Supplement for Neuro-Biologically Plausible Probabilistic Inference Inspired by Topic Models

Anonymous Author(s)
Affiliation
Address
email

1 Full Network Derivation for LDA

Variational inference for LDA is summarized by iterative equations which take the form

$$p_{d,i,j} = \frac{\exp\left(\psi\left(\eta_{i,j}\right) - \psi\left(\bar{\eta}_j\right)\right)\exp\psi\left(\alpha_{d,j}\right)}{\sum_k \exp\left(\psi\left(\eta_{i,k}\right) - \psi\left(\bar{\eta}_k\right)\right)\exp\psi\left(\alpha_{d,k}\right)} \tag{1}$$

$$\alpha_{d,j} = \alpha_0 + \sum_i R_{d,i} p_{d,i,j} \tag{2}$$

$$\eta_{i,j} = \eta_0 + \sum_d R_{d,i} p_{d,i,j} \tag{3}$$

When the ith element of vector $\eta_j = \eta_{i,j}$ the approximate posterior is given by a product of marginals which take the form

$$q(\beta_j|\eta_j) = Dir(\eta_j) \tag{4}$$

$$q(\theta_d) = Dir(\alpha_d) \tag{5}$$

$$q(N_{d,i,:}) = Mult(p_{d,i,:}, R_i) \tag{6}$$

$$\tag{7}$$

To obtain the network equations we define intermediate computation quantities:

$$w_{ij} = \exp\left(\psi\left(\eta_{ij}\right)\right) \tag{8}$$

$$\bar{\eta}_j = \sum_i \eta_{ij} \tag{9}$$

$$\bar{R}_{d,i} = \sum_j w_{i,j} \exp\left(\psi\left(\alpha_{d,j}\right)\right) \tag{10}$$

$$\bar{N}_{d,i,j} = w_{i,j} \frac{R_{d,i}}{\bar{R}_{d,i}} \tag{11}$$

This allows us to rerwite the update equations for inference on single document as

$$\bar{N}_{d,i,j} = w_{i,j} \frac{R_{d,i}}{\bar{R}_{d,i}} \tag{12}$$

$$\alpha_{d,j} = \alpha_0 + \exp\left(\psi\left(\bar{\eta}_j\right)\right)\exp(\alpha_{d,j}) \sum_i \bar{N}_{d,i,j} \tag{13}$$

Clearly, for fixed w_{ij}, iterating between these two equations converges to $\alpha_{d,j}$. The network which performs inference is then built by constructing a continuous time dynamical system which builds converges to the same fixed point is these two iterative equations, i.e. it has the property that

$$\frac{d}{dt}\alpha_{d,j} = \frac{d}{dt}\bar{N}_{d,i,j} = 0 \tag{14}$$

when the iterative equations above are satisfied. One way to accomplish this is to set

$$\frac{d}{dt}\bar{N}_{d,i,j} = w_{i,j}R_{d,i} - \bar{N}_{d,i,j}\bar{R}_i \tag{15}$$

$$\frac{d}{dt}\alpha_{d,j} = \exp\left(\psi\left(\bar{\eta}_j\right)\right)(\alpha_0 - \alpha) + \exp(\alpha_{d,j})\sum_i \bar{N}_{d,i,j} \tag{16}$$

To implement the learning rule. We utilize the approximation $\exp\left(\psi\left(x\right)\right) \approx x - 1/2$ for $x > 1$ so that $w_{i,j} = \eta_{i,j} - 1/2$. This allows us to write Hoffman's online learning rule

$$\tau_L \frac{d}{dt}\eta_{i,j} = \eta_0 - \eta_{i,j} + R_{t,i}p_{t,i,j} \tag{17}$$

as

$$\tau_L \frac{d}{dt}w_{i,j} = \eta_0 - 1/2 - w_{i,j} + \sum_d N_{d,i,j}\exp\left(\psi\left(\alpha_{d,j}\right)\right) \tag{18}$$

