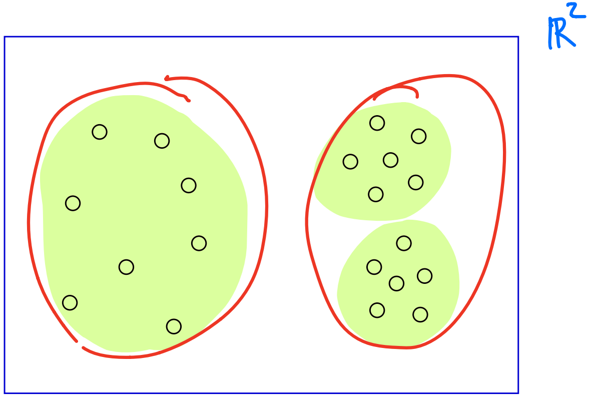
Unsupervised learning

Clustering

DSE 210

Clustering in $\mathbb{R}^d$



Two common uses of clustering:

- Vector quantization
  Find a finite set of representatives that provides good coverage of a complex, possibly infinite, high-dimensional space.
- Finding meaningful structure in data
  Finding salient grouping in data.

Widely-used clustering methods

1. K-means and its many variants
2. EM for mixtures of Gaussians
3. Agglomerative hierarchical clustering

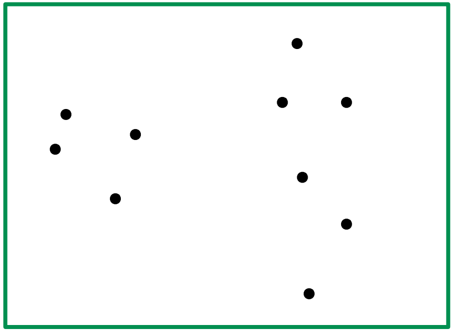
The k-means optimization problem

Vector quantization

- Input: Points $x_1, \ldots, x_n \in \mathbb{R}^d$; integer k
- Output: "Centers", or representatives, $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- Goal: Minimize average squared distance between points and their nearest representatives:

$$\text{cost}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2$$

squared distance from x_i to its closest rep.

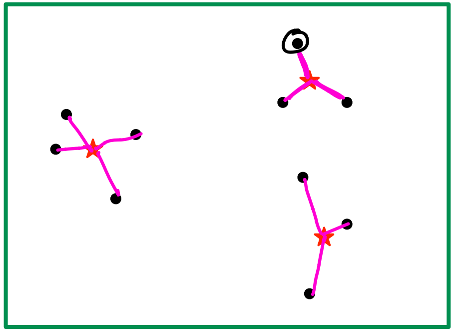


The k-means optimization problem

- Input: Points $x_1, \ldots, x_n \in \mathbb{R}^d$; integer k
- Output: "Centers", or representatives, $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- Goal: Minimize average squared distance between points and their nearest representatives:

$$\text{cost}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2$$

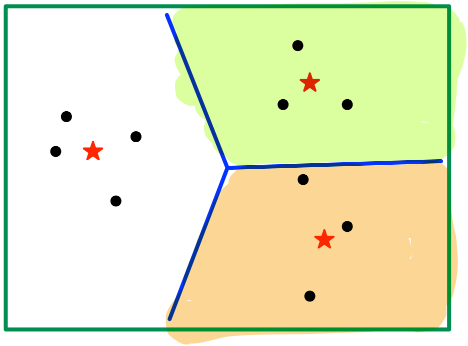
$k = 3$



The k-means optimization problem

- Input: Points $x_1, \ldots, x_n \in \mathbb{R}^d$; integer k
- Output: "Centers", or representatives, $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- Goal: Minimize average squared distance between points and their nearest representatives:

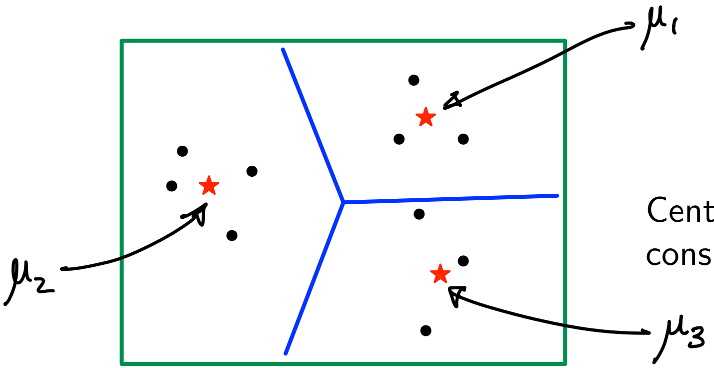
$$\text{cost}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2$$



