The computations of acting agents and the agents acting in computations

Philipp Hennig
ICERM
5 June 2017

Research Group for Probabilistic Numerics
Max Planck Institute for Intelligent Systems
Tübingen, Germany

Some of the presented work was supported by the Emmy Noether Programme of the DFG
Part I: The computations of acting agents 09:00–09:45
- a minimal introduction to machine learning
- the computational tasks of learning agents
- some special challenges, some house numbers

Part II: The agents acting in computations 10:30–11:15
- computation is inference
- new challenges require new answers
- a computer science view on numerical computations
An Acting Agent
autonomous interaction with a data-source

The Very Foundation

probabilistic inference

\[ p(x \mid D) = \frac{p(x)p(D \mid x)}{\int p(x)p(D \mid x) \, dx} \]

- **prior** explicit representation of assumptions about latent variables
- **likelihood** explicit representation of assumptions about generation of data
- **posterior** structured uncertainty over prediction
- **evidence** marginal likelihood of model

\[ \mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]
Gaussian Inference
the link between probabilistic inference and linear algebra

+ products of Gaussians are Gaussians

\[
C := (A^{-1} + B^{-1})^{-1} \quad c := C(A^{-1}a + B^{-1}b)
\]

\[
\mathcal{N}(x; a, A)\mathcal{N}(x; b, B) = \mathcal{N}(x; c, C)\mathcal{N}(a; b, A + B)
\]

+ marginals of Gaussians are Gaussians

\[
\int \mathcal{N} \left( \begin{pmatrix} x \\ y \end{pmatrix}; \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix} \right) dy = \mathcal{N}(x; \mu_x, \Sigma_{xx})
\]

+ (linear) conditionals of Gaussians are Gaussians

\[
p(x \mid y) = \frac{p(x, y)}{p(y)} = \mathcal{N} \left( x; \mu_x + \Sigma_{xy} \Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx} \right)
\]

+ linear projections of Gaussians are Gaussians

\[
p(z) = \mathcal{N}(z; \mu, \Sigma) \quad \Rightarrow \quad p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^T)
\]

Bayesian inference becomes linear algebra

\[
p(x) = \mathcal{N}(x; \mu, \Sigma) \quad p(y \mid x) = \mathcal{N}(y; A^T x + b, \Lambda)
\]

\[
p(B^T x + c \mid y) = \mathcal{N}[B^T x + c; B^T \mu + c + B^T \Sigma A (A^T \Sigma A + \Lambda)^{-1} (y - A^T \mu - b),
\]

\[
B^T \Sigma B - B^T \Sigma A (A^T \Sigma A + \Lambda)^{-1} A^T \Sigma B]
\]
A Minimal Machine Learning Setup

nonlinear regression problem

\[ p(y \mid f_x) = \mathcal{N}(y; f_x, \sigma I) \]
Gaussian Parametric Regression
aka. general linear least-squares

\[ f(x) = \phi(x)^T w = \sum_i w_i \phi_i(x) \quad p(w) = \mathcal{N}(w; \mu, \Sigma) \]

\[ \Rightarrow p(f) = \mathcal{N}(f, \phi^T \mu, \phi^T \Sigma \phi) \quad \phi_i(x) = \mathbb{I}(x > a_i) \cdot c_i(x - a_i) \quad \text{(RELU)} \]
Gaussian Parametric Regression
aka. general linear least-squares

\[ f(x) = \phi(x)^T w = \sum_i w_i \phi_i(x) \quad p(w) = \mathcal{N}(w; \mu, \Sigma) \]

\[ \Rightarrow \quad p(f) = \mathcal{N}(f, \phi^T \mu, \phi^T \Sigma \phi) \quad \phi_i(x) = \mathbb{I}(x > a_i) \cdot c_i(x - a_i) \quad \text{(RELU)} \]
Gaussian Parametric Regression
aka. general linear least-squares

\[ p(y \mid w, \phi_x) = \mathcal{N}(y; \phi_x^T w, \sigma^2 I) \]

\[ p(f_x \mid y, \phi_x) = \mathcal{N}(f_x; \phi_x^T \mu + \phi_x^T \Sigma \phi_x (\phi_x^T \Sigma \phi_x + \sigma^2 I)^{-1} (y - \phi_x^T \mu), \phi_x^T \Sigma \phi_x - \phi_x^T \Sigma \phi_x (\phi_x^T \Sigma \phi_x + \sigma^2 I)^{-1} \phi_x^T \Sigma \phi_x) \]
The Choice of Prior Matters

