



Bayesian Machine Learning

14/06/21 - 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

0 Remarks

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

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

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

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

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

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

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

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

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

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

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

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

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

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

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

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

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

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

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

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

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

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

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

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

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\begin{aligned}\mathbb{E}[X] &\approx \bar{x}^{(n)} = \frac{1}{n} \sum_{i=1}^n x^{(i)} \\ \mathbb{E}[h(X)] &\approx \frac{1}{n} \sum_{i=1}^n h(x^{(i)}) \\ V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] &\approx \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \bar{x}^{(n)})^2 \\ \mathbb{P}(X > 2) = \mathbb{E}[1_{\{X>2\}}] &\approx \frac{1}{n} \sum_{i=1}^n 1_{\{x^{(i)}>2\}}\end{aligned}$$

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

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 $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$ sample of a r.v. X and $U, V \sim \mathcal{U}(0,1)$

$$\begin{aligned}\mathbb{E}[X] &\approx \bar{x}^{(n)} = \frac{1}{n} \sum_{i=1}^n x^{(i)} \\ \mathbb{E}[h(X)] &\approx \frac{1}{n} \sum_{i=1}^n h(x^{(i)}) \\ V(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] &\approx \frac{1}{n} \sum_{i=1}^n (x^{(i)} - \bar{x}^{(n)})^2 \\ \mathbb{P}(X > 2) = \mathbb{E}[1_{\{X>2\}}] &\approx \frac{1}{n} \sum_{i=1}^n 1_{\{x^{(i)}>2\}} \\ \pi = 4 \times \mathbb{E}[U^2 + V^2 \leq 1] &\approx \frac{4}{n} \sum_{i=1}^n 1_{\{(u^{(i)})^2 + (v^{(i)})^2 \leq 1\}} \text{ for } u^{(i)}, v^{(i)} \sim \mathcal{U}(0,1)\end{aligned}$$

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**

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

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

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

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

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

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

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

easy : model output + prior fixed by us
difficult : as always ...

$$P(y | x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n} \sum_{i=1}^n P(y | x, \theta_i) \quad \text{if we can sample } \theta_1, \dots, \theta_n \sim P(\theta | X^{(train)}, Y^{(train)})$$

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

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

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

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 $\theta_1, \dots, \theta_n \sim P(\theta | X^{(train)}, Y^{(train)})$

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

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

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

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

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

easy : model output + prior fixed by us
difficult : as always ...

$$P(y | x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n} \sum_{i=1}^n P(y | x, \theta_i) \quad \text{if we can sample } \theta_1, \dots, \theta_n \sim P(\theta | X^{(train)}, Y^{(train)})$$

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

$$\max_{\theta} \mathbb{E}_T [\log P(X, T | \theta)] \approx \frac{1}{n} \sum_{i=1}^n \log P(X, T_i | \theta) \quad \text{if we can sample } T_1, \dots, T_n \sim T$$

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

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

with $P(\theta | X^{(train)}, Y^{(train)}) = \frac{p(Y^{(train)} | X^{(train)}, \theta) \cdot P(\theta)}{\text{constant}}$ **easy** : model output + prior fixed by us
difficult : as always ...

$$P(y | x, X^{(train)}, Y^{(train)}) \approx \frac{1}{n} \sum_{i=1}^n P(y | x, \theta_i) \quad \text{if we can sample } \theta_1, \dots, \theta_n \sim P(\theta | X^{(train)}, Y^{(train)})$$

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

$$\max_{\theta} \mathbb{E}_T [\log P(X, T | \theta)] \approx \frac{1}{n} \sum_{i=1}^n \log P(X, T_i | \theta) \quad \text{if we can sample } T_1, \dots, T_n \sim T$$

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 $U \sim \mathcal{U}(0,1)$

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)

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. $P \leq \text{const} \times Q$

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. $P \leq \text{const} \times Q$

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

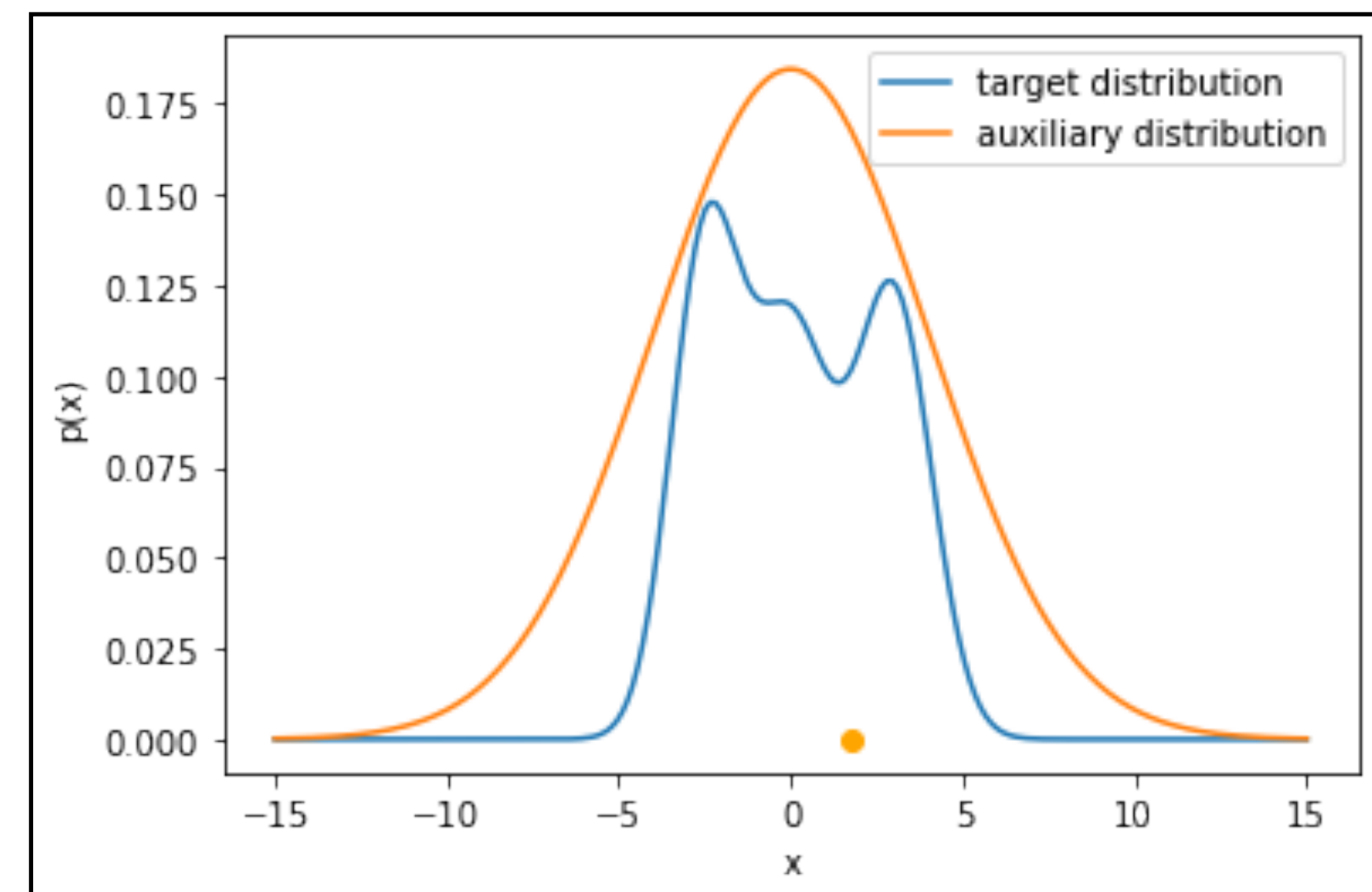
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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 $U \sim \mathcal{U}(0,1)$

