ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Bayesian Machine Learning

May 2022 - François HU
https://curiousml.github.io/

Outline

1 Bayesian statistics

2 Latent variable models

3 Variational Inference

4 **Markov Chain Monte Carlo**
- Monte Carlo Estimation
- MCMC and differences with VI

5 Extensions and oral presentations

1 Monte Carlo estimation

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X]$$

$$\mathbb{E}[h(X)]$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

$$\mathbb{P}(X > 2)$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1]$$

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X] \approx \bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

$$\mathbb{E}[h(X)]$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

$$\mathbb{P}(X > 2)$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1]$$

1. Monte Carlo estimation
Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X] \approx \bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

$$\mathbb{E}[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(x^{(i)})$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

$$\mathbb{P}(X > 2)$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1]$$

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X] \approx \bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

$$\mathbb{E}[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(x^{(i)})$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] \approx \frac{1}{n}\sum_{i=1}^{n} \left(x^{(i)} - \bar{x}^{(n)}\right)^2$$

$$\mathbb{P}(X > 2)$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1]$$

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X] \approx \bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

$$\mathbb{E}[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(x^{(i)})$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] \approx \frac{1}{n}\sum_{i=1}^{n} \left(x^{(i)} - \bar{x}^{(n)}\right)^2$$

$$\mathbb{P}(X > 2) = \mathbb{E}[1_{\{X>2\}}] \approx \frac{1}{n}\sum_{i=1}^{n} 1_{\{x^{(i)}>2\}}$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1]$$

1. Monte Carlo estimation

Classic estimation methods

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Example : Let us denote $\mathbf{x} = (x^{(1)}, x^{(2)}, \cdots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\mathbb{E}[X] \approx \bar{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

$$\mathbb{E}[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(x^{(i)})$$

$$V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] \approx \frac{1}{n}\sum_{i=1}^{n} \left(x^{(i)} - \bar{x}^{(n)}\right)^2$$

$$\mathbb{P}(X > 2) = \mathbb{E}[1_{\{X>2\}}] \approx \frac{1}{n}\sum_{i=1}^{n} 1_{\{x^{(i)}>2\}}$$

$$\pi = 4 \times \mathbb{P}[U^2 + V^2 \leq 1] \approx \frac{4}{n}\sum_{i=1}^{n} 1_{\{(u^{(i)})^2+(v^{(i)})^2\leq 1\}} \text{ for } u^{(i)}, v^{(i)} \sim \mathcal{U}(0,1)$$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y \mid x, X^{(train)}, Y^{(train)})$$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) \;=\; \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta$$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) \; = \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta \; = \; \mathbb{E}_{P(\theta|X^{(train},Y^{(train)})} P(y\,|\,x,\theta)$$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y \mid x, X^{(train)}, Y^{(train)}) = \int_\theta P(y \mid x, \theta) \cdot P(\theta \mid X^{(train}, Y^{(train)}) \cdot d\theta = \mathbb{E}_{P(\theta \mid X^{(train}, Y^{(train)})} P(y \mid x, \theta)$$

with $P(\theta \mid X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)} \mid X^{(train)}, \theta) \cdot P(\theta)}{\text{constant}}$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) = \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta = \mathbb{E}_{P(\theta|X^{(train},Y^{(train)})} P(y\,|\,x,\theta)$$

with $P(\theta\,|\,X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)}\,|\,X^{(train)},\theta)\cdot P(\theta)}{\text{constant}}$

easy : model output + prior fixed by us

difficult : as always …

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) = \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta = \mathbb{E}_{P(\theta|X^{(train},Y^{(train)})}P(y\,|\,x,\theta)$$

with $P(\theta\,|\,X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)}\,|\,X^{(train)},\theta)\cdot P(\theta)}{\text{constant}}$

easy : model output + prior fixed by us

difficult : as always …

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n}\sum_{i=1}^{n} P(y\,|\,x,\theta_i)$$

if we can sample $\theta_1, \ldots, \theta_n \sim P(\theta\,|\,X^{(train)}, Y^{(train)})$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) = \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta = \mathbb{E}_{P(\theta|X^{(train},Y^{(train)})} P(y\,|\,x,\theta)$$

with $P(\theta\,|\,X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)}\,|\,X^{(train)},\theta)\cdot P(\theta)}{\text{constant}}$

easy : model output + prior fixed by us

difficult : as always …

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n}\sum_{i=1}^{n} P(y\,|\,x,\theta_i)$$

if we can sample $\theta_1, \ldots, \theta_n \sim P(\theta\,|\,X^{(train)}, Y^{(train)})$

Lecture 2 (and 3) : M-step of EM-algorithm

$$\max_\theta \mathbb{E}_T\left[\log P(X, T\,|\,\theta)\right]$$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) = \int_\theta P(y\,|\,x,\theta)\cdot P(\theta\,|\,X^{(train}, Y^{(train)})\cdot d\theta = \mathbb{E}_{P(\theta|X^{(train}, Y^{(train)})} P(y\,|\,x,\theta)$$

with $P(\theta\,|\,X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)}\,|\,X^{(train)},\theta)\cdot P(\theta)}{\text{constant}}$

easy : model output + prior fixed by us

difficult : as always …

$$P(y\,|\,x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n}\sum_{i=1}^{n} P(y\,|\,x,\theta_i)$$

if we can sample $\theta_1, \ldots, \theta_n \sim P(\theta\,|\,X^{(train)}, Y^{(train)})$

Lecture 2 (and 3) : M-step of EM-algorithm

$$\max_\theta \mathbb{E}_T\left[\log P(X, T\,|\,\theta)\right] \approx \frac{1}{n}\sum_{i=1}^{n} \log P(X, T_i\,|\,\theta)$$

if we can sample $T_1, \ldots, T_n \sim T$

1. Monte Carlo estimation

Motivation

Monte Carlo : Estimate an expected value by **sampling**. A naïve method would be approximating it by its empirical value

Why do we care ?

- Estimate a large spectrum of probabilistic models up to a normalization constant

Lecture 1 : Bayesian inference

$$P(y \mid x, X^{(train)}, Y^{(train)}) = \int_\theta P(y \mid x, \theta) \cdot P(\theta \mid X^{(train}, Y^{(train)}) \cdot d\theta = \mathbb{E}_{P(\theta \mid X^{(train}, Y^{(train)})} P(y \mid x, \theta)$$

with $P(\theta \mid X^{(train)}, Y^{(train)}) = \dfrac{p(Y^{(train)} \mid X^{(train)}, \theta) \cdot P(\theta)}{\text{constant}}$

easy : model output + prior fixed by us

difficult : as always …

$$P(y \mid x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n} \sum_{i=1}^{n} P(y \mid x, \theta_i)$$

if we can sample $\theta_1, \ldots, \theta_n \sim P(\theta \mid X^{(train)}, Y^{(train)})$

Lecture 2 (and 3) : M-step of EM-algorithm

$$\max_\theta \mathbb{E}_T \left[\log P(X, T \mid \theta)\right] \approx \frac{1}{n} \sum_{i=1}^{n} \log P(X, T_i \mid \theta)$$

if we can sample $T_1, \ldots, T_n \sim T$

Yes with usual simulations or MCMC

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathscr{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
>
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathcal{U}(0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