Bayesian framework provides flexible yet explicit modelling language

$$\phi_i(x) = \theta \exp \left( - \frac{(x - c_i)^2}{2\lambda^2} \right)$$
The Choice of Prior Matters

Bayesian framework provides flexible yet explicit modelling language

\[ \phi_i(x) = \theta \exp \left( -\frac{(x - c_i)^2}{2\lambda^2} \right) \]
popular extension no. 1
requires large-scale linear algebra

\[ p(f_x \mid y, \phi_x) = \mathcal{N}(f_x; \phi_x^T \mu + \phi_x^T \Sigma \phi_x (\phi_x^T \Sigma \phi_x + \sigma^2 I)^{-1} (y - \phi_x^T \mu), \]
\[ \phi_x^T \Sigma \phi_x - \phi_x^T \Sigma \phi_x (\phi_x^T \Sigma \phi_x + \sigma^2 I)^{-1} \phi_x^T \Sigma \phi_x) \]

- set \( \mu = 0 \)
- aim for closed-form expression of kernel \( \phi_a^T \Sigma \phi_b \)
Features are cheap, so let’s use a lot

an example

For simplicity, let’s fix $\sum = \frac{\sigma^2(c_{\text{max}} - c_{\text{min}})}{F} I$

thus: $\phi(x_i)^T \sum \phi(x_j) = \frac{\sigma^2(c_{\text{max}} - c_{\text{min}})}{F} \sum_{\ell=1}^{F} \phi_\ell(x_i) \phi_\ell(x_j)$

especially, for $\phi_\ell(x) = \exp\left(-\frac{(x - c_\ell)^2}{2\lambda^2}\right)$

$\phi(x_i)^T \sum \phi(x_j)$

$= \frac{\sigma^2(c_{\text{max}} - c_{\text{min}})}{F} \sum_{\ell=1}^{F} \exp\left(-\frac{(x_i - c_\ell)^2}{2\lambda^2}\right) \exp\left(-\frac{(x_j - c_\ell)^2}{2\lambda^2}\right)$

$= \frac{\sigma^2(c_{\text{max}} - c_{\text{min}})}{F} \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \sum_{\ell} \exp\left(-\frac{c_\ell - \frac{1}{2}(x_i + x_j)^2}{\lambda^2}\right)$
Features are cheap, so let’s use a lot

an example

\[ \phi(x_i)^T \Sigma \phi(x_j) = \]
\[ \frac{\sigma^2 (c_{\text{max}} - c_{\text{min}})}{F} \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \sum_{\ell} \exp \left( -\frac{(c_{\ell} - \frac{1}{2}(x_i + x_j))^2}{\lambda^2} \right) \]

\[ \frac{F \cdot \delta c}{(c_{\text{max}} - c_{\text{min}})} \]

† now increase \( F \) so # of features in \( \delta c \) approaches \( \frac{F \cdot \delta c}{(c_{\text{max}} - c_{\text{min}})} \)

\[ \phi(x_i)^T \Sigma \phi(x_j) \rightarrow \]
\[ \sigma^2 \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \int_{c_{\text{min}}}^{c_{\text{max}}} \exp \left( -\frac{(c - \frac{1}{2}(x_i + x_j))^2}{\lambda^2} \right) \, dc \]

† let \( c_{\text{min}} \rightarrow -\infty, c_{\text{max}} \rightarrow \infty \)

\[ k(x_i, x_j) := \phi(x_i)^T \Sigma \phi(x_j) \rightarrow \sqrt{2\pi} \lambda \sigma^2 \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \]
Gaussian Process Regression
aka. Kriging, kernel-ridge regression, . . .

\[ p(f) = \mathcal{GP}(0, k) \quad k(a, b) = \exp \left( -\frac{(a - b)^2}{2\lambda^2} \right) \]
Gaussian Process Regression
aka. Kriging, kernel-ridge regression,

\[
p(f \mid y) = \mathcal{GP}(f_x; k_{xx}(k_{xx} + \sigma^2 I)^{-1}y, k_{xx} - k_{xx}(k_{xx} + \sigma^2 I)^{-1}k_{xx})
\]
For \( \phi_i(x) = \mathbb{I}(x > c_i)(x - c_i) \), an analogous limit gives
The prior still matters
just one other example out of the space of kernels

\[ p(f) = \mathcal{GP}(0, k) \text{ with } k(a, b) = \theta^{2/3} \min(a, b)^3 + |a - b| \min(a, b)^2. \]

the integrated Wiener process, aka. cubic splines.

