# Naive Bayes

Suppose categories indexed by c, and features represented in a vector **x**.

Assume features in **x** are independent given the category.

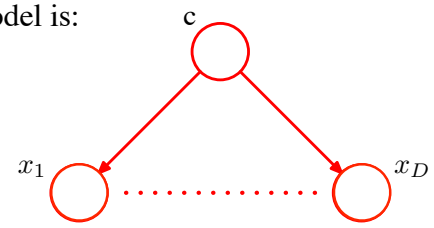
(Feature independence is the "naive" part).

$$p(\mathbf{x}|c) = \prod_i p(x_i|c)$$

# Naive Bayes

$$p(\mathbf{x},c) = \mathrm{p}(c)\prod_i p(x_i|c)$$

Graphical model is:




# Naive Bayes

$$p(\mathbf{x},c) = \mathrm{p}(c)\prod_i p(x_i|c)$$

Note that:

The forms of $p(x_i|c)$ need not all be the same (but often are)

If $p(\mathbf{x}|c)$ is a Gaussian, then it has diagonal covariance matrix. (This simplifying assumption is nearly always needed with Gaussians if the dimension, D, is large).

# Naive Bayes

Typically, $p(x_i|c)$ come from training data linked to known (labeled) classes (supervised learning).

Example (1) fit a univariate Gaussian to each variable, $x_i$, for each class, c.

Example (2), record a histrogram for each variable, $x_i$, for each class, c.

## Inference using Naive Bayes

$p(\mathbf{x}|c) = \prod_i p(x_i|c)$ (forward model)

$p(c|\mathbf{x}) \propto p(\mathbf{x}|c)p(c)$ (the Bayes part)

This leaves us with simple, and often very effective model and associated inference. We combine the likelihood $p(\mathbf{x}|c)$ with the prior $p(c)$ over categories.

## Naive Bayes for face identification

- Example features
  - Location, color, texture, of left eye
  - Location, color, texture, of right eye
  - Location, color, texture, of mouth
  - Location, color, texture, of nose
- We can imagine training these with different facial expressions, lighting conditions, etc.
- Notice that these are not independent.
- This sort of thing often works pretty well anyway.
- Possible explanation is that, while the model allows for the eyes to be different, this rarely occurs in training or testing data.

## Clustering

Clustering is the canonical case of "unsupervised" learning.

Given the data, what are the categories (clusters), c?

(Given a cluster, the features might be independent like Naive Bayes, or they might not be).

We will focus on clustering based on statistical models, but first review clustering in general.

## Why is clustering hard?

**Main reason**

- The number of possible clusterings is exponential in the number of data points

**Other important issues**

- The number of clusters (and a good way to check) is usually **not** known
- A good distance function between points may not be known
- A good model explaining the existence of clusters may not be known.
- High dimensionality

# Clustering based on distance measure

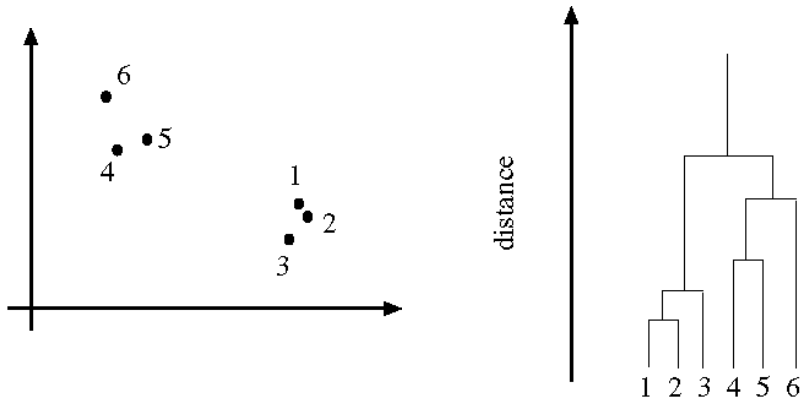
- Most common data representation is an N dimensional "feature" vector.
- Most common distance is Euclidian distance.
- Be careful with scaling and units!
- Probabilistic models can finesse scaling and multiple modalities
- Problems with correlated variables can be mitigated using transformations and data reduction methods such as PCA, ICA.

# Clustering approaches

- Agglomerative clustering
  - initialize: every item is a cluster
  - attach item that is "closest" to a cluster to that cluster
  - repeat
- Divisive clustering
  - split cluster along best boundary
  - repeat recursively
- Probabilistic clustering
  - define a probabilistic grouping model
  - use statistical inference to fit the model

# Simple agglomerative approaches

- Point-Cluster or Cluster-Cluster distance
  - single-link clustering (minimum distance from point to points in clusters or among pairs of points, one from each cluster)
  - complete-link clustering (maximum)
  - group-average clustering (average)
  - (terms are not important, but concepts are worth thinking about)
- Dendrograms
  - classic picture of output as clustering process continues



# K-Means

- Choose a fixed number of clusters ("K")
- Choose cluster centers (**means**) and point-cluster allocations (membership) to minimize the error

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2 \right\}$$

- **x**'s could be any set of features for which we can compute a distance (careful with scaling)

# K-Means

- Want to minimize

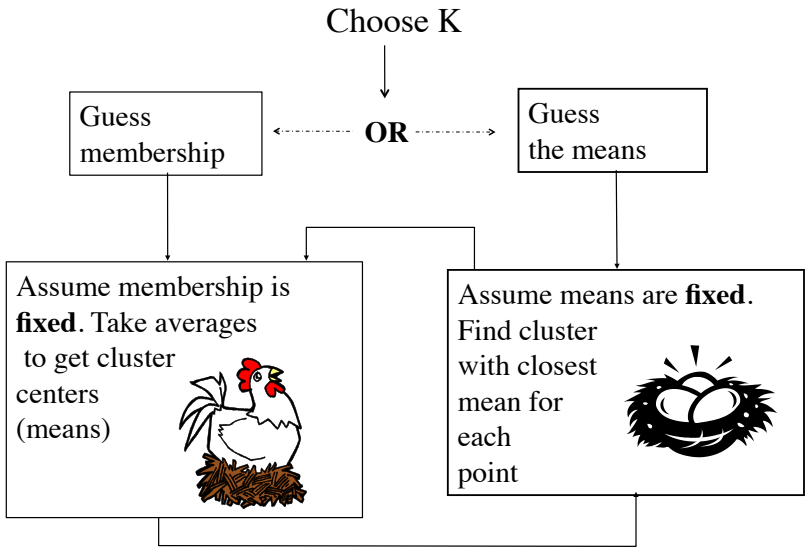
$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2 \right\}$$

