Bayesian Optimal Design - Weekly Report

Rasmus Hag Løvstad, pgq596

May 18, 2023

1 Week 3

1.1 Objective

The objective of this week is to learn about and implement variational inference for Bayesian linear regression. Then we will study convergence rates and accuracy by playing around with different parameters.

1.2 Theory

The point of variational inference is to approximate the posterior distribution $p(\theta|\mathbf{y}, \mathbf{d})$ using optimization techniques. The space of which to optimize in is the parameter space for some distribution $q(\theta)$, which we aim to make as close to the posterior as possible by KL-divergence. For reference, we will recite the definition of our linear model:

$$\theta \sim \mathcal{N}(\mu, \mathbf{\Sigma})$$

$$\epsilon \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2)$$

$$\mathbf{y}|\mathbf{d}, \theta \sim \theta^T \mathbf{d} + \epsilon$$

The optimization problem we wish to solve is defined like so:

$$q^*(\theta) = \arg\min_{q(\theta)\in\mathcal{L}} \mathrm{KL}(q(\theta)||p(\theta|\mathbf{y}, \mathbf{d}))$$

Computing the KL-divergence is hard though, because it requires computing the evidence $p(\mathbf{y}, \mathbf{d})$, so instead we try to optimize in regards to the expectation lower bound (ELBO):

TODO: show why

$$\mathrm{ELBO}(q) = \mathbb{E}_{\theta'}[\log p(\theta', \mathbf{d}, \mathbf{y})] - \mathbb{E}_{\theta'}[\log q(\theta')]$$

Let us pick $q(\theta')$ from the family of multivariate normal distributions. This is reasonable, since we expect the posterior to be normally distributed due to conjugacy between the prior and likelihood.
It is possible to rewrite the ELBO into a shape that is easier to optimize:

$$\mathrm{ELBO}(q) = \mathbb{E}_{\theta'}[\log p(\theta', \mathbf{d}, \mathbf{y})] - \mathbb{E}_{\theta'}[\log q(\theta')]$$

$$= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d}) p(\theta'|\mathbf{d}) p(\mathbf{d})] - \mathbb{E}_{\theta'}[\log q(\theta')]$$

$$
\begin{aligned}
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d}) + \log p(\theta'|\mathbf{d}) + \log p(\mathbf{d})] - \mathbb{E}_{\theta'}[\log q(\theta')] \\
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d})] + \mathbb{E}_{\theta'}[\log p(\theta'|\mathbf{d})] + \mathbb{E}_{\theta'}[\log p(\mathbf{d})] - \mathbb{E}_{\theta'}[\log q(\theta')] \\
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d})] + \mathbb{E}_{\theta'}[\log p(\theta')] - \mathbb{E}_{\theta'}[\log q(\theta')] + \text{const} \\
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d})] - (-\mathbb{E}_{\theta'}[\log p(\theta')] + \mathbb{E}_{\theta'}[\log q(\theta')]) + \text{const} \\
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d})] - \mathbb{E}_{\theta'}[\log \frac{q(\theta')}{p(\theta')}] + \text{const} \\
&= \mathbb{E}_{\theta'}[\log p(\mathbf{y}|\theta', \mathbf{d})] - \text{KL}(q||p) + \text{const}
\end{aligned}
$$

Since q and p are multivariate normal by assumption, we can use the closed-form solution for $\text{KL}(q||p)$. Thus, we only need to worry about the first part of the expression, which is an integral over the likelihood.
Now, we need to use the assumption about the multivariate normal distribution, that the covariance matrix must satisfy $\Sigma = AA^T$ for some matrix A . This means that

$$\theta \sim \mathcal{N}(\mu_\theta, A_\theta A_\theta^T)$$
$$\theta = \mu_\theta + A_\theta z$$

TODO: add this assumption to week 1

where

$$z \sim \mathcal{N}(0, 1)$$

TODO: why? (see eq 25 from lecture notes)

Thus we can rewrite our ELBO function as

$$\text{ELBO}(q) = \mathbb{E}_z[\log p(\mathbf{y}|\mu_\theta + A_\theta z, \mathbf{d})] - \text{KL}(q||p) + \text{const}$$

thus the log-likelihood is independent of θ'.
Now consider the first expectation:

$$
\begin{aligned}
&\mathbb{E}_z[\log p(\mathbf{y}|\mu_\theta + A_\theta z, \mathbf{d})] \\
&= \int_z p(z) \log p(\mathbf{y}|\mu_\theta + A_\theta z, \mathbf{d}) \; dz \\
&\approx \frac{1}{N} \sum_{i=1}^{N} p(z^{(i)}) \log p(\mathbf{y}|\mu_\theta + A_\theta z^{(i)}, \mathbf{d})
\end{aligned}
$$

where $N \in \mathbb{Z}$ and $z_i \sim \mathcal{N}(0, 1)$.

