Machine learning - trends and tools

“introducing the field and some of its key concepts”

Thomas Schöen
Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: thomas.schon@it.uu.se,
www: user.it.uu.se/~thosc112

“Machine learning gives computers the ability to learn without being explicitly programmed for the task at hand.”
“Anyone making confident predictions about anything having to do with the future of artificial intelligence is either kidding you or kidding themselves.”

Andrew McAfee, MIT
Background – what we do in the team

We automate the extraction of knowledge and understanding from data.

Both basic research and applied research (with companies).

Create probabilistic models for dynamical systems and their surroundings.

Develop methods to learn models from data.

The models can then be used by machines (or humans) to understand or take decisions about what will happen next.
What do I hope to achieve today?

1. Briefly introduce the scientific field of Machine Learning.
2. Create an awareness/interest around this technology.
3. Course style introduction to the Gaussian process.
What is machine learning all about?

Machine learning is about learning, reasoning and acting based on data.

Machine learning gives computers the ability to learn without being explicitly programmed for the task at hand.

“It is one of today’s most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science.”


A probabilistic approach

Machine learning is about methods allowing computers/machines automatically make use of data to solve tasks.

Data on its own is typically useless, it is only when we can extract knowledge from the data that it becomes useful.

Representation of the data: A model with unknown (a.k.a. latent or missing) variables related to the knowledge we are looking for.

Key concept: Uncertainty.

Key ingredient: Data.

Probability theory and statistics provide the theory and practice that is needed for representing and manipulating uncertainty about data, models and predictions.

Learn the unknown variables from the data.
The three cornerstones

Cornerstone 1 (Data) Typically we need lots of it.

Cornerstone 2 (Mathematical model) A mathematical model is a compact representation of the data that in precise mathematical form captures the key properties of the underlying situation.

Cornerstone 3 (Learning algorithm) Used to compute the unknown variables from the observed data using the model.
The performance of an algorithms typically depends on which representation that is used for the data.

Learned representations often provide better solutions than hand-designed representations.

When solving a problem – start by thinking about which model/representation to use!
Problem: How can we learn good representations of data?

Ex. Deep learning (DL) solves the problem by introducing representations that are expressed in terms of other, simpler representations.

International Conference on Learning Representations

http://www.iclr.cc/
The two basic rules from probability theory

Let $x$ and $y$ be continuous random variables. Let $p(\cdot)$ denote a general probability density function.

1. Marginalization (integrate out a variable):

$$p(x) = \int p(x, y) \, dy.$$

2. Conditional probability:

$$p(x, y) = p(x \mid y)p(y).$$

Combine them into Bayes’ rule:

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)} = \frac{p(x \mid y)p(y)}{\int p(x \mid y)p(y) \, dy}.$$
Key objects in learning a model

$D$ - measured data.
$z$ - unknown model variables.

The **full probabilistic model** is given by

$$ p(D, z) = \underbrace{p(D | z)}_{\text{data distribution}} \underbrace{p(z)}_{\text{prior}} $$

Inference amounts to computing the **posterior distribution**

$$ p(z | D) = \frac{\underbrace{p(D | z)}_{\text{data distribution}} \underbrace{p(z)}_{\text{prior}}}{p(D)} $$

Let’s make this more concrete.
Ex) Probabilistic modeling of linear SSMs

A linear Gaussian state space model (SSM) consists of a Markov process \( \{x_t\}_{t \geq 1} \) that is indirectly observed via a linear, Gaussian measurement process \( \{y_t\}_{t \geq 1} \),

\[
x_{t+1} = Ax_t + Bu_t + v_t, \quad v_t \sim \mathcal{N}(0, Q)
\]
\[
y_t = Cx_t + Du_t + e_t, \quad e_t \sim \mathcal{N}(0, R)
\]
\[
x_1 \sim \mu_\eta(x_1),
\]
\[
\theta \sim \pi(\theta),
\]

where \( \theta = \{A, B, C, D, Q, R, \eta\} \).