- **Cannot** do this optimization by search, because there are too many possible allocations.
- Standard difficulty which we handle with an iterative process (chicken and egg)

# K-Means algorithm (intuition)

- If we know the cluster centers, the best cluster for each point is easy to compute
  - Just compute the distance to each to find the closest
- If we know the best cluster for each point, the cluster centers are also easy to compute
  - Just average the points in each cluster
- Algorithm
  - 1) Guess one of the two.
  - 2) Alternatively re-compute the values for each

K-means flow chart



# Notes on K-Means

- K-means is "hard" clustering. This means that each point is completely in exactly one cluster
- What you get is a function of starting "guess"
- The error goes down with every iteration
  - This means you get a local minimum, but not necessarily a global one.
- Unfortunately, the dimension of the space is usually large, and high-dimensional space has lots of room for local maximum (standard problem!)
  - Dimensionality here is K*dim(**x**)
- Finding the global minimum for a real problem is very optimistic!

you should be able to argue why this is true

# Clustering using a generative statistical model

Associate each cluster with (usually) the same model type, but with different parameters.

Example (Gaussian Mixture Model (GMM)),

$$p(\mathbf{x}|c) = \mathrm{N}(\mathrm{u}_c, \Sigma_c)$$

or, assuming feature independence,

$$p(\mathbf{x}|c) = \mathrm{N}(\mathrm{u}_c, \sigma_c^2)$$

$p(\mathbf{x}|c)$ could also be a product of independent multinomials, or, even a product of different distributions (roll your own!).

# Clustering using a generative statistical model

These models are quite straight-forward to apply if we know the parameters.

In addition, establishing the model parameters is usually easy if we know the correspondence (e.g., we have labeled data).

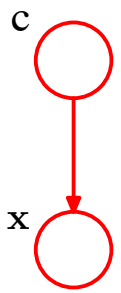
(We have already seen both these case with Naive Bayes).

However, "clustering" implies learning the model without knowing the correspondence.

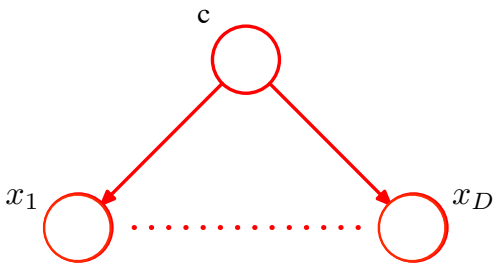
Doing this is a new kind of inference (missing value problem) that is different from max-sum and max-product.

# Clustering using a generative statistical model

Graphical model




(and for independent features)




(We saw this one when we discussed Naive Bayes)

## Inference given a clustering

Given a learned clustering model (either supervised or unsupervised), we can compute a posterior probability of which cluster an instance belongs to.

$$p(c|\mathbf{x}) \propto p(\mathbf{x}|c)p(c)$$

Easily normalized since the number of clusters is finite:

$$p(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)p(c)}{\sum_c p(\mathbf{x}|c)p(c)}$$

## Clustering models representing data statistics

What is the distribution of data best described by clusters? (Example, data coming from a bimodal distribution?)

$$p(\mathbf{x}) = \sum_c p(\mathbf{x}, c)$$
$$= \sum_c p(c)p(\mathbf{x}|c)$$

Generative story:

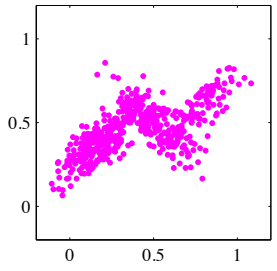
1) choose a cluster with probability, $p(c)$.
2) sample from $p(\mathbf{x}|c)$.
3) rinse and repeat.

## Clustering models representing data statistics

Distribution of data described by clusters.

$$p(\mathbf{x}) = \sum_c p(c)p(\mathbf{x}|c)$$

Distribution modeled by 3 multivariate Gaussians.




Even if we know the exact model, we cannot be sure from the data which point comes from which cluster. We only have the distribution for this.

## Learning the parameters from data

For concreteness, assume GMM

Assume K clusters

The goal is to learn mixing coefficients, $p(c)$, and cluster parameters for $p(\mathbf{x}|c)$ for all K clusters indexed by $c$.

## Learning the parameters from data

The goal is to learn mixing coefficients, $p(c)$, and cluster parameters for $p(\mathbf{x}|c)$ for all K clusters indexed by $c$.

From previous arguments, given $p(\mathbf{x}|c)$, we know the distribution over clusters for each data point.

We simultaneously cluster points and learn the cluster model.

## Learning the parameters from data

$$p(\mathbf{x}_i|\theta) = \sum_c p(c)\, p(\mathbf{x}_i|c,\theta_c)$$

Probability of all observed data will be the objective function. It is:

$$p(\{\mathbf{x}_i\}|\theta) = \prod_i \left( \sum_c p(c)\, p(\mathbf{x}_i|c,\theta_c) \right) \qquad \text{(want this to be large)}$$

or

$$\sum_i \log \left( \sum_c p(c)\, p(\mathbf{x}_i|c,\theta_c) \right) \qquad \text{(should be large)}$$

## Expectation Maximization (EM)

Operationally this is similar to K-means.

Observe that:

If we knew the cluster assignments, we could estimate the parameters for $p(\mathbf{x}|c)$.

If we knew $p(\mathbf{x}|c)$, we could make cluster assignments by computing the distribution $p(c|\mathbf{x})$

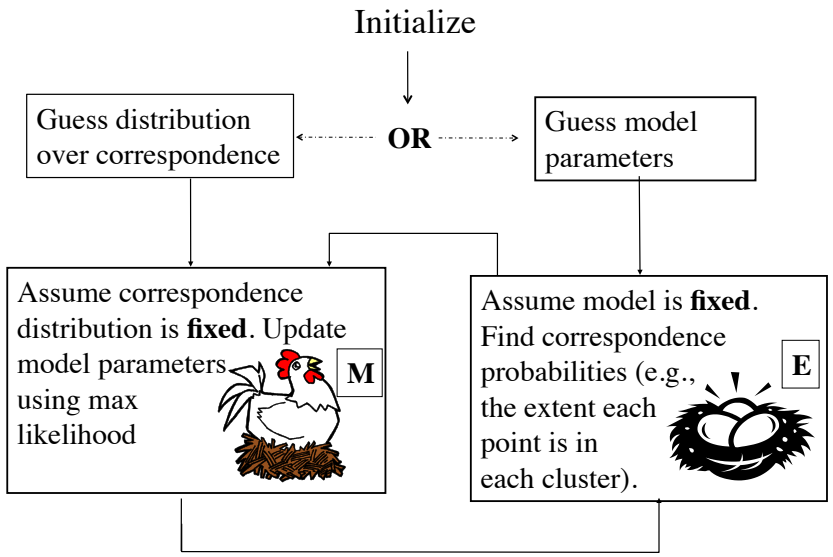
## Expectation Maximization (EM)

Difference with K-means.

