Drug Bandits

Hannah Bruce Macdonald, Chodera Lab

November 2018

1 Introduction

Computationally chemistry methods can be applied to drug design projects to indicate which molecules are likely to have the highest affinity to, or the best selectivity for a given target. Computational free energy methods able to calculate free energies to within 2 kcal·mol^{-1} would speed lead optimization three-fold[**?**]. For free energy calculations to be used in active drug design projects, the results should be achievable on reasonable time-scales.

Drug bandits are currently written to prioritize calculation of the most soluble molecules of a set of eight substituted benzene molecules.

1.1 Bayesian bandits

A Bayesian bandit problem is where, given N slot machines all of unknown reward-rates, how can the maximum reward be achieved given limited resources. Early on, it is better to try all of the machines, given that little is known about each, but as more plays are made, and the reward-rate is better understood for each machine, then the machines with better reward rate should be played more frequently. When to shift from playing to 'learn' to playing to 'gain' is known as the explore-exploit tradeoff.

A Bayesian Bandit starts with a *prior* for each slot machine (or drug molecule) and then samples a random variable from all of the priors, picking the maximum. The chosen slot machine is then pulled, and the prior is then updated based on the reward of the slot machine.

- Sample random variable from N bandit's priors ($x_1 ... x_N$)
- Select a bandit
- Pull bandit n, by sampling from it's likelihood
- Update prior accordingly

This protocol is repeated, and over time, the prior better matches the likelihood. The bandit with the highest reward is increasingly likely to be chosen, as the prior improves. A consequence of this is that the prior of the highest-rewarding bandit will be the most accurate.

1.2 Bayesian bandit for drug design

If each bandit is considered to be a ligand system, the free energy quantity (FE) can be considered as the reward of the bandit. For a protein-ligand system, the binding free energy would be the reward, or for a solvated ligand, the hydration free energy can be considered.

The *prior* is the expected FE of the system, where a pull of a given bandit would involve sampling from the likelihood of the FE, through a short simulation, where the calculated FE can be used to update the *prior*. The priors improve through subsequent pulls to better match the real simulated FE's, and future pulls will prioritize those with maximum reward — i.e those with the highest affinity for a target. Initially, the protocol will be designed to prioritize solubility in a set of ligands.

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} \tag{1}$$

where herein the likelihood, the FE of a molecule, is a Gaussian of unknown expectation (μ) and unknown variance (σ^2);

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

where the conjugate prior is the normal-inverse gamma distribution;

$$f(\mu, \sigma^2|\nu, \lambda, \alpha, \beta) = \frac{\sqrt{\lambda}}{\sigma\sqrt{2\pi}} \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left[\frac{2\beta + \lambda(x-\nu)^2}{2\sigma^2}\right] \tag{3}$$

which has 4 hyper-parameters.

Generating priors

A prior needs to be chosen for each bandit. Given that the mean values from the normal-inverse gamma function can be afforded using;

$$E(\mu) = \nu \tag{4a}$$

$$E(\sigma^2) = \frac{\beta}{\alpha - 1}; \quad \alpha > 1 \tag{4b}$$

The prior values of the normal-inverse gamma function ($\nu_0, \lambda_0, \alpha_0, \beta_0$) could be chosen to generate very broad normal distribution centered on 0 for all ligands, or a reasonable guess of the centre and distribution of the gaussian could be made following an initial simulation. Any initial value of these should converge to the same posterior, as the influence of the prior is diminishing, as shown in Equations 8 (*unsure if this is true?*), but a reasonable initial guess should speed convergence.

Currently running 2000 steps of SAMS, and using the FE as ν_0, and the error in the FE as β_0. $_0 = 1$ and $\alpha_0 = 2$ have been chosen arbitrarily, but these need to be optimised.

Sampling bandits

Bandits are sampled by sampling a random variable from their posteriors, and then selecting one bandit to 'pull'.

Generating a random variate from a normal-inverse gamma function, the variance (σ^2) is sampled from in inverse gamma function:

$$f(\sigma^2|\alpha, \lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left[-\frac{\lambda}{\sigma^2}\right] \tag{5}$$

In practise random variates can be sampled from an inverse gamma function using;

```
scipy.stats.invgamma(a,scale=lambd)
```

The expectation can then be sampled from a normal distribution:

$$\mathcal{N}(\nu, \frac{\sigma^2}{\lambda}) \tag{6}$$

Once each of the bandits has been sampled, there exists several methods to select the bandit to sample.

ϵ-greedy

With probability ϵ select a bandit at random, and with probability $1 - \epsilon$, pick the bandit argmax($x_1 ... x_N$). This method samples any two sub-optimal arms with the same frequency.

Boltzmann

Picking a bandit proportional to it's expected reward, x_i

$$P_i(n) = \frac{e^{\frac{x_i}{\tau}}}{\sum_{j=1}^{N} e^{\frac{x_j}{\tau}}} \tag{7}$$

where τ is a temperature parameter that controls the randomness of the choice. This method will sample any arm proportional to its expectation.

'Pulling' bandit

Pulling a given bandit involves generating a FE of the system. For testing, this can be done by generating a random variable from a normal distribution of the experimental FE and uncertainty, or by performing a short simulation.

The simulation protocol needs optimisation — should the 'pull' extend the previous simulation, or start from a random starting point? How many steps should a 'pull' involve? How should SAMS/repex be used?

The pull of the bandit results in a FE, which can be used to update the posterior

Updating posterior

The normal-inverse gamma function has four hyper-parameters, ν, λ, α and β. Each of these values after n pulls of the given bandit, where y_0 is the prior value. The FE result of the i^{th} pull is x_i, with the average over n pulls is $\overline{x}$.

$$\lambda_n = \lambda_0 + n \tag{8a}$$

$$\nu_n = \frac{\nu_0 \lambda_0 + n\overline{x}}{\lambda_n} \tag{8b}$$

$$\alpha_n = \alpha_0 + \frac{n}{2} \tag{8c}$$

$$\beta_n = \beta_0 + \frac{n\lambda_0}{\lambda_n} \frac{(\overline{x} - \nu_0)^2}{2} + \frac{1}{2} \sum_{i=1}^{n} (x_i - \overline{x})^2 \tag{8d}$$

The updated posterior can be related back to a normal distribution using the means defined in Equation 4.

2 TODO

Difficulty indicated by *'s

- Sort out optimal protocol for 'pull'. Length of simulation, SAMS/repex, etc. *
- FE rewards are unbound, whereas BB assume that the rewards are bound **
- how informed should the priors be? *
- look into the 'regret' of the process **
- currently using Thomspson sampling mixed with random sampling, need to ensure that this is optimal **

- implement purely in openmmtools *
- integrate relative free energies into protocol ***