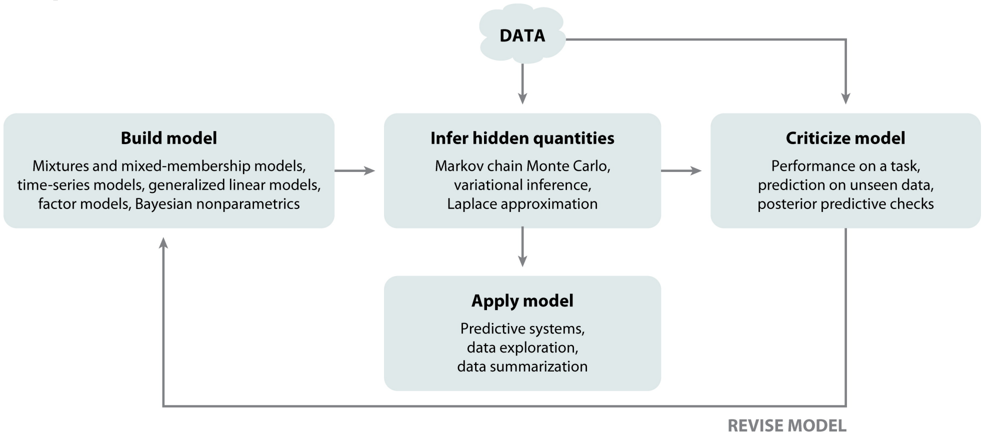
STATS271/371: Applied Bayesian Statistics

Victory Lap

Scott Linderman

May 26, 2021

Box's Loop


DATA
Build model
Mixtures and mixed-membership models, time-series models, generalized linear models, factor models, Bayesian nonparametrics
Infer hidden quantities
Markov chain Monte Carlo, variational inference, Laplace approximation
Criticize model
Performance on a task, prediction on unseen data, posterior predictive checks
Apply model
Predictive systems, data exploration, data summarization
REVISE MODEL

Blei DM. 2014.
Annu. Rev. Stat. Appl. 1:203–32

Blei, *Ann. Rev. Stat. App.* 2014.

PIRELLI
THE RITZ-CARLTON
THE RITZ-CARLTON
PETRONAS
AMG
UBS
TOMMY HILFIGER
Hewlett Packard Enterprise
CROWDSTRIKE
PETRONAS
MOËT & CHANDON

Victory Lap: The great beyond

- Highlights
- Some things we missed

Highlights

	Model	Algorithm	Criticism
Lap 1	Linear Regression	Exact Inference	Bayes Factors
Lap 2	Logistic Regression	Laplace Approx.	Predictive Likelihood
Lap 3	Robust Regression	MCMC / HMC	MCMC Diagnostics
Lap 4	Mixture Models	Gibbs Sampling	Posterior Predictive Checks
Lap 5	Mixed Membership Models	Coordinate Ascent VI	-
Lap 6	Factor Analysis & VAEs	BBVI & Amortization	-
Lap 7	State Space Models	Message Passing & SMC	-
Lap 8	Gaussian Processes	Elliptical Slice Sampling	-

The road not taken

https://radiowest.kuer.org/

Hierarchical models

Many datasets are organized into groups of observations, $\boldsymbol{X} = \{\{x_{g,n}\}_{n=1}^{N_g}\}_{g=1}^{G}$ where G is the number of groups and N_g is the number of observations in group g.

Two extreme models:

1. All groups share the same parameters:

$$p(\boldsymbol{X}, \theta) = p(\theta) \prod_{g=1}^{G} \prod_{n=1}^{N_g} p(x_{g,n} \mid \theta) \tag{1}$$

2. All groups have independent parameters:

$$p(\boldsymbol{X}, \boldsymbol{\Theta}) = \prod_{g=1}^{G} p(\theta_g) \prod_{n=1}^{N_g} p(x_{g,n} \mid \theta_g) \tag{2}$$

Model 1 pools information across all data points when estimating θ, but ignores variability between groups. Model 2 allows variability but shares no information.

Hierarchical Models II

Of course, there is a third way:

$$p(\boldsymbol{X}, \boldsymbol{\Theta}) = p(\theta_0) \prod_{g=1}^{G} p(\theta_g \mid \theta_0) \prod_{n=1}^{N_g} p(x_{g,n} \mid \theta_g) \tag{3}$$

Each group has its own parameters, but they are tied together via a hierarchical prior. Thus, we get the best of both worlds: parameter sharing and group-to-group variability.

