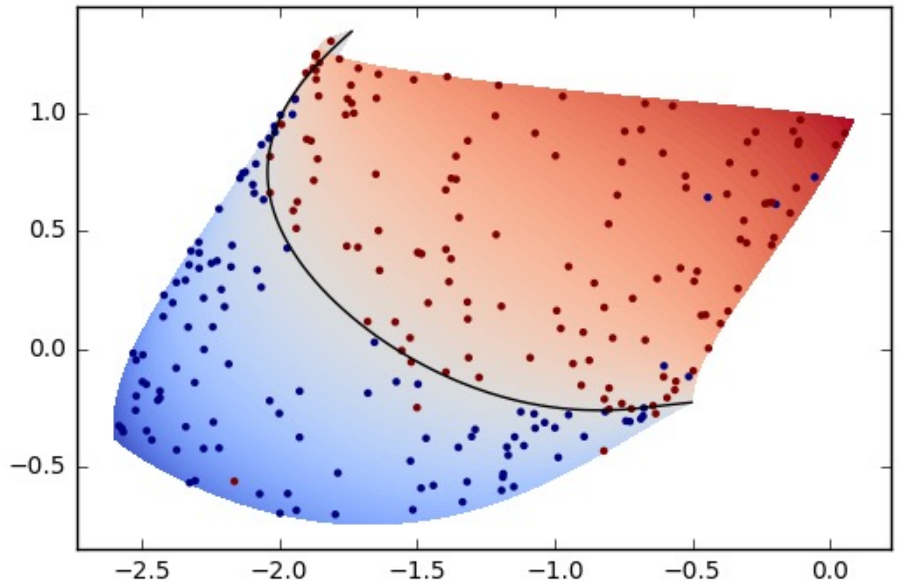
Lecture 7: Nearest Centroids (Part 2)



Gavin Kerrigan

Spring 2023

Some materials courtesy Padhraic Smyth, Alex Ihler.

Announcements

- HW1: grades & solutions released by Friday

- HW2 posted soon (by tomorrow)
  - Announcement via Ed
  - Due in 2 weeks (Friday 4/28)
  - Problem 1: nearest centroids on MNIST (today's lecture)
  - Problem 2: kNN (today / tomorrow's lecture)

- Keep an eye out for a survey
  - Announcement via Ed

Today's Lecture

Nearest Centroids

Classifier Evaluation

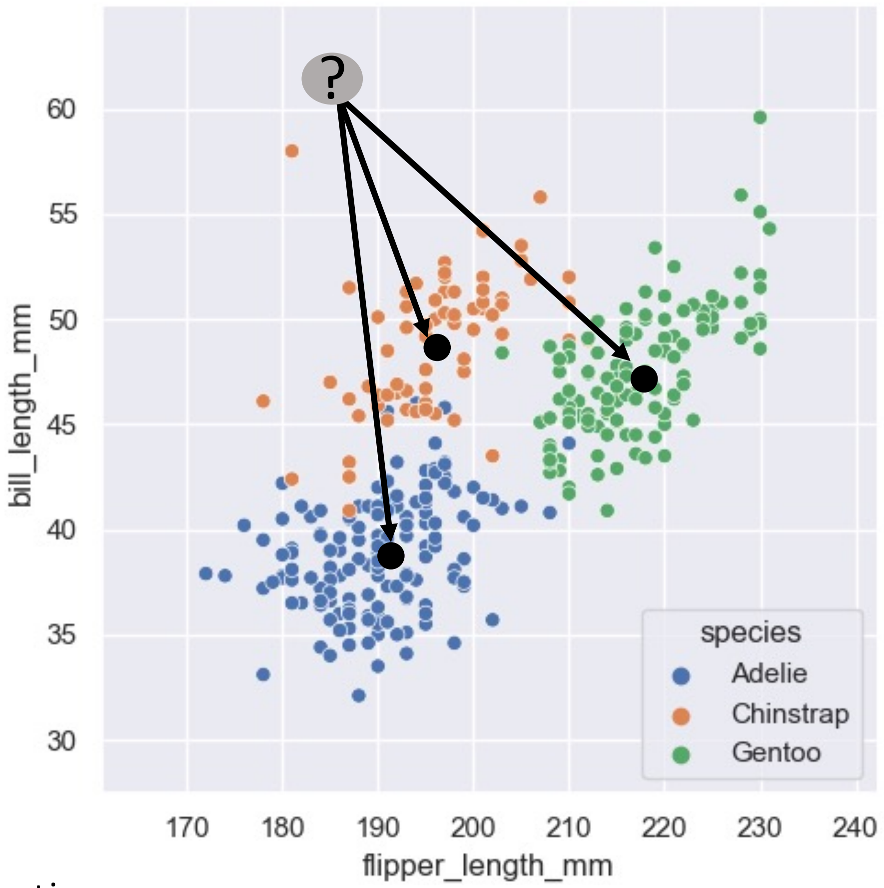
k Nearest Neighbors

The Nearest-Centroid Classifier

This is a very simple idea

Compute the centroid for each class
(i.e. centroid of datapoints in that class)

For a new unlabeled datapoint, assign
it to the class it is closest to
(in terms of Euclidean distance)



Even though it is very simple, it is plausible
as a simple cognitive model of human classification,
i.e., represent each class by a single “template”

Illustration of a Nearest-Centroid Classifier

- We will use the Penguin dataset for illustration

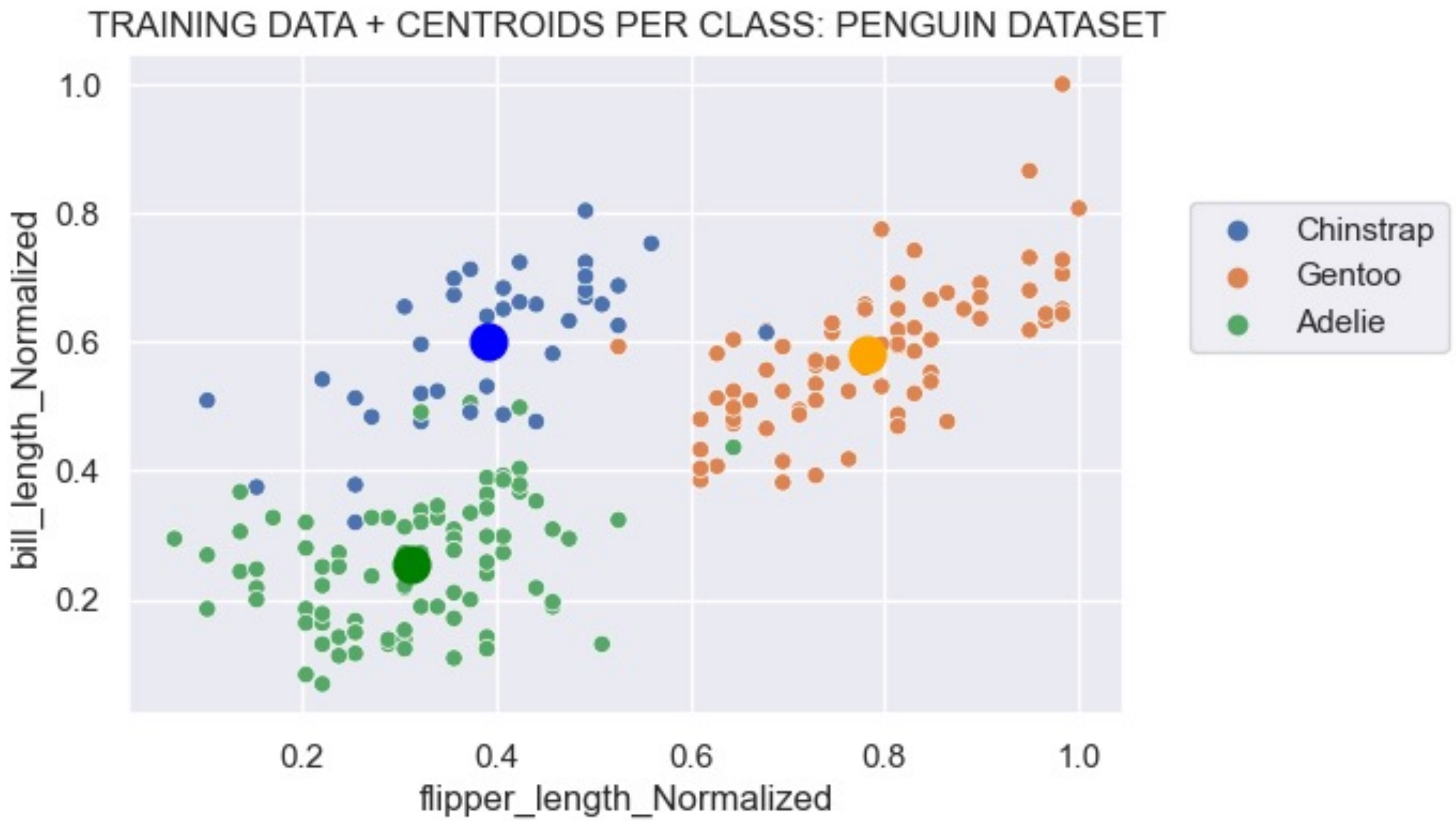
- Split the data (random selection of rows) into
  - 200 for training (where “training” = define the class centroids)
  - 133 for test (which we can use to evaluate accuracy)

