Foundations of Data Science
Lecture 4

Rumi Chunara, PhD

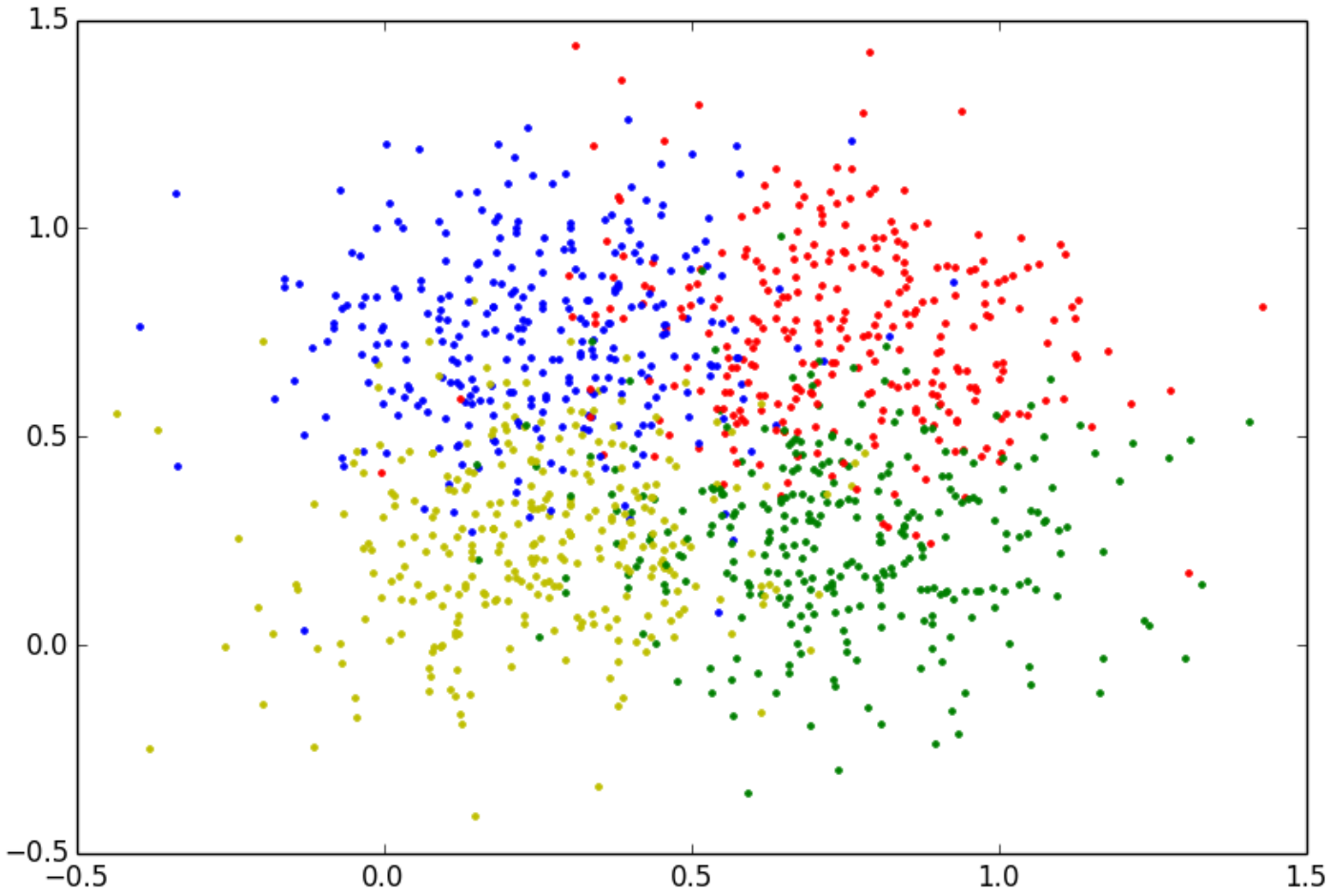
CS6053

k-Nearest Neighbors

- No training needed.
- Accuracy generally improves with more data.
- Matching is simple and fast (and single pass).
- Usually need data in memory, but can be run off disk.

