5th Symposium on Advances in Approximate Bayesian Inference, 2023 1–8

Variational-Bayes Derivation Made Easy

Anonymous Authors
Anonymous Institution

Abstract

Variational Bayes is a popular method for approximate inference but its derivation can be cumbersome. Here, we give a 3-step recipe to simplify the process. The key idea is to identify the posterior form by explicitly looking for linearity with respect to certain expectations. Given the form, we can simply "read-off" the terms needed in the VB update. We give several examples and also show the usefulness of the recipe in deriving other algorithms such as expectation-maximization and message passing.

1. Introduction

Since its introduction in the early 90s (Hinton and Van Camp, 1993; Saul et al., 1996; Jaakkola and Jordan, 1996), variational Bayes (VB) has become a prominent method for approximate inference but, despite significant progress in probabilistic programming, deriving VB updates still remain cumbersome for many. The main source of the difficulty is the need to derive closed-form expressions for the integrals. For example, consider observed data $\mathbf{y}$ and latent vector $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K)$ for a model $p(\mathbf{y}, \mathbf{z})$, and assume that we seek a mean-field posterior approximation: $p(\mathbf{z}|\mathbf{y}) \approx q(\mathbf{z}) = \prod_i q_i(\mathbf{z}_i)$. Then, a fixed point $q^*(\mathbf{z})$ of the evidence lower-bound (ELBO) satisfies the following (Bishop, 2006, Eq. 10.9):

$$\log q_i^*(\mathbf{z}_i) = \mathbb{E}_{q^*_{/i}}\left[\log p(\mathbf{y}, \mathbf{z})\right] + \text{const}, \tag{1}$$

where $q_i^*(\mathbf{z}_i)$ denotes the marginal of $\mathbf{z}_i$ and the right hand side marginalizes out the rest of the variables (denote by $\mathbf{z}_{/i}$) using an expectation with respect to their marginal $q^*_{/i}(\mathbf{z}_{/i})$.

For conjugate-exponential (CE) family models, the integrals have closed-form expressions, but deriving them can still be cumbersome. It is not uncommon to find papers where these derivations take many pages of work. With such long derivations, it is easy to lose the main intuition and elegance of the updates. Difficulty increases when non-conjugate factors are also present. For such cases, despite using automatic differentiation, we still need to make several difficult choices: Which optimizer to use for the non-conjugate parts and how to integrate them with the conjugate updating? How to set the learning rate so that the overall updates converge? And, when using coordinate-wise updates, in what sequence should we update the variables? Here, we present a recipe to simplify these difficulties.

The recipe we present consists of 3-steps which are based on an alternate way to express Eq. 1 (Khan and Lin, 2017; Khan and Rue, 2021): assuming q_i^* to have an exponential-family form, $q_i(\mathbf{z}_i) = \exp\left(\langle \boldsymbol{\lambda}_i, \mathbf{T}_i(\mathbf{z}_i)\rangle - A_i(\boldsymbol{\lambda}_i)\right)$ with natural parameter $\boldsymbol{\lambda}_i$, sufficient statistics $\mathbf{T}_i(\cdot)$, and log-parition $A_i(\cdot)$, we can write Eq. 1 as follows,

$$\boldsymbol{\lambda}_i^* = \nabla_{\boldsymbol{\mu}_i} \mathbb{E}_q\left[\log p(\mathbf{y}, \mathbf{z})\right]\big|_{\boldsymbol{\mu}_i = \boldsymbol{\mu}_i^*}, \tag{2}$$

© A. Authors.

where $\boldsymbol{\mu}_i = \mathbb{E}_{q_i}[\mathbf{T}(\mathbf{z}_i)]$ is the expectation parameter. The advantage of this formulation is that we do not have to think about the distribution forms of q_i or any integrals, rather simply use the definition of $(\boldsymbol{\lambda}_i, \boldsymbol{\mu}_i)$ from a look-up table (see Table 1 or Wikipedia).