- Use just 2 features in the classifier so that we can visualize the results

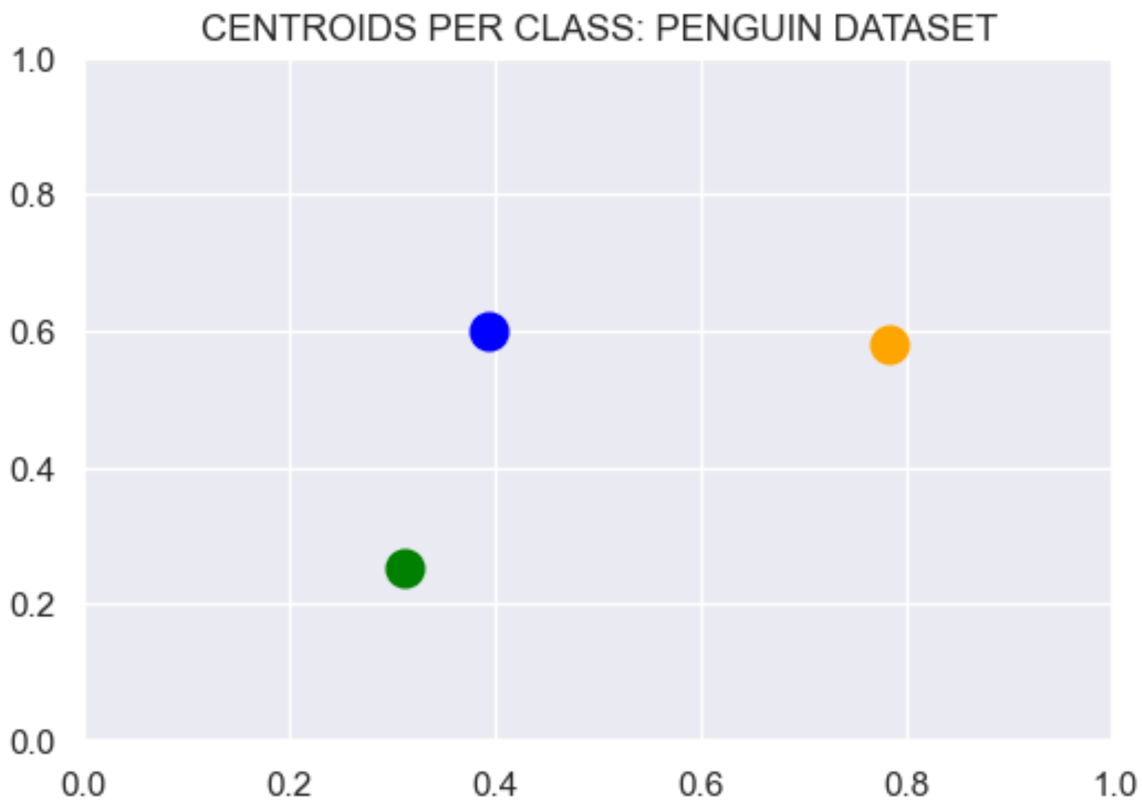
- Scale the features in each dimension so that they range from 0 to 1

The 200 Training Data Points

Training Data + Class Centroids

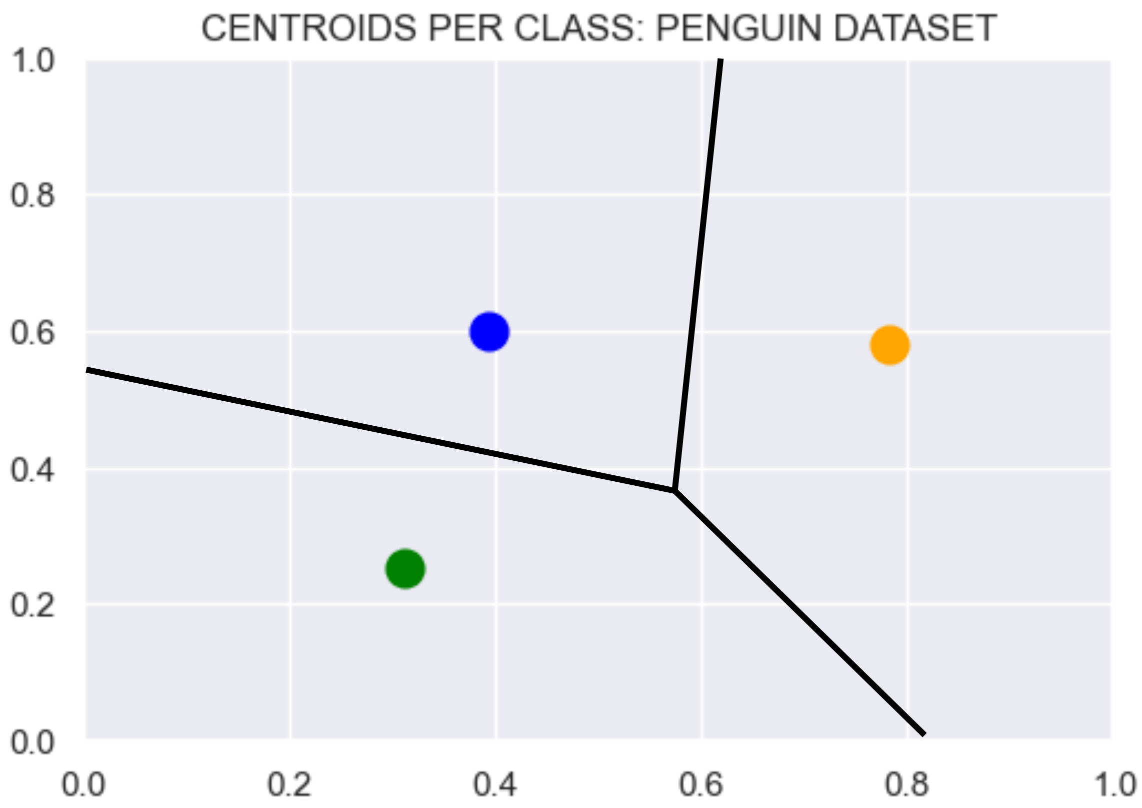


The Classifier: 3 Centroids



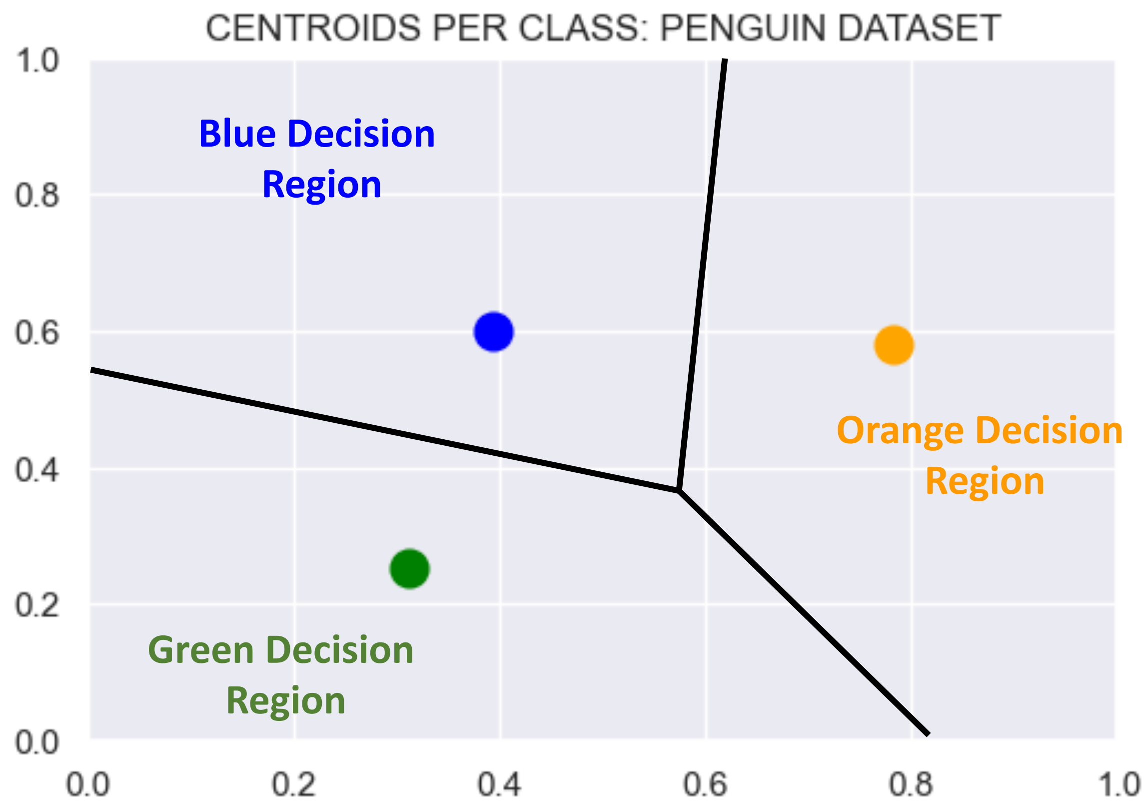
The centroids define (i.e., are) the Classifier

