

# Introduction to Machine Learning

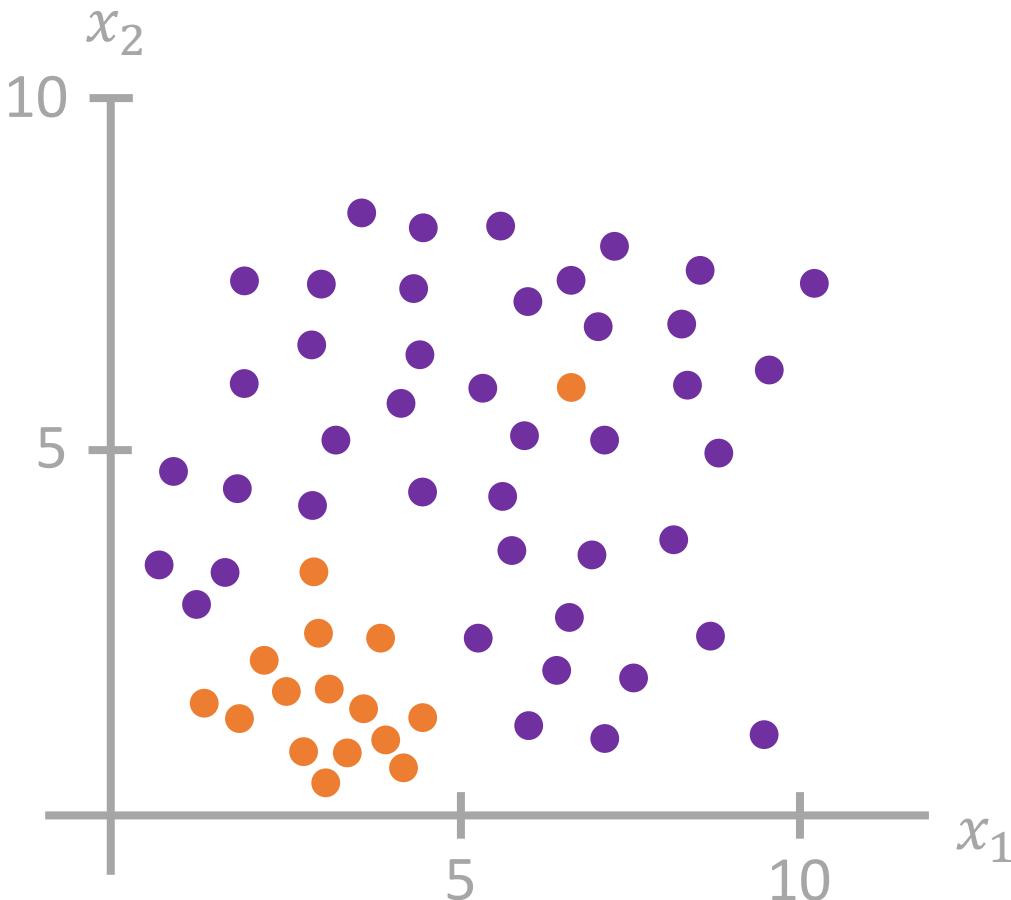
## Nearest Neighbor and Model Selection

Instructor: Pat Virtue

# Warm-up as you walk in

Consider input features  $x \in \mathbb{R}^2$ .

Draw a reasonable decision tree.



# Poll 1

## Decision tree generalization

Which of the following generalize best to unseen examples?

- A. Small tree with low training accuracy
- B. Large tree with low training accuracy
- C. Small tree with high training accuracy
- D. Large tree with high training accuracy

# Poll 1

## Decision tree generalization

Which of the following generalize best to unseen examples?

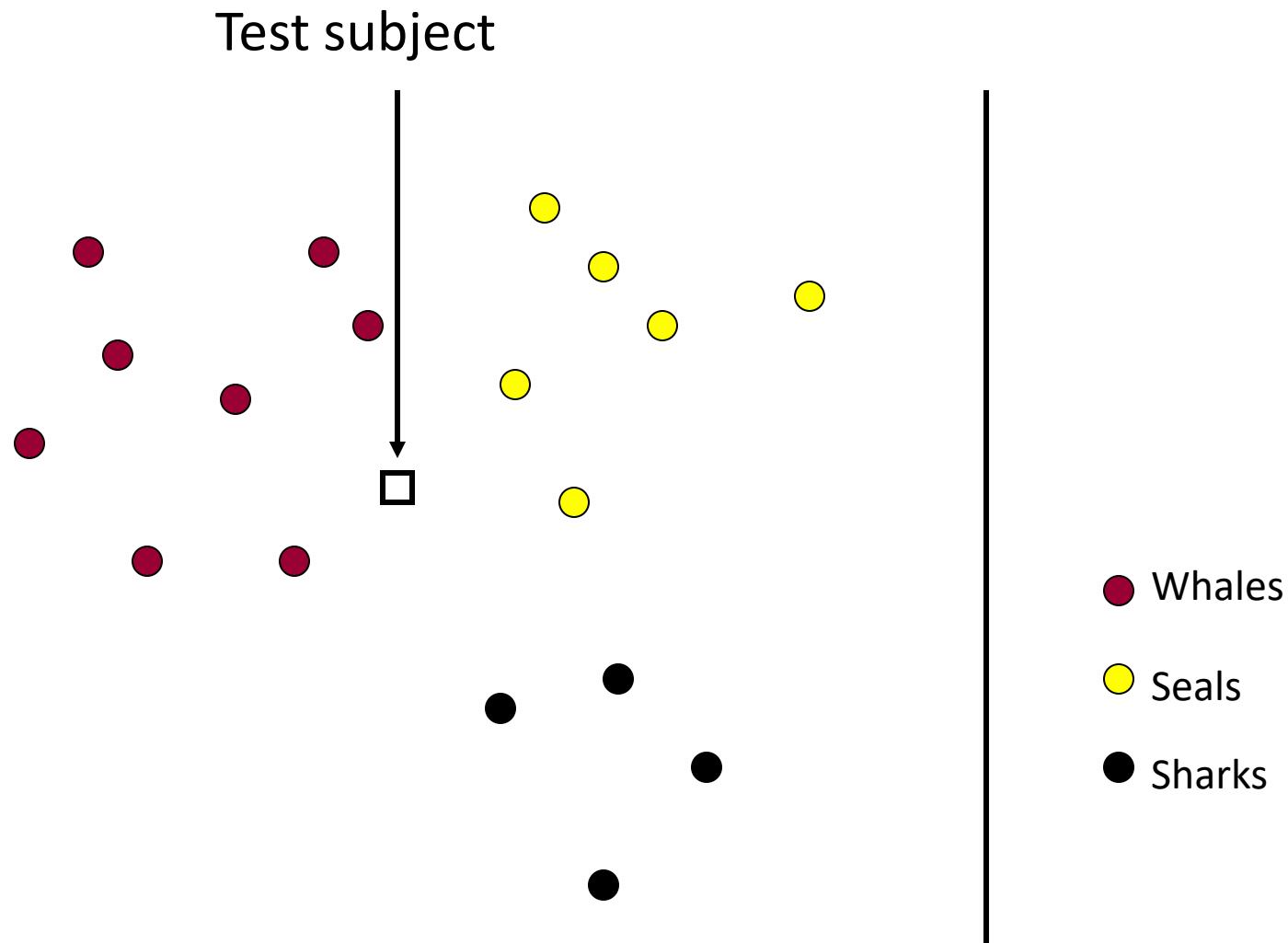
- A. Small tree with low training accuracy
- B. Large tree with low training accuracy
- C. Small tree with high training accuracy
- D. Large tree with high training accuracy

## Poll 2

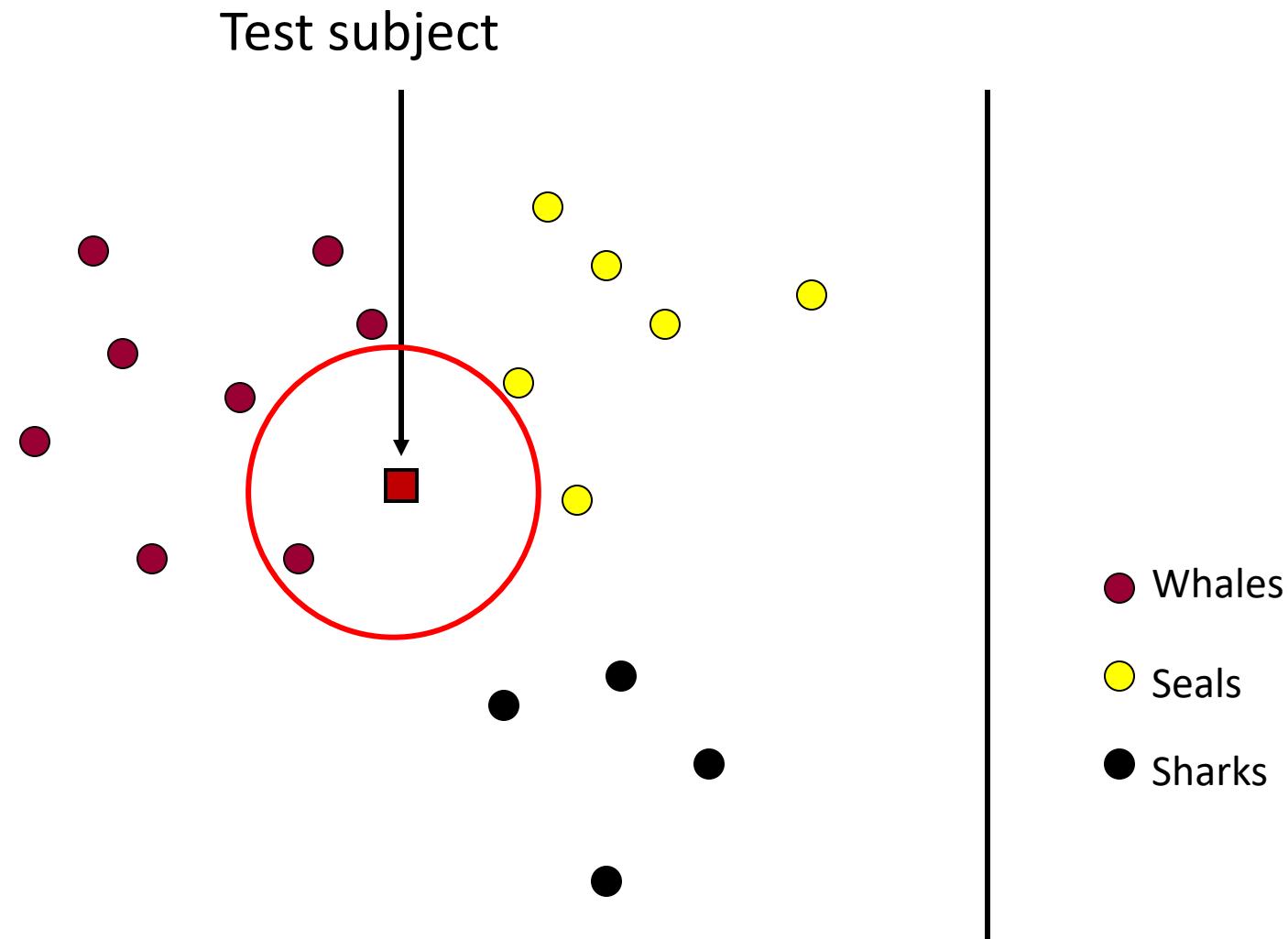
True or False:

For any dataset, you can find a decision tree that can perfectly classify the training data?

# Nearest Neighbor Classifier



# Nearest Neighbor Classifier



# Nearest Neighbor Classification

Given a training dataset  $\mathcal{D} = \{y^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^N$ ,  $y \in \{1, \dots, C\}$ ,  $\mathbf{x} \in \mathbb{R}^M$

and a test input  $\mathbf{x}_{test}$ , predict the class label,  $\hat{y}_{test}$ :

- 1) Find the closest point in the training data to  $\mathbf{x}_{test}$

$$n = \underset{n}{\operatorname{argmin}} d(\mathbf{x}_{test}, \mathbf{x}^{(n)})$$

- 2) Return the class label of that closest point

$$\hat{y}_{test} = y^{(n)}$$

Need distance function! What should  $d(\mathbf{x}, \mathbf{z})$  be?

# 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

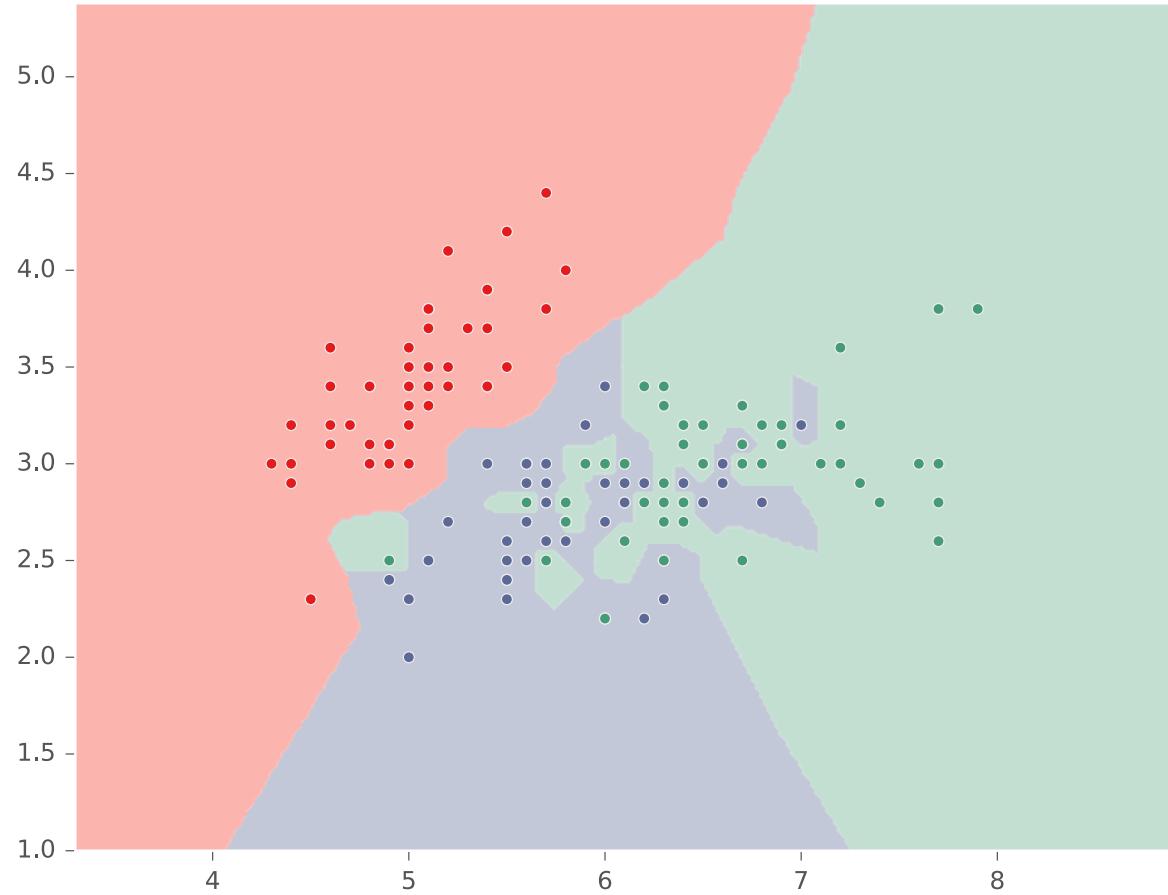
Deleted two of the four features, so that input space is 2D



