Learning mixture models

Stephen Tu

August 30, 2015

Abstract

This note is completely expository, and contains a whirlwind abridged introduction to the topic of mixture models by focusing on the application of clustering. More detailed and complete expositions are available in the literature; for instance, the standard machine learning texts (e.g. [Mur12, Bis06]) provide a thorough treatment of this material.

1 Introduction

We will focus on the problem of clustering a set of n points $x_1, ..., x_n$, where $x_i \in \mathcal{X}$. The $\mathcal{X}$ here is an abstract set; typical examples include $\mathbb{R}^d, \{0,1\}^d, \{1,2,...,k\}^d$. We assume the number of clusters k is known and fixed. Non-parametric methods exist which relax this assumption, but they are beyond the scope of this exposition.

1.1 k-means/medians/medoids clustering

The most basic approach to clustering attempts to cluster points by working directly in some metric space induced by $\mathcal{X}$. The most famous (and most widely used, probably) one is k-means, which assumes there is some Euclidean metric on $\mathcal{X}$. For now on, let us just assume $\mathcal{X} = \mathbb{R}^d$, and work with the ℓ_2 norm: $\|x\|^2 := \sum_{i=1}^d x_i^2$. k-means simply aims to solve

$$\operatorname*{argmin}_{S_1,...,S_k} \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2,$$

where $\mu_i := \frac{1}{|S_i|} \sum_{x \in S_i} x$. The standard heuristic algorithm for solving this problem works by:

1. $s_i \leftarrow \operatorname{argmin}_{1 \le j \le k} \|x_i - \mu_j^{(t)}\|$.
2. $\mu_j^{(t+1)} \leftarrow$ mean of points now assigned to the j-th cluster.

Solving the k-means optimization problem in general is NP-hard (in k and d). This algorithm, called Lloyd's algorithm, is a heuristic which actually works pretty well in practice, but *can* get stuck at local minima. See k-means++ [AV07] for an initialization scheme (for the means) which also works really well.

2 A probabilistic approach

A drawback (and arguably feature) of k-means as discussed above is that it makes no effort to model the data probabilistically. It assumes that clustering is purely a geometric problem. While this makes a lot of sense when $\mathcal{X} = \mathbb{R}^d$, it makes less sense e.g. when $\mathcal{X} = \{\spadesuit, \bigstar, \clubsuit, \blacksquare\}^d$. One approach in this case is to develop

a generative model for how the data arises. Of course, keeping in mind that all models are wrong [1], we cannot hope to possibly specify explicitly how natural data arises. But nevertheless, this approach does yield some useful results.

Let us define explicitly what we mean by mixture model. As before, fix k. Let π live on the $(k-1)$-dimensional simplex (that is, the set of $\pi \in \mathbb{R}^k_{\geq 0}$ such that $\mathbf{1}^T\pi = 1$). Consider a family of likelihood models $\{F_j(\cdot) : 1 \leq j \leq k\}$, where each F_j is parametrized by some θ_j, and the support of $F_j(\cdot)$ is some subset of $\mathcal{X}$. For simplicity sake, we will now assume that $F_j \equiv F$, that is each likelihood model is the same (the *parameter* of the each likelihood model, however, can be and will be in general, different). The mixture model is simply the following probabilistic model:

$$\begin{aligned} z_i &\sim \text{Cat}(\pi) \\ x_i | z_i &\sim F(\theta_{z_i}) \,. \end{aligned}$$

For now, we will assume that our model parameters $(\pi, \{\theta_j\}_{j=1}^k)$ are fixed, but unknown. Later on, when we talk about Bayesian methods, we will put prior distributions on π and $\{\theta_j\}_{j=1}^k$, but let us not complicate things for now. Suppose that $F(\theta)$ admits a density[2] $p(\cdot;\theta)$. Then, our probabilistic model admits a density as

$$p(x; \pi, \{\theta_j\}_{j=1}^k) = \sum_{j=1}^k \pi_j p(x; \theta_j) \,.$$

Gaussian mixture models as an example. Consider a multivariate normal distribution, denoted as $\mathcal{N}(\mu, \Sigma)$. The Gaussian mixture model then is simply

$$\begin{aligned} z_i &\sim \text{Cat}(\pi) \\ x_i | z_i &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i}) \,, \end{aligned}$$

with the model parameters $(\pi, \{\mu_j, \Sigma_j\}_{j=1}^k)$. Recall that when $\Sigma \succ 0$ (which is the only case we will deal with), the PDF of a multivariate normal distribution is given by

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{k/2} \det(\Sigma)^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right\} .$$

