Curse of Dimensionality
&
Clustering

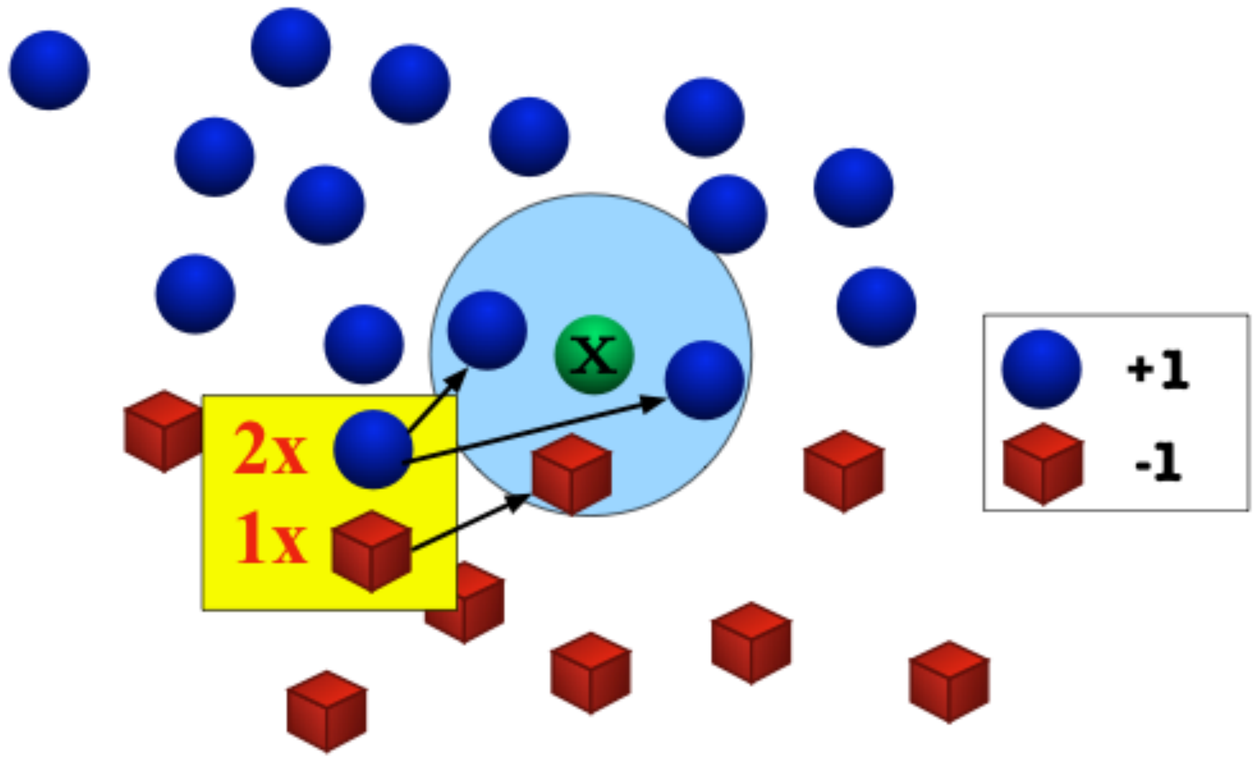
Announcement:

1. P1 will be out today

2. Office hour today 2-3 pm

Recap

The K-NN algorithm



Example: 3-NN with Euclidean distance on a binary classification data

Recap

Bayes optimal Predictor: $y_b = h_{opt}(x) = \arg \max_{y \in \{-1,1\}} P(y \mid x)$

Assume $x \in [-1,1]^2$, $P(x)$ has support everywhere $P(x) > 0, \forall x \in [-1,1]^2$, $n \to \infty$

$$\forall x \in [-1,1]^2 : \ \mathbb{P}(y \neq \text{1NN}(x)) \leq 2\mathbb{P}(y \neq h_{opt}(x))$$

Key point: K-NN can work well when $\frac{n}{d}$ is very large.

Outline for Today

1. Curse of dimensionality

2. Unsupervised Learning: Clustering and the K-means algorithm

3. Convergence of K-means

Finite sample error rate of 1-NN in high-dimension setting

(Informal result and no proof)

Fix $n \in \mathbb{N}^+$, assume $x \in [0,1]^d$, assume $P(y \mid x)$ is Lipschitz continuous with respect to x, i.e., $|P(y \mid x) - P(y \mid x')| \leq d(x, x')$

Then, we have:

$$\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq \text{1NN}(x))\right] \leq 2\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq h_{opt}(x))\right] + O\left(\left(\frac{1}{n}\right)^{1/d}\right)$$

Finite sample error rate of 1-NN in high-dimension setting

(Informal result and no proof)

Fix $n \in \mathbb{N}^+$, assume $x \in [0,1]^d$, assume $P(y \mid x)$ is Lipschitz continuous with respect to x, i.e., $|P(y \mid x) - P(y \mid x')| \leq d(x, x')$

Then, we have:

$$\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq \text{1NN}(x))\right] \leq 2\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq h_{opt}(x))\right] + O\left(\left(\frac{1}{n}\right)^{1/d}\right)$$

Q: Assume Bayes optimal has error zero, to make 1-NN's error upper bounded by 0.1, How many samples do we need?

Finite sample error rate of 1-NN in high-dimension setting

(Informal result and no proof)

Fix $n \in \mathbb{N}^+$, assume $x \in [0,1]^d$, assume $P(y \mid x)$ is Lipschitz continuous with respect to x, i.e., $|P(y \mid x) - P(y \mid x')| \leq d(x, x')$

Then, we have:

$$\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq \text{1NN}(x))\right] \leq 2\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq h_{opt}(x))\right] + O\left(\left(\frac{1}{n}\right)^{1/d}\right)$$

Q: Assume Bayes optimal has error zero, to make 1-NN's error upper bounded by 0.1, How many samples do we need?

$$(1/n)^{1/d} = 0.1$$

Finite sample error rate of 1-NN in high-dimension setting

(Informal result and no proof)

Fix $n \in \mathbb{N}^+$, assume $x \in [0,1]^d$, assume $P(y \mid x)$ is Lipschitz continuous with respect to x, i.e., $|P(y \mid x) - P(y \mid x')| \leq d(x, x')$

Then, we have:

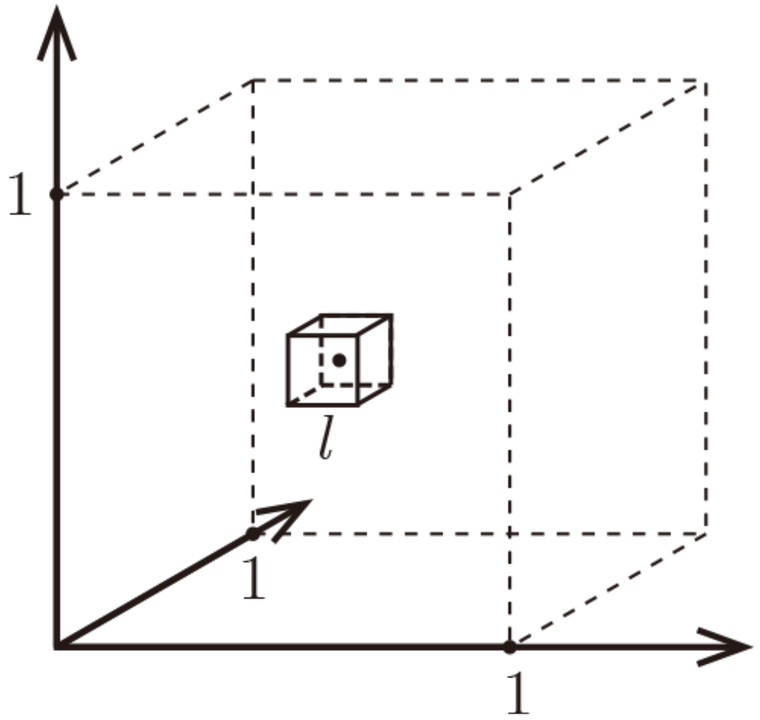
$$\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq \text{1NN}(x))\right] \leq 2\mathbb{E}_{x,y\sim P}\left[\mathbf{1}(y \neq h_{opt}(x))\right] + O\left(\left(\frac{1}{n}\right)^{1/d}\right)$$

Q: Assume Bayes optimal has error zero, to make 1-NN's error upper bounded by 0.1, How many samples do we need?

$$(1/n)^{1/d} = 0.1 \Rightarrow n = (10)^d$$

Curse of Dimensionality Intuitive Explanation

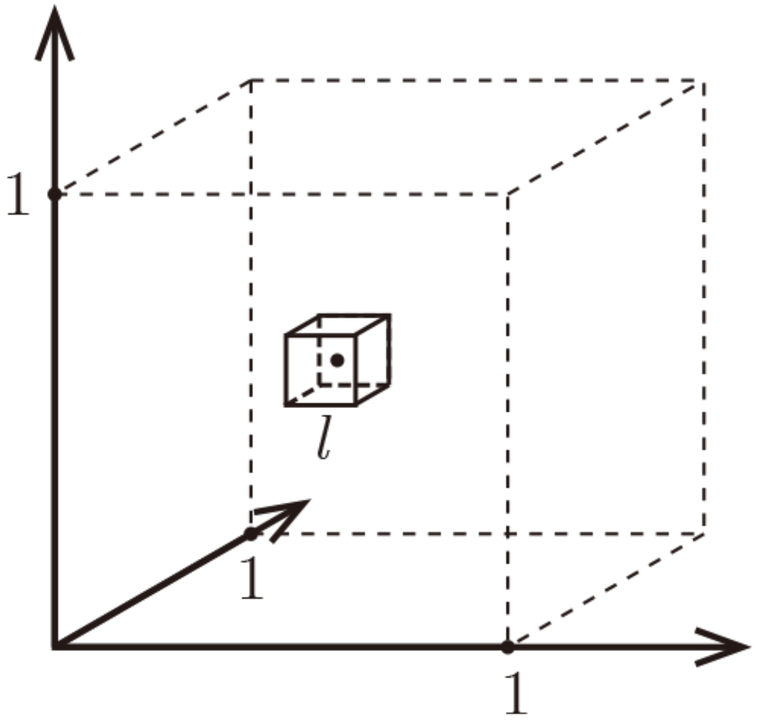
Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!



Curse of Dimensionality Intuitive Explanation

Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!

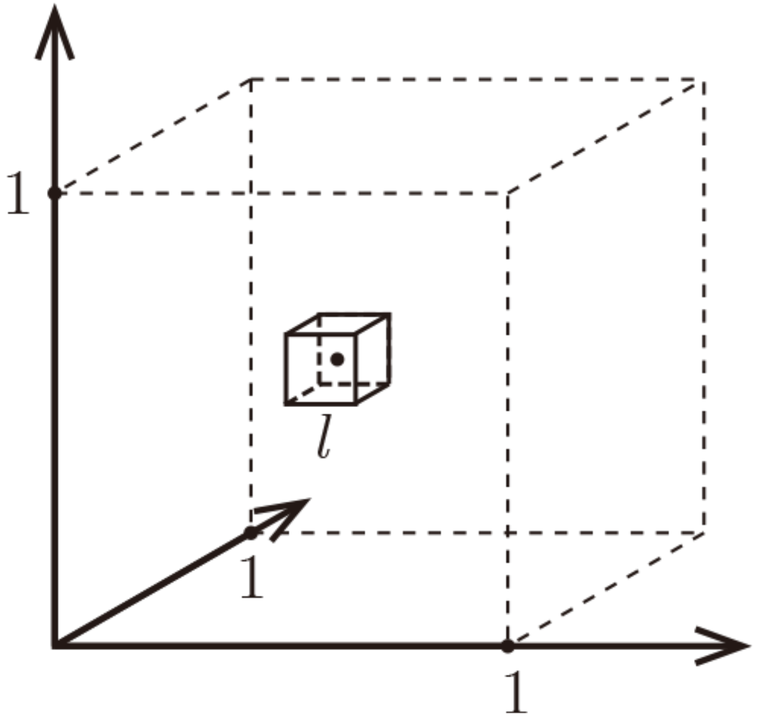
Example: let us consider uniform distribution over a cube $[0,1]^d$



Curse of Dimensionality Intuitive Explanation

Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!

