UPPSALA
UNIVERSITET

Composing stochastic quasi-Newton-type algorithms

Thomas Schön, Uppsala University

Joint work with **Adrian Wills** at the University of Newcastle, Australia.

Harvard University,
February 26, 2018.

Mindset — Numerical methods are inference algorithms

A numerical method **estimates** a certain **latent** property **given** the result of computations.

Computation is inference meaning that numerical methods can be interpreted as estimation/learning algorithms.

Basic numerical methods and basic statistical models are **deeply connected in formal ways**!

Poincaré, H. **Calcul des probabilités**. Paris: Gauthier-Villars, 1896.

Diaconis, P. **Bayesian numerical analysis**. *Statistical decision theory and related topics*, IV(1), 163–175, 1988.

O'Hagan, A. **Some Bayesian numerical analysis**. *Bayesian Statistics*, 4, 345–363, 1992.

Hennig, P., Osborne, M. A., and Girolami, M. **Probabilistic numerics and uncertainty in computations**. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

probabilistic-numerics.org/

Mindset — Numerical methods are inference algorithms

The task of a numerical algorithm is

to estimate unknown quantities from known ones.

Ex) basic algorithms that are equivalent to Gaussian MAP inference:

- Conjugate Gradients for linear algebra
- BFGS for nonlinear optimization
- Gaussian quadrature rules for integration
- Runge-Kutta solvers for ODEs

The structure of num. algs. is similar to statistical inference where

- The **tractable quantities** play the role of **"data"/"observations"**.
- The **intractable quantities** relate to **"latent"/"hidden"** quantities.

Problem formulation

If computation is inference maybe it is possible to use this in deriving new (and possibly more capable) algorithms.

What? Solve the non-convex stochastic optimization problem

$$\max_{x} f(x)$$

when we only have access to **noisy** evaluations of $f(x)$ and its derivatives.

Why? These stochastic optimization problems are common:

- The cost function cannot be evaluated on the entire dataset.
- When numerical methods approximate $f(x)$ and $\nabla^i f(x)$.
- ...

How? — our contribution

How? Learn a probabilistic nonlinear model of the Hessian.

Provides a local approximation of the cost function $f(x)$.

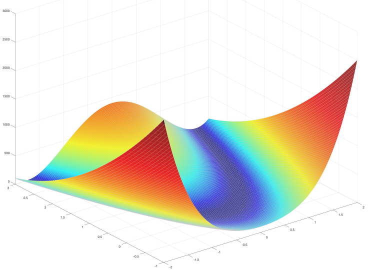
Use this local model to compute a search direction.

Captures second-order information (curvature) which opens up for better performance compared to a pure gradient-based method.

Let $f(x) = (a - x_1)^2 + b(x_2 - x_1^2)^2$, where $a = 1$ and $b = 100$.

Deterministic problem

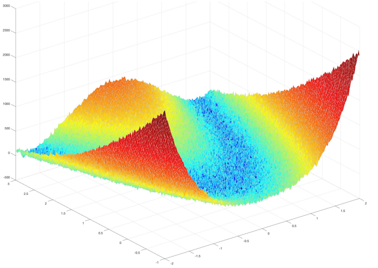
$$\max_{x} f(x)$$



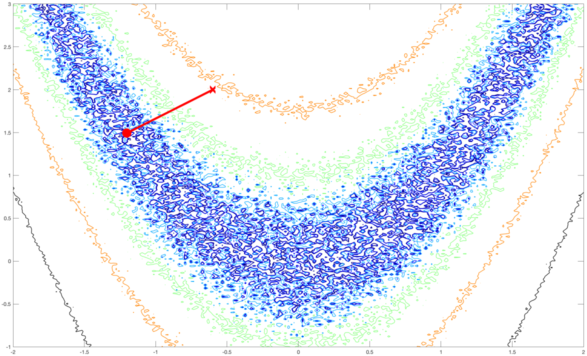
Stochastic problem

$$\max_{x} f(x)$$

when we only have access to noisy versions of the cost function ($\widetilde{f}(x) = f(x) + e$, $e = \mathcal{N}(0, 30^2)$) and its gradients.



