E6892 Bayesian Models for Machine Learning

Columbia University, Fall 2015

Lecture 3, 9/24/2015

Instructor: John Paisley

- Up to this point we've been discussing problems that are simple enough that we can calculate all posterior distributions in closed form. That is, we've assumed that given a model $X \sim p(X|\theta)$ and prior $\theta \sim p(\theta)$, we can analytically calculate the posterior using Bayes rule,

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{\int p(X|\theta)p(\theta)d\theta}. \tag{1}$$

  Since we have defined the model and priors, the joint likelihood term in the numerator is always known. Therefore, we specifically have assumed that the normalizing constant

$$p(X) = \int p(X|\theta)p(\theta)d\theta \tag{2}$$

  is an integral we can analytically solve.

- In practice, this is the exception and not the rule for machine learning applications. Again, since we will always be able to write an analytic form for $p(X|\theta)p(\theta)$ because we define it, this is equivalent to saying that the integral in the denominator is usually intractable.

- Why should we care? We know that $p(\theta|X) = \frac{1}{Z}p(X,\theta)$ for some number Z. This just scales the joint likelihood function $p(X,\theta)$ up or down. Imagine we want to calcualte the expectation of some function over its posterior, $\mathbb{E}_{p(\theta|X)}[f(\theta)] = \int f(\theta)p(\theta|X)d\theta$. This is equal to $\frac{1}{Z}\int f(\theta)p(X,\theta)d\theta$. What if this is an integral we can calculate? The problem here is that Z can be expected to vary over such a large range that, without knowing Z we are no closer to knowing anything about $\mathbb{E}_{p(\theta|X)}[f(\theta)]$.

- In this lecture we will discuss two methods for approximating posterior distributions. The first method is called the Laplace approximation and is useful when the number of model variables is relatively small. The second method is called Gibbs sampling, which is an instance of a wide variety of posterior inference algorithms collectively referred to as Markov chain Monte Carlo (MCMC) methods. Gibbs sampling is much more widely used than the Laplace approximation and is still useful in the case of models that have very many variables.

Example: Logistic regression

- Logistic regression is one instance of linear regression applied to the classification problem. To recall the setup, we have data in the form of labeled pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where $x \in \mathbb{R}^d$ and $y \in \{0, 1\}$. The goal is to use this information to help us predict the label of a new y given a new x.

- Model: As with linear regression, we assume that the output depends on the linear relationship between x and a weight vector $w \in \mathbb{R}^d$. This time the relationship is

$$y_i \sim \text{Bernoulli}(\sigma(x_i^T w)), \qquad \sigma(x_i^T w) = \frac{\mathrm{e}^{x_i^T w}}{1 + \mathrm{e}^{x_i^T w}} \tag{3}$$

  Last lecture we saw another modeling approach to binary classification using a Bayes classifier. There is no "correct" choice of model, only more and less appropriate models for the problem at hand; all models make a simplifying approximation about the underlying function generating of the data—the first one being that such a function exists. To find out which model is "better," you can try both on your data and see which performs better in practice.

- Prior: We assume a prior distribution on w of the form

$$w \sim \text{Normal}(0, cI). \tag{4}$$

  Again, this is a choice that we make and a different prior can be defined, perhaps one that makes stronger assumptions about what types of vectors we should learn in the posterior. For example, a prior that is only defined on $\mathbb{R}_+^d$ would *require* that the posterior be defined on $\mathbb{R}_+^d$ as well (simply look at Bayes rule to see this). With the normal prior there is no such requirement.

- Posterior: The posterior distribution on w is found using Bayes rule

$$p(w|X, \vec{y}) = \frac{\prod_{i=1}^N p(y_i|x_i, w)p(w)}{\int \prod_{i=1}^N p(y_i|x_i, w)p(w)dw} \tag{5}$$

  When we plug in $p(y_i|x_i, w) = \left(\frac{\mathrm{e}^{x_i^T w}}{1+\mathrm{e}^{x_i^T w}}\right)^{y_i} \left(1 - \frac{\mathrm{e}^{x_i^T w}}{1+\mathrm{e}^{x_i^T w}}\right)^{1-y_i}$, we quickly find that the integral in the denominator can't be solved. Therefore, we can't get the posterior distribution. Furthermore, imagine that there existed an oracle that could magically gave us the value of the solution to this integral. We would still likely hit a roadblock later when trying to solve integrals, for example to calculate the predictive distribution. This is why many researchers like to keep things conjugate if they can. When they can't they often resort to approximations. We will next discuss one simple method to approximate this posterior distribution.