More on GPs in Paris Perdikaris’ tutorial.
more on nonparametric models in Neil Lawrence’s and Tamara Broderick’s talks?
The Computational Challenge

large-scale linear algebra

\[ \alpha := \begin{pmatrix} (k_{xx} + \sigma^2 I)^{-1} \\ k_{ax} (k_{xx} + \sigma^2 I)^{-1} k_{xb} \\ \log |k_{xx} + \sigma^2 I| \end{pmatrix} \in \mathbb{R}^{N \times N}, \text{symm. pos. def.} \]
The Computational Challenge
large-scale linear algebra

\[
\alpha := \underbrace{(k_{XX} + \sigma^2 I)^{-1}} y \quad \kappa_a X (k_{XX} + \sigma^2 I)^{-1} k_{Xb} \quad \log |k_{XX} + \sigma^2 I| \\
\in \mathbb{R}^{N \times N}, \text{symm. pos. def.}
\]

Methods in wide use:
- exact linear algebra (BLAS), for \( N \lesssim 10^4 \) (because \( \mathcal{O}(N^3) \))
- (rarely:) iterative Krylov solvers (in part. conjugate gradients), for \( N \lesssim 10^5 \)

For large-scale (\( \mathcal{O}(NM^2) \)):
- inducing point methods, Nyström, etc.: using iid. structure of data
  \[
  \kappa_{ab} \approx \tilde{\kappa}_{au} \Omega^{-1} \tilde{\kappa}_{ub} \quad \Omega^{-1} \in \mathbb{R}^{M \times M}
  \]
  - Williams & Seeger, 2001; Quiñonero & Rasmussen, 2005;
  - Snelson & Ghahramani, 2007; Titsias, 2009
- spectral expansions using algebraic properties of kernel
  - Rahimi & Recht 2008; 2009
- in univariate setting: filtering using Markov structure
  - Särkkä 2013

Both are linear time, with finite error. Bridge to iterative methods is beginning to form, via sub-space recycling (de Roos & P.H., arXiv 1706.00241 2017)
popular extensions no. 2:
requires large-scale nonlinear optimization

Maximum Likelihood estimation: Assume $\phi(x) = \phi_\theta(x)$

$$L(y; \theta, w) = \log p(y \mid \phi, w) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} \| y_i - \phi_\theta(x_i)^T w \|^2 + \text{const.}$$

(A feed-forward network)
Learning Features

a (in general) **non-convex**, non-linear optimization problem

\[
L(y; \theta, w) = \log p(y \mid \phi, w) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} \| y_i - \phi_\theta(x_i)^T w \|^2 + \text{const.}
\]

\[
\nabla_\theta L = \frac{1}{\sigma^2} \sum_{i=1}^{N} -(y_i - \phi_\theta(x_i)^T w) \cdot w^T \nabla_\theta \phi(x_i)
\]

“back-propagation”
Deep Learning
(really just a quick peek)

in practice:

- multiple input dimensions (e.g. pixel intensities)
- multi-dimensional output (e.g. structured sentences)
- multiple feature layers
- structured layers (convolutions, pooling, pyramids, etc.)
Deep Learning has become Mainstream
an increasingly professional industry

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is $150,528$-dimensional, and the number of neurons in the network’s remaining layers is given by $253,440 - 186,624 - 64,896 - 64,896 - 43,264 - 4096 - 4096 - 1000$.

Krizhevsky, Sutskever & Hinton
"ImageNet Classification with Deep Convolutional Neural Networks"

and continues to impress predicting whole-image semantic labels


Zhao, Mathieu & LeCun, "Energy-based generative adversarial networks". Int. Conf. on Learning Representations (ICLR) 2017
The Computational Challenge
high-dimensional, non-convex, stochastic optimization

- contemporary problems are extremely high-dimensional $N > 10^7$
- typically badly conditioned
- optimizer interacts with model
- biggest challenge: stochasticity

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i; \theta) \approx \frac{1}{M} \sum_{j=1}^{M} \ell(y_j; \theta) =: \hat{L}(\theta)$$

$$p(\hat{L} \mid L) \approx \mathcal{N} \left( \hat{L}; L, \mathcal{O} \left( \frac{N - M}{M} \right) \right)$$

classic optimization paradigms break down.

