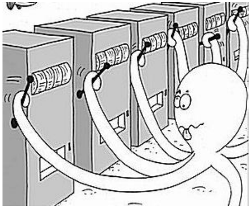
Bayesian optimization - part 2

Hrvoje Stojic

May 25, 2018

The multi-armed bandit (MAB) problem

Formulation



- A tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a (stationary) set of K actions/arms
- $\mathcal{R}^a(r) = P[r|a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $a_t \in \mathcal{A}$
- The environment generates a reward $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$

Regret

Regret

- The **action-value** is the mean reward for action a,
  $Q(a) = E[r|a]$

Regret

- The **action-value** is the mean reward for action a, $Q(a) = E[r|a]$

- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$

Regret

- The **action-value** is the mean reward for action a, $Q(a) = E[r|a]$
- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$
- The **cumulative regret** is the total opportunity loss $L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$

Regret

- The **action-value** is the mean reward for action a, $Q(a) = E[r|a]$
- The **optimal value** is $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$
- The **cumulative regret** is the total opportunity loss $L_t = E[\sum_{\tau=1}^{t} V^* - Q(a_\tau)]$
- Regret can be expressed in terms of counts and gaps:
  - The count $N_t(a)$ is expected number of selections for action a
  - The gap Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
  - The cumulative regret, stated differently:

$$L_t = \sum_{a \in \mathcal{A}} E[N_t(a)](V^* - Q(a)) = \sum_{a \in \mathcal{A}} E[N_t(a)]\Delta_a$$

Random exploration approaches

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$, by tracking the means

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

or

$$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \alpha(r_t - \hat{Q}_{t-1}(a))$$

Random exploration approaches

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$, by tracking the means

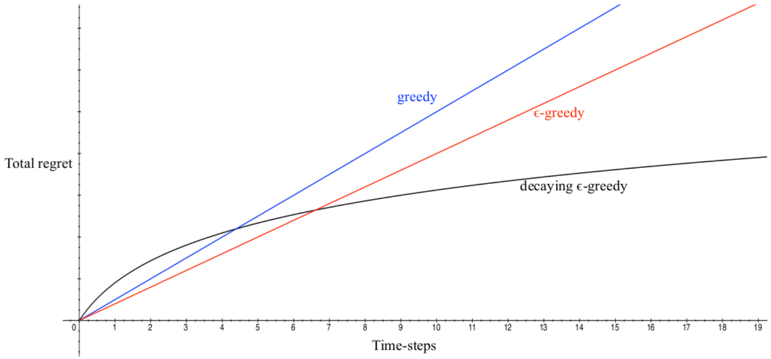
$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

or

$$\hat{Q}_t(a) = \hat{Q}_{t-1}(a) + \alpha(r_t - \hat{Q}_{t-1}(a))$$

- Three choice rules:
  - **greedy**: $a_t^* = \mathrm{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$
  - ϵ-**greedy**: With probability $1-\epsilon$ select $a_t^* = \mathrm{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$, with probability ϵ select a random action
  - **Softmax**: $P(a_t = a) = \frac{\exp(\hat{Q}_t(a)/\tau)}{\sum_{a'=1}^{K} \exp(\hat{Q}_t(a')/\tau)}$

Linear regret



Source: David Silver lectures

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t \to \infty} \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

- $\log t$ is the important bit, the second term is a constant, roughly task difficulty

Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

- $\log t$ is the important bit, the second term is a constant, roughly task difficulty
  - KL divergence says how similar the reward distributions of two arms are

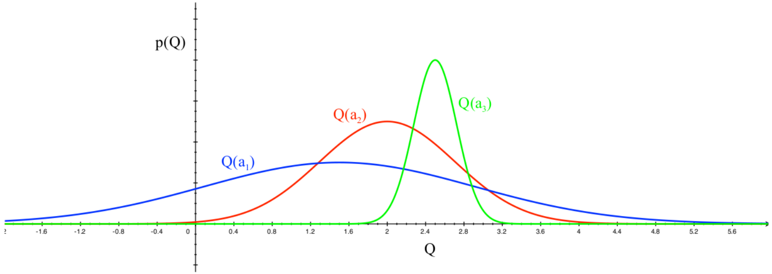
Lower bound on regret

- Asymptotic total regret is at least logarithmic in number of steps (Lai & Robbins, 1985)

$$\lim_{t\to\infty} \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

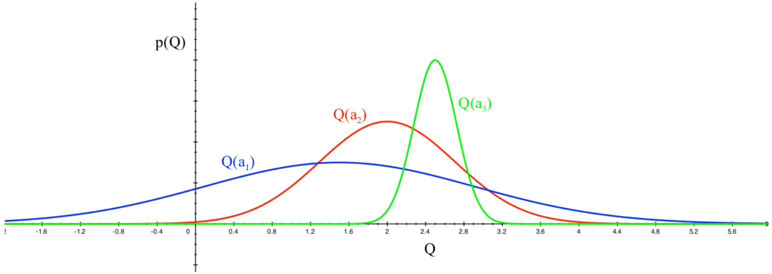
- $\log t$ is the important bit, the second term is a constant, roughly task difficulty
  - KL divergence says how similar the reward distributions of two arms are
  - The difference in expected rewards between the arms is described by the gap, Δ_a