The Classifier: Decision Boundaries

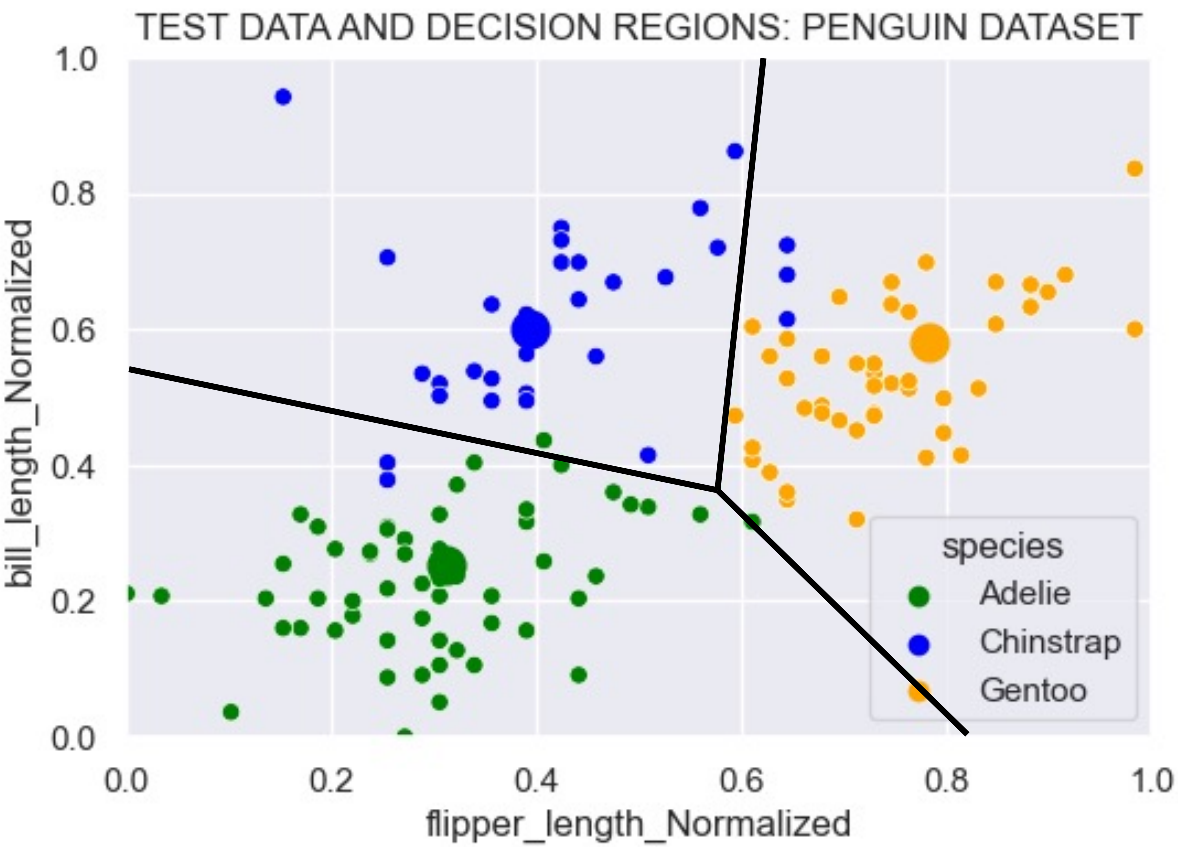


The three centroids (the “classifier”) implicitly determine the location of decision boundaries

The Classifier: Decision Regions

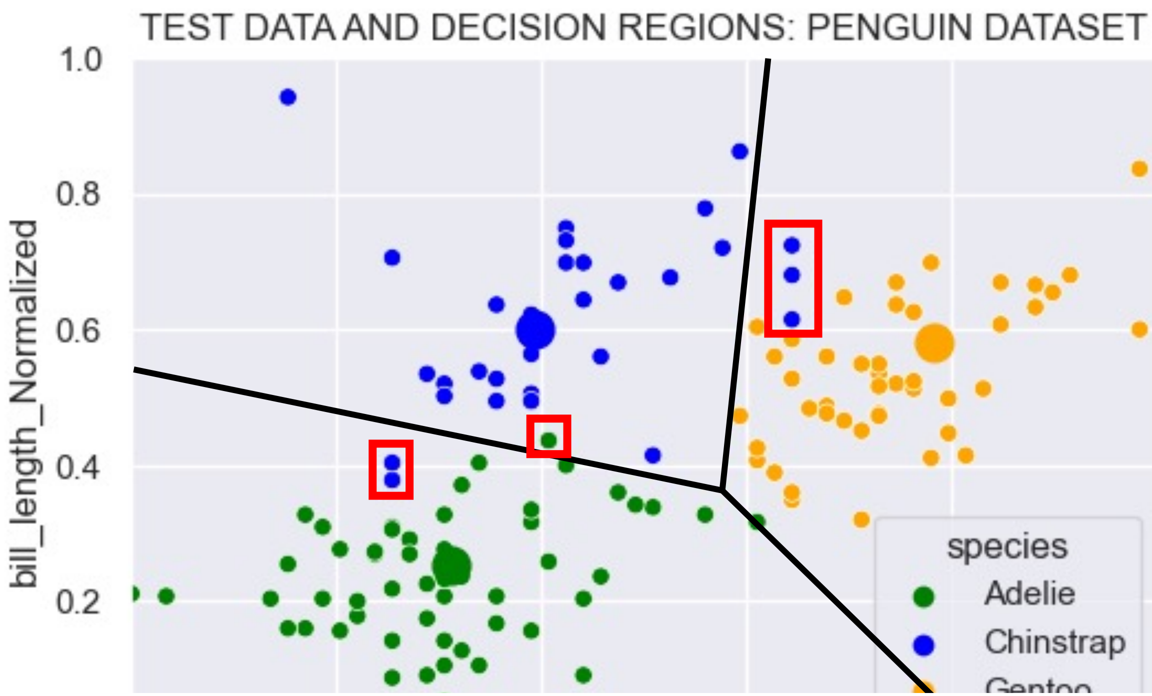


Test Data



The color for each datapoint is its true label

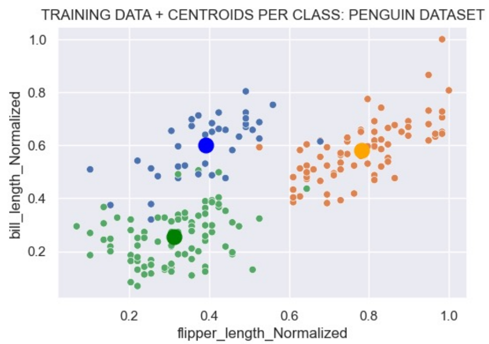
The Classifier’s Errors



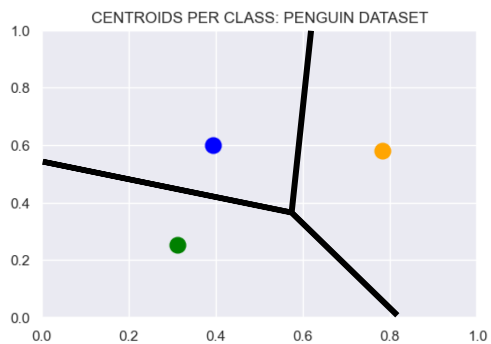
6 errors out of 133 examples = 6/133 = 4.5% error

“Error” means its in the incorrect decision region (incorrect prediction by the classifier)

Labeled Data



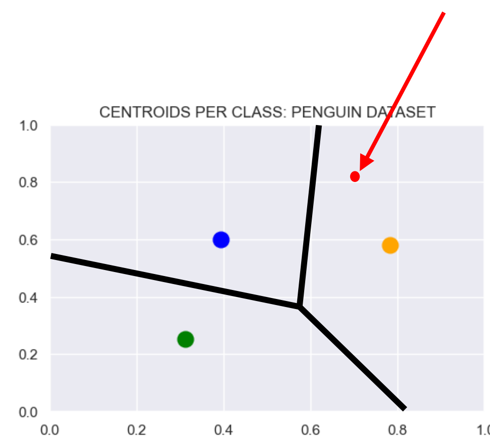
Compute Centroids



This is the Classifier

1. Building a Classifier

At prediction time:
Label is predicted for a new unlabeled datapoint



2. Using a Classifier for Prediction

MNIST Data

Classic dataset in ML

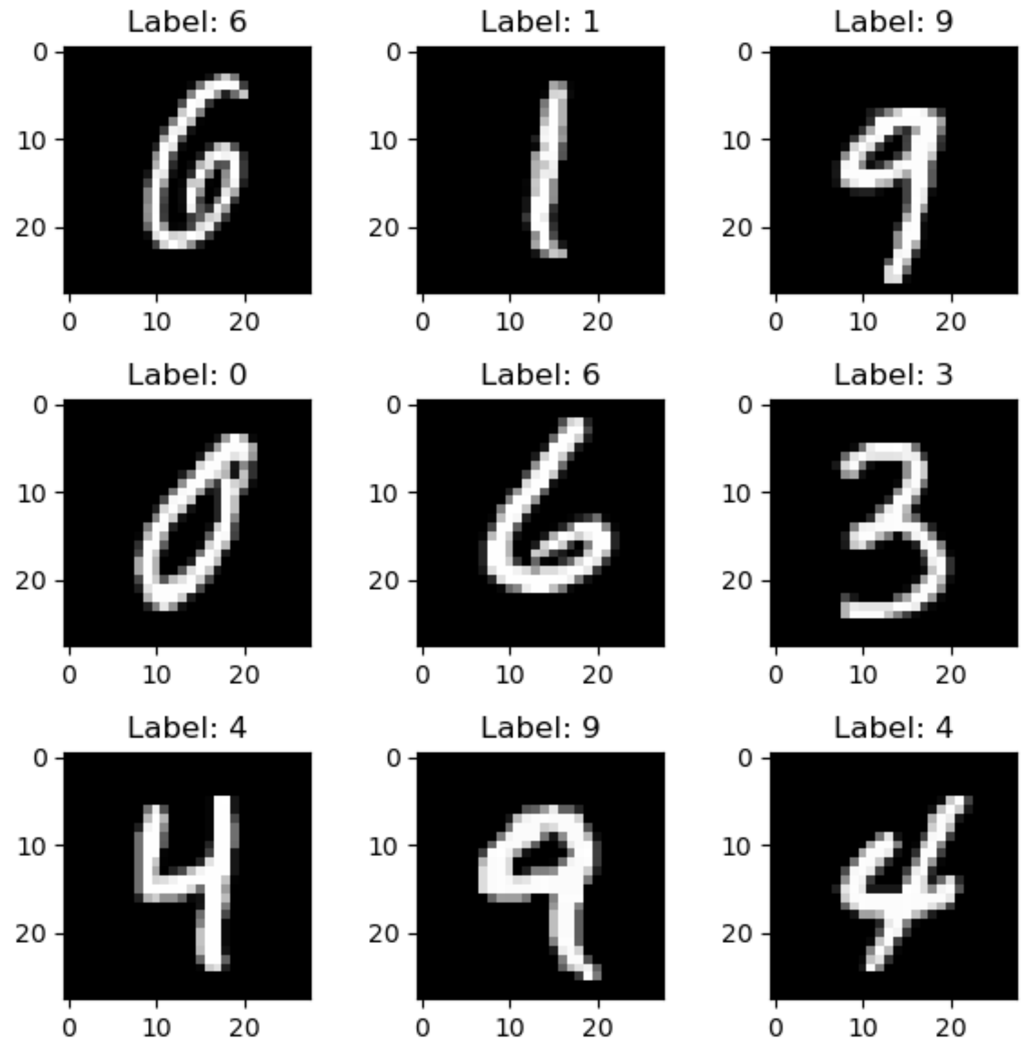
70,000 handwritten digits,
28 x 28 gray-scale pixels
Feature vector dim d = 28 x 28 = 784