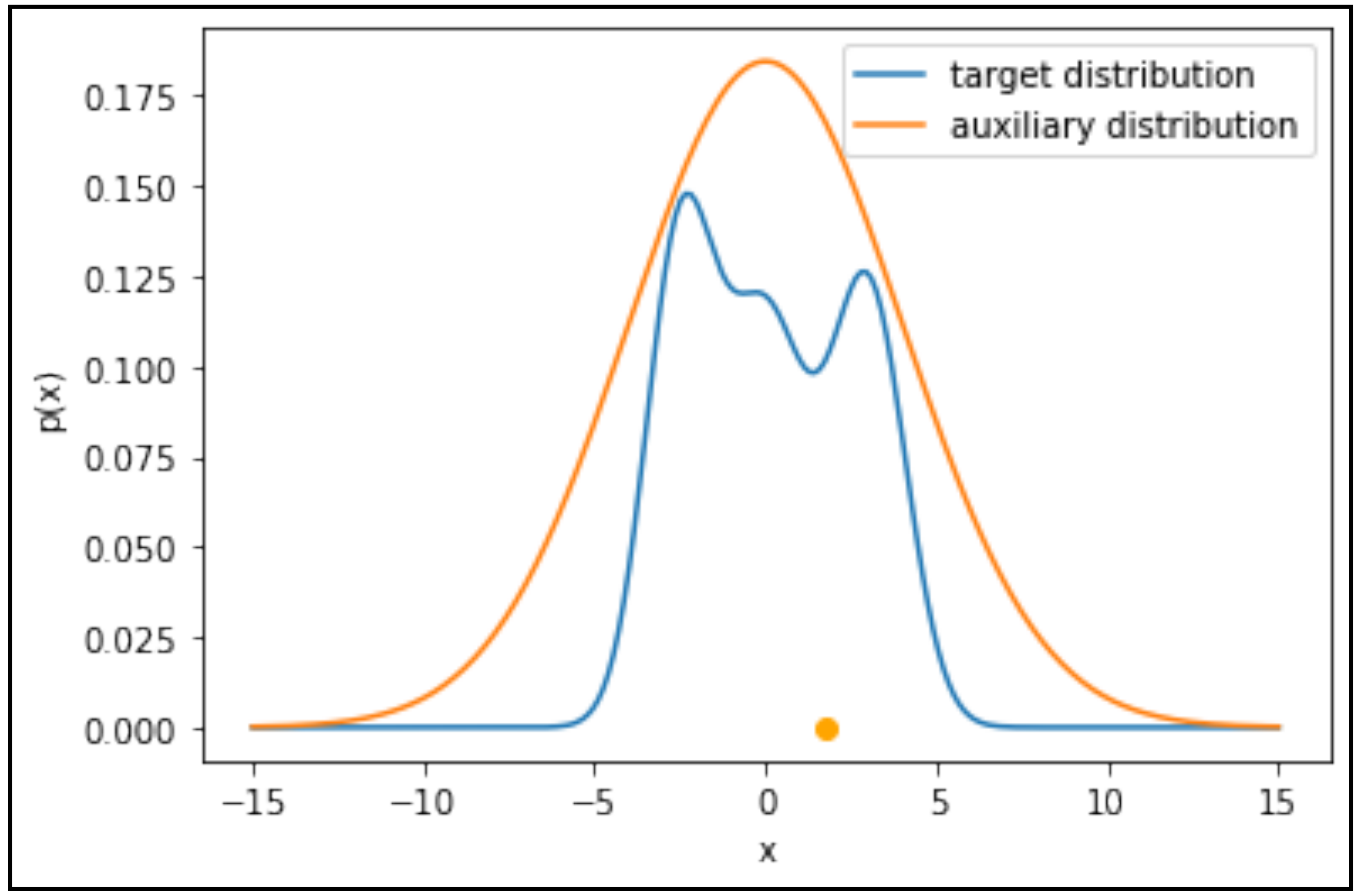
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathscr{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
>
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathscr{U}(0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



X_1

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathscr{U}(0,1)$

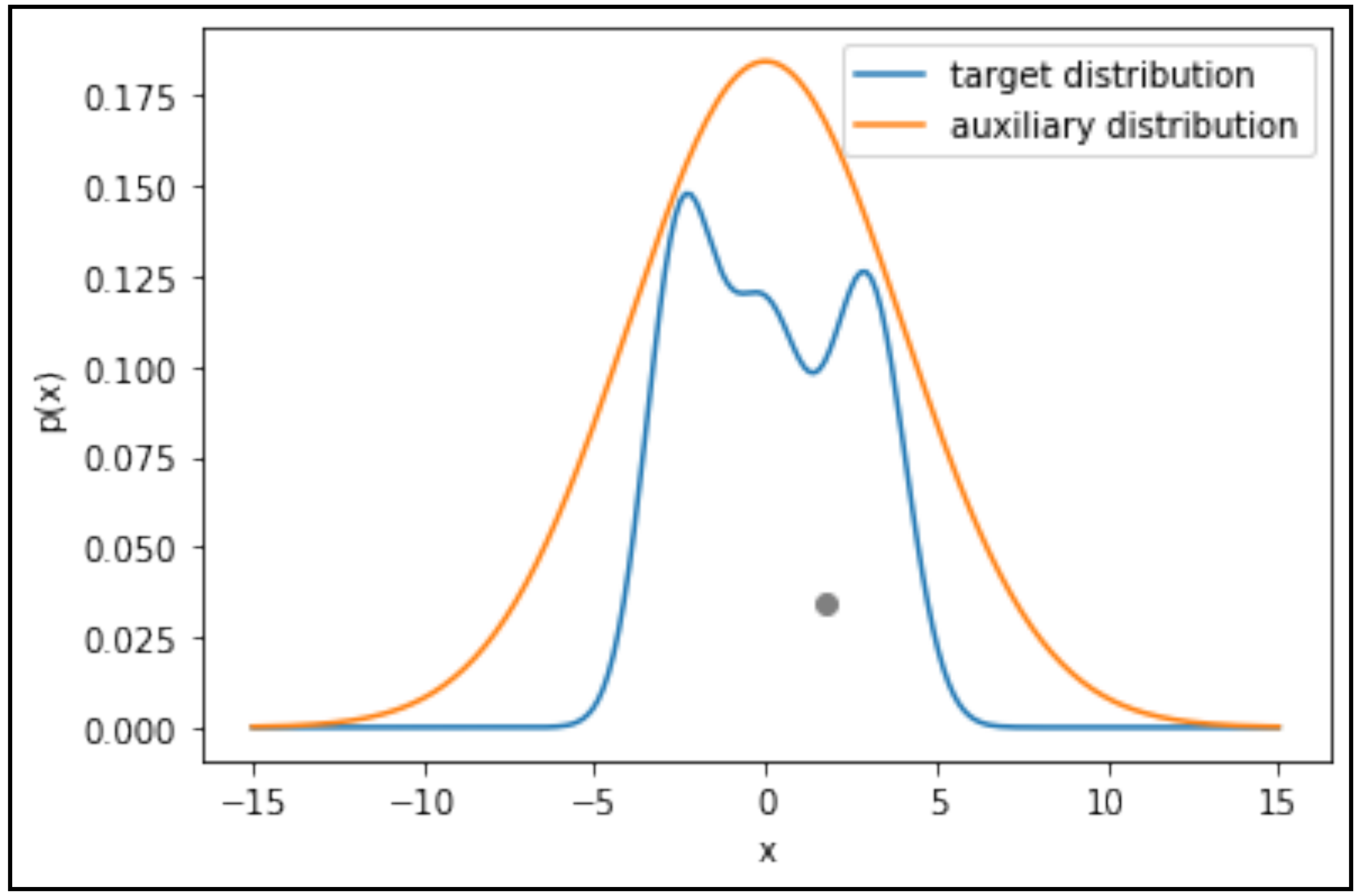
For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times \text{Q}$
>
> **Algorithm**
>
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathscr{U}(0, \text{const} \cdot \text{Q}(\text{x}_\text{i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.