We have **distributions** over the assignments, $p(c \mid \mathbf{x})$.

This leads us to work with expectations.

EM flow chart




# EM for GMM

$$p(\mathbf{x}) = \sum_c p(c)p(\mathbf{x} \mid c) \qquad \text{where} \qquad p(\mathbf{x} \mid c) = \mathbb{N}(\boldsymbol{\mu}_c, \Sigma_c)$$

$$\Theta = \{\Theta_c\}$$

And, for multiple points

$$p(\{\mathbf{x}_i\}|\theta) = \prod_i \left( \sum_c p(c)p(\mathbf{x} \mid c) \right)$$

This is our objective function.

# EM for GMM

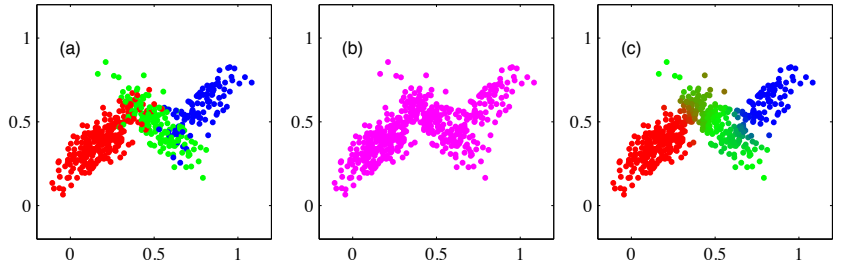
Assume we have estimates for the probability distribution over clusters for each point (the "egg"). Specifically we have:

$$p(c \mid \mathbf{x}_i, \Theta^{(s)}) \qquad \text{(s indexes interation (step))}.$$

These are called the *responsibilities*.

This is the extent to which each cluster explains the point. (Every point is in every cluster to some degree).

# Responsibilities illustrated




Points colored according to whether they were generated by the red, green, or blue clusters (normally not known).

Observed points without cluster information.

Points colored according the the degree that they are explained by the red, green, or blue clusters.

# EM for GMM

- We estimate the mean for each cluster naturally by:

Iteration (step)

$$\mu_c^{(s+1)} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot p(c \mid \mathbf{x}_i, \Theta_c^{(s)})}{\sum_{i=1}^{n} p(c \mid \mathbf{x}_i, \Theta_c^{(s)})} \quad \text{(weighted average)}$$

- Variances/covariances work similarly

# EM for GMM

- Also, intuitively,

$$p(c) = \frac{\sum_i p\left(c \middle| \mathbf{x}_i, \Theta^{(s)}\right)}{\sum_c \sum_i p\left(c \middle| \mathbf{x}_i, \Theta^{(s)}\right)} = \frac{\sum_i p\left(c \middle| \mathbf{x}_i, \Theta^{(s)}\right)}{N}$$

We can sort out the chicken!

# EM for GMM

Given the parameters (the chicken), the probability that a given point is associated with each cluster is computed by:

$$p(c \mid \mathbf{x}_i, \Theta^{(s)}) = \frac{\pi_c^{(s)} \cdot p(\mathbf{x}_i \mid \Theta_c^{(s)})}{\sum_{c'} \pi_{c'}^{(s)} \cdot p(\mathbf{x}_i \mid \Theta_{c'}^{(s)})}$$

(Note that we select $\Theta_c^{(s)}$ from $\Theta^{(s)}$.

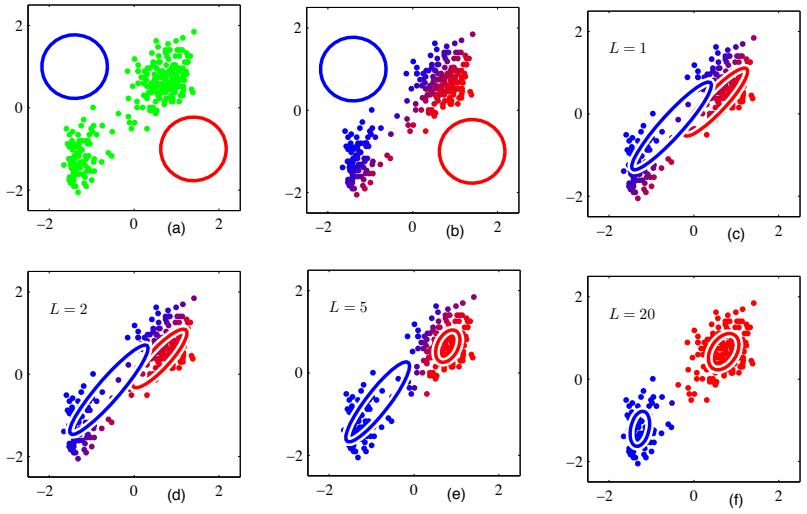
where $\pi_c^{(s)} = p\left(c \middle| \Theta_c^{(s)}\right)$ i.e., $\pi_c^{(s)}$ is part of $\Theta_c^{(s)}$.

This is the cluster membership discussed before,

with less formal notation: $p(c|x) \propto p(c)p(x|c)$

We can do the egg!

# EM illustrated



# EM (more formally)

Semi-optional technical material alert!

The formal treatment helps us use EM correctly in more complex situations. However, EM algorithms for many problems can "guessed at" using intuition.

The more formal treatment is not needed for the homework.

# EM (more formally)

- Assume K clusters. Index over clusters by $k$, over points by $n$.
- New notation for cluster membership:

  For each point, $n$, $z_n$ is a vector of K values where exactly one $z_{n,k} = 1$, and all others are 0. Note that $\sum_k z_{n,k} = 1$.

# EM (more formally)

- Denote cluster priors by:

$$\pi_k \equiv p\left(z_k = 1\right)$$

- Denote the responsibilities that each cluster has for each point by:

$$\gamma\left(z_{n,k}\right) \equiv p\left(z_{n,k} = 1 \middle| x_n, \theta^{(s)}\right) = \frac{\pi_k \; p\left(x_n \middle| z_{n,k} = 1, \theta_k^{(s)}\right)}{\sum_{k'} \pi_{k'} \; p\left(x_n \middle| z_{n,k'} = 1, \theta_{k'}^{(s)}\right)}$$

# EM (more formally)

Represent the entire data set of N points, $\mathbf{x}_n$,
as a matrix X with rows $\mathbf{x}_n^T$.