The k-means optimization problem

- Input: Points $x_1, \ldots, x_n \in \mathbb{R}^d$; integer k
- Output: "Centers", or representatives, $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$
- Goal: Minimize average squared distance between points and their nearest representatives:

$$\text{cost}(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \|x_i - \mu_j\|^2$$

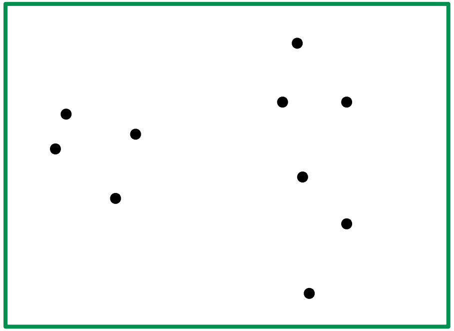


Centers carve $\mathbb{R}^d$ into k **convex** regions: μ_j's region consists of points for which it is the closest center.

Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

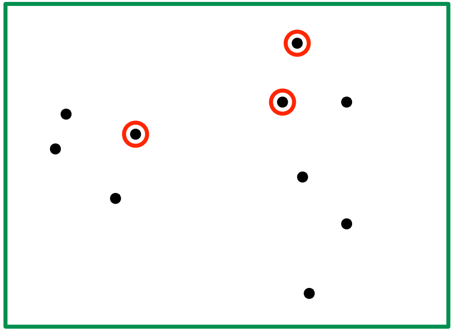
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

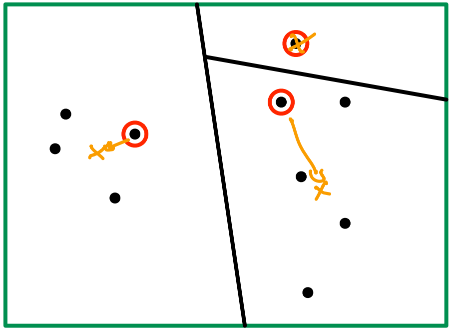
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

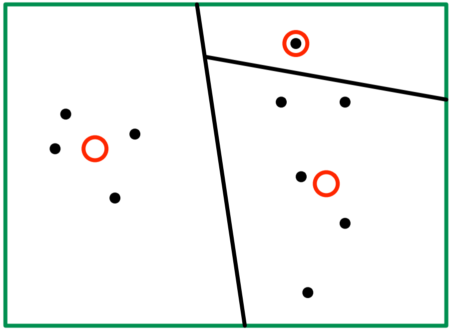
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

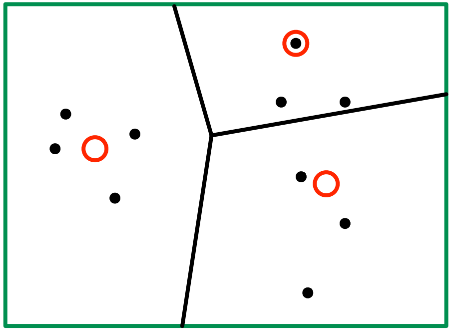
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

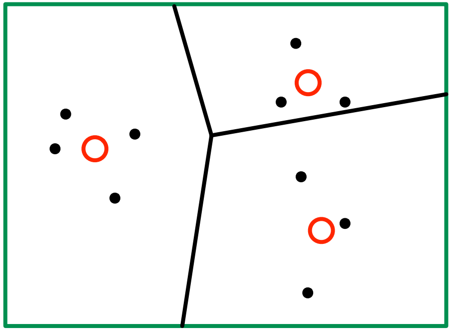
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

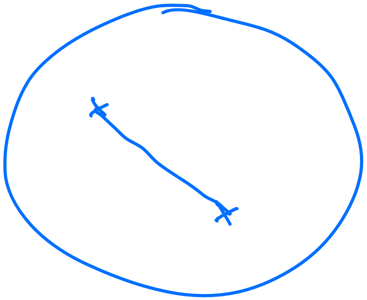
- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.

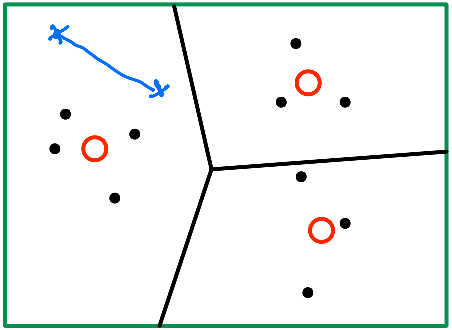


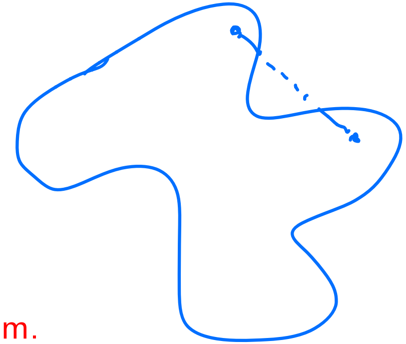
Lloyd's k-means algorithm

NP-hard optimization problem. Heuristic: "k-means algorithm".

