UPPSALA
UNIVERSITET

Sequential Monte Carlo and deep regression

Thomas Schön
Uppsala University
Sweden

Department of Engineering, University of Cambridge, Cambridge, UK
January 30, 2020.

Application – indoor localization using the magnetic field (I/II)

Aim: Compute the **position** using variations in the ambient magnetic field and the motion of the person (acceleration and angular velocities). All of this observed using sensors in a standard smartphone.

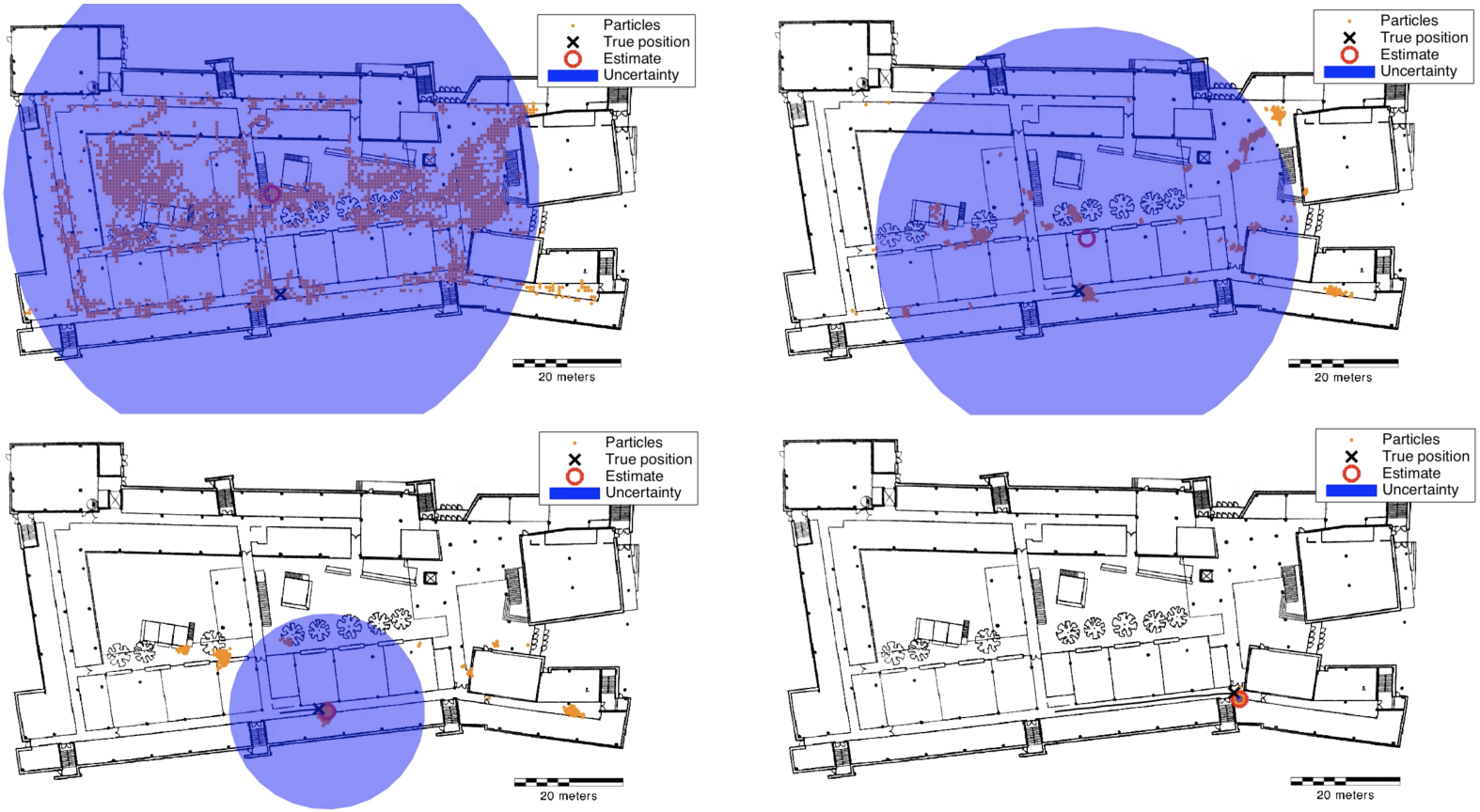
First we need a map, which we build using a tailored Gaussian process.

www.youtube.com/watch?v=enlMiUqPVJo

Arno Solin, Manon Kok, Niklas Wahlström, TS and Simo Särkkä. **Modeling and interpolation of the ambient magnetic field by Gaussian processes**. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018.

