14. Sampling Methods
Sampling Methods

Sampling Methods are widely used in Computer Science

- as an **approximation** of a deterministic algorithm
- to represent **uncertainty** without a parametric model
- to obtain higher computational **efficiency** with a small approximation error

Sampling Methods are also often called **Monte Carlo Methods**

Example: Monte-Carlo Integration

- Sample in the bounding box
- Compute fraction of inliers
- Multiply fraction with box size
Non-Parametric Representation

Probability distributions (e.g. a robot’s belief) can be represented:

- **Parametrically**: e.g. using mean and covariance of a Gaussian
- **Non-parametrically**: using a set of hypotheses (samples) drawn from the distribution

Advantage of non-parametric representation:

- No restriction on the type of distribution (e.g. can be multi-modal, non-Gaussian, etc.)
The more samples are in an interval, the higher the probability of that interval

But:

How to draw samples from a function/distribution?
Sampling from a Distribution

There are several approaches:

- Probability transformation
  - Uses inverse of the c.d.f \( h \)
- Rejection Sampling
- Importance Sampling
- MCMC

Probability transformation:
- Sample uniformly in \([0,1]\)
- Transform using \( h^{-1} \)

But:
- Requires calculation of \( h \) and its inverse

\[
h(y) = \int_{-\infty}^{y} p(\hat{y}) d\hat{y}
\]
Rejection Sampling

1. Simplification:
   - Assume $p(z) < 1$ for all $z$
   - Sample $z$ uniformly
   - Sample $c$ from $[0, 1]$

   • If $f(z) > c$ : keep the sample
   • otherwise: reject the sample
Rejection Sampling

2. General case:

Assume we can evaluate \( p(z) = \frac{1}{Z_p} \tilde{p}(z) \) (unnormalized)

- Find proposal distribution \( q \)
  - Easy to sample from \( q \)
- Find \( k \) with \( kq(z) \geq \tilde{p}(z) \)
- Sample from \( q \)
- Sample uniformly from \([0,kq(z_0)]\)
- Reject if \( u_0 > \tilde{p}(z_0) \)

**But:** Rejection sampling is inefficient.
Importance Sampling

• **Idea:** assign an importance weight $w$ to each sample

• With the importance weights, we can account for the “differences between $p$ and $q$”

$$w(x) = \frac{p(x)}{q(x)}$$

• $p$ is called **target**

• $q$ is called **proposal** (as before)
Importance Sampling

- **Explanation:** The prob. of falling in an interval $A$ is the **area** under $p$
- This is equal to the expectation of the **indicator function** $I(x \in A)$

\[
E_p[I(z \in A)] = \int p(z)I(z \in A)dz
\]
Importance Sampling

**Explanation:** The probability of falling in an interval $A$ is the area under $p$.

This is equal to the expectation of the indicator function $I(x \in A)$:

$$E_p[I(z \in A)] = \int p(z)I(z \in A)dz$$

$$= \int \frac{p(z)}{q(z)} q(z)I(z \in A)dz = E_q[w(z)I(z \in A)]$$

Requirement:

$$p(x) > 0 \implies q(x) > 0$$

Approximation with samples drawn from $q$:

$$E_q[w(z)I(z \in A)] \approx \frac{1}{L} \sum_{l=1}^{L} w(z_l)I(z_l \in A)$$
The Particle Filter

**Non-parametric** implementation of Bayes filter:

\[ \text{Bel}(x_t) = \eta \, p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) \text{Bel}(x_{t-1}) \, dx_{t-1} \]

- Represents the belief (posterior) \( \text{Bel}(x_t) \) by a set of random state samples.
- This representation is **approximate**.
- Can represent distributions that are **not Gaussian**.
- Can model **non-linear** transformations.

