CSC 580 Principles of Machine Learning

03 Geometry & Nearest Neighbors

Jason Pacheco

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Chicheng Zhang & Kwang-Sung Jun

*slides credit: Some material from Enfa Rose George CSC 380 Fall 2021

Motivation

Example Given student course survey data, predict whether Alice likes Algorithms course

Idea Find a student ``similar'' to Alice & has taken Algorithm course before, say Jeremy

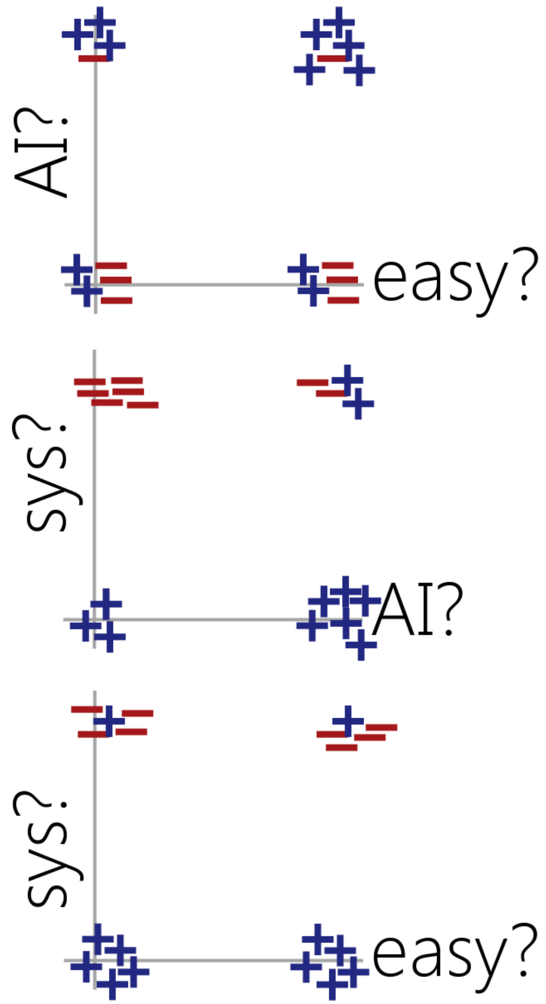
- If Jeremy likes Algorithms, then Alice is also likely to have the same preference.
- Or even better, find *several* similar students

- Prediction = mapping inputs to outputs
- Inputs = *features* that can be viewed as points in some space (possibly high-dimensional)
- “Similarity” = “distance” in feature space
- Suggests a **geometric** view of data

Example: Course Recommendation

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Features



ML begins by mapping data to feature vectors

Represented as points in 5-dimensional space for this example

That's too many dimensions to plot…so we look at 2D projections…

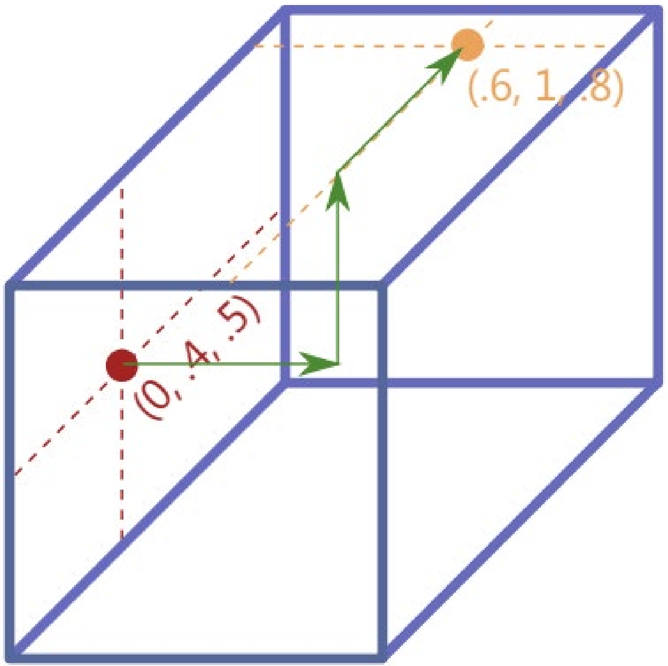
Measuring Nearest Neighbors

- Oftentimes convenient to work with feature $x \in \mathrm{R}^d$
- Distances in R^d:

notation $x(f)$: $x = (x(1), \dots, x(d))$

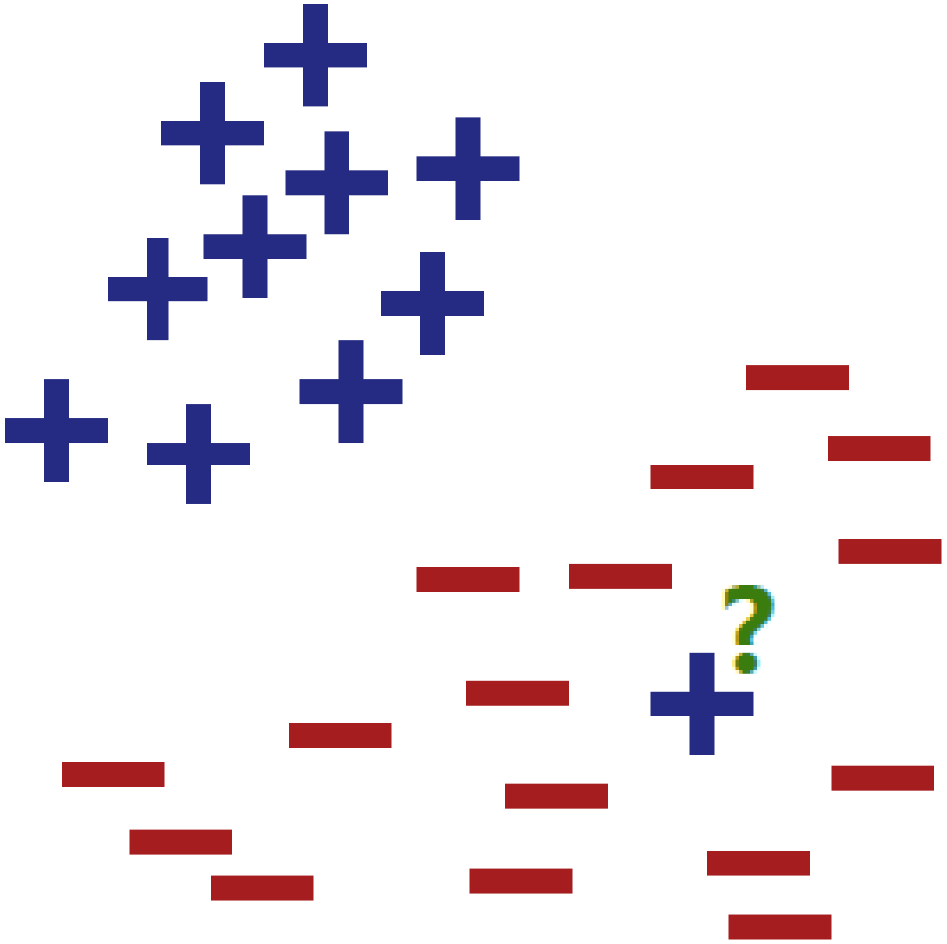
  - Euclidean distance $d_2(x, x') = \sqrt{\sum_{f=1}^{d} \left(x(f) - x'(f)\right)^2}$
  - Manhattan distance $d_1(x, x') = \sum_{f=1}^{d} |x(f) - x'(f)|$

  - If we shift a feature, would the distance change?
  - What about scaling a feature?



- How to extract features as **real values**?
  - Boolean features: {Y, N} -> {0,1}
  - Categorical features: {Red, Blue, Green, Black}
    - Convert to {1, 2, 3, 4}?
    - Better one-hot encoding: (1,0,0,0), .., (0,0,0,1) (IsRed?/isGreen?/isBlue?/IsBlack?)

Nearest Neighbor Classification



Query point ? Will be classified as +
but should be -

Inductive Bias Query points belong to same class as closes exemplar seen in training data

Question How can we reduce inductive bias?

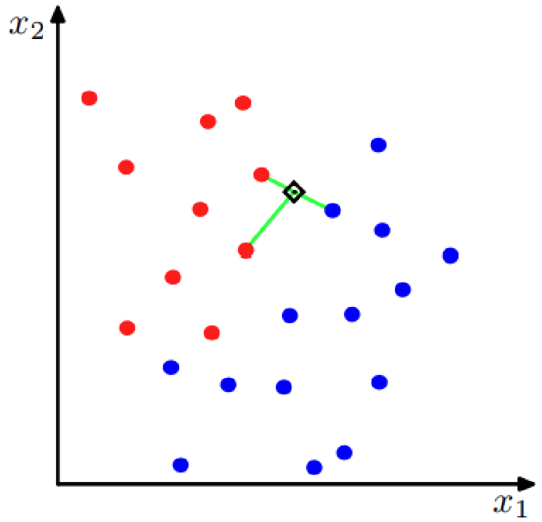
k-nearest neighbors (k-NN): main concept

