Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors

Christos Louizos, Max Welling

1 KL divergence between matrix variate Gaussian prior and posterior

Let $\mathcal{MN}_0(\mathbf{M}_0, \mathbf{U}_0, \mathbf{V}_0)$ and $\mathcal{MN}_1(\mathbf{M}_1, \mathbf{U}_1, \mathbf{V}_1)$ be two matrix variate Gaussian distributions for random matrices of size $n \times p$. We can use the fact that the matrix variate Gaussian is a multivariate Gaussian if we flatten the matrix, i.e. $\mathcal{MN}_0(\mathbf{M}_0, \mathbf{U}_0, \mathbf{V}_0) = \mathcal{N}_0(\text{vec}(\mathbf{M}_0), \mathbf{V}_0 \otimes \mathbf{U}_0)$, and as a result use the KL-divergence between two multivariate Gaussians:

$$
\begin{aligned}
KL(\mathcal{N}_0 || \mathcal{N}_1) &= \frac{1}{2}\Bigg(\text{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \log\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\Bigg) \\
&= \frac{1}{2}\Bigg(\text{tr}\left((\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1}(\mathbf{V}_0 \otimes \mathbf{U}_0)\right) + \big(\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)\big)^T \\
&\quad \big(\mathbf{V}_1 \otimes \mathbf{U}_1\big)^{-1}\big(\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)\big) - np + \log\frac{|\mathbf{V}_1 \otimes \mathbf{U}_1|}{|\mathbf{V}_0 \otimes \mathbf{U}_0|}\Bigg)
\end{aligned}
$$

Now to compute each term in the KL efficiently we need to use some properties of the vectorization and Kronecker product:

$$
\begin{aligned}
t_a &= \text{tr}\left((\mathbf{V}_1 \otimes \mathbf{U}_1)^{-1}(\mathbf{V}_0 \otimes \mathbf{U}_0)\right) \\
&= \text{tr}\left((\mathbf{V}_1^{-1} \otimes \mathbf{U}_1^{-1})(\mathbf{V}_0 \otimes \mathbf{U}_0)\right) \\
&= \text{tr}\left((\mathbf{V}_1^{-1}\mathbf{V}_0) \otimes (\mathbf{U}_1^{-1}\mathbf{U}_0)\right) \\
&= \text{tr}(\mathbf{U}_1^{-1}\mathbf{U}_0)\,\text{tr}(\mathbf{V}_1^{-1}\mathbf{V}_0)
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
t_b &= \big(\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)\big)^T \big(\mathbf{V}_1 \otimes \mathbf{U}_1\big)^{-1}\big(\text{vec}(\mathbf{M}_1) - \text{vec}(\mathbf{M}_0)\big) \\
&= \text{vec}(\mathbf{M}_1 - \mathbf{M}_0)^T(\mathbf{V}_1^{-1} \otimes \mathbf{U}_1^{-1})\,\text{vec}(\mathbf{M}_1 - \mathbf{M}_0) \\
&= \text{vec}(\mathbf{M}_1 - \mathbf{M}_0)^T\,\text{vec}(\mathbf{U}_1^{-1}(\mathbf{M}_1 - \mathbf{M}_0)\mathbf{V}_1^{-1}) \\
&= \text{tr}\left((\mathbf{M}_1 - \mathbf{M}_0)^T\mathbf{U}_1^{-1}(\mathbf{M}_1 - \mathbf{M}_0)\mathbf{V}_1^{-1}\right)
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
t_c &= \log\frac{|\mathbf{V}_1 \otimes \mathbf{U}_1|}{|\mathbf{V}_0 \otimes \mathbf{U}_0|} \\
&= \log\frac{|\mathbf{U}_1|^p|\mathbf{V}_1|^n}{|\mathbf{U}_0|^p|\mathbf{V}_0|^n} \\
&= p\log|\mathbf{U}_1| + n\log|\mathbf{V}_1| - \\
&\quad - p\log|\mathbf{U}_0| - n\log|\mathbf{V}_0|
\end{aligned} \tag{3}
$$

So putting everything together we have that:

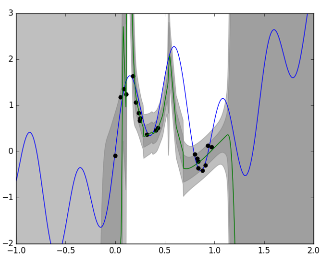
$$KL(\mathcal{MN}_0, \mathcal{MN}_1) = \frac{1}{2}\Big(\operatorname{tr}(\mathbf{U}_1^{-1}\mathbf{U}_0) \operatorname{tr}(\mathbf{V}_1^{-1}\mathbf{V}_0) + \operatorname{tr}\left((\mathbf{M}_1 - \mathbf{M}_0)^T \mathbf{U}_1^{-1}(\mathbf{M}_1 - \mathbf{M}_0)\mathbf{V}_1^{-1}\right) - \\ - np + p \log|\mathbf{U}_1| + n \log|\mathbf{V}_1| - p \log|\mathbf{U}_0| - n \log|\mathbf{V}_0| \Big) \tag{4}$$

2 Different toy dataset

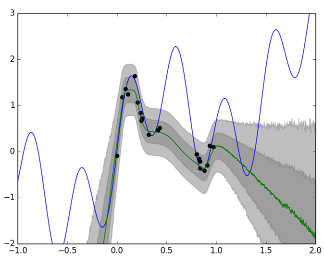
We also performed an experiment with a different toy dataset that was employed in [2]. We generated 12 inputs from $U[0, 0.6]$ and 8 inputs from $U[0.8, 1]$. We then transform those inputs via:

$$y_i = x_i + \epsilon_i + \sin(4(x_i + \epsilon_i)) + \sin(13(x_i + \epsilon_i))$$

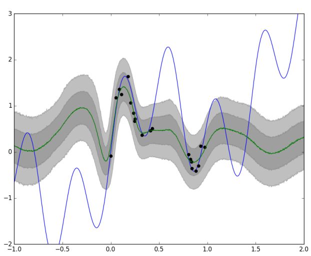
where $\epsilon_i \sim \mathcal{N}(0, 0.0009)$. We continued in fitting four neural networks that had two hidden-layers with 50 units each. The first was trained with probabilistic back-propagation [1], and the remaining three with our model while varying the nonlinearities among the layers: we used ReLU, cosine and hyperbolic tangent activations. For our model we set the upper bound of the variational dropout rate to 0.2 and we used 2 pseudo data pairs for the input layer and 4 for the rest. The resulting predictive distributions can be seen at Figure 1.



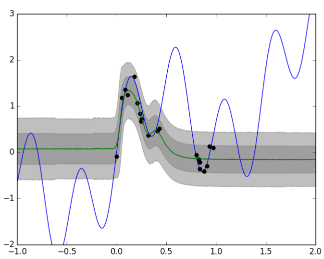
(a) PBP



(b) MG ReLU



(c) MG cosine



(d) MG tanh

Figure 1: Predictive distributions for the toy dataset. Grey areas correspond to $\pm\{1, 2\}$ standard deviations around the mean function.

References

[1] José Miguel Hernández-Lobato and Ryan Adams, *Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks*, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015.

[2] Ian Osband, Charles Blundell, Alexander Pritzel, Benjamin Van Roy, *Deep Exploration via Bootstrapped DQN*, arXiv preprint arXiv:1602.04621, 2016.