Basic principle:

- Set of state hypotheses ("particles")
- "Survival-of-the-fittest"
The Bayes Filter Algorithm (Rep.)

\[ \text{Bel}(x_t) = \eta \ p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \text{Bel}(x_{t-1}) \, dx_{t-1} \]

Algorithm Bayes_filter \((\text{Bel}(x), d)\)

1. if \(d\) is a sensor measurement \(z\) then
2. \(\eta = 0\)
3. for all \(x\) do
4. \(\text{Bel}'(x) \leftarrow p(z \mid x) \text{Bel}(x)\)
5. \(\eta \leftarrow \eta + \text{Bel}'(x)\)
6. for all \(x\) do \(\text{Bel}'(x) \leftarrow \eta^{-1} \text{Bel}'(x)\)
7. else if \(d\) is an action \(u\) then
8. for all \(x\) do \(\text{Bel}'(x) \leftarrow \int p(x \mid u, x') \text{Bel}(x') \, dx'\)
9. return \(\text{Bel}'(x)\)
Mathematical Description

Set of weighted samples:

\[ \mathcal{X}_t := \{ \langle x_t^{[1]}, w_t^{[1]} \rangle, \langle x_t^{[2]}, w_t^{[2]} \rangle, \ldots, \langle x_t^{[M]}, w_t^{[M]} \rangle \} \]

- State hypotheses
- Importance weights

The samples represent the probability distribution:

\[ p(x) = \sum_{i=1}^{M} w_t^{[i]} \cdot \delta_{x_t^{[i]}}(x) \]

Point mass distribution ("Dirac")
The Particle Filter Algorithm

Algorithm $\text{Particle\_filter}(x_{t-1}, u_t, z_t)$:

1. $\tilde{x}_t = x_t = \emptyset$
2. for $m = 1$ to $M$ do
3. sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
4. $w_t^{[m]} \leftarrow p(z_t \mid x_t^{[m]})$
5. $\tilde{x}_t \leftarrow \tilde{x}_t \cup \langle x_t^{[m]}, w_t^{[m]} \rangle$
6. for $m = 1$ to $M$ do
7. draw $i$ with prob. $\propto w_t^{[i]}$
8. $x_t \leftarrow x_t \cup \langle x_t^{[i]}, 1/M \rangle$
9. return $x_t$
Localization with Particle Filters

• Each particle is a potential pose of the robot
• Proposal distribution is the motion model of the robot (prediction step)
• The observation model is used to compute the importance weight (correction step)

Randomized algorithms are usually called Monte Carlo algorithms, therefore we call this:

Monte-Carlo Localization
A Simple Example

- The initial belief is a uniform distribution (global localization).
- This is represented by an (approximately) uniform sampling of initial particles.
Sensor Information

The sensor model $p(z_t | x_t^{[m]})$ is used to compute the new importance weights:

$$w_t^{[m]} \leftarrow p(z_t | x_t^{[m]})$$
After resampling and applying the motion model

\[ p(x_t | u_t, x_t^{[m], x_{t-1}}) \]

the particles are distributed more densely at three locations.
Again, we set the new importance weights equal to the sensor model.

\[ w_t^{[m]} \leftarrow p(z_t | x_t^{[m]}) \]
Robot Motion

Resampling and application of the motion model: One location of dense particles is left.

The robot is localized.
A Closer Look at the Algorithm...

Algorithm **Particle\_filter** \((\mathcal{X}_t, u_t, z_t)\):

1. \(\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset\)
2. for \(m = 1 \to M\) do
3. sample \(x_t^{[m]} \sim p(x_t | u_t, x_t^{[m-1]})\)
4. \(w_t^{[m]} \leftarrow p(z_t | x_t^{[m]})\)
5. \(\bar{\mathcal{X}}_t \leftarrow \mathcal{X}_t \cup \{x_t^{[m]}, w_t^{[m]}\}\)
6. for \(m = 1 \to M\) do
   - draw \(i\) with prob. \(\propto w_t^{[i]}\)
   - \(\mathcal{X}_t \leftarrow \mathcal{X}_t \cup x_t^{[i]}\)
7. return \(\mathcal{X}_t\)

Sample from proposal
Compute sample weights
Resampling
This can be done in the following ways:

- Adding the motion vector to each particle directly (this assumes perfect motion).
- Sampling from the motion model, e.g. for a 2D motion with translation velocity $v$ and rotation velocity $w$ we have:

$$p(x_t \mid u_t, x_{t-1}^{[m]})$$

$$u_t = \begin{pmatrix} v_t \\ w_t \end{pmatrix} \quad \quad x_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix}$$
Motion Model Sampling (Example)
Computation of Importance Weights

