Black-box optimization

Maxim Borisyak

National Research University Higher School of Economics (HSE)

August 10, 2018

Black-box optimization

–

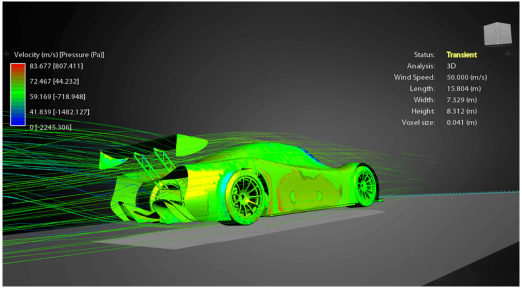
Black-box

Black-box optimization $\iff$ no gradient information.

Examples: car aerodynamics

Design optimization:

- expensive computations;
- even more expensive gradient:
    - and, possibly, unstable.



Status: Transient
Analysis: 3D
Wind Speed: 50.000 (m/s)
Length: 15.804 (m)
Width: 7.329 (m)
Height: 8.312 (m)
Voxel size: 0.041 (m)

Image source: spectre-design.com

Examples: SHiP shield optimization

$$\text{background} = \mathop{\mathbb{E}}_{\text{event}} \mathbb{I}\left[\text{muon passed} > 0 \mid \text{event}\right] \to \min$$

- expensive computations:
  - each evaulation involves thousands of event simulations;
- only estimates are available:
  - function might have a nice gradient;
  - no MC estimation have a usefull gradient.

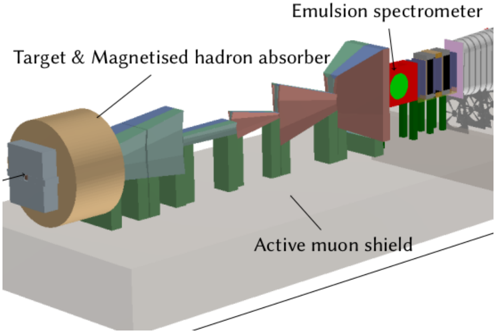


Image source: Oliver Lantwin. Bayesian optimisation

Examples: chess bot

$$\text{win rate} = \mathop{\mathbb{E}}_{\text{opponent}} \mathbb{I}\left[\text{win} \mid \text{opponent}\right] \to \max$$

- might not be expensive;
- only estimates are available:
  - function might have a nice gradient;
  - no MC estimation have a usefull gradient.

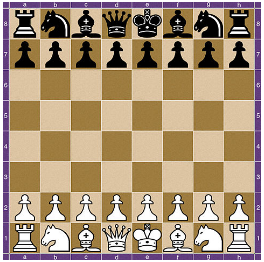


Image source: Wikimedia Commons.

Optimization methods categorization

black-box (zeroth order):

- **Bayesian Optimization**;
- **Variational Optimization**;
- evolutionary methods;
- Nelder–Mead, Powell, ...;

quasi-Newton:

- BFGS;

first order methods:

- SGD;
- adam;
- momentum;

second-order:

- Netwon's method;

* Lists of methods are by no means exhaustive.

Bayesian Optimization

Surrogate models

Let’s assume we have a function $f(x)$:

- expensive to evaluate;
- no gradient information.

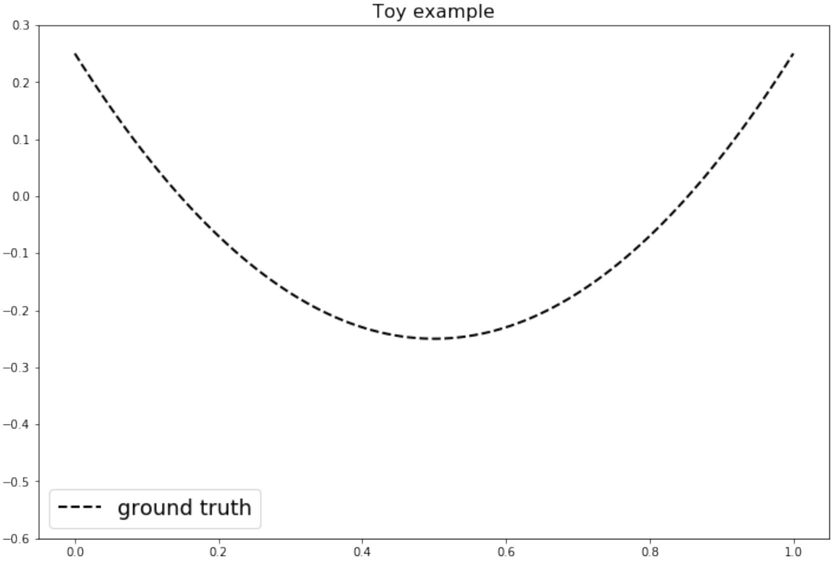
Example: detector optimization.

Bayesian optimization is primarely developed for computationally heavy objectives. Usually, gradient information for such target functions is absent as well. Nevertheless, there are some Bayesian optimization methods that make use of gradient information.

Toy example

$$\begin{aligned} f(x) &= 2\left(x - \frac{1}{2}\right)^2 - \frac{1}{4}; \\ x &\in [0, 1]. \end{aligned}$$

Toy example


Toy example
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
ground truth

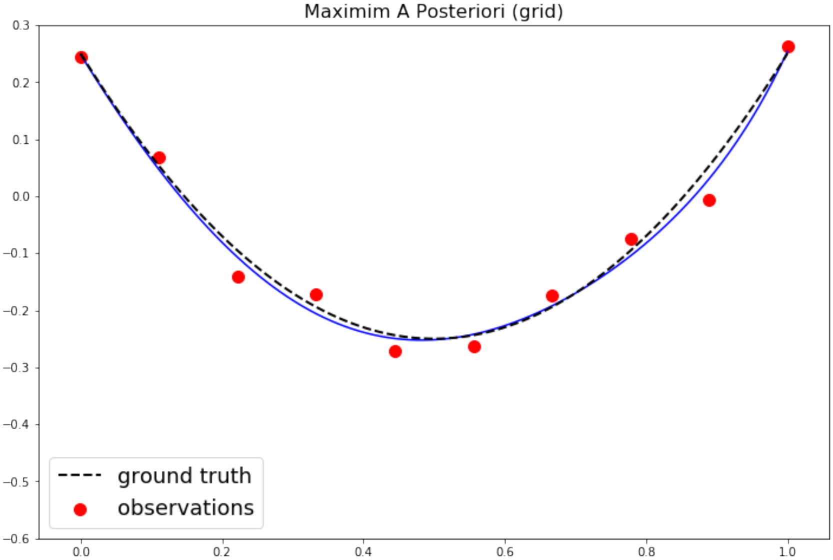
Surrogate model

- cover the whole parameter space (i.e. grid):
  - a.k.a experiment plan;
- evaluate target function in these points;
- introduce a regression model:
  - a.k.a. *surrogate model*;
  - preferably differentiable;
- fit model to these points;
- find minimum of the surrogate with a gradient method.

