K-Means

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

$$\sum_{i\in\text{clusters}}\left\{\sum_{j\in\text{elements of i'th cluster}}\left\|x_j-\mu_i\right\|^2\right\}$$

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

K-Means

- Want to minimize

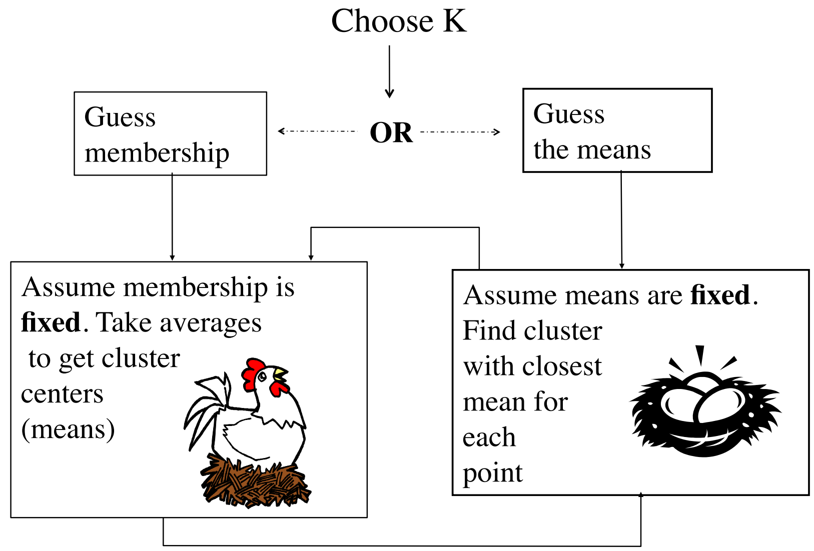
$$\sum_{i\in\text{clusters}}\left\{\sum_{j\in\text{elements of i'th cluster}}\left\|x_j-\mu_i\right\|^2\right\}$$

- Note that if we know the cluster allocation, we can compute the means
- **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



Choose K

Guess membership

OR

Guess the means

Assume membership is **fixed**. Take averages to get cluster centers (means)

Assume means are **fixed**. Find cluster with closest mean for each point

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.

you should be able to argue why this is true

- Unfortunately, the dimension of the space is usually large, and high-dimensional space have lots of local maximum (standard problem!)
  - Dimensionality here is K*dim(**x**)
- Finding the global minimum for a real problem is very optimistic!

Clustering using a generative statistical model

Associate each cluster with 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,

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

$p(\mathbf{x}|c)$ 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 model.

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

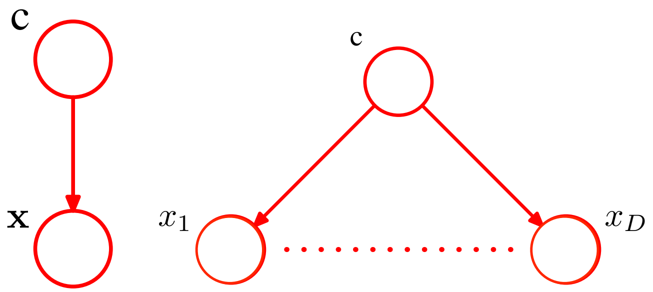
(We have already seen both these case with Naive Bayes).

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?)

$$\begin{aligned} p(\mathbf{x}) &= \sum_c p(\mathbf{x},c) \\ &= \sum_c p(c)\,p(\mathbf{x}|c) \end{aligned}$$

Generative story:

1) choose a cluster with probability, $p(c)$.

2) sample from $p(\mathbf{x}|c)$.

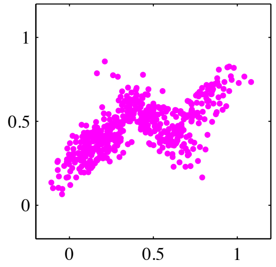
3) rinse and repeat.

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 $p\left(\mathbf{x}|c\right)$ for all K clusters indexed by c.

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

We simultaneously cluster points and learn the cluster model.

Learning the parameters from data

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

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

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

or

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

Expectation Maximization (EM)

Operationally this is similar to K-means.

