Probabilistic Modeling in Stan

Alp Kucukelbir

October 28, 2015

1 Learning Goals

By the end of this lecture, you will

1. understand the process of implementing a probabilistic model in practice;
2. grasp the advantages and disadvantages of probabilistic programming (Stan); and
3. have seen how to implement a probabilistic model and analyze a dataset in Stan.

2 Probabilistic Modeling in Practice

This class is about developing and using probability models.

Let's say that we already have a probability model in mind. Here is how we would apply that model to a dataset.

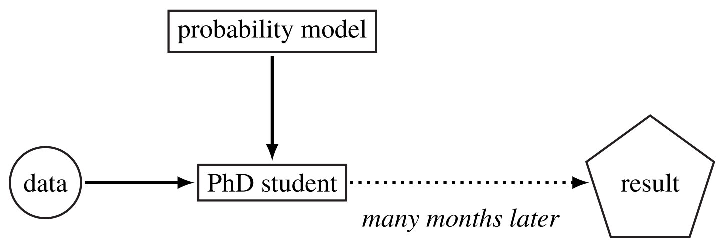

Figure 1: The "how hard could it be?"$^{\text{TM}}$ way of probabilistic modeling.

This is unsatisfying for a few reasons:

1. (Obviously) inefficient use of super-smart PhD students.
2. The algorithm is tied to the model.
3. No feedback.

Why is this the *status quo*?

Linear Regression. Consider the following situation. We want to understand what factors contribute to the price of houses. Let's say we are studying the greater New York City area and that we have measured the sale price of a bunch of homes. We have also gathered factors that we think may contribute to pricing, such as proximity of each home to the closest school, the rating of these schools, crime rates, etc.

We might want to begin with a linear regression approach. So let's put down some notation. There are N houses in our dataset. The $N \times 1$ vector $\boldsymbol{y} = \{y_n\}_1^N$ contains the sale price of each house. Accompanying each house is a D-dimensional row-vector $\boldsymbol{x}_n$ of factors, as described above. We collate these in a $N \times D$ matrix $\mathbf{X}$; treat this matrix as a fixed given quantity.

I like to start describing probabilistic models at the likelihood level. We want to treat house prices $\boldsymbol{y}$ as a probabilistic quantity that depends on the factors $\mathbf{X}$ in some unknown fashion. Let $\boldsymbol{\beta}$ be a $D \times 1$ latent vector. The simplest linear regression likelihood posits the following

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{\beta} \,;\, \mathbf{X}) &= \mathcal{N}(\boldsymbol{y} \,;\, \mathbf{X}\boldsymbol{\beta}, \mathbf{I}) \\ &= \prod_{n=1}^{N} \mathcal{N}(y_n \,;\, \boldsymbol{x}_n\boldsymbol{\beta}, 1). \end{aligned}$$

I also like to read equations out loud in words. This says that $\boldsymbol{\beta}$ should describe a linear combination of factors that explain house prices; it should do so treating each house independently. Any deviations from this structure should follow an independent Gaussian distribution with unit variance.

The Bayesian modeler would then want to posit a prior $p(\boldsymbol{\beta})$ on the latent variable $\boldsymbol{\beta}$ to capture her prior beliefs. Computing the posterior $p(\boldsymbol{\beta} \mid \boldsymbol{y} \,;\, \mathbf{X})$ would then let her follow through with her analysis. Here, we face our first challenge.

Challenge #1 | Conjugacy. The conjugate prior for $\boldsymbol{\beta}$ is another Gaussian. We have two options. We could choose the conjugate prior; this may or may not satisfy our modeling desires. But we can analytically calculate the posterior. Or we could choose a different prior but forsake a closed form posterior; we'll get to this second case later.

Let's choose a conjugate prior $p(\boldsymbol{\beta}) = \mathcal{N}(0, \mathbf{I})$. Skipping details, the posterior is

$$p(\boldsymbol{\beta} \mid \boldsymbol{y} \,;\, \mathbf{X}) = \mathcal{N}\left(\boldsymbol{\beta} \,;\, (\mathbf{I} + \mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}, (\mathbf{I} + \mathbf{X}^\top\mathbf{X})^{-1}\right).$$

See (Murphy, 2012) for a derivation.

So we're done! Right? Well...

Challenge #2 | Computation. We now need to implement this. This means making some decisions. This is a simple case, so we might follow something like this.

1. Figure out what you need.
   - Numerical linear algebra (dot products, matrix multiplication)
   - Probability distributions (library or hand-code, careful: numerical accuracy)
   - Plotting and diagnosis tools
2. Pick a language/library/environment (e.g. R, Python, C++)
   - Does it satisfy your requirements? If not, return to Step 1.
3. Write code. Debug code on simulated data.
4. Run analysis on real dataset.