Training set: $S = \{ (x_1, y_1), \ldots, (x_m, y_m) \}$

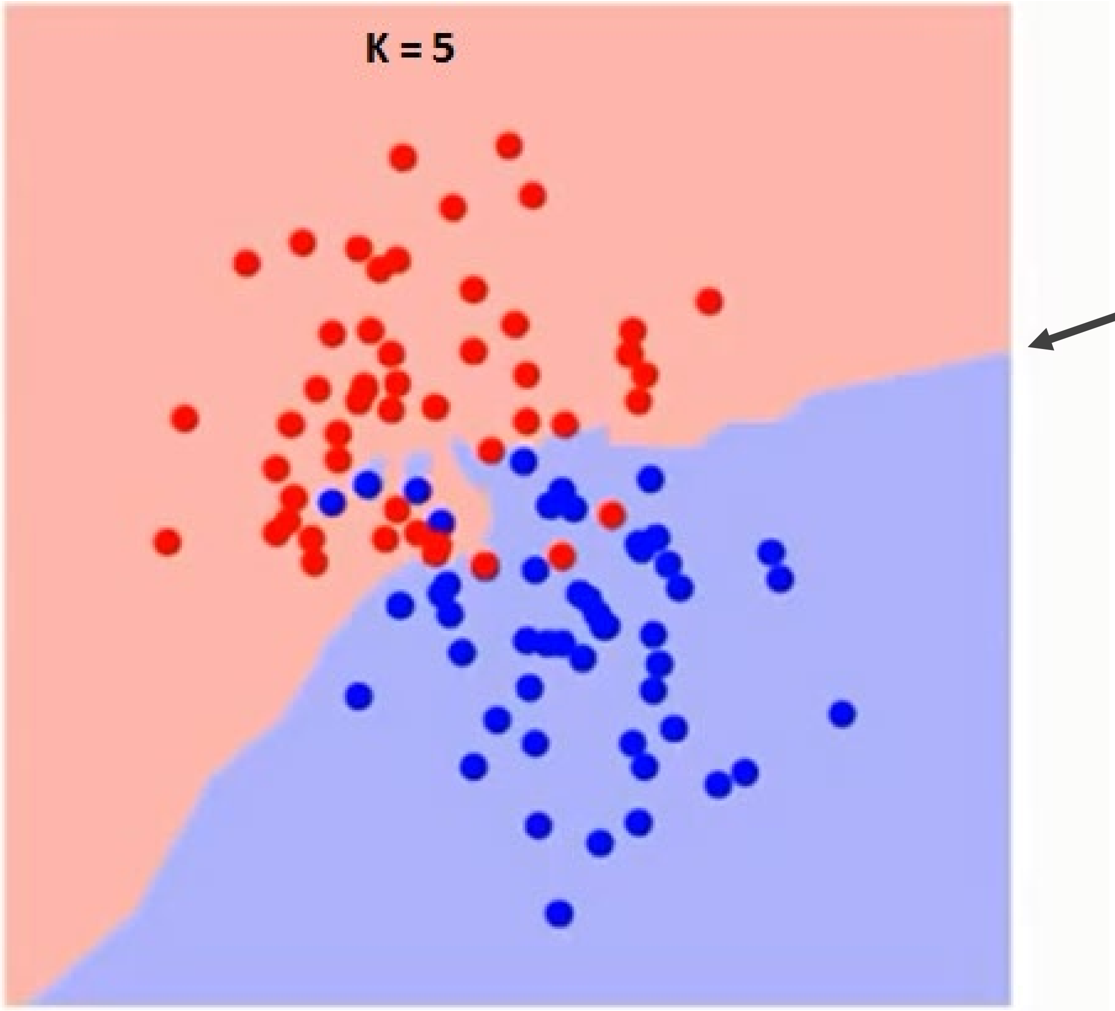
<u>**Inductive bias**</u>: given test example x, its label should resemble the labels of **nearby points**

Function

- input: x
- find the k nearest points to x from S; call their indices $N(x)$
- output: the majority vote of $\{y_i : i \in N(x)\}$
  - For regression, the average.



k-NN classification example



decision boundary

k-NN classification: pseudocode

- Training is trivial: store the training set
- Test:

Algorithm 3 KNN-PREDICT$(\mathbf{D}, K, \hat{x})$

list

$S \leftarrow [\,]$

for $n = 1$ **to** N **do**

append to list

    $S \leftarrow S \oplus \langle \mathrm{d}(x_n, \hat{x}), n \rangle$ // store distance to training example n

end for

sort in first coordinate

$S \leftarrow$ SORT(S) // put lowest-distance objects first

$\hat{y} \leftarrow 0$

for $k = 1$ **to** K **do**

    $\langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point

    $\hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point

end for

Majority vote of $\{y_i : i \in N(x)\}$

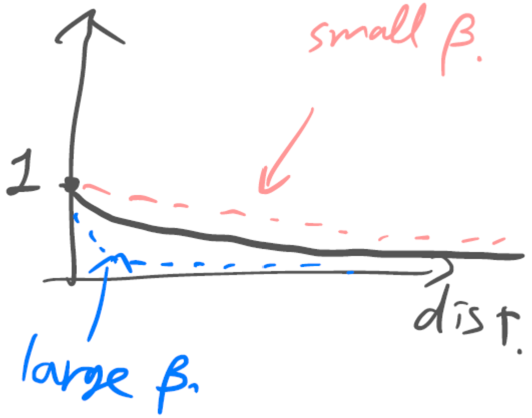
return SIGN$(\hat{y})$ // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

- Time complexity (assuming distance calculation takes $O(d)$ time)
  - $O(m\,d + m \log m + k) = O\big(m(d + \log m)\big)$
- Faster nearest neighbor search: k-d trees, locality sensitive hashing

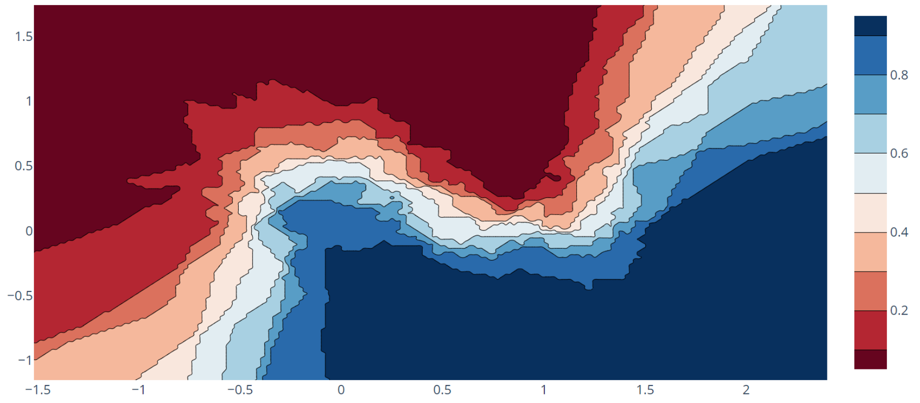
Variations

- Classification
  - Recall the majority vote rule: $\hat{y} = \arg\max_{y\in\{1,\ldots,C\}} \sum_{i\in N(x)} 1\{y_i = y\}$
  - Soft weighting nearest neighbors: $\hat{y} = \arg\max_{y\in\{1,\ldots,C\}} \sum_{i=1}^{m} w_i \, 1\{y_i = y\}$,

    where $w_i \propto \exp(-\beta\, d(x, x_i))$, or $\propto \frac{1}{1+d(x,x_i)^\beta}$



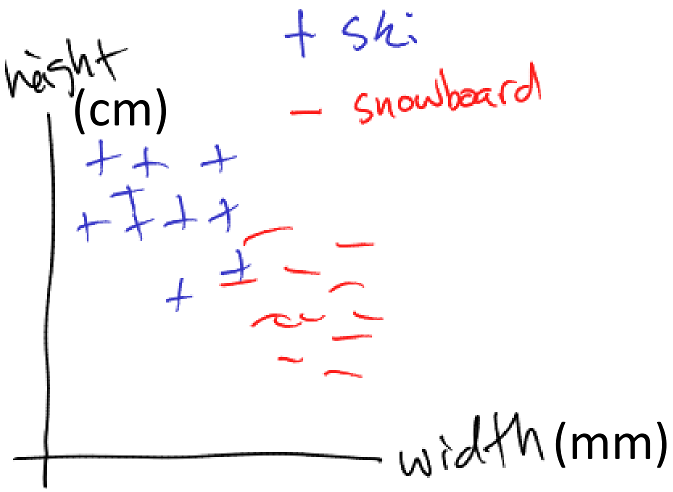
- Class probability estimates
  - $\hat{P}(Y = y \mid x) = \frac{1}{k}\sum_{i\in N(x)} 1\{y_i = y\}$
  - Useful for "classification with rejection"

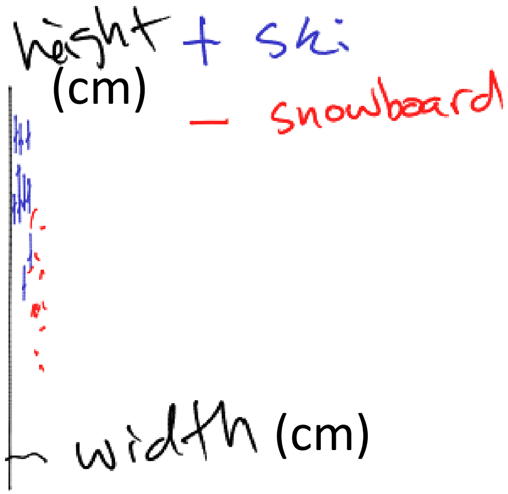


Feature issue 1: scaling

- Features having different scale can be problematic.

- Ex: ski vs. snowboard classification

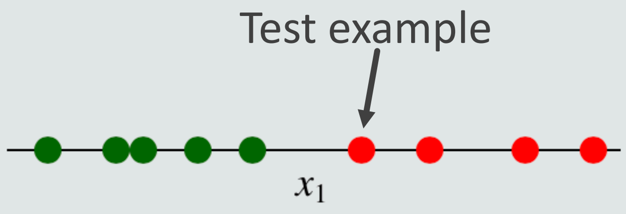




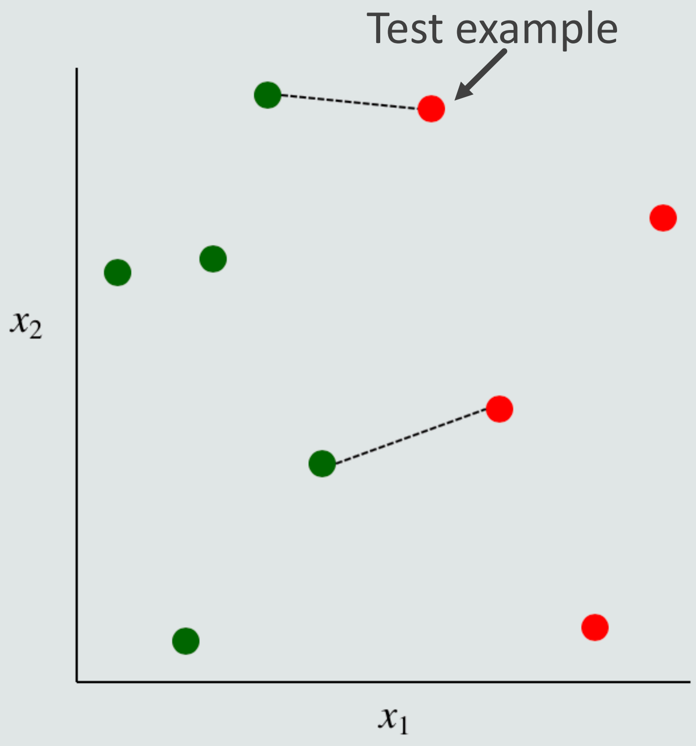
- Solution: feature standardization (later in the course)