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)

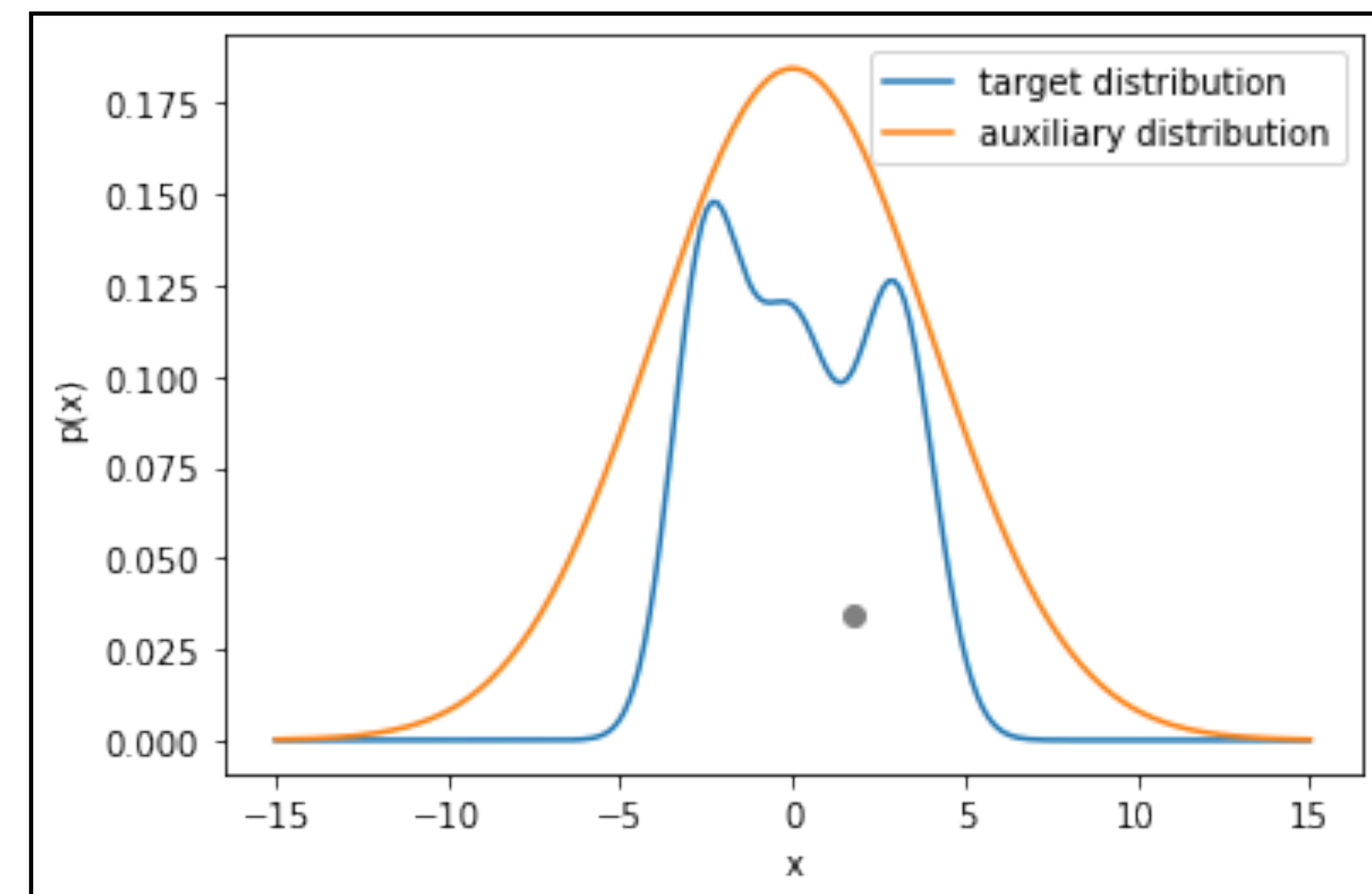
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

| Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times Q$

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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 $U \sim \mathcal{U}(0,1)$

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)

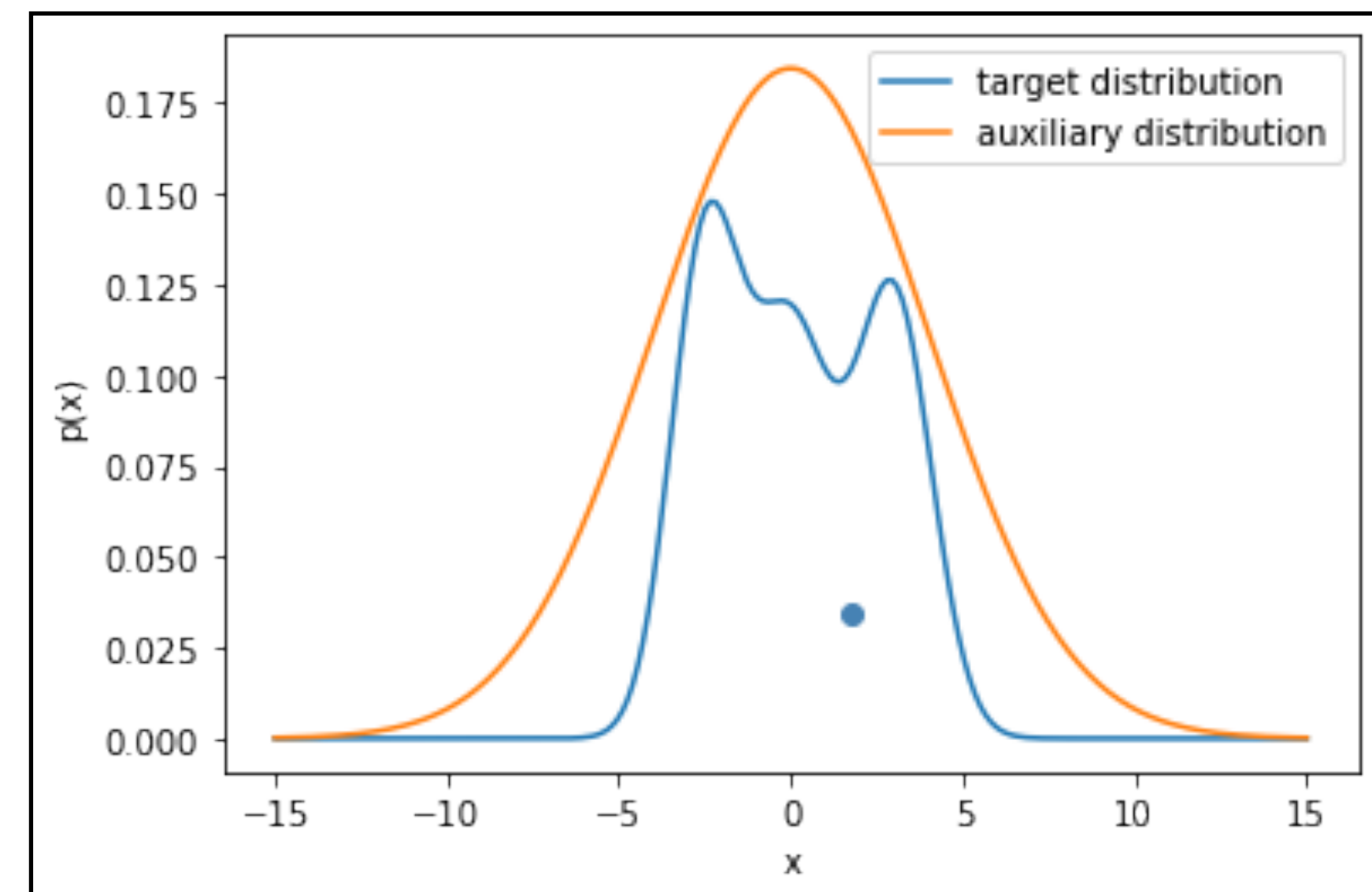
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

| Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times Q$

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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 $U \sim \mathcal{U}(0,1)$

