K-nearest neighbors

Yuhua Li (李玉华)

Intelligent and Distributed Computing Laboratory
Huazhong University of Science & Technology

idcliyuhua@hust.edu.cn

2019年04月29日

Table of contents

Table of Contents

Formal definition of k-NN:

Basic idea:

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.

- Assumption:Similar inputs have similar outputs
- Classification rule: For a test input **x**, assign the most common label amongest its k most similar training inputs
- Regression rule: The output is the property value for the object. This value is the average of the values of **k** nearest neighbors.

Formal definition of k-NN:

Formal definition

- Test point: $\mathbf{x}$
- Denote the set of the k nearest neighbors of $\mathbf{x}$ as $S_\mathbf{x}$. Formally $S_\mathbf{x}$ is defined as $S_\mathbf{x} \subseteq D$ s.t. $|S_\mathbf{x}| = k$ and $\forall(\mathbf{x}', y') \in D \backslash S_\mathbf{x}$,

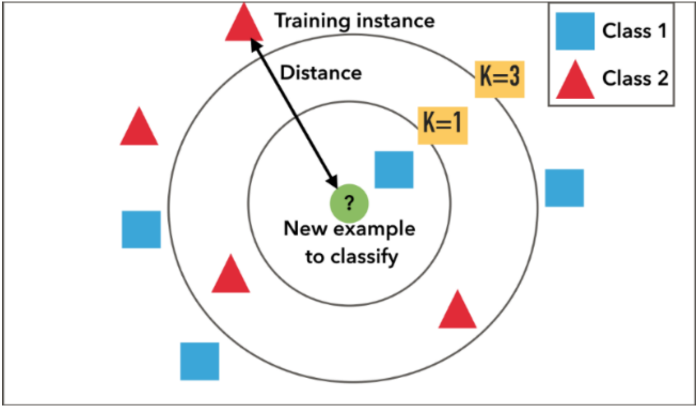
$$\text{dist}(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', y'') \in S_\mathbf{x}} \text{dist}(\mathbf{x}, \mathbf{x}''),$$

(i.e. every point in D but *not* in $S_\mathbf{x}$ is at least as far away from $\mathbf{x}$ as the furthest point in $S_\mathbf{x}$). We can then define the classifier $h()$ as a function returning the most common label in $S_\mathbf{x}$:

$$h(\mathbf{x}) = \text{mode}(\{y'' : (\mathbf{x}'', y'') \in S_\mathbf{x}\}),$$

where mode$(\cdot)$ means to select the label of the highest occurrence.

A binary classification example



☐: Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

Table of Contents

Distance function

The k-nearest neighbor classifier fundamentally relies on a distance metric. The better that metric reflects label similarity, the better the classified will be. The most common choice is the **Minkowski distance**

$$\mathrm{dist}(\mathbf{x}, \mathbf{z}) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p}.$$

Quiz

This distance definition is pretty general and contains many well-known distances as special cases. Can you identify the following candidates?

- $p = 1$
- $p = 2$
- $p \to \infty$

The best K

How to choos a K

The best choice of **k** depends upon the data; generally, larger values of **k** reduces effect of the noise on the classification, but make boundaries between classes less distinct. A good **k** can be selected by various heuristic techniques (see hyperparameter optimization). In binary (two class) classification problems, it is helpful to choose **k** to be an odd number as this avoids tied votes.

Some popular ways of choosing the empirically optimal **k** includes :

- bootstrap method
- cross validation
- Bayes method

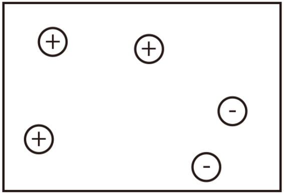
Table of Contents

1-NN classifier

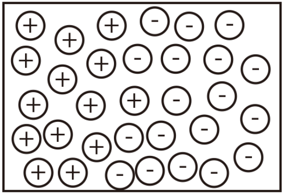
The most intuitive nearest neighbour type classifier is the one nearest neighbour classifier**(k=1)** that assigns a point **x** to the class of its closest neighbour in the feature space.
As the size of training data set approaches infinity, the one nearest neighbour classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data).

