Friedrich-Alexander-Universität
Technische Fakultät

Machine Learning
Data Analytics

# Machine Learning for Time Series

## (MLTS or MLTS-Deluxe Lectures)

Dr. Dario Zanca

**Machine Learning and Data Analytics (MaD) Lab**
**Friedrich-Alexander-Universität Erlangen-Nürnberg**
**30.11.2023**

# Topics overview

- Time series fundamentals and definitions (2 lectures)
- Bayesian Inference (1 lecture)
- Gussian processes (2 lectures)
- State space models (2 lectures) ←
- Autoregressive models (1 lecture)
- Data mining on time series (1 lecture)
- Deep learning on time series (4 lectures)
- Domain adaptation (1 lecture)

## Review concept: State Space Models (SSMs)

**SSMs** are characterized by:

- Transition density: $p(z_n|z_{n-1})$
- Observation density: $p(y_n|z_n)$

**Joint density** can be expressed using the *chain rule*:

$$p(z_{1:n}|y_{1:n}) = p(z_1)\prod_{i=2}^{n} p(z_i|z_{i-1})\prod_{i=1}^{n} p(y_i|z_i)$$

where $z_{1:n} = (z_1, \ldots, z_n)$ and $y_{1:n} = (y_1, \ldots, y_n)$.

**Filtering** is the task of estimating $p(z_n|y_{1:n})$.

## Review concept: Bayesian filtering

Let $p(z_{n-1}|y_{1:n-1})$ be the filtering density at time step $n-1$ and we wish to determine $p(z_n|y_{1:n})$.

We can use the following iterative steps:

- Prediction step:

$$p(z_n|y_{1:n-1}) = \int p(z_n|z_{n-1})\ p(z_{n-1}|y_{1:n-1})dz_n$$

- Correction step:

$$p(z_n|y_{1:n}) = \frac{p(y_n,|z_n)\, p(z_n|y_{1:n-1})}{p(y_n|y_{1:n-1})}$$

If the variables are normally distributed and the transitions are linear, the Bayes filter becomes equal to the Kalman filter.

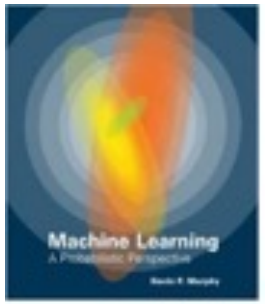
## In this lecture…

1. **Monte Carlo methods**
2. **Particle filtering: theory**
3. **Particle filtering: example and algorithmic view**

## References

**Machine learning: A Probabilistic Perspective,**

by Kevin Murphy (2012)



# Sequential Monte Carlo and Particle Filtering

## Monte Carlo methods

## Motivation

The fundamental goal of this section is **"how to find the expectation of a function $f(z)$ with respect to a probability distribution $p(z)$".**

$$\mathbb{E}[f] = \int f(z)p(z)dz$$

## Motivation

The fundamental goal of this section is **"how to find the expectation of a function $f(z)$ with respect to a probability distribution $p(z)$".**

$$\mathbb{E}[f] = \int f(z)p(z)dz$$

**Sampling methods** can be used to approximate this expectation using a set of samples $z^s \sim p(z)$, by the finite sum:

$$\frac{1}{S}\sum_{s=1}^{S} f(z^s)$$

where $S \in \mathbb{Z}^+$.

## Motivation

However, possible problems are:

- Samples $z^S$ might not be independent
- Depending on $f$, expectation might be dominated by examples with small probability

→ We need relatively large sample size $S$.

↳ need large sample size S.

# Monte Carlo methods (MC)

random sampling

Monte Carlo methods include a wide class of **algorithms that rely on random sampling** to obtain numerical results.

- Use randomness to solve problem that might be deterministic in principle.
- Used, e.g., in optimization, numerical integration and for generating draws from a probability distribution.

## General idea of MC methods:

- Define a domain of possible inputs → Domain
- Generate input randomly from a probability distribution → Random i/p
- Perform deterministic computations on the inputs → Deterministic computations
- Aggregate results → Aggregate result

## Rejection sampling

Rejection sampling is a Monte Carlo algorithm to sample data from a sophisticated ("difficult to sample from") distribution with the help of a proxy distribution.