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)

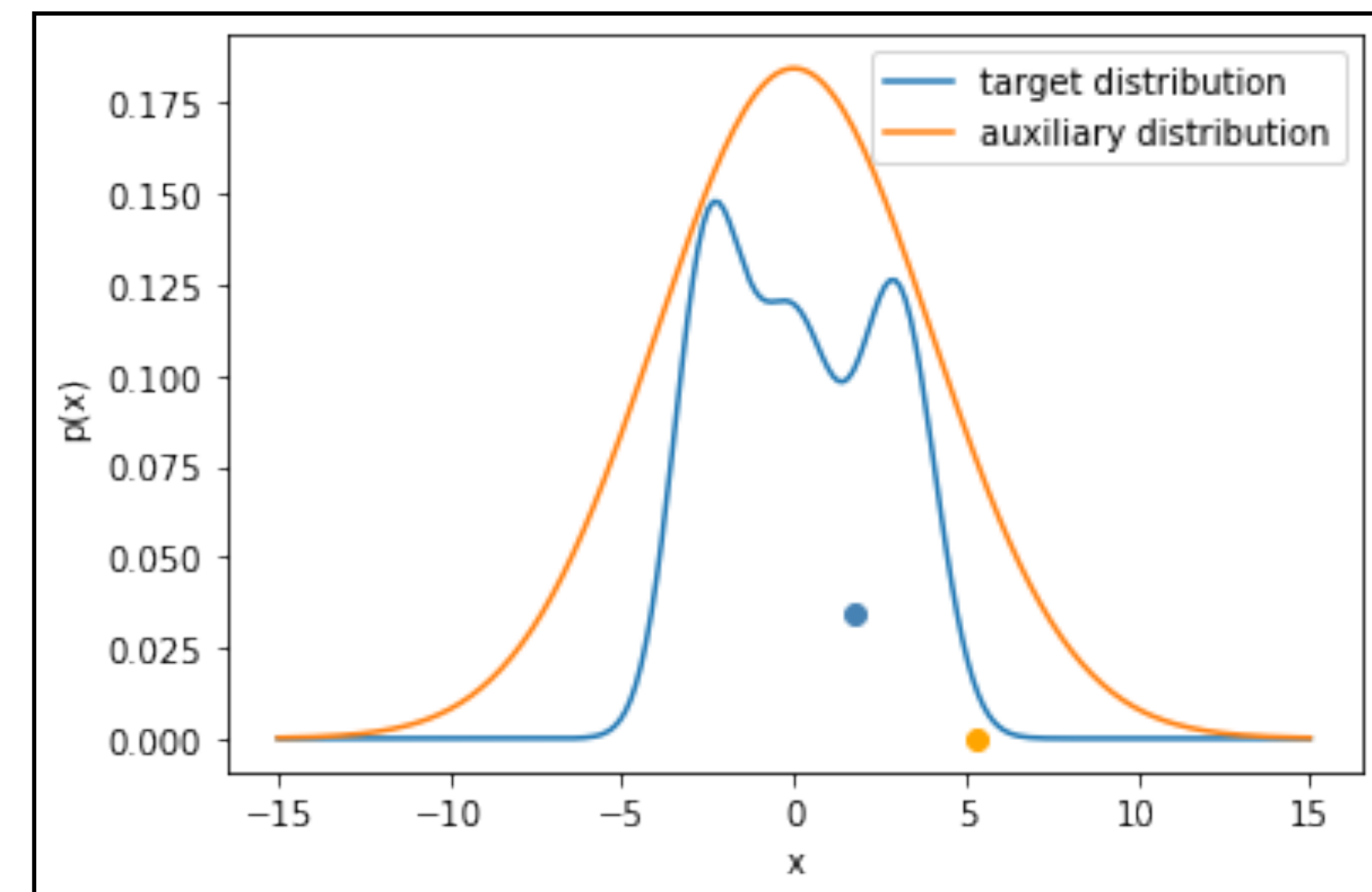
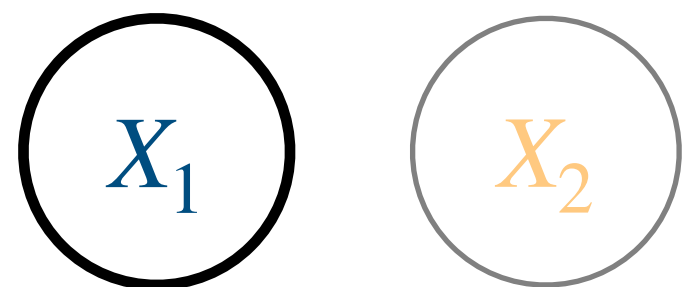
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