The **full probabilistic model** is given by

\[
p(x_{1:T}, \theta, y_{1:T}) = p(y_{1:T} | x_{1:T}, \theta) p(x_{1:T}, \theta)
\]

where \( p(x_{1:T}, \theta, y_{1:T}) \) is the **data distribution** and \( p(x_{1:T}, \theta) \) is the **prior**.
The Earth’s magnetic field sets a background for the ambient magnetic field. Deviations make the field vary from point to point.

**Aim:** Build a map (i.e., a model) of the magnetic environment based on measurements from magnetometers.

**Solution:** Customized Gaussian process that obeys Maxwell’s equations.

[www.youtube.com/watch?v=en1MiUqPVJo](http://www.youtube.com/watch?v=en1MiUqPVJo)

The model – learning relationship

The problem of learning (estimating) a model based on data leads to computational challenges, both

- **Integration**: e.g. the HD integrals arising during marg. (averaging over all possible parameter values $\mathbf{z}$):

  $$p(D) = \int p(D | \mathbf{z}) p(\mathbf{z}) \, d\mathbf{z}.$$  

- **Optimization**: e.g. when extracting point estimates, for example by maximizing the posterior or the likelihood

  $$\hat{\mathbf{z}} = \arg \max_\mathbf{z} p(D | \mathbf{z})$$

Typically impossible to compute exactly, use approximate methods

- Monte Carlo (MC), Markov chain MC (MCMC), and sequential MC (SMC).
- Variational inference (VI).
Use flexible models

Key lesson from modern Machine Learning:

Flexible models often gives the best performance.

How can we build flexible models?

1. Models that use a large (but fixed) number of parameters compared with the data set. (parametric, ex. deep learning)

2. Models that use more parameters as we get access to more data. (non-parametric, ex. Gaussian process)
Outline

1. What is machine learning?
2. A closer look at the models
3. Two concrete examples of flexible models
   a) Deep learning
   b) Gaussian processes
(4. A few other trends/tools)
5. Conclusion

Machine learning gives computers the ability to learn without being explicitly programmed for the task at hand.
Deep learning – an example

Automatically learn how to describe the contents of images.

A woman is throwing a frisbee in a park.

A large white bird standing in a forest.

A little girl sitting on a bed with a teddy bear.

A person is standing on a beach with a surfboard.

A few successful examples. A few examples where it failed...

Deep learning – another example

A machine defeated a human professional for the first time in the game of Go.

Interesting company: DeepMind

Deep learning – what is it?

The mathematical model has been around for 70 years, but over the last 5 years there has been a revolution. Key reasons:

1. Very large datasets
2. Better and faster computers
3. Enormous industrial interest (e.g. Google, Facebook, MS)
4. Some methodological breakthroughs

The underlying model is a big mathematical function with multiple layers of abstraction, commonly with millions of parameters.

The parameter values are automatically determined based on a large amount of training data.
Why deep? - Image classification example

Image classification (Input: pixels of an image. Output: object identity.)

1 megapixel (B/W) → $2^{1000000}$ possible images!

A deep neural network can solve this with a few million parameters!

Each hidden layer extracts increasingly abstract features.

A neural network (NN) is a hierarchical nonlinear function \( y = g_\theta(x) \) from an input variable \( x \) to an output variable \( y \) parameterized by \( \theta \).

Linear regression models the relationship between a continuous output variable \( y \) and an input variable \( x \),

\[
y = \sum_{i=1}^{n} x_i \theta_i + \theta_0 + \varepsilon = x^T \theta + \varepsilon,
\]

where \( \theta \) is the parameters composed by the “weights” \( \theta_i \) and the offset ("bias") term \( \theta_0 \),

\[
\theta = (\theta_0 \ \theta_1 \ \theta_2 \ \cdots \ \theta_n)^T, \\
x = (1 \ x_1 \ x_2 \ \cdots \ x_n)^T.
\]
Generalized linear regression and NNs

We can generalize this by introducing nonlinear transformations of the predictor \( u^T \theta \),

\[
y = f(u^T \theta).
\]

We can think of the neural network as a sequential construction of several generalized linear regressions.