Feature issue 2: irrelevant features

here's a case in which there is one relevant feature x_1 and a 1-NN rule classifies each instance correctly



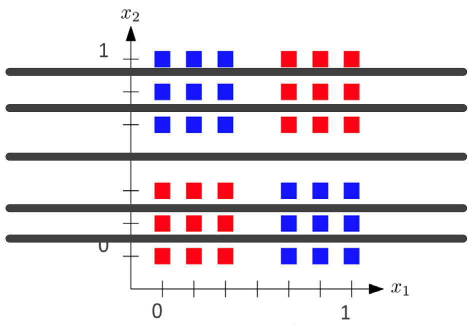
consider the effect of an irrelevant feature x_2 on distances and nearest neighbors



- Recall: how did we deal with these in decision trees?
- Solution: feature selection (later in the course)

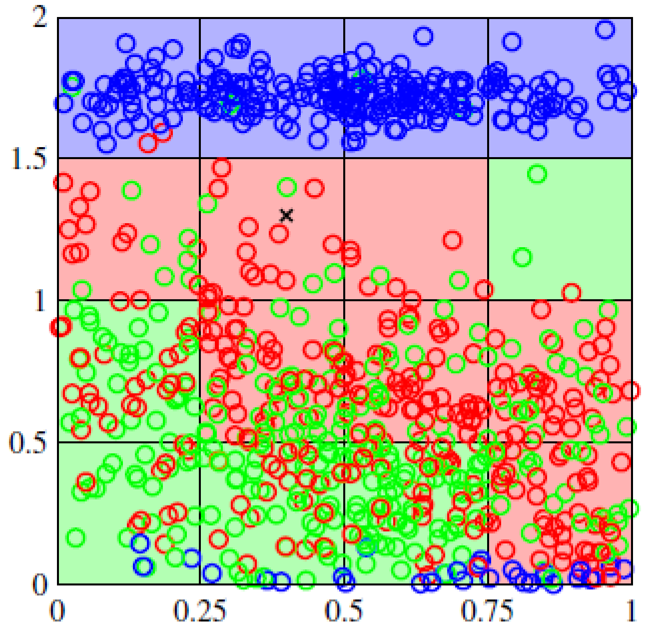
Comparison (feature $x \in \mathrm{R}^d$)

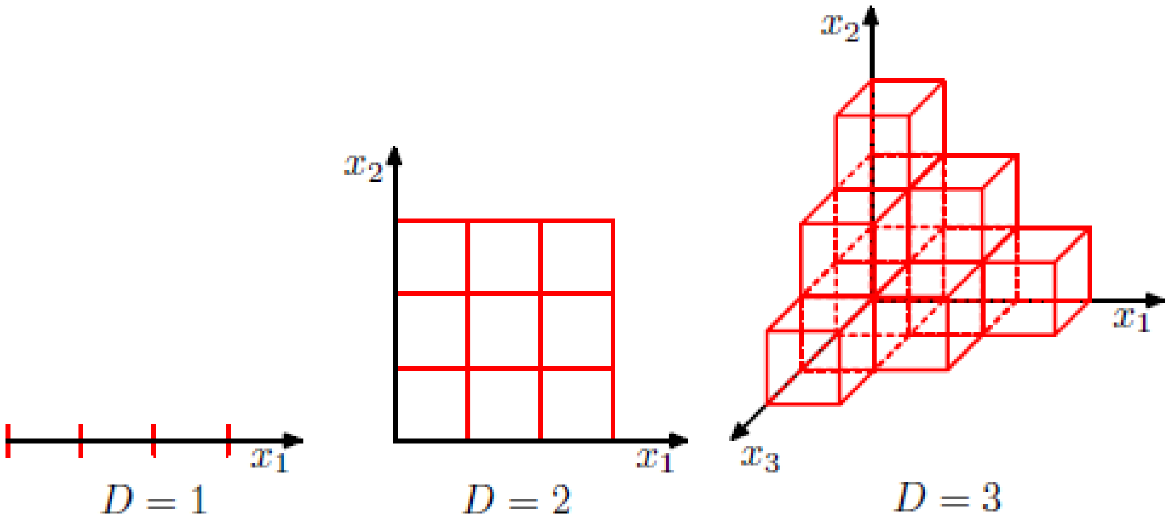
	Decision Tree	k-NN
• Interpretability	High	Medium (example-based)
• Sensitivity to irrelevant features	Low	High
• training time	$O(\#\text{nodes} \cdot d \cdot (m + m \log m))$ $\leq \tilde{O}(d\, m^2)$ (when no two points have the same feature)	0
• test time per example	$O(\text{depth})$	$O\big(m(d + \log m)\big)$ *Can reduce this with K-d trees or locality sensitive hashing*



Curse of Dimensionality - Computation

Divide space into regular intervals to avoid computing distances for each data

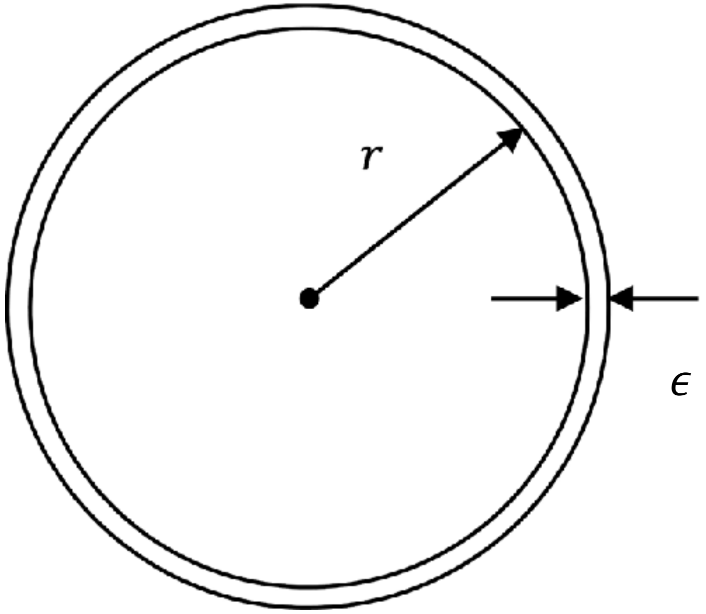


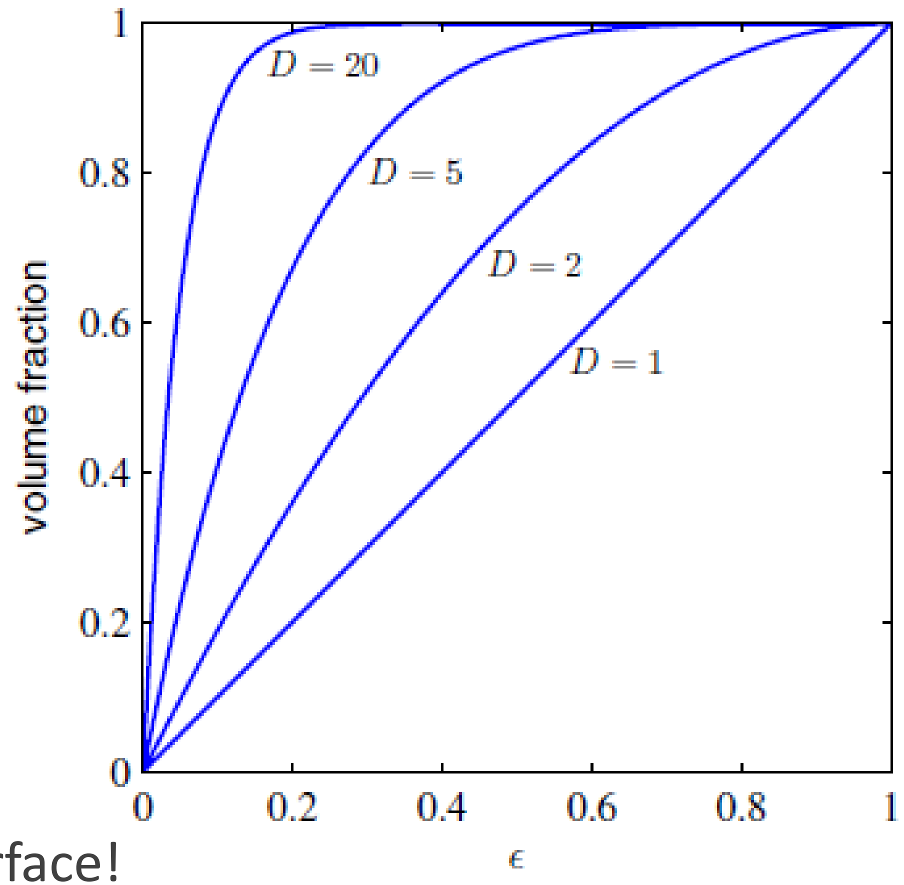


Number of required cells grows exponentially in dimension!

Curse of Dimensionality – Distance Weirdness

- Consider *D*-dimensional hypersphere of radius *r=1*
- What is the fraction of volume within shell of width ϵ?

