Bayesian inference using Markov chain Monte Carlo methods can be notoriously slow. In this blog post, we reframe Bayesian inference as an optimization problem using variational inference, markedly speeding up computation. We derive the variational objective function, implement coordinate ascent mean-field variational inference for a simple linear regression example in R, and compare our results to results obtained via variational and exact inference using Stan. Sounds like word salad? Then let’s start unpacking!

**Preliminaries**

Bayes’ rule states that

where $\mathbf{z}$ denotes latent parameters we want to infer and $\mathbf{x}$ denotes data.[1](https://fabiandablander.com/r/Variational-Inference.html#fn:1) Bayes’ rule is, in general, difficult to apply because it requires dealing with a potentially high-dimensional integral — the marginal likelihood. Optimization, which involves taking derivatives instead of integrating, is much [easier](https://xkcd.com/2117/) and generally faster than the latter, and so our goal will be to reframe this integration problem as one of optimization.

**Variational objective**

We want to get at the posterior distribution, but instead of sampling we simply try to find a density $q^\star(\mathbf{z})$ from a family of densities $\mathrm{Q}$ that best approximates the posterior distribution:

where $\text{KL}(. \lvert \lvert.)$ denotes the [Kullback-Leibler divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence):

We cannot compute this Kullback-Leibler divergence because it still depends on the nasty integral $p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z}) \, p(\mathbf{z}) \, \mathrm{d}\mathbf{z}$. To see this dependency, observe that:

where we have expanded the expectation to more clearly behold our nemesis. In doing so, we have seen that $\text{log } p(\mathbf{x})$ is actually a constant with respect to $q(\mathbf{z})$; this means that we can ignore it in our optimization problem. Moreover, minimizing a quantity means maximizing its negative, and so we maximize the following quantity:

We can expand the joint probability to get more insight into this equation:

This is cool. It says that maximizing the ELBO finds an approximate distribution $q(\mathbf{z})$ for latent quantities $\mathbf{z}$ that allows the data to be predicted well, i.e., leads to a high expected log likelihood, but that a penalty is incurred if $q(\mathbf{z})$ strays far away from the prior $p(\mathbf{z})$. This mirrors the usual balance in Bayesian inference between likelihood and prior (Blei, Kucukelbier, & McAuliffe, 2017).

ELBO stands for *evidence lower bound*. The marginal likelihood is sometimes called evidence, and we see that ELBO is indeed a lower bound for the evidence:

since the Kullback-Leibler divergence is non-negative. Heuristically, one might then use the ELBO as a way to select between models. For more on predictive model selection, see [this](https://fabiandablander.com/r/Law-of-Practice.html) and [this](https://fabiandablander.com/r/Bayes-Potter.html) blog post.

**Why variational?**

Our optimization problem is about finding $q^\star(\mathbf{z})$ that best approximates the posterior distribution. This is in contrast to more familiar optimization problems such as maximum likelihood estimation where one wants to find, for example, the *single best value* that maximizes the log likelihood. For such a problem, one can use standard calculus (see for example [this](https://fabiandablander.com/r/Curve-Fitting-Gaussian.html) blog post). In our setting, we do not want to find a single best value but rather a *single best function*. To do this, we can use *variational calculus* from which variational inference derives its name (Bishop, 2006, p. 462).

A function takes an input value and returns an output value. We can define a *functional* which takes a whole function and returns an output value. The *entropy* of a probability distribution is a widely used functional:

which takes as input the probability distribution $p(x)$ and returns a single value, its entropy. In variational inference, we want to find the function that minimizes the ELBO, which is a functional.

In order to make this optimization problem more manageable, we need to constrain the functions in some way. One could, for example, assume that $q(\mathbf{z})$ is a Gaussian distribution with parameter vector $\omega$. The ELBO then becomes a function of $\omega$, and we employ standard optimization methods to solve this problem. Instead of restricting the parametric form of the variational distribution $q(\mathbf{z})$, in the next section we use an independence assumption to manage the inference problem.

**Mean-field variational family**

A frequently used approximation is to assume that the latent variables $z\_j$ for $j = \{1, \ldots, m\}$ are mutually independent, each governed by their own variational density:

Note that this *mean-field variational family* cannot model correlations in the posterior distribution; by construction, the latent parameters are mutually independent. Observe that we do not make any parametric assumption about the individual $q\_j(z\_j)$. Instead, their parametric form is derived for every particular inference problem.

