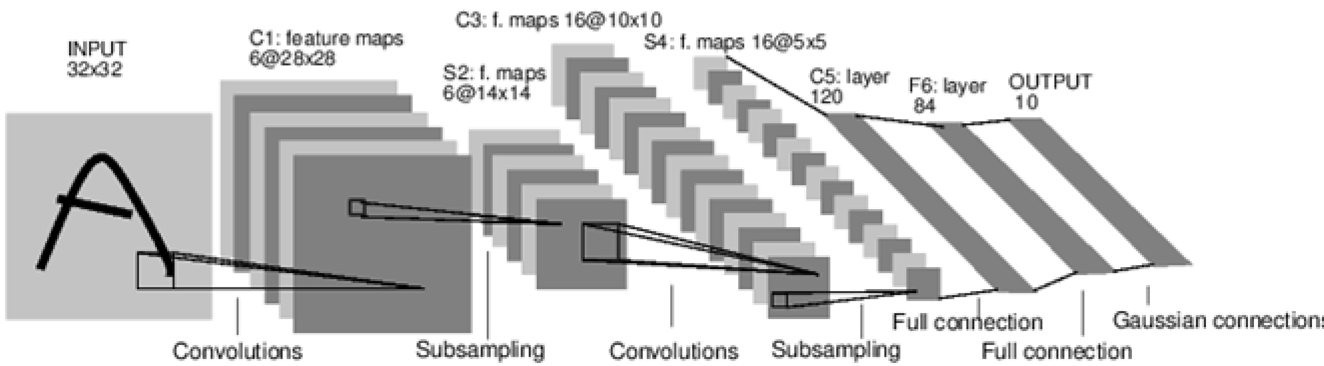
Bayesian Optimization

Brian Trippe & Paweł Budzianowski, MLG, 2016

Example - Picking Hyperparameters

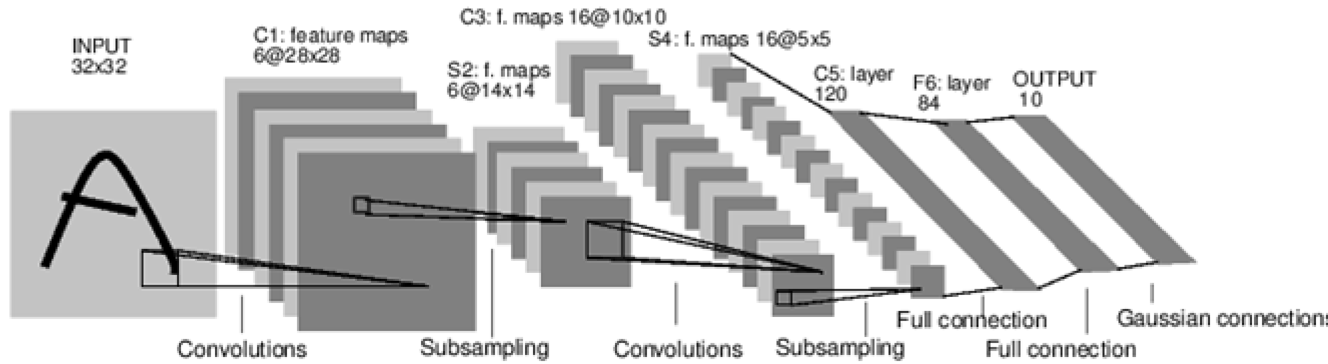
Designing Deep Neural Networks



A Full Convolutional Neural Network (LeNet)

Example - Picking Hyperparameters

Designing Deep Neural Networks



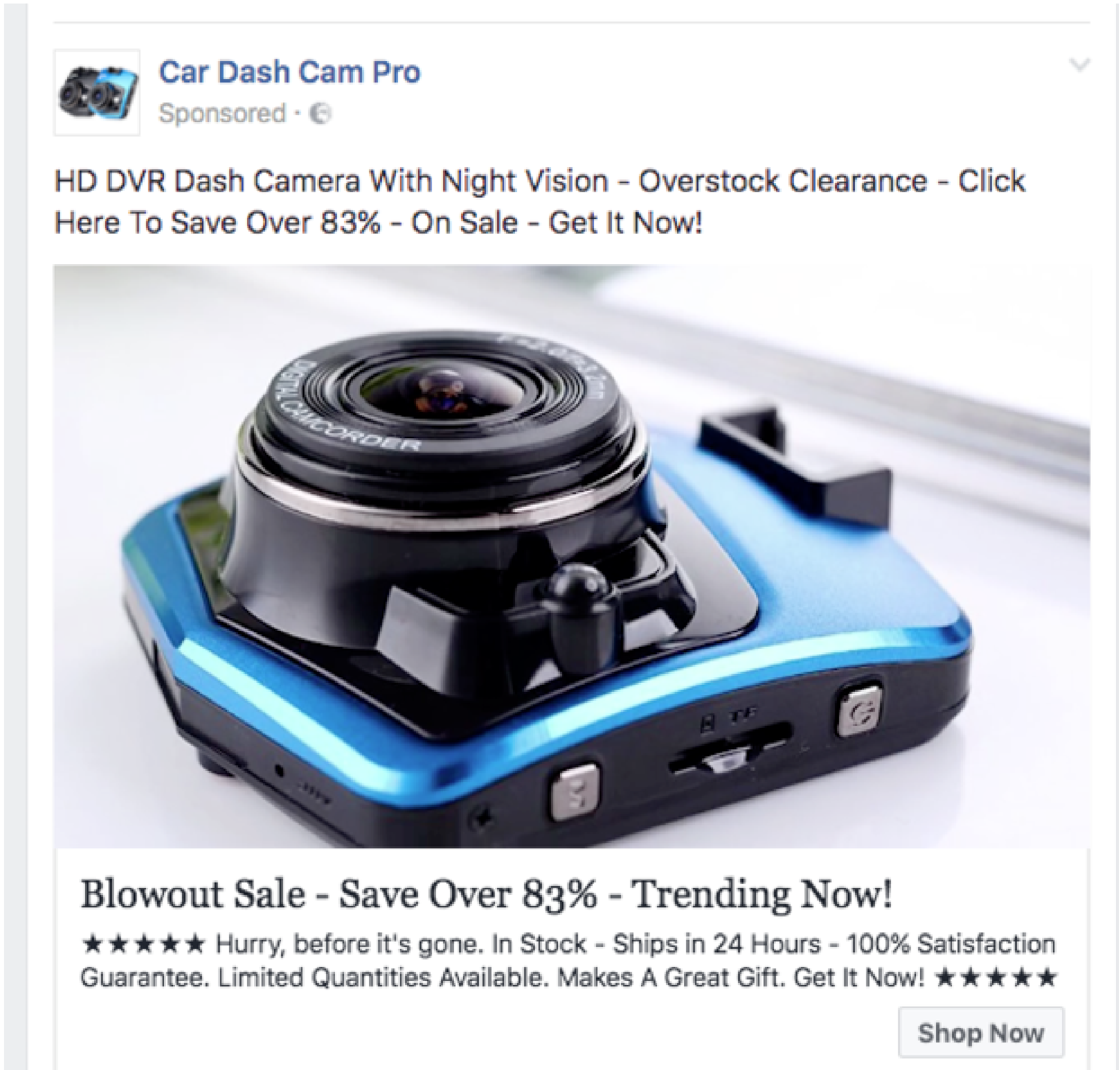
A Full Convolutional Neural Network (LeNet)

Validation accuracy is a function of hyperparamters

- learning rate, regularizations
- batch size, number/size of layers, convolution sizes
- Nonlinearity, Loss function

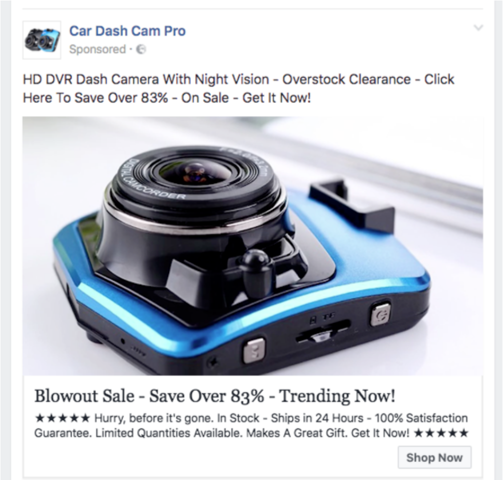
Example - A/B Testing

Tuning features of an Advertisement



Example - A/B Testing

Tuning features of an Advertisement

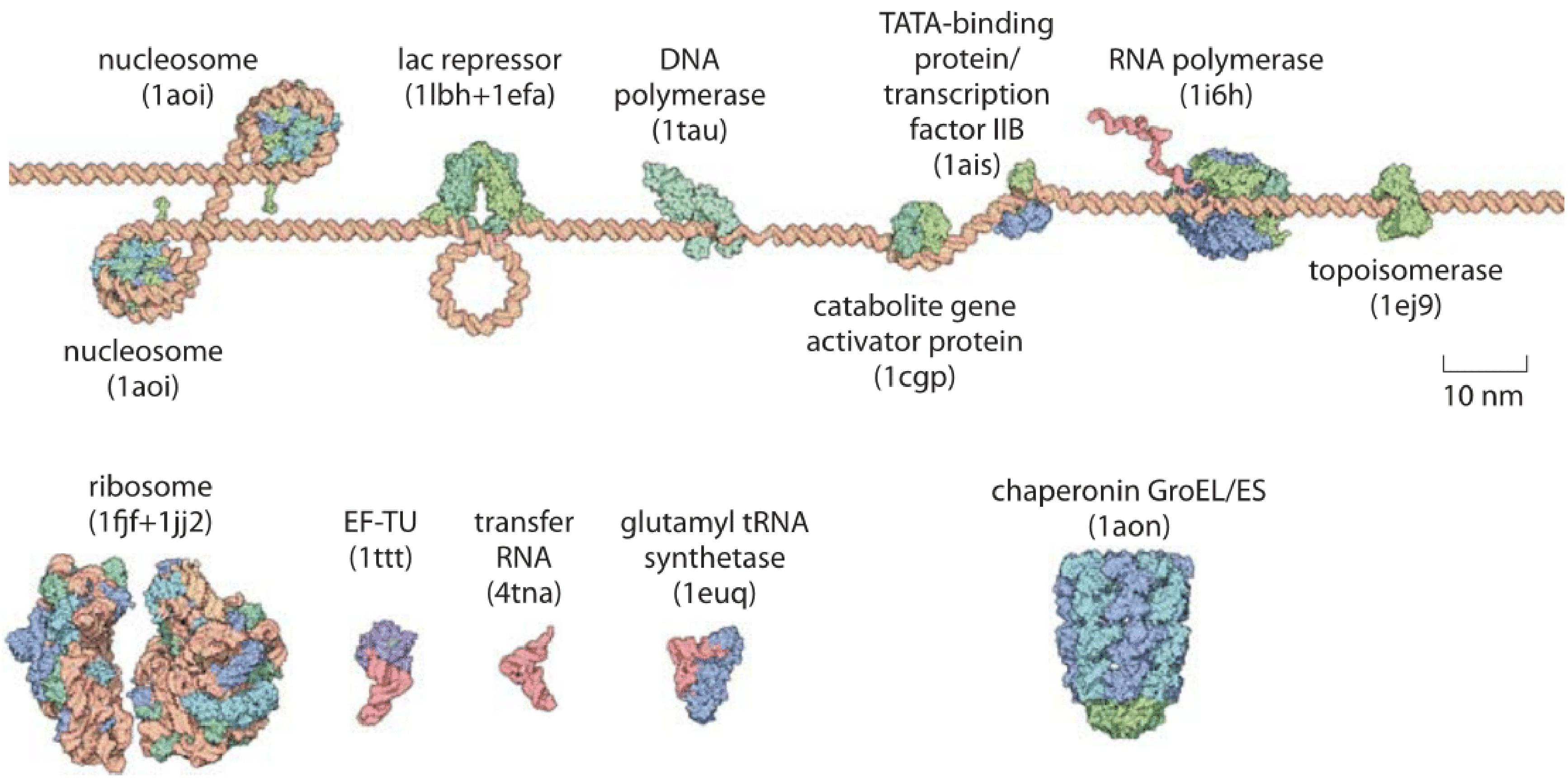


Click rates are a function of Advertiser's choices.

- Target Consumer Demographics and Interests
- Characteristics of what is displayed

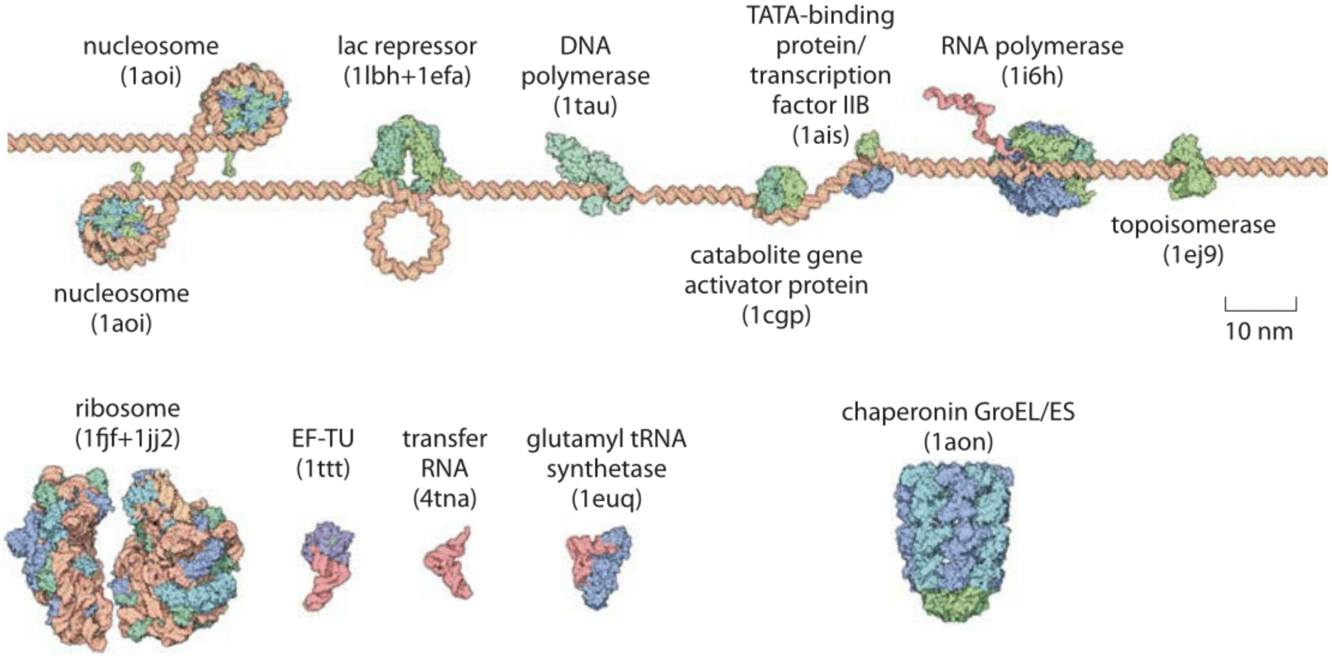
Example - Gene design

Choosing Regulatory Sequences for Genetic Engineering



Example - Gene design

Choosing Regulatory Sequences for Genetic Engineering



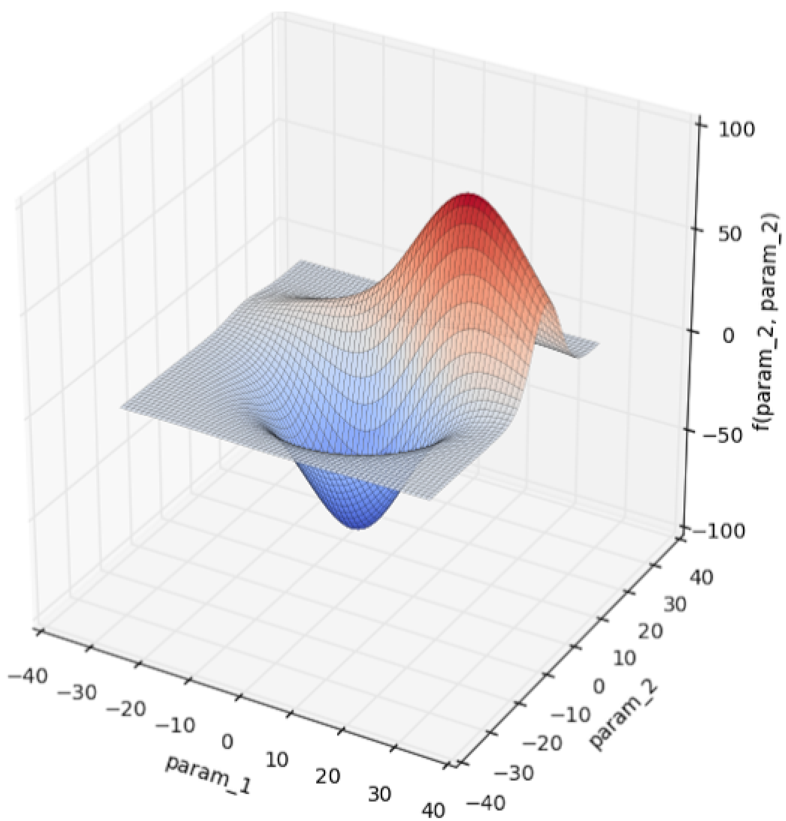
Transcription rate is a function of DNA sequence