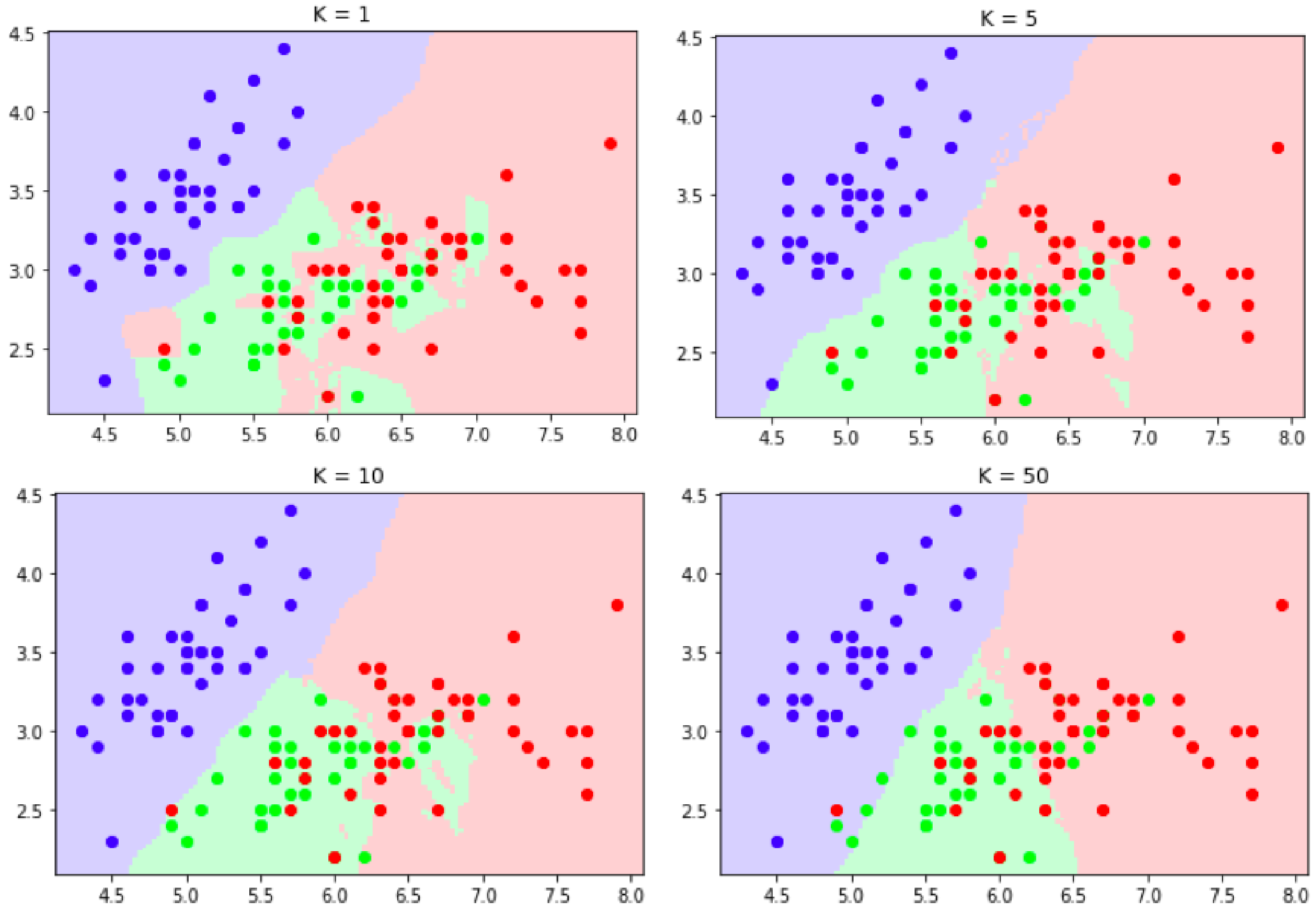




- Total volume of hypersphere concentrates onto shell at the surface!
- Distances go to zero!

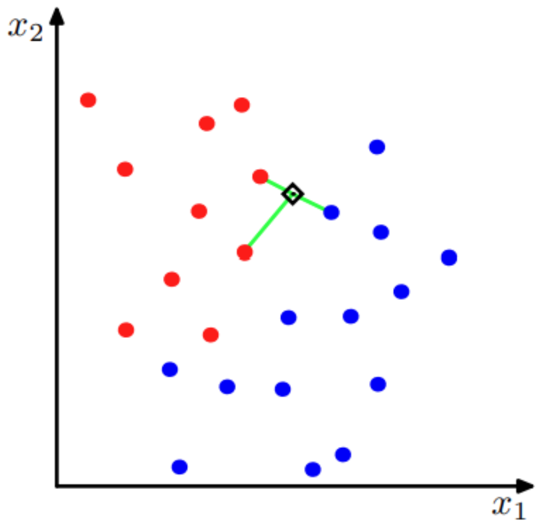
Intuition about lower dimensions doesn't extend to high dimensions

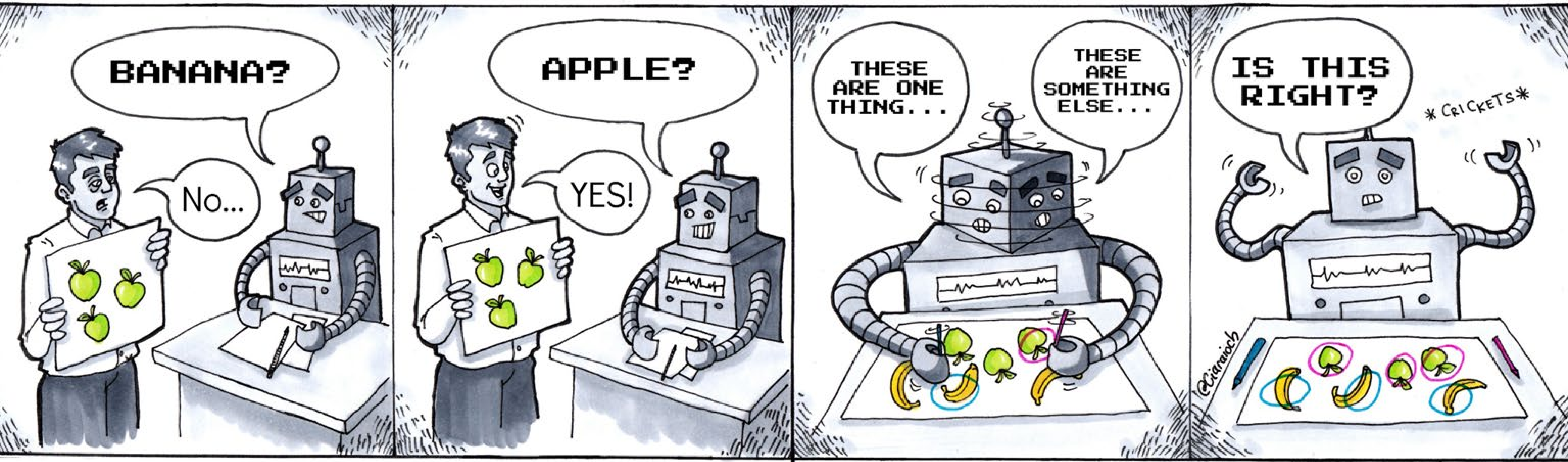
Hyperparameter tuning in k-NN



Hyperparameter tuning in k-NN

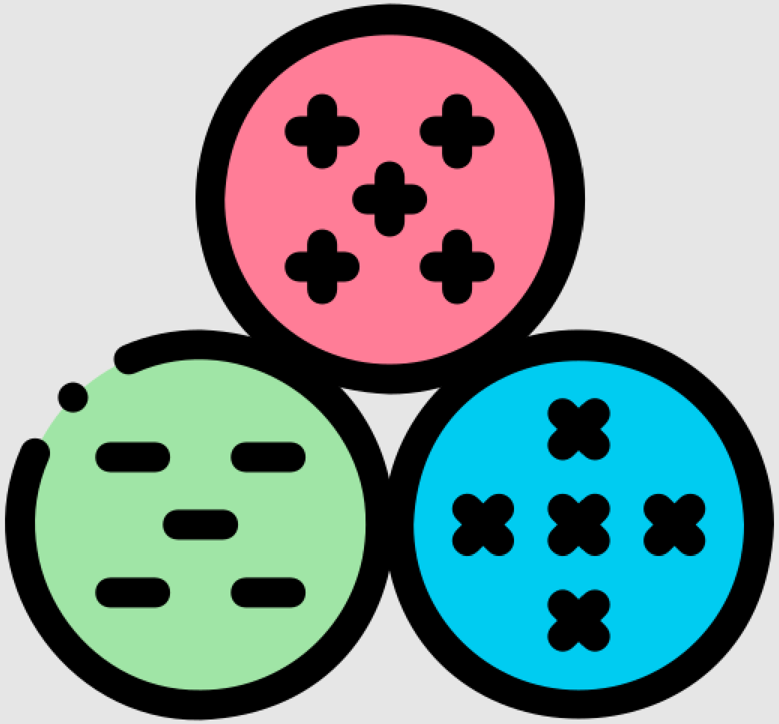
- Hyperparameter: k
- $k = 1$:
  - Training error = 0, overfitting
- $k = N$:
  - Output a constant (majority class) prediction, underfitting
- Can use hold-out validation to choose k