Conditionally Conjugate Linear Regression. That wasn't too bad! So let's look at a more realistic example. Consider a case where we have many factors in $\mathbf{X}$, such that D is large. We may not expect every factor to contribute towards explaining house prices; thus some entries of $\boldsymbol{\beta}$ could be "turned off" (set to zero). This would indicate that these factors have a negligible effect on house prices.

We could model this by positing an automatic relevance determination (ARD) prior on the latent variable $\boldsymbol{\beta}$. The likelihood is similar to before

$$p(\boldsymbol{y} \mid \boldsymbol{\beta}, \tau \,;\, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \,;\, \boldsymbol{x}_n \boldsymbol{\beta}, \tau^{-1}\right)$$

except that the variance τ^{-1} is also a latent variable.

The ARD prior is hierarchical. It looks like

$$\begin{aligned} p(\boldsymbol{\beta}, \tau, \boldsymbol{\alpha}) &= \underbrace{p(\boldsymbol{\beta}, \tau \mid \boldsymbol{\alpha})}_{\text{NormalGamma}} \underbrace{p(\boldsymbol{\alpha})}_{\text{Gamma}} \\ &= \mathcal{N}\left(\boldsymbol{\beta} \,;\, 0, (\tau \operatorname{diag} \boldsymbol{\alpha})^{-1}\right) \operatorname{Gam}(\tau \,;\, a_0, b_0) \prod_{d=1}^{D} \operatorname{Gam}(\alpha_d \,;\, c_0, d_0) \end{aligned}$$

where $\boldsymbol{\alpha}$ is a D-dimensional latent variable on the regression coefficients $\boldsymbol{\beta}$, where each component gets its own independent Gamma. The parameters a_0, b_0, c_0, d_0 are all fixed.

The graphical model looks like this.

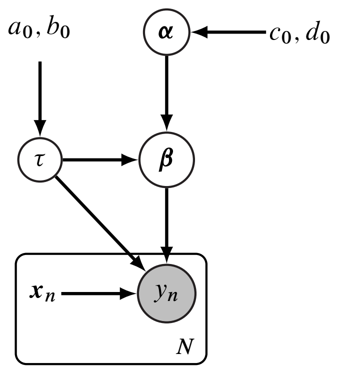

Figure 2: Graphical model of Bayesian linear regression with ARD.

For appropriately chosen parameter values, the latent variable $\boldsymbol{\alpha}$ will try to pick out factors that are "relevant". So even if $\mathbf{X}$ includes many factors, we expect this model to be able to identify the important ones that contribute to house prices. Let us now try to use this model.

Challenge #1 | Conjugacy. This model is not conjugate. However, the ARD prior is not arbitrary either. This model is actually conditionally conjugate, which means that we can compute an accurate approximation to the posterior using Gibbs sampling or variational inference.

Drugowitsch (2013) took the variational approach. It took him *pages*. This is not uncommon; deriving such algorithms is a significant reason why it takes months to get a result.

Challenge #2 | Computation. We now need to think about a few other things. We now need access to more probability distributions, such as the Gamma. Numerical stability is definitely a problem, so we may need to re-parameterize the model in terms of Inverse Gamma distributions. Also, we no longer have clean linear algebraic procedures to implement. We may need to loop through data structures. (Can we live with the cost of slow loops in Python and R?)

Non-conjugate Linear Regression. Now let's say we change something small in the model. For instance, let's change the prior on τ from a Gamma to a logNormal. The model is now completely non-conjugate. We need to carefully derive an "approximate approximate" algorithm; often times this requires brand new research.

What to do?

3 Probabilistic Programming

Probabilistic programming is a new field that seeks to generalize and automate probabilistic computation. I will only focus on Stan, which is an exciting project being developed here at Columbia. Moreover, I will only focus on a subset of what Stan can do.

The big picture goal is this.

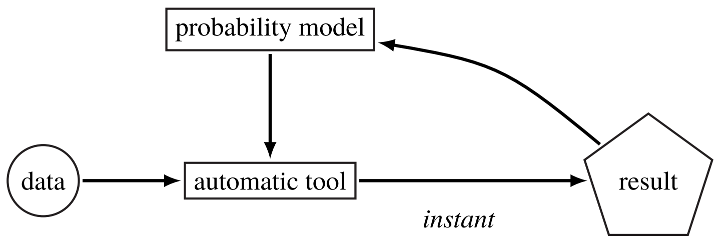

Figure 3: The probabilistic programming way of probabilistic modeling.

We replace the months of tedious work with a tool that automatically takes a probability model and applies it to a dataset. This allows a PhD student to spend her time on "closing the loop".

Stan is an "automatic tool" in this sense. As far as we are concerned, Stan offers us:

1. a probability model compiler (with incredible math/probability support); and
2. a variety of generic inference algorithms.

Stan has various pros/cons. (Let's discuss live.)

References

Drugowitsch, J. (2013). Variational bayesian inference for linear and logistic regression. *arXiv preprint arXiv:1310.5438*.

Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press.