Foundation of Data Science
Lecture 5, Module 3
Spring 2022

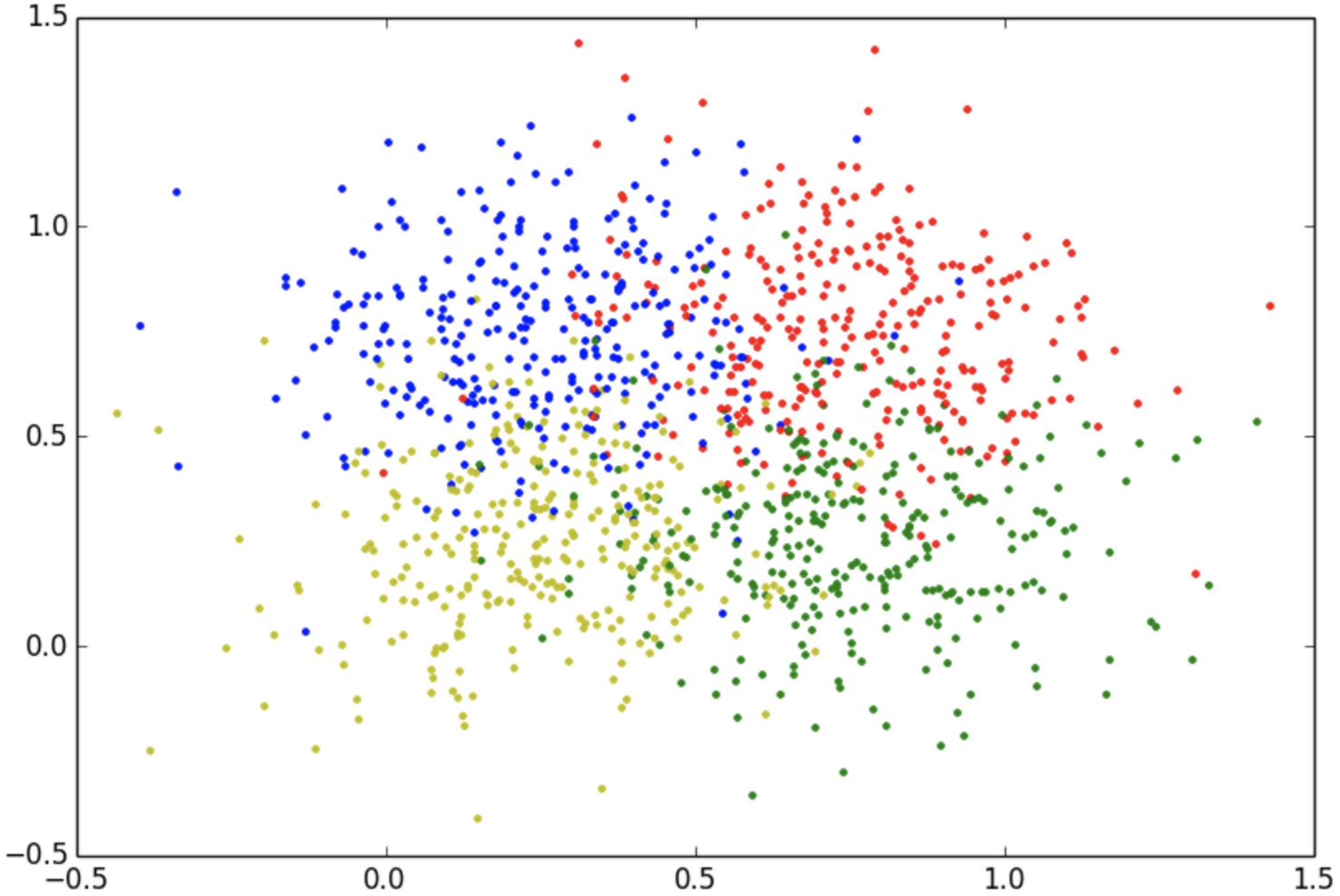
Rumi Chunara, PhD

Today

- Supervised Learning Algorithms
  - Decision Trees
  - Ensemble Methods
  - kNN