Laplace approximations

- Return to the generic model: $X \sim p(X|\theta)$ and $\theta \sim p(\theta)$. We make no assumptions about the forms of either of these distributions. Laplace approximations treat the posterior as being approximately Gaussian. That is, the goal of the Laplace approximation is to set

$$p(\theta|X) \approx \text{Normal}(\mu, \Sigma) \tag{6}$$

  for some good setting of μ and Σ. There are different ways one might find these parameter values; the Laplace approximation is one approach.

- First, it turns out that a normal approximation to the posterior can arise automatically from instead trying to approximate the joint likelihood $p(X, \theta) = p(X|\theta)p(\theta)$. That is, we will approximate the numerator and denominator in Bayes rule, after which the normal distribution will pop out.

- The first step is to write Bayes rule slightly differently

$$p(\theta|X) = \frac{\mathrm{e}^{\ln p(X,\theta)}}{\int \mathrm{e}^{\ln p(X,\theta)} d\theta} \tag{7}$$

- Next, rather than focus on $p(X, \theta)$, we focus on $\ln p(X, \theta)$. Notice that in more abstract terms we can just think of this as a function of θ. The interpretation of this function as the log of the joint likelihood is simply added later. We want an approximation of $\ln p(X, \theta)$ that is a good one, but also general enough that we don't need to worry about the fact that we don't actually know what this function is yet since we haven't defined anything other than abstract probability distributions.

- The Laplace approximation uses a 2nd order Taylor expansion as an approximation. Recall that we can approximate a function $f(\theta)$ with its 2nd order Taylor expansion as follows:

$$f(\theta) \approx f(\theta_0) + (\theta - \theta_0)^T \nabla f(\theta_0) + \frac{1}{2}(\theta - \theta_0)^T \nabla^2 f(\theta_0)(\theta - \theta_0) \tag{8}$$

  The point θ_0 is any arbitrarily chosen point in the domain of f (i.e., chosen among values that θ is allowed to take). The shorthand $\nabla f(\theta_0)$ means we evaluate the gradient of $f(\theta_0)$ at θ_0. For our problem $f(\theta) = \ln p(X, \theta)$.

- The point θ_0 is critical since the 2nd order Taylor approximation is very accurate around this point, and much less accurate as we move away from it. Therefore, we must decide a good setting for θ_0.

- For the Laplace approximation we are able to find a good point, and also a convenient one. Specifically, we set

$$\theta_0 = \theta_{\text{MAP}} = \arg\max_{\theta} \; p(\theta|X) = \arg\max_{\theta} \; \frac{p(X|\theta)p(\theta)}{p(X)} \tag{9}$$

  This is called the *maximum a posteriori* solution. Notice that it is the point that is the most probable (or, really, has the highest density) according to the posterior distribution. However, the whole reason we're on this endeavor is that we don't know the posteriori. Fortunately the maximum point of $p(\theta|X)$ is the same as for $p(X, \theta)$; the constant $p(X)$ doesn't depend on θ so it just scales the function and doesn't change the point at which it's maximized.

- Aside about MAP inference: MAP inference is the process of finding, for example, θ_{MAP} in the problem above. It first converts the joint likelihood to the log joint likelihood, $\ln p(X, \theta)$, making the observation that the monotonicity of the log transformation preserves all local maximum of $p(X, \theta)$. It then initializes θ and iteratively modifies θ to maximize $\ln p(X, \theta)$. One pervasive approach to this are the gradient ascent methods. For example, with steepest ascent, we iteratively update $\theta \leftarrow \theta + \rho \nabla_\theta \ln p(X, \theta)$ until it reaches a local maximum of $\ln p(X, \theta)$. $\rho \in [0, 1]$ scales the gradient and is usually small.

