Sampling II: Inference Control and Driving of Natural Systems

Nick Jones

nick.jones@imperial.ac.uk

Sampling II

What we have covered so far. The problem of un-normalized distributions.

Objectives for today:

1. Intuition behind two Markov chain Monte Carlo schemes (which is relevant for tomorrow's practical):
   - Metropolis Sampling
   - Gibbs Sampling
2. Introduction to Probabilistic Population Coding with neurons (*not* explicitly a sampling approach but relevant for today's practical)

Metropolis Hastings

Our treatment of MCMC methods will be very brief - engaging introductory online lectures on this topic can be found by Ian Murray on `videolectures.com`.
As discussed:
MCMC exploits relative probabilities - or, effectively, a local probability estimation rather than global probability estimation - it looks at the ratio of the unnormalized distribution at two points, $\mathbf{x}$, $P^*(\mathbf{x})$ and, $\mathbf{x}'$, $P^*(\mathbf{x}')$. This is, of course, the same as the ratio of $P(\mathbf{x})$ and $P(\mathbf{x}')$. So we can make judgements about ratios without worrying about normalization.

Metropolis method

We have an unnormalized distribution $P^*(x)$ that we'd like to sample from (or perhaps use to calculate expectations of functions $f(x)$ with x possibly a vector). We suppose we have a proposal distribution $Q(x'|x^{(t)})$ (notation and argument after MacKay [1] and Gelman et al [2]) which is symmetric so $Q(x_a|x_b) = Q(x_b|x_a)$. An example would be $Q(x'|x^{(t)}) \sim \mathcal{N}(x^{(t)}, \sigma^2)$ (where σ is a parameter we need to choose wisely - see later).

Metropolis Algorithm

[Initialize with some well motivated choice for $x^{(t=0)}$]

- Draw a proposal x' from $Q(x'|x^{(t)})$.
- If $r = \frac{P(x')}{P(x^{(t)})} > 1$ then let $x^{(t+1)} = x'$ (accept the proposal)
- If $r < 1$ let $x^{(t+1)} = x'$ with probability r (accept) and let $x^{(t+1)} = x^{(t)}$ with probability $1 - r$ (reject and stay put)
- advance time and go back to first step until time-out.

Remarks about the Metropolis Algorithm

Why does it work?

We need our Markov chain to be irreducible, aperiodic and not transient so that it has a single stationary distribution. We need this stationary distribution to remain the same under the action of our transitions *and* we require that this distribution is the one we'd like to draw from (this is sometimes obscured in introductions): $P(x)$. If we can show that $P(x)$ is invariant under the action of the chain (and we know the chain has a unique steady state) then we know that the only distribution we could be sampling from (in the long time limit) is $P(x)$.

Suppose $P(x_b) > P(x_a)$. Suppose that our chain has stationary distribution $S(x)$. We can consider the chance of transitioning $a \to b$: $P(x^{(t-1)} = x_a, x^{(t)} = x_b) = S(x_a)Q(x_b|x_a)$ and the chance of transitioning in the opposite direction: $b \to a$

$P(x^{(t-1)} = x_b, x^{(t)} = x_a) = S(x_b)Q(x_a|x_b)P^*(x_a)/P^*(x_b)$.

Remarks about the Metropolis Algorithm II

$P(x^{(t-1)} = x_a, x^{(t)} = x_b) = S(x_a)Q(x_b|x_a)$ and
$P(x^{(t-1)} = x_b, x^{(t)} = x_a) = S(x_b)Q(x_a|x_b)P^*(x_a)/P^*(x_b)$.

If $S(x) = P^*(x)/Z_p$ we discover that we have what is called detailed balance:
$P(x^{(t-1)} = x_a, x^{(t)} = x_b) = P(x^{(t-1)} = x_b, x^{(t)} = x_a)$ (check this please). If a distribution $P(X, Y) = P(Y, X)$ it has identical marginals. Since this is true from
$P(x^{(t-1)} = x_a, x^{(t)} = x_b) = P(x^{(t-1)} = x_b, x^{(t)} = x_a)$ it follows that the distribution at time t, $P(x^{(t)})$, is the same as the distribution at time $t-1$, $P(x^{(t-1)})$. From this we conclude that we have an invariant distribution over time and since there is only one invariant distribution of the chain we know that the candidate distribution $S(x) = P^*(x)/Z_p$ is the stationary distribution.