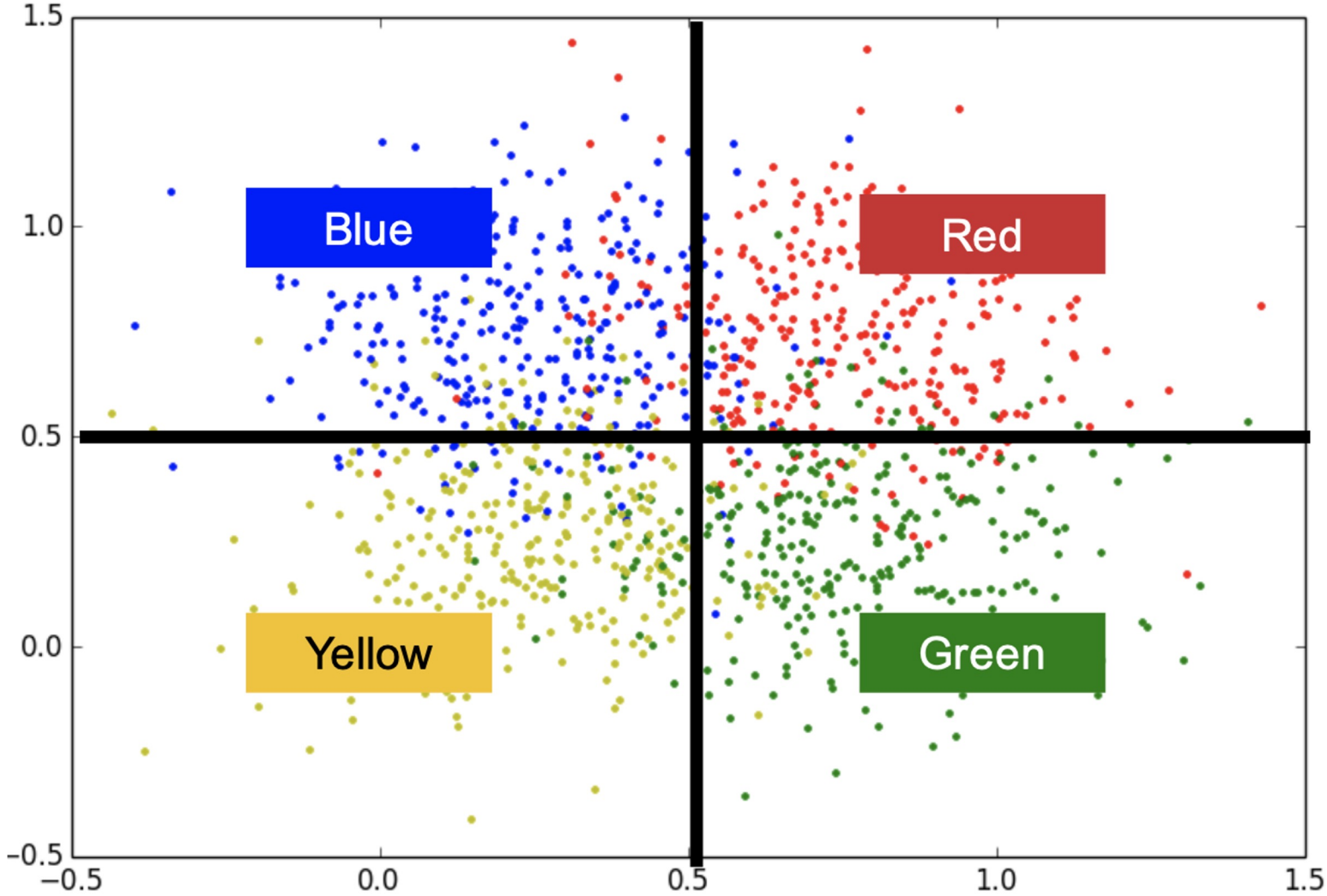
The Idea

If your training data is just like the picture below (two features, and one color per label), how would you decide what color to give to a test point (given its feature values)?