Let us suppose that:

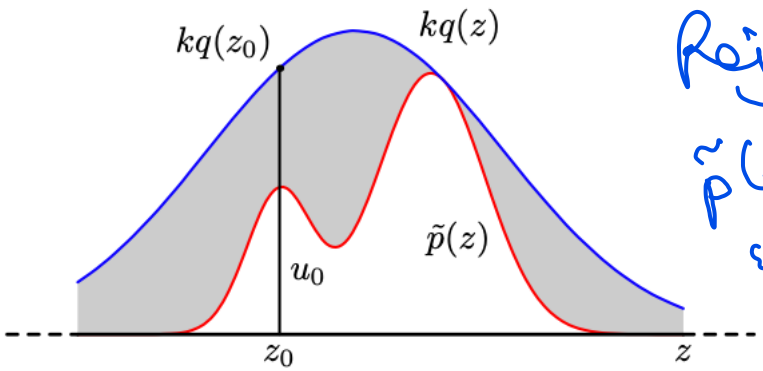
- We wish to sample from a "non-standard" distribution $p(z)$
- Sampling directly from $p(z)$ is difficult
- We are able to evaluate $\mathrm{p}(z)$ for any $z$, up to a certain (unknown) normalizing constant $Z_p$:

$$\mathrm{p}(z) = \tilde{p}(z)/Z_p$$

To apply rejection sampling, we make use of a simpler distribution $q(z)$, also called **proposal distribution** → We are able to readily draw samples from $q(z)$

## Rejection sampling

**Samples are generated** from the proposal distribution $q(z)$ and **rejected** if they fall between the unnormalized $\tilde{p}(z)$ and the scaled distribution $kq(z)$.




→ In the side figure, samples are rejected if they fall in the grey area.

→ Samples are accepted with probability $\frac{\tilde{p}(z)}{kq(z)}$

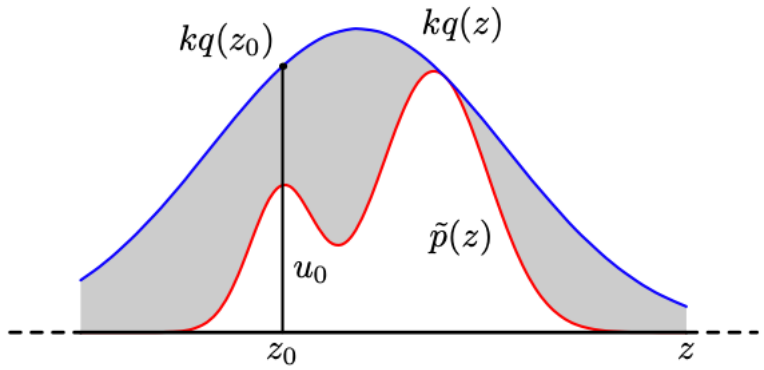
Note: $k$ is a constant value which is chosen such that $kq(z) \geq \tilde{p}(z)$ for all values of $z$.

Bishop, C. M. (2014). Bishop-Pattern Recognition and Machine Learning. Springer 2006.

# Rejection method

The success of the rejection method depend on the success in determining a suitable value for the constant $k$.

→ This is impractical in many cases, which leads to very small acceptance rates.




Bishop, C. M. (2014). Bishop-Pattern Recognition and Machine Learning. Springer 2006.

## Importance sampling

Importance sampling is a method which allows approximating expectations directly.

Suppose, similarly to the previous case, that:

- It is impractical to draw samples from $p(z)$
- But, we can evaluate $p(z)$ easily for any $z$

Uniform → inefficient
choose only significat samples

Uniform sampling from the space of $z$ is inefficient (high-dimensionality) and only few samples will have significant contribution.

→ We would like to chose samples from regions where $p(z)$ is large.

Bishop, C. M. (2014). Bishop-Pattern Recognition and Machine Learning. Springer 2006.

## Importance sampling

We draw samples from a proposal distribution $q(z)$. Then,

$$\mathbb{E}[f] = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \approx \frac{1}{S}\sum_{s=1}^{S}\frac{p(z^s)}{q(z^s)}f(z^s)$$

The quantity $\frac{p(z^s)}{q(z^s)}$ measures the importance of each sample and are called **importance weights**.