Our 3-step recipe goes as follows,

1. Identify q_i by looking for linearity of $\mathbb{E}_q[\log p(\mathbf{y}, \mathbf{z})]$ with respect to some $\boldsymbol{\mu}_i$.
2. Compute $\nabla_{\boldsymbol{\mu}_i} \mathbb{E}_q[\log p(\mathbf{y}, \mathbf{z})]$ (hint: for CE, it is just the coefficient in front of $\boldsymbol{\mu}_i$).
3. Update $\boldsymbol{\lambda}_i \leftarrow (1 - \rho_i)\boldsymbol{\lambda}_i + \rho_i \nabla_{\boldsymbol{\mu}_i} \mathbb{E}_q[\log p(\mathbf{y}, \mathbf{z})]$ with learning rate ρ_i.

The steps above are perhaps familiar to VB experts or resemble their own derivation process, but the recipe is simply making the process more transparent by giving explicit steps one should be taking, which we believe will be useful, at least for beginners. For example, Step 1 explicitly instructs to *look* for linearity, which is not a new result (Winn and Bishop, 2005, p. 668), but it is more instructive than the standard derivations where we try to identify q_i by looking for a known distributional form, for example, see the derivation in (Blei et al., 2017, App. A) or in (Bishop, 2006, Sec. 10.2).

The main benefit of the recipe is that, while it is as simple to follow as the standard approach to derive VB, it also covers a wide-variety of cases, including conjugate, non-conjugate, and even deterministic factors. This is because Step 2 and 3 follow the Bayesian learning rule (BLR) framework of Khan and Rue (2021); Khan and Lin (2017) who show the recovery of a wide-variety of as special cases of a variational approach. Below, we give additional sub-steps for Step 3 to recover standard updates used in the literature.

(3a) To recover Bayes-rule, rewrite the update in your parameter of choice.

(3b) For coordinate-wise updates (used in EM and message passing), set $\rho_i = 1$.

(3c) For deterministic $\mathbf{z}_i$, choose q_i to be Gaussian and use the delta method to approximate the expectation. Do the same for variables using the Laplace approximation.

(3d) For non-conjugate terms, just use the gradient. Simplify it using reparameterization trick whenever feasible; see Khan and Rue (2021).

In most cases, using the recipe, we can avoid the need to derive integrals by simply look up the pair $(\boldsymbol{\lambda}_i, \boldsymbol{\mu}_i)$. It may not be obvious but marginalization for CE terms in Eq. 1 is equivalent to computing gradients in Eq. 2. This can be checked by simply multiplying Eq. 2 by $\mathbf{T}_i(\mathbf{z}_i)$ and taking an exponent to recover Eq. 1 (Khan and Rue, 2021, Sec. 5.1).

There are also other advantages: (1) there is no need to use different optimizers for conjugate and non-conjugate parts, and (2) the learning rate and update schedule need not be set carefully because the update converges under fairly general conditions; see Khan et al. (2016). On the other hand, coordinate-wise update may not always converge and require the solution to be unique for each coordinate; see Paquet (2014) for an example.

2. Examples

We will now show two examples. First we show application to a simple mixture-model example to illustrate the usage of the recipe. This is a conjugate model with one variable and the recipe simply recovers Bayes' rule.

Table 1: Exponential-family used in this paper with sufficient statistics $\mathbf{T}(\mathbf{z})$, expectation parameters $\boldsymbol{\mu}$, and natural parameters $\boldsymbol{\lambda}$. For more exhaustive lists, see Wikipedia.

Distribution	$\mathbf{T}(\mathbf{z})$	$\boldsymbol{\mu}$	$\boldsymbol{\lambda}$
Bernoulli with $p(z=1)=\pi$	z	π	$\log \frac{\pi}{1-\pi}$
Beta $\propto z^{\alpha-1}e^{-\beta z}$ with parameters α and β	$\begin{pmatrix} \log z \\ \log(1-z) \end{pmatrix}$	$\begin{pmatrix} \psi(\alpha)-\psi(\alpha+\beta) \\ \psi(\beta)-\psi(\alpha+\beta) \end{pmatrix}$	$\begin{pmatrix} \alpha-1 \\ \beta-1 \end{pmatrix}$
Gaussian $\mathcal{N}(\mathbf{z}\vert\mathbf{m},\mathbf{S}^{-1})$	$\begin{pmatrix} \mathbf{z} \\ \mathbf{z}\mathbf{z}^\top \end{pmatrix}$	$\begin{pmatrix} \mathbf{m} \\ \mathbf{m}\mathbf{m}^\top+\mathbf{S}^{-1} \end{pmatrix}$	$\begin{pmatrix} \mathbf{S}\mathbf{m} \\ -\frac{1}{2}\mathbf{S} \end{pmatrix}$

