Topic 1: CLUSTERING

CMSC 35400/STAT 37710 Machine Learning
Risi Kondor, The University of Chicago

Clustering

In modern ML, more often than not, the inputs are high dimensional real vectors:

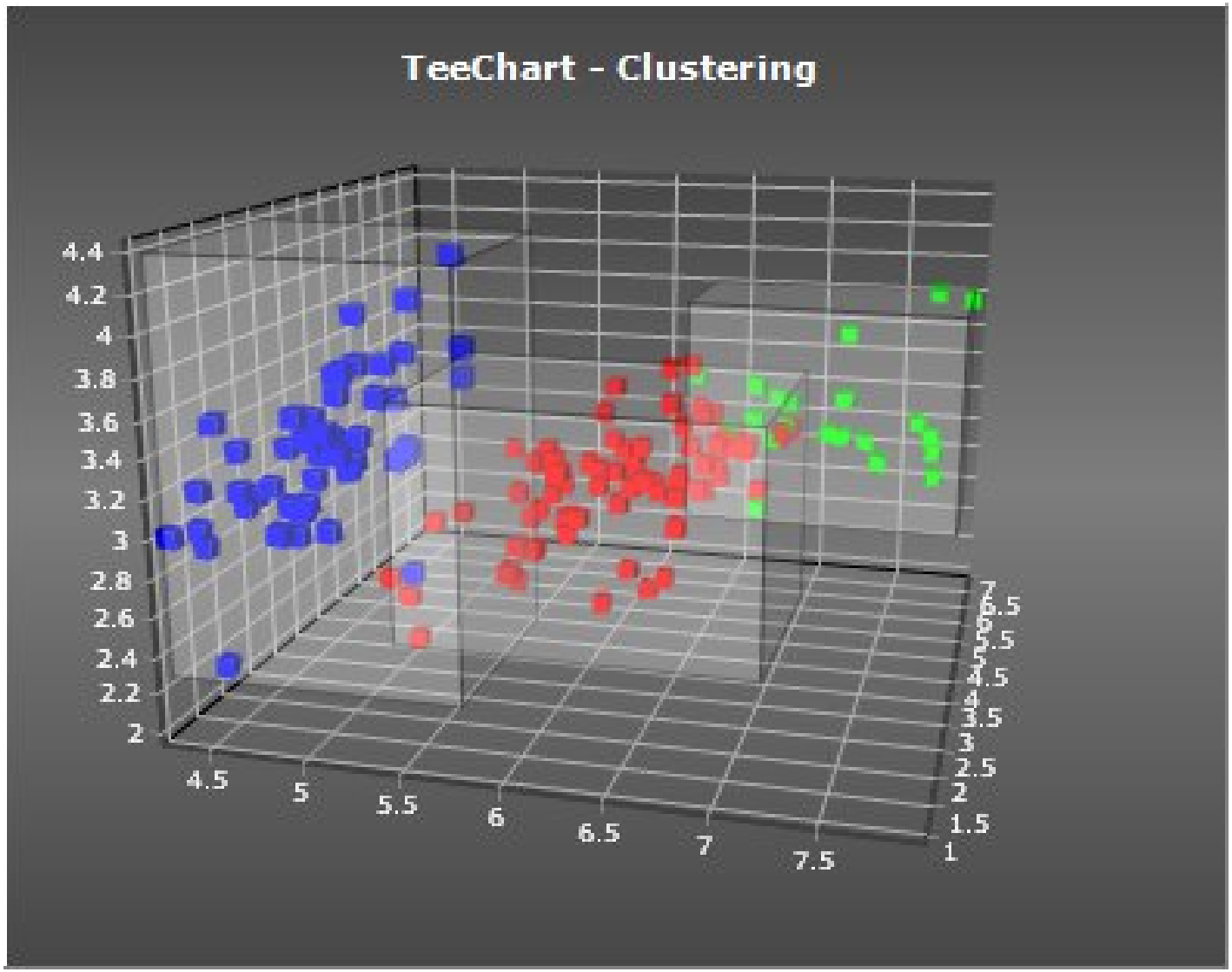
$$\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d.$$

Each x_i is called a **feature** (**covariate** in Stats).

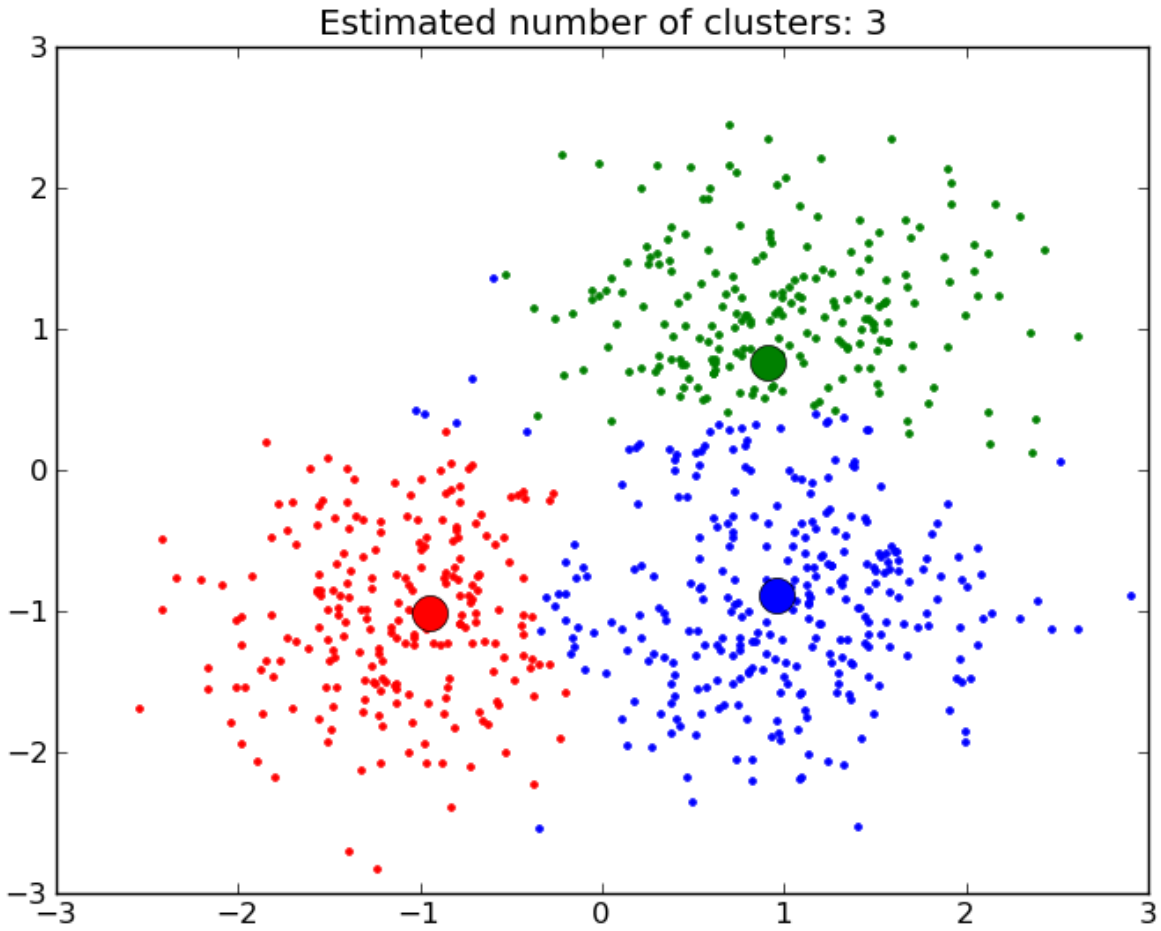
Example: $x_1 =$ age , $x_2 =$ weight , $x_3 =$ blood pressure ,...
Example: $x_i =$ intensity of a pixel i in an image