Remarks about the Metropolis Algorithm III

Can I use all of the samples? The difference between calculating expectations and requiring independent draws.
How long should I wait for my samples to be independent? The problem of burn-in, convergence to stationarity and de-correlation times.
Is my algorithm performing a random walk and how does it perform in higher dimensions? The Metropolis algorithm performs a biased random walk where the bias is dependent on the change between $P^*(x')$ and $P^*(x^t)$. Suppose that the characteristic lengthscale of the proposal distribution σ is such that the proposed x' is often such that $P^*(x') \sim P^*(x^t)$. Our proposals will be such that $P^*(x')/P^*(x^t) \sim 1$ and we only feel a weak bias per timestep.

Remarks about the Metropolis Algorithm IV

If our distribution $P^*(x)$ has most of its mass contained within a distance L then the minimum length of time to explore $P^*(x)$ will scale like $T \sim (L/\sigma)^2$ (after [1]). A simple argument shows that if $P^*(x)$ is a D-dimensional Gaussian with largest and smallest lengthscales σ_{max} and σ_{min} by noting that $\sigma_{max} = L$ and setting our proposal lengthscale $\sigma = \sigma_{min}$ we find that the time to make an independent sample is $T \sim (\sigma_{max}/\sigma_{min})^2$ which has the virtue that it is independent of the dimension D (thought it is quadratically dependent on a ratio which might be large $(\sigma_{max}/\sigma_{min})$).

Gibbs Sampling

Gibbs sampling can be seen as a particular type of Metropolis algorithm where proposals are accepted with probability 1 ($r = 1$). Because it is a Metropolis sampler it can successfully sample from a desired distribution.

It relies on an important extra condition about the unnormalized function we'd like to sample from $P^*(x)$: that its univariate conditional distributions $P^*(x_i|x_{/i})$ *can* be easily normalized so that we can draw from $P(x_i|x_{/i})$ for all i without problem (where $x_{/i}$ is all elements of x save the i^{th}).

Gibbs Sampling II

The condition that it is easy to sample from the univariate conditional distributions needn't be satisfied by $P^*(x)$ (it could have horrible conditional distributions also) but for physical/biological implementations it is very natural: we might imagine that each variable x_i is a distinct physical system which can generate random samples conditional on the configuration of the rest of the system $x_{/i}$. We'll be using this insight in the exercises in the next day's session. Gibbs sampling is called Glauber dynamics in the physics literature (go and have a very brief read about Glauber dynamics).

Gibbs Sampling III

I'll provide a (human) animated version of Gibbs Sampling on the board with a 2d Gaussian.

We suppose our sampler is at location $x^{(t)}$ at time t and we consider the proposal distribution $Q(x'|x^{(t)}) = P(x_i|x^{(t)}_{/i})$ where we pick the index i cyclically. Drawing from $P(x_i|x^{(t)}_{/i})$ yields a new value for x_i at time $t+1$ which we accept (deterministically) and update the i^{th} element of $x^{(t)}$ to its new value.

Advanced question: The energetic cost of sampling

We have considered natural systems that perform sampling. But what is the best one can do?

Advanced question: For a given $P(x)$ one can ask if there is a minimum expected energy cost associated with sampling from it repeatedly. A physical system which can be used to generate a sample has to have its state reset after each sample. Deleting information has an energy cost of $k_B T$ per bit (Landauer's principle). Is the minimum energy cost of sampling from a distribution its entropy scaled by $k_B T$?

Harder: if we want to be able to use randomness with a rate λ_{sample} how does this modulate my minimum energy cost?

Not advanced and expected: find and understand a brief proof of Landauer's principle (if you can't find one by the time we hit control theory ask me).

Representing Probability in the Brain

There are two major strands in approaches to constructing probabilistic models in the brain:

- Probabilistic Population Coding
- Sampling approaches

You'll find them discussed in these two review papers which are a reasonably easy read and also combine to give an introduction to Bayesian cognitive science more generally:

- Probabilistic brains: knowns and unknowns. Alexandre Pouget, Jeffrey Beck, Wei Ji Ma and Peter Latham, Nature Neuroscience, 2013.
- Statistically optimal perception and learning: from behavior to neural representations. József Fiser, Pietro Berkes, Gergő Orbán and Máté Lengyel, Trends in Cognitive Sciences, 2010.

Please read them (they are pretty interesting).