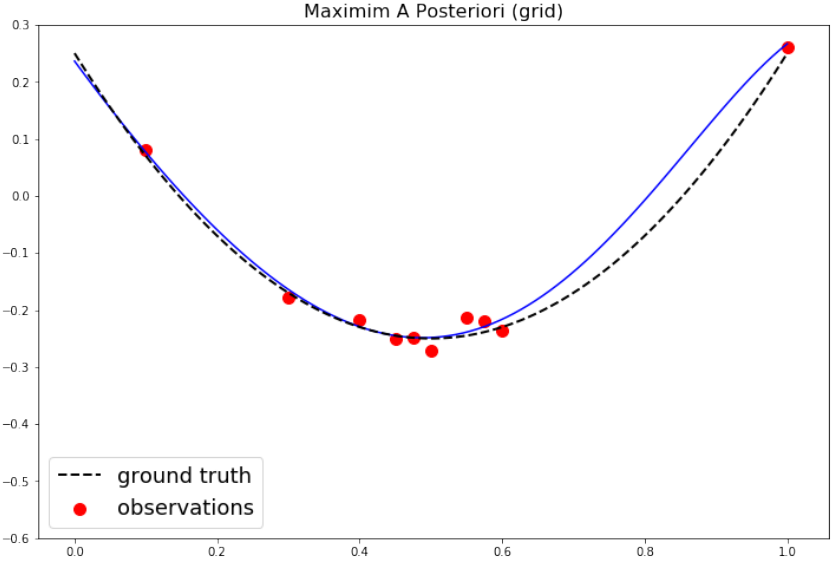
Toy example:

$$\mathrm{surrogate}(x) = w_0 + w_1 \cdot x + w_2 \cdot x^2 + \cdots + w_7 \cdot x^7$$

Surrogate model



Surrogate model



Discussion

- requires a lot of target function evaluations;
    - **to ensure surrogate model is not overfitted**;
- most of the points does not provide information about location of the minimum.

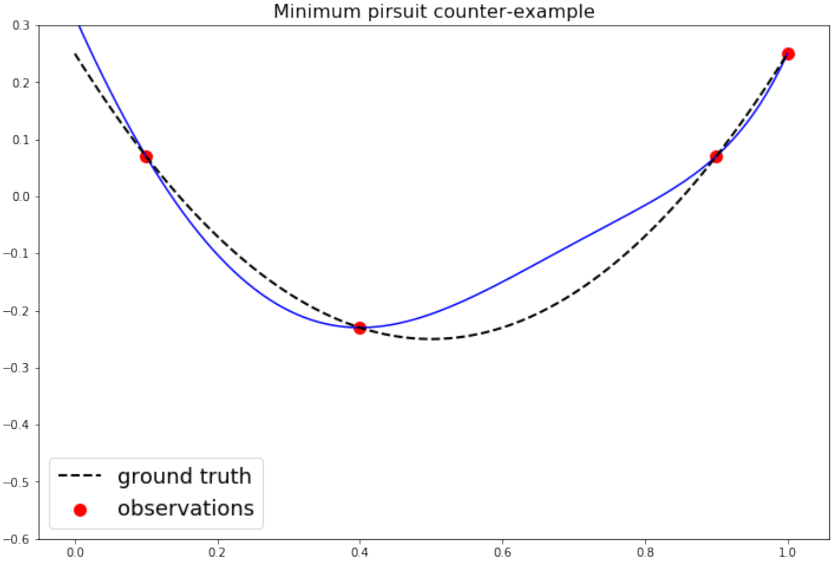
Can we update the surrogate model sequentially?

Follow surrogate minimim

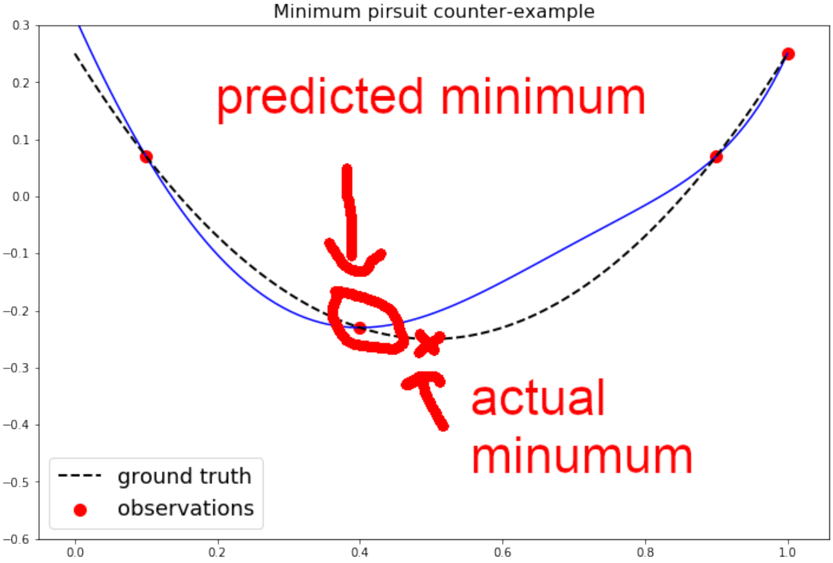
1. fit surrogate to observed values;
2. locate minimum of the surrogate;
3. evaluate target function in the predicted minimum;
4. repeat.

Sounds good, **does not work**!