Example: let us consider uniform distribution over a cube $[0,1]^d$

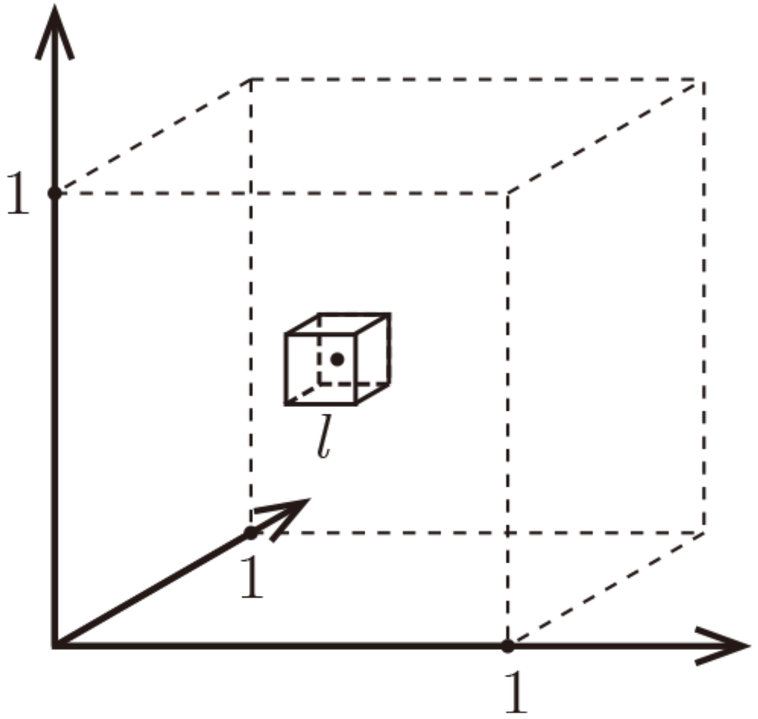


Q: sample x uniformly, what is the probability that x is inside the small cube?

Curse of Dimensionality Intuitive Explanation

Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!

Example: let us consider uniform distribution over a cube $[0,1]^d$



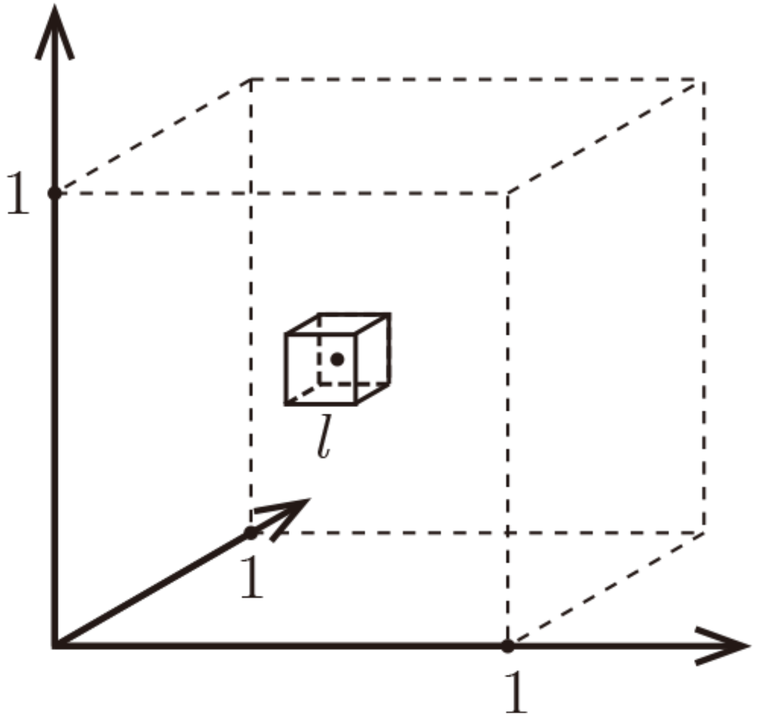
Q: sample x uniformly, what is the probability that x is inside the small cube?

A: Volume(small cube)/volume($[0,1]^d$)

Curse of Dimensionality Intuitive Explanation

Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!

Example: let us consider uniform distribution over a cube $[0,1]^d$

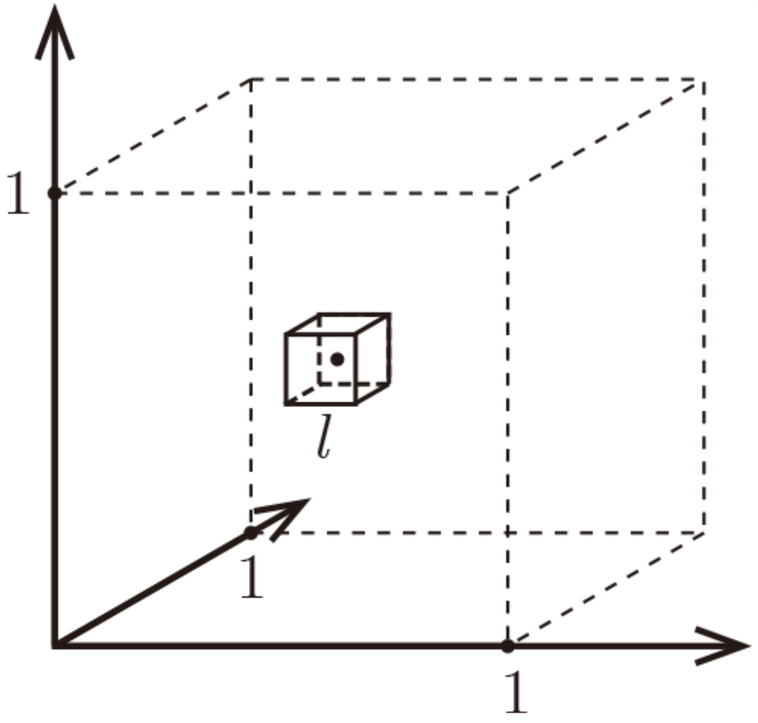


Q: sample x uniformly, what is the probability that x is inside the small cube?

A: Volume(small cube)/volume($[0,1]^d$) $= l^d$

Curse of Dimensionality Explanation

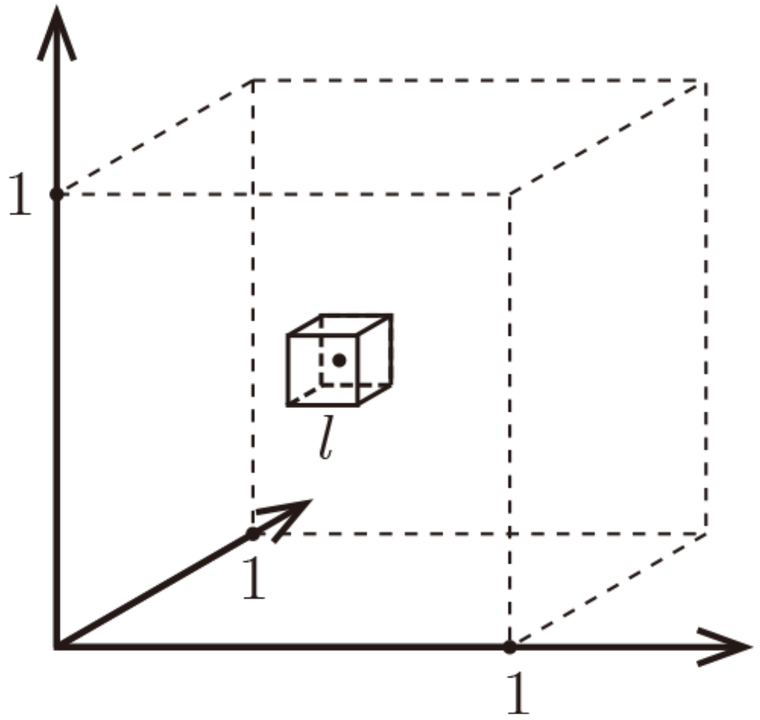
Example: let us consider uniform distribution over a cube $[0,1]^d$



Now assume we sample n points uniform randomly in the big cube, and we observe K points fall inside the small cube

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

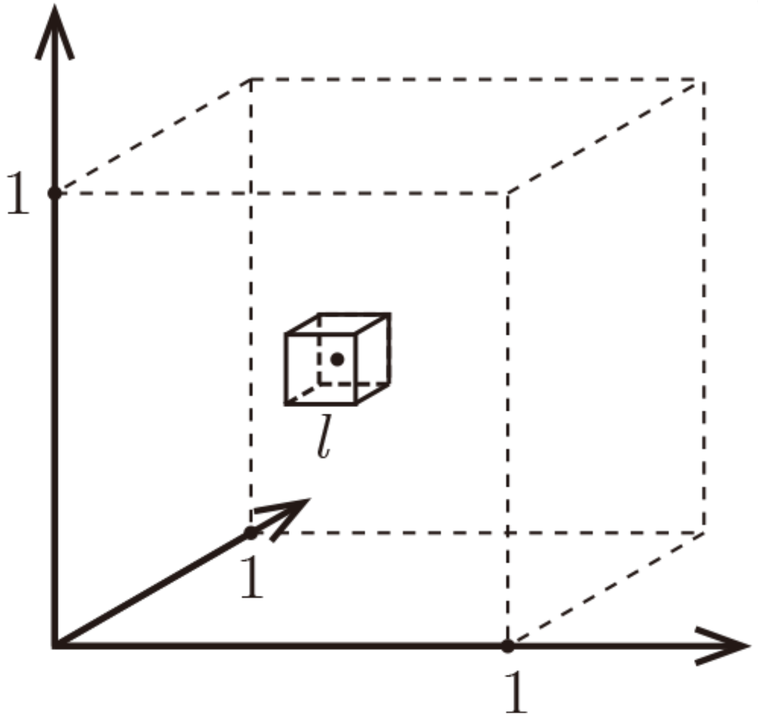


Now assume we sample n points uniform randomly in the big cube, and we observe K points fall inside the small cube

So empirically, the probability of sampling a point inside the small cube is roughly K/n

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$



Now assume we sample n points uniform randomly in the big cube, and we observe K points fall inside the small cube

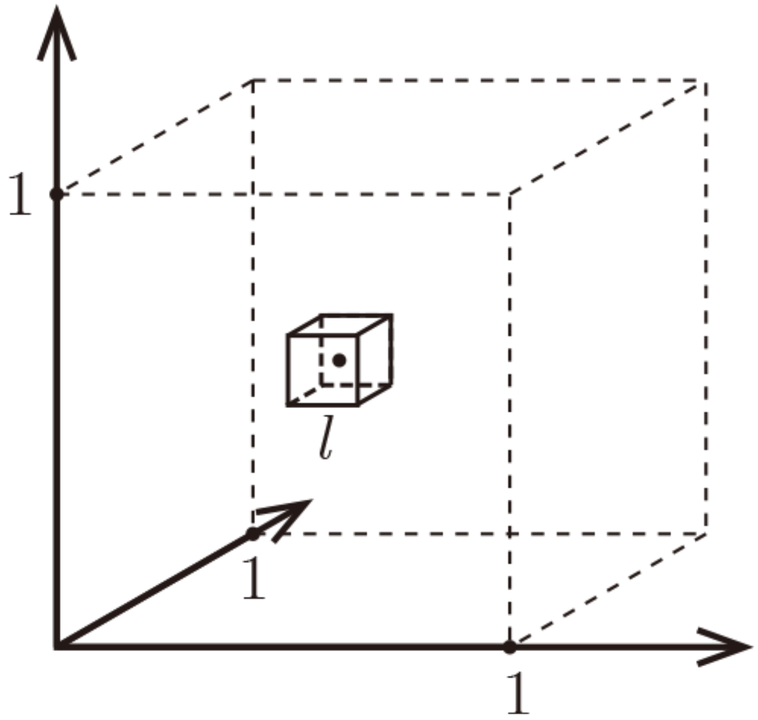
So empirically, the probability of sampling a point inside the small cube is roughly K/n