- And so for the Laplace approximation, the first step is to run an iterative algorithm to find θ_{MAP}. Then, using this as θ_0 in the 2nd order Taylor expansion we notice a nice simplification occurs,

$$\ln p(X,\theta) \approx \ln p(X,\theta_{\text{MAP}}) + \underbrace{(\theta-\theta_{\text{MAP}})^T \nabla \ln p(X,\theta_{\text{MAP}})}_{=0} + \frac{1}{2}(\theta-\theta_{\text{MAP}})^T \nabla^2 \ln p(X,\theta_{\text{MAP}})(\theta-\theta_{\text{MAP}}) \tag{10}$$

The first order term cancels out because the gradient of $\ln p(X,\theta)$ equals zero at the location of its maximum point, which is precisely what we get when we run the MAP algorithm to find θ_{MAP}.

- Directly plugging in this approximation,

$$p(X,\theta) \approx \frac{p(X,\theta_{\text{MAP}})\text{e}^{-\frac{1}{2}(\theta-\theta_{\text{MAP}})^T(-\nabla^2 \ln p(X,\theta_{\text{MAP}}))(\theta-\theta_{\text{MAP}})}}{\int p(X,\theta_{\text{MAP}})\text{e}^{-\frac{1}{2}(\theta-\theta_{\text{MAP}})^T(-\nabla^2 \ln p(X,\theta_{\text{MAP}}))(\theta-\theta_{\text{MAP}})} d\theta} \tag{11}$$

The terms $p(X,\theta_{\text{MAP}})$ cancel out and we are left with something we can immediately recognize as a multivariate normal distribution. In particular, we see that

$$p(X,\theta) \approx \text{Normal}(\theta_{\text{MAP}}, (-\nabla^2 \ln p(X,\theta_{\text{MAP}}))^{-1}) \tag{12}$$

We set the mean of our posterior approximation to the MAP solution θ_{MAP} and the covariance to the inverse of the negative of the Hessian of $\ln p(X,\theta)$ evaluated at θ_{MAP}. Therefore, after running the MAP algorithm, we calculate the matrix of second derivatives by hand and plug $\theta = \theta_{\text{MAP}}$ into the resulting matrix-organized set of functions of θ. We use this matrix to approximate the covariance. Notice that, from a computational perspective, the measure of uncertainty that we get on top of MAP from an approximate posterior distribution comes almost for free when $\dim(\theta)$ is small.

Logistic regression revisited

- We next derive the Laplace approximation for the logistic regression problem. We first write the joint likelihood,

$$p(\vec{y}, w|X) \propto \prod_{i=1}^{N} \sigma(x_i^T w)^{y_i}(1-\sigma(x_i^T w))^{1-y_i}\text{e}^{-\frac{1}{2c}w^T w}, \qquad \sigma(x_i^T w) = \frac{\text{e}^{x_i^T w}}{1+\text{e}^{x_i^T w}} \tag{13}$$

We initialize $w^{(0)} = 0$ and update $w^{(t)} = w^{(t-1)} + \rho \nabla_w \ln p(\vec{y}, w|X)|_{w^{(t-1)}}$ using gradient ascent. The derivative is directly calculated to be

$$\nabla_w \ln p(\vec{y}, w|X) \;=\; \sum_{i=1}^{N}(y_i - \sigma(x_i^T w))x_i \tag{14}$$

We simply plug the most recent value for w into this function to get the direction in which to step. When the algorithm converges to a point "close enough" to the maximum value, found by monitoring how much $\ln p(\vec{y}, w|X)$ is increasing with each step, we take the final value of $w^{(T)}$ at this iteration (call it T) and set w_{MAP} to be this vector and therefore the mean of the approximate posterior distribution of w.

To find the covariance, we directly calculate that

$$\nabla_w^2 \ln p(\vec{y}, w|X) \;=\; -\sum_{i=1}^{N}\sigma(x_i^T w)(1-\sigma(x_i^T w))x_i x_i^T \tag{15}$$

We plug the value of w_{MAP} into this function and invert the negative of it to obtain the the covariance matrix of the normal distribution used to approximate $p(w|\vec{y}, X)$.

Another modeling example

- Let's look at another example of a model where we can't calculate the posterior analytically, but for which the Laplace approximation isn't a feasible option.