Each layer in a multi-layer NN is modelled as

\[
z^{(l+1)} = f \left( \Theta^{(l+1)} z^{(l)} + \theta_0^{(l+1)} \right),
\]

starting with the input \( z^{(0)} = x \). (The nonlinearity operates element-wise.)
Deep neural networks

Let the computer **learn from experience** and understand the situation in terms of a **hierarchy of concepts**, where each concepts is defined in terms of its relation to simpler concepts.

If we draw a graph showing these concepts of top of each other, the graph is **deep**, hence the name deep learning.

It is accomplished by using **multiple levels of representation**. Each level transforms the representation at the previous level into a new and more abstract representation,

\[ z^{(l+1)} = f \left( \Theta^{(l+1)} z^{(l)} + \theta_0^{(l+1)} \right), \]

starting from the input (raw data) \( z^{(0)} = x \).

**Key aspect:** The layers are not designed by human engineers, they are generated from (typically lots of) data using a learning procedure and lots of computations.
Deep learning – example (skin cancer)

Start from a mathematical model trained on 1.28 million images (transfer learning). Make minor modifications of it, specializing to present situation.

Learn new model parameters using 129,450 clinical images (∼100 times more images than any previous study).

The results are on par with professional dermatologists on specific tasks. Still, far from being clinically useful, but at least they give us “valid reasons to remain cautiously optimistic” as someone said.

Deep learning – Want to know more?

Good introduction

Timely introduction (target audience include: software engineers who do not have a machine learning or statistics background)
http://www.deeplearningbook.org

Interesting discussion about why it works so well

Deep learning summer school 2016
https://sites.google.com/site/deeplearningsummerschool2016/

Geoffrey Hinton’s Coursera course
https://www.coursera.org/learn/neural-networks/home/welcome

NIPS and ICML conferences and workshops!
The Gaussian process (GP) is a non-parametric and probabilistic model for nonlinear functions.

- **Non-parametric** means that it does not rely on any particular parametric functional form to be postulated.
- **Probabilistic** means that it takes uncertainty into account in every aspect of the model.

An abstract idea

In probabilistic linear regression

\[ y_i = \beta^T x_i + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \]

we place a prior on \( \beta \), \( \beta \sim \mathcal{N}(0, \sigma^2 I_p) \).

(Abstract) idea: What if we instead place a prior directly on the function \( f(\cdot) \)

\[ f \sim p(f) \]

and look for \( p(f | y) \) rather than \( p(\beta | y) \)?!
What does it actually mean to have a prior over functions?

Can we construct a probabilistic object operating on functions?
One concrete construction

Well, one (arguably simple) idea on how we can reason probabilistically about an unknown function $f$ is by assuming that $f(x)$ and $f(x')$ are jointly Gaussian distributed:

$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} \sim \mathcal{N}(\mu, K)$

If we accept the above idea we can without conceptual problems generalize to any arbitrary set of input values $\{x_1, x_2, \ldots, x_N\}$. 
Definition and its implications

**Definition: (Gaussian Process, GP)** A GP is a (potentially infinite) collection of random variables such that any finite subset of it is jointly distributed according to a Gaussian.

Our definition means that for any arbitrary set of input values \( \{x_1, x_2, \ldots, x_N\} \) we have

\[
\begin{pmatrix}
    f(x_1) \\
    \vdots \\
    f(x_N)
\end{pmatrix}
\sim \mathcal{N}
\begin{pmatrix}
    \begin{pmatrix}
        m(x_1) \\
        \vdots \\
        m(x_N)
    \end{pmatrix},
    \begin{pmatrix}
        k(x_1, x_1) & \cdots & k(x_1, x_N) \\
        \vdots & \ddots & \vdots \\
        k(x_N, x_1) & \cdots & k(x_N, x_N)
    \end{pmatrix}
\end{pmatrix}
\]
We now have a prior!

