

Introduction to Data Science

Data Mining for Business Analytics

BRIAN D'ALESSANDRO

VP – DATA SCIENCE, DSTILLERY

ADJUNCT PROFESSOR, NYU

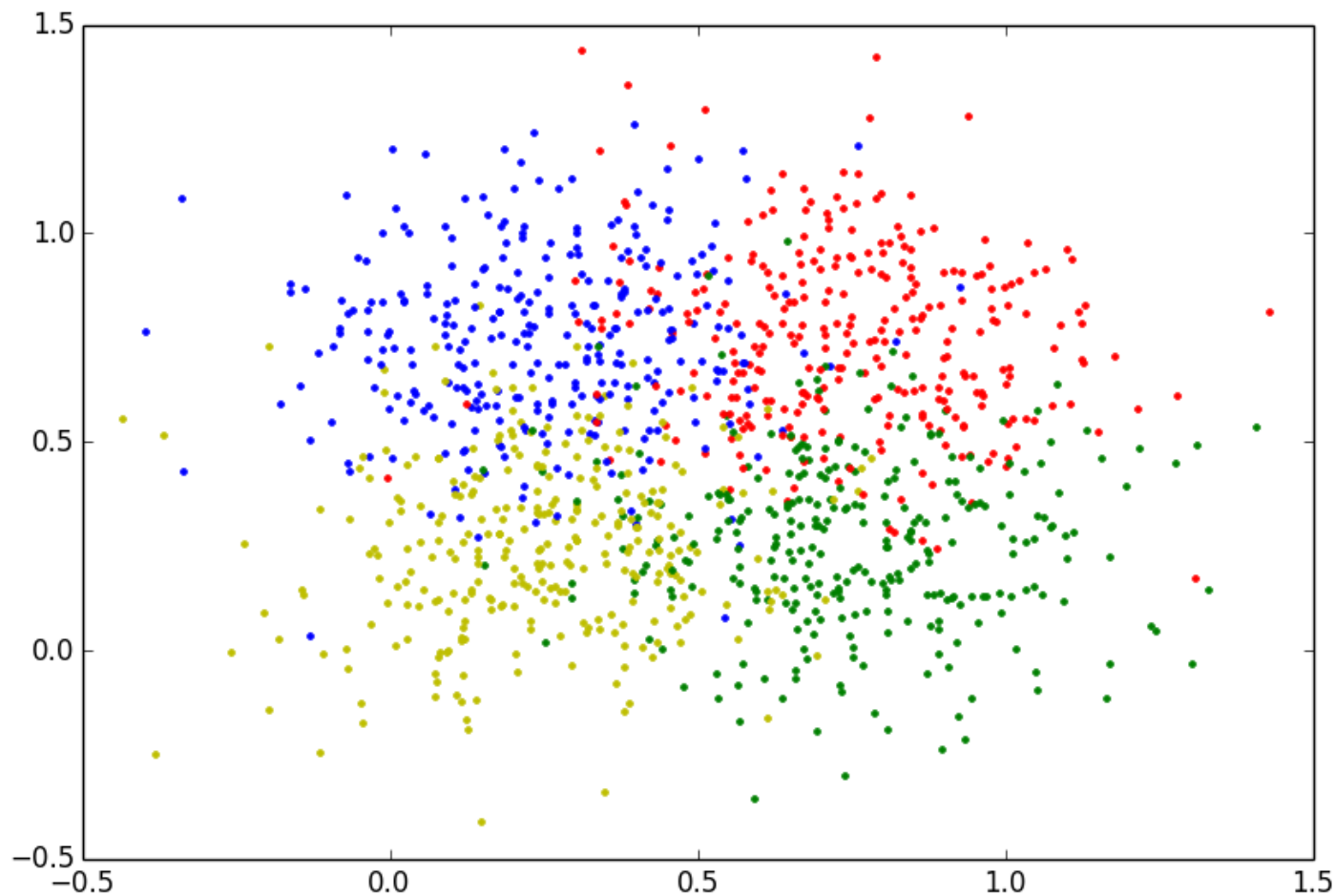
FALL 2014

Fine Print: these slides are, and always will be a work in progress. The material presented herein is original, inspired, or borrowed from others' work. Where possible, attribution and acknowledgement will be made to content's original source. Do not distribute, except for as needed as a pedagogical tool in the subject of Data Science.

