Optimization and Sampling

Goals

- Dual nature of **optimization** (differentiation) and **sampling** (integration) as core approaches to posterior inference.
  - Optimization: approximate the posterior by finding maximum/minimum, typically involving differentiation.
  - Sampling: draw samples from the posterior and approximate the integration.

Sampling: model selection

Many tasks in probabilistic machine learning boil down to computing an expectation (integral).

Marginal likelihood for model selection:

$$p(D|M) = \int p(D|\theta_M, M)p(\theta_M|M)d\theta_M.$$

Sampling: prediction

Predictive distribution:

$$p(D^*|D) = \int p(D^*|\theta)p(\theta|D)d\theta.$$

Sampling: hypothesis testing

p-values:

$$P(\theta > \theta_0|D) = \int_{\theta_0}^{\infty} p(\theta|D)d\theta.$$

Sampling: other examples

- Moments (mean, variance, skewness, kurtosis)
- EM-algorithm (expectation to marginalize out the latent variables)
- CDFs and quantiles

they all involve computing expectation.

Monte Carlo integration: idea

Let X be a RV with pdf given by p. For function g, we aim to compute

$$\mathbb{E}_{X\sim p}[g(X)] = \int g(x)p(x)dx.$$

1. Sample $x_i \sim p$ for $i = 1, ..., N$.
2. Approximation: $I = \int g(x)p(x)dx \approx \frac{1}{N}\sum_i g(x_i) = \hat{I}$.

Why is this valid?

Monte Carlo integration: unbiased

It is easy to show that

$$\mathbb{E}[\hat{I}] = I.$$

But does it concentrate around I as we increase N?

Monte Carlo integration: WLLN

(Weak law of large numbers) If $X_n \sim p$ i.i.d. and $\mathbb{E}[X_n] = \mu$, then

$$\frac{1}{N}\sum_i x_i \to_p \mu$$

as $N \to \infty$.

We can apply the LLN to $g(X_n)$ to show that it converges to $\mathbb{E}[g(X)]$.

Note: the assumptions are fairly weak, $\mathbb{E}[|X|] < \infty$ and that g be integrable.

Monte Carlo integration: basic properties

- Unbiased: $\mathbb{E}[N^{-1}\sum_i g(X_i)] = \mathbb{E}[g(X)]$.
- Consistency by LLN: we get closer to $\mathbb{E}[g(X)]$ as N increases.
- Rate of convergence: $\sqrt{\frac{1}{N}\text{var}[g(X)]} = O(1/\sqrt{N})$.
  - Rough translation: to obtain k decimal point accuracy, we need $N = 10^{2k}$.