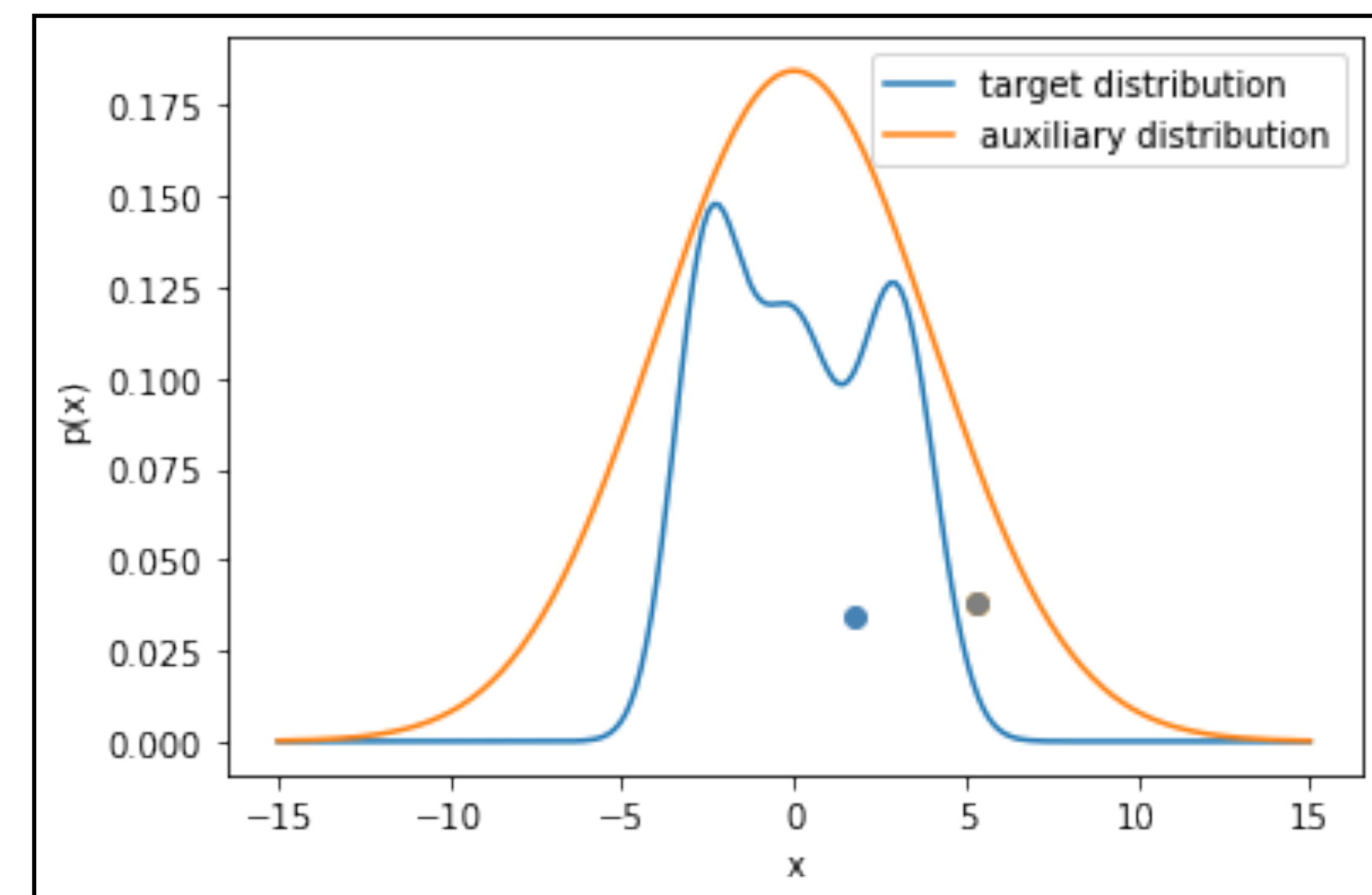
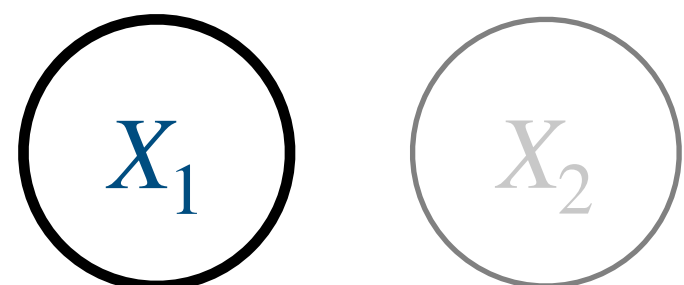
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

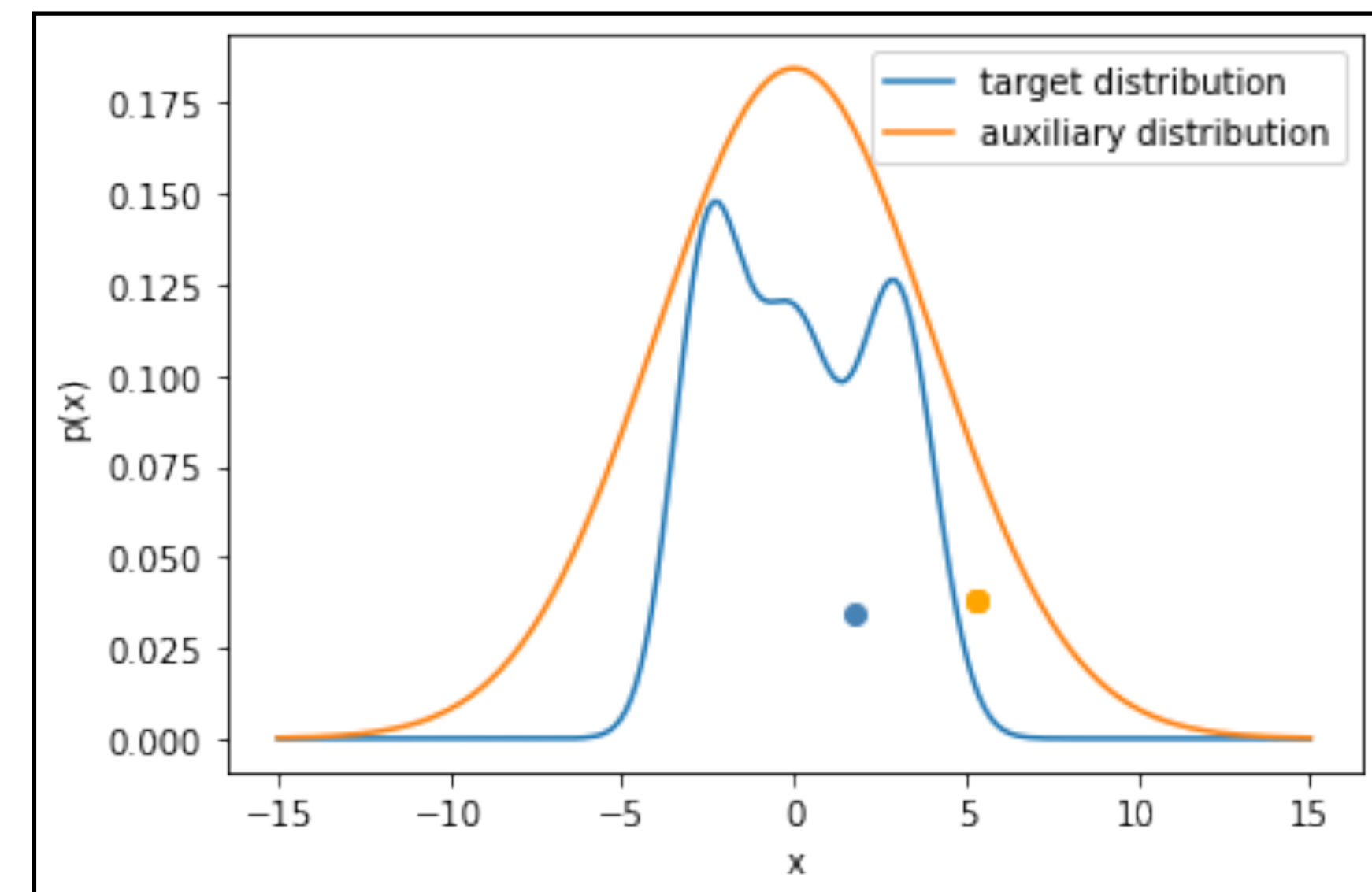
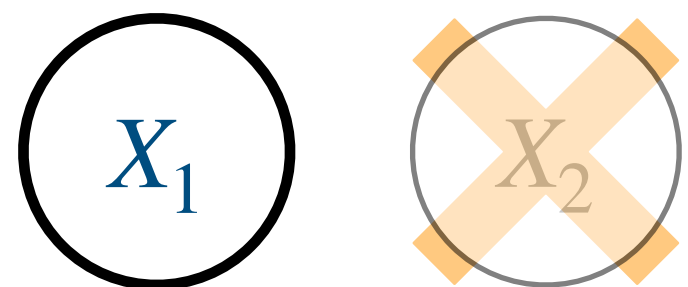
Otherwise : if there isn't an analytical way to sample it then