Carl Jidling, Niklas Wahlström, Adrian Wills and TS. **Linearly constrained Gaussian processes**. *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December, 2017.

Application – indoor localization using the magnetic field (II/II)



Show movie!

Arno Solin, Simo Särkkä, Juho Kannala and Esa Rahtu. **Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning.** In *Proceedings of the European Navigation Conference*, Helsinki, Finland, June, 2016.

Aim and outline

Aim: To provide intuition for the **key mechanisms** underlying sequential Monte Carlo (SMC), **hint at** a few ways in which SMC fits into the machine learning toolbox and show a new approach to deep regression.

Outline:

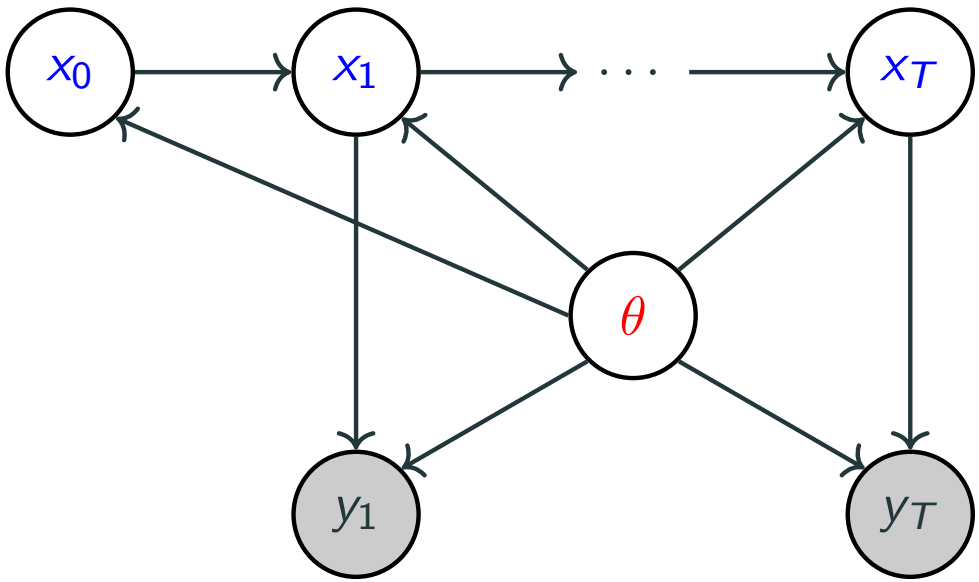
1. Introductory example
2. **SMC for dynamical systems**
3. SMC is a general method
4. Deep probabilistic regression

Representing a nonlinear dynamical systems

The state space model is a **Markov** chain that makes use of a **latent** variable representation to describe dynamical phenomena.

Consists of the unobserved (state) process $\{x_t\}_{t\geq 0}$ modelling the dynamics and the observed process $\{y_t\}_{t\geq 1}$ modelling the relationship between the measurements and the unobserved state process:

$$x_t = f(x_{t-1}, \theta) + v_t,$$
$$y_t = g(x_t, \theta) + e_t.$$



Representations using distributions and programmatic models

Representation using probability distributions

$$x_t \mid (x_{t-1}, \theta) \sim p(x_t \mid x_{t-1}, \theta),$$
$$y_t \mid (x_t, \theta) \sim p(y_t \mid x_t, \theta),$$
$$x_0 \sim p(x_0 \mid \theta).$$

Representation using a **programmatic model**

x[1] ∼ Gaussian(0.0, 1.0);	$p(x_1)$
y[1] ∼ Gaussian(x[1], 1.0);	$p(y_1 \mid x_1)$
for (t in 2..T) {	
x[t] ∼ Gaussian(a*x[t - 1], 1.0);	$p(x_t \mid x_{t-1})$
y[t] ∼ Gaussian(x[t], 1.0);	$p(y_t \mid x_t)$
}	

A **probabilistic program** encodes a **probabilistic model** using a particular probabilistic programming language (here Birch).

