Homework 6

Fred Boehm

12/2/2020

Part A.i

Compute the covariance matrix of Z

$$Var(Z_1) = \frac{1}{2}Var(X_1 + X_2) = \frac{1}{2}\left(\sigma_1^2 + \sigma_2^2\right)$$
$$Var(Z_2) = \frac{1}{2}Var(X_1 - X_2) = \frac{1}{2}\left(\sigma_1^2 + \sigma_2^2\right)$$

$$Cov(Z_1, Z_2) = \mathbb{E}(Z_1 Z_2) - \mathbb{E}Z_1 \mathbb{E}Z_2 = \mathbb{E}(Z_1 Z_2)$$

and

$$\mathbb{E}(Z_1 Z_2) = \frac{1}{2}\mathbb{E}\left(X_1^2 - X_2^2\right) = \frac{1}{2}\left(\sigma_1^2 - \sigma_2^2\right)$$

Thus,

$$\Sigma = \frac{1}{2}\begin{pmatrix} \sigma_1^2 + \sigma_2^2 & \sigma_1^2 - \sigma_2^2 \\ \sigma_1^2 - \sigma_2^2 & \sigma_1^2 + \sigma_2^2 \end{pmatrix}$$

Compute Λ

The precision matrix is:

$$\Lambda = \frac{1}{2\sigma_1^2\sigma_2^2}\begin{pmatrix} \sigma_1^2 + \sigma_2^2 & \sigma_2^2 - \sigma_1^2 \\ \sigma_2^2 - \sigma_1^2 & \sigma_1^2 + \sigma_2^2 \end{pmatrix}$$

Compute eigenvalues and eigenvectors of Σ

$$\begin{aligned} 0 &= det\begin{pmatrix} \frac{1}{2} & a+b-\lambda & b-a \\ & b-a & a+b-\lambda \end{pmatrix} \\ &= (\frac{a+b}{2} - \lambda)^2 - (\frac{b-a}{2})^2 \\ &= ab + \lambda^2 - \lambda(a+b) = (a-\lambda)(b-\lambda) \end{aligned}$$

Thus, the eigenvalues of Σ are σ_1^2 and σ_2^2.

So we now have the two eigenvalues, σ_1^2 and σ_2^2

We then seek the corresponding eigenvectors.

We see that these are $(1, 1)$ and $(1, -1)$.

Level curves of the pdf

```
s1 <- 1
s2 <- 10
```

```
x1 <- rnorm(n = 10000, mean = 0, sd = sqrt(s1))
x2 <- rnorm(n = 10000, mean = 0, sd = sqrt(s2))
z1 <- (x1 + x2) / sqrt(2)
z2 <- (x1 - x2) / sqrt(2)
```

```
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
s <- subplot(
  plot_ly(x = z1, type = "histogram"),
  plotly_empty(),
  plot_ly(x = z1, y = z2, type = "histogram2dcontour"),
  plot_ly(y = z2, type = "histogram"),
  nrows = 2, heights = c(0.2, 0.8), widths = c(0.8, 0.2), margin = 0,
  shareX = TRUE, shareY = TRUE, titleX = FALSE, titleY = FALSE
)
```

```
## Warning: `arrange_()` is deprecated as of dplyr 0.7.0.
## Please use `arrange()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

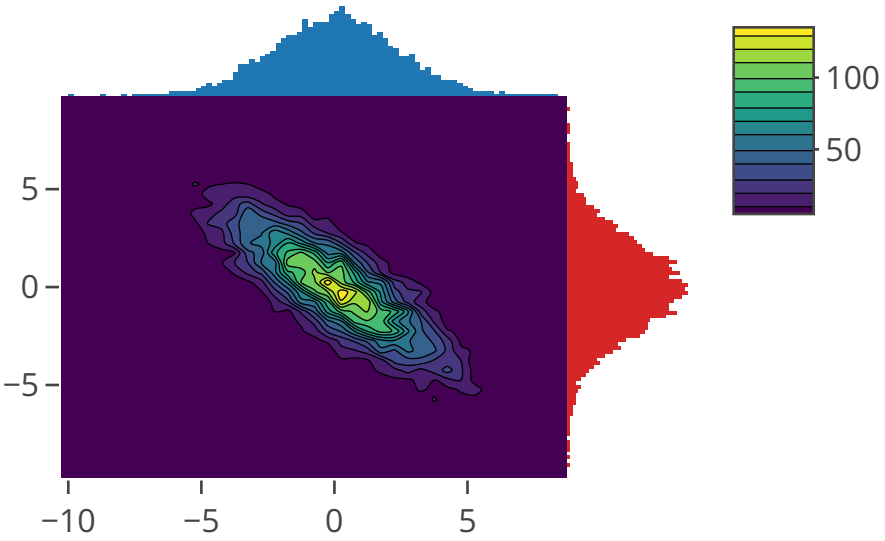
```
## Warning: No trace type specified and no positional attributes specified
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

```
fig <- layout(s, showlegend = FALSE)

fig
```



Part A.ii

Find the minimizer

We follow the steps in Example 2.41 to see that the minimizer is

$$q(z) = q_1(z_1)q_2(z_2)$$

With q_i being the density for a $N(0, \Lambda_{ii}^{-1})$ distribution.