Rejection sampling algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

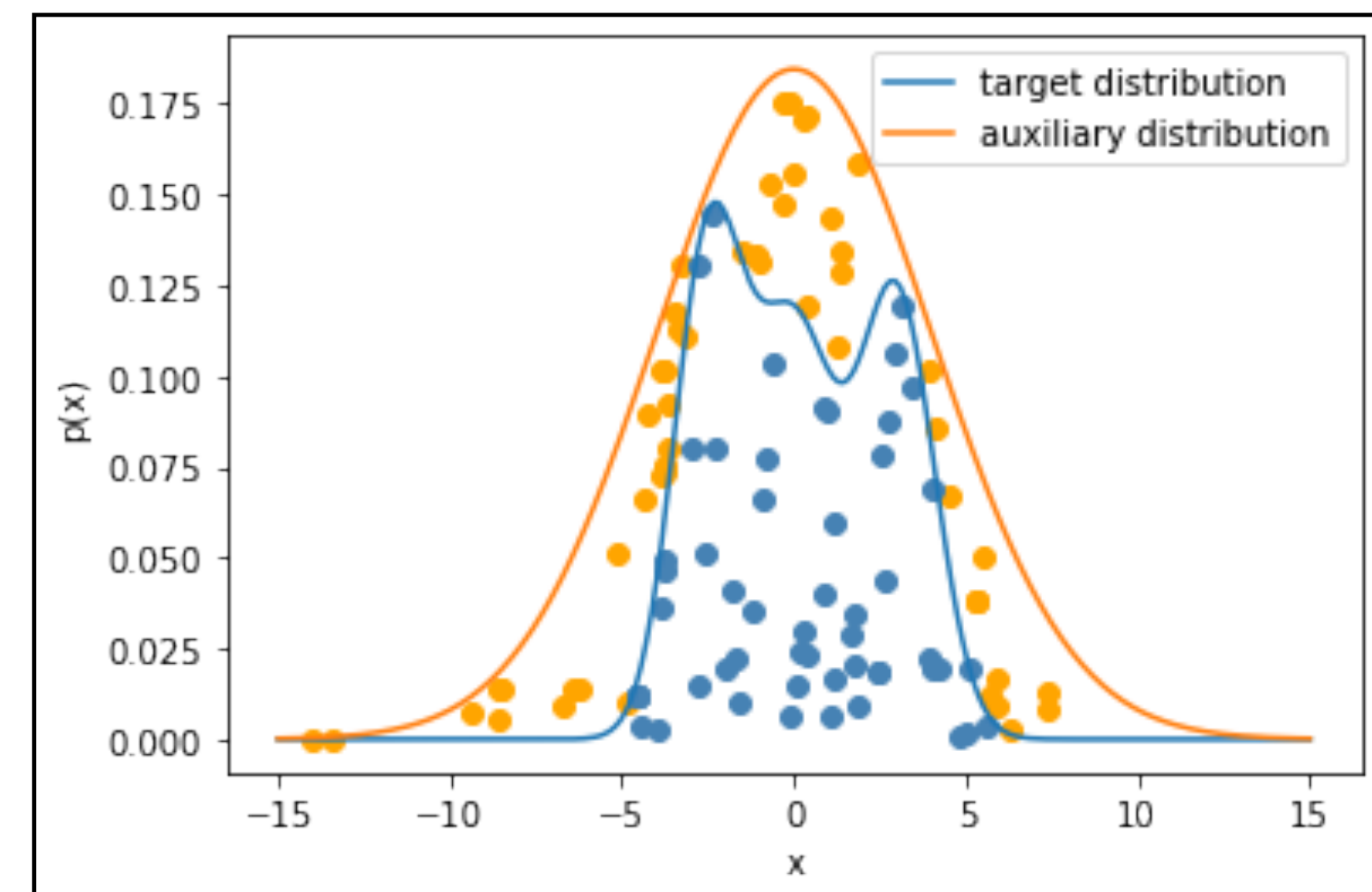
Otherwise : if there isn't an analytical way to sample it then

Rejection sampling algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

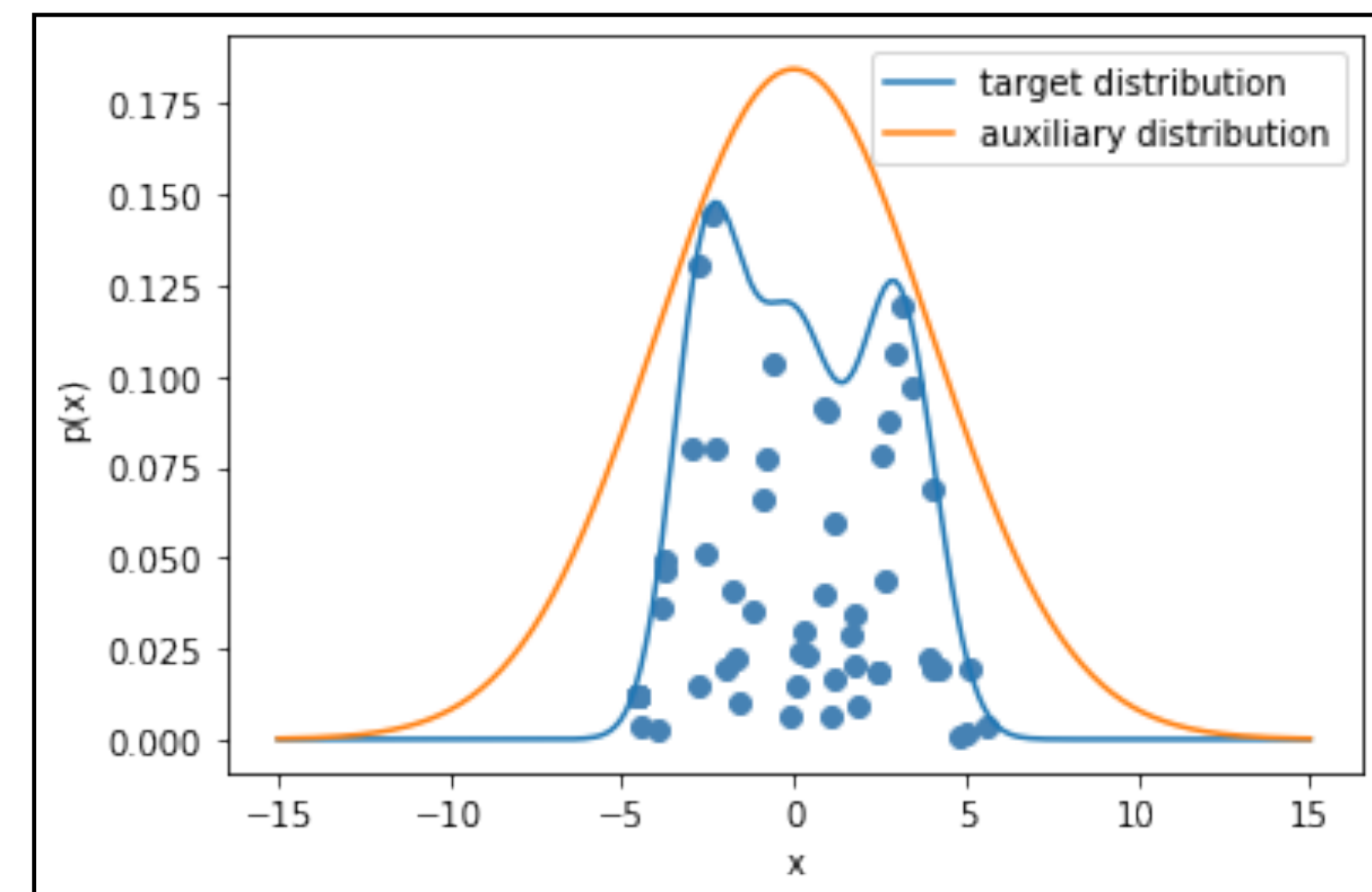
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

