The EP Algorithm

• Given: a joint distribution over data and variables

\[ p(D, \theta) = \prod_{i=1}^{M} f_i(\theta) \]

• Goal: approximate the posterior \( p(\theta \mid D) \) with \( q \)

• Initialize all approximating factors \( \tilde{f}_i(\theta) \)

• Initialize the posterior approximation \( q(\theta) \propto \prod_i \tilde{f}_i(\theta) \)

• Do until convergence:
  • choose a factor \( \tilde{f}_j(\theta) \)
  • remove the factor from \( q \) by division:

\[ q_{\setminus j}(\theta) = \frac{q(\theta)}{\tilde{f}_j(\theta)} \]
The EP Algorithm

• find $q^{\text{new}}$ that minimizes

$$
\text{KL} \left( \frac{f_j(\theta)q^{\backslash j}(\theta)}{Z_j} \bigg| q^{\text{new}}(\theta) \right)
$$

using moment matching, including the zero-th moment:

$$
Z_j = \int q^{\backslash j}(\theta)f_j(\theta)d\theta
$$

• evaluate the new factor

$$
\tilde{f}_j(\theta) = Z_j \frac{q^{\text{new}}(\theta)}{q^{\backslash j}(\theta)}
$$

• After convergence, we have

$$
p(D) \approx \int \prod_i \tilde{f}_j(\theta)d\theta
$$
yellow: original distribution
red: Laplace approximation
green: global variation
blue: expectation-propagation
The Clutter Problem

- Aim: fit a multivariate Gaussian into data in the presence of background clutter (also Gaussian)

\[ p(x \mid \theta) = (1 - w)\mathcal{N}(x \mid \theta, I) + w\mathcal{N}(x \mid 0, aI) \]

- The prior is Gaussian:

\[ p(\theta) = \mathcal{N}(\theta \mid 0, bI) \]
The Clutter Problem

The joint distribution for \( D = (x_1, \ldots, x_N) \) is

\[
p(D, \theta) = p(\theta) \prod_{n=1}^{N} p(x_n | \theta)
\]

this is a mixture of \( 2^N \) Gaussians! This is intractable for large \( N \). Instead, we approximate it using a spherical Gaussian:

\[
q(\theta) = \mathcal{N}(\theta | m, \nu I) = f_0(\theta) \prod_{n=1}^{N} f_n(\theta)
\]

the factors are (unnormalized) Gaussians:

\[
\tilde{f}_0(\theta) = p(\theta) \quad \tilde{f}_n(\theta) = s_n \mathcal{N}(\theta | m_n, \nu_n I)
\]
EP for the Clutter Problem

• First, we initialize $\tilde{f}_n(\theta) = 1$, i.e. $q(\theta) = p(\theta)$

• Iterate:
  • Remove the current estimate of $\tilde{f}_n(\theta)$ from $q$ by division of Gaussians:

  $$q_{-n}(\theta) = \frac{q(\theta)}{\tilde{f}_n(\theta)}$$
EP for the Clutter Problem

• First, we initialize  \( \tilde{f}_n(\theta) = 1 \), i.e.  \( q(\theta) = p(\theta) \)

• Iterate:
  • Remove the current estimate of  \( \tilde{f}_n(\theta) \) from  \( q \) by division of Gaussians:
    \[
    q_{-n}(\theta) = \frac{q(\theta)}{\tilde{f}_n(\theta)} \quad q_{-n}(\theta) = \mathcal{N}(\theta \mid m_{-n}, v_{-n}I)
    \]

• Compute the normalization constant:
    \[
    Z_n = \int q_{-n}(\theta) f_n(\theta) d\theta
    = (1 - w)\mathcal{N}(x_n \mid m_{-n}, (v_{-n} + 1)I) + w\mathcal{N}(x_n \mid 0, aI)
    \]
EP for the Clutter Problem