Optimism in the face of uncertainty



Source: David Silver lectures

Optimism in the face of uncertainty



- any disadvantages?

Source: David Silver lectures

Optimistic initialization & ϵ_t-greedy

- ϵ-greedy with optimistic initialization
  - Simple idea: initialise $\hat{Q}(a)$ to a high value
  - Update rule:

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1}(a))$$

  - Everything else stays the same!

Optimistic initialization & ϵ_t-greedy

- ϵ-greedy with optimistic initialization
  - Simple idea: initialise $\hat{Q}(a)$ to a high value
  - Update rule:

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1}(a))$$

  - Everything else stays the same!

- Decaying ϵ_t-greedy (Auer, Cesa-Bianchi, Fischer, 2002)
  - Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
    - e.g. $\epsilon_t = \min\{1, \frac{c|\mathcal{A}|}{\min_a \Delta_a t}\}$
  - Has logarithmic asymptotic total regret (for known gaps)
  - In practice very good performance
  - But difficult to tune the decay

Upper Confidence Bounds (UCB)

- The main principle

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
  - Select action maximising Upper Confidence Bound (UCB)

$$a_t = \mathrm{argmax} \hat{Q}_t(a) + \hat{U}_t(a)$$

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
  - Select action maximising Upper Confidence Bound (UCB)

$$a_t = \text{argmax} \hat{Q}_t(a) + \hat{U}_t(a)$$

- UCB1 algorithm (Auer et al, 2002)

$$a_t = \text{argmax}_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Upper Confidence Bounds (UCB)

- The main principle
  - Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
  - Select action maximising Upper Confidence Bound (UCB)

$$a_t = \mathrm{argmax} \hat{Q}_t(a) + \hat{U}_t(a)$$

- UCB1 algorithm (Auer et al, 2002)

$$a_t = \mathrm{argmax}_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- Easily converted to a probabilistic version

UCB1 derivation

- Hoeffding's Inequality

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$

UCB1 derivation

- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0,1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$

- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$
  - As $t \rightarrow \infty$ we want a tendency to select the optimal action, so we reduce p as a function of time, e.g. $p = t^{-4}$

UCB1 derivation

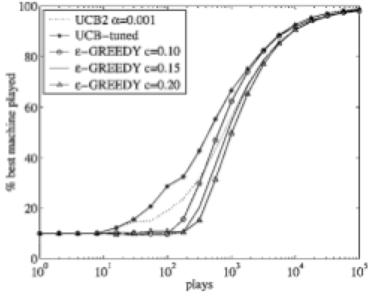
- Hoeffding's Inequality
  - Let $X_1, ..., X_t$ be IID random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$P[E[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

- When applied to bandit setting, conditioned on arm a, $P[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2N_t(a)U_t(a)^2}$
- Solving for $U_t(a)$, $U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$
  - As $t \to \infty$ we want a tendency to select the optimal action, so we reduce p as a function of time, e.g. $p = t^{-4}$
  - We arrive at

$$U_t(a) = \sqrt{\frac{2\log t}{N_t(a)}}$$

UCB comparison



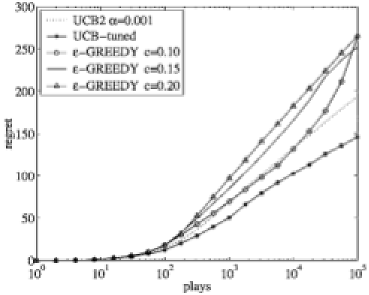


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Source: Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. Machine Learning, 47, 235-256.

Bayesian bandits

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R

- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R
- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$
- Use posterior instead of counts to guide exploration
  - Bayesian UCB, $a_t = \mathrm{argmax} \mu_a + \beta \sigma_a$
  - Probability matching, selects action a according to probability that a is the optimal action,

$$\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a | h_t]$$

Bayesian bandits

- So far we have made very few assumptions about the reward distribution R

- With Bayesian approach
  - We can exploit our prior knowledge of rewards, $P[R]$
  - We get full posterior distributions of rewards $P[R|h_t]$

- Use posterior instead of counts to guide exploration
  - Bayesian UCB, $a_t = \text{argmax} \mu_a + \beta \sigma_a$
  - Probability matching, selects action a according to probability that a is the optimal action,

$$\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|h_t]$$

- Wrong priors might cause issues.