Ex 1 *Consider a mixture model with two components $p_0(\mathbf{y})$ and $p_1(\mathbf{y})$ for an observation $\mathbf{y}$ where mixture indicator z is sampled from a Bernoulli prior,*

$$p(\mathbf{y}|z) = p_1(\mathbf{y})^z p_0(\mathbf{y})^{1-z}, \text{ where } p(z) = \pi_0^z(1-\pi_0)^{1-z}. \tag{3}$$

The posterior over z can be obtained by using Bayes' rule,

$$p(z=1|\mathbf{y}) = \frac{\pi_0 p_1(\mathbf{y})}{\pi_0 p_1(\mathbf{y}) + (1-\pi_0)p_0(\mathbf{y})} \tag{4}$$

We can recover this posterior by following our recipe. The first step is to figure out q by figuring out its expectation parameter $\mu = \mathbb{E}_q[T(z)]$ for some function $T(z)$.

$$\begin{aligned}\mathbb{E}_q[\log p(\mathbf{y},z)] &= \mathbb{E}_q\left[z\log p_1(\mathbf{y}) + (1-z)\log p_0(\mathbf{y}) + z\log\pi_0 + (1-z)\log(1-\pi_0)\right] \\ &= \underbrace{\mathbb{E}_q(z)}_{\mu}\underbrace{\log\frac{\pi_0 p_1(\mathbf{y})}{(1-\pi_0)p_0(\mathbf{y})}}_{\textit{Coeff. in front}} + \log[(1-\pi_0)p_0(\mathbf{y})].\end{aligned} \tag{5}$$

Therefore $T(z) = z$ with $z \in 0,1$ and $\boldsymbol{\mu} = \mathbb{E}_q(z)$, which corresponds to a Bernoulli distribution; see the first row in Table 1. The second step is to compute the gradient which is simply the coefficient in front of $\mathbb{E}_q(z)$. For the third step, we can get the natural parameter $\lambda = \log\pi_1/(1-\pi_1)$ where we denote the Bernoulli parameter by π_1 to contrast it with the prior π_0. We use this in the third step to get,

$$\lambda_* = \log\frac{\pi_0 p_1(\mathbf{y})}{(1-\pi_0)p_0(\mathbf{y})} \quad \Longrightarrow \pi_1^* = \frac{\pi_0 p_1(\mathbf{y})}{\pi_0 p_1(\mathbf{y}) + (1-\pi_0)p_0(\mathbf{y})}$$

where, in the first equality, we follow Step 3b to use $\rho = 1$, and in the second equality, we use Step 3a to rewrite it in terms of π_1 by using $\pi_1 = \frac{1}{1+e^{-\lambda}}$. This recovers the posterior obtained by using the Bayes rule.

Let us see another example with a slightly more complex model where a mean-field approximation is required. We will see that, unlike standard derivation, we do not have to start with a mean-field structure and we can figure this out by simply looking for linearity.

This is unlike standard derivation, where the factorization is often chosen first and the form is identified by looking for familiar distributional.

Ex 2 *We will add an additional latent variable:* $\pi_0 \to z_i \to \mathbf{y}_i$ *where we have multiple* vector *observations* $\mathbf{y}_i$*, and two sets of latent variables* z_i*, for all* $i = 1$ *to* N*, and* π_0*. We assume the following model with a beta prior over* π_0*,*

$$p(\mathbf{Y}|\mathbf{z}) = \prod_{i=1}^{N} p_1(\mathbf{y}_i)^{z_i} p_0(\mathbf{y}_i)^{1-z_i},\ p(z_i|\pi_0) = \pi_0^{z_i}(1-\pi_0)^{1-z_i},\ p(\pi_0) \propto \pi_0^{\alpha_0-1}(1-\pi_0)^{\beta_0-1}$$

Our goal is to estimate $p(\mathbf{z}, \pi_0|\mathbf{Y})$ *but this is not conjugate and we have to use a mean-field candidate for tractability. However, we can still apply our recipe as we did before.*