Rescaling the right hand side by a positive quantity does not affect the location of the fixed point of the dynamics and results in

$$\tau_L \frac{d}{dt}\eta_{i,j} = \exp\left(\psi\left(\bar{\eta}_j\right)\right)\left(\eta_0 - \eta_{i,j}\right) + \exp\left(\psi\left(\bar{\eta}_j\right)\right)\sum_d R_{d,i}p_{d,i,j} \tag{19}$$

$$= \exp\left(\psi\left(\bar{\eta}_j\right)\right)\left(\eta_0 - \eta_{i,j}\right) + \exp(\alpha_{d,j})\bar{N}_{d,i,j} \tag{20}$$

as desired.

2 Full VB Derivation for Online DDM inference

2.1 Network Implementation

We assume that the quantity of interest is the current distribution of latent causes, $p(\theta(t)|R(\tau), \tau = 0..T)$, where $R_i(t)$ represents the spikes from neuron i as a sum of delta functions. If no spikes occur then no evidence is presented and posterior inference over $\theta(t)$ is simply given by an undriven Kalman filter with parameters (Λ, Σ_D). A recurrent neural network which uses a PPC to encode a posterior which evolves according to a Kalman filter has the property that neural responses are linearlly related to the inverse covariance matrix of the posterior and inverse covariance times the posterior mean. In the absence of evidence, it is easy to show that these quantities must evolve according to quadratic recurrent dynamics which implement divisive normalization. Thus, the patterns of neural activity which linearlly encode them must do so as well. When a new spike arrives, optimal inference is no longer possible and a variational approximation must be utilized. As is shown in the supplement this variational approximation is similar to the variational approximation used for LDA. As a result, a network which can divisively inhibit its synapses is once again sufficient for the purposes of implementing approximate Bayesian inference.

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{\Sigma}^{-1} &= \mathrm{Diag}\left(\exp\left(\boldsymbol{\mu}+\frac{1}{2}\mathrm{Diag}\left(\boldsymbol{\Sigma}\right)\right)\right)+\boldsymbol{\Lambda}^{T}\boldsymbol{\Sigma}^{-1}+\boldsymbol{\Sigma}^{-1}\boldsymbol{\Lambda}-\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{D}\boldsymbol{\Sigma}^{-1} \qquad (21)\\
\frac{d}{dt}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)_{j} &= \sum_{i}R_{i}^{t}\frac{w_{ij}\exp\left(\mu_{t,j}\right)}{\sum_{k}w_{ik}\exp\left(\mu_{t,k}\right)}-\exp\left(\mu_{j}+\frac{1}{2}\mathrm{Diag}\left(\boldsymbol{\Sigma}\right)_{j}\right)+\left(\boldsymbol{\Lambda}^{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}-\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_{D}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)_{j}
\end{aligned}$$

2.2 Full Derivation

Just as the Dirichlet-Multinomial formulation of LDA has an equivalent Gamma-Poisson formulation, so too can we consider a Log Normal-Poisson formulation of the correlated topic model. The generative model takes the form

$$\begin{gathered}
\mathbf{z}\left(t\right)\sim N\left(\mathbf{z}(t-\Delta t)+\Delta t\boldsymbol{\Lambda}\left(\mu_{0}-\mathbf{z}(t-\Delta t)\right),\Delta t\boldsymbol{\Sigma}_{D}\right)\\
c_{j}\left(t\right)=\exp\left(z_{j}\left(t\right)\right)\\
g\left(t\right)\sim N\left(g(t-\Delta t)+\Delta t\lambda_{g}\left(\mu_{g_{0}}-g(t-\Delta t)\right),\Delta t\sigma_{g}^{2}\right)\\
\mathbf{w}_{i}\sim Dir\left(\alpha\right)\\
N_{ij}\left(t\right)\sim Poi\left(w_{ij}c_{j}\left(t\right)\Delta t\right)\\
R_{i}\left(t\right)=\sum_{j}N_{ij}\left(t\right)\sim Poi\left(\Delta t\sum_{j}w_{ij}c_{j}\left(t\right)\right)
\end{gathered} \qquad (22)$$