- Initialize centers $\mu_1, \ldots, \mu_k$ in some manner.
- Repeat until convergence:
  - Assign each point to its closest center.
  - Update each μ_j to the mean of the points assigned to it.



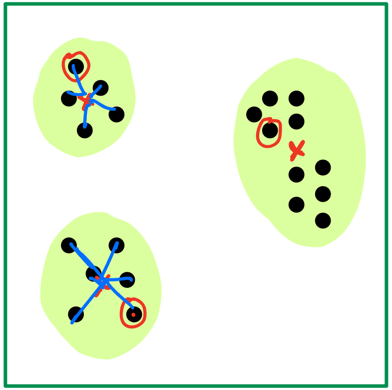


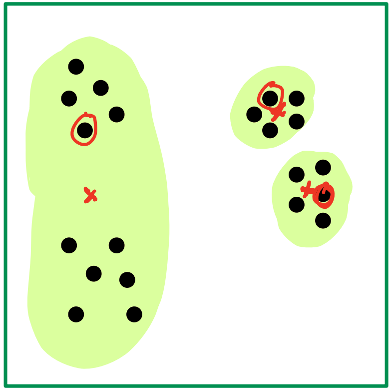


Each iteration reduces the cost $\Rightarrow$ convergence to a local optimum.

Initialization matters

Global optimum: the solution (set of "centers" $\mu_1, .., \mu_k$) with the smallest achievable cost.





Local optimum but not global optimum

Initializing the k-means algorithm

Typical practice: choose k data points at random as the initial centers.

Initializing the k-means algorithm

Typical practice: choose k data points at random as the initial centers.

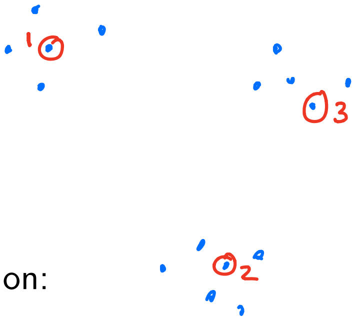
Another common trick: start with extra centers, then prune later.

Initializing the k-means algorithm

Typical practice: choose k data points at random as the initial centers.

Another common trick: start with extra centers, then prune later.

A particularly good initializer: k-**means++** (2006)



- Pick a data point x at random as the first center
- Let $C = \{x\}$ (centers chosen so far)
- Repeat until desired number of centers is attained:
  - Pick a data point x at random from the following distribution:

$$\Pr(x) \propto \text{dist}(x, C)^2,$$

where $\text{dist}(x, C) = \min_{z \in C} \|x - z\|$

  - Add x to C

Two common uses of clustering

- Vector quantization
  Find a finite set of representatives that provides good coverage of a complex, possibly infinite, high-dimensional space.

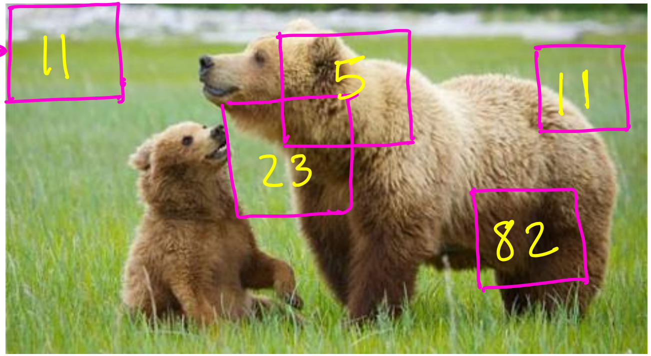
- Finding meaningful structure in data
  Finding salient grouping in data.

Representing images using k-means codewords

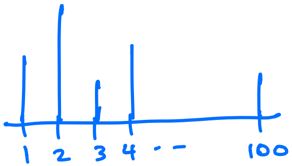
Vector quantization example

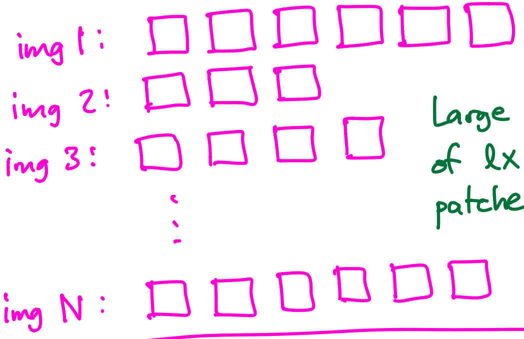
How to represent a collection of images as fixed-length vectors?