Parametric Bayesian approach: Beta-Bernoulli bandit

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a
- We can fully describe f with parameter $\mathbf{w} \in (0, 1)^K$ so that $f_{\mathbf{w}}(a) = w_a$

Parametric Bayesian approach: Beta-Bernoulli bandit

- A generic probabilistic model parametrized by $\mathbf{w}$, with $\mathcal{D}$ denoting the data
- We can express our prior beliefs about the parameter values through $P[\mathbf{w}]$
- Posterior is then obtained by applying the Bayes rule,

$$P[\mathbf{w}|\mathcal{D}] = \frac{P[\mathcal{D}|\mathbf{w}]P[\mathbf{w}]}{P[\mathcal{D}]}$$

- Consider the MAB version where reward distribution of each arm follows Bernoulli distribution with unknown parameter $p \in (0, 1)$ with rewards, $r \in 0, 1$
- Reward of each arm is determined by function f that takes index of an arm $a \in 1, ..., K$ and returns parameter p_a
- We can fully describe f with parameter $\mathbf{w} \in (0, 1)^K$ so that $f_{\mathbf{w}}(a) = w_a$
- Observations are collected in $\mathcal{D}_t = \{(a_\tau, r_\tau)\}_\tau^t$ as a set of tuples, where a_τ identifies the arm and r_τ is the reward

Thompson Sampling for Beta-Bernoulli MAB problem

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha,\beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha,\beta)$$

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha, \beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha, \beta)$$

- With such conjugate prior we can efficiently compute the posterior,

$$P[\mathbf{w}|\mathcal{D}] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha + n_{,1}, \beta + n_{a,0})$$

  - $n_{,1}$ is a count of 1 outcomes whenever for arm a
  - $n_{a,0}$ is a count of 0 outcomes whenever for arm a

Thompson Sampling for Beta-Bernoulli MAB problem

- Classical choice for the prior is a conjugate to the Bernoulli likelihood, Beta distribution

$$P[\mathbf{w}|\alpha, \beta] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha, \beta)$$

- With such conjugate prior we can efficiently compute the posterior,

$$P[\mathbf{w}|\mathcal{D}] = \prod_{a=1}^{K} \mathrm{Beta}(w_a|\alpha + n_{,1}, \beta + n_{a,0})$$

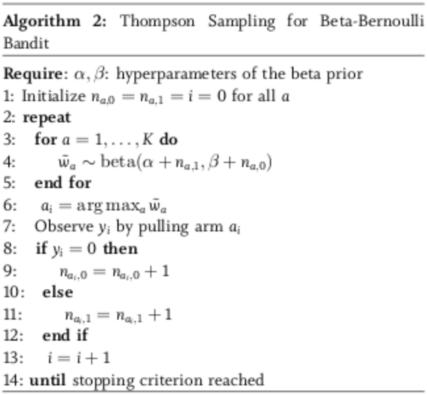
  - $n_{,1}$ is a count of 1 outcomes whenever for arm a
  - $n_{a,0}$ is a count of 0 outcomes whenever for arm a

- Thompson sampling (Thompson, 1933; Chapelle, Li, 2010)
  - Sample $\mathbf{w}'$ from each posterior and then maximize,

$$a_{t+1} = \mathrm{argmax}_a f_{\mathbf{w}'}(a), \mathrm{where}\ \mathbf{w}' \sim P[\mathbf{w}|\mathcal{D}_t]$$

  - Thompson sampling achieves Lai and Robbins lower bound!

Algorithm & Example



Algorithm 2: Thompson Sampling for Beta-Bernoulli Bandit

```
Require: $\alpha, \beta$: hyperparameters of the beta prior
1: Initialize $n_{a,0} = n_{a,1} = i = 0$ for all $a$
2: repeat
3:   for $a = 1, \dots, K$ do
4:     $\tilde{w}_a \sim \text{beta}(\alpha + n_{a,1}, \beta + n_{a,0})$
5:   end for
6:   $a_i = \arg\max_a \tilde{w}_a$
7:   Observe $y_i$ by pulling arm $a_i$
8:   if $y_i = 0$ then
9:     $n_{a_i,0} = n_{a_i,0} + 1$
10:  else
11:    $n_{a_i,1} = n_{a_i,1} + 1$
12:  end if
13:  $i = i + 1$
14: until stopping criterion reached
```