10 class labels $y \in \{0, 1, 2, \ldots 9\}$

In these slides we shuffle data, and
pick 20% for train, 80% for test

Used in Homework 2 (with 75% for train)

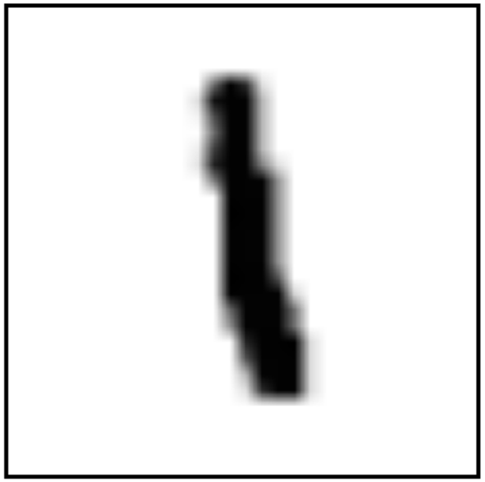
Data originally created by
Yann LeCun (Turing Award winner)
in 1998 for ML research.
See: Le Cun, Bottou, Bengio, Haffner,
Proceedings of IEEE, 1998



Variation in Examples per Class

From: https://en.wikipedia.org/wiki/MNIST_database

Representing Images as Feature Vectors



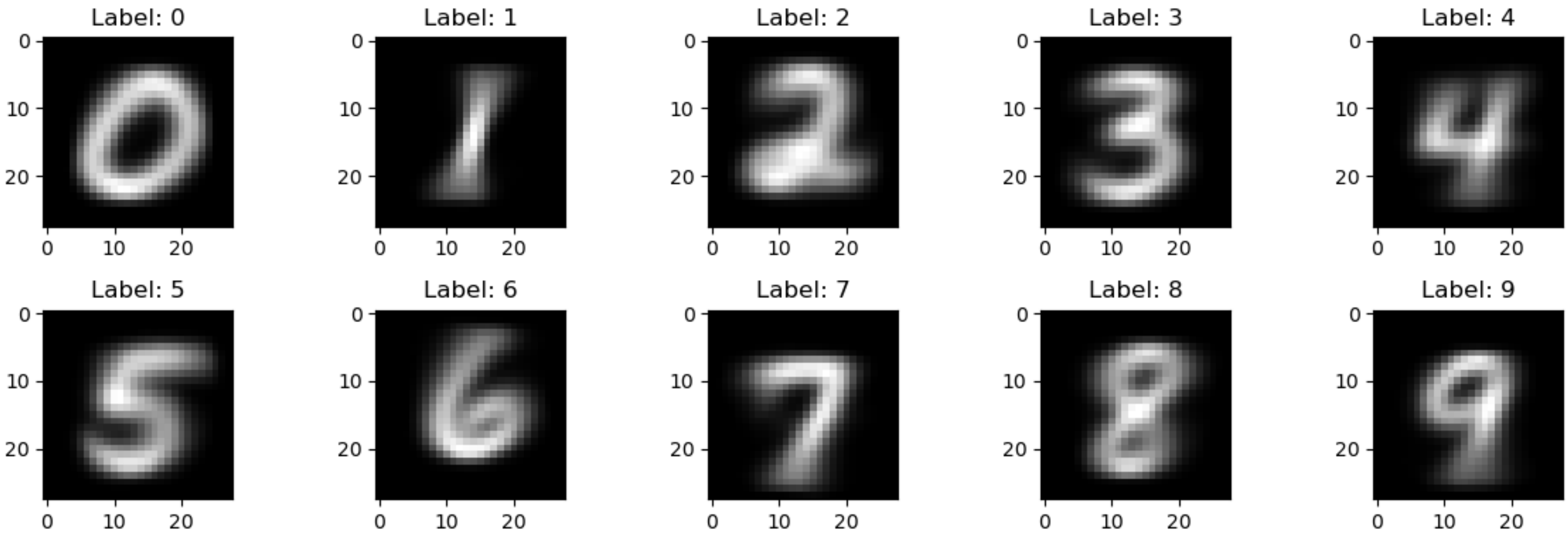
$$\cong \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\ 0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

This m x m array of numbers is then written out as a feature vector of length d x 1, where $d = m^2$

For example, for MNIST (homework 2):

- images are 28 x 28 (i.e., m=28)
- the resulting feature vector (per image) has $d = 28^2 = 784$

Fitted Centroids per Class

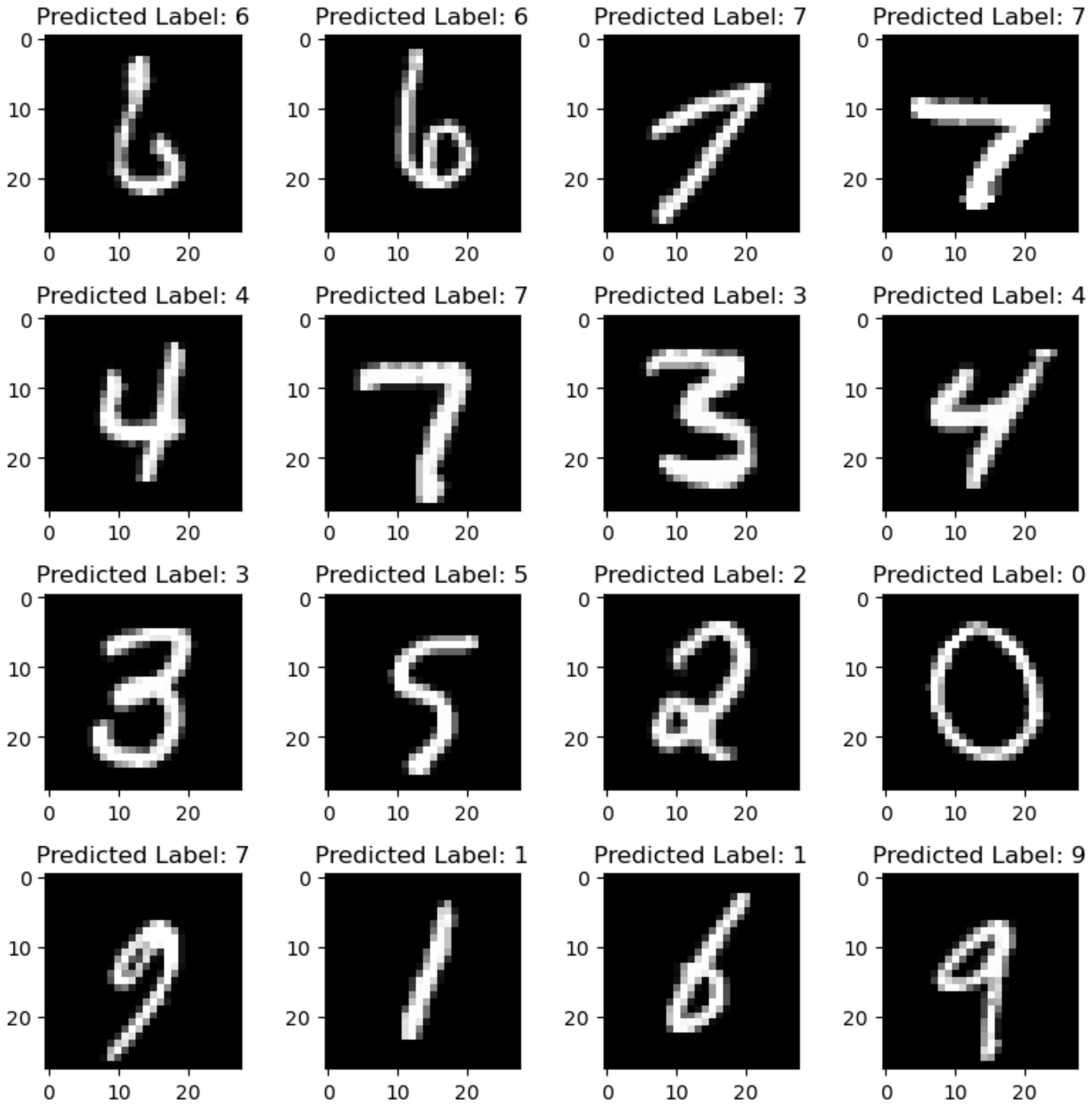


Note that the centroids are ”blurry”: why?

Predictions on Test Examples

(from NC model fit to 20% of data)

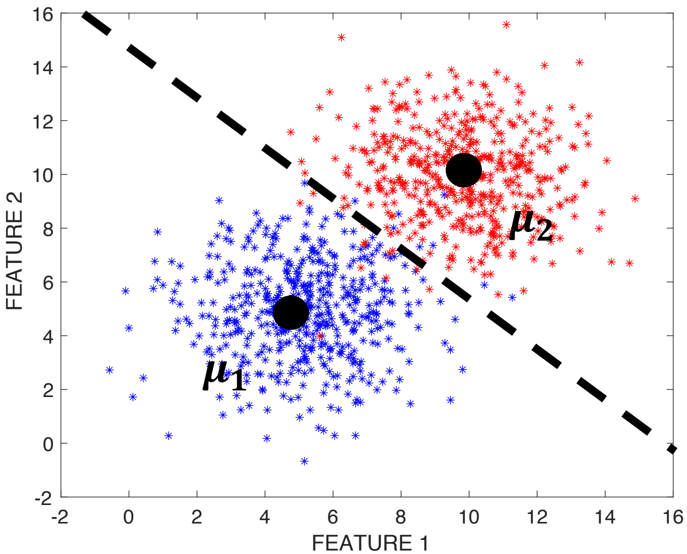
Accuracy:
About 80% on both training and test data



Decision Boundaries for Nearest Centroids

The nearest centroids classifier has *piecewise linear* decision boundaries

Here's a proof with two classes (C=2)
and two features (d = 2)



Two centroids:

$$\boldsymbol{\mu}_1 = (\mu_{11}, \mu_{12}) \qquad \boldsymbol{\mu}_2 = (\mu_{21}, \mu_{22})$$