$l \times l$ patch



Represent image as a k-histogram





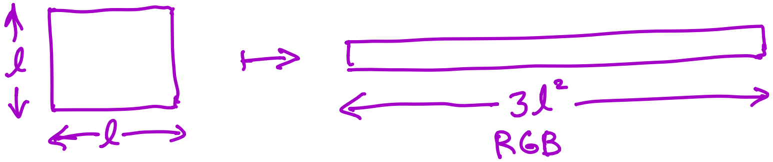
Large set of $l \times l$ patches

apply k-means to the entire collection

→ yields k "representative" patches

(eg. $k = 100$)

Each patch becomes a vector :



"Bag of patches"

Representing images using k-means codewords

How to represent a collection of images as fixed-length vectors?

- Take all $\ell \times \ell$ patches in all images. Extract features for each.
- Run k-means on this entire collection to get k centers.
- Now associate any image patch with its nearest center.
- Represent an image by a histogram over $\{1, 2, \ldots, k\}$.

Looking for natural groups in data

"Animals with attributes" data set

- 50 animals: antelope, grizzly bear, beaver, dalmatian, tiger, ...
- 85 attributes: longneck, tail, walks, swims, nocturnal, forager, desert, bush, plains, ...
- Each animal gets a score $(0 - 100)$ along each attribute
- 50 data points in $\mathbb{R}^{85}$

Apply k-means with $k = 10$ and look at grouping obtained.

Two different solutions starting from random initializations

1. zebra
2. spider monkey, gorilla, chimpanzee
3. tiger, leopard, wolf, bobcat, lion
4. hippopotamus, elephant, rhinoceros
5. killer whale, blue whale, humpback whale, seal, walrus, dolphin
6. giant panda
7. skunk, mole, hamster, squirrel, rabbit, bat, rat, weasel, mouse, raccoon
8. antelope, horse, moose, ox, sheep, giraffe, buffalo, deer, pig, cow
9. beaver, otter
10. grizzly bear, dalmatian, persian cat, german shepherd, siamese cat, fox, chihuahua, polar bear, collie

1. zebra
2. spider monkey, gorilla, chimpanzee
3. tiger, leopard, fox, wolf, bobcat, lion
4. hippopotamus, elephant, rhinoceros, buffalo, pig
5. killer whale, blue whale, humpback whale, seal, otter, walrus, dolphin
6. dalmatian, persian cat, german shepherd, siamese cat, chihuahua, giant panda, collie
7. beaver, skunk, mole, squirrel, bat, rat, weasel, mouse, raccoon
8. antelope, horse, moose, ox, sheep, giraffe, deer, cow
9. hamster, rabbit
10. grizzly bear, polar bear

Streaming and online computation

Streaming computation: for data sets too large to fit in memory.

- Make one pass (or maybe a few passes) through the data.
- On each pass:
  - See data points one at a time, in order.
  - Update models/parameters along the way.
- There is only enough space to store a tiny fraction of the data, or a perhaps short summary.

Streaming and online computation

Streaming computation: for data sets too large to fit in memory.

- Make one pass (or maybe a few passes) through the data.
- On each pass:
  - See data points one at a time, in order.
  - Update models/parameters along the way.
- There is only enough space to store a tiny fraction of the data, or a perhaps short summary.

Online computation: an even more lightweight setup, for data that is continuously being collected.

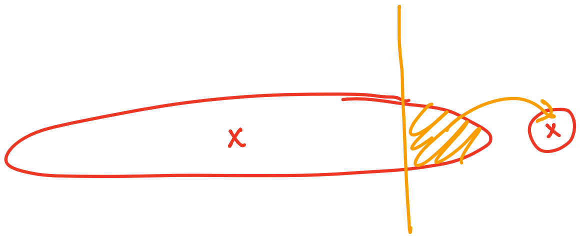
- Initialize a model.
- Repeat forever:
  - See a new data point.
  - Update model if need be.

Example: sequential k-means

1. Set the centers $\mu_1, \ldots, \mu_k$ to the first k data points
2. Set their counts to $n_1 = n_2 = \cdots = n_k = 1$
3. Repeat, possibly forever:
   - Get next data point x
   - Let μ_j be the center closest to x
   - Update μ_j and n_j:

$$\underbrace{\mu_j = \frac{n_j \mu_j + x}{n_j + 1}}_{\text{average } x \text{ into } \mu_j} \quad \text{and} \quad n_j = n_j + 1$$

K-means: the good and the bad



The good:

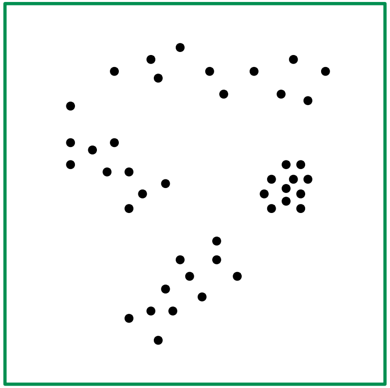
- Fast and easy.
- Effective in quantization.