# Nearest Neighbor on Fisher Iris Data



# Nearest Neighbor on Fisher Iris Data



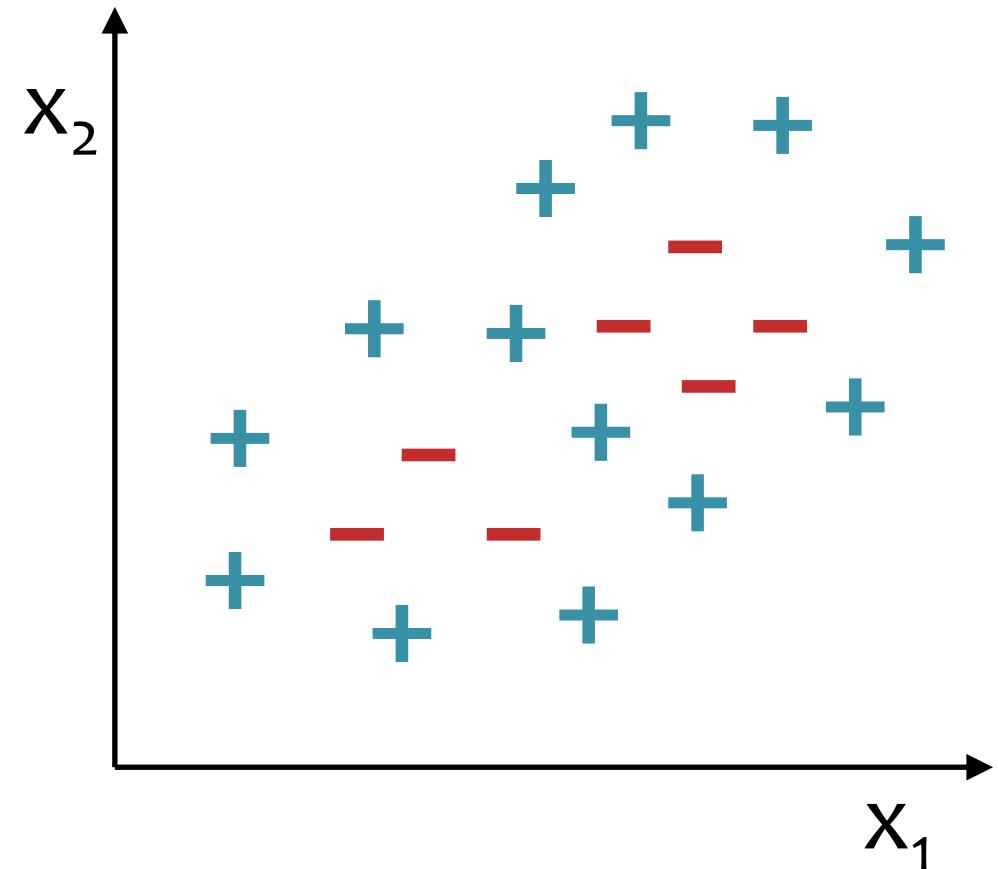
# Poll 3

Which methods can achieve zero training error on this dataset?

- A. Decision trees
- B. 1-Nearest Neighbor
- C. Both
- D. Neither

If zero error, draw the decision boundary.

# Otherwise, why not?



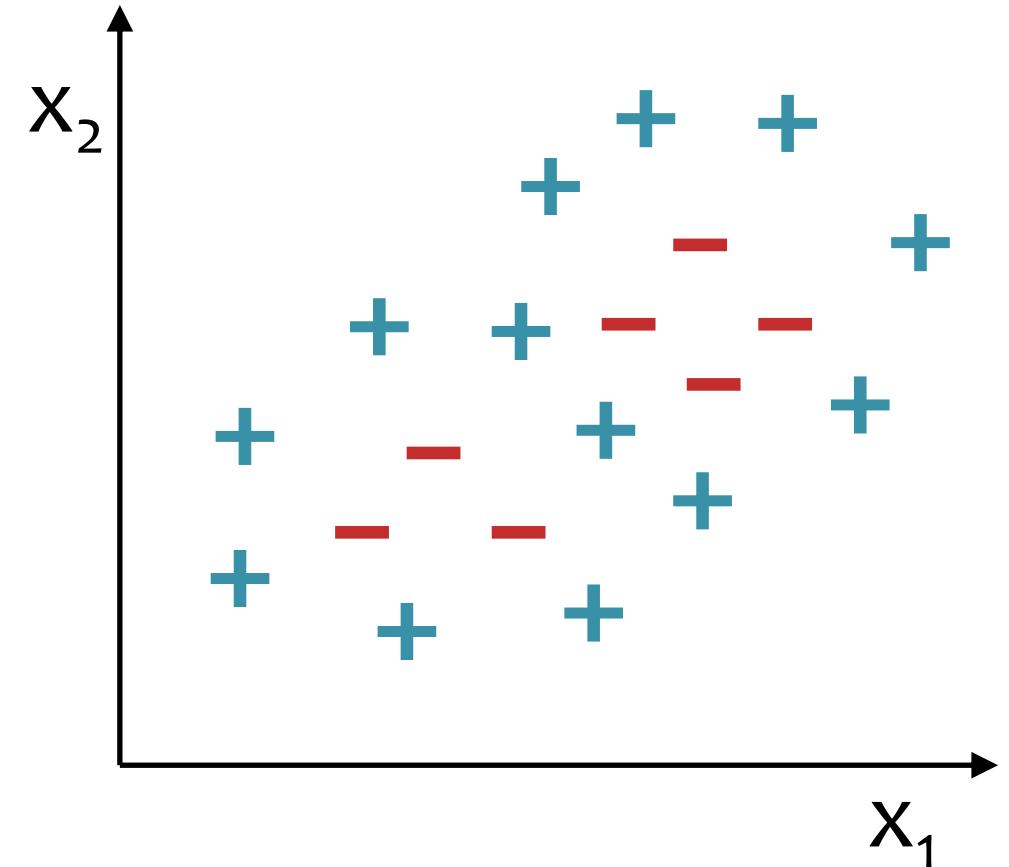
## Poll 3

Which methods can achieve zero training error on this dataset?

- A. Decision trees
- B. 1-Nearest Neighbor
- C. Both
- D. Neither

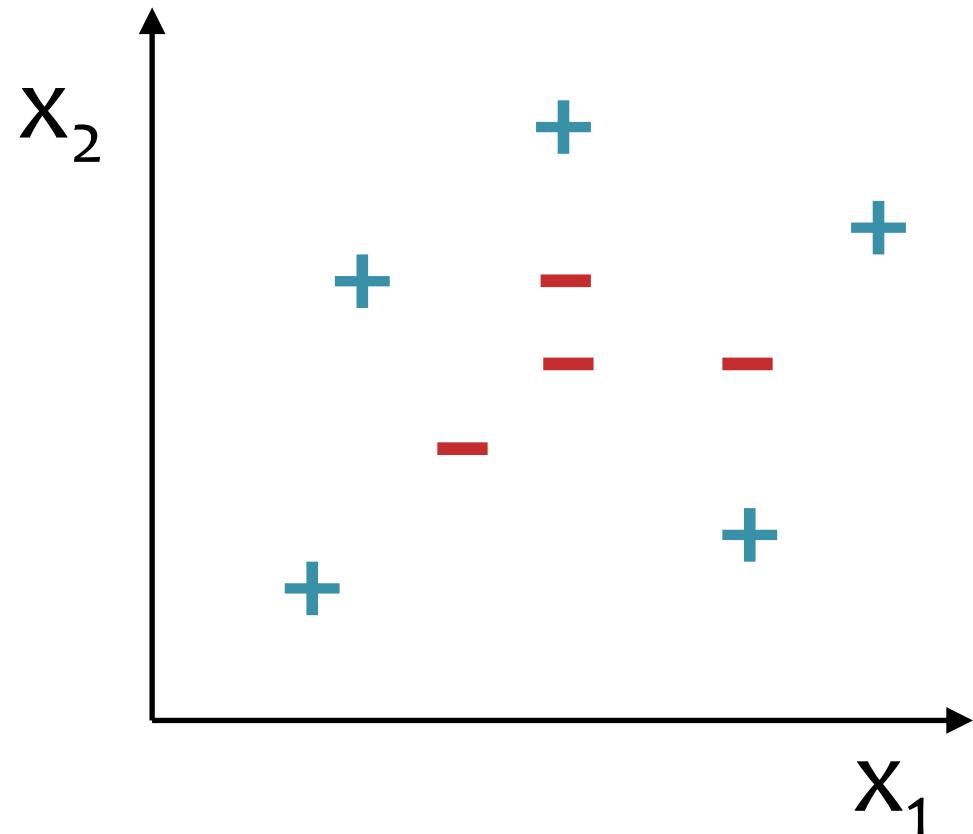
If zero error, draw the decision boundary.

Otherwise, why not?

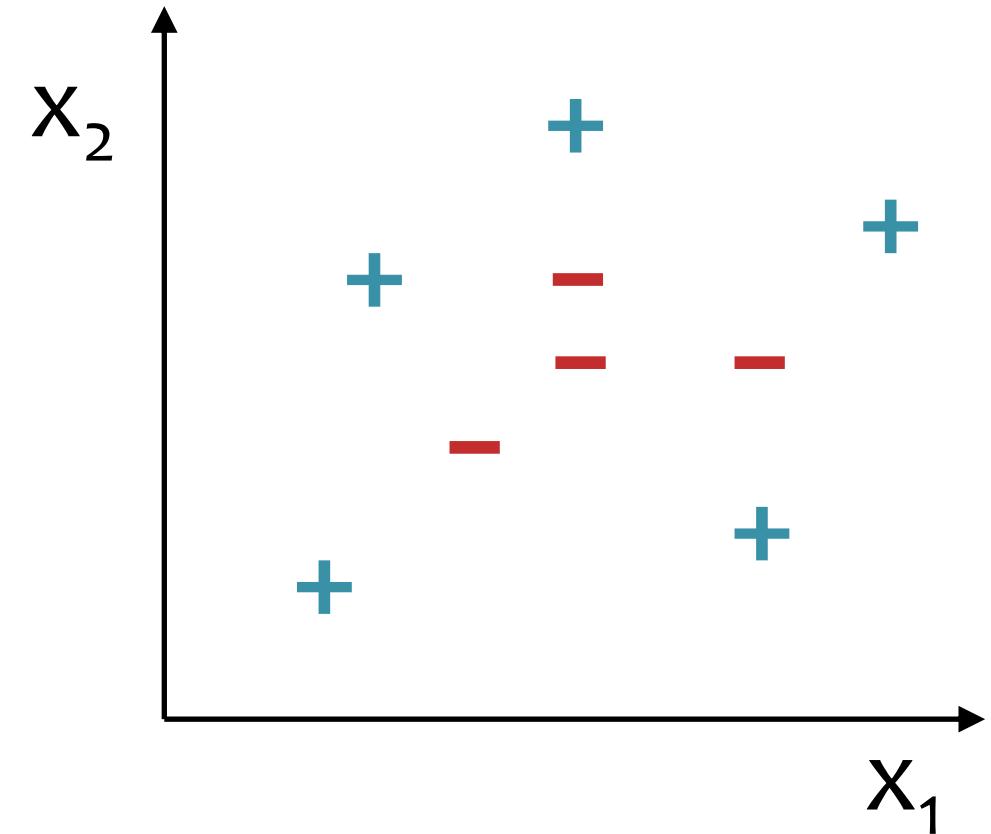


# Decision Boundaries

# Decision tree

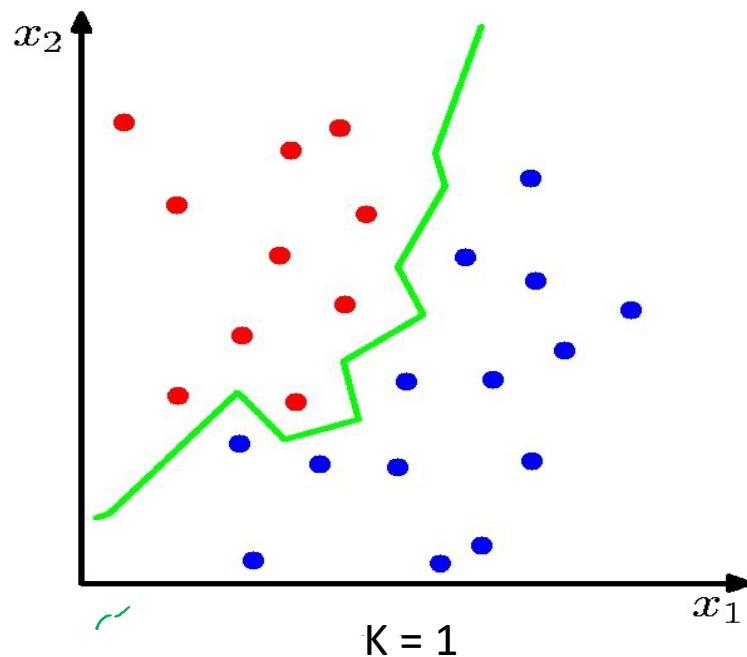


# Nearest neighbor

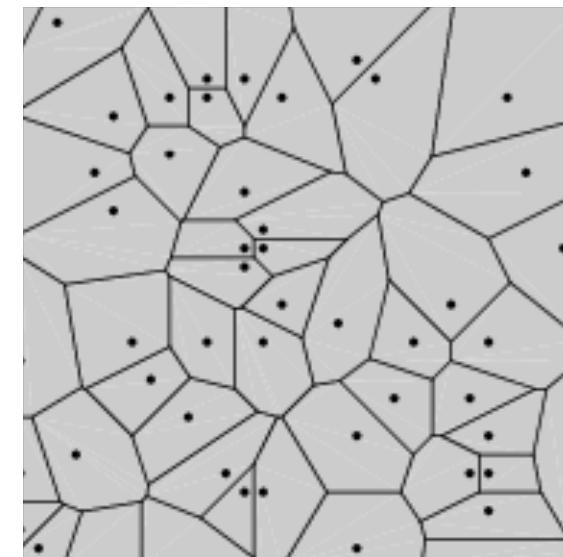


# Nearest Neighbor Decision Boundary

1-nearest neighbor classifier decision boundary



Voronoi Diagram



## Poll 4

1-nearest neighbor will likely:

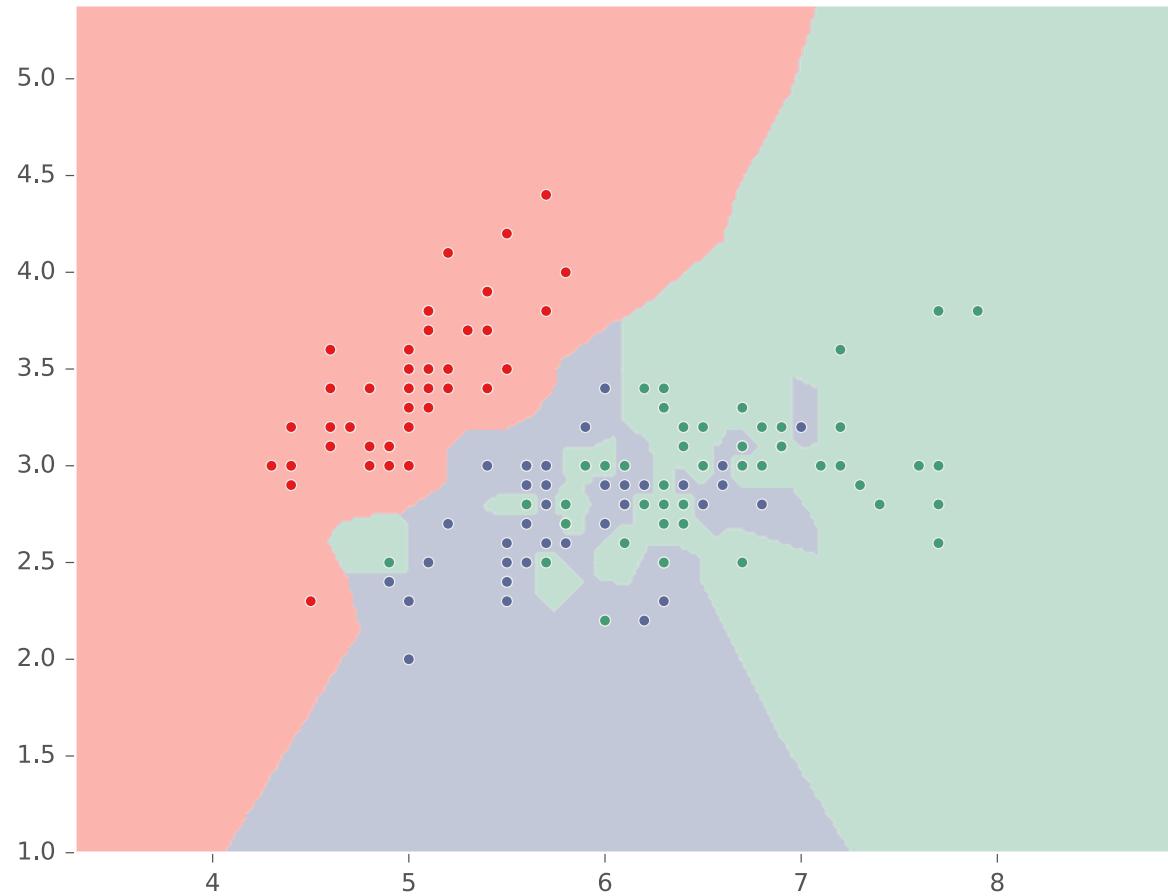
- A. Overfit
- B. Underfit
- C. Neither (it's a great learner!)

## Poll 4

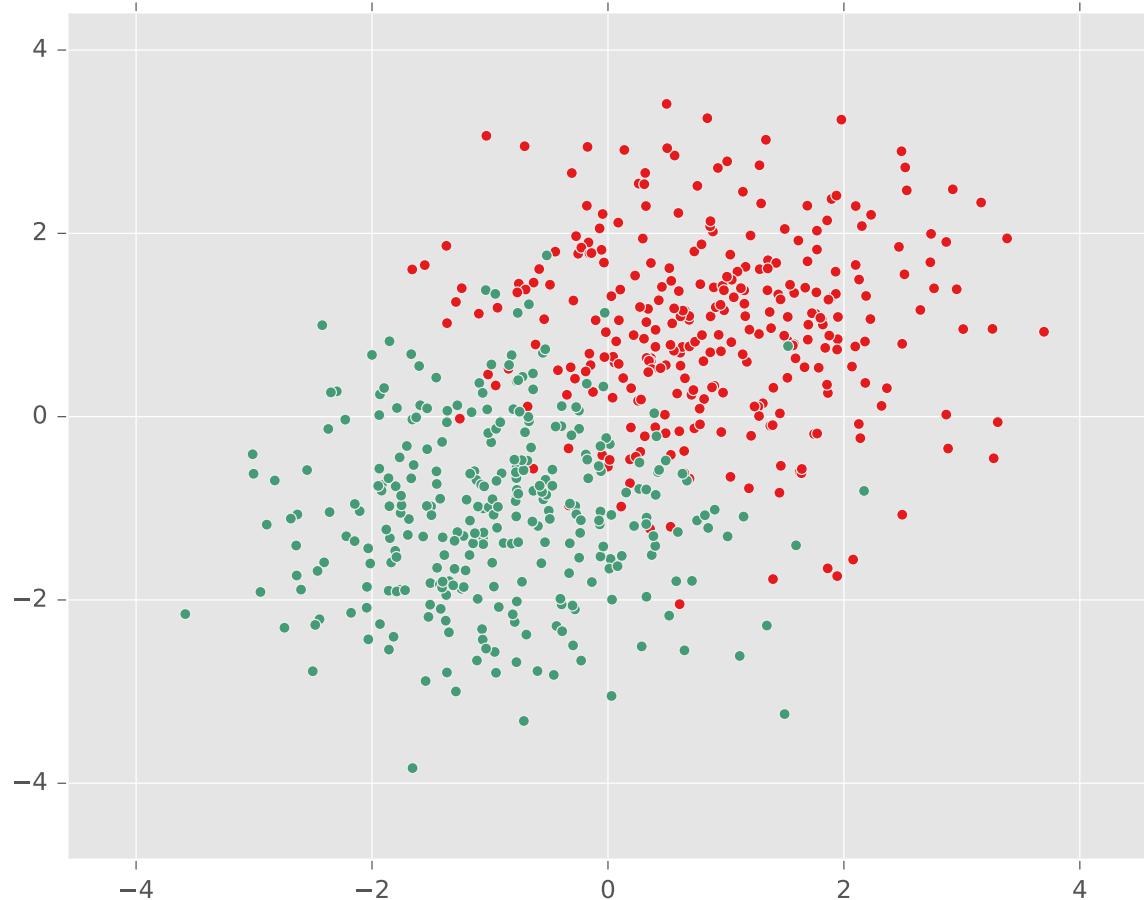
1-Nearest neighbor will likely:

- A. Overfit
- B. Underfit
- C. Neither (it's a great learner!)

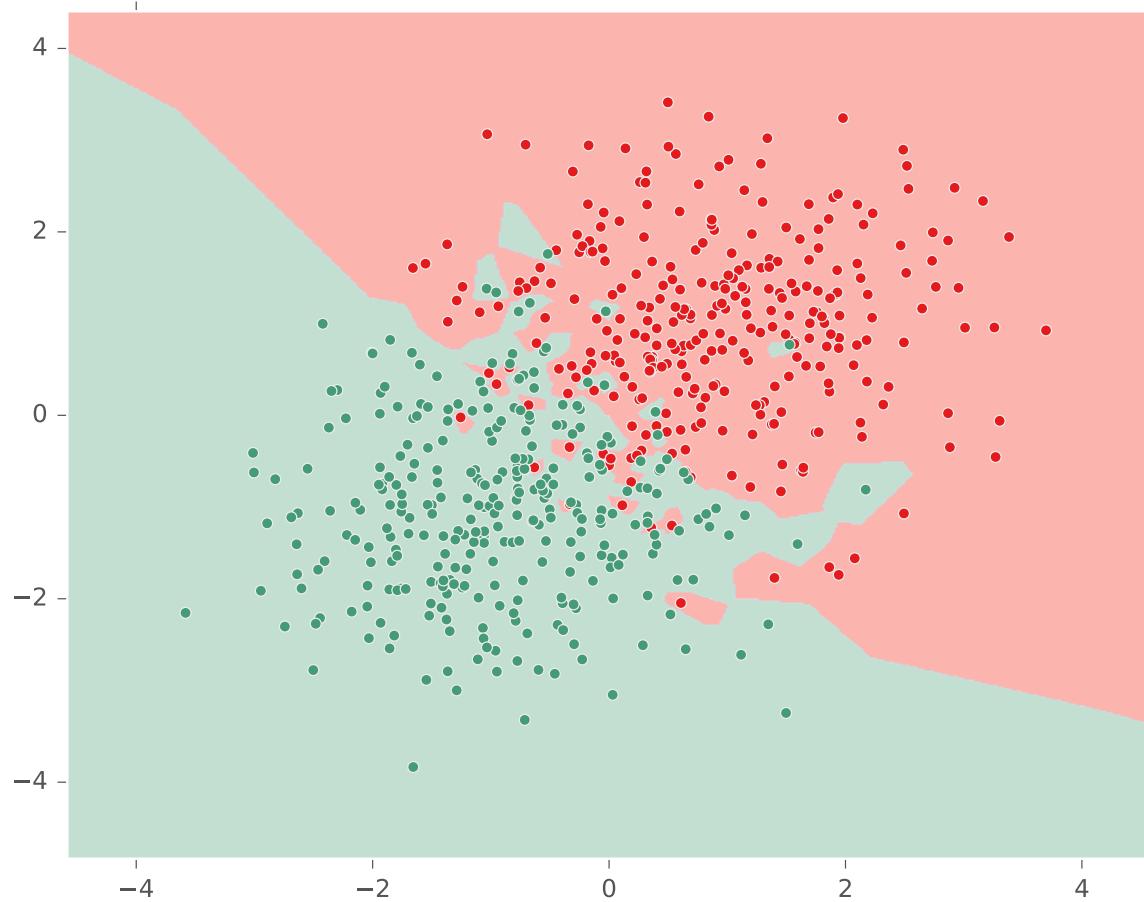
# Nearest Neighbor on Fisher Iris Data



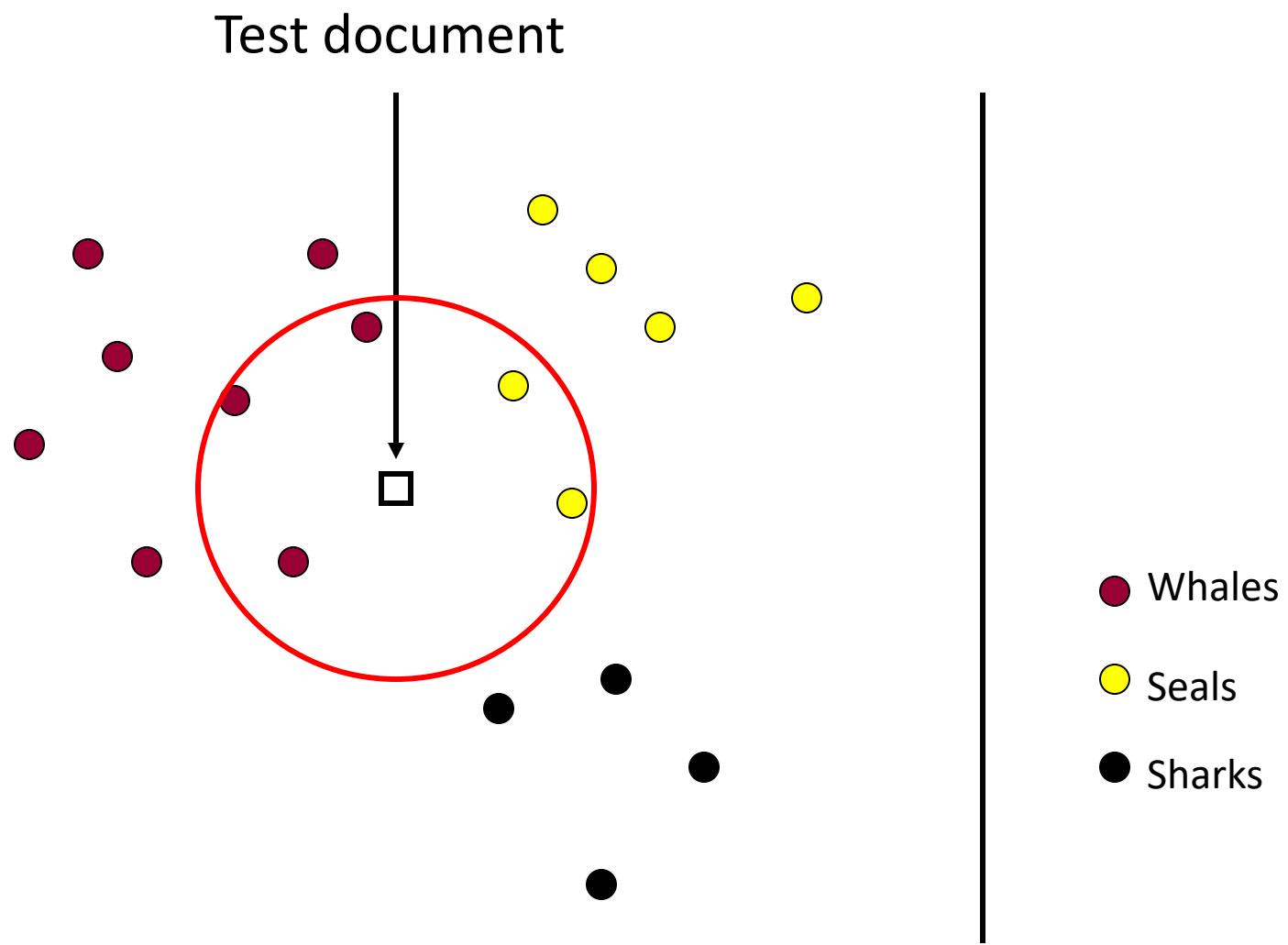
# Nearest Neighbor on Gaussian Data



# Nearest Neighbor on Gaussian Data



# kNN classifier (k=5)



# Nearest Neighbor Classification

Given a training dataset  $\mathcal{D} = \{y^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^N$ ,  $y \in \{1, \dots, C\}$ ,  $\mathbf{x} \in \mathbb{R}^M$

and a test input  $\mathbf{x}_{test}$ , predict the class label,  $\hat{y}_{test}$ :

- 1) Find the closest point in the training data to  $\mathbf{x}_{test}$

$$n = \underset{n}{\operatorname{argmin}} d(\mathbf{x}_{test}, \mathbf{x}^{(n)})$$

- 2) Return the class label of that closest point

$$\hat{y}_{test} = y^{(n)}$$

# $k$ -Nearest Neighbor Classification

Given a training dataset  $\mathcal{D} = \{y^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^N$ ,  $y \in \{1, \dots, C\}$ ,  $\mathbf{x} \in \mathbb{R}^M$

and a test input  $\mathbf{x}_{test}$ , predict the class label,  $\hat{y}_{test}$ :