K-NEAREST NEIGHBORS

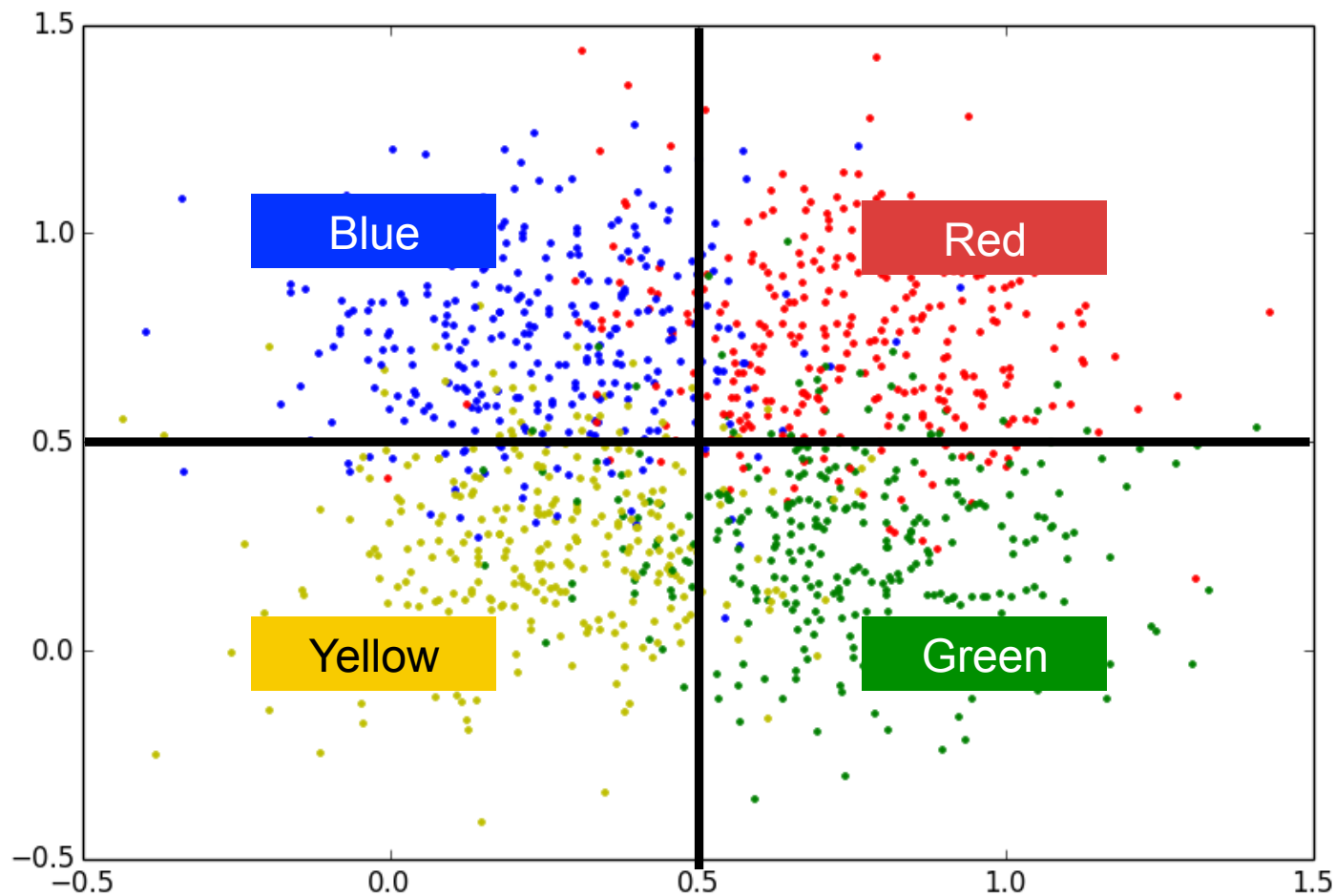
THE IDEA

How would you decide what color to give a point?