Represent the latent variable assignments with a matrix Z.
(For the true assignment, each row is zero except for a single element that is 1.)

We call $\{Z, X\}$ the *complete* data set (everthing is known).
The observed data, $\{X\}$, is called the *incomplete* data set.

# EM (more formally)

We assume that computing the MLE of parameters,

$$\arg\max_{\theta}\left\{\log\left\{p(Z,X|\theta)\right\}\right\}$$

with complete data is relatively easy.

Recall our intuitive treatment of EM for GMM. If we knew the cluster membership, we would know how to compute the means.

Since we did not know the cluster membership we did a weighted computation, which happens to be an expectation of the complete log likelihood, over the assignment (responsibility) distribution.

# EM (more formally)

Notice the complexity of the incomplete log likelihood:

$$\log\left(p(X|\theta)\right)=\sum_{n}\log\underbrace{\left(\sum_{k}\pi_k\ p(x_n|\theta)\right)}_{\text{nasty sum in log}}$$

By constrast, for complete log likelihood we can incorporate the assignment by:

$$p(X,Z|\theta)=\prod_{n}\prod_{k}\pi_k^{z_{n,k}}\left\{p(x_n|\theta)\right\}^{z_{n,k}}$$

So

$$\log\left(p(X,Z|\theta)\right)=\sum_{n}\sum_{k}\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta)\right)\right)\right\}$$

(No nasty sum in log; well suited for the expectation calculation).

# EM (more formally)

For the E step, we compute the responsibilities which is straightforward.

Define $\mathrm{Q}\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_{Z}p\left(Z|X,\theta^{(s)}\right)\log\left(p\left(X,Z|\theta^{(s+1)}\right)\right)$

(Expectation of $\log\left(p\left(X,Z|\theta^{(s+1)}\right)\right)$ over $p\left(Z|X,\theta^{(s)}\right)$).

The M step then computes $\theta^{(s+1)}=\arg\max_{\theta}\left\{\mathrm{Q}\left(\theta^{(s+1)},\theta^{(s)}\right)\right\}$

Maximizing Q is generally feasible and corresponds to the intuitive development.

# General EM algorithm

1. Choose initial values for $\theta^{(s=1)}$
   (can also do assignments, but then jump to M step).
2. E step: Evalute $p\left(Z|X,\theta^{(s)}\right)$
3. M step: Evalute $\theta^{(s+1)}=\arg\max_{\theta}\left\{\mathrm{Q}\left(\theta^{(s+1)},\theta^{(s)}\right)\right\}$
   where $\mathrm{Q}\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_{Z}p\left(Z|X,\theta^{(s)}\right)\log\left(p\left(X,Z|\theta^{(s+1)}\right)\right)$
4. Check for convergence; If not done, goto 2.

★ At each step, our objective function increases unless it is at a local maximum. It is important to check this is happening for debugging!

# General EM algorithm

★ At each step, our objective function (conditioned on the current model) increases unless it is at a local maximum. It is important to check this is happening for debugging!

Recall our objective function for the case of a mixture model:

$$p(X|\theta)=\prod_n \sum_k p(k)p(x_n|\theta_k)$$

or

$$\log\left(p(X|\theta)\right)=\sum_n \log\left(\sum_k p(k)p(x_n|\theta_k)\right)$$

**Implementation tip**. This is conveniently available from the computation of responsibilities (before normalization).

# Deriving the GMM M-step

Evalute $\theta^{(s+1)} = \arg\max_\theta \left\{Q\left(\theta^{(s+1)},\theta^{(s)}\right)\right\}$

where $Q\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_Z p\left(Z|X,\theta^{(s)}\right)\log\left(p\left(X,Z|\theta^{(s)}\right)\right)$

Recall that $\log\left(p\left(X,Z|\theta\right)\right)=\sum_n\sum_k\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}$

So $Q\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_Z p\left(Z|X,\theta^{(s)}\right)\sum_n\sum_k\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}$

# Deriving the GMM M-step

$$Q\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_Z p\left(Z|X,\theta^{(s)}\right)\sum_n\sum_k\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}$$
$$=\sum_n\sum_k\sum_Z p\left(Z|X,\theta^{(s)}\right)\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}$$

This exchanging of summing order says that **instead** of summing over points and clusters for all correspondences Z, we sum over all correspondences for a **given** point and cluster.

The quantity in parentheses only interacts with the given point and cluster from the outside. So intuitively, most of the sum ver

# Deriving the GMM M-step

$$Q\left(\theta^{(s+1)},\theta^{(s)}\right)=\sum_Z p\left(Z|X,\theta^{(s)}\right)\sum_n\sum_k\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}$$
$$=\sum_n\sum_k\underbrace{\sum_Z p\left(Z|X,\theta^{(s)}\right)\left\{z_{n,k}\left(\log(\pi_k)+\log\left(p(x_n|\theta_k)\right)\right)\right\}}_{\text{inner sum}}$$

This exchanging of summing order says that **instead** of summing over points and clusters for all correspondences Z, we sum over all correspondences for a **given** point and cluster.

We will focus on the inner sum.

Z is all possible correspondences. To generate them all more explicitly, we can consider the first point. For each possible assignment of the first point, we then need all possible combinations of the other points. To get that, we consider all possible assignments of the second point, together with all possible assignments of the remaining points. This shows:

$$\sum_{Z}(\ ) \equiv \sum_{\mathbf{z}_1}\sum_{\mathbf{z}_2}\sum_{\mathbf{z}_3}\cdots\sum_{\mathbf{z}_N}(\ )$$

Note that a sum over $z_n$ is short hand for a sum over cluster assignments for point n. Hence each of the sums on the right are over clusters.