It often makes sense to ask whether a dataset $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ can be partitioned into a small number of **clusters** of similar datapoints.
$\rightarrow$ Clustering is a typical unsupervised learning problem.

Clustering

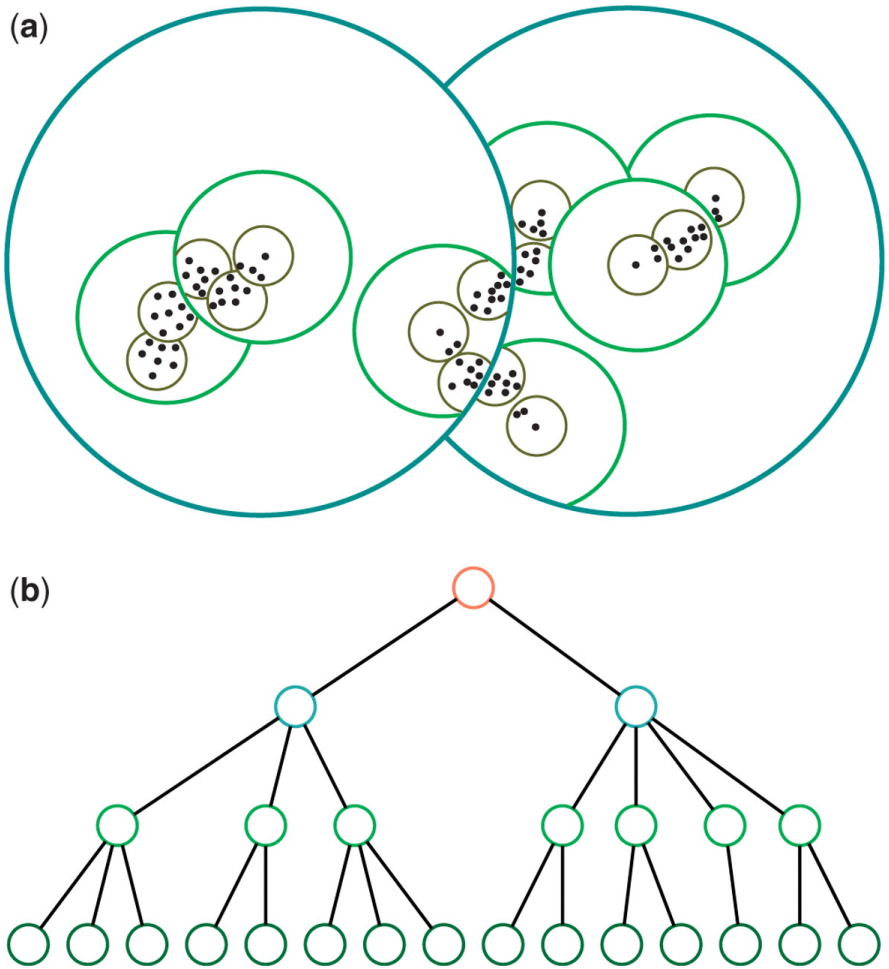

TeeChart - Clustering
4.4
4.2
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
4.5
5
5.5
6
6.5
7
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1