- 1) Find the closest  $k$  points in the training data to  $\mathbf{x}_{test}$

$$\mathcal{N}_k(\mathbf{x}_{test}, \mathcal{D})$$

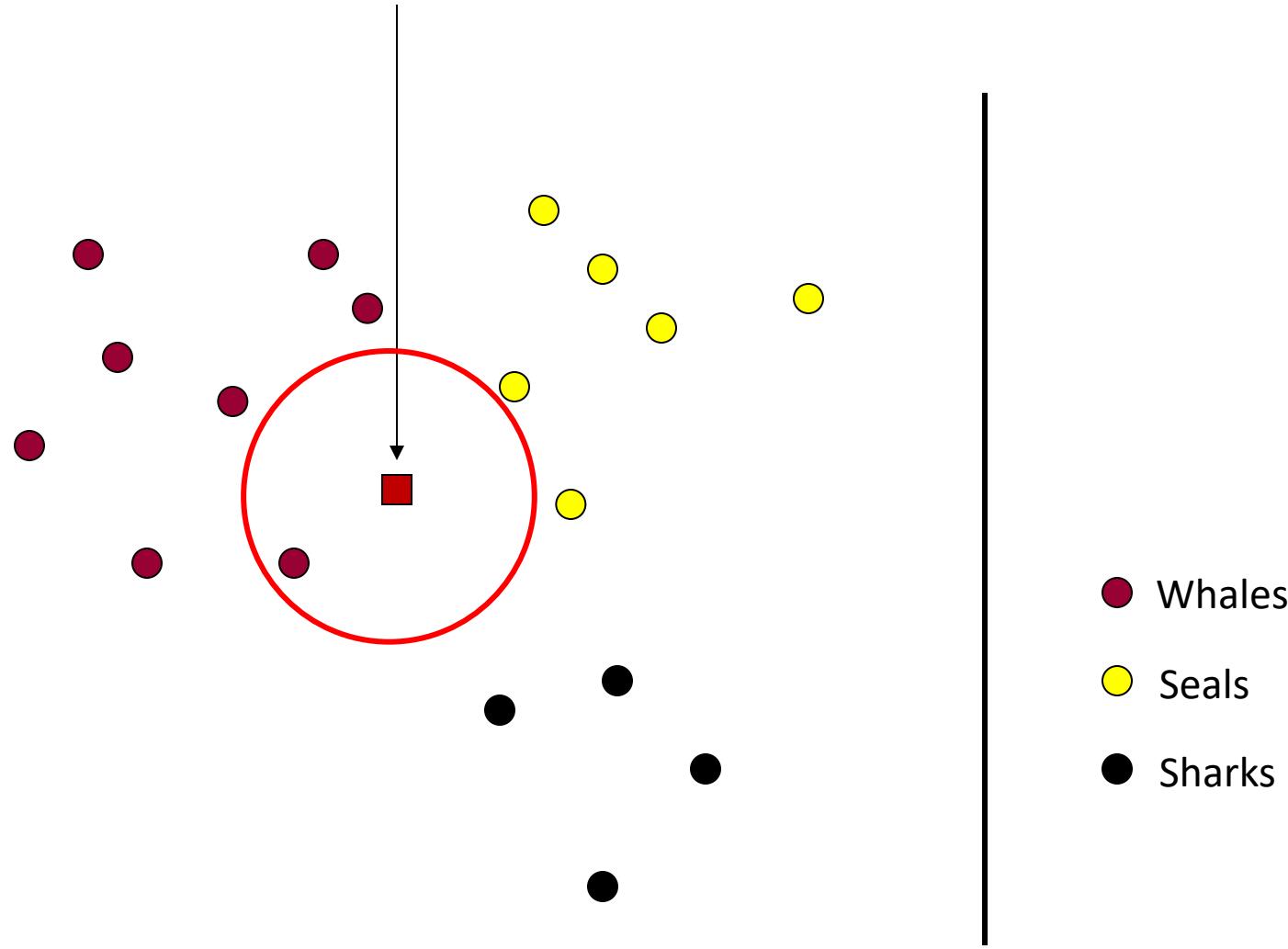
- 2) Return the class label of that closest point

$$\begin{aligned}\hat{y}_{test} &= \operatorname{argmax}_c p(Y = c \mid \mathbf{x}_{test}, \mathcal{D}, k) \\ &= \operatorname{argmax}_c \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x}_{test}, \mathcal{D})} \mathbb{I}(y^{(i)} = c)\end{aligned}$$

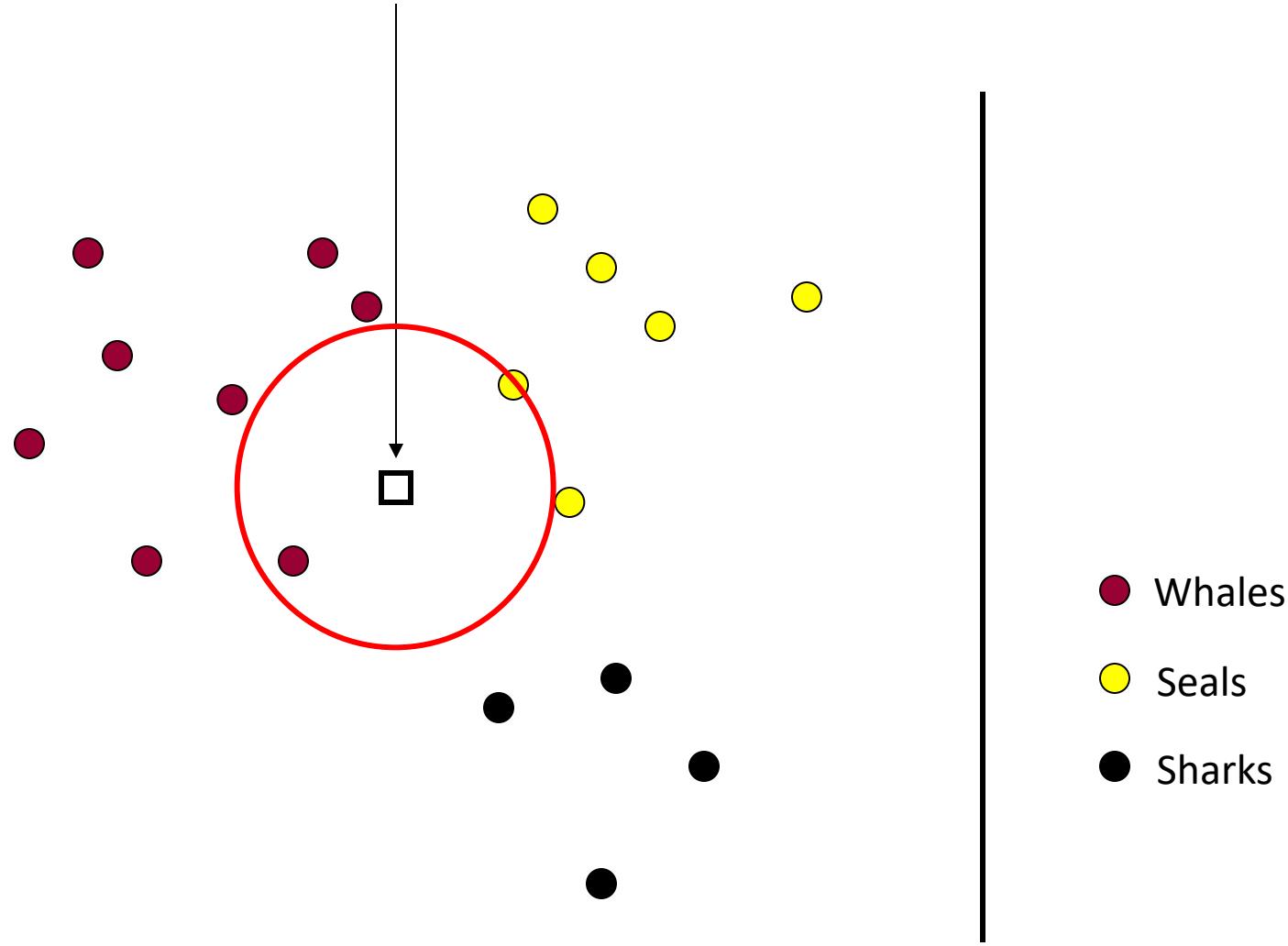
$$= \operatorname{argmax}_c \frac{k_c}{k},$$

