CS423 project Pitch

Group 4 - Taeyoung Kim[1], Janggun Lee[2], Geon Park[2]

April 27, 2020

[1] School of Computing

[2] Department of Mathematical Sciences

Table of Contents

Gaussian Mixture Model-1

GMM(Gaussian Mixture Model) is a well known generative model in machine learning. It views data as mixture of some multivariate normal random variable.

$$p(x_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k), \sum_{k=1}^{K} \pi_k = 1$$

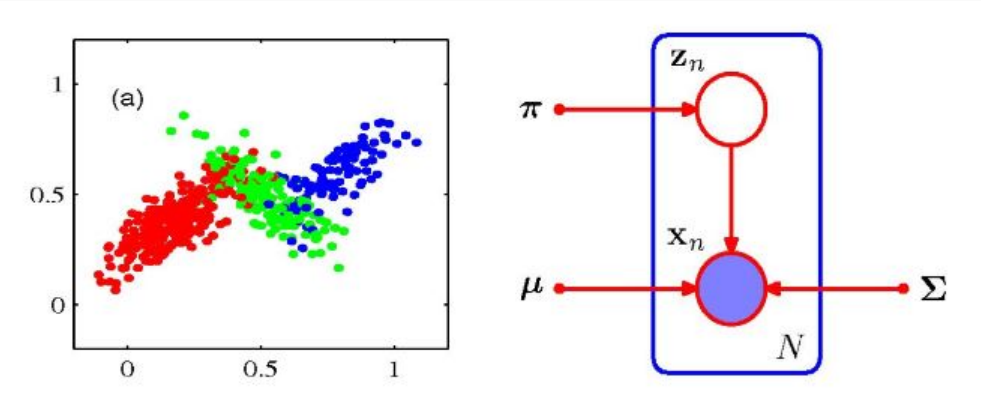


Figure 1: Visualization of GMM and model structure

Gaussian Mixture Model-2

Unlike simplicity of model itself, deriving learning algorithm is not simple. It uses **EM(Expectation-Maximization)** algorithm

Step E : With current parameters, compute value $p(z_{ik} = 1|x_i)$, which is probability that data x_i is contained in k-th cluster.

$$p(z_{ik} = 1|x_i) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(x_i|\mu_l, \Sigma_l)}$$

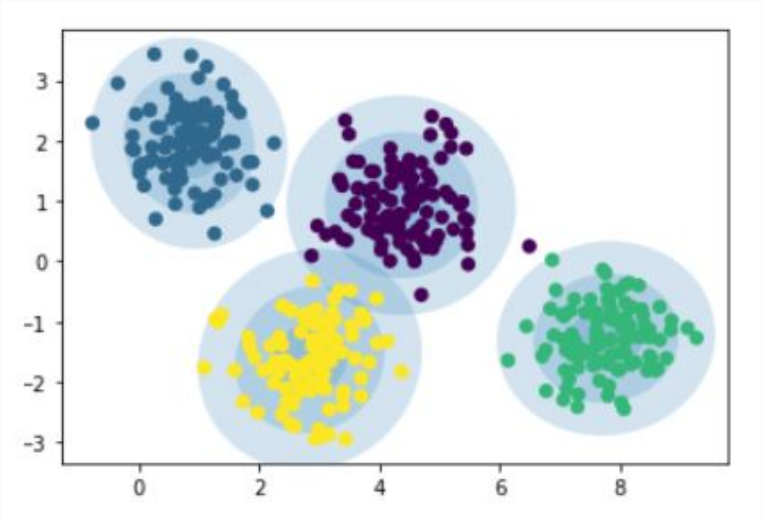
Step M : Update parameters with computed values $p(z_i k = 1|x_i)$.

$$\mu_k = \sum_{i=1}^{N} \frac{p(z_{ik} = 1|x_i)}{\sum_{j=1}^{N} p(z_{jk} = 1|x_j)} x_i$$