Solving the clustering problem. Now, suppose someone gave us the correct model parameters. How can we use the mixture model to solve our original clustering problem? Well, observe that we can compute the posterior probabilities via Bayes rule for each z_i as

$$\mathbb{P}(z_i = j | x_i; \pi, \theta) = \frac{\pi_j p(x_i; \theta_j)}{\sum_{j'=1}^k \pi_{j'} p(x_i; \theta_{j'})} \,. \tag{1}$$

This is neat– we actually have a distribution over which topic each data point belongs to. If we want a hard answer, we can take the (maximum a posteriori) MAP estimator:

$$\widehat{z_i} := \operatorname*{argmax}_{1 \leq j \leq k} \mathbb{P}(z_i = j | x_i; \pi, \theta) \,.$$

[1] "Essentially, all models are wrong, but some are useful." – George E.P. Box

[2] We will only deal with nice distributions which admit densities. No need to dig out those measure theory books!

2.1 The learning problem

Now that we have developed a probabilistic model, given a collection of data $x_1,...,x_n$, we can define a learning problem. Our learning problem is to infer $z_1,...,z_n$ from the data. Recall from Eq. 1 that we can accomplish this by learning $\pi, \{\theta_j\}_{j=1}^k$ from the data. A natural question to ask is, why do we not try to learn $z_1,...,z_n$ directly? As will become clear, this is a combinatorial optimization problem and requires (in the general case) k^n enumeration. However, we will develop techniques that let us optimize our parameters very efficiently, sidestepping this issue[3]. To make our life easy, we will make a very critical assumption, that the data is generated i.i.d. from our mixture model.

Now, there are actually two problems we need to address here: (a) what *statistical* method do we want to use to estimate our unknown parameter, and (b) what *computational* procedure do we use to implement our estimator from (a)? Note, a lot of ink has been spilled on both (a) and (b), so rather than try to rehash these arguments, I will just talk about the most commonly used methods.

The statistical method we will focus on here is the principle of *maximum likelihood* estimation. This is a very general principle, and I will talk about it generically at first. Suppose we have a parametric family of distributions $\mathcal{P} := \{P_\theta : \theta \in \Theta\}$. Given some data x, we will estimate θ via the MLE estimator

$$\widehat{\theta} := \operatorname*{argmax}_{\theta \in \Theta} P_\theta(x)\,.$$

I will make no attempt to justify the statistical performance of this estimator other than to say that the MLE estimator is the minimum variance unbiased estimator (MVUE)– see e.g. the relevant chapters in [Mur12].

Turning back to our mixture model learning problem, observe that our family of distributions is given by

$$\log p(x_1,...,x_n;\pi,\theta) = \log \prod_{i=1}^n p(x_i;\pi,\theta) = \sum_{i=1}^n \log \sum_{j=1}^k p(x_i,z_i;\pi,\theta)\,,$$

and hence our MLE estimator is[4]

$$\operatorname*{argmax}_{\pi,\theta} \sum_{i=1}^n \log \sum_{j=1}^k p(x_i,z_i;\pi,\theta)\,.$$

In general, this function is a nasty non-concave function, so generic local search methods (e.g. gradient ascent) will not work all the time. This motivates the study of a practical algorithm to estimate (π,θ).

2.2 Expectation-maximization (EM) algorithm

Rant: Before we begin our study, I must first say that a *lot* of ink has been spilled on this algorithm (a quick google search reveals about 800k results). You would think that with such titles as "EM Demystified" and "A Gentle Tutorial of the EM Algorithm", all pedagogical issues surrounding this algorithm would be resolved by now. However, I must admit that I find most explanations to be completely incomprehensible. Below, I will try to provide what is, in my opinion, the most clear exposition of the EM algorithm (which I borrowed from MIT's 6.437 course). But of course, you might disagree.

The general setup for EM works as follows. Suppose we have a generative model on $F_\theta(x,z)$, but we only get to observe x. We are interested in solving

$$\operatorname*{argmax}_{\theta \in \Theta} \ell(\theta) := \log \sum_z p(x,z;\theta)\,,$$

[3] These techniques are, strictly speaking, heuristics, but they work pretty well in practice.

[4] Recall that log is monotonically increasing.

where I have assumed that z is discrete (these results mostly extend to continuous r.v.s by replacing sums with integrals), and I have absorbed the π into the θ for notational simplicity. The EM algorithm iterates the following two steps:

1. (**E-step**) $Q(\theta, \theta^{(t)}) := \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(x,z;\theta)]$.
2. (**M-step**) $\theta^{(t+1)} \leftarrow \text{argmax}_{\theta\in\Theta} Q(\theta, \theta^{(t)})$.