Further, because the points are independent, we have:

$$p(Z|\bullet) = \prod_{\mathbf{z}_i} p(\mathbf{z}_i|\bullet)$$

$$\underbrace{\sum_{Z} p(Z|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)}_{\text{inner sum from previous}}$$

$$= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N}\prod_{n'} p(z_{n'}|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)$$

(Using the two formulas from the previous page)

$$\underbrace{\sum_{Z} p(Z|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)}_{\text{inner sum from previous}}$$

$$= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N}\prod_{n'} p(z_{n'}|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)$$

$$= \sum_{z_n} p(z_n|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right) \underbrace{\sum_{z_1}\cdots\sum_{z_{n-1}}\sum_{z_{n+1}}\cdots\sum_{z_N}\prod_{n'\neq n} p(z_{n'}|X,\theta^{(s)})}_{\text{All possibilities without point n. This entire mess evaluates to unity!}}$$

(Here we move in all the sums that do not interact with the function we are taking the expectation over that do not interact with the outer *n* variable. )

$$\underbrace{\sum_{Z} p(Z|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)}_{\text{inner sum from previous}}$$

$$= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N}\prod_{n'} p(z_{n'}|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)$$

$$= \sum_{z_n} p(z_n|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right) \underbrace{\sum_{z_1}\cdots\sum_{z_{n-1}}\sum_{z_{n+1}}\cdots\sum_{N}\prod_{n'\neq n} p(z_{n'}|X,\theta^{(s)})}_{\text{All possibilities without point n. This entire mess evaluates to unity!}}$$

$$= \sum_{z_n} p(z_n|X,\theta^{(s)}) \bullet z_{n,k} \bullet \left(\log(\pi_k) + \log\left(p(x_n|\theta_k)\right)\right)$$

(As noted, the big sum on the right is unity. It is the probability of all possible configurations that do not involve point *n*. Since this covers all cases, it is one.)

$$\underbrace{\sum_{Z} p\left(Z \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)}_{\text{inner sum from previous}}$$

$$= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N}\prod_{n'} p\left(z_{n'} \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$

$$= \sum_{z_n} p\left(z_n \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right) \underbrace{\sum_{z_1}\cdots\sum_{z_{n-1}}\sum_{z_{n+1}}\cdots\sum_{N}\prod_{n' \neq n} p\left(z_{n'} \middle| X,\theta^{(s)}\right)}_{\substack{\text{All possibilities without point n.} \\ \text{This entire mess evaluates to unity!}}}$$

$$= \sum_{z_n} p\left(z_n \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$

$$= p\left(z_{n,k} = 1 \middle| X,\theta^{(s)}\right)\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$

(Here, remember that $k$ is sitting outside the sum. The indicator variable $z_{n,k}$, selects the probability for $k$ from the sum over $z_n$, which is a sum over clusters.)


$$\underbrace{\sum_{Z} p\left(Z \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)}_{\text{inner sum from previous}}$$

$$= \sum_{z_1}\sum_{z_2}\cdots\sum_{z_N}\prod_{n'} p\left(z_{n'} \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$

$$= \sum_{z_n} p\left(z_n \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right) \underbrace{\sum_{z_1}\cdots\sum_{z_{n-1}}\sum_{z_{n+1}}\cdots\sum_{N}\prod_{n' \neq n} p\left(z_{n'} \middle| X,\theta^{(s)}\right)}_{\substack{\text{All possibilities without point n.} \\ \text{This entire mess evaluates to unity!}}}$$

$$= \sum_{z_n} p\left(z_n \middle| X,\theta^{(s)}\right) \bullet z_{n,k} \bullet \left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$

$$= p\left(z_{n,k} = 1 \middle| X,\theta^{(s)}\right)\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$


$$= \gamma\left(z_{n,k}\right)\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)$$ (definition of $\gamma\left(z_{n,k}\right)$, the responsibility)

# Deriving the M-step

$$Q\left(\theta^{(s+1)},\theta^{(s)}\right) = \sum_{Z} p\left(Z \middle| X,\theta^{(s)}\right) \sum_{n}\sum_{k}\left\{z_{n,k}\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)\right\}$$

$$= \sum_{n}\sum_{k}\left\{\gamma\left(z_{n,k}\right)\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)\right\}$$

We need to maximize this with respect to the parameters for each cluster, $k$. Notice that:

$$\frac{\delta}{\delta\theta_{k^*}} Q\left(\theta^{(s+1)},\theta^{(s)}\right) = \sum_{n}\left\{\gamma\left(z_{n,k^*}\right)\frac{\delta}{\delta\theta_{k^*}}\left(\log\left(\pi_{k^*}\right) + \log\left(p\left(x_n \middle| \theta_{k^*}\right)\right)\right)\right\}$$

(The values of $k$ not of current interest, i.e., not $k$*, die)

# Example—deriving the GMM M-step

$$\frac{\delta}{\delta\mu_k} Q\left(\theta^{(s+1)},\theta^{(s)}\right) = \sum_{n}\left\{\gamma\left(z_{n,k}\right)\frac{\delta}{\delta\mu_k}\left(\log\left(\pi_k\right) + \log\left(p\left(x_n \middle| \theta_k\right)\right)\right)\right\}$$

$$= \sum_{n}\left\{\gamma\left(z_{n,k}\right)\frac{\delta}{\delta\mu_k}\left(\log\left(p\left(x_n \middle| \theta_k\right)\right)\right)\right\}$$

$$\sum_{n}\left\{\gamma\left(z_{n,k}\right)\frac{\delta}{\delta\mu_k}\left(\log\left(\mathrm{N}\left(x_n \middle| \mu_k, \Sigma_k\right)\right)\right)\right\}$$

## Example—deriving the GMM M-step

$$\mathrm{N}\left(\mathbf{x}_n \middle| \mu_k, \Sigma_k\right) = \frac{1}{(2\pi)^{D/2}\left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2}\left(\mathbf{x}_n - \mu_k\right)^T \Sigma_k^{-1}\left(\mathbf{x}_n - \mu_k\right)\right)$$

$$\log\left(\mathrm{N}\left(\mathbf{x}_n \middle| \mu_k, \Sigma_k\right)\right) = \log\left(\frac{1}{(2\pi)^{D/2}\left|\Sigma_k\right|^{1/2}}\right) - \frac{1}{2}\left(\mathbf{x}_n - \mu_k\right)^T \Sigma_k^{-1}\left(\mathbf{x}_n - \mu_k\right)$$

$$\frac{\delta}{\delta\mu_k}\log\left(\mathrm{N}\left(\mathbf{x}_n \middle| \mu_k, \Sigma_k\right)\right) = \Sigma_k^{-1}\left(\mathbf{x}_n - \mu_k\right)$$

## Example—deriving the GMM M-step

$$\frac{\delta}{\delta\mu_k}\mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) = \sum_n \left\{\gamma\left(z_{n,k}\right) \frac{\delta}{\delta\mu_k}\left(\log\left(\mathrm{N}\left(x_n \middle| \mu_k, \Sigma_k\right)\right)\right)\right\}$$

