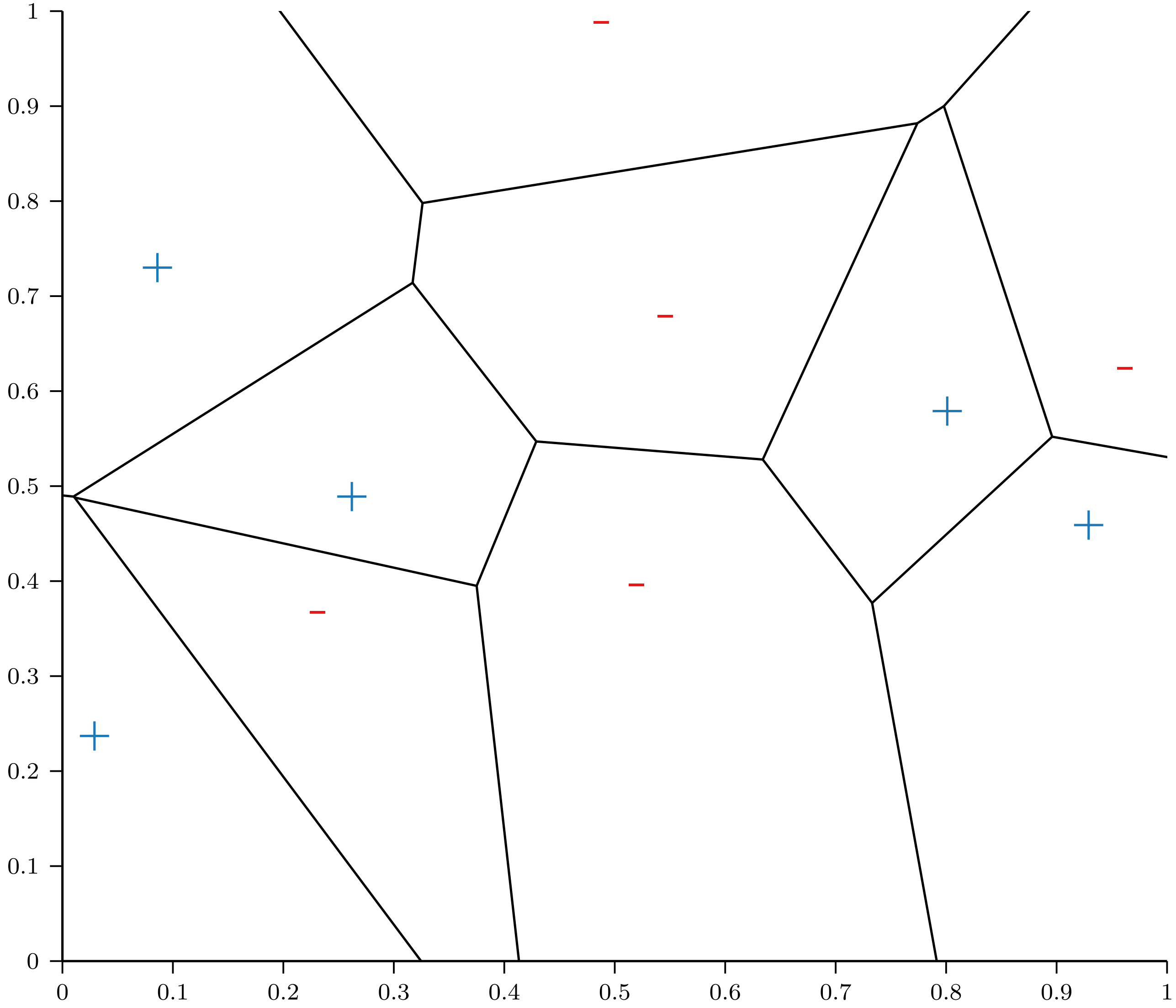
- An alternative is the Manhattan distance:

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

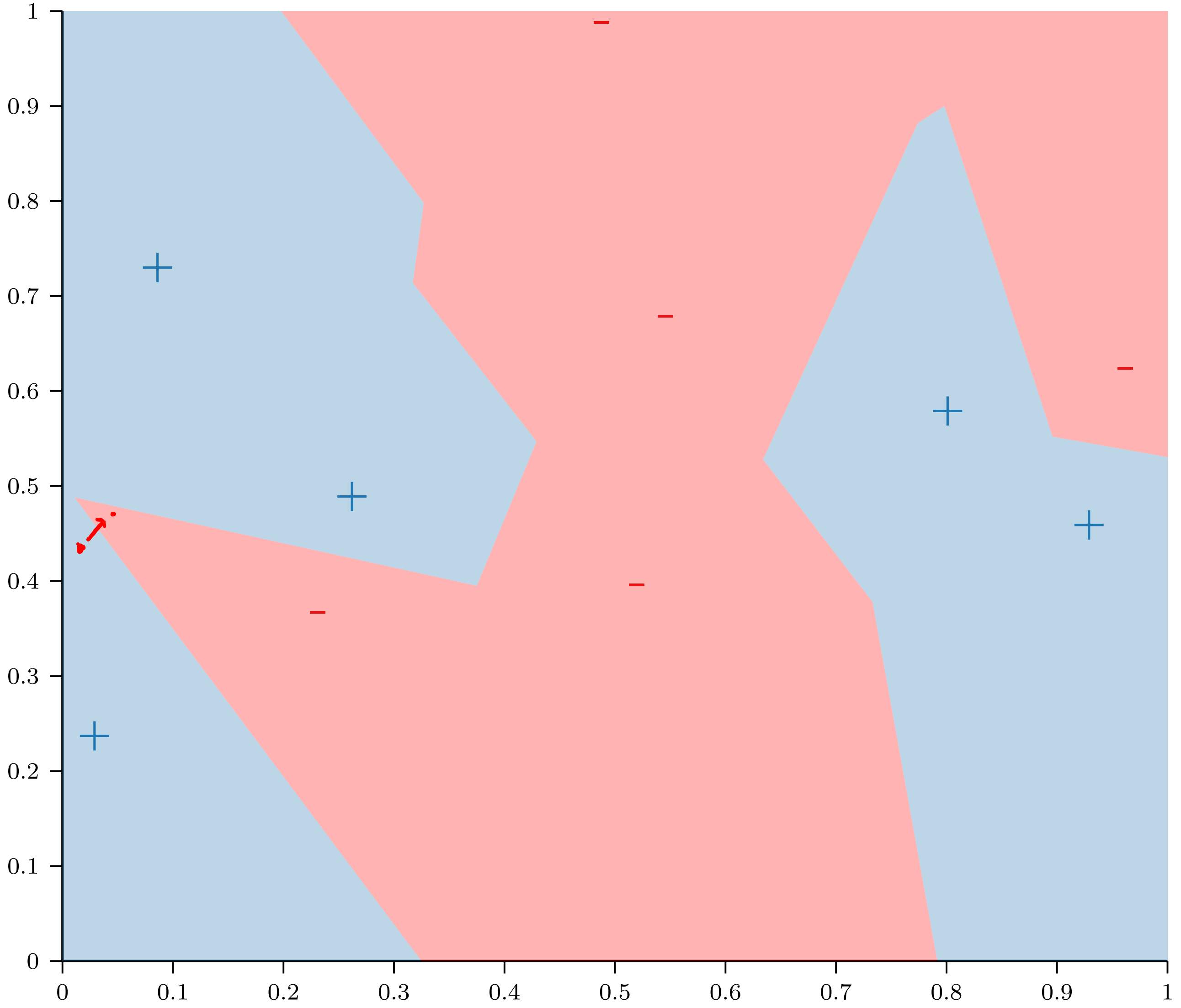
Nearest Neighbor: Example



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 $\underline{n \to \infty}$, with high probability, the true error rate of the nearest neighbor model $\leq 2\ *$ 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.”

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

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 $\leq 2\ *$ 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.”