1.3 Design

The implementation is heavily utilizes the `autograd` library, which automatically calculates the gradient of the ELBO. To do this, it was nescessary to manually implement the PDF for the likelihood, since the one provided by Autograd doesn't support optimizing over the mean of the distribution. Several measures were taken to ensure numerical stability in this implementation including using the log-sum-exp trick as well as using the logarithm of the determinant instead of the actual determinant. It is also nescessary to make sure that enough samples are drawn of $\mathbf{z}$ to ensure our expectation over the log-likelihood is accurate enough. 100 samples were used in this implementation. Another issue that arises is that sometime the log-likelihood becomes a very small negative number (often denoted as $-\infty$

by Numpy). This happens when a sample z is picked such that the log-likelihood $p(\mathbf{y}|\mu_\theta + A_\theta z^(i), \mathbf{d})$ becomes very close to zero. This was solved by simply rejecting any sample that results in such a log-likelihood.

Another problem that arises is overflows when the amount of datapoints become larger. To handle this, it first of all makes sense to work directly with the log of the likelihood instead of first calculating the likelihood and then taking the log of it.

$$-\frac{1}{2}(n\log(2\pi) + \log(\det(\Sigma)) + (x-\mu)^T\Sigma^{-1}(x-\mu))$$

This also means that we don't need to apply the log-sum-exp trick, since we are working directly with logarithms. A disadvantage with this method is first of all that we can't guarantee that the covariance has an inverse, and secondly that computing the inverse of large matrices is very inefficient. To solve the first issue, we can try regularize the covariance matrix by adding a small constant to the diagonal. The second could potentially be solved by using some sort of approximation like Cholesky decomposition, but this was not regarded for now. The stop condition for the optimization is decided by the default gradient tolerance of the scipy-implementation, but in practice the optimization is stopped when the precision of the gradient is lost due to the stocastic nature of the objective function. This typically happens when trying to take small steps around the minimum, but as can be seen in Figure ... the precision is increased when the amount of datapoints is increased.

