Classification and K-Nearest Neighbors

Administrivia

- **Reminder**: Homework 1 is due by 5pm Friday on Moodle
- **Reading Quiz** associated with today's lecture. Due before class Wednesday.
- NOTE TAKER

Regression vs Classification

So far, we've taken a vector of features $(x_1, x_2, \ldots, x_p)$ and predicted a numerical response

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots \hat{\beta}_p x_p$$

$$\$360,000 = \beta_0 + \beta_1 \times (2000 \text{ sq-ft}) + \beta_2 \times (3 \text{ bath rooms})$$

Regression deals with **quantitative** predictions

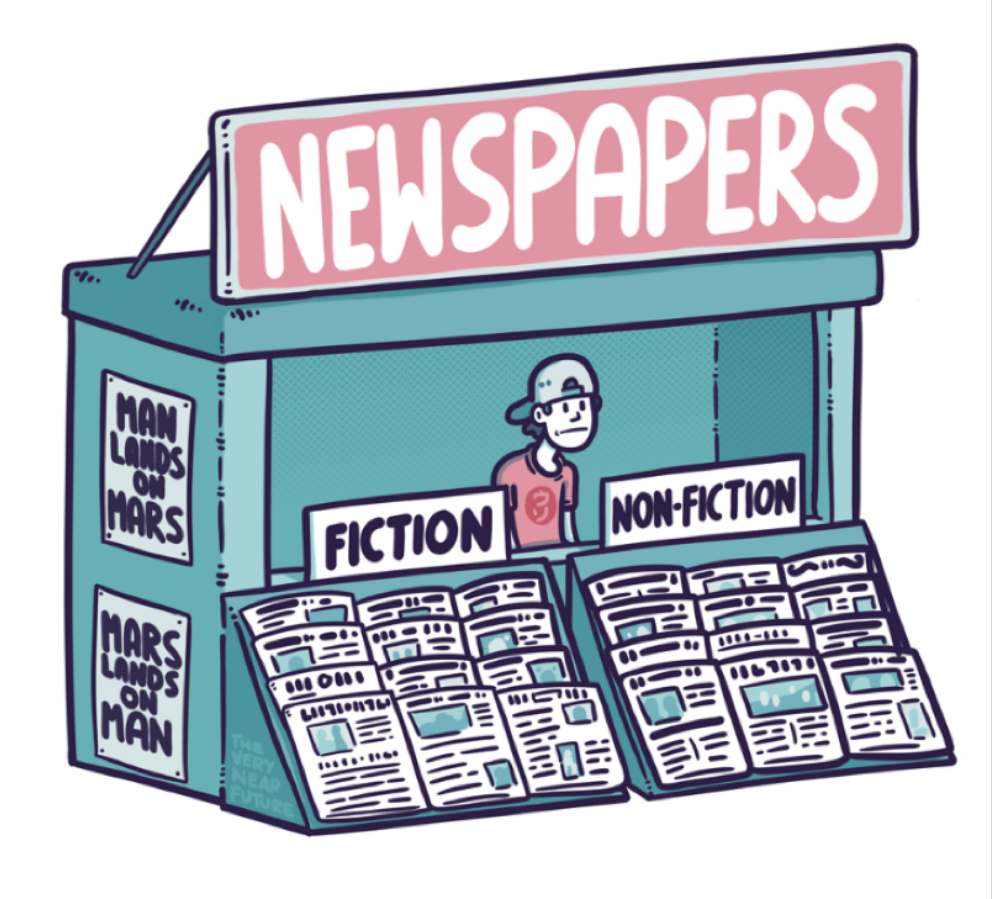
But what if we're interested in making **qualitative** predictions?

Classification Examples

Text Classification: Spam or Not Spam

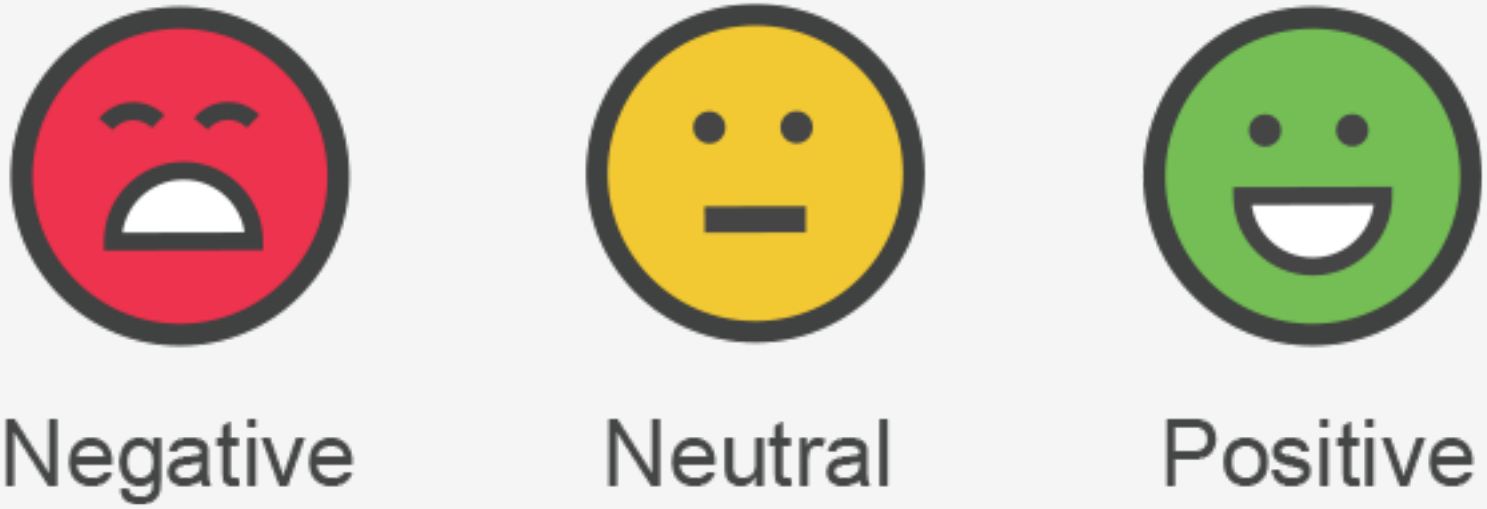
Classification Examples

Text Classification: Fake News or Real News



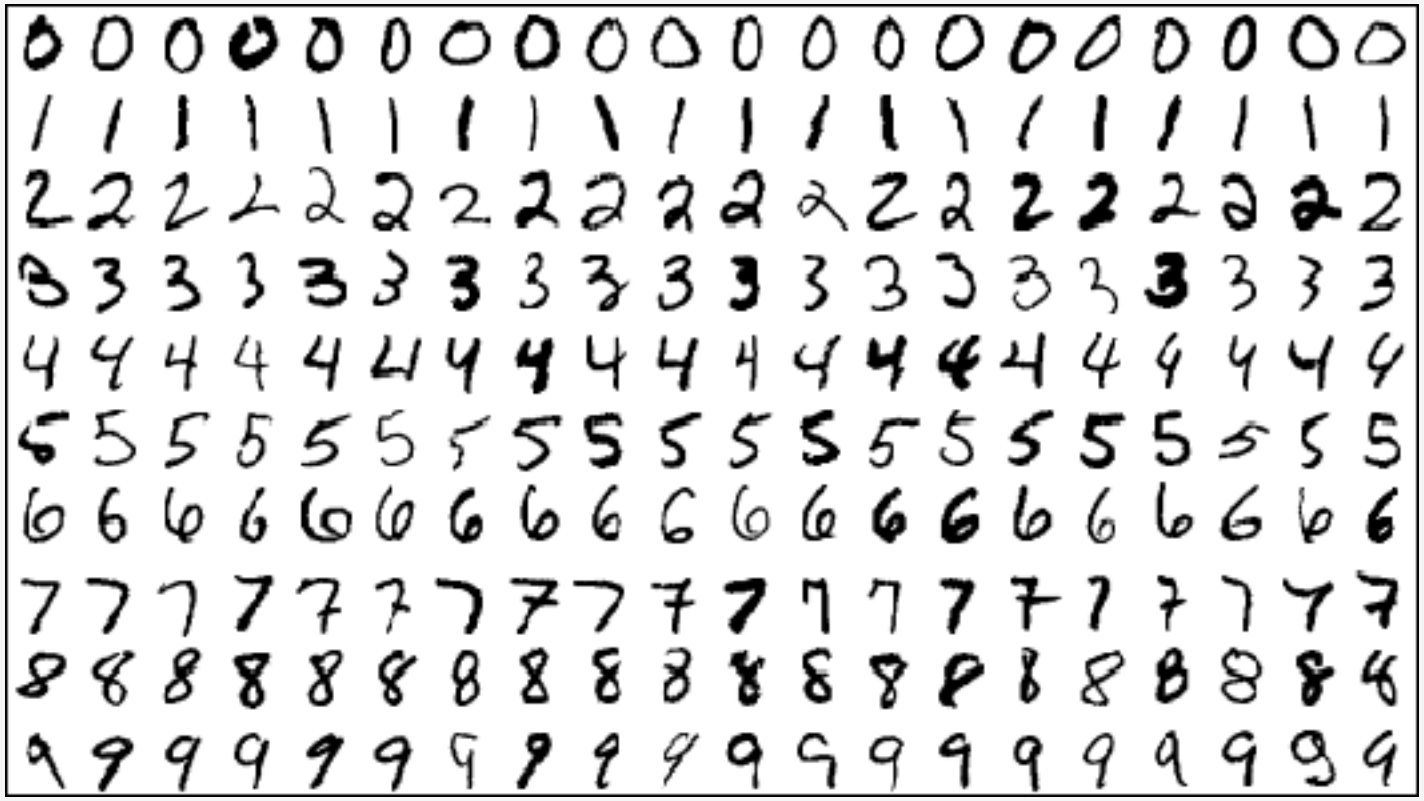
Classification Examples

Text Classification: Sentiment Analysis (Tweets, Amazon Reviews)