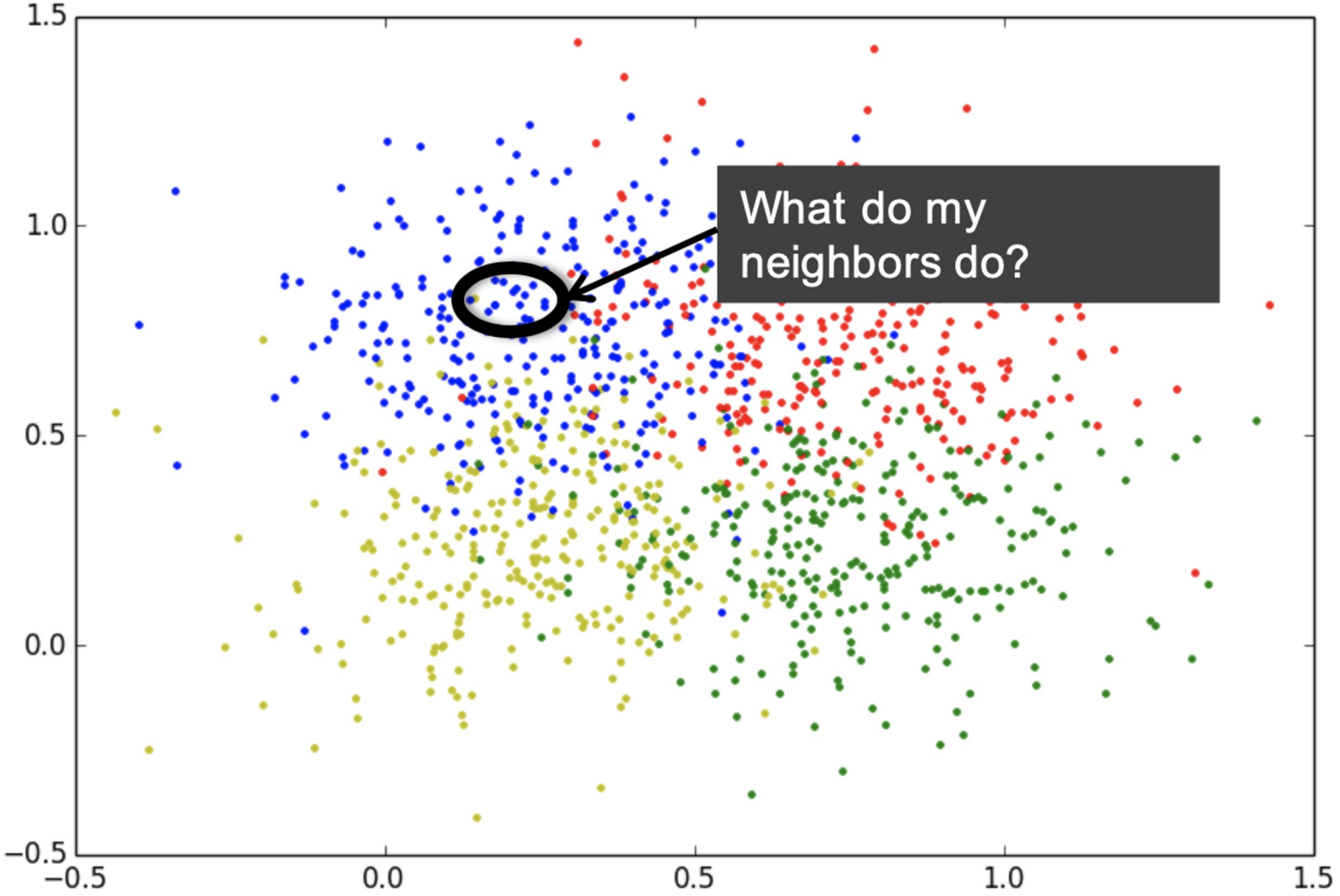
The Idea

Many supervised algorithms learn the shape of the decision boundary and use functions or rules to draw them



The Idea

K-NN ignores any global structure and just focuses on local information, determining the label of a new point “on demand”



Definition

Let ***D={X,Y}*** be a training dataset where each ***X=<x1,x2,...xm>*** is a vector of ***m*** features and ***Y*** is its corresponding target. Given a test point ***X'***, we define $N_k(X')$ as the set of the k nearest neighbors of ***X'*** in ***D*** (neighborhood of ***X'***) in some metric space.