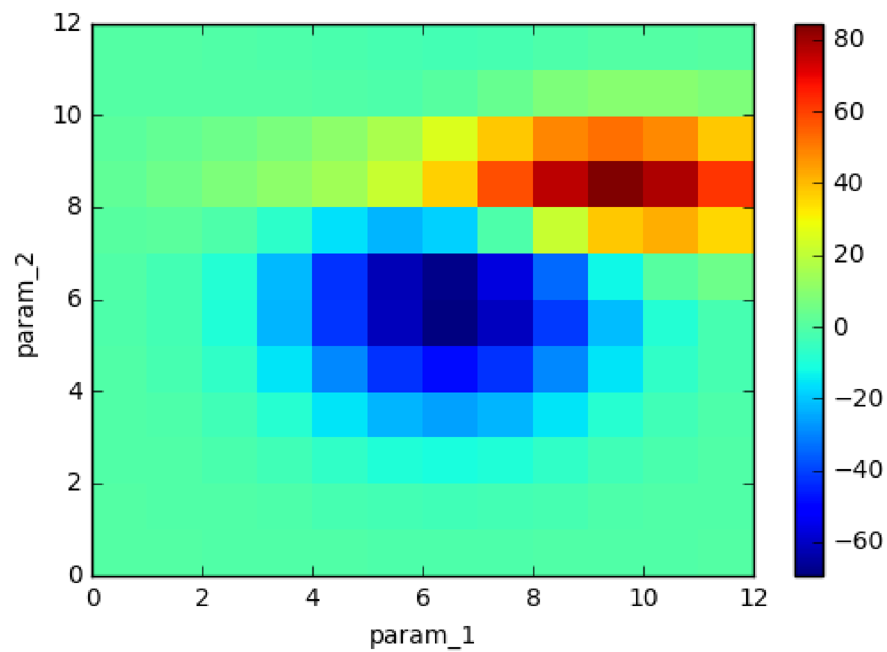
- regulatory motifs
- stability signals

González, Javier, et al. "Bayesian optimization for synthetic gene design." arXiv preprint arXiv:1505.01627 (2015).

How do we perform optimization?

We can use grid-search





How do we perform optimization?

Why does grid-search fail?

- How do we pick the interval?

How do we perform optimization?

Why does grid-search fail?

- How do we pick the interval?
- What if we are in a high dimensional space?
  - The number of grid points will increase exponentially with the number of dimensions, $O(c^n)$
  - Maximum distance between points increases as $\sqrt{D}$, so points need to be closer together.
  - $O(c^n * 2^{\sqrt{n}})$

How do we perform optimization?

Why does grid-search fail?

- How do we pick the interval?
- What if we are in a high dimensional space?
  - The number of grid points will increase exponentially with the number of dimensions, $O(c^n)$
  - Maximum distance between points increases as $\sqrt{D}$, so points need to be closer together.
  - $O(c^n * 2^{\sqrt{n}})$
- What if there is noise?

How do we perform optimization?

By being Bayesian!

How do we perform optimization?

By being Bayesian!

- Propose p(f)
- We can work with uncertainty and noise
- Make smarter choices about next points to query
- Use information from other problems to inform priors
- Hopefully, we get a good idea of the space with far less than an exponential number of points

What is Bayesian optimization?

What is Bayesian optimization?

The goal of Bayesian optimization is to solve optimization problems of the form:

$$\mathbf{x}_* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

What is Bayesian optimization?

The goal of Bayesian optimization is to solve optimization problems of the form:

$$\mathbf{x}_* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

where f is often a **nonlinear** and **non-convex** function over a domain $\mathcal{X}$.

What is Bayesian optimization?

The goal of Bayesian optimization is to solve optimization problems of the form:

$$\mathbf{x}_* = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

where f is often a **nonlinear** and **non-convex** function over a domain $\mathcal{X}$.

Assume that f is a **black-box** function - we can query it at any point but its derivatives are unavailable.

General algorithm

General algorithm

Algorithm 1 Bayesian optimization

Input: a black-box function f

1: **for** $n = 1, \ldots, N$ **do**

2: select $\mathbf{x}_n = \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha_{n-1}(\mathbf{x})$

3: query f at $\mathbf{x}_n$ to obtain y_n

4: augment data $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{(\mathbf{x}_n, y_n)\}$

return $\widetilde{\mathbf{x}}_N = \arg\max_{\mathbf{x} \in \mathcal{X}} \mu_N(\mathbf{x})$

Two ingredients

Two ingredients

1. Probabilistic framework

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

Two ingredients

1. Probabilistic framework

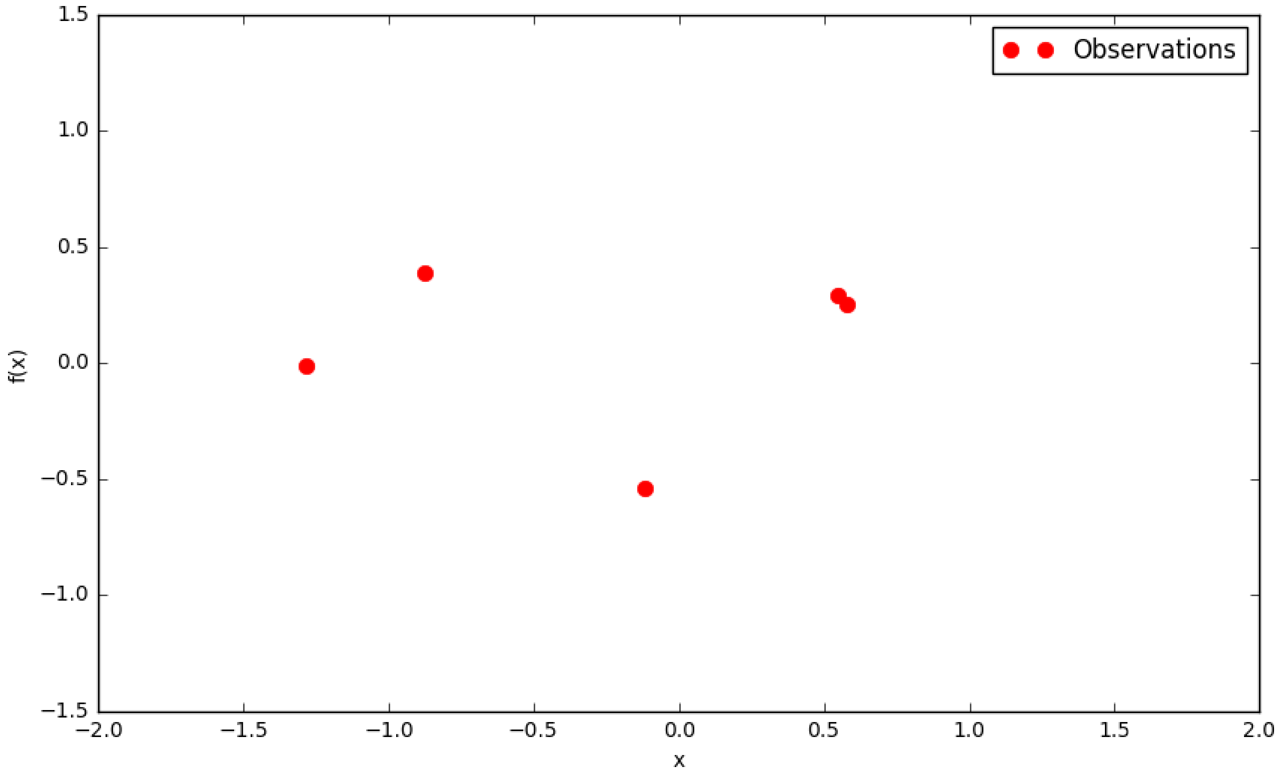
$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

2. Acquisition function

$$U : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}$$

$$\alpha(\mathbf{x};\ \mathcal{D}_n) = \mathbb{E}_{f(\mathbf{x})|\mathbf{x}}[U(\mathbf{x}, f(\mathbf{x}))]$$

A Concrete Example



- How do we pick the next point?
- We need a prior over functions

Gaussian Processes

The Gausian process provides a distribution over functions, $f : \mathbb{X} \to \mathbb{R}$

$$f \sim GP(\mu_0, k)$$

for $\mathbf{x} := (x_1, x_2, ..., x_n)$, $\mathbf{f} = (f(x_1), f(x_2), ..., f(x_n))$

$$\mathbf{f}|\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$$

where $m_i = \mu_0(x_i)$ and $K_{i,j} = k(x_i, x_j)$

Often we have noisy observations $\mathbf{y} := (y_1, y_2, ..., y_n)$

$$y|f, \sigma^2 \sim \mathcal{N}(f, \sigma^2\mathbf{I})$$

Gaussian Processes

The posterior predictive distribution at an unseen point, x_{n+1} is Gaussian.

$$y_{n+1} \sim \mathcal{N}(\mu_n(x_{n+1}), \sigma_n(x_{n+1})^2)$$

The predictive mean and variances have a closed form.

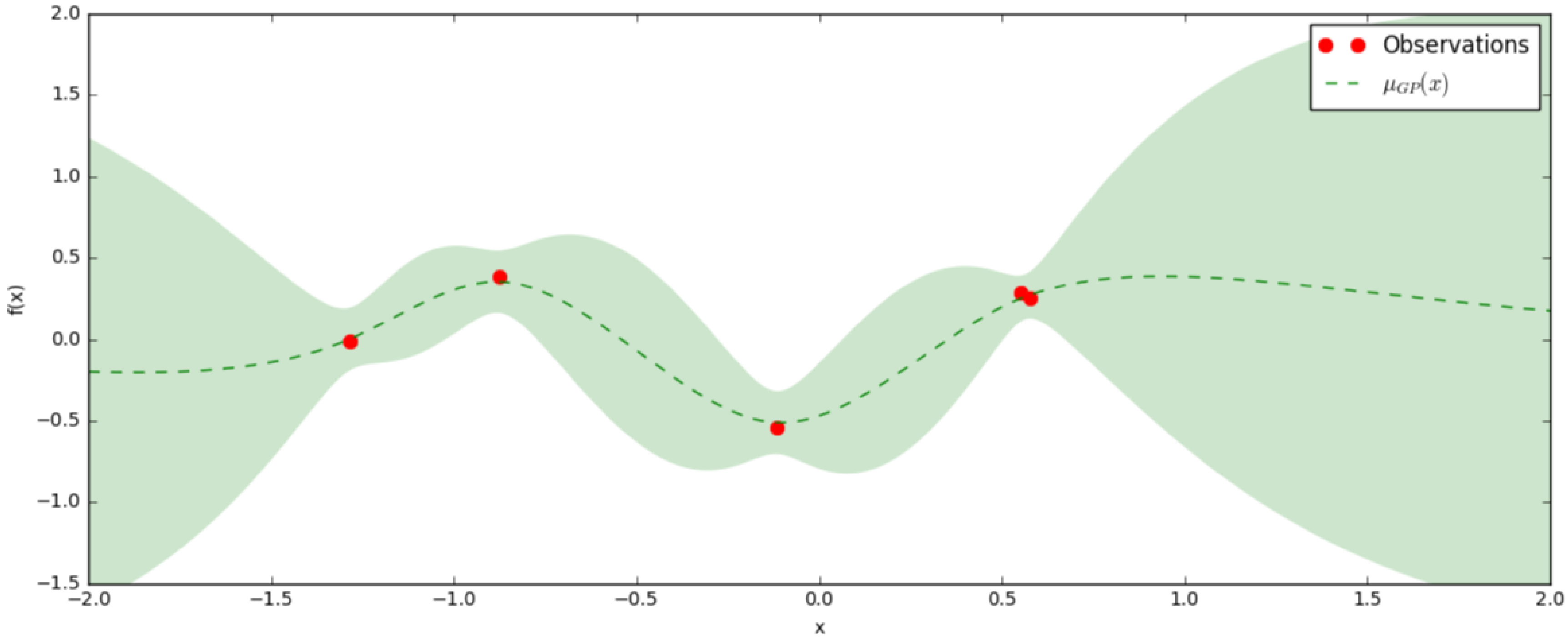
$$\mu_n(x_{n+1}) = \mu_o(x_{n+1}) + k(x_{n+1})^T(K + \sigma^2 I)^{-1}(\mathbf{y} - \mathbf{m})$$

$$\sigma_n^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - \mathbf{k}(x_{n+1})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(x_{n+1})$$

where $\mathbf{k}(x_{n+1})$ is a vector of covariance terms with other observations.

Gaussian Processes

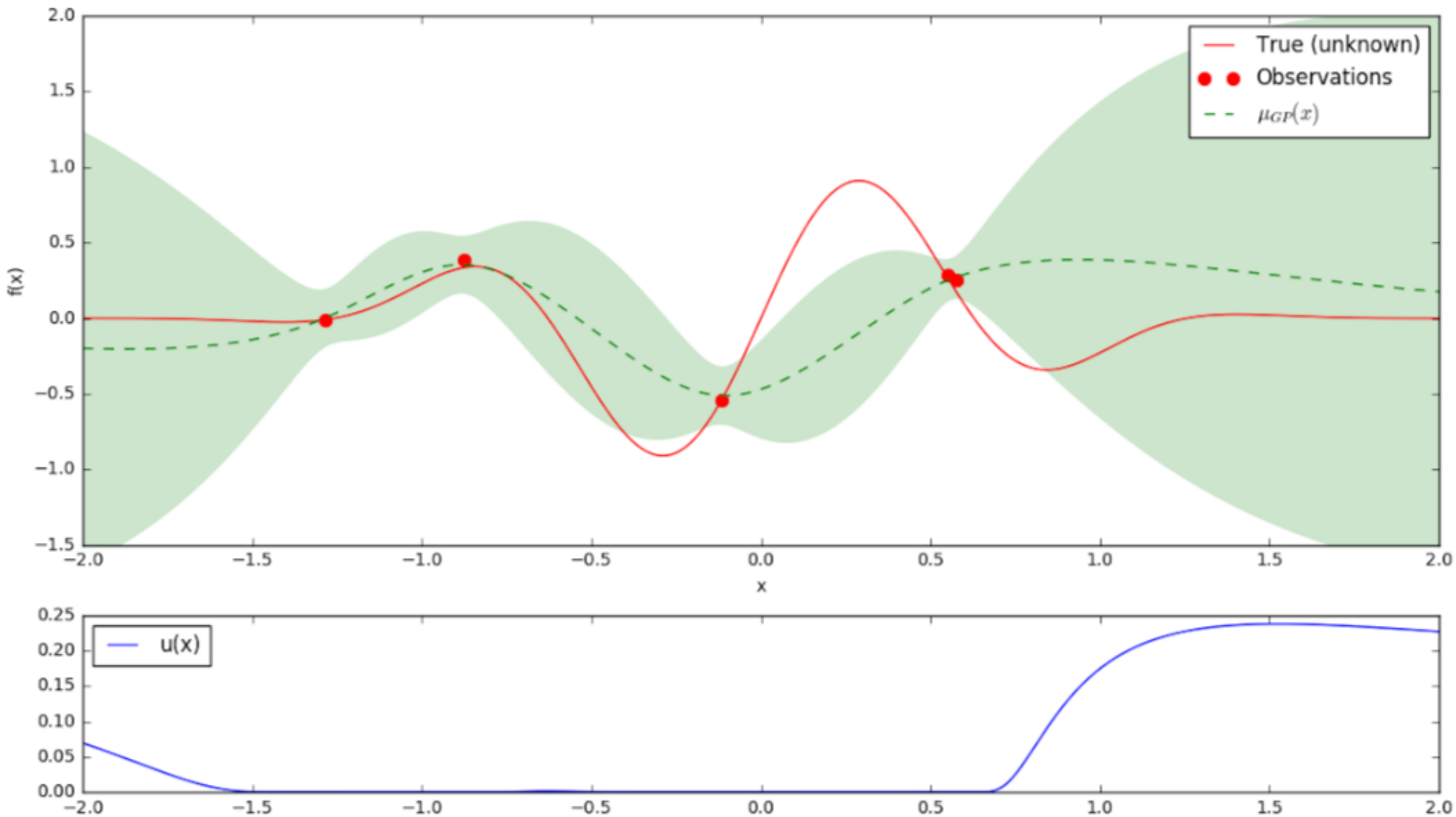
We fit a Gaussian process, picking hyperparameters by maximizing marginal likelihood.