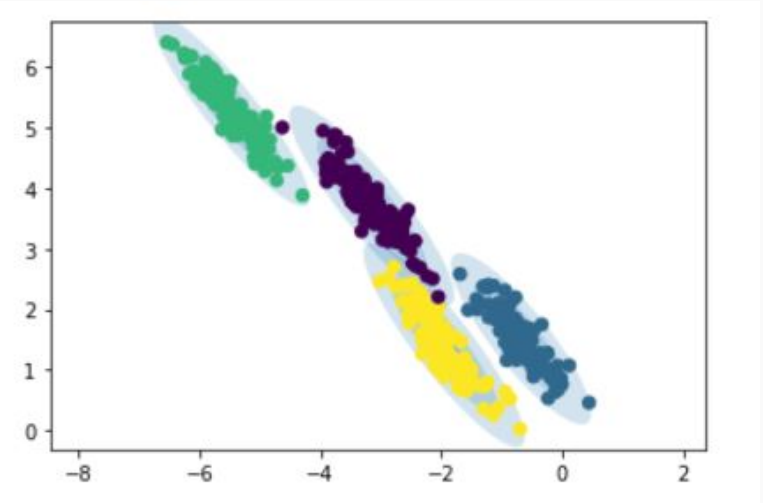
$$\Sigma_k = \sum_{i=1}^{N} \frac{p(z_{ik} = 1|x_i)}{\sum_{j=1}^{N} p(z_{jk} = 1|x_j)} (x_i - \mu_k)(x_i - \mu_k)^T$$

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} p(z_{ik} = 1|x_i)$$

Gaussian Mixture Model-3



(a) 4-componenets GMM with circular clusters



(b) 4-componenets GMM with squashed clusters

Figure 2: Examples of GMM in different dataset

Guassian Mixture Model-4

GMM in Anglican

```
(with-primitive-procedures [row-mean invert shape identity-matrix get-row]
  (defquery gmm [data & [hyperparams]]
    (let [[N D] (shape data)
          K         (:K         hyperparams 10)
          alpha     (:alpha     hyperparams 1.0)
          mu-0      (:mu-0      hyperparams (row-mean data))
          lambda-0  (:lambda-0  hyperparams (identity-matrix D))
          nu        (:nu        hyperparams (inc D))
          kappa     (:kappa     hyperparams 1.0)
          pi (sample (dirichlet (repeat K alpha)))
          lambda (into [] (map (fn [x] (sample x))
                               (repeat K (wishart nu lambda-0))))
          mu (into [] (map
                       (fn [k]
                         (sample (mvn mu-0
                                      (invert kappa
                                              (get lambda k)))))
                       (range K)))
          sigma (into [] (map invert lambda))]
      (loop [n 0
             z []]
        (if (= n N)
          z
          (let [row (get-row data n)
                k (sample (discrete pi))]
            (observe (mvn (get mu k) (get sigma k)) row)
            (recur (inc n) (conj z k)))))))))
```

Mixture of Experts-1

MoE(Mixture of Experts) consists of **gating model** and **experts**, where gating model divides the input into subsets, and experts learn on the subset.

Mixture models like GMM are good for generative modelings. By MoE, we can use them on discriminative models, like regression or classification problem.

Figure 3: Mixture of Experts

Definition (Inverse Problem)

Given data (x_i, y_i) such that $y_i = f(x_i) + \epsilon$, inverse problem is finding some function g such that $x \approxeq g(f(x))$.

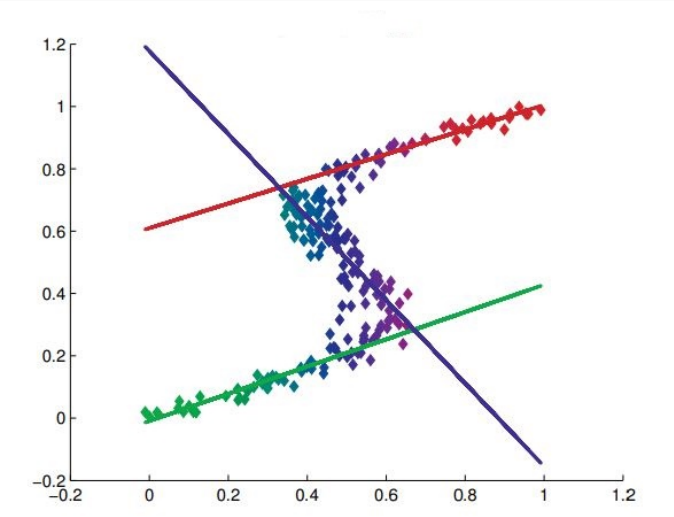


Figure 4: Example of GMM based MoE on inverse problem

MoE in inverse problems:

If f is not invertible, the model needs to choose one of x that $g(x) \approxeq y$. Choosing one of them is done by the gates of MoE, and approximation of subspace is done by each experts.

Mixture of Experts-3

We may have experts of MoE as MoEs. This creates **hierarchical MoE**, which is a finite tree where every leaf nodes are experts, and internal nodes are gates.

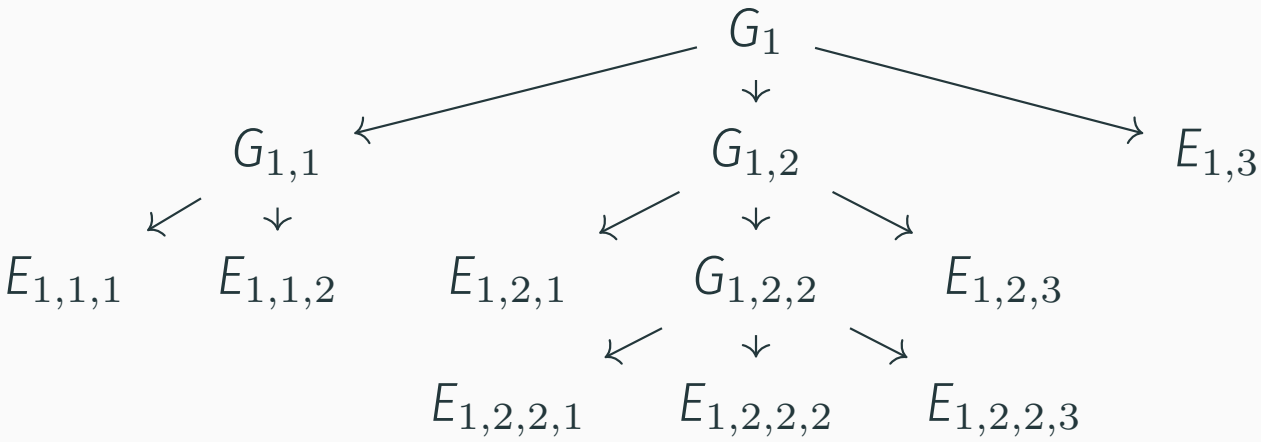


Figure 5: Example of hierarchical mixture of experts model

Experiment

We will use hierarchical mixture model, where gates are given by GMM. Target is inverse problem mentioned above, but specific dataset is not yet chosen.

Our algorithm is composed of 3 parts, GMM Generator, Expanding Decider, Expert Generator.