X_1



1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

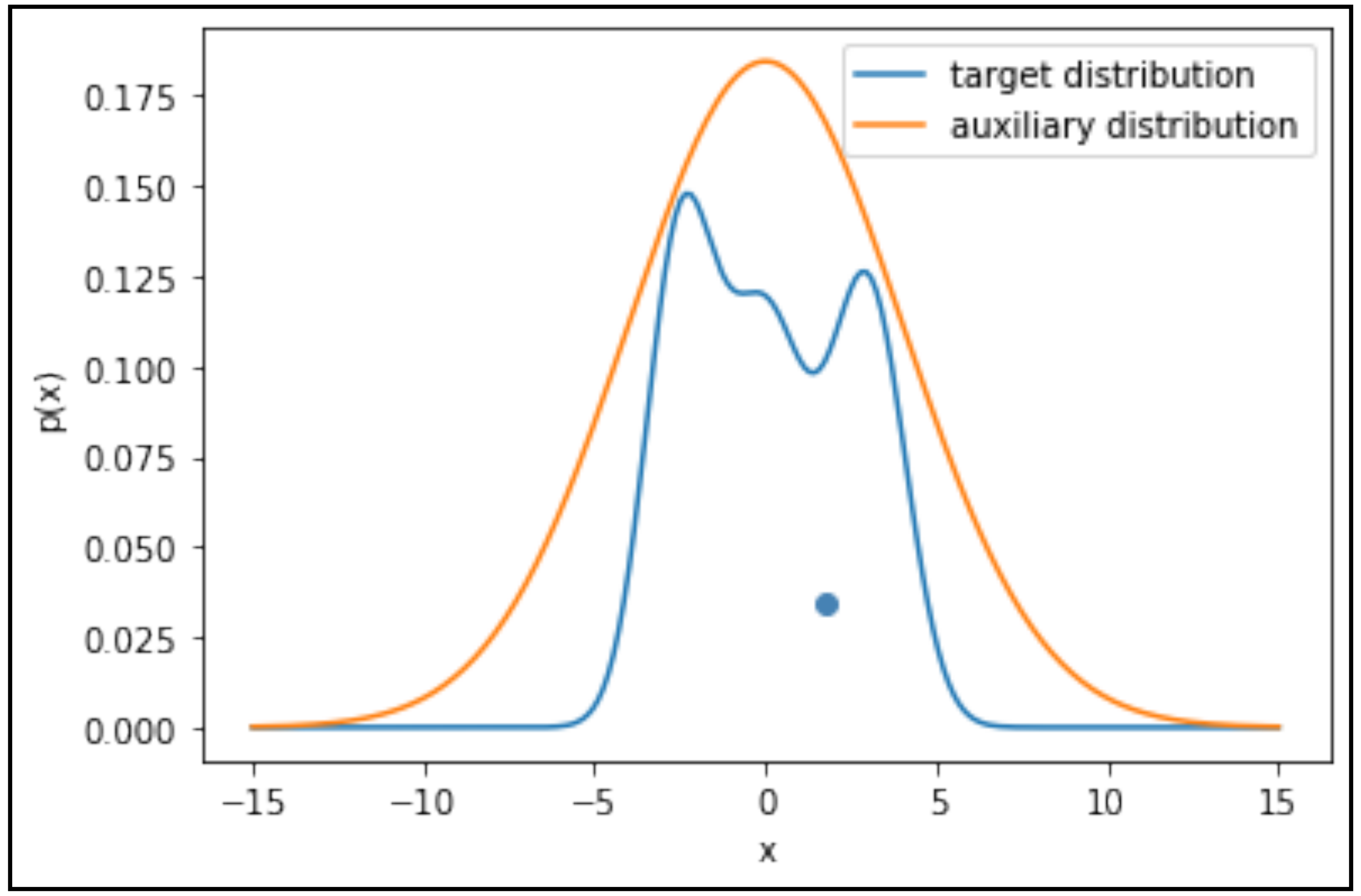
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times \text{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot \text{Q}(\text{x}_\text{i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



X_1

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

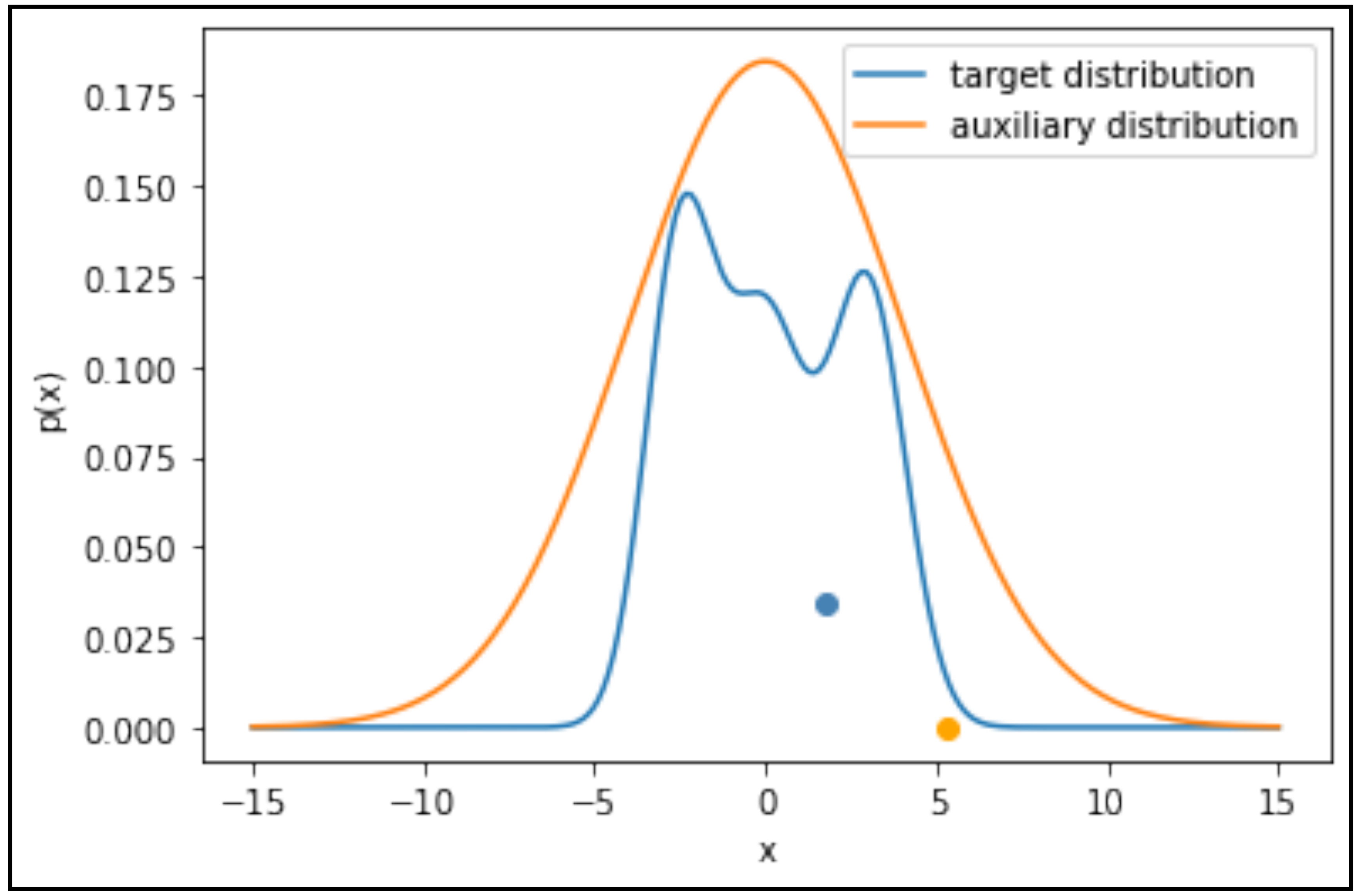
For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim \; Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \; \mathcal{U}(\,0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i})\,)$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.