- Compared to rejection sampling, no samples are rejected.
- Importance sampling do not provide itself a mechanism for drawing samples from a distribution $q(z)$.

Bishop, C. M. (2014). Bishop-Pattern Recognition and Machine Learning. Springer 2006.

## Sampling-importance-resampling approach

The **sampling-importance-resampling** approach also makes use of a proposal distribution $q(z)$ and **avoids determining a constant $k$**.

It consists of there *general* steps:

1. Sampling of $\{z_1, \dots, z_s\}$ from the proposal distribution $q(z)$
2. Construction of importance weights $\{w_1, \dots, w_s\}$
3. Resampling from a discrete distribution with probabilities given by the weights

**Property.** The resulting samples are approximately distributed according to $p(z)$ and the distribution becomes correct for $S \to \infty$.

Bishop, C. M. (2014). Bishop-Pattern Recognition and Machine Learning. Springer 2006.

# Sequential Monte Carlo and Particle Filtering

## Particle Filtering (PF): Theory

# Motivations

| Estimator | State-transition / Measurement models assumptions | Assumed noise distribution | Computational cost |
|---|---|---|---|
| Kalman Filter | Linear | Gaussian | Low |
| Extended Kalman Filter | Non-linear (but locally linear) | Gaussian | Low / Medium (depending on the difficulty of computing the Jacobian) |
| Unscented Kalman Filter | Non-linear | Gaussian | Medium |

## Particle Filtering (PF)

→ MC for recursive Bayesian inference

We can use sampling-importance-resampling formalism to obtain a **sequential Monte Carlo**, also said **particle filtering**.

PF is a Monte Carlo (or simulation-based) approach for recursive Bayesian inference.

- It approximates the prediction-correction cycle
- It can be **used for not linear-Gaussian systems** to make tractable inference algorithms
- It is widely applied in many areas (e.g., tracking, forecasting, online parameter learning, …)

🔍 The term “particle filters” originated in in reference to mean-field interacting particle methods used in fluid mechanics since early 1960s (Del Moral, 1966).

## Particle Filtering (PF)

**First,** we update the belief state using importance sampling.

If the proposal distribution has the form $q(z_{1:n}^s|y_{1:n})$, then the importance weights are given by

$$w_n^s \propto \frac{p(z_{1:n}^s|y_{1:n})}{q(z_{1:n}^s|y_{1:n})}$$

which can be normalized as follow:

$$\hat{w}_n^s \propto \frac{w_n^s}{\sum_{s'} w_n^{s'}}$$

## Particle Filtering (PF)

We observe that we can rewrite the numerator recursively as follows:

$$p(z_{1:n}|y_{1:n}) = \frac{p(y_n|z_{1:n}, y_{1:n-1})p(z_{1:n}|y_{1:n-1})}{p(y_n|y_{1:t-1})}$$

$$= \frac{p(y_n|z_n)p(z_n|z_{1:n-1}, y_{1:n-1})p(z_{1:n-1}|y_{1:n-1})}{p(y_n|y_{1:t-1})}$$

$$\propto p(y_n|z_n)p(z_n|z_{n-1})p(z_{1:n-1}|y_{1:n-1})$$

Next only on recent past

Markov property

And, similarly, the denominator:

$$q(z_{1:n}|y_{1:n}) = q(z_n|z_{1:n-1}, y_{1:n})\, q(z_{1:n-1}|\, y_{1:n-1})$$

## Particle Filtering (PF)

Therefore, we use this formulation to derive an iterative update for the weights:

$$w_n^s \propto \frac{p(z_{1:n}^s|y_{1:n})}{q(z_{1:n}^s|y_{1:n})}$$

$$\propto \frac{p(y_n|z_n^s)p\left(z_n^s\middle|z_{n-1}^s\right)p(z_{1:n-1}^s|y_{1:n-1})}{q(z_n^s|z_{1:n-1}^s,y_{1:n})\ q(z_{1:n-1}^s|\ y_{1:n-1})}$$

$$= w_{n-1}^s \frac{p(y_n|z_n^s)p\left(z_n^s\middle|z_{n-1}^s\right)}{q(z_n^s|z_{1:n-1}^s,y_{1:n})}$$

posterior filtered density

Hence, we can approximate the posterior filtered density using

$$p(\mathrm{z_n}|\mathrm{y_{1:n}}) \approx \sum_{s=1}^{S} \widehat{w}_n^s \delta_{z_n^s}(z_n)$$

## Particle Filtering: the degeneracy problem

The basic sequential importance fails after a few steps because most of the particles will have negligible weights.