This looks very simple, but why does this procedure work? I first present the intuitive reason for why this works, and then sketch a more formal proof. The intuitive argument starts by observing that for any distribution $q(z)$,

$$\begin{aligned}
\log \sum_z p(x,z;\theta) &= \log \sum_z \frac{q(z)}{q(z)} p(x,z;\theta) \\
&= \log \mathbb{E}_{z\sim q(z)}[\frac{1}{q(z)} p(x,z;\theta)] \\
&\overset{(a)}{\geq} \mathbb{E}_{z\sim q(z)}[\log p(x,z;\theta) - \log q(z)] \\
&= \mathbb{E}_{z\sim q(z)}[\log p(x,z;\theta)] + H(q)\,.
\end{aligned}$$

where in (a) we used Jensen's inequality[5], and where $H(q) := \mathbb{E}_{z\sim q(z)}[-\log q(z)]$ is the (Shannon) entropy of the distribution $q(z)$. Now, since $q(z)$ was arbitrary, we can pick any distribution we want. At time step t, we have available to us $p(z|x;\theta^{(t)})$, so we might as well use that. From this, we see immediately that the EM algorithm works by maximizing a lower bound on $\ell(\theta)$ (the entropy term is not a function of θ).

We now outline a more formal argument. Before we continue, we first state a few basic definitions from information theory which will make the exposition much simpler.

2.2.1 Basic tools from information theory

In what follows, let $p(x)$ and $q(x)$ denote discrete distributions on some common probability space.

Definition 1. *(KL-divergence between two distributions) Let* $D(p,q) := \mathbb{E}_p[\log \frac{p(x)}{q(x)}]$.

Now a simple fact which we will not bother to prove (use Jensen's inequality).

Proposition 1. *(Gibbs' inequality)* $D(p,q) \geq 0$, *with* $D(p,q) = 0$ *iff* $p = q$.

2.2.2 Proof that the EM algorithm "works"

We have to define what "works" means. We formalize this in the following proposition.

Proposition 2. *Let* $\theta^{(t)}$ *denote the* t*-th iterate of the EM algorithm. Then* $\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)})$.

Proof. (This proof is courtesy of Greg Wornell from MIT.) The proof starts by first noting the following identity:

$$\log p(x;\theta) = \log p(x,z;\theta) - \log p(z|x;\theta)\,.$$

Since z only appears on the RHS, we can take the expectation of both sides w.r.t. any distribution on z and obtain

$$\log p(x;\theta) = \mathbb{E}_{z\sim q(z)}[\log p(x,z;\theta)] - \mathbb{E}_{z\sim q(z)}[\log p(z|x;\theta)]\,.$$

[5] Jensen's inequality states that for convex $f(x)$, $f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$.

In our expression above, both θ and $q(\cdot)$ were arbitrary. So setting $\theta = \theta^{(t+1)}$ and $q(z) = p(z|x;\theta^{(t)})$, we get

$$\begin{aligned}\log p(x;\theta^{(t+1)}) &= \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(x,z;\theta^{(t+1)})] - \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(z|x;\theta^{(t+1)})] \\ &= Q(\theta^{(t+1)},\theta^{(t)}) - \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(z|x;\theta^{(t+1)})] \\ &\overset{(a)}{\geq} Q(\theta^{(t)},\theta^{(t)}) - \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(z|x;\theta^{(t+1)})]\,.\end{aligned}$$

where (a) comes by definition of the M-step. We now must figure out what to do with this curious quantity on the RHS. Let us define a shorthand $p_\theta := p(z|x;\theta)$. Then,

$$\begin{aligned}-\mathbb{E}_{p_{\theta^{(t)}}}[\log p_{\theta^{(t+1)}}] &= -\mathbb{E}_{p_{\theta^{(t)}}}[\log p_{\theta^{(t+1)}} - \log p_{\theta^{(t)}} + \log p_{\theta^{(t)}}] \\ &= -\mathbb{E}_{p_{\theta^{(t)}}}[\log \frac{p_{\theta^{(t+1)}}}{p_{\theta^{(t)}}}] - \mathbb{E}_{p_{\theta^{(t)}}}[\log p_{\theta^{(t)}}] \\ &= D(p_{\theta^{(t)}}, p_{\theta^{(t+1)}}) - \mathbb{E}_{p_{\theta^{(t)}}}[\log p_{\theta^{(t)}}] \\ &\overset{(a)}{\geq} -\mathbb{E}_{p_{\theta^{(t)}}}[\log p_{\theta^{(t)}}]\,.\end{aligned}$$

where in (a) we used Gibbs' inequality and lower bounded the KL-divergence by zero. Now we are done, since

$$\begin{aligned}Q(\theta^{(t)},\theta^{(t)}) - \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(z|x;\theta^{(t+1)})] &\geq Q(\theta^{(t)},\theta^{(t)}) - \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(z|x;\theta^{(t)})] \\ &= \log p(x;\theta^{(t)})\,,\end{aligned}$$

and therefore we have shown $\log p(x;\theta^{(t+1)}) \geq \log p(x;\theta^{(t)})$, which is what we wanted to show. □