X_1 X_2



1. Monte Carlo estimation
Usual simulations : the power of uniform distribution

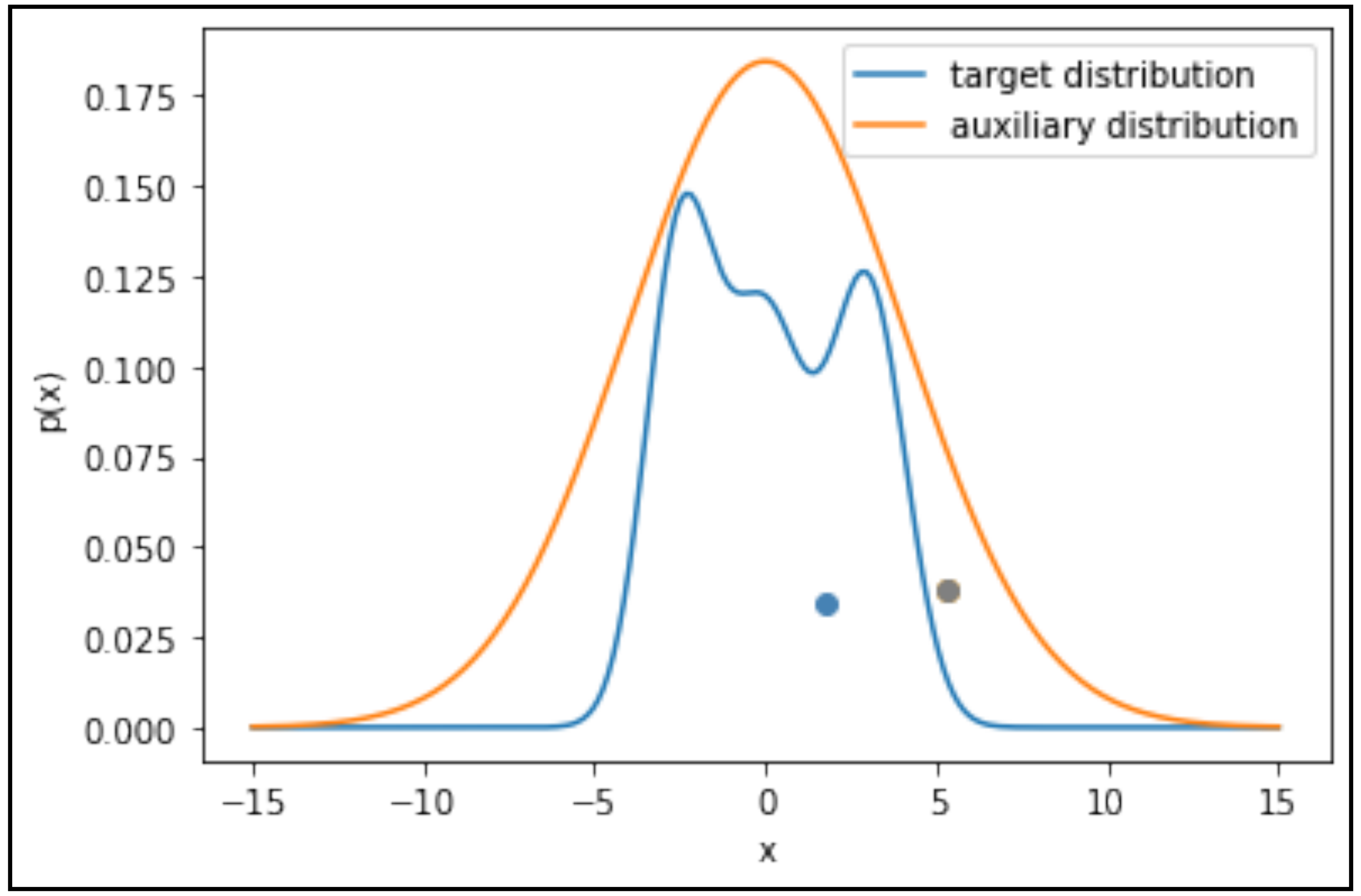
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathcal{U}(0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



X_1 X_2

1. Monte Carlo estimation
Usual simulations : the power of uniform distribution

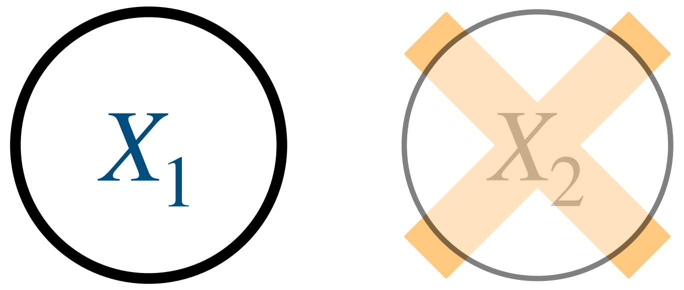
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathscr{U}(0,1)$

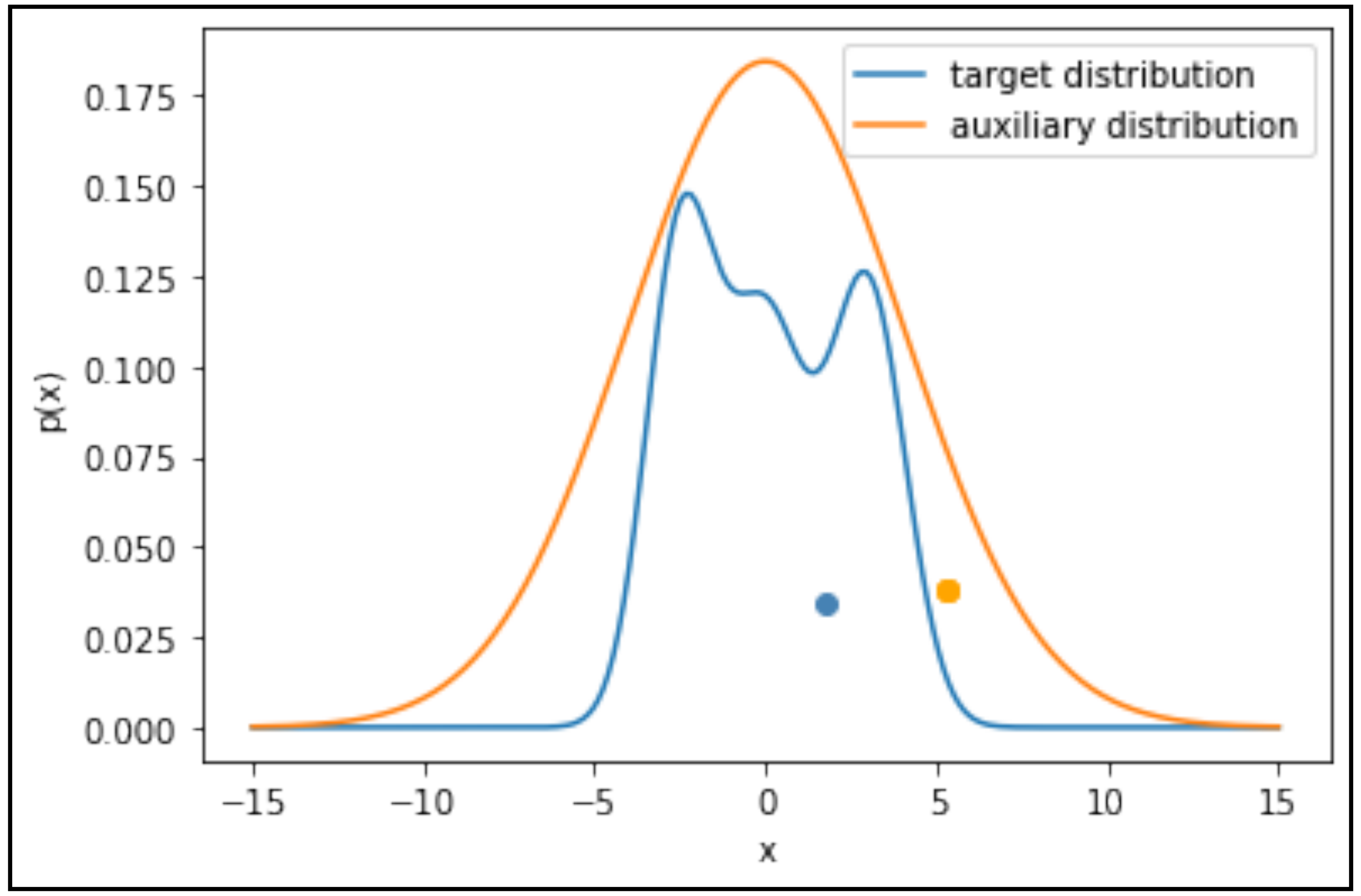
For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathscr{U}(0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.