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

k-Nearest Neighbors (kNN)

- 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)

- (weighted) randomly
- majority vote in some larger subset of the training data (e.g. k+c NNs)
- distance-weight the neighbors
- add noise to the query data point
- try another distance metric

kNN on Fisher Iris Data

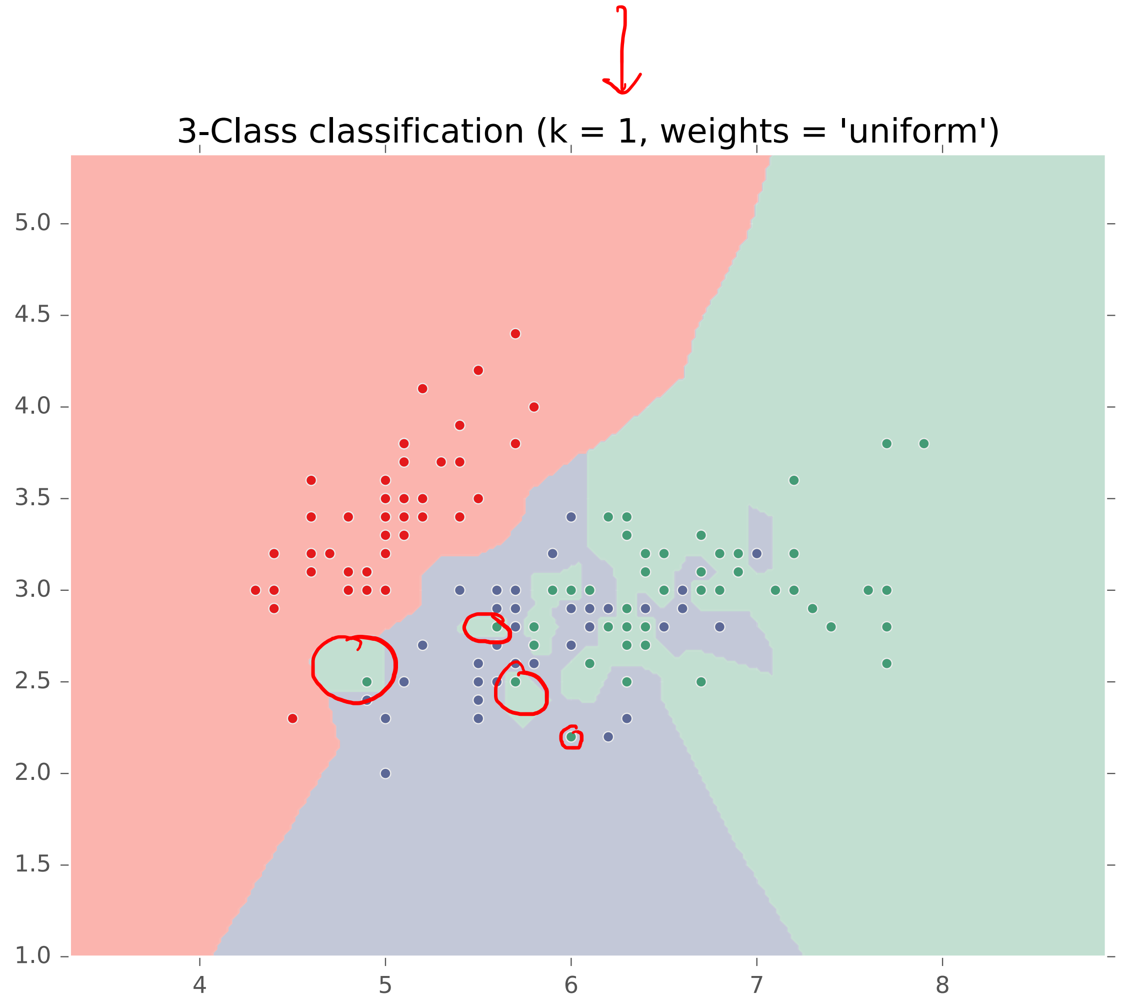


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

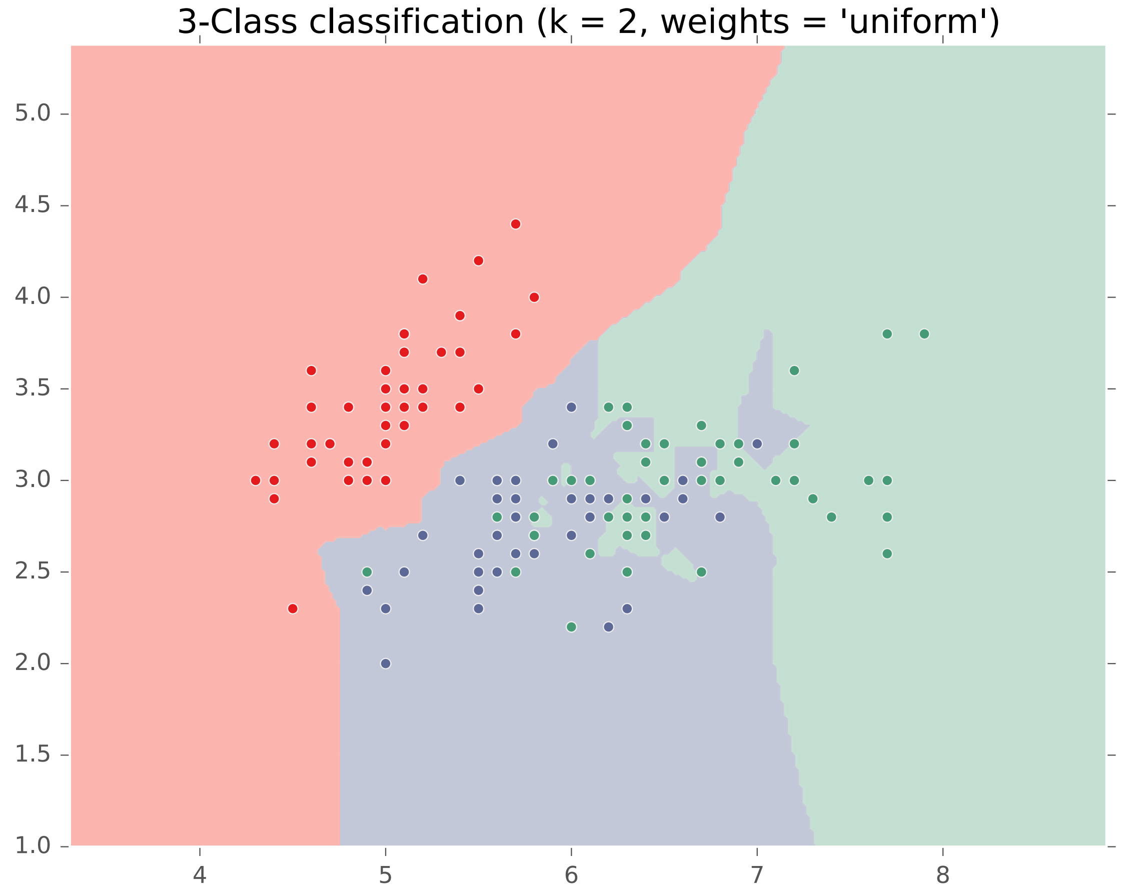


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

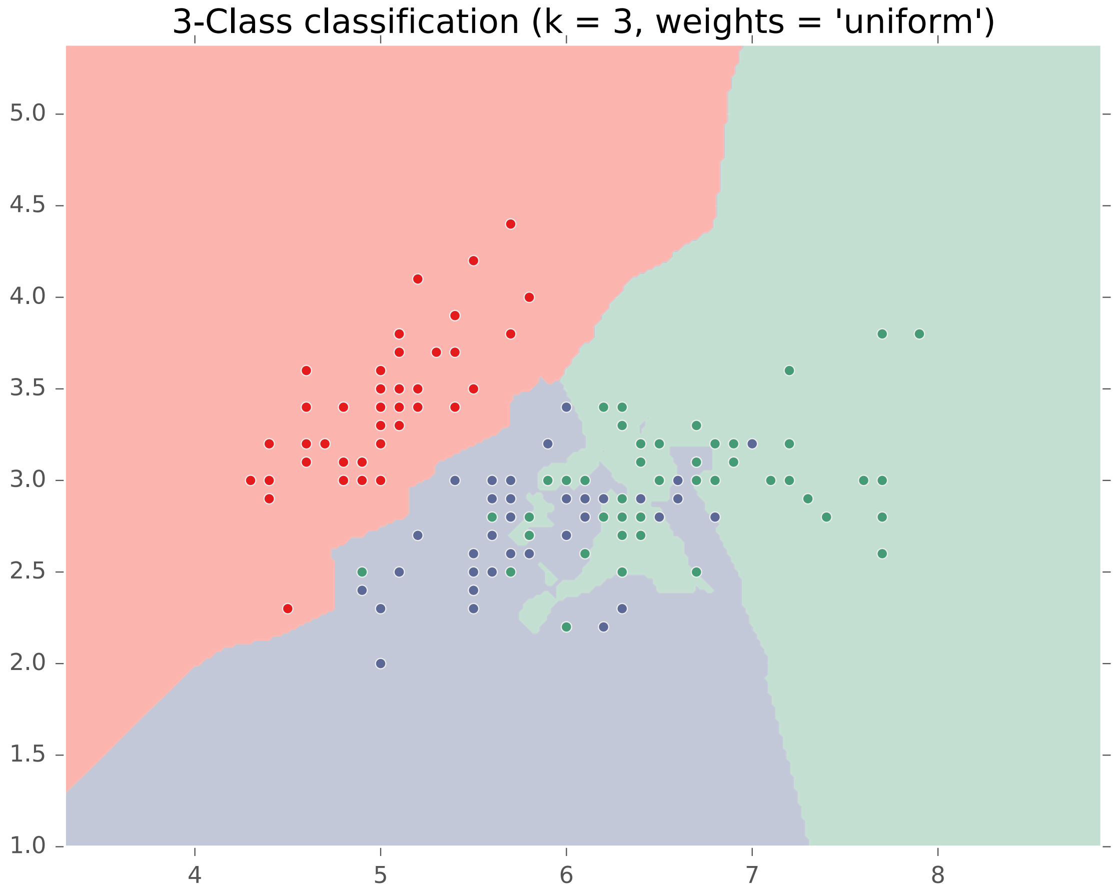


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

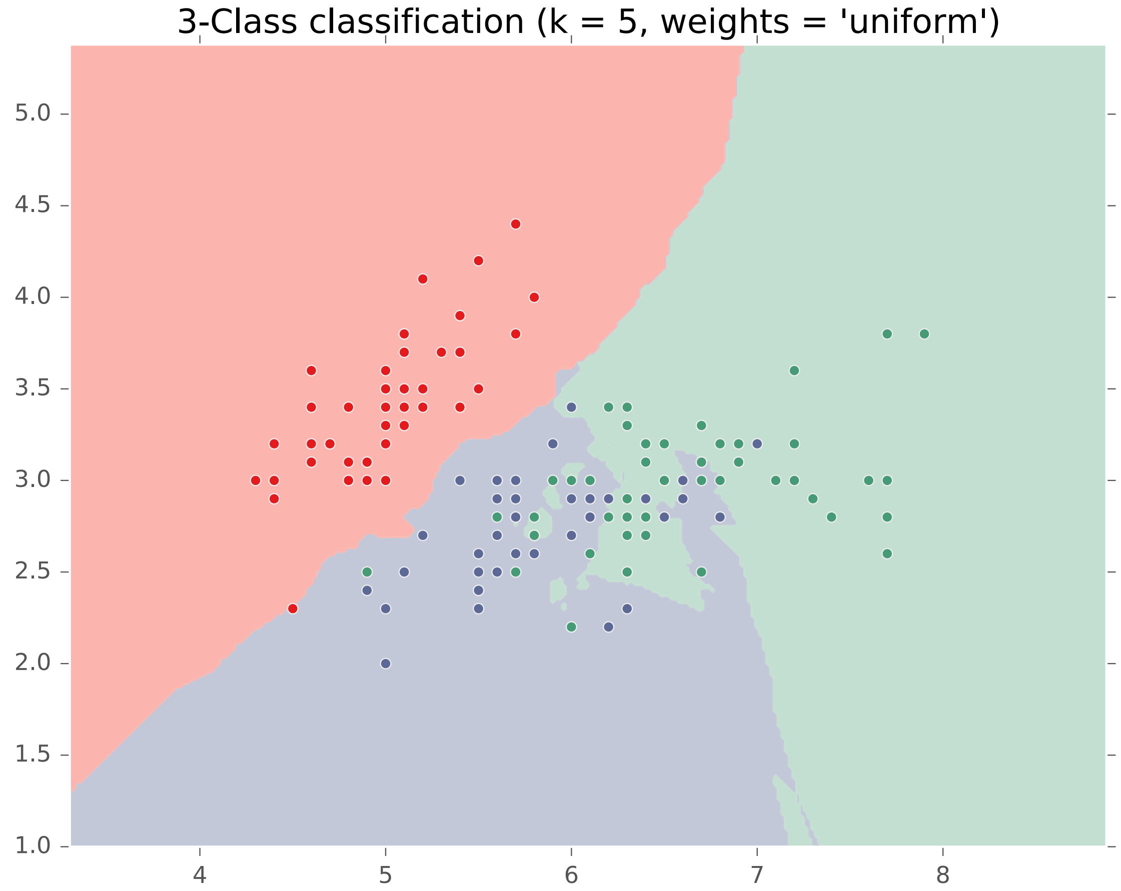


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

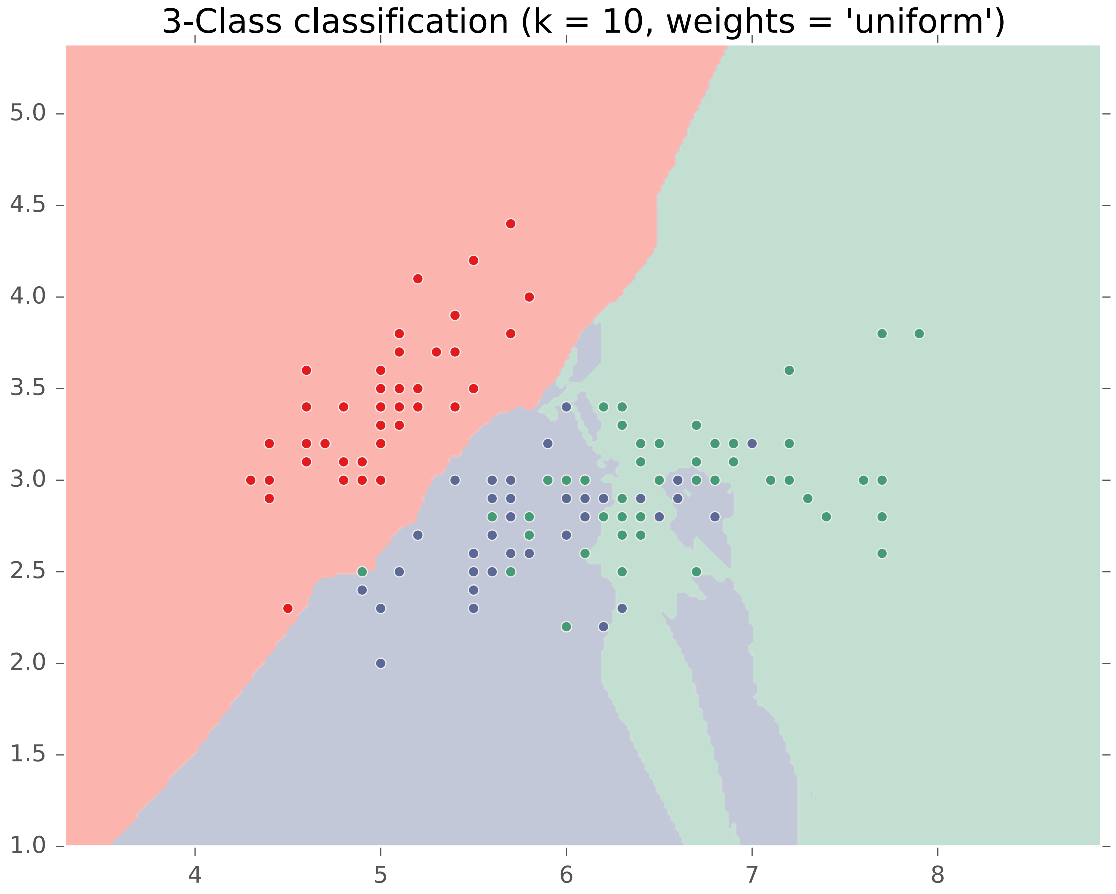


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

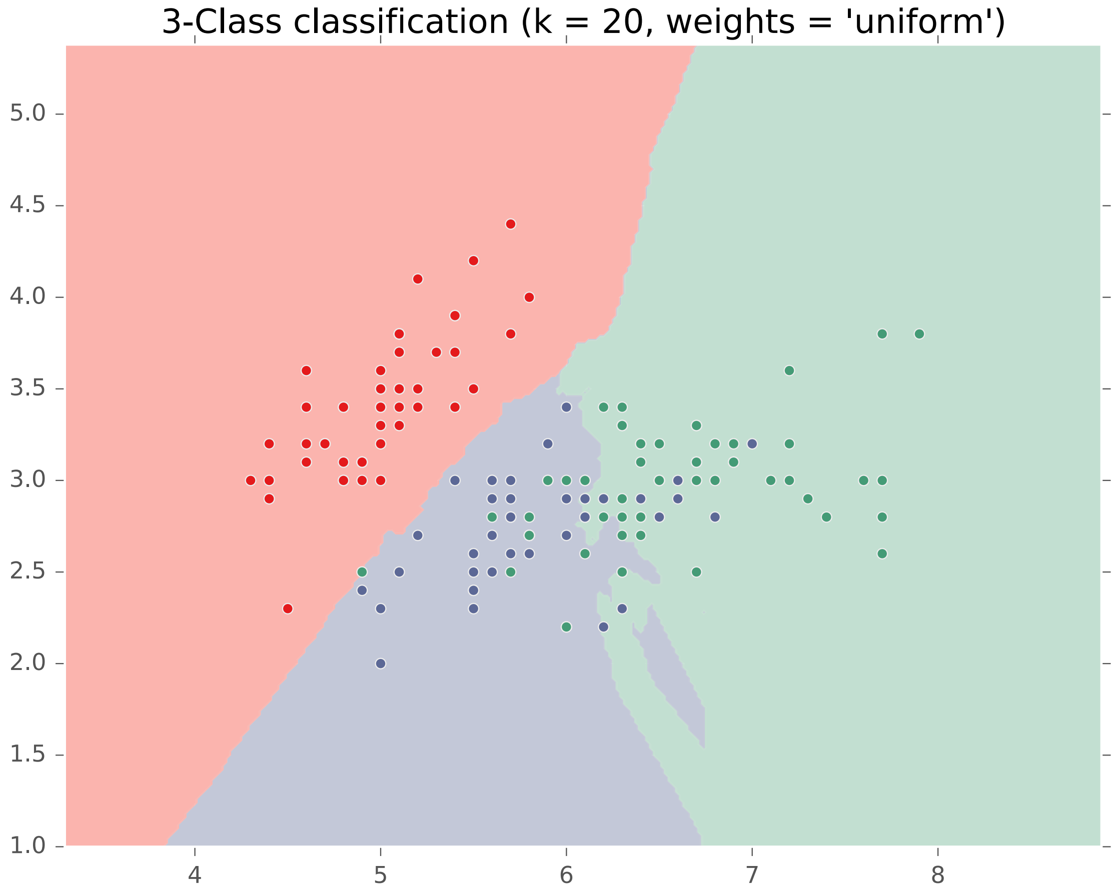


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

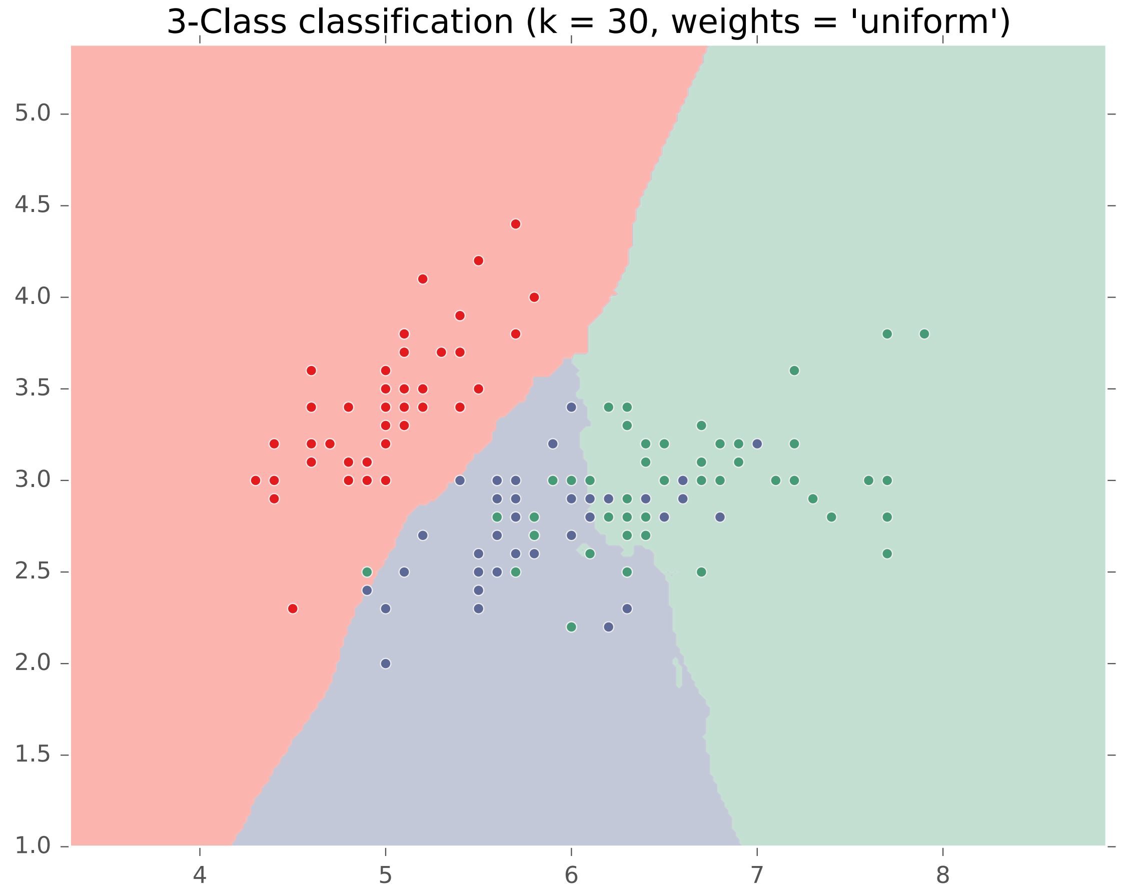