The first step is to figure out q *by looking for sufficient statistics of* z_i *and* π_0 *in which the expected log-joint is (multi-)linear (no need to decide the factorization yet),*

$$\begin{aligned}
&\mathbb{E}_q[\log p(\mathbf{Y}|\mathbf{z})p(\mathbf{z})p(\pi_0)] = \mathbb{E}_q\left[\left(\sum_{i=1}^{N} \log\left(p(\mathbf{y}_i|z_i)p(z_i)\right)\right) + \log p(\pi_0)\right] \\
&= \mathbb{E}_q\left[\left(\sum_{i=1}^{N} z_i \frac{\pi_0 p_1(\mathbf{y}_i)}{(1-\pi_0)p_0(\mathbf{y}_i)} + \log[(1-\pi_0)p_0(\mathbf{y}_i)]\right) + \log \pi_0^{(\alpha_0-1)} + \log(1-\pi_0)^{(\beta_0-1)}\right] \\
&= \left(\sum_{i=1}^{N} \underbrace{\mathbb{E}_q(z_i)}_{=\mu_i} \underbrace{\mathbb{E}_q\begin{bmatrix} \log \pi_0 \\ \log(1-\pi_0) \end{bmatrix}^\top}_{=\boldsymbol{\mu}_0^\top} \underbrace{\begin{bmatrix} p_1(\mathbf{y}_i) \\ p_0(\mathbf{y}_i) \end{bmatrix}}_{\textit{Coeff. in front}}\right) + \underbrace{\mathbb{E}_q\begin{bmatrix} \log \pi_0 \\ \log(1-\pi_0) \end{bmatrix}^\top}_{=\boldsymbol{\mu}_0^\top} \underbrace{\begin{bmatrix} \alpha_0 - 1 \\ N + \beta_0 - 1 \end{bmatrix}}_{\textit{Coeff. in front}} \\
&\quad + \sum_i \log p_0(\mathbf{y}_i).
\end{aligned}$$

We made a mean-field assumption: $q(\mathbf{z}, \pi_0) = \prod_i q(z_i)q(\pi_0)$ *but we did this only in the last step and simply to ensure that the expected log-joint is linear w.r.t. both* μ_i *and* $\boldsymbol{\mu}_0$*.*

After this, the second step is straightforward to simply read the coefficient in front of μ_i *and* $\boldsymbol{\mu}_0$*, and using the third step we can write the update,*

$$\lambda_i \leftarrow (1-\rho_i)\lambda_i + \rho_i\, \boldsymbol{\mu}_0^\top \begin{bmatrix} p_1(\mathbf{y}_i) \\ p_0(\mathbf{y}_i) \end{bmatrix},\ \textit{for all } i = 1, 2, \ldots, N \tag{6}$$

$$\boldsymbol{\lambda}_0 \leftarrow (1-\rho_0)\boldsymbol{\lambda}_0 + \rho_0 \left(\begin{bmatrix} \alpha_0 - 1 \\ N + \beta_0 - 1 \end{bmatrix} + \sum_{i=1}^{N} \mu_i \begin{bmatrix} p_1(\mathbf{y}_i) \\ p_0(\mathbf{y}_i) \end{bmatrix}\right). \tag{7}$$

The updates can be rewritten in terms of the parameters of Bernoulli and Beta distributions by using the expressions for $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ *given in first two rows of Table 1.*

In the above example, one can verify the following:

1. If we set $\rho_i = \rho_0 = 1$ then the update reduce to variational message passing.

2. If we set $\rho_i = 1$ only, then this is equivalent to stochastic variational inference where only one (local variable) z_i is updated in every iteration followed by the update of (the global variable) π_0.

3. Finally, if we additionally do a coordinate-wise update, where only one variable is picked at a time, then this will be a variational-EM like procedure which is sometime called co-ordinate ascent VI or CAVI.

Another example of matrix factorization, illustrating another point regarding obtaining updates of deterministic factors, is given in App. A. There we show that a delta approximation in the expectation parameters can recover the Maximum-a-Posterior (MAP) estimate.

We now briefly discuss the inclusion of non-conjugate terms.

