

# As you login



1. Rename yourself in Zoom to \*pre\*-pend your house number
  - e.g. “0 – Pat Virtue”
2. Open Piazza (getting ready for polls)
3. Download preview slides from course website
4. Grab something to write with/on 😊

# Announcements

## Assignments

- HW1 Feedback
- HW2
  - Due Mon, 9/21, 11:59 pm
  - Start now! OH will be \*super\* crowded as the deadline gets closer

## Breakout rooms

- Video on
- Unmute
- Introduce yourself if you haven't already met

An abstract graphic on the left side of the slide, featuring a sphere-like shape composed of a dense grid of intersecting red, green, and blue lines. The lines are curved and follow the contour of the sphere, creating a complex, woven pattern. The sphere is set against a dark gray background.

# Introduction to Machine Learning

## Nearest Neighbor and Model Selection

Instructor: Pat Virtue

# Plan

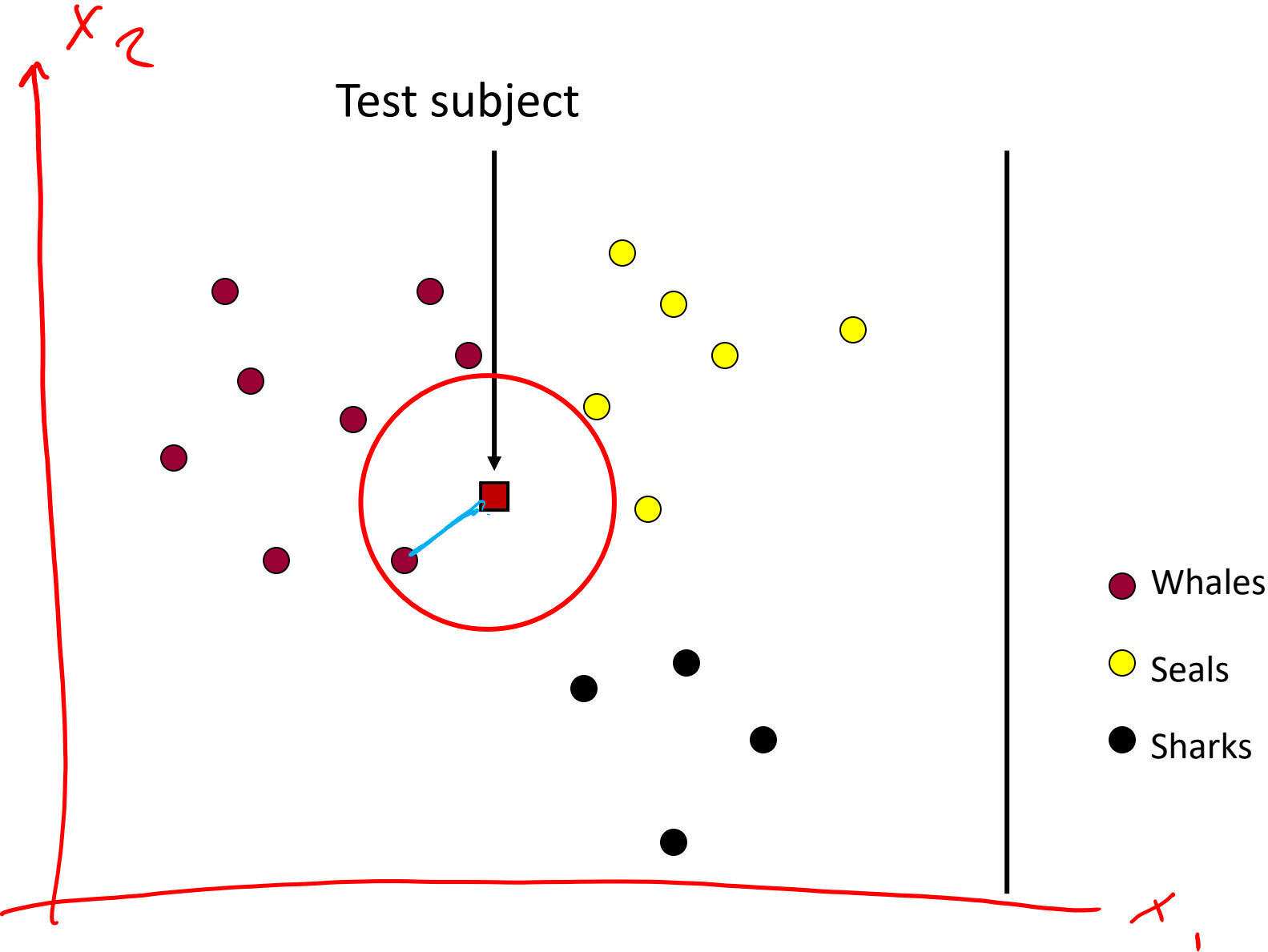
## Last time

- Decision trees
  - Continuous features, Overfitting
- Nearest neighbor methods

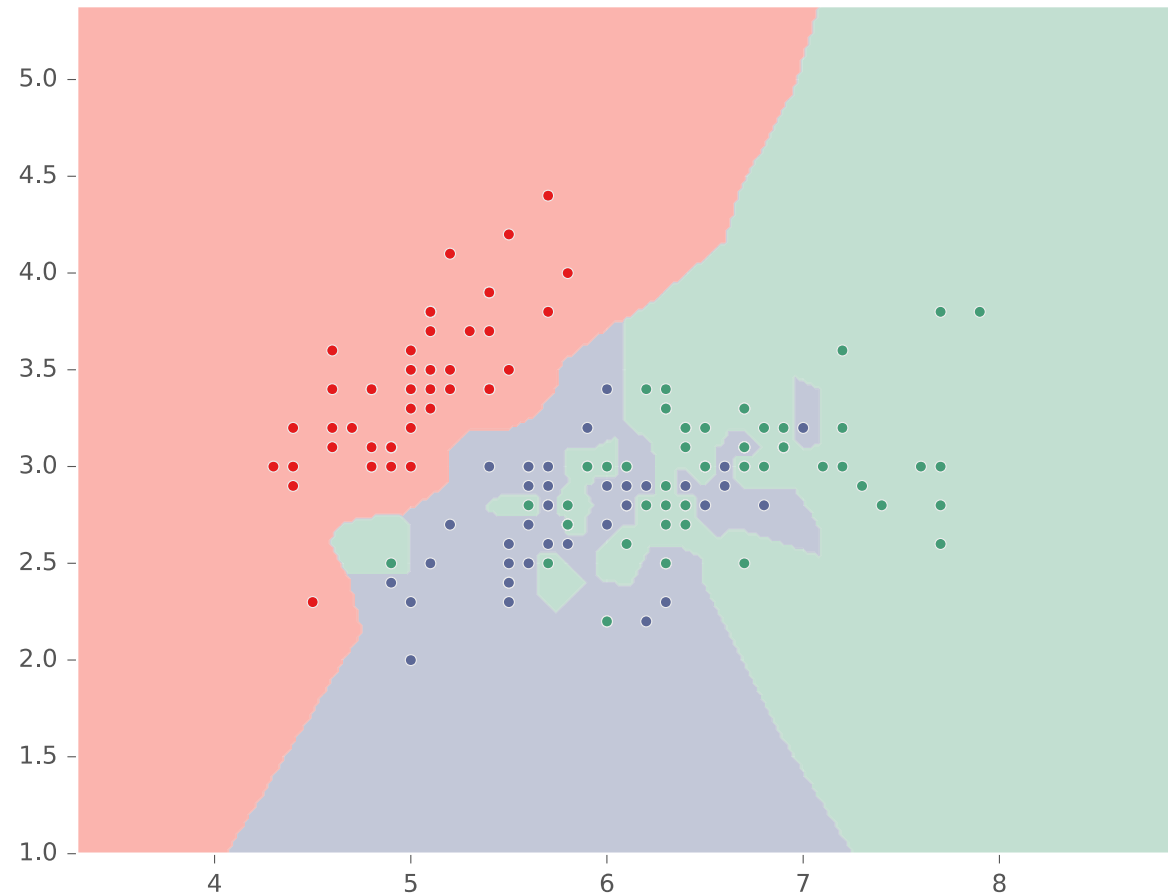
## Today

- K-nearest neighbor
- Nearest neighbor remarks
- Model selection / hyperparameter optimization
  - Validation methods

# Nearest Neighbor Classifier



# Nearest Neighbor on Fisher Iris Data

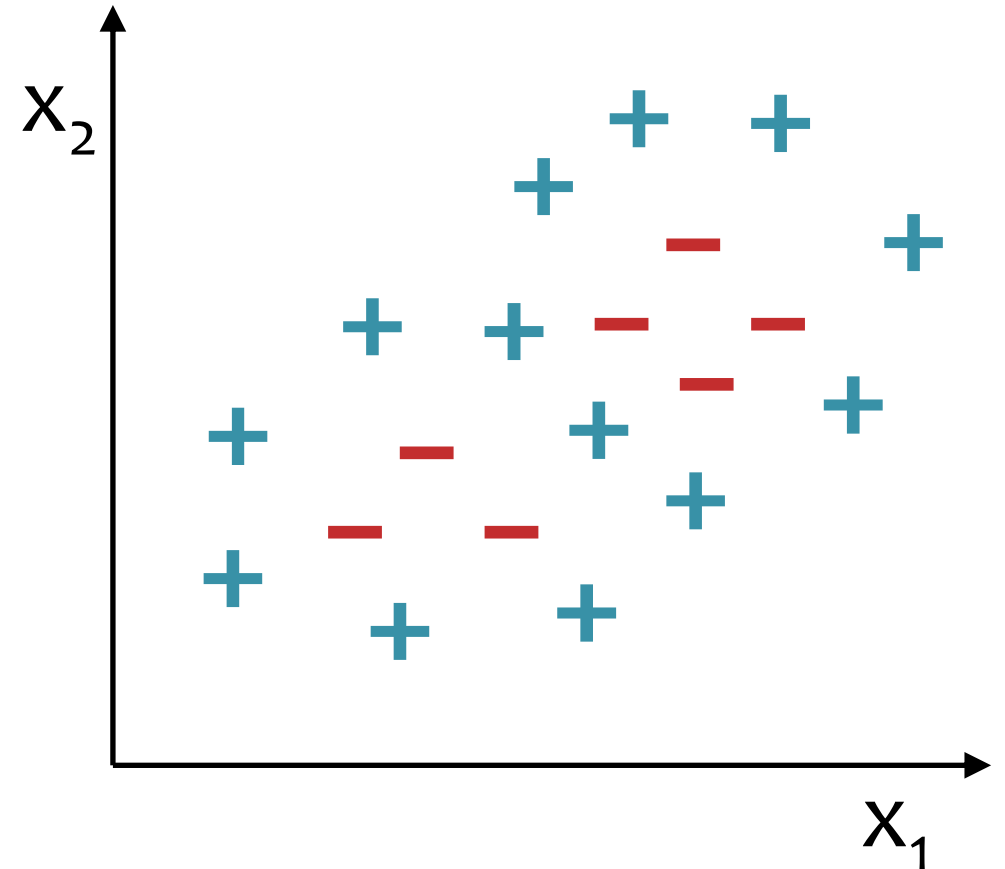


# Piazza Poll 1

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?

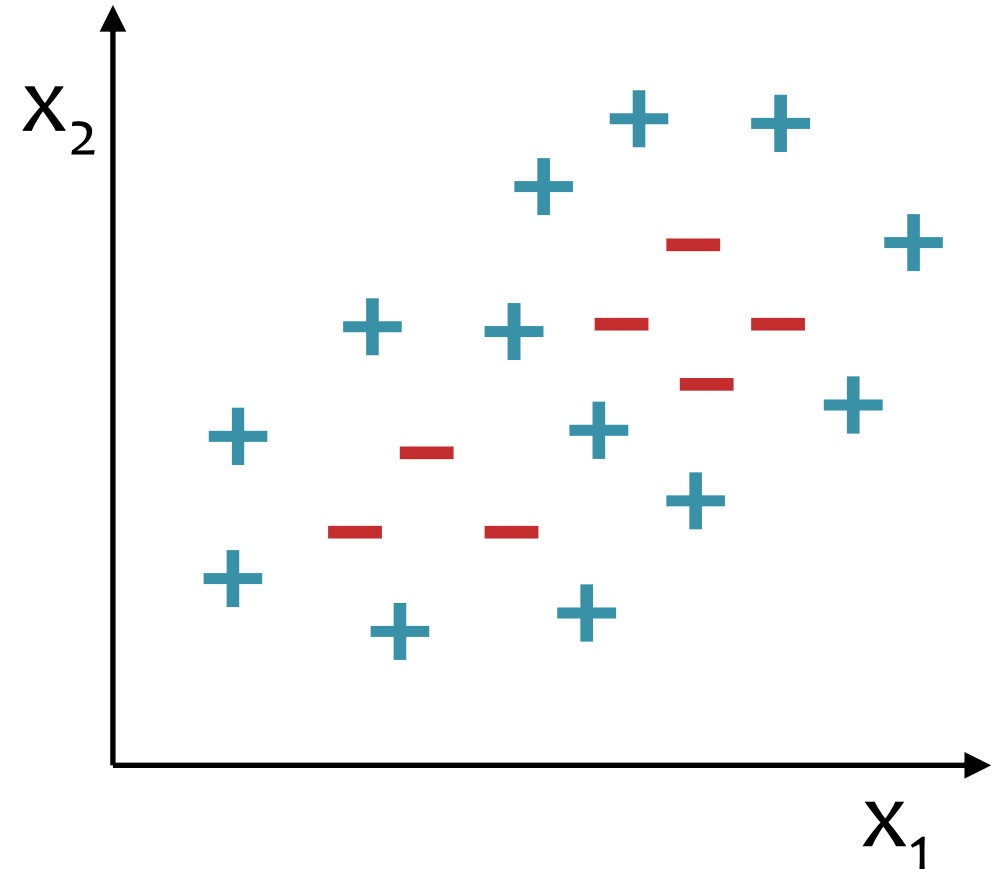


# Piazza Poll 1

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?