Computation of the sample weights:

- **Proposal distribution:** \( g(x_t^{[m]}) = p(x_t^{[m]} \mid u_t, x_{t-1}^{[m]}) \text{Bel}(x_{t-1}^{[m]}) \) (we sample from that using the motion model)

- **Target distribution (new belief):** \( f(x_t^{[m]}) = \text{Bel}(x_t^{[m]}) \) (we can not directly sample from that → importance sampling)

- **Computation of importance weights:**

\[
w_t^{[m]} = \frac{f(x_t^{[m]})}{g(x_t^{[m]})} \propto \frac{p(z_t \mid x_t^{[m]})p(x_t^{[m]} \mid u_t, x_{t-1}^{[m]}) \text{Bel}(x_{t-1}^{[m]})}{p(x_t^{[m]} \mid u_t, x_{t-1}^{[m]}) \text{Bel}(x_{t-1}^{[m]})} = p(z_t \mid x_t^{[m]})
\]
Proximity Sensor Models

- How can we obtain the sensor model \( p(z_t | x^m_t) \)?
- Sensor Calibration:

\[\text{Laser sensor} \quad \text{Sonar sensor}\]
Resampling

- Given: Set $\tilde{X}_t$ of weighted samples.
- Wanted: Random sample, where the probability of drawing $x_i$ is equal to $w_i$.
- Typically done $M$ times with replacement to generate new sample set $\tilde{X}_t$. 

```plaintext
for $m = 1$ to $M$ do
    draw $i$ with prob. $\propto w_t^{[i]}$
    $X_t \leftarrow X_t \cup x_t^{[i]}$
```

$\tilde{X}_t$
Resampling

- Standard n-times sampling results in high variance
- This requires more particles
- $O(n \log n)$ complexity

- Instead: low variance sampling only samples once
- Linear time complexity
- Easy to implement
Initial Distribution
After Ten Ultrasound Scans
After 65 Ultrasound Scans
Estimated Path
Kidnapped Robot Problem

The approach described so far is able to

• track the pose of a mobile robot and to
• globally localize the robot.

• How can we deal with localization errors (i.e.,
the kidnapped robot problem)?

Idea: Introduce uniform samples at every
resampling step

• This adds new hypotheses
Summary

• There are mainly 4 different types of sampling methods: Transformation method, rejections sampling, importance sampling and MCMC

• Transformation only rarely applicable

• Rejection sampling is often very inefficient

• Importance sampling is used in the particle filter which can be used for robot localization

• An efficient implementation of the resampling step is the low variance sampling
Markov Chain Monte Carlo
Markov Chain Monte Carlo

• In high-dimensional spaces, rejection sampling and importance sampling are very inefficient
• An alternative is Markov Chain Monte Carlo (MCMC)
• It keeps a record of the current state and the proposal depends on that state
• Most common algorithms are the Metropolis-Hastings algorithm and Gibbs Sampling
Markov Chains Revisited

A Markov Chain is a distribution over discrete-state random variables \( x_1, \ldots, x_M \) so that

\[
p(x_1, \ldots, x_T) = p(x_1)p(x_2 \mid x_1) \cdots = p(x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1})
\]

The graphical model of a Markov chain is this:

We will denote \( p(x_t \mid x_{t-1}) \) as a row vector \( \pi_t \)

A Markov chain can also be visualized as a state transition diagram.
The State Transition Diagram

Time

States

k=1

A_{11}

k=2

A_{33}

k=3

A_{33}

states

A_{11}

A_{33}

......

......

......

A_{33}

A_{11}

......

......

......

A_{33}

A_{11}

......

......

......

A_{33}

A_{11}
Some Notions

• The Markov chain is said to be **homogeneous** if the transitions probabilities are all the same at every time step $t$ (here we only consider homogeneous Markov chains).

• The transition matrix is **row-stochastic**, i.e. all entries are between 0 and 1 and all rows sum up to 1.