Classification Examples

Image Classification: Handwritten Digits

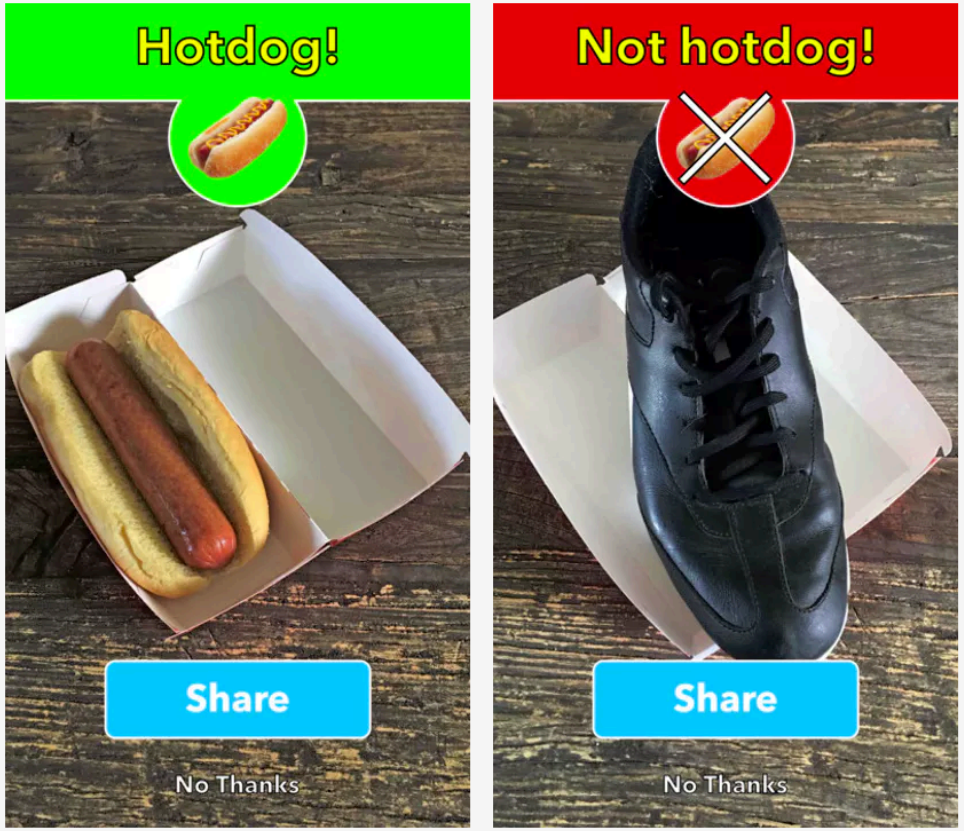


Classification Examples

Image Classification: Stop Sign or No Stop Sign (Autonomous Vehicles)

Classification Examples

Image Classification: Hot Dog or Not Hot Dog (No real application whatsoever)



Classification Examples

Image Classification: Labradoodle or Fried Chicken (Important for not eating your dog)

Classification with Probability

In classification problems our outputs are discrete **classes**

Goal: Given features $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ predict which class $Y = k$, $\mathbf{x}$ belongs to

Classification with Probability

In classification problems our outputs are discrete **classes**

Goal: Given features $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ predict which class $Y = k$, $\mathbf{x}$ belongs to

Conditional probability gives a nice framework for many classification models

Suppose our data can belong to one of $k = 1, 2, \ldots, K$ classes

We model the conditional probability $p(Y = k \mid \mathbf{x})$ for each $k = 1, 2, \ldots, K$

and we make the prediction $\hat{y} = \underset{k}{\operatorname{argmax}}\, p(Y = k \mid \mathbf{x})$

Classification with Probability

In classification problems our outputs are discrete **classes**

Goal: Given features $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ predict which class $Y = k$, $\mathbf{x}$ belongs to

Example: In Spam Classification, the features are the words in the email

The classes are SPAM and HAM

We estimate $\hat{p}(\text{SPAM} \mid \text{email})$ and $\hat{p}(\text{HAM} \mid \text{email})$

$= 0.75$ $= 0.25$

$\hat{y} = \text{SPAM}$

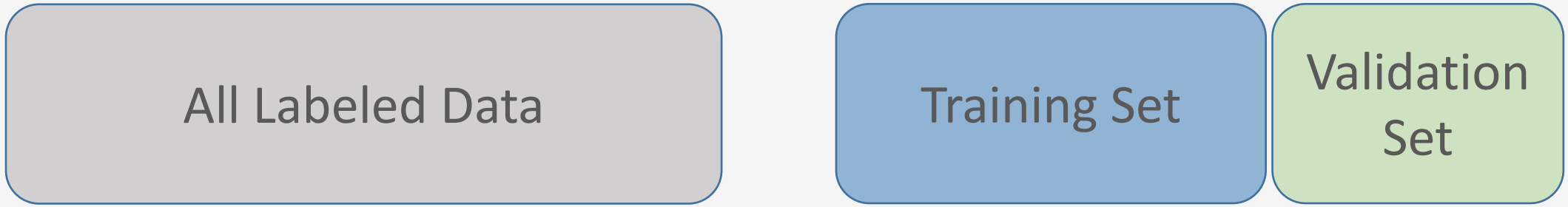
K-Nearest Neighbors

Our first classifier is a very simple one

Assume we have a labeled training set

Features can be anything: words, pixel intensities, numerical quantities

Instead of continuous response, now have discrete labels (encoded as 0, 1, 2, etc)



K-Nearest Neighbors

General Idea: Suppose we have an example $\mathbf{x}$ that we want to classify

- Look in training set for K examples that are "nearest" to $\mathbf{x}$
- Predict label for $\mathbf{x}$ using majority vote of labels for K nearest neighbors

Questions:

- What does "nearest" mean?
- What if there's a tie in the vote?

K-Nearest Neighbors

KNN:

- Find $\mathcal{N}_K(\mathbf{x})$: the set of K training examples nearest to $\mathbf{x}$
- Predict $\hat{y}$ to be majority label in $\mathcal{N}_K(\mathbf{x})$

Question:

- What does "nearest" mean?

Answer:

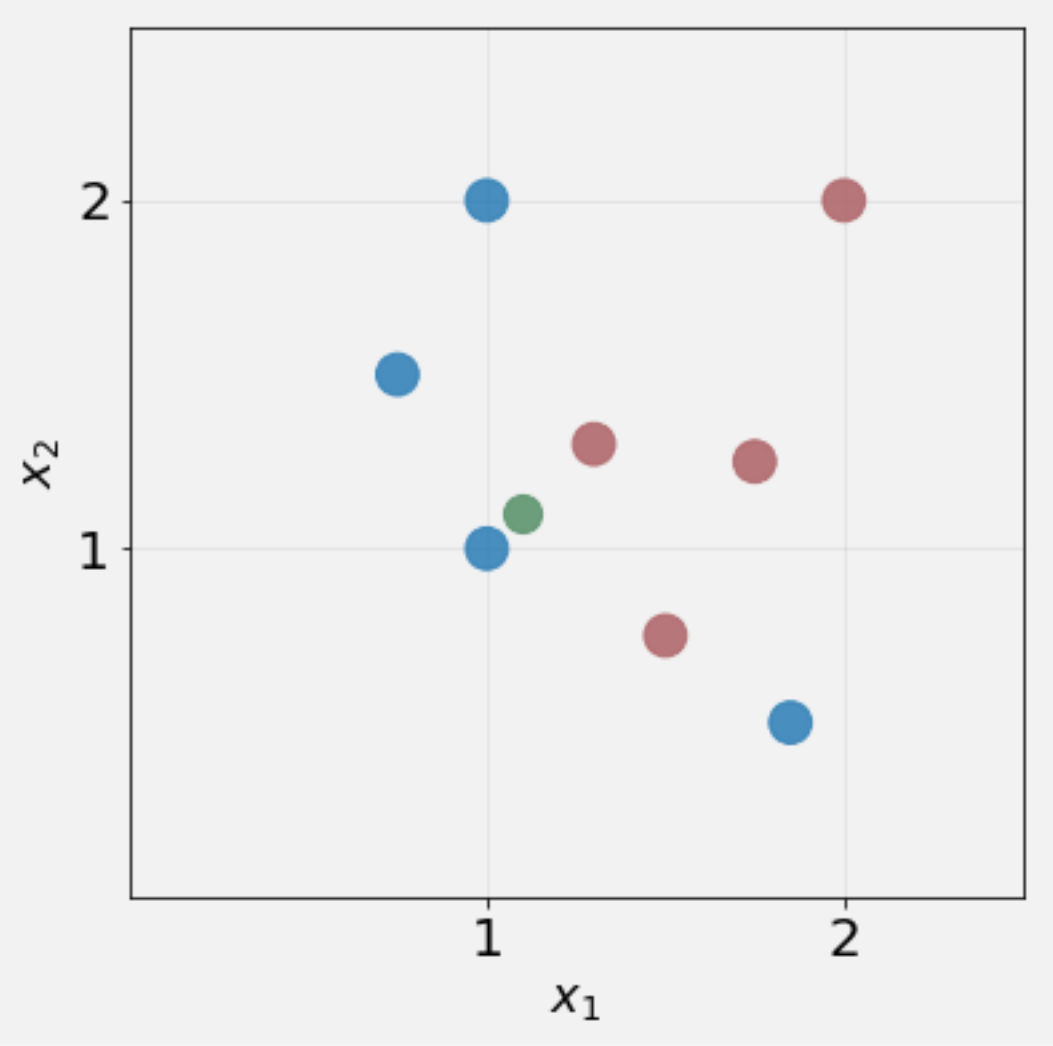
- Measure closeness using Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