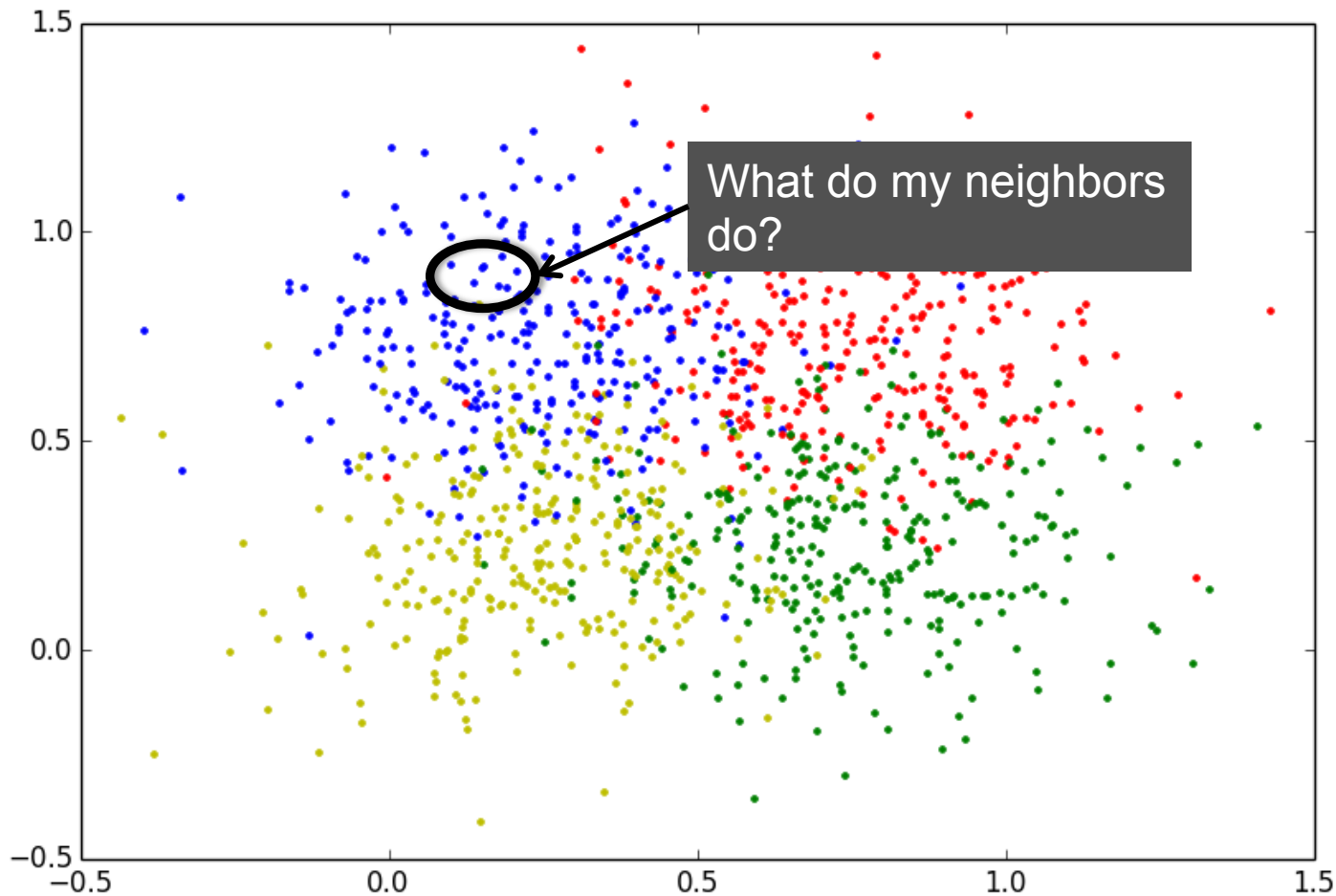
THE IDEA

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 $T = \langle X, Y \rangle$, where $X = \langle x_1, x_2, \dots, x_k \rangle$ 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.