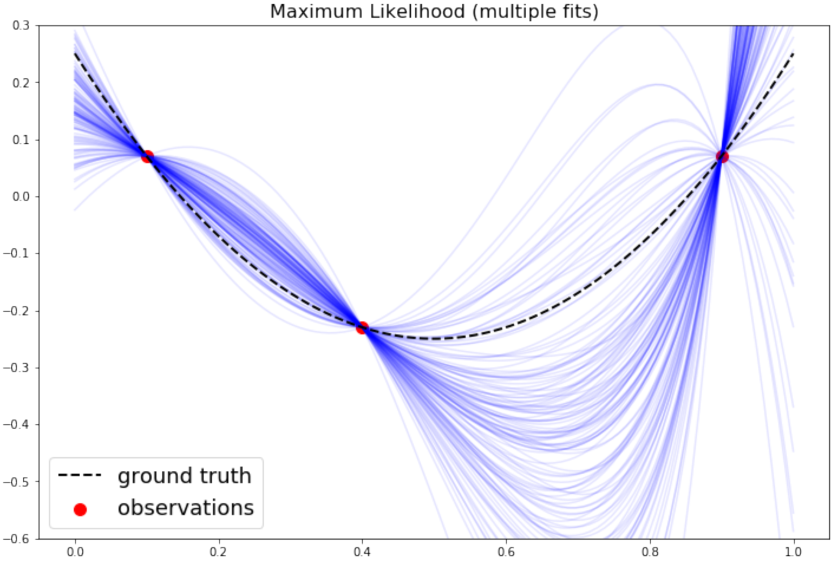
Does not follow surrogate minimim



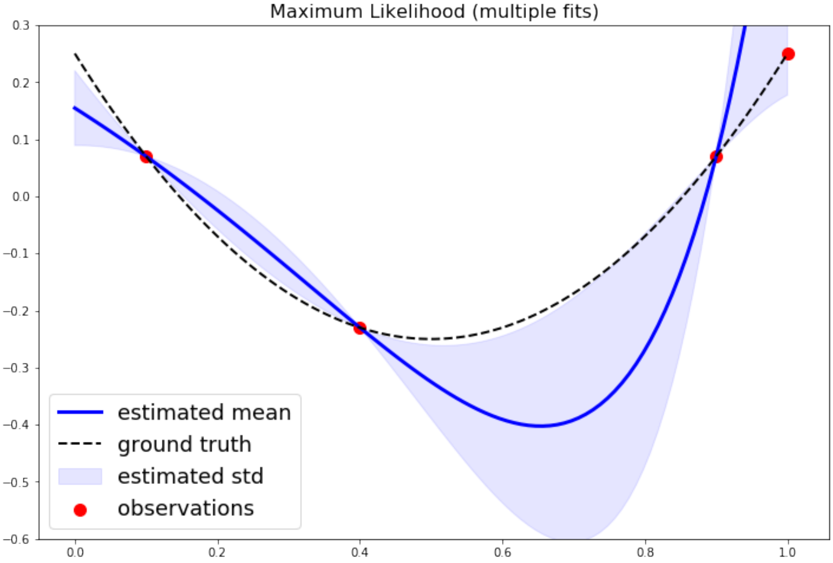
Does not follow surrogate minimim



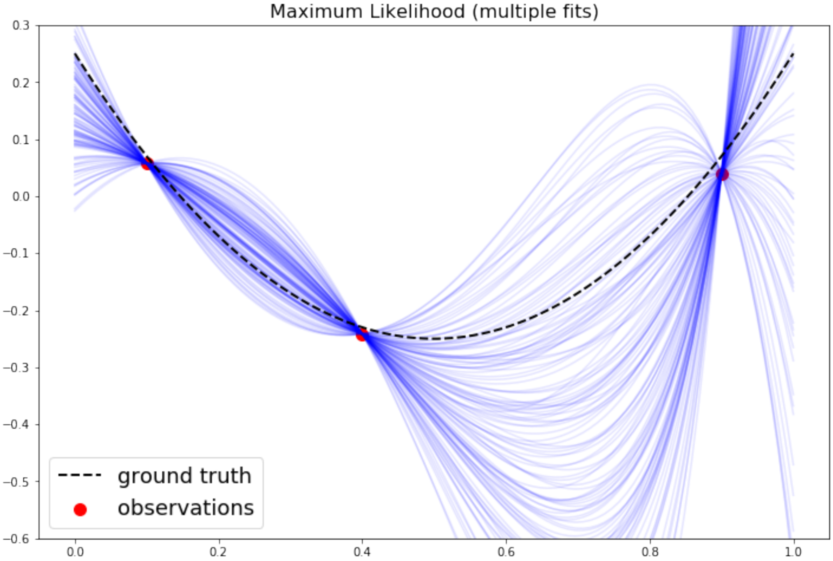
Multiple fits



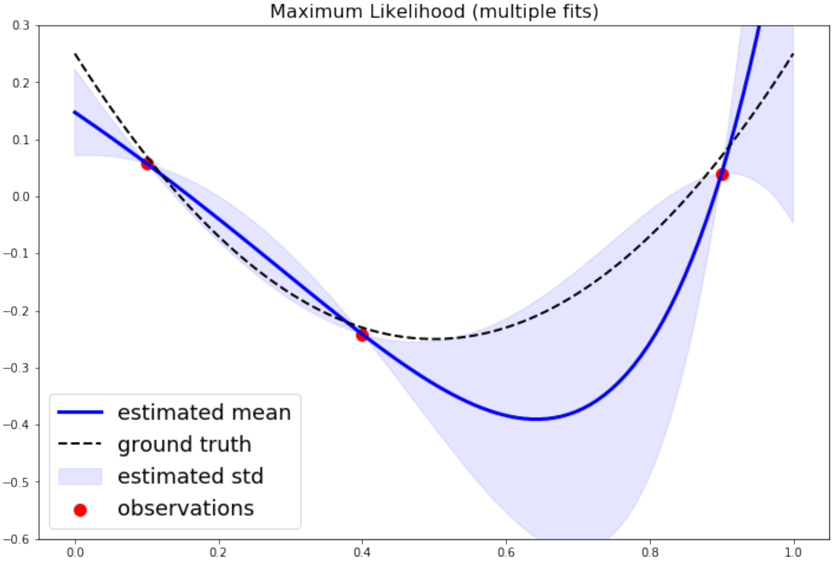
Multiple fits



Multiple fits (with noise)



Multiple fits (with noise)



Discussion

Surrogate models trained on few datapoints:

- the model overfits:
  - exist multiple models that fit data perfectly;
- regions with no points have unreliable predictions;
- multiple initial guesses result in an hardly predictable prior over model parameters.

Preliminaries: Maximum Likelihood estimates

Maximum Likelihood estimate θ_{ML} for parameters θ of a distribution family $P(x \mid \theta)$ given data $X = \{x_1, x_2, \ldots,_n\}$:

$$\begin{aligned} \theta_{\mathrm{ML}} &= \arg\max_{\theta} \mathcal{L}(\theta, X); \\ \mathcal{L}(\theta, X) &= P(x_1, x_2, \ldots, x_n \mid \theta); \end{aligned}$$

- $\mathcal{L}(\theta, X)$ - likelihood function.