Thus, we have $l^d \approx \frac{K}{n}$

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

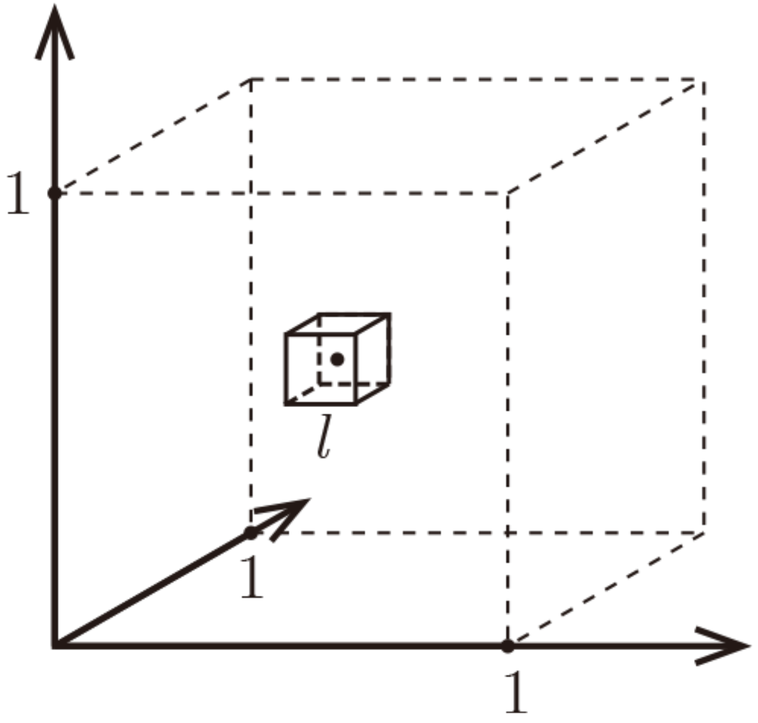
We have $l^d \approx \frac{K}{n}$



Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$

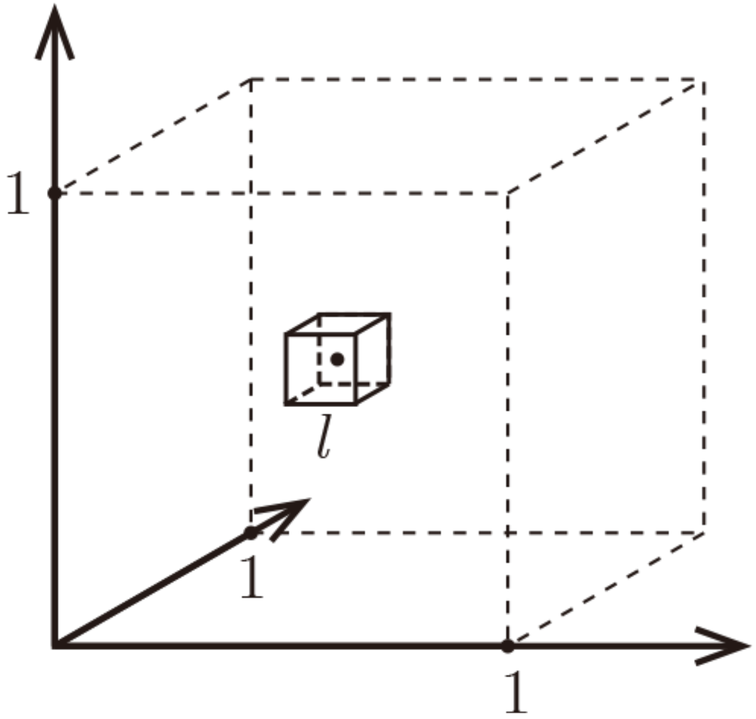


Q: how large we should set l, s.t., we will have K examples (out of n) fall inside the small cube?

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$



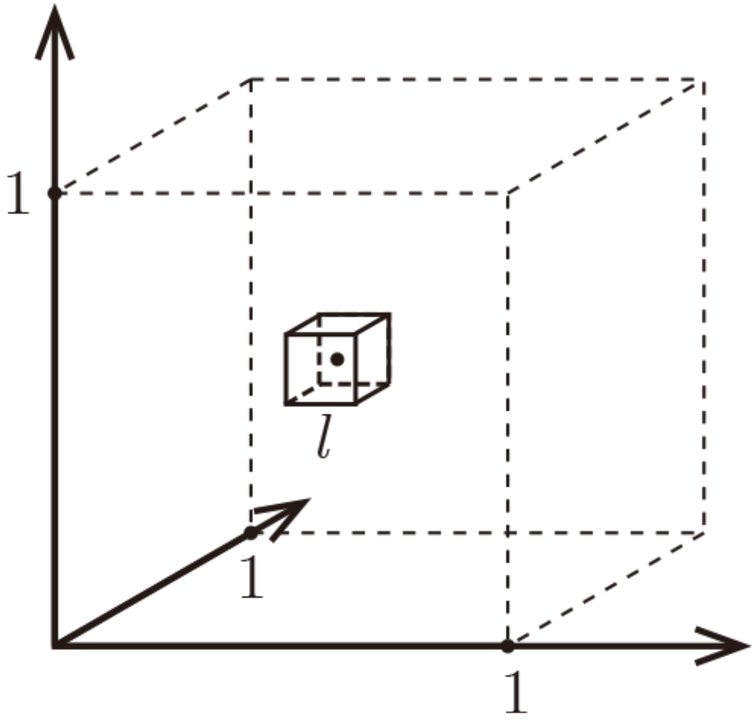
Q: how large we should set l, s.t., we will have K examples (out of n) fall inside the small cube?

$$l \approx (K/n)^{1/d}$$

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$



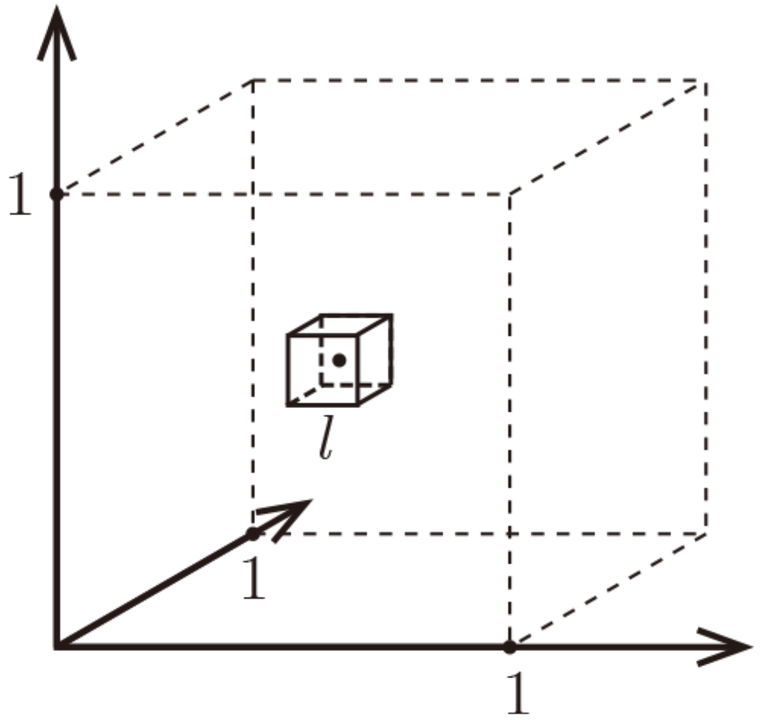
Q: how large we should set l, s.t., we will have K examples (out of n) fall inside the small cube?

$$l \approx (K/n)^{1/d} \rightarrow 1, \text{ as } d \rightarrow \infty$$

Curse of Dimensionality Explanation

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$



Q: how large we should set l, s.t., we will have K examples (out of n) fall inside the small cube?

$$l \approx (K/n)^{1/d} \rightarrow 1, \text{ as } d \rightarrow \infty$$

Bad news: when $d \rightarrow \infty$, the K nearest neighbors will be all over the place!
(Cannot trust them, as they are not nearby points anymore!)

The distance between two sampled points increases as d grows

The distance between two sampled points increases as d grows

In $[0,1]^d$, we uniformly sample two points x, x', calculate

$$d(x, x') = \|x - x'\|_2$$

The distance between two sampled points increases as d grows

In $[0,1]^d$, we uniformly sample two points x, x', calculate

$$d(x, x') = \|x - x'\|_2$$

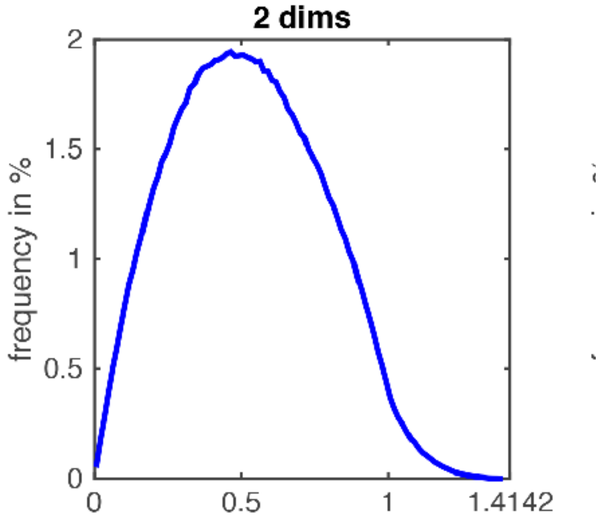
Let's plot the distribution of such distance:

The distance between two sampled points increases as d grows

In $[0,1]^d$, we uniformly sample two points x, x', calculate

$$d(x, x') = \|x - x'\|_2$$

Let's plot the distribution of such distance:

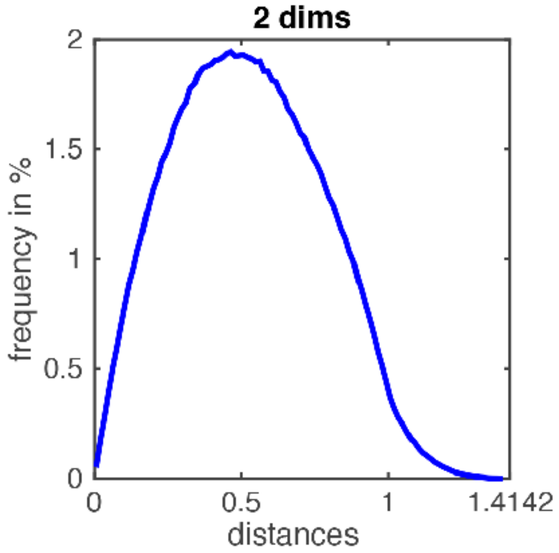


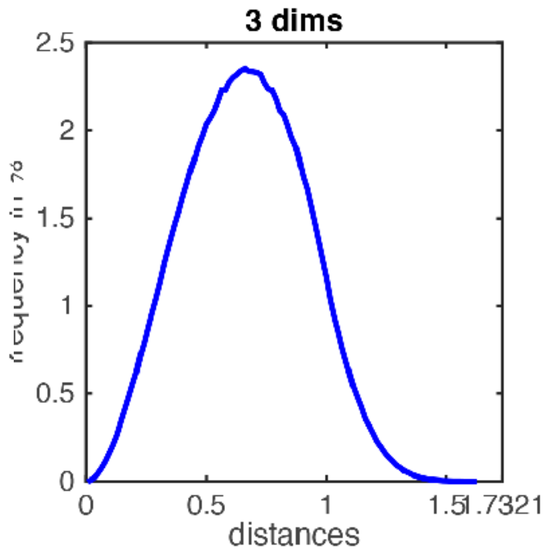
The distance between two sampled points increases as d grows