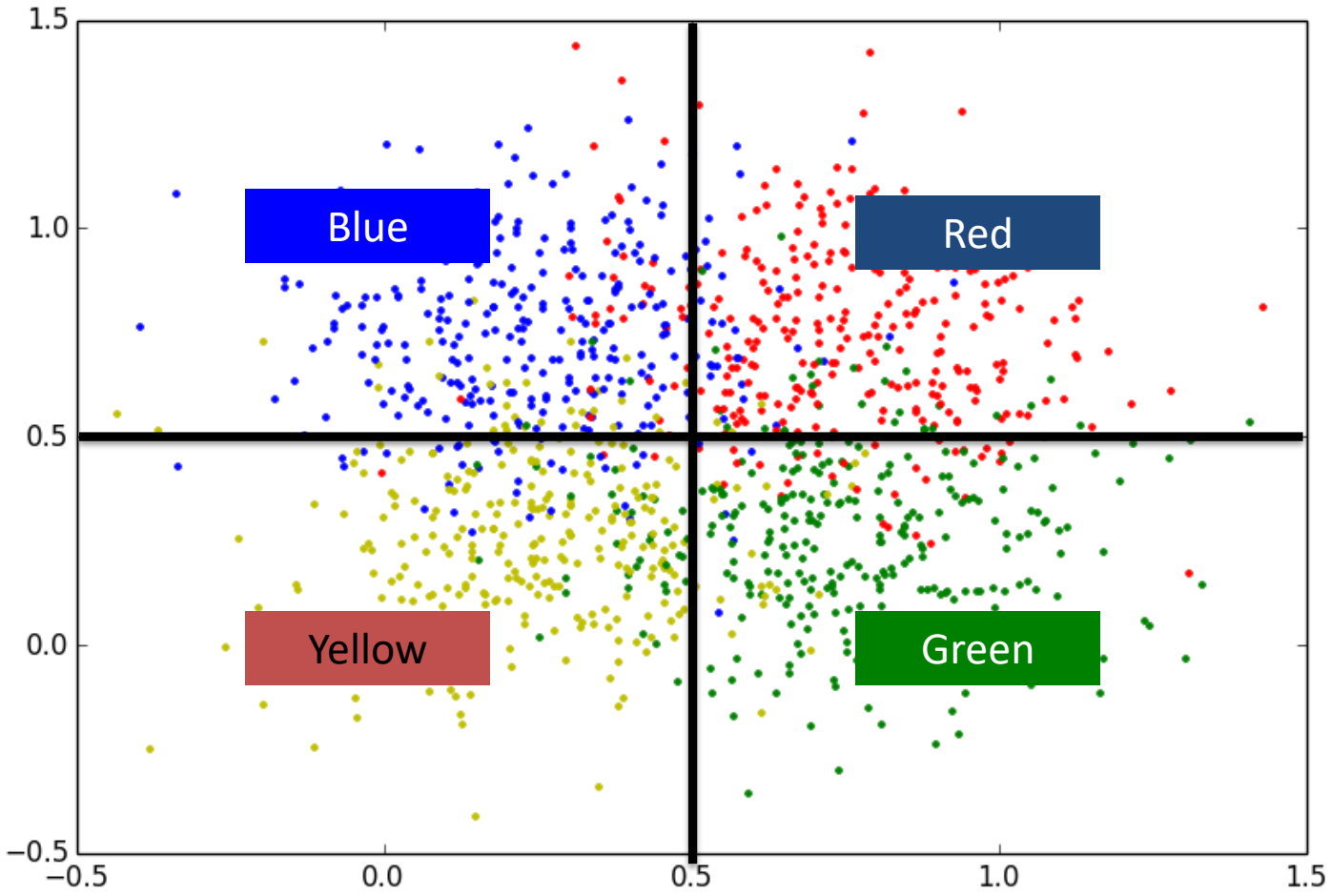
The Idea

How would you decide what color to give a point?



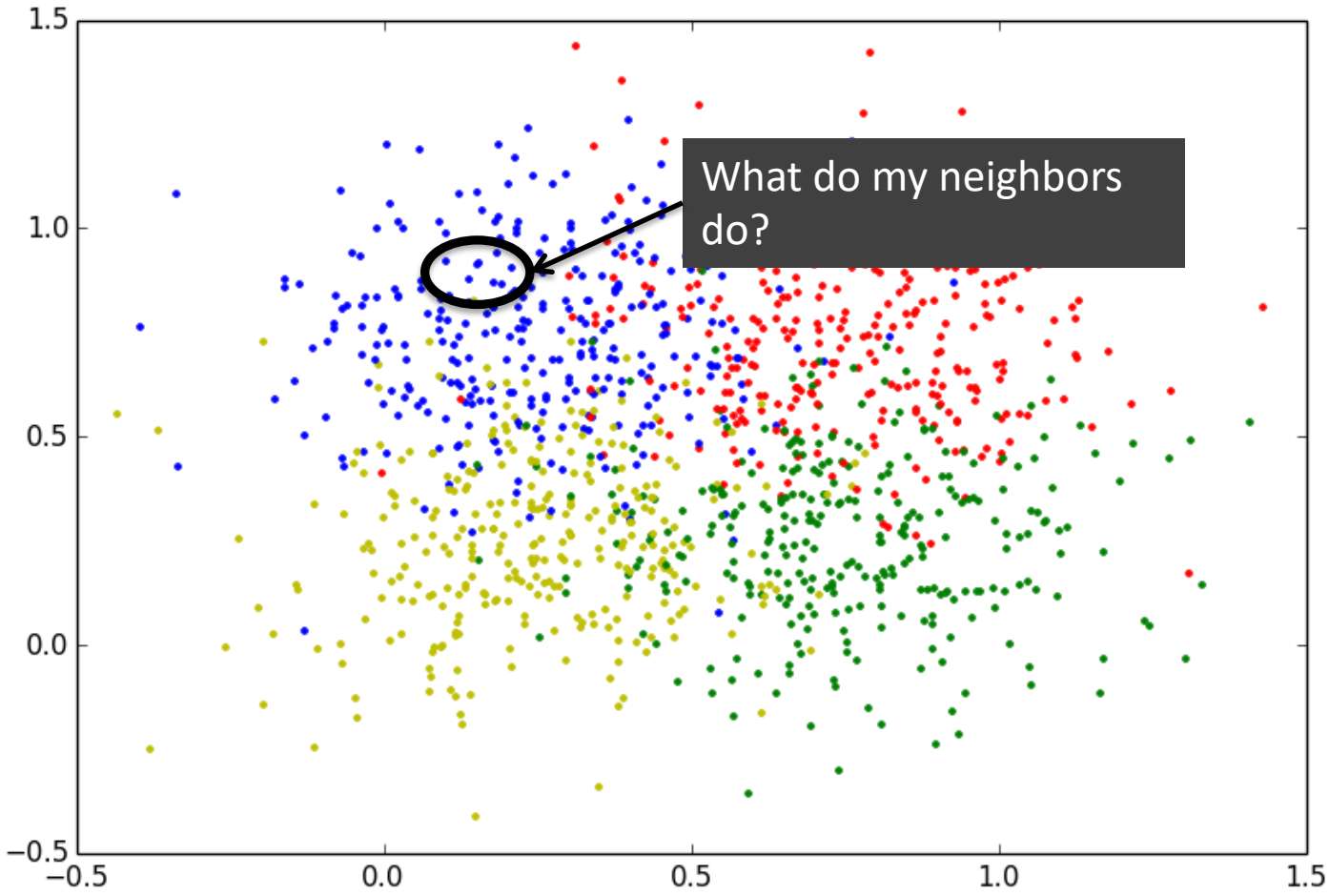
The Idea

Many algorithms focus on the shape/structure of the decision boundary (and we'll soon learn how to define and find those boundaries).



The Idea

K-NN on the other hand ignores any global structure and just focuses on local information.



Definition

More formally…

If we have a data set ***T=<X,Y>***, where ***X=<x_1,x_2,…x_k>*** is a vector of ***k*** features and ***Y*** is a real valued number (either a continuous number or binary indicator of class membership).