test example

training vector

K-Nearest Neighbors

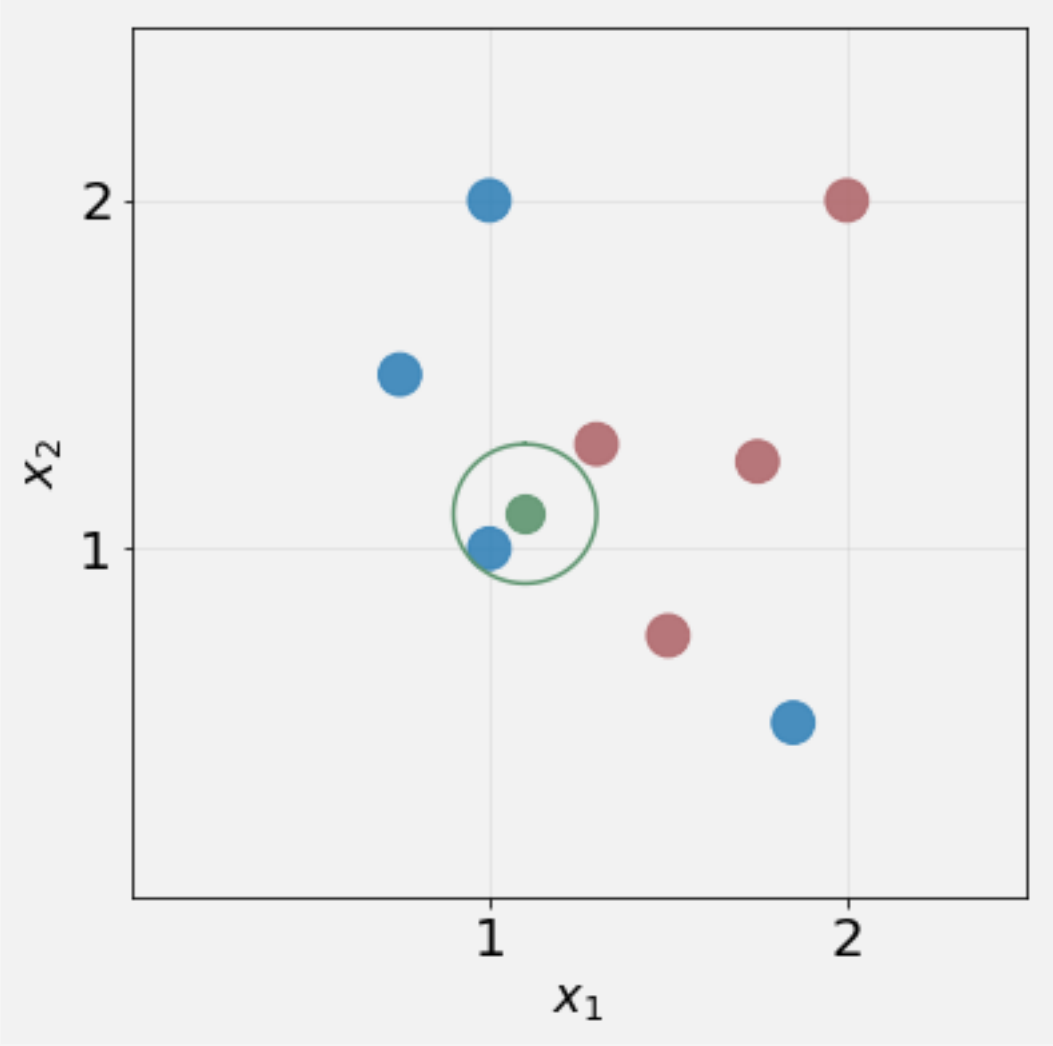
Classes = {Red, Blue}



Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

- 1-NN predicts: _____
- 2-NN predicts: _____
- 5-NN predicts: _____

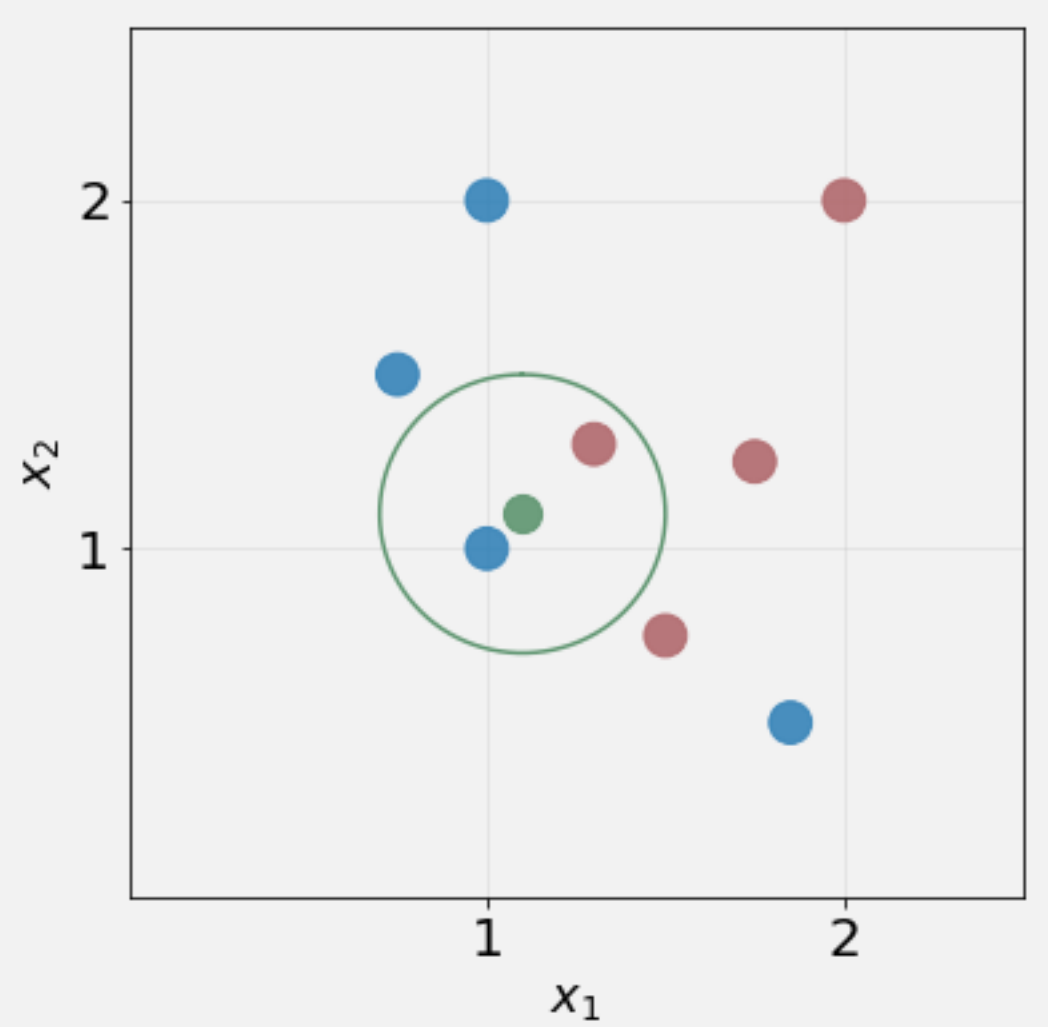
K-Nearest Neighbors



Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

- 1-NN predicts: BLUE
- 2-NN predicts: _____
- 5-NN predicts: _____

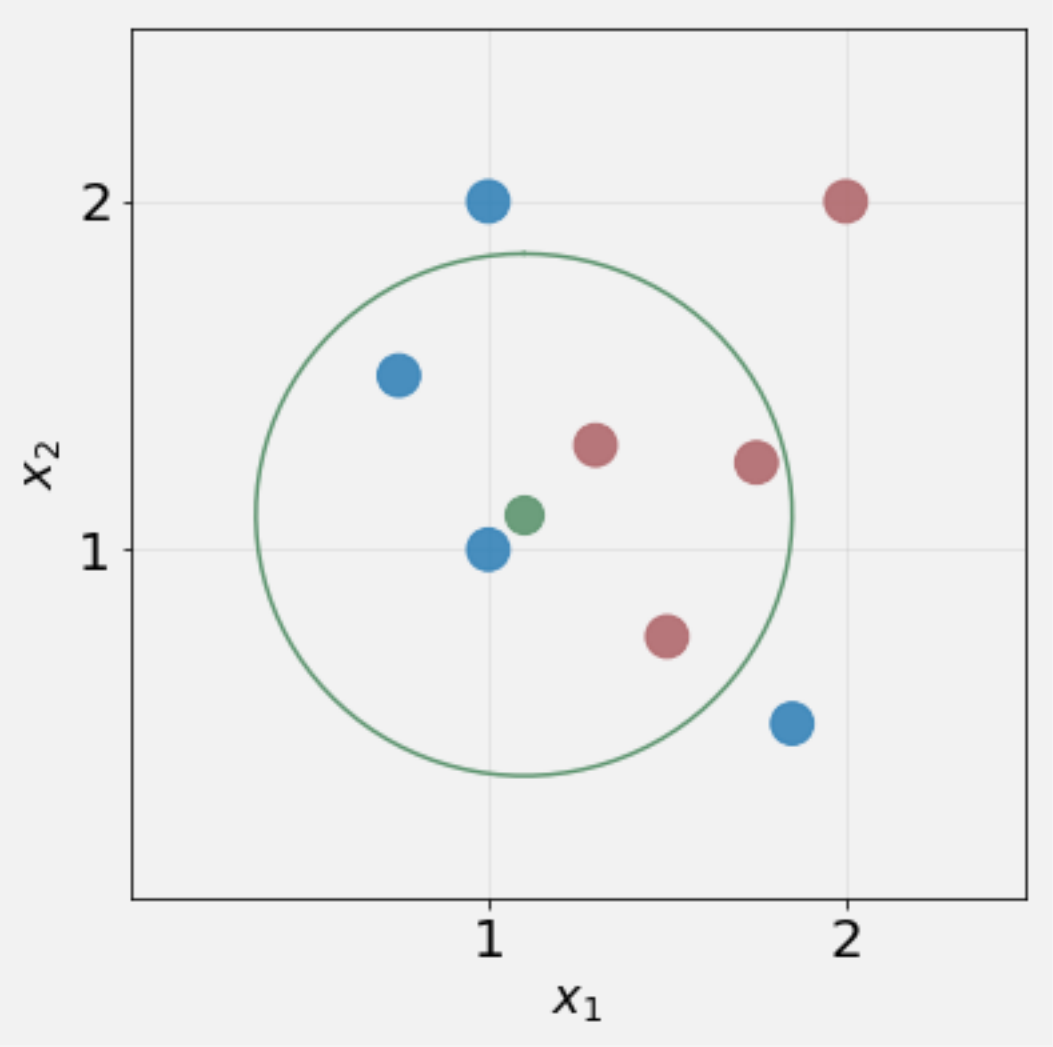
K-Nearest Neighbors



Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

- 1-NN predicts: blue
- 2-NN predicts: unclear
- 5-NN predicts: _____

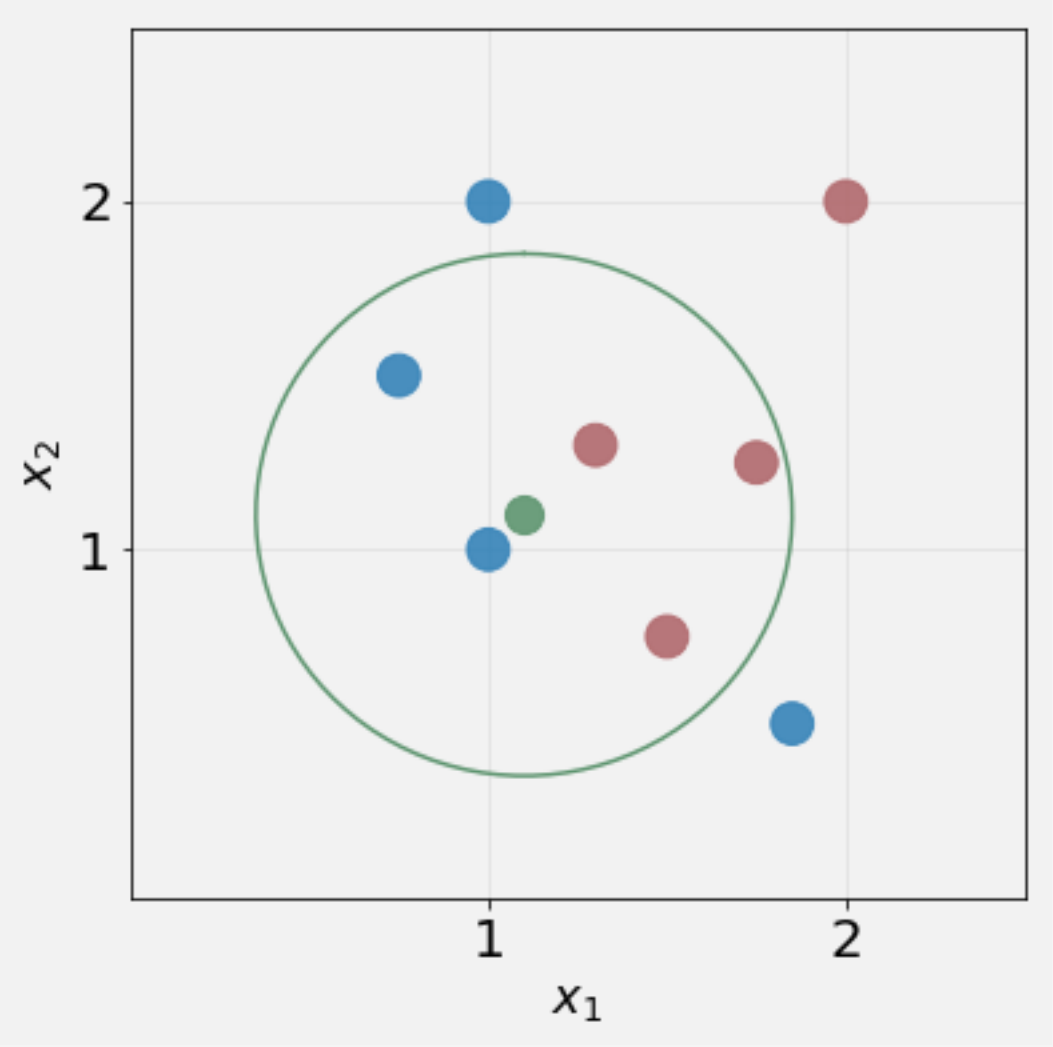
K-Nearest Neighbors



Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

- 1-NN predicts: blue
- 2-NN predicts: unclear