2.3 Derivation of EM algorithm for GMM

In this section we derive the EM algorithm specific to the Gaussian mixture model. We do this by following the algorithm prescribed in the previous section. To ease notation, define r_{ij} as

$$r_{ij} := \mathbb{P}(z_i = j|x_i;\theta^{(t)}), \qquad r_j := \sum_{i=1}^{n} r_{ij}\,.$$

We want to compute $Q(\theta,\theta^{(t)})$. This is given as

$$\begin{aligned}Q(\theta,\theta^{(t)}) &= \mathbb{E}_{z\sim p(z|x;\theta^{(t)})}[\log p(x_1,...,x_n,z_1,...,z_n;\theta)] \\ &= \sum_{i=1}^{n} \mathbb{E}_{z_i\sim p(z_i|x_i;\theta^{(t)})}[\log p(x_i,z_i;\theta)] \\ &= \sum_{i=1}^{n} \mathbb{E}_{z_i\sim p(z_i|x_i;\theta^{(t)})}[\log p(z_i)] + \mathbb{E}_{z_i\sim p(z_i|x_i;\theta^{(t)})}[\log p(x_i|z_i;\theta)] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}\log\pi_j + \mathbb{E}_{z_i\sim p(z_i|x_i;\theta^{(t)})}[\log p(x_i|z_i;\theta)]\,.\end{aligned}$$

Note that our objective is separable in terms of the π's and the θ's. Furthermore, so far everything we have done applies to any mixture model; immediately we see that the M-step update for π is the same for any mixture model.

Updating π. We want to optimize the following function of π:

$$\pi^{(t+1)} = \underset{\pi_1,...,\pi_k}{\operatorname{argmax}} \sum_{j=1}^{k} r_j \log \pi_j : \pi_1 + ... + \pi_k = 1, \pi_j \geq 0, j = 1, ..., k\,.$$

This is a nice (constraint) concave maximization program, so we can simply set the gradient of the Lagrangian to zero. That is, introducing Lagrange multiplier λ, our Lagrangian is

$$\mathcal{L}(\pi_1, ..., \pi_k, \lambda) = \sum_{j=1}^{k} r_j \log \pi_j + \lambda(\sum_{j=1}^{k} \pi_j - 1)\,.$$

By setting $\nabla_\pi \mathcal{L} = 0$, we get that

$$\pi_j^{(t+1)} = \frac{r_j}{n}, j = 1, ..., k\,.$$

Updating μ and Σ. We first expand out $\mathbb{E}_{z_i \sim p(z_i|x_i;\theta^{(t)})}[\log p(x_i|z_i;\theta)]$ by plugging in the Gaussian PDF:

$$\mathbb{E}_{z_i \sim p(z_i|x_i;\theta^{(t)})}[\log p(x_i|z_i;\theta)] = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{k} r_{ij}[\log\det(\Sigma_j) + (x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j)]\,.$$

Notice the first two terms do not depend on μ, so we can ignore them. Summing over our data points, we want to solve

$$\mu^{(t+1)} = \underset{\mu_1,...,\mu_k}{\operatorname{argmin}} \frac{1}{2}\sum_{j=1}^{k}\sum_{i=1}^{n} r_{ij}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j)\,.$$

Since each $r_{ij} \geq 0$ and $\Sigma_j^{-1} \succ 0$, this is a (unconstrained) convex minimization problem, so we can just set the gradient to zero. Recalling that for any $A = A^T$ and any b,

$$\nabla_x \left\{\frac{1}{2}(b - x)^T A(b - x)\right\} = -A(b - x)\,,$$

it is not hard to see that the update is given by

$$\mu_j^{(t+1)} = \frac{\sum_{i=1}^{n} r_{ij} x_i}{r_j}\,.$$

Now, updating Σ is conceptually the same as with μ, but requires a bit more machinery from matrix calculus[6]. We will need both facts below. Let X and A both be symmetric. Then,

$$\nabla_X \mathbf{Tr}(X^{-1}A) = -X^{-1}AX^{-1}, \qquad \nabla_X \log\det(X) = X^{-1}\,.$$