- Setup: We are given a set of measurements $\mathcal{D} = \{r_{ij}\}$ where the index pair $(i, j) \in \Omega$. That is, we let $i = 1, \ldots, N$ and $j = 1, \ldots, M$, but not every (i, j) is measured (i.e., in Ω). For example, r_{ij} could be the rating given by user i to object j (think Netflix, Yelp, Amazon, etc.), in which case $|\Omega| \ll NM$. We want to come up with a model to predict those $r_{ij} \notin \mathcal{D}$.

- Model: A typical model for this uses matrix factorization (we will not directly use the matrix notation here) and a low rank assumption. Here, we assume each user i is represented by a vector $u_i \in \mathbb{R}^d$ and each object j by a vector $v_j \in \mathbb{R}^d$. The low rank assumption comes from $d \ll \min(N, M)$. Then we let

$$r_{ij} \sim \text{Normal}(u_i^T v_j, \sigma^2) \qquad \text{for all } (i, j) \in \Omega \tag{16}$$

  Comment: This model does not seem ideal for all cases since it assumes $r_{ij} \in \mathbb{R}$. For example, in the context of user/object ratings alarm bells should be going off because we are modeling a finite set of positive integers as continuous. In this case where $r_{ij} \notin \mathbb{R}$ we are making a simplifying modeling assumption. This won't lead to any bugs during inference time, and the justification for it is in the performance. Still, we will later see how a few simple modifications can make us more comfortable with the modeling assumptions while still being very close to this model.

- Priors: We will assume that $u_i \overset{iid}{\sim} \text{Normal}(0, cI)$ and $v_j \overset{iid}{\sim} \text{Normal}(0, cI)$.

- Posterior: Let U contain all u_i, V contain all v_i and R contain all measured r_{ij}. Using Bayes rule, we want to calculate

$$p(U, V|R) = \frac{p(R|U, V)p(U, V)}{\int \cdots \int p(R|U, V)p(U, V)du_1 \cdots du_N dv_1 \cdots dv_M} \tag{17}$$

- The problem, is that the denominator is intractable and so we can't solve Bayes rule. Just like before we hit a roadblock. However, unlike before it seems that Laplace can't help us here. Even though we could find a local optimal solution for U_{MAP} and V_{MAP} to approximate a large, $Nd + Md$ length mean vector, the covariance matrix obtained from the Hessian would be $(Nd + Md) \times (Nd + Md)$. This can easily exceed one billion values to estimate!

- Step back: Let's take a step back and talk about what our objectives typically are with Bayes rule in contexts like these. Let $p(U, V|R)$ be the posterior distribution on the model variables. What we really want is the predictive distribution $p(r_{ij}|R)$ for some $(i, j) \notin \Omega$. Or, at least, we would like to calculate the expectation $\mathbb{E}[r_{ij}|R]$ under this posterior where we integrate out all of our uncertainty in the model variables. For now let's just focus on the expectation.

- We develop this further in the context of the above problem to make it concrete, but observe that this is a very general problem in Bayesian modeling. Using the notational shorthand for this

problem, the expectation of a new r_{ij} under the predictive distribution is

$$\begin{aligned}
\mathbb{E}[r_{ij}|R] &= \int r_{ij} p(r_{ij}|R) dr_{ij} && (18)\\
&= \int r_{ij} \int\int p(r_{ij}|U,V) p(U,V|R) dU dR dr_{ij} && (19)\\
&= \int\int \left[\int r_{ij} p(r_{ij}|U,V) dr_{ij} \right] dU dV && (20)
\end{aligned}$$

Notice that we've swapped integrals above. Since $p(r_{ij}|U,V) = \text{Normal}(u_i^T v_j, \sigma^2)$, the bracketed term is equal to $u_i^T v_j$. Therefore, we ultimately want $\mathbb{E}[u_i^T v_j|R]$ where the expectation is under the posterior distribution $p(U,V|R)$. Since we don't know this distribution, we are forced to seek other options. As we will see, one of these options employs Monte Carlo integration.

- Monte Carlo integration: MC integration is a very general technique. Imagine we want to calculate $\mathbb{E}_{p(X)}[f(x)]$ but run into a mathematical problem in doing so. We can approximate this expectation by first sampling $X_1, \ldots, X_N \overset{iid}{\sim} p(X)$ and then approximating

$$\mathbb{E}_{p(X)}[f(X)] \approx \frac{1}{N} \sum_{n=1}^{N} f(X_n) \tag{21}$$

  As $N \to \infty$ this approximation converges to the true expectation.