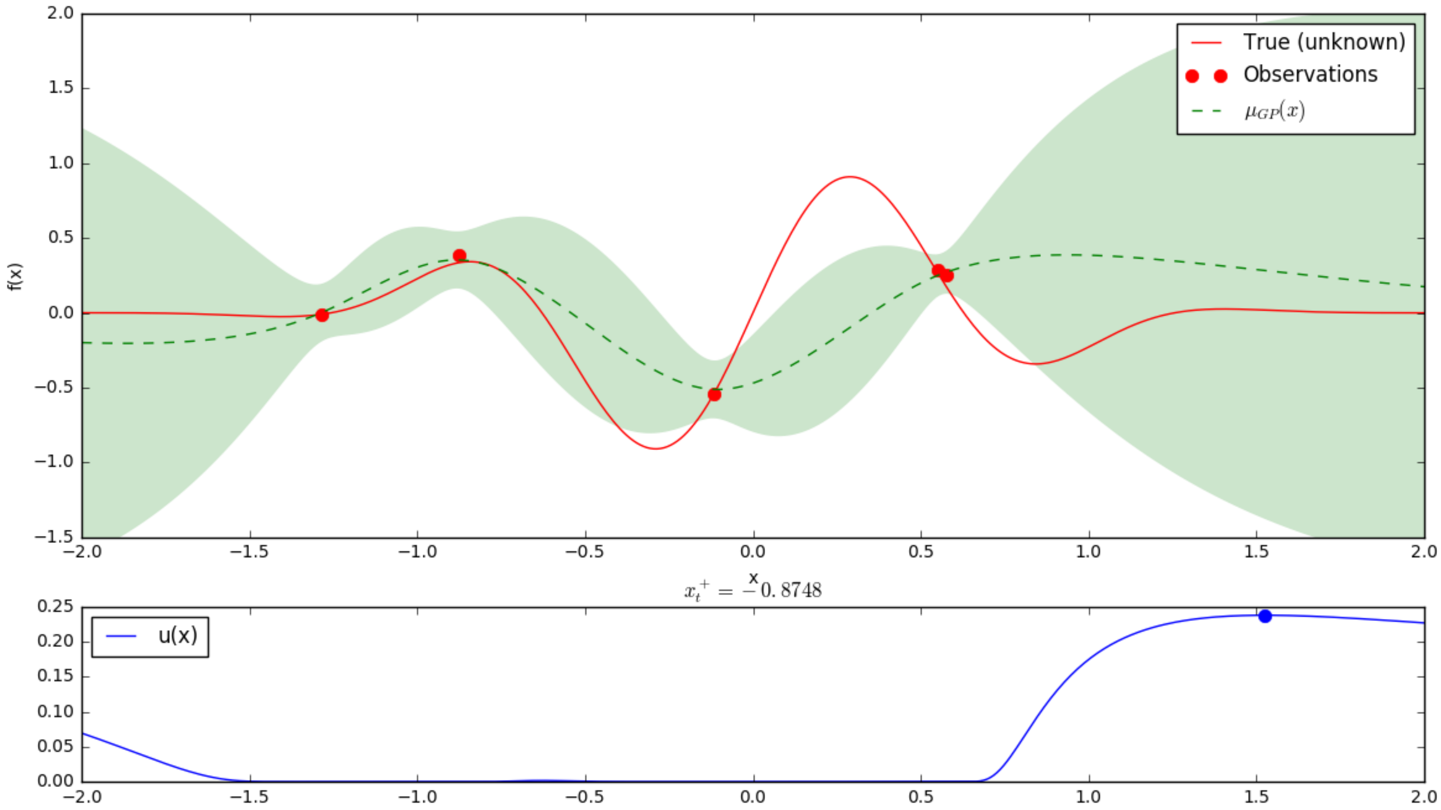
Acquisition Functions

Probability of Improvement

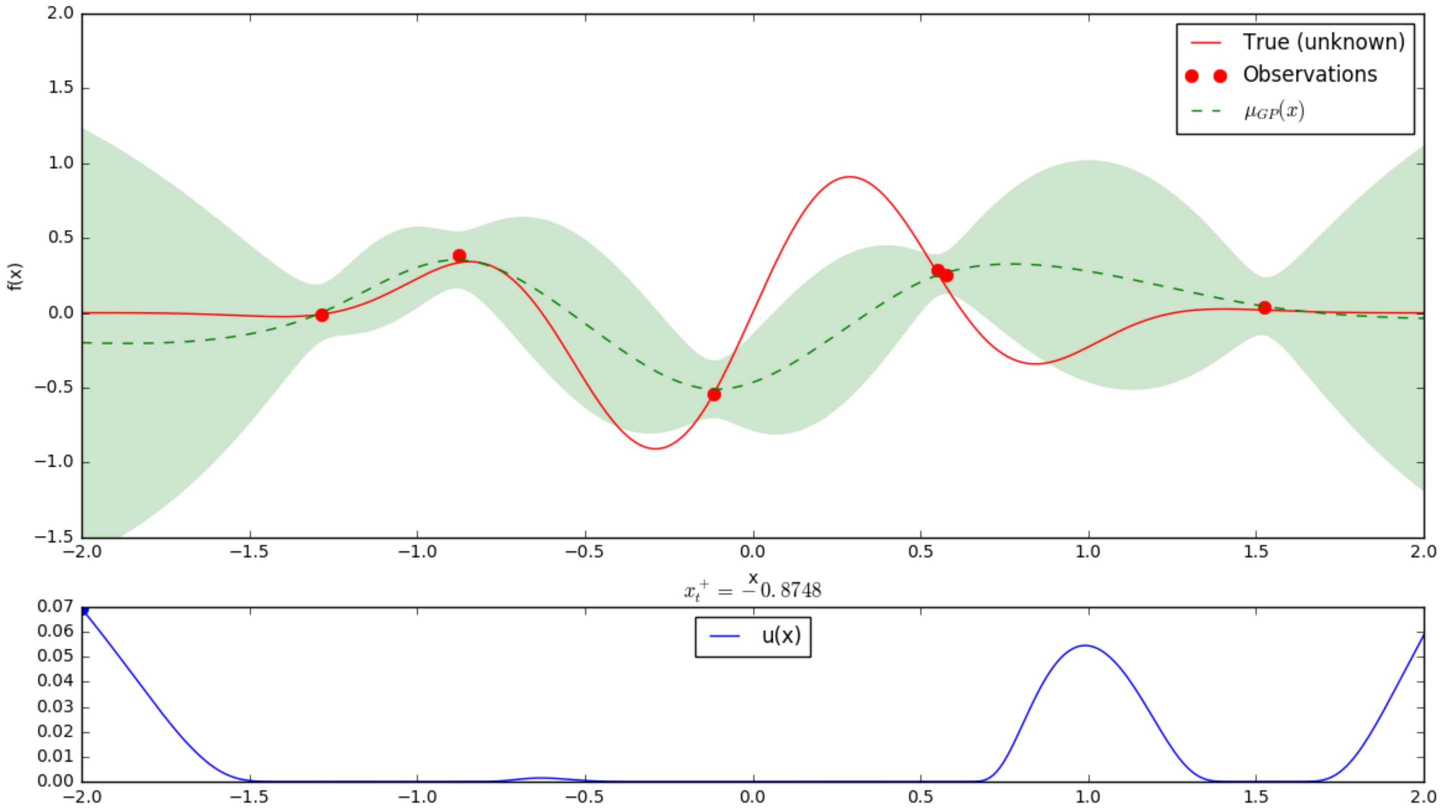
$$\alpha_{PI}(x;\mathcal{D}_n) = p(f_n(x) > y^*) = \Phi(\frac{\mu_n(x)-y^*}{\sigma_n(x)})$$