Supervised Learning | **Unsupervised Learning**

K Means Intuition



Icons are used from FlatIcon.com

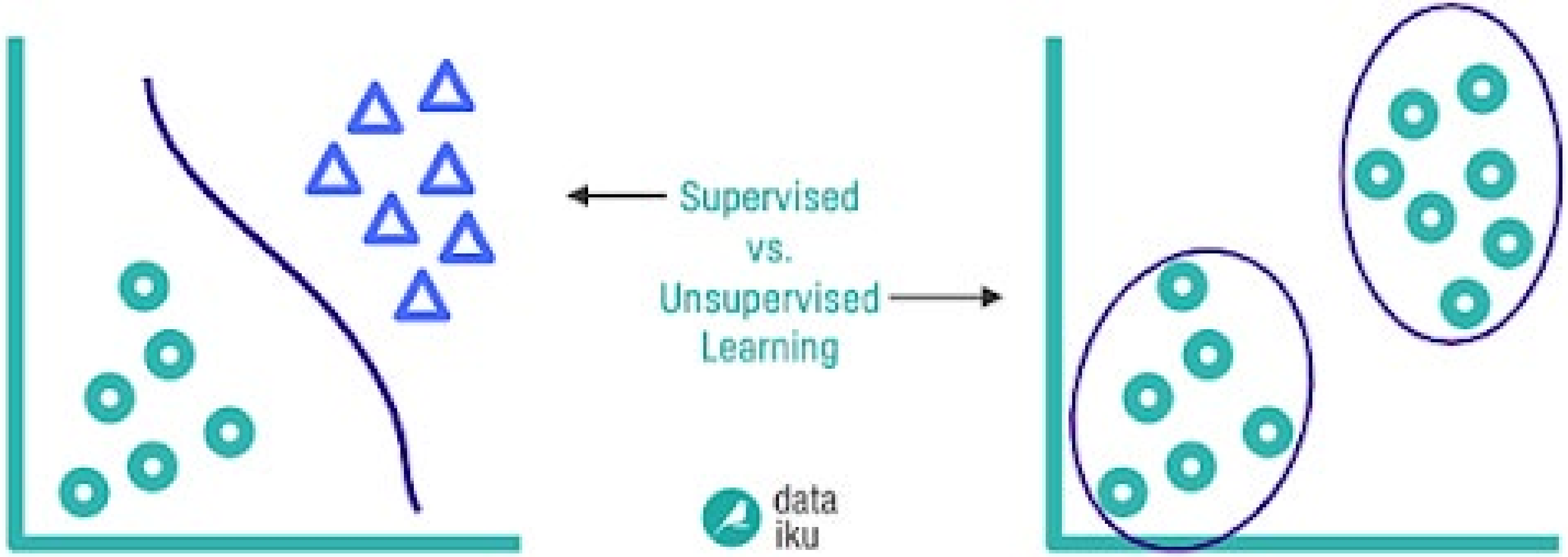
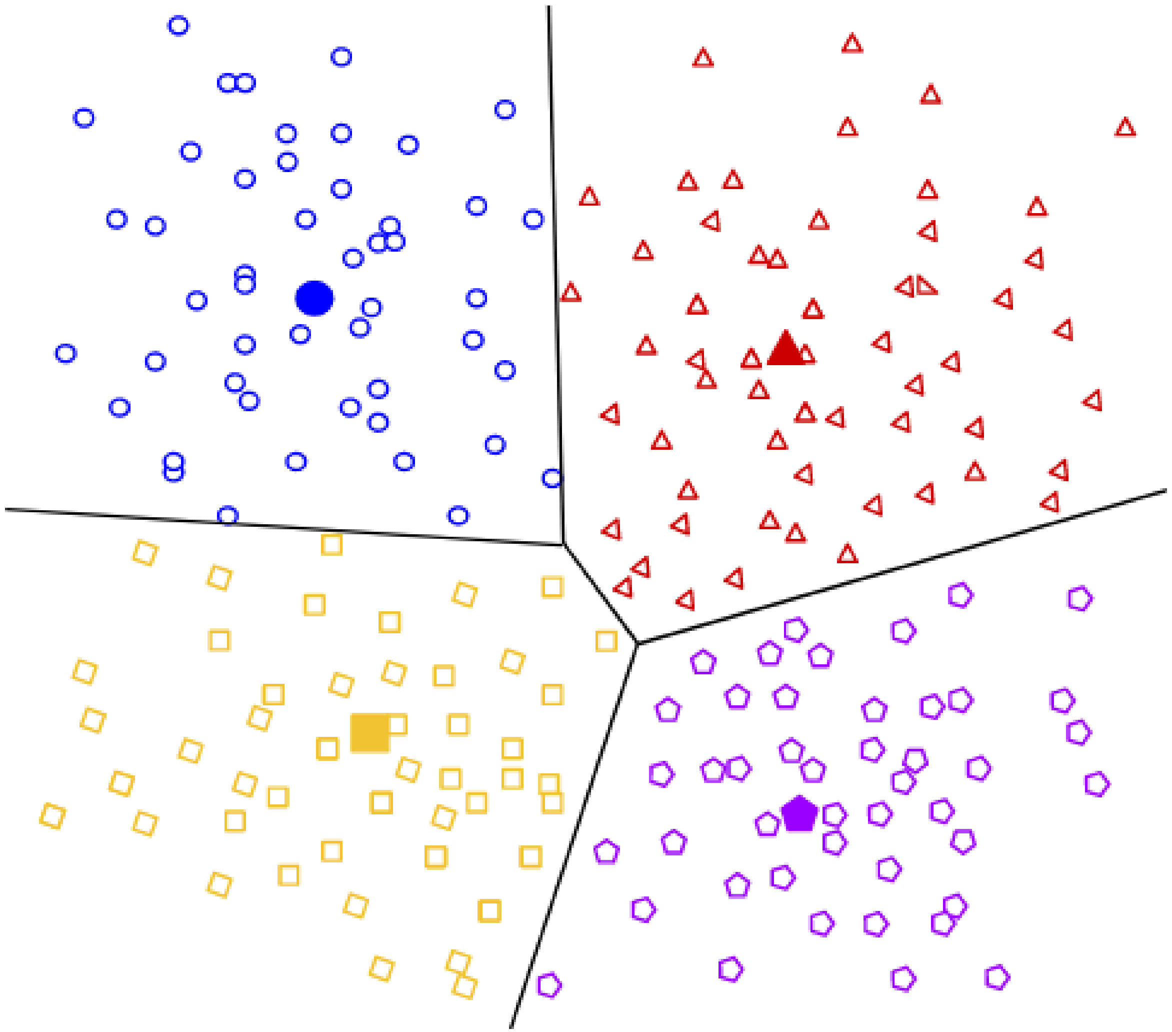


Image from [Dataiku]