- So if we could find a way to get i.i.d. samples

$$(U^{(1)}, V^{(1)}), \ldots, (U^{(N)}, V^{(N)}) \overset{iid}{\sim} p(U,V|R) \tag{22}$$

  we could say

$$\mathbb{E}[u_i^T v_j|R] \approx \frac{1}{N} \sum_{n=1}^{N} \langle u_i^{(n)}, v_j^{(n)} \rangle \tag{23}$$

- In fact, having access to these samples would allow us to calculate much more than an approximation to $\mathbb{E}[r_{ij}|R]$. For example, if we wanted a measure of confidence in this prediction, we could calculate the variance under the predictive distribution, $Var(r_{ij}|R) = \mathbb{E}[r_{ij}^2|R] - \mathbb{E}[r_{ij}|R]^2$. Using similar reasoning as above, we would find that

$$Var(r_{ij}|R) \approx \sigma^2 + \frac{1}{N} \sum_{n=1}^{N} \left(\langle u_i^{(n)}, v_j^{(n)} \rangle \right)^2 - \left(\frac{1}{N} \sum_{n=1}^{N} \langle u_i^{(n)}, v_j^{(n)} \rangle \right)^2 \tag{24}$$

  Many other calculations of interest under the predictive distribution can also be obtained this way. Therefore, a lot can be done if we can somehow get access to i.i.d. samples from the posterior distribution on the model $p(U,V|R)$.

- A major procedure for (approximately) getting these samples is called *Markov chain Monte Carlo (MCMC)* sampling. We'll next discuss one instance of this general technique, and arguably the easiest, called *Gibbs sampling*.

Gibbs sampling

- We will present the *recipe* for Gibbs sampling. Unfortunately a discussion on the *correctness* of the recipe is beyond the scope of this class. Every statistics department will have courses that focus heavily on MCMC methods, and to truly appreciate MCMC and it's theoretical correctness, it's worthwhile to take one of those classes. This is in contrast to *variational* inference methods, which are generally not taught in statistics departments and will be discussed more fully later in this course. Variational inference is a development of machine learning research.

- To use simpler and more general notation, imagine we have a model with two variables, θ_1 and θ_2. We have data from this model, $X \sim p(X|\theta_1, \theta_2)$ and independent priors $p(\theta_1)$ and $p(\theta_2)$. We find that we can't calculate $p(\theta_1, \theta_2|X)$ because the normalizing constant is an intractable integral. This is the typical story thus far.

- However, Gibbs sampling makes the added assumption that we *can* calculate $p(\theta_1|X, \theta_2)$ and $p(\theta_2|X, \theta_1)$ using Bayes rule. In other words, if we had access to all of the model variables except for one of them, then we could calculate the posterior distribution of this one unknown variable given the data and all other variables.

- Gibbs sampling uses this property of the tractability of all *conditional* posterior distributions to get samples from the unknown *full* posterior distribution of all model variables. Below, we write the Gibbs sampling procedure for this generic model.

  Gibbs sampling algorithm for the toy problem

  1. Randomly initialize $\theta_1^{(0)}$ and sample $\theta_2^{(0)} \sim p(\theta_2|X, \theta_1^{(0)})$
  2. For step $t = 1, \ldots, T$
     (a) Sample $\theta_1^{(t)} \sim p(\theta_1|X, \theta_2^{(t-1)})$
     (b) Sample $\theta_2^{(t)} \sim p(\theta_2|X, \theta_1^{(t)})$

- In other words, we first initialize all the variables in some way. Step 1 above is one possible initialization method. Then, we iterate through each variable and update it by sampling from its posterior distribution conditioned on the *current values* of everything els. MCMC theory proves that we get (approximate) samples from $p(\theta_1, \theta_2|X)$ this way, and Gibbs sampling MCMC requires us to be able to calculate the conditional posterior distributions.

- Discussion: First, notice that the conditional posterior uses the *most recent* values for all variables. Therefore, when sampling a particular θ_i during iteration t, some variables we condition on will use their value at iteration $t-1$ (because they haven't been updated yet during iteration t), while others will use their new value at iteration t (because they were sampled before sampling θ_i during the current iteration). What Gibbs sampling does *not* do is partition the variables into two sets and sample the new set at step t conditioned on the old set from $t-1$.