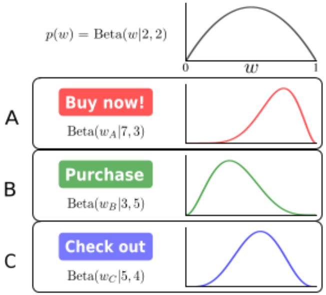


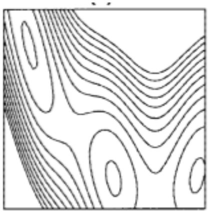
Fig. 2. *Example of the beta-Bernoulli model for A/B testing. Three different buttons are being tested with various colors and text. Each option is given two successes (click-throughs) and two failures as a prior (top). As data are observed, each option updates its posterior over w. Option A is the current best with five successes and only one observed failure.*

Source: Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE, 104(1), 148-175.

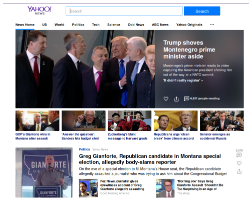
Contextual Multi-armed Bandit (CMAB) problem

Formulation

- A tuple $\langle \mathcal{A}, \mathcal{S}, \mathcal{R} \rangle$
- We introduce the state representation again
- $\mathcal{A}$ is a set of actions/arms
- $\mathcal{S} = P[s]$ is an unknown distribution over states (or “contexts”)
- $\mathcal{R}^a(r) = P[r|s,a]$ is an unknown probability distribution over rewards
- At each step t
  - The environment generates state $s_t \sim \mathcal{S}$
  - The agent selects an action $a_t \in \mathcal{A}$
  - The environment generates a reward $r_t \sim \mathcal{R}^{a_t}_{s_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$



CLiCK HERE!
Inflatable Fun For Everyone!
Bounce Magic
677-JUMP
explore&more
a children's museum
click
Pumpkinville
Fall Family Fun!
The Buffalo & Erie County Public Library presents
The Big Read and Find the Falcon... A Contest for All Ages!
BIG READ
Click here for more information


YAHOO! NEWS

Search

News Home · US · World · Politics · Tech · Science · Odd News · ABC News · Yahoo Originals · ···

Trump shoves Montenegro prime minister aside

Montenegro's prime minister reacts to video capturing the American president shoving him out of the way at a NATO summit.

'It didn't really register' »

6,837 people reacting

GOP's Gianforte wins in Montana after assault

'Answer the question': Sanders hits budget chief

Zuckerberg's blunt message to Harvard grads

Republicans urge 'clean break' from climate accord

Senator emerges as accidental Russia

Politics · Yahoo News

Greg Gianforte, Republican candidate in Montana special election, allegedly body-slams reporter

On the eve of a special election to fill Montana's House seat, the Republican candidate allegedly assaulted a journalist who was trying to ask him about the Congressional Budget

3,578

Fox News journalist gives eyewitness account of Greg Gianforte allegedly assaulting
Good Morning America

'Morning Joe' Says Greg Gianforte Assault 'Shouldn't Be Too Surprising in an Age of
The Wrap

Bayesian nonparametric approach: Gaussian Processes (GP)

- Inducing a Gaussian prior over functions:

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

- Here, $m(\mathbf{x})$ is a mean function modeling the expected output of the function and $k(\mathbf{x}, \mathbf{x}')$ is a kernel function modeling the covariance between different points.

$$m(\mathbf{x}) = E[f(\mathbf{x})]$$

and

$$k(x, x') = E[(f(\mathbf{x}) - m(x))(f(\mathbf{x}') - m(x'))]$$

- The choice of an appropriate kernel is normally based on assumptions such as smoothness and likely patterns to be expected in the data.

Gaussian Processes (GP)

Gaussian Processes (GP)

- Popular choice is the squared exponential (also called Gaussian or Radial Basis Function) kernel.

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\lambda^2}\right)$$

Gaussian Processes (GP)

- Popular choice is the squared exponential (also called Gaussian or Radial Basis Function) kernel.

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\lambda^2}\right)$$

- Correlation between two points decays according to a power function in dependency of the distance between the two points
- Covariance is symmetric, that is that only the distance between two points matters, but not the direction.

Gaussian Processes (GP)

- Popular choice is the squared exponential (also called Gaussian or Radial Basis Function) kernel.

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\lambda^2}\right)$$

- Correlation between two points decays according to a power function in dependency of the distance between the two points
- Covariance is symmetric, that is that only the distance between two points matters, but not the direction.