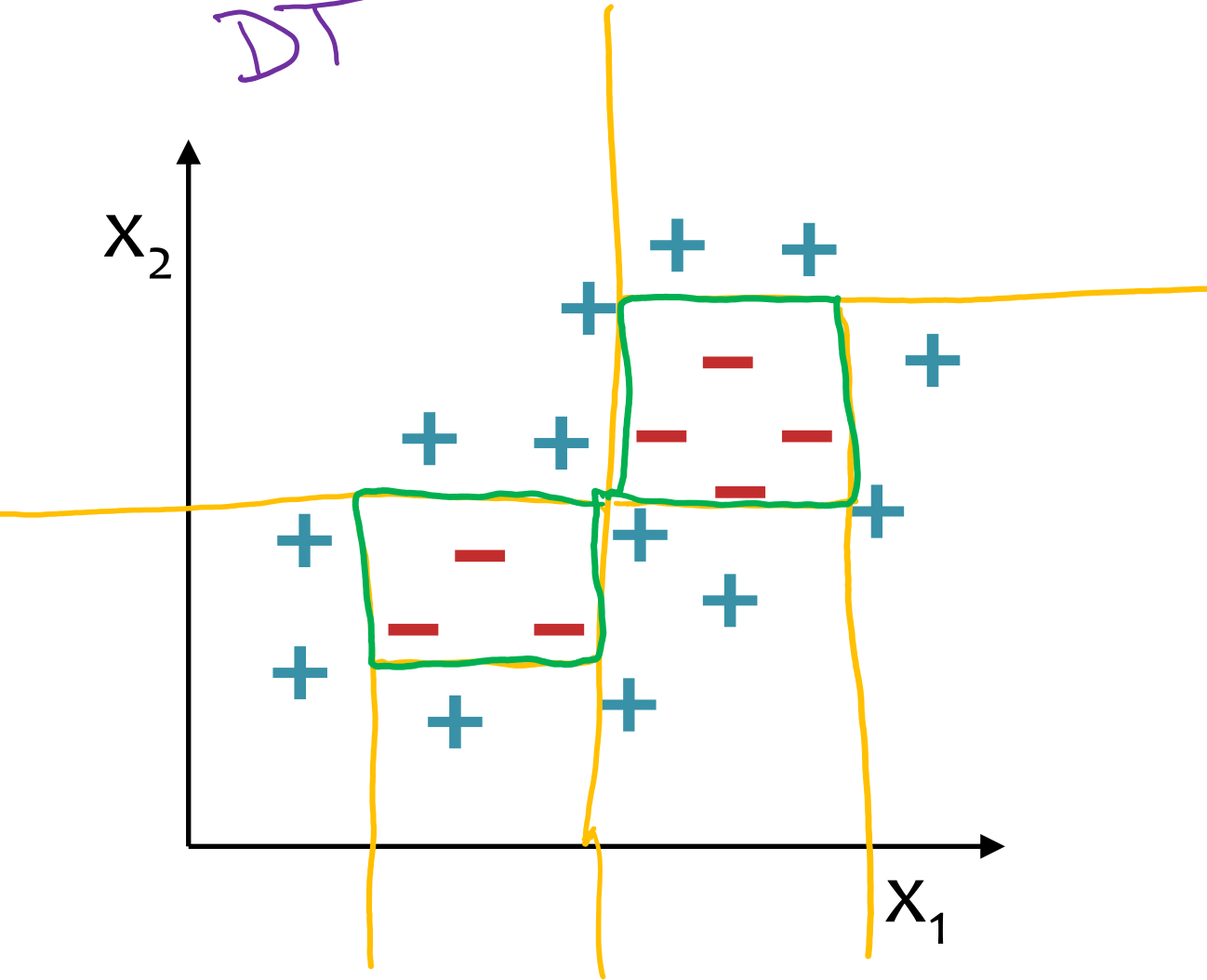




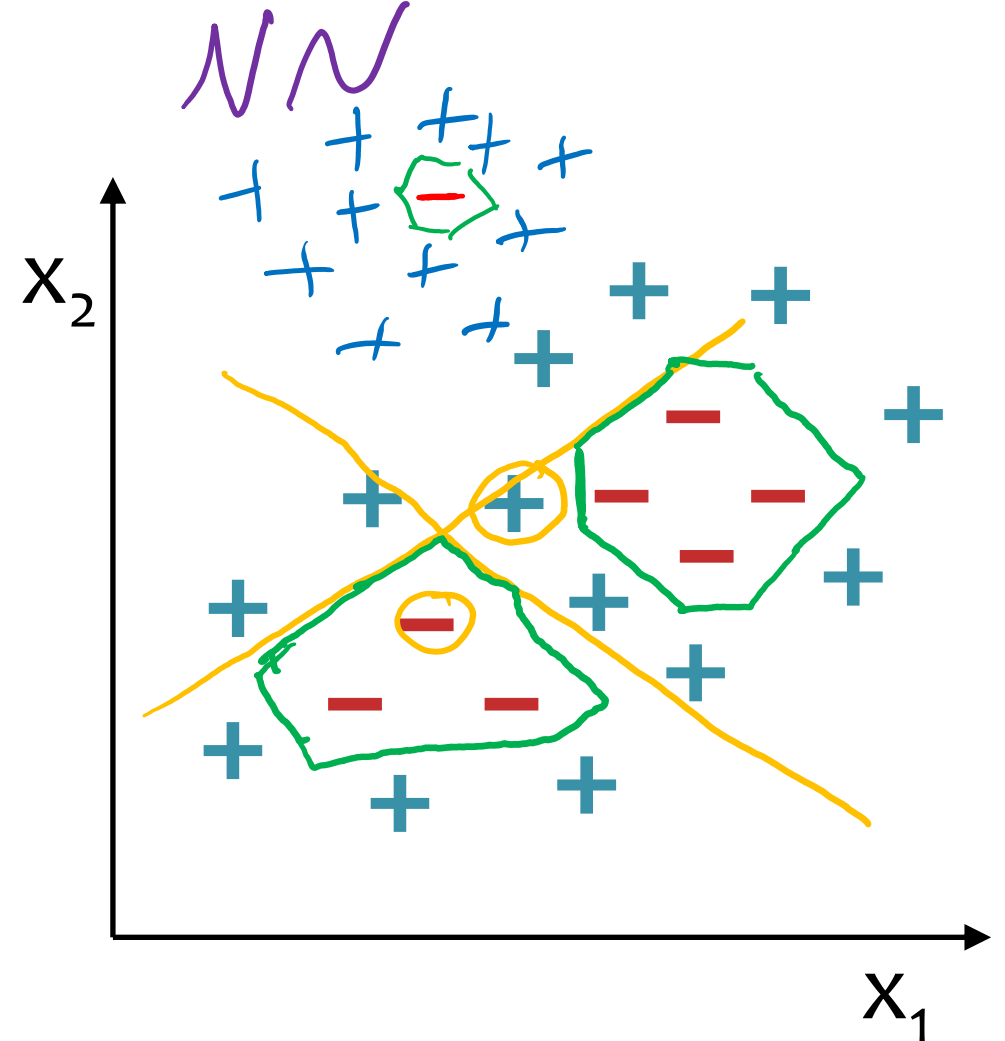
# Piazza Poll 1

Which methods can achieve zero training error on this dataset?

DT

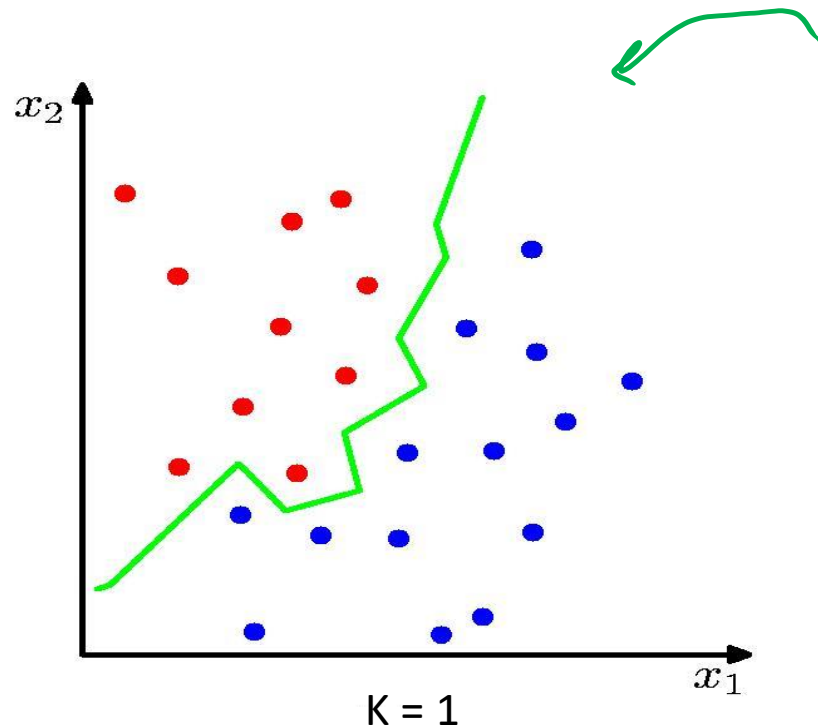


NN

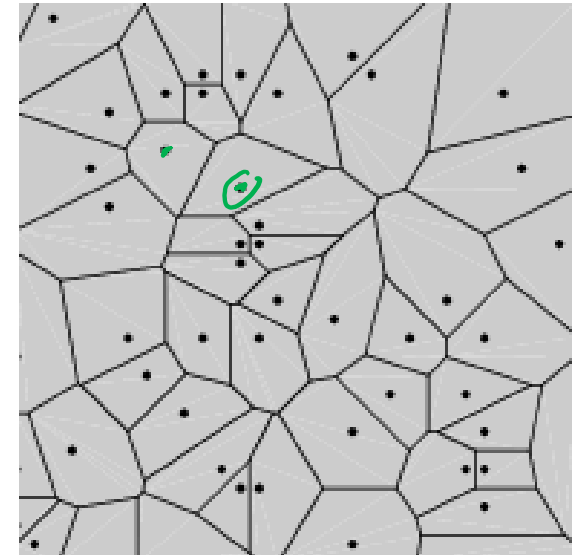


# Nearest Neighbor Decision Boundary

1-nearest neighbor classifier decision boundary



Voronoi Diagram



# Piazza Poll 2

1-nearest neighbor will likely:

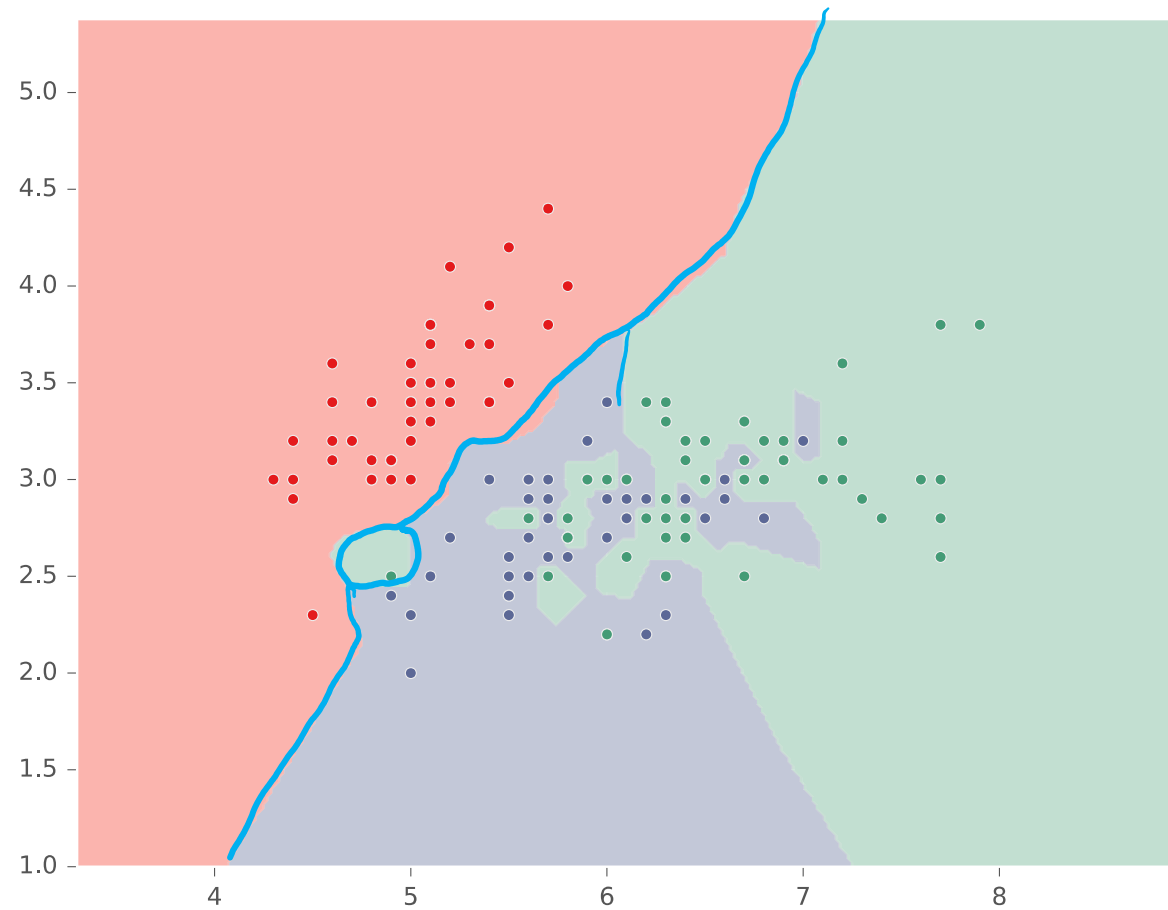
- A. Overfit 75%
- B. Underfit
- C. Neither (it's a great learner!)