- currently dominant optimizers are surprisingly simple:
  - stochastic gradient descent
  - RMSPROP
  - ADADELTA
  - ADAM

more in part II ...
popular extension no. 3 requires high-dimensional integration of probability measures

- in $p(f) = GP(0, k)$, what should $k$ be?
- parametrize $k = k^\theta$, $\mu = \mu^\theta$, $\Lambda = \Lambda^\theta$

\[
p(y \mid \theta) = \int p(y \mid f, \theta)p(f \mid \theta) df = \int \mathcal{N}(y; f_x, \Lambda^\theta)GP(f; \mu^\theta, k^\theta) \\
= \mathcal{N}(y, \mu_x^\theta, \Lambda^\theta + k_{XX}^\theta)
\]

\[
p(f \mid y) = \int p(f \mid y, \theta)p(\theta \mid y) d\theta
\]
Learning the kernel
hierarchical Bayesian inference

- practical cases can be extremely high-dimensional
  (→ Bayesian deep learning)

- standard approaches:
  - free energy minimization of a parametric approximation
  - Markov Chain Monte Carlo

- elaborate toolboxes available
  (→ probabilistic programming)

- but few (practically relevant) finite-time guarantees

more about hierarchical Bayesian inference in Tamara Broderick’s talk?
non-convex (multi-modal!) global optimization
expensive evaluations

more about optimization of architectures in Roman Garnett's talk
Summary: The Computations of Acting Agents

- machine intelligence requires computations
  - integration for marginalization
  - optimization for fitting
  - differential equations for control
  - linear algebra for all of the above

- contemporary AI problems pose very challenging numerical problems
- **uncertainty from data-subsampling** plays a crucial, intricate role
- classic numerical methods leave room for improvement

**after coffee:**
Learning machines don’t just pose problems—they also promise some answers.
Is there room at the bottom?
ML computations are dominated by numerical tasks

<table>
<thead>
<tr>
<th>task</th>
<th>... amounts to ...</th>
<th>... using black box</th>
</tr>
</thead>
<tbody>
<tr>
<td>marginalize</td>
<td>integration</td>
<td>MCMC, Variational, EP, ...</td>
</tr>
<tr>
<td>train/fit</td>
<td>optimization</td>
<td>SGD et al., quasi-Newton, ...</td>
</tr>
<tr>
<td>predict/control</td>
<td>ord. diff. Eq.</td>
<td>Euler, Runge-Kutta, ...</td>
</tr>
<tr>
<td>Gauss/kernel/LSq.</td>
<td>linear Algebra</td>
<td>Chol., CG, spectral, low-rank, ...</td>
</tr>
</tbody>
</table>

- Scientific computing has produced a very efficient toolchain, but we are (usually) only using generic methods!
- **methods on loan** do not address some of ML’s special needs
  - overly generic algorithms are inefficient
  - Big Data-specific challenges not addressed by “classic” methods

ML deserves customized numerical methods.
And as it turns out, we already have the right concepts!
Numerical methods estimate latent quantities given the result of computations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Estimate</th>
<th>Given</th>
</tr>
</thead>
<tbody>
<tr>
<td>integration</td>
<td>( \int_a^b f(x) , dx )</td>
<td>{f(x_i)}</td>
</tr>
<tr>
<td>linear algebra</td>
<td>( x ) s.t. ( Ax = b )</td>
<td>{As = y}</td>
</tr>
<tr>
<td>optimization</td>
<td>( x ) s.t. ( \nabla f(x) = 0 )</td>
<td>{\nabla f(x_i)}</td>
</tr>
<tr>
<td>analysis</td>
<td>( x(t) ) s.t. ( x' = f(x, t) )</td>
<td>{f(x_i, t_i)}</td>
</tr>
</tbody>
</table>

It is thus possible to build probabilistic numerical methods that use probability measures as in- and outputs, and assign a notion of uncertainty to computation.
Integration as Gaussian regression

\[ f(x) = \exp(-\sin(3x)^2 - x^2) \]  

\[ F = \int_{-3}^{3} f(x) \, dx = ? \]
A Wiener process prior \( p(f, F) \).