1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

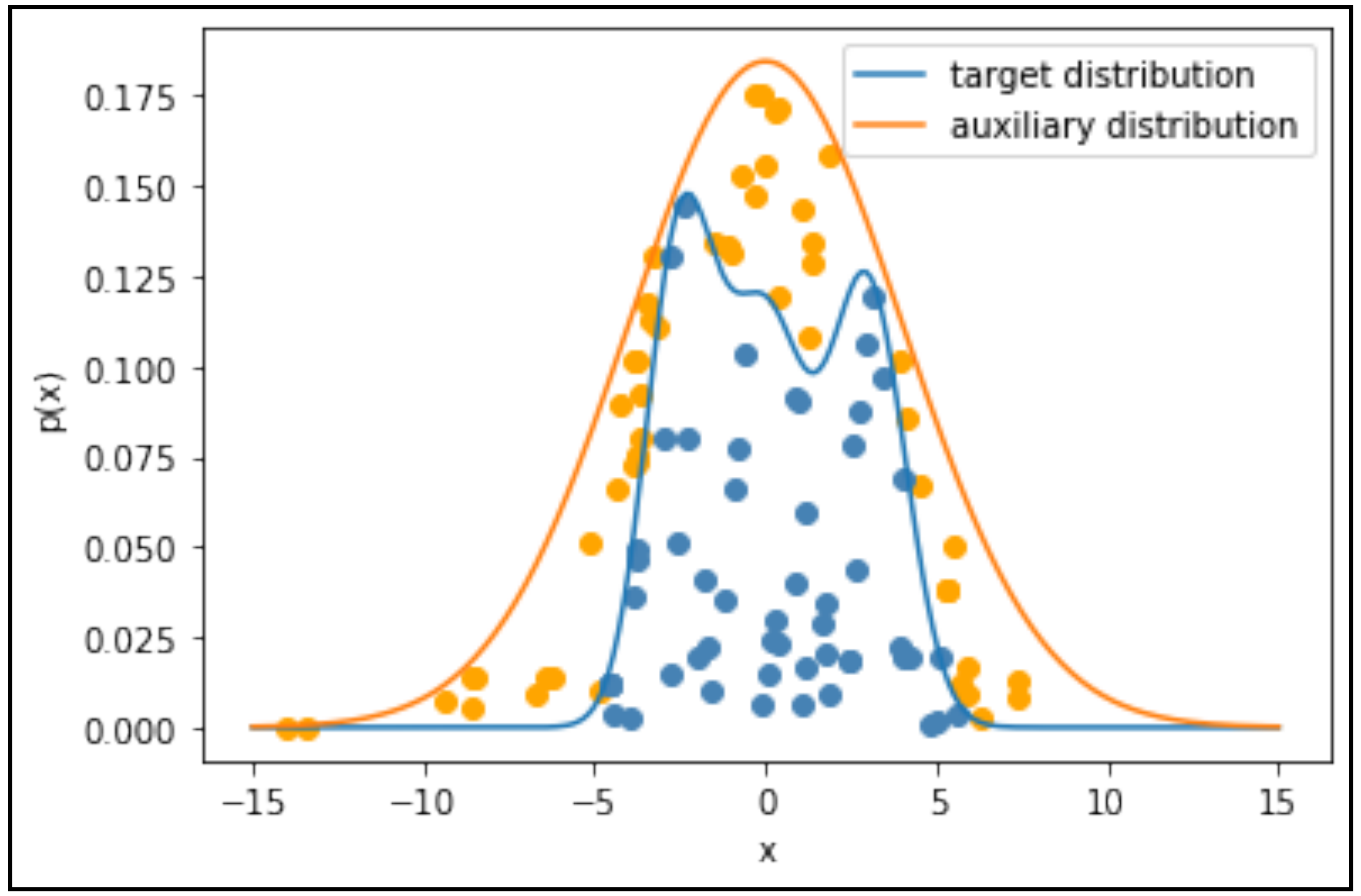
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times \text{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot \text{Q}(\text{x}_\text{i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathscr{U}(0,1)$

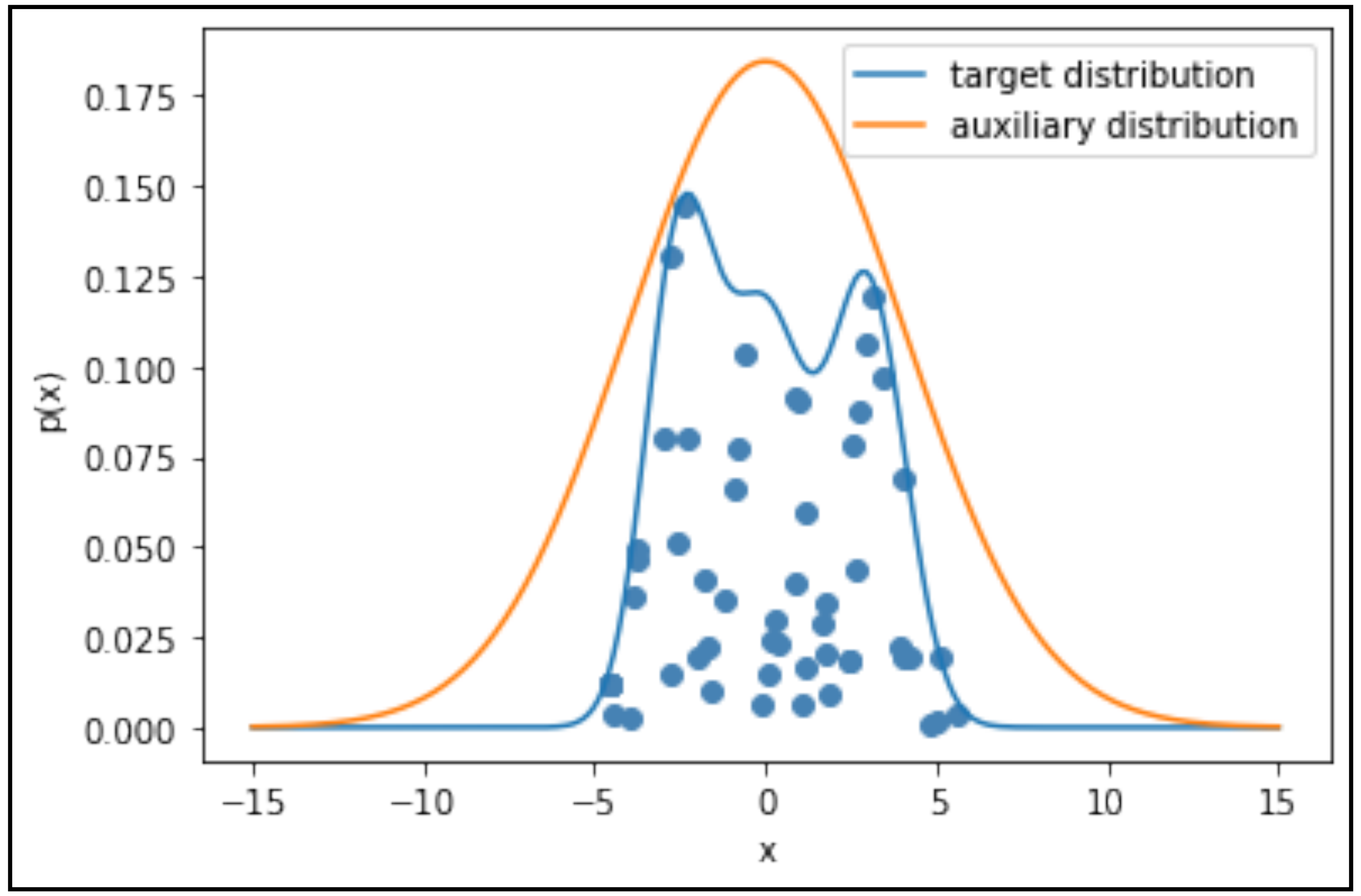
For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathscr{U}(0, \mathrm{const} \cdot \mathrm{Q(x_i)})$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.

