Module 1: Introduction to probabilistic modelling

5th June 2019

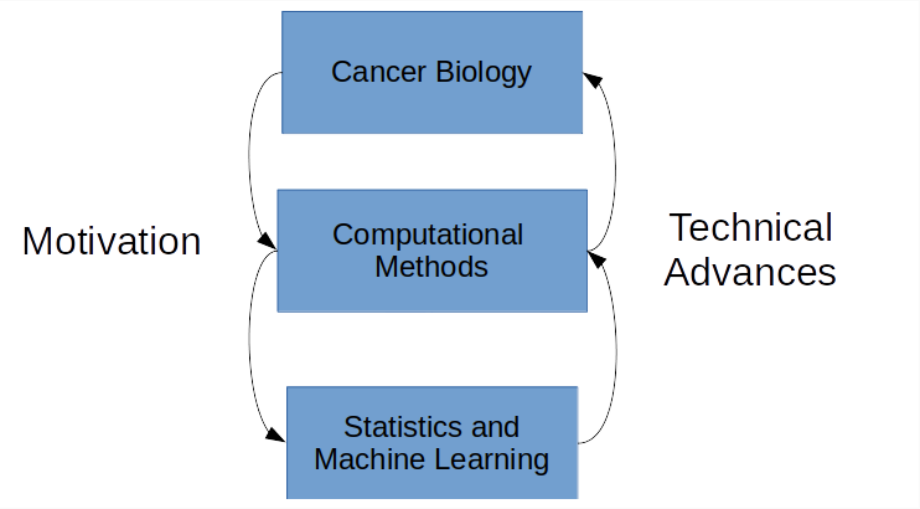
Housekeeping

- Slides and notes are at `https://github.com/aroth85/contra_workshop`.
- Time permitting I will post some code in the form of Jupyter notebooks.
  - I would like some feedback on what examples people would find helpful.
  - Is ther a colloborative forum we could use to communicate? Maybe Slack or Google groups?
- Please ask questions while we go.
  - This is not a meant to be a research talk.

About me

- Assistant professor at UBC in the Departments of CS and Pathology
- Scientist at the BC Cancer agency
- Personal website `https://aroth85.github.io`
- Email `aroth@bccrc.ca`
- Currently hiring students and post-docs

My research


Cancer Biology
Motivation
Computational Methods
Technical Advances
Statistics and Machine Learning

Overview of probabilistic modelling

What do we mean by probabilistic modelling

- In general any model that treats the data as a random quantity.
- More specifically a hierarchical model where parameters are also random.

Questions

- Can anyone give examples of probabilistic models?
- Can anyone give examples of non-probabilistic models?

Why do we need probabilistic models

- Data is always “noisy” in some way.
- In computational cancer the noise can come from:
  - Technical error due to the measurement technology (random variation).
  - Biological variation (systematic).
- Probabilistic models naturally capture random variation through the assumption the data is a random variable.
- Systematic variation can be handled by adding layers of hidden or latent variables to the model.

Question
Consider the problem of calling SNVs from bulk sequencing. What are some sources of technical and biological variation that could affect the data?

Building probabilistic models

- In this workshop we will look at how to build probabilistic models.
- Some key questions we address:
  1. How do we construct an appropriate model for the data?
  2. How can estimate the model parameters?
  3. How do report these model parameters i.e. how certain are we?
- When working through later modules, the most focus should be on the modelling process not the model we are building.
  - With that said the models we will construct can easily be turned into useful tools for real data.

The steps to building a probabilistic model

1. Identify a well defined problem.
2. Explore the data and qualitatively understand its properties.
3. Decompose the main problem into smaller pieces which can be iteratively extended.
4. Identify and implement a means to estimate model parameters.
5. Validate the model and inference approach.

What is the problem

- Identifying a well defined problem is the most important step.
- Some key considerations:
  - What are the quantities we wish to measure?
  - What type of data will we have available?
  - What are the current approaches to address the problem?

Remark

It is easy to skip the step of looking at existing approaches. Even if no formal method has been published, there are often simple approaches which can be applied. One useful exercise is to look at how biologists and bioinformaticians are currently analysing the data. It may be something simple like a t-test. Having baseline approaches to compare to is critical to ensure any new method actually provides an improvement.

Explore the data

- Looking at real data through some form of qualitative analysis is critical.
- Feasibility of problem may depend on features that may not be obvious.
- Exploratory analysis can also reveal unexpected noise.
- Some examples:
  - How practical is it to call variants from 5' single cell RNA-Seq?
  - What about single cell DNA sequencing?
  - What does binned read count data look from a diploid cell line?

Building a model

- Once you have a well defined problem and understand the data, then you can begin building a model.
- Often attempting to build a very complex model initially will be challenging.
- A useful strategy is to identify smaller problems which are easier to model.
- Using the framework of Bayesian hierarchical modelling it easy to extend simple models.
- A common idea is to share *statistical strength* across data points.
  - This typically manifest as sharing parameters between data points, samples, cells etc.