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

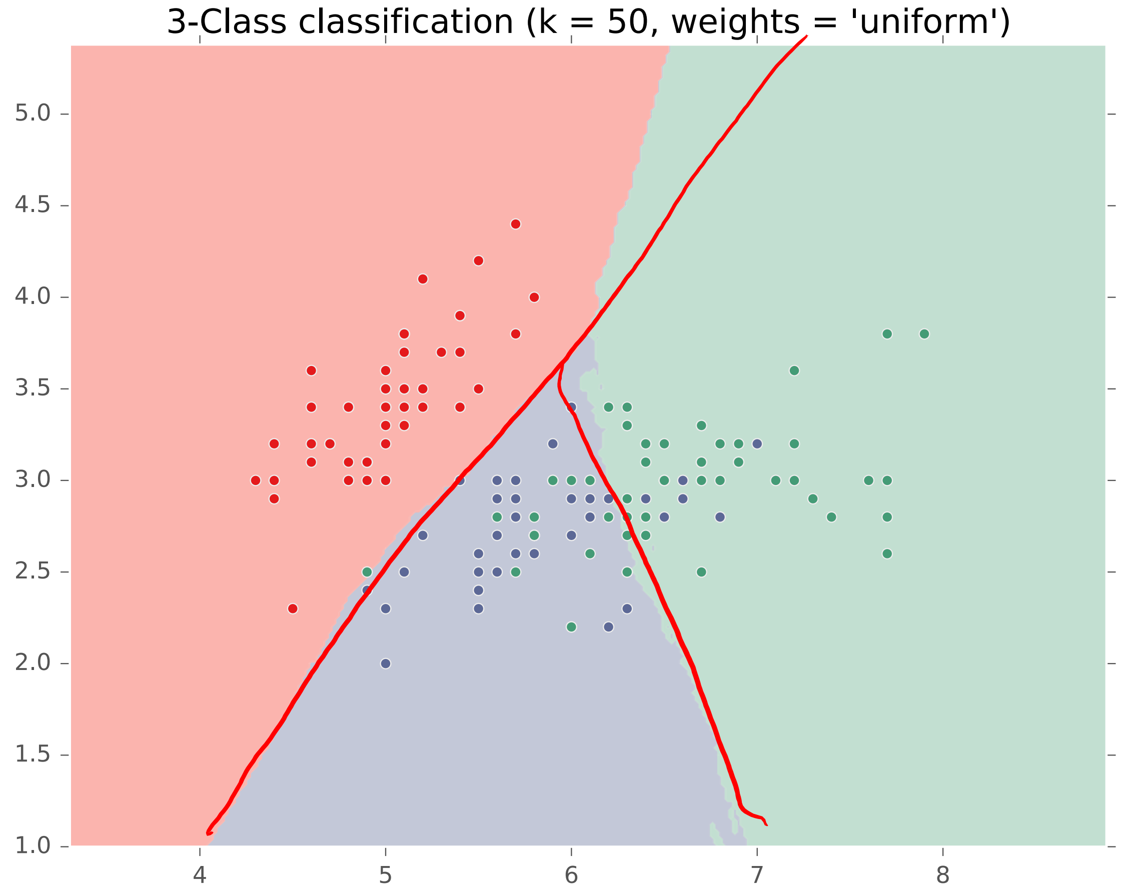


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

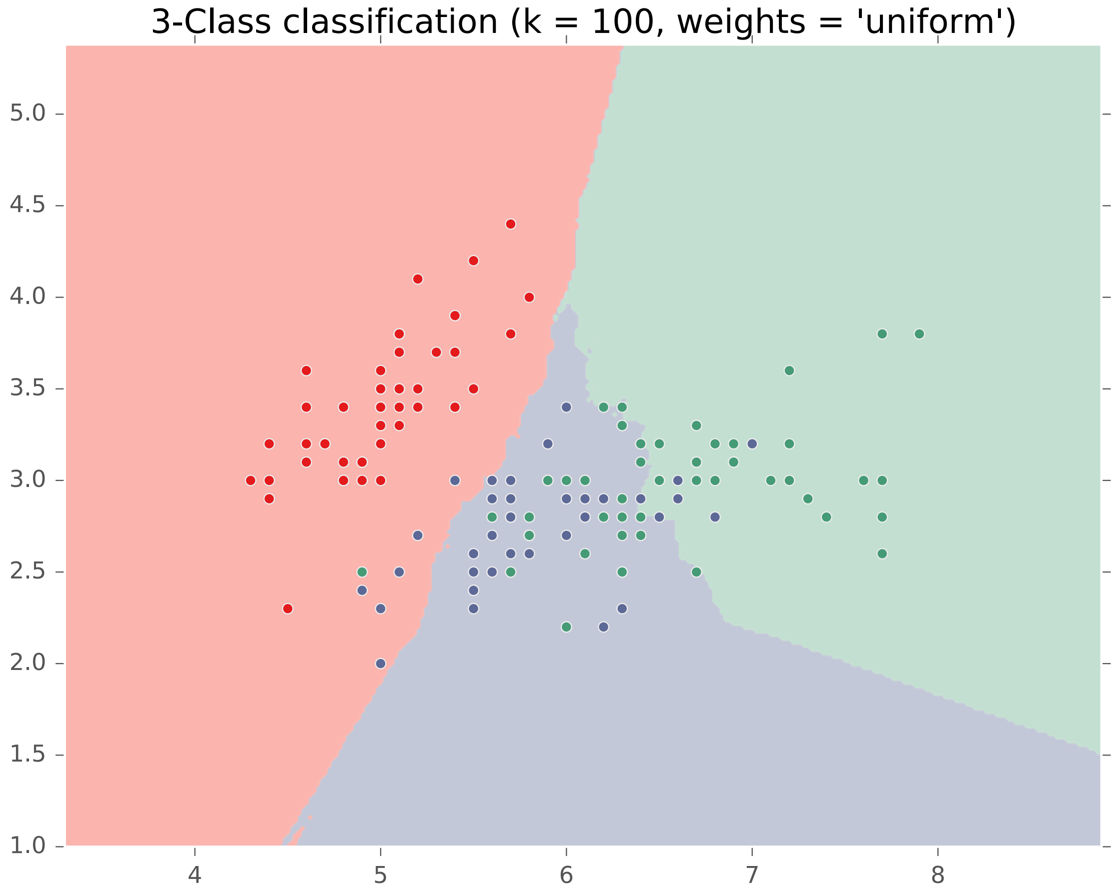


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

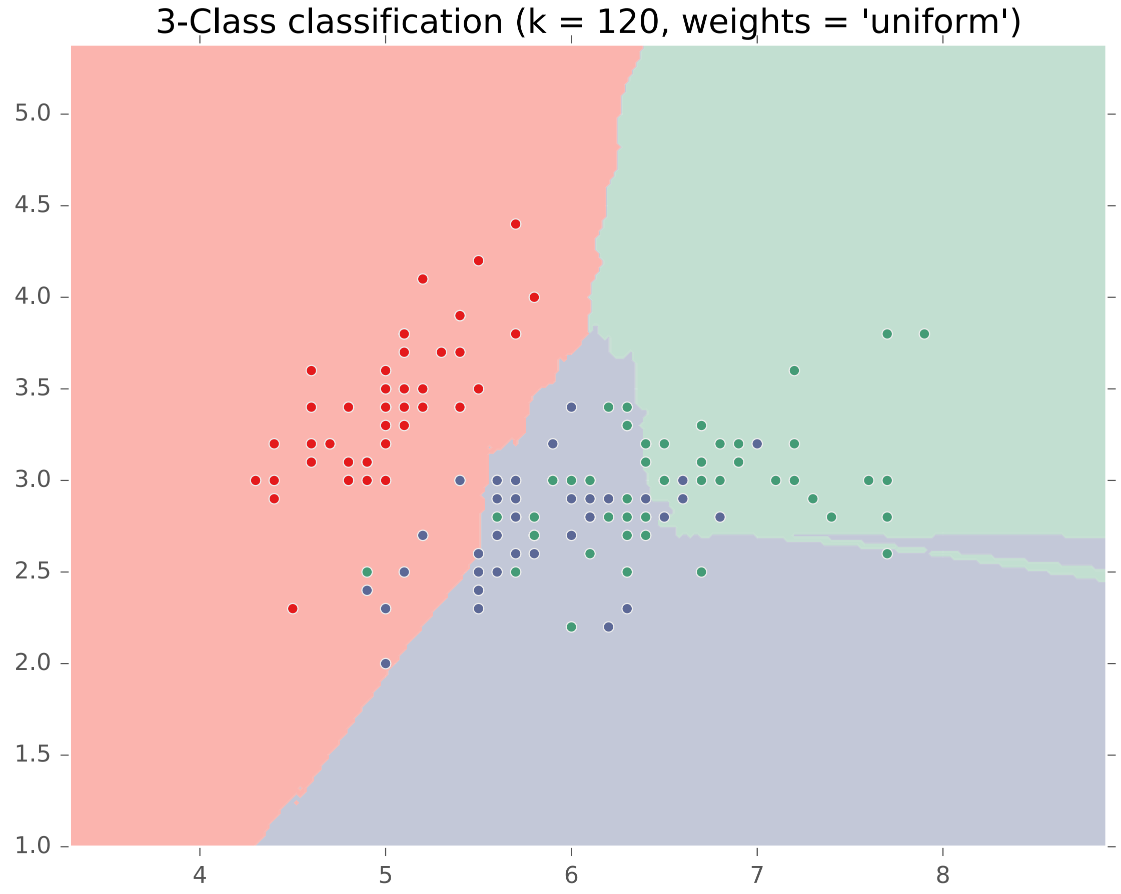


Figure courtesy of Matt Gormley

kNN on Fisher Iris Data

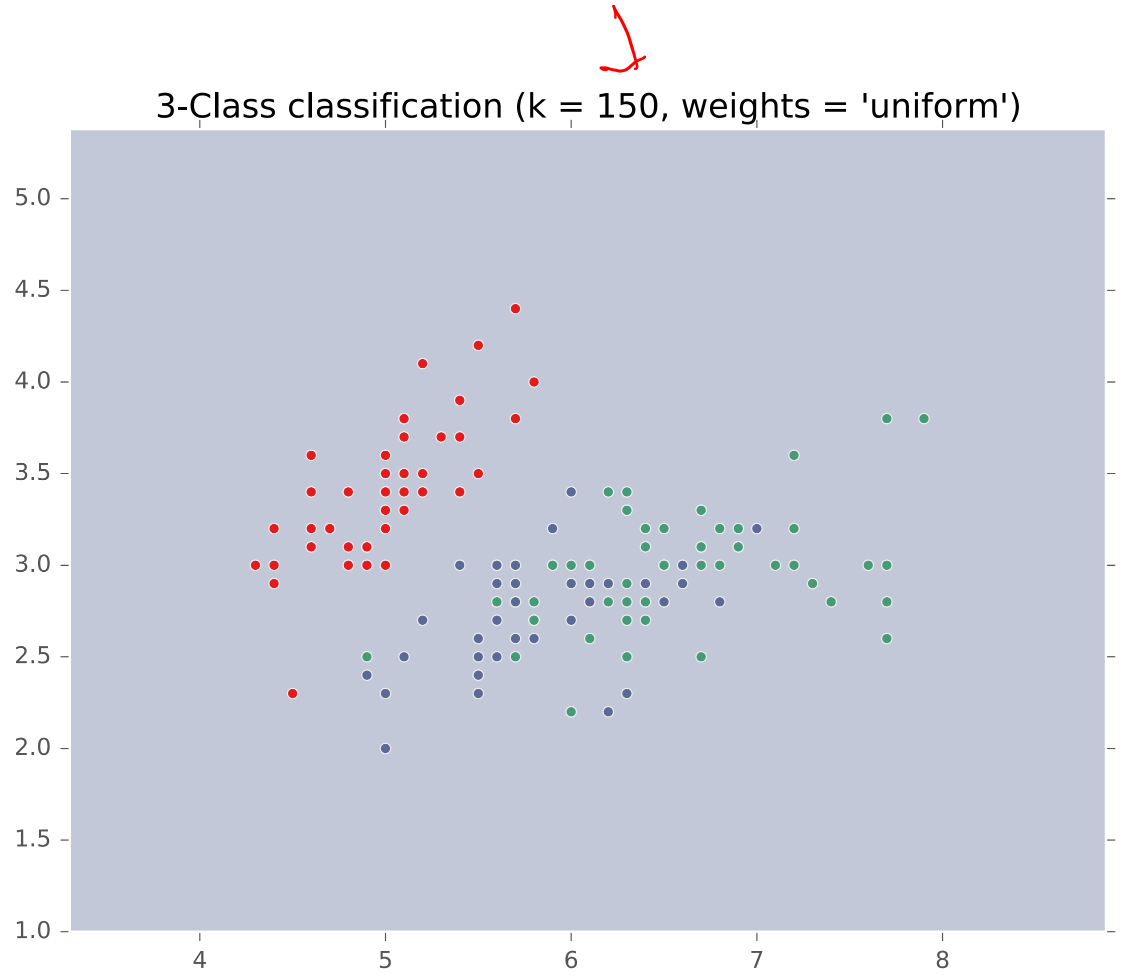


Figure courtesy of Matt Gormley

Aside: kNN and Categorical Features

- kNNs are compatible with categorical features, either by:

1. Converting categorical features into binary ones:

Cholesterol
Normal
Normal
Abnormal

→

Normal Cholesterol?	Abnormal Cholesterol?
1	0
1	0
0	1

2. Using a distance metric that works over categorical features e.g., the Hamming distance:

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

kNN:
Inductive Bias

- What is the inductive bias of a kNN model that uses the Euclidean distance metric?

similar data points behave similarly and all features are equally important