Bayesian Quadrature

\[ p(f) = \mathcal{GP}(f; 0, k) \]

\[ k(x, x') = \min(x, x') + c \]

\[ \Rightarrow p \left( \int_a^b f(x) \, dx \right) = \mathcal{N} \left( \int_a^b f(x) \, dx; \int_a^b m(x) \, dx, \int_a^b \int_a^b k(x, x') \, dx \, dx' \right) \]

\[ = \mathcal{N}(F; 0, -\frac{1}{6}(b^3 - a^3) + \frac{1}{2}[b^3 - 2a^2b + a^3] - (b - a)^2c) \]
...conditioned on actively collected information...

computation as the collection of information

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1, \ldots, x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information...

computation as the collection of information

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1, \ldots, x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information...

computation as the collection of information

\[ x_t = \arg \min \left[ \text{var}_p(F|x_1,\ldots,x_{t-1})(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information ...

computation as the collection of information

\[ f(x) \]

\[ |F - \hat{F}| \]

\[ x_t = \arg \min \left[ \text{var}_p(F|x_1, \ldots, x_{t-1})(F) \right] \]

* maximal reduction of variance yields regular grid
...conditioned on actively collected information ...

computation as the collection of information

\[ f(x) \]

\[ |F - \hat{F}| \]

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1,\ldots,x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information ...

computation as the collection of information

\[ f(x) \]

\[ |F - \hat{F}| \]

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1,\ldots,x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information...
computation as the collection of information

\[
x_t = \arg \min \left[ \text{var}_{p(F|x_1, \ldots, x_{t-1})}(F) \right]
\]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information...

Computation as the collection of information

\[ f(x) \]

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1,\ldots,x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information ...

computation as the collection of information

\[ f(x) = \text{arg min} \left[ \text{var}_{p(F|x_1, \ldots, x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...conditioned on actively collected information...

computation as the collection of information

\[ f(x) \]

\[ |F - \hat{F}| \]

\[ x_t = \arg \min \left[ \text{var}_p(F | x_1, \ldots, x_{t-1}) (F) \right] \]

+ maximal reduction of variance yields regular grid
...conditioned on actively collected information...

computation as the collection of information

\[ x_t = \arg \min \left[ \text{var}_{p(F|x_1,\ldots,x_{t-1})}(F) \right] \]

- maximal reduction of variance yields regular grid
...yields the trapezoid rule!

![Graph showing trapezoid rule](image)

\[
E_y[F] = \int E_{y[f(x)]} \, dx = \sum_{i=1}^{N-1} (x_{i+1} - x_i) \frac{1}{2} (f(x_{i+1}) + f(x_i))
\]

- **Trapezoid rule** is **MAP** estimate under Wiener process prior on \( f \)
- regular grid is optimal expected information choice
- error estimate is **under-confident**

more about **calibration** of uncertainty in the talks of **Chris Oates** and **John Cockayne**.
Estimate \( z \) from computations \( c \), under model \( m \).

\[
p(z \mid c, m) = \frac{p(z \mid m)p(c \mid z, m)}{\int p(z \mid m)p(c \mid z, m) \, dz}
\]
Classic methods as basic probabilistic inference
maximum a-posteriori estimation in Gaussian models

Quadrature
- Gaussian Quadrature
- GP Regression

Linear Algebra
- Conjugate Gradients
- Gaussian Regression

Nonlinear Optimization
- BFGS / Quasi-Newton
- Autoregressive Filtering

Differential Equations
- Runge-Kutta; Nordsieck Methods
- Gauss-Markov Filters


[Hennig 2014]

[Hennig & Kiefel 2013]

[Schober, Duvenaud & Hennig 2014; Kersting & Hennig 2016; Schober & Hennig 2016]
There is a class of **solvers for initial value problems** that

- has the same **complexity** as multi-step methods
- has **high local approximation order** $q$ (like classic solvers)
- has **calibrated posterior uncertainty** (order $q + 1/2$)

- this method $\mapsto$ Hans Kersting’s talk.  
  https://github.com/ProbabilisticNumerics/pfos
- calibration $\mapsto$ Oksana Chkrebtii’s talk.
- convergence $\mapsto$ Tim Sullivan’s talk.
There is a class of solvers for initial value problems that

- has the same **complexity** as multi-step methods
- has **high local approximation order** $q$ (like classic solvers)
- has **calibrated posterior uncertainty** (order $q + 1/2$)