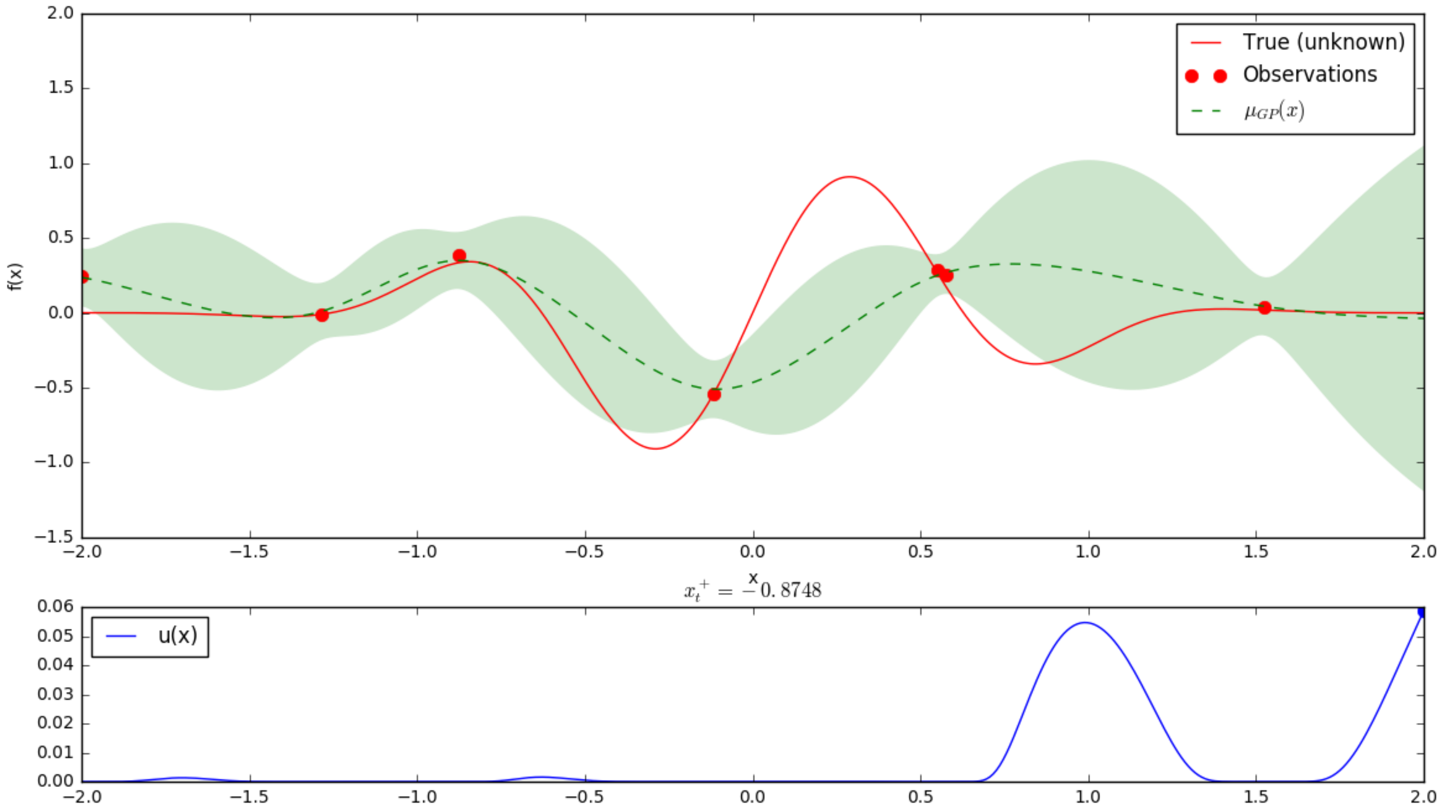
Bayesian Optimization in Action



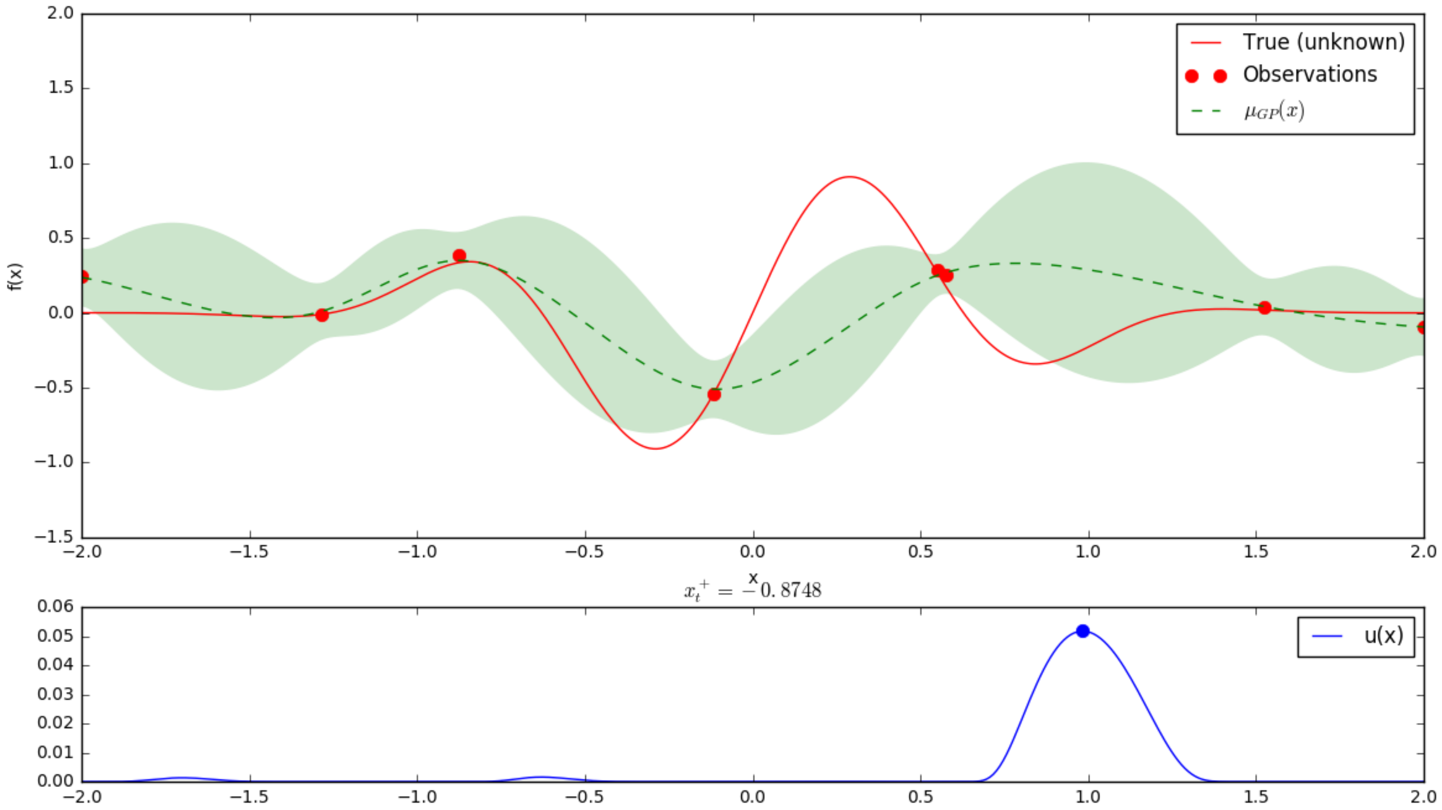
Bayesian Optimization in Action



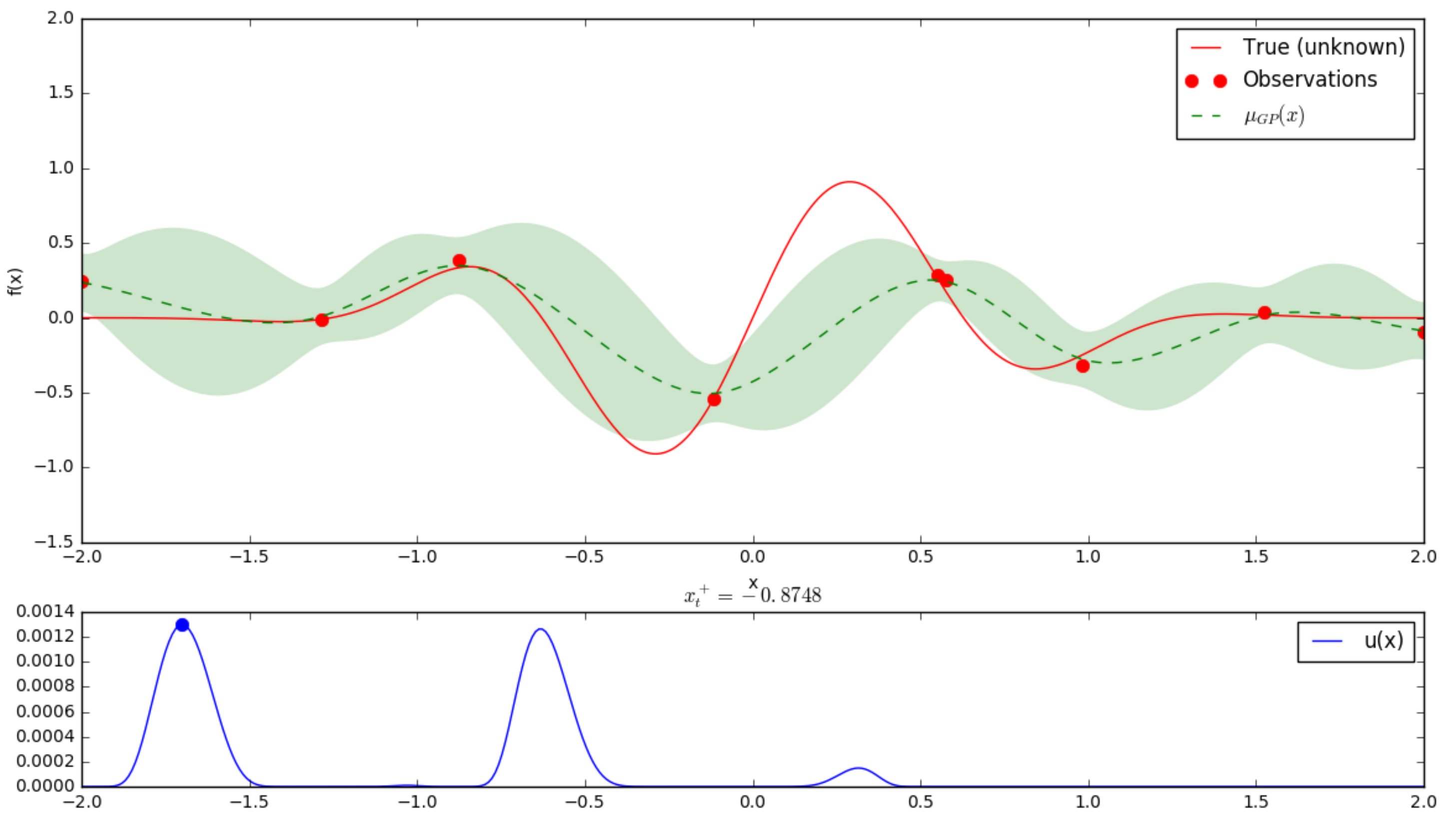
Bayesian Optimization in Action



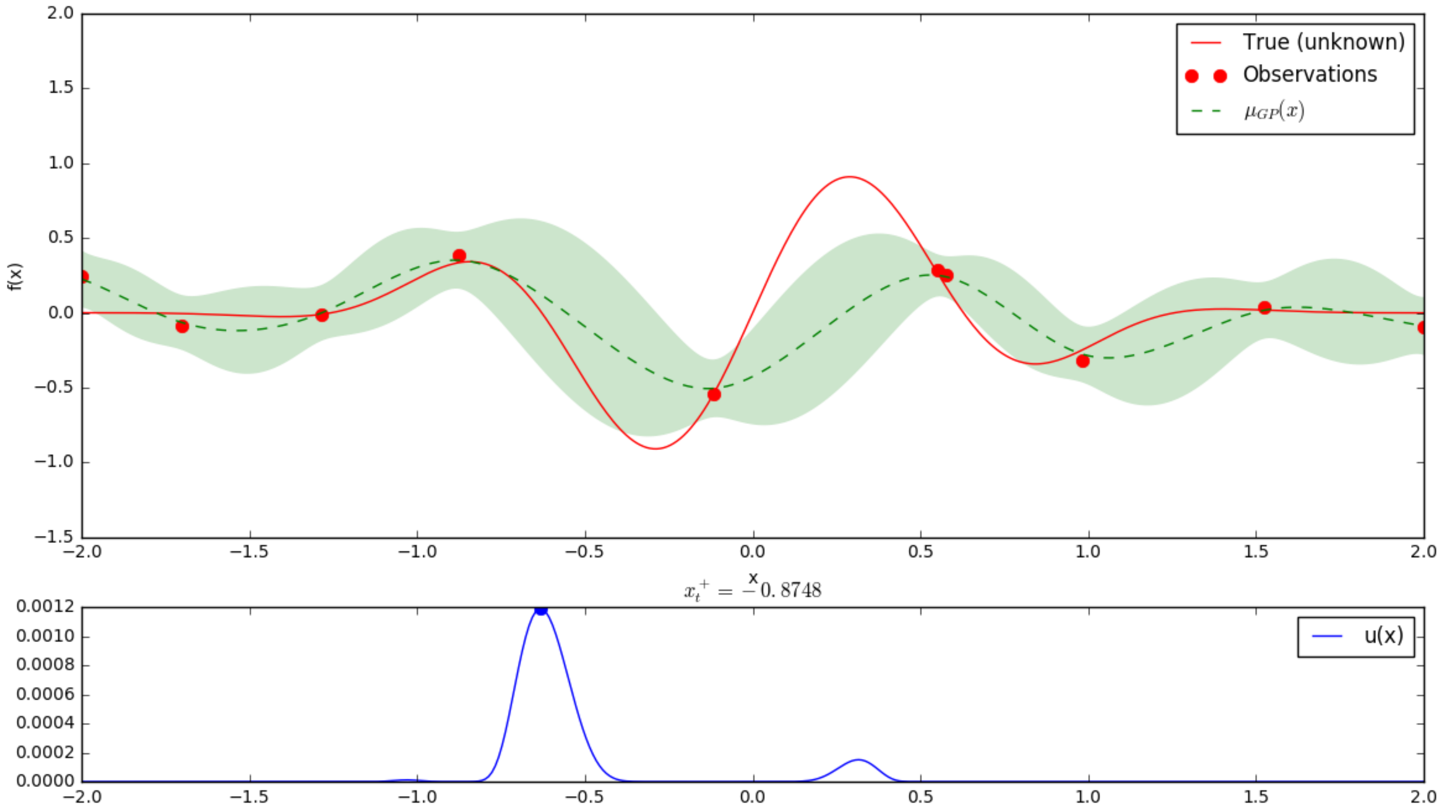
Bayesian Optimization in Action



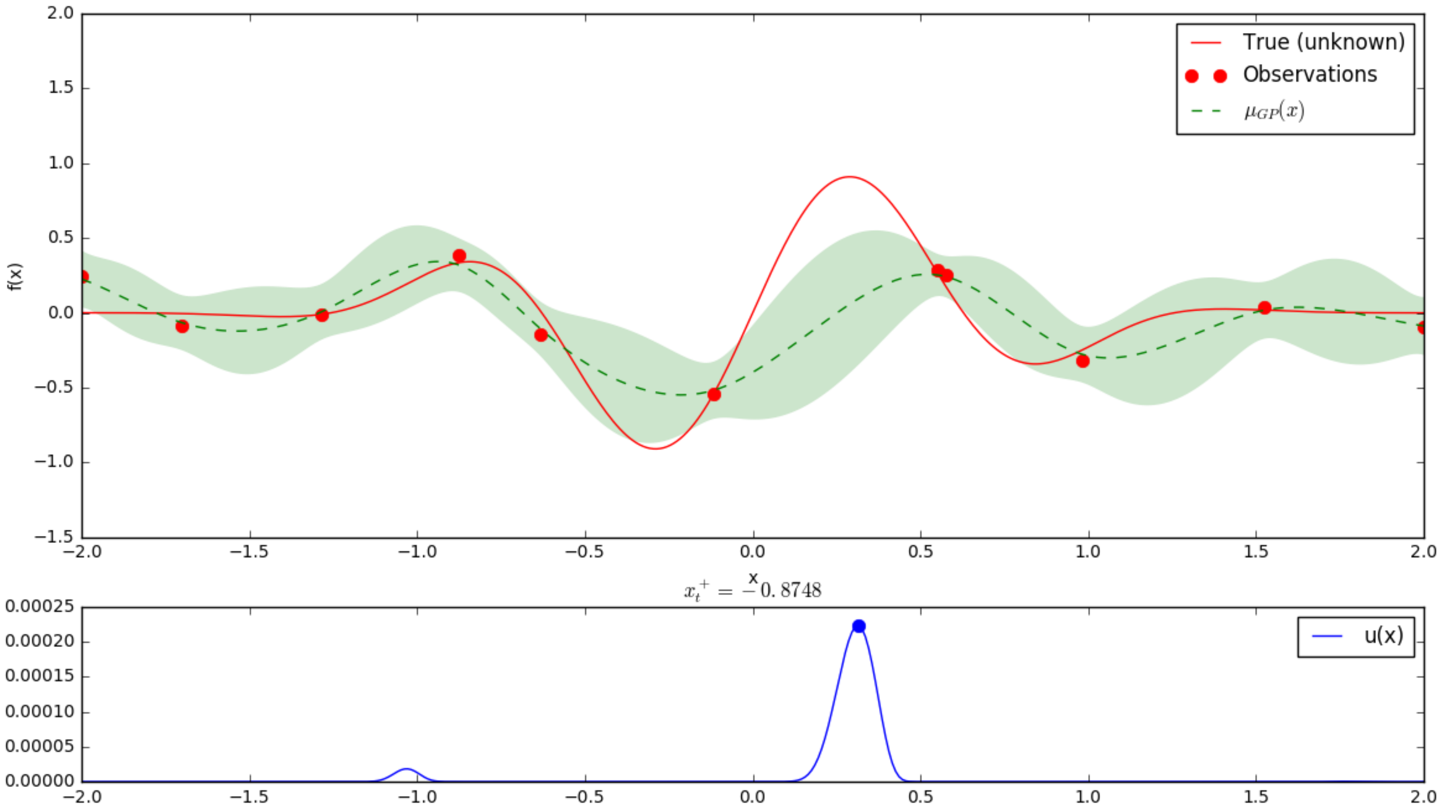
Bayesian Optimization in Action



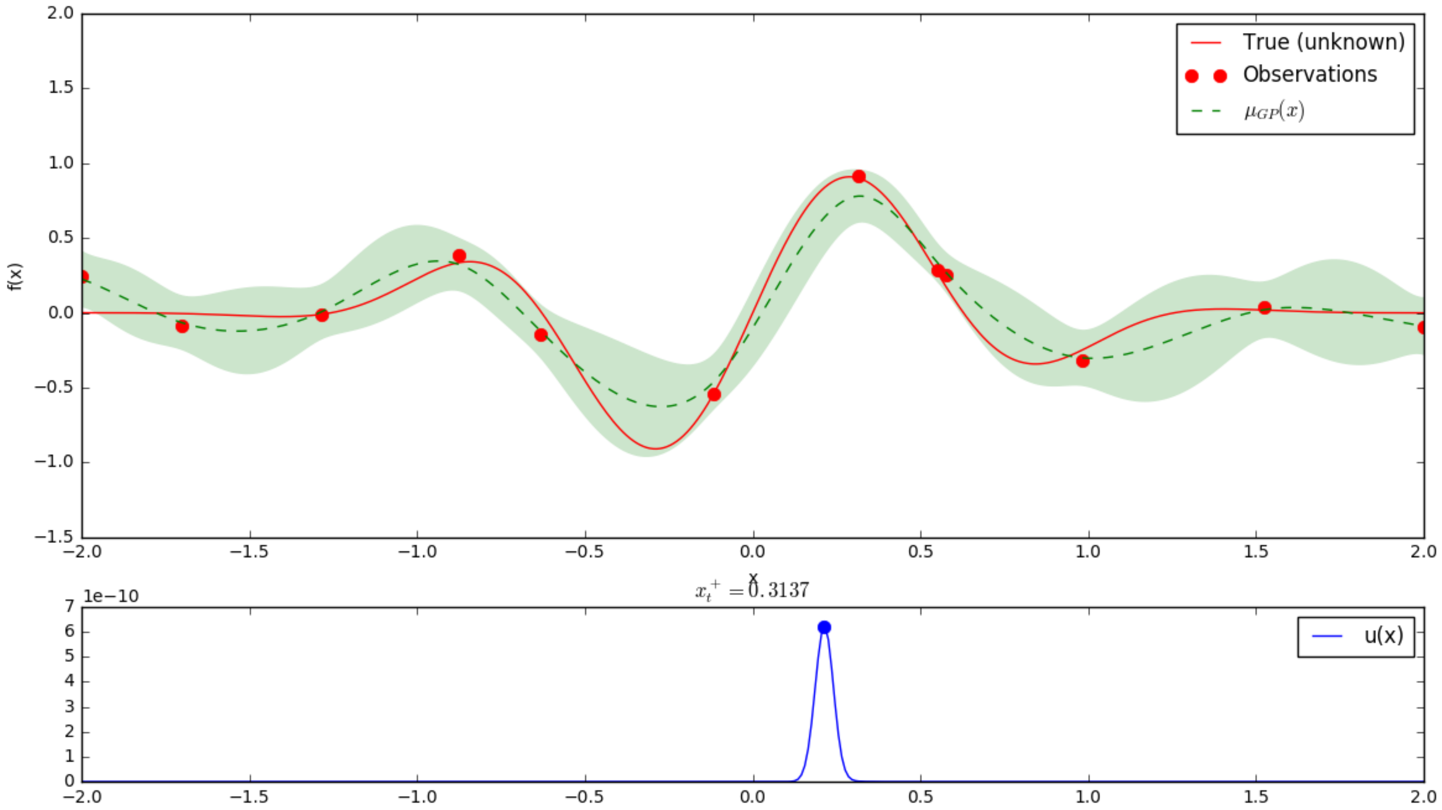
Bayesian Optimization in Action



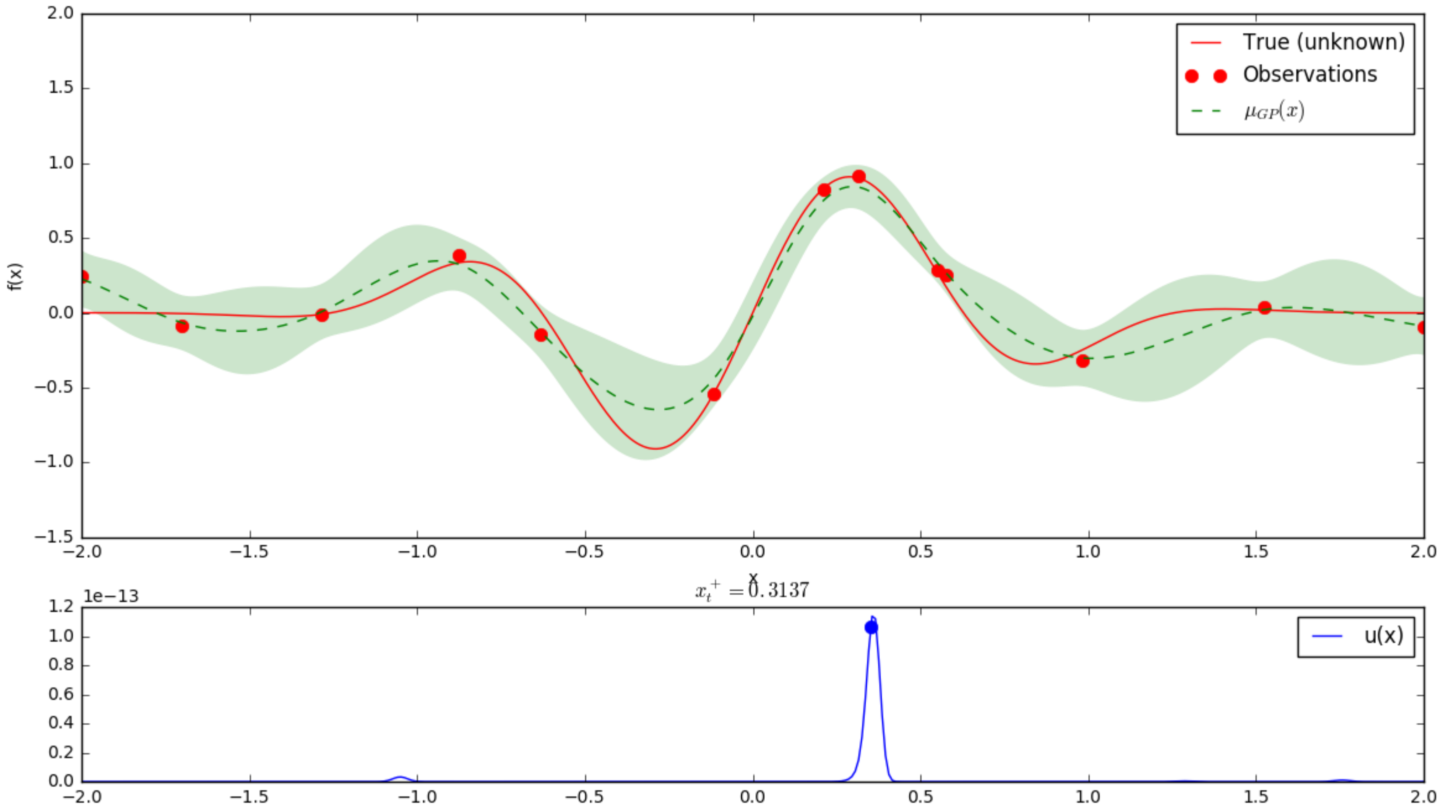
Bayesian Optimization in Action



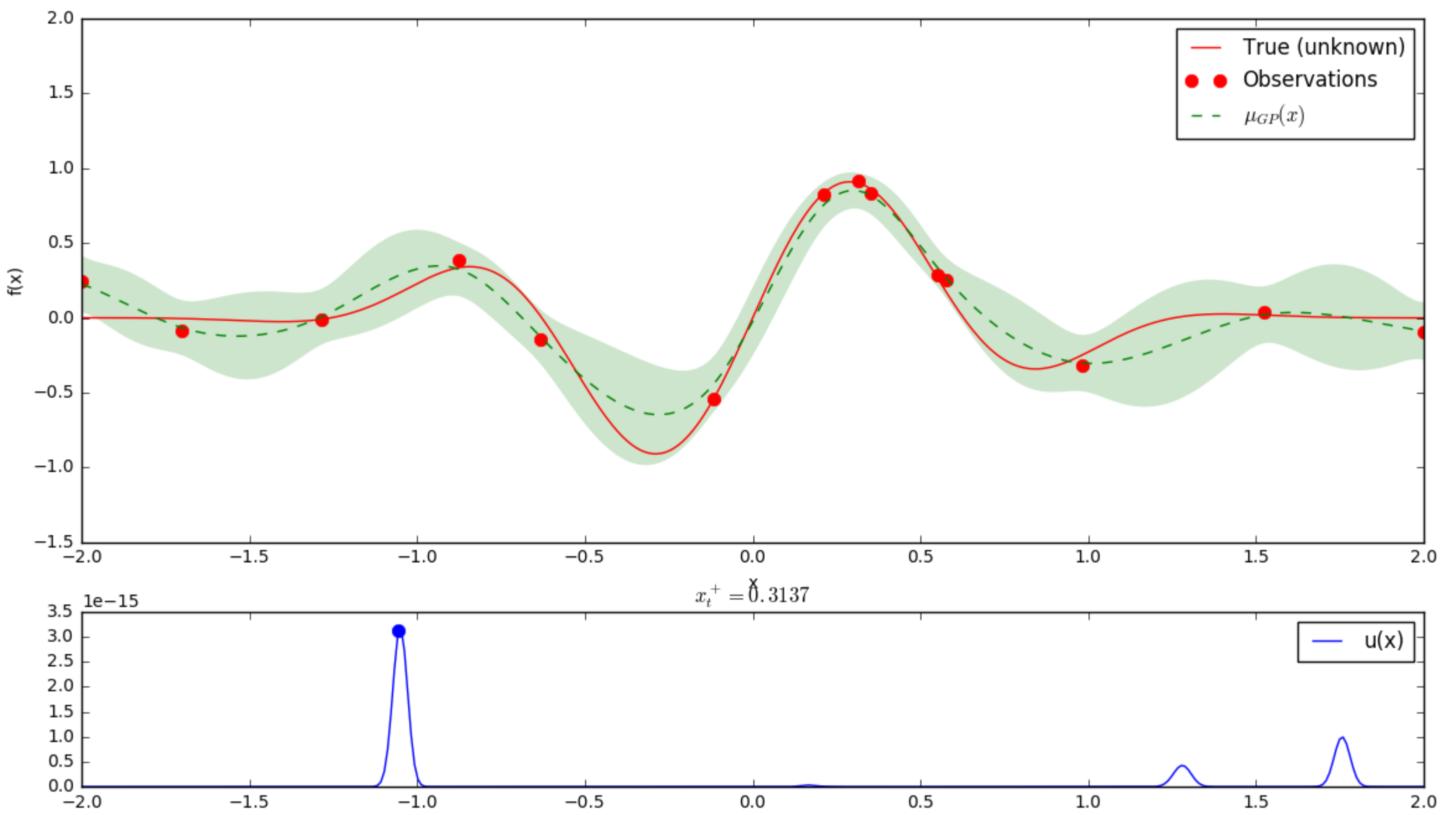
Bayesian Optimization in Action



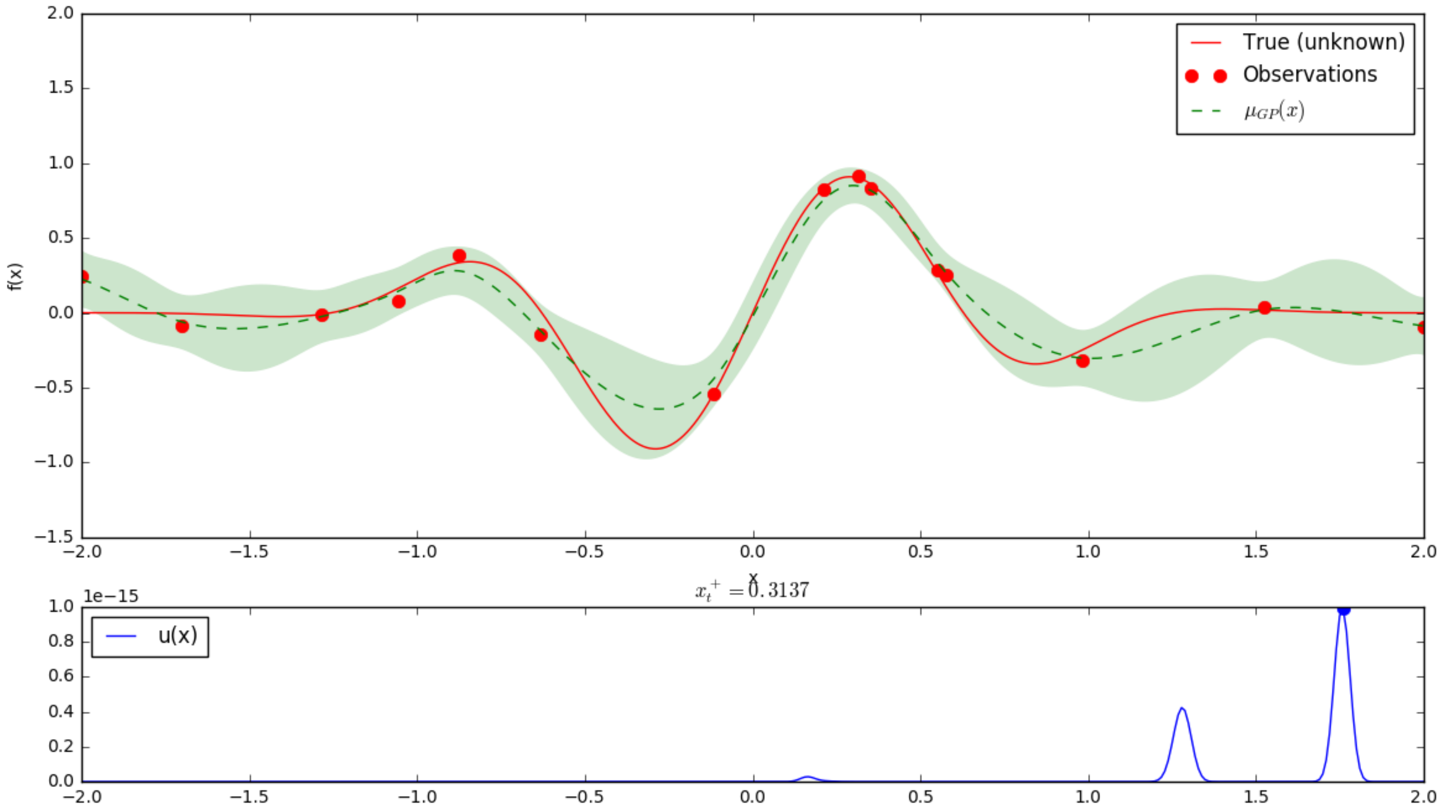
Bayesian Optimization in Action



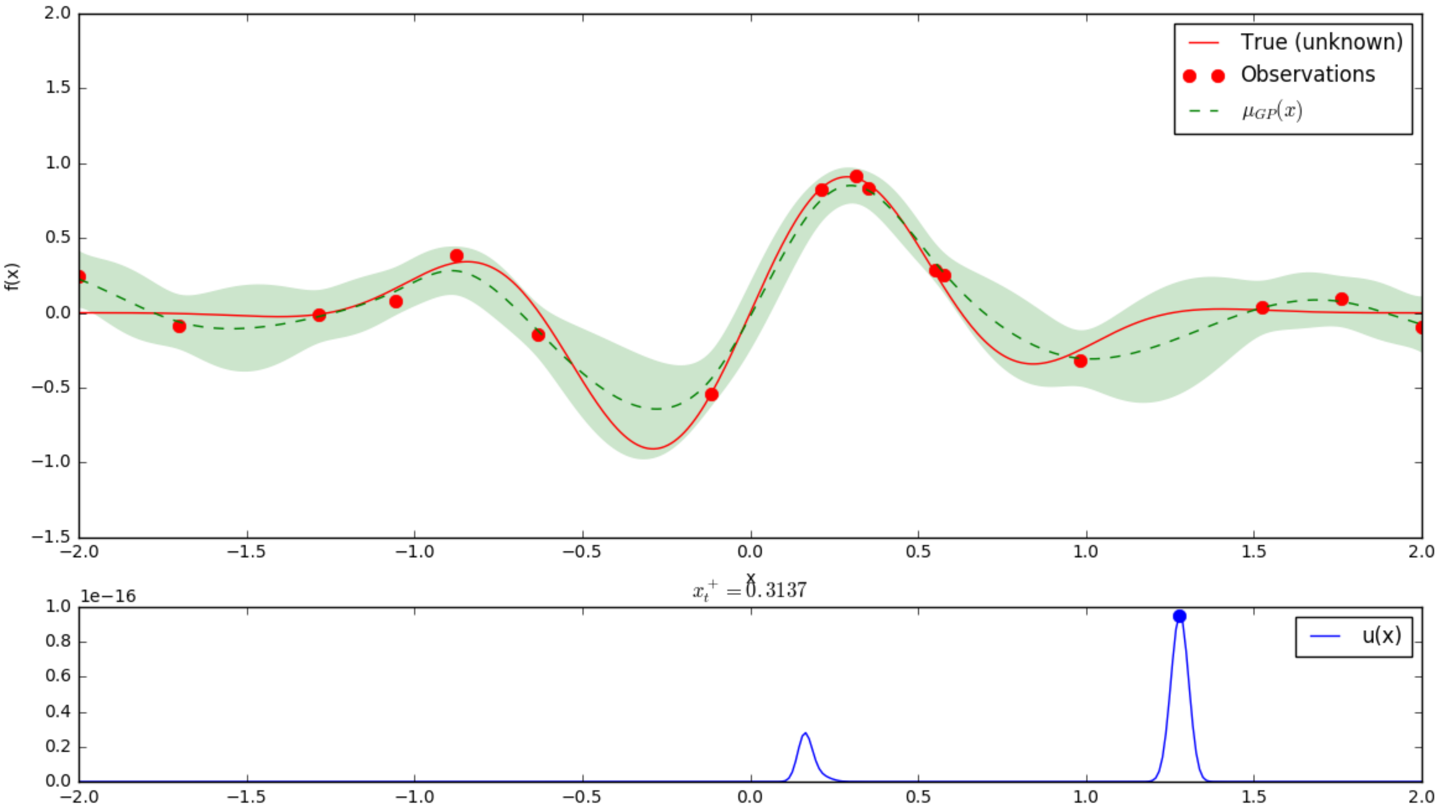
Bayesian Optimization in Action



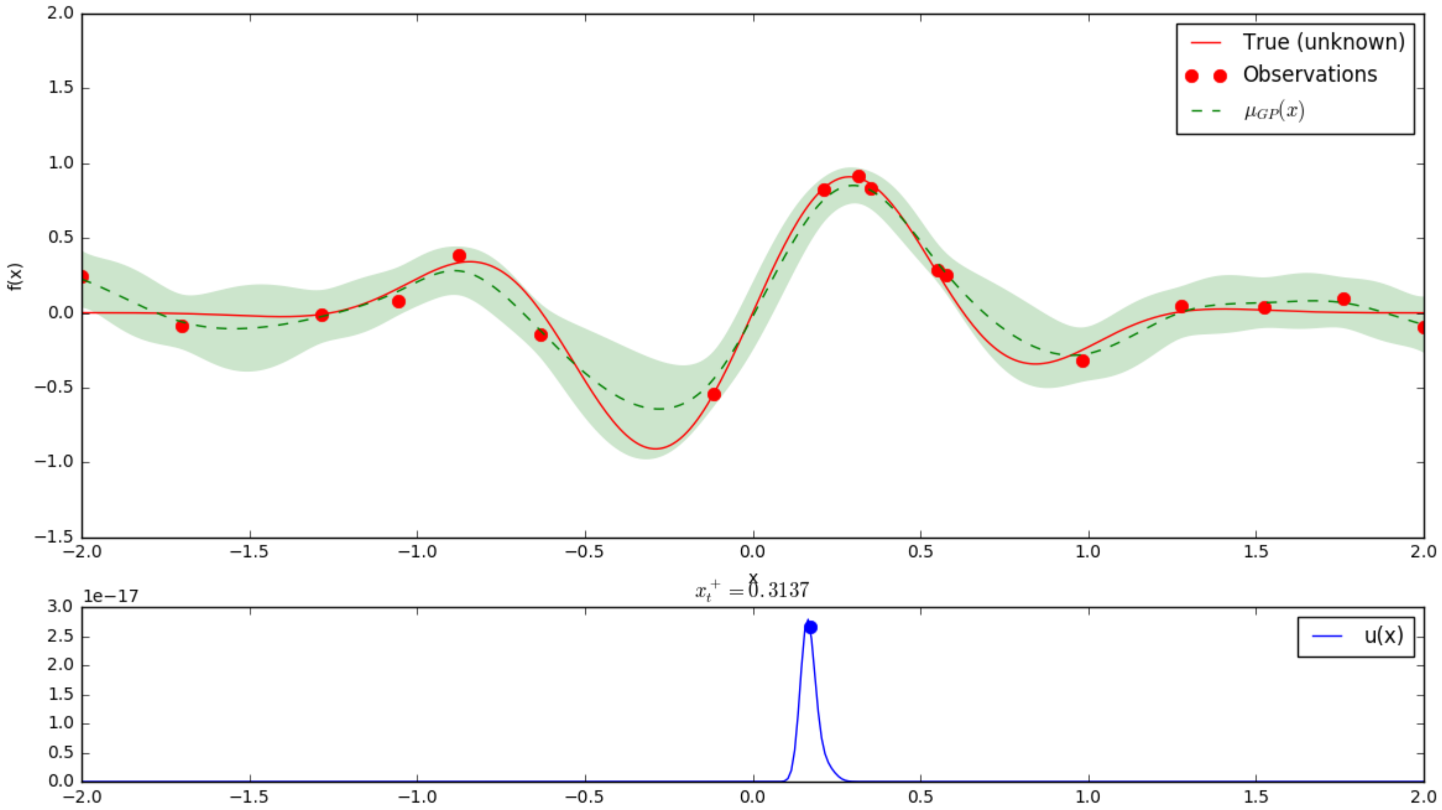
Bayesian Optimization in Action



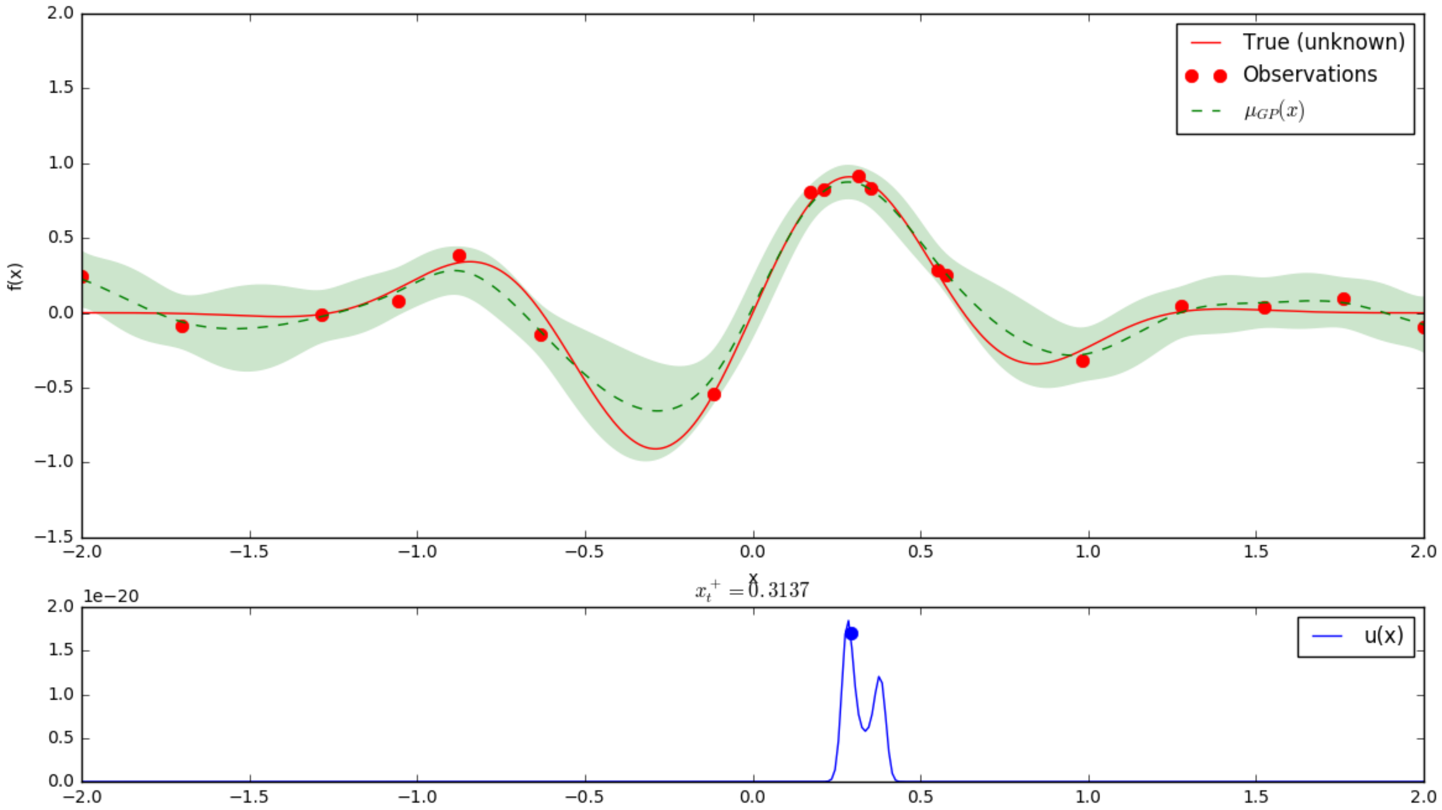
Bayesian Optimization in Action



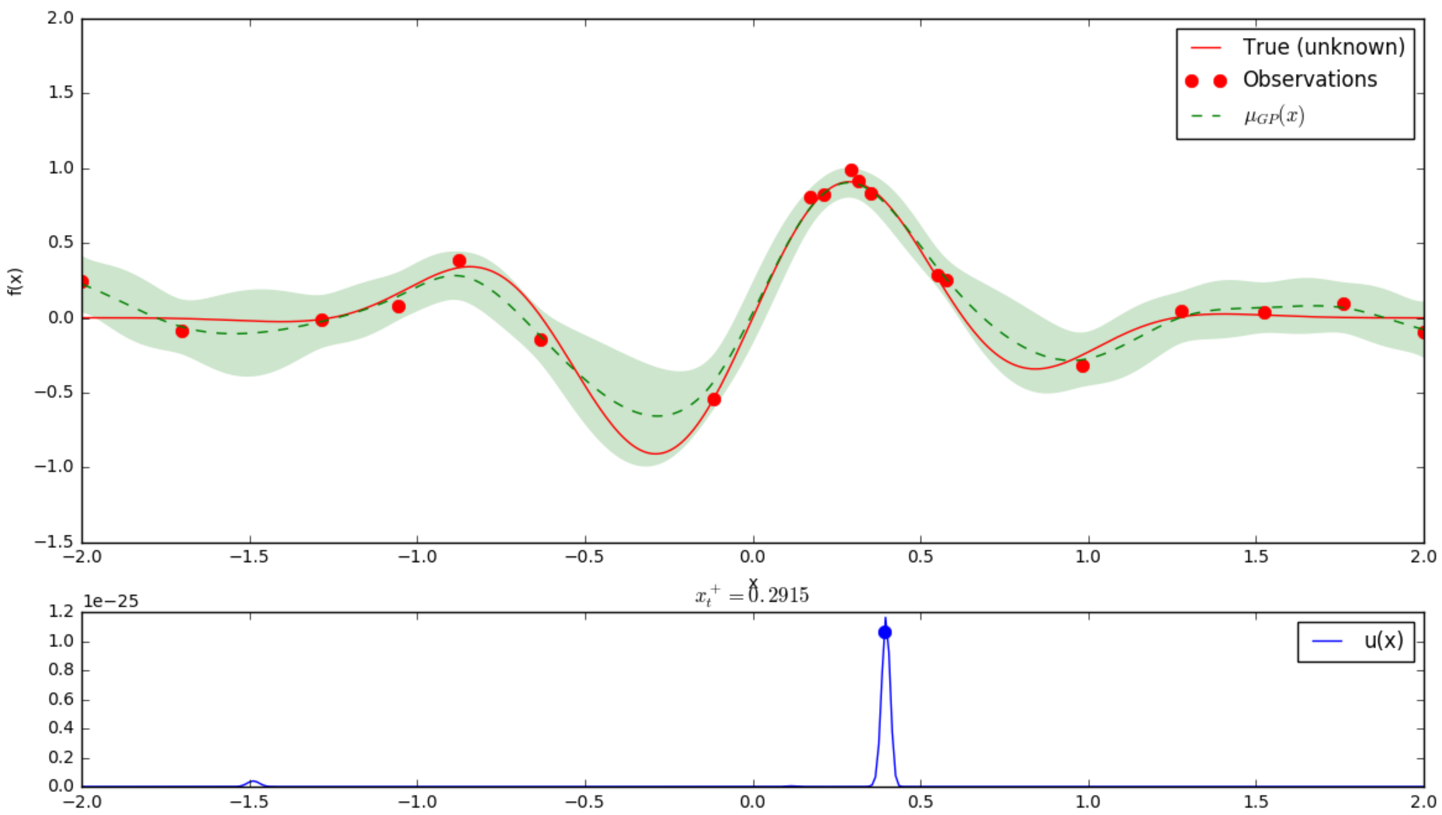
Bayesian Optimization in Action



Bayesian Optimization in Action



Bayesian Optimization in Action



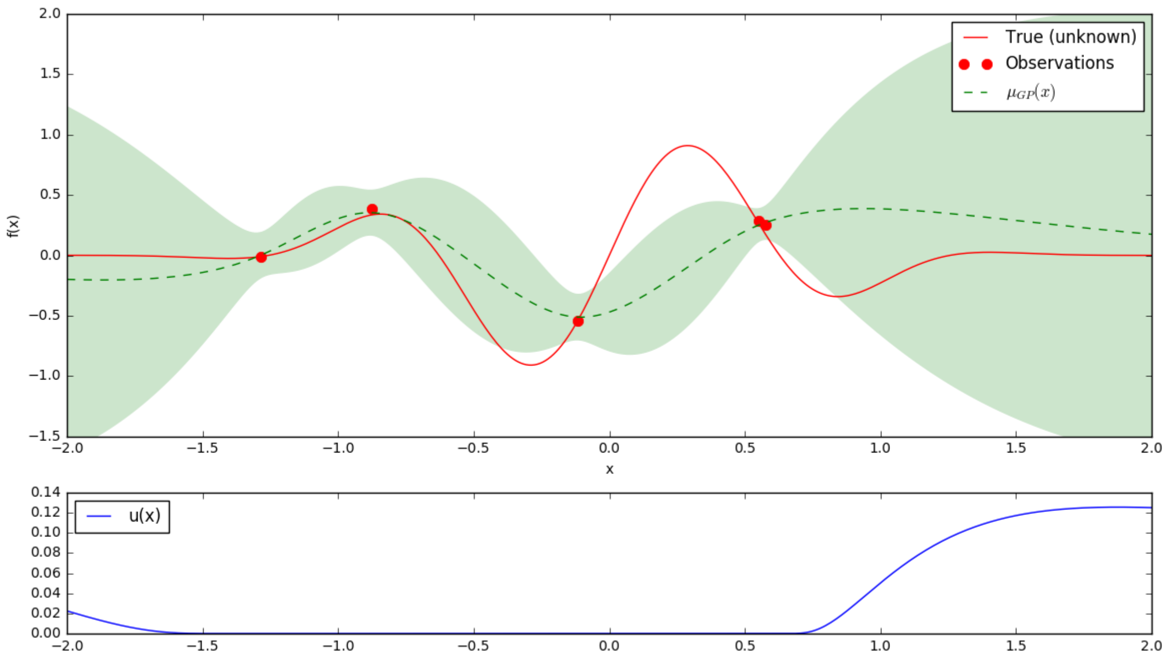
Acquisition Functions

Expected Improvement (EI)

$$\alpha_{\mathrm{EI}}(x|\mathcal{D}_n) = \mathbb{E}_{y\sim p(y|\mathcal{D}_n,x)}[I(x,y)]$$

where $I(x,y) = (y - y^*)\mathbb{I}[y > y^*]$

