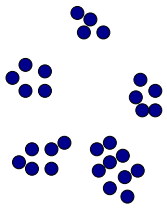
Lecture 18: EM for mixture models

- Basic review of Gaussians
- Mixture models
- EM for mixture models

Recall from last time: Expectation Maximization (EM)

- A general purpose method for learning from incomplete data
- Main idea:
  - If we had complete data (sufficient statistics) we could easily maximize the likelihood
  - So in the case of missing values, we will “fantasize” what they should be, based on the current parameter setting
  - This means we compute **expected sufficient statistics**
  - Then we improve the parameter setting, based on these statistics

Mixture models



- Suppose you have a bunch of data
- You'd like to know how the data was generated
- In particular, the data is coming from some underlying probability distribution. Can we figure out what it is?
- In this case, it seems that there are really 5 "distributions" generating the data
- This is called a **mixture model**
- It also seems these distributions are normal, or Gaussian (especially because everyone loves Gaussians :)

Quick refresher: Gaussian distributions

- The univariate Gaussian (Normal distribution) with mean μ and variance σ:

$$p(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Suppose we have data with n attributes (so now a data instance $\vec{x}$ is an n-dimensional vector)

  The *multivariate* Gaussian distribution with **mean vector** $\vec{\mu}$ (n-dimensional vector) and **covariance matrix** Σ ($n \times n$ matrix, symmetric, positive semi-definite) is:

$$p(\vec{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})\right)$$

  where $|\Sigma|$ denotes the determinant of Σ

Mixture of Gaussians

- Suppose we have a set of data points, $\langle \vec{x_1}, \ldots \vec{x_m} \rangle$
- **Gaussian assumption:** Suppose there are k Gaussian distributions $\mathcal{N}(\vec{\mu_j}, \Sigma_j), j = 1 \ldots k$ which generate data.
- Every data point is generated by first selecting one of these distributions, with probability $p_j, j = 1, \ldots k$, and then generating a point from the distribution.
- Can we estimate $p_j, \vec{\mu_j}, \Sigma_j, j = 1 \ldots k$ from the data?
- We'll do the 1-dimensional case first...

Maximum likelihood to the rescue!

- Write down the log likelihood of the data given a set of parameters p_i, μ_i, σ_i, $i = 1, \ldots k$:

$$\begin{aligned}
\log L &= \log p(\langle x_1, \ldots x_n \rangle | p_1, \ldots p_k, \mu_1, \ldots \mu_k, \sigma_1 \ldots \sigma_k) \\
&= \log \prod_{i=1}^{m} p(x_i | p_1, \ldots p_k, \mu_1, \ldots \mu_k, \sigma_1 \ldots \sigma_k) \text{ (assuming data is iid)} \\
&= \sum_{i=1}^{m} \log \left(\sum_{j=1}^{k} p(x_i | \mu_j, \sigma_j) p_j \right) \text{ (given how the data is drawn)}
\end{aligned}$$

- Now compute derivative wrt p_j, μ_j, σ_j, set to 0 and solve...

This cannot be solved analytically!

Latent variables

- In the previous model, we do not know *which Gaussian* generates each data point.
- The identity of the Gaussian is called a **latent (hidden, unobserved) variable**
- Latent variables are very important in practical applications
- **Latent variables make the problem difficult!**

A simple problem!

- Suppose we knew the latent variables (i.e., we knew which Gaussian generated which point)
- Let k_i be the Gaussian that generated point i. Then we have:

$$\begin{aligned}
\log L &= \sum_{i=1}^{m} \log\left(p(x_i|\mu_{k_i}, \sigma_{k_i}, p_{k_i})\right) \\
&= \sum_{i=1}^{m} \left(\log p(\mathbf{x_i}|\mu_{k_i}, \sigma_{k_i}) + \log p_{k_i}\right) \\
&= \sum_{i=1}^{m} \left(-\frac{(x_i - \mu_{k_i})^2}{2\sigma_{k_i}^2} - \frac{1}{2}\log\left(2\pi\sigma_{k_i}^2\right) + \log p_{k_i}\right)
\end{aligned}$$

- Now it is easy to maximize the likelihood!