Let ***$N_k(x)$*** be the neighborhood of the k nearest neighbors in some metric space to an instance of interest. Then the regression or classification estimate for the given instance is:

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

Equation Source: Elements of Statistical Learning, Tibshirani et al.

What is a neighborhood?

A *neighborhood* is a set of examples that are *close* to the given instance.

To be *close* we need some *metric* that defines *distance* between two examples.

k-Nearest Neighbors

Given a query item:

Find k closest matches in a labeled dataset ↓

k-Nearest Neighbors

Given a query item:
Find k closest matches

Return the most
Frequent label

k-Nearest Neighbors

k = 3 votes for “cat”

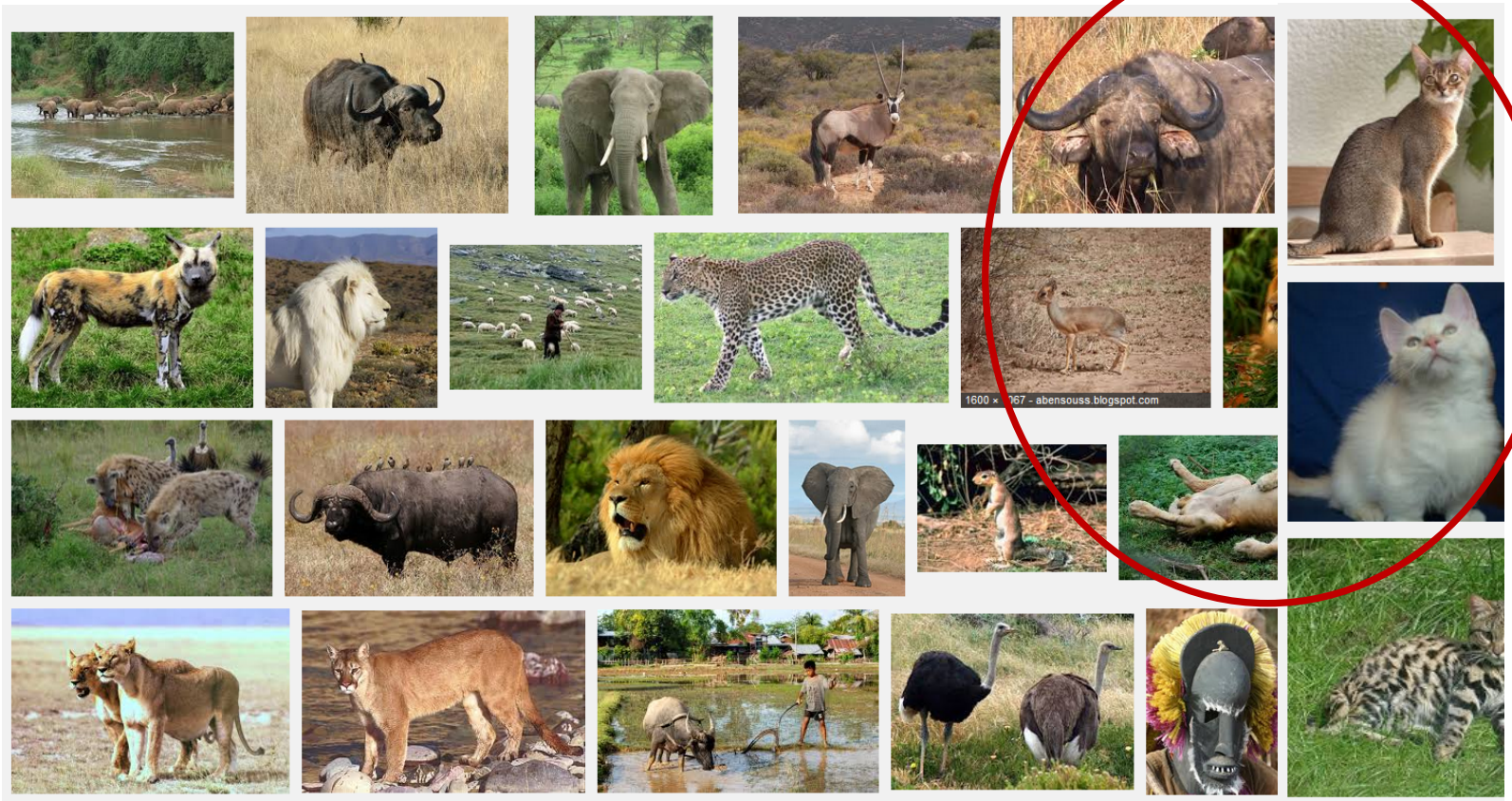
k-Nearest Neighbors

2 votes for cat,

1 each for Buffalo,

Deer, Lion

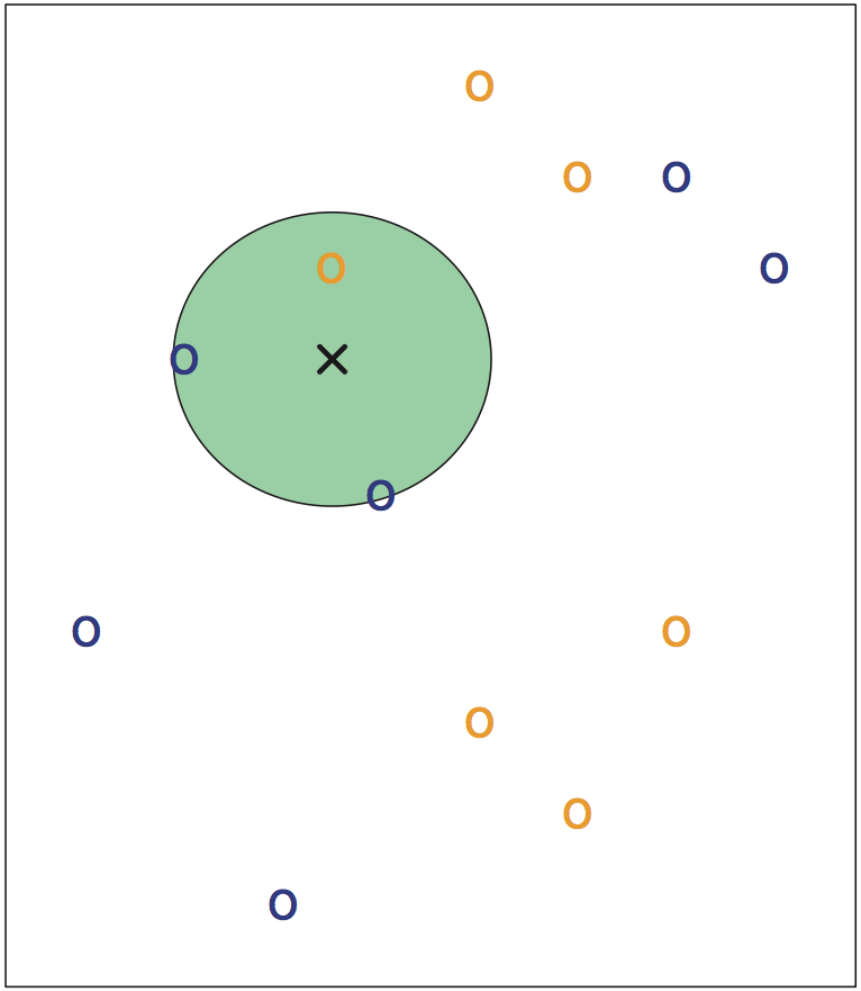
Cat wins…

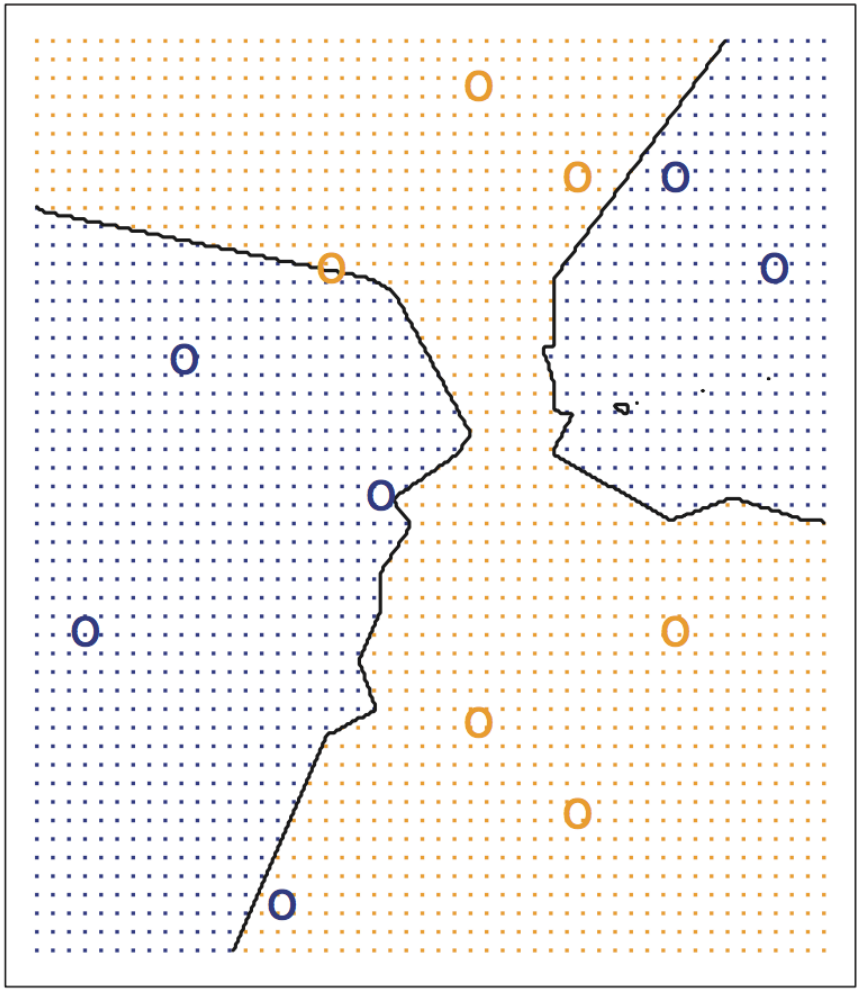


KNN – Method Motivation and Overview

- In theory we would always like to predict qualitative responses using the Bayes classifier (known probabilities; $P(Y = j \mid X = x_0)$
- For real data, we do not know the conditional distribution of Y given X