Clustering



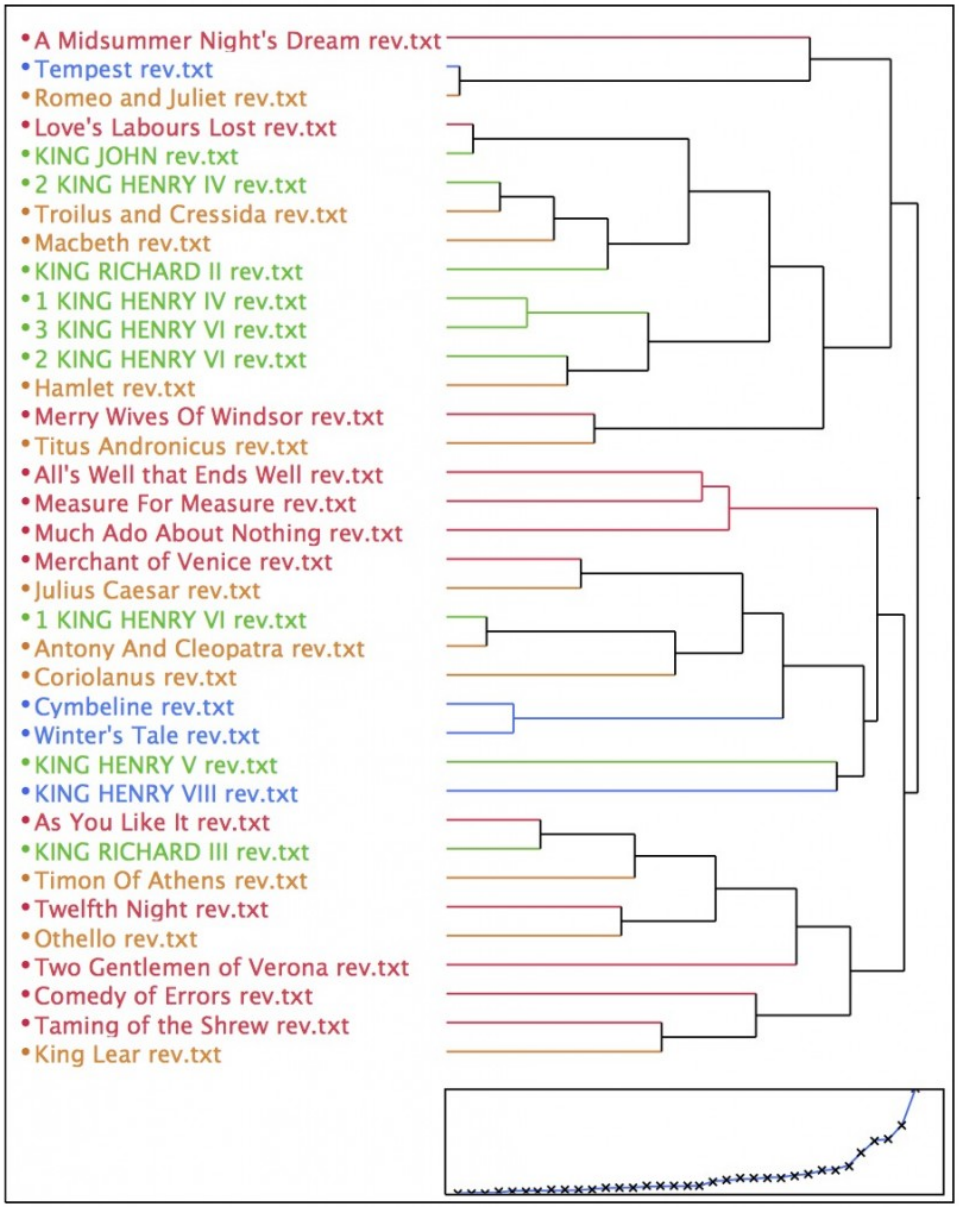
Cluster representatives indicated.

Hierarchical clustering

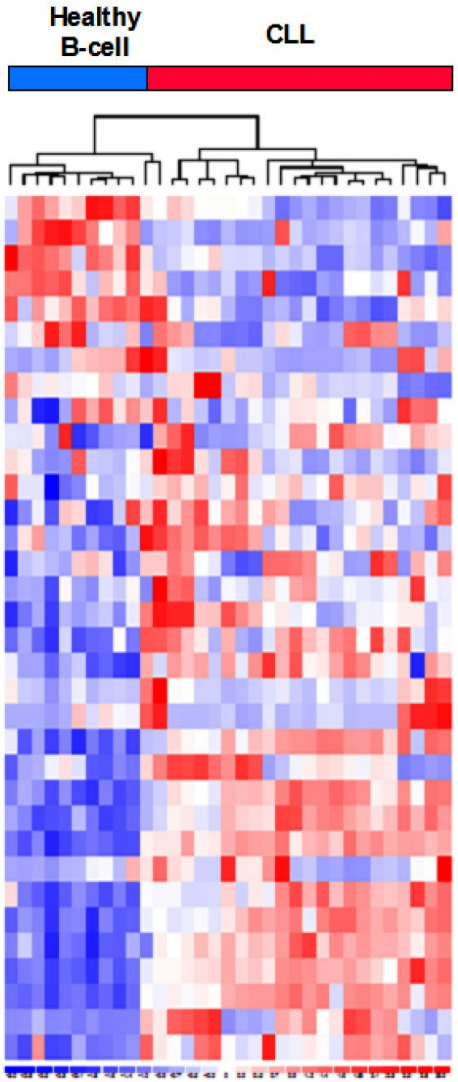


Cutting the tree at any level gives a flat clustering. Thanks to this freedom, don't have to decide the number of clusters in advance.