Examples: mean of a normal distribution

Mean of a normally distributed variable:

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$

$$\mu_{\mathrm{ML}} = \arg\max_{\mu} \mathcal{L}(\theta, X) = \arg\max_{\mu} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x_i - \mu)^2\right] =$$

$$\arg\max_{\mu} \log\left\{\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(x_i - \mu)^2\right]\right\} = \arg\min_{\mu} \sum_{i=1}^{N}(x_i - \mu)^2 =$$

$$\frac{1}{N}\sum_{i=1}^{N} x_i$$

Examples: regression

- $y = f(x) + \varepsilon$ - observations;
- $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ - noise in observations;
- $y = g(x, \theta)$ - regression model with parameters θ.

$$\theta_{\mathrm{ML}} = \arg\max_\theta \mathcal{L}(\theta, X) = \arg\max_\theta \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(g(x_i, \theta) - y_i)^2\right] =$$

$$\arg\max_\theta \log\left\{\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(g(x_i, \theta) - y_i)^2\right]\right\} =$$

$$\arg\min_\theta \sum_{i=1}^{N}(g(x_i, \theta) - y_i)^2 = \arg\min_\theta \mathrm{MSE}(\theta, X, y)$$

Bayesian inference and uncertainty

Bayesian inference recovers the whole posterior distribution.

- prior over surrogate model parameters $P(\theta)$;
- data model: $P(y \mid x, \theta)$;
- posterior:

$$P(\theta \mid X, y) = \frac{P(y \mid X, \theta) P(\theta)}{P(y \mid X)} = \frac{P(y \mid X, \theta) P(\theta)}{\int_\theta P(y \mid X, \theta) P(\theta) \, d\theta} \propto P(y \mid X, \theta) P(\theta)$$

Maximum A Posteriori

- Baysian inference is not always computationally possible;
- instead **Maximum A Posteriori** or **MAP** estimation is often used:

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} P(\theta \mid X, y) = \arg\max_{\theta} P(y \mid X, \theta) P(\theta)$$

- Maximum Likelihood is a MAP estimate with a uniform prior $P(\theta) = \mathrm{const}$:

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} P(y \mid X, \theta) \cdot \mathrm{const} = \theta_{\mathrm{ML}}$$

Examples: regression with prior

- $y = f(x) + \varepsilon$ - observations;
- $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ - noise in observations;
- $y = g(x, \theta)$ - regression model with parameters θ;
- $P(\theta) = \alpha \exp\left(-2\alpha|\theta|\right)$ - prior over model parameters.

$$\theta_{\text{MAP}} = \arg\max_\theta \ \mathcal{L}(\theta, X) P(\theta) = \dots =$$

$$\arg\min_\theta \ \sum_{i=1}^{N} (g(x_i, \theta) - y_i)^2 + 2\alpha|\theta| = \arg\min_\theta \ \text{MSE}(\theta, X, y) + C \cdot |\theta|$$

- structural risk minimization is a MAP estimation.

Bayesian inference and uncertainty, toy example

$$\begin{aligned}
\mathrm{surrogate}(x) &= w_0 + w_1 \cdot x + w_2 x^2 + \cdots + w_7 \cdot x^7; \\
y &= f(x) + \varepsilon; \\
\varepsilon &\sim \mathcal{N}(0, \sigma_\varepsilon^2).
\end{aligned}$$

- prior: $P(\theta = (w_0, \ldots, w_7)) = \mathcal{N}(0, \Sigma_w)$, $\Sigma_w = \mathrm{diag}(\sigma_w, \ldots, \sigma_w)$;
- data model:

$$P(y \mid x, \theta) \propto \exp\left[-\frac{1}{2\sigma_\varepsilon^2}\left(\mathrm{surrogate}(x) - y\right)^2\right]$$

Bayesian inference and uncertainty, toy example

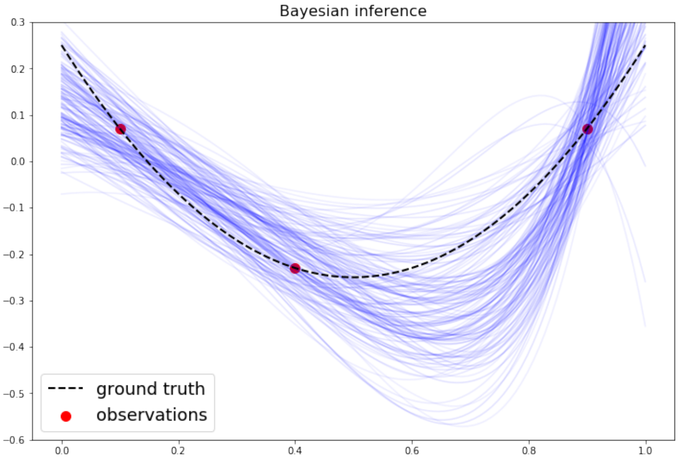
$$P(\theta \mid X, y) = \prod_i \frac{1}{\sqrt{(2\pi\sigma_\varepsilon^2)}} \exp\left[-\frac{(y_i - \text{surrogate}_\theta(x_i))^2}{2\sigma_\varepsilon^2}\right] \cdot \prod_j \frac{1}{\sqrt{(2\pi\sigma_w^2)}} \exp\left[-\frac{w_j^2}{2\sigma_w^2}\right]$$

$$- \log P(\theta \mid X, y) =$$

$$\sum_i \left[\text{const} + \frac{(y_i - \text{surrogate}_\theta(x_i))^2}{2\sigma_\varepsilon^2}\right] + \sum_j \left[\text{const} + \frac{w_j^2}{2\sigma_w^2}\right] \propto$$

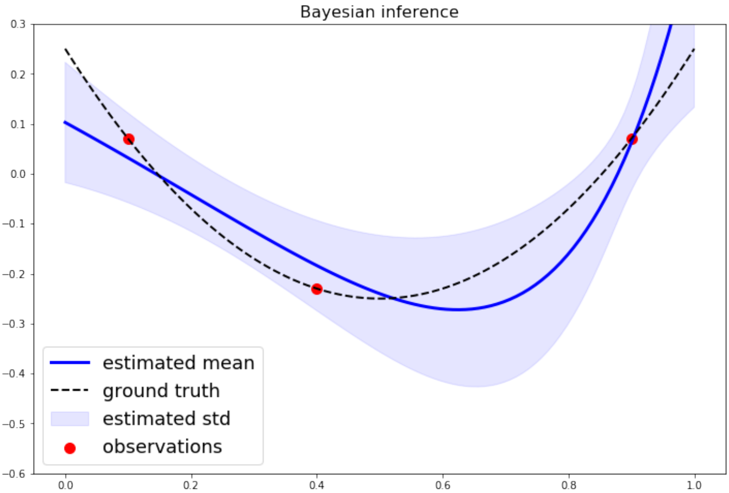
$$\text{const} + \sum_i (y_i - \text{surrogate}_\theta(x_i))^2 + C \cdot \|w\|^2$$

Bayesian inference and uncertainty



Transparency is proportional to the posterior of the model.