The bad:

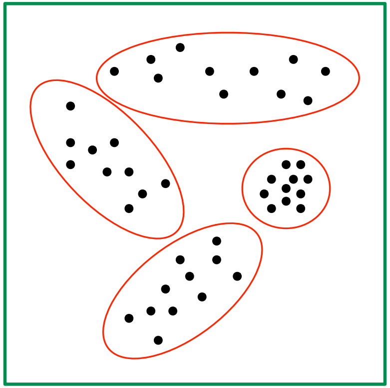
- Geared towards data in which the clusters are spherical, and of roughly the same radius.

Is there is a similarly-simple algorithm in which clusters of more general shape are accommodated?

Mixtures of Gaussians



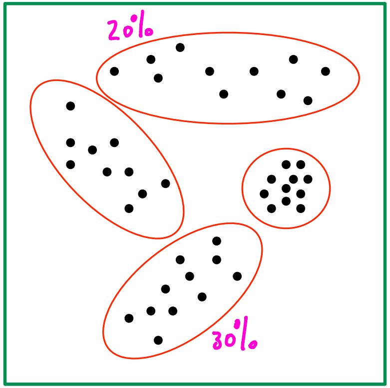
Mixtures of Gaussians



Mixtures of Gaussians

K-means: each cluster only gets a center

Now: each cluster gets its own Gaussian distribution



Each of the k clusters is specified by:

- a Gaussian distribution $P_j = N(\mu_j, \Sigma_j)$
- a mixing weight π_j

Overall distribution over $\mathbb{R}^d$: a **mixture of Gaussians**

$$\Pr(x) = \pi_1 P_1(x) + \cdots + \pi_k P_k(x)$$

The clustering task

We are given data $x_1, \ldots, x_n \in \mathbb{R}^d$.

For any mixture model $\pi_1, \ldots, \pi_k$, $P_1 = N(\mu_1, \Sigma_1), \ldots, P_k = N(\mu_k, \Sigma_k)$,

$$\begin{aligned}
&\Pr\left(\text{data} \mid \pi_1 P_1 + \cdots + \pi_k P_k\right) \\
&= \prod_{i=1}^{n} \underbrace{\left(\pi_1 P_1(x_i) + \cdots + \pi_k P_k(x_i)\right)}_{\text{probability of } x_i \text{ under the mixture model}} \\
&= \prod_{i=1}^{n} \left(\sum_{j=1}^{k} \frac{\pi_j}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2} (x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j) \right) \right)
\end{aligned}$$

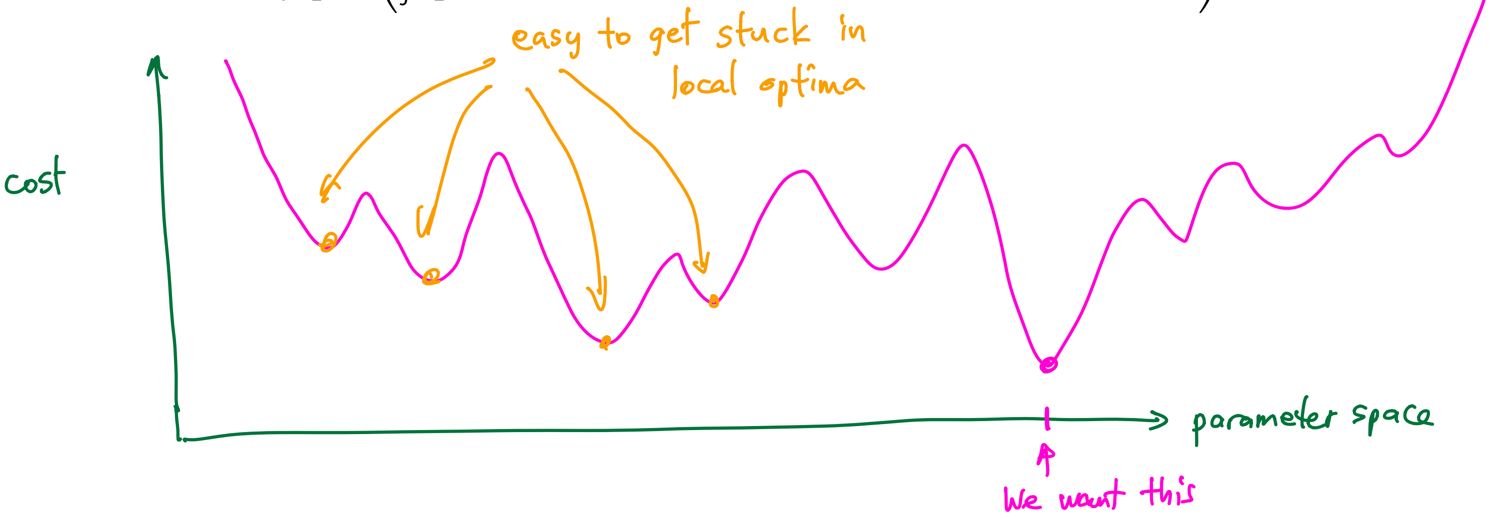
Find the **maximum-likelihood mixture of Gaussians**: parameters $\{\pi_j, \mu_j, \Sigma_j : j = 1 \ldots k\}$ maximizing this function.