- Some approaches attempt to estimate the conditional distribution of Y given X, and then classify a given observation to the class with highest estimated probability (e.g. KNN classifier).
- Given a positive integer K and a test observation x_0, the KNN classifier first identifies the K points in the training data that are closest to x_0, represented by N_0. It then estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j
- Finally, KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability.

KNN Visual Explanation





KNN Qualities

The Data is the Model

- No training needed.
- Accuracy generally improves with more data.
- Matching is simple and fast (and single pass).
- Usually need data in memory, but can be run off disk.

Minimal Configuration:

- Only parameter is k (number of neighbors)
- Two other choices are important:
  - Weighting of neighbors (e.g. inverse distance)
  - Similarity metric

Qualities

Non-parametric

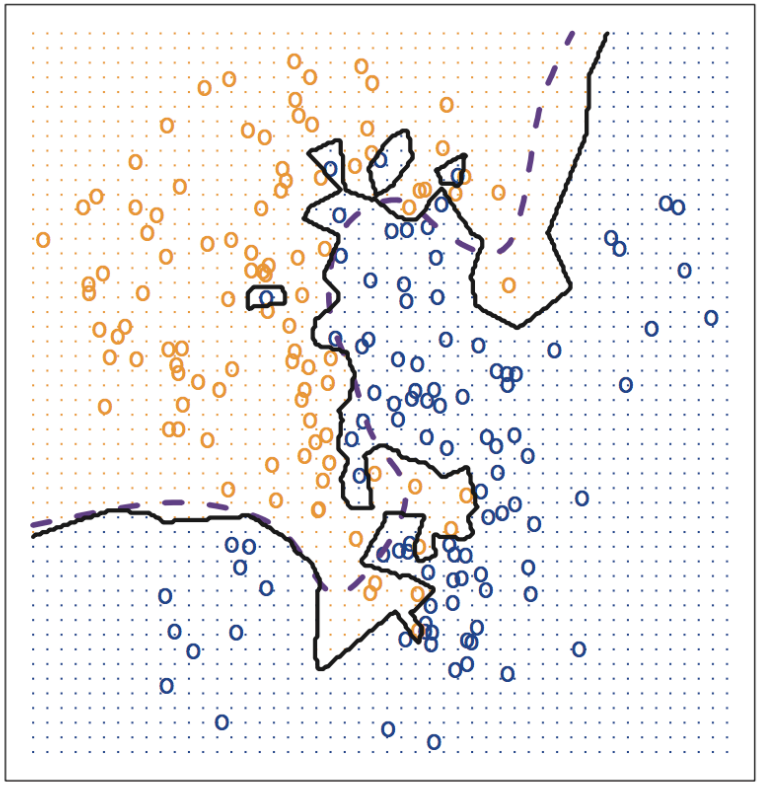
You don't have to make any assumptions about the functional form that estimates $E[Y|X]$. This makes it very powerful for estimating any arbitrary decision curve, but extreme flexibility always risks overfitting.