# Piazza Poll 2

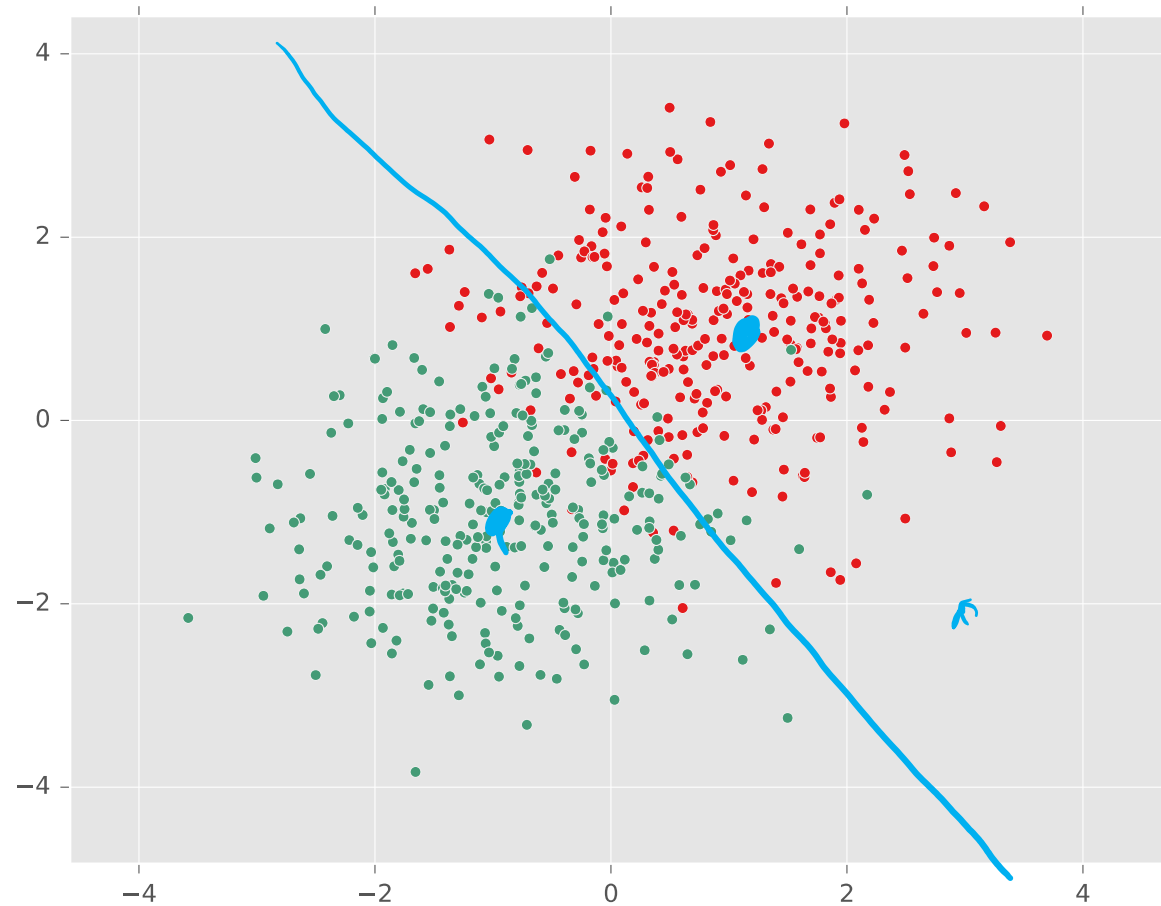
1-Nearest neighbor will likely:

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

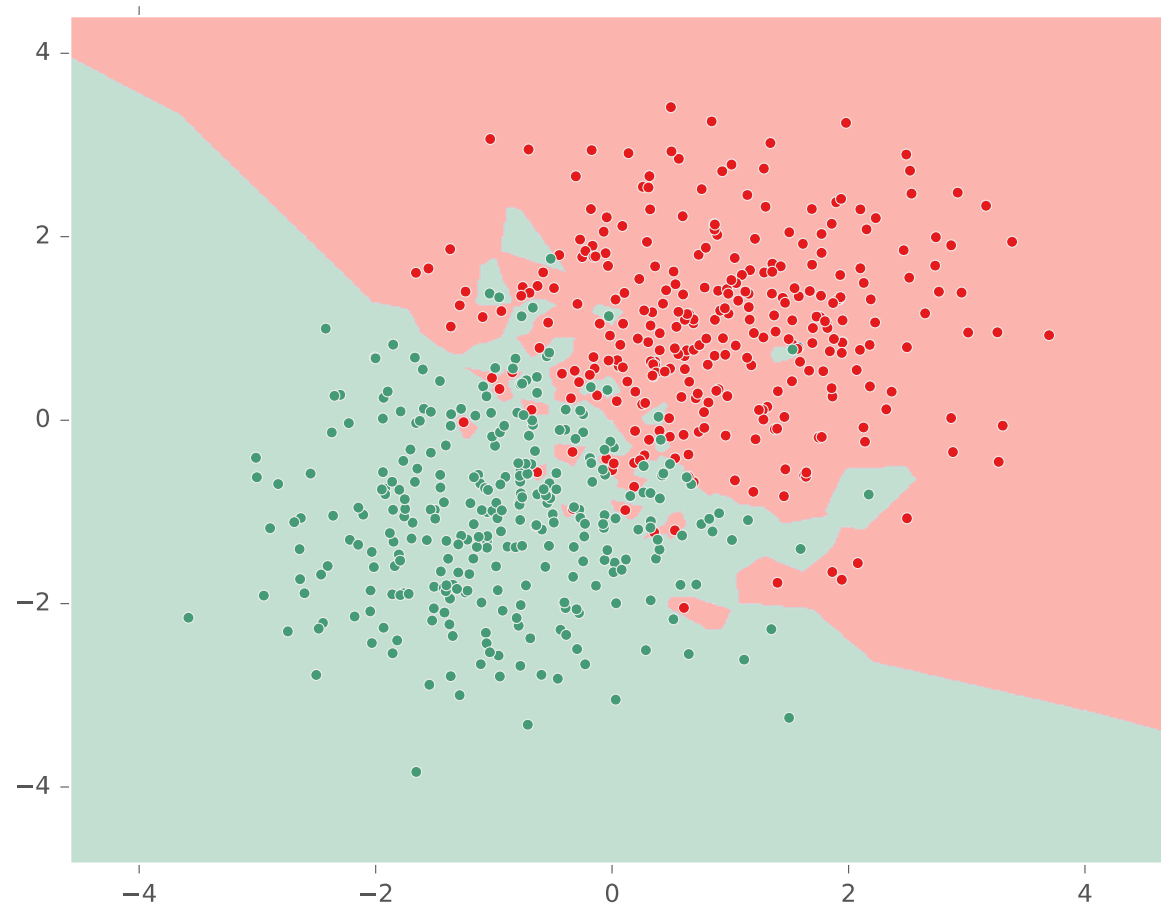
# Nearest Neighbor on Fisher Iris Data



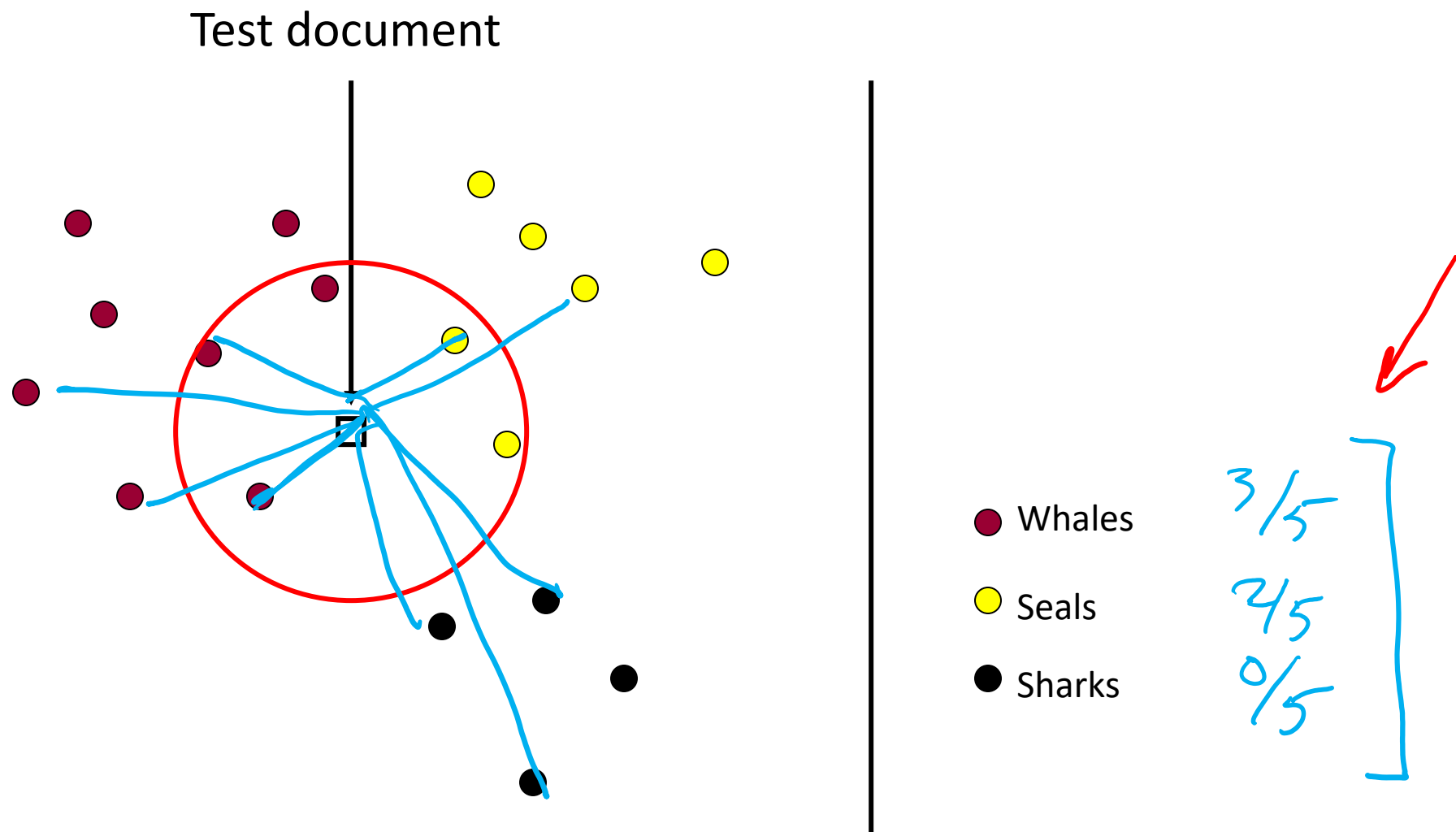
# Nearest Neighbor on Gaussian Data



# Nearest Neighbor on Gaussian Data



# kNN classifier (k=5)





# Nearest Neighbor Classification

→ train(D)  
store D

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

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

→  $h(\vec{x}_{test})$

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

$$\begin{aligned} d(\vec{x}, \vec{z}) &= \|\vec{x} - \vec{z}\|_2 \\ l_2 &= \left( \sum_{i=1}^M (x_i - z_i)^2 \right)^{1/2} \end{aligned}$$

$$\begin{aligned} d(\vec{x}, \vec{z}) &= \|\vec{x} - \vec{z}\|_1 \\ l_1 &= \sum_{i=1}^M |x_i - z_i| \end{aligned}$$

# k-Nearest Neighbor Classification

def train(D)  
store D

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  $\mathbf{x}_{test}$

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

- 2) Return the class label of that closest point

$$\hat{y}_{test} = \operatorname{argmax}_c p(Y = c \mid \mathbf{x}_{test}, \mathcal{D}, k) \leftarrow$$

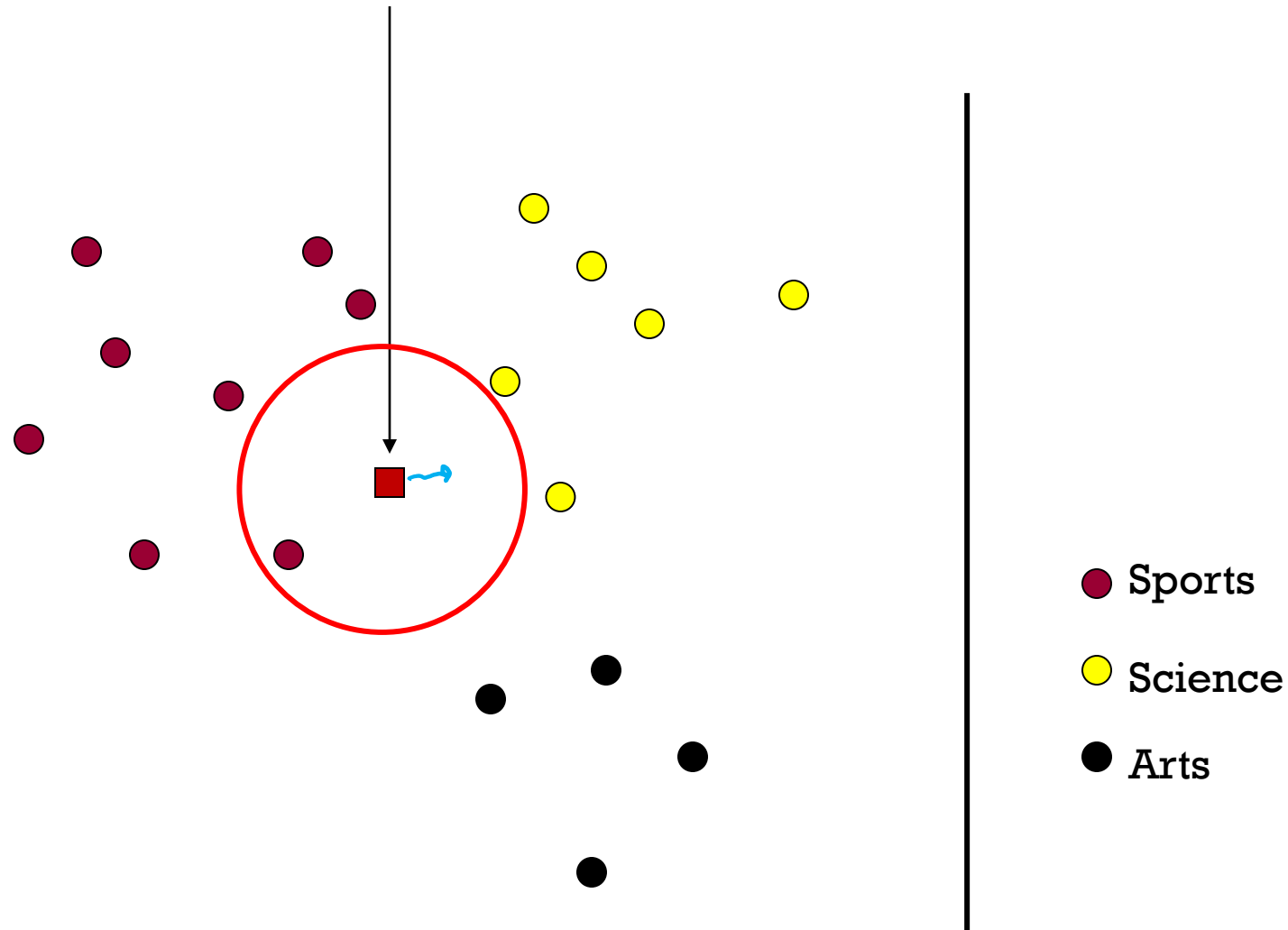
$$= \operatorname{argmax}_{\underline{c}} \frac{1}{k} \sum_{\underline{i} \in \mathcal{N}_k(\mathbf{x}_{test}, \mathcal{D})} \mathbb{I}(\underline{y}^{(i)} = c)$$

