Bayesian optimisation

Gilles Louppe
ATLAS ML workshop

March 30, 2016
Problem statement

\[ x^* = \arg \max_x f(x) \]

Constraints:

- \( f \) is a black box for which no closed form is known;
  - gradients \( \frac{df}{dx} \) are not available.
- \( f \) is expensive to evaluate;
- (optional) uncertainty on observations \( y_i \) of \( f \)
  - e.g., \( y_i = f(x_i) + \epsilon_i \) because of Poisson fluctuations.

Goal: find \( x^* \), while minimizing the number of evaluations \( f(x) \).
If you do not have these constraints, there is certainly a better optimisation algorithm than Bayesian optimisation.

(e.g., L-BFGS-B, Powell’s method (as in Minuit), etc)
Bayesian optimisation

for $t = 1 : T$,

1. Given observations $(x_i, y_i)$ for $i = 1 : t$, build a probabilistic model for the objective $f$.
   - Integrate out all possible true functions, using Gaussian process regression.

2. Optimise a cheap utility function $u$ based on the posterior distribution for sampling the next point.

   \[ x_{t+1} = \arg \max_x u(x) \]

   Exploit uncertainty to balance exploration against exploitation.

3. Sample the next observation $y_{t+1}$ at $x_{t+1}$. 
Where shall we sample next?
Build a probabilistic model for the objective function

This gives a posterior distribution over functions that could have generated the observed data.
Acquisition functions

Acquisition functions $u(x)$ specify which sample $x$ should be tried next:

- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa \sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where $x_t^+$ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., $\kappa$) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)
Plugging everything together \((t = 0)\)

\[
x_t^+ = 0.1000
\]

\[
x_{t+1} = \arg\max_x UCB(x)
\]
... and repeat until convergence ($t = 1$)
... and repeat until convergence ($t = 2$)
... and repeat until convergence \((t = 3)\)
... and repeat until convergence \((t = 4)\)
... and repeat until convergence ($t = 5$)
Limitations

• Bayesian optimisation has parameters itself!
  ■ Choice of the acquisition function
  ■ Choice of the kernel (i.e. design of the prior)
  ■ Parameter wrapping
  ■ Initialization scheme

• Gaussian processes usually do not scale well to many observations and to high-dimensional data.
  ■ Sequential model-based optimization provides a direct and effective alternative (i.e., replace GPs by a tree-based model).
Applications

- Bayesian optimization has been used in many scientific fields, including robotics, machine learning or life sciences.

- Use cases for high energy physics?
  - Optimisation of simulation parameters in event generators;
  - Optimisation of compiler flags to maximize execution speed;
  - Optimisation of hyper-parameters in machine learning for HEP;
  - ... let’s discuss further ideas?
Software

- **Python**
  - Spearmint [https://github.com/JasperSnoek/spearmint](https://github.com/JasperSnoek/spearmint)
  - RoBO [https://github.com/automl/RoBO](https://github.com/automl/RoBO)
  - scikit-optimize [https://github.com/MechCoder/scikit-optimize](https://github.com/MechCoder/scikit-optimize) (work in progress)

- **C++**
  - MOE [https://github.com/yelp/MOE](https://github.com/yelp/MOE)

Check also this [Github](https://github.com) repo for a vanilla implementation reproducing these slides.
Summary

- Bayesian optimisation provides a principled approach for optimising an expensive function $f$;

- Often very effective, provided it is itself properly configured;

- Hot topic in machine learning research. Expect quick improvements!