SOME 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.

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.



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.

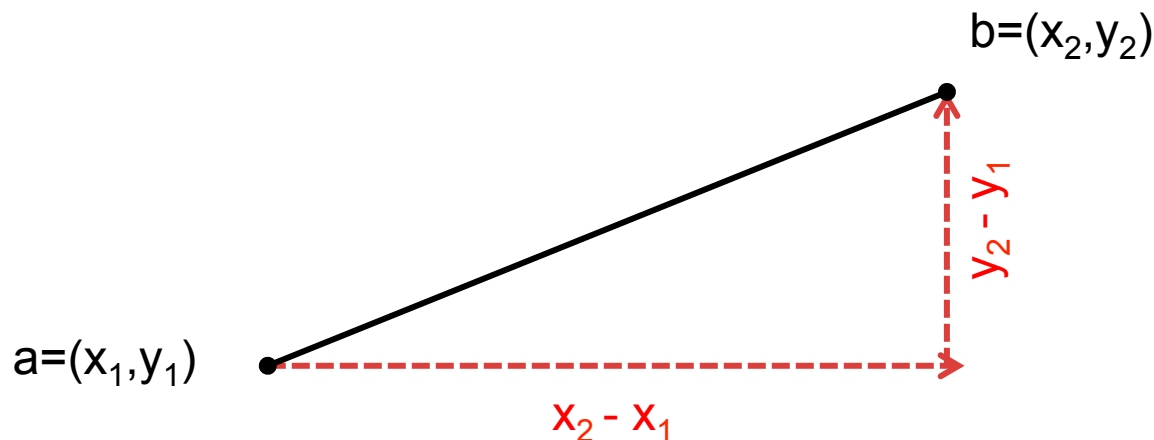
EUCLIDEAN DISTANCE

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:

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

The two dimensional case is the famous Pythagorean Theorem:



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
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

D
0.56
1.02
0.46
0.74
0.88
0.56
0.83
0.74
0.61
0.43

IMPLEMENTING THE “MODEL”

3. Find the k nearest neighbors (and 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
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

D
0.56
1.02
0.46
0.74
0.88
0.56
0.83
0.74
0.61
0.43

K-NN IN SCIKIT LEARN

Scikit learn has a fairly efficient implementation of kNN. It is recommended to use this should you ever need kNN, as the developers have used an efficient search algorithm for finding the top-k neighbors.

More info here <http://scikit-learn.org/stable/modules/neighbors.html>

```
from sklearn.neighbors import KNeighborsClassifier

#Specify k and use p=2 for Euclidean distance
neigh = KNeighborsClassifier(n_neighbors=k,p=2)

#doesn't really fit the model in a classic sense, it does some intelligent
partitioning of the data
neigh.fit(X_train,y_train)

#Get the predictions...note, this returns a class label prediction, not a
probability
ypred = neigh.predict(X_test)
```

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 $O(N_{\text{train}})$ searches when a brute force search is used.