$\frac{\delta}{\delta\mu_k}\mathrm{Q}\left(\theta^{(s+1)}, \theta^{(s)}\right) = 0$ means that

$$\sum_n \left\{\gamma\left(z_{n,k}\right) \Sigma_k^{-1}\left(\mathbf{x}_n - \mu_k\right)\right\} = 0$$ (Inner sigma is a matrix, not a sum).

$$\sum_n \left\{\gamma\left(z_{n,k}\right)\left(\mathbf{x}_n - \mu_k\right)\right\} = 0$$ (Multiply by $\Sigma_k^{-1}$)

## Example—deriving the GMM M-step

So, $\sum_n \left\{\gamma\left(z_{n,k}\right)\left(\mathbf{x}_n - \mu_k\right)\right\} = 0$

and $\mu_k \sum_n \left\{\gamma\left(z_{n,k}\right)\right\} = \sum_n \left\{\gamma\left(z_{n,k}\right)\left(\mathbf{x}_n\right)\right\}$

and $\mu_k = \dfrac{\sum_n \left\{\gamma\left(z_{n,k}\right)\left(\mathbf{x}_n\right)\right\}}{\sum_n \left\{\gamma\left(z_{n,k}\right)\right\}}$ (same as before)

## Example—deriving the GMM M-step

Finding variances/covariances is similar.

Finding the mixing coefficients is also similar, except we also need to enforce that they sum to one.

(Here the equations for the *k*'s are coupled).

So we use Lagrange Multipliers.

# Using Lagrange Multipliers

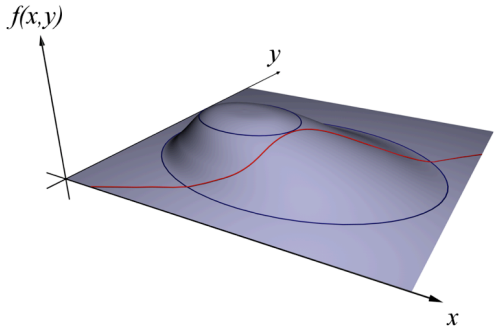
Now we find stationary points with respect to $\{\pi_k, \lambda\}$ of

$$Q\left(\theta^{(s+1)}, \theta^{(s)}\right) + \lambda\left(\sum_k \pi_k - 1\right)$$

Note that differentiating with respect to $\lambda$, and setting the result to zero puts the constraint into the equations.

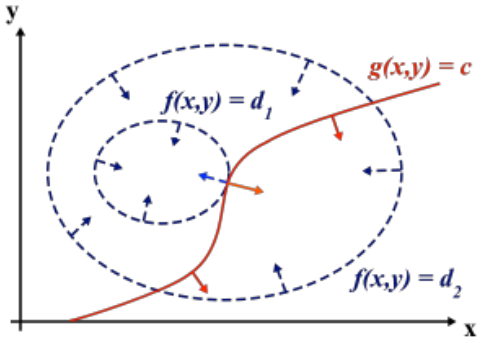
But the real problem is doing the optimization under the constraint.

# Using Lagrange Multipliers




From WikiPedia

# Using Lagrange Multipliers




From WikiPedia

# Using Lagrange Multipliers

$\nabla f \parallel \nabla g$

$\nabla f = \lambda \nabla g$

So, $\nabla(f - \lambda g) = 0$

or, $\nabla(f + \lambda g) = 0$ (negate $\lambda$)

# Using Lagrange Multipliers

Now we find stationary points with respect to $\{\pi_k, \lambda\}$ of

$$Q\left(\theta^{(s+1)}, \theta^{(s)}\right) + \lambda\left(\sum_k \pi_k - 1\right)$$

$$\frac{\delta}{\delta \pi_k}\left\{Q\left(\theta^{(s+1)}, \theta^{(s)}\right) + \lambda\left(\sum_k \pi_k - 1\right)\right\}$$
$$= \sum_n \left\{\gamma\left(z_{n,k}\right) \frac{\delta}{\delta \pi_k}\left(\log\left(\pi_k\right) + \log\left(N\left(x_n \middle| \mu_k, \Sigma_k\right)\right)\right)\right\} + \lambda$$
$$= \sum_n \left\{\gamma\left(z_{n,k}\right) \frac{1}{\pi_k}\right\} + \lambda$$

Setting the result to zero, $\sum_n \left\{\gamma\left(z_{n,k}\right) \frac{1}{\pi_k}\right\} + \lambda = 0$

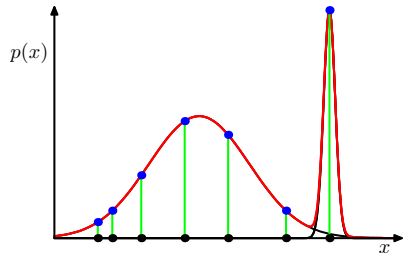
So $\pi_k = \frac{\sum_n \left\{\gamma\left(z_{n,k}\right)\right\}}{-\lambda}$

Summing over $k$ gives, $1 = \frac{\sum_k \sum_n \left\{\gamma\left(z_{n,k}\right)\right\}}{-\lambda} = \frac{N}{-\lambda}$

So, $\lambda = -N$, and $\pi_k = \frac{\sum_n \left\{\gamma\left(z_{n,k}\right)\right\}}{N}$ as before.

# EM in practice

- For GMM we need to consider clusters that have essentially one point:




- Easily fixed by adding a constant to the variance (prior).

# EM in practice

- Tying parameters (using GMM as an example)
  - We can improve stability by assuming the variances (or covariances) for all clusters are the same.
  - Updates work as you expect. Instead of multiple weighted sums, you just use one big one.
  - But note that one advantage of GMM over K-means is that the scale is naturally taken care of, and the clusters **can** have different variances.

# EM in practice

- You must check that the log likelihood increases!
- A simple way to compute it during an iteration:

Recall our objective function:

$$p(X)=\prod_n \sum_k p(k)p(x_n|k)$$

Consider how we might compute the responsibilities
$\gamma(n,k) \propto p(k)p(x_n|k)$
(Then normalize once you have them all).

So, make a running sum of the unormalized values

# EM in practice

- Precision problems --> must work with logs
- But we need to exponentiate to normalize --> rescaling tricks

Let $P=\{p_i\}$.

Suppose we want $Q=\frac{1}{\sum_i p_i}\{p_i\}$

Where we need to use $V=\{\log(p_i)\}$

and $\exp(p_i)$ is too small, and the sum of them might be zero.