Hierarchical clustering



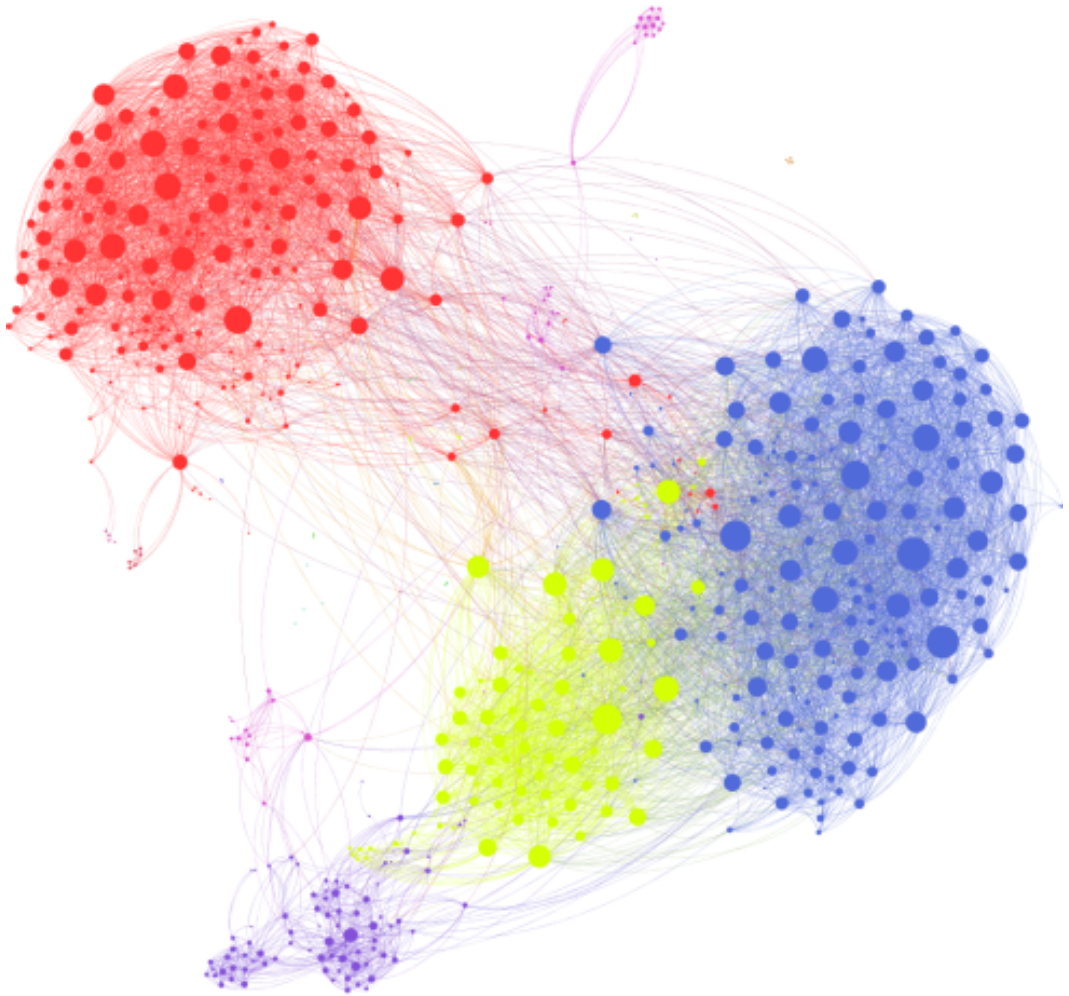
Hierarchical clustering



FLNA	filamin A, alpha
CRIP2	cysteine-rich protein 2
AKAP8	A kinase
VEGFB	vascular endothelial growth factor B
RNASEH1	ribonuclease H1
MONDOA	Mlx interactor
TRAF1	TNF receptor-associated factor 1
ZC3HAV1	zinc finger CCCH type, antiviral 1
CNK1	connector enhancer of KSR-like
TNNT3	troponin T3, skeletal, fast
LILRB3	leukocyte immunoglobulin-like receptor, subfamily B
MAPK11	mitogen-activated protein kinase 11
CLTB	clathrin, light polypeptide
SMARCD2	SWI/SNF related, subfamily d, member 2
P2RX1	purinergic receptor P2X, ligand-gated ion channel, 1
PTP4A3	protein tyrosine phosphatase type IVA, member 3
QSCN6	quiescin Q6
CDC42EP4	CDC42 effector protein
PGF	placental growth factor, vascular endothelial growth factor-related protein
CDKN1C	cyclin-dependent kinase inhibitor 1C
NFATC1	nuclear factor of activated T-cells, cytoplasmic, calcineurin-dependent 1
PLEC1	plectin 1
LSP1	lymphocyte-specific protein 1
NPTXR	neuronal pentraxin receptor
VEGFC	vascular endothelial growth factor C
CGB1	chorionic gonadotropin, beta polypeptide 1
ATF5	activating transcription factor 5
CHST1	carbohydrate
AQP1	aquaporin 1
CREB3L1	cAMP responsive element binding protein 3-like 1
MAPK11	mitogen-activated protein kinase 11
AQP1	aquaporin 1
COL9A3	collagen, type IX, alpha 3
BCL2	B-cell CLL/lymphoma 2

[Pallasch et al., Blood, 2009]

Clustering of nodes in a graph



Also known as **graph partitioning** (these are somebody's Facebook friends).

Clustering: the good

Clustering is important because

- It is a natural thing to want to do with large data.
- Can reveal a lot about the structure of data $\rightarrow$ exploratory data analysis.
  e.g., finding new types of stars, patients with similar disease profiles, ...
- Allows us to compress data by replacing points by their cluster representatives (called **vector quantization**).
- Key part of finding structure in large graphs & networks.

Clustering: the bad

- Unsupervised problem $\rightarrow$ always harder to formalize.
- Ill-defined: different objective functions possible, no clear winner. Even after we've clustered the data it's hard to say whether the clustering is good or bad $\rightarrow$ subjective.
- What is the "correct" number of clusters? Also subjective. Often data is very ambiguous in this regard.
- End users may attribute too much significance to the clusters with unforeseeable consequences.
- Compared to supervised ML, the theory is in its infancy.

Outline

1. Flat clustering: k–means
2. Hierarchical clustering: agglomerative clustering
3. Model based clustering: mixture of Gaussians

Flat clustering

Flat clustering

Input: the datapoints $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$;
the desired number of clusters $k \in \mathbb{N}$.
Output: k disjoint sets $C_1, C_2, \ldots, C_k$ whose union is $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Clustering is driven by a distance metric, d. In the simplest case it is just the Euclidean distance

$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\| = \left(\sum_{i=1}^{d} (x_i - x_i')^2\right)^{1/2}.$$

Let's assign each cluster a representative point $\boldsymbol{m}_i$. Depending on context, we might or might not require $\boldsymbol{m}_i$ to be one of the $\mathbf{x}_1, \ldots, \mathbf{x}_n$ datapoints.

Cost functions

Start with a **cost function** (in this context also called **distortion**) that our algorithm tries minimize:

- Max distance to cluster center:
$$J_{\text{max}} = \max_{i \in \{1,\ldots,k\}} \max_{\mathbf{x} \in C_i} d(\mathbf{x}, \boldsymbol{m}_i).$$
- Average distance to cluster center:
$$J_{\text{avg}} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \boldsymbol{m}_i).$$
- Average squared distance to cluster center:
$$J_{\text{avg}^2} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \boldsymbol{m}_i)^2.$$
- Sum of squared intra-cluster distances:
$$J_{\text{IC}} = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_i} d(\mathbf{x}, \mathbf{x}')^2.$$

(Prove that $J_{\text{IC}} \sim J_{\text{avg}^2}$)

The k–means algorithm

Problem: find $C_1, C_2, \ldots, C_k$ and centroids $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k \in \mathbb{R}^d$ that minimize

$$J_{\text{avg}^2} = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \boldsymbol{m}_j)^2,$$

where $d(\mathbf{x}, \boldsymbol{m}_j) = \|\mathbf{x} - \boldsymbol{m}_j\|$.