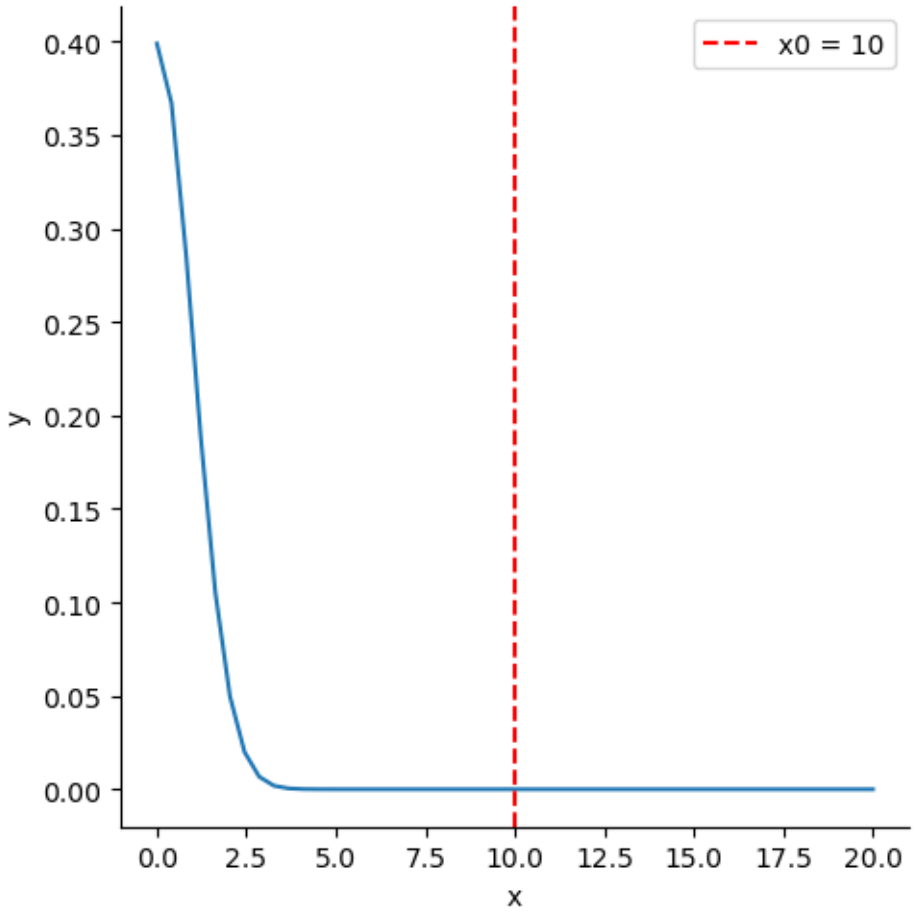
Importance sampling: tail probability

Let $X \sim N(0, 1)$. Estimate the tail probability:

$$P(X > x_0),$$

for some large x_0.

Importance sampling: tail probability plot



Compute $P(X > 10)$?

Importance sampling: motivation 1

```
import scipy
x0 = 10
y = scipy.stats.norm.pdf(x0, 0, 1)
print(y)
prob = 1 - scipy.stats.norm.cdf(x0, 0, 1)
print(prob)
```

```
7.69459862670642e-23
0.0
```

- Density is non-zero but CDF returns zero.

Importance sampling: motivation 2

$$P(X > x_0) = \mathbb{E}[1[X > x_0]] \tag{1}$$

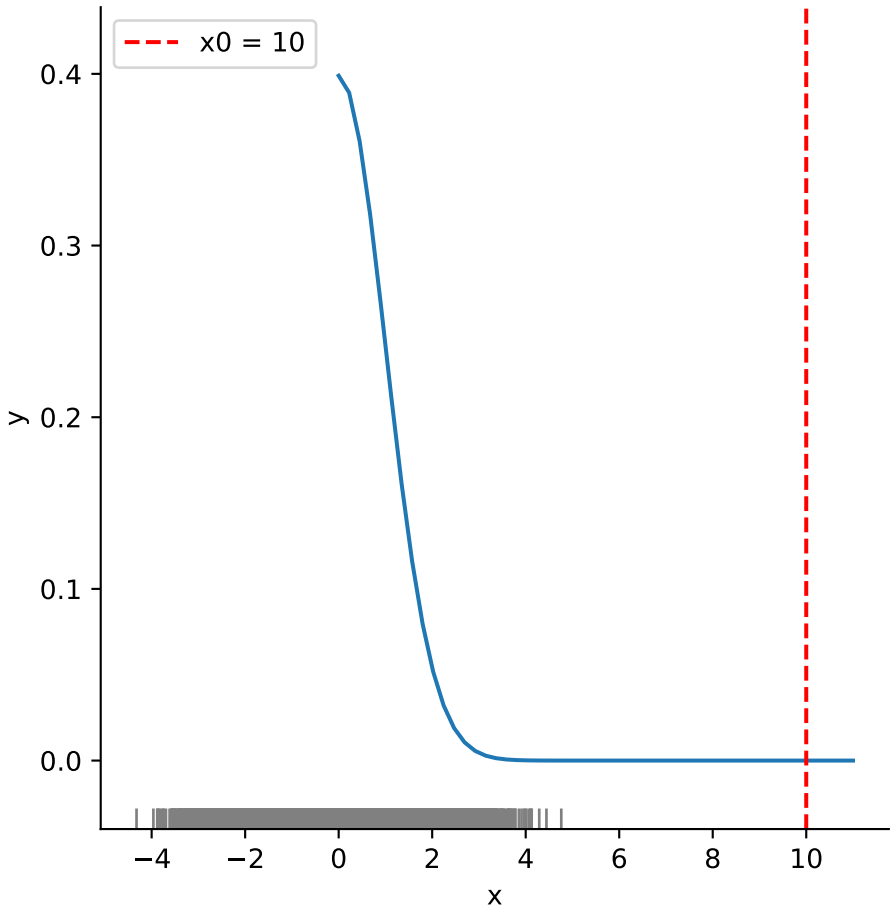
$$= \int 1[X > x_0]\phi(x)dx, \tag{2}$$

where $\phi(x)$ denotes standard Normal PDF. Let's use Monte Carlo sampling:

$$\mathbb{E}[1[X > x_0]] \approx \frac{1}{N}\sum_i 1[x_i > x_0]. \tag{3}$$

Importance sampling: MC estimate fails

```
Number of samples > 10: 0
```



Importance sampling: idea

$$\mathbb{E}[g(X)] = \int g(x)f(x)dx \tag{4}$$

$$= \int g(x)\frac{f(x)}{h(x)}h(x)dx \tag{5}$$

$$= \int g(x)w(x)h(x)dx. \tag{6}$$

Rather than sampling from $X_i \sim f$, we sample from more convenient distribution with density function h.