1-NN Convergence Proof

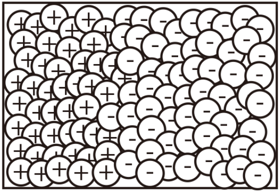
As $n \to \infty$, **the 1-NN error is no more than twice the error of the Bayes Optimal classifier.** (Similar guarantees hold for $k > 1$.)



□: n small



□: n large



□: $n \to \infty$

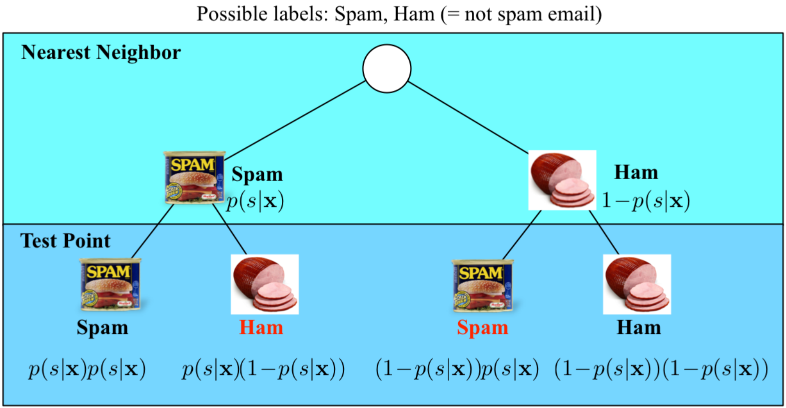
1-NN Convergence Proof

Let $\mathbf{x}_{\mathrm{NN}}$ be the nearest neighbor of our test point $\mathbf{x}_{\mathrm{t}}$. As $n \to \infty$, $\mathrm{dist}(\mathbf{x}_{\mathrm{NN}}, \mathbf{x}_{\mathrm{t}}) \to 0$, i.e. $\mathbf{x}_{\mathrm{NN}} \to \mathbf{x}_t$. (This means the nearest neighbor is identical to $\mathbf{x}_{\mathrm{t}}$.)
You return the label of $\mathbf{x}_{\mathrm{NN}}$. What is the probability that this is not the label of $\mathbf{x}_{\mathrm{t}}$? (This is the probability of drawing two different label of $\mathbf{x}$)

$$\begin{aligned}\epsilon_{NN} &= \mathrm{P}(y^*|\mathbf{x}_t)(1 - \mathrm{P}(y^*|\mathbf{x}_{NN})) + \mathrm{P}(y^*|\mathbf{x}_{NN})(1 - \mathrm{P}(y^*|\mathbf{x}_t)) \\ &\leq (1 - \mathrm{P}(y^*|\mathbf{x}_{NN})) + (1 - \mathrm{P}(y^*|\mathbf{x}_t)) = 2(1 - \mathrm{P}(y^*|\mathbf{x}_t) = 2\epsilon_{\mathrm{BayesOpt}},\end{aligned}$$

where the inequality follows from $\mathrm{P}(y^*|\mathbf{x}_t) \leq 1$ and $\mathrm{P}(y^*|\mathbf{x}_{NN}) \leq 1$. We also used that $\mathrm{P}(y^*|\mathbf{x}_t) = \mathrm{P}(y^*|\mathbf{x}_{NN})$.

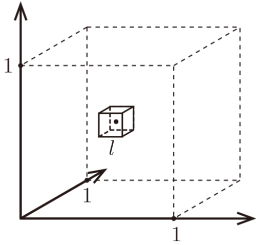
1-NN Convergence Proof



□: In the limit case, the test point and its nearest neighbor are identical. There are exactly two cases when a misclassification can occur: when the test point and its nearest neighbor have different labels. The probability of this happening is the probability of the two red events: $(1-p(s|\mathbf{x}))p(s|\mathbf{x}) + p(s|\mathbf{x})(1-p(s|\mathbf{x})) = 2p(s|\mathbf{x})(1-p(s|\mathbf{x}))$.