→ This problem is know as **degeneracy problem** and it occurs when sampling in high dimensional spaces

We can quantify the degree of degeneracy by using the effective sampling size, defined by:

$$S_{eff} = \frac{1}{\sum_{s=1}^{S} (w_n^s)^2}$$

When the variance of the weights is too large, we are wasting resources updating particles with negligible weights.

## Particle filtering: the resampling step

The degeneracy problem can be solved by **adding a resampling step.**

Whenever the effective sampling size $S_{eff}$ drops below a threshold:

- we eliminate particles with low weights and → eliminate low weight particles
- we create replicates of the survival particles. → create replicas of survival particles.

(re-sampling step)

In particular, we generate a new set $\{z_n^{s*}\}_{s=1}^{S}$ by sampling replacement $S$ times according to the weighted distribution obtained previously $\sum_{s=1}^{S} \hat{w}_n^s \delta_{z_n^s}(z_n)$.

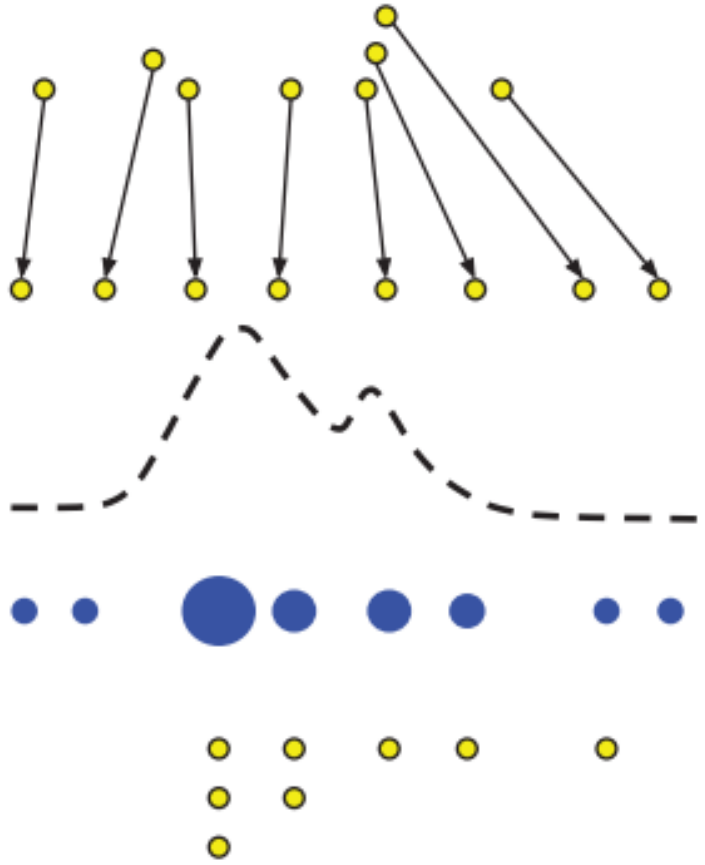
The result is an *i.i.d. unweighted sample* from the discrete density, so we set the new weights to $w_n^s = 1/S$. ← new weights.