This is an **optimization problem**.

- Is it continuous? No. Is it combinatorial? No. $\rightarrow$ Mixed.
- Is it convex? No.
- How do we solve it? Alternating minimization strategy.

The k–means algorithm

Let γ_i be the cluster that $\mathbf{x}_i$ is assigned to, i.e., $C_j = \{\, \mathbf{x}_i \mid \gamma_i = j \,\}$.

- Given the $\gamma_1, \gamma_2, \ldots, \gamma_n$ cluster assignments, J_{avg^2} is minimized by setting

$$\boldsymbol{m}_j \leftarrow \frac{1}{|\, C_j \,|} \sum_{i\,:\,\gamma_i = j} \mathbf{x}_i \qquad\qquad j = 1, 2, \ldots, k.$$

- Given the $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ cluster centroids, J_{avg^2} is minimized by setting

$$\gamma_i = \operatorname*{argmin}_{j \in \{1,2,\ldots,k\}} d(\mathbf{x}, \boldsymbol{m}_j) \qquad\qquad i = 1, 2, \ldots, n.$$

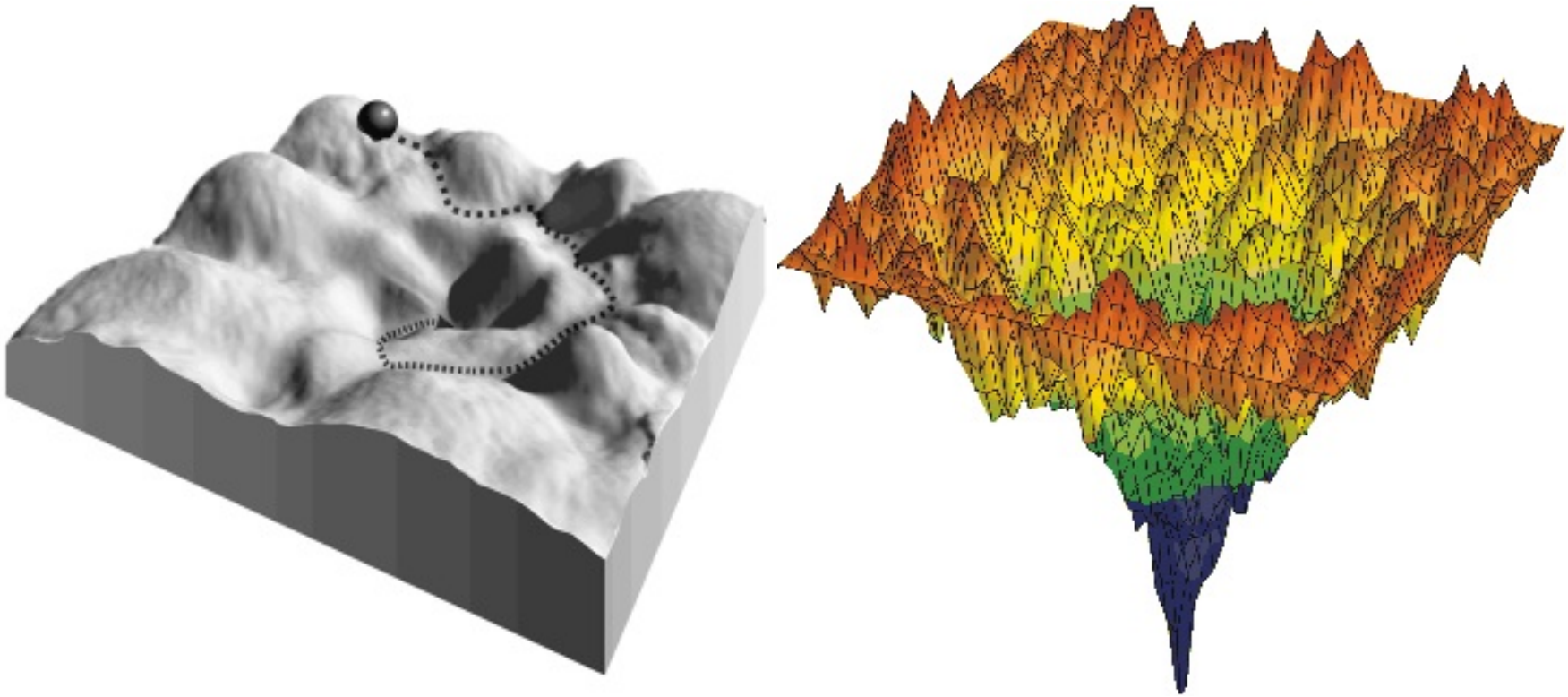
The k–means algorithm

$\{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k\} \leftarrow$ k random points in Ω ;
`while(convergence){`
  $C_1, C_2, \ldots, C_k \leftarrow \emptyset$;
  `for` $i = 1$ `to` n `{` // Assign each point to the closest center
    $\widehat{j} \leftarrow \arg\min_{j \in \{1,\ldots,k\}} d(\mathbf{x}_i, \boldsymbol{m}_j)$;
    $C_{\widehat{j}} \leftarrow C_{\widehat{j}} \cup \{\mathbf{x}_i\}$;
  `}`
  `for` $j = 1$ `to` k // Recompute cluster centers
    $\boldsymbol{m}_j \leftarrow \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$;
`}`

The k–means algorithm

- Probably the most popular clustering algorithm.
- Effectively does alternating minimization on
$$J_{\text{avg}^2} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \boldsymbol{m}_i)^2.$$
- Converges in a finite number of steps (Why?) but best upper bound is n^{kd} [Inaba et al., 1989].
- Finding the optimal clustering is NP-hard for general d (even for $k = 2$) or general k (even $d = 2$) [Dasgupta et al., 2009]
- There is no guarantee that the algorithm converges to the globally optimal solution (in most cases it won't). This is a serious problem. Often end up with some clusters only having a single datapoint. Solutions:
  - Random restarts
  - Merge clusters that are too small
  - Split clusters that are too large
  - Annealing and other methods for dealing with complicated energy surfaces
  - etc.