Our goal now is to minimize the following function

$$\Sigma^{(t+1)} = \underset{\Sigma}{\operatorname{argmin}}\, g(\Sigma_1, ..., \Sigma_k) := \sum_{j=1}^{k} r_j \log\det(\Sigma_j) + \sum_{i=1}^{n} r_{ij}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j),\ \Sigma_j \succ 0, j = 1, ..., k\,.$$

Typically, derivations of EM sort of "ignore" the positive definite constraint and simply set $\nabla_\Sigma g = 0$. Doing so yields

$$\nabla_{\Sigma_j} g = r_j \Sigma_j^{-1} - \sum_{i=1}^{n} r_{ij}\Sigma_j^{-1}(x_i - \mu_j)(x_i - \mu_j)^T \Sigma_j^{-1} = 0 \Longrightarrow \Sigma_j^{(t+1)} = \frac{1}{r_j}\sum_{i=1}^{n} r_{ij}(x_i - \mu_j)(x_i - \mu_j)^T\,.$$

Observe that, when doing this, nothing prevents the next iterate of Σ from becoming singular. One can add ϵI_d to the estimate for some $\epsilon > 0$ to ensure non-degeneracy, but this is mostly a hack.

[6] See the Matrix Cookbook (`http://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`) for many useful facts.

Putting it together. We can now describe the entire EM algorithm, specialized to the GMM. Let $\pi^{(0)}, \mu^{(0)}, \Sigma^{(0)}$ denote the initial guess at the latent parameters. Repeat the following until some termination condition is reached (e.g. the parameters have stopped changing significantly).

(a) (E-Step). Calculate the r_{ij}'s and r_j's via the formulas above.

(b) (M-Step). Update

$$\begin{aligned}
\pi_j^{(t+1)} &= \frac{r_j}{n} \\
\mu_j^{(t+1)} &= \frac{\sum_{i=1}^n r_{ij} x_i}{r_j} \\
\Sigma_j^{(t+1)} &= \frac{1}{r_j} \sum_{i=1}^n r_{ij} (x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T .
\end{aligned}$$

3 Bayesian mixture models: extending the probabilistic approach

In our previous discussion, we stipulated a probabilistic model for the likelihood of the data, but then proceeded to assume that our prior parameters were fixed, but unknown. The Bayesian way of thinking asks why stop there? Indeed, we can add a probabilistic model on the *parameters* of the model, just as we had a probabilistic model for the data.

3.1 A brief introduction to Bayesian inference

Let us forget about the mixture model for a second, and setup a general framework for Bayesian inference. Suppose we are given a set of data x generated by a probabilistic model $p(x|\theta)$, and we want to learn $p(\theta|x)$. Using Bayes's rule, this is simply

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} = \frac{p(x|\theta)p(\theta)}{\int_{\theta'} p(x|\theta')p(\theta')\, d\theta'} .$$

The way to think about this is as follows. Before I see my data, I have some belief on what the prior distribution is, $p(\theta)$. This belief is informed by, say e.g. my background knowledge on the problem at hand. Now, given a set of observations, I update my belief to reduce my uncertainty about θ, and obtain a posterior distribution $p(\theta|x)$. This is the Bayesian way of thinking, and it is very appealing, since it speaks intuitively to how we as humans process information. However, in this framework, we are confronted with two main issues:

(a) What is a reasonable prior $p(\theta)$?

(b) How do I compute the normalizing factor to get my posterior belief?

These are both hard questions, and one could spend a lot of time on this. To address both issues at once (by sort of "punting"), we introduce the notion of conjugate priors.

3.1.1 Conjugate priors

The driving philosophy behind conjugate priors is that, since we do not really have a good idea for what prior $p(\theta)$ to pick, we might as well pick one that makes the math work out well. Sounds reasonable, right?

Let us make this very concrete. Suppose our likelihood model is the Bernoulli distribution Bern(p), where $p \in [0,1]$ is the probability of drawing a 1. We now need a distribution on p; notice that such a distribution must be supported only on $[0,1]$.

Beta distribution. Let me introduce this distribution first, and then we will see why it is a nice prior to pick for the Bernoulli distribution. The Beta distribution, written $\text{Beta}(\alpha, \beta)$, is parameterized by $\alpha, \beta > 0$. It has a PDF given by:

$$p(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} : x \in (0,1),$$

where $B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\, dx$ is the normalizing constant.