Level curves

Setting $\sigma_1^2 = 1$ and $\sigma_2^2 = 10$, we get

$$\Lambda = \begin{pmatrix} 5.5 & 4.5 \\ 4.5 & 5.5 \end{pmatrix}$$

```
x <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
y <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
```

```
s <- subplot(
  plot_ly(x = x, type = "histogram"),
  plotly_empty(),
  plot_ly(x = x, y = y, type = "histogram2dcontour"),
  plot_ly(y = y, type = "histogram"),
  nrows = 2, heights = c(0.2, 0.8), widths = c(0.8, 0.2), margin = 0,
  shareX = TRUE, shareY = TRUE, titleX = FALSE, titleY = FALSE
)
```

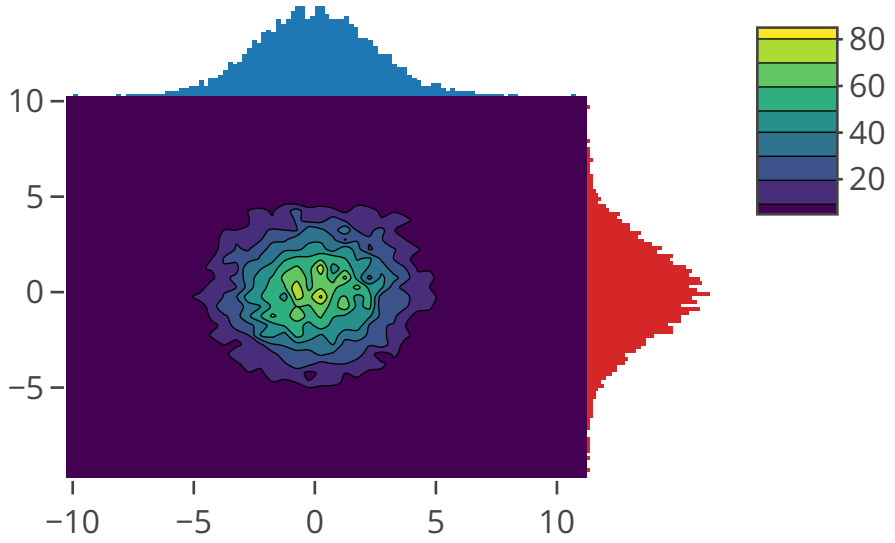
```
## Warning: No trace type specified and no positional attributes specified
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

```
fig <- layout(s, showlegend = FALSE)

fig
```



Part A.iii

Find the minimizer

Section 2.5.3 tells us that the minimizer is the product of the marginals of $\mathbb{P}$.

```
x <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
y <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
```

Level curves

```
x <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
y <- rnorm(n = 10000, mean = 0, sd = sqrt(11 / 2))
```

```
s <- subplot(
  plot_ly(x = x, type = "histogram"),
  plotly_empty(),
  plot_ly(x = x, y = y, type = "histogram2dcontour"),
  plot_ly(y = y, type = "histogram"),
  nrows = 2, heights = c(0.2, 0.8), widths = c(0.8, 0.2), margin = 0,
  shareX = TRUE, shareY = TRUE, titleX = FALSE, titleY = FALSE
)
```

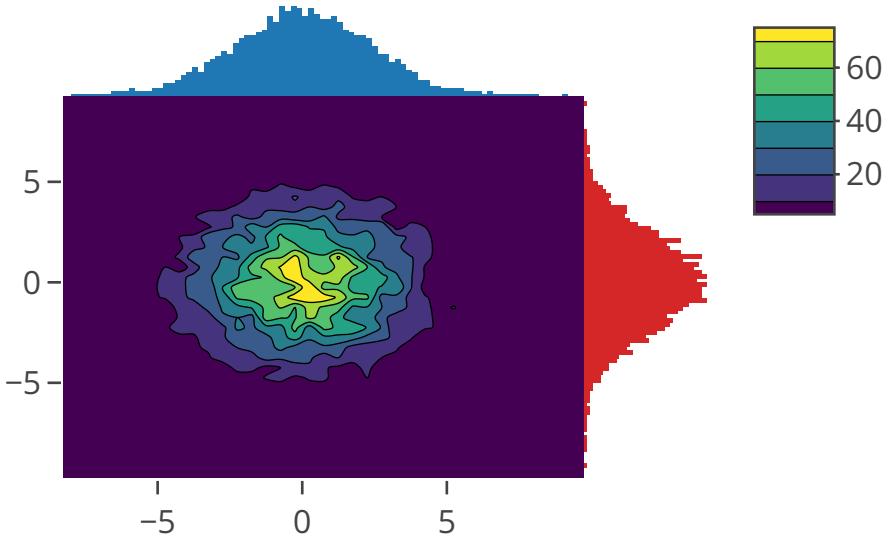
```
## Warning: No trace type specified and no positional attributes specified
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

```
fig <- layout(s, showlegend = FALSE)

fig
```