- 5-NN predicts: red

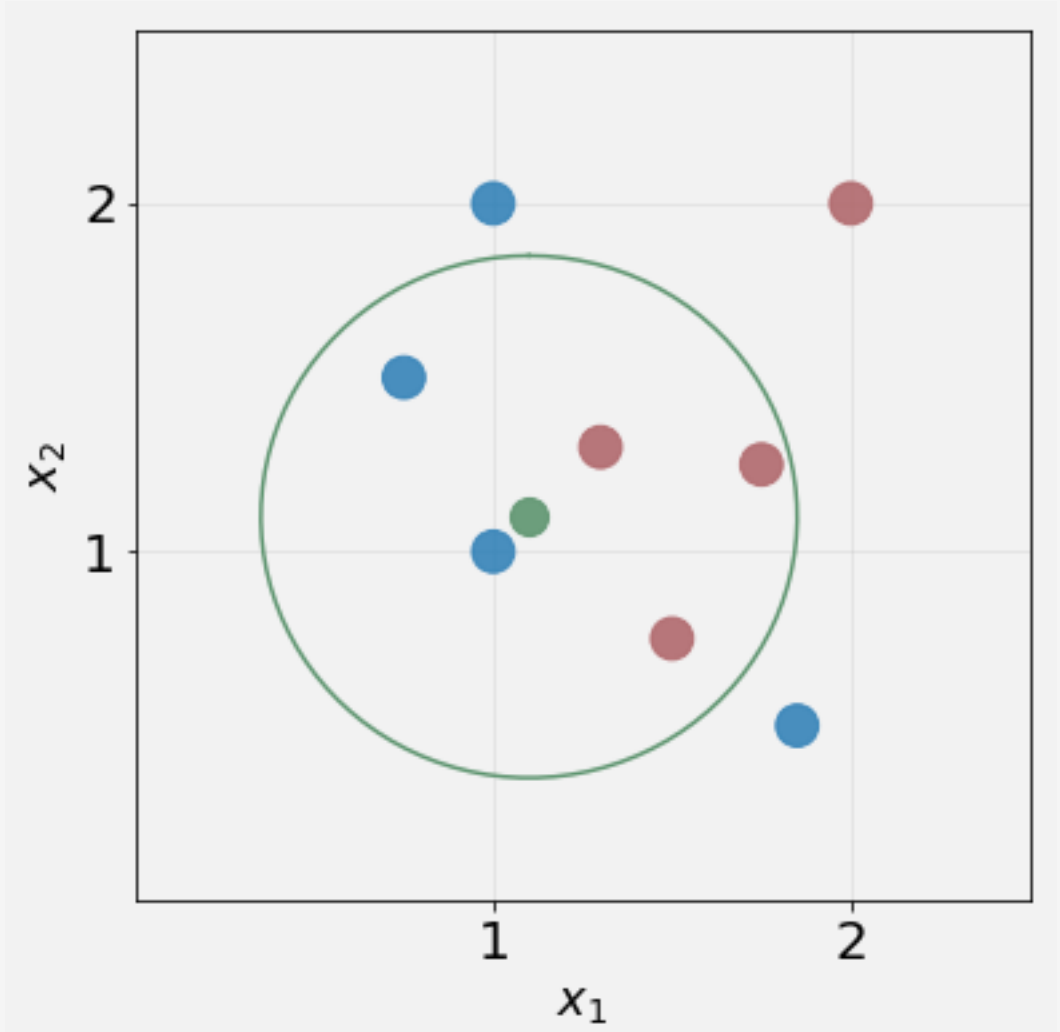
K-Nearest Neighbors



Euclidean Distance: $\|\mathbf{x}_i - \mathbf{x}\|^2$

- 1-NN predicts: blue
- 2-NN predicts: unclear
- 5-NN predicts: red

What About Probability?



INDICATOR FUNCTION

$$\hat{p}(Y = k \mid \mathbf{x}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K} I(y_i = k)$$

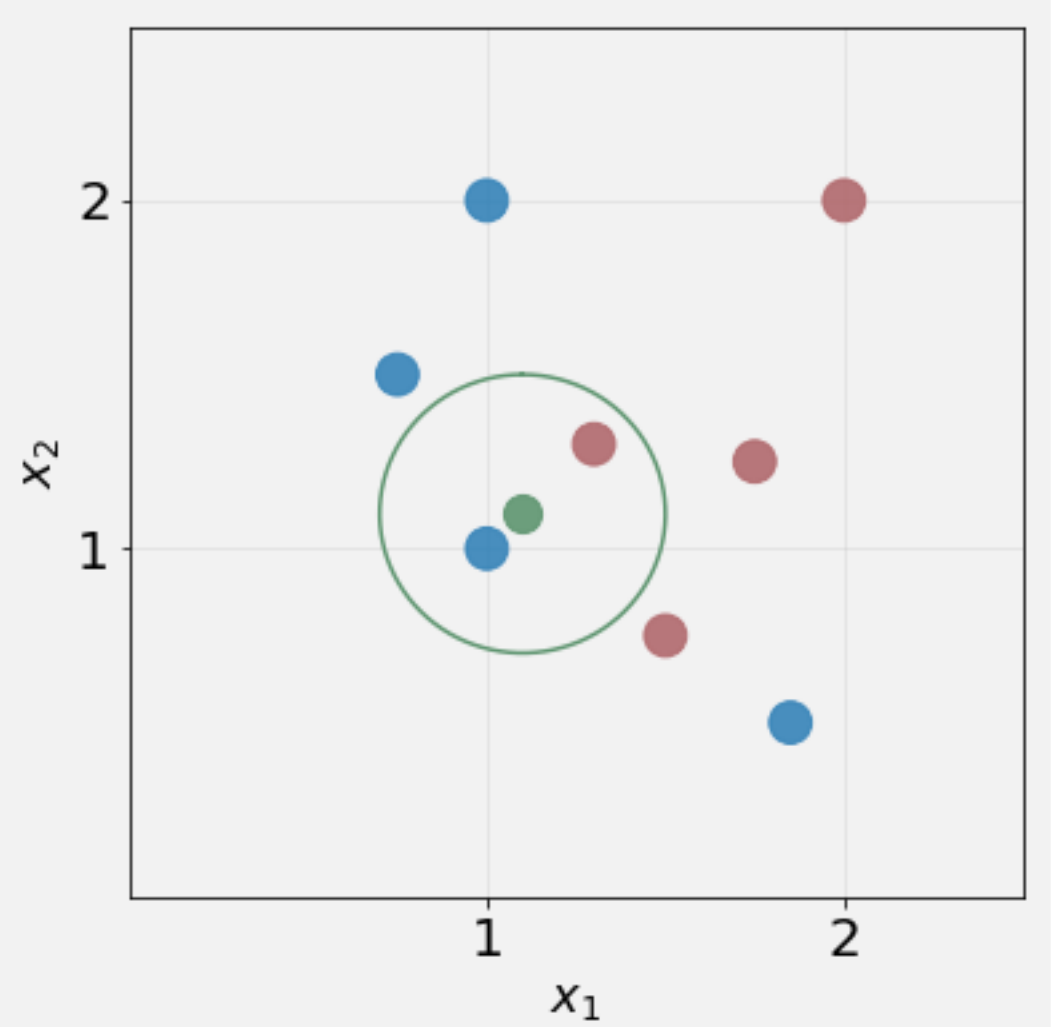
- 5-NN:

$\hat{p}(Y = \text{BLUE} \mid x) = \frac{2}{5}$

$\tilde{p}(Y = \text{RED} \mid x) = \frac{3}{5}$

$$I(b) = \begin{cases} 1 & \text{IF } b \text{ IS TRUE} \\ 0 & \text{IF } b \text{ IS FALSE} \end{cases}$$

What About Ties?

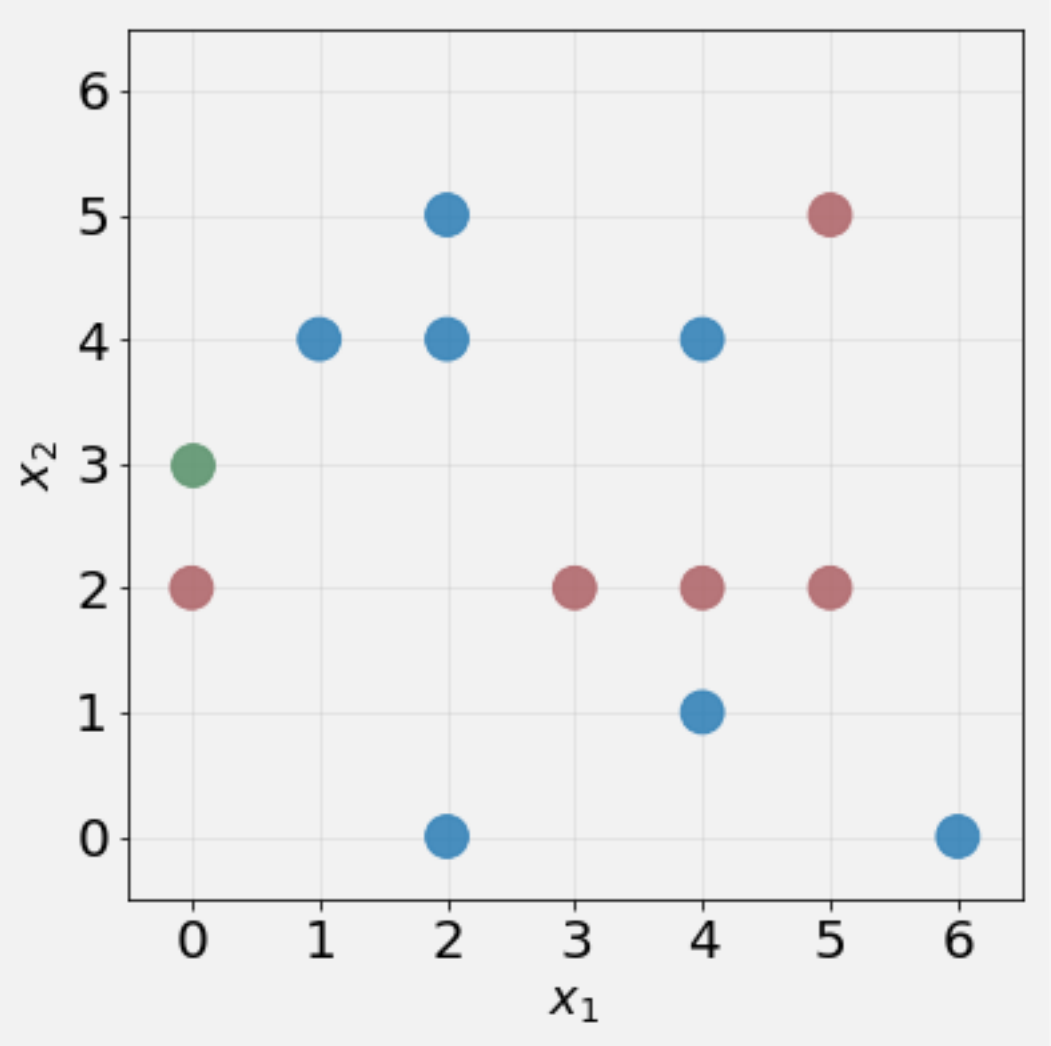


Lots of reasonable strategies

Example: If there is a tie:

- reduce K by 1 and try again.
- repeat until the tie is broken.

