CS281: Advanced ML **October 25, 2017**

Lecture 14: Mixture Models

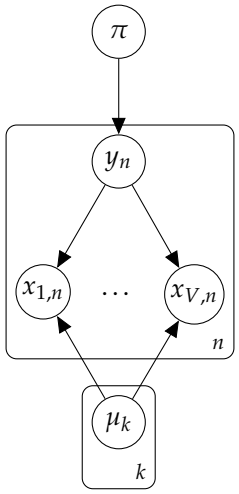
Lecturer: Sasha Rush *Scribes: Eric Dunipace, Andrew Fai, Isaac Xia*

14.1 Administrative Announcements

Overview: Second half of class is on Inference, especially on harder problems. There is only one problem set left, but the bulk of the time will be on the project.
There is some extra time left in the schedule, for which Prof. Rush will poll topics from the class.

14.2 Mixture Models: Introduction

Up until now, we've been considering mostly supervised learning, where we had an input x, an output y, and parameters θ.
Previously when doing MLE, MAP, marginalizing over nodes in a graphical model, we weren't trying to find a distribution, we were finding a **point-estimation** rather than a **distribution**.
For example, suppose we want to think about models from Naive Bayes:

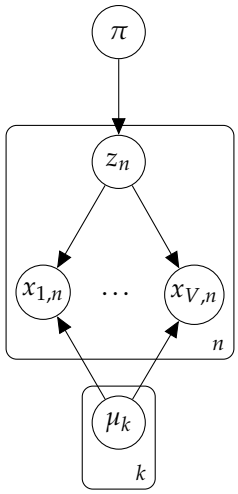


with joint probability

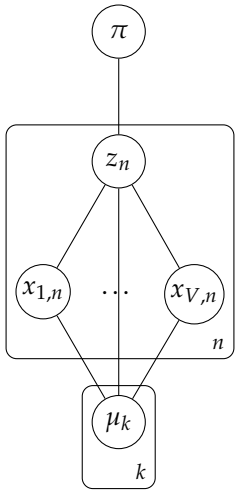
$$p(y|\pi)\prod_{j=1}^{V} p(x_j|y,\mu),$$

where the priors could be Categorical on $p(y|\pi)$ and Bernoulli on each $p(x_j|y,\mu)$.
In this setting, with data $\{x_i, y_i\}$, we can perform log-likelihood maximization with $p(\{x\},\{y\})$ to infer the parameters μ and π.

In a new setting, instead suppose we have the same graphical plate model, but we now do not observe the categories; instead of y_n, we denote these latent variables as z_n.



which also can be represented with the following undirected graphical model:



What does this graphical model look like, and what dependencies do we need to consider?

- There's clearly a dependency on π, but we can't observe z_n.
- There are the μ_k from the x_{in}.

All of these Naive Bayes models without observed y_n are used for **different types of clustering**. These models are referred to as **mixture models**, and can handle multimodality well.

14.3 Types of Mixture Models

Up until now, most models we've used have unimodal distributions, e.g. normal distributions.

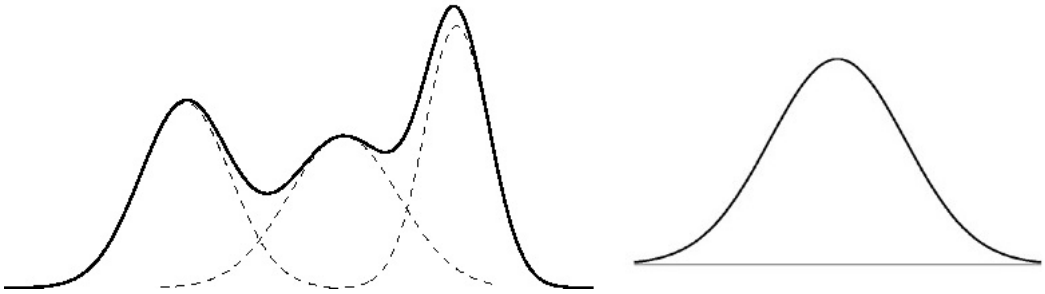


Figure 14.1: Multimodality of Multiple Gaussians vs Unimodality of One Gaussian

Although we won't limit ourselves to only using continuous mixture models, we'll be primarily working with mixtures of Gaussians. Below are examples of mixture models we'll see more of.

14.3.1 Mixture of Gaussian Modes

We can have a mixture of Gaussian models, by simply mixing different Gaussian conditional distributions together:

$$p(x|y) \sim \mathcal{N}(\mu, \Sigma)$$

An application example is speech recognition, which must deal with different accents (e.g. British vs American) for pronunciations of the same word. In this case, Gaussian mixture models are able to consolidate the seemingly different pronunciations.

14.3.2 Mixture of Bernoulli/Multinomial Models

We can instead mix together various Bernoulli or Multinomial conditionals.

$$p(x_j|y) \sim \text{Bern}(\mu_j) \text{ or } \text{Multinom}(\alpha)$$

14.4 Math behind Mixture Models

Now, we delve more deeply into the math behind using such mixture models. To start, the models involve x (data), z (latent variables), π (prior of latent variable), and $\{\mu_k\}$ (parameters for conditional distribution). We will use x and z to denote all n observations of the data and latent variables.