Instance-based learning

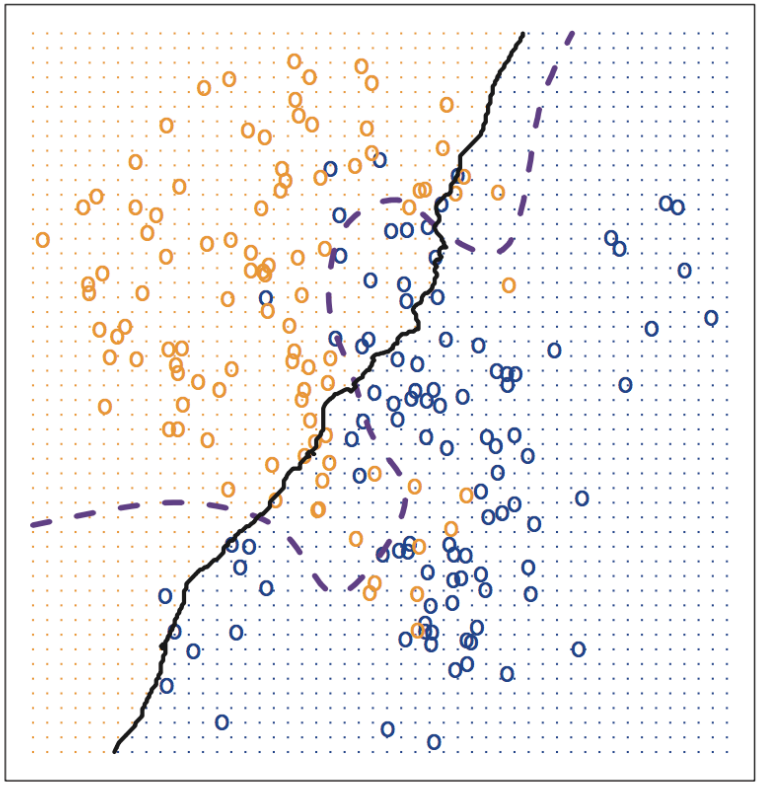
You technically don't need to train this. Estimation of $E[Y|X]$ is done locally and a scoring time by taking the average of Y of the k nearest neighbors of the instance. There is effectively no model, just all of the training data stored in memory.

Choice of “K”

KNN: K=1



KNN: K=100



Distance Metric

Definition: A metric is a function that defines a distance between elements of a set.

Properties:

Given two points a and b, and a distance function d()

1. $d(a,b) \geq 0$ … non-negativity
2. $d(a,b)=0$ only if and only if $a=b$
3. $d(a,b)=d(b,a)$ … symmetry
4. $d(a,c) \leq d(a,b)+d(b,c)$ … triangle inequality

Understanding these properties is helpful for using and validating various available distance metrics.

The metrics we'll explore in our examples is Euclidean Distance, though Hamming Distance and Mahalanobis Distance are often used.

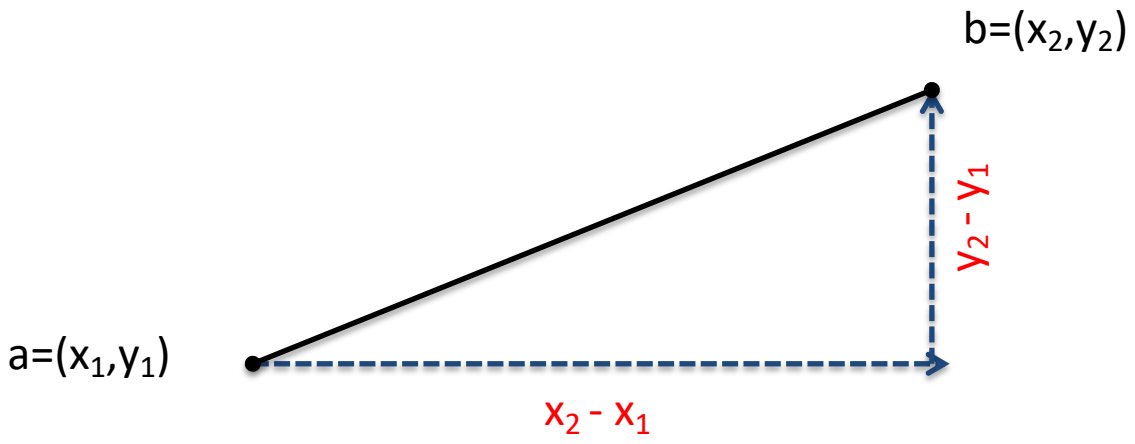
Euclidean Distance

This metric derives from basic geometry, and is the way distance is often defined in physical coordinate systems.

Let ***a*** and ***b*** be two k-dimensional vectors in Euclidean space. The Euclidean distance between them is defined as:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_k - b_k)^2} = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2}$$

The two dimensional case is the famous Pythagorean Theorem:



KNN metrics

- **Euclidean Distance:** Simplest, fast to compute:

$$= \left|\vec{a} - \vec{b}\right|^2 = (\vec{a} - \vec{b})^T (\vec{a} - \vec{b}) = 2(1 - \cos(\vec{a}, \vec{b}))$$

- **Cosine Similarity:** Good for documents, images, etc.

