Bayesian Optimisation with a Probabilistic Model
(in Python)

Ross Tooley
Motivation
Inspired by ProBO

ProBO: Versatile Bayesian Optimization Using Any Probabilistic Programming Language

Coming soon. Read the paper [here](#).

For questions, please email willie@cs.cmu.edu.
BO Recap

Bayesian Optimisation

Model

Next Input

Expensive Objective Function

Next Output

observation (x)

objective fn (f(·))

acquisition (·)

acquisition function (u(·))

acquisition max
ProBO Recap

ProBO

Probabilistic Model

Next Input

Expensive Objective Function

Next Output

observation (x)

objective fn \( f(\cdot) \)

acquisition max

acquisition function \( u(\cdot) \)
Algorithm 1 ProBO\((\mathcal{D}_0, \text{inf}, \text{gen})\)

1: for \(n = 1, \ldots, N\) do
2: \hspace{1em} post \leftarrow \text{inf}(\mathcal{D}_{n-1})
3: \hspace{1em} x_n \leftarrow \operatorname{argmin}_{x \in \mathcal{X}} a(x, \text{post}, \text{gen})
4: \hspace{1em} y_n \sim s(x_n)
5: \hspace{1em} \mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup (x_n, y_n)
6: Return \(\mathcal{D}_N\).

Algorithm 2 \(a_{\text{EI}}(x, \text{post}, \text{gen})\) \hspace{1em} \triangleright \text{EI}

1: for \(m = 1, \ldots, M\) do
2: \hspace{1em} z_m \leftarrow \text{post}(s_m)
3: \hspace{1em} y_m \leftarrow \text{gen}(x, z_m, s_m)
4: \hspace{1em} f_{\text{min}} \leftarrow \min_{y \in \mathcal{D}} f(y)
5: \hspace{1em} \text{Return } \sum_{m=1}^{M} \mathbbm{1}[f(y_m) \leq f_{\text{min}}] (f_{\text{min}} - f(y_m))
Plan: implement pseudocode in Python
Enter BoTorch

BoTorch

BAYESIAN OPTIMIZATION IN PYTORCH
Enter BoTorch

Bayesian Optimization in PyTorch
Enter BoTorch... and Pyro
Enter BoTorch... and Pyro

New Plan:
1. Implement ProBO in Python
2. Implement in BoTorch + Pyro
3. Compare
Work so far

(very little)
Some BoTorch + Pyro code

class BO_Model(Model):
    def probabilistic_model(marriage=None, age=None, divorce=None):
        a = numpyro.sample('a', dist.Normal(0., 0.2))
        M, A = 0., 0.
        if marriage is not None:
            bM = numpyro.sample('bM', dist.Normal(0., 0.5))
            M = bM * marriage
        if age is not None:
            ba = numpyro.sample('ba', dist.Normal(0., 0.5))
            A = ba * age
        sigma = numpyro.sample('sigma', dist.Exponential(1.))
        mu = a + M + A
        numpyro.sample('obs', dist.Normal(mu, sigma), obs=divorce)
    
    def condition_on_observations(X, Y):
        # Start from this source of randomness.
        rng_key = random.PRNGKey(0)
        rng_key, rng_key_ = random.split(rng_key)
        # Num samples
        num_warmup, num_samples = 1000, 2000
        # Run NUTS.
        kernel = NUTS(probabilistic_model)
        mcmc = MCMC(kernel, num_warmup, num_samples)
        mcmc.run(rng_key, marriage=X, divorce=X)
        mcmc.print_summary()
        self.samples_1 = mcmc.get_samples()

    def posterior(X):
        rng_key, rng_key_ = random.split(rng_key)
        predictive = Predictive(probabilistic_model, self.samples_1)
        predictions = predictive(rng_key_, marriage=X)['obs']
        return lambda () -> np.random.choice(predictions)
Work to do

(everything else)
Q&A

Thanks for listening!