Bayesian inference and uncertainty



Bayesian inference and uncertainty

Naive Bayesian inference:

- sample θ_i from the prior $P(\theta)$;
- evaluate $P(\theta_i \mid X, y)$;
- compute $P(y = t \mid X) = \frac{1}{\sum_i P(\theta_i|X,y)} \sum_i P(y = t \mid X, \theta_i) P(\theta_i \mid X, y)$

The next step

How the next point for evaluation should be chosen?

- exploration:
  - decrease uncertanty of the estimations;
- exploitation:
  - using current estimates to locate the minimum;
- acqusition function:

$$\mathrm{candidate} = \arg\max_x \ \mathrm{acqusition}(x)$$

Acqusition function can be maximized by any standard optimization procedure, i.e. GD if differentiable.

Acqusition

- probability of improvement:

$$PI(x) = P(y > y^* \mid x)$$

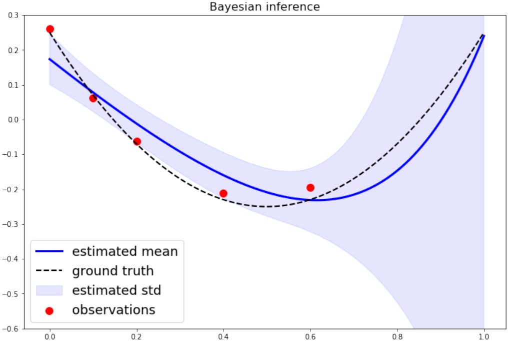
- expected improvement:

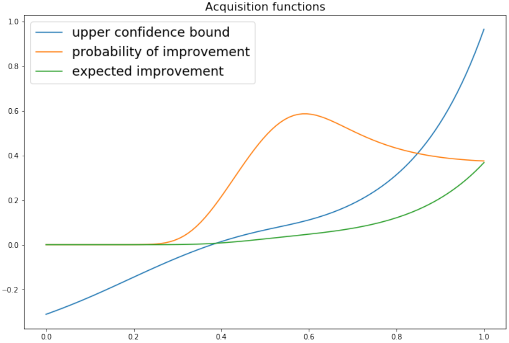
$$EI(x) = \mathbb{E}\left[\mathbb{I}[y > y^*] \mid x\right]$$

- upper confidence bound:

$$LCB(x) = \mathbb{E}\left[y \mid x\right] - \kappa \cdot \mathrm{Var}\left[y \mid x\right]$$

The next step





BO cycle

```
given $X^0$, $y^0$ — initial dataset;
for $t := 1, \dots$ do
    compute posterior: $P^t = P(\theta \mid X^{t-1}, y^{t-1})$
    compute candidate: $x' = \arg\max_x \mathbb{E}\left[\mathbb{I}[y > y^*] \mid x, P^t\right]$
    evaluate: $f^t = f(x')$;
    $X^t = [X^{t-1}, x']$
    $y^t = [X^{t-1}, f^t]$
end for
```

BO cycle

```
bo = BO(
  acqusition='EI',
  regressor=GaussianProcess(kernel=...)
)

for i in range(...):
  candidate = bo.ask()
  value = target_function(candidate)
  bo.tell(candidate, value)
```

The problem with naive Bayesian Inference

- the prior is usually selected to be wide-spread:
    - to ensure the actual dependency has relative high prior probability;
- most of the sampled models have low posterior probability:
    - wasting a lot of computational power.

Linear Models

- fixed basis functions: $\phi(x) \in \mathbb{R}^m$;
- vector of parameters: $w \in \mathbb{R}^m$:

$$\begin{aligned} y(x) &= w \cdot \phi(x) + \varepsilon; \\ \varepsilon &\sim \mathcal{N}(0, \sigma^2_\varepsilon); \\ \\ w &\sim \mathcal{N}(0, \Sigma_w); \end{aligned}$$

For simplicity the constant term is omitted in the derivations as it can be modeled by introducing a constant basis function: $\phi_0(x) = 1$. Nevertheless, additional care should be taken with priors for corresponding coefficient w_0.

Linear models

Theorem 1

If:

- prior $P(w)$ of parameters w of a linear model is normally distributed,
- noise is Gaussian with constant variance,

then posterior $P(w \mid X, y)$ is also normally distributed.

Linear models

Assume values $y \in \mathbb{R}^n$ were observed in points X (with $\Phi = \phi(X)$):

$$P(w \mid y, X) \propto P(y \mid w, X) P(w) \propto$$

$$\exp\left[-\frac{1}{2\sigma_\varepsilon^2}(y - \Phi^T w)^T (y - \Phi^T w)\right] \cdot \exp\left[-\frac{1}{2} w^T \Sigma_w^{-1} w\right] =$$

$$\exp\left[-\frac{1}{2}(w - w^*)^T A_w (w - w^*)\right]$$

where:

- $A_w = \frac{1}{\sigma_\varepsilon^2} \Phi \Phi^T + \Sigma_w^{-1}$;
- $w^* = \frac{1}{\sigma_\varepsilon^2} A_w^{-1} \Phi y$.

Linear models

Theorem 2

With normally distributed posterior $P(w \mid X, y)$ joint distribution $P(y_1, y_2, \ldots, y_n \mid X, y, x_1, \ldots, x_n)$ of any finite set of $(y_1, \ldots, y_n)$ is normal.

Linear models

To make prediction y' in point x':

$$P(y' \mid y, X, x') = \int P(y' \mid w, x') P(w \mid X, y) = \mathcal{N}\left(\frac{1}{\sigma_\varepsilon^2}\phi'^T A^{-1}\Phi y, \phi'^T A^{-1}\phi'\right)$$

- posterior distribution of model parameters is Gaussian;
- (posterior) joint distribution of any number of $y(x)$ is a Gaussian distribution.

Kernels

$$P(y' \mid y, X, x') = \mathcal{N}\left(\frac{1}{\sigma_\varepsilon^2}\underbrace{\phi'^T A^{-1}\Phi}_{\text{dot products}}\, y, \underbrace{\phi'^T A^{-1}\phi'}_{\text{dot product}}\right)$$

- basis function $\phi(x)$ occurs only in scalar product;
- kernel trick can be used:

$$\phi^T(x)\Sigma_w\phi(x') \to k(x, x')$$

Gaussian process

In general Gaussian process $f(x)$ is defined by:

- $m(x) = \mathbb{E}\, f(x)$ - mean function;
- $k(x, x') = \mathbb{E}\,(f(x) - m(x))(f(x') - m(x'))$.