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a}^T \vec{b}}{\left|\vec{a}\right|\left|\vec{b}\right|}$$

- **Jaccard Index:** For set data, measure of similarity:

 # matching elements / # unique elements in both sets

- **Hamming Distance:** For string data: gives the result of how many attributes were different for vectors of equal length.

$$D_H = \sum_{i=1}^{k} x_i \neq y_i$$

KNN metrics

- **Manhattan Distance:** Coordinate-wise distance:

$$\sum_{i=1}^{k} | x_i - y_i |$$

- **Edit Distance:** for strings, especially genetic data.

Given two strings *a* and *b* on an alphabet Σ (e.g. the set of ASCII characters, the set of bytes [0..255], etc.), the edit distance d(*a*,*b*) is the minimum-weight series of edit operations that transform *a* into *b*.

Selecting Distance Measures

- Suppose that you are comparing how similar two organisms of different species are in terms of the number of genes they share. Describe which measure: Hamming or Jaccard, you think would be more appropriate for comparing the genetic makeup of two organisms.
- Assume that each animal is represented as a binary vector, where each attribute is 1 if a particular gene is present in the organism and 0 otherwise.

Selecting Distance Measures

- When comparing if two organisms of similar species the important genes to compare are the alike genes. In the example below when comparing two organisms represented by a binary vector with an attribute of 1 indicating the gene is present; there are 2 genes that are alike out of 10.

Example:

x = (0,0,0,1,1,1,0,0,1,0)

y = (1,0,0,1,0,0,1,0,1,1)

- Thus Jaccard Coefficient is informative because the only genes that are important are the alike genes that are that are present. Genes that are not present in either organism does not support that the two organisms are similar.

Selecting Distance Measures

- If you wanted to compare the genetic makeup of two organisms of the same species, e.g., two human beings, would you use the Hamming distance, the Jaccard coefficient, or a different measure of similarity or distance?
- Note that two human beings share > 99.9% of the same genes.

Selecting Distance Measures

- Since the genes of the two humans are 99.9% alike genes, the only genes of interest are ones that are different. The Hamming distance approach only looks at incidents at are different. For example human x and y represented as binary vectors, where each attribute of a gene present is a 1 and otherwise 0 if the gene is not present.

- x = (1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,
  1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,
  1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0, 1,1,1,1,0,0,0,1,1,0)

- y = (1,1,1,1,0,1,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,
  1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,
  1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0,1,1,1,1,0,0,0,1,1,0, 1,1,1,1,0,0,0,1,1,0)

- The Hamming distance results 1, one difference out of 100 attributes revealing that 99 genes are the same.

Implementing the “model”

1. Build a training set – must have a label and numeric features.

User	Y	X1	X2	X3
1	1	0.64	0.72	0.48
2	0	0.02	0.01	0.21
3	0	0.19	0.20	0.87
4	0	0.39	0.16	0.29
5	1	0.87	0.69	0.17
6	0	0.41	0.09	0.54
7	0	0.54	0.46	0.12
8	1	0.97	0.26	0.35
9	0	0.56	0.52	0.35
10	1	0.96	0.29	0.84

Implementing the "model"

2. Choose a non-training instance and compute distance (we'll use Euclidean) from every member of the training set.

User	Y	X1	X2	X3
11	?	0.57	0.43	0.95

User	Y	X1	X2	X3	D
1	1	0.64	0.72	0.48	0.56
2	0	0.02	0.01	0.21	1.02
3	0	0.19	0.20	0.87	0.46
4	0	0.39	0.16	0.29	0.74
5	1	0.87	0.69	0.17	0.88
6	0	0.41	0.09	0.54	0.56
7	0	0.54	0.46	0.12	0.83
8	1	0.97	0.26	0.35	0.74
9	0	0.56	0.52	0.35	0.61
10	1	0.96	0.29	0.84	0.43

Implementing the "model"

3. Find the k nearest neighbors (k should be chosen in advance), and average the labels Y in the training set.

$$E[Y|X_{11}] = 1/3$$

User	Y	X1	X2	X3
11	?	0.57	0.43	0.95

User	Y	X1	X2	X3	D
1	1	0.64	0.72	0.48	0.56
2	0	0.02	0.01	0.21	1.02
3	0	0.19	0.20	0.87	0.46
4	0	0.39	0.16	0.29	0.74
5	1	0.87	0.69	0.17	0.88
6	0	0.41	0.09	0.54	0.56
7	0	0.54	0.46	0.12	0.83
8	1	0.97	0.26	0.35	0.74
9	0	0.56	0.52	0.35	0.61
10	1	0.96	0.29	0.84	0.43

Challenges

Expensive Search

The algorithm becomes more accurate when the number of samples N increases. However, as N increases the search cost increases. Scoring a new instance requires $O(N_{train})$ searches when a brute force search is used.

