CS281B/Stat241B: Advanced Topics in Learning & Decision Making

Reversible Jump MCMC and Introduction to Kernel Methods

Lecturer: Michael I. Jordan *Scribes: Andrea Frome*

1 Reversible Jump MCMC

Returning to the mixture model at the end of last lecture (Figure 1), we could try to find $P(k|y)$ by running the Gibbs sampler. This would require moralizing the graph (Figure 2), and at each sampling step, set the value of a node given the values of the nodes in its Markov blanket. We already have one problem in that we have continuous and discrete nodes, and we don't have a way to integrate over the continuous nodes.

Another problem is that the number of components, k, can change, and the number of parameters changes with k. For fixed k, we could calculate $P(k|y)$ and compare models for different values of k, but we cannot move between models using Gibbs sampler. In contrast to this approach, reversible jump MCMC seeks to calculate the probability of *all* nodes in the graph by transitioning between models of different k values as it samples.

Review: Metropolis-Hastings (Chapter 1 of Jordan & Bishop [1]): Let x be a state including all variables except y, and let $\pi(x)$ be a distribution on x. We want to sample to determine $\pi(x)$. To do this, we set up a Markov chain with a transition function $p(x, x')$ that has $\pi(x)$ as an equilibrium distribution, i.e. if the sampling algorithm runs long enough, then we would be sampling from $\pi(x)$. A sufficient condition for ensuring $\pi(x)$ is an equilibrium distribution is *detailed balance*. (While it isn't a necessary condition, it is usually the easiest way to show that the sampling algorithm would find the equilibrium distribution if run long enough). Detailed balance is the reversibility condition:

$$\forall x, x' : \pi(x)p(x, x') = \pi(x')p(x', x) \tag{1}$$

Metropolis-Hastings satisfies detailed balance, and thus reaches equilibrium. When sampling at node x, the algorithm proposes a move to x' with some probability $q(x, x')$, and accepts the move with probability

$$\alpha = min\left\{1, \frac{\pi(x')}{\pi(x)}\frac{q(x', x)}{q(x, x')}\right\} \tag{2}$$

(The Metropolis algorithm would be just $min\left\{1, \frac{\pi(x')}{\pi(x)}\right\}$.)

So the question is how to jump between subspaces with different dimensionality. An example to frame the problem: we have two models, one in one-dimensional space ($k = 0$) and the other in two-dimensional space ($k = 1$) (Figure 3). The probability mass is uniform in both; in model 0, we use $U(0, 1)$, a uniform distribution on the interval 0 and 1, and in model 1, we use a uniform distribution over a triangle. The probability of being in model 0 is p_1, and the probability of being in model 1 is $1 - p_1$. What we need is a *reversible* way to jump between the two models that will satisfy detailed balance.

For the given example, we consider three possible moves in a given step:

(a) for $k = 0$, $x \to U(x - \epsilon, x + \epsilon)$

(b) for $k = 1$, swap x_1 and x_2

(c) jump between $k = 0$ and $k = 1$

On a given step, we could either move within a model or jump to the other model, so we say that our model will:

$$\begin{array}{lll} \text{for } k = 0, & \text{choose (a)} & \text{with probability } 1 - r_1 \\ & \text{choose (c)} & \text{with probability } r_1 \\ \text{for } k = 1, & \text{choose (b)} & \text{with probability } 1 - r_2 \\ & \text{choose (c)} & \text{with probability } r_2 \end{array}$$

To execute step (c) from $k = 0$, we choose u from $U(0, 1)$ and propose a move such that $(x_1, x_2) = (x, u)$. If we are in $k = 1$, then set $x = x_1$. This is a *bijection* between $(x, u) \leftrightarrow (x_1, x_2)$. Figure 3 shows a possible path through the models following these rules.

Now we need to define a way to accept the move that will ensure reversibility. We choose to accept with probability

$$\alpha = min\left\{1, \frac{p(y|x')}{p(x|y)} \frac{p(x')}{p(x)} \frac{g(k, u'|x')}{g(k, u|x)} \left| \frac{\partial(x', u')}{\partial(x, u)} \right| \right\}. \tag{3}$$

The first two terms on the right side of the *min* come from the acceptance expression for Metropolis-Hastings. The third term is the joint distribution of k and u while at x in the kth model. The fourth term is the determinant of the Jacobian matrix which allows us to change dimensionality. This choice statisfies detailed balance, i.e. there is an equal probability of going in either direction.