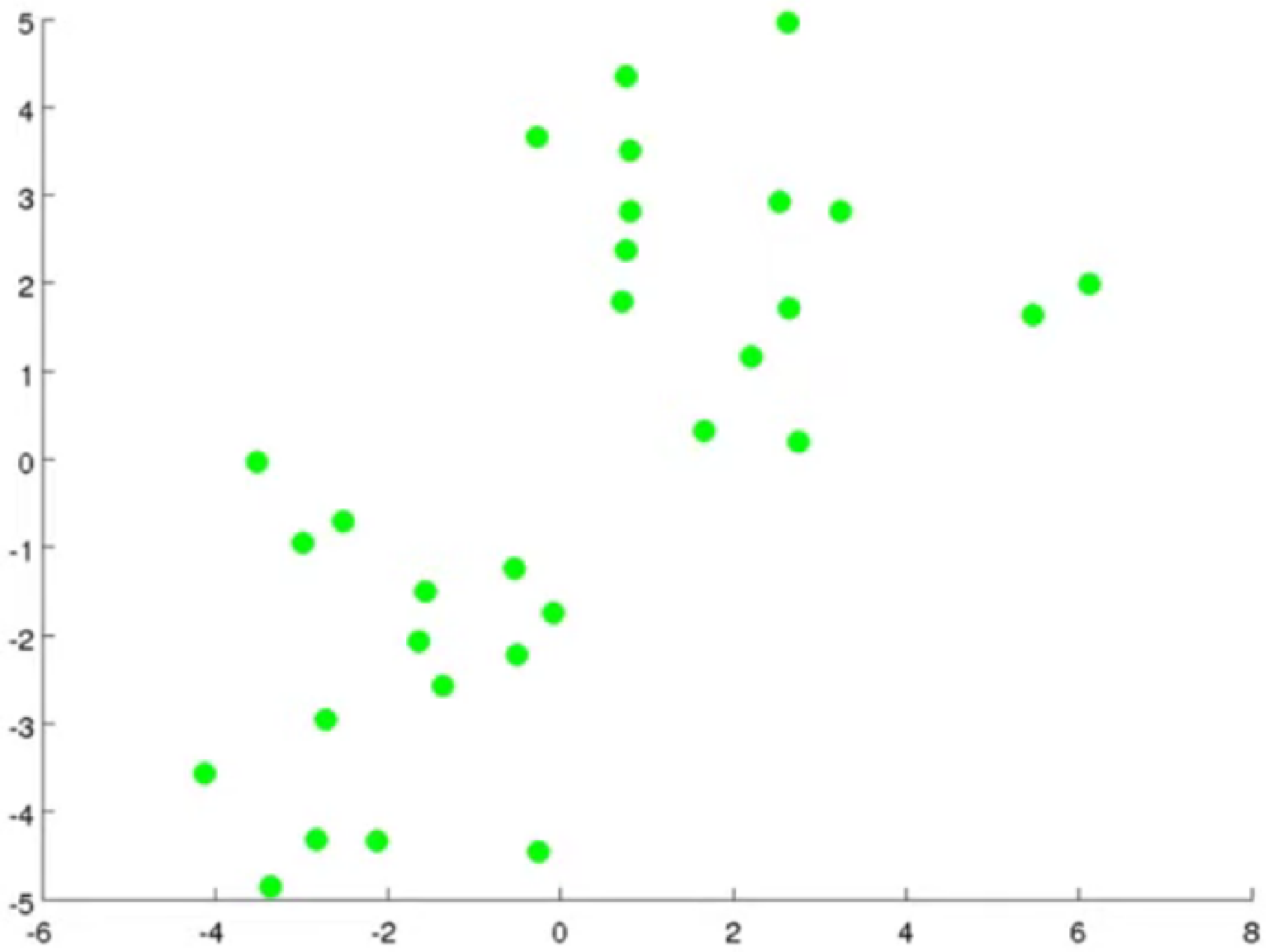
Centroid-based Clustering



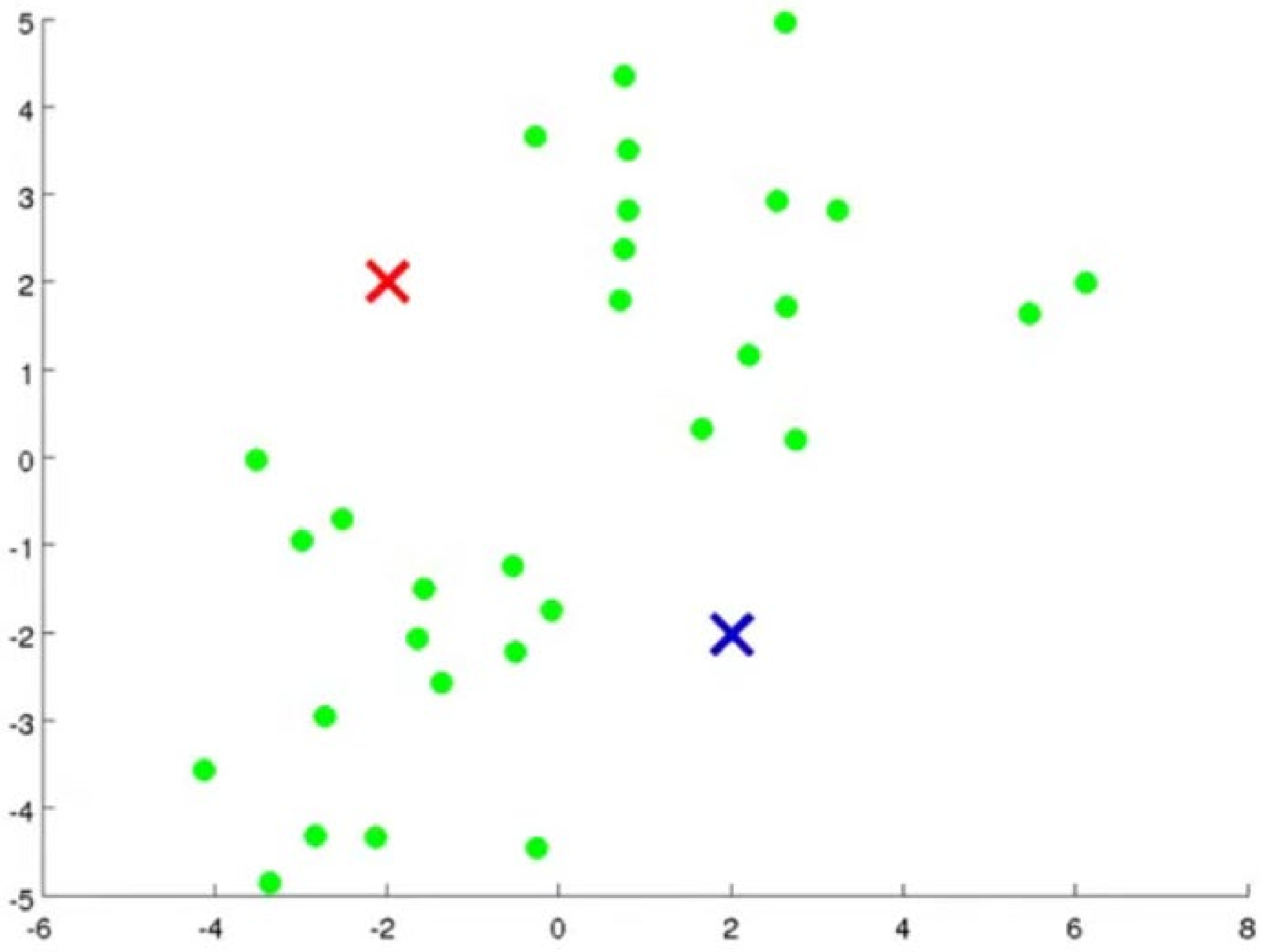
Basic Steps

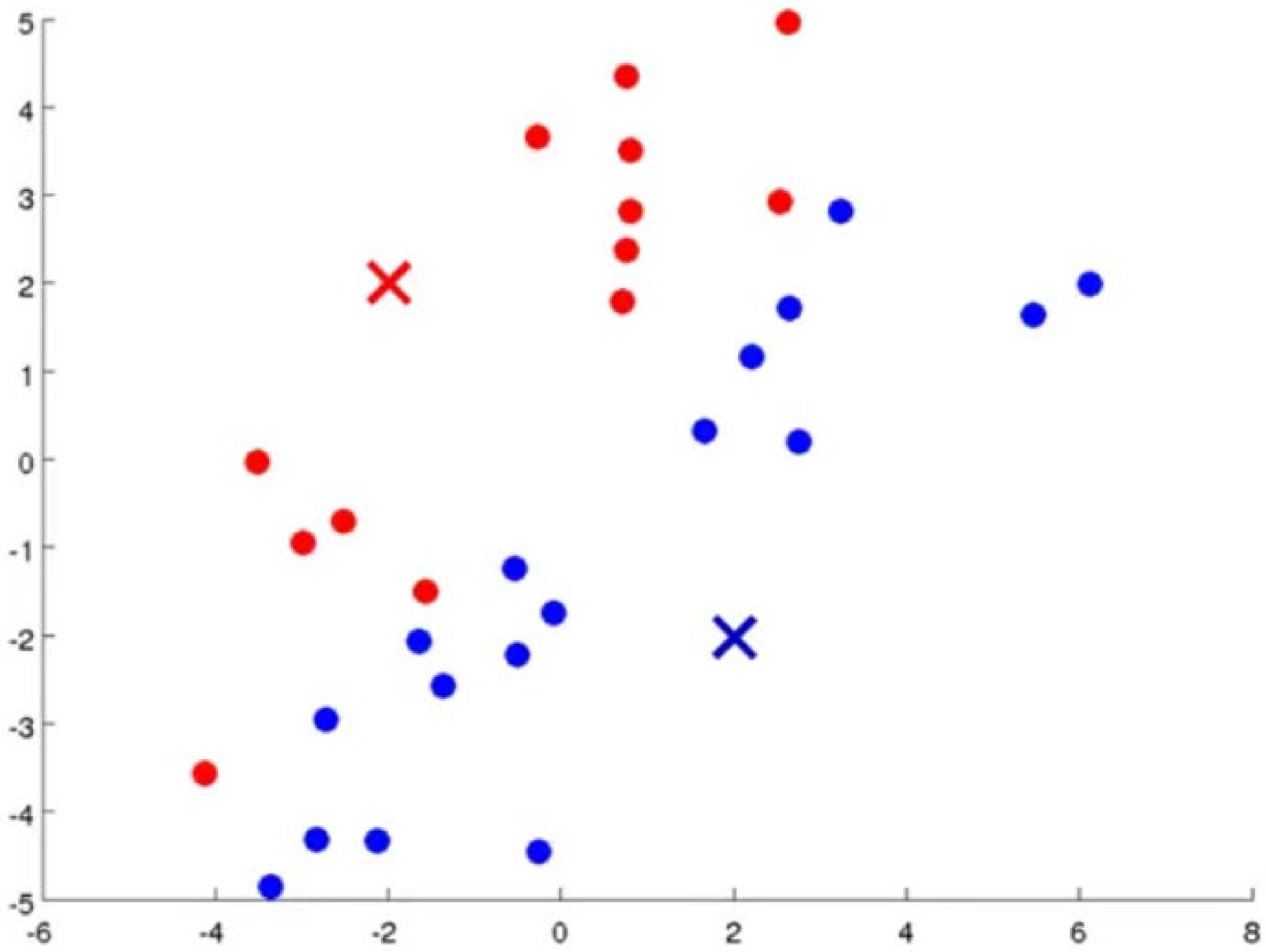
- Assign Cluster Centroids

- Until Convergence :
  - Cluster Assignment Step
  - Re-assigning Centroid Step