X_1 X_2 ... X_n



1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

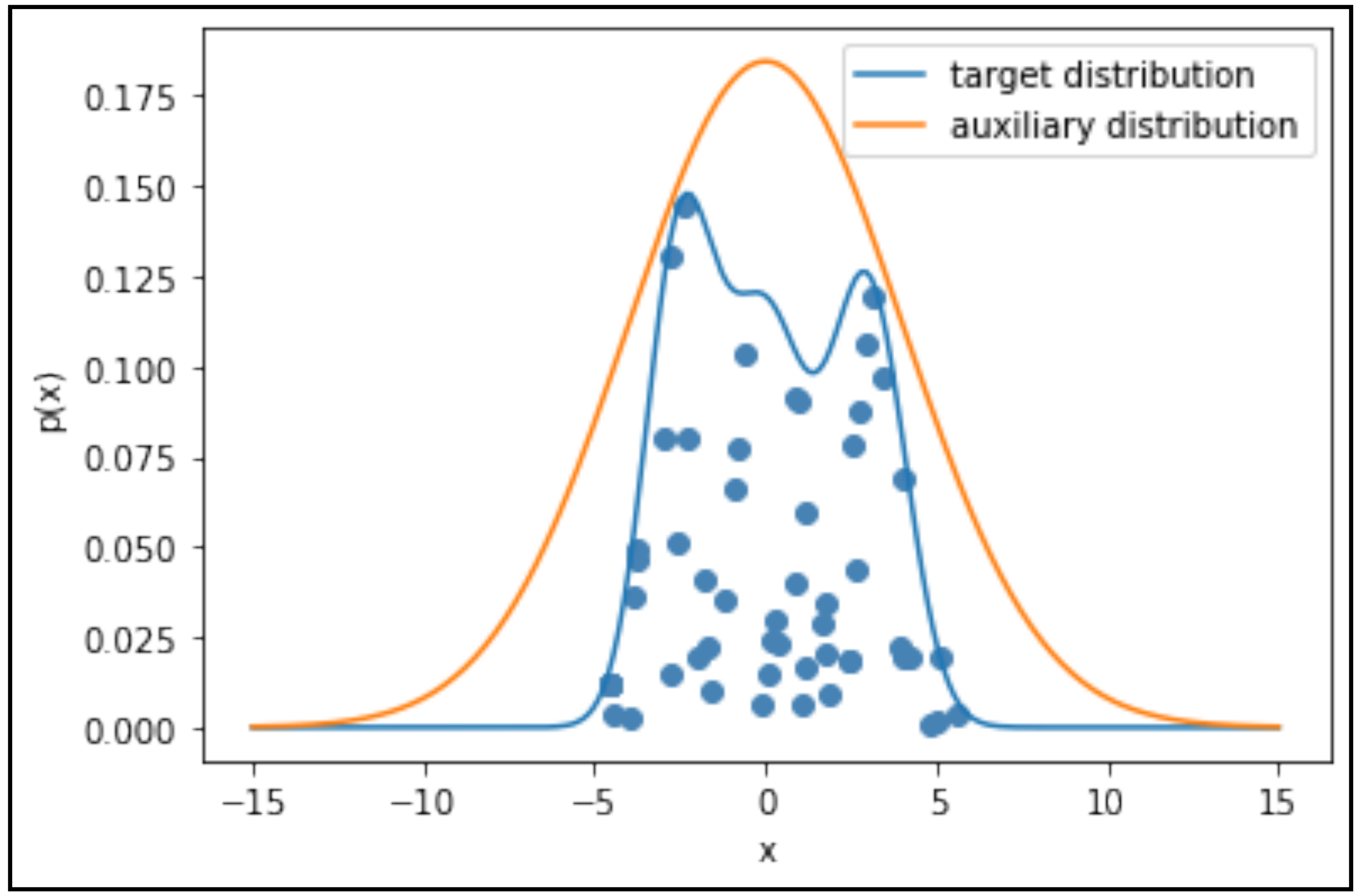
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \mathrm{const} \times \mathrm{Q}$
>
> **Algorithm**
>
> 1. generate sample $x_i \sim \; Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \; \mathcal{U}(\,0, \mathrm{const} \cdot \mathrm{Q}(\mathrm{x_i})\,)$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



X_1

X_2

...

X_n

works for most distribution

if the « **gaps** » between P and Q are too large,
we reject most of the sample

1. Monte Carlo estimation

Usual simulations : the power of uniform distribution

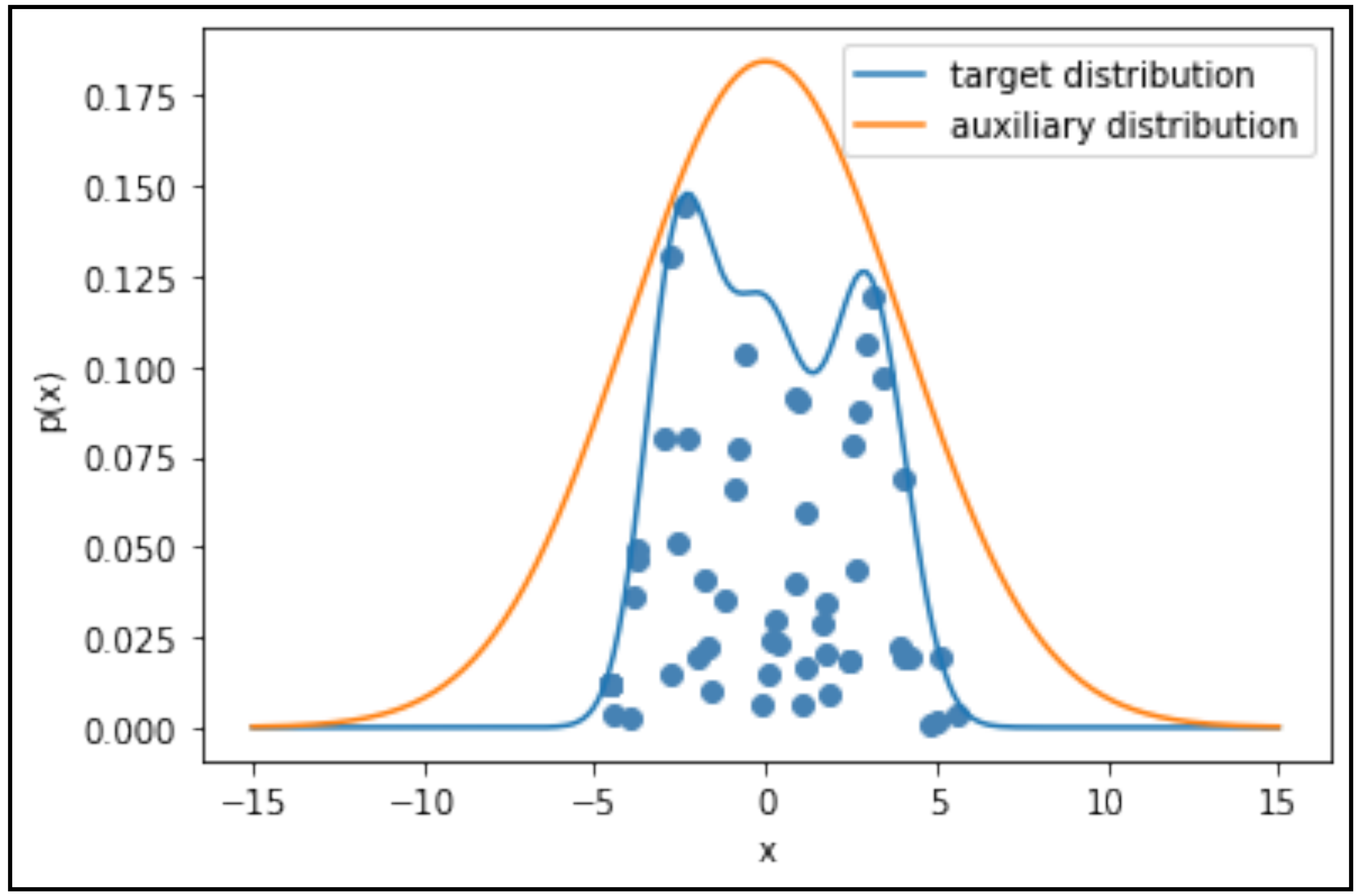
Starting point : we know how to simulate a pseudo-random uniform $U \sim \mathcal{U}(0,1)$

For « usual » distributions : both **discrete** and **continuous r.v.** can be sampled thanks to the uniform distribution

> **In practice (with python)** we can easily sample them (via **scipy** and **numpy** for example)

Otherwise : if there isn't an analytical way to sample it then

> **Rejection sampling** algorithm.
> Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times \mathrm{Q}$
>
> **Algorithm**
> 1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
> 2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot \mathrm{Q}(\mathrm{x_i}))$
> 3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



X_1

X_2

...