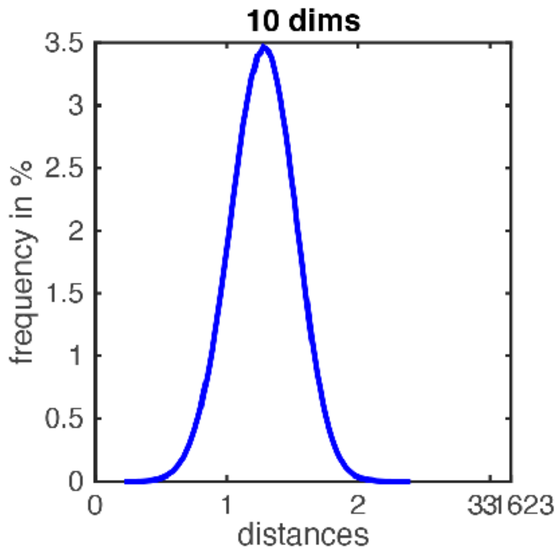
In $[0,1]^d$, we uniformly sample two points x, x', calculate

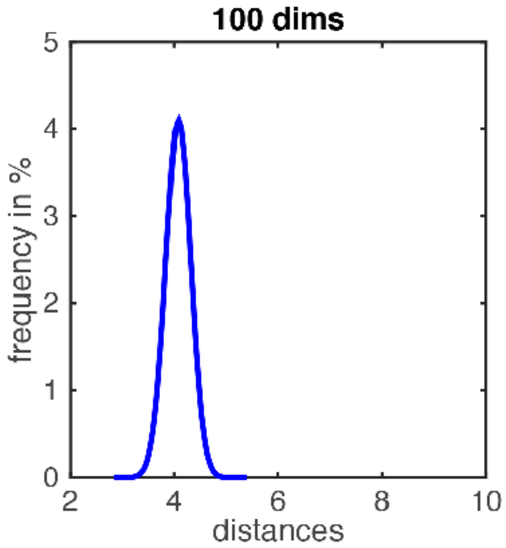
$$d(x, x') = \|x - x'\|_2$$

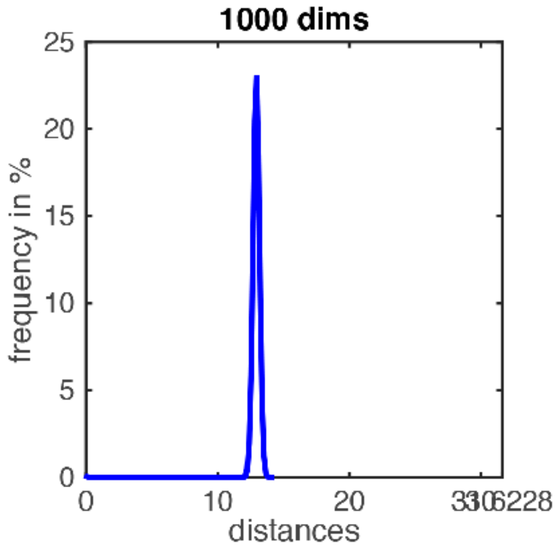
Let's plot the distribution of such distance:

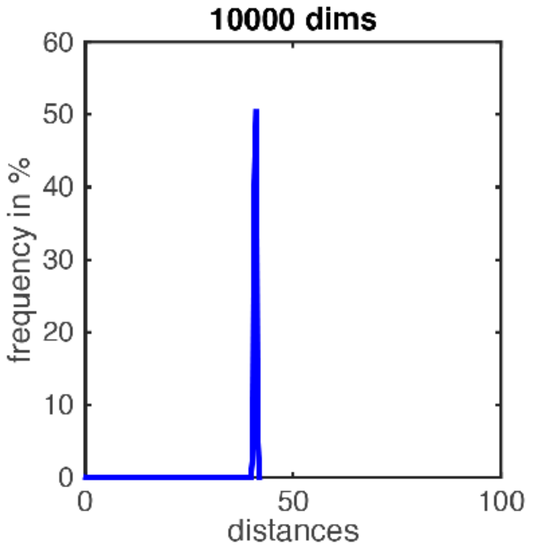










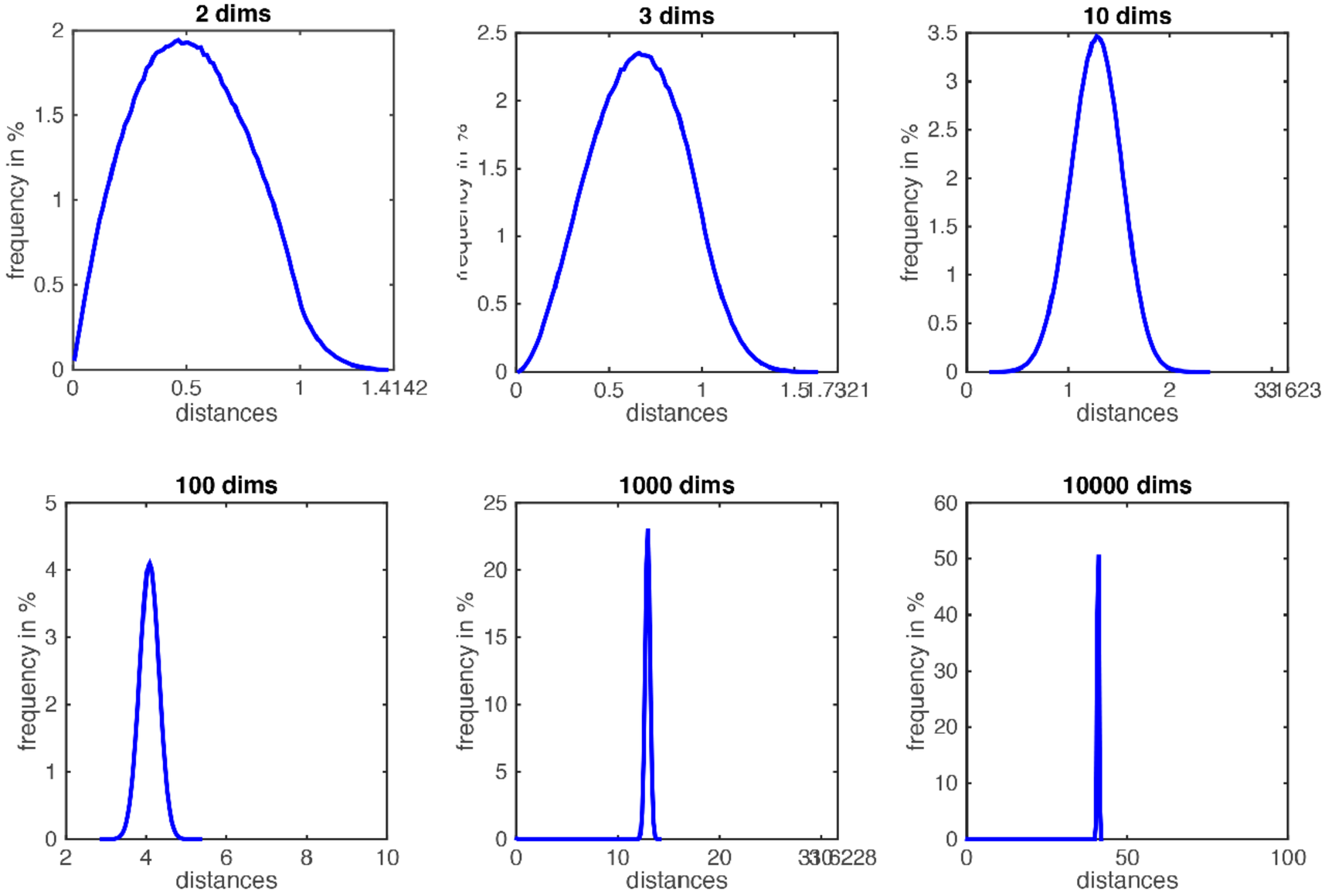


The distance between two sampled points increases as d grows

In $[0,1]^d$, we uniformly sample two points x, x', calculate

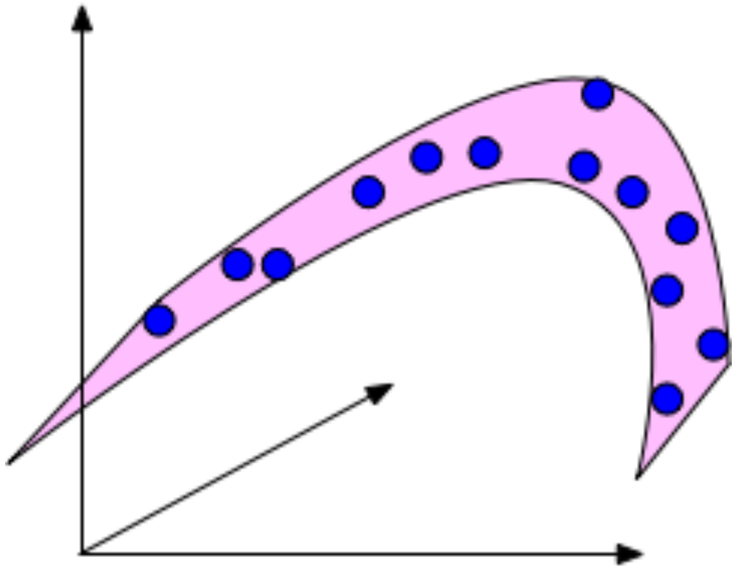
$$d(x, x') = \|x - x'\|_2$$

Let's plot the distribution of such distance:



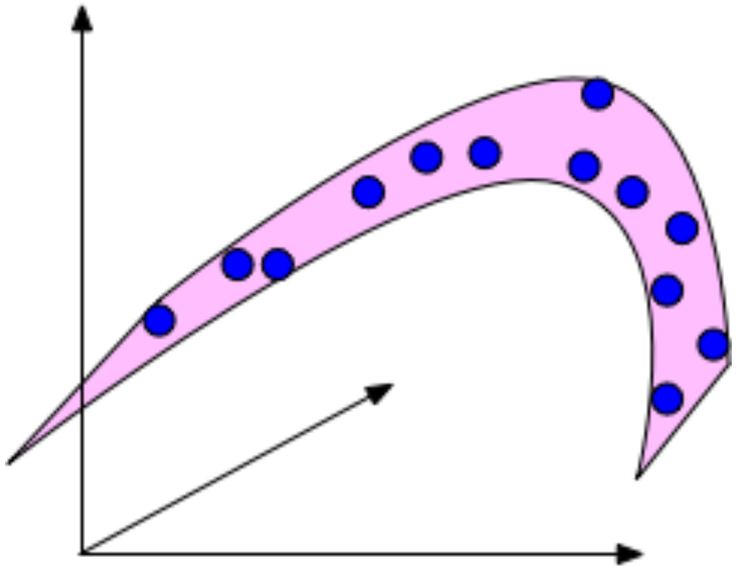
Distance increases as $d \to \infty$

Luckily, real world data often has low-dimensional structure!