where  $k_c$  is the number of the  $k$ -neighbors with class label  $c$

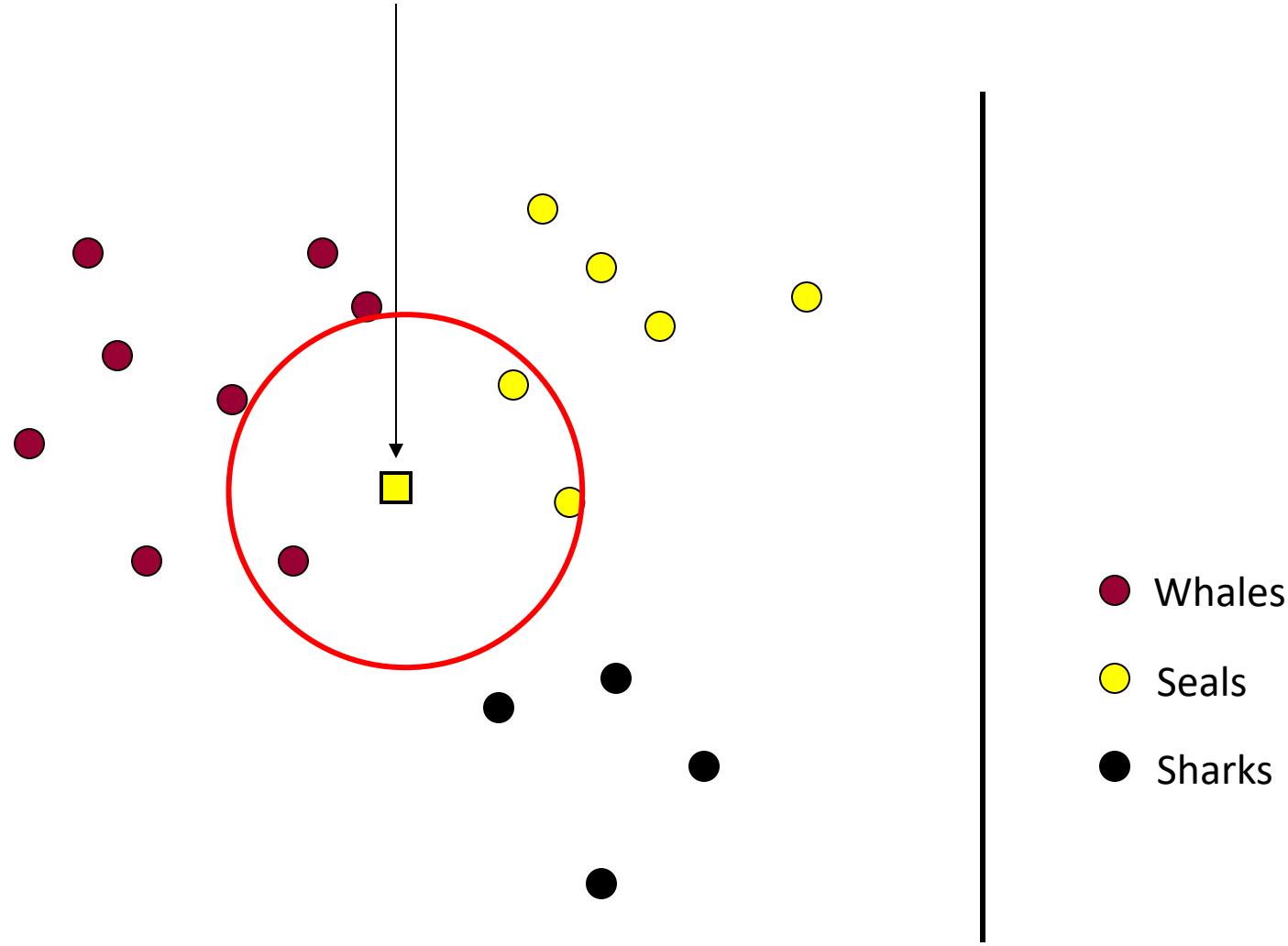
# 1-Nearest Neighbor (kNN) classifier



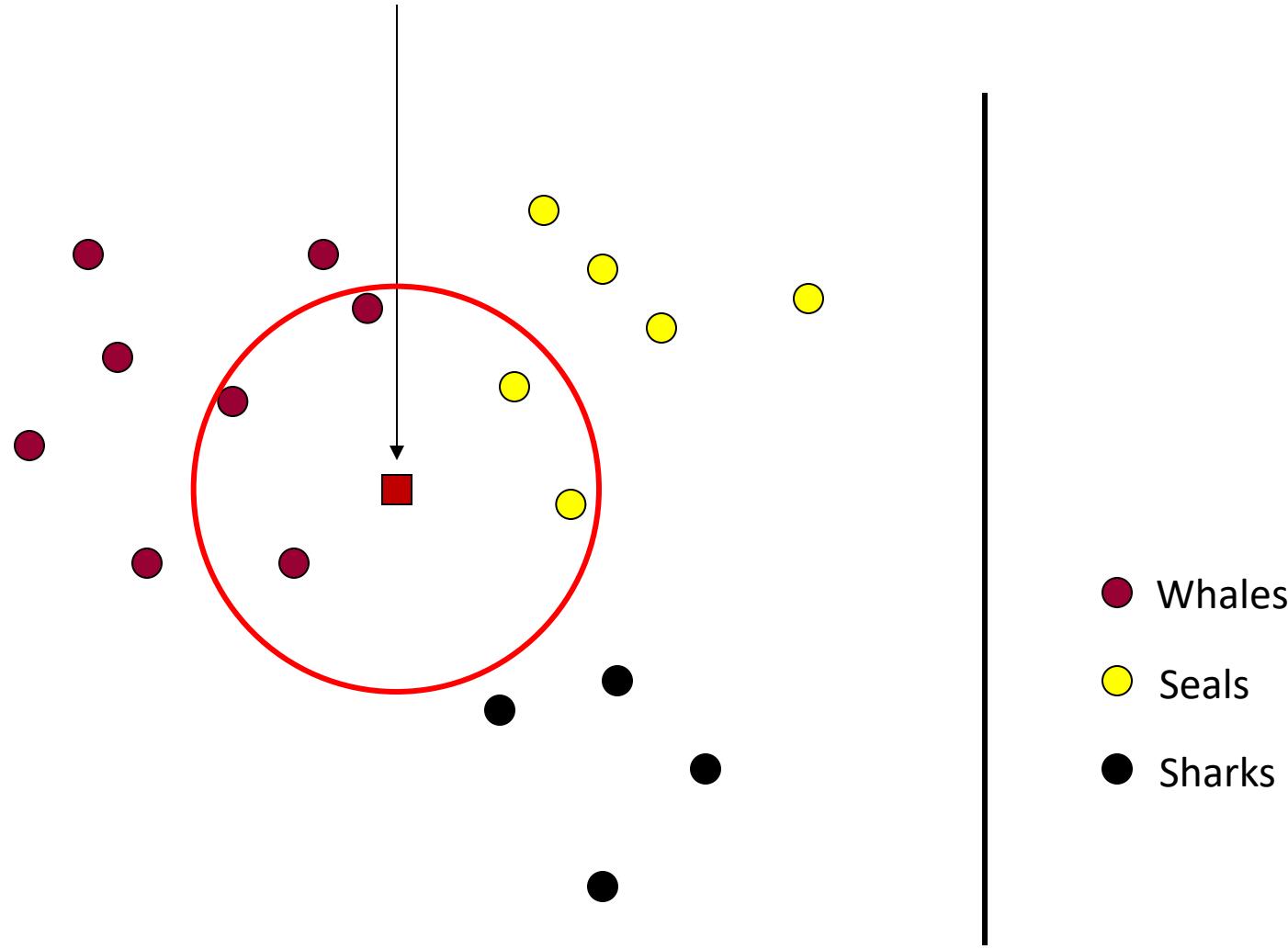
# 2-Nearest Neighbor (kNN) classifier



# 3-Nearest Neighbor (kNN) classifier



# 5-Nearest Neighbor (kNN) classifier



# What is the best $k$ ?

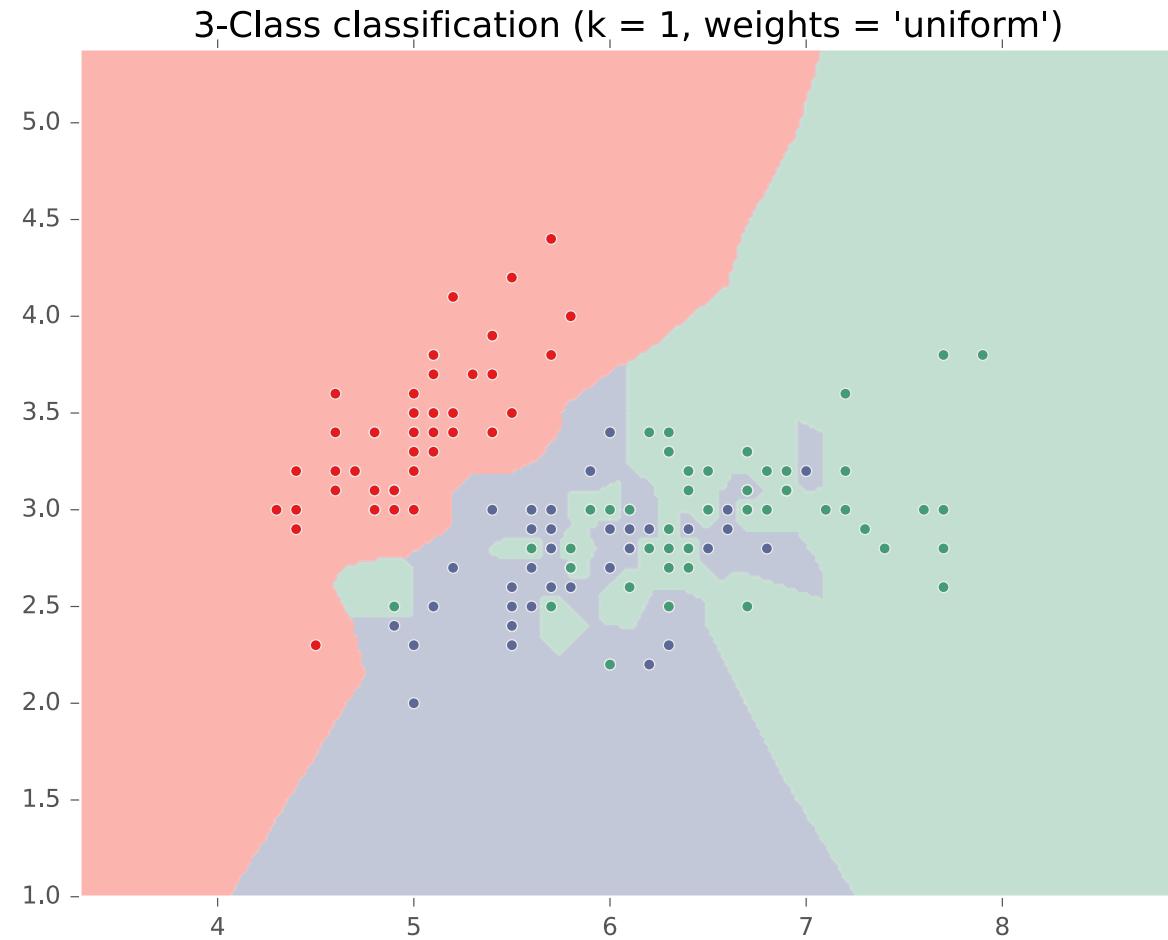
How do we choose a learner that is accurate and also generalizes to unseen data?

- Larger  $k \rightarrow$  predicted label is more stable
- Smaller  $k \rightarrow$  predicted label is more affected by individual training points

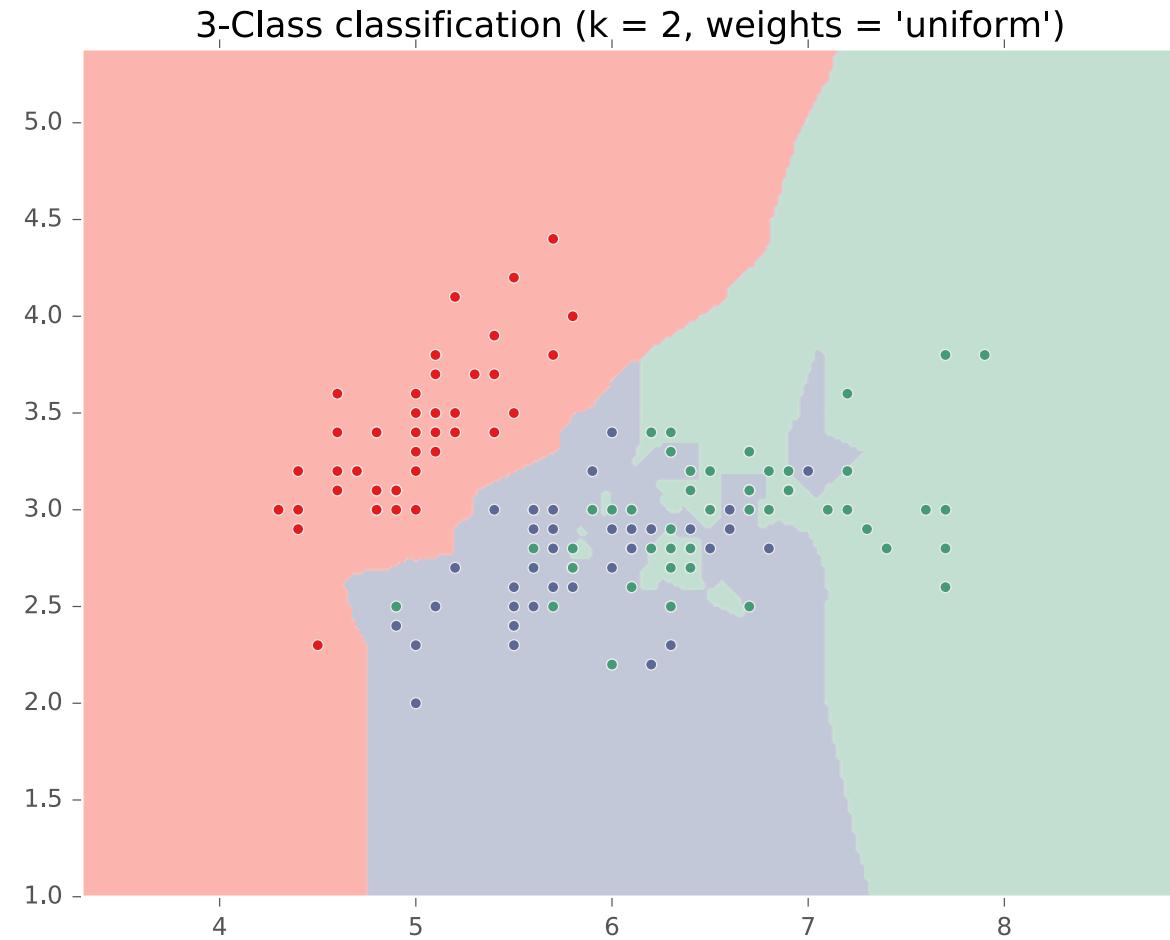
But how to choose  $k$ ?