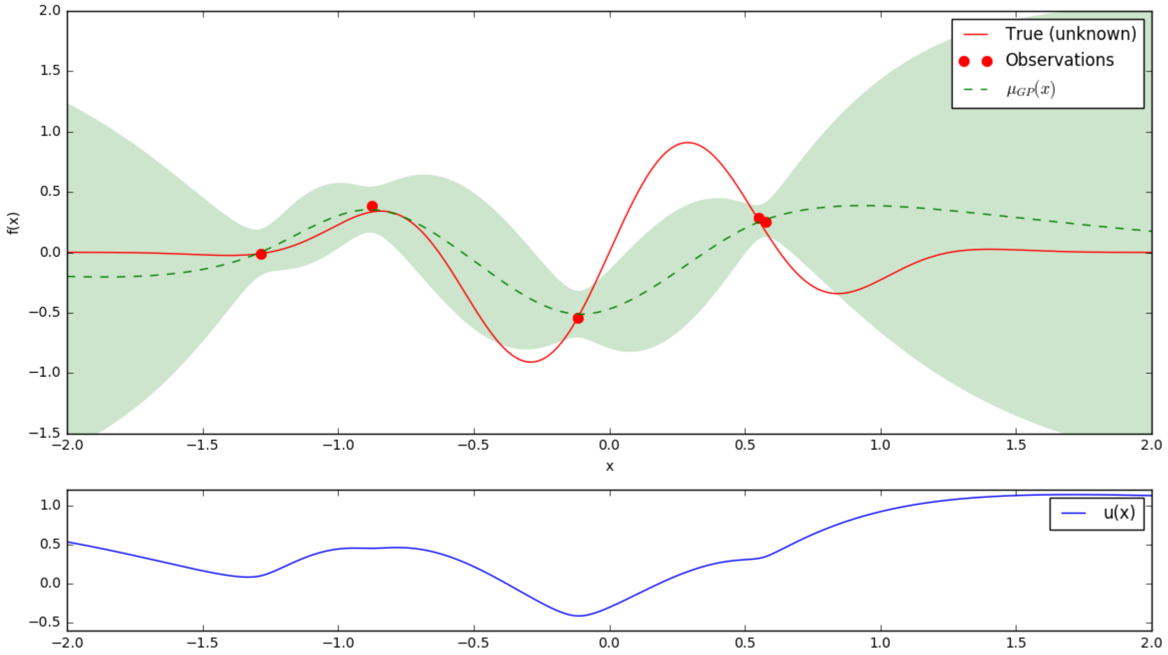
$$= (\mu_n(x) - y^*)\Phi(\frac{\mu_n(x)-y^*}{\sigma_n(x)}) + \sigma_n(x)\phi(\frac{\mu_n(x)-y^*}{\sigma_n(x)})$$



Acquisition Functions

Upper Confidence Bound (UCB)

$$\alpha_{\mathrm{UCB}}(x|\mathcal{D}_n) = \mu_n(x) + \beta_n \sigma_n(x)$$



Information theoretic approach

Entropy search

Information theoretic approach

Entropy search

Consider posterior over the unknown maximizer:

$$p(\mathbf{x}_* | \mathcal{D}_n).$$

Information theoretic approach

Entropy search

Consider posterior over the unknown maximizer:

$$p(\mathbf{x}_*|\mathcal{D}_n).$$

We aim on reducing the uncertainty in the location of $\mathbf{x}_*$:

$$\alpha_{\mathrm{ES}}(\mathbf{x}) := H\left[p(\mathbf{x}_*|\mathcal{D}_n)\right] - \mathbb{E}_{p(y|D_n,\mathbf{x})}\left[H\left[p(\mathbf{x}^*|\mathcal{D}_n \cup \{\mathbf{x}, y\})\right]\right].$$

Predictive Entropy Search

We can take advantage of the symmetry of the mutual information between $\mathbf{x}_*$ and y, i.e:

Predictive Entropy Search

We can take advantage of the symmetry of the mutual information between $\mathbf{x}_*$ and y, i.e:

$$\alpha_{\mathrm{ES}}(\mathbf{x}) = H[\mathbf{x}_*] - H[\mathbf{x}_*|y] = I(\mathbf{x}_*; y) = I(y;\mathbf{x}_*) = H[y] - H[y|\mathbf{x}_*]$$

Predictive Entropy Search