K-Nearest Neighbors



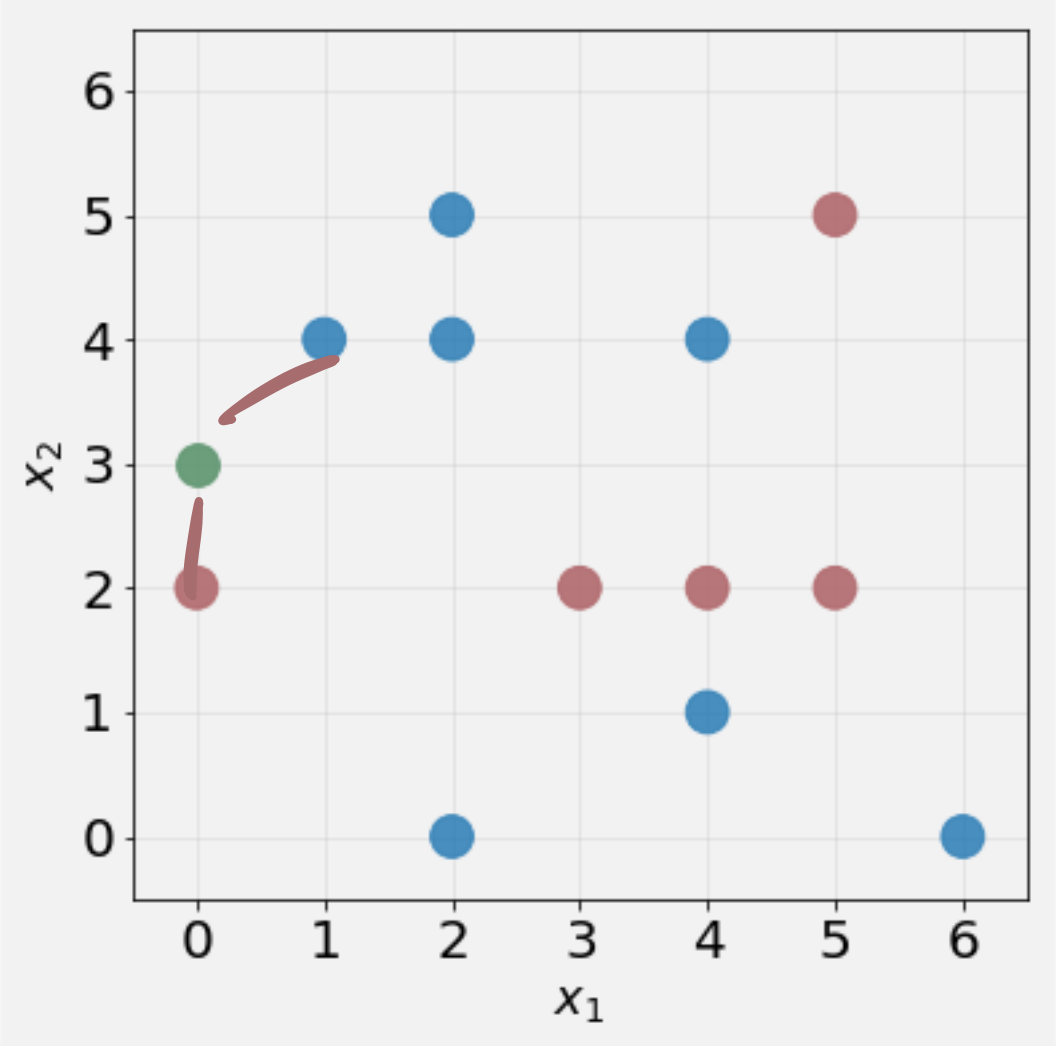
Predict $\hat{y}$ with $K = 1$

Closest Points:

$N_1(x) = \{(0,2)\}$

Prediction: $\hat{y} = \text{RED}$

K-Nearest Neighbors



Predict $\hat{y}$ with $K = 2$

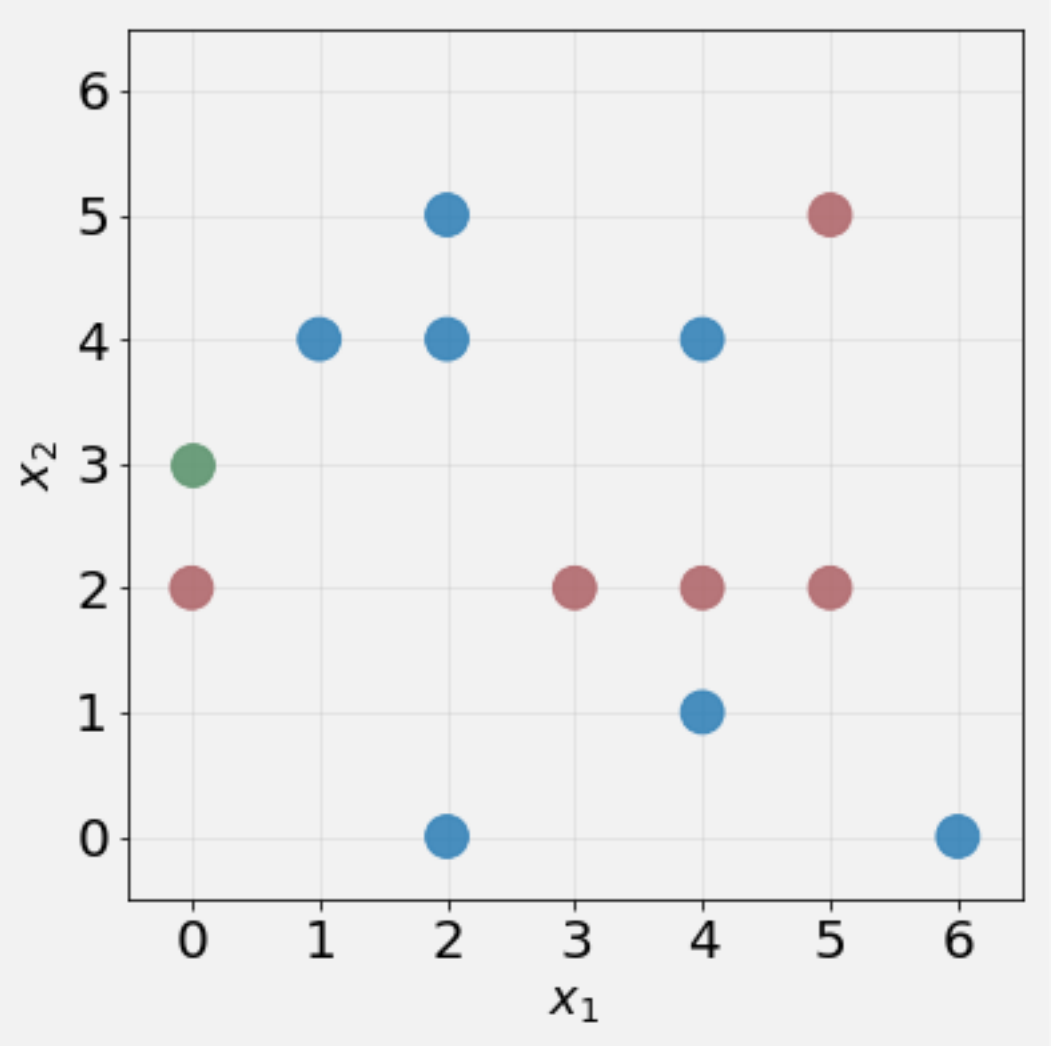
Closest Points:

$N_2(x) = \{(0,2), (1,4)\}$

$N_1(1) = \{(0,2)\}$

Prediction: RED $= \tilde{y}$

K-Nearest Neighbors



Predict $\hat{y}$ with $K = 3$

Closest Points:

$N_3(x) = \{(0,2), (1,4), (2,4)\}$

Prediction: $\hat{y}$ = BLUE

A Subtle Distinction

- KNN is an **instance-based method**
- Given an unseen query point, look directly at training data and then predict
- Compare to regression where we use training data to **learn** parameters
- When we learn parameters we call the model a **parametric model**
- Instance-based methods like KNN are called **non-parametric models**

Question: When would parametric vs non-parametric make a big difference?

Naïve Implementation

How much does KNN cost?

$\|x_i - x\|^2$

Suppose your training set has n examples, each with p features

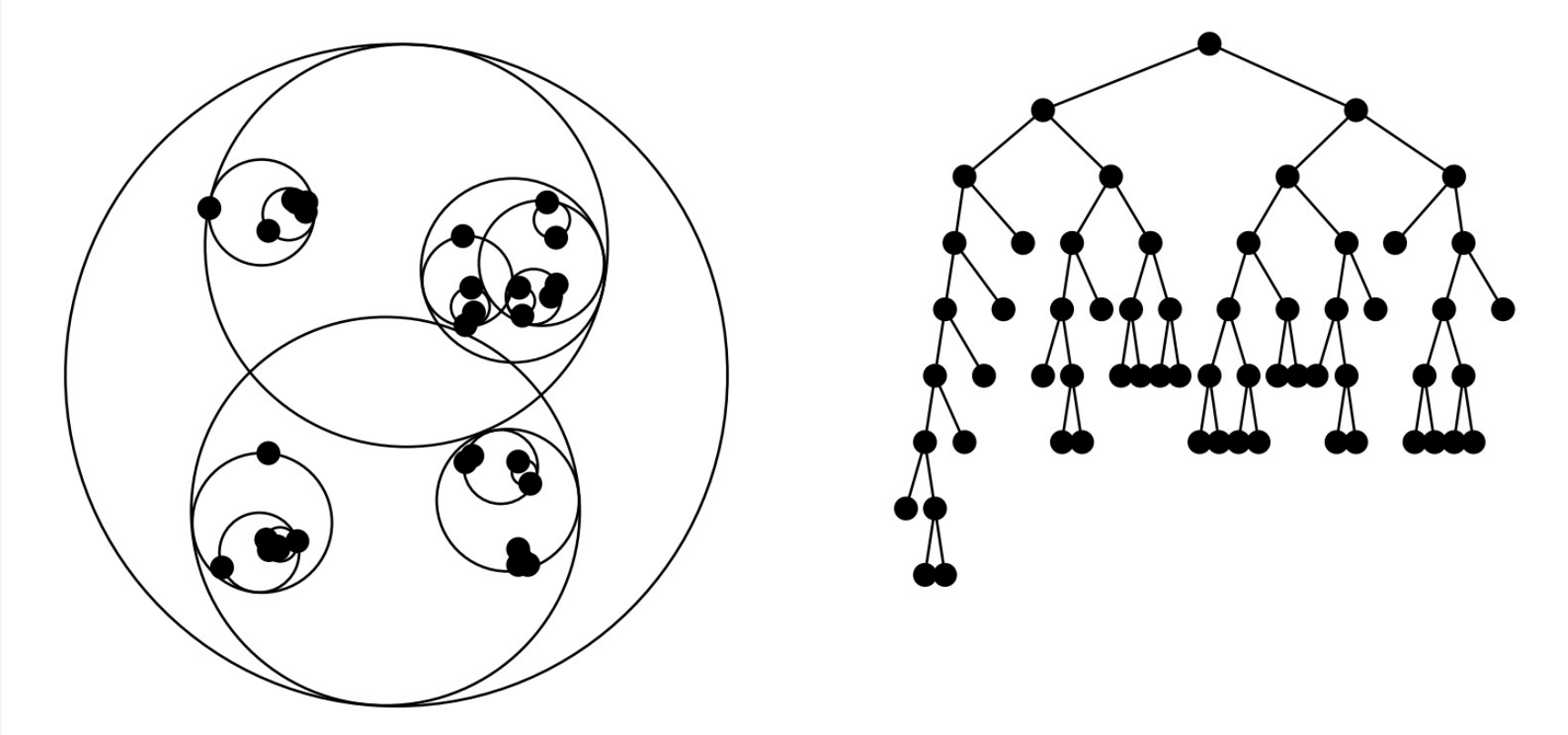
Now you classify a single example $\mathbf{x}$