A point $\boldsymbol{x} = (x_1, x_2)$ is on the decision boundary if and only if it has equal distance to both centroids:

$$d_E(\mathbf{x}, \boldsymbol{\mu}_1) = d_E(\mathbf{x}, \boldsymbol{\mu}_2)$$

Decision Boundaries for Nearest Centroids

A point $\boldsymbol{x} = (x_1, x_2)$ is on the decision boundary if and only if it has equal distance to both centroids:

$$d_E(\mathbf{x}, \boldsymbol{\mu}_1) = d_E(\mathbf{x}, \boldsymbol{\mu}_2)$$

1.) Plug in to the formula for Euclidean distance:

$$\sqrt{(x_1 - \mu_{11})^2 + (x_2 - \mu_{21})^2} = \sqrt{(x_1 - \mu_{21})^2 + (x_2 - \mu_{22})^2}$$

2.) Square both sides and expand everything out:

$$\cancel{x_1^2} - 2x_1\mu_{11} + \mu_{11}^2 + \cancel{x_2^2} - 2x_2\mu_{12} + \mu_{12}^2 = \cancel{x_1^2} - 2x_1\mu_{21} + \mu_{21}^2 + \cancel{x_2^2} - 2x_2\mu_{22} + \mu_{22}^2$$

3.) Move all x_2 terms to the left; all x_1 terms to the right:

$$x_2(2\mu_{22} - 2\mu_{12}) = x_1(2\mu_{11} - 2\mu_{21}) + \mu_{22}^2 + \mu_{21}^2 - \mu_{11}^2 - \mu_{12}^2$$

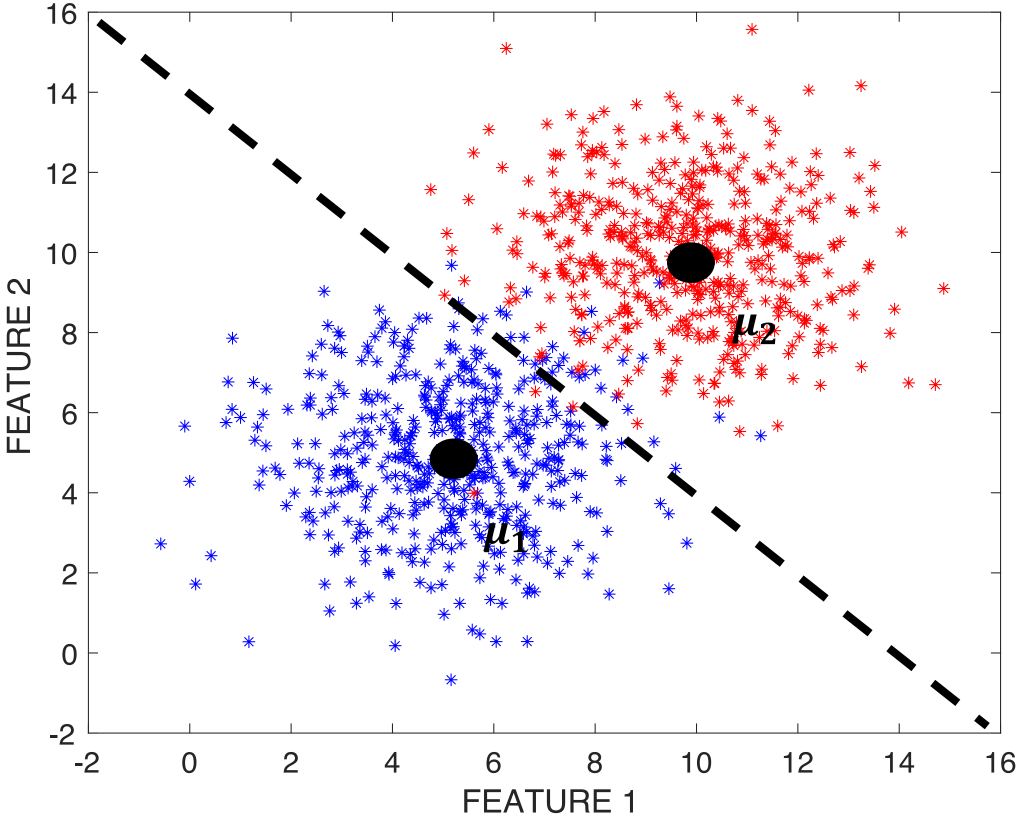
4.) Solve for x_2:

$$x_2 = \left(\frac{\mu_{11} - \mu_{21}}{\mu_{22} - \mu_{12}}\right) x_1 + \frac{\mu_{22}^2 + \mu_{21}^2 - \mu_{11}^2 - \mu_{12}^2}{2(\mu_{22} - \mu_{12})}$$

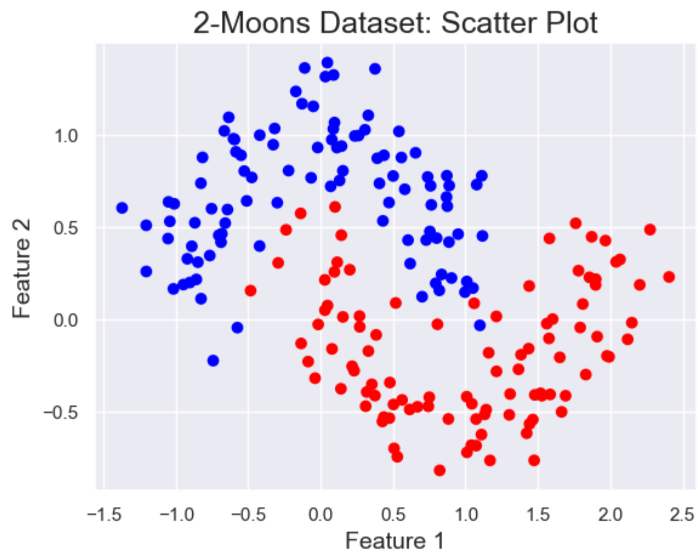
Decision Boundaries for Nearest Centroids

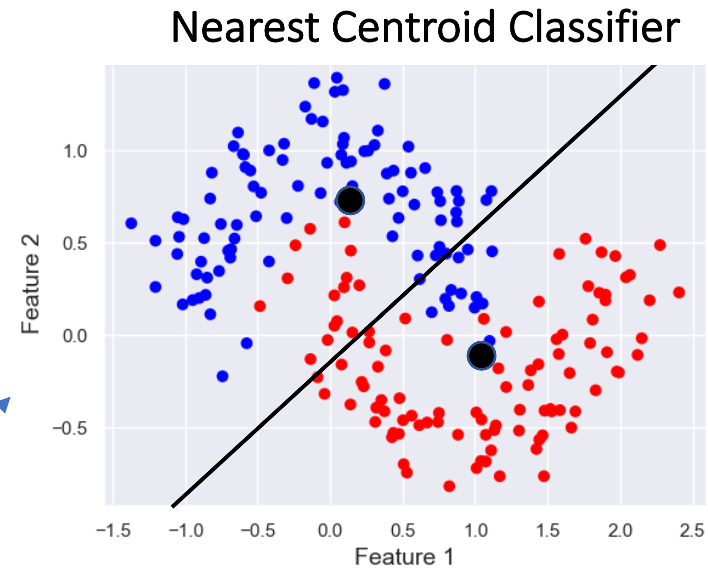
$$d_E(\mathbf{x}, \boldsymbol{\mu}_1) = d_E(\mathbf{x}, \boldsymbol{\mu}_2) \quad \text{if and only if} \quad \underbrace{x_2 = \left(\frac{\mu_{11} - \mu_{21}}{\mu_{22} - \mu_{12}}\right) x_1 + \frac{\mu_{22}^2 + \mu_{21}^2 - \mu_{11}^2 - \mu_{12}^2}{2(\mu_{22} - \mu_{12})}}$$

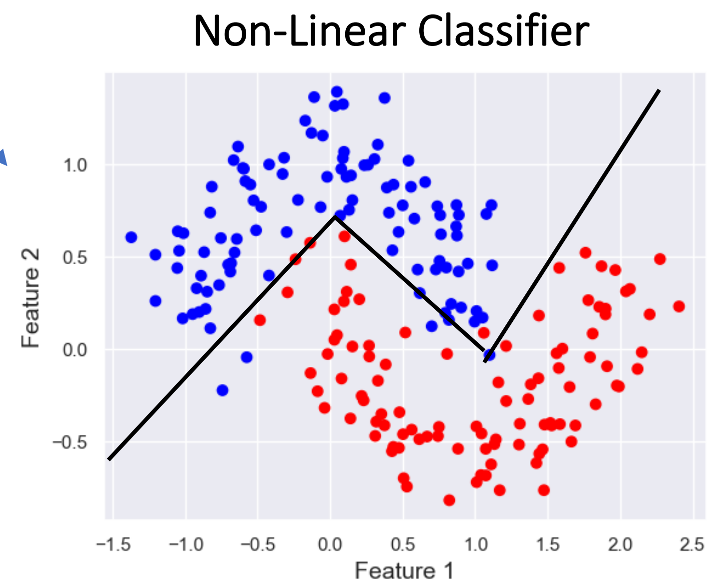
Linear equation representing the decision boundary



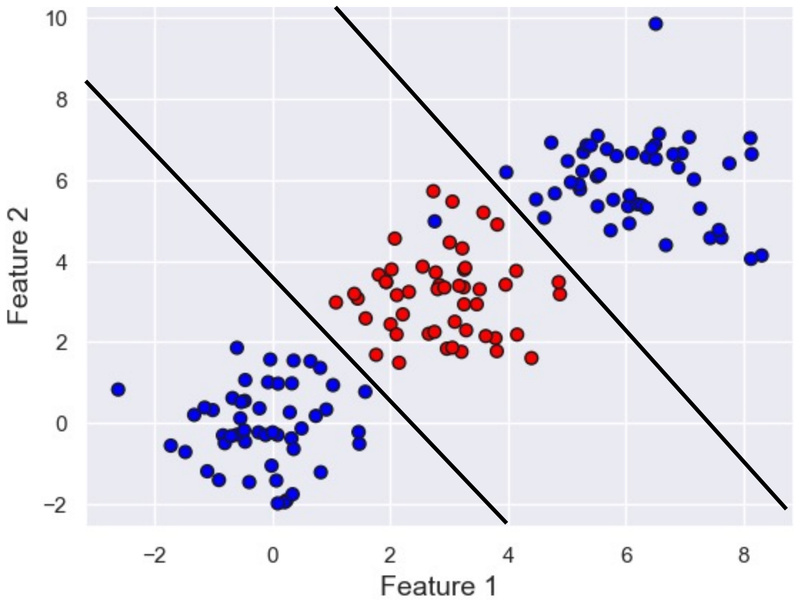
Limitations of Nearest-Centroid







Limitations of Nearest-Centroid



Here the ideal classifier has 2 separate decision regions for the blue class

The nearest-centroid classifier cannot model multiple decision regions per class

Strengths and Weaknesses of Nearest-Centroid

- Strengths
  - Simple to implement in code
  - Easy to explain
  - Does not need a lot of training data

- Main Weakness
  - Too simple to model complex classification boundaries

Questions?