X_n

works for most distribution

if the « **gaps** » between P and Q are too large,
we reject most of the sample

use MCMC

2 Markov Chain Monte Carlo : Definition

2. Markov Chain Monte Carlo

Definition : Monte Carlo sampling

Monte Carlo sampling : generates independent samples from the probability distribution in order to estimate an expected value

X_1 X_2 … X_n where $X_1, \ldots, X_n \sim P$ i.i.d

2. Markov Chain Monte Carlo

Definition : Markov Chain

Monte Carlo sampling : generates independent samples from the probability distribution in order to estimate an expected value

X_1 X_2 ... X_n where $X_1, \ldots, X_n \sim P$ i.i.d

Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

P is called **stationary** if $P(x') = \sum_{x \in \mathrm{supp(X)}} T(x, x') \cdot P(x)$

$T(x, x')$ the transition probability of being in the state x' given the current state x

2. Markov Chain Monte Carlo

Definition : Markov Chain Monte Carlo

Monte Carlo sampling : generates independent samples from the probability distribution in order to estimate an expected value

$X_1 \quad X_2 \quad \dots \quad X_n$ where $X_1, \dots, X_n \sim P$ i.i.d

Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

P is called **stationary** if $P(x') = \sum_{x \in \text{supp(X)}} T(x, x') \cdot P(x)$

$T(x, x')$ the transition probability of being in the state x' given the current state x

Markov Chain Monte Carlo sampling : a sequence of *Monte Carlo Samples* where the *next* sample is dependent upon the *current* sample

$$X_1 \xrightarrow{T} X_2 \xrightarrow{T} \cdots \xrightarrow{T} X_b \xrightarrow{T} X_{b+1} \xrightarrow{T} \cdots \xrightarrow{T} X_{n+b}$$

Burn-in samples

$X_{b+1}, \dots, X_{b+n} \sim P$ **with some possible correlations**

2. Markov Chain Monte Carlo

Definition : Markov Chain Monte Carlo

Monte Carlo sampling : generates independent samples from the probability distribution in order to estimate an expected value

$$X_1 \quad X_2 \quad \dots \quad X_n$$

where $X_1, \dots, X_n \sim P$ i.i.d

Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

P is called **stationary** if $P(x') = \sum_{x \in \mathrm{supp(X)}} T(x, x') \cdot P(x)$

$T(x, x')$ the transition probability of being in the state x' given the current state x

Markov Chain Monte Carlo sampling : a sequence of *Monte Carlo Samples* where the *next* sample is dependent upon the *current* sample

$$X_1 \xrightarrow{T} X_2 \xrightarrow{T} \dots \xrightarrow{T} X_b \xrightarrow{T} X_{b+1} \xrightarrow{T} \dots \xrightarrow{T} X_{n+b}$$

Burn-in samples

$X_{b+1}, \dots, X_{b+n} \sim P$ **with some possible correlations**

Objective : Build a Markov Chain that converges to the target distribution P no matter the starting point

2. Markov Chain Monte Carlo

Definition : Markov Chain Monte Carlo

Monte Carlo sampling : generates independent samples from the probability distribution in order to estimate an expected value

$$X_1 \qquad X_2 \qquad \dots \qquad X_n$$

where $X_1, \dots, X_n \sim P$ i.i.d

Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

P is called **stationary** if $P(x') = \sum_{x \in \mathrm{supp(X)}} T(x, x') \cdot P(x)$

$T(x, x')$ the transition probability of being in the state x' given the current state x

Markov Chain Monte Carlo sampling : a sequence of *Monte Carlo Samples* where the *next* sample is dependent upon the *current* sample

$$\underbrace{X_1 \xrightarrow{T} X_2 \xrightarrow{T} \cdots \xrightarrow{T}}_{\text{Burn-in samples}} \underbrace{X_b \xrightarrow{T} X_{b+1} \xrightarrow{T} \cdots \xrightarrow{T} X_{n+b}}_{X_{b+1}, \dots, X_{b+n} \sim P \text{ with some possible correlations}}$$

Objective : Build a Markov Chain that converges to the target distribution P no matter the starting point

Theorem : if $T(x, x') > 0$ for all x, x' then there *exists* an *unique* **stationary** and **convergent** distribution

3 Markov Chain Monte Carlo : Algorithms

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(x_j \mid x_{-j})$ can be sampled
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(x_j \mid x_{-j})$ can be sampled
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \ldots, x_d^{(i-1)}\right)$

$$x_1^{(i)} \sim P(x_1 \mid x_2^{(i-1)}, x_3^{(i-1)}, \ldots, x_d^{(i-1)})$$

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(x_j \mid x_{-j})$ can be sampled
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \ldots, x_d^{(i-1)}\right)$

$$x_1^{(i)} \sim P(x_1 \mid x_2^{(i-1)}, x_3^{(i-1)}, \ldots, x_d^{(i-1)})$$

$$x_2^{(i)} \sim P(x_2 \mid x_1^{(i)}, x_3^{(i-1)}, \ldots, x_d^{(i-1)})$$

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \dots, x^{(n)} \sim P\left(x_1, x_2, \dots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \dots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \dots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(\, x_j \,|\, x_{-j}\,)$ can be sampled
- **Initialisation** : $x^{(0)} = (0,\dots,0)$ or random values
- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \dots, x_d^{(i-1)}\right)$

$$x_1^{(i)} \sim P(\, x_1 \,|\, x_2^{(i-1)}, x_3^{(i-1)}, \dots,\; x_d^{(i-1)}\,)$$

$$x_2^{(i)} \sim P(\, x_2 \,|\, x_1^{(i)}, x_3^{(i-1)}, \dots,\; x_d^{(i-1)}\,)$$

…

$$x_d^{(i)} \sim P(\, x_d \,|\, x_2^{(i)}, x_3^{(i)}, \dots,\; x_d^{(i-1)}\,)$$

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(\, x_j \,|\, x_{-j}\,)$ can be sampled
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \ldots, x_d^{(i-1)}\right)$

for each position, $x_k^{(i)} \sim P(\, x_1 \mid x_{1:k-1}^{(i)}, x_{k+1:d}^{(i-1)}\,)$

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(\, x_j \,|\, x_{-j}\,)$ can be sampled

- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values

- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \ldots, x_d^{(i-1)}\right)$

for each position, $x_k^{(i)} \sim P(\, x_1 \mid x_{1:k-1}^{(i)}, x_{k+1:d}^{(i-1)}\,)$

sometimes it can converge slowly to the desired distribution

sometimes Gibbs samples can be too correlated

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Gibbs Sampling Algorithm

- **Hypothesis** : The conditional $P(x_j \mid x_{-j})$ can be sampled
- **Initialisation** : $x^{(0)} = (0, \ldots, 0)$ or random values
- **Repeat** :

sample $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$ based on $x^{(i-1)} = \left(x_1^{(i-1)}, \ldots, x_d^{(i-1)}\right)$

for each position, $x_k^{(i)} \sim P(\, x_1 \mid x_{1:k-1}^{(i)}, x_{k+1:d}^{(i-1)}\,)$

sometimes it can converge slowly to the desired distribution

sometimes Gibbs samples can be too correlated

Use a variant Gibbs sampling : Metropolis-Hastings

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Metropolis-Hastings Algorithm

- **Hypothesis** : Let $P = \hat{P}/\mathrm{const}$ where $\hat{P}$ can be calculated and let Q be an **auxiliary distribution** we can sample from

- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values

- **Repeat** :

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Metropolis-Hastings Algorithm

- **Hypothesis** : Let $P = \hat{P}/\text{const}$ where $\hat{P}$ can be calculated and let Q be an **auxiliary distribution** we can sample from
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

  sample a **candidate** $x^{(i)} \sim Q(\, x^{(i)} \mid x^{(i-1)}\,) =$ (example of auxiliary distribution) $\mathcal{N}(x^{(i-1)}, \sigma^2 I)$