Interpretation:

- a set of random variables $y(x)$;
- each $y(x)$ is normally distributed;
- each subset $\{y(x_1), y(x_2), \dots, y(x_m)\}$ has normal joint distribution:
    - with covariance matrix defined be the kernel $\mathrm{cov}(y(x_1), y(x_2)) = k(x_1, x_2)$.

Gaussian process

Bayesian linear model:

- $m(x) = \mathbb{E}\, w \cdot \phi(x) = \phi(x)\mathbb{E}w = 0$;
- $k(x, x') = \phi^T(x)\mathbb{E}\left[ww^T\right]\phi(x') = \phi^T(x)\Sigma_w\phi(x')$.

Usually:

- $m(x) \equiv 0$;
- $k(x, x') = \exp(-\frac{1}{2}\|x - x'\|^2)$ - RBF kernel;
- $k(x, x') =$ crazy expression - Matern kernel;
- $k(x, x') = \mathbb{I}\left[x = x'\right]$ - white kernel (equivalent to Gaussian noise).

Every proper kernel corresponds to a set basis funstions, i.e. every kernel defines a scalar product in a corresponding *Reproducing kernel Hilbert space* or RKHS. Thus, Gaussian processes defined by kernels are equivalent to Bayesian liner regressions in corresponding RKHS.

Predictions with kernel

- dataset:

$$
\begin{aligned}
X &= [x_1, x_2, \ldots, x_n]\,; \\
y &= [y_1, y_2, \ldots, y_n]
\end{aligned}
$$

- points to evaluate: X^*, y^*

Predictions with kernel

$$\begin{bmatrix} y \\ y' \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right)$$

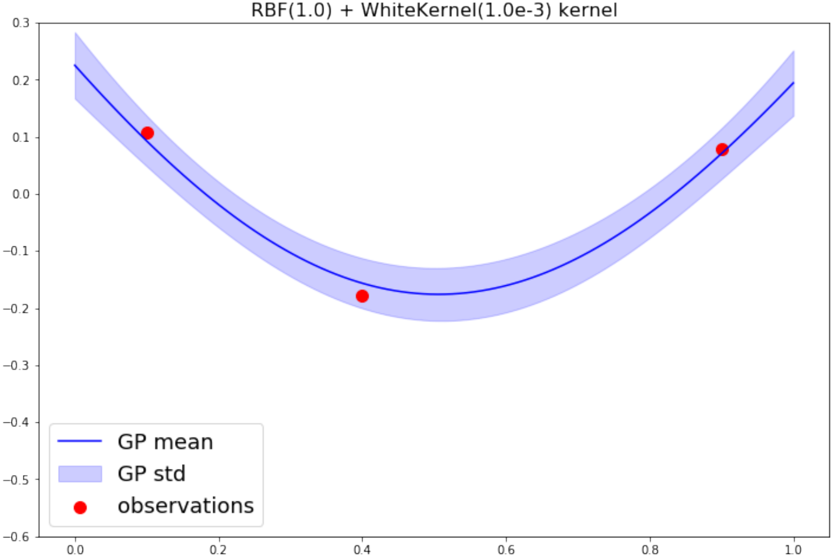
- where:

$$K(X,X) = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix}$$

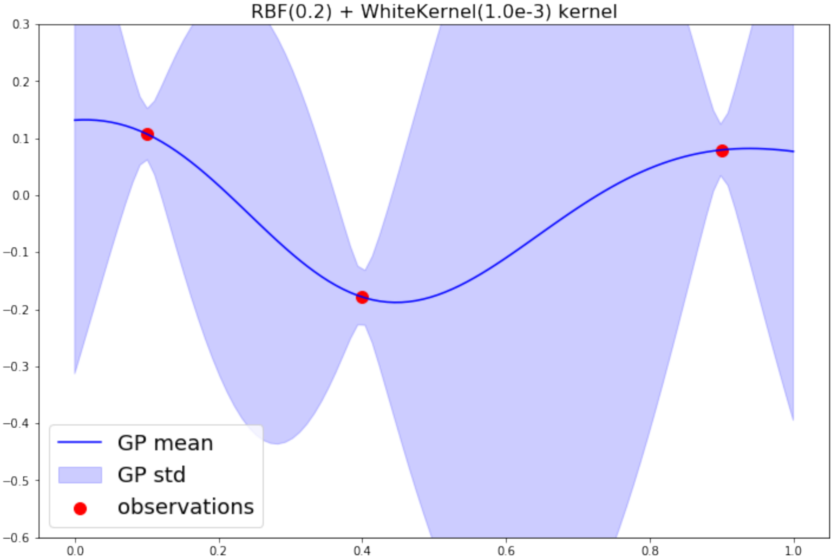
Predictions with kernel

$$\begin{aligned}
y' \mid X, y, X' &\sim \mathcal{N}(\mu', \sigma'); \\
\mu' &= K(X', X)\left[K(X, X) + \sigma_\varepsilon^2\right]^{-1} y; \\
\sigma' &= K(X', X') - K(X', X)\left[K(X, X) + \sigma_\varepsilon^2\right]^{-1} K(X, X').
\end{aligned}$$

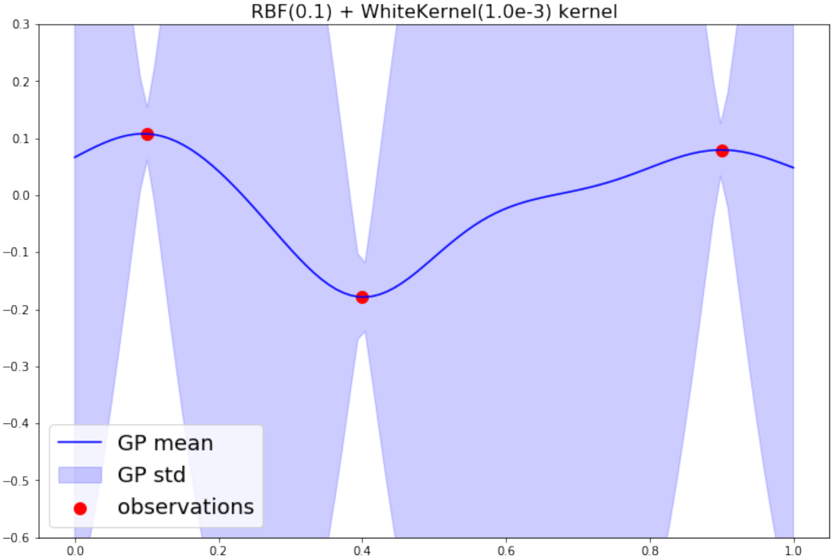
Examples


RBF(1.0) + WhiteKernel(1.0e-3) kernel
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
GP mean
GP std
observations