Since we are doing online inference we are only interested in updating the posterior for a single time step in the forward direction using the posterior from the previous time step as the posterior for the next. For a time step of size Δt.

$$\begin{gathered}
\log\left(\mathbf{R}^{t},\mathbf{N}^{t},\mathbf{z}^{t}\right)\sim\log\left(\delta\left(R_{i}^{t}-\sum_{j}N_{ij}^{t}\right)\right)+\sum_{ij}N_{ij}^{t}\left(\log w_{ij}+z_{j}^{t}+\log\Delta t\right)-\Delta t\sum_{j}\exp\left(z_{j}^{t}\right)\\
-\tfrac{1}{2}\left(\mathbf{z}^{t}-\mu_{t-\Delta t}-\Delta t\boldsymbol{\Lambda}\left(\mu_{0}-\mu_{t-\Delta t}\right)\right)\cdot\\
\left(\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)\boldsymbol{\Sigma}_{t-\Delta t}\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)^{T}+\Delta t\boldsymbol{\Sigma}_{D}\right)^{-1}\left(\mathbf{z}^{t}-\mu_{t-\Delta t}-\Delta t\boldsymbol{\Lambda}\left(\mu_{0}-\mu_{t-\Delta t}\right)\right)
\end{gathered} \qquad (23)$$

A variational posterior of the form

$$q\left(\mathbf{N},\mathbf{z}\right)=q\left(\mathbf{N}_{i,:}|{R_{i}}^{t},p_{i,:}^{t}\right)q\left(\mathbf{z}|\mu^{t},\boldsymbol{\Sigma}^{t}\right) \qquad (24)$$

This immediately results in an update rule for the Multinomial distribution which takes the familiar form,

$$\left\langle N_{ij}^{t}\right\rangle=R_{i}\frac{w_{ij}\exp(\mu_{j}^{j})}{\sum_{k}w_{ik}\exp(\mu_{k}^{j})} \qquad (25)$$

Update rules for the Gaussian posterior on $\mathbf{z}(t)$ are obtained for two different cases. Case 1: no spikes are present. In this case we minimizing the residual KL divergence by gradient descent, i.e. by taking the derivative with respect to the parameters of interest of

$$\begin{gathered}
KL\left(q\left(z|\mu_{t},\boldsymbol{\Sigma}_{t}\right)|p\right)\sim\sum_{ij}\left\langle N_{ij}^{t}\right\rangle\mu_{t,j}-\Delta t\sum_{j}\exp\left(\mu_{t,j}+\tfrac{1}{2}\sigma_{t,j}^{2}\right)\\
-\tfrac{1}{2}Tr\left(\left(\boldsymbol{\Sigma}_{t}+\mu_{t}\mu_{t}^{T}\right)\left(\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)\boldsymbol{\Sigma}_{t-\Delta t}\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)^{T}+\Delta t\boldsymbol{\Sigma}_{D}\right)^{-1}\right)\\
+\mu_{t}\cdot\left(\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)\boldsymbol{\Sigma}_{t-\Delta t}\left(\mathbf{I}-\Delta t\boldsymbol{\Lambda}\right)^{T}+\Delta t\boldsymbol{\Sigma}_{D}\right)^{-1}\left(\mu_{t-\Delta t}+\Delta t\boldsymbol{\Lambda}\left(\mu_{0}-\mu_{t-\Delta t}\right)\right)\\
+\tfrac{1}{2}\log\left|\boldsymbol{\Sigma}_{t}\right|
\end{gathered} \qquad (26)$$