Maximum likelihood estimation: Univariate Gaussian

- Working with k_i (the component which generated instance i) is awkward.
- To make things easy, we'll define an indicator variable, δ_{ij}, which is equal to 1 if and only if $k_i = j$, 0 otherwise.
- With this notation, the likelihood becomes:

$$\log L = \sum_{i=1}^{m} \sum_{j=1}^{k} \delta_{ij} \left(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2} - \frac{1}{2} \log (2\pi\sigma_j^2) + \log p_j \right)$$

- Note that we can reverse the sums at will

Maximum likelihood estimation: Univariate Gaussian (2)

- To compute the means μ_j, we have:

$$\frac{\partial \log L}{\partial \mu_j} = \sum_{i=1}^{m} \delta_{ij} \left(-\frac{2(x_i - \mu_j)(-1)}{2\sigma_j^2} \right) = 0$$

  Solving this, we get:

$$\mu_j = \frac{\sum_{i=1}^{m} \delta_{ij} x_i}{\sum_{i=1}^{m} \delta_{ij}}$$

  In other words, to estimate the mean of the jth component, we compute the sample mean of the instances generated by it
- Similarly, we get for the standard deviation:

$$\sigma_j^2 = \frac{\sum_{i=1}^{m} \delta_{ij} (x_i - \mu_j)^2}{\sum_{i=1}^{m} \delta_{ij}}$$

Maximum likelihood estimation: Univariate Gaussian (3)

- To find p_j, we also have to take into account that $\sum_j p_j = 1$.
- Assume without loss of generality (wlog) that we replace $p_k = 1 - \sum_{j=1}^{k-1} p_j$:

$$\frac{\partial \log L}{\partial p_j} = \sum_{i=1}^{m} \left(\delta_{ij} \frac{1}{p_j} + \delta_{ik} \frac{1}{1 - \sum_{j=1}^{k-1} p_j} (-1) \right) = 0, \forall j = 1, \ldots k-1$$

- By manipulating this equation, we get:

$$\frac{\sum_{i=1}^{m} \delta_{ij}}{\sum_{i=1}^{m} \delta_{ik}} = \frac{p_j}{p_k}, \forall j = 1, \ldots k-1$$

- Now we take sums on both sides:

$$\sum_{j=1}^{k-1} \frac{\sum_{i=1}^{m} \delta_{ij}}{\sum_{i=1}^{m} \delta_{ik}} = \sum_{j=1}^{k-1} \frac{p_j}{p_k}$$

Maximum likelihood estimation: Univariate Gaussian (4)

- We know that $\sum_{i=1}^{m} \sum_{j=1}^{k} \delta_{ij} = m$ (because each instance is generated by exactly one Gaussian component)
- We also know that $\sum_{j=1}^{k-1} p_j = 1 - p_k$
- Plugging everything back in the equation, we get:

$$\frac{m - \sum_{i=1}^{m} \delta_{ik}}{\sum_{i=1}^{m} \delta_{ik}} = \frac{1 - p_k}{p_k}$$

- Solving, we get that the probability p_k is given by the empirical fraction of the points coming from the kth distribution:

$$p_k = \frac{\sum_{i=1}^{m} \delta_{ik}}{m}$$

And because we did this wlog, it holds for any p_j.

Maximum likelihood estimation: Multivariate Gaussian

- It is just a straightforward generalization of the univariate case
- The probability p_j is just the number of points coming from the jth distribution

$$p_j = \frac{\sum_{i=1}^{m} \delta_{ij}}{m}$$

- The mean of the distribution is given by the empirical mean of the points coming from it:

$$\vec{\mu_j} = \frac{\sum_{i=1}^{m} \delta_{ij} \vec{x_i}}{\sum_{i=1}^{m} \delta_{ij}}$$

- The covariance matrix is the covariance of the points from the distribution

$$\Sigma_j = \frac{\sum_{i=1}^{m} \delta_{ij} \left(\vec{x_i} - \vec{\mu_j}\right) \left(\vec{x_i} - \vec{\mu_j}\right)^T}{\sum_{i=1}^{m} \delta_{ij}}$$

EM for Mixture of Gaussians

- We start with an initial guess for the parameters p_j, $\vec{\mu_j}$, Σ_j
- We will iterate an e-step, in which we "complete" the data, with an M-step, in which we re-compute the parameters
- In the "hard EM" version, completing the data means that each data point is assumed to be generated by *exactly one Gaussian*
- This is very related to *k-means clustering* (which you may know)
- In the "soft EM" version (also usually known as EM), we assume that each data point could have been generated from *any component*
- In this case, each point will contribute to the mean and variance estimate of each component.