Table of Contents

Formally, imagine the unit cube $[0,1]^d$. All training data is sampled *uniformly* within this cube, i.e. $\forall i, x_i \in [0,1]^d$, and we are considering the $k = 10$ nearest neighbors of such a test point.



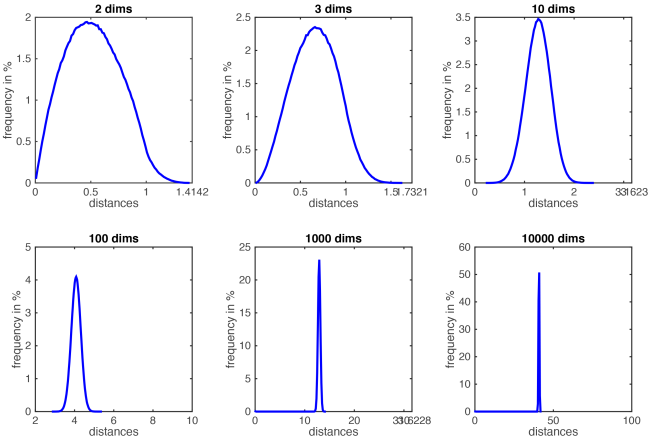
Let ℓ be the edge length of the smallest hyper-cube that contains all k-nearest neighbor of a test point. Then $\ell^d \approx \frac{k}{n}$ and $\ell \approx \left(\frac{k}{n}\right)^{1/d}$.

Distances between points

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

So as $d \gg 0$ almost the entire space is needed to find the 10-NN. This breaks down the k-NN assumptions, because the k-NN are not particularly closer (and therefore more similar) than any other data points in the training set. Why would the test point share the label with those k-nearest neighbors, if they are not actually similar to it?

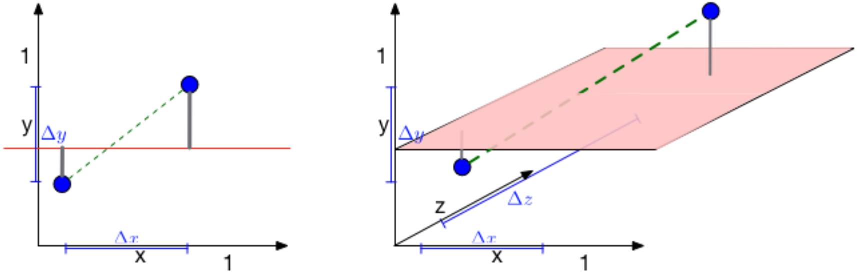
Distances between points



□: Figure demonstrating "the curse of dimensionality". The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares. As the number of dimensions d grows, all distances concentrate within a very small range .

Thinking

The distance between two randomly drawn data points increases drastically with their dimensionality. How about the distance to a hyperplane?



☐: The curse of dimensionality has different effects on distances between two points and distances between points and hyperplanes .