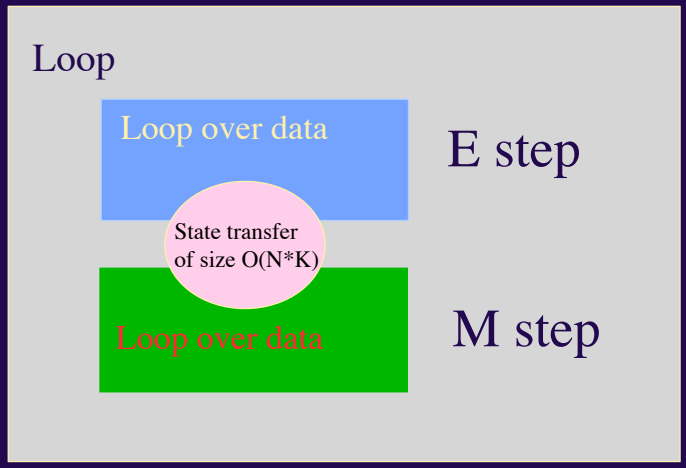
Let $M = \max\{\log(p_i)\}$

Observe that working with $V'=\{\log(p_i)-M\}$ does the trick.

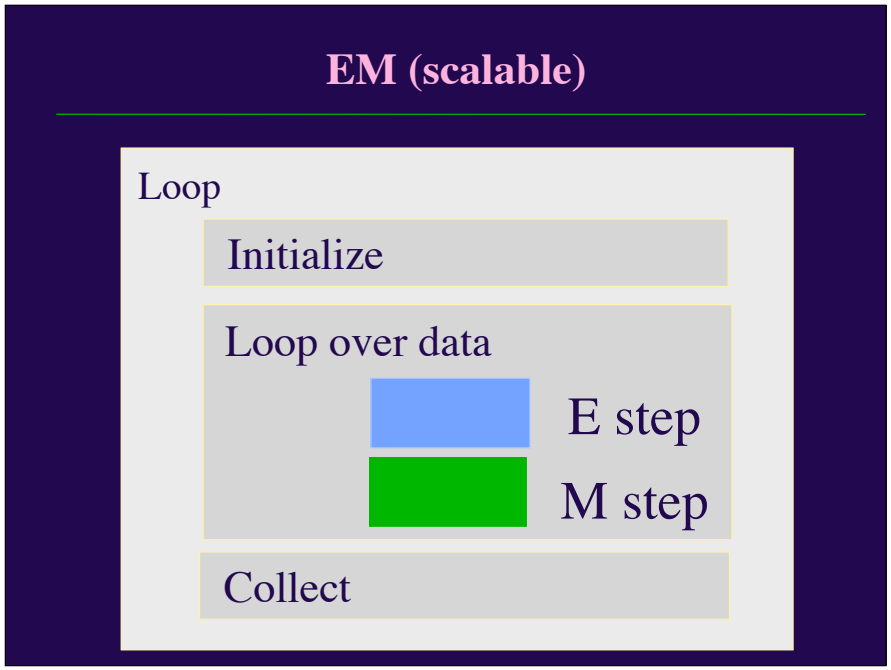
# EM in practice (continued)

- Memory problems ---> we can compute means, etc., as running totals so that we do not need to store responsibilities for all points over all clusters.

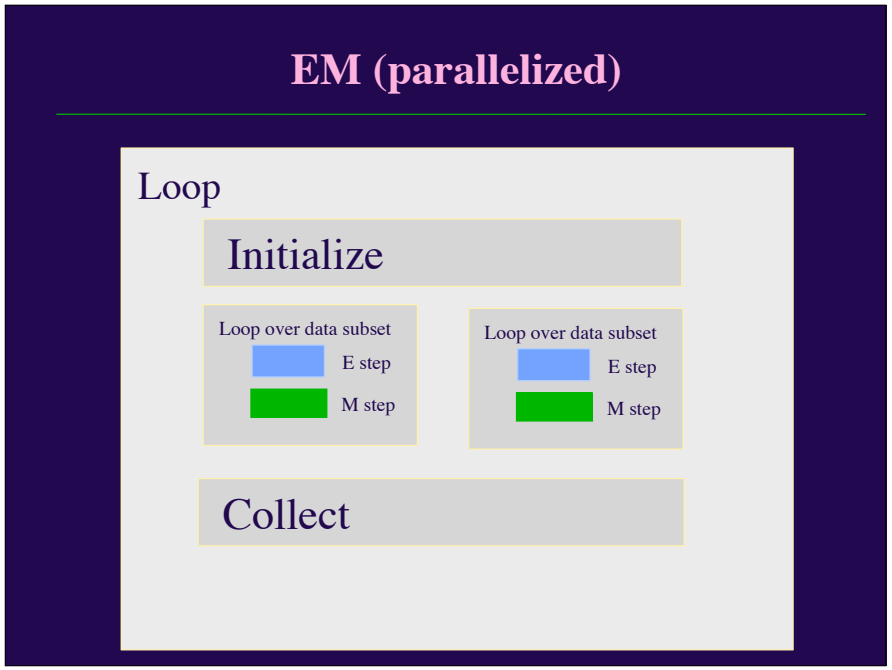
# EM (Straight Forward Implementation)



## EM (scalable)




## EM (parallelized)




## Analysis of EM

- Maximizing the Q function provided a new parameter estimate which increased the likelihood
- Showing this typically uses Jensen's inequality
  - Bishop (§9.4), instead, uses the fact that the KL divergence between two distributions is non-negative, but showing this uses Jensen's.
- Given a bounded likelihood, this means the algorithm converges to a stationary point
  - Typically a local maximum but examples where it is a saddle point can be constructed.

## Analysis of EM

- We will sketch the summary provided in the online resource "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman
- This follows "The EM Algorithm and Extensions" by Geoffrey McLachlan and Thriyambakam Krishnan.
- See also Bishop (§9.4)

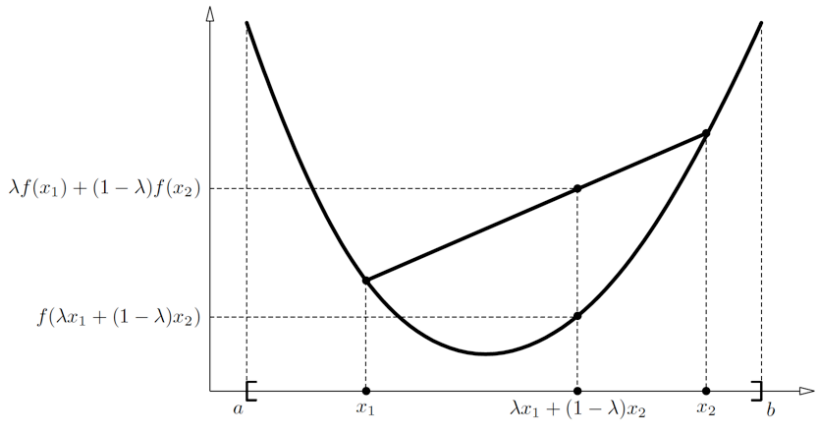



Figure 1: $f$ is *convex* on $[a, b]$ if $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$ $\forall x_1, x_2 \in [a, b], \quad \lambda \in [0, 1]$.