Lawrence Murray and TS. **Automated learning with a probabilistic programming language: Birch**. *Annual Reviews in Control*, 46:29–43, 2018.

State space model – full probabilistic model

The **full probabilistic model** is given by

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{\prod_{t=1}^{T} \underbrace{p(y_t \mid x_t, \theta)}_{\text{observation}}}_{\text{likelihood } p(y_{1:T} \mid x_{0:T}, \theta)} \underbrace{\prod_{t=1}^{T} \underbrace{p(x_t \mid x_{t-1}, \theta)}_{\text{dynamics}} \underbrace{p(x_0 \mid \theta)}_{\text{state}} \underbrace{p(\theta)}_{\text{param.}}}_{\text{prior } p(x_{0:T}, \theta)}$$

The **nonlinear filtering problem** involves the measurement update

$$p(x_t \mid y_{1:t}) = \frac{\overbrace{p(y_t \mid x_t)}^{\text{measurement}} \overbrace{p(x_t \mid y_{1:t-1})}^{\text{prediction pdf}}}{p(y_t \mid y_{1:t-1})},$$

and the time update

$$p(x_t \mid y_{1:t-1}) = \int \underbrace{p(x_t \mid x_{t-1})}_{\text{dynamics}} \underbrace{p(x_{t-1} \mid y_{1:t-1})}_{\text{filtering pdf}} \mathrm{d}x_{t-1}.$$

Sequential Monte Carlo (SMC)

The need for approximate methods (such as SMC) is tightly coupled to the intractability of the integrals above.

SMC provide approximate solutions to **integration** problems where there is a **sequential structure** present.

The **particle filter** approximates $p(x_t \mid y_{1:t})$ for

$$x_t = f(x_{t-1}) + v_t,$$
$$y_t = g(x_t) + e_t,$$

by maintaining an **empirical distribution** made up of N samples (particles) $\{x_t^i\}_{i=1}^N$ and the corresponding weights $\{w_t^i\}_{i=1}^N$

$$\underbrace{\widehat{p}(x_t \mid y_{1:t})}_{\widehat{\pi}(x_t)} = \sum_{i=1}^{N} \frac{w_t^i}{\sum_{l=1}^{N} w_t^l} \delta_{x_t^i}(x_t).$$

SMC – in words

1. **Propagation:** Sample a new successor state and append it to the earlier.
2. **Weighting:** The weights corrects for the discrepancy between the proposal distribution and the target distribution.
3. **Resampling:** Focus the computation on the promising parts of the state space by randomly pruning particles, while still preserving the asymptotic guarantees of importance sampling.

Sequential Monte Carlo (SMC) – abstract

The distribution of interest $\pi(x)$ is called the **target distribution**.

(Abstract) problem formulation: **Sample from a sequence** of probability distributions $\{\pi_t(x_{0:t})\}_{t\geq 1}$ defined on a sequence of spaces of increasing dimension, where

$$\pi_t(x_{0:t}) = \frac{\widetilde{\pi}_t(x_{0:t})}{Z_t},$$

such that $\widetilde{\pi}_t(x_t) : \mathcal{X}^t \to \mathbb{R}^+$ is known point-wise and $Z_t = \int \pi(x_{0:t})\mathrm{d}x_{0:t}$ is often computationally challenging.