Table of Contents

Dimension reduction

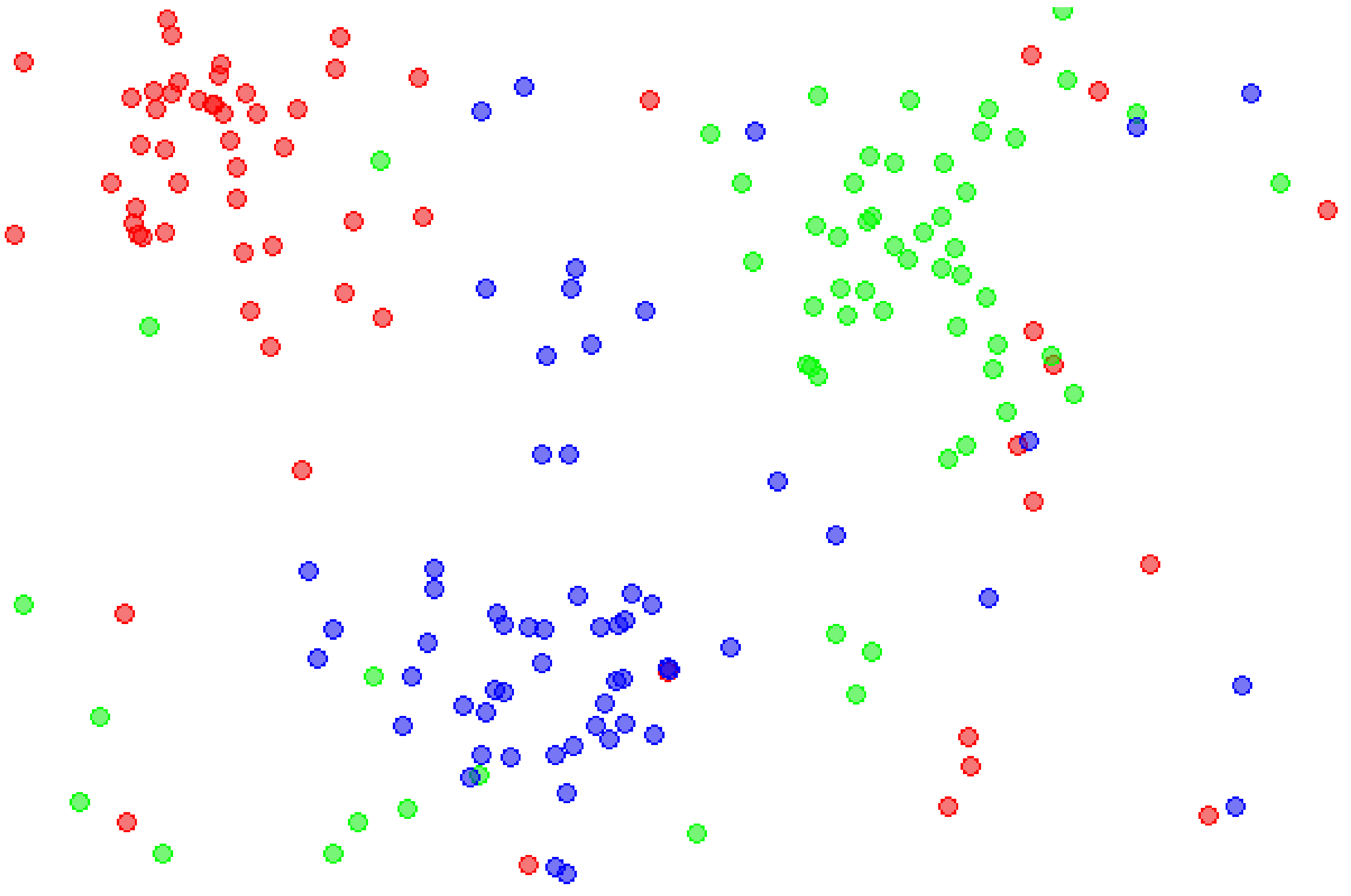
For high-dimensional data (e.g., with number of dimensions more than 10) dimension reduction is usually performed prior to applying the k-NN algorithm in order to avoid the effects of the curse of dimensionality.
Feature extraction and dimension reduction can be combined in one step using principal component analysis (**PCA**), linear discriminant analysis (**LDA**), or canonical correlation analysis (**CCA**) techniques as a pre-processing step, followed by **clustering** by k-NN on feature vectors in reduced-dimension space. In machine learning this process is also called **low-dimensional embedding**.