A concrete example of **hierarchical Bayesian linear regression**:

$$\begin{aligned} &p(\{\{\boldsymbol{x}_{g,n}, y_{g,n}\}_{n=1}^{N_g}\}_{g=1}^{G}, \{\boldsymbol{w}_g, \sigma_g^2\}_{g=1}^{G}, \boldsymbol{w}_0) = \\ &\qquad \mathcal{N}(\boldsymbol{w}_0 \mid \boldsymbol{\mu}_0, \lambda_0^{-1}\boldsymbol{I}) \prod_{g=1}^{G} \left[\mathcal{N}(\boldsymbol{w}_g \mid \boldsymbol{w}_0, \lambda^{-1}\boldsymbol{I}) \, \mathrm{Inv}{-}\chi^2(\sigma_g^2 \mid \nu, \tau^2) \prod_{n=1}^{N_g} \mathcal{N}(y_{n,g} \mid \boldsymbol{w}_g^\top \boldsymbol{x}_n, \sigma_g^2) \right] \end{aligned} \tag{4}$$

Questions: What happens when $\lambda \to \infty$? What happens when $\lambda \to 0$? Same for λ_0. How could you share information about the variances, σ_g^2?

Hierarchical Models III

Consider the question of determining K, the number of components in a mixture model.

Rather than using predictive log likelihood to choose a single K (recall Lap 4), we treat K as a random variable in a hierarchical model,

$$p(\{\boldsymbol{x}_n, z_n\}_{n=1}^N, \boldsymbol{\Theta}, K) = p(K)p(\boldsymbol{\pi} \mid K)\prod_{k=1}^{K} p(\theta_k)\prod_{n=1}^{N} p(z_n \mid \boldsymbol{\pi})\, p(\boldsymbol{x}_n \mid \theta_{z_n}) \tag{5}$$

Suppose K can take on a finite range of values in the set $\mathscr{K}$. The marginal distribution, summing over K, is a **mixture of finite mixture models** [Miller and Harrison, 2018]

$$p(\{\boldsymbol{x}_n, z_n\}_{n=1}^N, \boldsymbol{\Theta}) = \sum_{\kappa \in \mathscr{K}} p(K = \kappa)\, p(\boldsymbol{\pi} \mid \kappa)\prod_{k=1}^{\kappa} p(\theta_k)\prod_{n=1}^{N} p(z_n \mid \boldsymbol{\pi})\, p(\boldsymbol{x}_n \mid \theta_{z_n}) \tag{6}$$

Note that the size of $\boldsymbol{\pi}$ and the set $\{\theta_k\}_{k=1}^K$ changes with the random variable K.

Bayesian nonparametrics

Taking this idea to its countably infinite limit $\mathscr{K} = 1, 2, \ldots$, we arrive at Bayesian nonparametric mixture models.

We discussed such a model—the Dirichlet process mixture model—in Lap 4, but that's one of many.

In fact, we've seen another Bayesian model already: Gaussian processes!

In general, BNP deals with priors on infinite dimensional objects; i.e. stochastic processes.

Other examples include the beta process, which is useful for constructing nonparametric **factorial** latent variable models, and the Pitman-Yor process, which generalizes the DP to allow for power-law distributed weights.

Orbanz [2014] is a great introduction to this research area.

Bayesian deep learning

Gaussian processes gave us tools to reason about random functions. Not just point estimates; full posterior distributions. Of course, there's another cool function class in town—neural networks!

Recall the standard feedforward neural network from Lap 6. Each layer applies a linear function followed by an elementwise nonlinearity.

$$\mathbb{E}[\boldsymbol{y}_n \mid \boldsymbol{x}_n, \boldsymbol{\Theta}] = g(\boldsymbol{W}_L g(\boldsymbol{W}_{L-1} g(\ldots g(\boldsymbol{W}_1 \boldsymbol{x}_n + \boldsymbol{b}_1)\ldots) + \boldsymbol{b}_{L-1}) + \boldsymbol{b}_L) \tag{7}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=1}^{L}$ is the set of weights and biases.