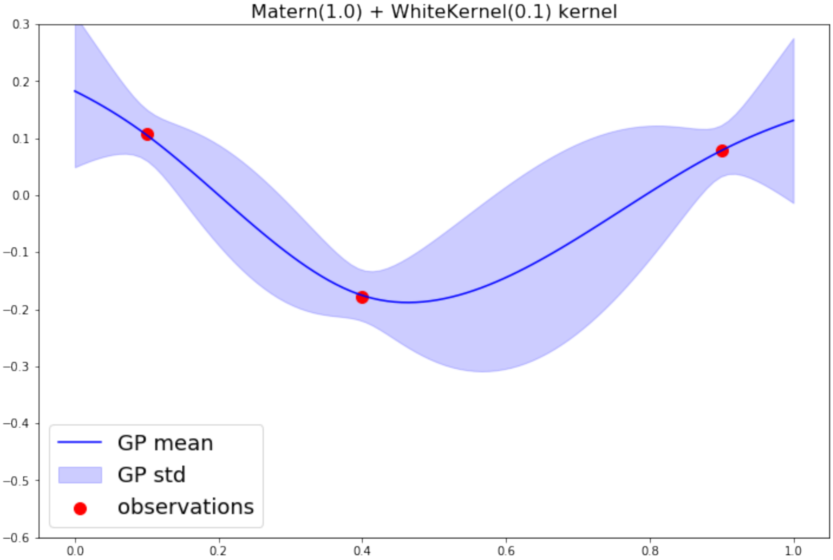
Examples


RBF(0.2) + WhiteKernel(1.0e-3) kernel
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
GP mean
GP std
observations

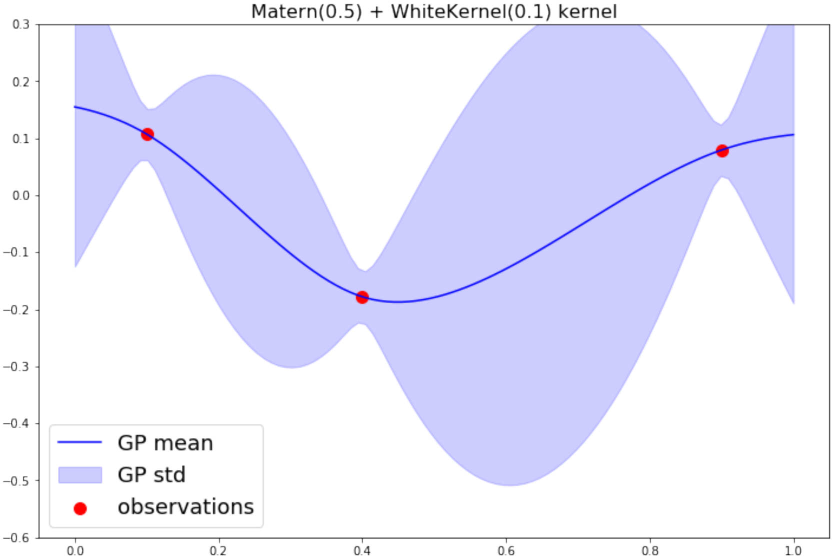
Examples


RBF(0.1) + WhiteKernel(1.0e-3) kernel
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
GP mean
GP std
observations

Examples


Matern(1.0) + WhiteKernel(0.1) kernel
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
GP mean
GP std
observations

Examples


Matern(0.5) + WhiteKernel(0.1) kernel
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
GP mean
GP std
observations

Hyper-parameter optimization for GP

How kernel parameters should be chosen?

- prior knowledge about the problem:
    - e.g. noise level of the target function;
- Maximum Likelihood fit:
    - setting acceptable ranges for hyper-parameters;
    - employing Gradient Descent;

Yes, it is hyper-parameter optimization for the model that optimizes hyper-parameters of another model.

Gaussian Process

- full Baysian inference:
  - guaranteed convergence with a proper kernel and a proper acquisition function;
- computational compexity scales **cubically** with the number of training points.

Usage

Low-mid (≤ 20) dimensionality problems with few training points (≤ 100).

Alternatives to GP

Random Forest:

- uncertanty estimation in the leaves:
- much faster to fit:
  - good for large datasets (100+ points);
- non-differentiable:
  - much slower to find acqusition minimum.
- not a true Bayesian inference:
  - but quite close due to Random Forest properties.

Usage

Low dimensionality problems with a lot of training points (100+).

Alternatives to GP

A variant of Neural Networks for BO:

- a regression Neural Net is trained to fit the data (MAP estimation);
- the last layer is replaced by a Bayesian linear regression (no basis expansion);
- dimensionality reduction allows to handle large dimesionality problems.
- partial Bayesian inference might backfire.

Usage

You are lucky and the problem is high dimensional with internal structure, a lot of training points (100+).

Snoek J et al. Scalable bayesian optimization using deep neural networks.

Variational Optimization

Variational bound

Variational Optimization replaces problem:

$$f(\theta) \to_\theta \min;$$

with:

$$J(\psi) = \mathop{\mathbb{E}}_{\theta \sim P(\cdot|\psi)} f(\theta) \to_\psi \min$$

where:

- $J(\psi)$ - variational bound;
- $P(\cdot \mid \psi)$ - search distribution.

This variational bound is not the only one, nevertheless, it is the most common one in Varitional Optimization.

Properties

$$J(\psi) = \mathop{\mathbb{E}}_{\theta \sim P(\cdot \mid \psi)} f(\theta) \to_{\psi} \min$$

- upper bound:

$$\forall \psi : J(\psi) \geq \min f(\theta);$$

- if $P(\cdot \mid \psi)$ is allowed to (nearly) collapse into delta function, then

$$P(\cdot \mid \psi^{*}) \approx \delta(\theta^{*}).$$

Gradient of the variational bound

$$\frac{\partial}{\partial \psi} J(\psi) = \frac{\partial}{\partial \psi} \mathop{\mathbb{E}}_{\theta \sim P(\cdot|\psi)} f(\theta) = \frac{\partial}{\partial \psi} \int_\theta d\theta\ f(\theta) P(\theta \mid \psi) =$$

$$\int_\theta d\theta\ f(\theta)\ \frac{\partial}{\partial \psi} P(\theta \mid \psi) = \int_\theta d\theta\ f(\theta)\ P(\theta \mid \psi)\ \frac{\partial}{\partial \psi} \log P(\theta \mid \psi) =$$

$$\mathop{\mathbb{E}}_{\theta \sim P(\cdot|\psi)} f(\theta) \frac{\partial}{\partial \psi} \log P(\theta \mid \psi)$$