• First, we initialize $\tilde{f}_n(\theta) = 1$, i.e. $q(\theta) = p(\theta)$

• Iterate:
  • Remove the current estimate of $\tilde{f}_n(\theta)$ from $q$ by division of Gaussians:
    \[
    q_{-n}(\theta) = \frac{q(\theta)}{\tilde{f}_n(\theta)} \quad q_{-n}(\theta) = \mathcal{N}(\theta \mid \mathbf{m}_{-n}, \mathbf{v}_{-n}I)
    \]
  • Compute the normalization constant:
    \[
    Z_n = \int q_{-n}(\theta) f_n(\theta) \, d\theta
    \]
  • Compute mean and variance of $q^{\text{new}}(\theta) = q_{-n}(\theta) f_n(\theta)$
  • Update the factor $\tilde{f}_n(\theta) = Z_n \frac{q^{\text{new}}(\theta)}{q_{-n}(\theta)}$
A 1D Example

- blue: true factor $f_n(\theta)$
- red: approximate factor $\tilde{f}_n(\theta)$
- green: cavity distribution $q_{-n}(\theta)$

The form of $q_{-n}(\theta)$ controls the range over which $\tilde{f}_n(\theta)$ will be a good approximation of $f_n(\theta)$
Summary

• Variational Inference uses approximation of functions so that the KL-divergence is minimal

• In mean-field theory, factors are optimized sequentially by taking the expectation over all other variables

• Variational inference for GMMs reduces the risk of overfitting; it is essentially an EM-like algorithm

• Expectation propagation minimizes the reverse KL-divergence of a single factor by moment matching; factors are in the exp. family
10. 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
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 prob. of falling in an interval $A$ is the area under $p$.
- This is equal to the expectation of the indicator function $I(x \in A)$

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

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

Requirement: $p(x) > 0 \Rightarrow q(x) > 0$

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
- 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 \hspace{2cm} 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. \( \text{for } m = 1 \text{ to } M \text{ do} \)
3. \( \text{sample } x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]}) \)
4. \( w_t^{[m]} \leftarrow p(z_t | x_t^{[m]}) \)
5. \( \tilde{x}_t \leftarrow \tilde{x}_t \cup \langle x_t^{[m]}, w_t^{[m]} \rangle \)
6. \( \text{for } m = 1 \text{ to } M \text{ do} \)
\( \text{draw } i \text{ with prob. } \propto w_t^{[i]} \)
\( x_t \leftarrow x_t \cup \langle x_t^{[i]}, 1/M \rangle \)
7. 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.
The sensor model $p(z_t \mid x_t^{[m]})$ is used to compute the new importance weights:

$$w_t^{[m]} \leftarrow p(z_t \mid x_t^{[m]})$$
After resampling and applying the motion model \( p(x_t \mid u_t, x_{t-1}^{[m]}) \) the particles are distributed more densely at three locations.
Again, we set the new importance weights equal to the sensor model.

\[ w_t^{[m]} \leftarrow p(z_t \mid x_t^{[m]}) \]
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 $\text{Particle}\_\text{filter} (x_t, u_t, z_t)$:

1. $\tilde{x}_t = x_t = \emptyset$
2. for $m = 1$ to $M$ do
3. sample $x_t^{[m]} \sim p(x_t | u_t, x_t^{[m]})$
4. $w_t^{[m]} \leftarrow p(z_t | x_t^{[m]})$
5. $\bar{x}_t \leftarrow \bar{x}_t + (x_t^{[m]}, w_t^{[m]})$
6. for $\bar{x}_t$ do
7. $\text{return } x_t$

Sample from proposal
Compute sample weights
Resampling
Sampling from Proposal

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:

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

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

Computation of the sample weights:

- Proposal distribution: \( g(x_t^{[m]}) = p(x_t^{[m]} | u_t, x_{t-1}^{[m]})\text{Bel}(x_{t-1}^{[m]}) \) (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:

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

- How can we obtain the sensor model $p(z_t \mid x_t^{[m]})$?
- Sensor Calibration:

![Graphs showing probability distribution for Laser and Sonar sensors.](image-url)

Laser sensor

Sonar sensor
Resampling

- Given: Set $\mathcal{X}_t$ 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.

\begin{align*}
\text{for } m = 1 \text{ to } M \\
\text{draw } i \text{ with prob. } \propto w_t^{[i]} \\
\mathcal{X}_t \leftarrow \mathcal{X}_t \cup x_t^{[i]}
\end{align*}
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
Sample-based Localization (sonar)
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 and reduces the
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