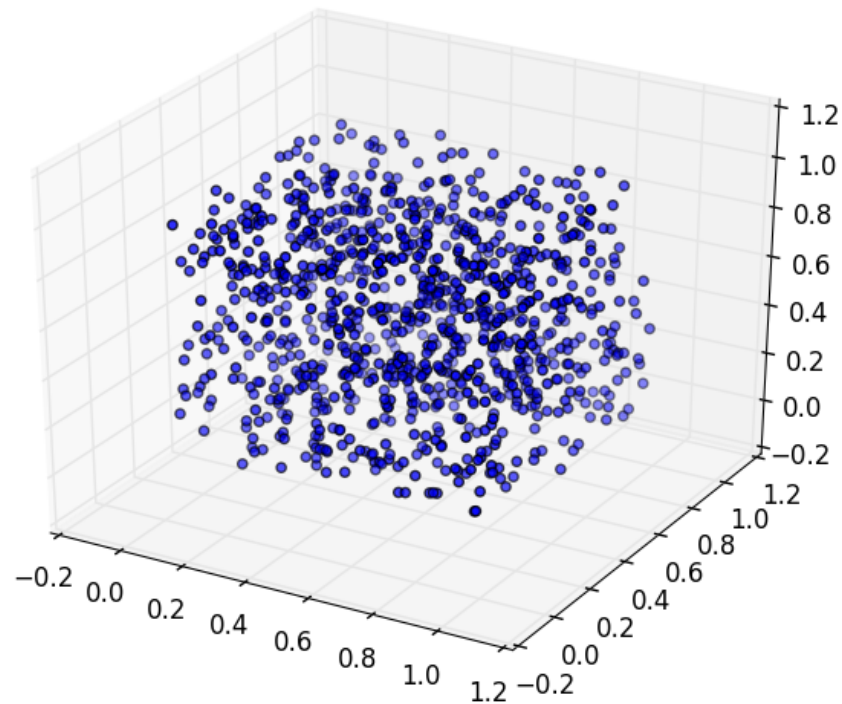
Distance Metrics

The Curse of Dimensionality (CODA) – when using Euclidean distance (as well as other L_p -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.