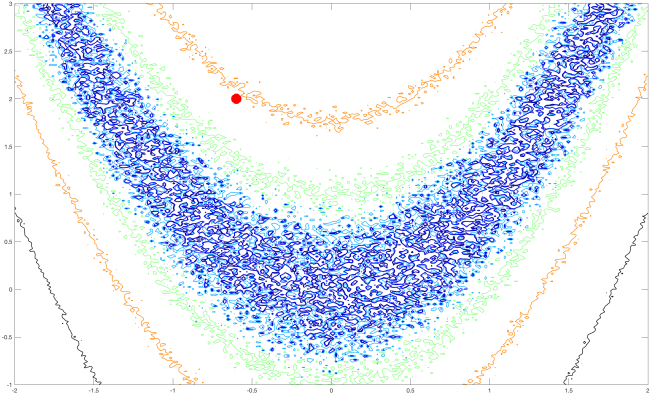
fminunc at work



Terminates at the wrong solution after 3 iterations.

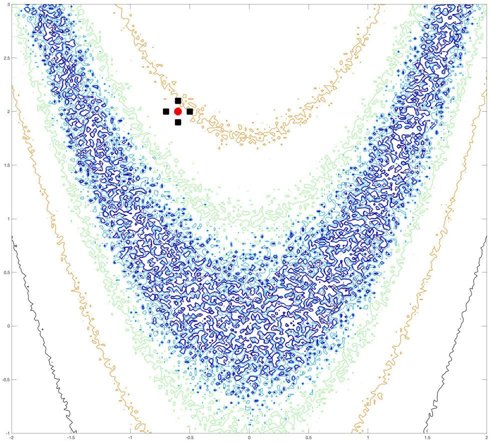
The true solution is $(1, 1)$.

Adam at work

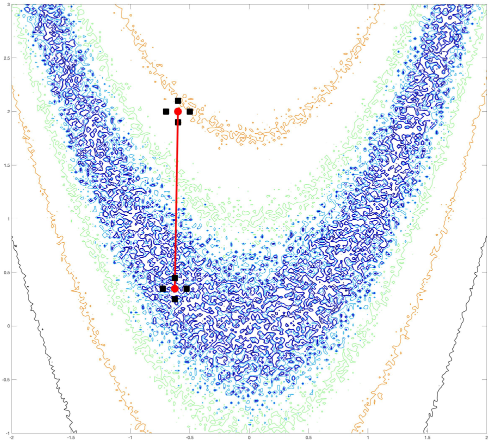


By not using the curvature information we expose ourself to the "banana-problem".

New algorithm at work — iteration 1

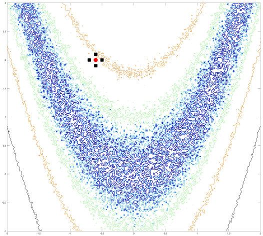


New algorithm at work — iteration 2

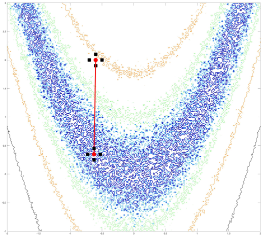


New algorithm at work — overall result

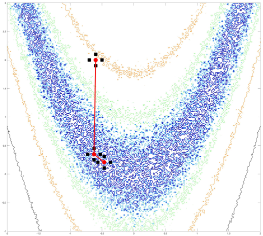
Initial value



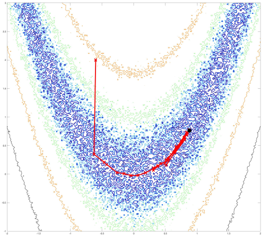
Iteration 1



Iteration 2



Iteration 50



Aim: Derive a stochastic quasi-Newton algorithm.

Spin-off: Combine it with particle filters for maximum likelihood identification in nonlinear state space models.

1. Mindset (probabilistic numerics) and problem formulation
2. **A non-standard take on quasi-Newton**
3. μ on the Gaussian process (GP)
4. Assembling a new stochastic optimization algorithm
   a. Representing the Hessian with a GP
   b. Learning the Hessian
5. Testing ground – maximum likelihood in SSMs
6. Some ongoing research (if there is time)

Quasi-Newton — A non-standard take

Our problem is of the form

$$\max_{x} f(x)$$

Idea underlying (quasi-)Newton methods: Learn a local quadratic model $q(x_k, \delta)$ of the cost function $f(x)$ around the current iterate x_k

$$q(x_k, \delta) = f(x_k) + g(x_k)^\mathsf{T}\delta + \frac{1}{2}\delta^\mathsf{T} H(x_k)\delta$$

A second-order Taylor expansion around x_k, where

$$g(x_k) = \nabla f(x)\big|_{x=x_k},$$
$$H(x_k) = \nabla^2 f(x)\big|_{x=x_k},$$
$$\delta = x - x_k.$$

Available data

We have measurements of the

- cost function $f_k = f(x_k)$
- and its gradient $g_k = g(x_k)$.