We can take advantage of the symmetry of the mutual information between $\mathbf{x}_*$ and y, i.e:

$$\alpha_{\mathrm{ES}}(\mathbf{x}) = H[\mathbf{x}_*] - H[\mathbf{x}_*|y] = I(\mathbf{x}_*;y) = I(y;\mathbf{x}_*) = H[y] - H[y|\mathbf{x}_*]$$

This leads to the new form of the acquisition function:

$$\alpha_{\mathrm{PES}}(\mathbf{x};\mathcal{D}_n) := H\left[p(y|\mathcal{D}_n,\mathbf{x})\right] - \mathbb{E}_{p(\mathbf{x}_*|\mathcal{D}_n)}\left[H\left[p(y|\mathcal{D}_n,\mathbf{x},\mathbf{x}_*)\right]\right]$$

Sampling from $p(\mathbf{x}_* | \mathcal{D}_n)$

We will use Monte Carlo approximation to compute the expectation.

Sampling from $p(\mathbf{x}_*|\mathcal{D}_n)$

We will use Monte Carlo approximation to compute the expectation.

A natural approach is to produce sample from posterior:

$$p(\mathbf{f}|\mathcal{D}_n)$$

and maximize the function $\mathbf{f}$ to find $\mathbf{x}_*$.

Sampling from $p(\mathbf{x}_*|\mathcal{D}_n)$

We will use Monte Carlo approximation to compute the expectation.

A natural approach is to produce sample from posterior:

$$p(\mathbf{f}|\mathcal{D}_n)$$

and maximize the function $\mathbf{f}$ to find $\mathbf{x}_*$.

However, this would cost $\mathcal{O}(m^3)$ where m is the number of function evaluations.

Thus, we need the analytic approximation of $\mathbf{f}$.