- **this method**  →  Hans Kersting’s talk.  
  https://github.com/ProbabilisticNumerics/pfos
- **calibration**  →  Oksana Chkrebtii’s talk.
- **convergence**  →  Tim Sullivan’s talk.
There is a class of solvers for initial value problems that
- has the same complexity as multi-step methods
- has high local approximation order $q$ (like classic solvers)
- has calibrated posterior uncertainty (order $q + 1/2$)

- this method → Hans Kersting’s talk. https://github.com/ProbabilisticNumerics/pfos
- calibration → Oksana Chkreibti’s talk.
- convergence → Tim Sullivan’s talk.
There is a class of **solvers for initial value problems** that

- has the same **complexity** as multi-step methods
- has **high local approximation order** $q$ (like classic solvers)
- has **calibrated posterior uncertainty** (order $q + 1/2$)

- **this method** → Hans Kersting’s talk.  
  [https://github.com/ProbabilisticNumerics/pfos](https://github.com/ProbabilisticNumerics/pfos)
- **calibration** → Oksana Chkrebtii’s talk.
- **convergence** → Tim Sullivan’s talk.
There is a class of **solvers for initial value problems** that

- has the same **complexity** as multi-step methods
- has high local approximation order $q$ (like classic solvers)
- has calibrated posterior uncertainty (order $q + 1/2$)

- **this method** → Hans Kersting’s talk. [https://github.com/ProbabilisticNumerics/pfos](https://github.com/ProbabilisticNumerics/pfos)
- **calibration** → Oksana Chkrebtii’s talk.
- **convergence** → Tim Sullivan’s talk.
There is a class of solvers for initial value problems that
- has the same complexity as multi-step methods
- has high local approximation order $q$ (like classic solvers)
- has calibrated posterior uncertainty (order $q + 1/2$)

- this method $\rightarrow$ Hans Kersting’s talk.
- calibration $\rightarrow$ Oksana Chkrebtii’s talk.
- convergence $\rightarrow$ Tim Sullivan’s talk.

$x'(t) = f(x(t), t), \quad x(t_0) = x_0$
There is a class of solvers for initial value problems that
- has the same complexity as multi-step methods
- has high local approximation order $q$ (like classic solvers)
- has calibrated posterior uncertainty (order $q + 1/2$)

- this method $\rightarrow$ Hans Kersting’s talk.
- calibration $\rightarrow$ Oksana Chkrebtii’s talk.
- convergence $\rightarrow$ Tim Sullivan’s talk.

$x'(t) = f(x(t), t), \quad x(t_0) = x_0$
- Probabilistic numerics can be as fast and reliable as classic ones.
- Computation can be phrased on ML language!
- Meaningful (calibrated) uncertainty can be constructed at minimal computational overhead (dominated by cost of point estimate)

So what does this mean for Data Science / ML / AI?
New Functionality, and new Challenges
making use of the probabilistic numerics perspective

Prior: structural knowledge reduces complexity.

$$p(z \mid c, m) = \frac{p(z \mid m)p(c \mid z, m)}{\int p(z \mid m)p(c \mid z, m) \, dz}$$

Likelihood:

Posterior:

Evidence:
An integration prior for probability measures

WARped Sequential Active Bayesian Integration (WSABI)  Guenter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

A prior specifically for integration of probability measures

- $f > 0$ ($f$ is probability measure)
- $f \propto \exp(-x^2)$ ($f$ is product of prior and likelihood terms)
- $f \in C^\infty$ ($f$ is smooth)

Explicit prior knowledge yields reduces complexity.

cf. **information-based complexity**.  

more on this connection in Houman Owhadi's tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)  
Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)  Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)

Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI) □ Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)  
Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI) □ Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)  
Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.

more on this connection in Houman Owhadi’s tutorial?
An integration prior for probability measures

WArped Sequential Active Bayesian Integration (WSABI)  Gunter, Osborne, Garnett, Hennig, Roberts. NIPS 2014

- adaptive node placement
- scales to, in principle, arbitrary dimensions
- faster (in wall-clock time) than MCMC

Explicit prior knowledge yields reduces complexity.

cf. information-based complexity.


more on this connection in Houman Owhadi's tutorial?
Estimate $z$ from computations $c$, under model $m$.