Data lives in 2-d manifold

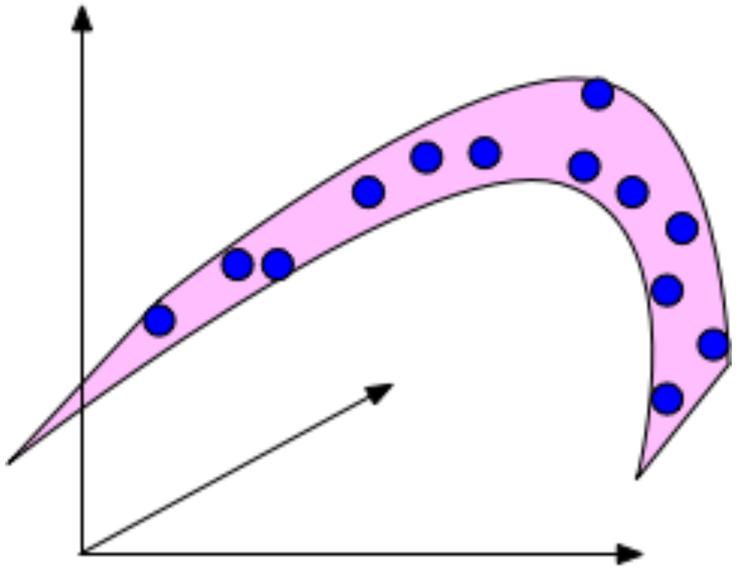
Luckily, real world data often has low-dimensional structure!



Data lives in 2-d manifold

Example: face images

Luckily, real world data often has low-dimensional structure!

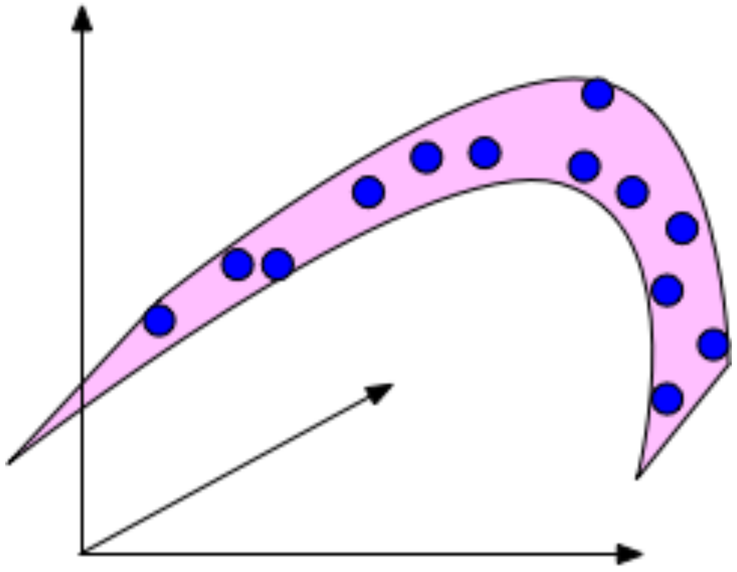


Data lives in 2-d manifold

Example: face images

Original image: $\mathbb{R}^{64^2}$

Luckily, real world data often has low-dimensional structure!



Data lives in 2-d manifold

Example: face images

Original image: $\mathbb{R}^{64^2}$

Next week: we will see that these faces approximately live in 100-d space!

Outline for Today

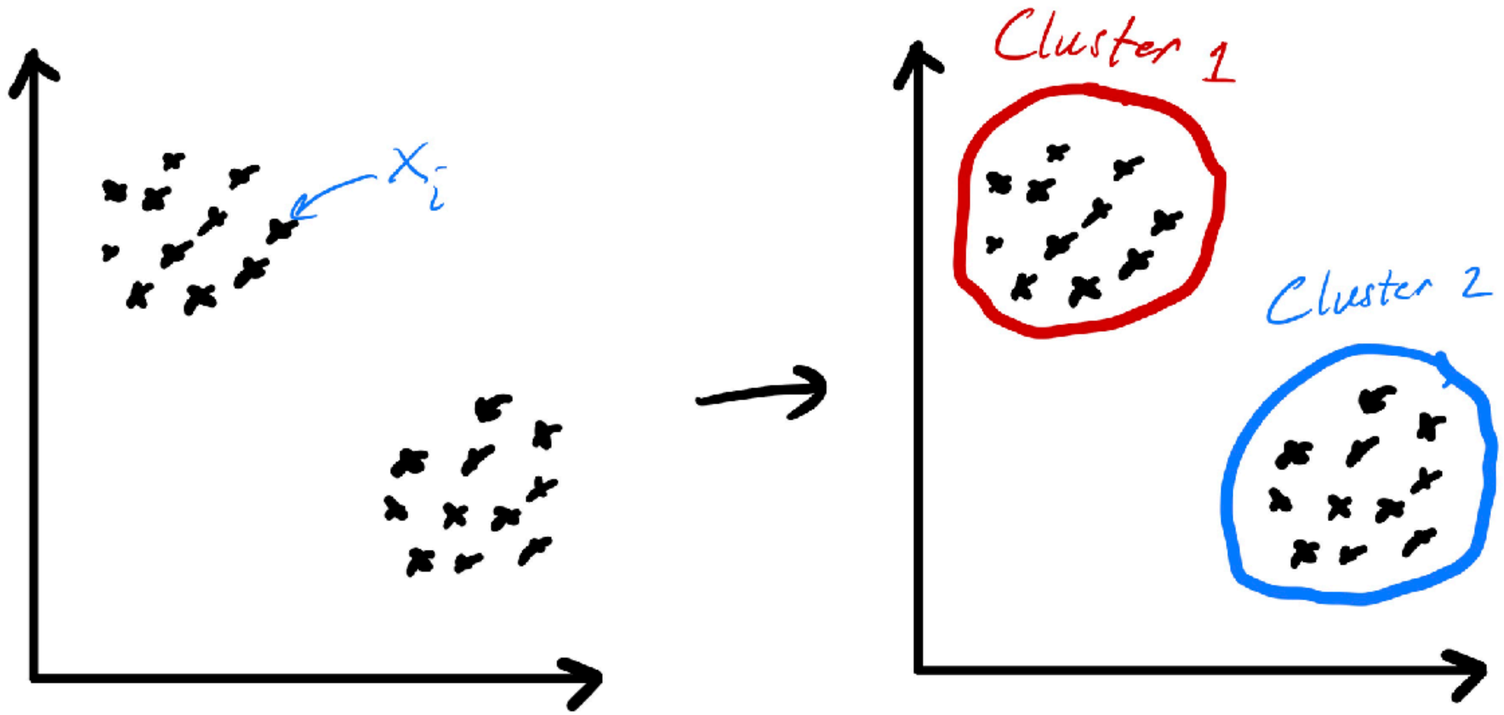
1. Curse of dimensionality 

2. Unsupervised Learning: Clustering and the K-means algorithm

3. Convergence of K-means

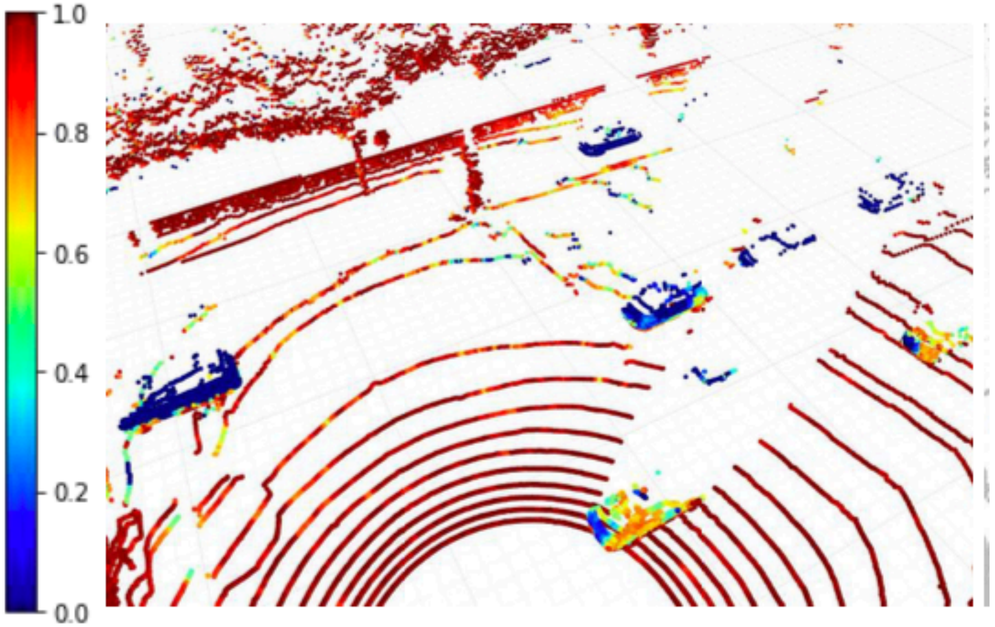
What is clustering?

It is an unsupervised learning procedure (i.e., applies to data without ground truth labels)



Usage of clustering algorithms in real world

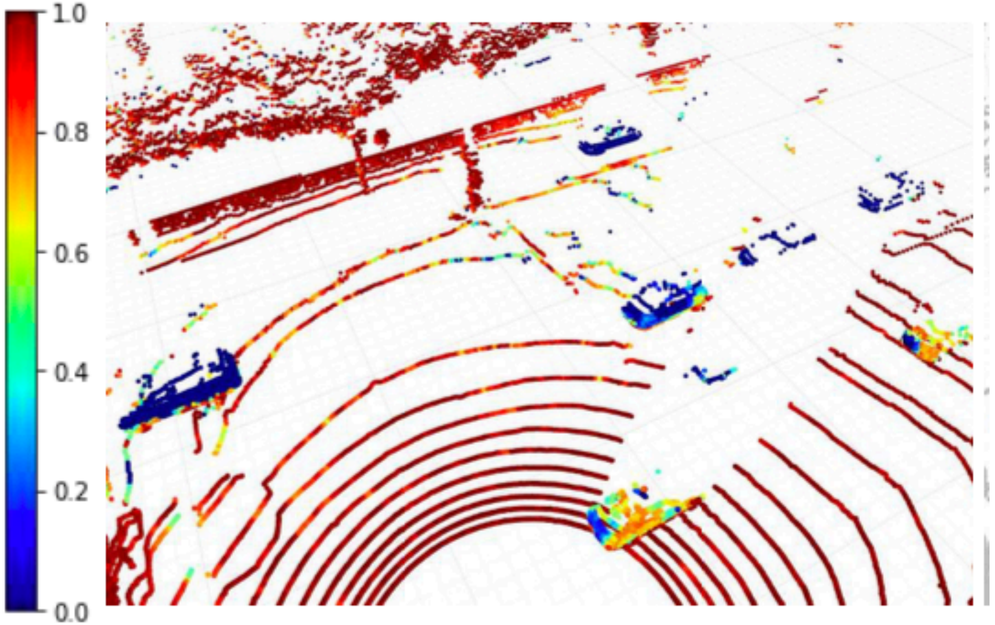
Example: Learning to detect cars without ground truth label



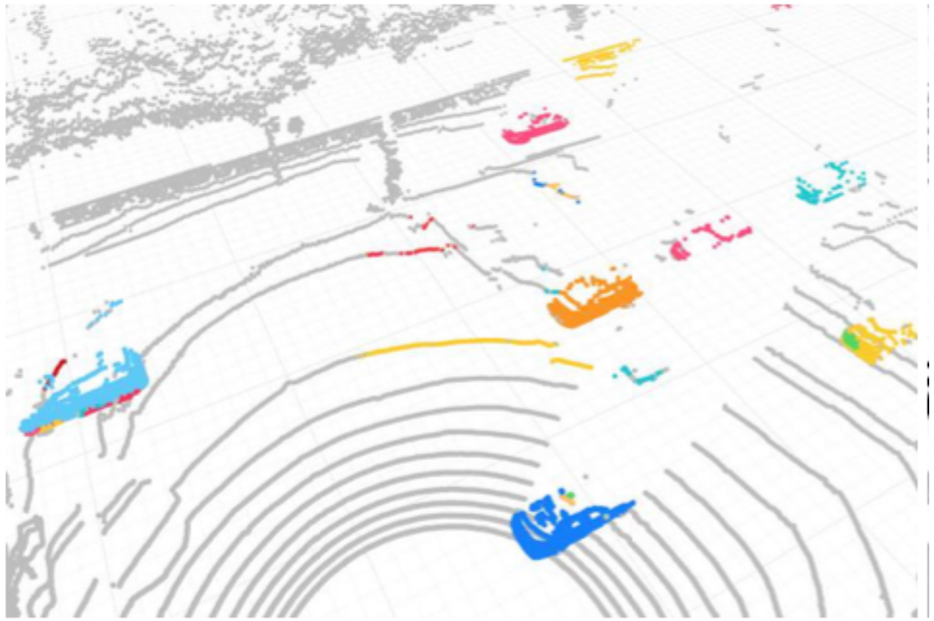
A point cloud from a Lidar sweep (4-d data)