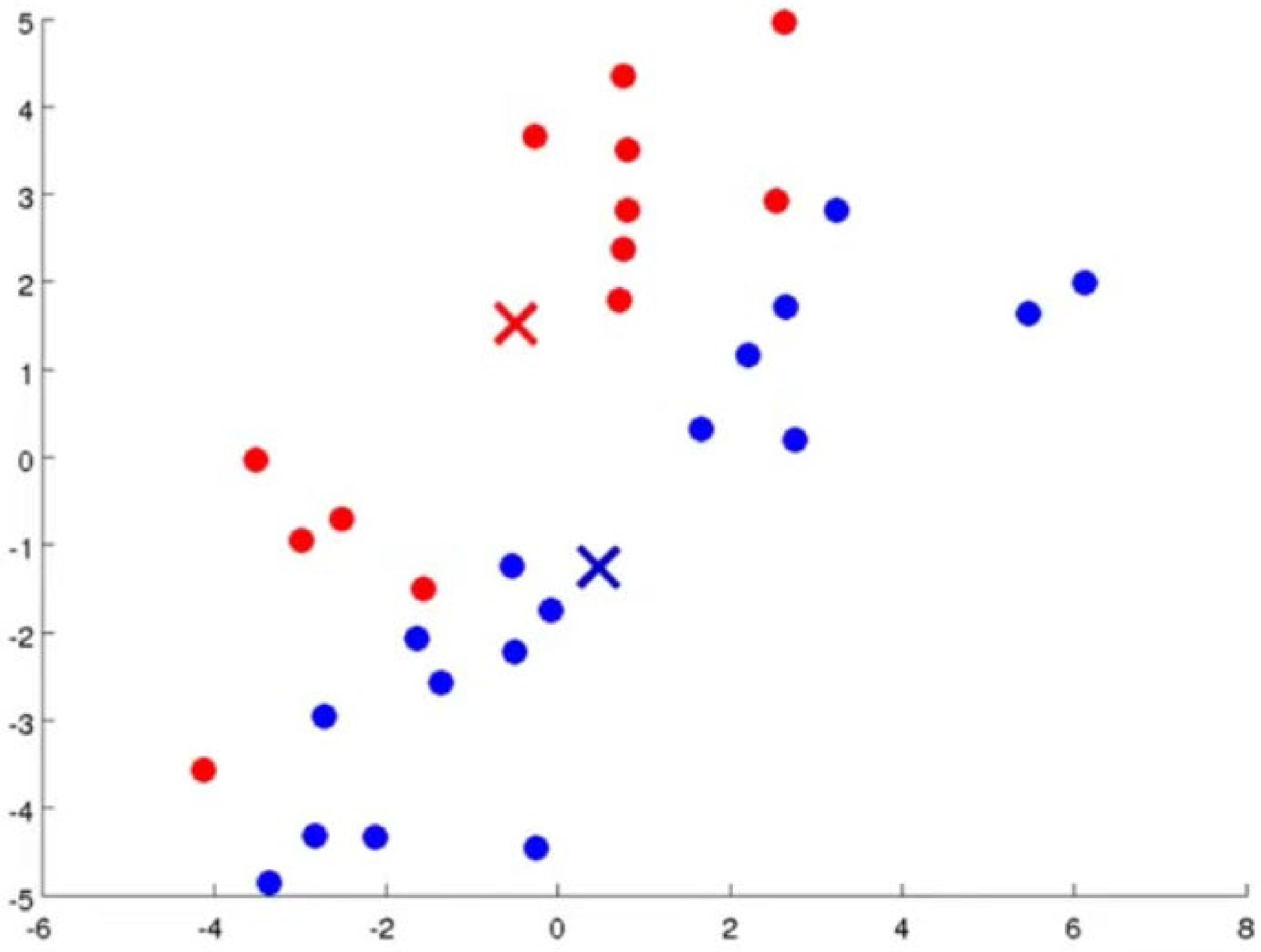


Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2

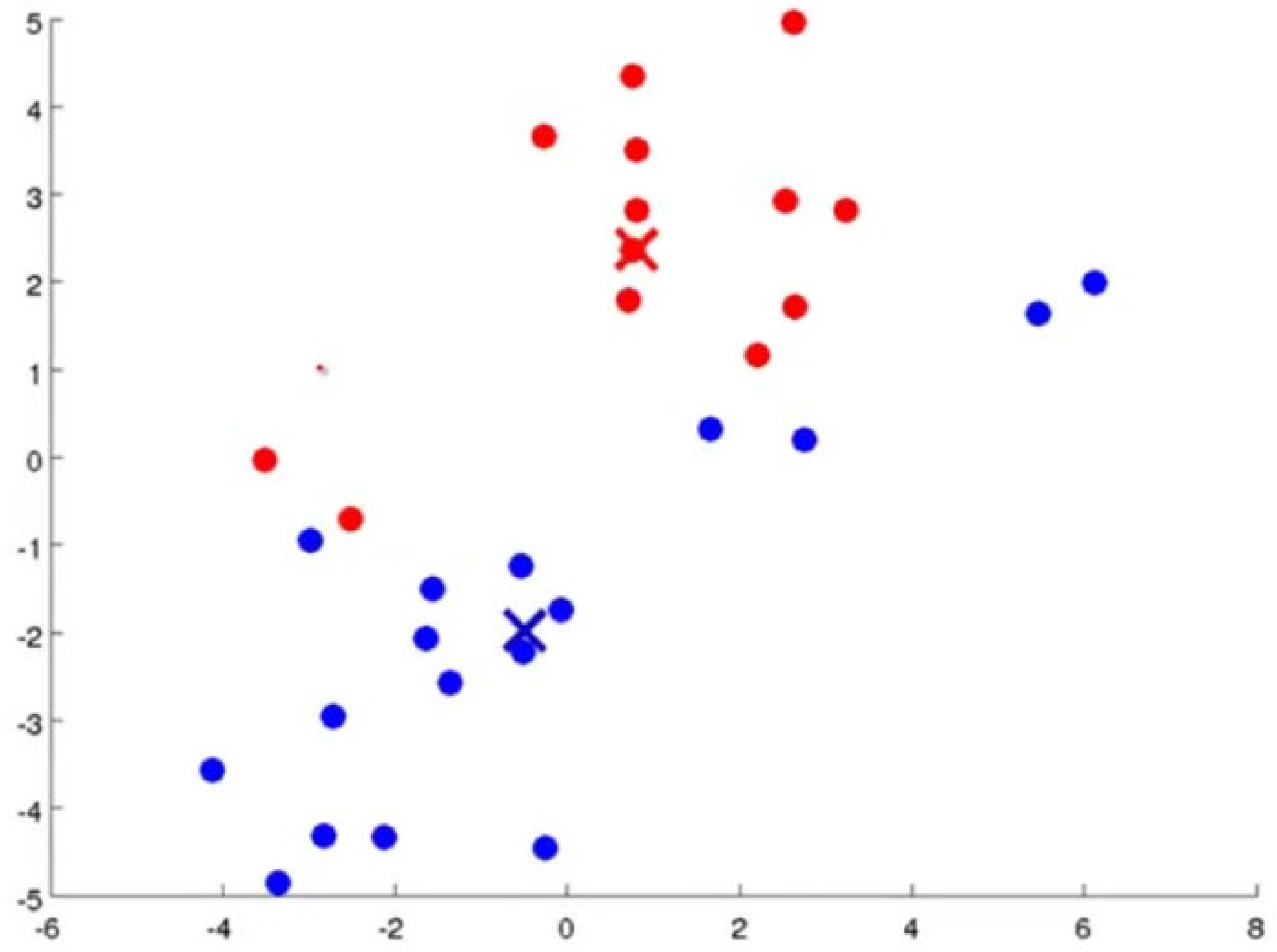

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8



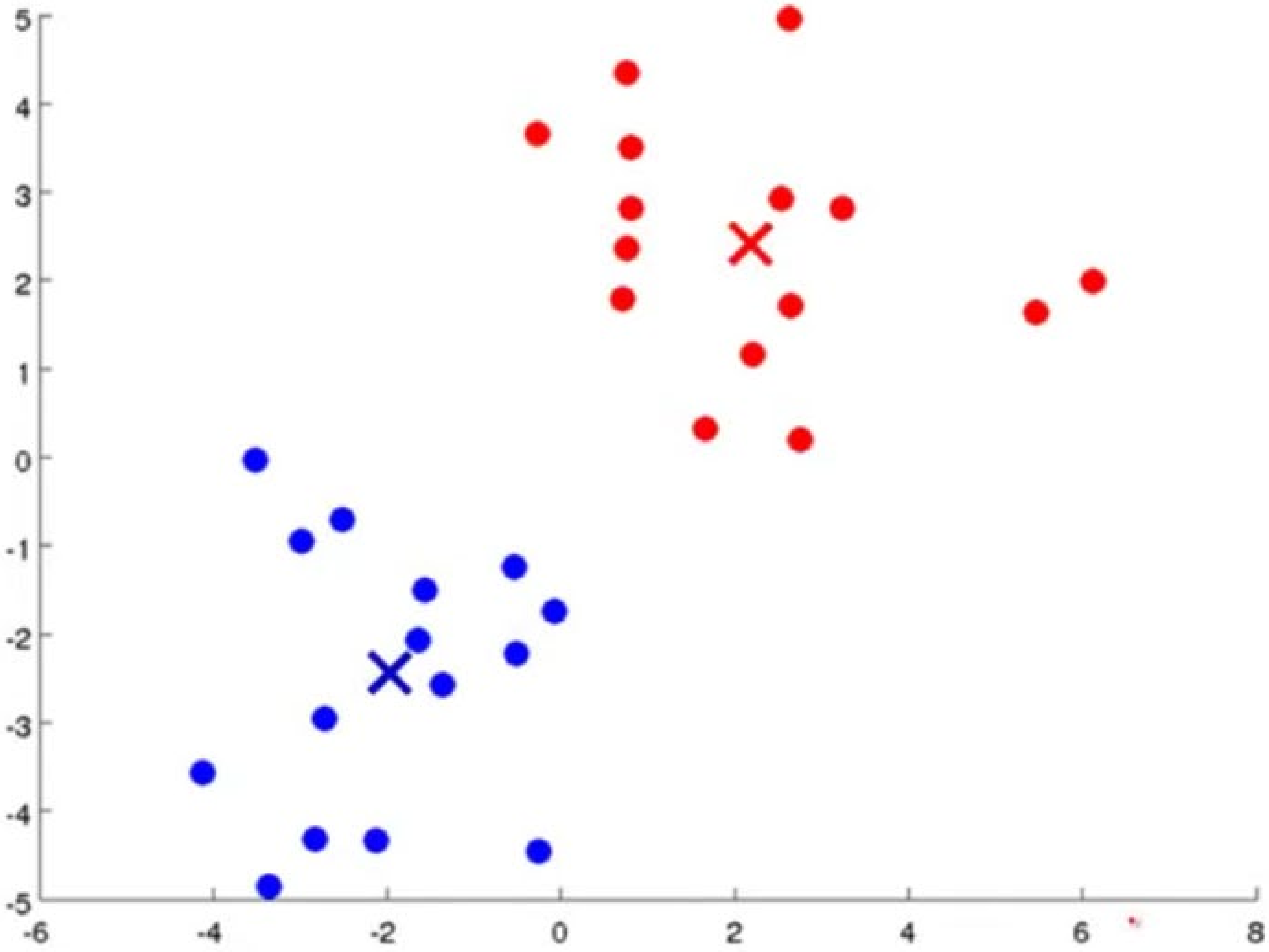
Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2