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

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
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)

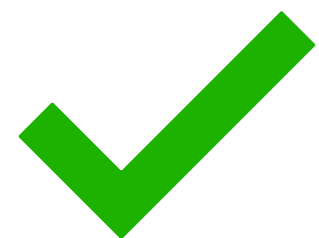
Otherwise : if there isn't an analytical way to sample it then

| **Rejection sampling** algorithm.

| Assumption : we can compute **distribution's pdf** P and sample from an **auxiliary distribution** Q s.t. $P \leq \text{const} \times Q$

Algorithm

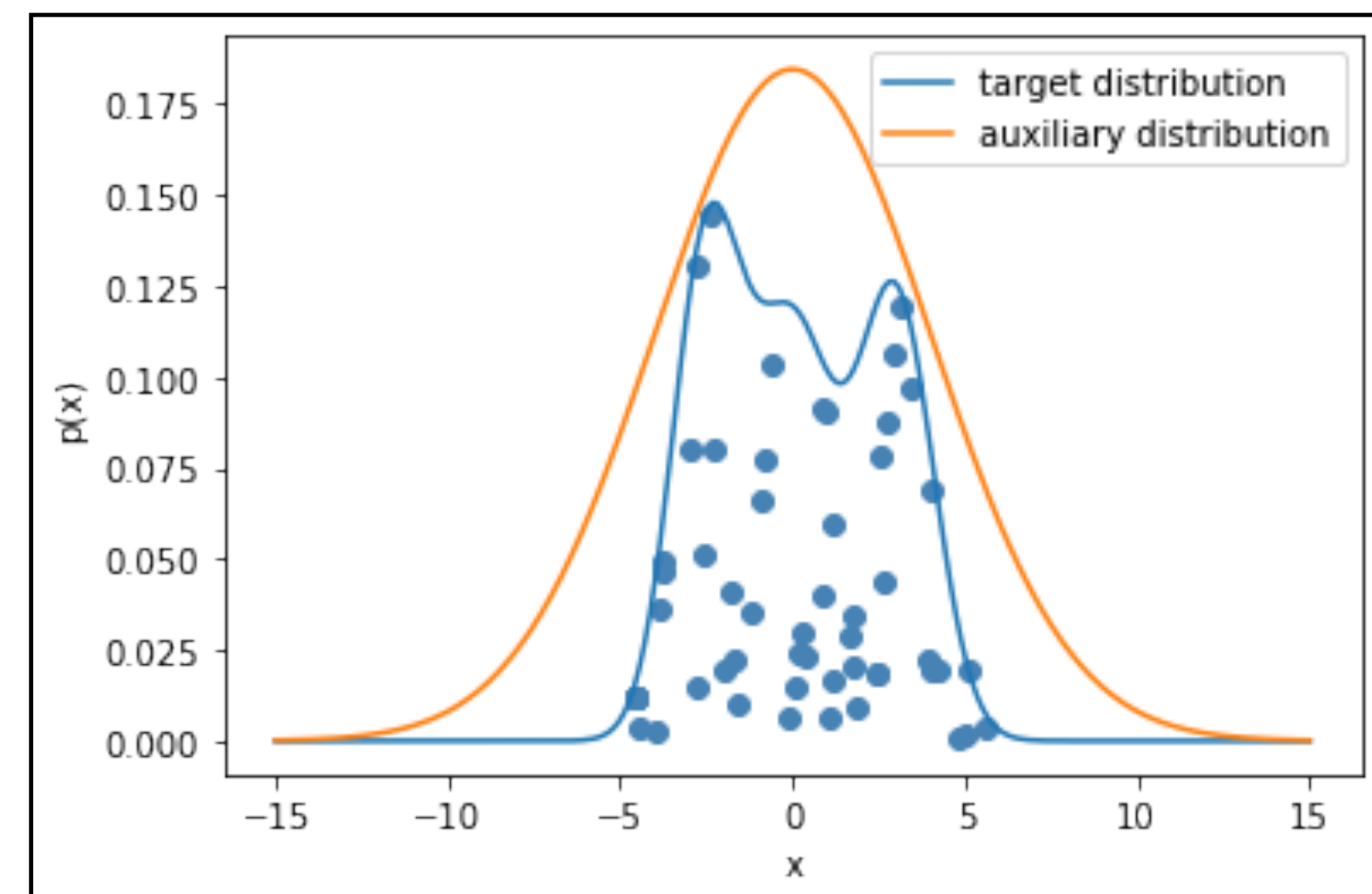
1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.



works for most distribution



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



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. $P \leq \text{const} \times Q$

Algorithm

1. generate sample $x_i \sim Q$ (**auxiliary distribution**)
2. generate sample $u \sim \mathcal{U}(0, \text{const} \cdot Q(x_i))$
3. if $u \leq P(x_i)$ then **accept** x_i else **reject**.