Sampling from $p(\mathbf{x}_* | \mathcal{D}_n)$

As a consequence of the Bochner's theorem every stationary kernel k has an associated normalized spectral density $p(\mathbf{w})$ and you can show that:

Sampling from $p(\mathbf{x}_* | \mathcal{D}_n)$

As a consequence of the Bochner's theorem every stationary kernel k has an associated normalized spectral density $p(\mathbf{w})$ and you can show that:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= 2\alpha \mathbb{E}_{p(\mathbf{w}, b)}[\cos(\mathbf{w}^T \mathbf{x} + b)\cos(\mathbf{w}^T \mathbf{x}' + b)] \\ &\approx \phi(\mathbf{x})^T \phi(\mathbf{x}') \end{aligned}$$

Sampling from $p(\mathbf{x}_*|\mathcal{D}_n)$

As a consequence of the Bochner's theorem every stationary kernel k has an associated normalized spectral density $p(\mathbf{w})$ and you can show that:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x'}) &= 2\alpha\mathbb{E}_{p(\mathbf{w},b)}[\cos(\mathbf{w}^T\mathbf{x} + b)\cos(\mathbf{w}^T\mathbf{x'} + b)] \\ &\approx \phi(\mathbf{x})^T\phi(\mathbf{x'}) \end{aligned}$$

We approximate the Gaussian process prior for f with a linear model:

$$f(\mathbf{x}) \approx \phi(\mathbf{x})^T\theta$$

where $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.

Sampling from $p(\mathbf{x}_* | \mathcal{D}_n)$

As a consequence of the Bochner's theorem every stationary kernel k has an associated normalized spectral density $p(\mathbf{w})$ and you can show that:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= 2\alpha \mathbb{E}_{p(\mathbf{w},b)}[\cos(\mathbf{w}^T \mathbf{x} + b) \cos(\mathbf{w}^T \mathbf{x}' + b)] \\ &\approx \phi(\mathbf{x})^T \phi(\mathbf{x}') \end{aligned}$$

We approximate the Gaussian process prior for f with a linear model:

$$f(\mathbf{x}) \approx \phi(\mathbf{x})^T \theta$$

where $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. This can be maximized to obtain:

$$\mathbf{x}_*^{(i)} = \arg\max_{\mathbf{x} \in \mathcal{X}} f^{(i)}(\mathbf{x}).$$

Approximating $p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n, \mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

Approximating $p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n, \mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

If only we wouldn't have to condition on $\mathbf{x}_*$...

Approximating $p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n, \mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

If only we wouldn't condition on $\mathbf{x}_*$...

To circumvent the difficulty, we introduce the following constraints:

Approximating $p(y|\mathcal{D}_n, \mathbf{x},\mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x},\mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n,\mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

If only we wouldn't condition on $\mathbf{x}_*$...

To circumvent the difficulty, we introduce the following constraints:

1. $\mathbf{x}_*$ is only a local maximum,

Approximating $p(y|\mathcal{D}_n, \mathbf{x},\mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x},\mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n,\mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

If only we wouldn't condition on $\mathbf{x}_*$...

To circumvent the difficulty, we introduce the following constraints:

1. $\mathbf{x}_*$ is only a local maximum,
2. $f(\mathbf{x}_*)$ is larger than past observations,

Approximating $p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n, \mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

If only we wouldn't condition on $\mathbf{x}_*$...

To circumvent the difficulty, we introduce the following constraints:

1. $\mathbf{x}_*$ is only a local maximum,
2. $f(\mathbf{x}_*)$ is larger than past observations,
3. $f(\mathbf{x})$ is smaller than $f(\mathbf{x}_*)$.

Approximating $p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*)$

Let's note that:

$$p(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}_*) = \int p(y|f(\mathbf{x}))p(f(\mathbf{x})|\mathcal{D}_n, \mathbf{x}_*)\mathrm{d}f(\mathbf{x})$$

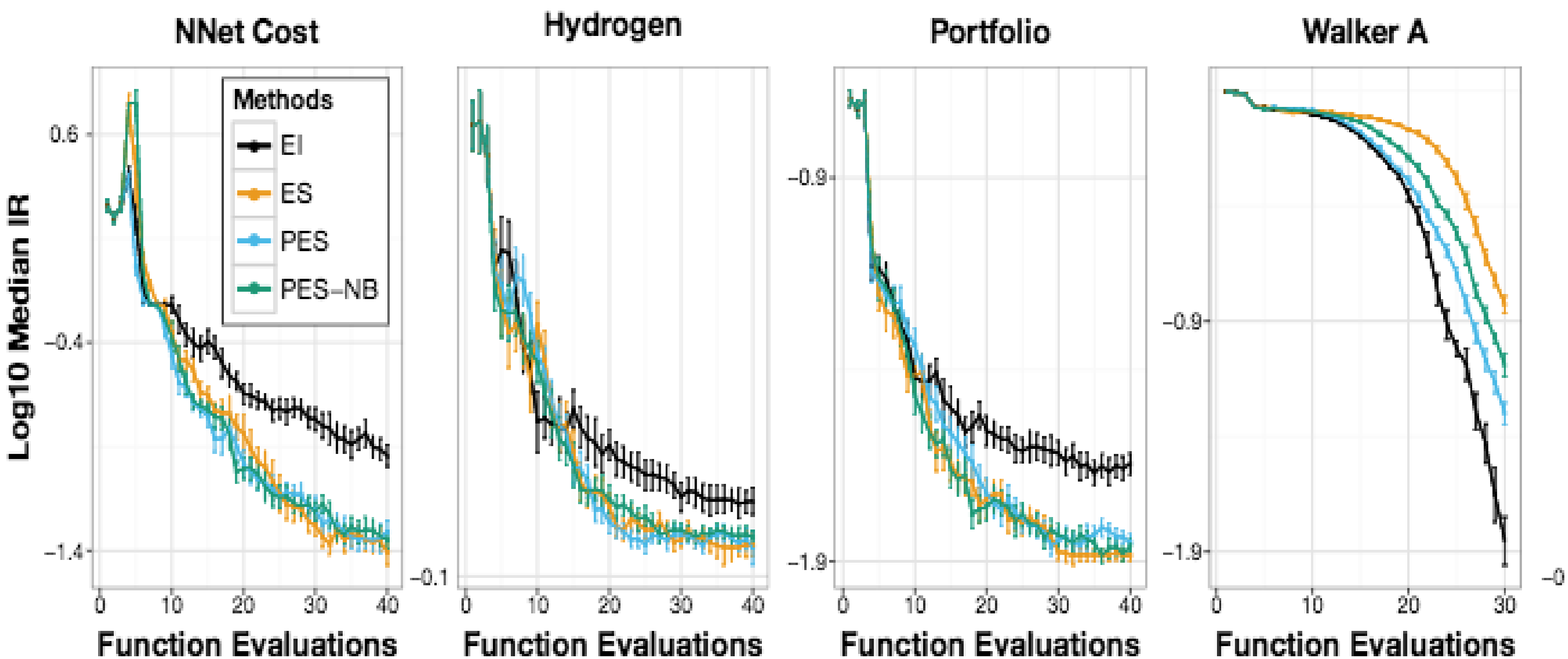
If only we wouldn't condition on $\mathbf{x}_*$...

To circumvent the difficulty, we introduce the following constraints:

1. $\mathbf{x}_*$ is only a local maximum,
2. $f(\mathbf{x}_*)$ is larger than past observations,
3. $f(\mathbf{x})$ is smaller than $f(\mathbf{x}_*)$.

You can find more in Hernandez-Lobato et al. 2014.

How does it work?



Probabilistic Frameworks

Using GPs

How do we choose a kernel and hyper-parameters?

- A common approach is by empirical Bayes
  - $\theta^* = \operatorname{argmax}_\theta p(\mathcal{D}_n|\theta)$
  - $\alpha(x) = \mathrm{E}_{f|\theta^*}[U(f(x))]$

Using GPs

How do we choose a kernel and hyper-parameters?

- A common approach is by empirical Bayes
  - $\theta^* = \ \mathrm{argmax}_\theta \, p(\mathcal{D}_n|\theta)$
  - $\alpha(x) = \ \mathrm{E}_{f|\theta^*}[U(f(x))]$
- Alternatively we can be more Bayesian
  - $\theta \sim p(\theta|\lambda)$
  - $\alpha(x) = \ \mathrm{E}_{\theta|\mathcal{D}_n,\lambda}[\mathrm{E}_{f|\theta}[U(f(x))]]$

Shortcomings of GPs

- Variable length scales
  - non-stationary kernels
  - input warping

Shortcomings of GPs

- Variable length scales
    - non-stationary kernels
    - input warping
- Scalability in N
    - Naively calculating posterior predictive is $O(N^3)$

Shortcomings of GPs

- Variable length scales
  - non-stationary kernels
  - input warping
- Scalability in N
  - Naively calculating posterior predictive is $O(N^3)$
- Scalability with Dimensionality

Multi-Task Bayesian Optimization

We are provided with T related functions, $(f_1, f_2, ...f_T)$, and are interested in optimizing one of them, f_t.

We can simultaneously model all of the functions with a multi-output GP

Multi-Task Bayesian Optimization

We are provided with T related functions, $(f_1, f_2, ... f_T)$, and are interested in optimizing one of them, f_t.

We can simultaneously model all of the functions with a multi-output GP

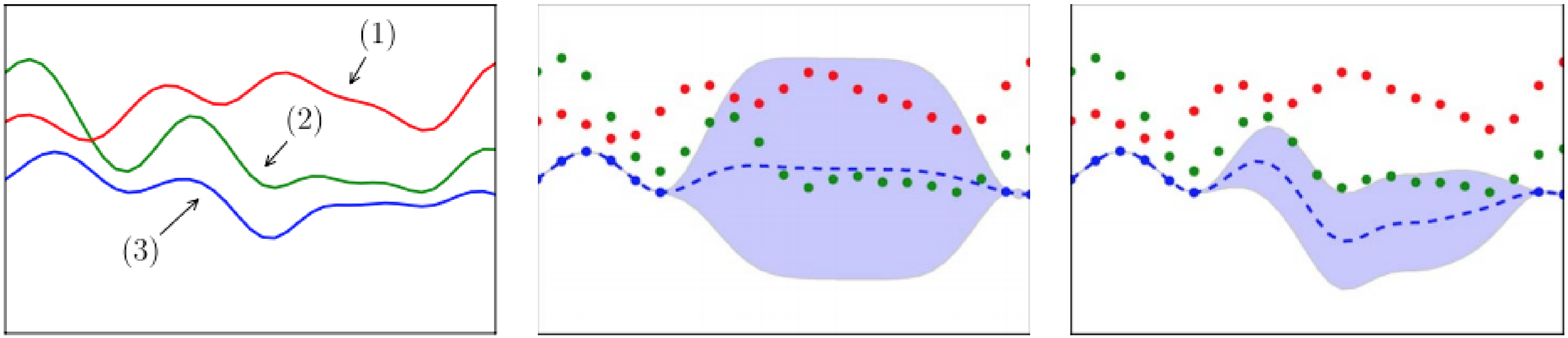
We use the *intrinisic model of coregionalization*

- $k((x,t),(x',t')) = k_{\mathbb{X}}(x,x')k_T(t,t')$

Matthias Seeger, Yee-Whye Teh, and Michael I. Jordan. Semiparametric latent factor models. In AISTATS, 2005.

Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In NIPS, 2008.

Multi-Task Bayesian Optimization



Transferring knowledge from other tasks, f_1 and f_2 informs our prior over f_3

We can better cope with the size of the space!

Swersky, Kevin, Jasper Snoek, and Ryan P. Adams. "Multi-task bayesian optimization." Advances in neural information processing systems. 2013.

Bayesian Neural Nets

Short intro

Bayesian Neural Nets

Let's denote by θ parameters of out network. We are interested in computing the posterior predictive distribution

$$p(y|\mathbf{x},\mathcal{D}_n).$$

Bayesian Neural Nets

Let's denote by θ parameters of out network. We are interested in computing the posterior predictive distribution

$$p(y|\mathbf{x},\mathcal{D}_n).$$

Using posterior distribution over parameters:

$$p(\theta|\mathcal{D}_n) \propto p(\theta)\prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\theta)$$

we have:

$$p(y|\mathbf{x},\mathcal{D}_n) = \int p(y|\mathbf{x},\theta)p(\theta|\mathcal{D}_n)\mathrm{d}\theta$$

Approximations for $p(\theta|\mathcal{D}_n)$

Approximations for $p(\theta|\mathcal{D}_n)$

1. Probabilistic backpropagation (PBP), Hernandez-Lobato and Adams, 2015,

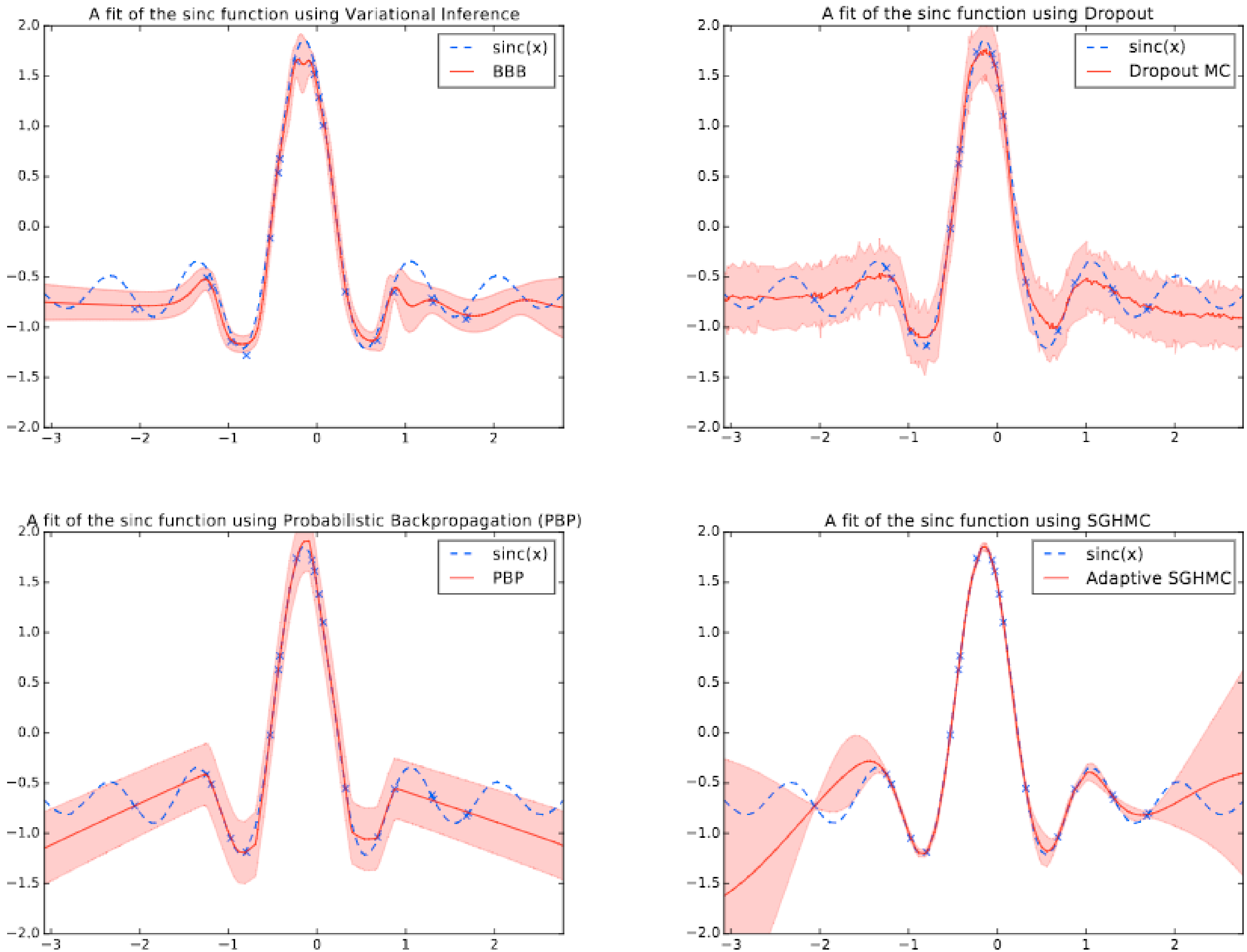
Approximations for $p(\theta|\mathcal{D}_n)$

1. Probabilistic backpropagation (PBP), Hernandez-Lobato and Adams, 2015,
2. Variational inference - Bayes by backprop (BBB), Blundell et al., 2015,

Approximations for $p(\theta|\mathcal{D}_n)$

1. Probabilistic backpropagation (PBP), Hernandez-Lobato and Adams, 2015,
2. Variational inference - Bayes by backprop (BBB), Blundell et al., 2015,
3. Dropout MC, Gal and Ghahramani, 2015

Comparison


A fit of the sinc function using Variational Inference
sinc(x)
BBB
A fit of the sinc function using Dropout
sinc(x)
Dropout MC
A fit of the sinc function using Probabilistic Backpropagation (PBP)
sinc(x)
PBP
A fit of the sinc function using SGHMC
sinc(x)
Adaptive SGHMC

MCMC methods for $p(\theta|\mathcal{D}_n)$

MCMC methods for $p(\theta|\mathcal{D}_n)$

Stochastic gradient Langevin dynamics (SGLD), Welling and Teh, 2011. This is just:

$$\theta_{t+1} = \theta_t + \epsilon_t/2\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(\mathbf{x}_{ti}|\theta_t)\right) + \eta_t,$$

where $\eta_t \sim \mathcal{N}(0, \epsilon_t)$.