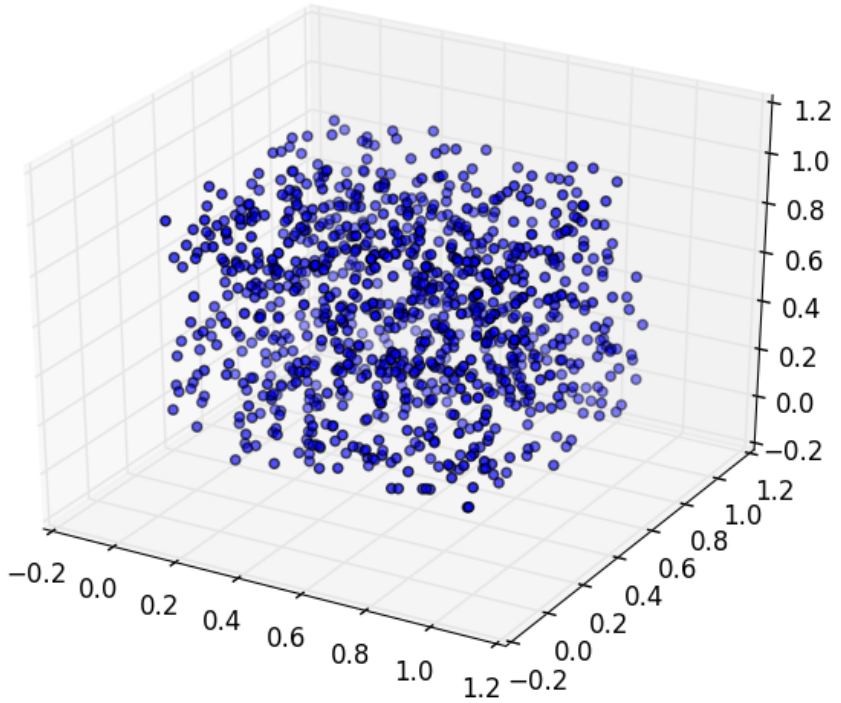
Distance Metrics

The Curse of Dimensionality (CODA) – when using Euclidean distance (as well as other Lp-norm distance measures), avg. distances increase as dimensionality increases. In a high-dimensional space, most parts are far from all other points.

Scale matters – features with higher avg. magnitude tend to dominate distance metrics. It may be required to normalize data first.

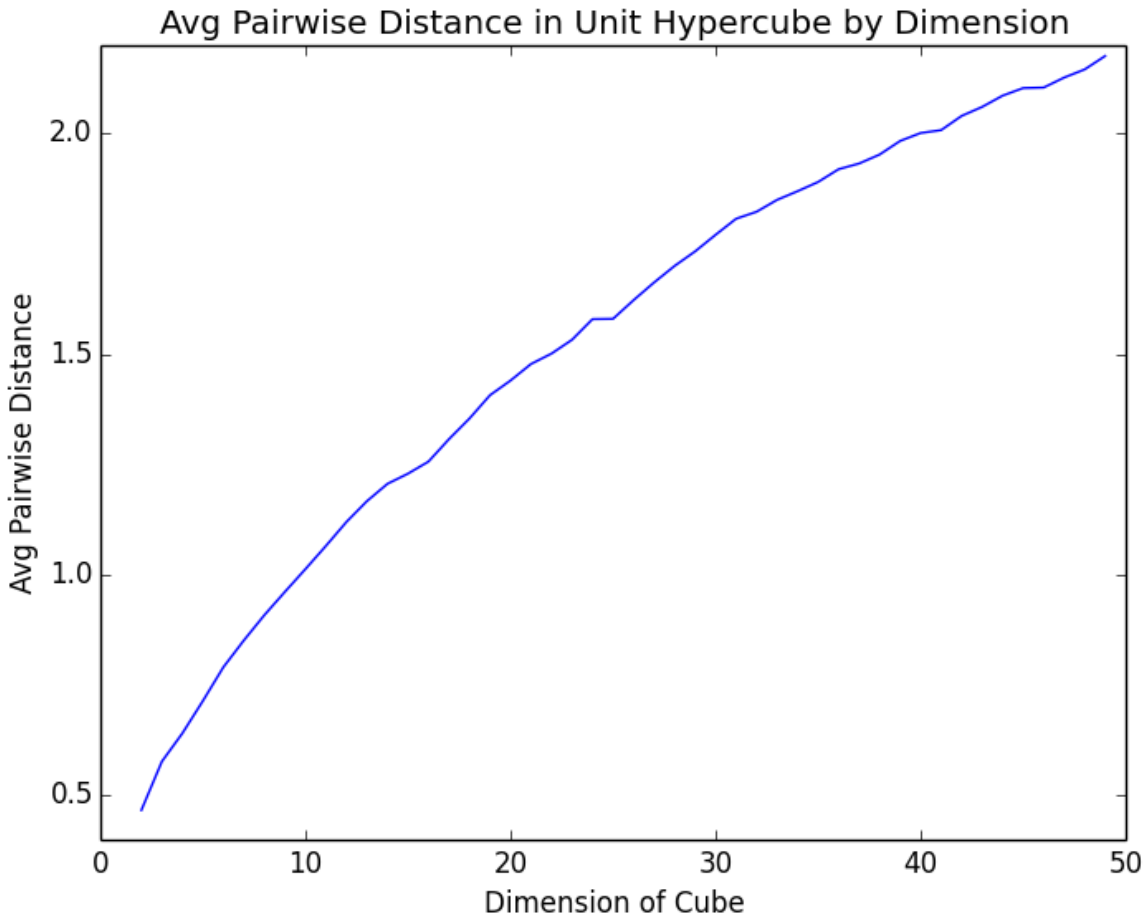
CODA - Simulation

1. Sample 1000 random points uniformly in the d-dimensional hypercube: i.e, $X \sim \text{Uniform}([0,1])^d$
2. Compute all pairwise distances across a range of d
3. Look at the distribution of distance, given d, and show E[distance|d]