Now, suppose we have the following probabilistic model:

$$\begin{aligned} \theta &\sim \text{Beta}(\alpha, \beta) \\ x_i|\theta &\sim \text{Bern}(\theta). \end{aligned}$$

The Beta-Bernoulli model has a very convenient fact that the posterior distribution of θ given the data is another Beta distribution! Specifically,

$$p(\theta|x_1, ..., x_n) \sim \text{Beta}(\alpha + \sum_{i=1}^n x_i, \beta + n - \sum_{i=1}^n x_i).$$

To see this, observe that

$$\begin{aligned} p(\theta, x_1, ..., x_n) &= p(\theta)p(x_1, ..., x_n|\theta) \\ &= \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} \prod_{i=1}^n \theta^{x_i}(1-\theta)^{1-x_i} \\ &= \frac{1}{B(\alpha, \beta)} \theta^{\alpha+\sum_{i=1}^n x_i - 1}(1-\theta)^{\beta+n-\sum_{i=1}^n x_i - 1}. \end{aligned}$$

Hence,

$$\begin{aligned} p(x_1, ..., x_n) &= \int_0^1 \frac{1}{B(\alpha, \beta)} \theta^{\alpha+\sum_{i=1}^n x_i - 1}(1-\theta)^{\beta+n-\sum_{i=1}^n x_i - 1}\, d\theta \\ &= \frac{B(\alpha + \sum_{i=1}^n x_i - 1, \beta + n - \sum_{i=1}^n x_i)}{B(\alpha, \beta)}. \end{aligned}$$

Immediately, we see that

$$\begin{aligned} p(\theta|x_1, ..., x_n) &= \frac{p(\theta, x_1, ..., x_n)}{p(x_1, ..., x_n)} \\ &= \frac{1}{B(\alpha + \sum_{i=1}^n x_i - 1, \beta + n - \sum_{i=1}^n x_i)} \theta^{\alpha+\sum_{i=1}^n x_i - 1}(1-\theta)^{\beta+n-\sum_{i=1}^n x_i - 1}, \end{aligned}$$

which is exactly the PDF for a $\text{Beta}(\alpha + \sum_{i=1}^n x_i, \beta + n - \sum_{i=1}^n x_i)$ distribution as claimed.

It turns out that this convenience is not just for the Beta-Bernoulli model, but also for many other models. When such a relation holds, we call the prior-likelihood pair as *conjugate priors*. Wikipedia has a whole list of these pairs which I encourage you to check out[7]. We will point out another one, the Dirichlet-Categorical[8] pair.

[7] See `https://en.wikipedia.org/wiki/Conjugate_prior`.

[8] This is sometimes referred to as the Dirichlet-Multinomial. See [Tu14] for clarification.

Dirichlet distribution. The Dirichlet distribution, written Dirichlet$(\alpha_1,...,\alpha_k)$, is a k-dimensional generalization of the Beta distribution ($k = 2$ can be thought of as a Beta distribution). It is parametrized by $(\alpha_1,...,\alpha_k) \in \mathbb{R}^k_{>0}$ with PDF

$$p(x_1,...,x_k;\alpha_1,...,\alpha_k) = \frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{i=1}^k x_i^{\alpha_i - 1} : (x_1,...,x_k) \in \mathcal{S}^{k-1},$$

where $\mathcal{S}^{k-1}$ denotes the open $(k-1)$-dimensional simplex.

Similar to before, suppose we have the model:

$$\begin{aligned} \theta &\sim \text{Dirichlet}(\alpha_1,...,\alpha_k) \\ z_i|\theta &\sim \text{Cat}(\theta). \end{aligned}$$

Then,

$$p(\theta|z_1,...,z_n) \sim \text{Dirichlet}(\widehat{\alpha}_1,...,\widehat{\alpha}_k),$$

where $\widehat{\alpha}_j = \alpha_j + \sum_{i=1}^n \mathbf{1}\{z_i = j\}, j = 1,...,k$. The proof of this follows the same line of reasoning as with the Beta-Bernoulli case.

3.2 Extending to mixture models

Now we have a nice choice of prior distributions on our model parameters, we need to put everything back into our mixture model framework. Let us write down the probabilistic model for the Beta-Bernoulli mixture model (each $x_i \in \{0,1\}^d$ here):

$$\begin{aligned} \pi &\sim \text{Dirichlet}(\gamma/k,...,\gamma/k) \\ z_i|\pi &\sim \text{Cat}(\pi) \\ \theta_j^\ell &\sim \text{Beta}(\alpha,\beta) \\ x_i^\ell|z_i,\theta &\sim \text{Bern}(\theta_{z_i}^\ell). \end{aligned}$$

We simplify the model a bit in that we only use one γ to parameterize our Dirichlet distribution, and all $k \times d$ Beta-Bernoulli likelihood pairs share the same α,β. Here, we assume that γ,α,β are fixed and known. You could imagine also putting a prior on these prior parameters, but at *some* point you have to stop and assume something!