Gradient of the variational bound

$$\boxed{\nabla_\psi J(\psi) = \mathop{\mathbb{E}}_{\theta \sim P(\cdot|\psi)} f(\theta)\ \nabla_\psi \log P(\theta \mid \psi)}$$

$\nabla_\psi J(\psi)$ does not depend on $\nabla_\theta f(\theta)$

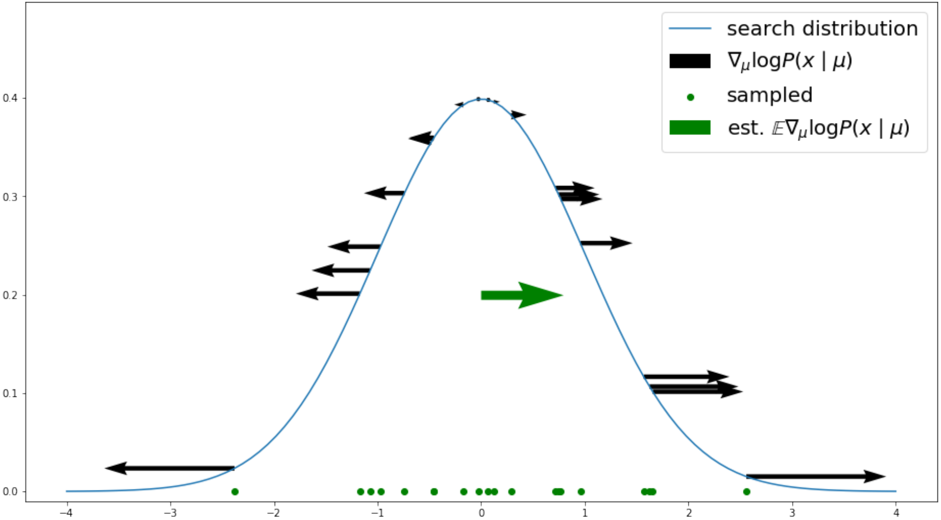
VO for Machine Learning loss functions

Typical loss function:

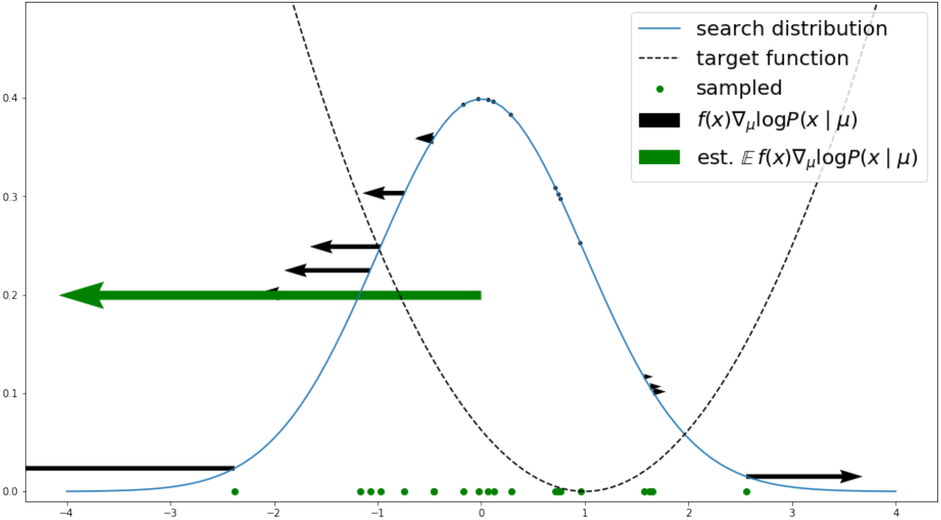
$$\begin{aligned} f(\theta) &= \mathop{\mathbb{E}}_{x} l(\theta, x); \\ \nabla_\psi J(\psi) &= \mathop{\mathbb{E}}_{\theta \sim P(\cdot|\psi)} \mathop{\mathbb{E}}_{x} l(\theta, x)\ \nabla_\psi \log P(\theta \mid \psi); \end{aligned}$$

- batch estimation of $\nabla_\psi J(\psi)$ is unbiased;
- can be used with SGD.

Gradient of the variational bound



Gradient of the variational bound



SGD-VO

```
initialize $P(\cdot \mid \psi)$
while not converged do
    sample $\theta$ from $P(\cdot \mid \psi)$;
    $\nabla_\psi J(\psi) \leftarrow f(\theta) \nabla_\psi \log P(\theta \mid \psi)$;
    $\psi \leftarrow \psi - \gamma \nabla_\psi J(\psi)$;
end while
```

Variational Optimization

- allows usage of stochastic gradient methods for black-box problems:
    - VO is much slower in contrast to using analytical gradient;
- search distribution is chosen to be simple:
    - e.g. normal distribution;

Usage

Low dimensionality problems with a cheap target function or cheap **unbiased** estimates.

Summary

Summary

Bayesian Optimization:

- efficient for heavy objectives;
- wide model selection:
  - precise Gaussian Processes;
  - approximations for ensamblies and Neural Networks.

Variational Optimization:

- cheap updates;
- efficient for cheap objectives.

References and links

Gaussian Processes:

- Bishop C. Pattern Recognition And Machine Learning.
- Snelson E. Tutorial: Gaussian process models for machine learning.

Bayesian Optimization:

- Wang Z, Zoghi M, Hutter F, Matheson D, De Freitas N. Bayesian Optimization in High Dimensions via Random Embeddings.
- Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Patwary M, Prabhat M, Adams R. Scalable bayesian optimization using deep neural networks.

References and links

Variational Optimization:

- Wierstra D, Schaul T, Glasmachers T, Sun Y, Peters J, Schmidhuber J. Natural evolution strategies.

Backup

Basis expansion

$$\begin{aligned} x &\in \mathbb{R}^n; \\ \phi(x) &= \{\phi_1(x), \phi_2(x), \dots, \phi_m(x)\}; \\ f(x) &= w \cdot \phi(x). \end{aligned}$$

- $f(x)$ - is still a linear model (w.r.t w);
- $\phi(x)$ is called feature map;
- can produce a rich models, e.g.:
    - polynomials: $\phi_k(x) = x^k$;
    - Fourier series: $\phi_k(x) = \cos(kx)$.
- size of the basis can grows very quickly with x dimensionality:
    - $O(n^2)$ for polynomials;

Basis expansion