Prior: structural knowledge reduces complexity

Likelihood: modeling imprecise computation reduces cost

$$p(z \mid c, m) = \frac{p(z \mid m)p(c \mid z, m)}{\int p(z \mid m)p(c \mid z, m) \, dz}$$

Posterior: Evidence:
New numerics for Big Data

Uncertainty on Inputs directly effecting numerical decisions

In Big Data setting, batching introduces (Gaussian) noise

\[
\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i; \theta) \approx \frac{1}{M} \sum_{j=1}^{M} \ell(y_j; \theta) =: \hat{\mathcal{L}}(\theta) \quad M \ll N
\]

\[
p(\hat{\mathcal{L}} \mid \mathcal{L}) \approx \mathcal{N} \left( \hat{\mathcal{L}}; \mathcal{L}, \mathcal{O} \left( \frac{N - M}{M} \right) \right)
\]
New numerics for Big Data
Uncertainty on Inputs directly effecting numerical decisions

In Big Data setting, batching introduces (Gaussian) noise

\[
\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i; \theta) \approx \frac{1}{M} \sum_{j=1}^{M} \ell(y_j; \theta) =: \hat{\mathcal{L}}(\theta)
\]

\[
p(\hat{\mathcal{L}} | \mathcal{L}) \approx \mathcal{N}\left(\hat{\mathcal{L}}; \mathcal{L}, \mathcal{O}\left(\frac{N-M}{M}\right)\right)
\]

Classic methods are unstable to noise. E.g.: step size selection

\[
\theta_{t+1} = \theta_t - \alpha_t \nabla \hat{\mathcal{L}}(\theta_t)
\]
Probabilistic Line Searches

Step-size selection stochastic optimization

Mahsereci & Hennig, NIPS 2015

classic line search: unstable

probabilistic line search: stable

two-layer feed-forward perceptron on CIFAR 10. Details, additional results in Mahsereci & Hennig, NIPS 2015.

https://github.com/ProbabilisticNumerics/probabilistic_line_search

+ **batch-size selection**
+ **early stopping**
+ **search directions**

[batch-size selection](https://github.com/ProbabilisticNumerics/batch-size_selection)  □ Balles & Hennig, arXiv 1612.05086
[early stopping](https://github.com/ProbabilisticNumerics/early_stopping) □ Mahsereci, Balles & Hennig, arXiv 1703.09580
[search directions](https://github.com/ProbabilisticNumerics/search_directions) □ Balles & Hennig, arXiv 1705.07774
Estimate \( z \) from computations \( c \), under model \( m \).

Prior: structural knowledge reduces complexity

Likelihood: modeling imprecise computation reduces cost

Posterior: tracking uncertainty for robustness

Evidence:

\[
p(z \mid c, m) = \frac{p(z \mid m)p(c \mid z, m)}{\int p(z \mid m)p(c \mid z, m) \, dz}
\]

Uncertainty Across Composite Computations

interacting information requirements

Hennig, Osborne, Girolami, Proc. Royal Society A 2015

probabilistic numerical methods taking and producing uncertain inputs and outputs allow management of computational resources

more on uncertainty propagation in Ilias Bilionis’ talk.
Estimate $z$ from computations $c$, under model $m$.

**Prior:** structural knowledge reduces complexity  

**Likelihood:** modeling imprecise computation reduces cost  

$$p(z \mid c, m) = \frac{p(z \mid m)p(c \mid z, m)}{\int p(z \mid m)p(c \mid z, m) \, dz}$$

**Posterior:** tracking uncertainty for robustness  

**Evidence:** checking models for safety

Probabilistic Certification?


\[ f(x) - \mu(x) = (f(x) - \mu(x))^T K^{-1} (f(x) - \mu(x)) - N \]
Uncertain computation as and for machine learning

- **computation is inference** $\rightarrow$ **probabilistic numerical methods**
  - probability measures for **uncertain** inputs and outputs
  - classic methods as special cases

New concepts not just for Machine Learning:

- **prior**: structural knowledge reduces complexity
- **likelihood**: imprecise computation lowers cost
- **posterior**: uncertainty propagated through computations
- **evidence**: model mismatch detectable at run-time

- ML & AI pose **new** computational challenges
- computational methods can be phrased in the concepts of ML
- **but** related results of mathematics are currently “under-explored”
- more about all of this in **this seminar**!

http://probnum.org  https://pn.is.tue.mpg.de