Question: How do we update the Hessian model?

Line segment connecting two adjacent iterates x_k and x_{k+1}:

$$r_k(\tau) = x_k + \tau(x_{k+1} - x_k), \qquad \tau \in \{0, 1\}.$$

Useful basic facts

The fundamental theorem of calculus states that

$$\int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) \mathrm{d}\tau = \nabla f(r_k(1)) - \nabla f(r_k(0)) = \underbrace{\nabla f(x_{k+1})}_{g_{k+1}} - \underbrace{\nabla f(x_k)}_{g_k}$$

and the chain rule tells us that

$$\frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) = \nabla^2 f(r_k(\tau)) \frac{\partial r_k(\tau)}{\partial \tau} = \nabla^2 f(r_k(\tau))(x_{k+1} - x_k).$$

$$\underbrace{g_{k+1} - g_k}_{=y_k} = \int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) \mathrm{d}\tau = \int_0^1 \nabla^2 f(r_k(\tau)) \mathrm{d}\tau (\underbrace{x_{k+1} - x_k}_{s_k}).$$

Result — the quasi-Newton integral

With the definitions $y_k \triangleq g_{k+1} - g_k$ and $s_k \triangleq x_{k+1} - x_k$ we have

$$y_k = \int_0^1 \nabla^2 f(r_k(\tau)) \mathrm{d}\tau s_k.$$

Interpretation: The difference between two consecutive gradients (y_k) constitute a *line integral observation of the Hessian*.

Problem: Since the Hessian is unknown there is no functional form available for it.

Solution 1 — recovering existing quasi-Newton algorithms

Existing quasi-Newton algorithms (e.g. BFGS, DFP, Broyden's method) assume the Hessian to be constant

$$\nabla^2 f(r_k(\tau)) \approx H_{k+1}, \qquad \tau \in \{0, 1\},$$

implying the following approximation of the integral (**secant condition**)

$$y_k = H_{k+1} s_k.$$

Find H_{k+1} by **regularizing** H:

$$\begin{aligned} H_{k+1} = \min_H \quad & \|H - H_k\|_W^2, \\ \text{s.t.} \quad & H = H^\top, \quad H s_k = y_k, \end{aligned}$$

Equivalently, the existing quasi-Newton methods can be interpreted as **particular instances of Bayesian linear regression**.

Philipp Hennig. **Probabilistic interpretation of linear solvers**, *SIAM Journal on Optimization*, 25(1):234–260, 2015.

Solution 2 — use a flexible nonlinear model

Our approach is fundamentally different.

Recall that the problem is **stochastic** and **nonlinear**.

Hence, we need a model that can deal with such a problem.

Idea: Represent the Hessian using a **Gaussian process** learnt from data.

Two of the remaining challenges:

1. Can we use line integral observations when learning a GP?
2. How do we ensure that the resulting GP represents a Hessian?

μ on the Gaussian process (GP)

The Gaussian process is a model for nonlinear functions

Q: Why is the Gaussian process used everywhere?

It 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 (Bayesian) linear regression

$$y_t = \underbrace{\theta^\mathsf{T} \mathbf{x}_t}_{f(\mathbf{x}_t)} + e_t, \qquad e_t \sim \mathcal{N}(0, \sigma^2),$$

we place a prior on θ, e.g. $\theta \sim \mathcal{N}(0, \alpha^2 I)$.

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

$$f \sim p(f)$$

and look for $p(f \mid y_{1:T})$ rather than $p(\theta \mid y_{1:T})$?!

