CSC321 Tutorial 10: Assignment 3 Review

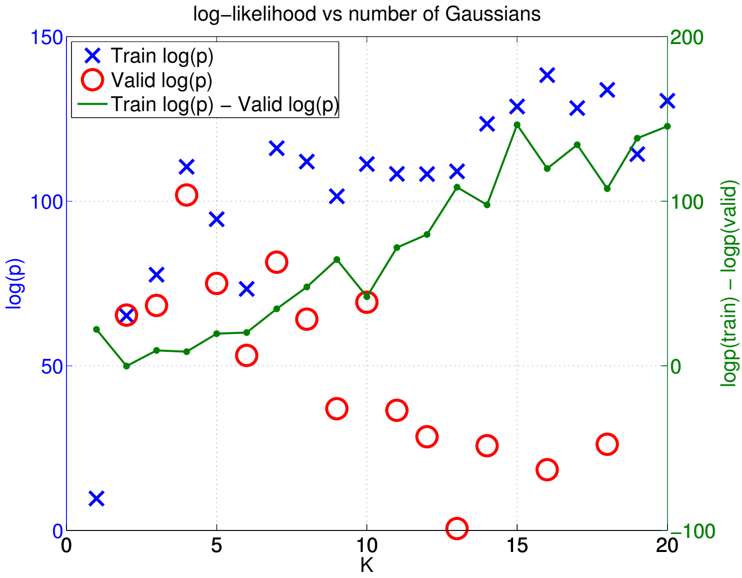
Yue Li
Email: yueli@cs.toronto.edu

Wed 11-12 March 26
Fri 10-11 March 28

Assignment 3 PART 1 (5 points)

- Run `moginit` to create training and validation datasets from 4 random Gaussians.
- Then use the function `mogem` to fit various numbers of Gaussians to the training data.
- Using performance on the *validation data*, determine the optimal number of Gaussians to fit to the training data.
- Present your results as a graph that plots both the validation density and the training density as a function of the number of Gaussians.
- Include a brief statement of what you think the graph shows.
- Also include a brief statement about the effects of changing the initial standard deviation used in `mogem`.
- Please do not change the random seeds in `mogem` (this will produce different data).

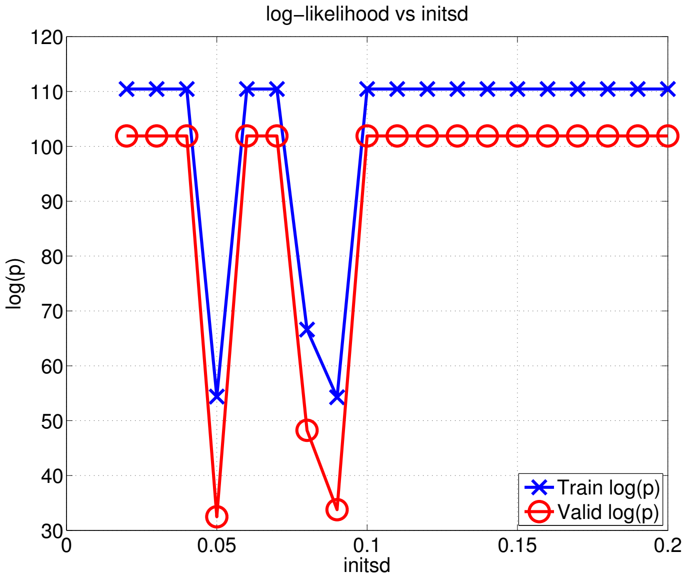
3 points: a graph that plots both the validation density and the training density as a function of the number of Gaussians.
Fixed other parameters: `nIter = 100; initsd = 0.04;`



1 point:

- Since the data were sampled from 4 random Gaussians (30 cases per Gaussian), the optimal number of K should be around 4. Indeed, at initsd $= 0.04$ (NB: different initsd is also accepted) and nIter $= 100$, the K $= 4$ corresponds to the highest test logp $= 101.8960$.
- A sign of overfitting is also observed when K becomes greater than 4 because the difference between train and validation log p starts to increase (green line).

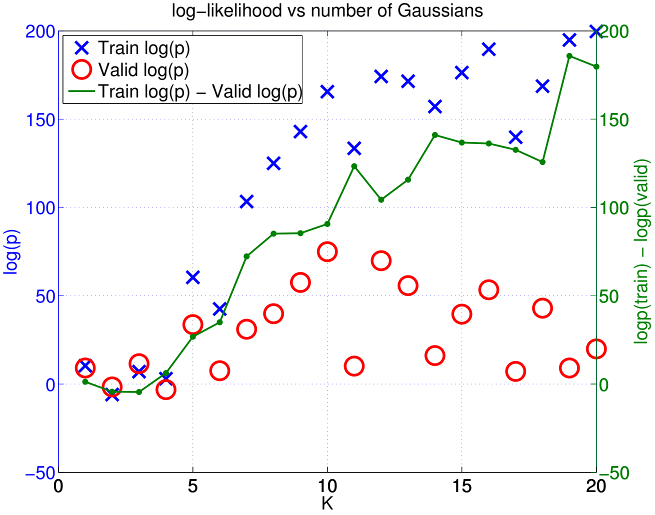
1 point: Fixing $K = 4$, changing the initsd has little effect on the results when initsd > 0.1, indicating that the EM algorithm is fairly robust to different initial initsd.