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

More generally, if $f$ is convex, then, for

$\lambda_i \geq 0$, and $\sum_i \lambda_i = 1$

we have

$$f\left(\sum_i x_i \lambda_i\right) \leq \sum_i \lambda_i f(x_i)$$

(Jensen's inequality)

Result from calculas (prove via mean value theorem)

If $f$ is twice differentiable on $[a,b]$ and $f'' \geq 0$ on $[a,b]$, then $f(x)$ is convex on $[a,b]$.

Notice that $f(x) = -\log(x)$ is convex

Proof?

$$f'(x) = -\frac{1}{x}$$

$$f''(x) = \frac{1}{x^2}$$

$$f\left(\sum_i x_i \lambda_i\right) \le \sum_i \lambda_i f(x_i) \quad \text{(Jensen's inequality)}$$

$$\log\left(\sum_i x_i \lambda_i\right) \ge \sum_i \lambda_i \log(x_i) \qquad (-\log(x) \text{ is convex})$$

In EM, we seek $\theta$ to maximize $L(\theta) = \ln P(\mathbf{X}|\theta)$

Suppose at step $n$ we have $L(\theta_n)$

$$L(\theta) - L(\theta_n) = \ln\left(\sum_{\mathbf{z}} \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)\right) - \ln \mathcal{P}(\mathbf{X}|\theta_n). \qquad (11)$$

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

$$\begin{aligned}
L(\theta) - L(\theta_n) &= \ln\left(\sum_{\mathbf{z}} \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)\right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&= \ln\left(\sum_{\mathbf{z}} \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta) \cdot \frac{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)}\right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&= \ln\left(\sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)}\right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \\
&\ge \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln\left(\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)}\right) - \ln \mathcal{P}(\mathbf{X}|\theta_n) \quad (12) \\
&= \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln\left(\frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)\mathcal{P}(\mathbf{X}|\theta_n)}\right) \quad (13) \\
&\triangleq \Delta(\theta|\theta_n). \quad (14)
\end{aligned}$$

Jensen's

$\ln P(\mathbf{X}|\theta_n) = \sum_z \ln P(\mathbf{X}|\theta_n) P(z|X,\theta_n)$
because $P(\mathbf{X}|\theta_n)$ does not depend on z,
and $\sum_z P(z|X,\theta_n) = 1$

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

$$L(\theta) \ge L(\theta_n) + \Delta(\theta|\theta_n)$$

$$l(\theta|\theta_n) \triangleq L(\theta_n) + \Delta(\theta|\theta_n)$$

$$L(\theta) \ge l(\theta|\theta_n).$$

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

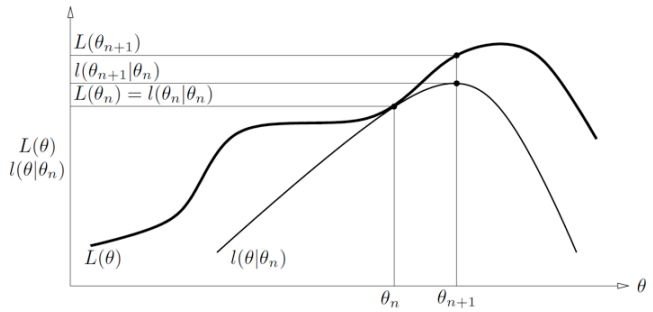



Figure 2: Graphical interpretation of a single iteration of the EM algorithm: The function $l(\theta|\theta_n)$ is bounded above by the likelihood function $L(\theta)$. The functions are equal at $\theta = \theta_n$. The EM algorithm chooses $\theta_{n+1}$ as the value of $\theta$ for which $l(\theta|\theta_n)$ is a maximum. Since $L(\theta) \geq l(\theta|\theta_n)$ increasing $l(\theta|\theta_n)$ ensures that the value of the likelihood function $L(\theta)$ is increased at each step.

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

$$
\begin{aligned}
l(\theta_n|\theta_n) &= L(\theta_n) + \Delta(\theta_n|\theta_n) \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta_n)\mathcal{P}(\mathbf{z}|\theta_n)}{\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)\mathcal{P}(\mathbf{X}|\theta_n)} \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \frac{\mathcal{P}(\mathbf{X},\mathbf{z}|\theta_n)}{\mathcal{P}(\mathbf{X},\mathbf{z}|\theta_n)} \\
&= L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln 1 \\
&= L(\theta_n), \qquad (16)
\end{aligned}
$$

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman

$$
\begin{aligned}
\theta_{n+1} &= \arg\max_{\theta} \{l(\theta|\theta_n)\} \\
&= \arg\max_{\theta} \left\{ L(\theta_n) + \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \frac{\mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta)}{\mathcal{P}(\mathbf{X}|\theta_n)\mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n)} \right\} \\
&\quad \text{Now drop terms which are constant w.r.t. } \theta \\
&= \arg\max_{\theta} \left\{ \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \mathcal{P}(\mathbf{X}|\mathbf{z},\theta)\mathcal{P}(\mathbf{z}|\theta) \right\} \\
&= \arg\max_{\theta} \left\{ \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \frac{\mathcal{P}(\mathbf{X},\mathbf{z},\theta)}{\mathcal{P}(\mathbf{z},\theta)} \frac{\mathcal{P}(\mathbf{z},\theta)}{\mathcal{P}(\theta)} \right\} \\
&= \arg\max_{\theta} \left\{ \sum_{\mathbf{z}} \mathcal{P}(\mathbf{z}|\mathbf{X},\theta_n) \ln \mathcal{P}(\mathbf{X},\mathbf{z}|\theta) \right\} \\
&= \arg\max_{\theta} \left\{ \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta_n} \{\ln \mathcal{P}(\mathbf{X},\mathbf{z}|\theta)\} \right\} \qquad (17)
\end{aligned}
$$

From "The Expectation Maximization Algorithm: A short tutorial" by Sean Borman