Local vs. global minima



Complicated energy landscapes with lots of local minimia are the bane of modern science (ML, optimization, protein folding, etc.).

k -means++

Arthur and Vassilvitskii (2007)

k-means++

choose $\boldsymbol{m}_1$ uniformly at random from $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
for($i = 2$ to k){
  choose $\boldsymbol{m}_i$ from $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ with probability

$$p(\boldsymbol{m}_i = \mathbf{x}_j) = \frac{(D_{i-1}(\mathbf{x}_j))^2}{\sum_\ell (D_{i-1}(\mathbf{x}_\ell))^2}$$

  where

$$D_{i-1}(\mathbf{x}) = \min_{p \in \{1,2,\ldots,i-1\}} \| \mathbf{x} - \boldsymbol{m}_p \| .$$

}
Run k-means initialized with $(\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k)$ as usual

k-means++

Theorem [Arthur and Vassilvitskii (2007)] Let $\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k$ be the initial cluster centers returned by the k-means++ initialization procedure. Then

$$\mathbb{E}[J_{\text{avg}^2}(\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k)] \leq 8(\ln k + 2) J^*_{\text{avg}^2},$$

where $J^*_{\text{avg}^2}$ is the minimum of J_{avg^2} over all possible clusterings.

Hierarchical clustering

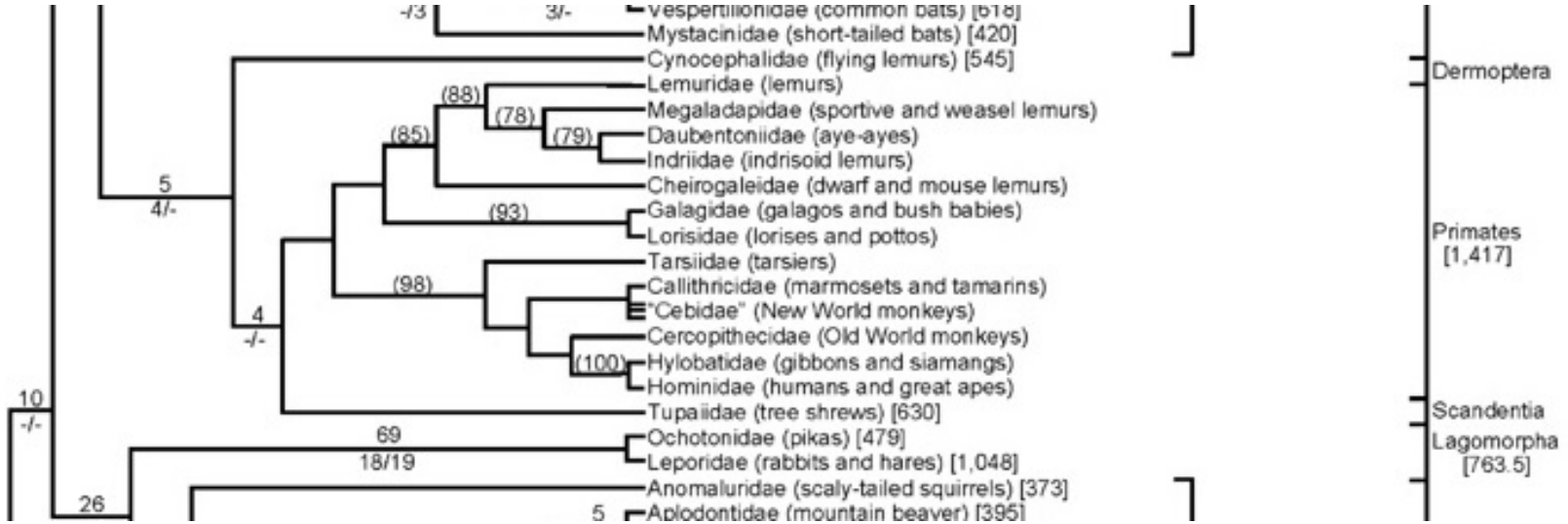
Hierarchical clustering

Input: the datapoints $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$

Output: a clustering tree (**dendrogram**)

Advantages: Don't need to decide number of clusters in advance

Hierarchical structure is often very informative



Hierarchical clustering

- **Agglomerative:** start with n clusters containing one datapoint each, and then merge clusters pairwise until only one cluster is left.
- **Divisive:** Start with a single cluster containing all the datapoints and then split it into smaller and smaller clusters. $\rightarrow$ Recurisvely clustering clusters into smaller clusters.

Merging criteria for agglomerative

Agglomerative algorithms always merge the pair of clusters closest to each other according to some distance measure:

- Single linkage: $d(C_i, C_j) = \min_{\mathbf{x} \in C_i,\, \mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')$
  $\rightarrow$ tends to generate long "chains"
- Complete linkage: $d(C_i, C_j) = \max_{\mathbf{x} \in C_i,\, \mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')$
  $\rightarrow$ tends to generate compact "round" clusters, k–center cost
- Average linkage:
  - $d(C_i, C_j) = \frac{1}{|\, C_i \,|} \frac{1}{|\, C_j \,|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_j} d(\mathbf{x}_i, \mathbf{x}_j)$
  - Ward's method $\rightarrow$ k–means cost of resulting clustering

Agglomerative clustering algorithm

$\mathcal{C} \leftarrow \emptyset$;
`for` $i = 1$ `to` n
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\{\mathbf{x}_i\}\}$; // At first each point has its own cluster
`while` $(|\mathcal{C}| > 1)$ `{`
  `find` the pair of clusters $C_1, C_2 \in \mathcal{C}$ for which $d(C_1, C_2)$ is smallest ;
  $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$; // Merge C_1 and C_2
`}`

Model based clustering (flat case)