# k-NN on Fisher Iris Data

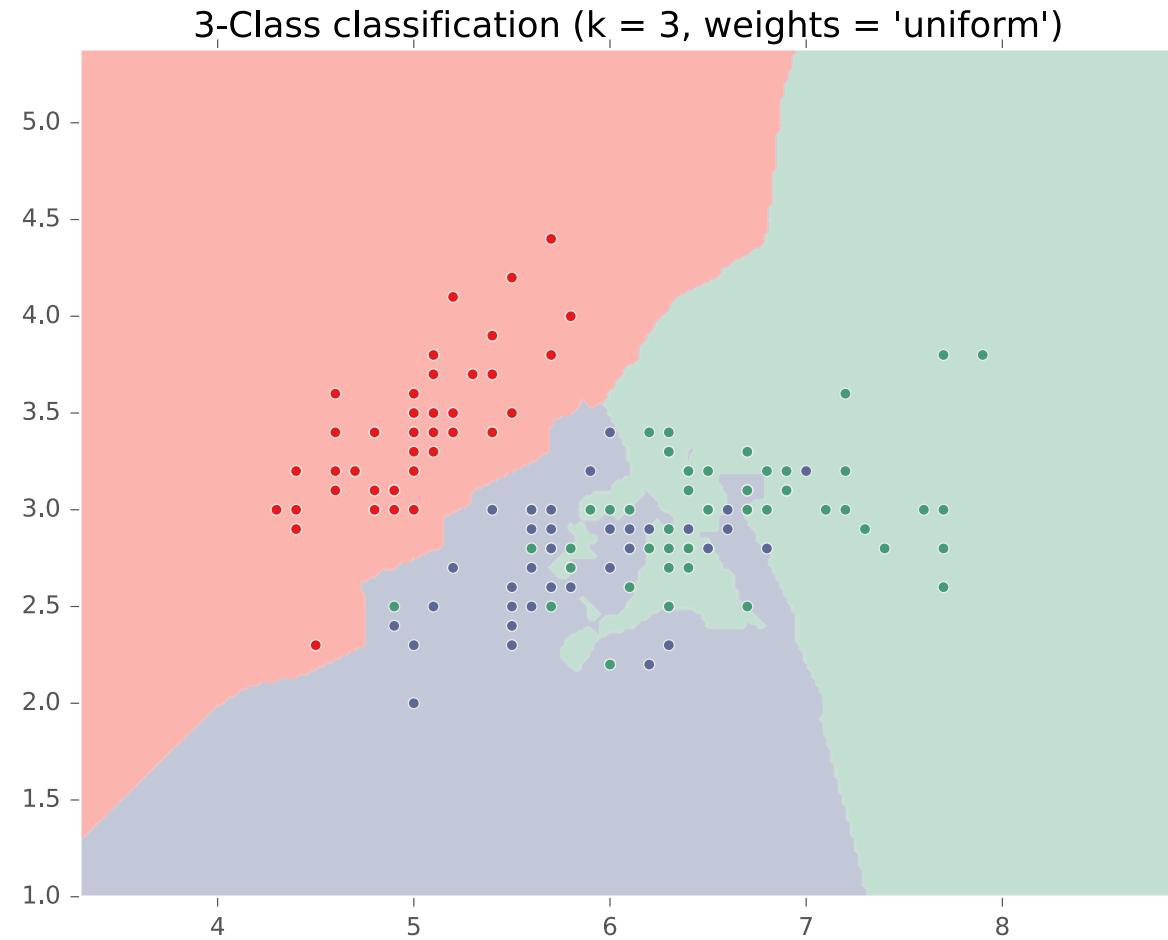
## Special Case: Nearest Neighbor



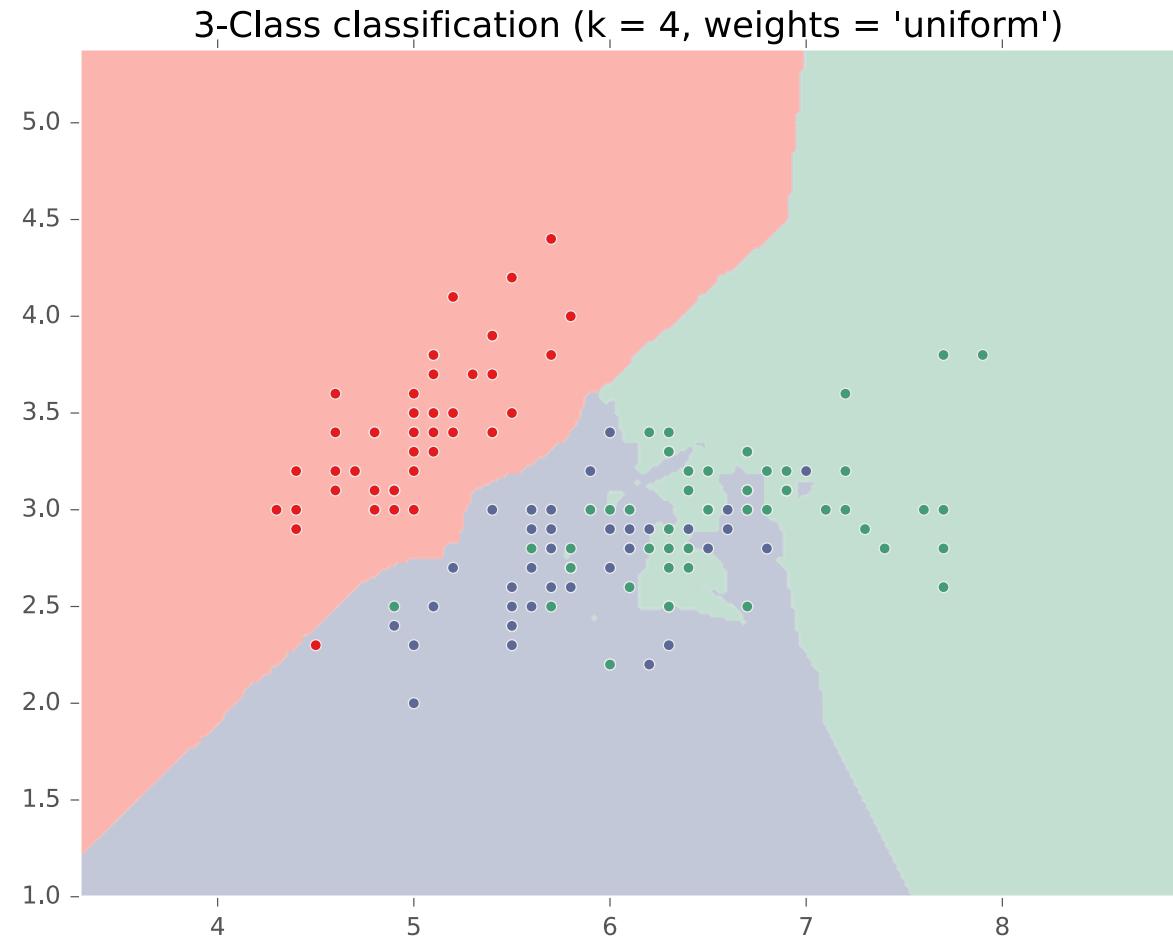
# k-NN on Fisher Iris Data



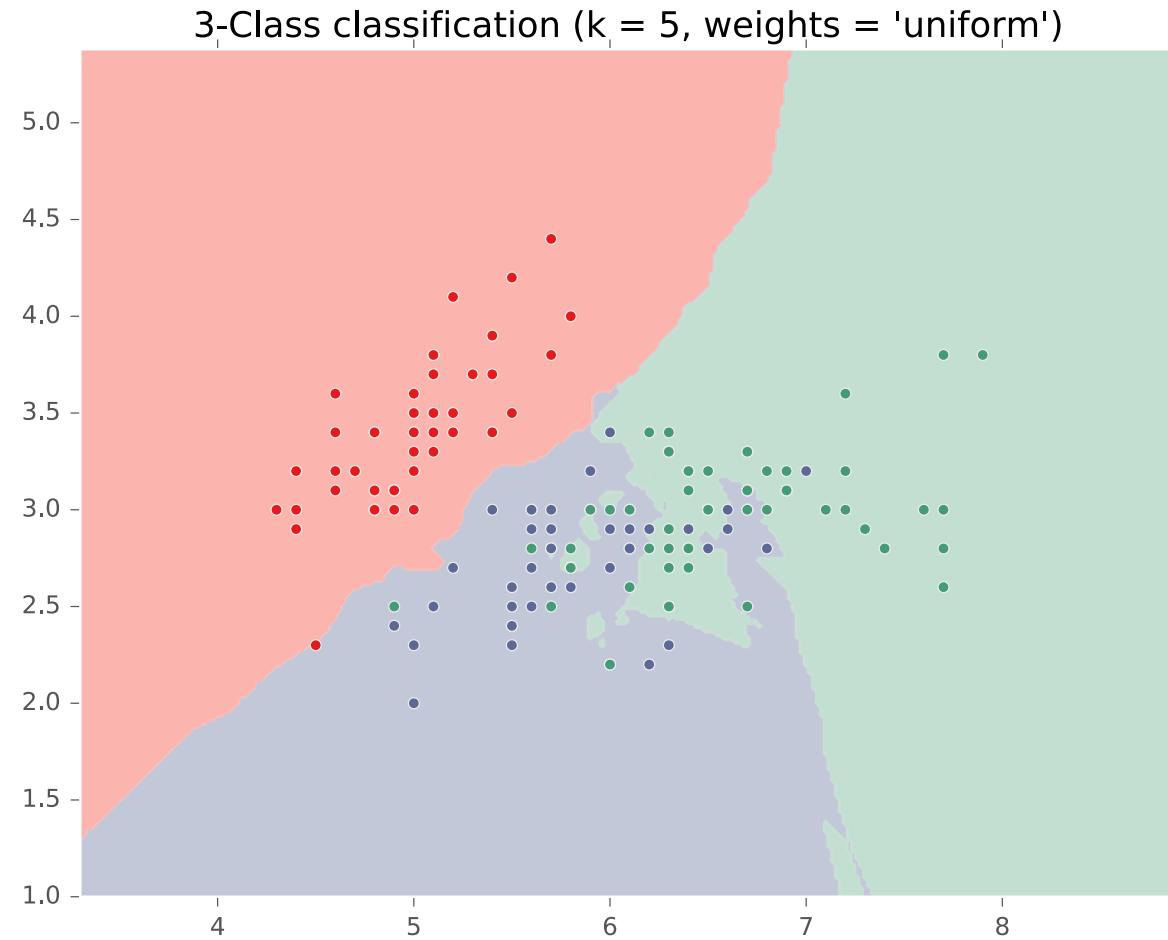
# k-NN on Fisher Iris Data



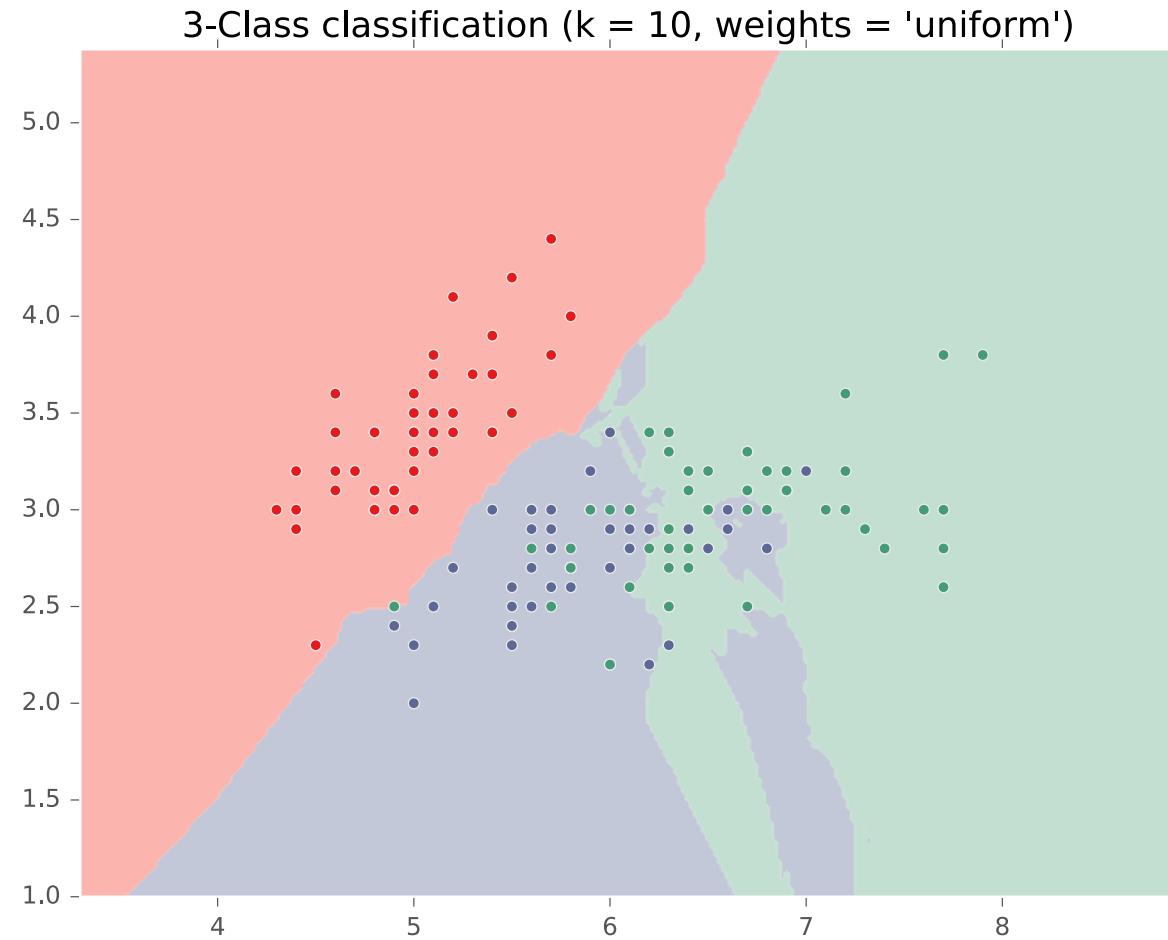
# k-NN on Fisher Iris Data



# k-NN on Fisher Iris Data

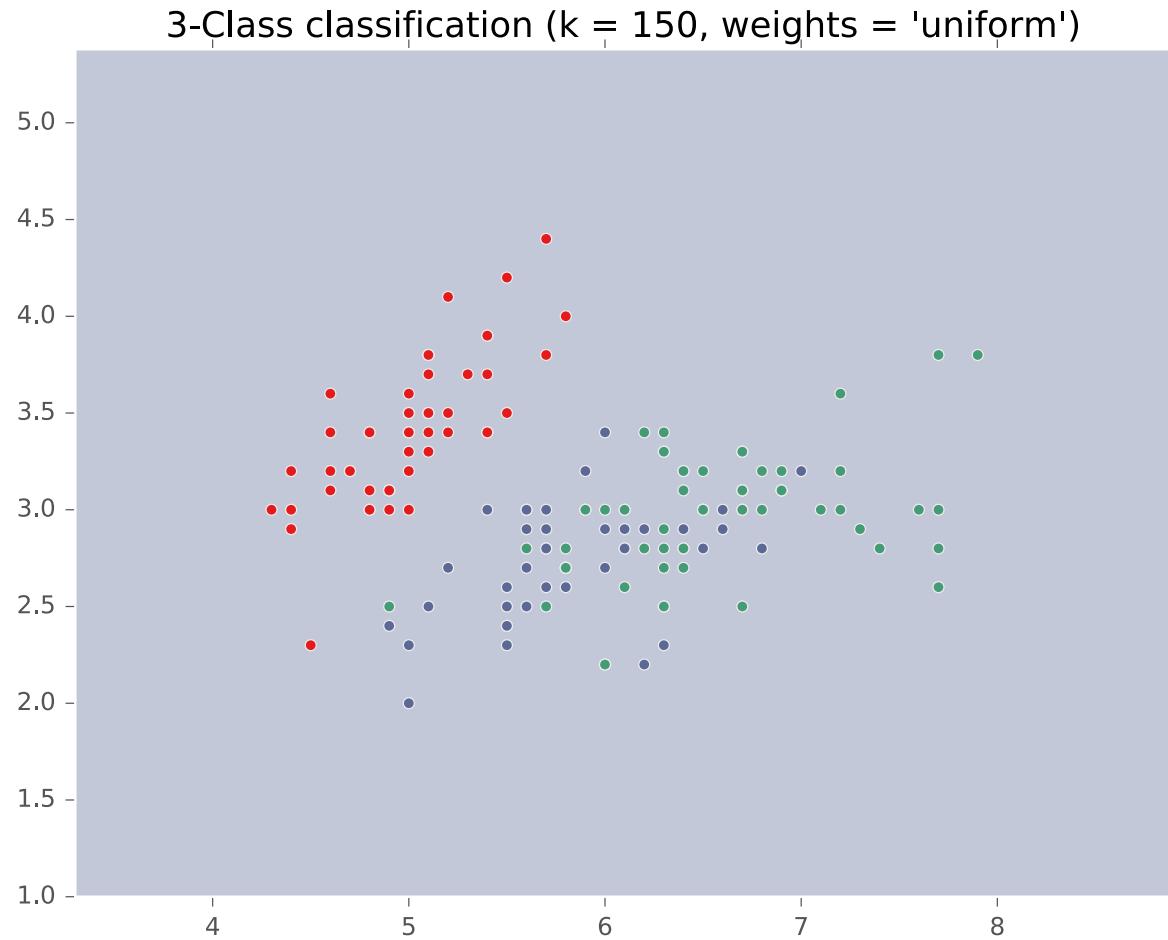


# k-NN on Fisher Iris Data



# k-NN on Fisher Iris Data

## Special Case: Majority Vote



# k-NN: Remarks

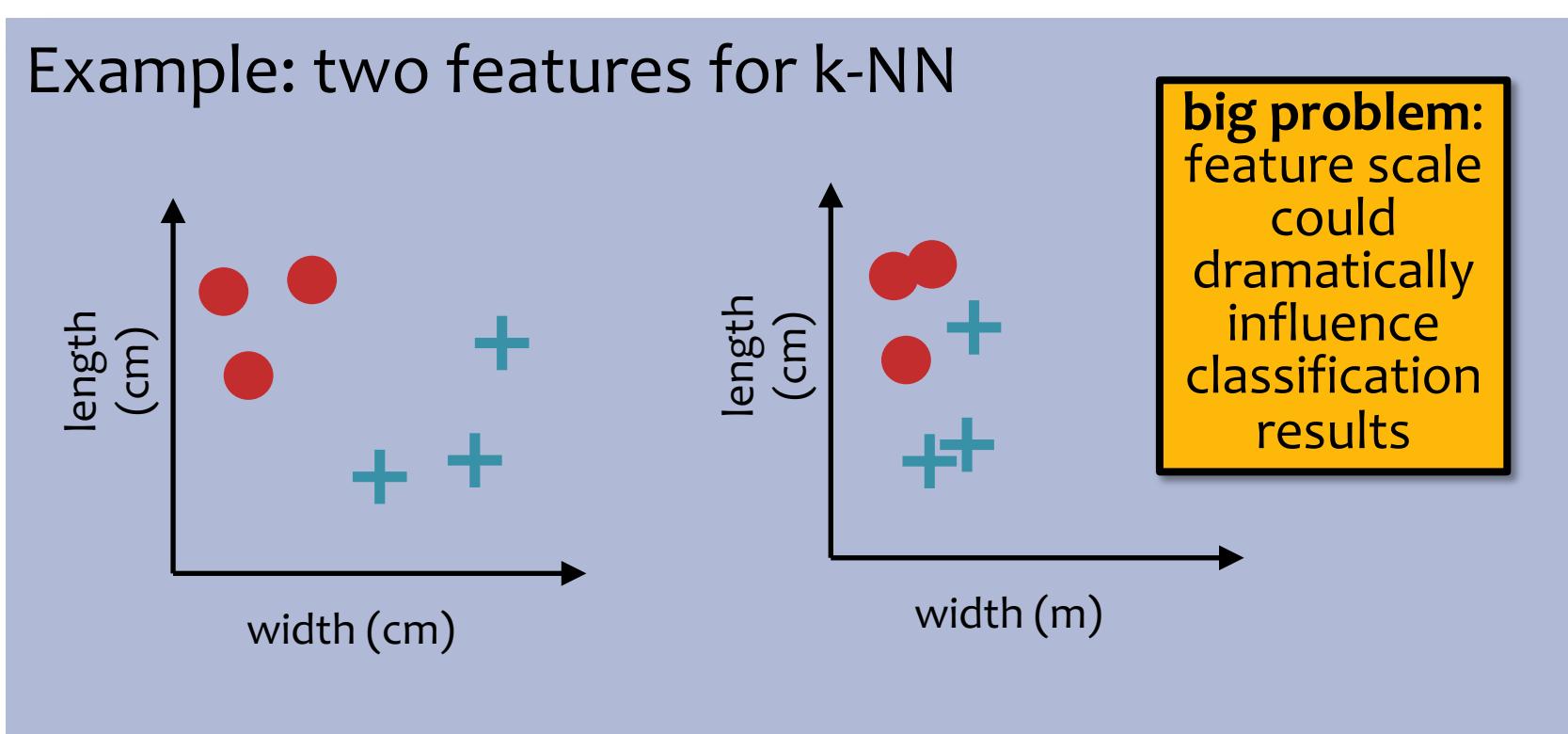
## Inductive Bias:

1. Close points should have similar labels
2. All dimensions are created equally!

# k-NN: Remarks

## Inductive Bias:

1. Close points should have similar labels
2. All dimensions are created equally!



# k-NN: Remarks

## **Computational Efficiency:**

- Suppose we have  $N$  training examples, and each one has  $M$  features
- Computational complexity for the special case where  $k=1$ :

## Poll 5 (train) and Poll 6 (predict)

Suppose we have  $N$  training examples, and each one has  $M$  features

Computational complexity for the special case where  $k=1$ :

- A.  $O(1)$
- B.  $O(\log N)$
- C.  $O(\log M)$
- D.  $O(\log NM)$
- E.  $O(N)$
- F.  $O(M)$
- G.  $O(NM)$
- H.  $O(N^2)$
- I.  $O(N^2M)$

## Poll 5 (train) and Poll 6 (predict)

Suppose we have  $N$  training examples, and each one has  $M$  features

Computational complexity for the special case where  $k=1$ :

- A.  $O(1)$
- B.  $O(\log N)$
- C.  $O(\log M)$
- D.  $O(\log NM)$
- E.  $O(N)$
- F.  $O(M)$
- G.  $O(NM)$
- H.  $O(N^2)$
- I.  $O(N^2M)$

# k-NN: Remarks

## Computational Efficiency:

- Suppose we have  $N$  training examples, and each one has  $M$  features
- Computational complexity for the special case where  $k=1$ :

Task	Naive	k-d Tree
Train	$O(1)$	$\sim O(M N \log N)$
Predict (one test example)	$O(MN)$	$\sim O(2^M \log N)$ 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)

# MODEL SELECTION

# Model Selection

## **WARNING:**

- In some sense, our discussion of model selection is premature.
- The models we have considered thus far are fairly simple.
- The models and the many decisions available to the data scientist wielding them will grow to be much more complex than what we've seen so far.