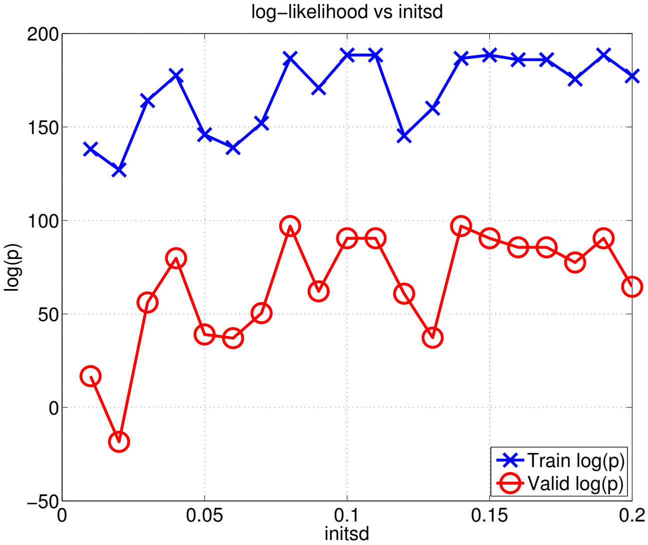
Assignment 3 PART 2 (2 points)

- Change `moginit.m` to use only **12** cases per Gaussian, and **12** axis-aligned gaussians to generate the data and repeat the experiment above (without changing the random seeds).
- Present your results as a graph and include a brief statement of what you think the graph shows and why it differs from the graph in PART 1.

1 point: With only 12 cases per Gaussian and initsd = 0.04 (NB: different initsd is also accepted), overfitting occurs after K = 10 as shown by the decreasing trend of validation log p despite ever-increasing training log p. The overfitting can be seen more clearly from the increasing trend of the difference b/w training log p and validation log p.



Fixing K = 12, EM becomes less robust to different initsd for more complex model (optional).

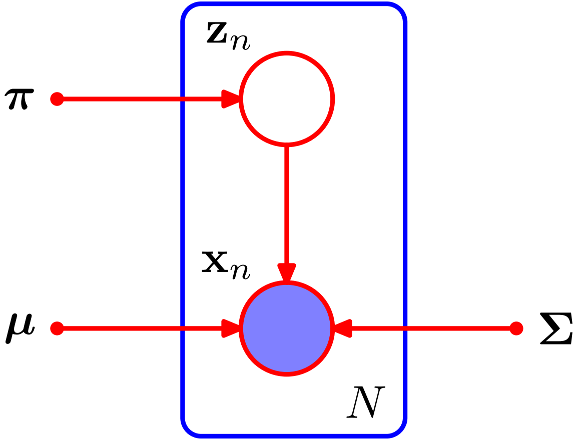


Comment: comparing to the maximum likelihood approach used here, Bayesian estimation can provide a more robust estimate (i.e. less dependent on the data size and initial parameters; recall A2 part 3). An approximation schemes such as sampling or variational inference (in particular, variational Bayesian EM) must be used to estimate the expectation over the entire model space. Interested readers can refer to Chapter 10.1, 10.2 from Bishop textbook.

Assignment 3 PART 3 (3 points) - some programming

- Change `mogem.m` so that in addition to fitting the means and axis-aligned variances, it also fits the **mixing proportions**.
- Currently, `mogem` does not mention mixing proportions so it is currently assuming that they are all equal (which makes them all cancel out when computing the posterior probability of each Gaussian for each datapoint.).
- So the first thing to do is to include mixing proportions when computing the posterior, but keep them fixed (and not all equal).

Review of MoG: a generative model



Pattern recognition and machine learning, Chapter 9 p433, (Bishop, 2006)

- Objective function:

$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Maximum likelihood (ML) solutions for $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\mu}_k} = 0 \Longrightarrow \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k) \mathbf{x_n}$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \boldsymbol{\Sigma}_k} = 0 \Longrightarrow \boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T$$

$$\frac{\partial \ln p(\mathbf{X})}{\partial \pi_k} = 0 \Longrightarrow \pi_k = \frac{N_k}{N}$$

where

- $N_k = \sum_{n=1}^{N} \gamma(z_k)$
- $\gamma(z_k) = p(z_k = 1 | \mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$