Ex 3 *Suppose in the previous example we decided to use a non-conjugate prior, say, a logit-normal distribution with mean m,*

$$p(\pi_0) \propto \frac{1}{\pi_0(1-\pi_0)} e^{-\frac{(logit(\pi_0)-m)^2}{2}}.$$

Still, the derivation proceeds in the same way by expanding $\mathbb{E}_q[\log p(\mathbf{Y}|\mathbf{z})p(\mathbf{z})p(\pi_0)] =$

$$\left(\sum_{i=1}^{N} \underbrace{\mathbb{E}_q(z_i)}_{=\mu_i} \underbrace{\mathbb{E}_q \begin{bmatrix} \log \pi_0 \\ \log(1-\pi_0) \end{bmatrix}^\top}_{=\boldsymbol{\mu}_0^\top} \underbrace{\begin{bmatrix} p_1(\mathbf{y}_i) \\ p_0(\mathbf{y}_i) \end{bmatrix}}_{\textit{Coeff. in front}} \right) + N \underbrace{\mathbb{E}_q[\log(1-\pi_0)]}_{\mu_0^{(2)}} + \underbrace{\mathbb{E}_q[\log p(\pi_0)]}_{\textit{non-conjugate term}} .$$

and combine the last two terms by collecting the linear terms together,

$$\underbrace{\mathbb{E}_q \begin{bmatrix} \log \pi_0 \\ \log(1-\pi_0) \end{bmatrix}^\top}_{=\boldsymbol{\mu}_0^\top} \underbrace{\begin{bmatrix} -1 \\ N-1 \end{bmatrix}}_{\textit{Coeff. in front}} + \underbrace{\mathbb{E}_q \left[-\frac{(logit(\pi_0)-m)^2}{2} \right]}_{\textit{non-conjugate term}} .$$

We will now just have to add the last term to the derivative of $\nabla_{\boldsymbol{\mu}_0}$, and rewrite the update. The form of the update does not change much, rather the non-conjugate prior can simply be written as a "pseudo" conjugate Beta prior,

$$\boldsymbol{\lambda}_0 \leftarrow (1-\rho_0)\boldsymbol{\lambda}_0 + \rho_0 \left(\begin{bmatrix} \hat{\alpha}_0 - 1 \\ N + \hat{\beta}_0 - 1 \end{bmatrix} + \sum_{i=1}^{N} \mu_i \begin{bmatrix} p_1(\mathbf{y}_i) \\ p_0(\mathbf{y}_i) \end{bmatrix} \right).$$

where $\hat{\alpha}_0 = \nabla_{\mu_0^{(1)}} \mathbb{E}_q \left[-\frac{1}{2}(logit(\pi_0)-m)^2 \right]$ and $\hat{\beta}_0 = \nabla_{\mu_0^{(2)}} \mathbb{E}_q \left[-\frac{1}{2}(logit(\pi_0)-m)^2 \right]$ are the parameters of the pseudo beta prior. The gradients automatically convert the non-conjugate prior form into a conjugate one, which is the main idea of Khan and Lin (2017). The gradients are easy to compute through automatic differentiation since we know the expression for $\boldsymbol{\mu}_0$ from Table 1.

3. Limitations and Future Work

One drawback of the recipe is that it depends heavily on the $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ parameterization. There are several works that extend beyond, for example, to mixture distributions (Lin et al., 2019) and also to transformation families (Kıral et al., 2023). Such approaches have shown to be useful in deriving existing algorithms, as well as designing new ones (Khan and Rue, 2021). We believe that such extensions can be found in general and that it is possible to derive all sorts of updates in the same fashion as described in this recipe.

Acknowledgments

Acknowledgements go here.

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Annual Conference on Computational Learning Theory*, pages 5–13, 1993.

T. Jaakkola and M. Jordan. A variational approach to Bayesian logistic regression problems and their extensions. In *International conference on Artificial Intelligence and Statistics*, 1996.

Mohammad Emtiyaz Khan and Wu Lin. Conjugate-computation variational inference: converting variational inference in non-conjugate models to inferences in conjugate models. In *International Conference on Artificial Intelligence and Statistics*, pages 878–887, 2017.

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

Mohammad Emtiyaz Khan, Reza Babanezhad, Wu Lin, Mark Schmidt, and Masashi Sugiyama. Faster stochastic variational inference using proximal-gradient methods with general divergence functions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2016.