# Model Selection

## Statistics

- *Def:* a **model** defines the data generation process (i.e. a set or family of parametric probability distributions)
- *Def:* **model parameters** are the values that give rise to a particular probability distribution in the model family
- *Def:* **learning** (aka. estimation) is the process of finding the parameters that best fit the data
- *Def:* **hyperparameters** are the parameters of a prior distribution over parameters

## Machine Learning

- *Def:* (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def:* **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def:* the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def:* **hyperparameters** are the tunable aspects of the model, that the learning algorithm does not select

# Model Selection

## Example: Decision Tree

- model = set of all possible trees, possibly restricted by some hyperparameters (e.g. max depth)
- parameters = structure of a specific decision tree
- learning algorithm = ID3, CART, etc.
- hyperparameters = max-depth, threshold for splitting criterion, etc.

## Machine Learning

- *Def:* (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def:* **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def:* the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def:* **hyperparameters** are the tunable aspects of the model, that the learning algorithm does not select

# Model Selection

## Example: k-Nearest Neighbors

- model = set of all possible nearest neighbors classifiers
- parameters = none (KNN is an instance-based or non-parametric method)
- learning algorithm = for naïve setting, just storing the data
- hyperparameters =  $k$ , the number of neighbors to consider

## Machine Learning

- Def: (loosely) a **model** defines the hypothesis space over which learning performs its search
- Def: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- Def: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- Def: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does not select

# Model Selection

## Statistics

- Def: a **model** defines the data generation process (i.e. a set or family of probability distributions)
- Def: **model parameters** are the numeric parameters that give rise to a particular probability distribution in the model family
- Def: **learning** (aka. estimation) is the process of finding the parameters that best fit the data
- Def: **hyperparameters** are the parameters of a prior distribution over parameters

## Machine Learning

- Def: (loosely) a **model** defines the hypothesis which learning performs its task
- **parameters** are the numeric parameters that give rise to a hypothesis
- Def: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- Def: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does not select

# Model Selection

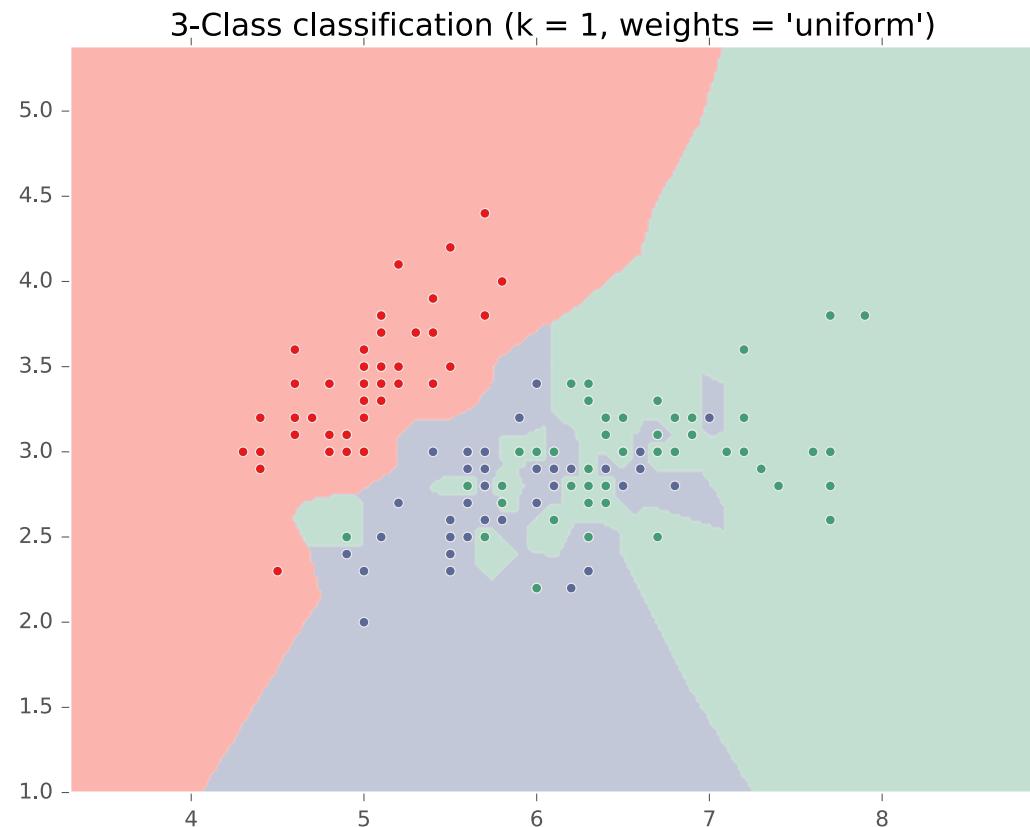
- Two very similar definitions:
  - *Def: model selection* is the process by which we choose the “best” model from among a set of candidates
  - *Def: hyperparameter optimization* is the process by which we choose the “best” hyperparameters from among a set of candidates (**could be called a special case of model selection**)
- **Both** assume access to a function capable of measuring the quality of a model
- **Both** are typically done “outside” the main training algorithm --- typically training is treated as a black box

# Experimental Design

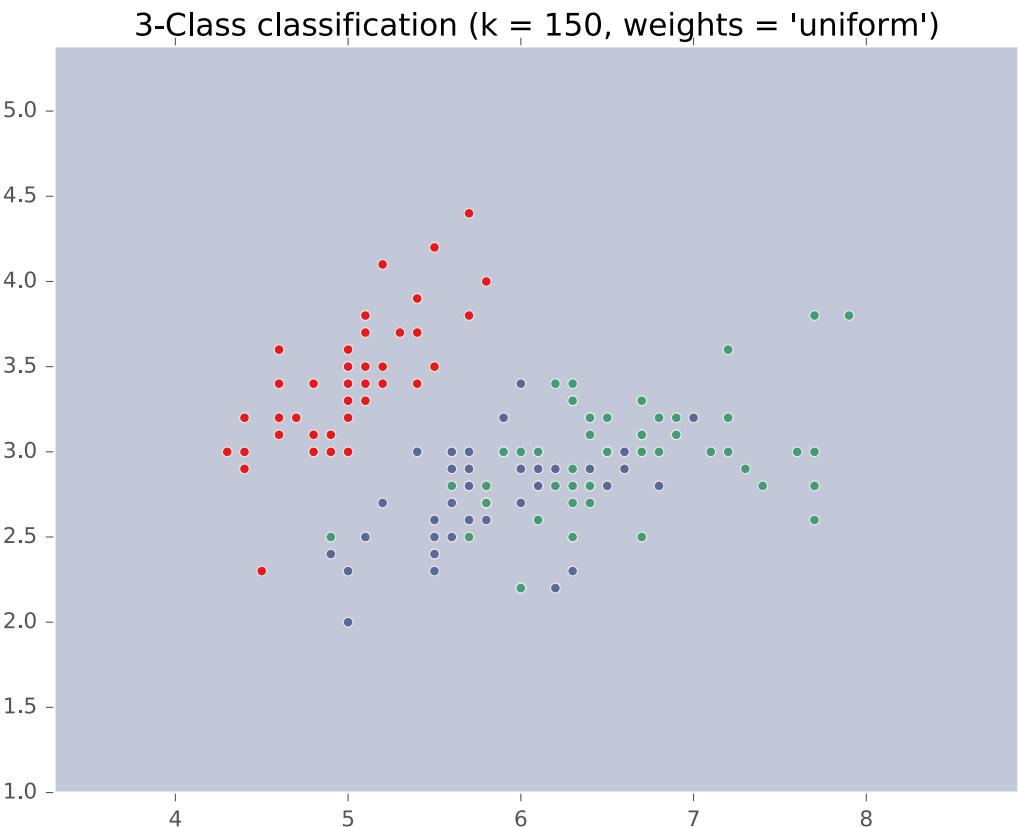
	<b>Input</b>	<b>Output</b>	<b>Notes</b>
Training	<ul style="list-style-type: none"><li>• training dataset</li><li>• hyperparameters</li></ul>	<ul style="list-style-type: none"><li>• best model parameters</li></ul>	We pick the best model parameters by learning on the training dataset for a fixed set of hyperparameters
Hyperparameter Optimization	<ul style="list-style-type: none"><li>• training dataset</li><li>• validation dataset</li></ul>	<ul style="list-style-type: none"><li>• best hyperparameters</li></ul>	We pick the best hyperparameters by learning on the training data and evaluating error on the validation error
Testing	<ul style="list-style-type: none"><li>• test dataset</li><li>• hypothesis (i.e. fixed model parameters)</li></ul>	<ul style="list-style-type: none"><li>• test error</li></ul>	We evaluate a hypothesis corresponding to a decision rule with fixed model parameters on a test dataset to obtain test error

# Special Cases of k-NN

## k=1: Nearest Neighbor

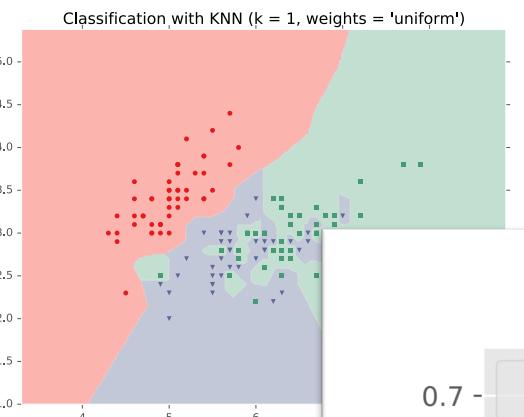


## k=N: Majority Vote

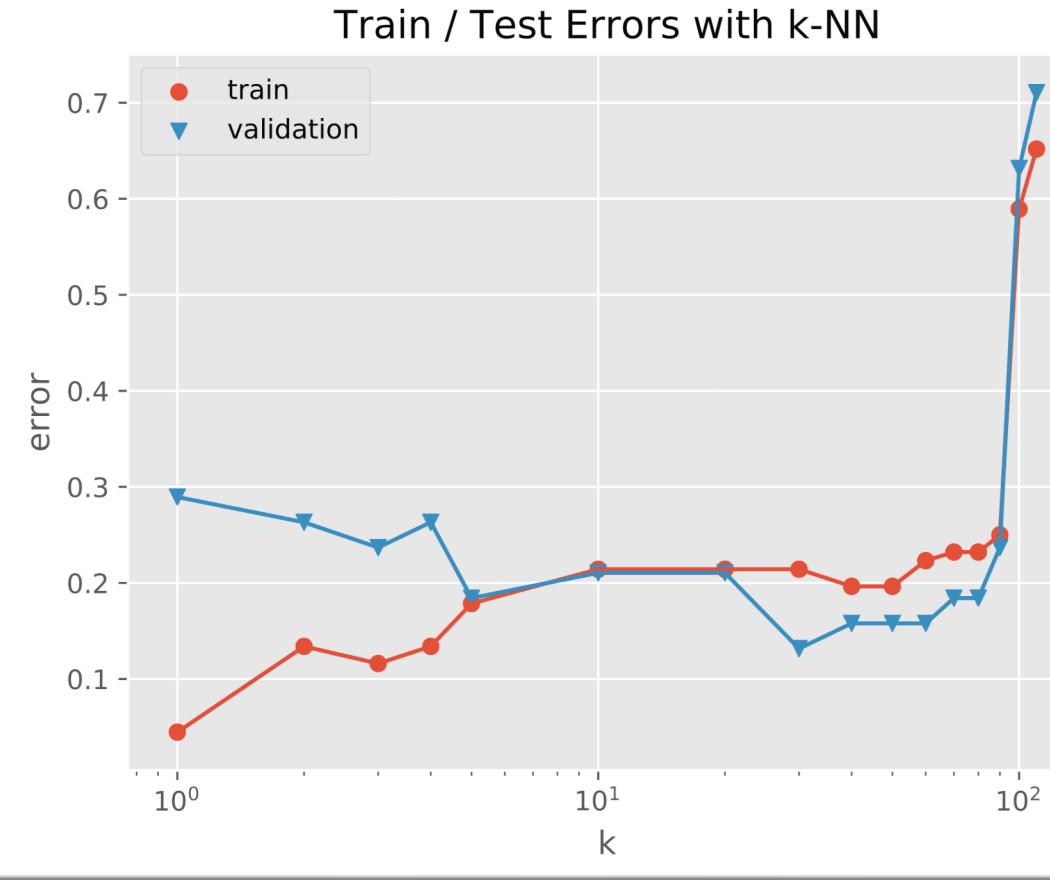
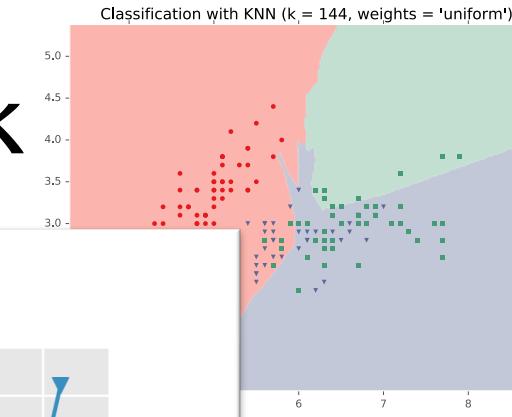