complicated optimization problem

Optimization surface

Minimize the negative log-likelihood,

$$\sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} \frac{\pi_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)\right)\right)$$



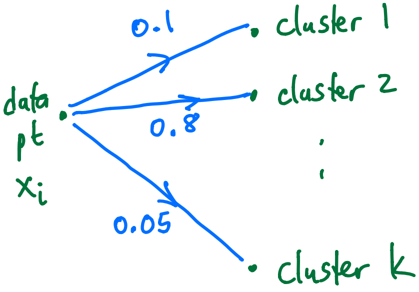
The EM algorithm

1. Initialize $\pi_1, \ldots, \pi_k$ and $P_1 = N(\mu_1, \Sigma_1), \ldots, P_k = N(\mu_k, \Sigma_k)$.
2. Repeat until convergence:
   - Assign each point x_i fractionally between the k clusters:

$$w_{ij} = \Pr(\text{cluster } j \mid x_i) = \frac{\pi_j P_j(x_i)}{\sum_\ell \pi_\ell P_\ell(x_i)}$$

   - Update mixing weights, means, and covariances:

soft assignment



$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} w_{ij}$$

$$\mu_j = \frac{1}{n\pi_j} \sum_{i=1}^{n} w_{ij} x_i$$

$$\Sigma_j = \frac{1}{n\pi_j} \sum_{i=1}^{n} w_{ij} (x_i - \mu_j)(x_i - \mu_j)^T$$

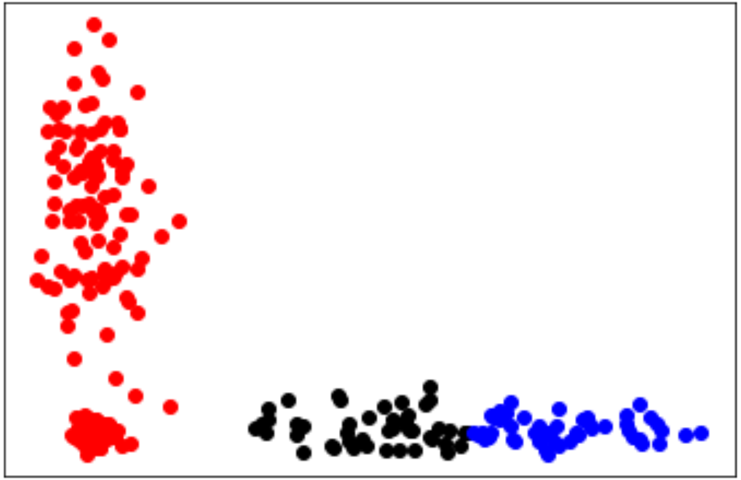
Example

Data with 3 clusters, each with 100 points.

The data set

Example

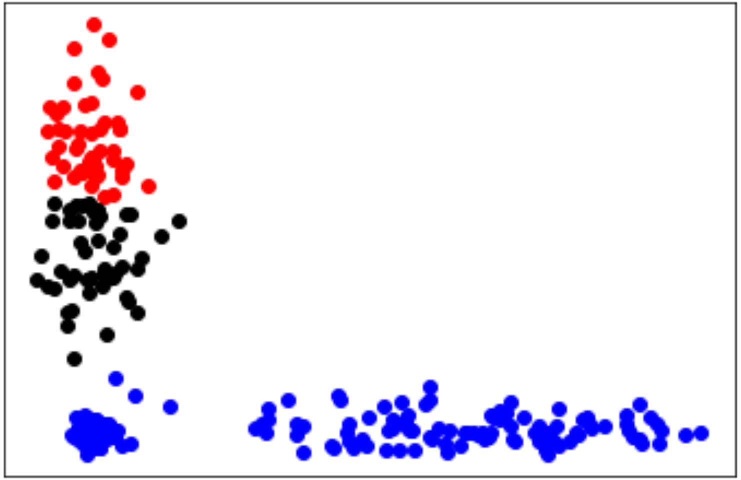
Data with 3 clusters, each with 100 points.