The natural question is, now, how do we go about learning the parameters? What does it even mean to learn the parameters? To be fully Bayesian, we want to learn *distributions*. That is, we are after

$$p(z,\pi,\theta|x).$$

Let us write down the full joint likelihood $p(x,z,\pi,\theta)$:

$$\begin{aligned} p(x,z,\pi,\theta) &= p(\pi)p(z|\pi)p(\theta)p(x|z,\theta) \\ &= p(\pi)\left[\prod_{i=1}^n p(z_i|\pi)\right]\left[\prod_{j=1}^k \prod_{\ell=1}^d p(\theta_j^\ell)\right]\left[\prod_{i=1}^n \prod_{\ell=1}^d p(x_i^\ell|z_i,\theta)\right] \end{aligned}$$

From this we see that specifying the joint $p(x,z,\pi,\theta)$ is easy. In theory, computing $p(x)$ (which in turn gives us $p(z,\pi,\theta|x)$) is therefore also easy; we just integrate out z,π,θ. However, this task is daunting, involving a combinatorial sum over z (k^n choices, to be exact). Remember we ran into the same problem before in Section 3.2, and we employed the EM algorithm to help us out. We cannot use EM here since we are after distributions instead of point estimates. We instead rely on the theory of Markov Chain Monte Carlo (MCMC) methods, specifically Gibbs sampling. At a high level, we will set up a procedure such that eventually this procedure will produce independent samples from our target posterior distribution. From these samples, we can then estimate quantities of our distribution of interest (e.g. MAP estimate).

3.2.1 Gibbs sampling in a nutshell

An oversimplified view of Gibbs sampling works as follows. Suppose we have a target distribution $p(x_1, ..., x_n)$ (note these x are not necessarily data points, they just denote a particular parameterization of $p(\cdot)$) which we wish to sample from, but directly obtaining samples from $p(\cdot)$ is difficult. However, if it is the case that sampling from $p(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$ is easy, then we can generate the $t+1$-th sample $x^{(t+1)}$ from $x^{(t)}$ via the update

$$x_i^{(t+1)} \sim p(x_i|x_1^{(t+1)}, ..., x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, ..., x_n^{(t)}),$$

where we cycle i from 1 to n. The reason this works is beyond the scope of this exposition; see e.g. [Liu08] for a more thorough treatment. For now it suffices to say that this sets up a Markov chain where $p(\cdot)$ is the stationary distribution, and as $t \to \infty$, we know that this Markov chain will converge to its stationary distribution so that we will essentially be drawing samples from $p(\cdot)$.

3.2.2 Gibbs sampling for mixture models

We can apply the Gibbs sampler from the previous section to the Beta-Bernoulli mixture model. We need to be able to sample from the following distributions:

$$p(z|\pi, \theta, x), \qquad p(\pi|z, \theta, x), \qquad p(\theta|z, \pi, x).$$

But these distributions are all easy to specify! $p(\pi|z)$ is easy by the fact that the Dirichlet prior is conjugate to the Categorical distribution. $p(\theta|z, \pi, x)$ is also easy, given the fact that the Beta prior is conjugate to the Bernoulli distribution. Finally, $p(z|\pi, \theta, x)$ is easy and was already done in Eq. 1 for the EM algorithm (remember the r_{ij}'s?).

3.2.3 Collapsed Gibbs sampling for mixture models

The Gibbs sampler specified previously is correct, but inefficient. We are really after z, but we are sampling a much higher dimensional distribution including π, θ. The collapsed Gibbs sampler simply integrates these parameters out, allowing us to sample z directly. To see how we can do this, note that

$$\begin{aligned} p(z_i|z_{\neg i}, x) &\propto p(z_i|z_{\neg i})p(x|z_i, z_{\neg i}) \\ &\propto p(z_i|z_{\neg i})p(x_i|x_{\neg i}, z_i, z_{\neg i})p(x_{\neg i}|z_{\neg i}) \\ &\propto p(z_i|z_{\neg i})p(x_i|x_{\neg i}, z_i, z_{\neg i}). \end{aligned}$$