Condition: $h(x) > 0$ where $f(x) > 0$

Importance sampling: does it work?

```
x = np.random.normal(size=10000, loc=x0, scale=1)
weights = scipy.stats.norm.pdf(x, 0, 1) / scipy.stats.norm.pdf(x, x0, 1)
print(np.mean(weights * (x > x0)))
```

```
7.512162906570435e-24
```

How can we verify that this is correct?

```
true_val = scipy.stats.norm.sf(x0)
print(f"True value: {true_val}")
```

```
True value: 7.61985302416047e-24
```

Importance sampling: how to choose a proposal

- $f(x) > 0 \Rightarrow h(x) > 0$.
- $\mathbb{E}_{X\sim h}[g(X)f(X)/h(X)]$ to be defined.
- Variance of the weights should be finite.

Importance sampling: variance of the weights

The variance of the importance estimator is finite only when

$$\mathbb{E}_{X\sim h}\left[g^2(X)\frac{f^2(x)}{h^2(x)}\right] < \infty.$$

- We need the ratio f/h to be bounded, so $f(x) < Mh(x)$ for some $M > 0$.
- More specifically, we want $h(x)$ to match the shape of $g(x)f(x)$ reasonably well where $g(x)f(x)$ has high density.
- The optimal proposal h is proportional to $|g(x)f(x)|$.

Importance sampling: variance of the weights

Effective sample size:

$$\text{ESS} = \frac{\left(\sum_n w_n\right)^2}{\sum_n w_n^2}.$$

- ESS roughly measures how many i.i.d. samples from the target we would effectively have if we used direct sampling from f.
- High variance in the weights means few samples dominate the approximation.

Optimization

$$\theta^* = \text{argmin} L(\theta).$$

- Continuous optimization: $\theta \in \Theta \subseteq \mathbb{R}^D$.
- We will assume that L is "smooth": continuously differentiable.
  - If not smooth, there is a way around this problem by decomposing the function over its domain (smooth part vs non-smooth part).
- Examples: mean squared error function or posterior distribution function involving continuous-valued parameters.

Global optimization

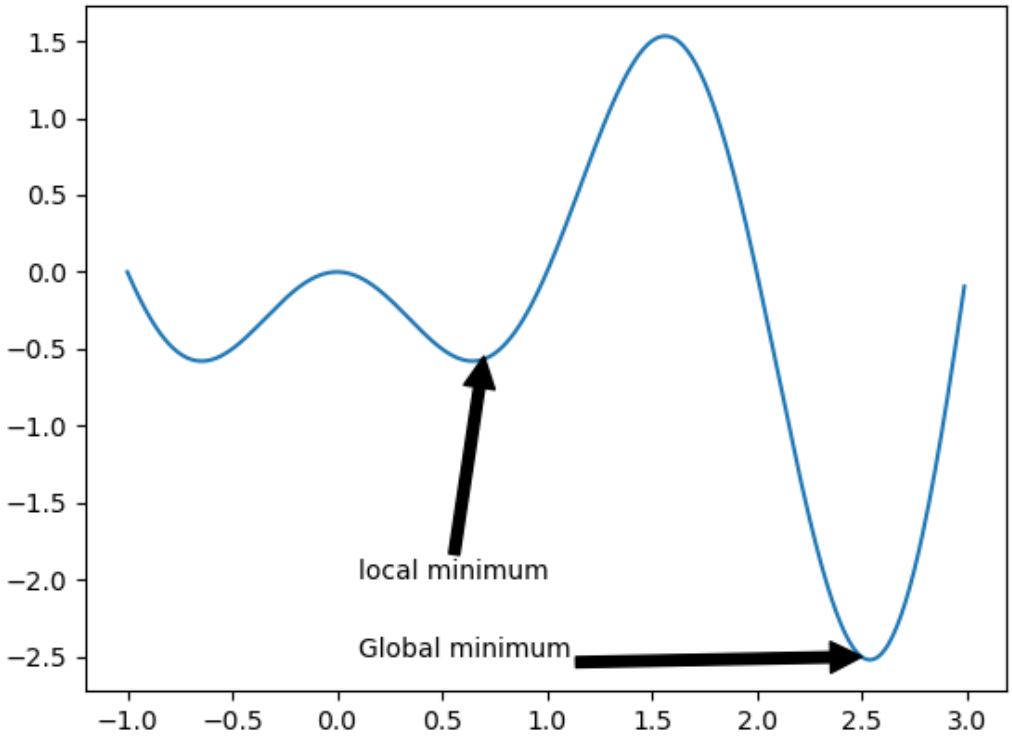
- θ^* is a global minimum.
- Algorithm to solve the optimization problem is called a solver.
- Most problems and solvers do not guarantee global optima, except in special cases (e.g., convex problems).