• Observation: the probabilities of reaching the states can be computed using a vector-matrix multiplication.
The Stationary Distribution

The probability to reach state \( k \) is

\[
\pi_{k,t} = \sum_{i=1}^{K} \pi_{i,t-1} A_{ik}
\]

Or, in matrix notation:

\[
\pi_t = \pi_{t-1} A
\]

We say that \( \pi_t \) is **stationary** if

\[
\pi_t = \pi_{t-1}
\]

Questions:

• How can we know that a stationary distributions exists?
• And if it exists, how do we know that it is unique?
The Stationary Distribution (Existence)

To find a stationary distribution we need to solve the eigenvector problem $A^T v = v$

The stationary distribution is then $\pi = v^T$ where $v$ is the eigenvector for which the eigenvalue is 1. This eigenvector needs to be normalized so that it is a valid distribution.

**Theorem:** Every row-stochastic matrix has such an eigen vector, but this vector may not be unique.

Proof based on Perron-Frobenius.
Stationary Distribution (Uniqueness)

• A Markov chain can have many stationary distributions

• Sufficient for a unique stationary distribution: we can reach every state from any other state in finite steps at non-zero probability, i.e. the chain is **ergodic** (without proof)

• This is equivalent to the property that the transition matrix is **irreducible**: 

\[ \forall i, j \ \exists m \quad (A^m)_{ij} > 0 \]
Main Idea of MCMC

• So far, we specified the transition probabilities and analysed the resulting distribution.
• This was used, e.g. in HMMs.

Now:

• We want to sample from an arbitrary distribution.
• To do that, we design the transition probabilities so that the resulting stationary distribution is our desired (target) distribution!
**Definition:** A transition distribution $\pi_t$ satisfies the property of **detailed balance** if $\pi_i A_{ij} = \pi_j A_{ji}$

The chain is then said to be **reversible**.
Making a Distribution Stationary

**Theorem:** If a Markov chain with transition matrix $A$ is irreducible and satisfies detailed balance wrt. the distribution $\pi$, then $\pi$ is a stationary distribution of the chain.

**Proof:**

\[
\sum_{i=1}^{K} \pi_i A_{ij} = \sum_{i=1}^{K} \pi_j A_{ji} = \pi_j \sum_{i=1}^{K} A_{ji} = \pi_j \quad \forall j
\]

it follows $\pi = \pi A$.

This is a sufficient, but not necessary condition.
Sampling with a Markov Chain

The idea of MCMC is to sample state transitions based on a **proposal distribution** \( q \).

The most widely used algorithm is the Metropolis-Hastings (MH) algorithm.

In MH, the decision whether to stay in a given state is based on a given probability.

If the proposal distribution is \( q(x' \mid x) \), then we stay in state \( x' \) with probability

\[
\min \left(1, \frac{\tilde{p}(x')q(x \mid x')}{\tilde{p}(x)q(x' \mid x)} \right)
\]

Unnormalized target distribution
The Metropolis-Hastings Algorithm

- Initialize $x^0$
- for $s = 0, 1, 2, \ldots$
  - define $x = x^s$
  - sample $x' \sim q(x' \mid x)$
  - compute acceptance probability
    $$\alpha = \frac{\tilde{p}(x') q(x \mid x')}{\tilde{p}(x) q(x' \mid x)}$$
  - compute $r = \min(1, \alpha)$
  - sample $u \sim U(0, 1)$
  - set new sample to
    $$x^{s+1} = \begin{cases} 
    x' & \text{if } u < r \\
    x^s & \text{if } u \geq r
    \end{cases}$$

Aim: create samples from (unnormalized) distribution $\tilde{p}$
Why Does This Work?

We have to prove that the transition probability of the MH algorithm satisfies detailed balance wrt the target distribution.

**Theorem:** If \( p_{MH}(x' \mid x) \) is the transition probability of the MH algorithm, then

\[
p(x)p_{MH}(x' \mid x) = p(x')p_{MH}(x \mid x')
\]

**Proof:**
Why Does This Work?

We have to prove that the transition probability of the MH algorithm satisfies detailed balance wrt the target distribution.

**Theorem:** If \( p_{MH}(x' | x) \) is the transition probability of the MH algorithm, then

\[
p(x)p_{MH}(x' | x) = p(x')p_{MH}(x | x')
\]

**Note:** All formulations are valid for discrete and for continuous variables!
Choosing the Proposal

• A proposal distribution is valid if it gives a non-zero probability of moving to the states that have a non-zero probability in the target.