Today's Lecture

Nearest Centroids

Classifier Evaluation

k Nearest Neighbors

Classifier Accuracy

Input = feature vector

Output = predicted label

$$\mathbf{x} \longrightarrow \boxed{\text{Classifier}} \longrightarrow \hat{y}$$

Classifier is a mapping from features to predicted labels

It takes an input x (a feature vector) and produces a prediction of a class label

Classifier Accuracy

Input = feature vector

$\mathbf{x}$

Classifier

Output = predicted label

$\hat{y}$

True label

y

Compare

If $\hat{y} = y$, then Acc = 1

If $\hat{y} \neq y$, then Acc = 0

In math notation:

Acc = $I(\hat{y}, y)$

Indicator function

Evaluates to 1 if arguments are equal

Evaluates to 0 otherwise

Classifier Accuracy

In general, we want to know classifier accuracy on average, over many **x** examples

We can estimate this on a **test data set** to get test accuracy
Say the test dataset has N_{test} labeled examples, i.e., feature vectors with labels

Accuracy on the ith datapoint from the test set is $Acc_i = I(\hat{y}_i, y_i)$

Average accuracy is defined as:

$$\begin{aligned} \text{Acc} &= \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \text{Acc}_i \\ &= \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} I(\hat{y}_i, y_i) \end{aligned}$$

Sum over all test examples

Compute Accuracy for each example

Simple Example of Accuracy Calculation

Test Data Features				Predicted Labels from Classifier	True Labels
21.4	6.1	200	→	0	0
28.1	5.5	145	→	0	1
24.7	2.2	94	→	1	1
32.0	4.2	155	→	1	1

Acc = ¼(1 + 0 + 1 + 1) = 3/4 = 0.75

Note:

1. We also often look at error rate = Err = 1 – Acc
2. We could express Accuracy as a fraction (0 to 1) or as a percentage

Computing a Confusion Matrix

- Say we make predictions on all n examples in a test set

- Confusion Matrix
  - A matrix with C rows and C columns
  - Rows correspond to the true class
  - Columns correspond to the predicted class
  - Entry (r, c) in the matrix, corresponding to (row r, column c), means
    - The true class is r, and the classifier predicted class c

  - To compute a Confusion Matrix:
    - Given a list of pairs (true_class, predicted_class)
    - Initialize the confusion matrix entries to be all zeros
    - Go through the list, and for each pair add 1 to the corresponding row/column cell in the matrix

Simple Example of a Confusion Matrix

C = 3 classes

List of (true, predicted) class labels:

(1,1)
(2,1)
(2,2)
(3,3)
(3,3)
(3,1)
(1,1)
(2,2)
(1,2)
(2,2)
(2,2)

	1	2	3
1	2	1	0
2	1	4	0
3	1	0	2

Predicted Label

True Label

Accuracy and the Confusion Matrix

n = sum of all counts in the confusion matrix

Accuracy = (sum of diagonal counts)/n

This makes sense: diagonal entries are predictions the classifier was correct on

Error = (sum of off-diagonal counts)/n = 1 - Accuracy

Off-diagonal entries are the errors, predictions the classifier was "confused" on

From previous slide: n = 11, sum of diagonals = 8

$\Rightarrow$ Accuracy = 8/11 = 0.73

$\Rightarrow$ Error = 3/11 = 0.27

2	1	0
1	4	0
1	0	2

Confusion Matrix

for Nearest-Centroid Classifier
with test data = 10% of data

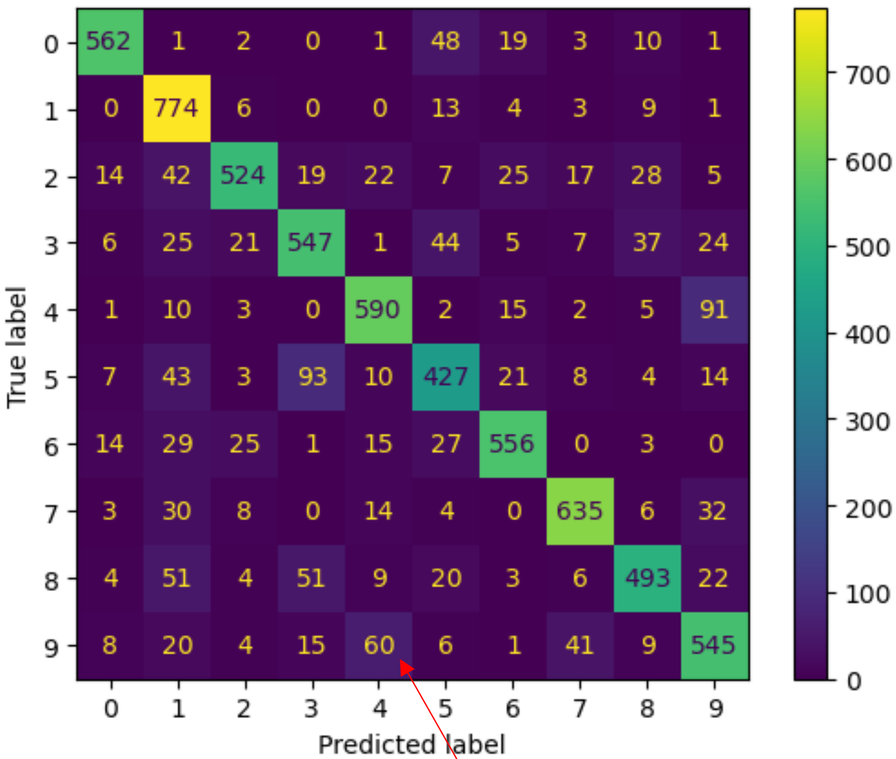
Entry for row i, column j, is the number of test examples that have true label i but were predicted as j

On-diagonal cells are correct predictions

Off-diagonal cells are errors (“confusions”)

Some pairs of labels are rarely confused, e.g., “1” and “0”, or “4” and “3”

Others are much more frequently confused, e.g., “5” and “3”, or “4” and “9”



Indicates that 60 examples in the test data that had true class label “9” were predicted by the NC classifier as “4”’s

Confusion Matrix for Binary Classification

	Predicted 0	Predicted 1	
True Label 0	True Negative (TN)	False Positive (FP)	0
True Label 1	False Negative (FN)	True Positive (TP)	1
	0	1	

Predicted Label

y=0 often the “negative” class
y=1 often called the “positive” class

e.g. y=0 indicates no disease, y=1 indicates disease

Recall = TP/(TP + FN)

- Of the datapoints with <u>true label</u> y=1 (i.e. TP+FN) how many do we correctly classify?
- If a patient has a disease, how good are we at detecting it?

Confusion Matrix for Binary Classification

	Predicted Label 0	Predicted Label 1
True Label 0	True Negative (TN)	False Positive (FP)
True Label 1	False Negative (FN)	True Positive (TP)

y=0 often the “negative” class
y=1 often called the “positive” class

e.g. y=0 indicates no disease, y=1 indicates disease

Precision: TP/(TP + FP)

- Of the datapoints with <u>predicted</u> label $\hat{y}$=1 (i.e. TP+FP) how many do we correctly classify?
- If we predict that a patient has a disease, how likely are we to be correct?

Confusion Matrix for Binary Classification

12	2	0
6	18	1
0	1	

True Label

Predicted Label

y=0 often the "negative" class
y=1 often called the "positive" class

e.g. y=0 indicates no disease, y=1 indicates disease

Accuracy = (12 + 18) / (12 + 18 + 6 + 2) = 0.789

Recall = 18/(18 + 6) = 0.75

Precision = 18/(18 + 2) = 0.9

Questions?

Today's Lecture

Nearest Centroids

Classifier Evaluation

k Nearest Neighbors

Could we extend the Nearest-Centroid Classifier?