Local optimization

θ^* is a local minimum if

$$\exists \delta > 0, \forall \theta \in \Theta \text{ s.t. } \|\theta - \theta^*\| < \delta \Rightarrow L(\theta^*) \leq L(\theta)$$

Local vs global optimization



Source: extreme_fig_1d.ipynb

Gradient and Hessian

Let $g(\theta) = \nabla L(\theta)$ be the gradient vector of L.

Let $H(\theta) = \nabla^2 L(\theta)$ denote the Hessian matrix.

Let $g^* = g(\theta)\Big|_{\theta*}$ and $H^* = H(\theta)\Big|_{\theta^*}$.

Necessary and sufficient conditions of local optima

If θ^* is a local optimum, then

- $g^* = 0$ and

- H^* is positive semi-definite.

If $g^* = 0$ and H^* is positive definite, then θ^* is a local optimum.

Note: zero gradient alone is not sufficient, since we could have a saddle point.

PSD ensures $L(\theta^*) \leq L(\theta)$.

First-order method

Class of algorithms that utilize the gradient vector.

1. Initialize algorithm: set $\theta = \theta_0$.
2. At each iteration t: update $\theta_{t+1} = \theta_t + \eta_t d_t$:
    - η_t: learning rate or step size
    - d_t descent direction.
3. Repeat until stationary point is reached (i.e., gradient is zero).

The descent direction is given by $d_t = -g_t$. Recall: gradient points to the direction of maximal increase.

Second-order method

Utilize gradient and Hessian to find the optima.

(Newton's method) At each iteration t:

$$\theta_{t+1} = \theta_t + \eta H_t^{-1} g_t.$$

Because the Hessian encodes the local curvature of the function, multiplying the gradient by H^{-1} "preconditions" the search direction and counteracts the function's curvature, yielding more direct steps toward the optimum.

Stochastic optimization

The goal is to minimize an expected loss with respect to some random variable z:

$$L(\theta) = \mathbb{E}_{z \sim q}[L(\theta, z)].$$

Example: Monte Carlo Expectation Maximization where the expectation is intractable.

Stochastic gradient descent

We need the gradient of the expectation of the loss function:

$$g_t = \nabla \mathbb{E}_{z_t \sim q}[L(\theta, z_t)].$$

Update:

$$\theta_{t+1} = \theta_t - \eta_t g_t.$$

Stochastic gradient descent: finite sum

$$L(\theta_t) = \frac{1}{N} \sum_{n=1}^{N} l(y_n, f_\theta(x)).$$

- In fitting a large scale model with large data, evaluating the gradient of $L(\theta_t)$ can be time consuming.
- Sample a minibatch B_t of size $B \ll N$:

$$g_t = \frac{1}{B} \sum_{n \in B_t} \nabla l(y_n, f_\theta(x)).$$

Learning rate

To ensure convergence of stochastic gradient descent the learning rate schedule needs to satisfy Robbins-Monro conditions:

$$\eta_t \to 0,$$

and

$$\frac{\sum \eta_t^2}{\sum \eta_t} \to 0,$$

or $\sum \eta_t \to \infty$ and $\sum \eta_t^2 \to 0$.

Learning rate

- Inverse decay $\eta_t = \eta_0/(1 + \alpha t)$
- Exponential decay $\eta_0 \exp(-\alpha t)$ for $\eta_0 > 0, \alpha > 0$.

But these may not be practical choices for many problems.

Momentum

Take larger steps in directions of continued movements; slow down when the gradients change abruptly.

$$m_t = \beta m_{t-1} + g_{t-1} \tag{7}$$
$$\theta_t = \theta_{t-1} + \eta_t m_t, \tag{8}$$

$0 < \beta < 1$, typically $\beta \approx 0.9$.

SGD algorithms

- AdaGrad
- RMSProp
- Adam

None of these methods satisfy Robbins-Monro and therefore, are not guaranteed to converge. However, they work well in practice.

Combining ideas from optimization for sampling

- Monte Carlo Expectation Maximization:
  - Intractable E-step is replaced by Monte Carlo E-step,
  - Maximize model parameters θ in the M-step.
- Variational inference:
  - draw samples from a variational distribution q_ψ, optimize ψ to be close to the posterior p.
- Non-reversible MCMC:
  - Hamiltonian Monte Carlo and Bouncy Particle Sampler: uses gradients to explore the sample space.