In our example:

$$\begin{aligned} \alpha &= min\left\{1, \frac{p_2 \cdot I(x_2 < x_1)}{p_1} \cdot \frac{r_2}{r_2 \cdot 1} |1| \right\} & (4) \\ &= min\left\{1, \frac{2 p_2 r_2}{p_1 r_1}\right\} I(u < x) & (5) \end{aligned}$$

where I is the identify function ($I(s)$ is equal to 1 if statement s is true, and 0 otherwise), and the determinant of the Jacobian is 1 because we have a uniform density on our uppermost model.

To apply this to the mixture model in Figure 2, we use Gibbs for sampling all nodes except node k, and use reversible jump MCMC for node k. We choose Gibbs sampling over Metropolis-Hastings for the other nodes because the conjugate priors that we chose for our model makes sampling with Gibbs easy since we don't have to decide whether to accept. In general, the sampling methods, from simplest to most complex are:

Direct calculation → Gibbs sampling → Metropolis-Hastings → Reversible Jump MCMC

2 Introduction to Mercer Kernel Methods

The basic idea behind *kernel methods* is the "kernel trick" for transforming problems into higher-dimensional, but easier, problems. If we move the problem into a higher space, we are able to use linear methods (e.g. PCA) since a complicated manifold in the original space may be well described by a hyperplane in some higher-dimensional space (Figure 4). What's more, we can do our calculations in our original space; from functional analysis we have the result that inner products in higher dimensions often have a representation as a kernel function in lower dimensions.

For example, take Gaussian kernels:

$$K(x, x') = e^{-\frac{1}{2}||x-x'||^2} \tag{6}$$

For a given x, we evaluate the kernel function and we see that the contours are round (Figure 5). For some $\Phi(x)$, $K(x, x') = \Phi(x)^T \Phi(x')$, where $\Phi(x)$ is a *infinite-dimensional* vector.

Similarly with polynomial kernels:

$$\begin{aligned} K(x, x') &= (1 + x^T x')^n & (7) \\ &= \Phi(x)^T \Phi(x') & (8) \end{aligned}$$

for some Φ with many more dimensions than the original space.

2.1 Introduction to Support Vector Machines

The first kernel method we'll look at is *support vector machines (SVMs)*, a method where we take points close to the classification boundary, and evaluate kernels at these points. SVMs can be used for either classification or regression, but we'll be considering their use in classification. SVMs come out of two basic ideas: (1) the kernel trick and (2) optimal-margin classifiers.

If we want a linear classifier, there are an infinite number of possible classifiers (Figure 6). The optimal classifier is the one that maximizes the margin between the boundary and the data points. What's more, when calculating the optimal-margin classifier, we only need to pay attention to the points closest to the boundary.

References

[1] Jordan, Michael I. and Bishop, Christopher M. *Probabilistic Graphical Models*. To appear.

[2] Richardson, S. and Green, P. J. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society*, series B, 59, 731–792. 1997.

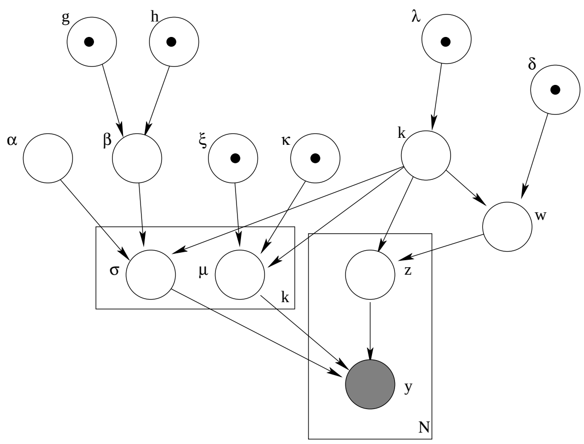

Figure 1: Graph of the mixture model. Nodes with dots in the center are set to fixed values.