MCMC methods for $p(\theta|\mathcal{D}_n)$

MCMC methods for $p(\theta|\mathcal{D}_n)$

Methods based on hybrid Monte Carlo - many variations.

MCMC methods for $p(\theta|\mathcal{D}_n)$

Methods based on hybrid Monte Carlo - many variations.

MCMC methods for $p(\theta|\mathcal{D}_n)$

Methods based on hybrid Monte Carlo - many varations.

We consider a joint system of θ and auxiliary momentum variables, r:

$$p(\theta, r|\mathcal{D}_n) \propto \exp\left(-\log p(\theta,\mathcal{D}_n) - \frac{1}{2}r^T M^{-1} r\right).$$

MCMC methods for $p(\theta|\mathcal{D}_n)$

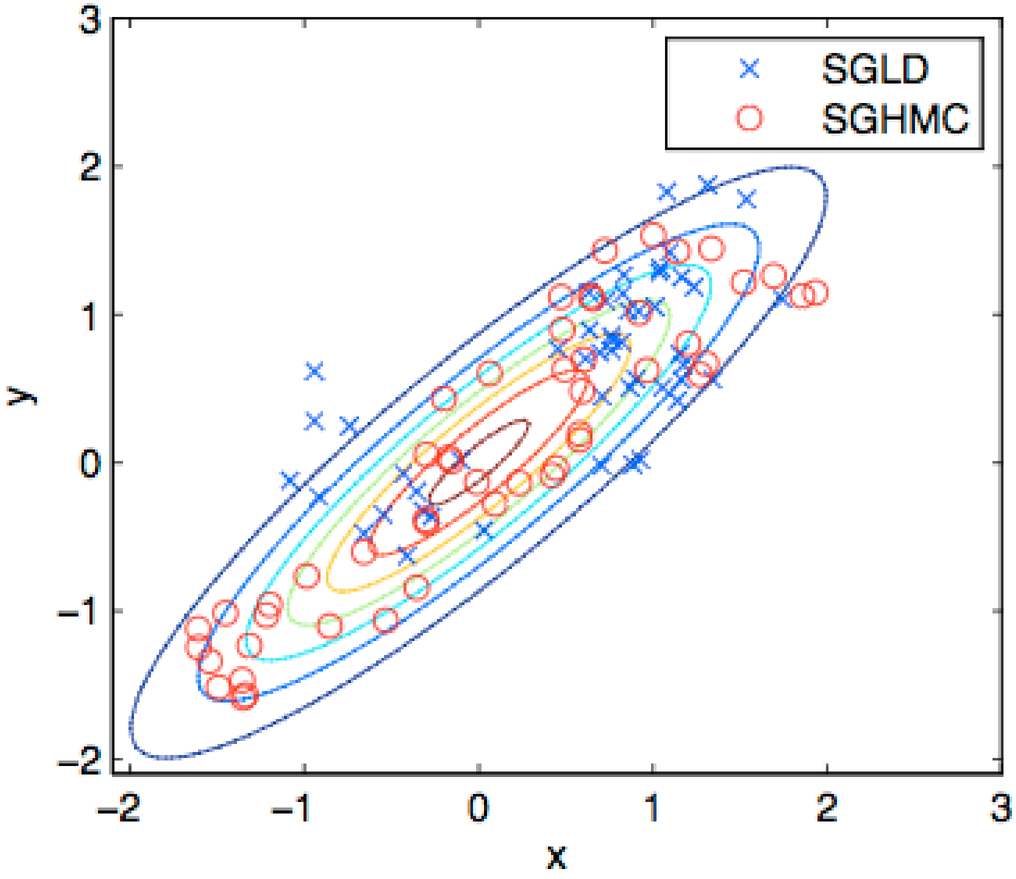
Methods based on hybrid Monte Carlo - many varations.

We consider a joint system of θ and auxiliary momentum variables, r:

$$p(\theta, r|\mathcal{D}_n) \propto \exp\left(-\log p(\theta,\mathcal{D}_n) - \frac{1}{2}r^T M^{-1} r\right).$$

Stochastic gradient Hamiltionian Monte Carlo (SGHMC), Chen et. al. 2014.

SGLD vs SGHMC



Bayesian Optimization with Hamiltonian Monte Carlo Artificial Neural Networks (BOHAMIANN)

Springenberg et al., 2016

Bayesian Optimization with Hamiltonian Monte Carlo Artificial Neural Networks (BOHAMIANN)

Springenberg et al., 2016

In the context of BNN, our probabilistic model takes the form:

$$p(f_t(\mathbf{x})|\mathbf{x}, \theta) = \mathcal{N}(\hat{f}(\mathbf{x}, t; \theta_\mu), \theta_{\sigma^2})$$

where $\hat{f}(\mathbf{x}, t; \theta_\mu)$ is the output of a parametric model with parameters θ_μ and noise is assumed to be homoscedastic.

How does it work?

How does it work?

We follow SGHMC algorithm to reach the posterior distribution - this requires running a chain for a 'long' time.

How does it work?

We follow SGHMC algorithm to reach the posterior distribution - this requires running a chain for a 'long' time.

At test time we use S samples θ^s generated using our chain which can be seen as samples from the posterior $p(\theta|\mathcal{D}_n)$.

What about acquisition function?

We managed to generate approximate samples $\theta_i \sim p(\theta|\mathcal{D}_n)$ from the posterior. For the EI we can obtain:

What about acquisition function?

We managed to generate approximate samples $\theta_i \sim p(\theta|\mathcal{D}_n)$ from the posterior. For the EI we can obtain:

$$p(f_t(\mathbf{x})|\mathbf{x},\mathcal{D}_n) \approx \frac{1}{S}\sum_{i=1}^{S}\mathcal{N}(\hat{f}(\mathbf{x},t;\theta_\mu^s),\theta_{\sigma^2})$$

What about acquisition function?

We managed to generate approximate samples $\theta_i \sim p(\theta|\mathcal{D}_n)$ from the posterior. For the EI we can obtain:

$$p(f_t(\mathbf{x})|\mathbf{x},\mathcal{D}_n) \approx \frac{1}{S}\sum_{i=1}^{S}\mathcal{N}(\hat{f}(\mathbf{x},t;\theta_\mu^s),\theta_{\sigma^2})$$

and

$$\mu(f_t(\mathbf{x}|\mathcal{D}_n)) = \frac{1}{S}\sum_{i=1}^{S}\hat{f}(\mathbf{x};\theta_\mu^s).$$

What about acquisition function?

We managed to generate approximate samples $\theta_i \sim p(\theta|\mathcal{D}_n)$ from the posterior. For the EI we can obtain:

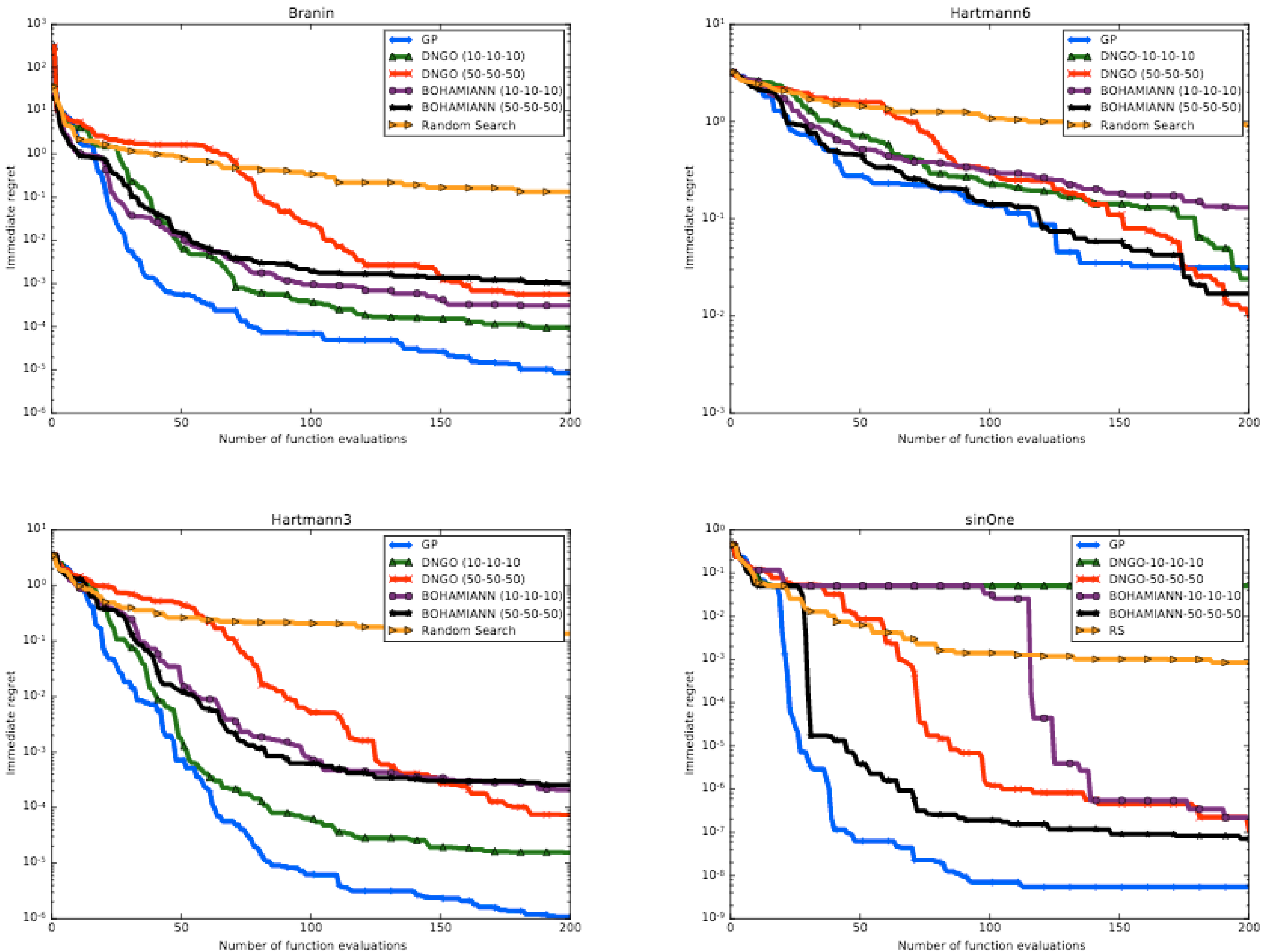
$$p(f_t(\mathbf{x})|\mathbf{x},\mathcal{D}_n) \approx \frac{1}{S}\sum_{i=1}^{S}\mathcal{N}(\hat{f}(\mathbf{x},t;\theta_\mu^s),\theta_{\sigma^2})$$

and

$$\mu(f_t(\mathbf{x}|\mathcal{D}_n)) = \frac{1}{S}\sum_{i=1}^{S}\hat{f}(\mathbf{x};\theta_\mu^s).$$

Note - we can compute partial derivatives of α_{EI} with respect to $\mathbf{x}$ which allows us to use standard gradient-based techniques to maximize acquisition function!

Final comparison



Open Questions and Problems

- Parallelization
    - Non-sequential samples
- Cost sensitivity
    - When we have variable cost for different queries (e.g. layer sizes)
    - Expected improvement per second

Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012.

Open Questions and Problems

- Exploration vs Exploitation tradeoff changes over time
  - 'Portfolios' of acquisition functions are often the best solution.
  - A meta criterion is used to pick the best solution across the portfolio

- Acquisition functions are often don't answer the questions we want to ask
  - Short-sighted
  - Don't consider utility of exploitation/exploration

Hoffman, Matthew D., Eric Brochu, and Nando de Freitas. "Portfolio Allocation for Bayesian

Optimization." UAI. 2011.

Active learning

Active learning

1. Contextual bandits,

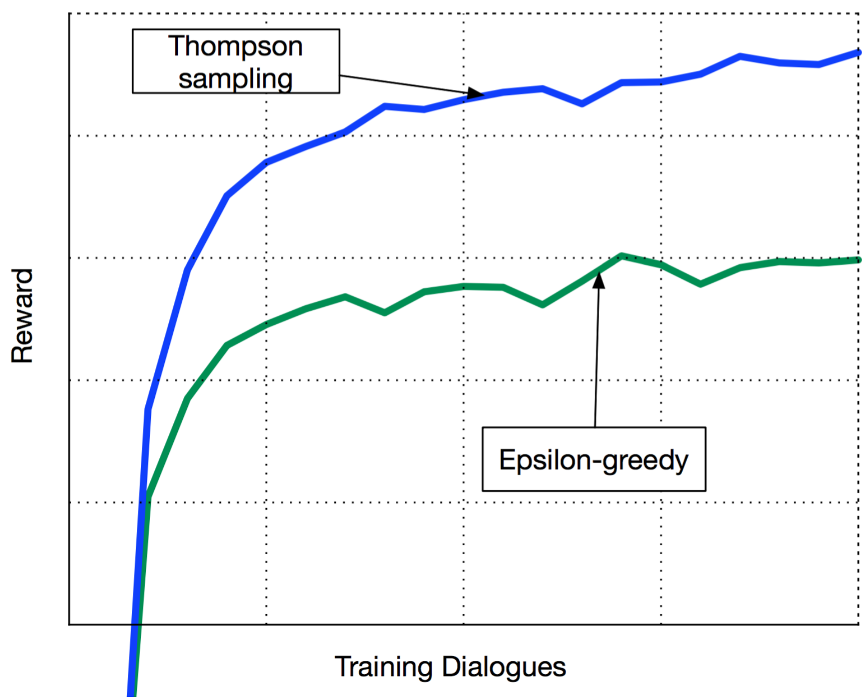
Active learning

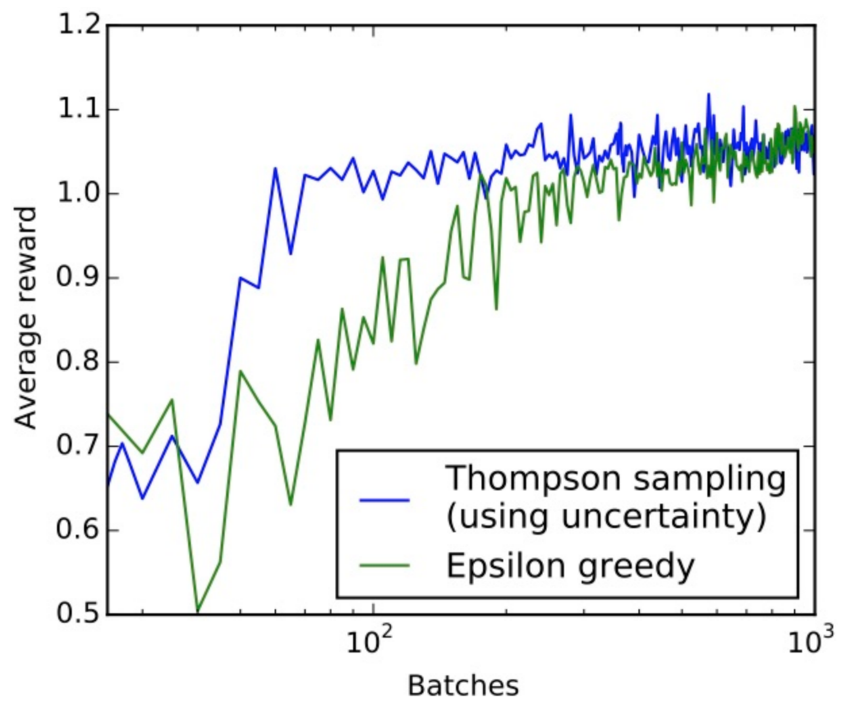
1. Contextual bandits,
2. Thompson Sampling (1933),

Active learning

1. Contextual bandits,
2. Thompson Sampling (1933),
3. Bayesian RL - Gasic et al. 2009, Gal 2016.

Thompson sampling vs ϵ-greedy approach





Thank you!