SMC methods are a class of sampling-based algorithms capable of:

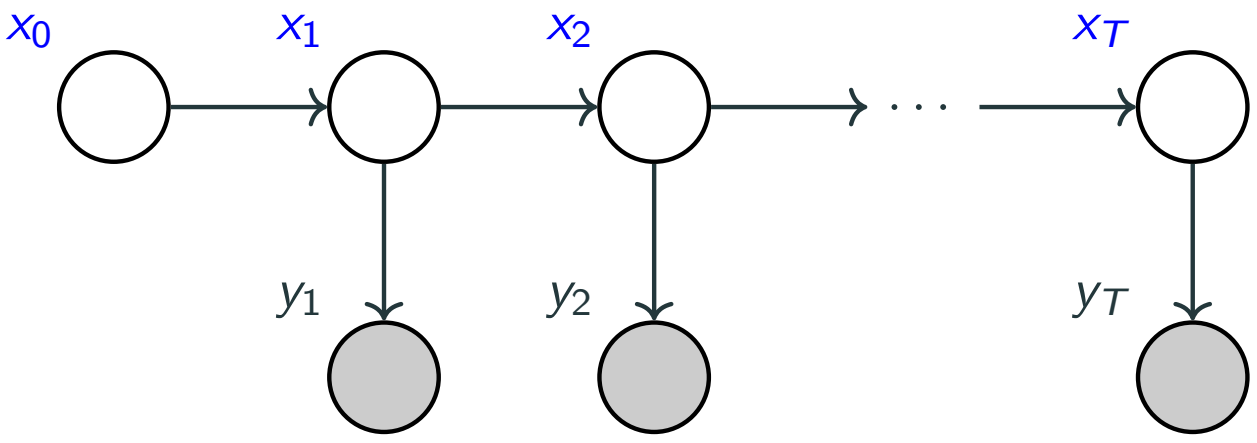
1. Approximating $\pi(x)$ and compute integrals $\int \varphi(x)\pi(x)\mathrm{d}x$.
2. Approximating the normalizing constant Z (unbiased).

Important question: How general is this formulation?

SMC is actually more general than we first thought

The sequence of target distributions $\{\pi_t(x_{1:t})\}_{t=1}^n$ can be constructed in **many** different ways.

The most basic construction arises from **chain-structured graphs**, such as the state space model.



$$\overbrace{p(x_{1:t} \mid y_{1:t})}^{\pi_t(x_{1:t})} = \frac{\overbrace{p(x_{1:t}, y_{1:t})}^{\widetilde{\pi}_t(x_{1:t})}}{\underbrace{p(y_{1:t})}_{Z_t}}$$

SMC can be used for general graphical models

SMC methods are used to approximate a **sequence of probability distributions** on a sequence of spaces of increasing dimension.

Key idea:

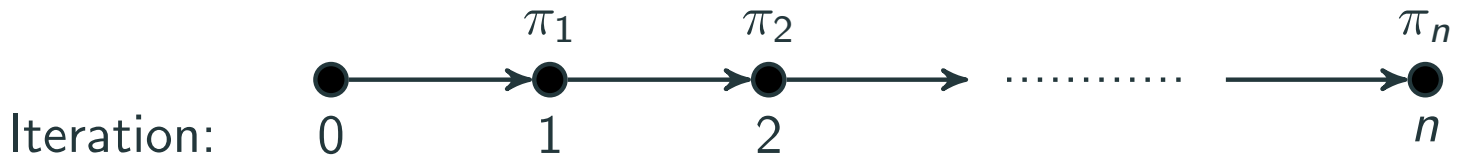
1. Introduce a **sequential decomposition** of any probabilistic graphical model.
2. Each **subgraph** induces an intermediate target dist.
3. Apply SMC to the sequence of intermediate target dist.

SMC also provides an unbiased estimate of the **normalization constant**!

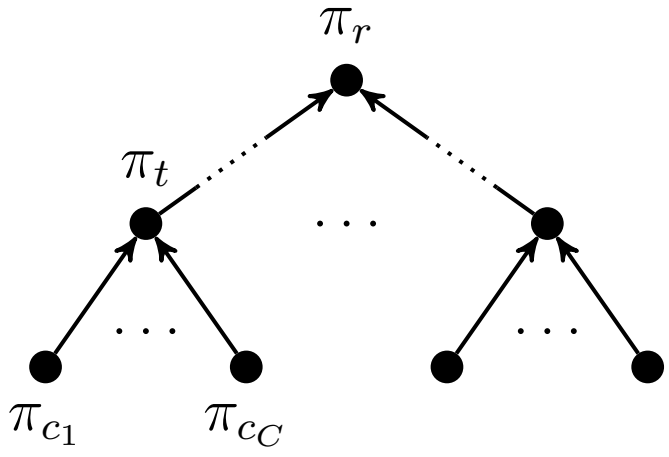
Christian A. Naesseth, Fredrik Lindsten and TS. **Sequential Monte Carlo methods for graphical models**. In *Advances in Neural Information Processing Systems (NIPS)* 27, Montreal, Canada, December, 2014.