Observe that:

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

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

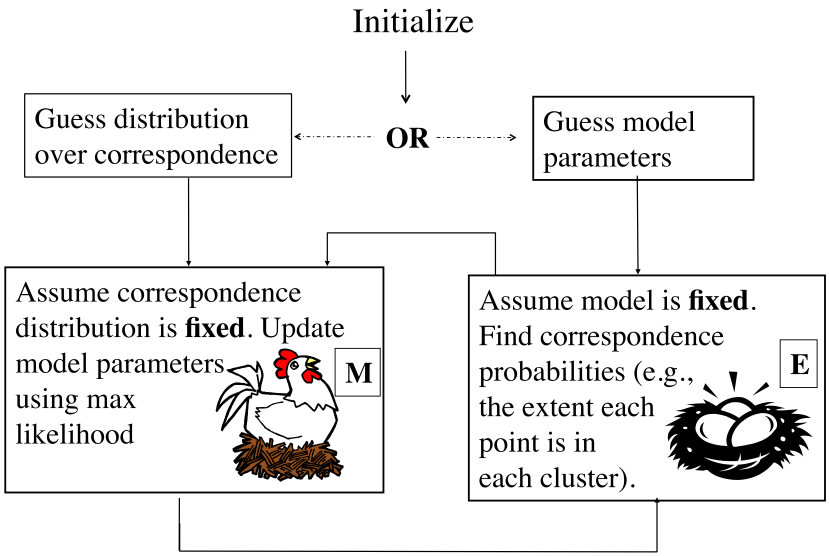
Expectation Maximization (EM)

Difference with K-means.

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

This leads us to work with expectations.

EM flow chart



Initialize

Guess distribution over correspondence

OR

Guess model parameters

Assume correspondence distribution is **fixed**. Update model parameters using max likelihood

M

Assume model is **fixed**. Find correspondence probabilities (e.g., the extent each point is in each cluster).

E

EM for GMM

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

$\Theta = \{\Theta_c\}$

And, for multiple points

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

This is our objective function.

EM for GMM

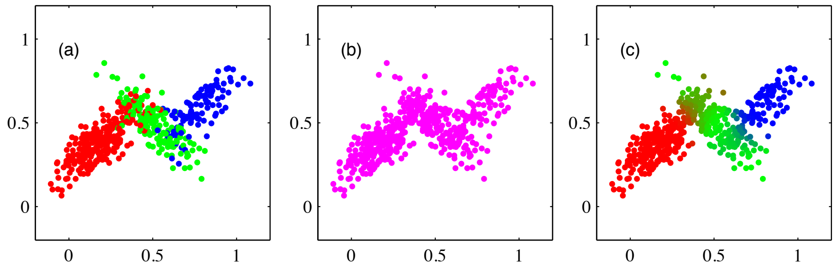
Assume we have estimates for the probability distribution over clusters for each point (the “egg”). Specifically we have:

$$p(c | \mathbf{x}_i, \Theta^{(s)})$$ (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)

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

- 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:

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

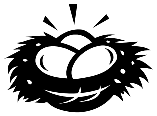
(Note that we select $\Theta_c^{(s)}$ from $\Theta^{(s)}$.

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

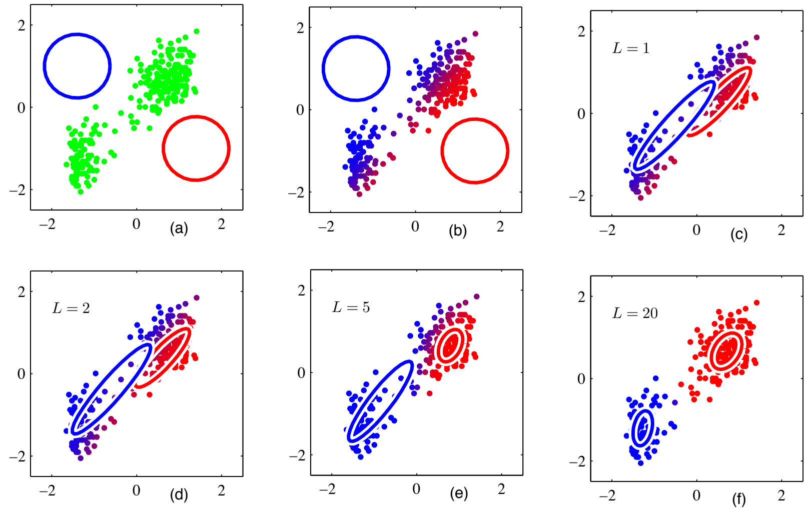
We can do the egg!

This is the cluster membership discussed before,

with less formal notation: $p\left(c \middle| x\right) \propto p(c)\, p\left(x \middle| c\right)$



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)

- Responsibilities for GMM using this notation:

$$\gamma\left(z_{n,k}\right) = \frac{\pi_k \mathrm{N}\left(x_n \middle| u_k^{(s)}, \Sigma_k^{(s)}\right)}{\sum_{k'} \pi_{k'} \mathrm{N}\left(x_n \middle| u_{k'}^{(s)}, \Sigma_{k'}^{(s)}\right)}$$

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.