We start from our definition of the ELBO and apply the mean-field assumption:

In the following, we optimize the ELBO with respect to a single variational density $q\_j(z\_j)$ and assume that all others are fixed:

One could use variational calculus to derive the optimal variational density $q\_j^\star(z\_j)$; instead, we follow Bishop (2006, p. 465) and define the distribution

where we need to make sure that it integrates to one by subtracting the (log) normalizing constant $\mathcal{Z}$. With this in mind, observe that:

Thus, maximizing the ELBO with respect to $q\_j(z\_j)$ is minimizing the Kullback-leibler divergence between $q\_j(z\_j)$ and $\tilde{p}(\mathbf{x}, z\_j)$; it is zero when the two distributions are equal. Therefore, under the mean-field assumption, the optimal variational density $q\_j^\star(z\_j)$ is given by:

see also Bishop (2006, p. 466). This is not an explicit solution, however, since each optimal variational density depends on all others. This calls for an iterative solution in which we first initialize all factors $q\_j(z\_i)$ and then cycle through them, updating them conditional on the updates of the other. Such a procedure is known as *Coordinate Ascent Variational Inference* (CAVI). Further, note that

which allows us to write the updates in terms of the conditional posterior distribution of $z\_j$ given all other factors $\mathbf{z}\_{-j}$. This looks *a lot* like Gibbs sampling, which we discussed in detail in a [previous](https://fabiandablander.com/r/Spike-and-Slab.html) blog post. In the next section, we implement CAVI for a simple linear regression problem.

**Application: Linear regression**

In a [previous](https://fabiandablander.com/r/Curve-Fitting-Gaussian.html) blog post, we traced the history of least squares and applied it to the most basic problem: fitting a straight line to a number of points. Here, we study the same problem but swap optimization procedure: instead of least squares or maximum likelihood, we use variational inference. Our linear regression setup is:

where we assume that the population mean of $y$ is zero (i.e., $\beta\_0 = 0$); and we assign the error variance $\sigma^2$ an improper Jeffreys’ prior and $\beta$ a Gaussian prior with variance $\sigma^2\tau^2$. We scale the prior of $\beta$ by the error variance to reason in terms of a standardized effect size $\beta / \sigma$ since with this specification:

As a heads up, we have to do a surprising amount of calculations to implement variational inference even for this simple problem. In the next section, we start our journey by deriving the variational density for $\sigma^2$.

**Variational density for $\sigma^2$**

Our optimal variational density $q^\star(\sigma^2)$ is given by:

To get started, we need to derive the conditional posterior distribution $p(\sigma^2 \mid \mathbf{y}, \beta)$. We write:

which is proportional to an inverse Gamma distribution. Moving on, we exploit the linearity of the expectation and write:

This, too, looks like an inverse Gamma distribution! Plugging in the normalizing constant, we arrive at:

Note that this quantity depends on $\beta$. In the next section, we derive the variational density for $\beta$.

**Variational density for $\beta$**

Our optimal variational density $q^\star(\beta)$ is given by:

and so we again have to derive the conditional posterior distribution $p(\beta \mid \mathbf{y}, \sigma^2)$. We write:

where we have “completed the square” (see also [this](https://fabiandablander.com/statistics/Two-Properties.html) blog post) and realized that the conditional posterior is Gaussian. We continue by taking expectations:

which is again proportional to a Gaussian distribution! Plugging in the normalizing constant yields:

Note that while the variance of this distribution, $\sigma^2\_\beta$, depends on $q(\sigma^2)$, its mean $\mu\_\beta$ does not.

To recap, instead of assuming a parametric form for the variational densities, we have derived the optimal densities under the mean-field assumption, that is, under the assumption that the parameters are independent: $q(\beta, \sigma^2) = q(\beta) \, q(\sigma^2)$. Assigning $\beta$ a Gaussian distribution and $\sigma^2$ a Jeffreys’s prior, we have found that the variational density for $\sigma^2$ is an inverse Gamma distribution and that the variational density for $\beta$ a Gaussian distribution. We noted that these variational densities depend on each other. However, this is not the end of the manipulation of symbols; both distributions still feature an expectation we need to remove. In the next section, we expand the remaining expectations.