Going from classical SMC fo D&C-SMC

The **computational graph** of classic SMC is a sequence (chain)



D&C-SMC generalize the classical SMC framework **from sequences to trees**.

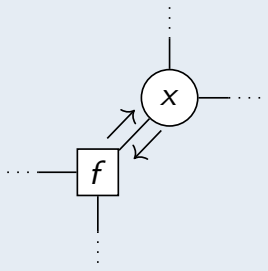


Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Bonnie Kirkpatrick, TS, John Aston and Alexandre Bouchard-Côté. **Divide-and-Conquer with Sequential Monte Carlo**. *Journal of Computational and Graphical Statistics (JCGS)*, 26(2):445-458, 2017.

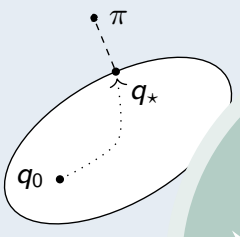
Approximate Bayesian inference – blending

Deterministic methods

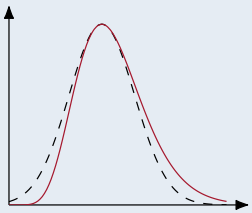
Message passing



Variational inf.

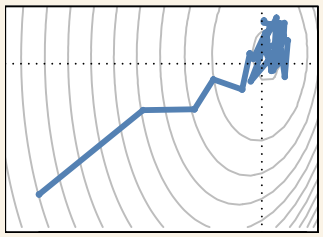


Laplace's method

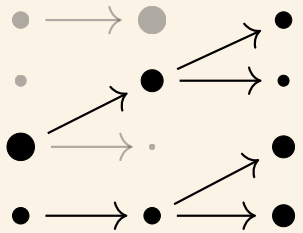


Monte Carlo methods

Markov chain Monte Carlo



Sequential Monte Carlo



VSMC
VMCMC
...

Blending deterministic and Monte Carlo methods

Deterministic methods:

- Good: Accurate and rapid inference
- Bad: Results in biases that are hard to quantify

Monte Carlo methods:

- Good: Asymptotic consistency, lots of theory available
- Bad: Can suffer from a high computational cost

Examples of freedom in the SMC algorithm that opens up for **blending**:

- The **proposal** distributions can be defined in many ways.
- The **intermediate target** distributions can be defined in many ways.

Leads to very interesting and useful algorithms, **many of them still remain to be discovered and explored**.

Deep probabilistic regression

Background: regression using deep neural networks

Supervised regression: learn to predict a continuous output (target) value $y^\star \in \mathcal{Y} = \mathbb{R}^K$ from a corresponding input $x^\star \in \mathcal{X}$, given a training set $\mathcal{D}$ of i.i.d. input-output data

$$\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N, \qquad (x_n, y_n) \sim p(x, y).$$

Deep neural network (DNN): a function $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterized by $\theta \in \mathbb{R}^P$, that maps an input $x \in \mathcal{X}$ to an output $f_\theta(x) \in \mathcal{Y}$.

Our ongoing work on deep regression

Deep learning for classification is handled using standard losses and output representations.

This is **not** the case when it comes to regression.

Train a model $p(y \mid x; \theta)$ of the conditional target density using a DNN to predict the un-normalized density **directly** from input-output pair (x, y).

Four existing approaches: 1. Direct regression

Train a DNN $f_\theta : \mathcal{X} \to \mathcal{Y}$ to directly predict the target $y^\star = f_\theta(x^\star)$.