Eren Mehmet Kıral, Thomas Möllenhoff, and Mohammad Emtiyaz Khan. The lie-group bayesian learning rule. In *International conference on Artificial Intelligence and Statistics*, 2023.

Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. 2019.

Ulrich Paquet. On the convergence of stochastic variational inference in Bayesian networks. *NIPS Workshop on variational inference*, 2014.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

John Winn and Christopher M Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

Appendix A. Deriving Matrix Factorization

Ex 4 *Matrix factorization and latent factor models aim to fit data matrix* $\mathbf{Y}$ *of size* $N \times D$ *by using two sets of factors* $\mathbf{U}$ *and* $\mathbf{V}$ *respectively of sizes* $N \times K$ *and* $D \times K$ *where* K *is the number of factors. The likelihood and the prior are given by,*

$$p(\mathbf{Y}|\mathbf{U},\mathbf{V}) = \prod_{i=1}^{N}\prod_{j=1}^{D}\mathcal{N}(y_{ij}|\mathbf{u}_i^\top\mathbf{v}_j,\mathbf{I}), \quad p(\mathbf{u}_i) = \mathcal{N}(\mathbf{u}_i|0,\mathbf{I}/\delta_u), \quad p(\mathbf{v}_j) = \mathcal{N}(\mathbf{v}_j|0,\mathbf{I}/\delta_v)$$

where y_{ij} *is the* ij*'th entry and* $\mathbf{v}_j$ *is the* j*'th row of* $\mathbf{V}$*. The most popular procedure is to use an alternating least-squares (ALS) procedure (more correctly "alternating ridge regression"), where we interpret* $\mathbf{y}_i \approx \mathbf{V}\mathbf{u}_i$ *as ridge regression, and estimate* $\mathbf{u}_i$ *given an estimate* $\hat{\mathbf{v}}_i$*,*

$$\hat{\mathbf{u}}_i \leftarrow \left(\sum_{j=1}^{D}\hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^\top + \delta_u\mathbf{I}_K\right)^{-1}\left(\sum_{j=1}^{D}\hat{\mathbf{v}}_j y_{ij}\right), \quad \textit{for all } i = 1,2,\ldots,N \tag{8}$$

In a similar fashion, we can estimate $\mathbf{v}_j$ *as well, and iterate.*

This algorithm and its Bayesian versions (e.g., expectation maximization and VB) can all be derived by using our recipe. Following Step 1, we look for linearity first,

$$\mathbb{E}_q[\log p(\mathbf{Y},\mathbf{U},\mathbf{V})] = \mathbb{E}_q\left[\sum_{i=1}^{N}\left(\sum_{j=1}^{D}-\frac{1}{2}(y_{ij}-\mathbf{u}_i^\top\mathbf{v}_j)^2 - \frac{\delta_v}{2}\mathbf{v}_j^\top\mathbf{v}_j\right) - \frac{\delta_u}{2}\mathbf{u}_i^\top\mathbf{u}_i\right] + \textit{const}$$

$$= \sum_{i=1}^{N}\left(\sum_{j=1}^{D}-\frac{1}{2}\left(Tr\left(\underbrace{\mathbb{E}_q(\mathbf{u}_i\mathbf{u}_i^\top)}_{=\boldsymbol{\mu}_{u_i}^{(2)}}\underbrace{\mathbb{E}_q(\mathbf{v}_j\mathbf{v}_j^\top)}_{=\boldsymbol{\mu}_{v_j}^{(2)}}\right) - 2y_{ij}\underbrace{\mathbb{E}_q(\mathbf{u}_i)}_{=\boldsymbol{\mu}_{u_i}^{(1)}}^\top\underbrace{\mathbb{E}_q(\mathbf{v}_j)}_{\boldsymbol{\mu}_{v_j}^{(1)}}\right)\right.$$

$$\left. - \frac{\delta_v}{2}Tr\underbrace{\mathbb{E}_q\left(\mathbf{v}_j\mathbf{v}_j^\top\right)}_{=\boldsymbol{\mu}_{v_j}^{(2)}}\right) - \frac{\delta_u}{2}Tr\underbrace{\mathbb{E}_q\left(\mathbf{u}_i\mathbf{u}_i^\top\right)}_{=\boldsymbol{\mu}_{u_i}^{(2)}} + \textit{const}$$

Here we had to use mean-field approximation between among all $\mathbf{u}_i$ and $\mathbf{v}_j$ because we want linearity with respect to each in the first and second terms (arising due to the likelihood). The sufficient statistics correspond to Gaussian (third row in Table 1).

