Applied Probabilistic Programming & Bayesian Machine Learning

§4-1: Mixture Models + Bayesian Variable Selection

MIT IAP 2017 Max W. Shen

1 Mixture Models

A mixture model assumes the existence of a structure of subpopulations that arise from a discrete mixture of distributions.

One example of a mixture model is:

$$\begin{aligned} y_i &\sim N(\mu_{z_i}, \sigma) \\ z_i &\sim bernoulli(\lambda_i) \end{aligned} \tag{1}$$

Here, $z_i \in \{0, 1\}$ so we have two unknown means: μ_0 and μ_1. The parameter λ_i indicates the probability of x_i being centered at μ_0 vs. μ_1.

One useful application of mixture models is clustering, where λ_i can be interpreted as membership to a cluster. To extend model 1 beyond 2 clusters to k clusters, one can replace the bernoulli distribution by a categorical distribution, which generalizes the bernoulli distribution to the set $\{1, ..., k\}$.

Model 1 is an instance of a Gaussian mixture model due to the Gaussian likelihood of data given a cluster. K-means using L2 or Euclidean distance (e.g., $dist(x, y) = \sqrt{(x-y)^2}$) is theoretically equivalent to a spherical (unit covariance matrix) Gaussian mixture model with K clusters.

1.1 Stan Implementation

Stan implements mixture models by marginalizing out the contribution of each mixture. That is,

$$\begin{aligned} P(y_i) &= \sum_{i=1}^{k} P(y_i, \mu_{z_i}, \sigma) \\ P(y_i) &= \sum_{i=1}^{k} P(y_i | \mu_{z_i}, \sigma) P(z_i, \sigma) \\ P(y_i) &= \sum_{i=1}^{k} P(y_i | \mu_{z_i}, \sigma) P(z_i) P(\sigma) \end{aligned} \tag{2}$$

For computational stability, it is preferable to perform the inner multiplication series under a log scale, then exponentiating to perform the outer $\sum$ sum, and finally taking the log again to return the log likelihood value. This is achieved in Stan by log-sum-exp.

In addition, Stan then requires you to explicitly add this likelihood to the log probability by accessing an internal variable named "target".

As an example, the mixture of a $Normal(1, -2)$ and $Normal(3, 1)$ with mixing proportions $[0.3, 0.7]$ is modeled in Stan as:

```
parameters {
  real y;
}
model {
  target += log_sum_exp(log(0.3) + normal_lpdf(y | -1, 2),
                        log(0.7) + normal_lpdf(y | 3, 1));
}
```

The function "normal_lpdf" returns the log likelihood of the data y given a mean and standard deviation. Here, "target" is the internal Stan variable tracking the log probability. Previous modeling statements, such as ...

```
model {
  y ~ normal(0, 1);
}
```

... implicitly add the log-likelihood to "target" under the hood. In this mixture model example, we are explicitly adding to "target" instead. Direct access to "target" from Stan gives users a lot of flexibility in expressing any model one can think of.

Exercise 1

Implement the following model. Inference may take a while, so we suggest a low number of iterations such as 100 warmup and 500 or 1000 iterations. Compare your results with 1 chain vs. 4 chains.

$$\begin{aligned} y_i &\sim N(\mu_{z_i}, \sigma = 1) \\ z_i &\sim bernoulli(\lambda_i) \end{aligned} \tag{3}$$

1.2 Identifiability for Mixture Models

Inference on model 3 should be stable for a single chain, but you should notice that the final mean value lies between the true means μ_0 and μ_1!

We chose to use $N = 20$, and means -5 and 5. We centered the first 10 datapoints around -5 and the last 10 datapoints around 5 for ease of posterior checking. Running with 4 cores, we observe the following fit:

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
mu_a	-2.54	3.09	4.37	-5.17	-5.09	-5.04	0.49	5.1	2.0	81.54
mu_b	2.51	3.09	4.37	-5.14	-0.52	5.01	5.05	5.13	2.0	83.5
lambda[0]	0.74	0.29	0.41	4.3e-3	0.44	0.97	0.99	1.0	2.0	14.85

lambda[1]	0.73	0.29	0.41	3.6e-3	0.47	0.97	0.99	1.0	2.0	14.07
lambda[2]	0.73	0.29	0.41	3.6e-3	0.46	0.97	0.99	1.0	2.0	15.17
lambda[3]	0.73	0.29	0.41	2.8e-3	0.5	0.97	0.99	1.0	2.0	14.73
lambda[4]	0.73	0.29	0.41	4.0e-3	0.43	0.97	0.99	1.0	2.0	15.29
lambda[5]	0.73	0.29	0.41	2.7e-3	0.46	0.97	0.99	1.0	2.0	14.15
lambda[6]	0.73	0.29	0.41	4.9e-3	0.46	0.96	0.99	1.0	2.0	15.71
lambda[7]	0.73	0.29	0.41	3.2e-3	0.46	0.97	0.99	1.0	2.0	13.98
lambda[8]	0.73	0.29	0.41	3.4e-3	0.47	0.96	0.99	1.0	2.0	14.74
lambda[9]	0.73	0.29	0.41	3.0e-3	0.49	0.97	0.99	1.0	2.0	15.3
lambda[10]	0.27	0.29	0.41	1.2e-3	0.01	0.04	0.56	1.0	2.0	14.49
lambda[11]	0.27	0.29	0.41	1.2e-3	0.01	0.03	0.54	1.0	2.0	14.17
lambda[12]	0.27	0.29	0.41	1.3e-3	0.01	0.04	0.56	1.0	2.0	14.39
lambda[13]	0.27	0.29	0.41	8.9e-4	0.01	0.03	0.56	1.0	2.0	14.63
lambda[14]	0.27	0.29	0.41	1.2e-3	0.01	0.03	0.52	1.0	2.0	14.61
lambda[15]	0.27	0.29	0.41	1.1e-3	0.01	0.04	0.56	1.0	2.0	14.9
lambda[16]	0.27	0.29	0.41	1.6e-3	0.01	0.03	0.52	1.0	2.0	14.32
lambda[17]	0.27	0.29	0.41	1.3e-3	0.01	0.04	0.56	1.0	2.0	14.32
lambda[18]	0.27	0.29	0.41	1.1e-3	0.01	0.04	0.55	1.0	2.0	14.28
lambda[19]	0.27	0.29	0.41	9.9e-4	0.01	0.04	0.57	1.0	2.0	13.8
lp__	-952.3	0.1	3.6	-960.5	-954.5	-952.0	-949.8	-946.4	1185.0	1.0

We note that the n_eff and Rhat values are very poor. The means and values of λ indicate that 3 chains found $\mu_a = -5$ and $\mu_b = 5$, and 1 chain found $\mu_a = 5$ and $\mu_b = 5$.

Our plotted fit looks like this:

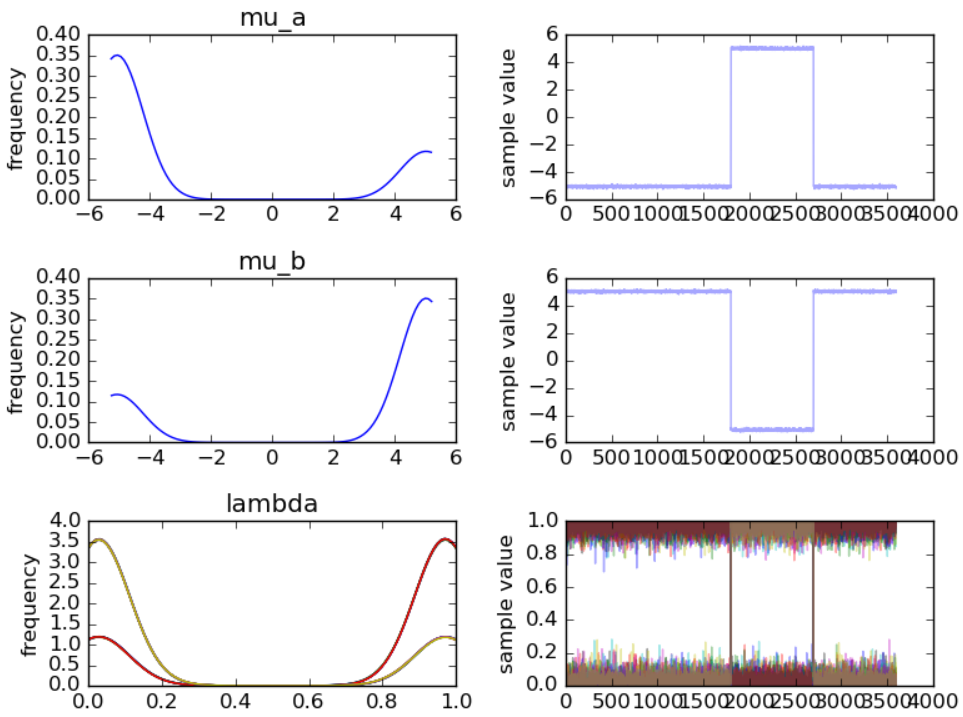

Figure 1: Problems with Identifiability

In general, identifiability issues can be solved by specifying additional prior knowledge. Here, all that is needed to get the chains to agree on a single solution is to place an ordering on μ_0 and μ_1, such as $\mu_0 < \mu_1$.

In Stan, we can do so with:

```
parameters {
  ordered[2] mu;
}
```

Values within the "ordered" data-type are ordered in increasing value.

Exercise 2

Solve the identifiability issue by placing an ordering on μ_0 and μ_1.

Exercise 3

Extend the model to $K = 10$ clusters. Use a for loop to add to "target".

Exercise 4

Simulate data from a uniform mixture of normals centered at -2, 0, and 2 all with standard deviation 1. Run inference on a model with 2 mixtures, 3 mixtures, and 4 mixtures. What do you observe?

2 Example: Hurdle Models

Material from this section is lifted directly from the Stan reference (pg 119), with a few added comments.

Consider a "hurdle model" with probability density function:

$$p(y|\theta,\lambda) = \begin{cases} \theta, & \text{if } y = 0, \text{ and} \\ (1-\theta)\frac{poisson(y|\lambda)}{1-poissonCDF(0|\lambda)}, & \text{if } y > 0 \end{cases} \tag{4}$$

The denominator $1 - poissonCDF(0|\lambda)$ is an important adjustment because the total probability density under the constraint that $y > 0$ is no longer 1. This model in Stan looks like:

```
if (y[n] == 0) {
  1 ~ bernoulli(theta);
}
else {
  0 ~ bernoulli(theta);
  y[n] ~ poisson(lambda) T[1, ];
}
```

The Stan manual notes that: "The Bernoulli statements are just shorthand for adding $\log\theta$ and $\log(1-\theta)$ to the log density. The T[1,] after the Poisson indicates that it is truncated below at 1; see Section 9.1 in the Stan reference for more about truncation."

The above model is equivalent to the explicitly written:

```
if (y[n] == 0)
  target += log(theta);
else
  target += log1m(theta) + poisson_lpmf(y[n] | lambda)
            - poisson_lccdf(0 | lambda));
```

The function "log1m(real x)" returns $\log(1-x)$ if $x \leq 1$.

The poisson "lpmf" is a function that returns the log probability of observation $y[n]$ under a poisson distribution with parameter λ. The primary difference with a statement like "$y[n] \sim poisson(lambda)$;" is that the lpmf function does not add to the log probability.

The poisson "lccdf" is the log of the "complementary cumulative density function" (CCDF) at 0 for the Poisson distribution with parameter λ, which is $(1 - poisson_C DF(-\infty, 0))$.

The Stan manual then notes that "This is an example where collecting counts ahead of time can also greatly speed up the execution speed without changing the density. For data size N = 200 and parameters $\theta = 0.3$ and $\lambda = 8$, the speedup is a factor of 10; it will be lower for smaller N and greater for larger N; it will also be greater for larger θ. To achieve this speedup, it helps to have a function to count the number of nonzero entries in an array of integers."

Here, a new block, the function block, is used:

```
functions {
  int num_zero(int[] y) {
    int nz;
    nz = 0;
    for (n in 1:size(y))
      if (y[n] == 0)
        nz = nz + 1;
    return nz;
  }
}
```

"Then a transformed data block can be used to store the sufficient statistics,":

```
transformed data {
  int<lower=0, upper=N> N0;
  int<lower=0, upper=N> Ngt0;
  int<lower=1> y_nz[N - num_zero(y)];

  N0 = num_zero(y);
  Ngt0 = N - N0;
  {
    int pos;
    pos = 1;
    for (n in 1:N) {
      if (y[n] != 0) {
        y_nz[pos] = y[n];
        pos = pos + 1;
      }
    }
  }
}
```

"The model block can then be reduced to three statements"

```
model {
  N0 ~ binomial(N, theta);
  y_nz ~ poisson(lambda);
  target += -Ngt0 * log1m_exp(-lambda);
}
```

"The first statement accounts for the Bernoulli contribution to both the zero and non zero counts. The second line is the Poisson contribution from the non-zero counts, which is now vectorized.