Usage of clustering algorithms in real world

Example: Learning to detect cars without ground truth label



Apply clustering
on this dataset
(point cloud)



A point cloud from a Lidar sweep (4-d data)

Different color represents different clusters

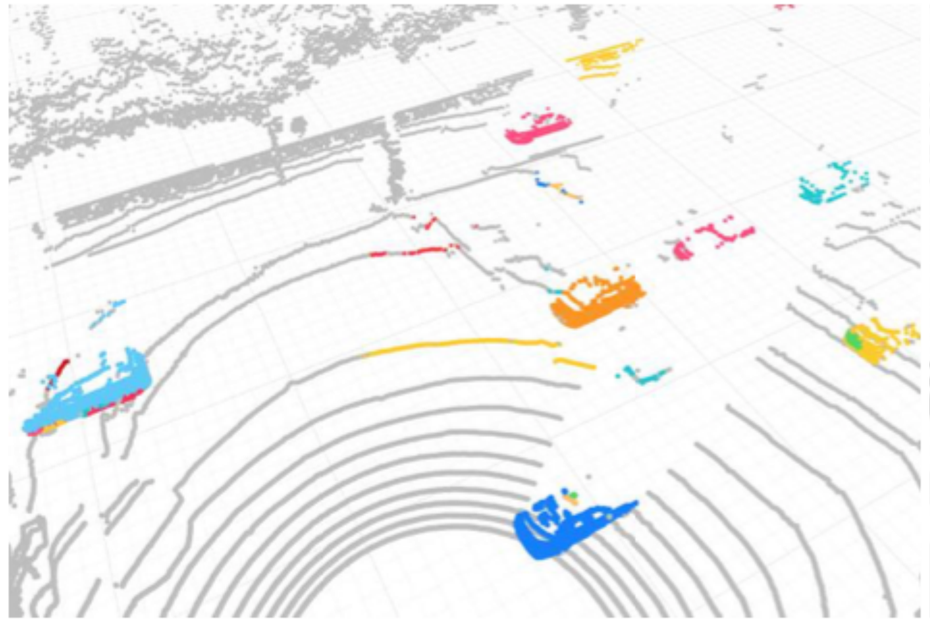
Usage of clustering algorithms in real world

Example: Learning to detect cars without ground truth label

3. Fit Bounding Boxes

These boxes are the pseudo-labels we use to train detector

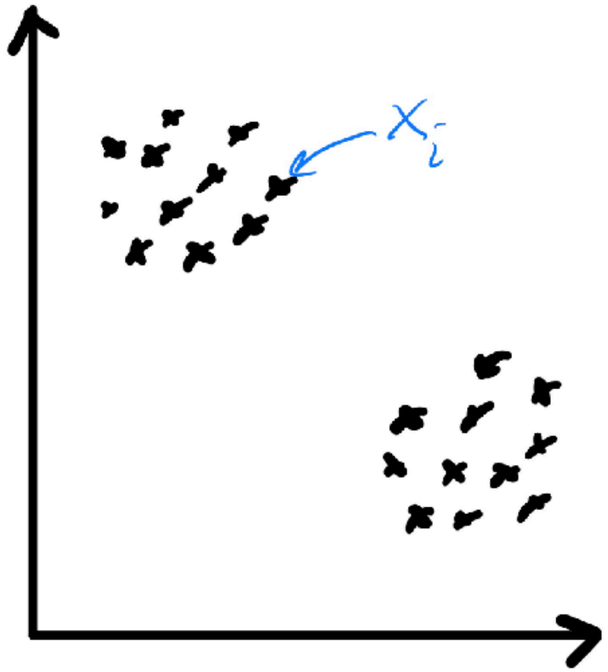
Fitting bounding box around clusters



Different color represents different clusters

The K-means algorithm

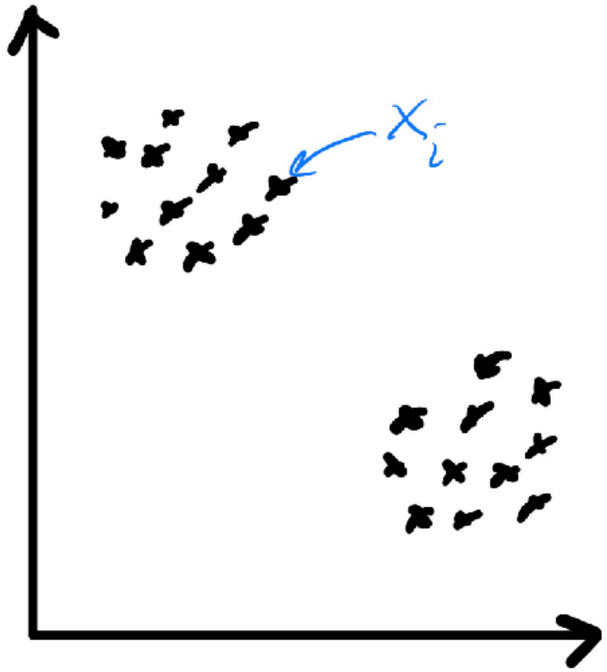
Input $\mathscr{D} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, parameters K



The K-means algorithm

Input $\mathscr{D} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, parameters K

Expected output: K centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}, \mu_i \in \mathbb{R}^d$, and K clusters $C_1 \ldots, C_K$

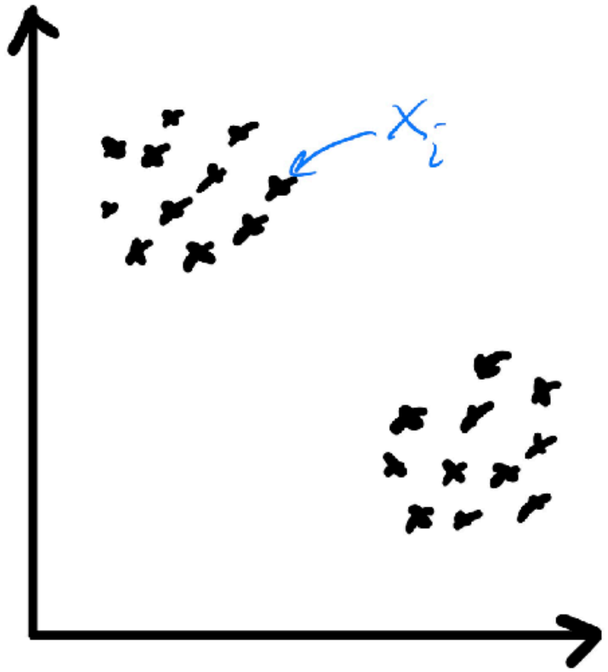


The K-means algorithm

Input $\mathscr{D} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, parameters K

Expected output: K centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}, \mu_i \in \mathbb{R}^d$, and K clusters $C_1 \ldots, C_K$

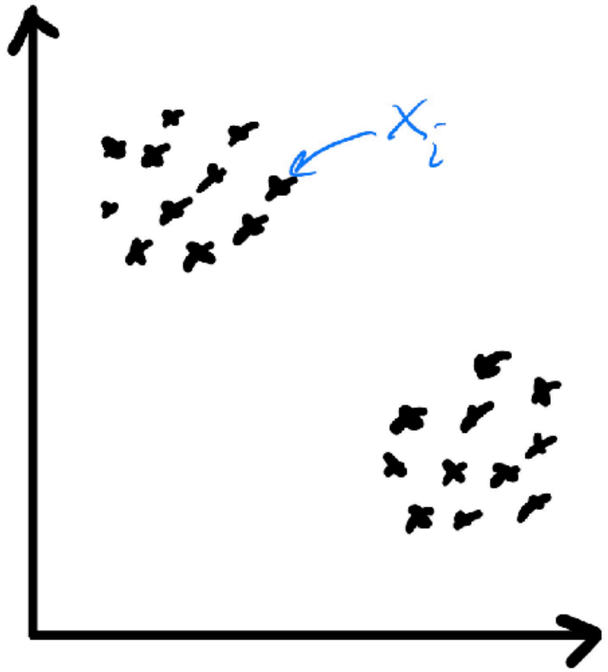
The data assignment procedure:



The K-means algorithm

Input $\mathcal{D} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, parameters K

Expected output: K centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}, \mu_i \in \mathbb{R}^d$, and K clusters $C_1 \ldots, C_K$



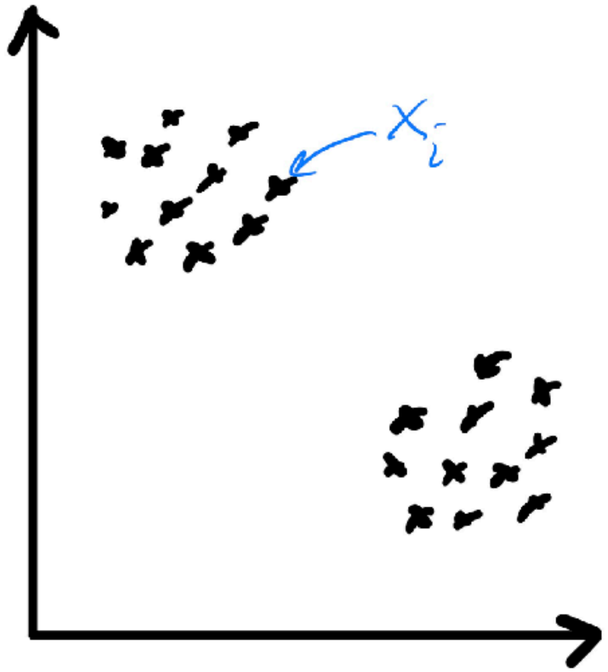
The data assignment procedure:

If we had K centroids, we could split the dataset into K clusters, $C_1, \ldots, C_K$, by assigning each data point to its nearest centroid

The K-means algorithm

Input $\mathcal{D} = \{x_1, \ldots, x_n\}, x_i \in \mathbb{R}^d$, parameters K

Expected output: K centroids $\{\mu_1, \mu_2, \ldots, \mu_k\}, \mu_i \in \mathbb{R}^d$, and K clusters $C_1 \ldots, C_K$



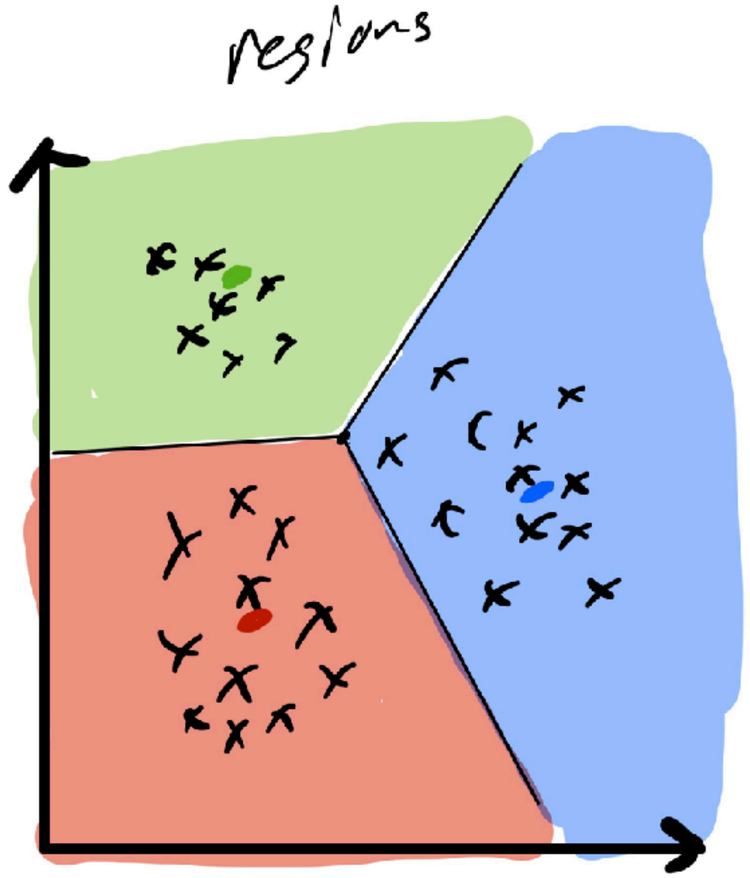
The data assignment procedure:

If we had K centroids, we could split the dataset into K clusters, $C_1, \ldots, C_K$, by assigning each data point to its nearest centroid

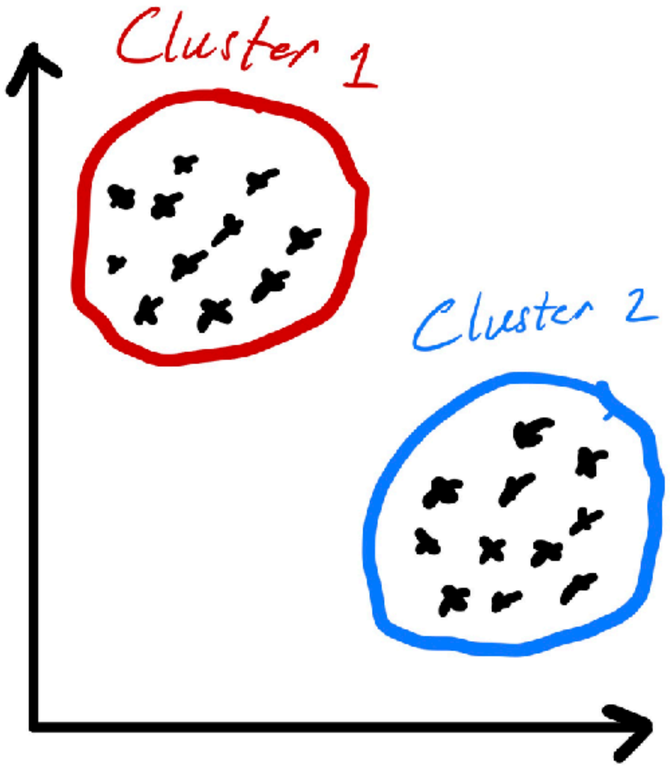
$$C_i = \{x \in \mathcal{D} \text{ s.t., } \mu_i \text{ is the closest centroid to } x\}$$

The data assignment procedure

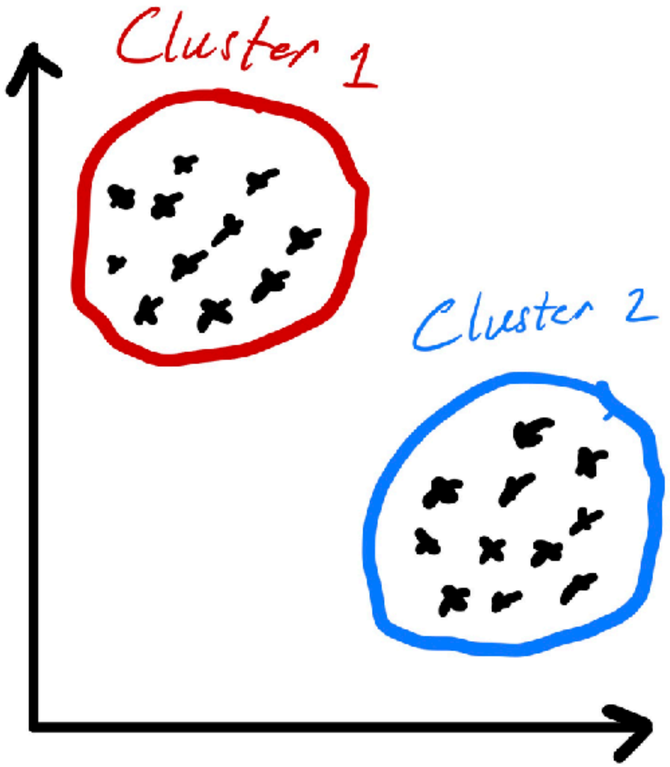
K centroids $\mu_1, \ldots, \mu_k$ splits the space into a voronoi diagram



The centroid computation procedure



The centroid computation procedure



If we magically had the clusters $C_1, \ldots, C_K$, we could compute centroids as follows:

μ_i : the mean of the data in C_i

The K-means algorithm

Iterate between Centroid computation and Data Assignment!

The K-means algorithm

Iterate between Centroid computation and Data Assignment!

Initialize K clusters $C_1, C_2, \ldots, C_K$, where $\cup_{i=1}^{K} C_i = \mathscr{D}$, and $C_i \cap C_j = \emptyset$, for $i \neq j$

The K-means algorithm

Iterate between Centroid computation and Data Assignment!

Initialize K clusters $C_1, C_2, \ldots, C_K$, where $\cup_{i=1}^{K} C_i = \mathscr{D}$, and $C_i \cap C_j = \emptyset$, for $i \neq j$

Repeat until convergence:

The K-means algorithm

Iterate between Centroid computation and Data Assignment!

Initialize K clusters $C_1, C_2, \ldots, C_K$, where $\cup_{i=1}^{K} C_i = \mathcal{D}$, and $C_i \cap C_j = \emptyset$, for $i \neq j$

Repeat until convergence:

1. centroids computation using $C_1, \ldots, C_K$, i.e.,for all i,
   $\mu_i = \sum_{x \in C_i} x / |C_i|$ (i.e., the mean of the data in C_i)

The K-means algorithm

Iterate between Centroid computation and Data Assignment!

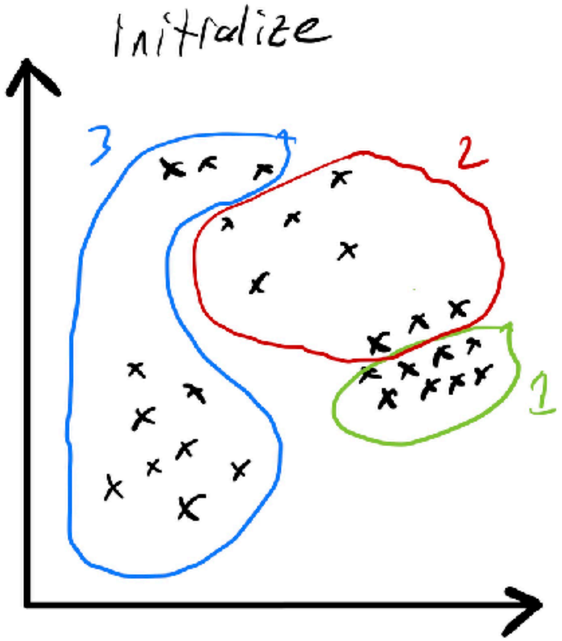
Initialize K clusters $C_1, C_2, \ldots, C_K$, where $\cup_{i=1}^{K} C_i = \mathcal{D}$, and $C_i \cap C_j = \emptyset$, for $i \neq j$

Repeat until convergence:

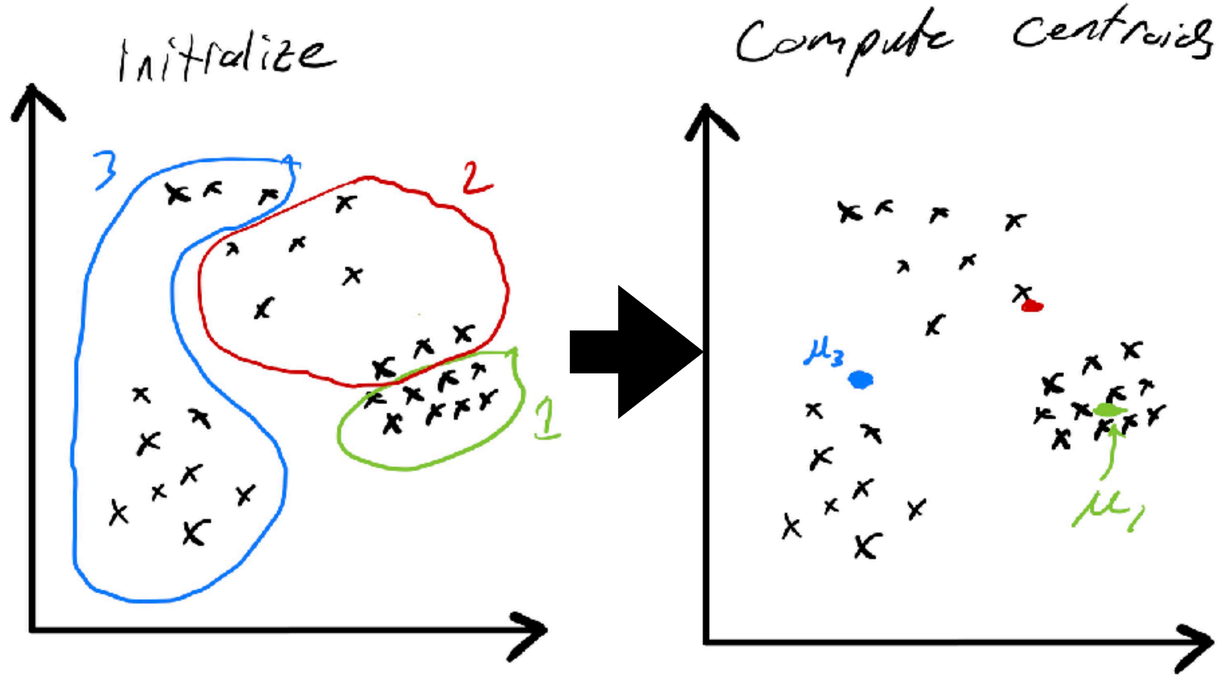
1. centroids computation using $C_1, \ldots, C_K$, i.e.,for all i,
$\mu_i = \sum_{x \in C_i} x / |C_i|$ (i.e., the mean of the data in C_i)

2. the data assignment procedure, i.e., re-split data into $C_1, \ldots, C_K$, using $\mu_1, \ldots, \mu_k$