Part B: A summary of the two articles

Bayesian inference for complex, multilevel statistical models historically involved sampling-based Markov chain Monte Carlo (MCMC) methods. In this approach, one constructs a Markov chain for which the stationary distribution is the posterior distribution for the unknowns in the statistical model. Empirically summaries of the posterior distribution are computed from thousands of samples.

Over the last 20 years, researchers in machine learning and statistics have developed variational inference methods for study of posterior distributions in complicated Bayesian models. In variational inference, one specifies a simpler family of distributions that approximate the posterior. One then chooses a distribution from the from the approximation family. The chosen distribution minimizes (over the approximation family members) the KL divergence with the posterior.

A major advantage of variational inference over sampling-based methods is the lesser computing time for variational inference. Despite this advantage, variational inference was plagued by the need for model-specific derivations. Ranganath, Gerrish, and Blei (2014) recognized this shortcoming and developed black box variational inference to address this issue. Their work enables rapid exploration of diverse collections of models.

Black box variational inference uses stochastic optimization by calculating noisy gradients of the evidence lower bound (ELBO). Ranganath, Gerrish, and Blei (2014) do this computation with Monte Carlo samples from the score function (Equation 3 in Ranganath, Gerrish, and Blei (2014)). They incorporate Rao-Blackwellization and control variates to reduce the noisiness of the gradients.

Ranganath, Gerrish, and Blei (2014) achieve impressive results in evaluating their black box variational inference methods. Figure 1 of their article illustrates the acceleration in computing time. Figure 2 shows the remarkable gradient variance reduction achieved with the Rao=-Blackwellization and control variates.

Kucukelbir et al. (2015) go a step further in developing automatic differentation variational inference (ADVI) for differentiable probability models. ADVI, which they implement in the Stan programming language, first determines an appropriate variational family. It then optimizes the variational objective function. With this implementation, ADVI can be used for inference with any model that can be expressed in Stan.

Some of the terminology from Kucukelbir et al. (2015) needs definitions. They use the term "differentiable probability model" to refer to members of a class of models that are widely used in machine learning. While

they don't provide a detailed rationale behind their definition, they state that only fully discrete models, like the Ising model, are not "differentiable probability models". Widely used model classes such as linear regression, matrix factorization, and gaussian processes are all differentiable probability models. Additionally, models with discrete latent variables, such as hidden Markov models, mixture models, and topic models belong to the collection of differentiable probability models.

To appreciate the significance of the work of Kucukelbir et al. (2015), we need to say more about why variational difference is difficult to automate. Classical variational inference relies on conditionally conjugate models, where the optimal approximating family matches the prior distribution. Additionally, in some models, variational inference is performed after designing model-specific approximations.

A key to automating variational inference is to choose a density with support matching that of the posterior distribution. In these settings, ELBO maximization can proceed by Monte carlo integration and stochastic optimization. This leads to the black box variational inference of Ranganath, Gerrish, and Blei (2014). Kucukelbir et al. (2015) use a related strategy. They take a "transformation-based" approach in which they automatically transform the support of the latent variables to the real numbers coordinate space, $\mathbb{R}^k$. Then they work with gaussian variational densities.

Kucukelbir et al. (2015) use Stan and its model compiler that handles transformations. Once the joint density of the differentiable probability model is $\mathbb{R}^k$, they can choose a variational distribution that doesn't depend on the model itself. They use a mean-field variational approximation, much like that in the above exercises.

The next step is to maximize the ELBO in $\mathbb{R}^k$. Automatic differentiation here fails; however, elliptic standardization provides a work-around and is implemented in their algorithm. The authors conclude with demonstrations of the utility of their new methods and software. They find that ADVI in Stan outperforms MCMC sampling methods in several metrics, including computing time.

Finally, I wanted to think about how I might adopt Stan for explorations of machine learning models. As an R user, I was excited to see that there is an R interface for Stan (https://mc-stan.org/users/interfaces/rstan). At first glance, it appears to be thoroughly documented with a detailed manual and many worked examples.

References

Kucukelbir, Alp, Rajesh Ranganath, Andrew Gelman, and David Blei. 2015. "Automatic Variational Inference in Stan." In *Advances in Neural Information Processing Systems*, 568–76.

Ranganath, Rajesh, Sean Gerrish, and David Blei. 2014. "Black Box Variational Inference." In *Artificial Intelligence and Statistics*, 814–22. PMLR.