k-means solution 1

Example

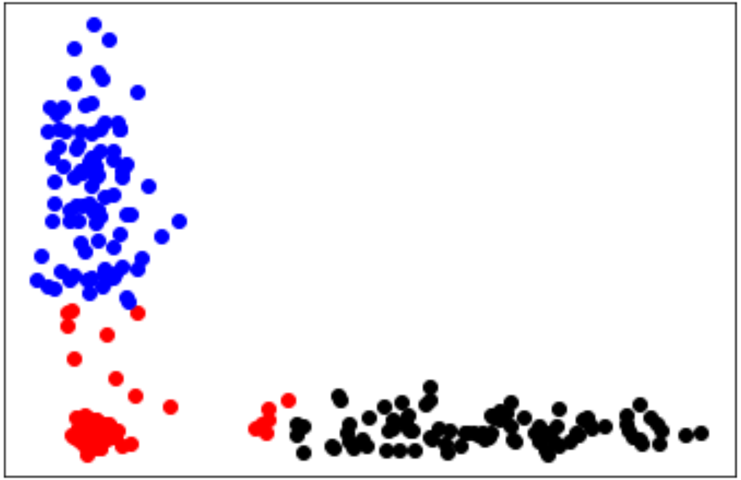
Data with 3 clusters, each with 100 points.



k-means solution 2

Example

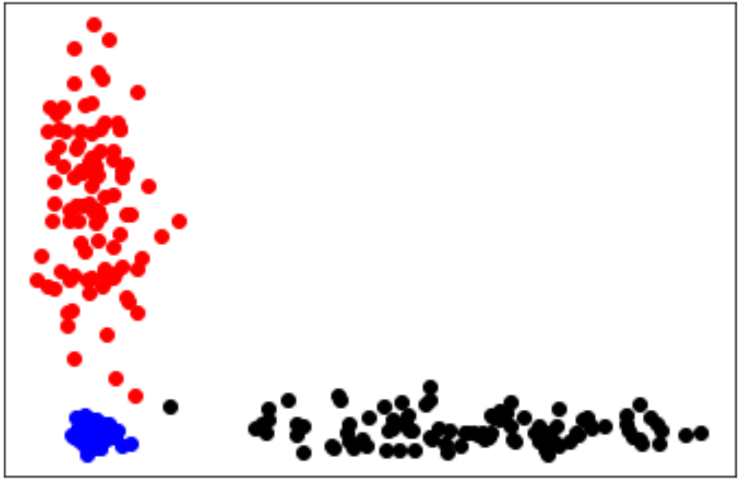
Data with 3 clusters, each with 100 points.



k-means solution 3

Example

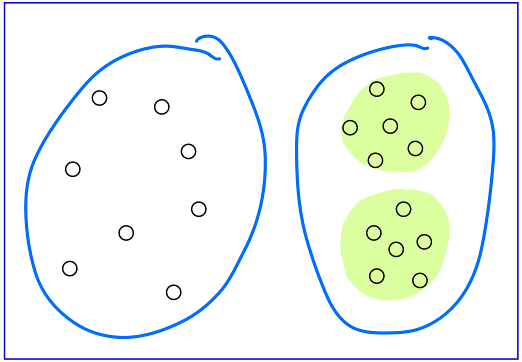
Data with 3 clusters, each with 100 points.



EM for mixture of Gaussians

Hierarchical clustering

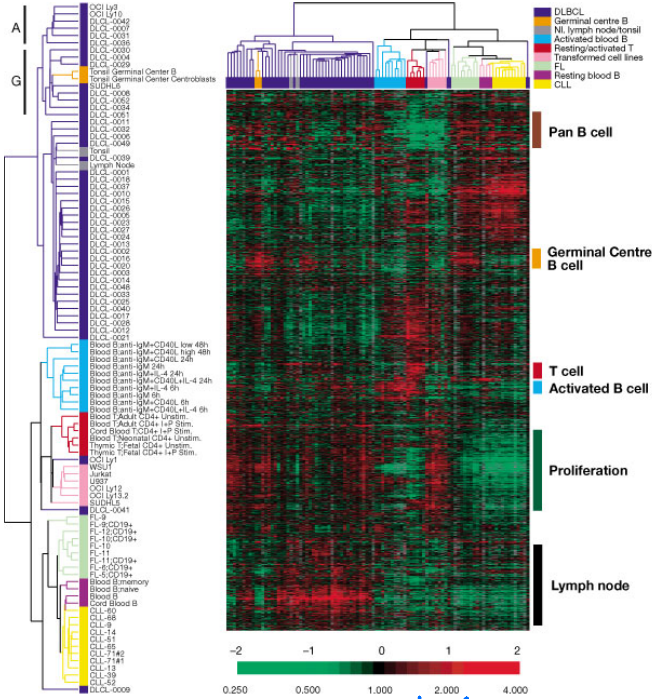
Choosing the number of clusters (k) is difficult.