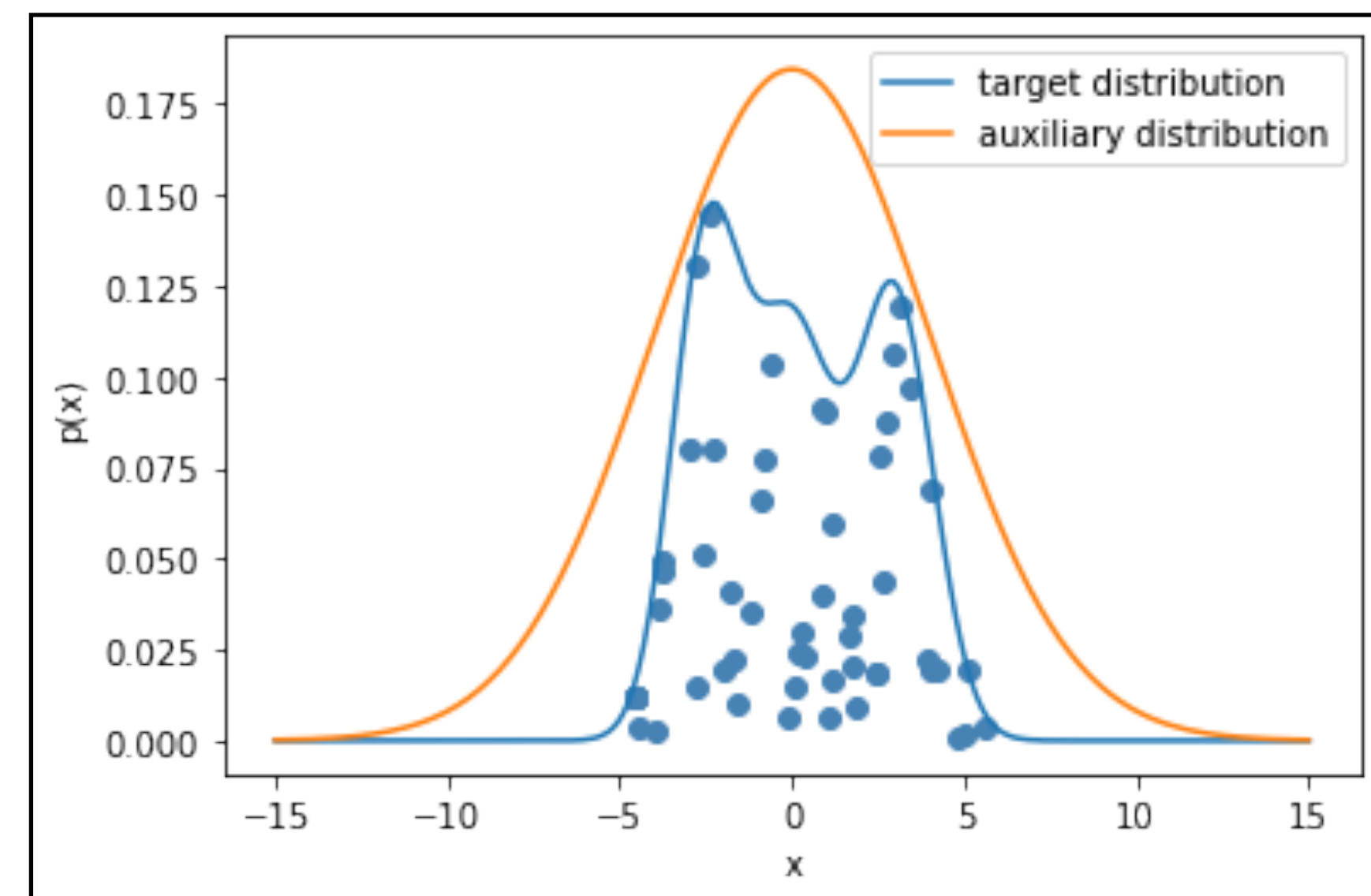


works for most distribution



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

use MCMC



X_1

X_2

...


X_n

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


$$X_1$$

$$X_2$$

...

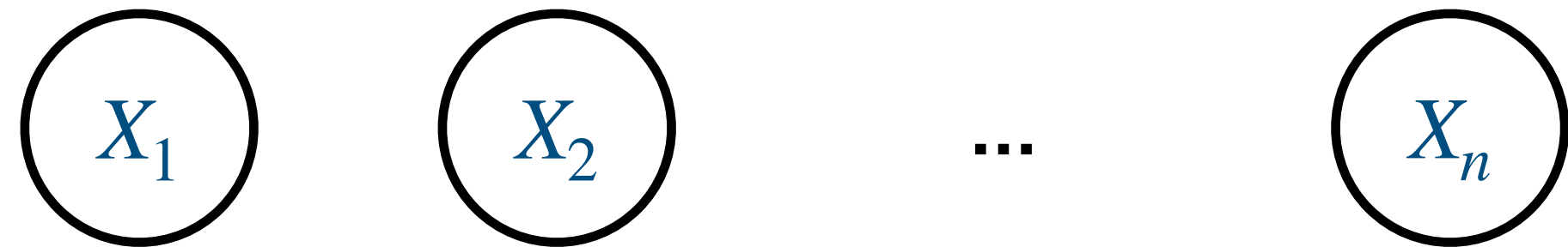
$$X_n$$

where $X_1, \dots, X_n \sim P$ 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



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

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

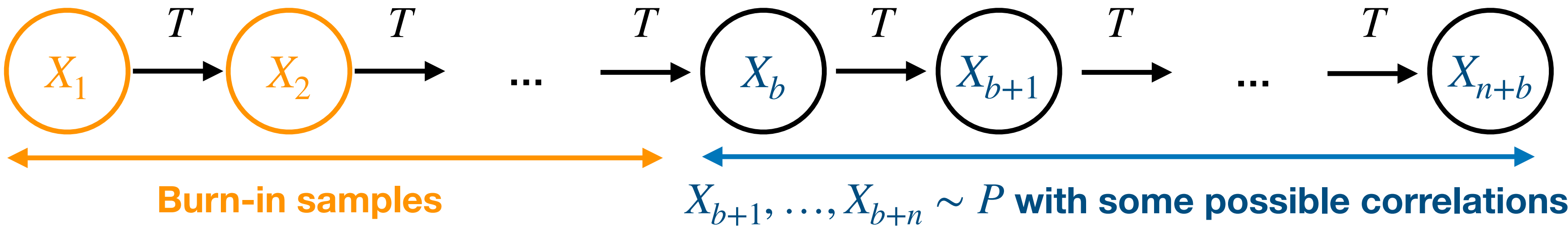


Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

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

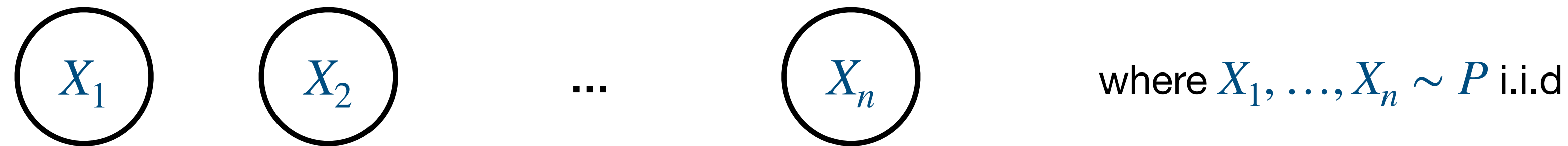
Markov Chain Monte Carlo sampling : a sequence of *Monte Carlo Samples* where the *next* sample is dependent upon the *current* sample



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

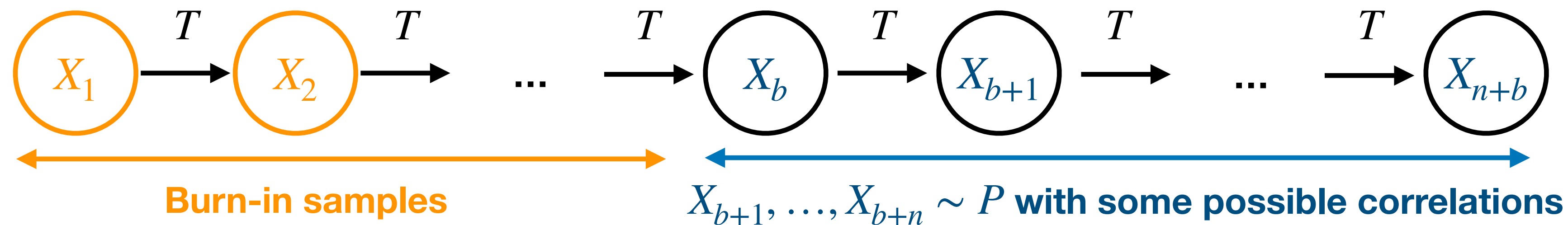


Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

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

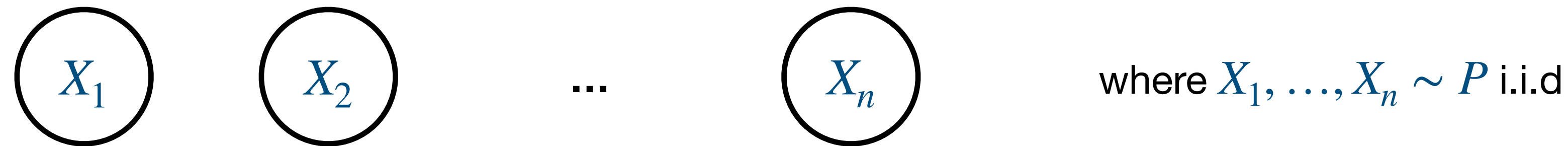


Objective : Build a Markov Chain that converges to the target distribution P no matter the starting point