# PF: the overall scheme

Proposal

Weighting

Resampling



$p(z_{1:n-1}|y_{1:n-1})$

$p(z_{1:n}|y_{1:n-1})$

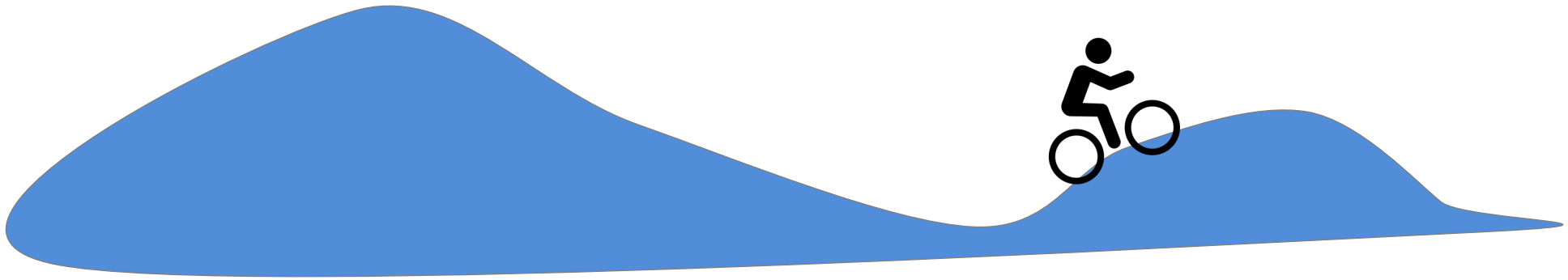
$p(y_n\ |z_n)$

$p(z_n\ |y_{1:n})$

# Sequential Monte Carlo and Particle Filtering

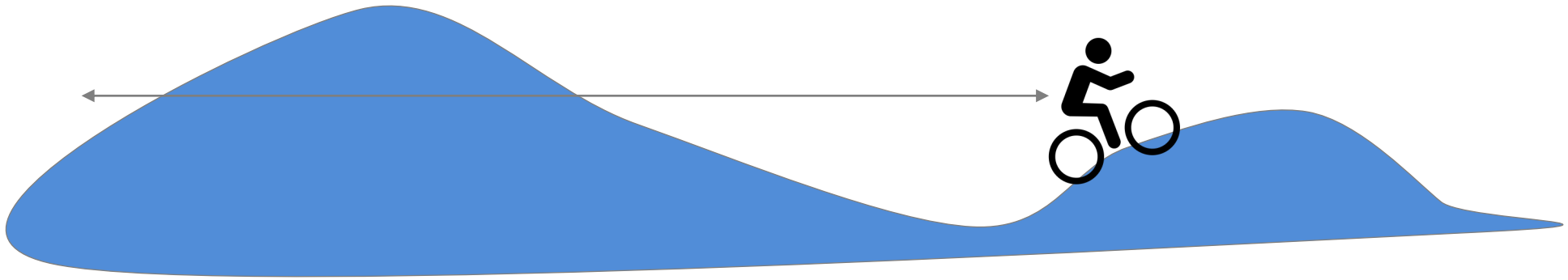
## Particle Filtering (PF): Example and Algorithmic View