**Removing expectations**

Now that we know the parametric form of both variational densities, we can expand the terms that involve an expectation. In particular, for the variational density $q^\star(\sigma^2)$ we write:

Noting that $\mathbb{E}\_{q(\beta)}[\beta] = \mu\_{\beta}$ and using the fact that:

the expectation becomes:

For the expectation which features in the variational distribution for $\beta$, things are slightly less elaborate, although the result also looks unwieldy. Note that since $\sigma^2$ follows an inverse Gamma distribution, $1 / \sigma^2$ follows a Gamma distribution which has mean:

**Monitoring convergence**

The algorithm works by first specifying initial values for the parameters of the variational densities and then iteratively updating them until the ELBO does not change anymore. This requires us to compute the ELBO, which we still need to derive, on each update. We write:

Let’s take a deep breath and tackle the second term first:

Note that there are three expectations left. However, we really deserve a break, and so instead of analytically deriving the expectations we compute $\mathbb{E}\_{q(\sigma^2)}\left[\text{log } \sigma^2\right]$ and $\mathbb{E}\_{p(\sigma^2)}\left[\text{log } q(\sigma^2)\right]$ numerically using Gaussian quadrature. This fails for $\mathbb{E}\_{q(\sigma^2)}\left[\text{log } q(\sigma^2)\right]$, which we compute using Monte carlo integration:

We are left with the expected log likelihood. Instead of filling this blog post with more equations, we again resort to numerical methods. However, we refactor the expression so that numerical integration is more efficient:

Since we have solved a similar problem already above, we evaluate the expecation with respect to $q(\beta)$ analytically:

In the next section, we implement the algorithm for our linear regression problem in R.

**Implementation in R**

Now that we have derived the optimal densities, we know how they are parameterized. Therefore, the ELBO is a function of these variational parameters and the parameters of the priors, which in our case is just $\tau^2$. We write a function that computes the ELBO:

library('MCMCpack')

#' Computes the ELBO for the linear regression example

#'

#' @param y univariate outcome variable

#' @param x univariate predictor variable

#' @param beta\_mu mean of the variational density for \beta

#' @param beta\_sd standard deviation of the variational density for \beta

#' @param nu parameter of the variational density for \sigma^2

#' @param nr\_samples number of samples for the Monte carlo integration

#' @returns ELBO

compute\_elbo <- function(y, x, beta\_mu, beta\_sd, nu, tau2, nr\_samples = 1e4) {

n <- length(y)

sum\_y2 <- sum(y^2)

sum\_x2 <- sum(x^2)

sum\_yx <- sum(x\*y)

# Takes a function and computes its expectation with respect to q(\beta)

E\_q\_beta <- function(fn) {

integrate(function(beta) {

dnorm(beta, beta\_mu, beta\_sd) \* fn(beta)

}, -Inf, Inf)$value

}

# Takes a function and computes its expectation with respect to q(\sigma^2)

E\_q\_sigma2 <- function(fn) {

integrate(function(sigma) {

dinvgamma(sigma^2, (n + 1)/2, nu) \* fn(sigma)

}, 0, Inf)$value

}

# Compute expectations of log p(\sigma^2)

E\_log\_p\_sigma2 <- E\_q\_sigma2(function(sigma) log(1/sigma^2))

# Compute expectations of log p(\beta \mid \sigma^2)

E\_log\_p\_beta <- (

log(tau2 / beta\_sd^2) \* E\_q\_sigma2(function(sigma) log(sigma^2)) +

(beta\_sd^2 + tau2) / (tau2) \* E\_q\_sigma2(function(sigma) 1/sigma^2)

)

# Compute expectations of the log variational densities q(\beta)

E\_log\_q\_beta <- E\_q\_beta(function(beta) dnorm(beta, beta\_mu, beta\_sd, log = TRUE))

# E\_log\_q\_sigma2 <- E\_q\_sigma2(function(x) log(dinvgamma(x, (n + 1)/2, nu))) # fails

# Compute expectations of the log variational densities q(\sigma^2)

sigma2 <- rinvgamma(nr\_samples, (n + 1)/2, nu)

E\_log\_q\_sigma2 <- mean(log(dinvgamma(sigma2, (n + 1)/2, nu)))