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

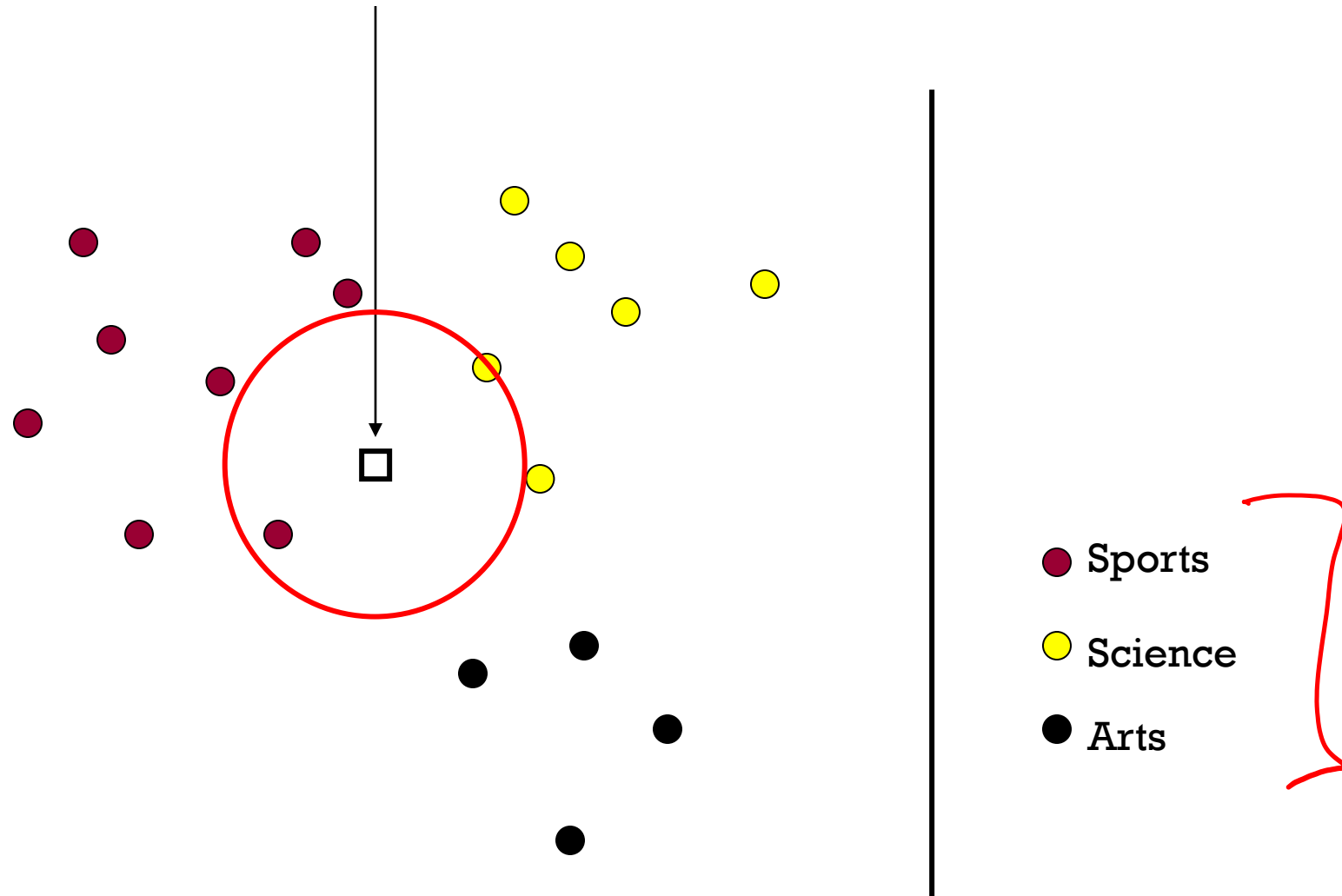
where  $k_c$  is the number of the  $k$ -neighbors with class label  $c$

def h( $\vec{x}_{test}$ )

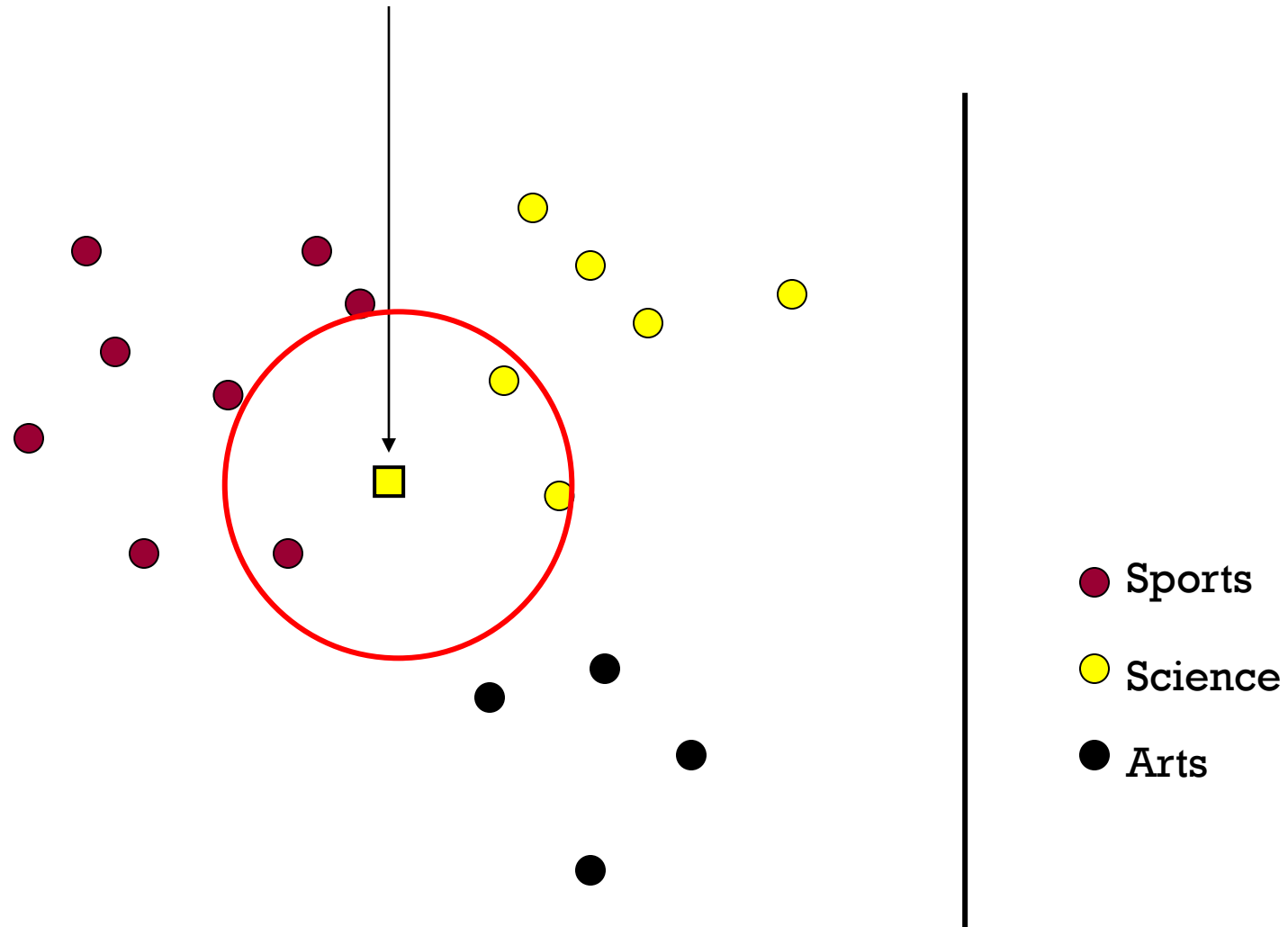
# 1-Nearest Neighbor (kNN) classifier



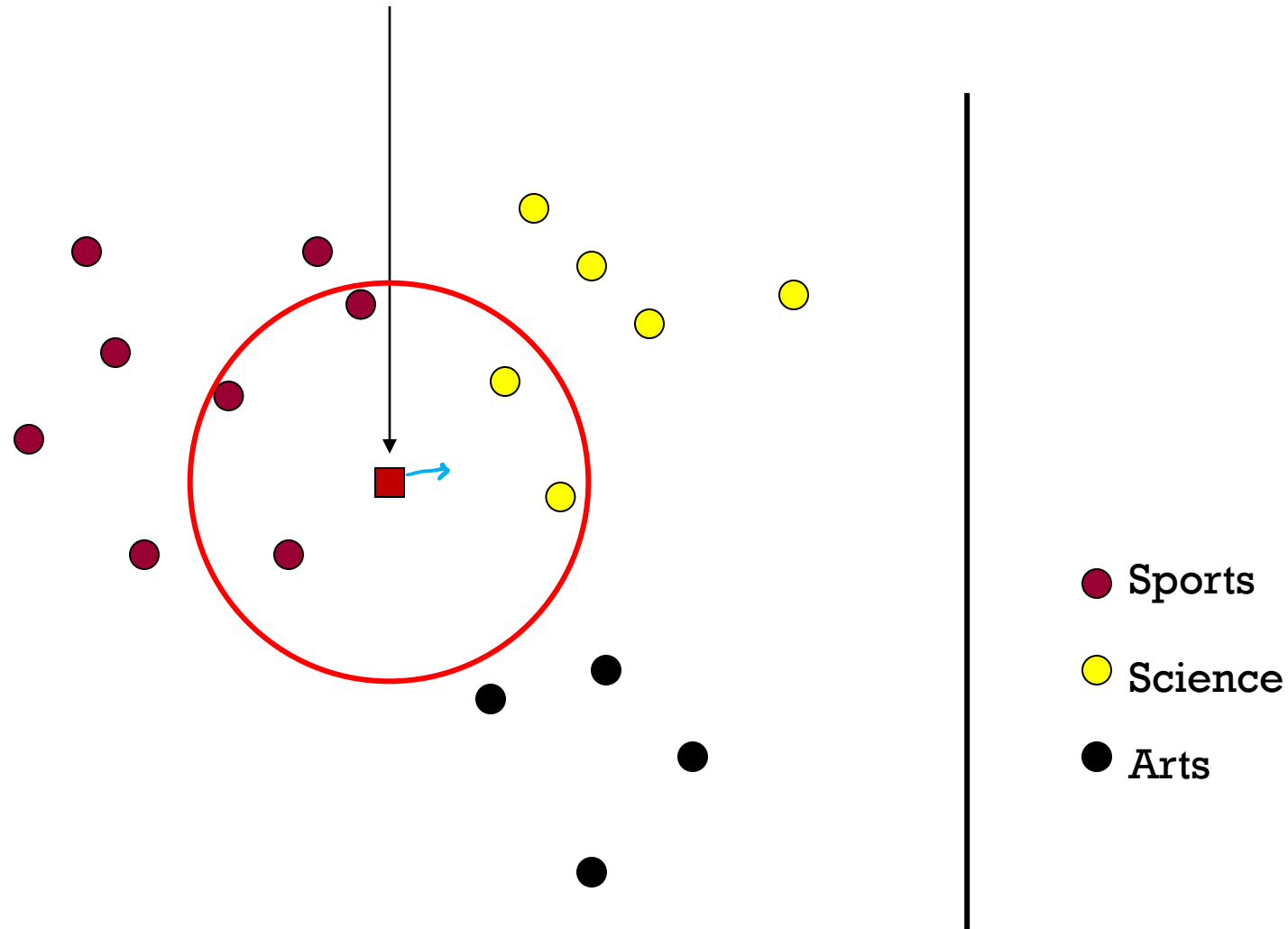
# 2-Nearest Neighbor (kNN) classifier



# 3-Nearest Neighbor (kNN) classifier



# 5-Nearest Neighbor (kNN) classifier



# What is the best $k$ ?

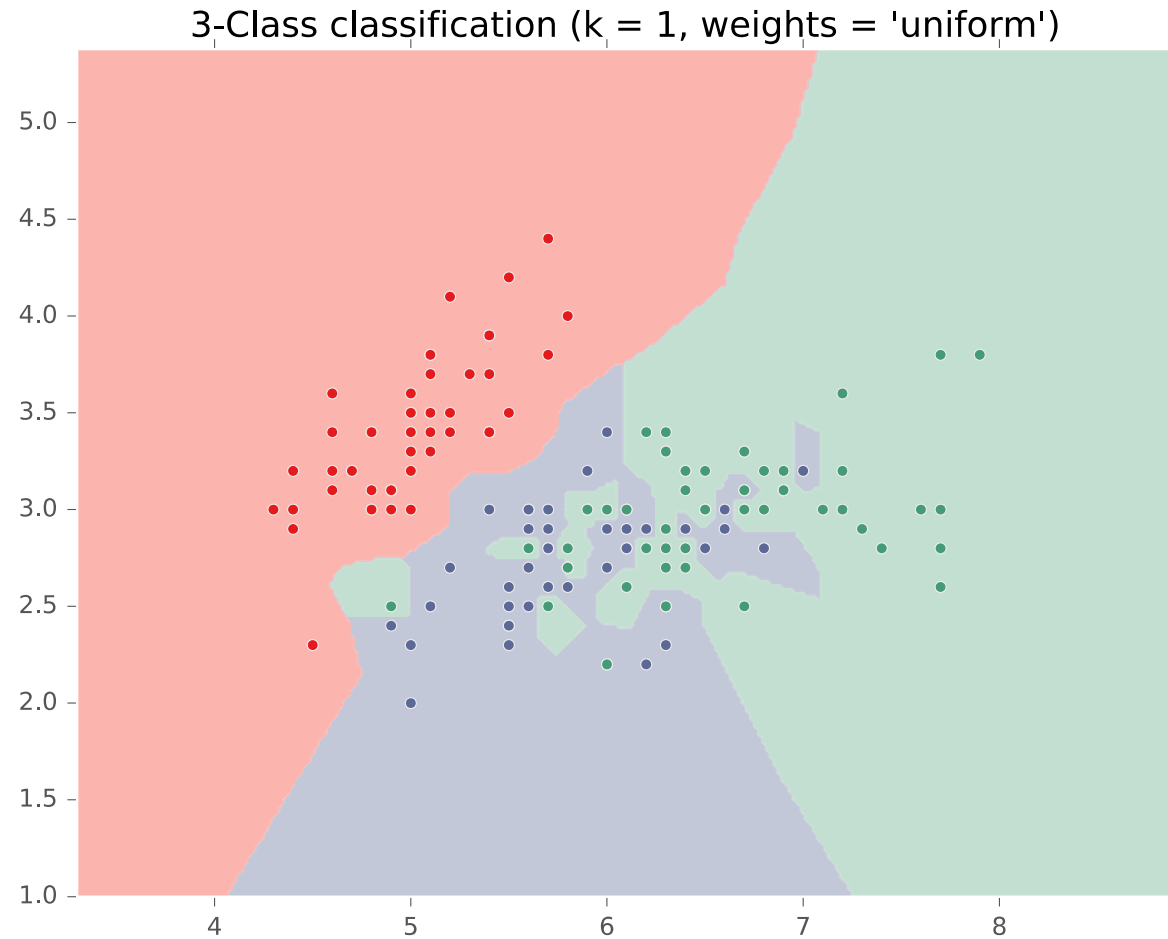
How do we choose a learner that is accurate and also generalizes to unseen data?