If we put a prior on the weights and biases, $p(\boldsymbol{\Theta})$, we have a random function $g : \mathbb{R}^{D_x} \to \mathbb{R}^{D_y}$. Both the prior on weights and the structure of the neural network determine the effective "kernel," $\mathrm{Cov}(g(\boldsymbol{x}_n), g(\boldsymbol{x}_{n'}))$.

Once again, Neal [1996] paved the way for this research area, deriving the "neural network kernel" for an infinitely wide single layer network with probit units.

Several recent works have extended these techniques to other architectures [Garriga-Alonso et al., 2018, de G. Matthews et al., 2018, Lee et al., 2017, Yang, 2019, Arora et al., 2019], including the **neural tangent kernel** [Jacot et al., 2018]. See Wilson [2020] for an introduction.

Probabilistic programming

Ultimately, this course has been about developing techniques for modeling generative processes as joint distributions over parameters, latent variables, and data. We introduced a series of algorithms for "inverting" the generative model to infer the latent states and parameters given data.

However, those inference algorithms were often quite tailored to the model.

The past ten years have seen great advances in **probabilistic programming languages** like Stan, Church, Vatican, Anglican (sensing a trend?), Turing, Gen, Edward, Pyro, etc. These languages provide a way to specify the generative process (i.e. how to sample from the model), and then automate the inference algorithm.

It's a compelling idea! There's still work to be done in order to design inference algorithms as effective as our handcrafted ones, but that gap is closing.

Automated model search

Probabilistic programming languages aim to automate inference. Is it possible to automate the model building process as well? In probabilistic programming parlance, is it possible to do program induction?

It's an exciting idea! Much of science is about building an increasingly refined model of the world, and it advances with the creative sparks (what Popper called **"bold ideas, unjustified anticipations, and speculative thought"**) that cause us to revise our current models. Could a machine do that?

The automated statistician (Lap 8) aims to do exactly that for the special case of Gaussian processes. Many other model search strategies have been put forward as well, but ultimately the combinatorial growth of the problem seems to necessitate something more intelligent than brute force or greedy approaches.

Exciting as it is, it's also somewhat sad... Box's loop of model building, inference, and criticism is what draws many of us to science and statistics in particular. As much as I love understanding, I also enjoy the pursuit.

Bayesian decision making

Ultimately, what do we use models for? Aside from understanding the universe and its complex generative processes, we use them to make decisions.

Put differently, Box's loop often involves some **action** on the part of the scientist or policy maker. Those actions can incur some **loss** (or reward, if you like thinking positively), and they can yield new data to inform subsequent models.

Bayesian decision theory involves averaging over uncertainty in latent variables and parameters to compute the expected loss of various decisions.

Often, decisions are made sequentially, and your actions can influence the future state of the world and hence the available future actions and expected loss. This is where Bayesian decision theory meets **Markov decision processes, reinforcement learning, and control theory**.

One particular instance of such problems is **active learning**, aka **optimal experimental design**, where the goal is to infer model parameters using strategically chosen experiments. E.g. to learn the weights of a regression by querying chosen inputs $\boldsymbol{x}_n$. **Bayesian optimization** [Shahriari et al., 2015] is one such example.

Ask me anything

What other topics are you interested in?

Final logistics

- We’ll have two poster sessions next week: **1pm Wednesday, June 2** and **1pm Thursday, June 3**.
- Please attend both if you can! Poster sessions are much more fun with an audience.
- Stay tuned for instructions on how to use GatherTown for the poster sessions.
- I am still planning to upload our solutions to the homework. They just need a bit of clean-up.
- Please fill out the course feedback questionnaire! It will help us improve this course in the future.

It's been a pleasure having all of you in class, virtually.
Thank you, and have a great summer!

References I

Jeffrey W Miller and Matthew T Harrison. Mixture models with a prior on the number of components. *J. Am. Stat. Assoc.*, 113(521):340–356, 2018.

Peter Orbanz. Lecture notes on Bayesian nonparametrics. May 2014. URL `http://www.gatsby.ucl.ac.uk/~porbanz/papers/porbanz_BNP_draft.pdf`.

Radford M Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 1996.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. August 2018.

Alexander G de G. Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. April 2018.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. November 2017.

References II

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. February 2019.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. April 2019.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. June 2018.

Andrew Gordon Wilson. Bayesian neural networks from a gaussian process perspective. `http://gpss.cc/gpss20/slides/Wilson2020_part2.pdf`, 2020. Accessed: 2021-5-26.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.