Hard EM for Mixture of Gaussians

1. Guess an initial parameter setting $p_j, \vec{\mu_j}, \Sigma_j, j = 1 \ldots k$
2. Repeat until convergence:
   (a) *E-step:* For each $i = 1, \ldots m$ and each $j = 1, \ldots k$:

$$k_i = \arg\max_j p(\vec{x_i} \text{ drawn from distribution } j | p_j, \vec{\mu_j}, \Sigma_j)$$

where $p(\vec{x_i}$ drawn from distribution $j | p_j, \vec{\mu_j}, \Sigma_j) \propto p_j p(\vec{x_i} | \vec{\mu_j}, \Sigma_j)$

   (b) *M-step:* Update the parameters of the model to maximize the likelihood of the data

$$p_j = \frac{1}{m} \sum_{i=1}^{m} \delta_{ij} \qquad \vec{\mu_j} = \frac{\sum_{i=1}^{m} \delta_{ij} \vec{x_i}}{\sum_{i=1}^{m} \delta_{ij}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} \delta_{ij} (\vec{x_i} - \vec{\mu_j}) (\vec{x_i} - \vec{\mu_j})^T}{\sum_{i=1}^{m} \delta_{ij}}$$

Soft EM for Mixture of Gaussians

1. Guess an initial parameter setting $p_j, \vec{\mu_j}, \Sigma_j, j = 1, \ldots k$
2. Repeat until convergence:
   (a) *E-step:* For each $i = 1, \ldots m$ and each $j = 1, \ldots k$:

$$w_j(i) = p(\vec{x_i} \text{ drawn from distribution } j | p_j, \vec{\mu_j}, \Sigma_j)$$

where $p(\vec{x_i}$ drawn from distribution $j | p_j, \vec{\mu_j}, \Sigma_j) \propto p_j p(\vec{x_i} | \vec{\mu_j}, \Sigma_j)$

   (b) *M-step:* Update the parameters of the model to maximize the likelihood of the data

$$p_j = \frac{1}{m} \sum_{i=1}^{m} w_j(i) \qquad \vec{\mu_j} = \frac{\sum_{i=1}^{m} w_j(i) \vec{x_i}}{\sum_{i=1}^{m} w_j(i)}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j(i) (\vec{x_i} - \vec{\mu_j}) (\vec{x_i} - \vec{\mu_j})^T}{\sum_{i=1}^{m} w_j(i)}$$

EM in general

- Whenever we are trying to model data drawn probabilistically, and we have missing values in the data, EM is an option
- We need some structured or parametric form of the distribution (we saw Bayes nets and mixtures of Gaussians as examples)
- We starts with a guess for the parameters of the distribution
- You can think of the E-step as trying to "complete" the data, by filling in the missing values
- The M-step will compute new parameters, given the completed data

Theoretical properties of EM

- Each iteration improves the likelihood:

$$L(\theta_{i+1}|D) \geq L(\theta_i|D)$$

- If the parameters do not change in one iteration, $\theta_{i+1} = \theta_i$, then the gradient of the log-likelihood function is 0 at θ_i:

$$\frac{\partial L(\theta|D)}{\partial \theta}(\theta_i) = 0$$

  This means that θ_i is a min, max or saddle point
- In practice, convergence only occurs at local maxima
- See textbook for a detailed description of general EM and its properties

EM pros and cons

- Much easier to implement than gradient descent; no parameter tuning is necessary, and no projection of the parameters (we compute them directly normalized)
- Converges much faster than vanilla gradient descent
- Not very sensitive to the starting point
- Speed comparison with fancy gradient descent is unclear

Summary

- Difficulty of inference with missing data is that it generates a complex likelihood function
- We have a non-linear optimization problem, and we can use two solutions:
  - Gradient descent: always works for non-linear optimization
  - EM: targeted towards optimizing likelihood
- Both are only guaranteed to converge to local maxima, so we need restarts
- Both use inference to compute expected sufficient statistics of the data
- Hence, the inference step is a bottleneck for both