# Compute the expected log likelihood

E\_log\_y\_b <- sum\_y2 - 2\*sum\_yx\*beta\_mu + (beta\_sd^2 + beta\_mu^2)\*sum\_x2

E\_log\_y\_sigma2 <- E\_q\_sigma2(function(sigma) log(sigma^2) \* 1/sigma^2)

E\_log\_y <- n/4 \* log(2\*pi) \* E\_log\_y\_b \* E\_log\_y\_sigma2

# Compute and return the ELBO

ELBO <- E\_log\_y + E\_log\_p\_beta + E\_log\_p\_sigma2 - E\_log\_q\_beta - E\_log\_q\_sigma2

ELBO

}

The function below implements coordinate ascent mean-field variational inference for our simple linear regression problem. Recall that the variational parameters are:

The following function implements the iterative updating of these variational parameters until the ELBO has converged.

#' Implements CAVI for the linear regression example

#'

#' @param y univariate outcome variable

#' @param x univariate predictor variable

#' @param tau2 prior variance for the standardized effect size

#' @returns parameters for the variational densities and ELBO

lmcavi <- function(y, x, tau2, nr\_samples = 1e5, epsilon = 1e-2) {

n <- length(y)

sum\_y2 <- sum(y^2)

sum\_x2 <- sum(x^2)

sum\_yx <- sum(x\*y)

# is not being updated through variational inference!

beta\_mu <- sum\_yx / (sum\_x2 + 1/tau2)

res <- list()

res[['nu']] <- 5

res[['beta\_mu']] <- beta\_mu

res[['beta\_sd']] <- 1

res[['ELBO']] <- 0

j <- 1

has\_converged <- function(x, y) abs(x - y) < epsilon

ELBO <- compute\_elbo(y, x, beta\_mu, 1, 5, tau2, nr\_samples = nr\_samples)

# while the ELBO has not converged

while (!has\_converged(res[['ELBO']][j], ELBO)) {

nu\_prev <- res[['nu']][j]

beta\_sd\_prev <- res[['beta\_sd']][j]

# used in the update of beta\_sd and nu

E\_qA <- sum\_y2 - 2\*sum\_yx\*beta\_mu + (beta\_sd\_prev^2 + beta\_mu^2)\*(sum\_x2 + 1/tau2)

# update the variational parameters for sigma2 and beta

nu <- 1/2 \* E\_qA

beta\_sd <- sqrt(((n + 1) / E\_qA) / (sum\_x2 + 1/tau2))

# update results object

res[['nu']] <- c(res[['nu']], nu)

res[['beta\_sd']] <- c(res[['beta\_sd']], beta\_sd)

res[['ELBO']] <- c(res[['ELBO']], ELBO)

# compute new ELBO

j <- j + 1

ELBO <- compute\_elbo(y, x, beta\_mu, beta\_sd, nu, tau2, nr\_samples = nr\_samples)

}

res

}

Let’s run this on a simulated data set of size $n = 100$ with a true coefficient of $\beta = 0.30$ and a true error variance of $\sigma^2 = 1$. We assign $\beta$ a Gaussian prior with variance $\tau^2 = 0.25$ so that values for $\lvert \beta \rvert$ larger than two standard deviations ($0.50$) [receive about $0.68$](https://fabiandablander.com/r/Variational-Inference.html) prior probability.

gen\_dat <- function(n, beta, sigma) {

x <- rnorm(n)

y <- 0 + beta\*x + rnorm(n, 0, sigma)

data.frame(x = x, y = y)

}

set.seed(1)

dat <- gen\_dat(100, 0.30, 1)

mc <- lmcavi(dat$y, dat$x, tau2 = 0.50^2)

mc

## $nu

## [1] 5.00000 88.17995 45.93875 46.20205 46.19892 46.19895

##

## $beta\_mu

## [1] 0.2800556

##

## $beta\_sd

## [1] 1.00000000 0.08205605 0.11368572 0.11336132 0.11336517 0.11336512

##

## $ELBO

## [1] 0.0000 -297980.0495 493.4807 -281.4578 -265.1289

## [6] -265.3197

From the output, we see that the ELBO and the variational parameters have converged. In the next section, we compare these results to results obtained with Stan.