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

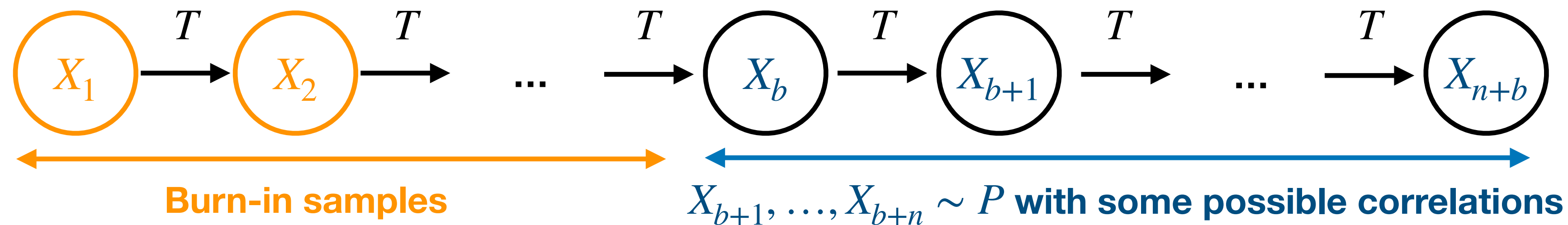


Markov Chain : generates a sequence of r.v. where the *next* variable is probabilistically dependent upon the *current* variable.

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



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 $x^{(1)}, \dots, x^{(n)} \sim P(x_1, x_2, \dots, x_d)$

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

Gibbs Sampling Algorithm

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

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

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

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

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

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

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

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

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

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

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

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

...

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

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

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

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Gibbs sampling

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

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

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

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

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 $x^{(1)}, \dots, x^{(n)} \sim P(x_1, x_2, \dots, x_d)$

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

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

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

sometimes it can converge slowly to the desired distribution

Use a variant Gibbs sampling : **Metropolis-Hastings**
sometimes Gibbs samples can be too correlated

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

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

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

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

sample a **candidate** $x^{(i)} \sim Q(x^{(i)} | x^{(i-1)})$ = (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 $x^{(1)}, \dots, x^{(n)} \sim P(x_1, x_2, \dots, x_d)$

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

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

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

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

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

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

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

3. Markov Chain Monte Carlo

Algorithm : Metropolis-Hastings

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

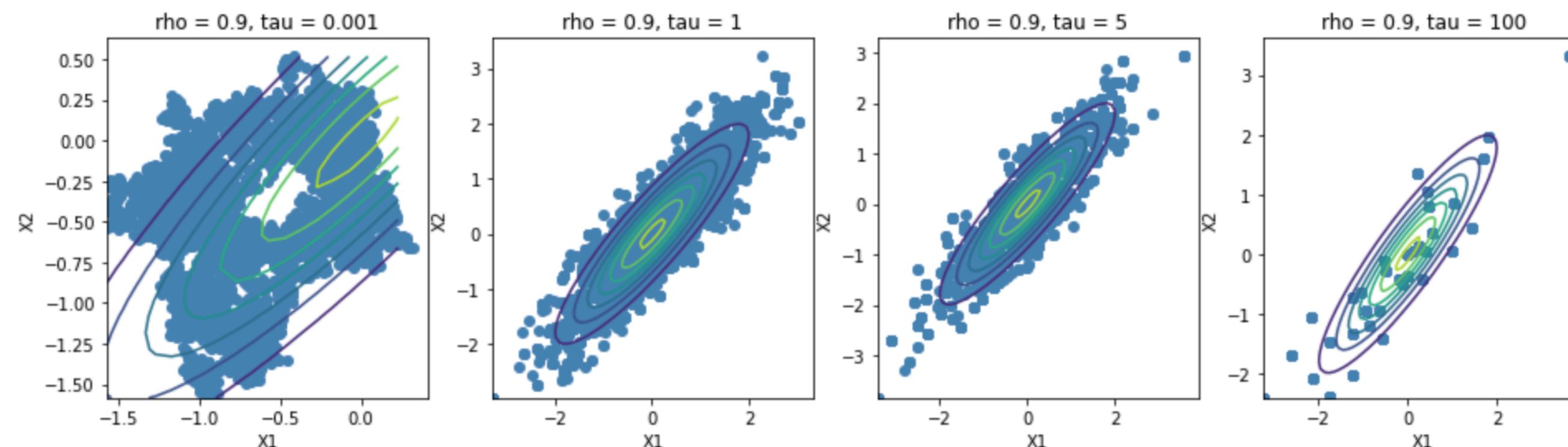
Remark : we denote $x^{(i)} := (x_1^{(i)}, \dots, x_d^{(i)})$; $x_{-j} = (x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_d)$; $x_{m:n} = (x_m, x_{m+1}, \dots, x_n)$

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

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

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



$\tau = \sigma^2$ 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 generate 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 and examples : notebook

Application and examples

website : <https://curiousml.github.io/>

EPITA - École pour l'informatique et les techniques avancées (2020 - ...)

- Master of Science in Artificial Intelligence Systems : **Bayesian Machine Learning** by [François HU](#)
 - Training session / prerequisite : [\[Statistics with python\]](#), [\[Data\]](#)
 - Lecture 1 : [\[Bayesian statistics\]](#)
 - Practical work 1 : [\[Conjugate distributions\]](#) [\[Correction\]](#)
 - Lecture 2 : [\[Latent Variable Models and EM-algorithm\]](#)
 - Practical work 2 : [\[Probabilistic K-means and probabilistic PCA\]](#) [\[corr1, HTML\]](#) [\[corr1, ipynb\]](#)
 - Lecture 3 : [\[Variational Inference and intro to NLP\]](#)
 - Practical work 3 : [\[Topic Modeling with LDA\]](#) [\[No correction\]](#)
 - Lecture 4 : [\[Markov Chain Monte Carlo\]](#)
 - Practical work 4 : [\[MCMC samples\]](#)
 - Lecture 5 : (soon available)

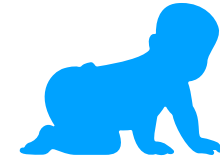


TODO



Road map

Bayesian statistics (03/05/21)



1

Bayesian perspective :

$$P(\theta|X) = \frac{P(X, \theta)}{P(X)} = \frac{\overset{\text{Likelihood}}{P(X|\theta)} \cdot \overset{\text{Prior distribution}}{P(\theta)}}{\underset{\text{Evidence}}{P(X)}}$$

Posterior distribution

θ parameters

X observations

Example :
Naive Bayes classifier,
Linear regression,

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

Conjugate distribution

Pros :

- exact posterior

Cons :

- conjugate prior maybe inadequate

Hard to compute !

Latent variable models (17/05/21)



2

Hidden variable models :

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

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

Example :
GMM, K-means, PCA/PPCA

Pros :

- fewer parameters / simpler models
- hidden variable sometimes meaningful
- clustering / dimensionality reduction

Cons :

- harder to work with
- requires math
- only local maximum or saddle point
- EM : the posterior of T could be intractable

Variational Inference (07/06/21)

3

Deterministic approximation of posterior :

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

Mean Field Approximation !

Example :

Topic modelling, LDA trained by VI

Pros :

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

Cons :

- can never generate exact result

Markov Chain Monte Carlo (14/06/21)

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 !

Example :

Topic modelling, LDA trained by MCMC

Pros :

- train / inference almost every probabilistic model
- asymptotically exact
- suited to small / medium dataset

Cons :

- Usually slower than alternatives (VI)
- can generate dependant samples from the distribution

Extensions (28/06/21)

5