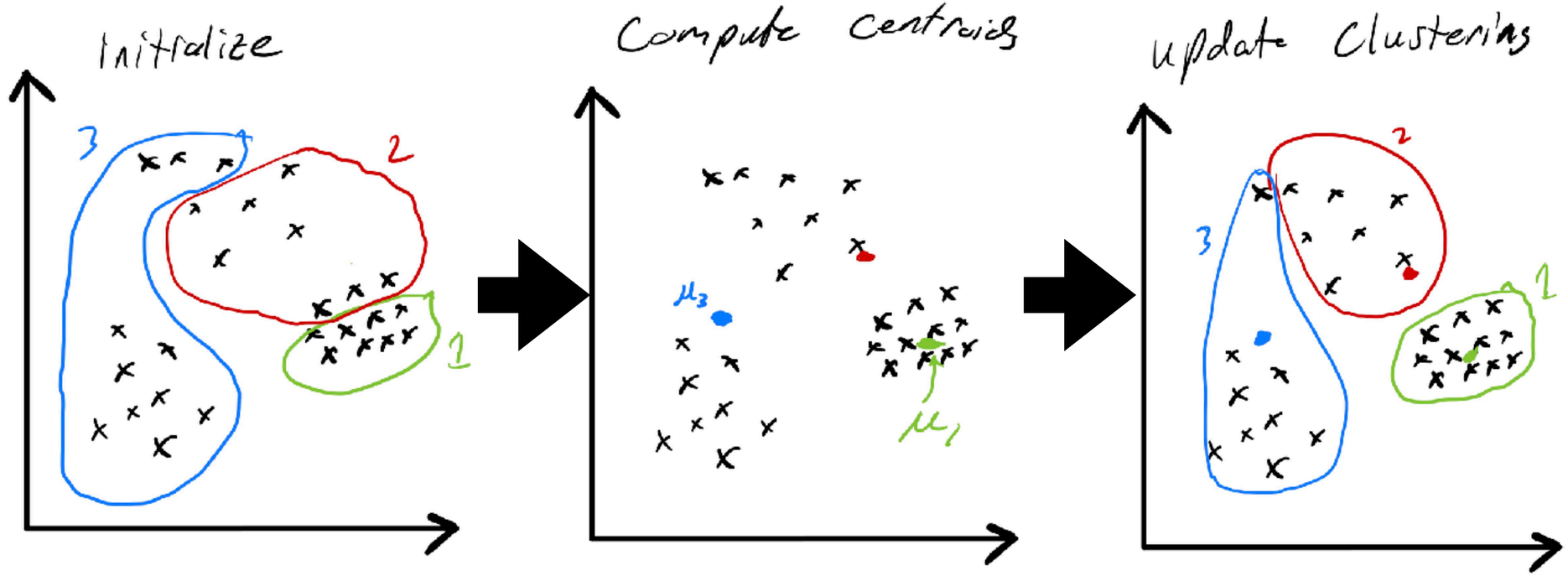
The K-means algorithm



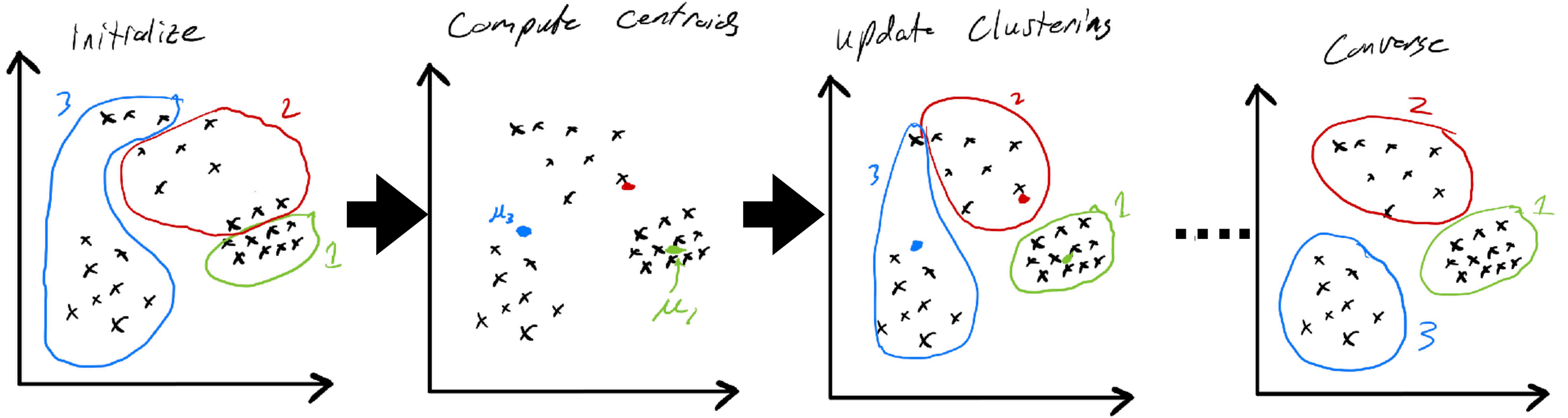
The K-means algorithm



The K-means algorithm



The K-means algorithm



Outline for Today

1. Curse of dimensionality ✓

2. Unsupervised Learning: Clustering and the K-means algorithm ✓

3. Convergence of K-means

Does K-means algorithm converge?

Yes, though it does not guarantee to return the globally optimal solution

Does K-means algorithm converge?

Yes, though it does not guarantee to return the globally optimal solution

Given any K disjoint groups $C_1, C_2, \ldots, C_K$, and any K centroids, define a loss function:

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

Does K-means algorithm converge?

Yes, though it does not guarantee to return the globally optimal solution

Given any K disjoint groups $C_1, C_2, \ldots, C_K$, and any K centroids, define a loss function:

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

Total distance of points in C_i to μ_i

K-means as a Coordinate Descent Algorithm

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means as a Coordinate Descent Algorithm

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means minimizes ℓ in an alternating fashion:

K-means as a Coordinate Descent Algorithm

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means minimizes ℓ in an alternating fashion:

Q1: w/ $C_1, \ldots, C_K$ fix, what is $\arg \min_{\mu_1, \ldots, \mu_k} \ell(\{C_i\}, \{\mu_i\})$?

K-means as a Coordinate Descent Algorithm

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means minimizes ℓ in an alternating fashion:

Q1: w/ $C_1, \ldots, C_K$ fix, what is $\arg \min_{\mu_1, \ldots, \mu_k} \ell(\{C_i\}, \{\mu_i\})$?

Q2: w/ $\mu_1, \ldots, \mu_K$ fix, what is $\arg \min_{C_1, \ldots, C_k} \ell(\{C_i\}, \{\mu_i\})$?

K means is doing Coordinate Descent here

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means Algorithm: (re-stated from a different perspective)

Initialize $\mu_1, \ldots, \mu_K$

Repeat until convergence:

K means is doing Coordinate Descent here

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means Algorithm: (re-stated from a different perspective)

Initialize $\mu_1, \ldots, \mu_K$

Repeat until convergence:

$$C_1, \ldots, C_K = \arg \min_{C_1, \ldots, C_k} \ell(\{C_i\}, \{\mu_i\})$$

K means is doing Coordinate Descent here

$$\ell(\{C_i\}, \{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

K-means Algorithm: (re-stated from a different perspective)

Initialize $\mu_1, \ldots, \mu_K$

Repeat until convergence:

$$C_1, \ldots, C_K = \arg \min_{C_1, \ldots, C_k} \ell(\{C_i\}, \{\mu_i\})$$

$$\mu_1, \ldots, \mu_K = \arg \min_{\mu_1, \ldots, \mu_k} \ell(\{C_i\}, \{\mu_i\})$$

How to pick K?

Given K, we can look at the minimum loss

$$\ell_K := \min_{C_1,\ldots,C_K,\mu_1,\ldots,\mu_K} \ell(\{C_i\},\{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

How to pick K?

Given K, we can look at the minimum loss

$$\ell_K := \min_{C_1,\ldots,C_K,\mu_1,\ldots,\mu_K} \ell(\{C_i\},\{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

Note that exactly compute the $\min$ is NP-hard, but we can approximate it w/ K-means solutions

How to pick K?

Given K, we can look at the minimum loss

$$\ell_K := \min_{C_1,\ldots,C_K,\mu_1,\ldots,\mu_K} \ell(\{C_i\},\{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

Note that exactly compute the $\min$ is NP-hard, but we can approximate it w/ K-means solutions

Q: Should we just naively pick a K that the ℓ_K is zero?

How to pick K?

Given K, we can look at the minimum loss

$$\ell_K := \min_{C_1,\ldots,C_K,\mu_1,\ldots,\mu_K} \ell(\{C_i\},\{\mu_i\}) = \sum_{i=1}^{K} \left[\sum_{x \in C_i} \|x - \mu_i\|_2^2 \right]$$

Note that exactly compute the $\min$ is NP-hard, but we can approximate it w/ K-means solutions

Q: Should we just naively pick a K that the ℓ_K is zero?

No! When $K = n$, loss is zero (every data point is a cluster!)

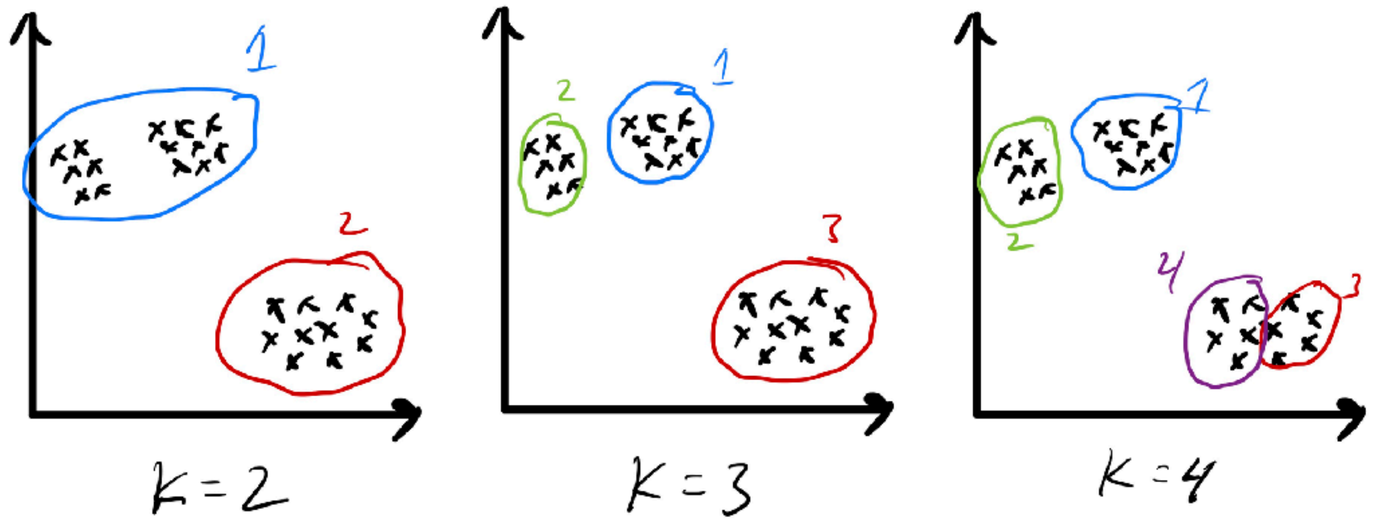
How to pick K?

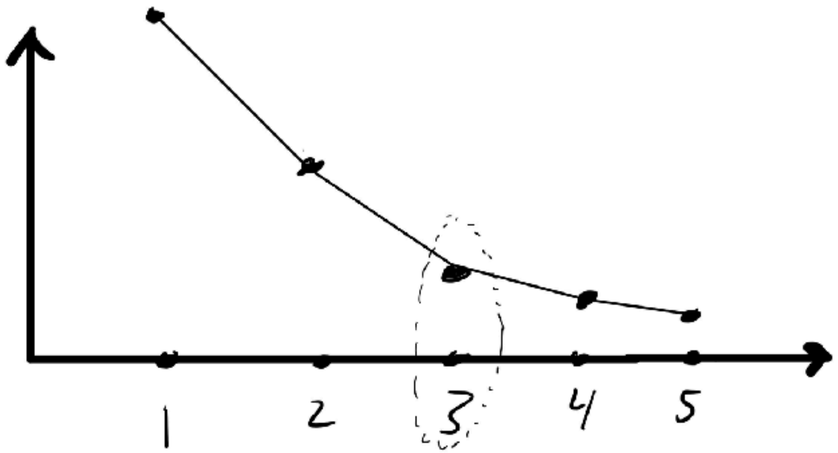
How to pick K?

In practice, we can gradually increase K, and keep track the loss ℓ_K, and stop when ℓ_K does not drop too much

How to pick K?

In practice, we can gradually increase K, and keep track the loss ℓ_K, and stop when ℓ_K does not drop too much





Summary

1. Curse of Dimensionality:

Data points in high-dim space tends to spread far from each other

2. The first Unsupervised Learning Algorithm — K means

Coordinate Descent on the loss $\ell(\{C_i\}, \{\mu_i\})$