- Good for smooth functions, hyperparameters λ (called the length-scale) and σ^2 (the noise constant) are normally optimized by using the marginal likelihood.

Gaussian Processes (GP)

- This implies the aforementioned distribution over functions as we can easily generate samples for new input points at location $X_\star$.

$$\mathbf{f}_\star \sim \mathcal{N}(0, K\left(X_\star, X_\star\right))$$

Gaussian Processes (GP)

- This implies the aforementioned distribution over functions as we can easily generate samples for new input points at location $X_\star$.

$$\mathbf{f}_\star \sim \mathcal{N}(0, K\left(X_\star, X_\star\right))$$

- Given observations $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ with a noise level σ, we can draw new predictions from our function $\mathbf{f}_\star$ for inputs $X_\star$ as described below.

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, X_\star) \\ K(X_\star, X) & K(X_\star, X_\star) \end{bmatrix}\right)$$

Gaussian Processes (GP)

- Treat a function as a vector of infinite size.
- We can simply draw outputs for finite points by using a multivariate normal distribution with a covariance matrix generated by our kernel.
- Calculating the expectation of the Gaussian Process at the new points is then

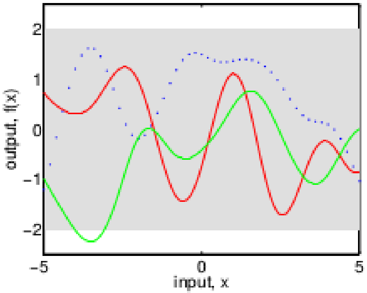
$$\mathbf{f}_\star | X, \mathbf{y}, X_\star \sim \mathcal{N}(\bar{\mathbf{f}}_\star, \mathsf{cov}(\mathbf{f}_\star))$$

- Predictions for new points are generated based on the expected mean value and covariance function of the posterior Gaussian Process.

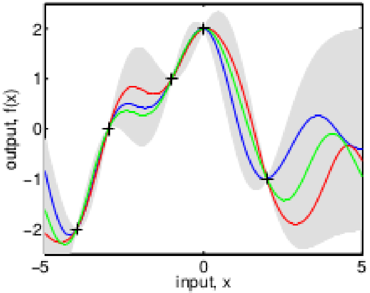
$$\mathbb{E}[\mathbf{f}_\star | X, \mathbf{y}, X_\star] = K(X_\star, X)[K(X, X) + \sigma^2 I]^{-1}\mathbf{y}$$

$$\mathsf{cov}(\mathbf{f}_\star) = K(X_\star, X_\star) - K(X_\star, X)[K(X, X) + \sigma^2 I]^{-1} K(X, X_\star)$$

Drawing from the GP prior and posterior



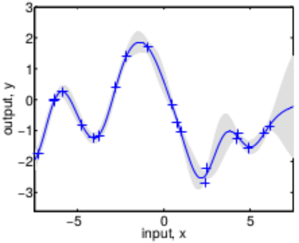
(a), prior



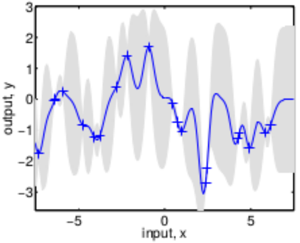
(b), posterior

Source: Rasmussen, Williams (2006)

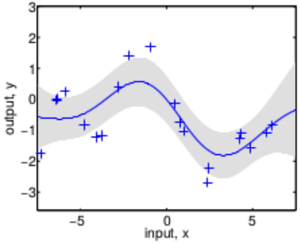
Dependence on hyperparameters



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

Source: Rasmussen, Williams (2006)

Matern kernel

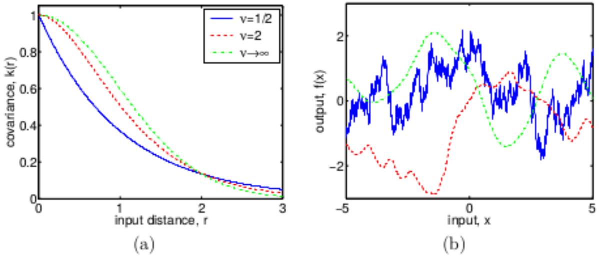
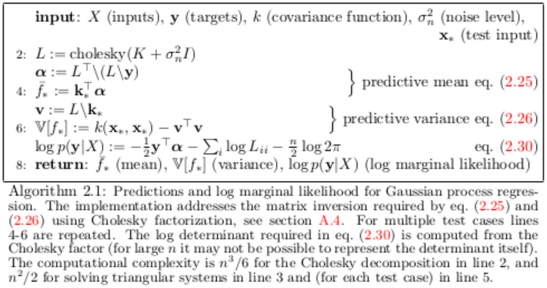


Figure 4.1: Panel (a): covariance functions, and (b): random functions drawn from Gaussian processes with Matérn covariance functions, eq. (4.14), for different values of ν, with $\ell = 1$. The sample functions on the right were obtained using a discretization of the x-axis of 2000 equally-spaced points.