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

Setting k

- Theorem:
  - If k is some function of N s.t. $k(N) \rightarrow \infty$ and $\frac{k(N)}{N} \rightarrow 0$ as $N \rightarrow \infty$ …
  - … then (under certain assumptions) the true error of a kNN model $\rightarrow$ the Bayes error rate
- Practical heuristics:
  - $k = \lfloor\sqrt{N}\rfloor$
  - $k = 3$
- This is a question of **model selection**: each value of k corresponds to a different “model”

Model Selection

- A **model** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one (the ”hypothesis space”)
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

Example: Decision Trees

- Model = set of all possible trees, potentially narrowed down according to the hyperparameters (see below)
- Model parameters = structure of a specific tree e.g., splits, split order, predictions at leaf nodes,
- Hyperparameters = splitting criterion, max-depth, tie-breaking procedures, etc…

Model Selection

- A **model** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one (the ”hypothesis space”)
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

Example: kNN

- Model = set of all possible nearest neighbors classifiers
- Model parameters = none! kNN is a “non-parametric model”
- Hyperparameters = k

distance metric,

tie-breaking ...

Model Selection with Test Sets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$, suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$$

- Learn a classifier from each model using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \ldots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{test}$ and choose the one with lowest test error:

$$\hat{m} = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{test})$$

Model Selection with Test Sets?

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$, suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$$

- Learn a classifier from each model using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \ldots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{test}$ and choose the one with lowest test error:

$$\hat{m} = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{test})$$

- Is $err(h_{\hat{m}}, \mathcal{D}_{test})$ a good estimate of $err(h_{\hat{m}})$?

Model Selection with Validation Sets

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$$

- Learn a classifier from each model using only $\mathcal{D}_{train}$:

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \dots, h_M \in \mathcal{H}_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

Hyperparameter Optimization with Validation Sets

- Given $\mathcal{D} = \boxed{\mathcal{D}_{train} \cup \mathcal{D}_{val}} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\underline{\theta_1}, \underline{\theta_2}, \dots, \underline{\theta_M}$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

Setting k for kNN with Validation Sets

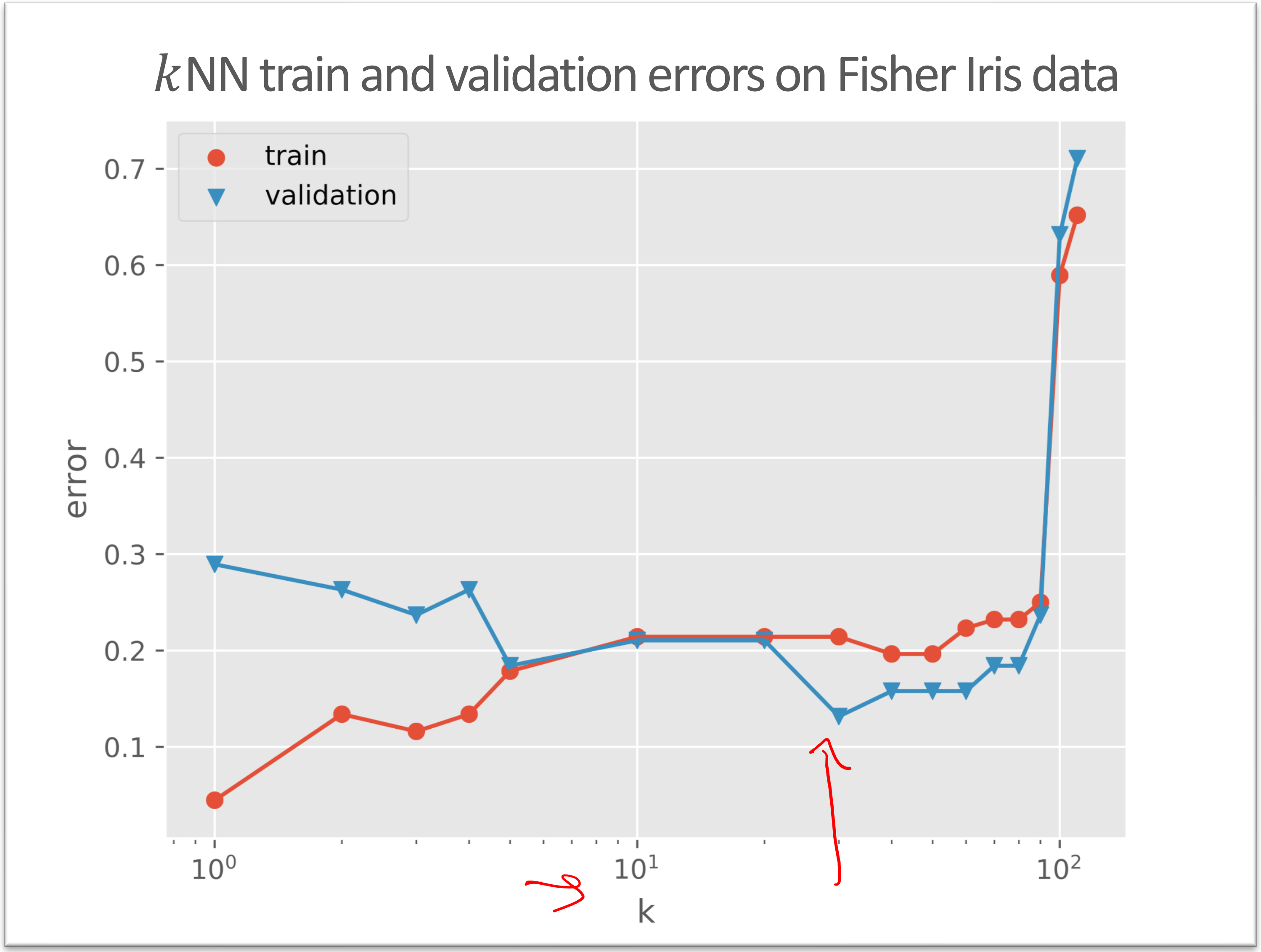


Figure courtesy of Matt Gormley

How should we partition our dataset?