This results in

$$\begin{aligned}\frac{d}{dt}\mathbf{\Sigma}^{-1} &= Diag\left(\exp\left(\mu + \tfrac{1}{2}diag\left(\mathbf{\Sigma}\right)\right)\right) + \mathbf{\Lambda}^T\mathbf{\Sigma}^{-1} + \mathbf{\Sigma}^{-1}\mathbf{\Lambda} - \mathbf{\Sigma}^{-1}\mathbf{\Sigma}_D\mathbf{\Sigma}^{-1} \\ \frac{d}{dt}\mathbf{\Sigma}^{-1}\mu &= -\exp\left(\mu + \tfrac{1}{2}diag\left(\mathbf{\Sigma}\right)\right) + \mathbf{\Lambda}^T\mathbf{\Sigma}^{-1}\mu - \mathbf{\Sigma}^{-1}\mathbf{\Lambda}\mu_0 - \mathbf{\Sigma}^{-1}\mathbf{\Sigma}_D\mathbf{\Sigma}^{-1}\mu\end{aligned} \tag{27}$$

In the second case, we suppose that a spike in the input layer has occurred. Since the underlying spike generation process is Poisson, this event occurs in a time window if size zero. This allows us to greatly simplify inference by considering only those terms in the log probability which are present in the limit as Δt goes to zero.

This results in update equation:

$$\begin{aligned}\left\langle N_{ij}^t \right\rangle &= R_i^t \tfrac{w_{ij}\exp(\mu_{t,j})}{\sum_k w_{ik}\exp(\mu_{t,k})} \\ \mathbf{\Sigma}_t^{-1} &= \mathbf{\Sigma}_{t^-}^{-1} \\ \mathbf{\Sigma}_t^{-1}\mu_t &= \mathbf{\Sigma}_{t^-}^{-1}\mu_{t^-} + \sum_i \left\langle \mathbf{N}_i^t \right\rangle\end{aligned} \tag{28}$$

Now, full VB inference would require that we evolve the dynamics of equation 27 until a spike occurs then iterate the equations 28 until convergence before continuing. This was the procedure used to generate the x-axis values of the blue dots in Figure 6 right in the main text. A more biologically plausible scheme would only iterate 28 only once. This results in the DDM inference equation in the main text. This approximation results in almost zero information loss due in large part to the fact that prior provided by knowledge of the dynamics is actually quite informative.

3 Simulations

The points in Figure 6 were generated by sampling from the generative models for LDA and the DDM and then applying the network implementation of VBEM or VBEM itself to those samples. The output of both algorithms is a set of natural parameters associated with each olfactory scene or in the case of the DDM each time step.

3.1 LDA

For LDA, we only plotted the natural parameters of the gamma distributed posterior over concentration, $q_{\mathbf{c}}(\mathbf{c}|\alpha)$. Here, the natural parameter is the shape parameter of the gamma distributed posterior which is proportional to the network's estimate of the concentration. We considered a bank of 100 odors and 20 olfactory receptor neurons. Odors were drawn randomly from a symmetric Dirichlet distribution with parameter $\alpha = 1/2$. Each olfactory scene contained on average 5 odors with concentration significantly different from zero. Average concentration level was chosen so that each olfactory receptor neuron fired on average 25 spikes per trial.

3.2 DDM

For the DDM the natural parameters of the posterior distribution over log concentration are the precision matrix (inverse covariance matrix) and the precision matrix times them posterior mean. In this case we also considered 100 odors and 20 sensory input neurons and generated odors from a symmetric Dirichlet with $\alpha = 1/2$. The dynamics of the OU process which governed the behaviour of the odor concentration were given by $\mathbf{\Lambda} = 2\mathbf{I}$ and $\mathbf{\Sigma}_D = 4log(21)\mathbf{I}$. This ensures that, once again, there are about 5 odors which are present at a significant level of concentration at any given time.