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

But how to choose  $k$ ?

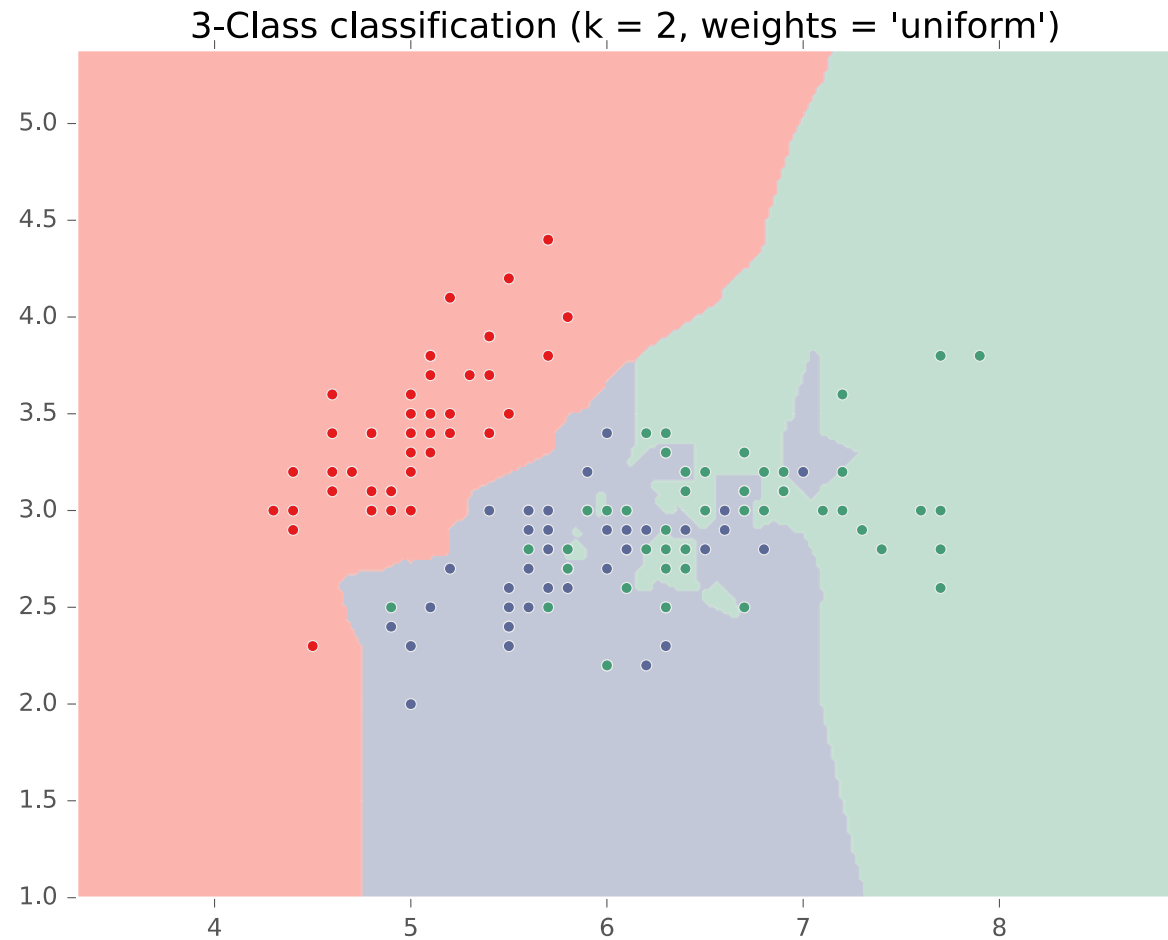
# k-NN on Fisher Iris Data

## Special Case: Nearest Neighbor

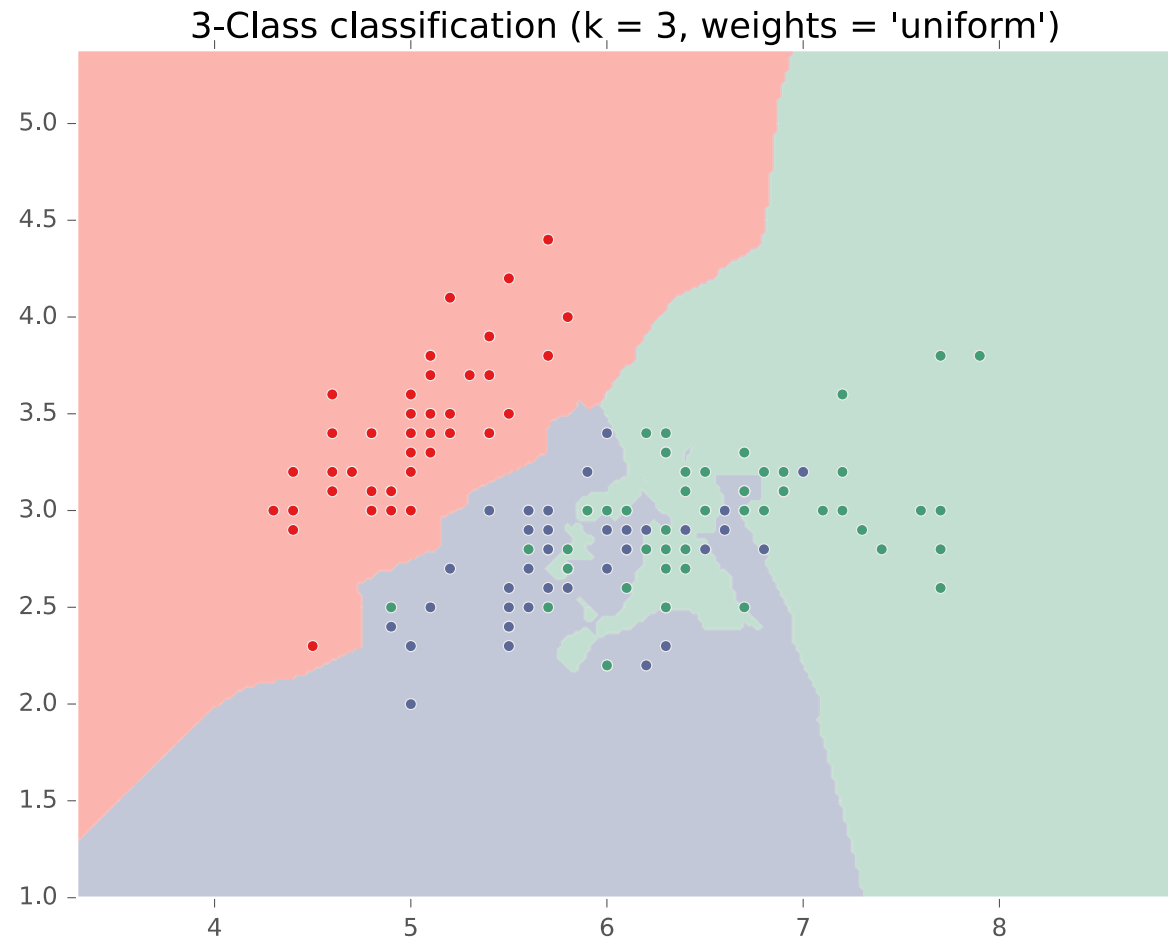




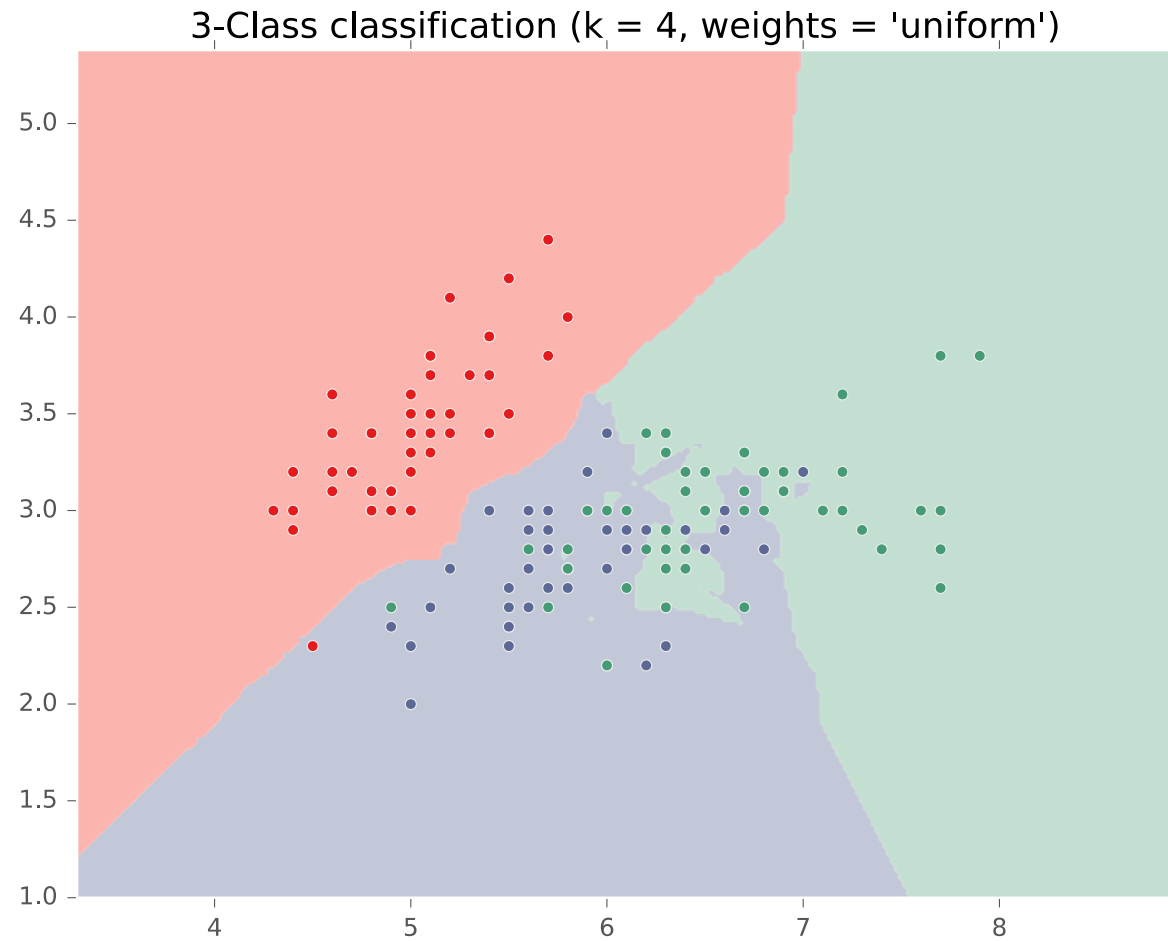
# k-NN on Fisher Iris Data



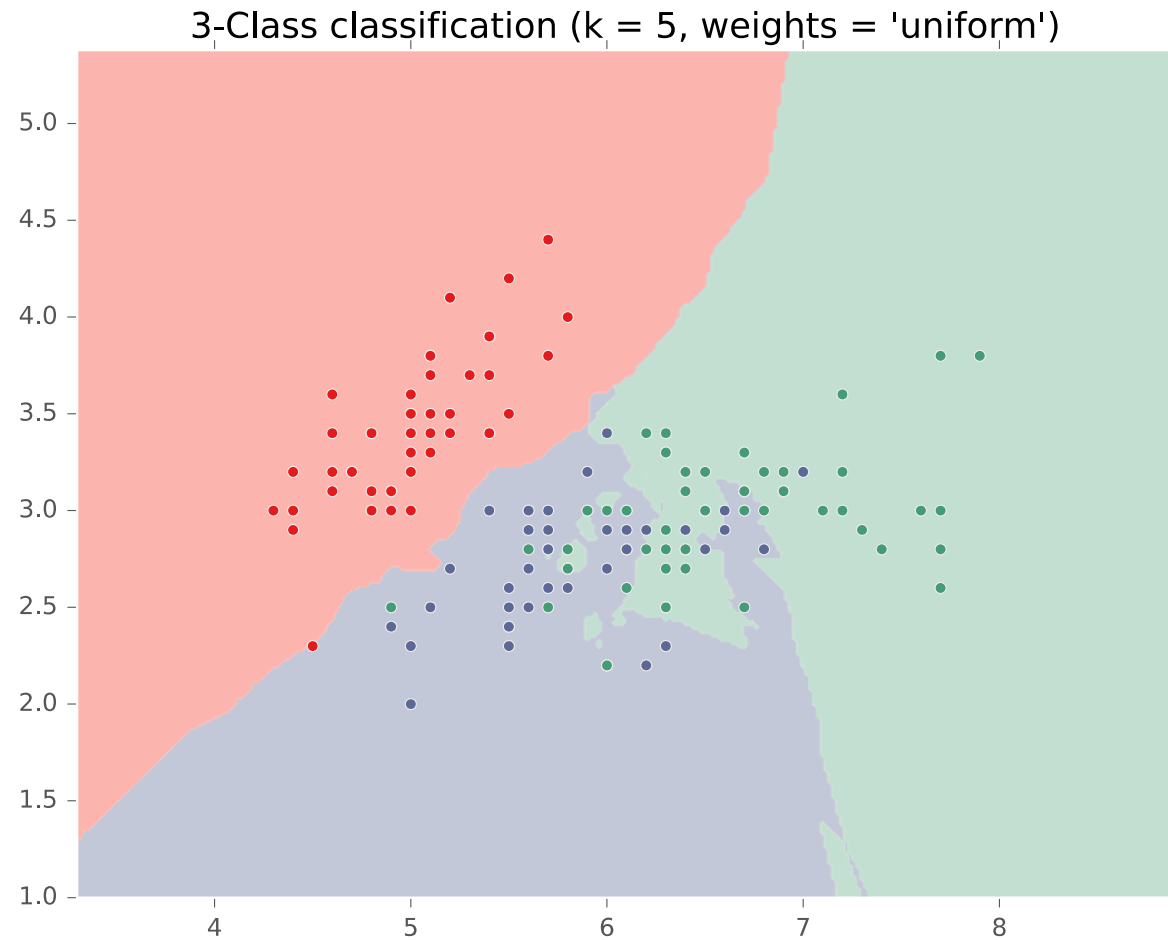
# k-NN on Fisher Iris Data



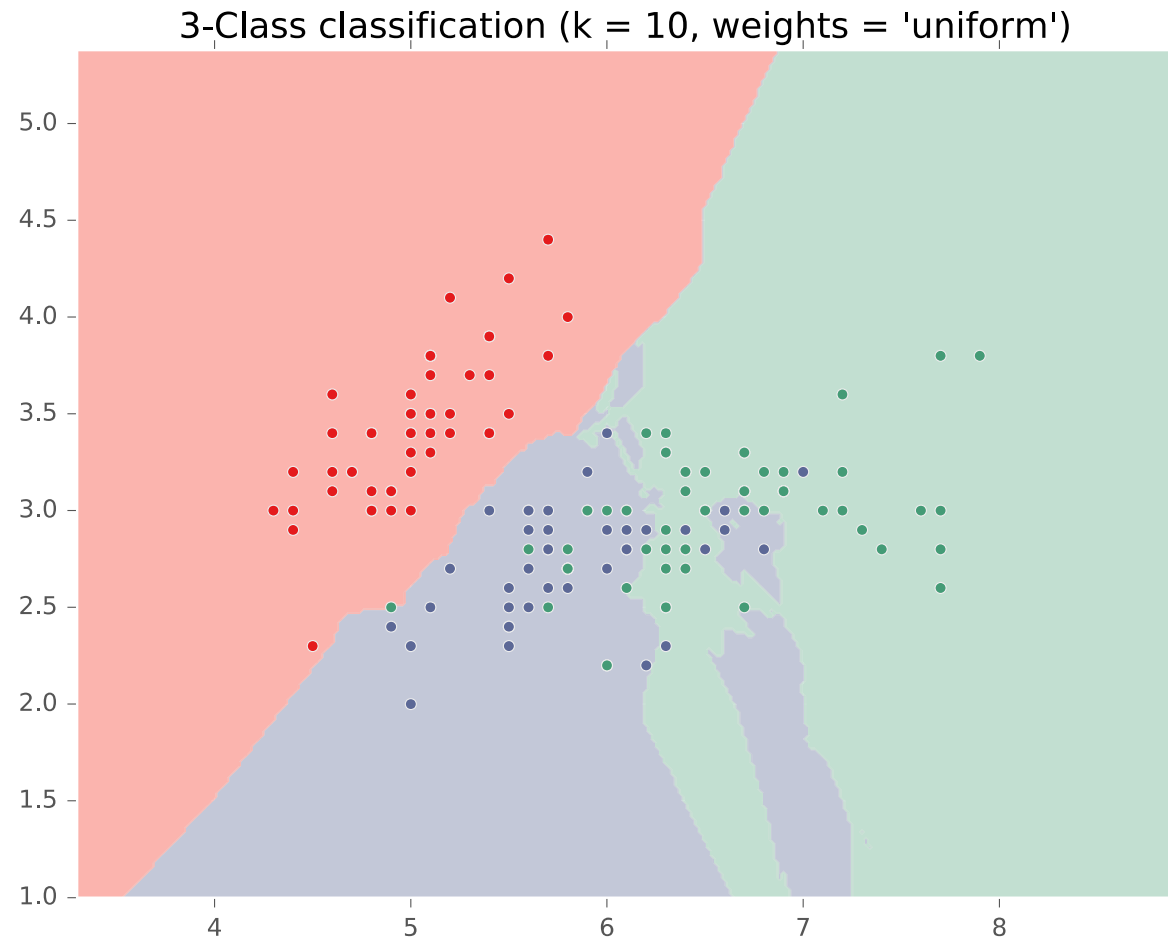
# k-NN on Fisher Iris Data



# k-NN on Fisher Iris Data

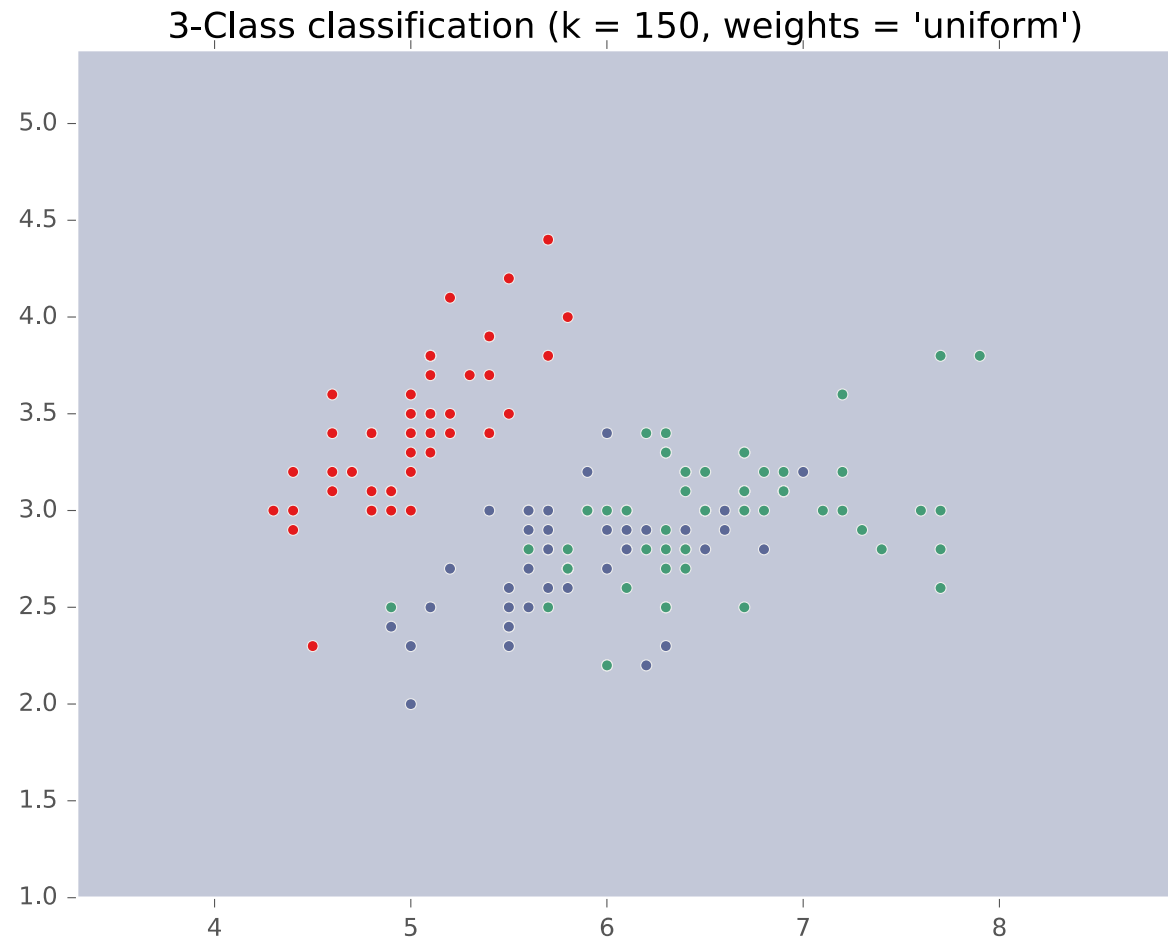


# k-NN on Fisher Iris Data



# k-NN on Fisher Iris Data

## Special Case: Majority Vote



# k-NN: Remarks

## **Inductive Bias:**

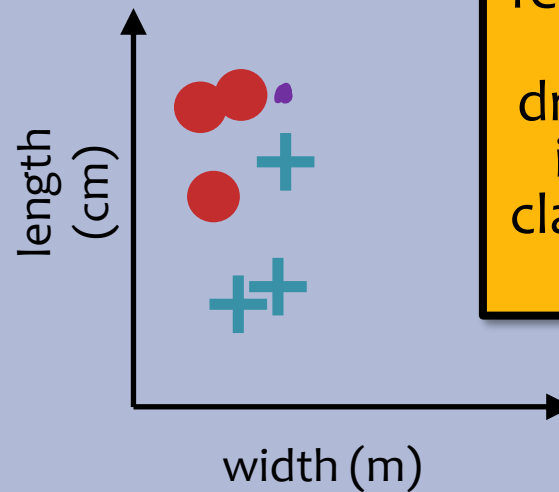
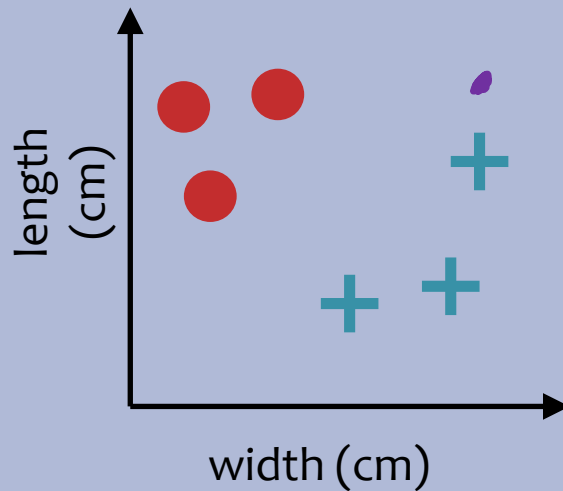
1. Close points should have similar labels
2. All dimensions are created equally!

# k-NN: Remarks

## Inductive Bias:

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

Example: two features for k-NN



**big problem:**  
feature scale  
could  
dramatically  
influence  
classification  
results



# k-NN: Remarks

## Computational Efficiency:

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

# Piazza Poll 3 (train) and Poll 4 (test)

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

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

A.  $O(1)$  *train*

B.  $O(\log N)$

C.  $O(\log M)$

D.  $O(\log NM)$

E.  $O(N)$

F.  $O(M)$

G.  $O(NM)$  *test*

H.  $O(N^2)$

I.  $O(N^2M)$

*def train(D)*

*(3)*

*store D*

*def h( $\vec{x}_{\text{test}}$ )*

*(4)*

*for i in 1..N*

*$\rightarrow d(\vec{x}_{\text{test}}, \vec{x}^{(i)})$*

*def d( $\vec{x}, \vec{z}$ )*

*for j in 1..M*

*$x_j - z_j$*

# Piazza Poll 3 (train) and Poll 4 (test)


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)

# k-NN Learning Objectives

*You should be able to...*

- Describe a dataset as points in a high dimensional space
- Implement k-Nearest Neighbors with  $O(N)$  prediction
- Describe the inductive bias of a k-NN classifier and relate it to feature scale
- Sketch the decision boundary for a learning algorithm (compare k-NN and DT)

# **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. C4.5
- 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 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 space which learning performs its search over
- **parameters** are the numeric values of the model 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

If “learning” is all about picking the best **parameters** how do we pick the best **hyperparameters**?