• A good proposal is the Gaussian, because it has a non-zero probability for all states.

• **However:** the variance of the Gaussian is important!
  • with low variance, the sampler does not explore sufficiently, e.g. it is fixed to a particular mode
  • with too high variance, the proposal is rejected too often, the samples are a bad approximation
Target is a mixture of 2 1D Gaussians.
Proposal is a Gaussian with different variances.
Gibbs Sampling

- Initialize \( \{ z_i : i = 1, \ldots, M \} \)
- For \( \tau = 1, \ldots, T \)
  - Sample \( z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, \ldots, z_M^{(\tau)}) \)
  - Sample \( z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, \ldots, z_M^{(\tau)}) \)
  - ...
  - Sample \( z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)}) \)

**Idea:** sample from the full conditional

This can be obtained, e.g. from the Markov blanket in graphical models.
**Gibbs Sampling is a Special Case of MH**

- The proposal distribution in Gibbs sampling is

\[
q(x' | x) = p(x'_i | x_{-i}) \mathbb{I}(x'_i = x_{-i})
\]

- This leads to an acceptance rate of:

\[
\alpha = \frac{p(x')q(x | x')}{p(x)q(x' | x)} = \frac{p(x'_i | x_{-i})p(x'_{-i})p(x_i | x_{-i})}{p(x_i | x_{-i})p(x_{-i})p(x'_i | x_{-i})} = 1
\]

- Although the acceptance is 100%, Gibbs sampling does not converge faster, as it only updates one variable at a time.
Gibbs Sampling: Example

- Use an MRF on a binary image with edge potentials $\psi(x_s, x_t) = \exp(J x_s x_t)$ ("Ising model") and node potentials $\psi(x_t) = \mathcal{N}(y_t | x_t, \sigma^2)$

$x_t \in \{-1, 1\}$
Gibbs Sampling: Example

- Use an MRF on a binary image with edge potentials \( \psi(x_s, x_t) = \exp(J x_s x_t) \) ("Ising model") and node potentials \( \psi(x_t) = \mathcal{N}(y_t | x_t, \sigma^2) \)

- Sample each pixel in turn

![Sample 1, Gibbs](image1)

![Sample 5, Gibbs](image2)

![Mean after 15 sweeps of Gibbs](image3)

After 1 sample

After 5 samples

Average after 15 samples
Gibbs Sampling for GMMs

- Again, we start with the full joint distribution:

\[ p(X, Z, \mu, \Sigma, \pi) = p(X \mid Z, \mu, \Sigma)p(Z \mid \pi)p(\pi) \prod_{k=1}^{K} p(\mu_k)p(\Sigma_k) \]

- It can be shown that the full conditionals are:

\[ p(z_i = k \mid x_i, \mu, \Sigma, \pi) \propto \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \]

\[ p(\pi \mid z) = \text{Dir}(\{\alpha_k + \sum_{i=1}^{K} z_{ik}\}_{k=1}^{K}) \]

\[ p(\mu_k \mid \Sigma_k, Z, X) = \mathcal{N}(\mu_k \mid m_k, V_k) \quad \text{(linear-Gaussian)} \]

\[ p(\Sigma_k \mid \mu_k, Z, X) = \mathcal{IW}(\Sigma_k \mid S_k, \nu_k) \]
Gibbs Sampling for GMMs

- First, we initialize all variables
- Then we iterate over sampling from each conditional in turn
- In the end, we look at $\mu_k$ and $\Sigma_k$
How Often Do We Have To Sample?

- Here: after 50 sample rounds the values don’t change any more

- In general, the **mixing time** $\tau_\epsilon$ is related to the **eigen gap** $\gamma = \lambda_1 - \lambda_2$ of the transition matrix:

$$\tau_\epsilon \leq O\left(\frac{1}{\gamma} \log \frac{n}{\epsilon}\right)$$
Summary

• Markov Chain Monte Carlo is a family of sampling algorithms that can sample from arbitrary distributions by moving in state space.

• Most used methods are the Metropolis-Hastings (MH) and the Gibbs sampling method.

• MH uses a proposal distribution and accepts a proposed state randomly.

• Gibbs sampling does not use a proposal distribution, but samples from the full conditionals.