Model based clustering

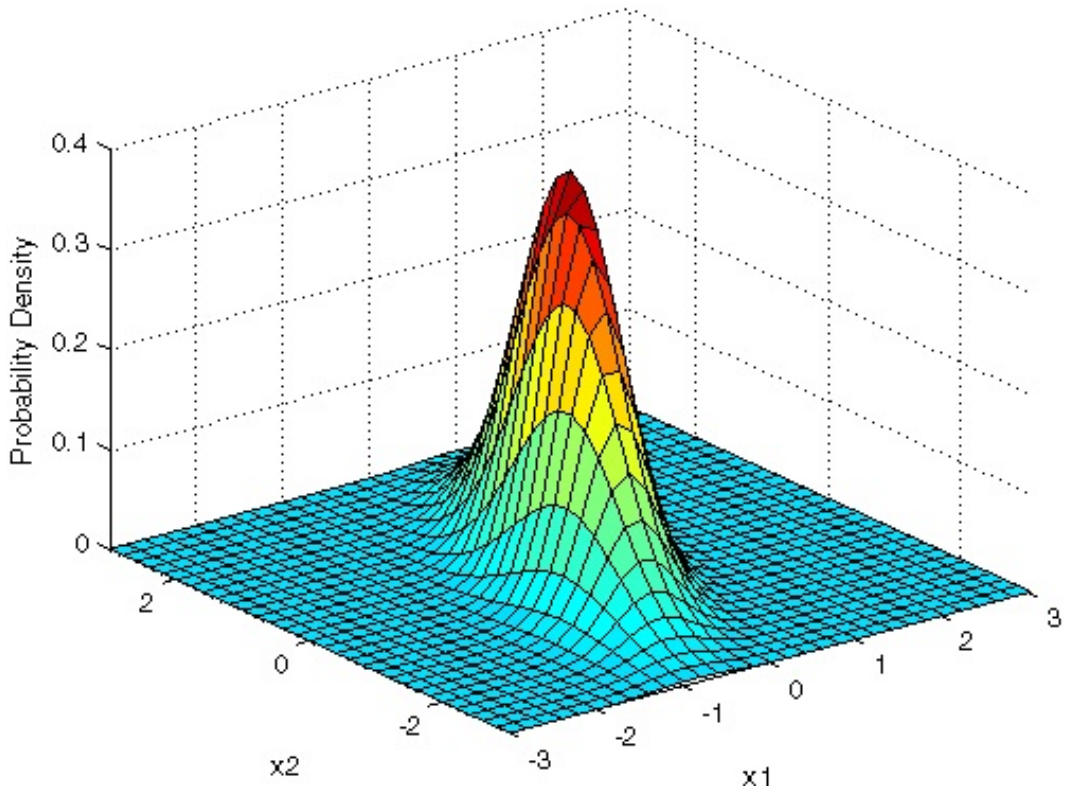
- Regard each datapoint as consisting of two random quantities (**random variables**):
  - $\boldsymbol{X}_i \in \mathbb{R}^d$: the location of the i'th datapoint $\rightarrow$ **observed**
  - $Z_i \in \{1, \ldots, k\}$: the cluster assignment of the i'th datapoint $\rightarrow$ **hidden**
- Assume that each $(\mathbf{x}_i, z_i)$ pair is drawn independently from some probability distribution (model) with parameters θ :

$$(\mathbf{x}_i, z_i) \sim p_\theta.$$

Here θ can be any bunch of parameters, depends on the model.

The probability distribution p_θ is said to **generate** the data. $\rightarrow$ **generative modeling** (typical Bayesian idea)

The multivariate Gaussian (Normal)



$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \,|\, \boldsymbol{\Sigma} \,|}} \, e^{-(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})/2} := \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Mixture of Gaussians model

The most common generative model for clustering is a mixture of k Gaussians:

$$p_\theta(\mathbf{x}, z) = \pi_z \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

where $\pi_1, \ldots, \pi_k \geq 0$ and $\sum_{j=1}^{k} \pi_j = 1$. The **marginal** of $\mathbf{x}$ is

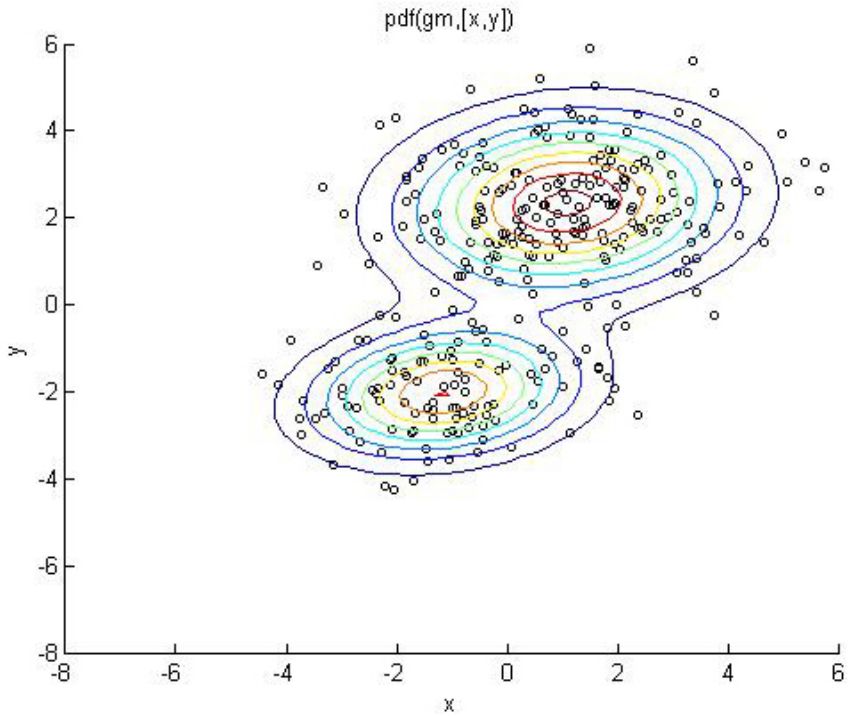
$$p(\mathbf{x}) = \sum_{i=1}^{k} \pi_i \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