Data reduction

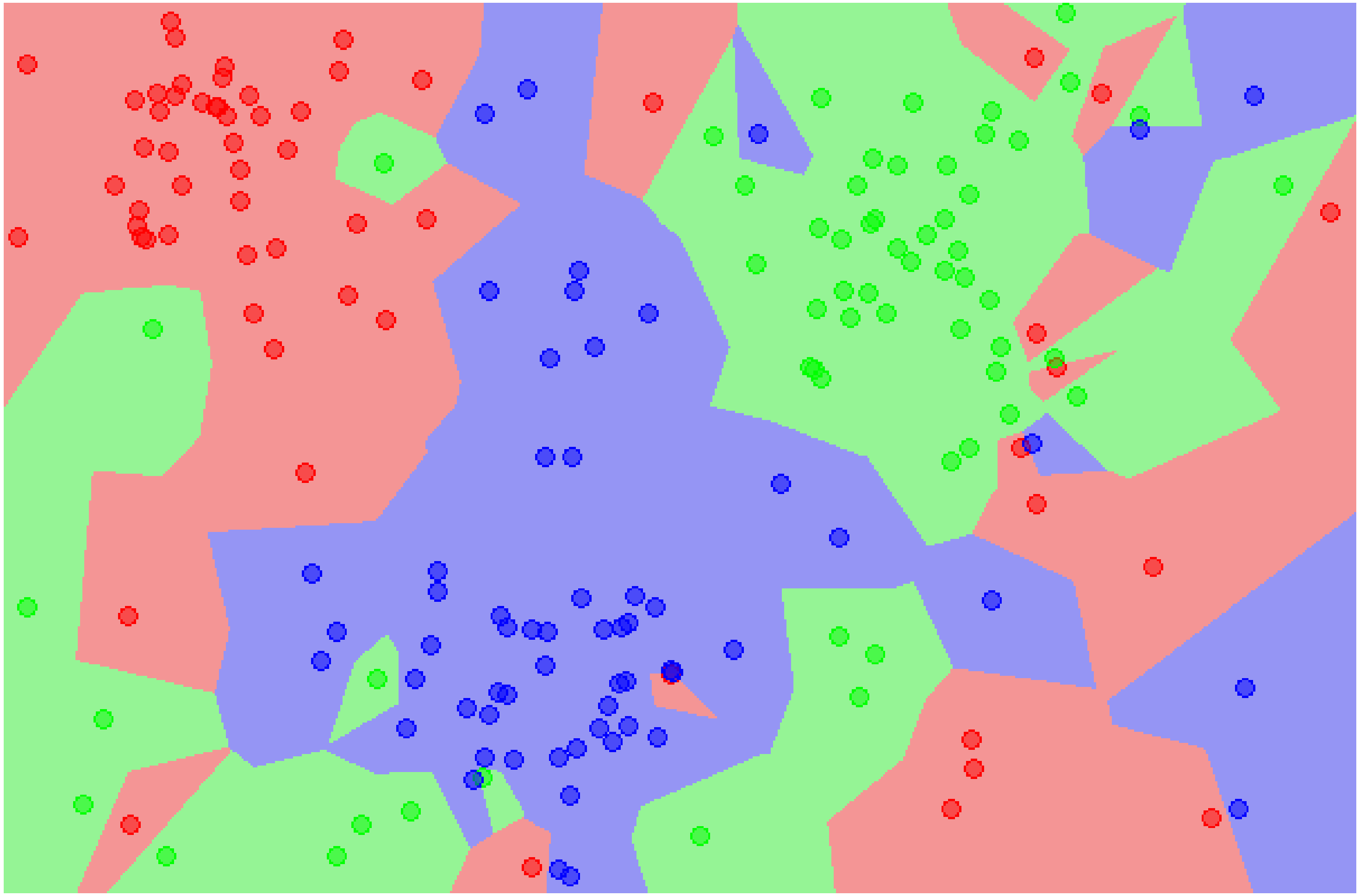
Data reduction is one of the most important problems for work with huge data sets.

1. **Dimensionality reduction**: remove dimensions irrelevant;
2. **Numerosity reduction**: smaller data representation such as parametric model;
3. Discretization and concept hierarchy generation.

A three-class problem

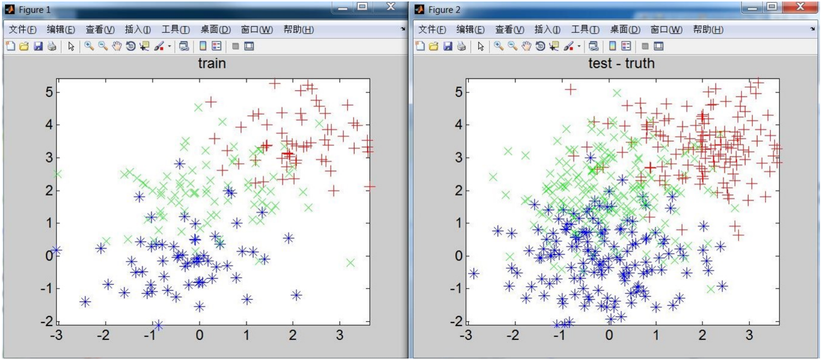


Result of NN Classifier (*k*=1)