Now both of these remaining expressions can be derived explicitly with our conjugate prior assumptions. Let the notation $Z_{\neg i}^j := \{i' \in [n] \setminus \{i\} : z_{i'} = j\}$. We have that

$$p(z_i = j|z_{\neg i}) = \frac{|Z_{\neg i}^j| + \frac{\gamma}{k}}{n - 1 + \gamma}, \qquad p(x_i|x_{\neg i}, z_i = j, z_{\neg i}) = \prod_{\ell=1}^{d} \frac{(\alpha + \sum_{i' \in Z_{\neg i}^j} x_{i'}^\ell)^{x_i^d} (\beta + |Z_{\neg i}^j| - \sum_{i' \in Z_{\neg i}^j} x_{i'}^\ell)^{1 - x_i^d}}{\alpha + \beta + |Z_{\neg i}^j|},$$

and therefore,

$$p(z_i = j|z_{\neg i}, x) \propto \frac{|Z_{\neg i}^j| + \frac{\gamma}{k}}{n - 1 + \gamma} \prod_{\ell=1}^{d} \frac{(\alpha + \sum_{i' \in Z_{\neg i}^j} x_{i'}^\ell)^{x_i^d} (\beta + |Z_{\neg i}^j| - \sum_{i' \in Z_{\neg i}^j} x_{i'}^\ell)^{1 - x_i^d}}{\alpha + \beta + |Z_{\neg i}^j|}.$$

We will demonstrate how $p(z_i = j|z_{\neg i})$ is derived, leaving the other conditional probability as an exercise. Define the (generalized) Beta function

$$B(\gamma_1, ..., \gamma_k) = \int_{x_1, ..., x_k} \prod_{j=1}^{k} x_j^{\gamma_j - 1} \, d\mathcal{S}^{k-1} \overset{(a)}{=} \frac{\prod_{j=1}^{k} \Gamma(\gamma_j)}{\Gamma(\sum_{j=1}^{k} \gamma_j)},$$

where $d\mathcal{S}^{k-1}$ denotes integration w.r.t. the $(k-1)$-dimensional simplex and where (a) is a well known fact. We have that for an arbitrary m,

$$\begin{aligned} p(z_1,...,z_m) &= \int_\pi p(z_1,...,z_m,\pi)\, d\mathcal{S}^{k-1} \\ &= \int_\pi p(\pi) \prod_{i=1}^{m} \pi_{z_i}\, d\mathcal{S}^{k-1} \\ &= \frac{1}{B(\gamma_1,...,\gamma_k)} \int_\pi \prod_{j=1}^{k} \pi_j^{\sum_{i=1}^n \mathbf{1}\{x_i=j\}+\gamma_j-1}\, d\mathcal{S}^{k-1}\,. \end{aligned}$$

Hence,

$$p(z_{\neg i}) = \frac{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^k_{\neg i}|+\gamma_k)}{B(\gamma_1,...,\gamma_k)}, \qquad p(z_i=j,z_{\neg i}) = \frac{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^j_{\neg i}|+1+\gamma_j,...,|Z^k_{\neg i}|+\gamma_k)}{B(\gamma_1,...,\gamma_k)}\,.$$

Therefore,

$$p(z_i=j|z_{\neg i}) = \frac{p(z_i=j,z_{\neg i})}{p(z_{\neg i})} = \frac{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^j_{\neg i}|+1+\gamma_j,...,|Z^k_{\neg i}|+\gamma_k)}{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^k_{\neg i}|+\gamma_k)}\,.$$

By our assumption that $\gamma_j = \gamma/k$, we get

$$\frac{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^j_{\neg i}|+1+\gamma_j,...,|Z^k_{\neg i}|+\gamma_k)}{B(|Z^1_{\neg i}|+\gamma_1,...,|Z^k_{\neg i}|+\gamma_k)} = \frac{\Gamma(|Z^j_{\neg i}|+1+\gamma/k)}{\Gamma(|Z^j_{\neg i}|+\gamma/k)} \frac{\Gamma(n-1+\gamma)}{\Gamma(n+\gamma)} \overset{(a)}{=} \frac{|Z^j_{\neg i}|+\gamma/k}{n-1+\gamma}\,,$$

where in (a) we used the fact that $\Gamma(t+1) = t\Gamma(t)$.

Our Gibbs sampler now only works on the space $(z_1,...,z_n)$ and does not have to explicitly sample the model parameters; this is potentially much more efficient.

References

[AV07] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *SODA*, 2007.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. 2006.

[Liu08] Jun Liu. *Monte Carlo Strategies in Scientific Computing*. 2008.

[Mur12] Kevin Murphy. *Machine Learning: A Probabilistic Perspective*. 2012.

[Tu14] Stephen Tu. The dirichlet-multinomial and dirichlet-categorical models for bayesian inference. `http://www.cs.berkeley.edu/~stephentu/writeups/dirichlet-conjugate-prior.pdf`, 2014.