- One possible idea would be to allow each class to be represented by multiple points (rather than just one)
- A new point could be classified as having the class of the point it is closest to in the training data
  - But how many points per class? And how would they be chosen?
- The Nearest Neighbor Classifier takes this idea to an extreme

The Single Nearest Neighbor Classifier

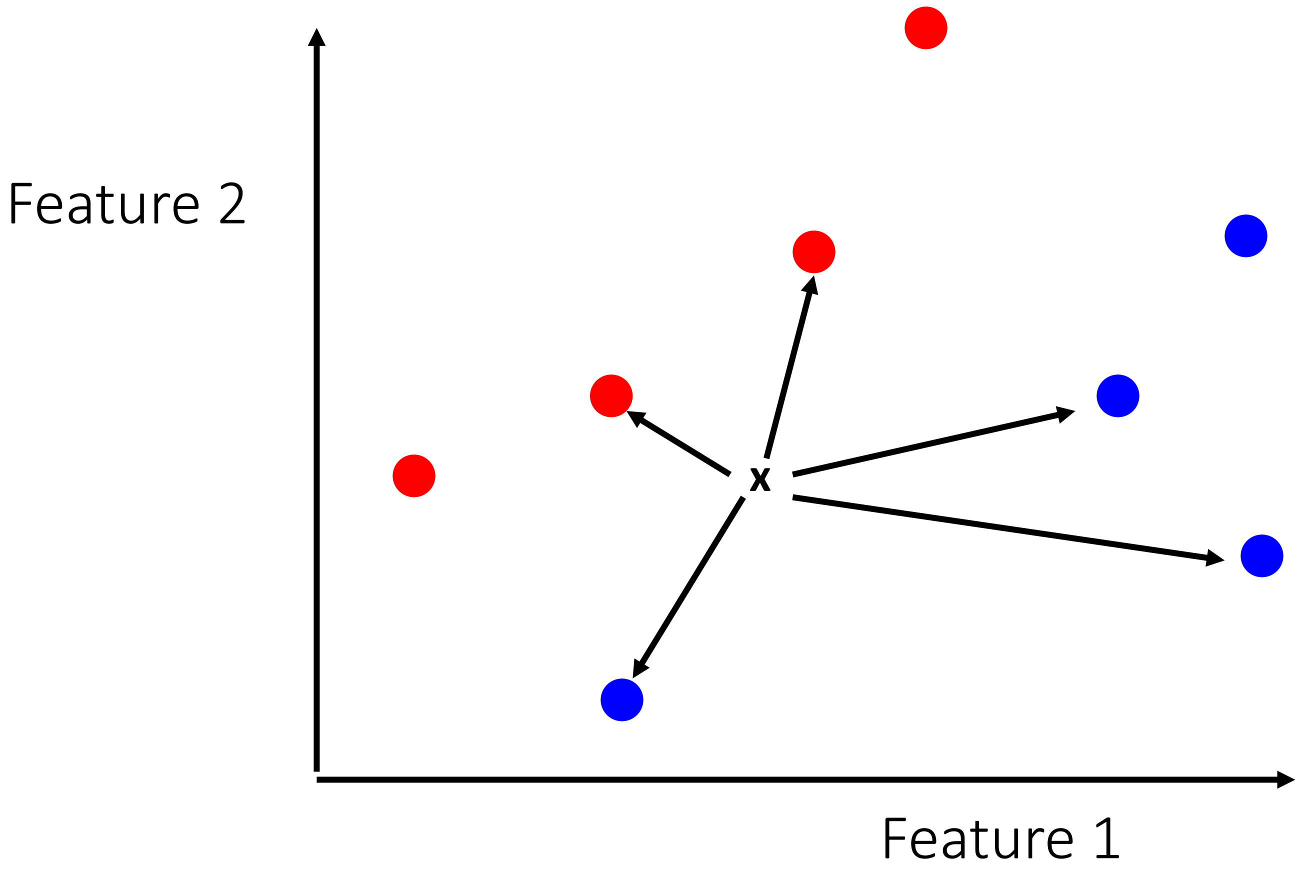
- Given:
  - Training data = **X** (feature vectors), **y** (class labels)

- Prediction Method using Single Nearest Neighbor
  - given an unlabeled feature vector **x**
  - Find the feature vector $\mathbf{x_i}$, in **X** that is closest to x
  - Let the class label y_i be the predicted label for x

- Very simple idea
  - Find the “nearest neighbor”
  - Assign the class label of this neighbor to **x**

- An example of “memory-based” prediction

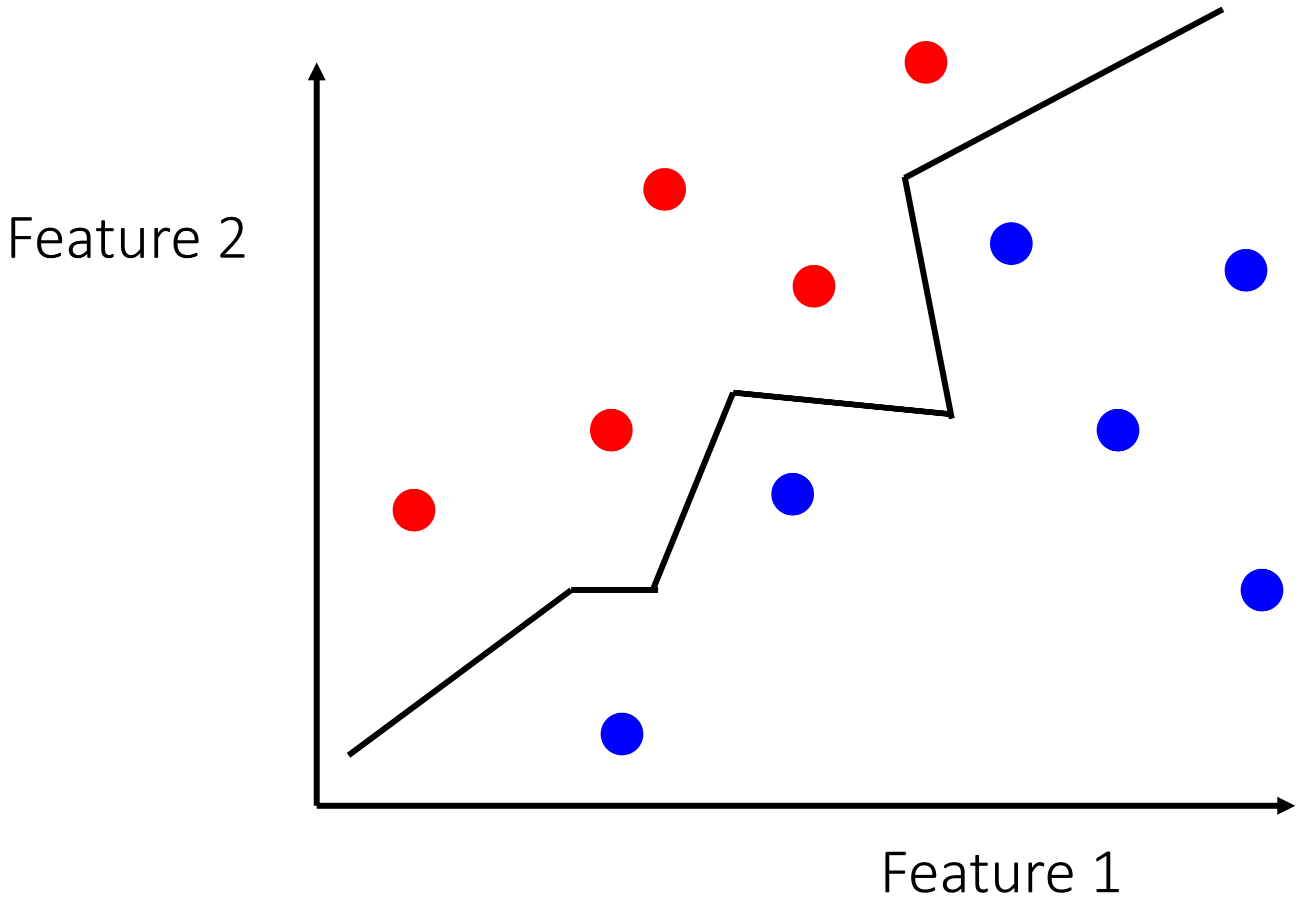
Single Nearest Neighbor Classification



Compute distances to all neighbors

Select closest neighbor and use its label for prediction

Single Nearest Neighbor Classification



kNN has piecewise linear decision boundaries (shown here for k=1)

More complex than nearest-centroid

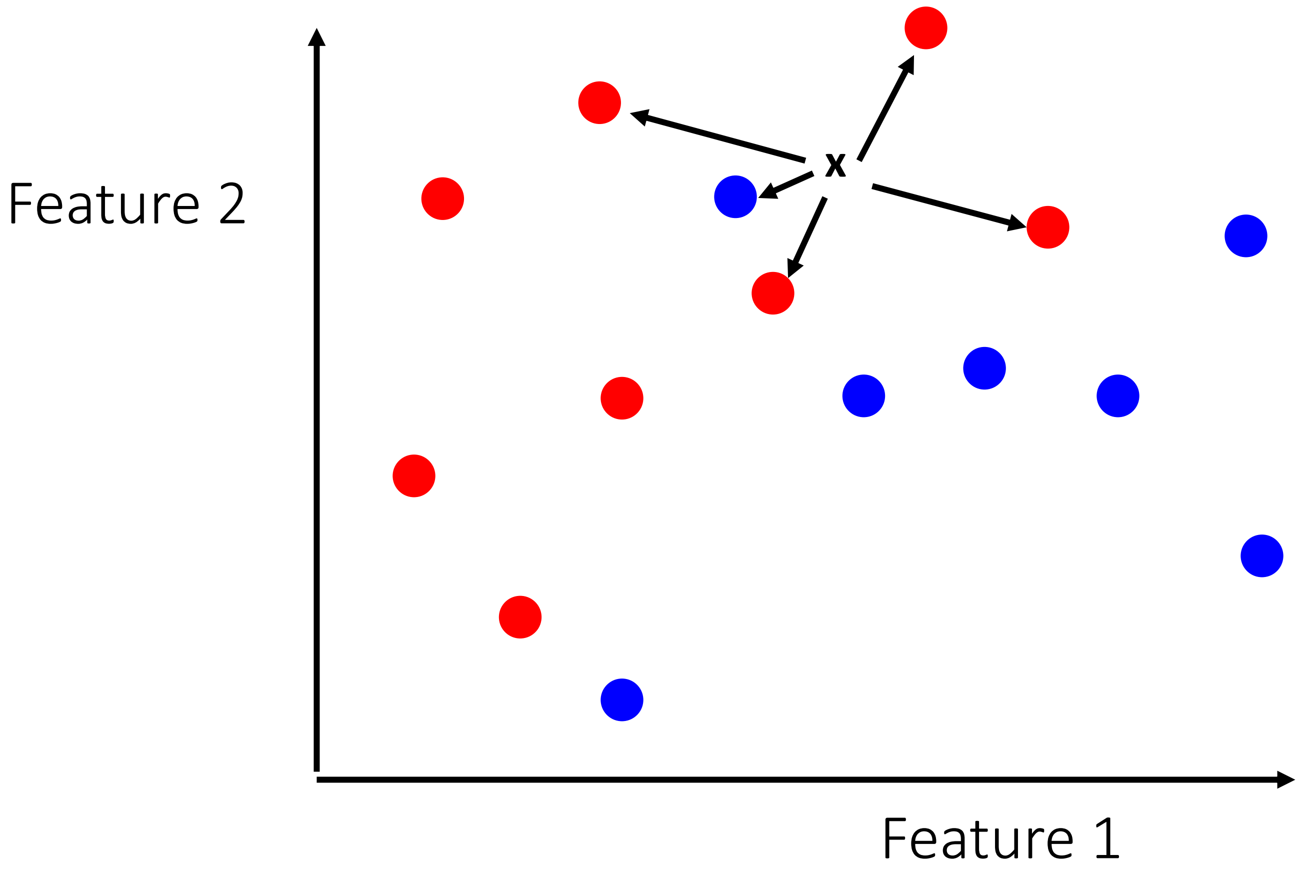
k-Nearest Neighbor Classification

- More general version

- Prediction Method for unlabeled **x**
  - Find the k closest points in the training data to **x**
  - Take the k labels of these k points
  - Predict the label is that is the majority of the k closest points
  - Example:
    - Labels from k nearest neighbors = {1, 3, 3, 1, 3} => majority label = 3