Finally, the normalization for the truncation is a single line, so that the expression for the log CCDF at 0 isnt repeated. Also note that the negation is applied to the constant Ngt0; whenever possible, leave subexpressions constant because then gradients need not be propagated until a non-constant term is encountered."

3 Bayesian Variable Selection

Variable selection, informally, is the problem of learning which variables/features are relevant for our prediction task. Ideally, we learn which features are useful to include for prediction and which can be excluded - a binary, 0/1 decision.

One example of a variable selection problem is attempting to learn a sparse weight vector in a regression setting. This can be useful for interpretability and generalization to related tasks.

One approach that may come to mind is placing a sparsity-inducing prior over the weights vector. However, as we have seen, sparsity-inducing regularization from the domain of statistical learning theory (such as L1 regularization) do not have exact Bayesian counterparts because true sparsity (setting values to exactly 0) requires summarizing the full posterior distribution using a point estimate.

With the tool of mixture models in our box, we can approach "true sparsity" or variable selection in a fully Bayesian manner.

Consider the following mixture model for a single weight w_i in our weight vector:

$$\begin{aligned} \lambda_i &\sim bernoulli(q_i) \\ p(w_i) &\sim \begin{cases} normal(0,0), & \text{if } \lambda_i = 0, \text{ and} \\ uniform(-\infty, +\infty), & \text{if } \lambda_i = 1 \end{cases} \end{aligned} \tag{5}$$

Here, we use an improper normal distribution with $\sigma = 0$ to indicate a "spike" distribution with zero probability mass everywhere except at 0. The uniform distribution is a "slab" distribution, and together they form a theoretically perfect "spike and slab" model. We note that, in practice, these ideal forms of the spike and slab distributions should be relaxed for an effective Stan implementation.

The output of our Bayesian model is a posterior distribution over parameters, but in practice often a single discrete model (or final set of non-zero weights) is necessary. In such a case, a sensible post-processing step is to set a cutoff threshold such as 0.5 where you set each weight w_i to 0 if its associated $q_i < 0.5$.

Bayesian variable selection is particularly useful when building non-linear models with splines and basis functions, which gain modeling expressivity at the cost of learning over a large number of features. Under such circumstances, Bayesian variable selection is a natural tool since it is expected *a priori* that not all features will be useful in the model.

3.1 Two Approaches

In practice, the two approaches of mixture models and sparsity-inducing priors have different pros and cons, so we recommend that you keep both in mind. Stan's inference engine performs worse on discrete variables (which appear in finite mixture models) than continuous variables. However, sparsity-inducing priors do not set variables to exactly 0, which can be undesirable in some applications, but some Bayesians (such as Andrew Gelman) find this philosophically appealing.

3.2 Sparsity-Inducing Priors, Revisited

Previously, the double-exponential prior was discussed, which in combination with taking the posterior mode, is equivalent to L1-regularization.

A spike and slab mixture model can be approximated by a single distribution if we ensure it has high probability density at 0 (akin to the spike distribution) and also has arbitrarily long tails (resembling a uniform distribution) which avoids over-shrinking the parameter.

With these properties in mind, the **generalized double Pareto distribution** is a good choice. It became popular in 2011, where it was described by: "While it has a spike at zero like the Laplace density, it also has a Students t-like tail behavior." See https://arxiv.org/pdf/1104.0861.pdf

The generalized double Pareto distribution has a scale parameter $\xi > 0$ and a shape parameter $\alpha > 0$. Its probability density function is:

$$gdP(y|\xi, \alpha) = \frac{1}{2\xi}\left(1 + \frac{|y|}{\alpha\xi}\right)^{-(\alpha+1)} \tag{6}$$

The generalized double Pareto distribution has an interpretation as a scale mixture of normals, because it can be sampled using model 7, where the population is distributed by a mixture of normals with varying scale parameters σ_i, and the parameters ξ, α define the mixture.

$$\begin{aligned} y_i &\sim N(0, \sigma_i^2) \\ \sigma_i &\sim Expon(\lambda_i^2/2) \\ \lambda_i &\sim Gamma(\alpha, \eta = \xi\alpha) \end{aligned} \tag{7}$$

Figure 2 displays the shape of the generalized double Pareto distribution, revealing its properties.