Naïve Method: Loop over each training example, compute distances, cache min indices

- The naïve method is $O(pn)$ in time
- The naïve method is $O(pn)$ in space

Tree-Based Implementation

Build a KD- or Ball-tree on the training data



Tree-Based Implementation

Build a KD- or Ball-tree on the training data

Higher up-front cost in time and storage to build data structure

Up-front cost is $O(pn \log n)$ in time and $O(pn)$ in space

Now you classify a single example $\mathbf{x}$

Classification time for single query is $O(p \log n)$

Good strategy if lots of data and need to do lots of queries

Performance Metrics and Choosing K

Question: How do we evaluate our KNN classifier?

Answer: Perform prediction on labeled data, count how many we got wrong

$$\text{Error Rate} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

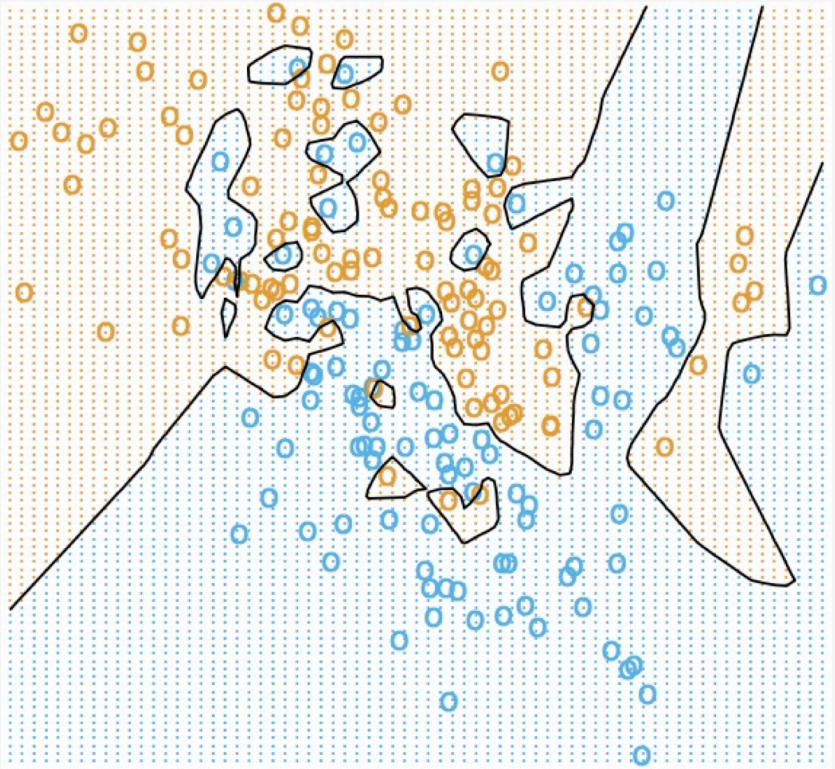
WRONG PRED / # PREDICTIONS

Question: How do we choose K?

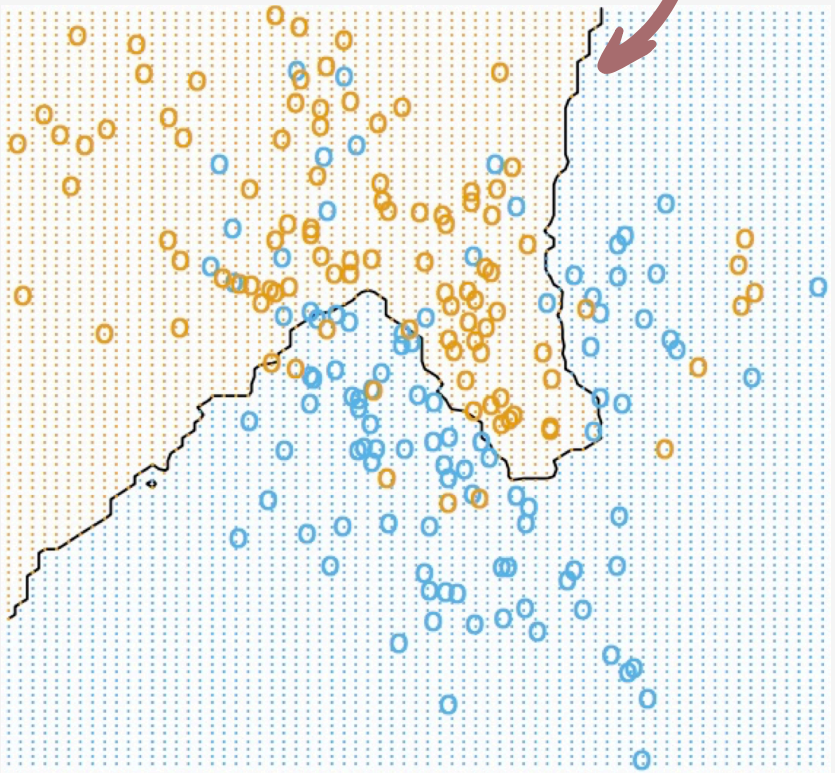
Performance Metrics and Choosing K

Like all models, KNN can overfit and underfit, depending on choice of K

K = 1



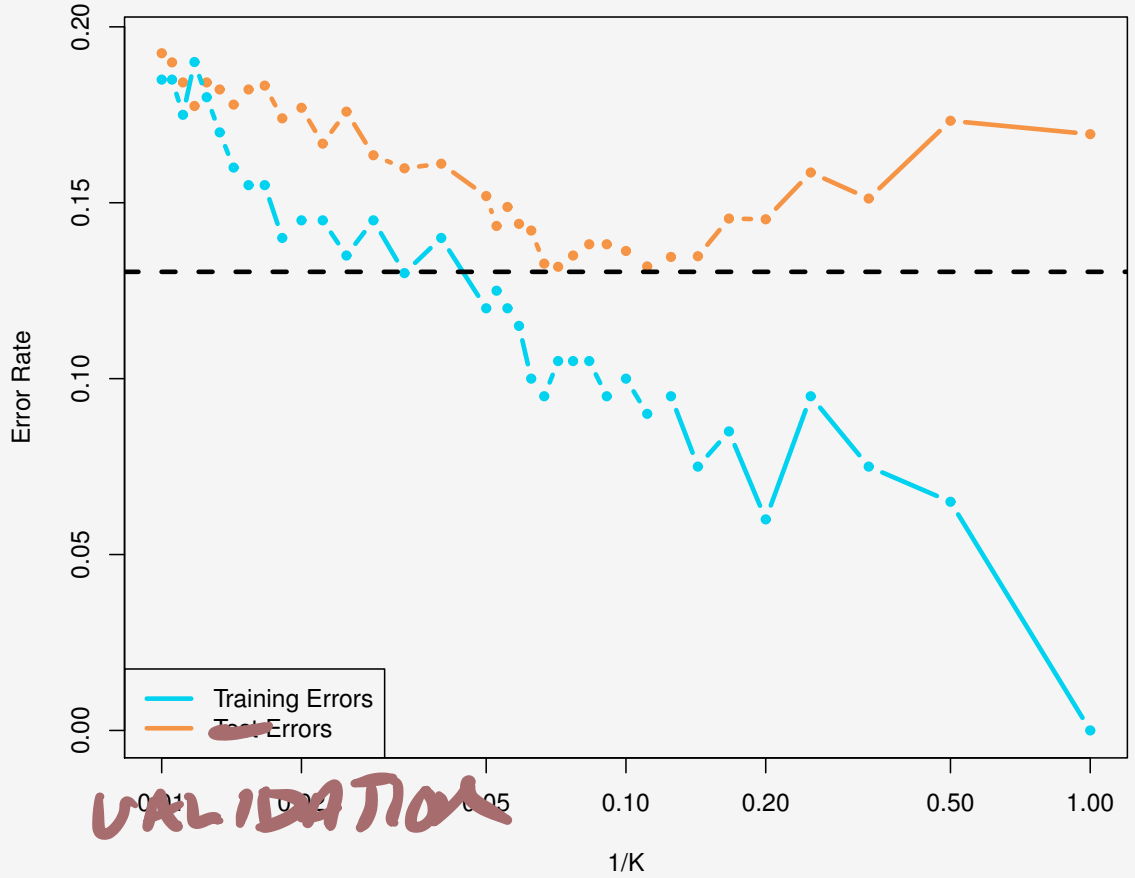
K = 15



Performance Metrics and Choosing K

Choose optimal K by varying K and computing error rate on train and validation set

- **Note**: Plot vs 1/K
- **Lower** K leads to **more** flexibility
- Choose K to minimize **validation error**