Source: Rasmussen, Williams (2006)

GP algorithm



```
   input: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function), $\sigma_n^2$ (noise level),
                                                                        $\mathbf{x}_*$ (test input)
2: $L := \text{cholesky}(K + \sigma_n^2 I)$
   $\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$                              } predictive mean eq. (2.25)
4: $\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
   $\mathbf{v} := L \backslash \mathbf{k}_*$                                  } predictive variance eq. (2.26)
6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
   $\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$          eq. (2.30)
8: return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)
```

Algorithm 2.1: Predictions and log marginal likelihood for Gaussian process regression. The implementation addresses the matrix inversion required by eq. (2.25) and (2.26) using Cholesky factorization, see section A.4. For multiple test cases lines 4-6 are repeated. The log determinant required in eq. (2.30) is computed from the Cholesky factor (for large n it may not be possible to represent the determinant itself). The computational complexity is $n^3/6$ for the Cholesky decomposition in line 2, and $n^2/2$ for solving triangular systems in line 3 and (for each test case) in line 5.

Source: Rasmussen, Williams (2006)

Computational considerations and alternative regression models

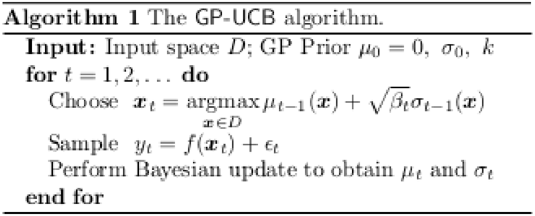
Computational considerations and alternative regression models

- Although we have analytic expressions, exact inference in GP regression has a cost of $\mathcal{O}(n^3)$
- Due to inversion of the covariance matrix
  - in practice using Cholesky decomposition reduces the cost to $\mathcal{O}(n^2)$
  - however, in BO we have to repeat it in every iteration as hyperparameters change
- With large budgets the cost might be prohibitive
  - Sparse GPs

Computational considerations and alternative regression models

- Although we have analytic expressions, exact inference in GP regression has a cost of $\mathcal{O}(n^3)$
- Due to inversion of the covariance matrix
  - in practice using Cholesky decomposition reduces the cost to $\mathcal{O}(n^2)$
  - however, in BO we have to repeat it in every iteration as hyperparameters change
- With large budgets the cost might be prohibitive
  - Sparse GPs
- Other alternative regression models?
  - Random Forests (SMAC, TPE)
  - Variance in predictions of the trees used as a proxy for uncertainty
  - Poor extrapolators
  - GPs are relatively bad as well, but they revert to prior far from the inputs, while RF is unrealistically confident

GP-UCB



Algorithm 1 The GP-UCB algorithm.

```
**Input:** Input space $D$; GP Prior $\mu_0 = 0,\ \sigma_0,\ k$
**for** $t = 1, 2, \ldots$ **do**
    Choose $x_t = \underset{x \in D}{\operatorname{argmax}}\, \mu_{t-1}(x) + \sqrt{\beta_t}\sigma_{t-1}(x)$
    Sample $y_t = f(x_t) + \epsilon_t$
    Perform Bayesian update to obtain $\mu_t$ and $\sigma_t$
**end for**
```

- Regret bounds, Srinivas et al (2010)
- Expected improvement, Probability of improvement, Entropy search, predictive entropy search, Portfolios of acquisition functions (Hedge, Entropy search portfolio)

Expected Improvement

- $EI(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - f(\hat{\mathbf{x}})\right\}\right]$, where $\hat{\mathbf{x}}$ is the current optimal set of hyperparameters.

Expected Improvement

- $EI(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - f(\hat{\mathbf{x}})\right\}\right]$, where $\hat{\mathbf{x}}$ is the current optimal set of hyperparameters.
- We can actually compute EI expectation under the GP model

$$EI(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\hat{\mathbf{x}}))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{1}$$

$$Z = \frac{\mu(\mathbf{x}) - f(\hat{\mathbf{x}})}{\sigma(\mathbf{x})} \tag{2}$$

where $\Phi(z)$, and $\phi(z)$, are the cumulative distribution and probability density function of the (multivariate) standard normal distribution.