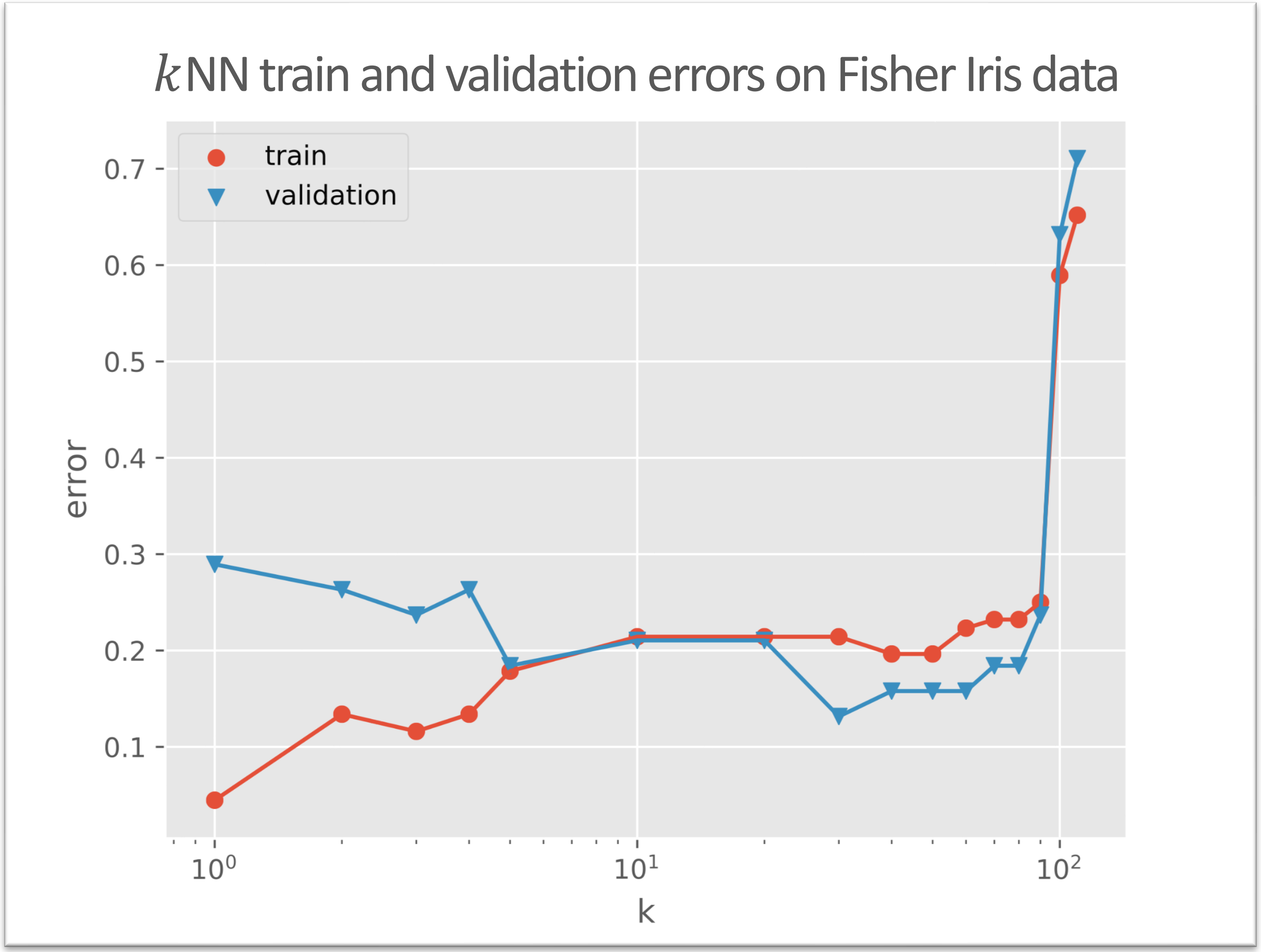
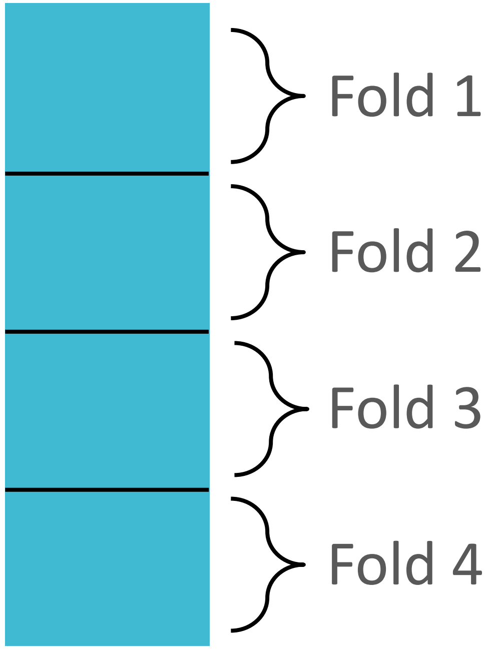


Figure courtesy of Matt Gormley

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

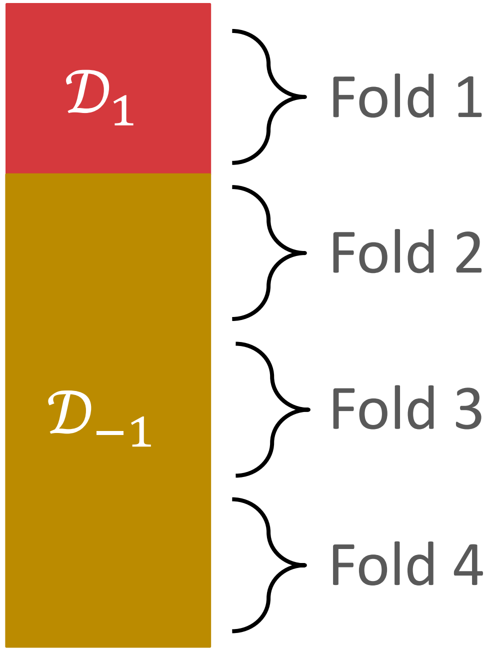


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$
- Use each one as a validation set once:

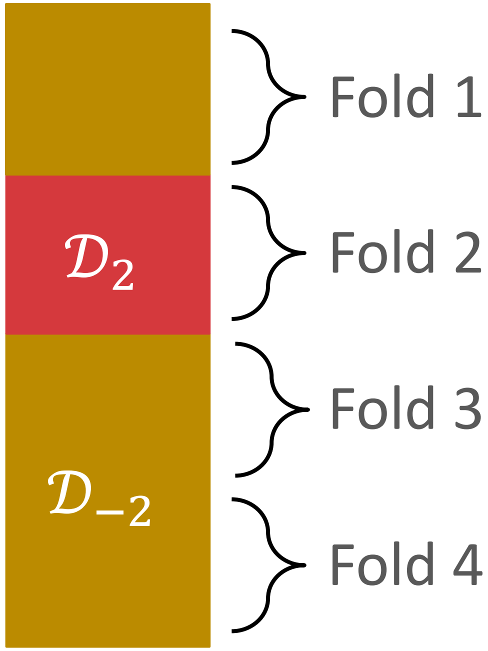


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

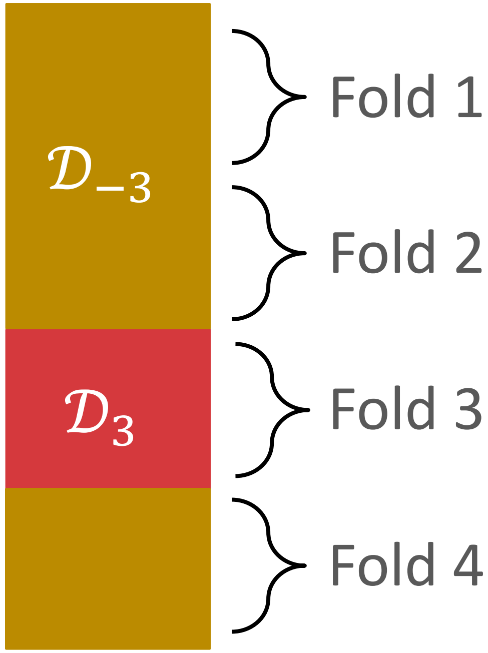


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

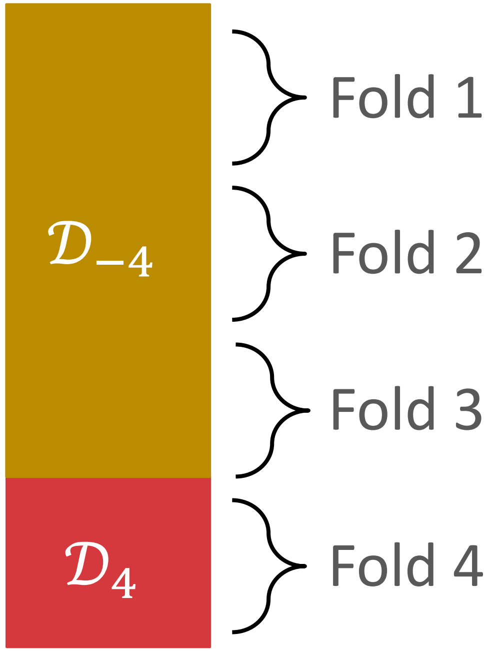


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:

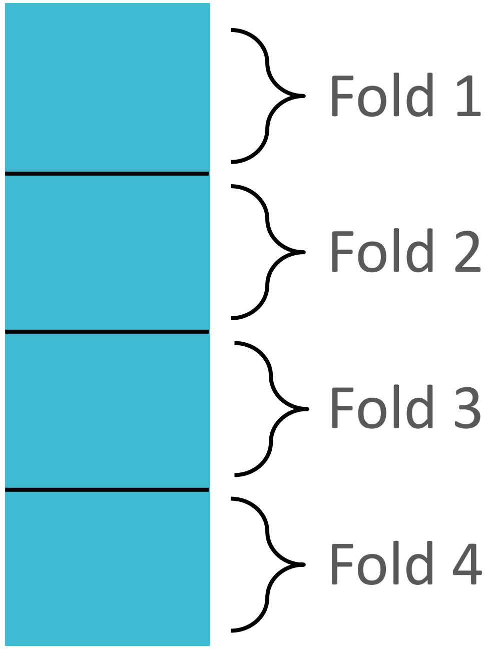


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \underbrace{\frac{1}{K} \sum_{i=1}^{K} e_i}$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$
- Use each one as a validation set once:



- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $\underline{e_i = err(h_{-i}, \mathcal{D}_i)}$
- The K-fold cross validation error is

$$err_{cv_K} = \underbrace{\frac{1}{K} \sum_{i=1}^{K} e_i}$$

- Special case when $K = N$: Leave-one-out cross-validation
- $\boxed{\mathcal{D}_{Test}}$ Choosing between m candidates requires training mK times

Summary

	Input	Output
Training	• training dataset • hyperparameters	• best model parameters
Hyperparameter Optimization	• training dataset • validation dataset	• best hyperparameters
Cross-Validation	• training dataset • validation dataset	• cross-validation error
Testing	• test dataset • classifier	• test error

Hyperparameter Optimization

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Now $err(h_{\hat{m}}^{+}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}}^{+})$!

Pro tip: train your final model using *both* training and validation datasets

- Given $\mathcal{D} = \boxed{\mathcal{D}_{train} \cup \mathcal{D}_{val}} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

- Train a new model on $\mathcal{D}_{train} \cup \mathcal{D}_{val}$ using $\theta_{\hat{m}}$, $h_{\hat{m}}^{+}$
- Now $err(h_{\hat{m}}^{+}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}}^{+})$!

How do we pick hyperparameter settings to try?

- Given $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$, suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only $\mathcal{D}_{train}$:

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using $\mathcal{D}_{val}$ and choose the one with lowest *validation* error:

$$\hat{m} = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; err(h_m, \mathcal{D}_{val})$$

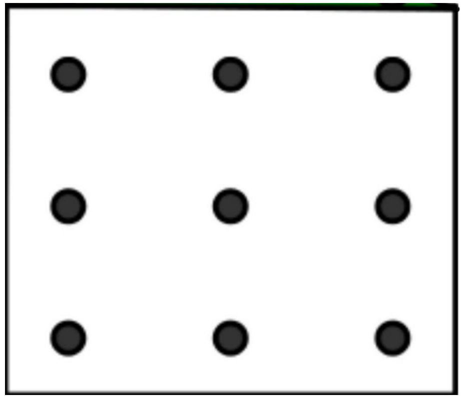
- Train a new model on $\mathcal{D}_{train} \cup \mathcal{D}_{val}$ using $\theta_{\hat{m}}$, $h_{\hat{m}}^{+}$
- Now $err(h_{\hat{m}}^{+}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}}^{+})$!

General Methods for Hyperparameter Optimization

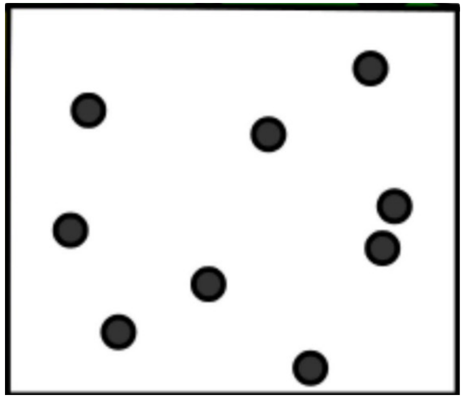
- Idea: set the hyperparameters to optimize some performance metric of the model
- Issue: if we have many hyperparameters that can all take on lots of different values, we might not be able to test all possible combinations
- Commonly used methods:
  - Grid search
  - Random search
  - Bayesian optimization (used by Google DeepMind to optimize the hyperparameters of AlphaGo: https://arxiv.org/pdf/1812.06855v1.pdf)
  - Evolutionary algorithms
  - Graduate-student descent

Grid Search vs. Random Search (Bergstra and Bengio, 2012)

Grid Layout

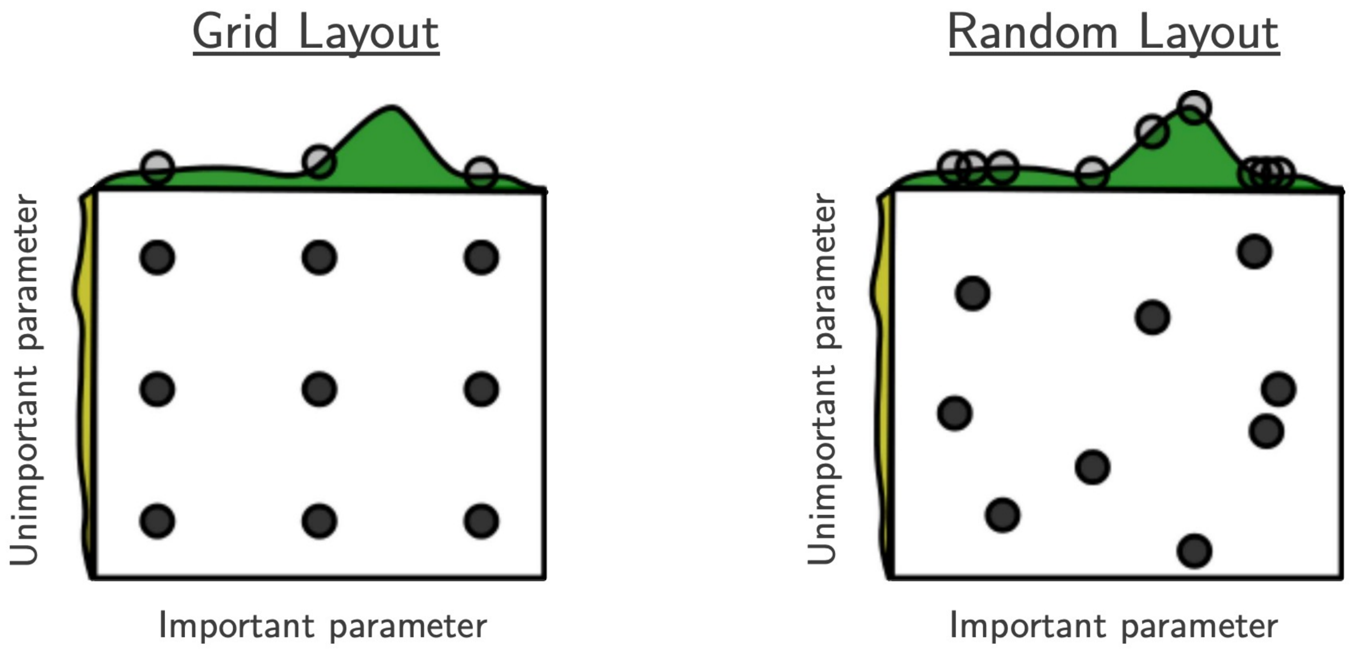


Random Layout



Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Grid Search vs. Random Search (Bergstra and Bengio, 2012)



Grid and random search of nine trials for optimizing a function $f(x, y) = g(x) + h(y) \approx g(x)$ with *low effective dimensionality*. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Key Takeaways

- Real-valued features and decision boundaries
- Nearest neighbor model and generalization guarantees
- kNN "training" and prediction
- Effect of k on model complexity
- kNN inductive bias
- Differences between training, validation and test datasets in the model selection process
- Cross-validation for model selection
- Relationship between training, hyperparameter optimization and model selection