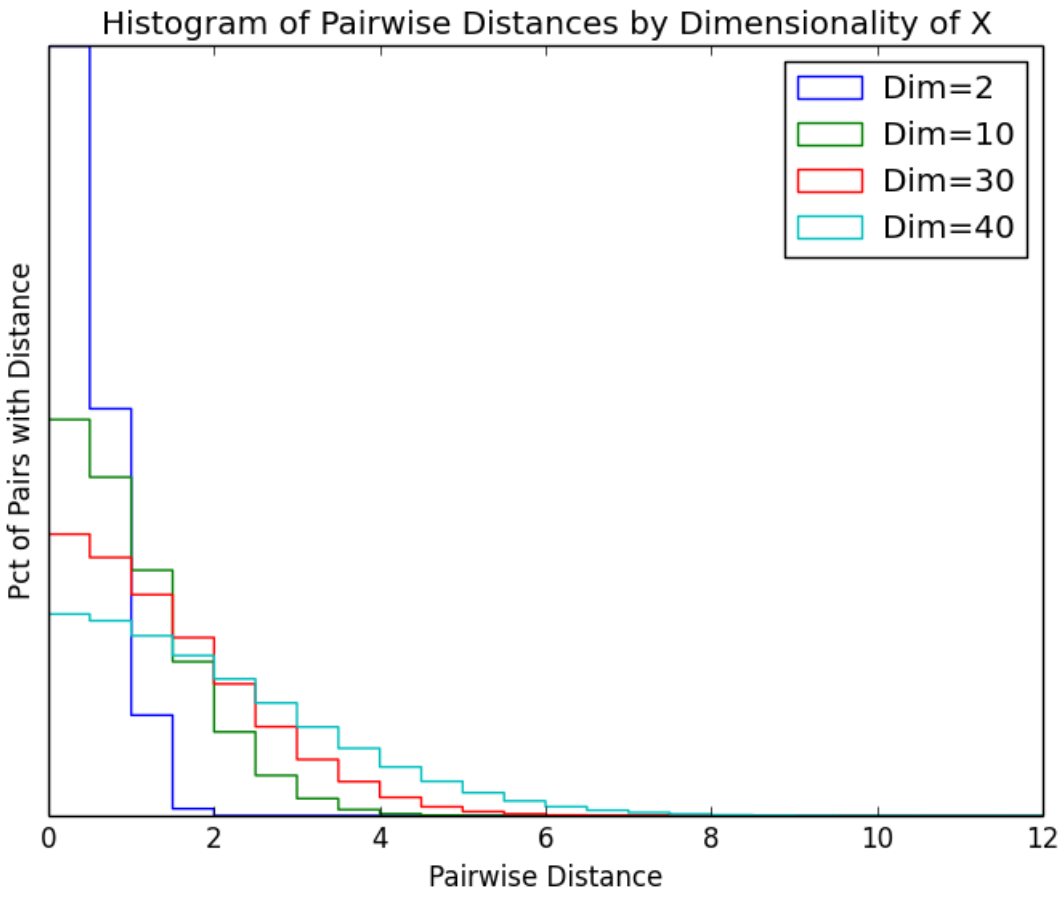
CODA - Simulation

E[distance|d] increases monotonically with d



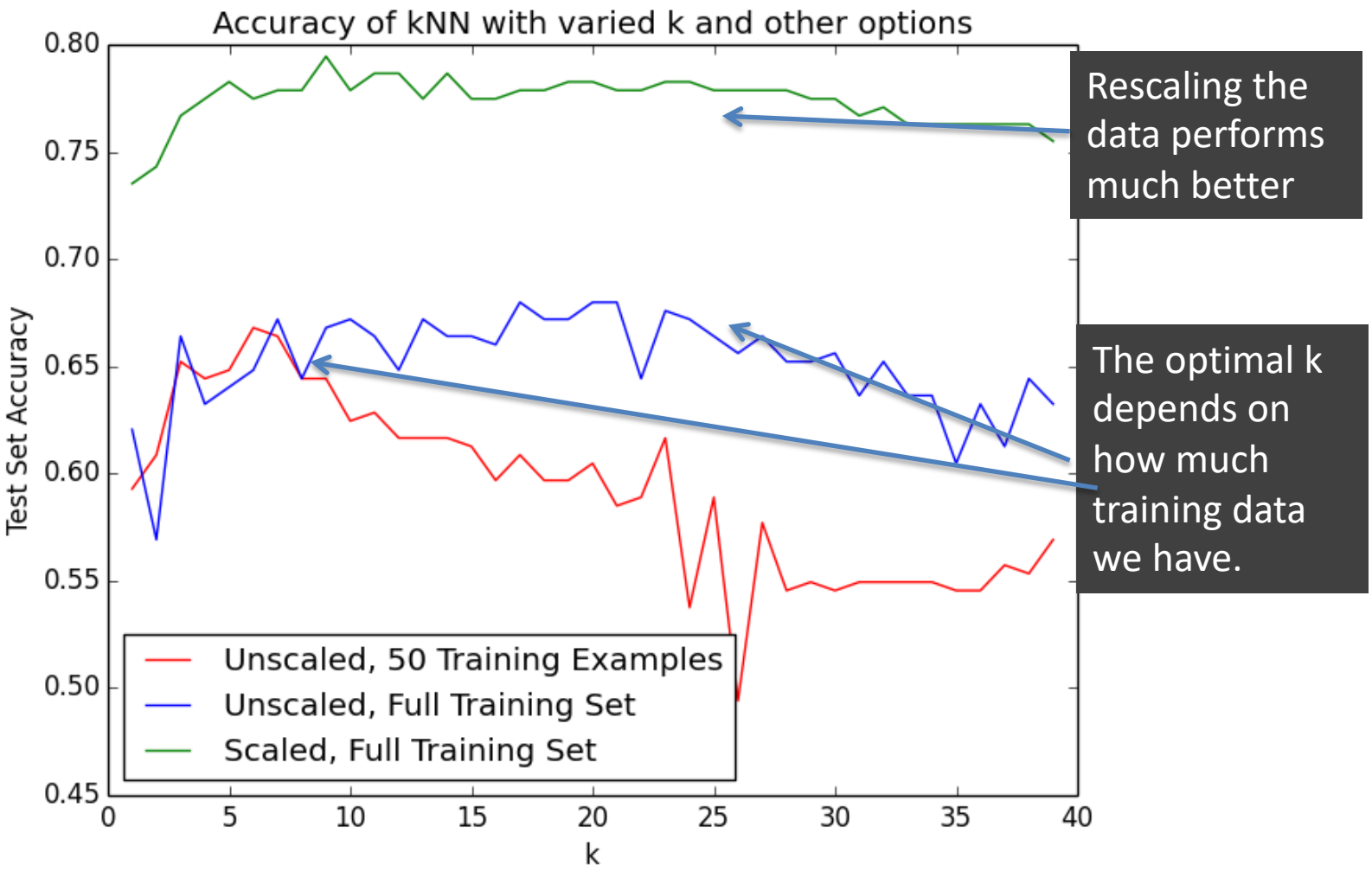
CODA - Simulation

We can see more points being further away from each other as d increases. As d increases, we need to go farther and farther away to find the k nearest neighbors. At a certain point, the k neighbors aren't exactly local.



Example – KNN classifier

This plot shows accuracy vs. k on a test set for kNN classification with several design options. This sort of analysis is how we would normally choose design parameters and hyper-paramaters.



Today

- Intro to ML – what is it
- Two Basic Algorithms
  - Linear Regression
  - kNN