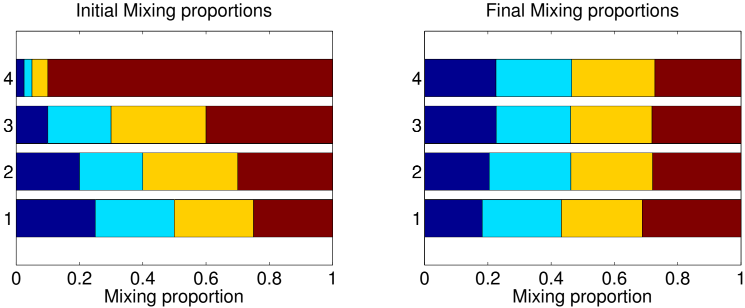
EM algorithm for MoG :

1. Initialize $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, $\pi_k = \frac{1}{K}$ or by K-means.
2. **E-step**. Evaluate the responsibilities $\gamma(z_k)$ using $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k:
$$\gamma(z_k) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{1}$$
3. **M-step**. Re-estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, π_k based on the ML solutions:
$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)\mathbf{x_n} \tag{2}$$
$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_k)(\mathbf{x} - \mu_k^{new})(\mathbf{x} - \boldsymbol{\mu}_k^{new})^T \tag{3}$$
$$\pi_k^{new} = \frac{N_k}{N} \tag{4}$$
where $N_k = \sum_{n=1}^{N} \gamma(z_k)$.
4. Evaluate the log likelihood:
$$\ln p(\mathbf{X}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{5}$$

- **1 point**: add fixed mixprop and change log p computation:
  - `mixprop = ones(1,numgaussians)/numgaussians;` %Initialize to uniform mixprop
  - `valid_densities(:,g) = mixprop(g) * (1/(2*pi*sqrt(xv*yv)))*exp(-xd.*xd/(2*xv) - yd.*yd/(2*yv));` %Numerator of the posterior, i.e., Eq 1
  - `train_densities(:,g)= mixprop(g) * (1/(2*pi*sqrt(xv*yv)))*exp(-xd.*xd/(2*xv) - yd.*yd/(2*yv));` %Numerator of the posterior, i.e., Eq 1
  - `trainLogp = sum(log( sum(train_densities, 2) ));` %Training likelihood Eq 5
  - `validLogp = sum(log( sum(valid_densities, 2) ));` %Validation likelihood Eq 5

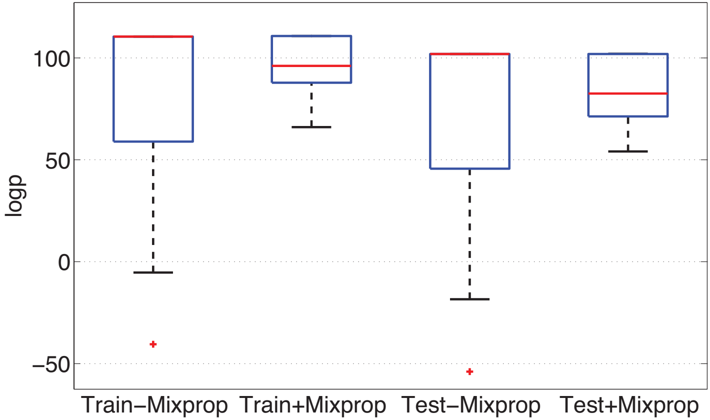
    Note: no division by `numgaussians` (i.e., assuming 1/4 for all mixprop) in the above likelihoods once the `mixprop` is fixed
- **1 point** learn mixprop:
  - `mixprop(g) = sum(r) / numcases;`%Eq 4

1 point: print out the final mixprops for 4 Gaussians (and initsd = 0.04) with initial mixprop set to [0.25, 0.25, 0.25, 0.25], [0.3, 0.2, 0.2, 0.3], [0.1, 0.2, 0.3, 0.4] and [0.9, 0.025, 0.025, 0.05]. Despite different initial mixprop, the final mixprop should be approximately evenly distributed reflecting the true mixprops (i.e., 0.25 for all 4 Gaussians)



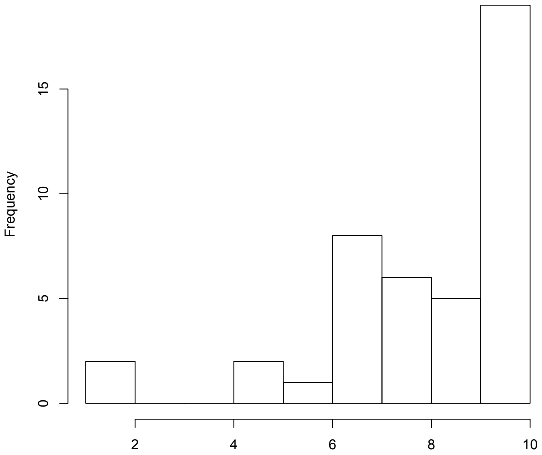
NOTE: full mark of 3 points are granted if your mixprops are correct; otherwise the above marks were counted

optional: compare log p before and after adding mixprop for 4 Gaussians for the same setting over multiple runs.



NB: the *final likelihoods* are plotted over 100 runs for each method

A3 Marks



Class average: 8.25; median: 9 (well done!)