The **complete data likelihood** of the model is then

$$p(x, z) = \prod_n \prod_k \pi_k p(x_n|\mu_k)^{z_{nk}},$$

where z_{nk} is shorthand for $\mathbf{1}(Z_n = k)$.
To get the **marginal likelihood** of the data, we can marginalize and sum over all possible values of z_n:

$$p(x) = \sum_z \prod_n \prod_k \pi_k p(x_n|\mu_k)^{z_{nk}} = \prod_n \underbrace{\sum_z}_{\text{Hard to compute}} \prod_k \pi_k p(x_n|\mu_k)^{z_{nk}}$$

Although we pushed the sum into the product, the sum over all possible values of z is still hard to compute. Our main alternative strategy is thus to utilize the complete data likelihood in order to find $p(x)$.
Below is a heuristic algorithm to find $p(x)$:

1. Initialize π and μ randomly.
2. Compute the log-likelihood $p(x, z|\pi, \mu) = \sum_n \sum_k z_{nk} \log \pi_k + z_{nk} \log p(x_n|\mu_k)$ using fixed π and μ.
3. Denote $q_{nk} = p(z_n = k|x_n; ...)$ as a "hallucinated" distribution over z_{nk}; the origin of q_{nk} is unclear, but it satisfies the given requirements. We can use this distribution of $z \sim q_{nk}$ to compute the expectation of the log-likelihood as:

$$\mathbb{E}_{z \sim q_{nk}} [\log p(x_n, z_n|\pi, \mu)] = \sum_n \sum_k q_{nk} \log \pi_k + q_{nk} \log p(x_n|\mu_k).$$

   We can then "hallucinate" parameters through the distribution of z_{nk}! Returning to the graphical model, recall that it was difficult to actually compute the parameters. Howevere, given some parameters π, μ, we can compute a marginal distribution over the z: $p(z|x, \pi, \mu)$.
4. We thus can perform a coordinate ascent algorithm; that is, we optimize one set of parameters, then the other, then continue to alternate optimization.

This heuristic is summarized below:

1: **procedure** EM_NB(x, z)
2: Initialize π and μ randomly.
3: **while** π and μ not converged **do**
4: (*Expectation*) Compute $q_{nk} = p(z|x, \pi, \mu)$ using fixed π and μ
5: (*Maximization*) Compute MLE of π and μ using q through $\mathbb{E}_{z \sim q} [\log p(x, z)]$

The above is called the **Expectation Maximization** (EM) Algorithm, where step 4 computes the actual expectation maximization, and step 5 actually maximizes through the MLE. π will be Categorical and μ will be the means of the Gaussians themselves here.

14.5 Expectation Maximization Algorithm

The above algorithm can be generalized, and is known as the **Expectation Maximization** (EM) algorithm.

1: **procedure** EM(x,z)
2: Initialize π and μ randomly.
3: **while** π and μ not converged **do**
4: (*Expectation*) $q_{nk} \leftarrow \dfrac{\pi_k p(x_n|\mu_c)}{\sum_{k'} \pi_{k'} p(x_n|\mu_{k'})}$
5: (*Maximization*) $\mathbb{E}_{z \sim q} [\log p(x, z|\pi, \mu)] = \sum_n \sum_k q_{nk} \log \pi_k + q_{nk} \log p(x_n|\mu_k)$

We'll thinking about this algorithm in the context of information theory. In the *maximization* step, we have the first term $\sum_n \sum_k q_{nk} \log \pi_k$ can be thought about as maximizing the π_k under the model, with the assumption they were sampled from distribution of q_{nk}.

$$\sum_n \sum_k \underbrace{q_{nk}}_{\text{Came from } q_{nk} \text{ guess}} \log \left(\underbrace{\pi_k}_{\text{Want to find } \pi_k} \right)$$

So, we can think of q_{nk} as "expected counts" from a distribution. So, the MLE is similar to standard counts of observations. In particular,

$$\pi_k = \frac{\sum\limits_n q_{nk}}{\sum\limits_{n,k} q_{nk}}.$$

Alternatively, we could also use the MAP estimate. The main point is that the prior is embedded in the max step.
A similar update happens for μ, but will be specific to the actual form of the model $p(x|\mu)$.

14.5.1 Assumptions on EM

One key point is that the model yields an easily computable $p(z|x, \pi, \mu)$. If we imagine that $p(x|z)$ is an arbitrary neural net with continuous output, this could be very difficult. A specific example is a Variational Autoencoder (VAE), which could be found from Kingsma and Welling.

14.6 Demo on EM

We did a demonstration on Expectation Maximization.
In general, we saw it was pretty hard to even infer the clusters of data especially as the number of true clusters increased.
Q: *How to infer number of true clusters?* It's actually a hard problem, no really good standout way to do this.
Q: *Difference between k-means and EM here?* k-means will choose q as the highest probability, instead of assigning a distribution to q_{nk} as EM does.