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

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

If we accept the above idea we can without conceptual problems generalize to any *arbitrary* finite set of input values $\{x_1, x_2, \ldots, x_T\}$.

$$\begin{pmatrix} f(x_1) \\ \vdots \\ f(x_T) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(x_1) \\ \vdots \\ m(x_N) \end{pmatrix}, \begin{pmatrix} k(x_1, x_1) & \ldots & k(x_1, x_T) \\ \vdots & \ddots & \vdots \\ k(x_T, x_1) & \ldots & k(x_T, x_T) \end{pmatrix}\right)$$

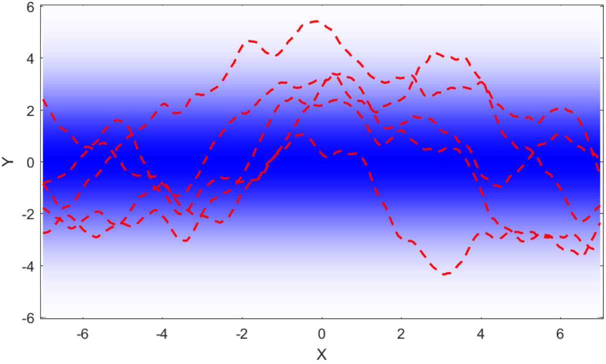
Definition

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.

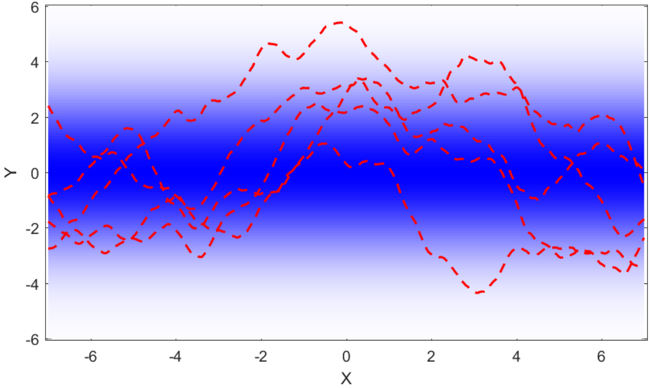
We now have a prior!

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

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



GP regression – illustration



Stochastic optimization

GP prior for the Hessian

Stochastic quasi-Newton integral

$$y_k = \int_0^1 \underbrace{B(r_k(\tau))}_{=\nabla^2 f(r_k(\tau))} s_k \mathrm{d}\tau + e_k,$$

corresponds to noisy (e_k) gradient observations.

Since $B(x)s_k$ is a column vector, the integrand is given by

$$\mathrm{vec}\left(B(x)s_k\right) = \left(s_k^\mathsf{T} \otimes I\right) \mathrm{vec}\left(B(x)\right) = \left(s_k^\mathsf{T} \otimes I\right) \mathrm{vec}\left(B(x)\right),$$

where $\mathrm{vec}\left(B(x)\right) = D \underbrace{\mathrm{vech}\left(B(x)\right)}_{\widetilde{B}(x)}$.

Let us use a GP model for the unique elements of the Hessian

$$\widetilde{B}(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')).$$

Resulting stochastic qN integral and Hessian model

Summary: resulting stochastic quasi-Newton integral:

$$y_k = \underbrace{(s_k^{\mathsf{T}} \otimes I)D}_{=\bar{D}_k} \int_0^1 \widetilde{B}(r_k(\tau))\mathrm{d}\tau + e_k,$$

with the following model for the Hessian

$$\widetilde{B}(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')).$$

The Hessian can now be estimated using tailored GP regression.

Linear transformations (such as an integral or a derivative) of a GP results in a new GP.

Resulting stochastic optimization algorithm

Standard non-convex numerical optimization loop with **non-standard components**.

Algorithm 1 Probabilistic optimization

1. **Initialization ($k = 1$)**
2. **while** *not terminated* **do**
   (a) Compute a search direction p_k using the current approximation of the gradient g_k and Hessian B_k.
   (b) Probabilistic line search to find a step length α_k and set
   $$x_{k+1} = x_k + \alpha_k p_k.$$
   (c) Set $k := k + 1$
   (d) Update the Hessian estimate (tailored GP regression)
3. **end while**

Testing ground – nonlinear sys.id.

Probabilistic modelling of dynamical systems

$$
\begin{aligned}
x_t &= f(x_{t-1}, \theta) + w_t, \\
y_t &= g(x_t, \theta) + e_t, \\
x_0 &\sim p(x_0 \mid \theta), \\
(\theta &\sim p(\theta)).
\end{aligned}
$$

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

Corresponding full probabilistic model:

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

Model = probability distribution!