We then determine ***Y'*** for ***X'*** as follows:

$$Y'(X') = \frac{1}{k} \sum_{x_i \in N_k(X')} Y_i$$ Regression

$$Y'(X') = Most\_Frequent\_Label(\{Y_i | X_i \in N_k(X')\})$$ Classification

kNN - Advantages

- **Non-parametric** - No assumptions need to be made about the function *f* that maps features onto labels
  - Very powerful for estimating any arbitrary function, but extreme flexibility risks overfitting

- **Instance-based learning** - There is not really a training phase! Target prediction is done locally -- there is effectively no model, just all of the training data stored in memory (cheap timewise, more expensive spacewise)
  - The “supervision” here has to do with using neighboring samples

Determining a Neighborhood

- A *neighborhood* is a set of examples that are *close* to the given instance
  - To find the *k* closest samples we need some metric that defines distance between every two points!

- **Distance metric** - Given two points *a* and *b*, a distance metric d():
  - d(a,b) ≥0 (non-negativity)
  - d(a,b) = 0 if and only if a = b
  - d(a,b) = d(b,a) (symmetry)
  - d(a,c) ≤ d(a,b) + d(b,c) (triangle inequality)

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

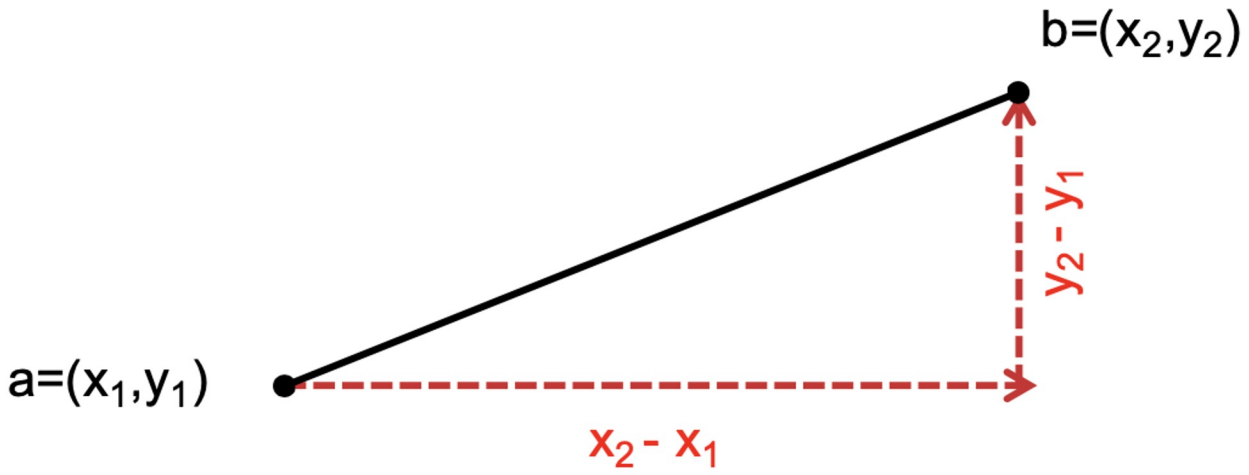
Euclidean Distance

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:



k-NN in Practice

User	Y	X1	X2	X3
11	?	0.57	0.43	0.95

K = 3
(determined in advance)

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

https://github.com/briandalessandro/DataScienceCourse/blob/master/ipython/Lectures/2020%20Intro%20to%20DS%20Module%2027%20-%20kNN.pdf

k-NN in Practice

User	Y	X1	X2	X3
11	?	0.57	0.43	0.95

Y = 0 (most frequent label in the neighborhood)

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

k-NN - Challenges

- k-NN gets more accurate as the size of the training dataset increases
  - However, as this size increases the search cost for the k closest neighbors also grows
    - O(|size of training data|) searches if using brute force
    - Best to use approximate solutions that are faster in practice!
    - Great video: https://www.youtube.com/watch?v=dZrGXYty3qc&ab_channel=caltech

Scikit-Learn algorithms for k-NN search

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'*

Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

k-NN - Challenges

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.

Let dist_{max} and dist_{min} be the farthest and closest point to the center. As *d* increases the ratio of the difference between max and min to the min tends to 0. This means all points are equally far apart, leaving no nearest neighbors to learn from.

As *d* grows, the notion of "nearest" becomes less meaningful...

https://www.youtube.com/watch?v=dZrGXYty3qc&ab_channel=caltech

$$\lim_{d \to \infty} \frac{\text{dist}_{max} - \text{dist}_{min}}{\text{dist}_{min}} \to 0$$

k-NN - Choosing k