Learn the parameters θ by minimizing a loss function $\ell(f_\theta(x_n), y_n)$, penalizing discrepancy between prediction $f_\theta(x_n)$ and ground truth y_n

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \ell(f_\theta(x_n), y_n), \qquad \theta = \arg\min_{\theta'} J(\theta').$$

Common choices for ℓ are the L^2 loss, $\ell(\hat{y}, y) = \|\hat{y} - y\|_2^2$, and the L^1 loss.

Minimizing $J(\theta)$ then corresponds to minimizing the negative log-likelihood $\sum_{n=1}^{N} -\log p(y_n \mid x_n; \theta)$, **for a specific model** $p(y \mid x; \theta)$ of the conditional target density.

Ex: The L^2 loss corresponds to a fixed-variance Gaussian model:

$$p(y \mid x; \theta) = \mathcal{N}(y; f_\theta(x), \sigma^2).$$

Four existing approaches: 2. Probabilistic regression

Why not explicitly employ this probabilistic perspective and try to create more **flexible** models $p(y \mid x; \theta)$ of the conditional target density $p(y \mid x)$?

Probabilistic regression: train a DNN $f_\theta : \mathcal{X} \to \mathcal{Y}$ to predict the parameters ϕ of a certain family of probability distributions $p(y; \phi)$, then model $p(y \mid x)$ with

$$p(y \mid x; \theta) = p(y; \phi(x)), \qquad \phi(x) = f_\theta(x).$$

The parameters θ are learned by minimizing $\sum_{n=1}^{N} -\log p(y_n \mid x_n; \theta)$.

Ex: A general 1D Gaussian model can be realized as:

$$p(y \mid x; \theta) = \mathcal{N}\big(y; \mu_\theta(x), \sigma^2_\theta(x)\big), \qquad f_\theta(x) = \begin{pmatrix} \mu_\theta(x) & \log \sigma^2_\theta(x) \end{pmatrix}^\mathsf{T} \in \mathbb{R}^2.$$

Four existing approaches: 3. Confidence-based regression

The quest for improved regression accuracy has also led to the development of more specialized methods.

Confidence-based regression: train a DNN $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to predict a scalar confidence value $f_\theta(x, y)$, and maximize this quantity over y to predict the target

$$y^\star = \arg\max_y f_\theta(x^\star, y)$$

The parameters θ are learned by generating **pseudo** ground truth confidence values $c(x_n, y_n, y)$, and minimizing a loss function $\ell\big(f_\theta(x_n, y), c(x_n, y_n, y)\big)$.

Four existing approaches: 4. Regression-by-classification

Discretize the output space $\mathcal{Y}$ into a finite set of C classes and use standard classification techniques...

High-level description of our idea

Confidence-based regression give impressive results, but:

1. they require important (and tricky) task-dependent design choices (e.g. how to generate the pseudo ground truth labels)
2. and usually lack a clear probabilistic interpretation.

Probabilistic regression is straightforward and generally applicable, but:

1. it can usually not compete in terms of regression accuracy.

Our construction **combines the benefits** of these two approaches while **removing the problems** above.

Our (simple and very general) construction

A general regression method with a **clear probabilistic interpretation** in the sense that we learn a model $p(y \mid x, \theta)$ **without** requiring $p(y \mid x, \theta)$ to belong to a particular family of distributions.