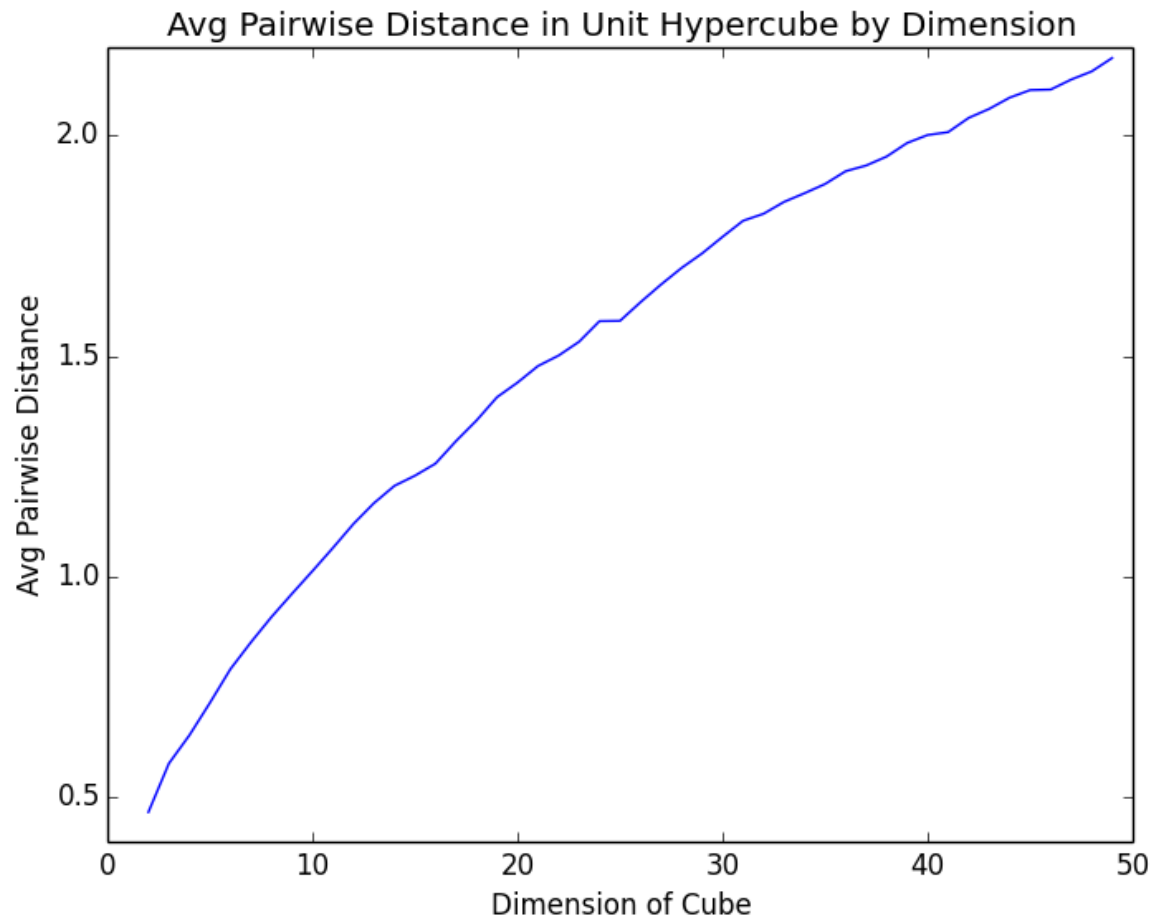
CODA - SIMULATION

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 $E[\text{distance}|d]$



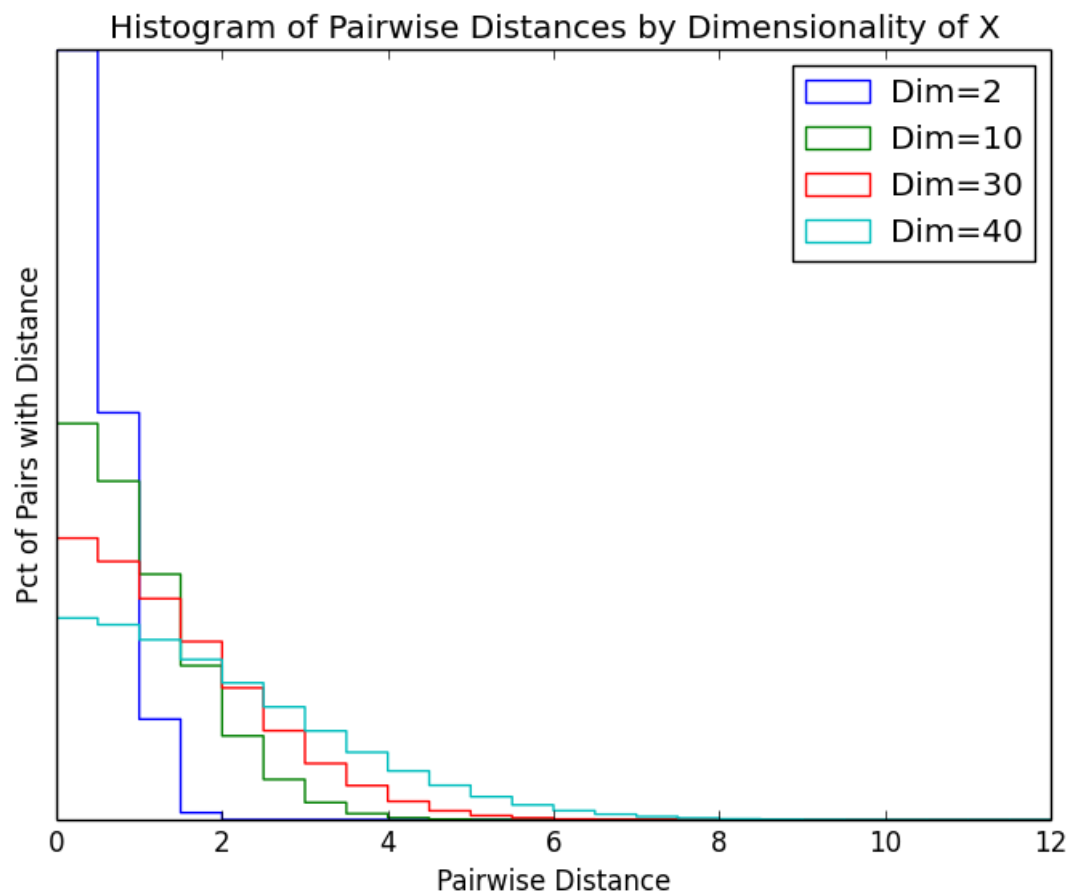
CODA - SIMULATION

$E[\text{distance}|d]$ increases monotonically with d



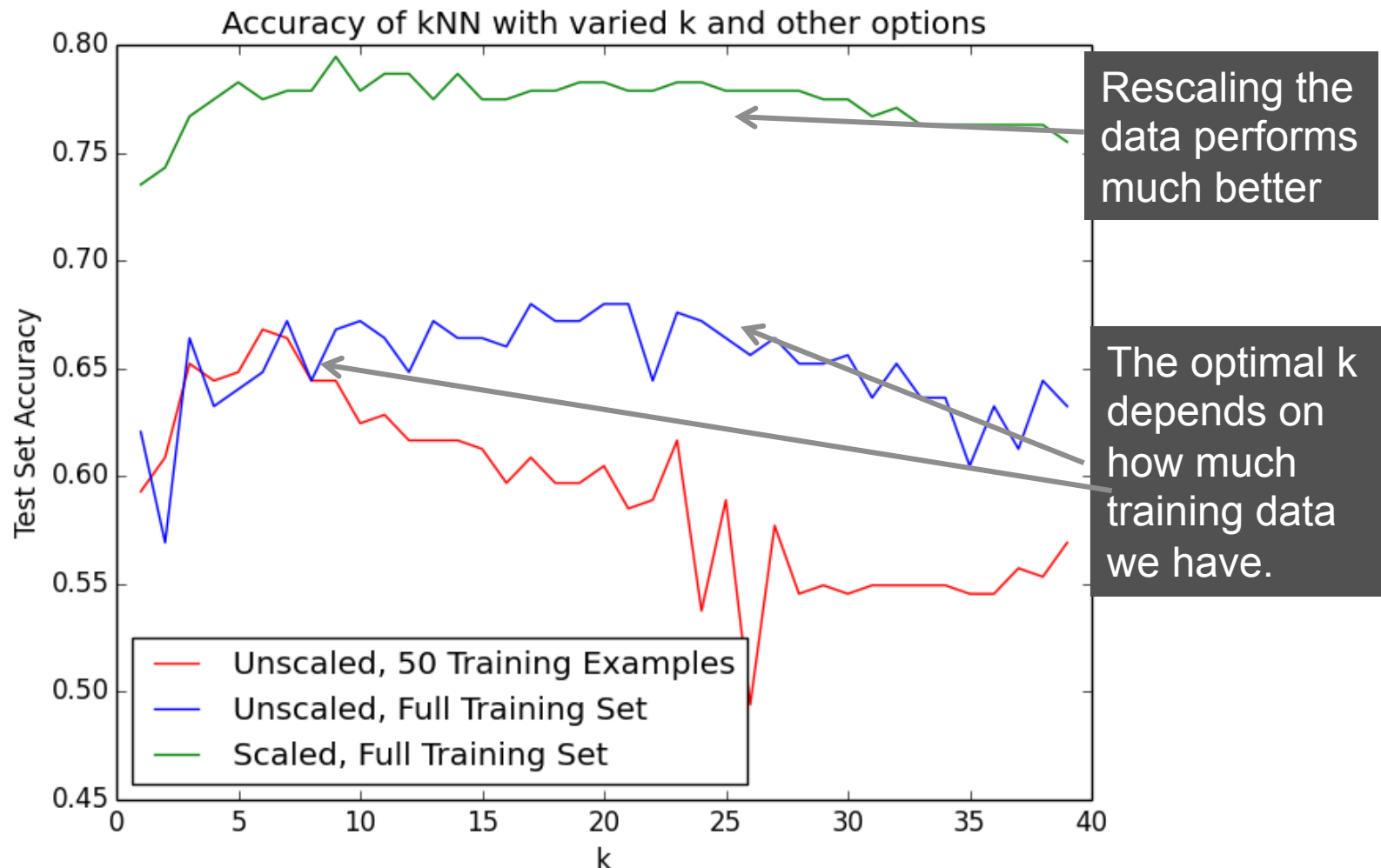
CODA - SIMULATION

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 hyperparameters.



FINAL THOUGHTS

kNN is best suited for...

- Low dimensionality
- High N
- Complex conditional distributions $E[Y|X]$
- Interpretability

Ultimately, of course, you'll always want to test.