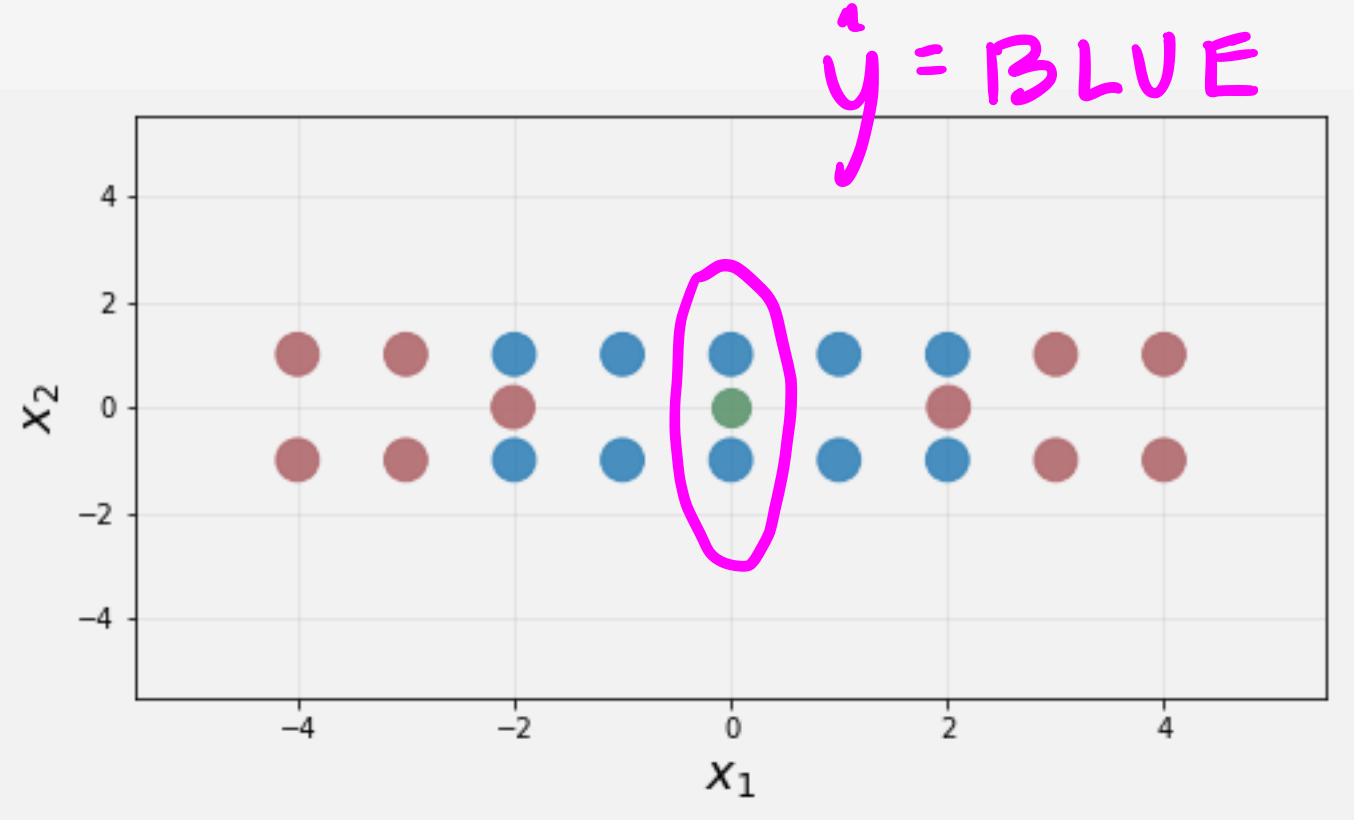


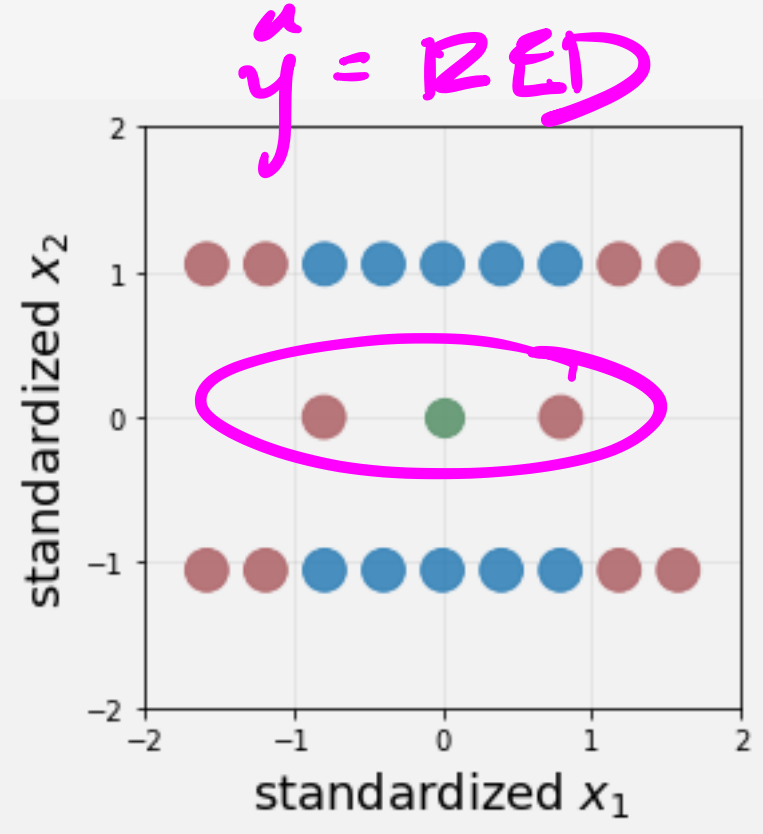
Feature Standardization

- Important to normalize/standardize features to similar sizes
- Consider doing 2-NN on unscaled and scaled data

Feature Standardization

- Important to normalize/standardize features to similar sizes
- Consider doing 2-NN on unscaled and scaled data





Generalization Error and Analysis

Bias-Variance Trade-Off:

- Pretty much analogous to our experience with polynomial regression
- As $1/K \to 1$, variance increases and bias decreases

Reducible and Irreducible Errors:

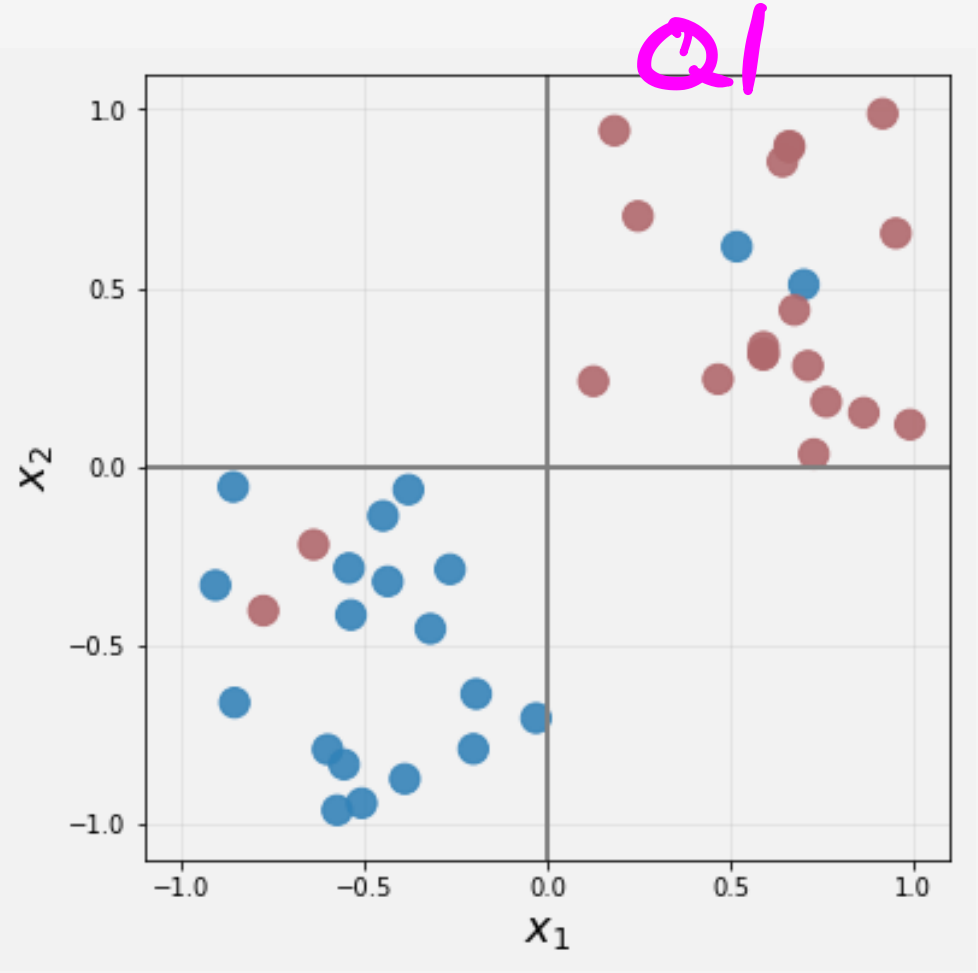
- This is more interesting, and more complicated
- What follows applies to ALL classification methods, not just KNN

The Optimal Bayes Classifier

Suppose we knew the true probabilistic distribution of our data: $p(Y = k \mid \mathbf{X})$

Example:

- Equal prob a 2D point is in Q1 or Q3
- If point in Q1, 0.9 probability it's red.
- If point in Q3, 0.9 probability it's blue.
- If classifier uses true model to predict...
- will be wrong 10% of time on average
- Bayes Error Rate = 0.1

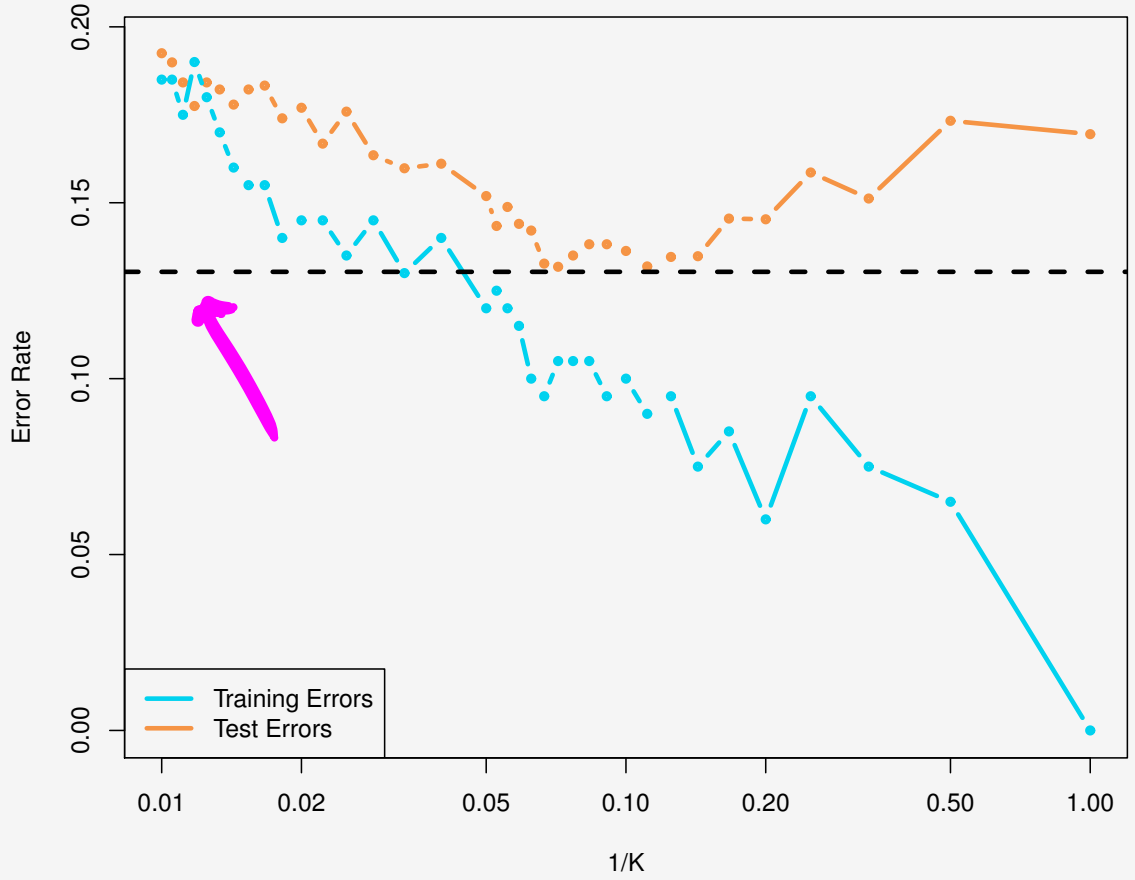


The Optimal Bayes Classifier

The **Bayes Error Rate** is the theoretical measure of **Irreducible Error**

- In long run, can never do better than the Bayes Error Rate on validation data

- Of course, just like $\mathrm{Var}(\epsilon)$ in the regression setting, we don't actually know what this error is. We just know it's there!