# Model Selection

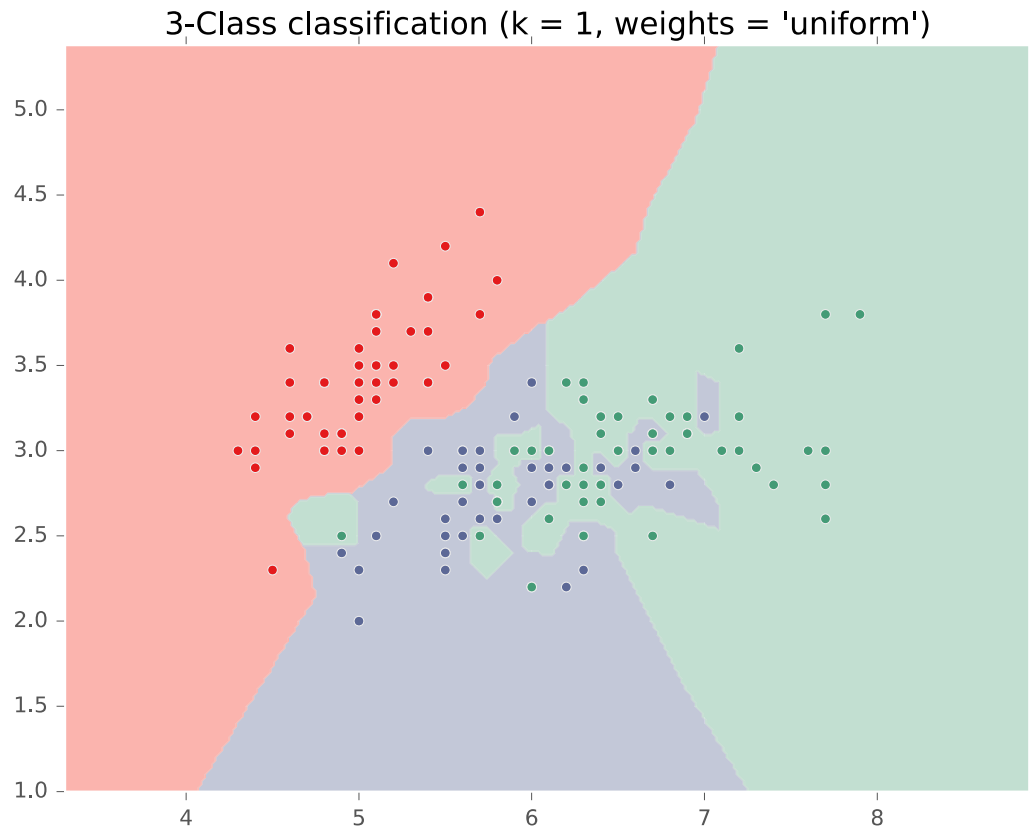
- Two very similar definitions:
  - Def: **model selection** is the process by which we choose the “best” model from among a set of candidates
  - Def: **hyperparameter optimization** is the process by which we choose the “best” hyperparameters from among a set of candidates (**could be called a special 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

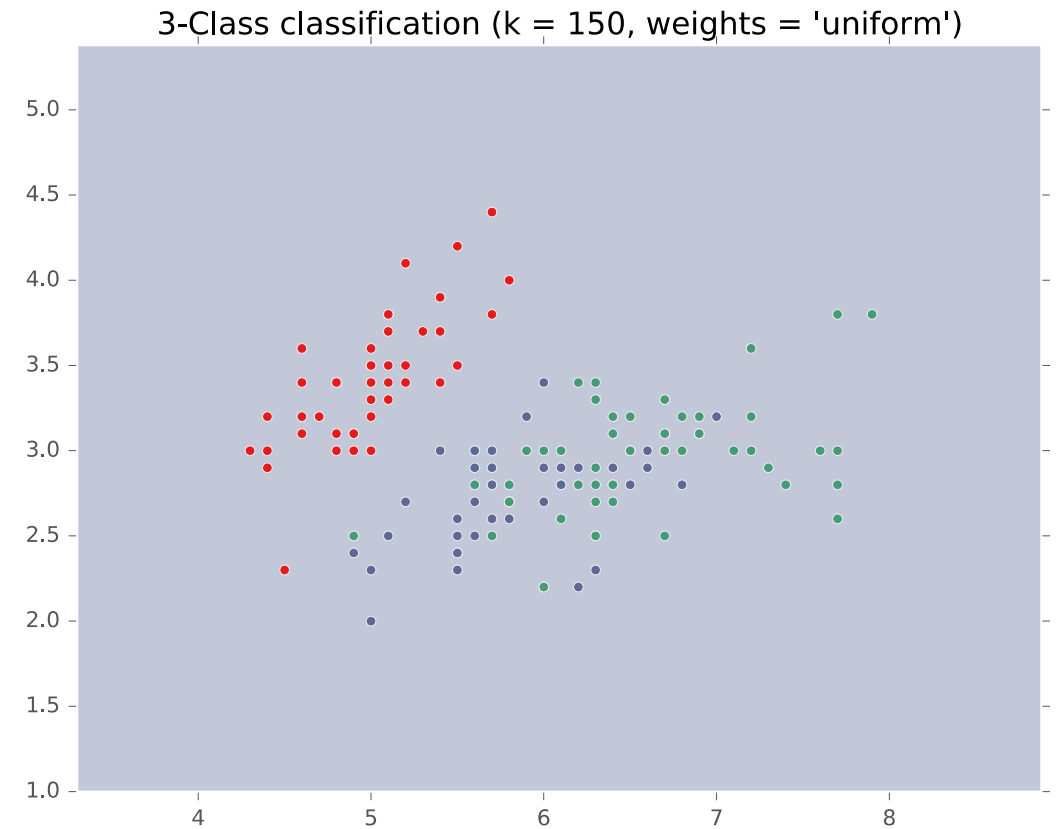
	Input	Output	Notes
Training	<ul style="list-style-type: none"><li>• training dataset</li><li>• hyperparameters</li></ul>	<ul style="list-style-type: none"><li>• best model parameters</li></ul>	We pick the best model parameters by learning on the training dataset for a fixed set of hyperparameters
Hyperparameter Optimization	<ul style="list-style-type: none"><li>• training dataset</li><li>• validation dataset</li></ul>	<ul style="list-style-type: none"><li>• <u>best hyperparameters</u></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

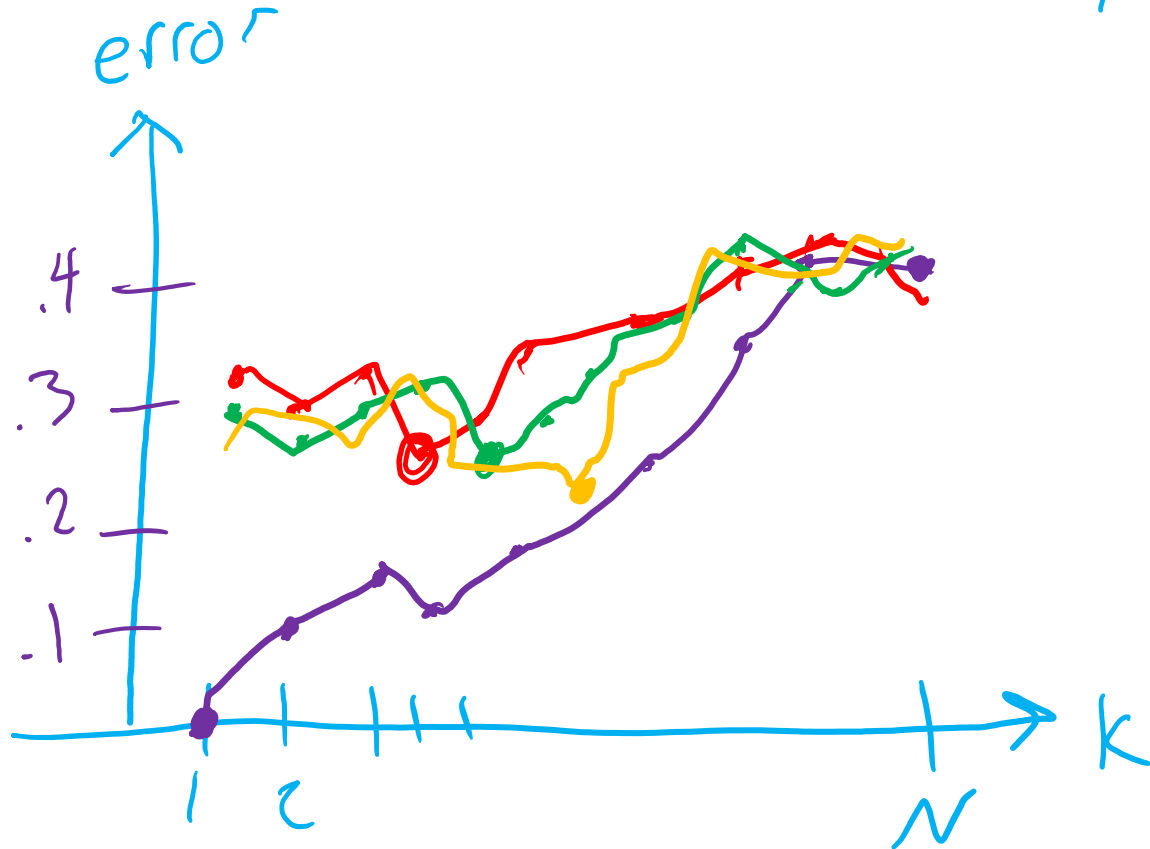


## k=N: Majority Vote



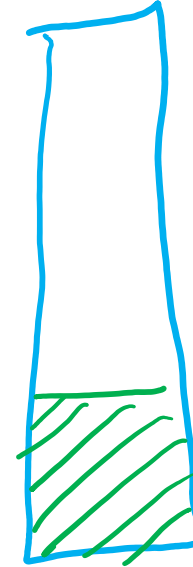
# Example of Hyperparameter Optimization

Choosing  $k$  for  $k$ -NN



$\gamma = 0$  40%  $\gamma = 1$  60%

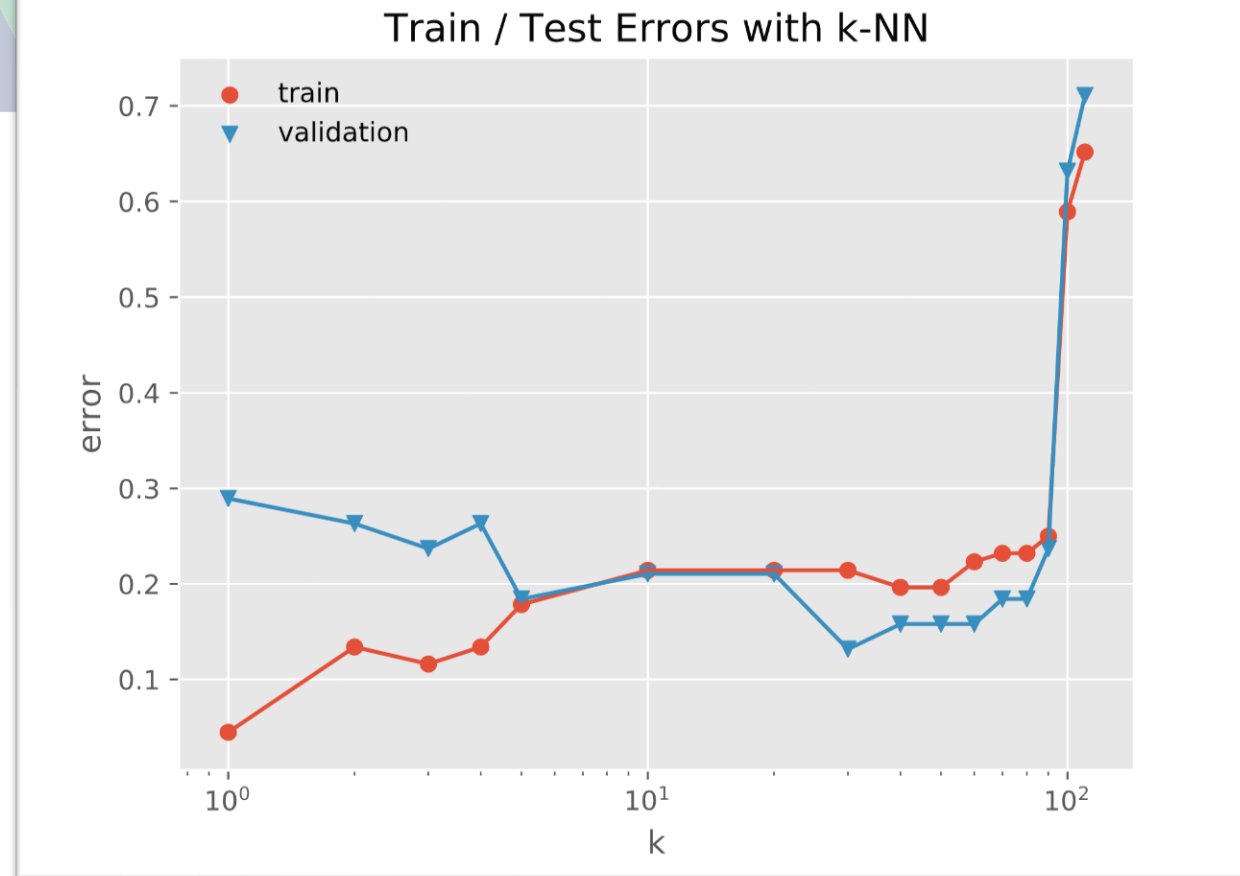
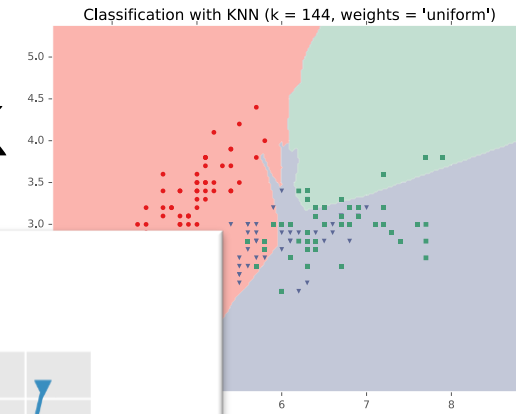
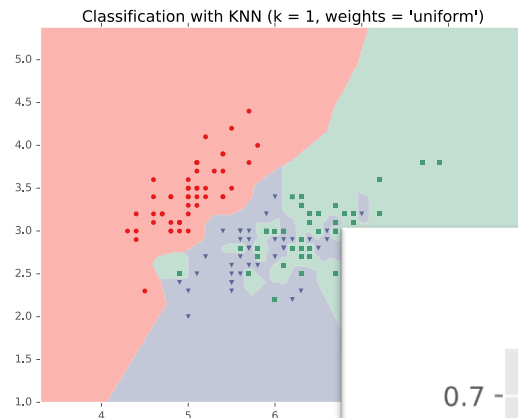
1200