What's in and what's out

What we're not going to do: we're not going to give a justification as to why it is reasonable to suppose the Brain performs Bayesian inference or why neural models are appropriate.

We will: look at how neurally inspired models can perform inference and comment on some of their properties.

Point Process Sensor Array

Today we'll be covering the first approach: Probabilistic Population Coding. We'll start by considering properties of a simple sensor system and then connect this back to the brain.

Consider the following sensor: a point process with rate $\lambda_i(g, x) = g\, exp[\frac{(x-\mu_i)^2}{2\sigma^2}]$ and inputs x and g. Where we suppose that there is a stimulus with value x and an intensity g.

Sensor i will have a high rate if g is large and if $x \sim \mu_i$.

We can specify a population of sensors $i \in \{1, ..., N\}$ with ordered means $\mu_i < \mu_{i+1}$. If all the sensors have a common input x (with intensity g) where the sensor i with μ_i closest to x shows the greatest response.

Point Process Sensor Array II

An interpretation of σ is an indication of the hard-wired trust that the system has that a value x input is really x. Alternatively one can think of it as a tolerance to ensure that the system responds to all possible inputs x: if $\mu_i - \mu_{i+1} \sim \sigma$ and if the input $x \in [\mu_1 - \sigma, \mu_N + \sigma]$ there will always be one sensor, i, that is likely to respond appreciably (on a timescale $1/\lambda_i$).

The sensors can be viewed as crude neurons where λ is the firing rate $exp[\frac{(x-\mu_i)^2}{2\sigma^2}]$ is the tuning curve of the neuron (its average firing rate as we tune over a variety of inputs x) and g is the strength of the input signal or gain. *We will be relating g to how convincing the stimulus is about the input value x and we'll be connecting it to the variance of a distribution.* g could encode how bright or dim a photo of arrow is and x could encode its orientation.

Point Process Sensor Array III

We can give a chemical interpretation:

$A_i + g \xrightarrow{exp[\frac{(x-\mu_i)^2}{2\sigma^2}]} B_i$

$B_i \xrightarrow{K} A_i + C_i$

if $K \gg g\, exp[\frac{(x-\mu_i)^2}{2\sigma^2}]\ \forall x, i, g$ then we can think of C_i as an instantaneous counter of the reaction making B_i.

Can you construct a chemical system with a rate approximating $exp[\frac{(x-\mu_i)^2}{2\sigma^2}]$?

Point Process Sensor Array to encode posteriors

Given a sustained input x with intensity g for a time T what is the probability of observing $\mathbf{n}$ events with n_i being the number of events associated with neuron/sensor i. Since each neuron is independent of every other then the likelihood will be a product of Poisson distributions of the following form:
$P(n_i|x,g) = (T\lambda_i)^{n_i} exp^{(-T\lambda_i)}/n_i!$
recalling that $\lambda_i(g,x) = g\, exp[\frac{(x-\mu_i)^2}{2\sigma^2}]$.

Point Process Sensor Array to encode posteriors II

$P(\mathbf{n}|x,g) = \Pi_i P(n_i|x,g) = \Pi_i (T\lambda_i)^{n_i} exp^{(-T\lambda_i)}/n_i!$
From Bayes theorem we can find $P(x,g|\mathbf{n}) \propto P(\mathbf{n}|x,g)P(g)P(x)$ and assuming for simplicity that $P(x)$ is uniform then one can integrate out over g (please convince yourself of this) and find: $P(x|\mathbf{n}) \sim \mathcal{N}(\frac{\sum_i \mu_i n_i}{\sum_i n_i}, \frac{\sigma^2}{\sum_i n_i})$.
To show this you'll need to use the condition that for σ sufficiently large $\sum_i \lambda_i(x,g)$ is independent of x.
We'll discuss the interpretation of the behaviour of the mean and variance but note that, by construction, $< n_i >= T\lambda_i$ so this expression for $P(x|\mathbf{n})$ implicitly depends on T and g. *In particular the variance* $\frac{\sigma^2}{\sum_i n_i}$ *is inversely proportional to the intensity/gain* g.