- To give some MCMC background and how it is run in practice (in the context of the toy model):

    1. MCMC is a special case of a Markov chain where the stationary distribution is the desired full posterior distribution. Therefore, we are only guaranteed to get *one* sample from the full posterior in the infinite limit. (It is necessary to take a class devoted to these methods to appreciate this fully.)

    2. In practice, one first runs MCMC (here, the Gibbs sampler) for a large number of steps, let that number be T_1, and then approximates the first sample from the posterior distribution as $(\theta_1^{(T_1)}, \theta_2^{(T_1)})$. The first $T_1 - 1$ steps are called the "burn-in" phase and is intended to minimize the impact of the initialization, which likely does not start the chain off with a good setting for the model variables. The first $T_1 - 1$ iterations "fix" this bad initialization.

    3. Next we want to get a second sample from the posterior. However, remember that we need the samples to be i.i.d. for Monte Carlo integration. Clearly any two adjacent samples $(\theta_1^{(t)}, \theta_2^{(t)})$ and $(\theta_1^{(t+1)}, \theta_2^{(t+1)})$ are not independent of each other, since the values at t were used to get the values at $t+1$. Therefore, just like with the burn-in phase, we run the chain for several more iterations until step T_2 so that $(\theta_1^{(T_1)}, \theta_2^{(T_1)})$ and $(\theta_1^{(T_2)}, \theta_2^{(T_2)})$ are almost totally uncorrelated. This continues again until step T_3, then T_4, etc. Usually, $T_i - T_{i-1}$ for $i > 1$ is picked to be a constant number much less than T_1.

    4. Good numbers for T_1 and $T_i - T_{i-1}$ are not immediately obvious. MCMC textbooks give useful heuristics. In machine learning applications these are usually picked arbitrarily in practice and are usually determined more by the desired number of samples within a pre-allotted running time than by the statistical properties of the Markov chain.

    5. We then make the approximation that

    $$(\theta_1^{(T_1)}, \theta_2^{(T_1)}), (\theta_1^{(T_2)}, \theta_2^{(T_2)}), \ldots, (\theta_1^{(T_S)}, \theta_2^{(T_S)}) \overset{iid}{\sim} p(\theta_1, \theta_2 | X) \tag{25}$$

    Notice that we're simply throwing away the updates for many of the iterations. By writing out the integral form of the expectation below, we see we can use these samples for Monte Carlo integration to approximate

    $$\mathbb{E}[f(\hat{X})|X] \approx \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}[f(\hat{X})|\theta_1^{(T_s)}, \theta_2^{(T_s)}] \tag{26}$$

    6. The reason that MCMC, despite its precise theory, is still only an approximate method is the fact that T_1 and $T_i - T_{i-1}$ are finite. Therefore, at step T_1 we are *not* sampling from the stationary distribution of the Markov chain (remember that the "stationary distribution" is the desired posterior distribution), and the samples at steps T_i and T_{i-1} are *not* 100% uncorrelated with each other, and therefore not independent. Nevertheless, MCMC often works extremely well in practice.

    7. Much of the development of MCMC techniques not covered in this class—in addition to accounting for models in which we don't have all conditional posteriors—are on ways to sample such that the values of T_1 and $T_i - T_{i-1}$ can be reduced without sacrificing the quality of the samples (i.e., their approximate independence and closeness to the posterior distribution).

- Returning to the original model we were discussing, we want $p(U, V|R)$ where U is the set of N "user" vectors and V the set of M "object" vectors, and R is the set of measured values $\{r_{ij} : (i, j) \in \Omega\}$ often corresponding to a rating given by user i to object j.

- We can't find this posterior exactly. A next step is to check if we can calculate all the *conditional* posteriors exactly. If so, we can approximately sample from $p(U, V|R)$ using Gibbs sampling.