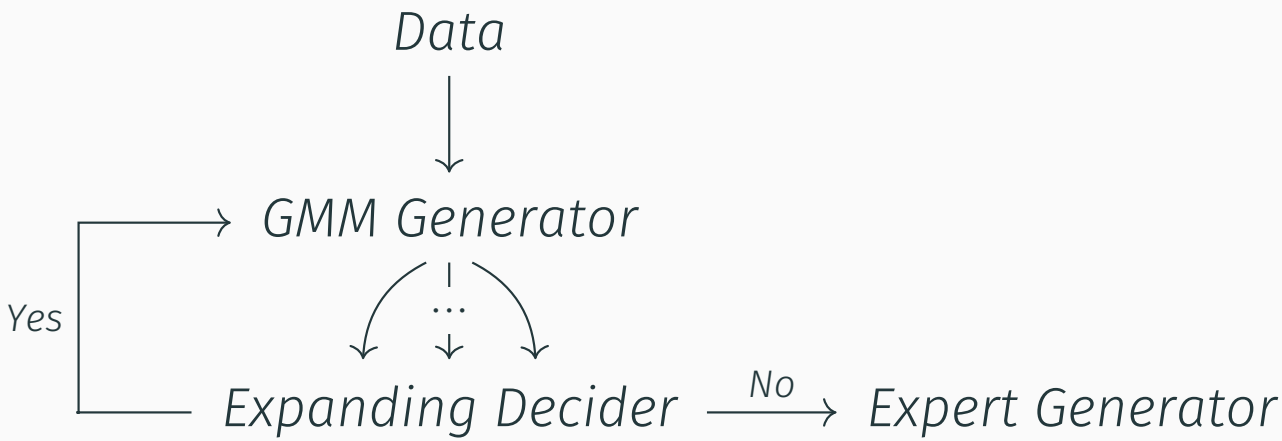


Figure 6: Data flow diagram for our model

Advantage of using ProbProg-1

EM algorithm is hard to implement, hard to understand, and even prone to local minima. Though EM algorithm for GMM is simpler, its derivation includes some matrix calculus, and EM algorithm for mixture of experts is more complex.

Using probabilistic programming will solve this computation problem.

Advantage of using ProbProg-2

Hyperparameter - for example, number of clusters - tuning in GMM is not simple. Usual approach is trying for various choices, then evaluate them, and this is time-consuming, as well as difficult to automate.

We expect that including hyperparameters as distribution, then inferring posterior distribution on themselves will solve this.

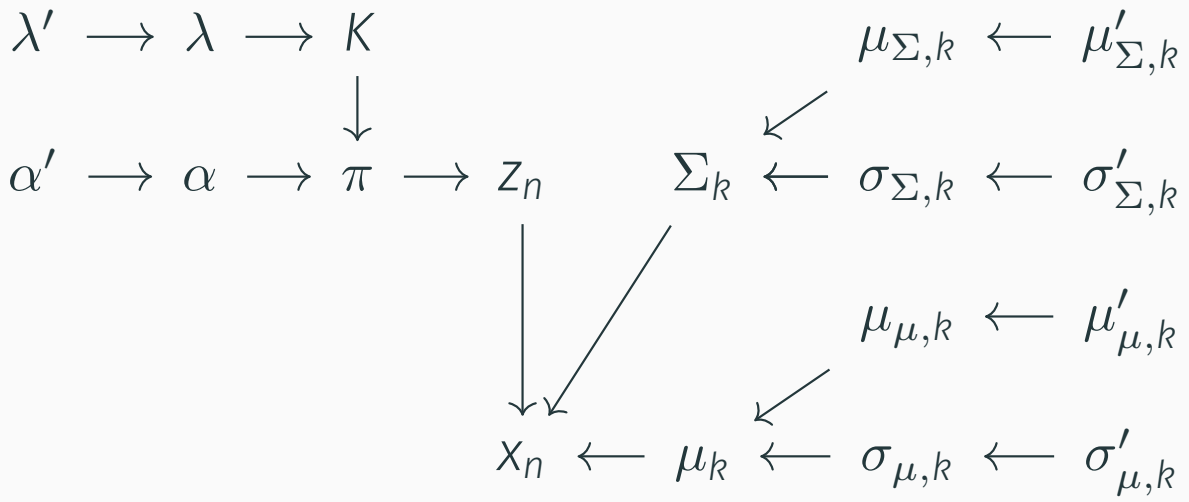


Figure 7: Independence structure of auto tuning GMM

Advantage of using ProbProg-3

Major application in using hierarchical MoE is that splitting condition can be included in inference. This allows regularization on depth of MoE while learning.

Using recursion, hierarchical MoE can by expressed as a finite tree, a structure that classical modeling in statistics can not simply express.

References

Figure 1, 2 : Mathematical Foundation for Artificial Intelligence (2019, Fall), Department of Mathematical Sciences, Prof. Ganguk Hwang.

Slide 5 : Anglican page, examples ‘GAUSSIAN MIXTURE MODEL: IRIS DATA’

Figure 4 : Machine Learning: a Probabilistic Perspective by Kevin Patrick Murphy, Ch 11.

Thanks!