Maximum likelihood nonlinear system identification

Maximum likelihood – model the unknown parameters as a deterministic variable θ and solve

$$\max_{\theta} p(y_{1:T} \mid \theta),$$

Challenge: The optimization problem is stochastic!

Cost function – the likelihood

Each element $p(y_t \mid y_{1:t-1}, \theta)$ in the likelihood

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \theta),$$

can be computed by averaging over all possible values for the state x_t,

$$p(y_t \mid y_{1:t-1}, \theta) = \int p(y_t \mid x_t, \theta) \underbrace{p(x_t \mid y_{1:t-1}, \theta)}_{\text{approx. by PF}} \mathrm{d}x_t.$$

Non-trivial fact: The likelihood estimates obtained from the particle filter (PF) are **unbiased**.

Tutorial paper on the use of the PF (an instance of sequential Monte Carlo, SMC) for nonlinear system identification

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wagberg, Christian A. Naesseth, Andreas Svensson and Liang Dai. **Sequential Monte Carlo methods for system identification**, *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, Beijing, China, October 2015.

ex) Simple linear toy problem

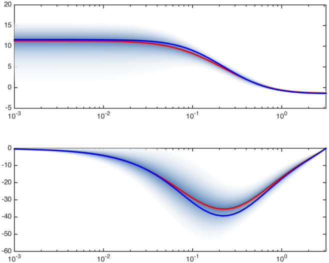
Identify the parameters $\theta = (a, c, q, r)^{\mathsf{T}}$ in

$$
\begin{aligned}
x_{t+1} &= a x_t + w_t, & w_t &\sim \mathcal{N}(0, q), \\
y_t &= c x_t + e_t, & e_t &\sim \mathcal{N}(0, r).
\end{aligned}
$$

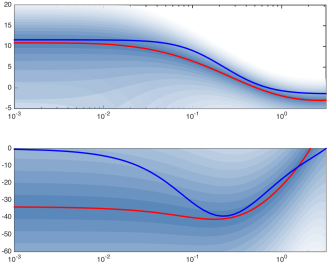
Observations:

- The likelihood $L(\theta) = p(y_{1:T} \mid \theta)$ and its gradient $\nabla_\theta L(\theta)$ are available in closed form via standard Kalman filter equations.
- Standard gradient-based search algorithms applies.
- Deterministic optimization problem ($L(\theta), \nabla_\theta L(\theta)$ noise-free).

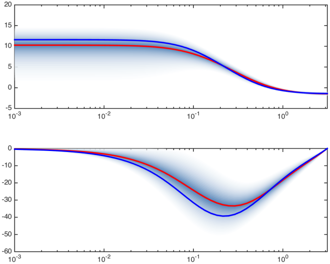
ex) Simple linear toy problem



Both alg. for in the noise-free case.

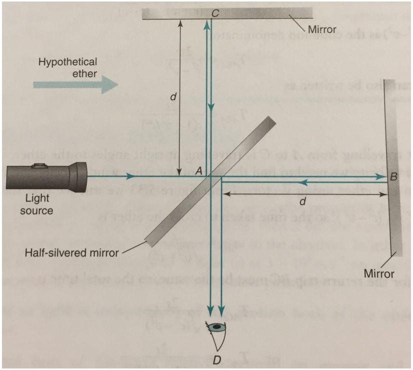


Classical BFGS alg. for noisy observations of $L(\theta)$ and $\nabla L(\theta)$.



GP-based BFGS alg. with noisy observations of $L(\theta)$ and $\nabla L(\theta)$.

ex) laser interferometry



The classic Michelson-Morley experiment from 1887.

Idea: Merge two light sources to create an interference pattern by superposition.