Val 300



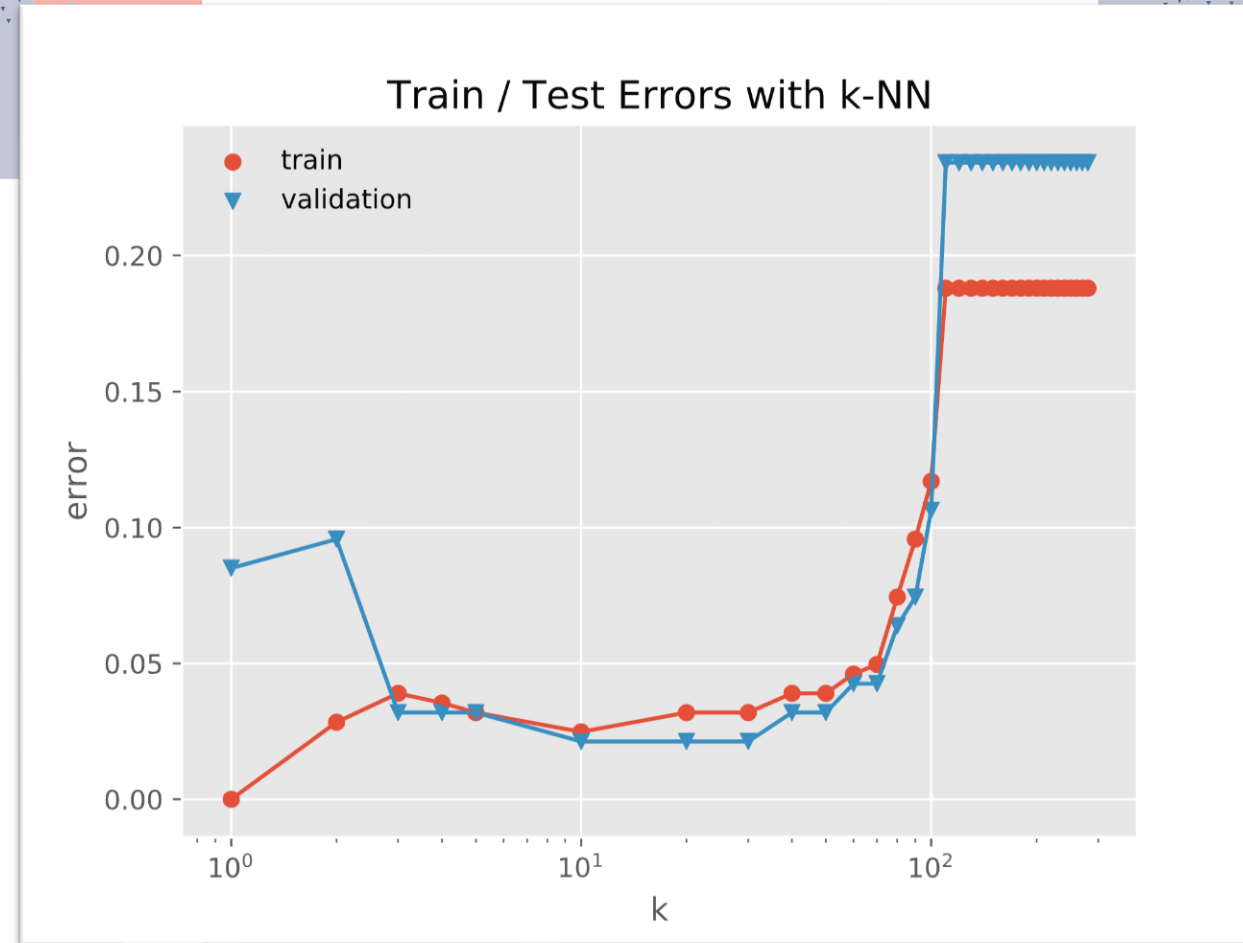
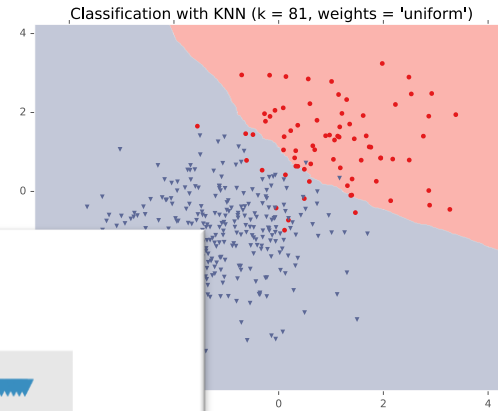
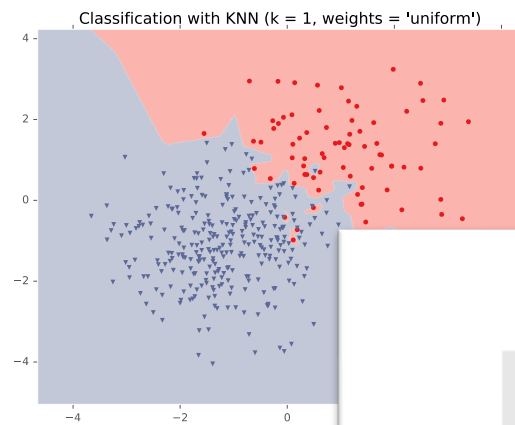
# k-NN: Choosing k



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!

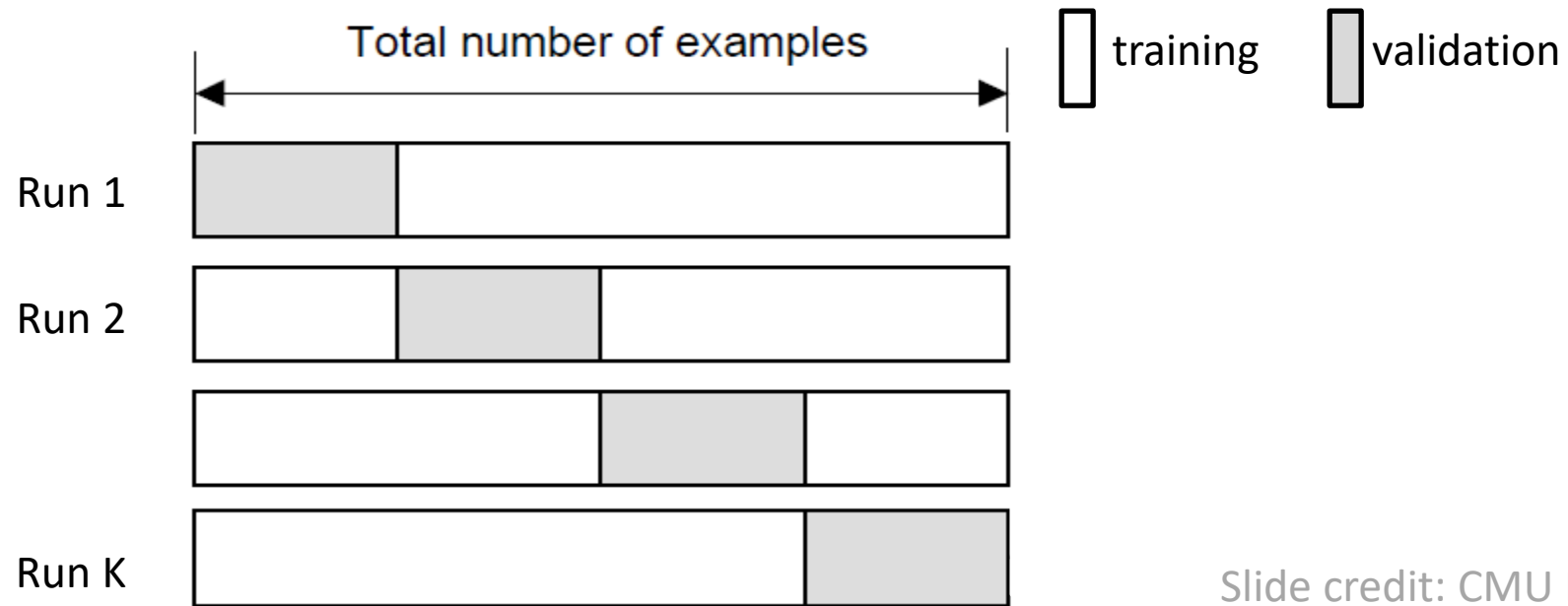
# Cross-validation

## 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

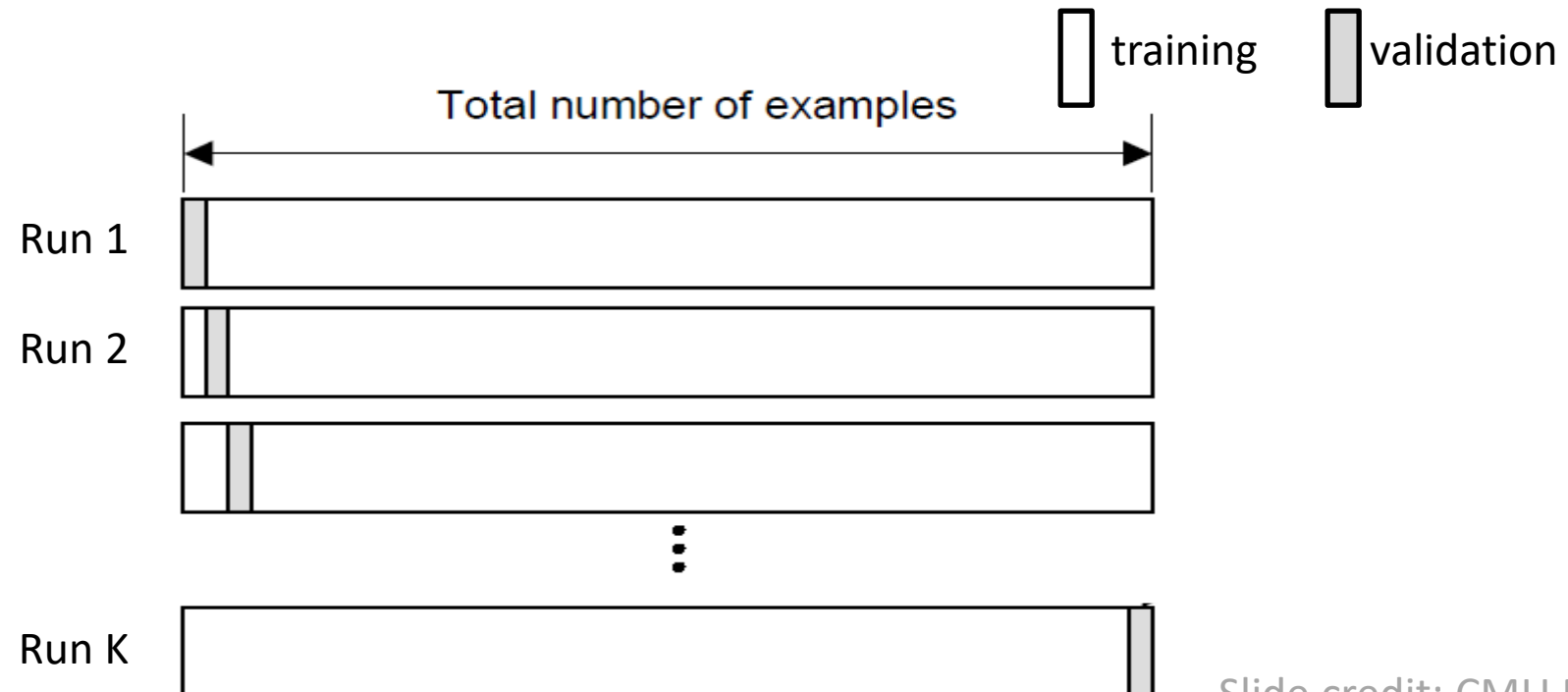


# Cross-validation

## 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

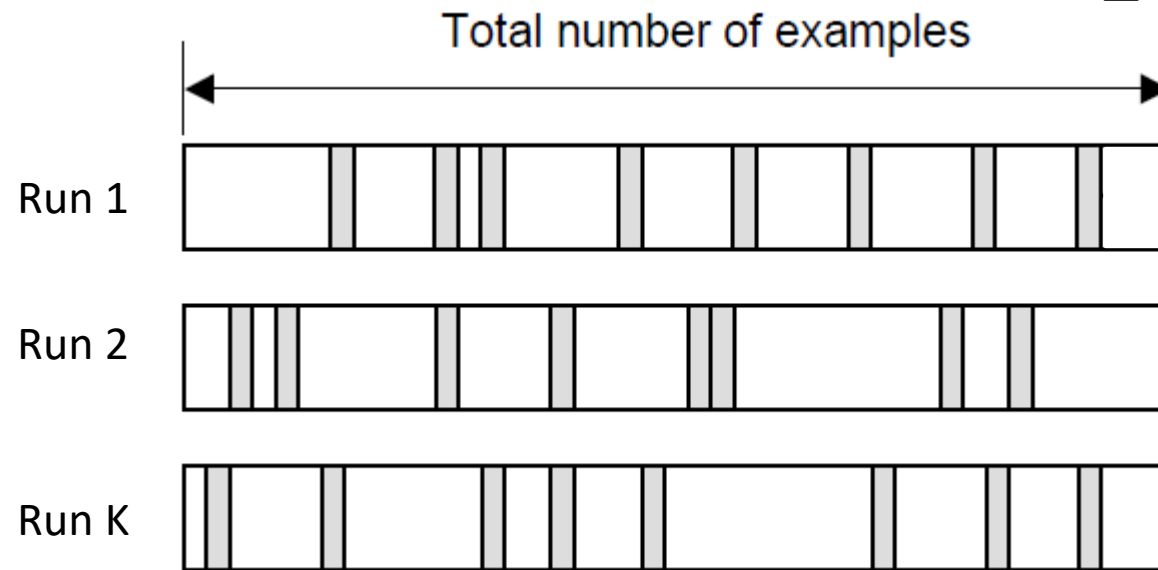
Randomly subsample a fixed fraction  $\alpha N$  ( $0 < \alpha < 1$ ) of the dataset for validation.

Compute validation error with remaining data as training data.

Repeat K times

Report average validation error

□ training    □ validation



# Practical Issues in Cross-validation

How to decide the values for  $K$  and  $\alpha$  ?

- Large  $K$ 
  - + Validation error can approximate test error well
  - Observed validation error will be unstable (few validation pts)
  - The computational time will be very large as well (many experiments)
- Small  $K$ 
  - + The # experiments and, therefore, computation time are reduced
  - + Observed validation error will be stable (many validation pts)
  - Validation error cannot approximate test error well

Common choice:  $K = 10$ ,  $\alpha = 0.1$  😊

# 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
  - Graduate-student descent
  - ...

## **Main Takeaway:**

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

# Model Selection Learning Objectives

*You should be able to...*

- Plan an experiment that uses training, validation, and test datasets to predict the performance of a classifier on unseen data (without cheating)
- Explain the difference between (1) training error, (2) validation error, (3) cross-validation error, (4) test error, and (5) true error
- For a given learning technique, identify the model, learning algorithm, parameters, and hyperparameters