\[ f \sim \mathcal{GP}(m, k) \]

The GP is a **generative** model so let us first sample from the prior.
GP regression

**Remaining problem:** Given training data $\mathcal{T} = \{x_i, y_i\}_{i=1}^N$ and our GP prior $f \sim \mathcal{GP}(m, k)$ compute $p(f_\star | y)$ for an arbitrary test point $(x_\star, y_\star)$.

\[
\begin{pmatrix} y \\ f_\star \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} m(x) \\ m(x_\star) \end{pmatrix}, \begin{pmatrix} k(x, x) + \sigma^2 I_N & k(x, x_\star) \\ k(x_\star, x) & k(x_\star, x_\star) \end{pmatrix} \right),
\]

The conditioning theorem for partitioned Gaussians results in

\[
f_\star | y \sim \mathcal{N}(\mu_\star, k_\star),
\]

\[
\mu_\star = m(x_\star) + s^T (y - m(x)),
\]

\[
k_\star = k(x_\star, x_\star) - s^T k(x, x_\star),
\]

where

\[
s^T = k(x_\star, x)(k(x, x) + \sigma^2 I_N)^{-1}.
\]
GP regression – illustration
Gaussian process state space model

“Inspired by the Gaussian process, enabled by the particle filter”

\[ x_{t+1} = f(x_t) + v_t, \quad \text{s.t.} \quad f(x) \sim \mathcal{GP}(0, \kappa \eta, f(x, x')) , \]
\[ y_t = g(x_t) + e_t, \quad \text{s.t.} \quad g(x) \sim \mathcal{GP}(0, \kappa \eta, g(x, x')) . \]

The model functions \( f \) and \( g \) are assumed to be realizations from Gaussian process priors and \( v_t \sim \mathcal{N}(0, Q) \), \( e_t \sim \mathcal{N}(0, R) \).

We can now find the posterior distribution

\[ p(f, g, Q, R, \eta \mid y_{1:T}) , \]

via some approximation (we use particle MCMC).


Regularization in nonlinear state spaces

Results in a flexible non-parametric model where the GP prior on \( f \) takes on the role of a regularizer.

Provides a data-driven way of tuning the model flexibility.

Toy example:

\[
x_{t+1} = -10 \frac{x_t}{1 + 3x_t^2} + v_t, \\
y_t = x_t + e_t.
\]

For all details on the new model and more examples, see

Gaussian Process – Want to know more?

Classic monograph on Gaussian processes


A review of Bayesian non-parametric modelling


Note from our new MSc level course

www.it.uu.se/edu/course/homepage/sml/literature/probabilistic_modeling_compendium.pdf

The standard GP is too expensive ($O(N^3)$) to be practically useful, but there are good approximations available, see e.g.

Machine learning gives computers the ability to learn without being explicitly programmed for the task at hand.
Probabilistic numerics

Reinterpreting numerical tasks (e.g., linear algebra, int., opt. and solving differential equations) as probabilistic inference problems.

A numerical method estimates a certain latent property given the result of computations, i.e. computation is inference.

Ex: Basic alg. that are equivalent to Gaussian MAP inference
- Conjugate Gradients for linear algebra
- BFGS etc. for nonlinear optimization
- Gaussian Quadrature rules for Integration
- Runge-Kutta solvers for ODEs


https://www.youtube.com/v/tZ9CP-kQAVI&showsearch=0&autoplay=1&rel=0
http://probabilistic-numerics.org
Bayesian optimization

Bayesian optimization is a **sequential model-based** approach to solve the problem of **optimizing an unknown (or expensive to evaluate) function** $f(x)$

$$x^* = \arg \max_x f(x)$$

**Place a prior** over the unknown objective function $f(x)$ (e.g. using a Gaussian process) and sequentially refine it as data becomes available via probabilistic posterior updating.