- We will check for one u_i. Clearly this then establishes the case for all u_i, and by symmetry we can perform the same exact derivation for all v_j. The conditional posterior of u_i is

$$p(u_i|V, R) \propto \prod_{j:(i,j)\in\Omega} p(r_{ij}|u_i, v_j)p(u_i) \tag{27}$$

  1. Notice with this that we decided to write the likelihood in terms of r_{ij} only and not v_j.

  2. Also, notice that on the LHS we condition on all V and R.

  3. On the RHS we use the structure of the model we defined: We only need to consider the r_{ij} for a fixed i and over those j such that $r_{ij} \in \mathcal{D}$. Also, the likelihood of these r_{ij} are independent given u_i and V. For a particular r_{ij}, conditioning on all V is fine to write out in the abstract, but when we actually write out the distribution we will see that, according to our model assumption, r_{ij} only depends on u_i and v_j, and no other u or v.

  4. Therefore, $p(r_{ij}|u_i, V) = p(r_{ij}|u_i, v_j)$. Conditioning on more than is necessary is just superfluous and makes our parsing of the distributions and the dependencies they induce among the model variables less transparent—it is good practice to only condition on what is necessary according to the model and prior definitions.

- Using the likelihood and prior defined for this problem, we have

$$p(u_i|V, R) \propto \prod_{j:(i,j)\in\Omega} \text{Normal}(r_{ij}|u_i^T v_j, \sigma^2)\text{Normal}(u_i|0, cI) \tag{28}$$

  We have actually already calculated this last lecture for the Bayesian linear regression problem. In fact, from the perspective of u_i it's identically the same problem! From that calculation, we know that

$$p(u_i|V, R) = \text{Normal}(\mu_{u_i}, \Sigma_{u_i}) \tag{29}$$

$$\Sigma_{u_i} = \left(\frac{1}{c}I + \frac{1}{\sigma^2}\sum_{j:(i,j)\in\Omega} v_j v_j^T\right)^{-1} \qquad \mu_{u_i} = \Sigma_{u_i}\left(\frac{1}{\sigma^2}\sum_{j:(i,j)\in\Omega} r_{ij}v_j\right)$$

- By symmetry, we get a similar posterior distribution for each v_j. The resulting Gibbs sampling algorithm for this model is the following.

Gibbs sampling algorithm for the matrix factorization model

1. Randomly initialize each $u_i^{(0)}$ and $v_j^{(0)}$
2. For step $t = 1, \ldots, T$
   (a) For $i = 1, \ldots, N$ sample a new u_i from its conditional posterior distribution,

$$u_i^{(t)} \sim \text{Normal}(\mu_{u_i}^{(t)}, \Sigma_{u_i}^{(t)})$$

   where at iteration t we use the parameters

$$\Sigma_{u_i}^{(t)} = \left(\frac{1}{c} I + \frac{1}{\sigma^2} \sum_{j:(i,j)\in\Omega} v_j^{(t-1)} (v_j^{(t-1)})^T \right)^{-1} \qquad \mu_{u_i}^{(t)} = \Sigma_{u_i}^{(t)} \left(\frac{1}{\sigma^2} \sum_{j:(i,j)\in\Omega} r_{ij} v_j^{(t-1)} \right)$$

   (b) For $j = 1, \ldots, M$ sample a new v_j from its conditional posterior distribution,

$$v_j^{(t)} \sim \text{Normal}(\mu_{v_j}^{(t)}, \Sigma_{v_j}^{(t)})$$

   where at iteration t we use the parameters

$$\Sigma_{v_j}^{(t)} = \left(\frac{1}{c} I + \frac{1}{\sigma^2} \sum_{i:(i,j)\in\Omega} u_i^{(t)} (u_i^{(t)})^T \right)^{-1} \qquad \mu_{v_j}^{(t)} = \Sigma_{v_j}^{(t)} \left(\frac{1}{\sigma^2} \sum_{i:(i,j)\in\Omega} r_{ij} u_i^{(t)} \right)$$

- Notice that, because we choose to update all u_i first in each iteration, their conditional posteriors use the values of v_j in the *previous* iteration—those are the most recent values for v_j. When we then update all v_j, we use the values of u_i in the *current* iteration because these are the most recent values. This might give the impression that the order in which we update each variable will make a difference. However, *order does not matter*. We could have chosen to update all v_j first during each iteration. Or we could alternate between u_i and v_j in an arbitrary way. All that matters is that, when calculating the conditional posterior, we condition on the most recent values for all variables.