- Single neighbor is a special case with k=1

k-Nearest Neighbor (kNN) Classification



k = 1: predict blue

k = 3: predict red

k = 5: predict red

Additional Details

- How do we measure distance or closeness?
  - Default choice is usually Euclidean distance

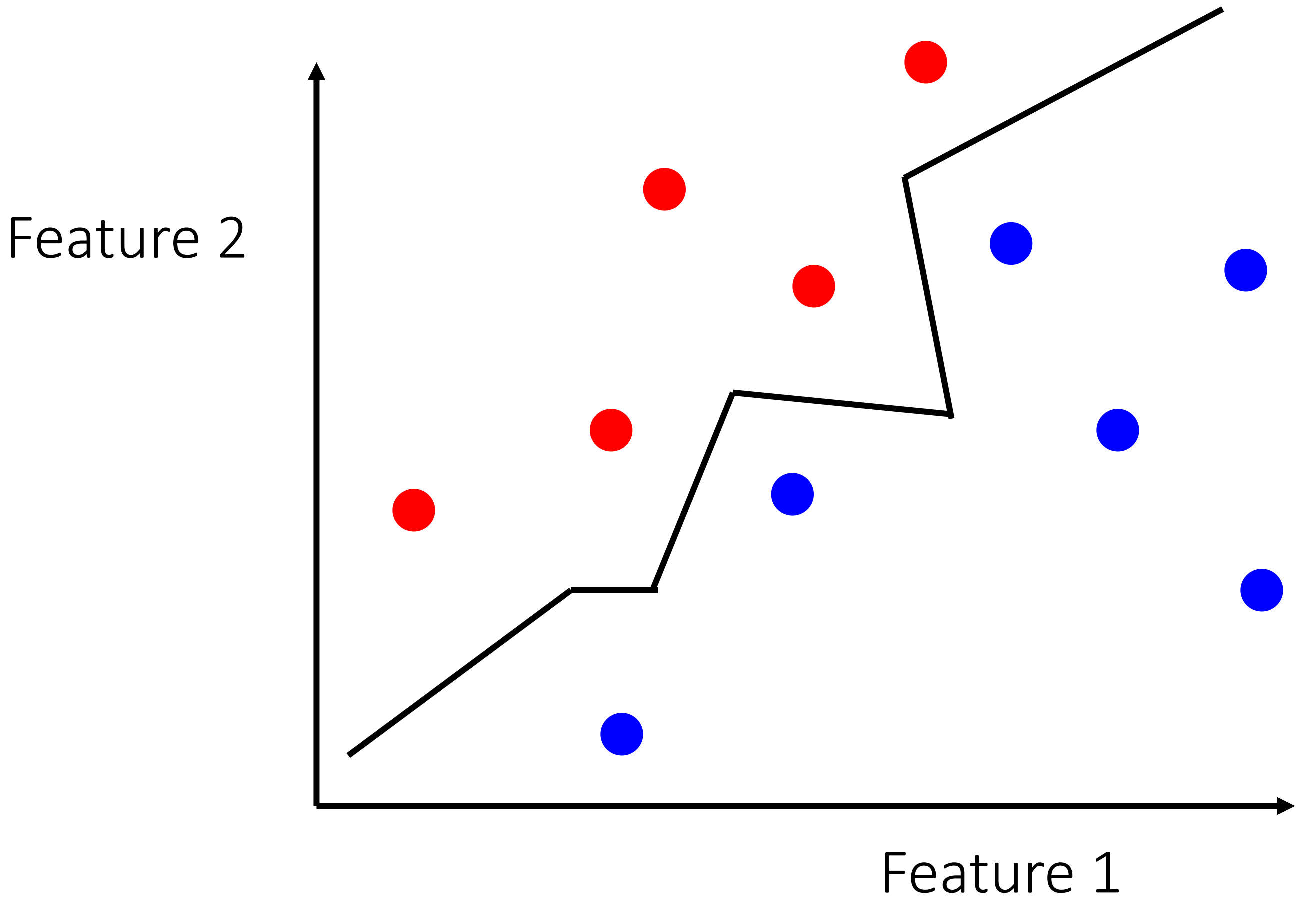
    ….but in principle any distance measure could be used

    ….. e.g., edit-distance if we had strings instead of feature vectors

- What do we do with ties?
  - Type 1: two or more points are equi-distant from x
  - Type 2: after we select k labels, there is no "winner"
  - In both cases we can break ties randomly

  - Note that if we have just 2 classes, we can avoid ties of Type 2 by selecting k to be an odd number, k = 1, 3, 5, …

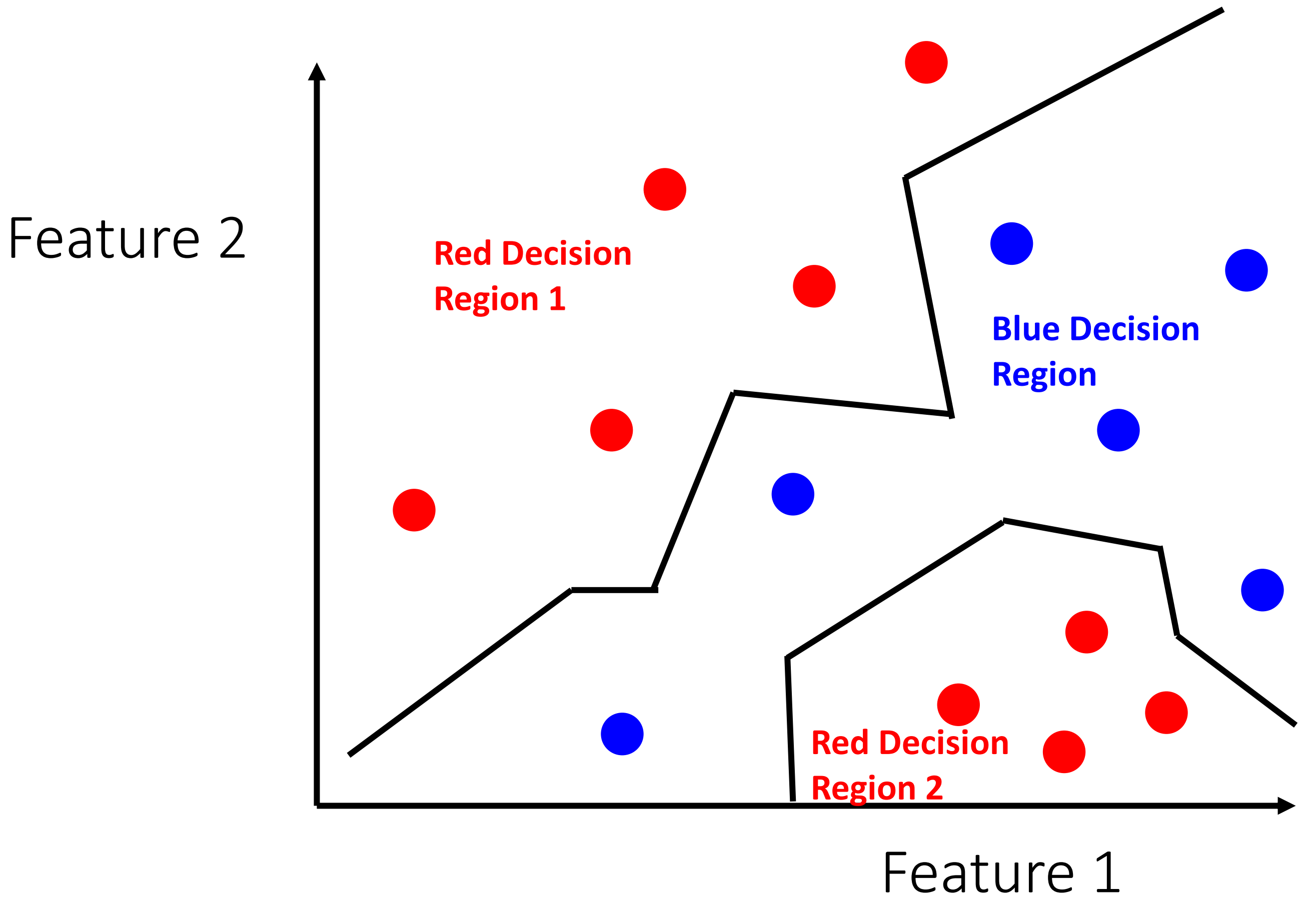
Decision Boundaries for kNN



kNN produces piecewise linear decision boundaries (shown here for k=1)

More complex than nearest-centroid

Disjoint Decision Regions for kNN



kNN can also model disjoint decision regions

Here there are two separate disjoint parts of the input space that will be predicted as the red class

Summary and Wrapup

- Nearest Centroids Classifiers
  - Classify new data by finding nearest class centroid
  - Piecewise linear decision boundaries
  - Simple to implement but can’t model complex decision boundaries

- Classifier evaluation
  - Accuracy
  - Confusion matrices
  - Precision, Recall

- k Nearest Neighbors
  - Classify new data by predicting the majority label of the nearest training data
  - More details next lecture

Questions?
(Outside after lecture)