Often: no single right answer, because of multiscale structure.

Hierarchical clustering avoids these problems.

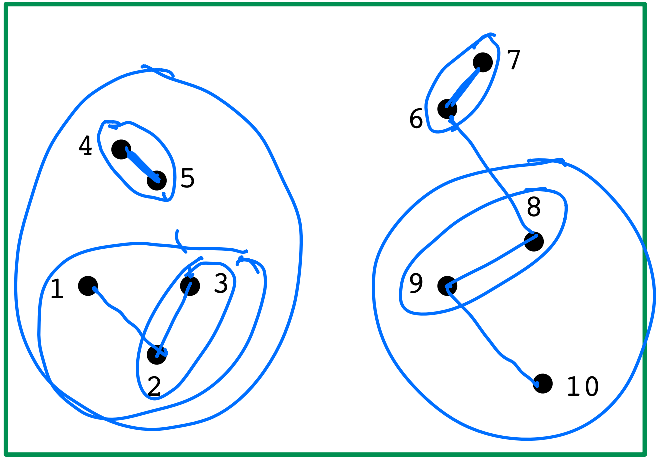
Example: gene expression data

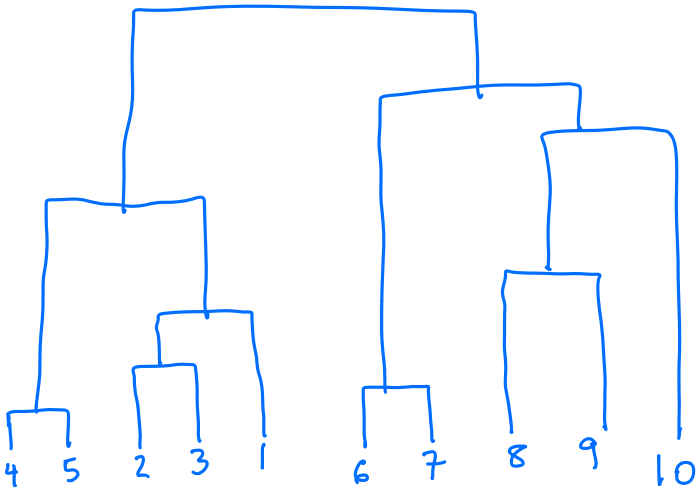


genes

samples/patients

The single linkage algorithm





- Start with each point in its own, singleton, cluster
- Repeat until there is just one cluster:
  - Merge the two clusters with the closest pair of points
- Disregard singleton clusters

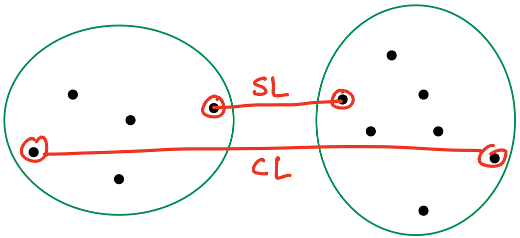
Linkage methods

- Start with each point in its own, singleton, cluster
- Repeat until there is just one cluster:
  - Merge the two “closest” clusters

Linkage methods

- Start with each point in its own, singleton, cluster
- Repeat until there is just one cluster:
    - Merge the two "closest" clusters

How to measure distance between two clusters C and C'?



- Single linkage

$$\mathrm{dist}(C, C') = \min_{x \in C, x' \in C'} \|x - x'\|$$

- Complete linkage

$$\mathrm{dist}(C, C') = \max_{x \in C, x' \in C'} \|x - x'\|$$

Average linkage

Three commonly-used variants:

1. Average pairwise distance between points in the two clusters

$$\text{dist}(C, C') = \frac{1}{|C| \cdot |C'|} \sum_{x \in C} \sum_{x' \in C'} \|x - x'\|$$

2. Distance between cluster centers

$$\text{dist}(C, C') = \|\text{mean}(C) - \text{mean}(C')\|$$

3. Ward's method: the increase in k-means cost occasioned by merging the two clusters

$$\text{dist}(C, C') = \frac{|C| \cdot |C'|}{|C| + |C'|} \|\text{mean}(C) - \text{mean}(C')\|^2$$