Fitting the model

- Probabilistic models will typically have many parameters.
- A core question is how to estimate these parameters given the data.
- As we will see, Bayesian inference provides a prescription for this problem.
  - However, there is still a lot of flexibility.
- A core question is how to report estimates of model parameters.
  - What point estimate do we report?
  - What measure of uncertainty?
- Note there is an interplay between inference and model building.
  - If we cannot fit the model it is not very useful for analysing data.
  - If we have better tools for model fitting we can confidently build more complex models.

Validation

- The final step in the process is to validate the model.
- Some important questions:
  1. Does the inference algorithm work well?
  2. Does the model fit real data and capture the biological quantities of interest accurately?
- The first question can be answered by simulating data from the model and assessing the accuracy of inferred parameters.
- The latter question can only be answered with *ground truth* real world data.

Bayesian inference

Basic probability

- Let $A, B, \{C_i\}$ be random events.
- Then there are three basic equations that are at the centre of Bayesian probabilistic modelling.

$$\begin{aligned} \text{Joint distribution - } p(A,B) &= p(A|B)p(B) \\ \text{Bayes' rule - } p(A|B) &= \frac{p(B|A)p(A)}{p(B)} \\ \text{Law of total probability - } p(B) &= \sum_i p(B|C_i)p(C_i) \end{aligned}$$

Background

- The Bayesian paradigm is one framework for probabilistic modelling.
- In the Bayesian setting model parameters are considered random like the data.
- The core quantity we need to compute in the Bayesian setting is the posterior

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}$$

- Here X is the data, θ are the parameters, $p(X|\theta)$ is the likelihood and $p(\theta)$ the prior.
- The normalisation constant (model evidence) $p(X) = \int p(X|\theta)p(\theta)\mathrm{d}\theta$ is typically hard to compute.

Background

- The Bayesian approach to model fitting states that all the information about the parameters is encapsulated in the posterior.
- We begin with prior belief about θ encoded in the prior $p(\theta)$.
- Our updated belief about θ after seeing the data is given by the posterior $p(\theta|X)$.
- If we see new data then our posterior becomes our new prior.

Summarising the posterior

- In principle the posterior tells us everything we could want to know about θ.
- In practice it can be hard to interpret high dimensional posteriors so we turn to point or region estimates.
- Simple point estimates include reporting summary statistics such as the mean and variance.
- Region estimates can be used to quantify uncertainty i.e. credible intervals.

Loss functions

- Loss functions provide a very general framework for summarising posteriors.
- A loss function $L(x, y)$ is a positive valued function which encodes our loss if we predict x when the true value is y.
  - L^1 loss $|x - y|$
  - L^2 loss $||x - y||^2$
- The Bayesian approach to point estimation is then to report the value with the minimum expected loss

$$\begin{aligned} \hat{\theta} &= \text{argmin}_{\theta'} \mathbb{E}_{p(\theta|X)}[L(\theta, \theta')] \\ &= \text{argmin}_{\theta'} \int L(\theta', \theta) p(\theta|X) \mathrm{d}\theta \end{aligned}$$

Hierarchical models

- Bayesian modelling is modular because parameters are random variables.
- We can thus make the hyper-parameters that govern the distribution of these variables random as well.
- This allows us to construct hierarchical models.
- We can use this to:
  - Embed simpler models in more complex ones.
  - Reduce the impact of the prior.
  - Share parameters to share statistical strength.

Posterior inference

Why Bayesian inference is challenging

- Bayesian inference is conceptually simple - we apply Bayes' rule and compute the posterior.
- In practice this means computing the normalisation constant $p(X) = \int p(X|\theta)p(\theta)\mathrm{d}\theta$.
  - If θ is high dimensional this is typically intractable.
- Bayesian inference becomes hard because we need to either:
  - Avoid explicitly computing $p(X)$.
  - Use advanced methods to estimate $p(X)$.
- As a result we almost always rely on some method to compute an approximation to the posterior.

MAP estimation

- The simplest approximation one can make to the posterior is to use a distribution that places all the mass on one point.
- This point is typically the maximum of $p(\theta|X)$.
- We call this value, $\hat{\theta}$, the maximum a posteriori (MAP) estimate.
- This is tractable because

$$\begin{aligned} \hat{\theta} &= \text{argmax}_\theta p(\theta|X) \\ &= \text{argmax}_\theta p(\theta, X) \end{aligned}$$

because $p(X)$ does not depend on θ.

MAP estimation

- It is often reasonably efficient to compute the MAP estimate
  - This is especially true if the parameters are continuous and we can use gradient descent.
  - It is trickier when the parameters are discrete and have a large state space.
- The MAP estimate is a bad estimate of the posterior distribution typically.
- There are two issues:
  - We approximate a distribution with a single point.
  - The mode of the distribution may be a typical of the distribution as a whole.