Expected Improvement

- $EI(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - f(\hat{\mathbf{x}})\right\}\right]$, where $\hat{\mathbf{x}}$ is the current optimal set of hyperparameters.
- We can actually compute EI expectation under the GP model

$$EI(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\hat{\mathbf{x}}))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{1}$$

$$Z = \frac{\mu(\mathbf{x}) - f(\hat{\mathbf{x}})}{\sigma(\mathbf{x})} \tag{2}$$

where $\Phi(z)$, and $\phi(z)$, are the cumulative distribution and probability density function of the (multivariate) standard normal distribution.

1. EI is high when the (posterior) expected value $\mu(\mathbf{x})$ is higher than the current best value $f(\hat{\mathbf{x}})$; or
2. EI is high when the uncertainty $\sigma(\mathbf{x})$ around the point $\mathbf{x}$ is high.

Acquisition functions illustration

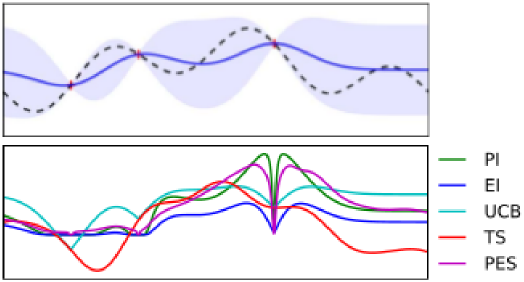


Fig. 5. *Visualization of the surrogate regression model and various acquisition functions. (Top) The true objective function is shown as a dashed line and the probabilistic regression model is shown as a blue line with a shaded region delimiting the $2\sigma_n$ credible intervals. Finally, the observations are shown as red crosses. (Bottom) Four acquisition functions are shown. In the case of PI, the optimal mode is much closer to the best observation as in the alternative methods, which explains its greedy behavior. In contrast, the randomization in TS allows it to explore more aggressively.*

Source: Shahriari et al (2016)

Optimizing acquisition functions

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach
  - Difficult to asses the performance and there are few theoretical guarantees

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach
  - Difficult to asses the performance and there are few theoretical guarantees

- Recent development: Optimistic optimization

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach
  - Difficult to asses the performance and there are few theoretical guarantees

- Recent development: Optimistic optimization
  - Use the same optimism in the face of uncertainty on acquisition function optimization level as well

Optimizing acquisition functions

- Only useful if cheap to evaluate relative to objective f
- Acquisition functions are often multimodal, not a trivial task

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach
  - Difficult to asses the performance and there are few theoretical guarantees

- Recent development: Optimistic optimization
  - Use the same optimism in the face of uncertainty on acquisition function optimization level as well
  - E.g. Wang, SShakibi, Jin and de Freitas (2014) propose BamSOO: shrink the region that we examine in every iteration to the most promising regions

Optimizing acquisition functions: Optimistic optimization (BamSOO)

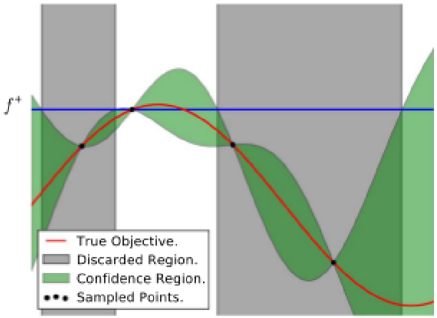
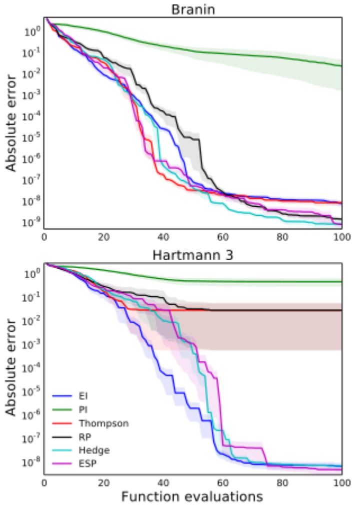


Fig. 7. *Conditioned on the unknown objective function (red) lying between the surrogate confidence bounds (green region) with high probability, we can discard regions of the space where the upper bound is lower than the best lower bound encountered thus far. Figure from [43].*

Source: Shahriari et al (2016)

Acquisition functions comparison



- Shahriari et al (2016) conclude choice of acquisition function matters less than the regression model

Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)

Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points

Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data

Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: cost and radial basis parameter