TODO: Why is log-pdf true? prove in theory

1.4 Results

TODO: Make subplots

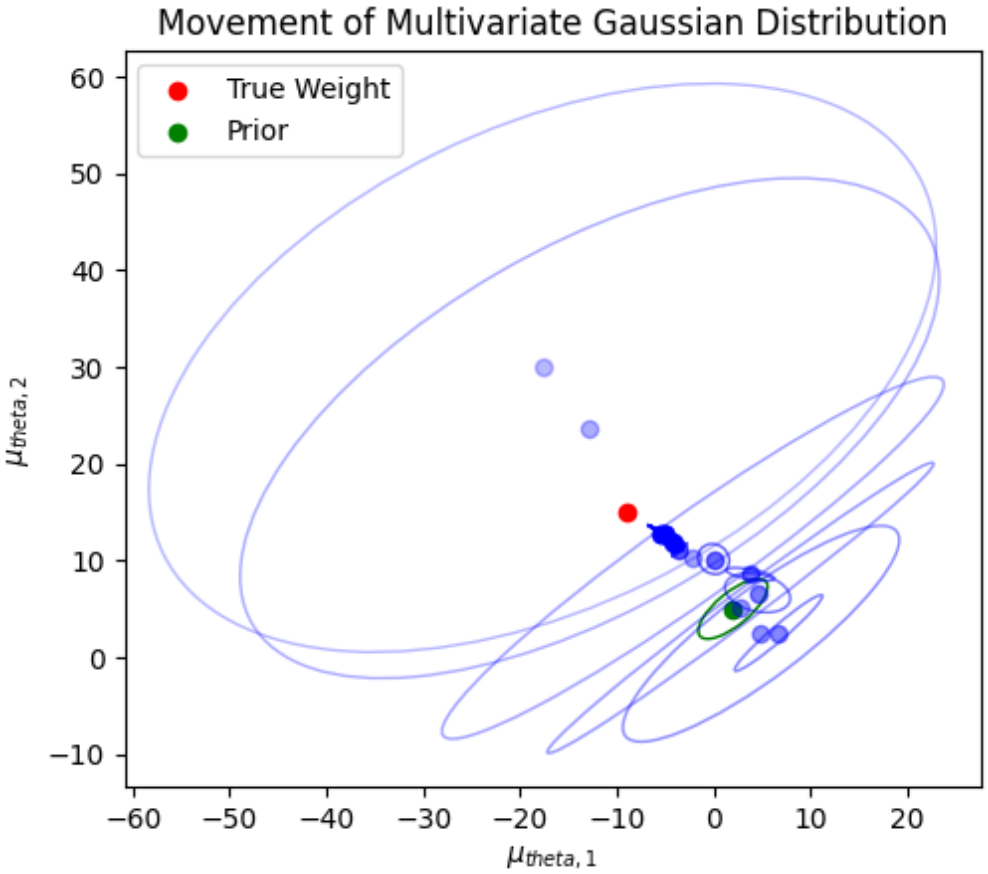

Figure 1: Optimization over the ELBO for 50 datapoints

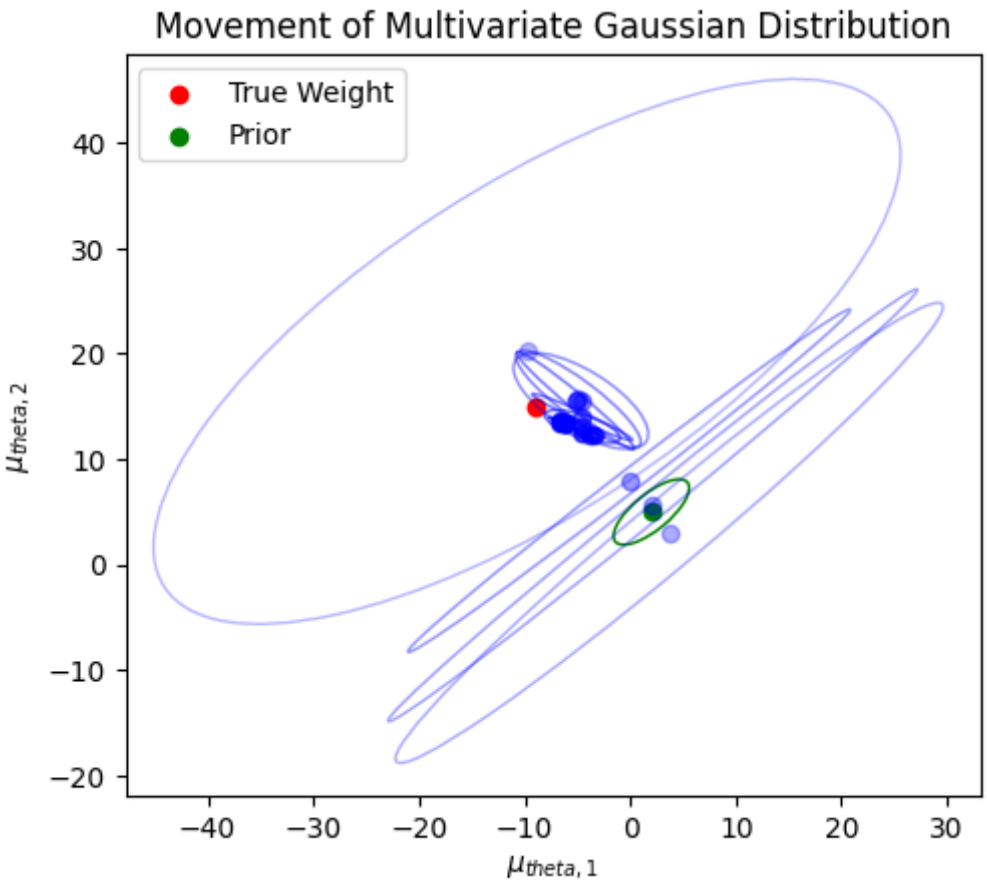

Figure 2: Optimization over the ELBO for 100 datapoints

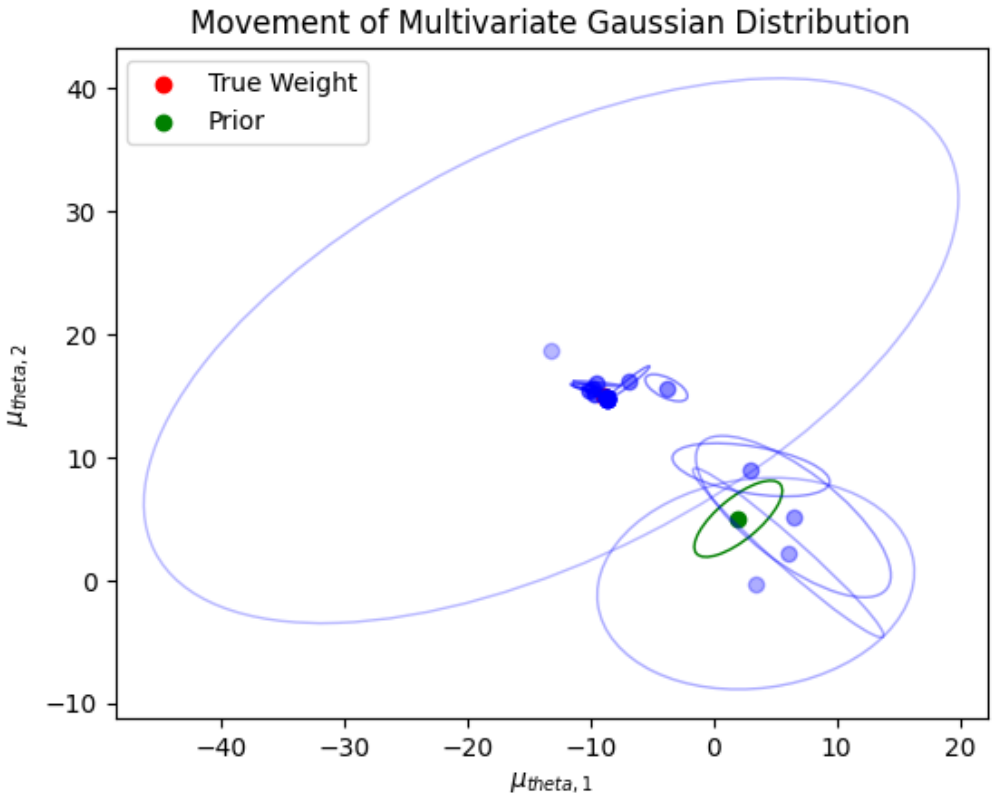

Figure 3: Optimization over the ELBO for 300 datapoints

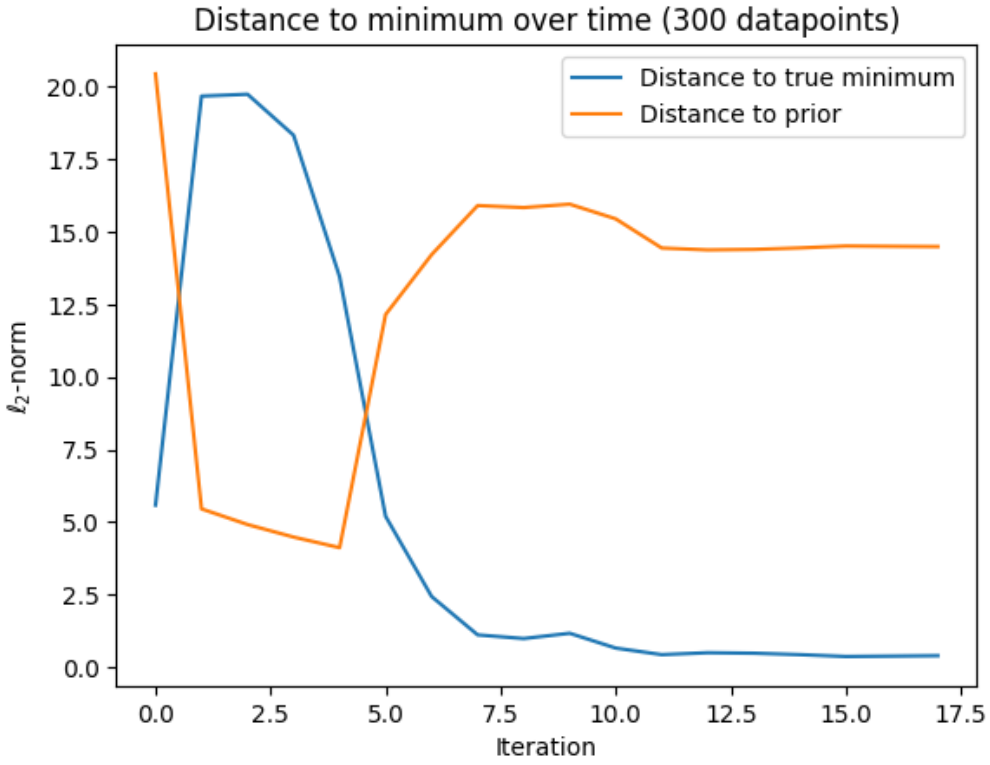

Figure 4: Optimization over the ELBO for 300 datapoints

As one can see, the optimization converges quite close to the true weights when the amount of datapoints increases as expected. First, the algorithm quickly moves towards the prior, and then slowly moves towards the true minimum. If the prior is close to the true minimum, one could expect that the algorithm would converge faster. It is also seen that the covariance of the parameters quickly become very small.

TODO: Interpret this - what does that mean?

1.5 Evaluation

The algorithm presented here seems to work - although it is not as efficient as the analytical solution, it represents a framework that could be more useful for other problems. From this we've also learned a bit about how we can expect the movement of the distribution - first towards the prior, then towards the minimum. It has also shown the importance of numerical stability as well as how we can use the reparameterization trick for efficient computation.