The gradient in the second step can be simply read off as the coefficients in the front, and get the update by substituting in Step 3,

$$\mathbf{S}_{u_i}\mathbf{m}_{u_i} \leftarrow (1-\rho_i)\mathbf{S}_{u_i}\mathbf{m}_{u_i} - \rho_i\left(\sum_{j=1}^{D}\boldsymbol{\mu}_{v_j}^{(1)}y_{ij}\right)$$

$$\mathbf{S}_{u_i} \leftarrow (1-\rho_i)\mathbf{S}_{u_i} + \rho_i\left(\sum_{j=1}^{D}\boldsymbol{\mu}_{v_j}^{(2)} + \delta_u\mathbf{I}_K\right)$$

$$\mathbf{S}_{v_j}\mathbf{m}_{v_j} \leftarrow (1-\rho_j)\mathbf{S}_{v_j}\mathbf{m}_{v_j} - \rho_j\left(\sum_{i=1}^{N}\boldsymbol{\mu}_{u_i}^{(1)}y_{ij}\right)$$

$$\mathbf{S}_{v_j} \leftarrow (1-\rho_j)\mathbf{S}_{v_j} + \rho_j\left(\sum_{i=1}^{N}\boldsymbol{\mu}_{u_i}^{(2)} + \delta_v\mathbf{I}_K\right).$$

We have written this update in terms of mean and precision below to show equivalent to ALS. In fact, we can obtain VMP, EM, and ALS as special cases,

1. *Setting $\rho_t = 1$ and updating posteriors of $\mathbf{U}$ and $\mathbf{V}$ in alternate iterations (that is coordinate wise updates), we recover the VMP update.*

2. *We can recover probabilistic PCA updates (Tipping and Bishop, 1999), where we only compute the posterior wrt $\mathbf{U}$, by additionally doing the following,*

   (a) denoting $\mathbf{m}_{v_j}$ by $\hat{\mathbf{v}}_j$, and

   (b) using the delta approximation for $\boldsymbol{\mu}_{v_j}^{(2)} = \mathbb{E}_q[\mathbf{v}_j\mathbf{v}_j^\top] \approx \mathbb{E}_q[\mathbf{v}_j]\mathbb{E}_q[\mathbf{v}_j]^\top = \hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^\top$.

   The updates are shown below

$$\mathbf{S}_{u_i}\mathbf{m}_{u_i} \leftarrow \sum_{j=1}^{D}\hat{\mathbf{v}}_j y_{ij}, \quad \mathbf{S}_{u_i} \leftarrow \sum_{j=1}^{D}\hat{\mathbf{v}}_j\hat{\mathbf{v}}_j^\top + \delta_u\mathbf{I}_K \quad \textit{for all } i \tag{9}$$

$$\hat{\mathbf{v}}_j \leftarrow \left(\sum_{i=1}^{N}\boldsymbol{\mu}_{u_i}^{(2)} + \delta_v\mathbf{I}_K\right)^{-1}\left(\sum_{i=1}^{N}\boldsymbol{\mu}_{u_i}^{(1)}\,y_{ij}\right), \quad \textit{for all } j. \tag{10}$$

3. *Similarly, the ALS updates in Eq. 8 is obtained by further denoting $\mathbf{m}_{u_i}$ by $\hat{\mathbf{u}}_i$, and adding another delta approximation: $\boldsymbol{\mu}_{u_i}^{(2)} = \mathbb{E}_q[\mathbf{u}_i\mathbf{u}_i^\top] \approx \mathbb{E}_q[\mathbf{u}_i]\mathbb{E}_q[\mathbf{u}_i]^\top = \hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^\top$.*

Our recipe provides a more flexible update where all quantities can be updated in parallel. Such parallel updates are often employed in VMP but do not always converge (Paquet, 2014). The update fix this issue by using $\rho_t < 1$.