# PF: Example



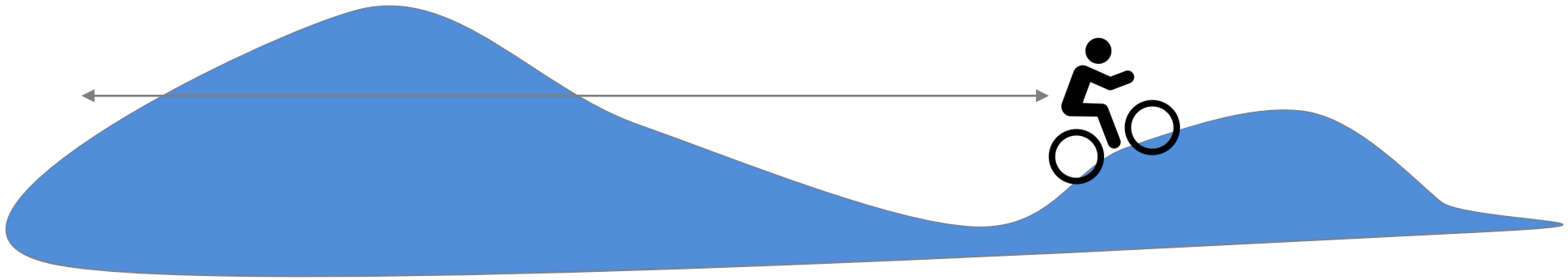
## PF: Example

1. We want to estimate the **horizontal position** of the cyclist.



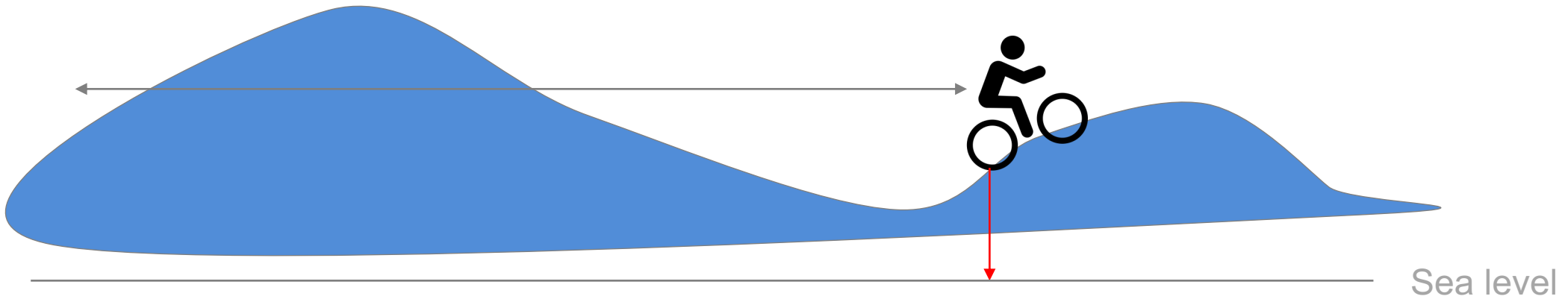
## PF: Example

1. We want to estimate the **horizontal position** of the cyclist.
2. We have **knowledge** about the hills' morphology

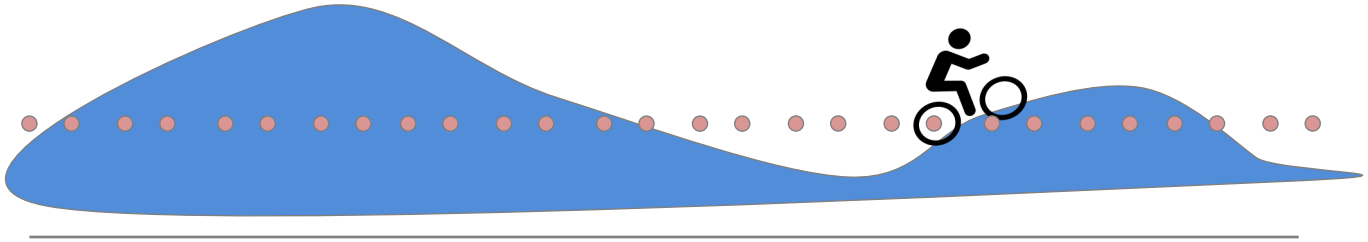


## PF: Example

1. We want to estimate the **horizontal position** of the cyclist.
2. We have **knowledge** about the hills' morphology
3. We receive noisy **measures of the altitude** (e.g., in relation to sea level).



# PF: Example



Particles (time $n$)

We draw a **set of particles** $\langle z_n^s, w_n^s \rangle_{s \in \{1, \ldots, S\}}$

(set of particles)

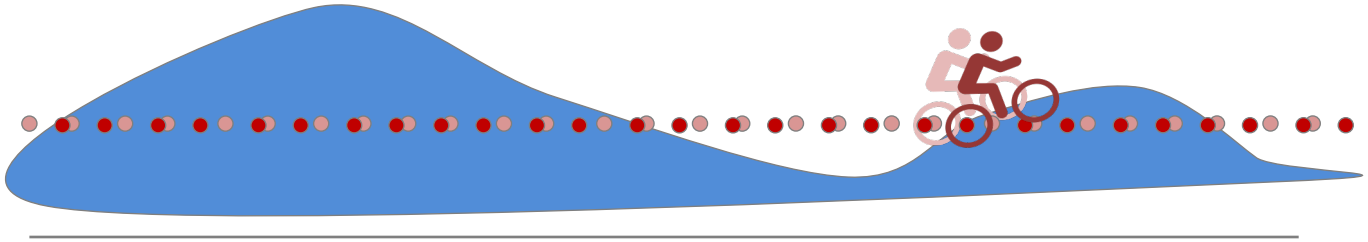
- $n$ is the time index,
- $z_n^s$ is a state hypothesis,
- $w_n^s$ corresponding weights.

prior probability.