**Comparison with Stan**

Whenever one goes down a rabbit hole of calculations, it is good to sanity check one’s results. Here, we use Stan’s variational inference scheme to check whether our results are comparable. It assumes a Gaussian variational density for each parameter after transforming them to the real line and automates inference in a “black-box” way so that no problem-specific calculations are required (see Kucukelbir, Ranganath, Gelman, & Blei, 2015). Subsequently, we compare our results to the exact posteriors arrived by Markov chain Monte carlo. The simple linear regression model in Stan is:

data {

int n;

vector[n] y;

vector[n] x;

real tau;

}

parameters {

real b;

real sigma;

}

model {

target += -log(sigma);

target += normal\_lpdf(b | 0, sigma\*tau);

target += normal\_lpdf(y | b\*x, sigma);

}

We use Stan’s black-box variational inference scheme:

library('rstan')

# save the above model to a file and compile it

# model <- stan\_model(file = 'regression.stan')

stan\_dat <- list('n' = nrow(dat), 'x' = dat$x, 'y' = dat$y, 'tau' = 0.50)

fit <- vb(

model, data = stan\_dat, output\_samples = 20000, adapt\_iter = 10000,

init = list('b' = 0.30, 'sigma' = 1), refresh = FALSE, seed = 1

)

This gives similar estimates as ours:

fit

## Inference for Stan model: variational-regression.

## 1 chains, each with iter=20000; warmup=0; thin=1;

## post-warmup draws per chain=20000, total post-warmup draws=20000.

##

## mean sd 2.5% 25% 50% 75% 97.5%

## b 0.28 0.13 0.02 0.19 0.28 0.37 0.54

## sigma 0.99 0.09 0.82 0.92 0.99 1.05 1.18

## lp\_\_ 0.00 0.00 0.00 0.00 0.00 0.00 0.00

##

## Approximate samples were drawn using VB(meanfield) at Wed Oct 30 13:20:01 2019.

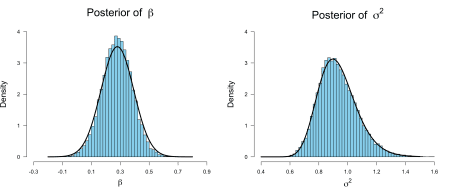
## We recommend genuine 'sampling' from the posterior distribution for final inferences!

Their recommendation is prudent. If you run the code with different seeds, you can get quite different results. For example, the posterior mean of $\beta$ can range from $0.12$ to $0.45$, and the posterior standard deviation can be as low as $0.03$; in all these settings, Stan indicates that the ELBO has converged, but it seems that it has converged to a different local optimum for each run. (For seed = 3, Stan gives completely nonsensical results). Stan warns that the algorithm is experimental and may be unstable, and it is probably wise to not use it in production.

Although the posterior distribution for $\beta$ and $\sigma^2$ is available in closed-form (see the *Post Scriptum*), we check our results against exact inference using Markov chain Monte carlo by visual inspection.

fit <- sampling(model, data = stan\_dat, iter = 8000, refresh = FALSE, seed = 1)

The Figure below overlays our closed-form results to the histogram of posterior samples obtained using Stan.



Note that the posterior variance of $\beta$ is slightly *overestimated* when using our variational scheme. This is in contrast to the fact that variational inference generally *underestimates* variances. Note also that Bayesian inference using Markov chain Monte Carlo is very fast on this simple problem. However, the comparative advantage of variational inference becomes clear by increasing the sample size: for sample sizes as large as $n = 100000$, our variational inference scheme takes less then a tenth of a second!

**Conclusion**

In this blog post, we have seen how to turn an integration problem into an optimization problem using variational inference. Assuming that the variational densities are independent, we have derived the optimal variational densities for a simple linear regression problem with one predictor. While using variational inference for this problem is unnecessary since everything is available in closed-form, I have focused on such a simple problem so as to not confound this introduction to variational inference by the complexity of the model. Still, the derivations were quite lengthy. They were also entirely specific to our particular problem, and thus generic “black-box” algorithms which avoid problem-specific calculations hold great promise.

We also implemented coordinate ascent mean-field variational inference (CAVI) in R and compared our results to results obtained via variational and exact inference using Stan. We have found that one probably should not trust Stan’s variational inference implementation, and that our results closely correspond to the exact procedure. For more on variational inference,