Two cases:

1. Mirror B and C at the **same** distance from mirror A.
2. Mirror B and C at **different** distances from mirror A.

ex) laser interferometry

Dynamics: constant velocity model (with unknown force w)

$$\begin{pmatrix} \dot{p} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ w \end{pmatrix}.$$

Measurements: generated using two detectors

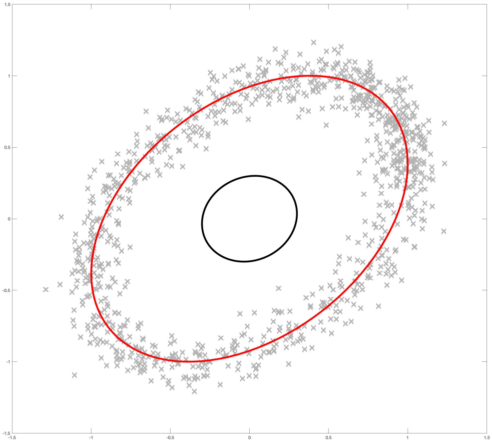
$$\begin{aligned} y_1 &= \alpha_0 + \alpha_1 \cos(\kappa p) + e_1, & e_1 &\sim \mathcal{N}(0, \sigma^2), \\ y_2 &= \beta_0 + \beta_1 \sin(\kappa p + \gamma) + e_2, & e_2 &\sim \mathcal{N}(0, \sigma^2). \end{aligned}$$

Unknown parameters: $\theta = \begin{pmatrix} \alpha_0 & \alpha_0 & \beta_0 & \beta_1 & \gamma & \sigma \end{pmatrix}^\mathsf{T}$.

Resulting maximum likelihood system identification problem

$$\max_\theta p(y_{1:T} \mid \theta)$$

ex) laser interferometry



Snapshots of some related ongoing research

Snapshot 1 – scaling up to large problems

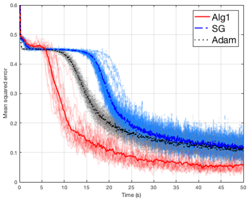
What is the key limitation of our GP-based optimization algorithm?

It **does not** scale to large-scale problems!

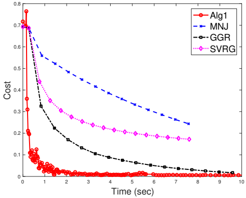
It is still highly useful and competitive for **small to medium** sized problems involving up to a coupled of hundred parameters or so.

We have developed a **new** technique that scales to **very large** problems.

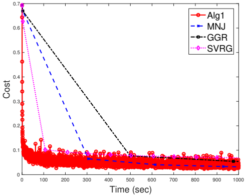
Snapshot 1 – scaling up to large problems



Training a deep CNN for MNIST data.



Logistic loss function with an L2 regularizer, `gisette`, 6 000 observations and 5 000 unknown variables.



Logistic loss function with an L2 regularizer, `URL`, 2 396 130 observations and 3 231 961 unknown variables.

Key innovations

- Replace the GP with a matrix updated using fast Cholesky routines.
- Exploit a receding history of iterates and gradients akin to L-BFGS.
- An auxiliary variable Markov chain construction.

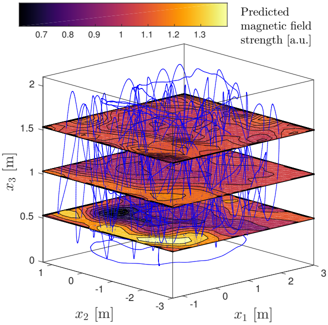
Adrian Wills and Thomas B. Schön. **Stochastic quasi-Newton with adaptive step lengths for large-scale problems**. *arXiv:1802.04310*, February, 2018.

Snapshot 2 – A linearly constrained GP

Innovation: Modification of the covariance function in a GP to correctly account for **known linear operator** constraints.

Contribution:

1. A probabilistic model that is **guaranteed** to fulfil known linear operator constraints.
2. A **constructive procedure** for designing the transformation.



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

Snapshot 3 – GP-based nonlinear state space model

"Inspired by the Gaussian process, enabled by the particle filter"

$$
\begin{aligned}
x_{t+1} &= f(x_t) + w_t, \qquad &&\text{s.t.} \quad f(x) \sim \mathcal{GP}(0, \kappa_{\eta,f}(x, x')),\\
y_t &= g(x_t) + e_t, \qquad &&\text{s.t.} \quad g(x) \sim \mathcal{GP}(0, \kappa_{\eta,g}(x, x')).
\end{aligned}
$$

Results in a **flexible** non-parametric model where the GP prior takes on the **role of a regularizer**.

We can now find the posterior distribution

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

via some approximation (we use **particle MCMC**).