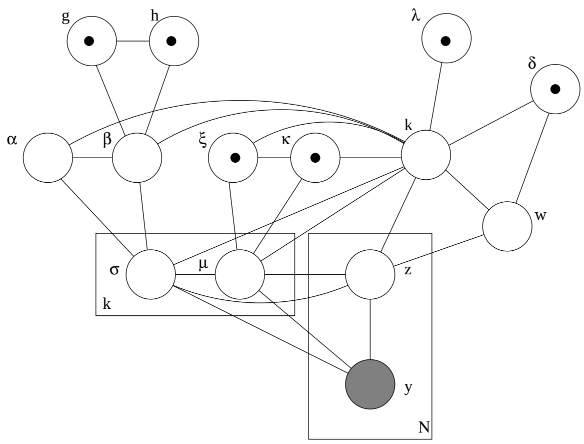

Figure 2: Moralized mixture model graph.

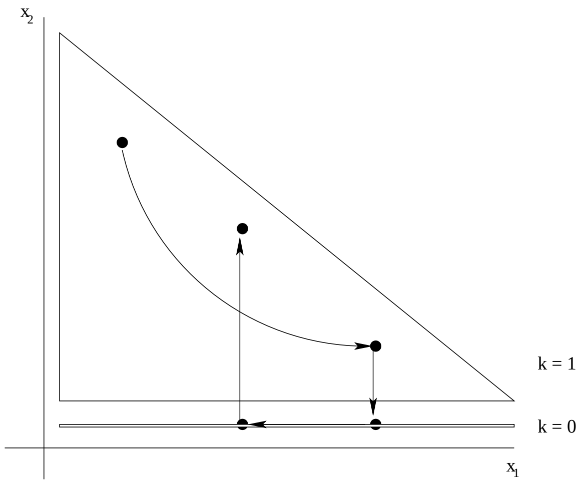

Figure 3: Graph for reversible jump MCMC example. The arrows show a possible sampling path through the models.

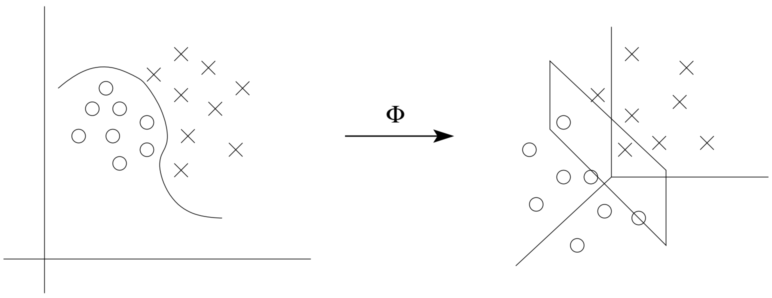

Figure 4: Example of mapping from lower-dimensional space to a higher-dimensional space.

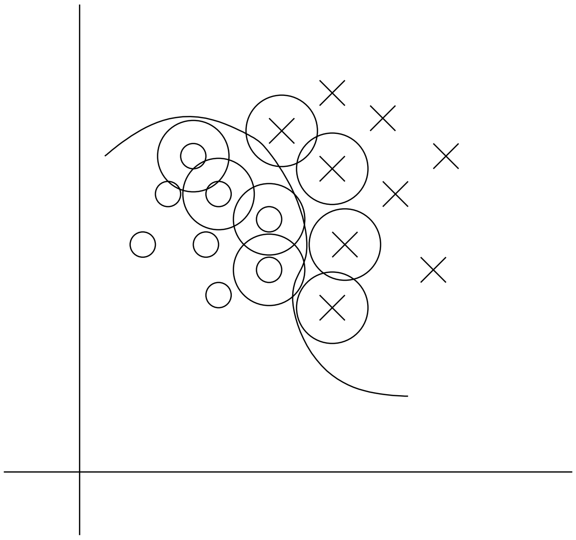

Figure 5: Contours showing Gaussian distribution around kernels.

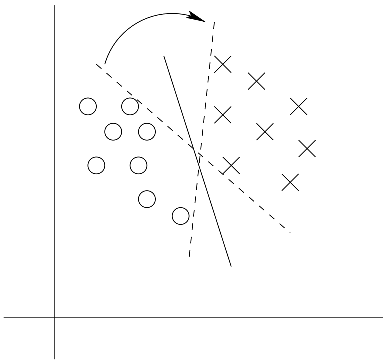

Figure 6: A graph with many possible classifiers. The two dotted lines show two extreme possibilities, and the solid line represents the line that is furthest from all the data points.