These samples represent the prior probability:

$$p(z_n | y_1, \ldots, y_n) \simeq \sum_{s=1}^{S} w_n^s * \delta(z_n^s)$$

# PF: Example



Particles (time $n$)

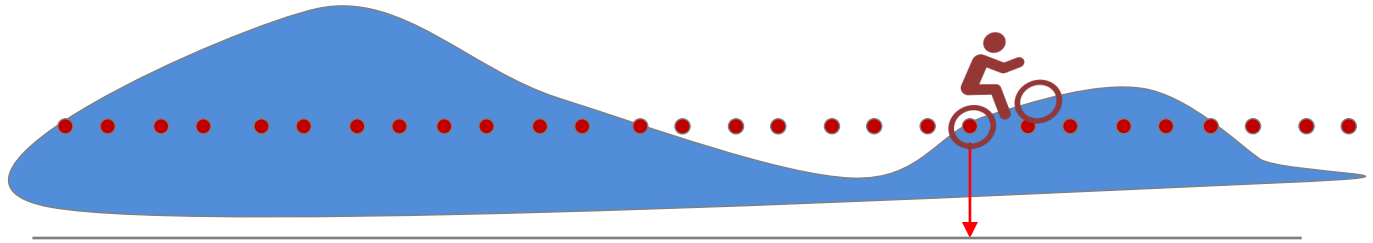
Propagated Particles (time $n+1$)

transition model

In the **prediction step,** we use our transition model $f$ to **propagate** particles forward in time:

$$z_{n+1}^{s} = f(z_n^{s}) + \epsilon$$

# PF: Example



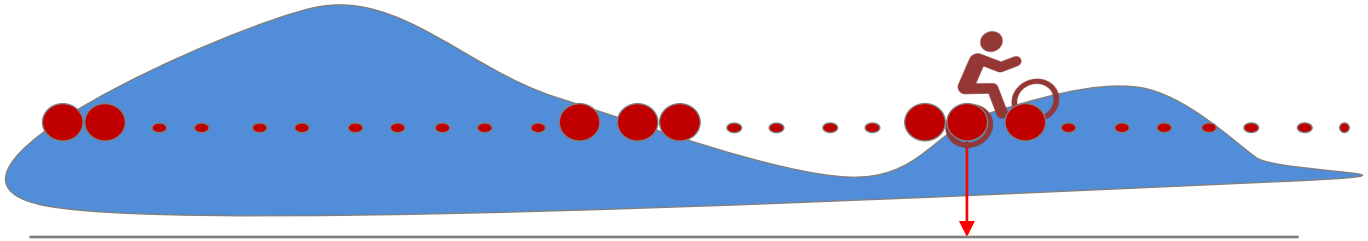
Sensory information

In the **correction step,** we **compute particle weights** based on the new sensory information $y_{n+1}$:

$$w_{n+1}^s = w_n^s * p(y_{n+1} | z_{n+1}^s)$$

↳ particle weights

# PF: Example



In the **correction step,** we **compute particle weights** based on the new sensory information $y_{n+1}$:

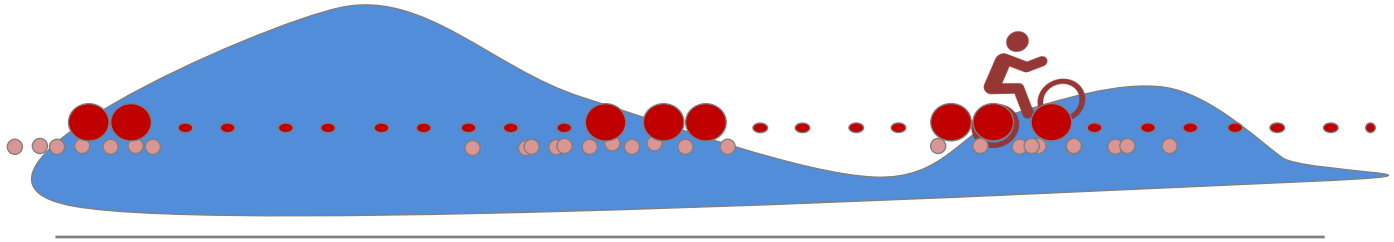
$$w_{n+1}^s = w_n^s * p(y_{n+1} | z_{n+1}^s)$$

distribution $p(z_{t+1} | y_{1:t+1})$ is particle set: $\langle z_{n+1}^s, w_{n+1}^s \rangle_{s \in \{1,\ldots,S\}}$