Question
Can anyone think of a case when the MAP estimator is hard to compute?

Monte Carlo methods

- In the Bayesian setting we typically want to compute expected values.
  - For example mimising the expected loss functions.
- Monte Carlo methods make the following simple observation

$$\begin{aligned} \mathbb{E}_p[h] &= \int h(x)p(x)\mathrm{d}x \\ &\approx \frac{1}{S}\sum_{s=1}^{S} h(x^{(s)}) \end{aligned}$$

  where $x^{(s)}$ are random draws from p.
- The accuracy of this estimator increases as the number of samples, S, increases.
  - This is the Law of Large Numbers in action.

MCMC methods

- Sampling from the posterior directly is typically as hard as computing the posterior.
- The basic idea of Markov Chain Monte Carlo (MCMC) methods is to construct a Markov chain that admits $p(\theta|X)$ as its invariant distribution.
- We can then sequentially draw samples from the Markov chain to obtain samples from $p(\theta|X)$.

MCMC burnin

- MCMC methods are only guaranteed to draw samples from $p(\theta|X)$ in the limit as we take an infinite number of iterations.
- In practice the Markov chain often gets very close to sampling from $p(\theta|X)$ in a finite number of iterations.
- However, early samples from the chain may come from a distribution which is quite different from $p(\theta|X)$.
  - Thus we discard some initial number of samples as *burnin*.

MCMC thinning

- Successive samples will not be independent draws from $p(\theta|X)$ they will be correlated.
- This does not affect the asymptotic result.
- This does mean we need more samples than if we could sample from $p(\theta|X)$ directly.
- Often people only retain every n^{th} sample, a procedure called *thinning*.
- Thinning reduces the storage cost.
- However, thinning does not improve accuracy for a given computational budget.

Metropolis-Hastings algorithm

- The MH algorithm is the simplest MCMC method to implement.
- We proceed as follows:
  - Propose a new value θ' from a distribution $q(\theta'|\theta)$.
  - Keep the proposed value with probability $\alpha(\theta,\theta') = \min\left\{1, \frac{p(\theta'|X)q(\theta|\theta')}{p(\theta|X)q(\theta'|\theta)}\right\} = \min\left\{1, \frac{p(\theta',X)q(\theta|\theta')}{p(\theta,X)q(\theta'|\theta)}\right\}$ otherwise keep the old value.
- The trick of the MH algorithm is that the intractable normalisation cancels in $\alpha(\theta,\theta')$ so we only need to evaluate the joint distribution $p(\theta',X)$.
- The key to obtaining efficient MH algorithms is to use a good proposal q.

Gibbs sampling

- Assume $\theta = (\theta_1, \theta_2)$ and we can sample from the conditional distributions $p(\theta_1|\theta_2, X)$ and $p(\theta_2|\theta_1, X)$.
- The Gibbs sampler works by alternatively sampling from $p(\theta_1|\theta_2, X)$ and then $p(\theta_2|\theta_1, X)$.
- We use the previous values from each step to compute the conditional in the next.
- Gibbs sampler are most useful when:
  - The model has conditional conjugacy so we have closed form distribution.
  - The parameters are discrete with a small state space.
- Gibbs sampling can get stuck in modes fairly easily. Often mixing MH moves that update θ_1 and θ_2 jointly can help.

Metropolised-Gibbs algorithm

- If θ is high dimensional, then it can be very hard to find a good proposal q.
  - This is the curse of dimensionality which will force us to make only small changes to any dimension.
- A useful strategy in this case is to break $\theta = (\theta_1, \ldots, \theta_B)$ into B blocks of low dimension.
- We can then use an MH algorithm targeting $p(\theta_i | \{\theta_j\}_{j \neq i}, X)$ with acceptance ratio

$$\begin{aligned} \alpha(\theta_i, \theta_i') &= \min\left\{1, \frac{p(\theta_i' | \{\theta_j\}_{j \neq i}, X) q(\theta_i | \theta_i')}{p(\theta_i | \{\theta_j\}_{j \neq i}, X) q(\theta_i' | \theta_i)}\right\} \\ &= \min\left\{1, \frac{p(\theta', X) q(\theta | \theta')}{p(\theta, X) q(\theta' | \theta)}\right\} \end{aligned}$$

Metropolised-Gibbs algorithm

- This blocked algorithm is often called Metropolised-Gibbs algorithm.
- In practice it is often the only way to get MH to work in high dimensional problems.
- Note that the normalisation $p(\{\theta_j\}_{j\neq i}, X)$ cancels so we get the same acceptance ratio as the normal MH algorithm.
- If $q(\theta_i'|\theta_i) = p(\theta_i'|\{\theta_j\}_{j\neq i}, X)$ the acceptance ratio becomes 1.
  - This is the Gibbs sampler