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Metropolis-Hastings Algorithm

- **Hypothesis** : Let $P = \hat{P}/\mathrm{const}$ where $\hat{P}$ can be calculated and let Q be an **auxiliary distribution** we can sample from

- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values

- **Repeat** :

  sample a **candidate** $x^{(i)} \sim Q(\, x^{(i)} \mid x^{(i-1)}\,)$ = (example of auxiliary distribution) $\mathcal{N}(x^{(i-1)}, \sigma^2 I)$

  with **acceptance probability** : $\min\left(1, \dfrac{Q(x^{(i-1)} \,|\, x^{(i)}) \times \hat{P}(x^{(i)})}{Q(x^{(i)} \,|\, x^{(i-1)}) \times \hat{P}(x^{(i-1)})}\right)$ accept $x^{(i)}$ **as an sample from** P

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Metropolis-Hastings Algorithm

- **Hypothesis** : Let $P = \hat{P}/\mathrm{const}$ where $\hat{P}$ can be calculated and let Q be an **auxiliary distribution** we can sample from
- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values
- **Repeat** :

  sample a **candidate** $x^{(i)} \sim Q(\, x^{(i)} \mid x^{(i-1)}\,)$ = (example of auxiliary distribution) $\boxed{\mathcal{N}(x^{(i-1)}, \sigma^2 I)}$

  with **acceptance probability** : $\min\left(1, \dfrac{\cancel{Q(x^{(i-1)} | x^{(i)})} \times \hat{P}(x^{(i)})}{\cancel{Q(x^{(i)} | x^{(i-1)})} \times \hat{P}(x^{(i-1)})}\right)$ accept $x^{(i)}$ **as an sample from** P

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

Reminder : we want to sample $x^{(1)}, \ldots, x^{(n)} \sim P\left(x_1, x_2, \ldots, x_d\right)$

Remark : we denote $x^{(i)} := \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)$; $x_{-j} = \left(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d\right)$; $x_{m:n} = \left(x_m, x_{m+1}, \ldots, x_n\right)$

Metropolis-Hastings Algorithm

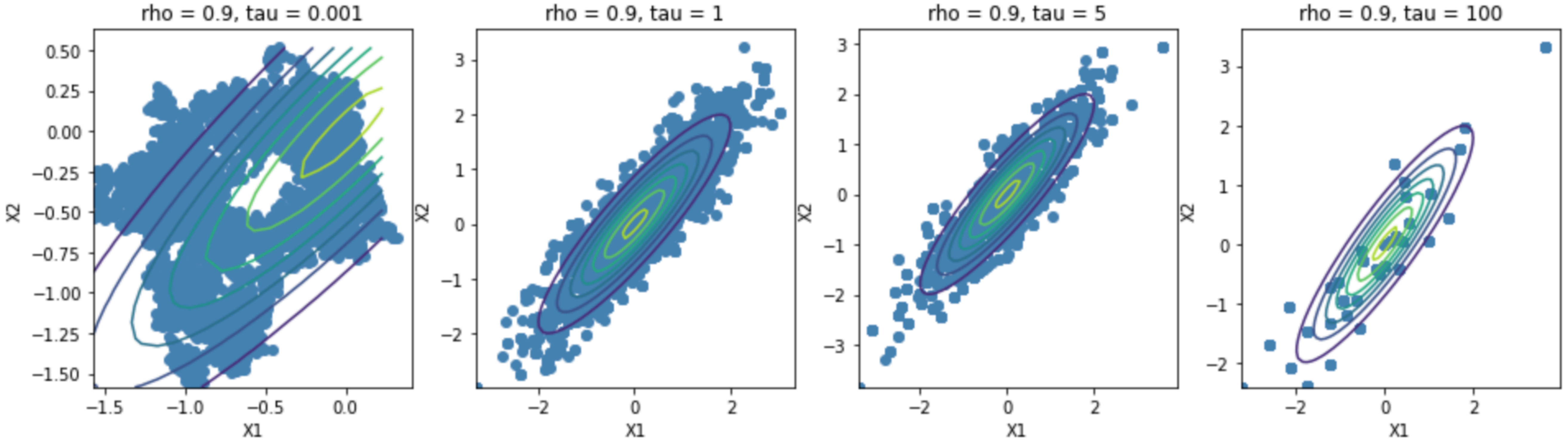
- **Hypothesis** : Let $P = \hat{P}/\mathrm{const}$ where $\hat{P}$ can be calculated and let Q be an **auxiliary distribution** we can sample from

- **Initialisation** : $x^{(0)} = (0,\ldots,0)$ or random values

- **Repeat** :

  sample a **candidate** $x^{(i)} \sim Q(\, x^{(i)} \mid x^{(i-1)}\,)$ = (example of auxiliary distribution) $\boxed{\mathcal{N}(x^{(i-1)}, \sigma^2 I)}$

  with **acceptance probability** : $\min\left(1, \dfrac{\cancel{Q(x^{(i-1)} | x^{(i)})} \times \hat{P}(x^{(i)})}{\cancel{Q(x^{(i)} | x^{(i-1)})} \times \hat{P}(x^{(i-1)})}\right)$ accept $x^{(i)}$ **as an sample from** P



$\tau = \sigma^2$ **and** ρ **the correlation between two gaussians** X_1 **and** X_2

3.b. MCMC vs VI

3.b. **MCMC vs VI**

pros and cons

MCMC

Pros :
- Useful when the posterior is intractable
- Asymptotically exact
- Suited to small / medium dataset

Cons :
- Usually slower than alternatives (VI)
- Can generates dependant samples from the distribution

VI (see lecture 3)

Pros :
- Useful when the posterior is intractable
- Suited to large dataset

Cons :
- Can never generate exact result

4 Applications : notebook

! Road map

Bayesian statistics

1

Bayesian perspective :

Likelihood — Prior distribution

$$P(\theta|X) = \frac{P(X,\theta)}{P(X)} = \frac{P(X|\theta)\cdot P(\theta)}{P(X)}$$

Posterior distribution

Evidence

Hard to compute !

θ parameters

X observations

Exemple :
Naive Bayes classifier,
Linear regression, ….

MAP : $\arg\max_{\theta} P(X|\theta)\cdot P(\theta)$

Conjugate distribution

Pros :	Cons :
- exact posterior	- conjugate prior maybe inadequate

Latent variable models

2

Hidden variable models :

$$P(X|\theta) = \sum_{t\in T_{indexes}} P(X,T=t|\theta)$$

$$P(X,T|\theta) = P(X|T,\theta)P(T|\theta)$$

Exemple :
GMM, K-means, PCA/PPCA

Pros :	Cons :
- fewer parameters / simpler models - hidden variable sometimes meaningful - clustering / dimensionality reduction	- harder to work with - requires math - only local maximum or saddle point - EM : the posterior of T could be intractable

Variational Inference

3

Deterministic approximation of posterior :

$$p(Z|X) = \frac{P(X|Z)\cdot P(Z)}{P(X)}$$

Mean Field Approximation !

Exemple :
Topic modelling, LDA trained by VI

Pros :	Cons :
- Useful when the posterior is intractable - Suited to large dataset	- can never generate exact result

Markov Chain Monte Carlo

4

Sampling techniques for estimate expected values :

$$\mathbb{E}_{p(x)}[h(x)] \approx \frac{1}{M}\sum_{s=1}^{M} f(x_s)$$

$f(x_s) \sim p(x)$ Gibbs sampling / Metropolis-Hastings !

Exemple :
Topic modelling, LDA trained by MCMC

Pros :	Cons :
- train / inference almost every probabilistic model - asymptotically exact - suited to small / medium dataset	- Usually slower than alternatives (VI) - can generates dependant samples from the distribution

Extensions

5

Oral presentations (20 points)

+

Exercice 1 of lecture 1 : 0.5 bonus point
Notebook 1 : 1 bonus point
Notebook 2 : 2 bonus points
Notebook 3 : 0.5 bonus point
Notebook 4 : 1 bonus point
Notebook 5 : 1 bonus point