Now the distribution $p(z_{n+1} | y_{1:t+1})$ is represented by the particle set:

$$\langle z_{n+1}^s, w_{n+1}^s \rangle_{s \in \{1, \ldots, S\}}$$

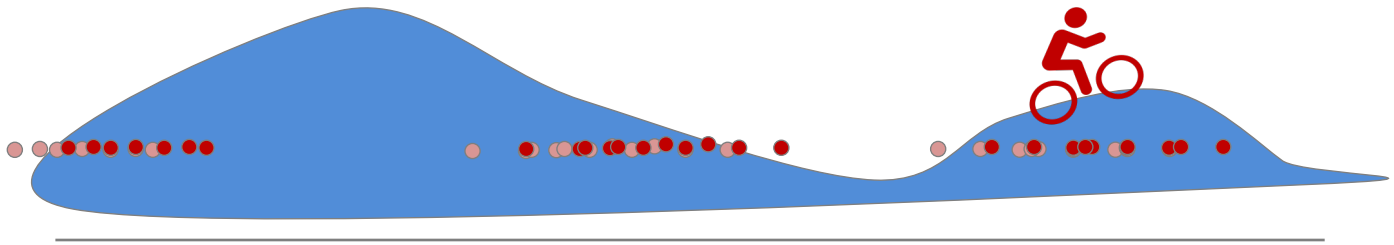
# PF: Example



We **repeat** the previous steps:

- Sampling again from the new distribution (we have more particles close to the more likely state's hypothesis).

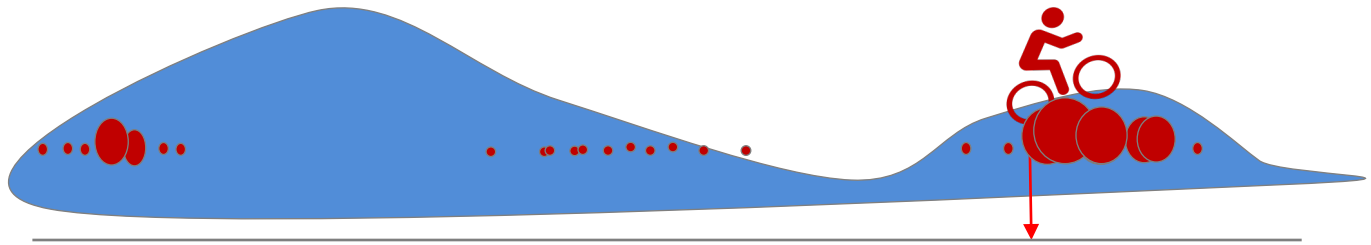
# PF: Example



We **repeat** the previous steps:

- Sampling again from the new distribution (we have more particles close to the more likely state's hypothesis).
- We propagate again through time.

## PF: Example



We **repeat** the previous steps:

close particles ↳ state's hypothesis.

- Sampling again from the new distribution (we have more particles close to the more likely state's hypothesis).
- We propagate again through time.
- We perform again a correction, based on the new measurement, and update the particles' weights.
- …

# PF: An algorithmic view

**Initially sample $S$ particles $z_1^S$**

**For $n = 1, \dots, N$**

For $s = 1, \dots, S$

Draw $z_n^s \sim q(z_n^s | z_{n-1}^s, y_n)$

update and normalize $w_{n+1}^s$

if $S_{eff} < S_{min}$:

Resample particles

Re-initialize weights

End

End

**End**

# Sequential Monte Carlo and Particle Filtering

## Recap

## Recap

- Monte Carlo methods
  - Rejection method
  - Importance sampling
  - Sampling-importance-resampling approach
- Particle filtering

# Particle Filtering: pros and cons

**Advantages:**

- Ability to represent arbitrary densities
- Adaptive focusing on probable regions of state-space
- Dealing with non-Gaussian noise

**Disadvantages:**

- High computational complexity
- It is difficult to determine optimal number of particles
- Number of particles increase with increasing model dimension
- Potential problems: degeneracy