# Example of Hyperparameter Optimization

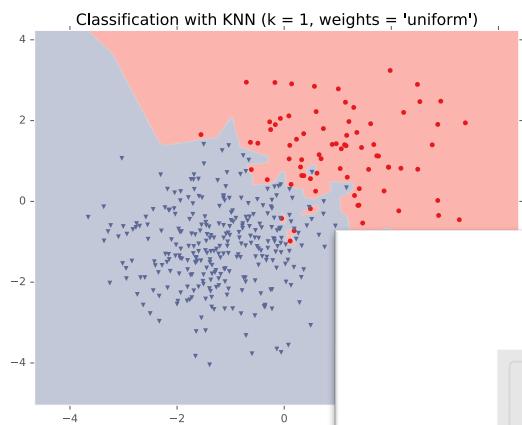
## Choosing $k$ for k-NN



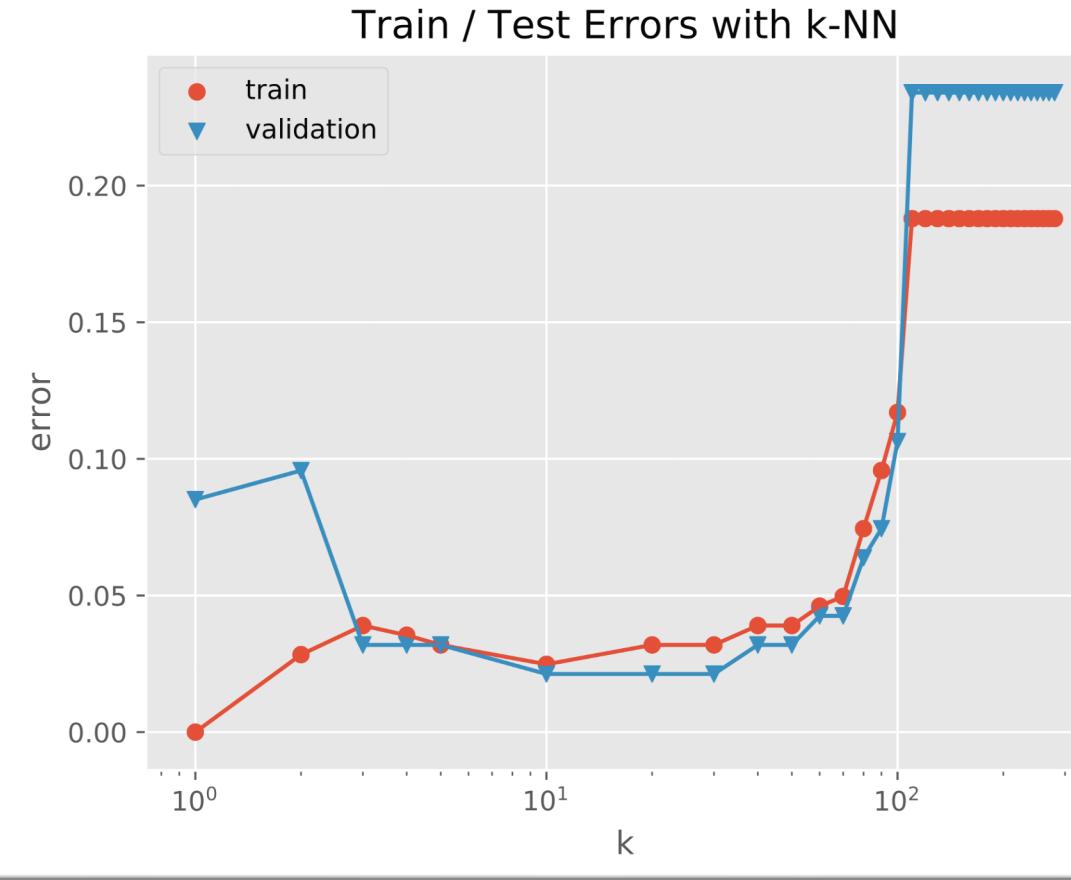
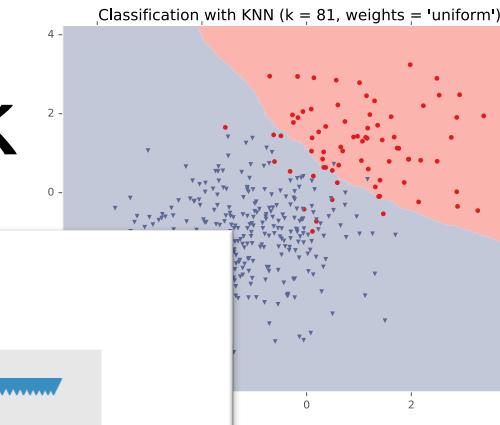
# k-NN: Choosing k



Fisher Iris Data: varying the value of k



# k-NN: Choosing k



## Gaussian Data: varying the value of k

# Validation

## Why do we need validation?

- Choose hyperparameters
- Choose technique
- Help make any choices beyond our parameters

## But now, we have another choice to make!

- How do we split training and validation?

## Trade-offs

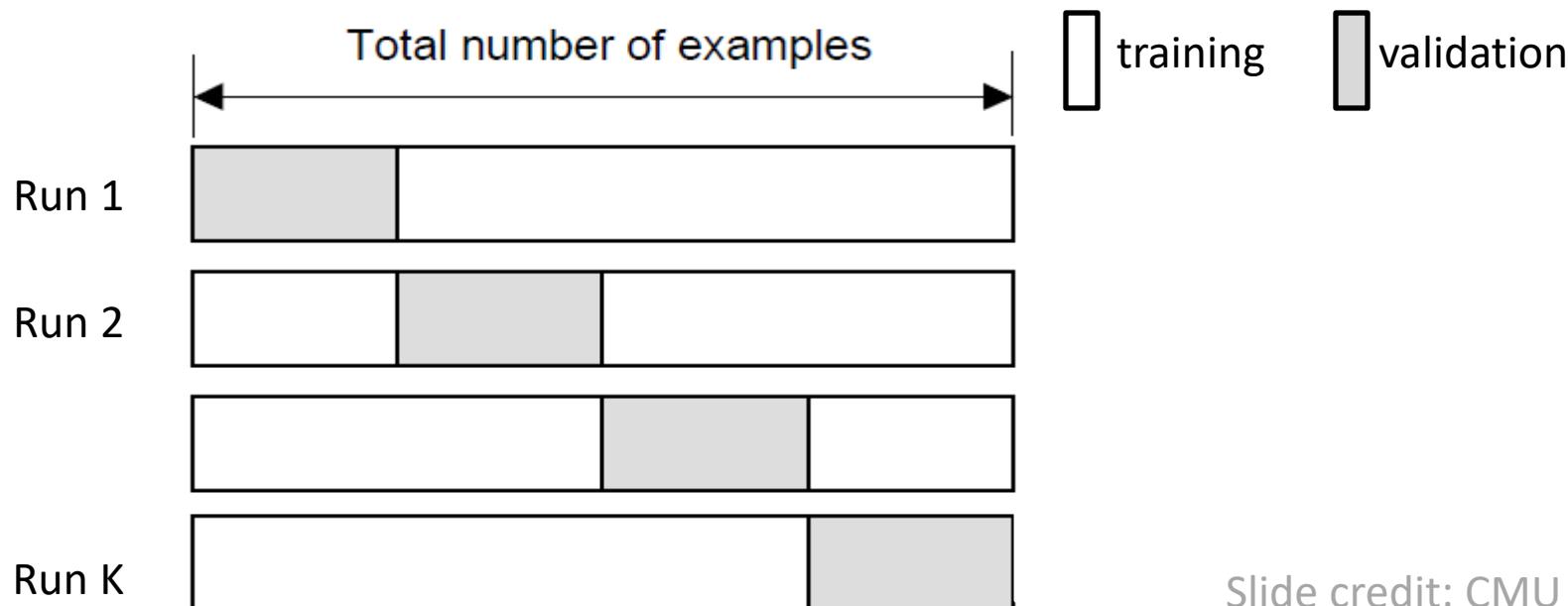
- More held-out data, better meaning behind validation numbers
- More held-out data, less data to train on!

# Cross-validation

## K-fold cross-validation

Create K-fold partition of the dataset.

Do K runs: train using K-1 partitions and calculate validation error on remaining partition (rotating validation partition on each run).  
Report average validation error

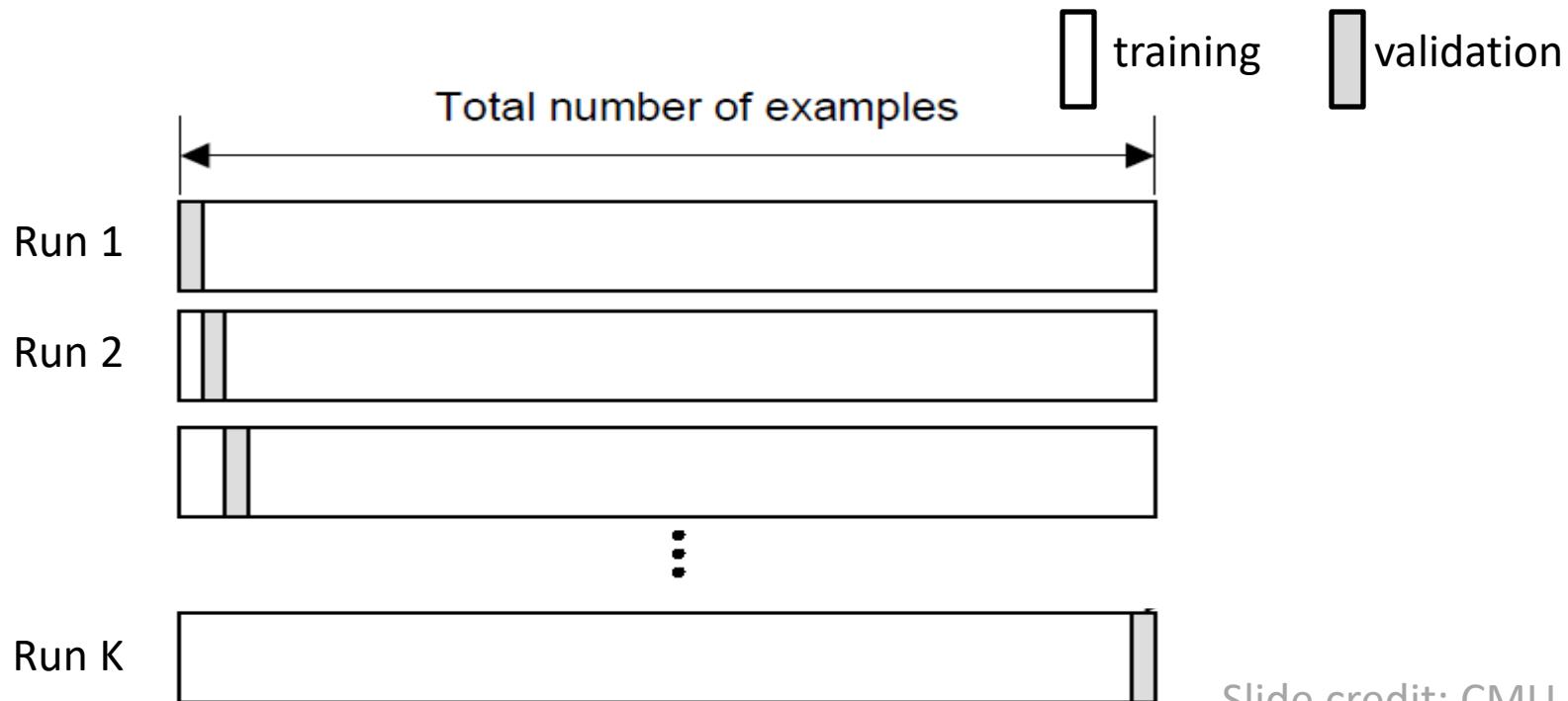


# Cross-validation

## Leave-one-out (LOO) cross-validation

Special case of K-fold with  $K=N$  partitions

Equivalently, train on  $N-1$  samples and validate on only one sample per run for  $N$  runs



# Cross-validation

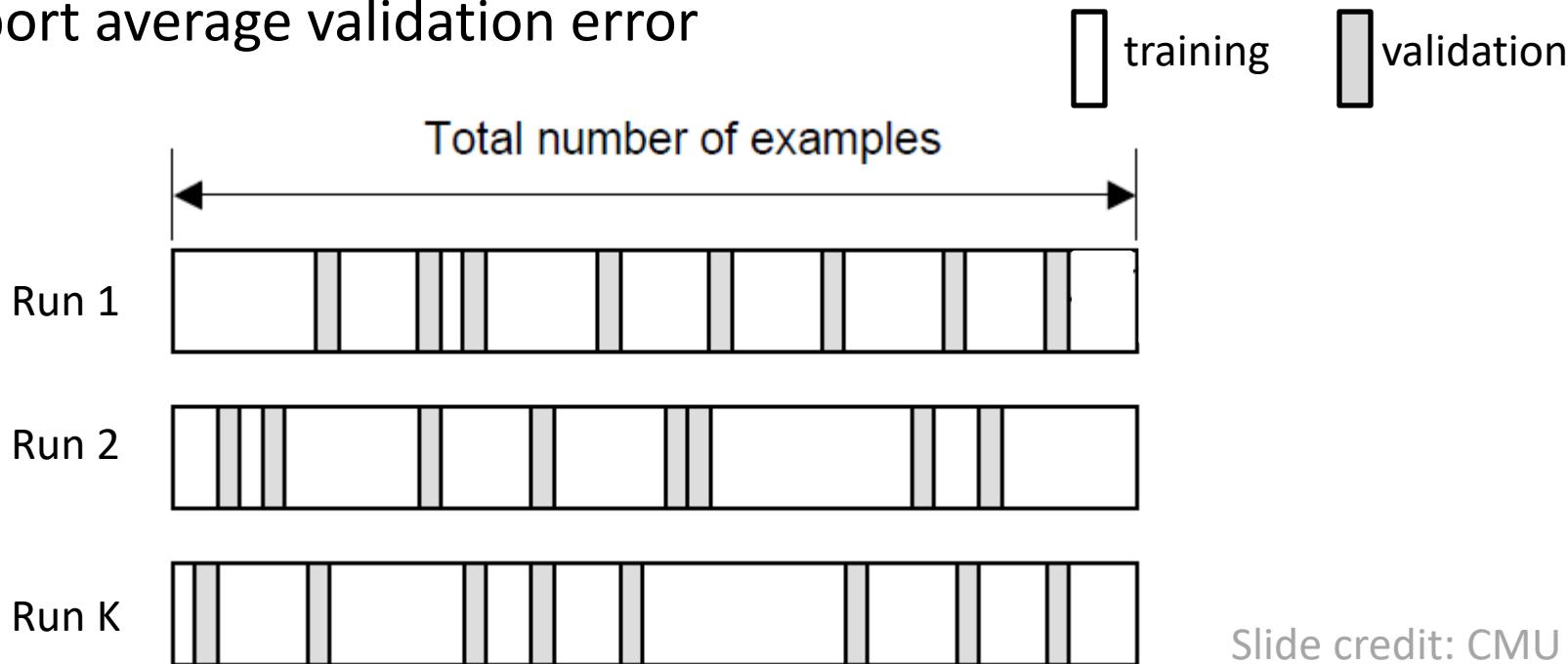
## Random subsampling

Randomly subsample a fixed fraction  $\alpha N$  ( $0 < \alpha < 1$ ) of the dataset for validation.

Compute validation error with remaining data as training data.

Repeat K times

Report average validation error



## Poll 7

Say you are choosing amongst 7 discrete values of a decision tree *mutual information threshold*, and you want to do K=5-fold cross-validation.

How many times do I have to train my model?

- A. 1
- B. 5
- C. 7
- D. 12
- E. 35
- F.  $5^7$

## Poll 7

Say you are choosing amongst 7 discrete values of a decision tree *mutual information threshold*, and you want to do K=5-fold cross-validation.

How many times do I have to train my model?

- A. 1
- B. 5
- C. 7
- D. 12
- E. 35
- F.  $5^7$

# Model Selection

## WARNING (again):

- This section is only scratching the surface!
- Lots of methods for hyperparameter optimization: (to talk about later)
  - Grid search
  - Random search
  - Bayesian optimization
  - Gradient-descent
  - ...

## Main Takeaway:

- Model selection / hyperparameter optimization is just another form of learning