Frigola, Roger, Fredrik Lindsten, Thomas B. Schön, and Carl Rasmussen. **Bayesian inference and learning in Gaussian process state-space models with particle MCMC**. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Andreas Svensson and Thomas B. Schön. **A flexible state space model for learning nonlinear dynamical systems**, *Automatica*, 80:189-199, June, 2017.

Snapshot 4 – The ASSEMBLE project and Birch

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

Swedish Foundation *for* Strategic Research

Keep the model and the learning algorithms **separated**.

Create a **market place** for SMC-based learning algorithms (think CVX).

Birch — Our prototype probabilistic programming language.

Lawrence M. Murray, Daniel Lundén, Jan Kudlicka, David Broman and Thomas B. Schön. **Delayed sampling and automatic Rao-Blackwellization of probabilistic programs**. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, Lanzarote, Spain, April, 2018.

Conclusions

Derived a **probabilistic** quasi-Newton algorithm that can be used with **noisy** observations of the cost function and its derivatives.

- Non-standard interpretation of quasi-Newton.
- Represent the Hessian using a Gaussian process.
- Application: Maximum likelihood estimation in nonlinear SSMs.
- We can scale up to large problems.

Remember to talk to people who work on **different problems** with **different tools**!!

Backup slides

Tailoring GP regression for Hessian estimation

Setting: We put a GP prior on part of the Hessian

$$\widetilde{B}(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')),$$

which is then updated using the measurements via the stochastic quasi-Newton integral:

$$y_k = \underbrace{(s_k^\mathsf{T} \otimes I)D}_{=\bar{D}_k} \int_0^1 \widetilde{B}(r_k(\tau))\mathrm{d}\tau + e_k.$$

The Gaussian process is closed under linear operators implying that

$$y_k \sim \mathcal{N}\left(m_k, K_{kk}\right),$$

where

$$\begin{aligned}
m_k &= \bar{D}_k \int_0^1 \mu(r_k(\tau))\mathrm{d}\tau, \\
K_{kk} &= \bar{D}_k \int_0^1 \int_0^1 \kappa(r_k(\tau), r_k(t))\mathrm{d}\tau\mathrm{d}t \bar{D}_k^\mathsf{T} + R.
\end{aligned}$$

Hessian posterior distribution

Setting: We have training data available in the form $\{s_i, y_i\}_{i=1}^N$.

Model assumptions:

$$\begin{pmatrix} \widetilde{B}_\star \\ \mathbf{y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m_{s_\star} \\ m_{\mathbf{s}} \end{pmatrix}, \begin{pmatrix} K_{s_\star s_\star} & K_{s_\star \mathbf{s}} \\ K_{\mathbf{s} s_\star} & K_{\mathbf{s}\mathbf{s}} \end{pmatrix} \right).$$

$$\mathbf{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_N \end{pmatrix}^\mathsf{T}, \qquad \mathbf{s} = \begin{pmatrix} s_1 & s_2 & \cdots & s_N \end{pmatrix}^\mathsf{T}.$$

Result of using the new Hessian information

$$\begin{aligned} \widetilde{B}_\star \mid \mathbf{y} &\sim \mathcal{N}\left(m_\mathrm{p}, K_\mathrm{p}\right), \\ m_\mathrm{p} &= m_{s_\star} - K_{s_\star \mathbf{s}} K_{\mathbf{s}\mathbf{s}}^{-1}(\mathbf{y} - m_{\mathbf{s}}), \\ K_\mathrm{p} &= K_{s_\star s_\star} - K_{s_\star \mathbf{s}} K_{\mathbf{s}\mathbf{s}}^{-1} K_{\mathbf{s} s_\star}. \end{aligned}$$

GP regression – general

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

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

The conditioning theorem for partitioned Gaussians results in

$$\begin{aligned} f_\star \mid \mathbf{y} &\sim \mathcal{N}\left(\mu_\star, k_\star\right), \\ \mu_\star &= m(x_\star) + \mathbf{s}^\mathsf{T}(\mathbf{y} - m(\mathbf{x})), \\ k_\star &= k(x_\star, x_\star) - \mathbf{s}^\mathsf{T} k(\mathbf{x}, x_\star), \end{aligned}$$

where $\mathbf{s}^\mathsf{T} = k(x_\star, \mathbf{x})(k(\mathbf{x}, \mathbf{x}) + \sigma^2 I_T)^{-1}$.