4 Table of properties for LDA, ICA, and L1-based networks

Neural network implementations of LDA, ICA, and L1 make very different predictions for neural computation. ICA is a feedforward algorithm. Network implementations of L1 typically use linear recurrent inhibition to compute a prediction of the input image which is then subtracted from the actual image. Sparsity is then achieved via a competitive race to explain mechanism. The network

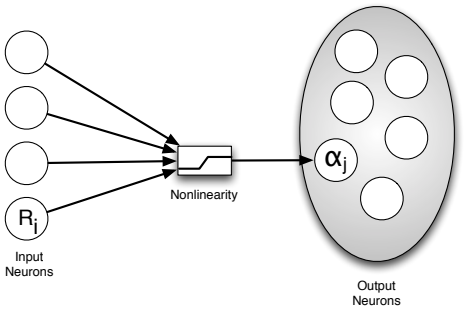

Figure 1: ICA network model

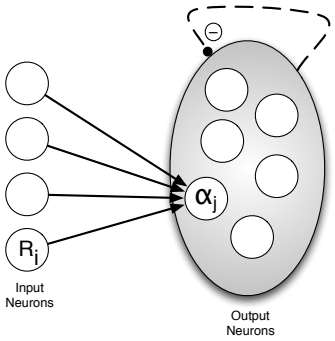

Figure 2: L1 network model

implementation of LDA, on the other hand, divides the image by the prediction to generate the residual signal which drives the refinement of the representation. Other differences are summarized in the table below.

Table 1: Network Model Comparison

Attribute	ICA	L1	LDA
Reccurently Modulated	N	Y	Y
Divisive Normalization	N	N	Y
Subtractive Inhibition	Y	Y	Y
Only Positive Weights	N	N	Y
Contrast Invariance	N	N	Y

5 Comparison of LDA, ICA and L1 regression on Natural Images

Like ICA and L1 penalized regression, LDA has at its core a linear mixing model capable of implementing a sparse prior. In the Gamma-Poisson formulation, this is achieve by setting the α parameter of the gamma distribution on the un-normalized topic distribution to a value less than one. This results is greater excess kurtosis than the L1 prior. The primary differences between the LDA and these other methods are that (1) LDA naturally functions in the domain of count data, and (2) LDA naturally enforces a non-negativity constraint on the mixture components. In this sense it is similar to Non-negative Matrix Factorization (NMF). In the visual domain NMF applied to a data set of whitened natural images has been shown to be capable of reproducing oriented receptive fields similar to those obtained by Olshausen and Field using L1 penalized regression and van Haterern and van der Schaff using ICA [1, 2, 3]. Here we extend this result by showing that NMF implemented via variational inference applied to LDA also generates oriented filters when applied to *un-whitened* natural images. We used ICA code and natural images from [3] and L1 code from [2]. This results in a marked difference between LDA and ICA and L1 penalized regression, the latter two of which fail miserably at reproducing any kind of a sensible representation when applied to un-whitened data. See Figure 3.

References

[1] Patrik O Hoyer. Non-negative Matrix Factorization with Sparseness Constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[2] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[3] J.H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1394):359, 1998.

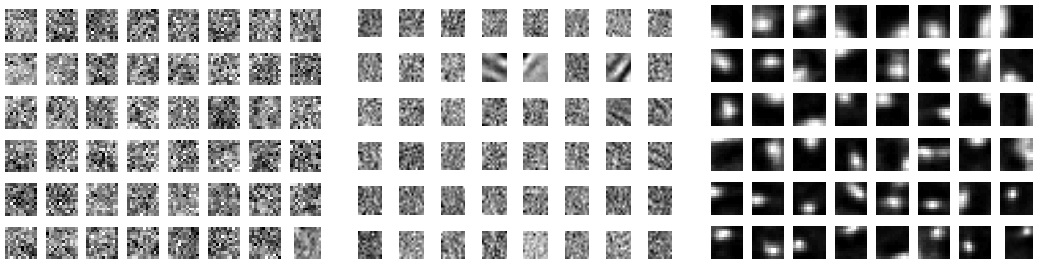

Figure 3: Filters found with (Left) L1 Olshausen&Field (Middle) ICA (Right) LDA when trained with *un-whitened* natural image data.