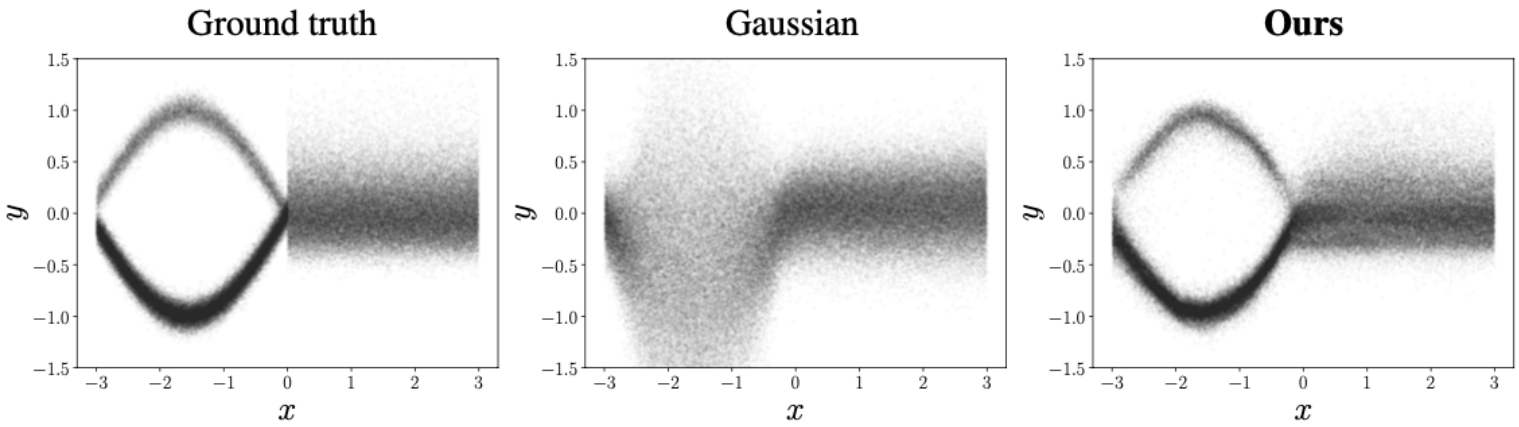
Let the DNN be a function $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that maps an input-output pair $\{x_n, y_n\}$ to a scalar value $f_\theta(x_n, y_n) \in \mathbb{R}$.

Define the resulting (flexible) probabilistic model as

$$p(y \mid x, \theta) = \frac{e^{f_\theta(x,y)}}{Z(x,\theta)}, \qquad Z(x,\theta) = \int e^{f_\theta(x,y)} dy$$

Learning flexible deep conditional target densities

1D toy illustration showing that we can learn multi-modal and asymmetric distributions, i.e. our model is **flexible**.



We train by maximizing the log-likelihood:

$$\max_{\theta} \sum_{n=1}^{N} \log p(y_n \mid x_n, \theta) = \max_{\theta} \sum_{n=1}^{N} -\log \underbrace{\left(\int e^{f_\theta(x_n, y)} dy \right)}_{Z(x_n, \theta)} + f_\theta(x_n, y_n)$$

Challenge: Requires the normalization constant to be evaluated...

Solution: Monte Carlo! (via a simple importance sampling construction)

Training the model

$$p(y \mid x, \theta) = \frac{e^{f_\theta(x,y)}}{Z(x,\theta)}, \qquad Z(x,\theta) = \int e^{f_\theta(x,y)} dy$$

The parameters θ are learned by minimizing $\sum_{n=1}^{N} -\log p(y_n \mid x_n; \theta)$.

Use importance sampling to evaluate $Z(x, \theta)$:

$$\begin{aligned} -\log p(y_n \mid x_n; \theta) &= \log \left(\int e^{f_\theta(x_n,y)} dy \right) - f_\theta(x_n, y_n) \\ &= \log \left(\int \frac{e^{f_\theta(x_n,y)}}{q(y)} q(y) dy \right) - f_\theta(x_n, y_n) \\ &\approx \log \left(\frac{1}{M} \sum_{k=1}^{M} \frac{e^{f_\theta(x_n,y^{(k)})}}{q(y^{(k)})} \right) - f_\theta(x_n, y_n), \quad y^{(k)} \sim q(y). \end{aligned}$$

Use a Gaussian mixture as proposal.

Prediction at test time

Train a DNN $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ to predict $f_\theta(x, y)$ and model $p(y \mid x)$ with

$$p(y \mid x, \theta) = \frac{e^{f_\theta(x,y)}}{Z(x, \theta)}, \qquad Z(x, \theta) = \int e^{f_\theta(x,y)} dy.$$

The parameters θ are learned by minimizing $\sum_{n=1}^{N} -\log p(y_n \mid x_n; \theta)$.