14.7 Theory behind Expectation Maximization

We'll discuss why the EM algorithm works; specifically, why q_{nk} eventually converges to $p(z_n|x_n, \pi, \mu)$.
First, consider the marginal distribution of the data $p(x) = \prod\limits_n \sum\limits_{z_n} p(x_n, z_n|\pi, \mu)$. We have that the log-marginal is (in terms of entropy)

$$\begin{aligned}
\log p(x) &= \sum_n \log \left[\sum_{z_n} p(x_n, z_n, |\pi, \mu) \right] \\
&= \sum_n \log \left[\sum_{z_n} q(z_n = k) \frac{p(x_n, z_n|\pi, \mu)}{q(z_n = k)} \right] \\
&= \sum_n \log \mathbb{E}_q \left[\frac{p(x_n, z_n|\pi, \mu)}{q(z_n = k)} \right] \\
&\geq \sum_n \sum_k q(z_{nk}) \log \frac{p(x_n, z_n|\pi, \mu_k)}{q(z_n)} \\
&= \left[\sum_n \sum_k q(z_{nk}) \log p(x_n, z_n|\pi, \mu_k) \right] - \sum_n \left[\sum_k q(z_{nk}) \log q(z_n) \right] \\
&= \sum_n \sum_k q_{nk} \log p(x_n, z_n = k|\pi, \mu_k) + \sum_n H\left[q(z_n)\right]
\end{aligned}$$

In the second line we only multiplied by $1 = \frac{q(z_n = k)}{q(z_n = k)}$. In the third line, we use the trick of applying Jensen's inequality when we see the log of an expectation.

In now analyzing the M-step, we are computing

$$\pi,\mu \leftarrow \underset{\pi,\mu}{\text{argmax}} \sum_n \sum_k q_{nk} \log \pi_k + q_{nk} \log p(x_n|\mu_k)$$

where $\sum_k q_{nk} \log \pi_k$ is the cross-entropy between parameters and expected counts.
The E-step, we find q_{nk}.

$$\begin{aligned}\arg\min_q \sum_n \left[\sum_k q(z) \log\left(\frac{p(z|x)p(x)}{q(z)}\right)\right] &= \arg\min_q \sum_n \left[\sum_k q(z) \log \frac{p(z|x)}{q(z)} + \text{const}\right] \\ &= \arg\min_q \sum_n KL\left(q(z) \,||\, p(z|x)\right),\end{aligned}$$

and we can minimize the KL divergence term by setting $q_{nk} = p(z_n|x_n, \pi, \mu)$ as the same distribution.

14.8 Problems and Answers

14.8.1 Comparing a Mixture Model and Hurdle Model

Recall the hurdle model in Homework 2, Problem 2. This allowed us to account for a bimodal distribution of count data with one mode on 0, and one mode with a Poisson distribution. Alternatively, we could model this data with a mixture model.
This is the hurdle model included in the previous problem set:

$$\begin{aligned}P(y = 0 \mid x) &= 1 - \pi \\ P(y = j \mid x) &= \pi \frac{f(j \mid \mathbf{x})}{1 - f(0 \mid \mathbf{x})},\ j = 1, 2, ...\end{aligned}$$

One choice of parameterization is to use a logistic regression model for π:

$$\pi = \sigma(\mathbf{x}^\top \mathbf{w}_1)$$

and use a Poisson GLM for fx with mean parameters λ (see Murphy 9.3):

$$\lambda = \exp(\mathbf{x}^\top \mathbf{w}_2)$$

Mixture Model: Consider a mixture of Poisson GLMS with 2 latent classes: z = 0 and z = 1. The Poisson GLM used when z = 0 described the people who rarely go to the gym. The Poisson GLM used when z = 1 describes the people who go to the gym frequently.

(a) **Question**: Intuitively, why might modeling this with a latent variable be useful?
Answer: The hypothesis is that there are 2 groups of people in this dataset: one group of gym goers who randomly miss the gym some weeks, and another set of non-gym goers who randomly go to the gym some weeks. A latent variable model could distinguish these 2 groups and learn a Poisson GLM for each of them. This allows us to better model the random weeks where gym-goers miss the gym, and non gym-goers hit the gym. This is also a particularly useful framework in the case where we know that there are 2 groups of people, but we don't know the gym-going habits of each group. E.g. if group one goes to the gym once most weeks instead of zero times, we wouldn't be able to model it as a hurdle model unless we knew that ahead of time.

(b) **Question**: Write the probability of y if we parameterize this with a mixture of Poisson GLMs determined by a latent class indicator, z.
Answer: $P(y = j \mid x, z) \sim Poi(\exp(\mathbf{x}^\top \mathbf{w}_z))$

(c) **Question**: How would you learn the parameters with E.M.?
Answer: Initialize zs and ws randomly
Expectation step: $z = argmax_k P(y = j|x, z = k)$
Maximization step: $w_z = argmax_{w_z} P(y = j|x, z)$
Iterate until convergence.