Cue Combination

Humans appear to combine information from different sensory modalities in a simple fashion.
Suppose a stimulus x is such that the response of "sight" is $P(c_1|x)$ and "sound" is $P(c_2|x)$ then the posterior is $P(x|c_1, c_2) \propto P(c_1|x)P(c_2|x)P(x)$ (where $P(x)$ is our prior which we set uniform for simplicity). If $P(c_1|x)$ and $P(c_2|x)$ are Gaussian with means μ_1, μ_1 and variances σ_1, σ_1 we can find very quickly that the posterior $P(x|c_1, c_2)$ has the following mean and variance:
$\mu_3 = \frac{\sigma_2^2\mu_1+\sigma_1^2\mu_2}{\sigma_1^2+\sigma_2^2}$
$1/\sigma_3^2 = 1/\sigma_1^2 + 1/\sigma_2^2$

A virtue of Probabilistic Population Coding

The array of sensors we considered was an example of Probabilistic Population Coding. This formulates a possible neural process that encodes a posterior distribution over our stimulus value: $P(x|\mathbf{n})$.

It has the virtue that it is very easy to perform cue combination. We suppose that we have two sensor populations corresponding to different sensory modalities and each population might have a different intensity of response g and $\widehat{g}$.

We define a third integrator population that simply adds the events in the i^{th} register from modality 1 (sight) with the i^{th} register from modality 2 (sound).

Cue Combination with Chemicals

Chemically this combination of cues would look like this:

$$A_i + g \xrightarrow{exp[\frac{(x-\mu_i)^2}{2\sigma^2}]} B_i$$

$$B_i \xrightarrow{K} A_i + C_i$$

for sensory modality 1 (sight)

and

$$\widehat{A}_i + \widehat{g} \xrightarrow{exp[\frac{(x-\mu_i)^2}{2\sigma^2}]} \widehat{B}_i$$

$$\widehat{B}_i \xrightarrow{K} \widehat{A}_i + C_i$$

for sensory modality 2 (sound). Where $\widehat{A}_i$ and $\widehat{B}_i$ are new sets of chemicals we've added to our system. Note that C_i is common to both modalities and is acting like an integrator that counts events. [For this to work we still need $K \gg g\, exp[\frac{(x-\mu_i)^2}{2\sigma^2}]\ \forall x, i, g$ and similarly for the $\widehat{}$ variables]. The virtue of using chemicals to implement this is that we can associate a temperature with our chemical reactions and so reason about the thermodynamics of this inference architecture.

Cue Combination with PPC

One can perform the calculation we did before to obtain $P(x|\mathbf{n})$ but now we have two populations which might have different intensities/gains g and $\widehat{g}$ and which will generate two data sets $\mathbf{n}, \widehat{\mathbf{n}}$.

$P(n_i|x,g) = (T\lambda_i)^{n_i} exp^{(-T\lambda_i)}/n_i!$ and

$P(\widehat{n}_i|x,g) = (T\widehat{\lambda}_i)^{\widehat{n}_i} exp^{(-T\widehat{\lambda}_i)}/\widehat{n}_i!$

Taking appropriate products we can find $P(x|\mathbf{n},\widehat{\mathbf{n}})$. The excellent property of the *PPC* approach is that the variance of $P(x|\mathbf{n},\widehat{\mathbf{n}})$ turns out to be $\sigma_3^2 \propto g + \widehat{g}$ or alternatively $1/\sigma_3^2 = 1/\widehat{\sigma}^2 + 1/\sigma^2$ where σ^2 and $\widehat{\sigma}^2$ are the variances of $P(x|\mathbf{n})$ and $P(x|\widehat{\mathbf{n}})$ respectively. This follows from our observation that the variance of $P(x|\mathbf{n})$ was $\propto g$ and from the log-additivity of the PPC formulation (see next slide): this is nothing more profound than the outputs of the third population being the integral of the two sensing populations.

Cue Combination with PPC II

It can be shown that the PPC approach (where cue-combination closely matches what we observe in experiment) doesn't just apply for the specific model we constructed but to a larger class (see the review on 'Probabilistic brains') and requires only that $P(\mathbf{n}_3|x) = P(\mathbf{n} + \widehat{\mathbf{n}}|x) \propto P(\mathbf{n}|x)P(\widehat{\mathbf{n}}|x)$ where $\mathbf{n}_3$ is the number of counts from the third population.

Practical

Implement the sensor array that we described earlier using the Gillespie algorithm.
If you have time consider cue-combination.

[1] D. J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Cambridge University Press, 2003.

[2] A. Gelman et al. Bayesian Data Analysis 2nd ed. Chapman and Hall 2003.