Given a test input $x^\star$, we predict the target $y^\star$ by maximizing $p(y \mid x^\star; \theta)$

$$y^\star = \arg\max_y \, p(y \mid x^\star; \theta) = \arg\max_y \, f_\theta(x^\star, y).$$

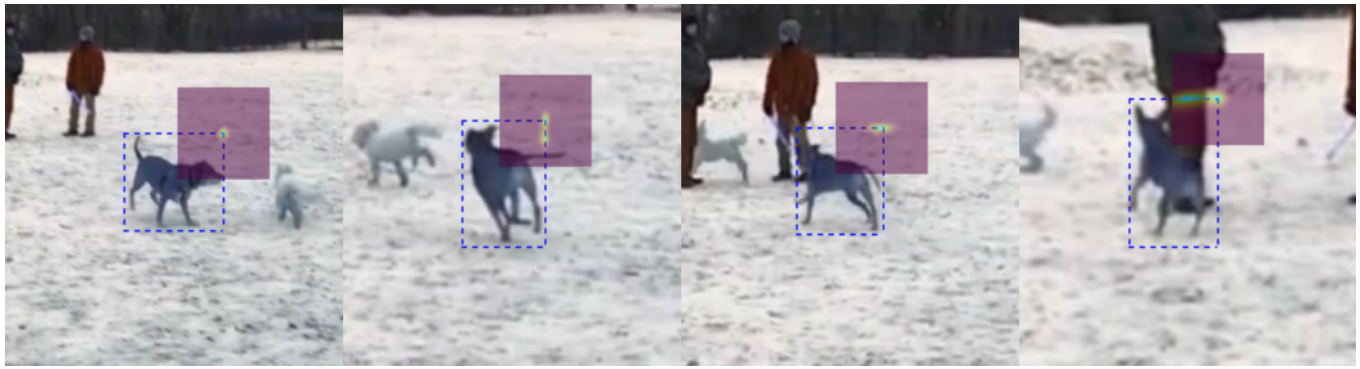
By designing the DNN f_θ to be differentiable w.r.t. targets y, the gradient $\nabla_y f_\theta(x^\star, y)$ can be efficiently evaluated using auto-differentiation.

Use gradient ascent to find a local maximum of $f_\theta(x^\star, y)$, starting from an initial estimate $\hat{y}$.

Experiments

Good results on four different computer vision (regression) problems:
1. Object detection, 2. Age estimation, 3. Head-pose estimation and
4. **Visual tracking**.

Task (visual tracking): Estimate a bounding box of a target object in every frame of a video. The target object is defined by a given box in the first video frame.



Show Movie!

Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat and TS. **Learning deep conditional target densities for accurate regression**. Submitted, November, 2019.

Conclusion

SMC provide approximate solutions to **integration** problems where there is a **sequential structure** present.

- SMC is **more general** than we first though.
- SMC can indeed be **computationally challenging**, but it comes with rather well-developed analysis and guarantees.
- There is still a lot of freedom **waiting to be exploited**.
- Constructed a practical deep flexible model for regression

Forthcoming SMC introduction written with an ML audience in mind

Christian A. Naesseth, Fredrik Lindsten, and TS. **Elements of sequential Monte Carlo**. *Foundations and Trends in Machine Learning*, 2019.

Backup slides

Recent developments working with the trend of blending

Develop new approximating families of distributions.

Naesseth, C. A., Linderman, S. W., Ranganath, R. and Blei, D. M. **Variational Sequential Monte Carlo**. *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A. and Teh, Y. W. **Filtering variational objectives**. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Le, T. A., Igl, M., Rainforth, T., Jin, T. and Wood, F. **Auto-encoding sequential Monte Carlo**. In *International Conference on Learning Representations (ICLR)*, 2018.

Alter the intermediate targets to take "future variables" into account.

Results in **"additional intractability"** – use deterministic methods.

Alternative interpretation: Use SMC as a post-correction for the bias introduced by the deterministic methods.

Lindsten, F., Helske, J. and Vihola, M. **Graphical model inference: Sequential Monte Carlo meets deterministic approximations**. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

"The combination of the two ideas mentioned above".

Lawson, D., Tucker, G., Naesseth, C. A., Maddison, C. J., Adams, R. P., and Teh, Y. W. **Twisted Variational Sequential Monte Carlo**. *Bayesian Deep Learning (NeurIPS Workshop)*, 2018.