Voronoi Tessellation (棋盘形分布）

KNN-example: data



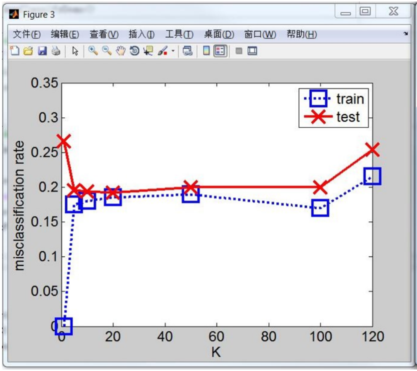


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
train
test
misclassification rate
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0 20 40 60 80 100 120
K

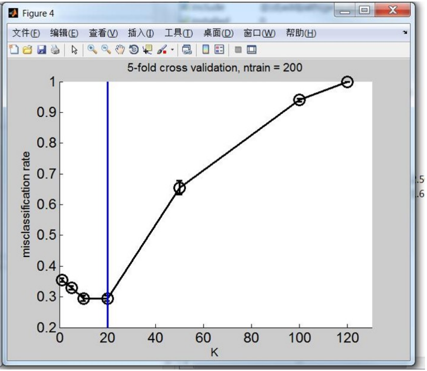


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
5-fold cross validation, ntrain = 200
misclassification rate
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0 20 40 60 80 100 120
K

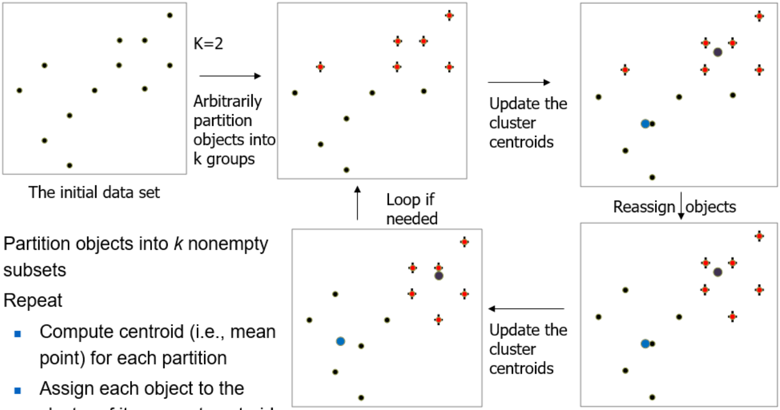
Table of Contents

Formal definition of k-Means Clustering:

Basic idea:

Given k, the k-means algorithm is implemented in four steps:

- Partition objects into k nonempty subsets
- Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., **mean point**, of the cluster)
- Assign each object to the cluster with the nearest seed point
- Go back to Step 2, stop when the assignment does not change



- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Table of Contents

Quick summary of k-NN

- k-NN is a simple and effective classifier if distances reliably reflect a semantically meaningful notion of the dissimilarity. (It becomes truly competitive through metric learning)
- As $n \to \infty$, k-NN becomes provably very accurate, but also very slow.
- As $d \gg 0$, points drawn from a probability distribution stop being similar to each other, and the k-NN assumption breaks down.
- k-NN stores the entire training dataset which it uses as its representation.
- k-NN does not learn any model.
- k-NN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

Pros and cons of k-NN

Some pros and cons of k-NN:

Pros

- No assumptions about data: useful, for example, for nonlinear data
- Simple algorithm to explain and understand/interpret
- High accuracy (relatively): it is pretty high but not competitive in comparison to better supervised learning models
- Versatile: useful for classification or regression

Cons

- Computationally expensive, because the algorithm stores all of the training data
- High memory requirement
- Stores all (or almost all) of the training data
- Prediction stage might be slow (with big N)
- Sensitive to irrelevant features and the scale of the data

The End