Bayesian Optimal Design - Weekly Report

Rasmus Hag Løvstad, pgq596

May 3, 2023

1 Week 2

1.1 Objective

The objective of this week is to implement and experiment with Bayesian Optimal Design for Bayesian linear regression as implemented in Week 1.

1.2 Theory

The Bayesian Optimal Design problem is about finding the a design $\mathbf{d}$ from a design space $\mathbf{D}$, that optimizes some kind of utility function. For this project, that utility function is the mutual information gained from an experiment by measuring at "location" $\mathbf{d}$.
Mutual information is in this case defined as

$$I(\mathbf{d}) = \int_{\Theta} \int_{\mathbf{Y}} p(\theta, \mathbf{y}|\mathbf{d})[\log p(\theta, \mathbf{y}|\mathbf{d}) - \log p(\mathbf{y}|\mathbf{d}) - \log p(\theta)] d\mathbf{y} d\theta$$

From last week, we assume that $p(\theta)$ (the prior in our linear model) is normally distributed with mean μ_θ and covariance Σ_θ. We can condition $p(\theta, \mathbf{y}|\mathbf{d})$ on θ and get

$$p(\theta, \mathbf{y}|\mathbf{d}) = p(\theta|\mathbf{y}, \mathbf{d}) p(\mathbf{y}|\mathbf{d})$$

such that

$$
\begin{aligned}
I(\mathbf{d}) &= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d})[\log p(\theta|\mathbf{y}, \mathbf{d}) + \log p(\mathbf{y}|\mathbf{d}) - \log p(\mathbf{y}|\mathbf{d}) - \log p(\theta)] d\mathbf{y} d\theta \\
&= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d})[\log p(\theta|\mathbf{y}, \mathbf{d}) - \log p(\theta)] d\mathbf{y} d\theta \\
&\text{Split integral over } \mathbf{Y} \\
&= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\mathbf{y} - \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta) d\mathbf{y} d\theta \\
&\text{Split joint distribution into conditional and marginal} \\
&= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\mathbf{y} - p(\theta) \log p(\theta) \int_\mathbf{Y} p(\mathbf{y}|\theta, \mathbf{d}) d\mathbf{y} d\theta \\
&\text{Use that probability distributions integrate to 1} \\
&= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\mathbf{y} - p(\theta) \log p(\theta) d\theta \\
&\text{Split integral over } \Theta \\
&= \int_\Theta \int_\mathbf{Y} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\mathbf{y} d\theta - \int_\Theta p(\theta) \log p(\theta) d\theta \\
&\text{Switch inner integral} \\
&= \int_\mathbf{Y} \int_\mathbf{\Theta} p(\theta, \mathbf{y}|\mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\theta d\mathbf{y} - \int_\Theta p(\theta) \log p(\theta) d\theta \\
&\text{Split joint distribution into conditional and marginal} \\
&= \int_\mathbf{Y} p(\mathbf{y}) \int_\Theta p(\theta|\mathbf{y}, \mathbf{d}) \log p(\theta|\mathbf{y}, \mathbf{d}) d\theta d\mathbf{y} - \int_\Theta p(\theta) \log p(\theta) d\theta \\
&\text{Use known integral of entropy (defined as } H(f) = \textstyle\int_x f(x) \log f(x) dx) \\
&\text{for multivariate Gaussian distributions, as well as the fact that} \\
&\text{the prior and posterior are multivariate Gaussian.} \\
&= \int_\mathbf{Y} p(\mathbf{y}) \frac{1}{2} \ln \det(2\pi e \Sigma_{\theta|\mathbf{y},\mathbf{d}}) d\mathbf{y} - \frac{1}{2} \ln \det(2\pi e \Sigma_\theta) \\
&\text{Use that } \Sigma_{\theta|\mathbf{y},\mathbf{d}} \text{ does not depend on } \mathbf{y} \text{ (from last week)} \\
&\text{and that probability distributions integrate to 1} \\
&= \frac{1}{2} \ln \det(2\pi e \Sigma_{\theta|\mathbf{y},\mathbf{d}}) - \frac{1}{2} \ln \det(2\pi e \Sigma_\theta)
\end{aligned}
$$

To find the optimal design, we want to find a maximizer $\mathbf{d}^*$ defined as such:

$$\mathbf{d}^* = \arg \max_{\mathbf{d} \in \mathbf{D}} I(\mathbf{d})$$

This can be done using an optimization approach, like a gradient-based line-search.

1.3 Design

The implementation is pretty straight-forward this week, as we only need to implement $I(\mathbf{d})$ and plug it in to any out-of-the-box optimizer, such as

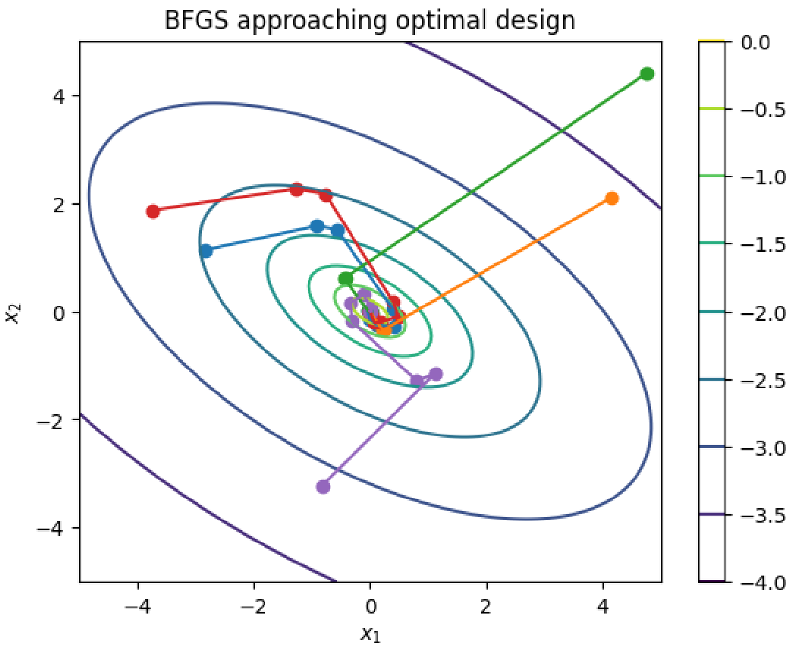

Figure 1: BFGS algorithm optimizing for the most mutual information

`scipy.optimize.minimize(method="BFGS")`.
If one implements the optimization using `autograd.np`, the gradient can also be calculated easily, leading to a more efficient optimization approach.

1.4 Results

Running the BFGS algorithm 5 times with random start points over $I(\mathbf{d})$ with $\Sigma_\theta = \begin{bmatrix} 7.9 & 3 \\ 4 & 5 \end{bmatrix}$ gives the plot seen in Figure 1. The algorithm usually converges after about 10 steps or so.

1.5 Evaluation

As it can be seen in Figure 1, the algorithm has no problem finding the optimal design for maximizing the mutual information between the prior and the posterior.