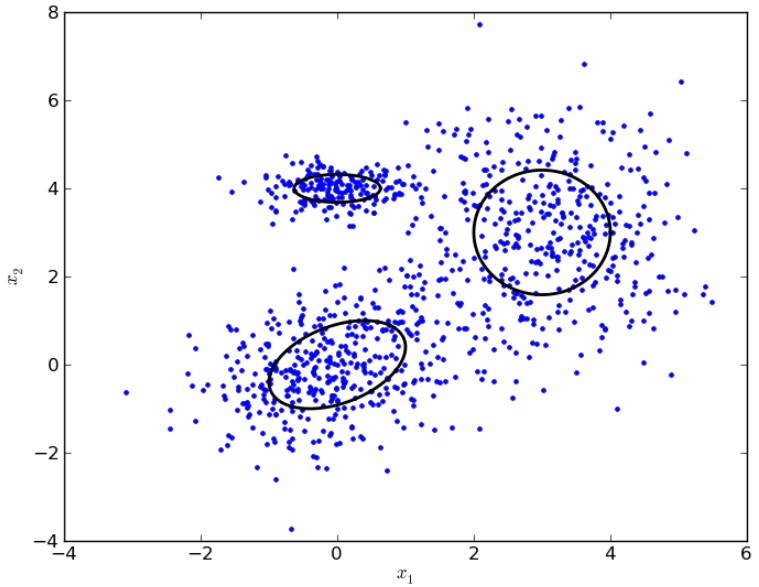
(this is why it is called a mixture).

The parameters $\theta = ((\pi_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \ldots, (\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))$ are:

- $\pi_z \in [0, 1]$: the prior probability of a new point coming from cluster z
- $\boldsymbol{\mu}_z \in \mathbb{R}^d$: the center of the z'th Gaussian
- $\boldsymbol{\Sigma}_z \in \mathbb{R}^{d \times d}$: the covariance matrix of the z'th Gaussian

Mixture of Gaussians





Big advantage: can capture clusters of different sizes and orientations.

But how do we find the parameters? $\rightarrow$ statistical estimation.

Sampling: model $\rightarrow$ data

It is easy to draw a new datapoint $(\mathbf{x}, z)$ from a mixture model like

$$p_\theta(\mathbf{x}, z) = \pi_z \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z).$$

1. First draw the cluster assignment variable, z from the discrete distribution

$$p(z) = \pi_z.$$

2. Then, given the cluster assignment z, draw the location variable $\mathbf{x}$ from

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z).$$

Estimation: data $\rightarrow$ parameters

Fitting a probabilistic model like

$$p_\theta(\mathbf{x}, z) = \pi_z \, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$$

means **estimating** the parameters $(\pi_1, \ldots, \pi_k, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k)$ from the data (called the sample) $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

Statistics is all about deriving such estimators. However, *there is no single best estimator*:

- **Probability:** $\theta \rightarrow \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ — stochastic, but well defined
- **Statistics:** $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \rightarrow \theta$ — not well defined.

Likelihood

Given a statistical model $p_\theta(x)$, the **likelihood** of θ given a sample (i.e., training set) $\{x_1, \ldots, x_n\}$ is

$$L_{(x_1,\ldots,x_n)}(\theta) = p_\theta(x_1, x_2, \ldots, x_n).$$

Usually drop $(x_1, \ldots, x_n)$ from the subscript.

It is important to understand the distinction between $L(\theta)$ and $p_\theta(x_1, \ldots, x_n)$. In particular, L is not invariant to reparametrizing x.

Maximum likelihood

The simplest type of statistical estimator is the **maximum likelihoood estimator (MLE)**:

$$\widehat{\theta} = \operatorname*{argmax}_{\theta} L(\theta).$$

In practice, often it is even easier to maximize the **log-likelihood**, $\ell(\theta) = \log L(\theta)$, especially because for IID data

$$\ell_{\{x_1,x_2,\ldots,x_n\}}(\theta) = \ell_{x_1}(\theta) + \ell_{x_2}(\theta) + \ldots + \ell_{x_n}(\theta).$$

Question: What are the advantages and drawbacks of the maximum likelihood principle?

MLE for one cluster model

If we had only one cluster, the model would be

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}\right|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2}.$$

It is easy to see that the MLE in this case is

$$\widehat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i \qquad \widehat{\boldsymbol{\Sigma}} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \widehat{\boldsymbol{\mu}})(\mathbf{x}_i - \widehat{\boldsymbol{\mu}})^\top.$$

Expectation maximization

But what about a model $p_\theta(x, z)$ where

- x is **observed**
- z is **unobserved** (**latent**)?

Can't compute the likelihood if some of the variables are not observed!

The idea of the **EM–algorithm** is to take the average of the log-likelihood over possible values of z, i.e., compute an **expected log-likelihood** $\overline{\ell}_{\widehat{\theta}_{\text{old}}}(\theta)$ (w.r.t. the old parameters $\widehat{\theta}_{\text{old}}$) and maximize that.

Need to iterate this until convergence.

Expectation maximization

1. **E-step:** Compute the *expected* log-likelihood (w.r.t. the hidden variables) under $\widehat{\theta}_{\text{old}}$
$$\overline{\ell}_{\widehat{\theta}_{\text{old}}}(\theta).$$
2. **M-step:** Maximize this to get the new estimate for $\widehat{\theta}$:
$$\widehat{\theta} = \operatorname*{argmax}_{\theta} \overline{\ell}_{\widehat{\theta}_{\text{old}}}(\theta).$$

Whether or not this is viable for a complicated model is not obvious.

EM for mixture of Gaussians

Starting from random settings, iterate the following two steps:

- **"E-step":** Given the $\boldsymbol{\mu}_j$'s and $\boldsymbol{\Sigma}_j$'s update the assignments

$$p_{i,j} = p(\mathbf{x}_i \text{ belongs to cluster } j) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{j'} \pi_{j'} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_{j'}, \boldsymbol{\Sigma}_{j'})}$$

- **"M-step":** Given the assignments, update $\boldsymbol{\pi}$ and the $\boldsymbol{\mu}_i$'s and Σ_i's

$$\pi_j \leftarrow \frac{1}{n} \sum_{i=1}^{n} p_{i,j} \qquad \boldsymbol{\mu}_j \leftarrow \frac{\sum_{i=1}^{n} p_{i,j} \mathbf{x}_i}{\sum_{i=1}^{n} p_{i,j}}$$

$$\boldsymbol{\Sigma}_j \leftarrow \frac{\sum_{i=1}^{n} p_{i,j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^{n} p_{i,j}}$$