The posterior distribution is then used to construct an **acquisition function** that determines what the next query point $x$ should be.

---

Bayesian optimization – illustration
Probabilistic programming makes use of computer programs to represent probabilistic models.

Probabilistic programming lies on the interesting intersection of
1. Programming languages: Compilers and semantics.

Creates a clear separation between the model and the inference methods, encouraging model based thinking. Automate inference!

http://probabilistic-programming.org/
Markov chain Monte Carlo (MCMC)

Represent distributions using a large number of samples.

Markov chain Monte Carlo (MCMC) methods are used to sample from a probability distribution by **simulating a Markov chain** that has the desired distribution as its stationary distribution.

Used to compute numerical approximations of intractable integrals.

Constructive algorithms:

1. The **Metropolis Hastings** sampler
2. The **Gibbs** sampler

“Visualizing” a Markov chain

Built a Markov chain to sample light paths connecting the sensor with light sources in the scene.

Results using equal time rendering

Our method that builds on MLT

Metropolis light transport (MLT)

Sequential Monte Carlo (SMC)

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

Important example where we have a sequential structure present is in dynamical systems, where we call SMC particle filtering.

We will host the SMC conference in Uppsala in August 2017!

http://www.it.uu.se/conferences/smc2017


Thomas Schönh"
Variational inference

Variational inference provides an approximation to the posterior distribution by assuming that it has a certain functional form that contain unknown parameters.

These unknown parameters are found using optimization, where some distance measure is minimized.

Variational inference methods are used to approximate intractable integrals and are thus an alternative to MCMC.

Ex. Variational Bayes (VB) and expectation propagation (EP).

A few other trends/tools (brief)

1. **Probabilistic numerics** – Reinterpreting numerical tasks as probabilistic inference problems.

2. **Bayesian optimization** – A sequential model-based approach to solve the problem of optimizing an unknown function $f(x)$.

3. **Probabilistic programming** – Makes use of computer programs to represent probabilistic models.

4. Two strategies to approximate intractable target distributions
   - **Markov chain Monte Carlo (MCMC)** – Construct a Markov chain with the target distribution as its stationary distribution. Then, we sample from the chain to eventually collect independent samples from the stationary distribution.
   - **Variational inference** – Assume a family of distributions and find the member (using optimization) of that family which is closest to the target distribution.
What did I hope to achieve today?

1. Briefly introduce the scientific field of Machine Learning.
2. Create an **awareness/interest** around this technology.
3. Course style introduction to the Gaussian process.
ASSEMBLE – an ML research project

**Aim:** Automate probabilistic modeling of dynamical systems (and their surroundings) via a formally defined **probabilistic modeling language**.

5 year project with a budget of SEK 29 000 000.

Probabilistic Modeling Research

- Application Model
- Inference Methods Research

Modeling Language Research

- Probabilistic Model Compiler

Application Specific Machine Learning Solution

Demonstrators
- Smart Meters (Greenely)
- Cell Tracking (Karolinska Institute)
- Energy-Aware Computing
- Container Crane Automation (ABB)
- Smart Automotive Safety (Autoliv)

Feedback from demonstrators enables:
- Improved modeling techniques
- Improved inference methods
- Enhanced modeling language
New course in Statistical Machine Learning


Introductory course to statistical machine learning (SML) focusing on classification and regression. Topics include:

- Regression, classification and boosting
- Regularization (ridge regression and the LASSO)
- Regression and classification trees
- Deep learning and neural networks

Deep learning lab: See how a state-of-the-art deep learning algorithm performs at classifying real world images.

Mini project: Predict which songs one of the TAs like the most.

www.it.uu.se/edu/course/homepage/sml
Machine learning gives computers the ability to **learn without being explicitly programmed** for the task at hand.

**Uncertainty** is a key concept!

The best predictive performance is currently obtained from **highly flexible** learning systems, e.g.

1. Deep learning
2. Gaussian processes

Remember to talk to people who work on **different problems** with **different tools**!! (Visit other fields!)