K-Nearest Neighbors Wrap-Up

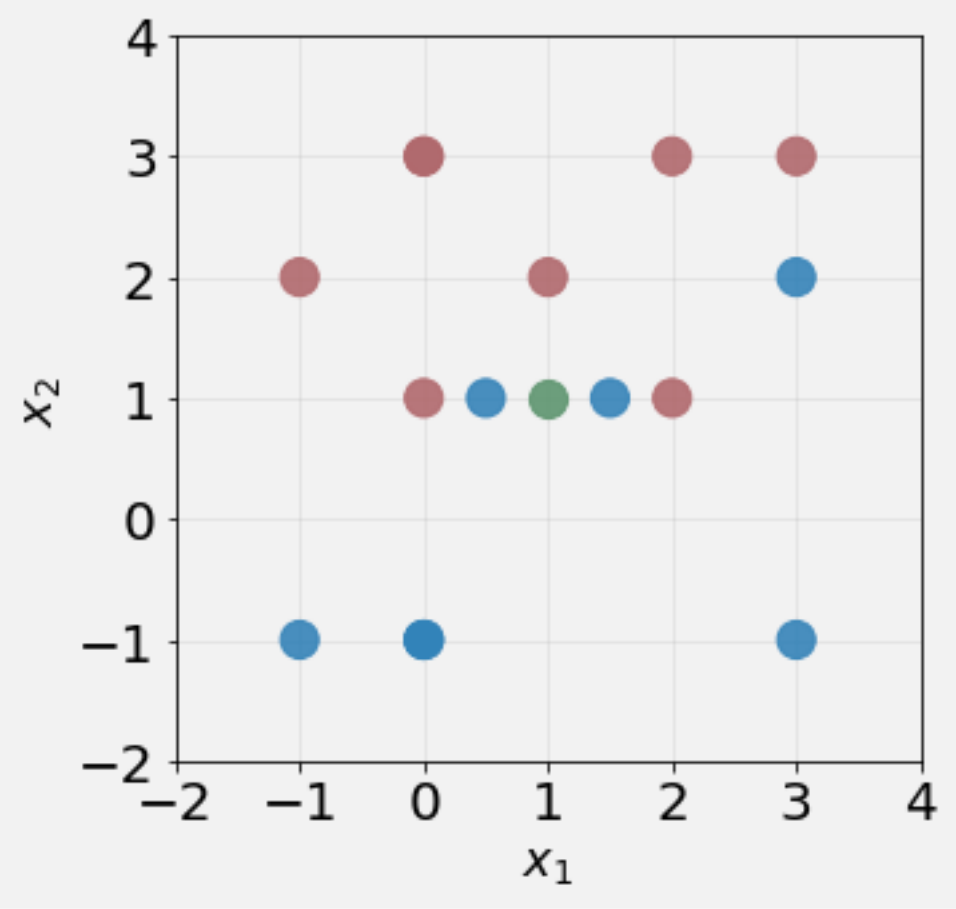
- Talked about the general classification setting
- Learned how to do simple KNN
- Looked at possible implementations and their complexity

Next Time:

- Look at our first parametric classifier: The Perceptron
- Kinda a weak model, but forms the basis for Neural Networks

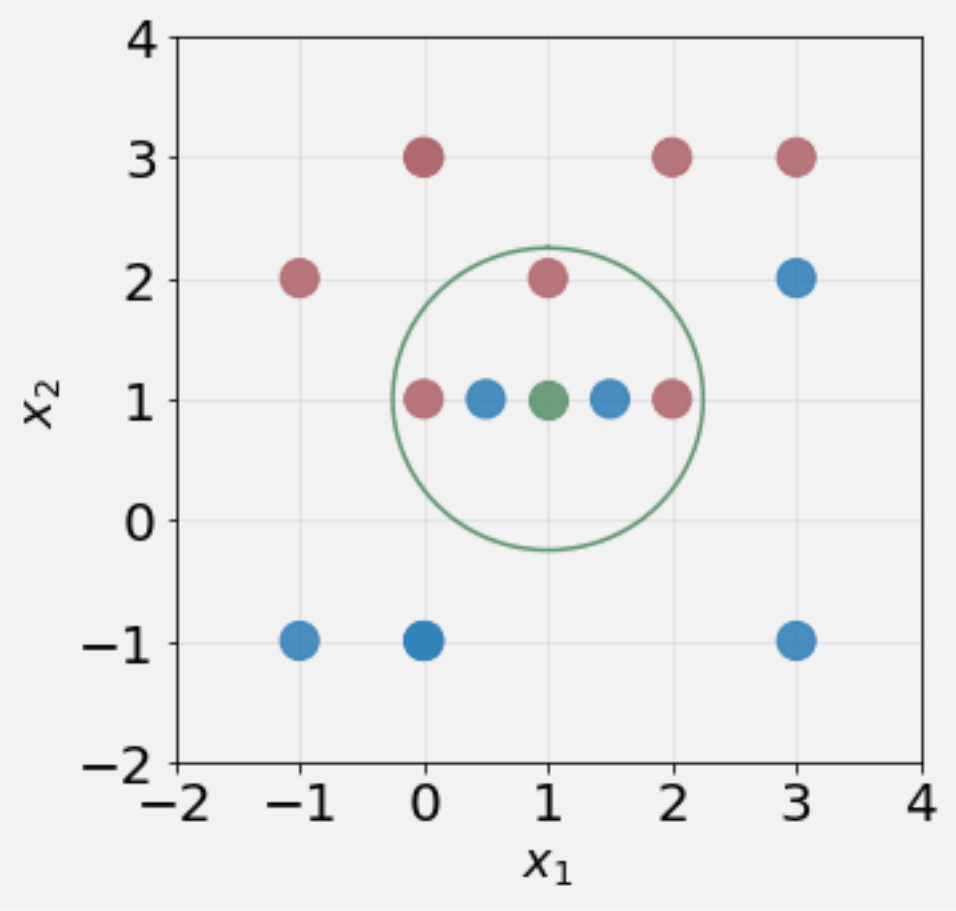
If-Time Bonus: Weighted KNN

Question: What should 5-NN predict in the following picture?



If-Time Bonus: Weighted KNN

Question: What should 5-NN predict in the following picture?



Standard KNN would predict red, but it seems like there are almost as many blues and they're closer.

If-Time Bonus: Weighted KNN

Weighted-KNN:

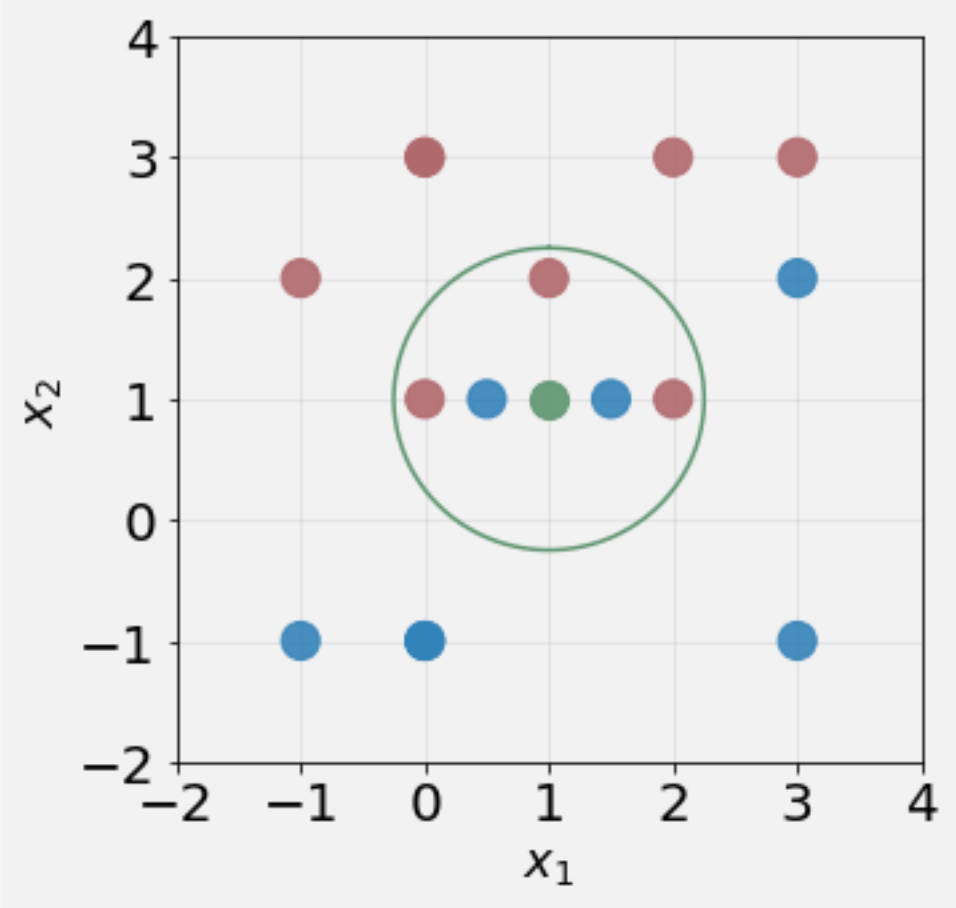
- Find $\mathcal{N}_K(\mathbf{x})$: the set of K training examples nearest to $\mathbf{x}$
- Predict $\hat{y}$ to be weighted-majority label in $\mathcal{N}_K(\mathbf{x})$, weighted by inverse-distance

Improvements over KNN:

- Gives more weight to examples that are very close to query point
- Less tie-breaking required

If-Time Bonus: Weighted KNN

Question: What should 5-NN predict in the following picture?



Red Distances:

1, 1, 1

Blue Distances:

0.5, 0.5

Red Weighted-Majority Vote:

1/1 + 1/1 + 1/1 = 3

Blue Weighted-Majority Vote:

1/0.5 + 1/0.5 = 2 + 2 = 4

Prediction: 4 > 3 -> predict Blue