The left side compares the generalized double Pareto distribution with the Laplace distribution and the Cauchy distribution. It is shown that the generalized double Pareto distribution has a similar spiky peak at 0 with the Laplace distribution, but has more probability mass at its tails than the Cauchy distribution.

The right side of the figure displays the distribution under different settings of the scale parameter $\xi > 0$ and the shape parameter $\alpha > 0$.

In practice, using $\xi = 1, \alpha = 1$ is a good default starting point.

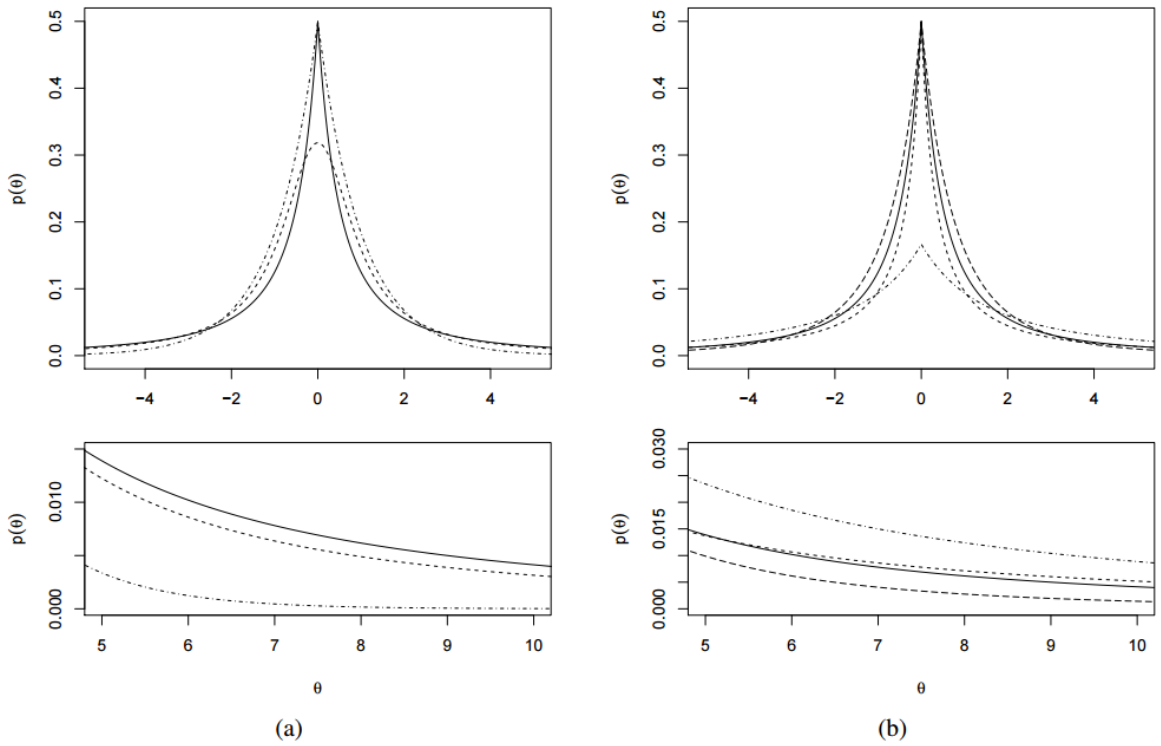

Figure 2.1: (a) Probability density functions for standard double Pareto (solid line), standard Cauchy (dashed line) and Laplace (dot-dash line) ($\lambda = 1$) distributions. (b) Probability density functions for the generalized double Pareto with (ξ, α) values of $(1, 1)$ (solid line), $(0.5, 1)$ (dashed line), $(1, 3)$ (long-dashed line), and $(3, 1)$ (dot-dash line).

Figure 2: The Generalized Double Pareto Distribution

Exercise 5

Implement model 7 (Write simulation code and Stan code).

4 Bibliography and Additional Resources

Chapter 10 of the Stan reference discusses implementation of finite mixture models in Stan.

In Gelman's Bayesian Data Analysis 3, chapter 20 discusses basis function models in greater depth, and chapter 22 addresses finite mixture models.

The Stan reference is available here: http://mc-stan.org/documentation/