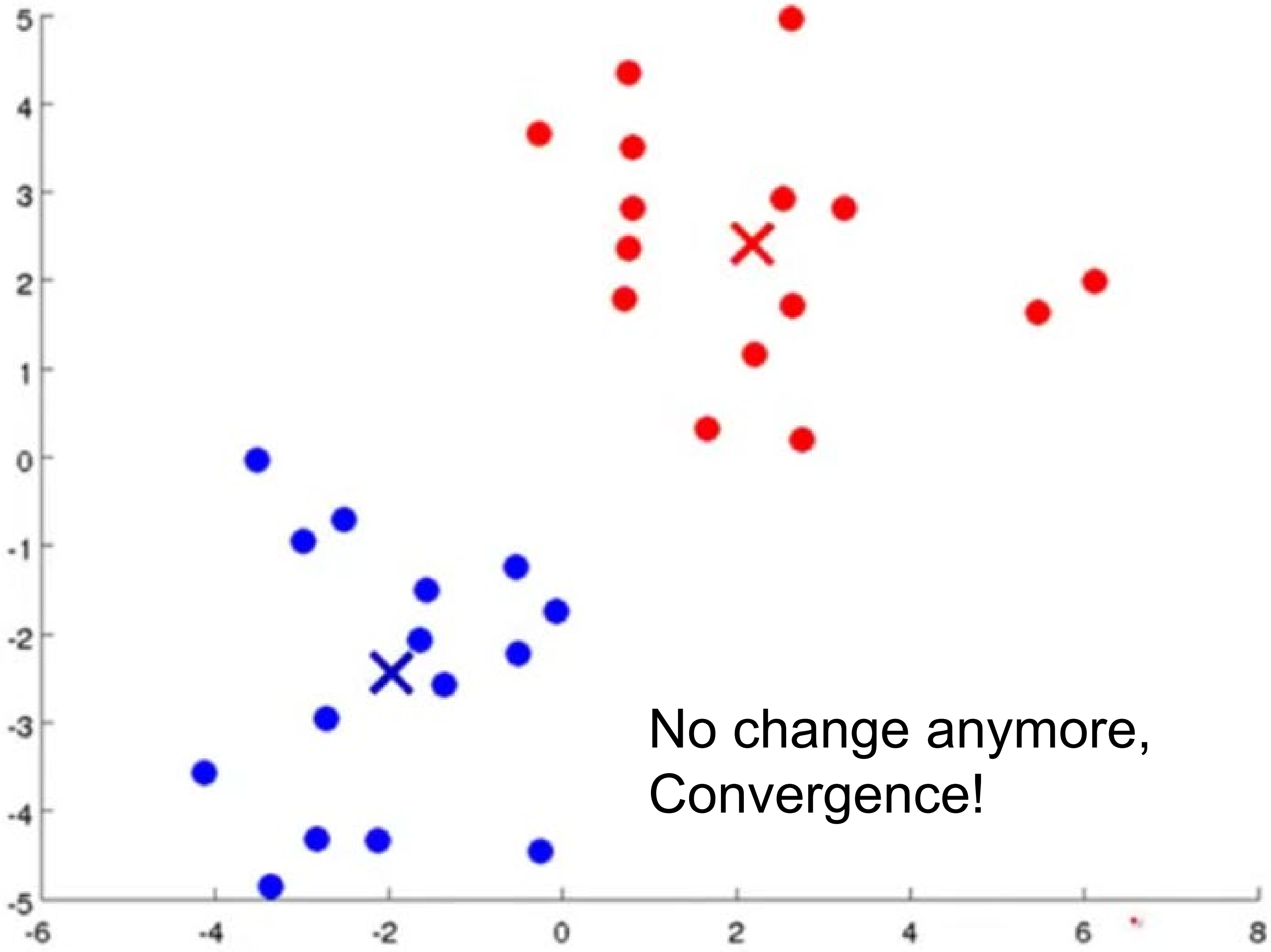
Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2



Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2



Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2

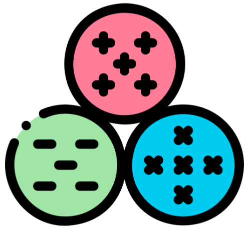


No change anymore,
Convergence!

Slides from Andrew Ng | Machine Learning Coursera Lecture 13.2

Basic Steps

- Assign Cluster Centroids
- Until Convergence :
    - Cluster Assignment Step
    - Re-assigning Centroid Step



Iterating until Convergence

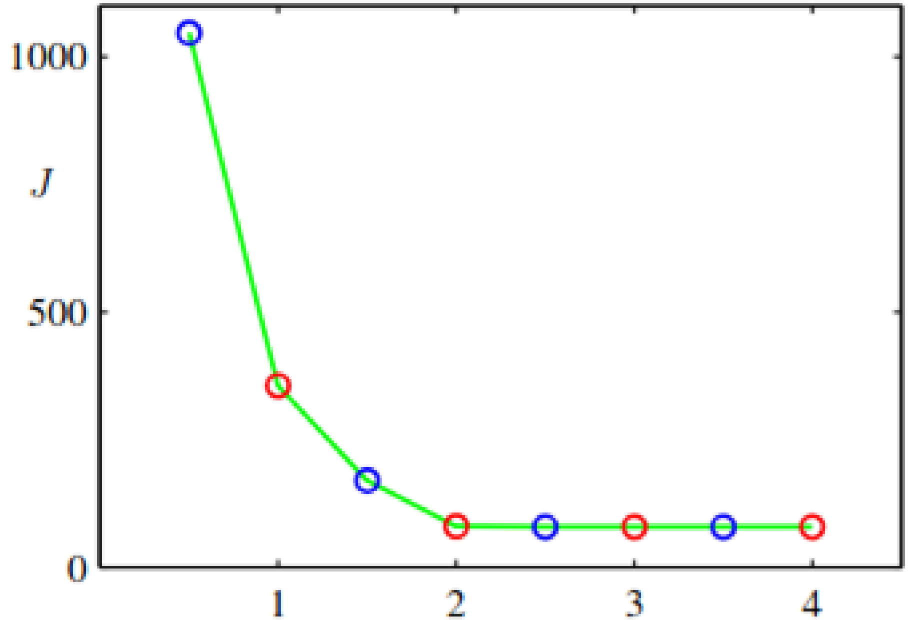
Animation from [Kaggle]()

Promise of Convergence

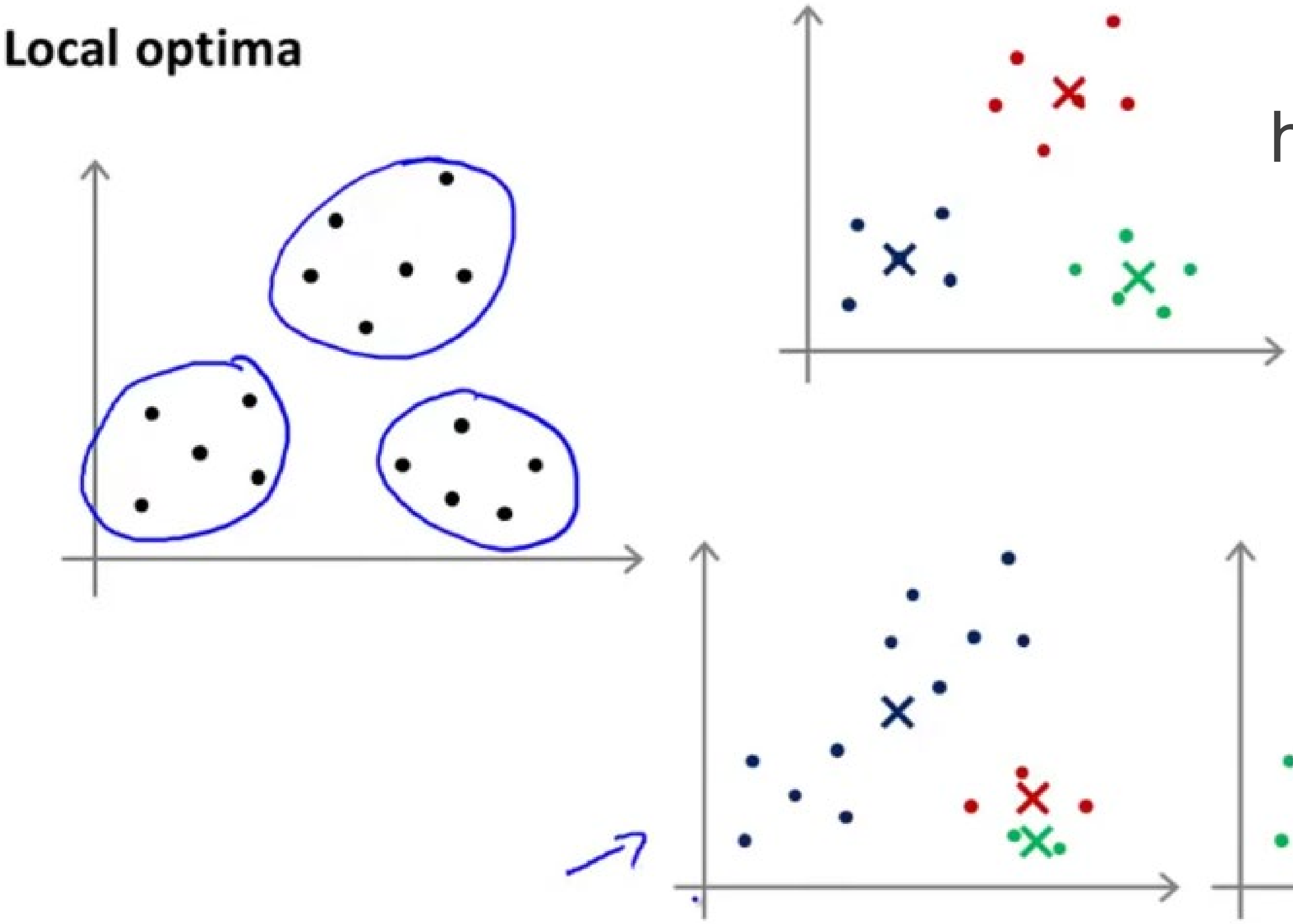
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

But, may converge to a local rather than global minimum of J.

Solution quality highly dependent on initialization!



Plot of the cost function J given by (9.1) after each E step (blue points) and M step (red points) of the K-means algorithm for the example shown in Figure 9.1.



Solution quality highly dependent on initialization!

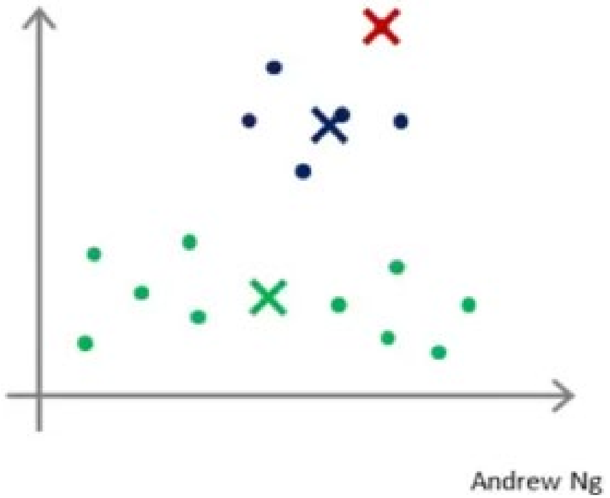


Image from Andrew NG Coursera Machine Learning Course

Next lecture (9/7)

- Linear classification; the Perceptron algorithm

- Assigned reading: CIML Chap. 4