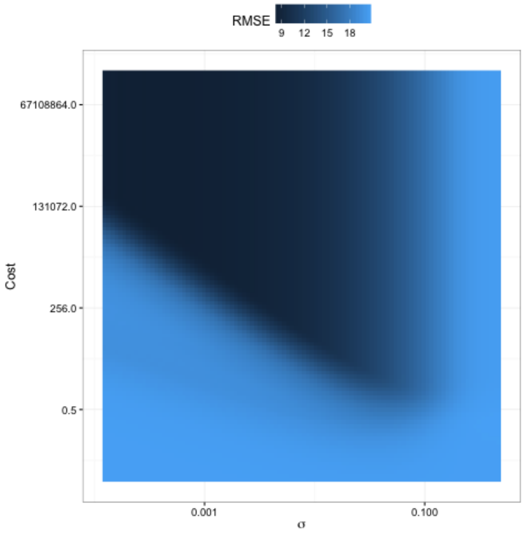
Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: cost and radial basis parameter
- example: Revolutionanalytics.com

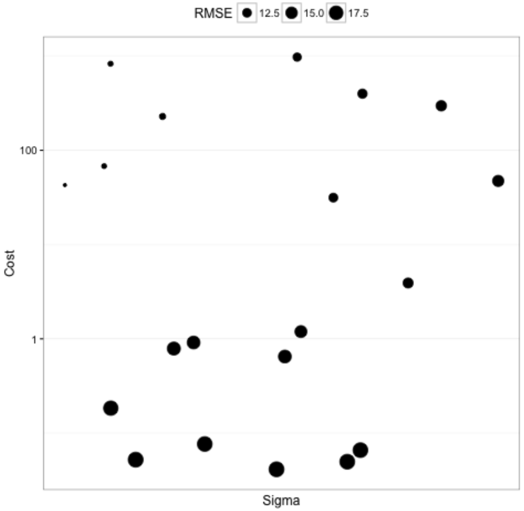
Example: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: cost and radial basis parameter
- example: Revolutionanalytics.com
  - using kernlab and rBayesianOptimization

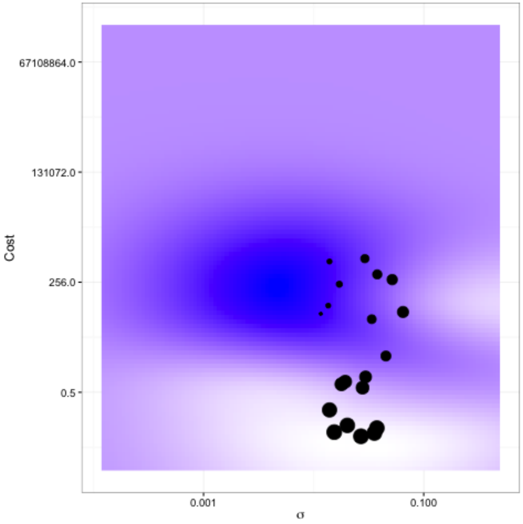
Example: RMSE surface



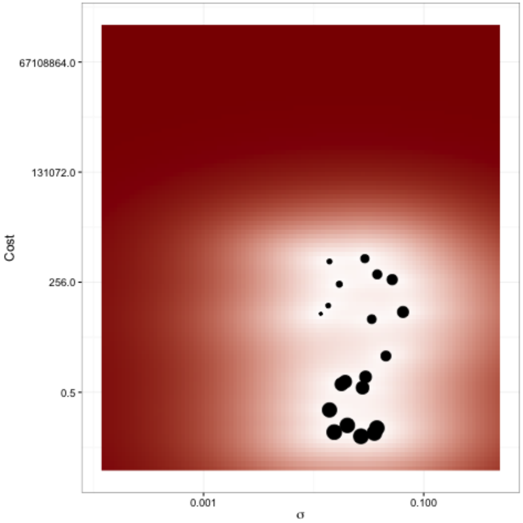
Example: Random search



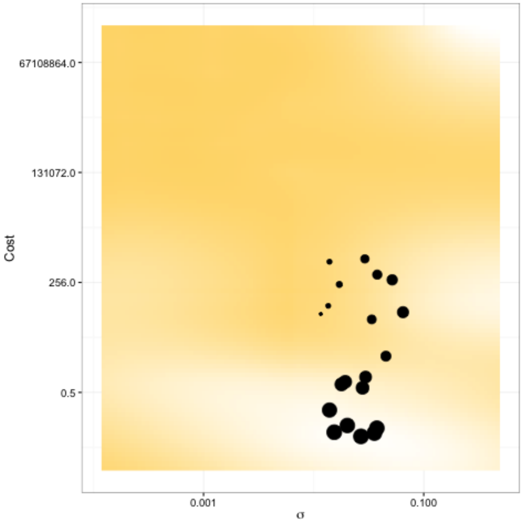
Example: GP predictive mean (based on initial random search)



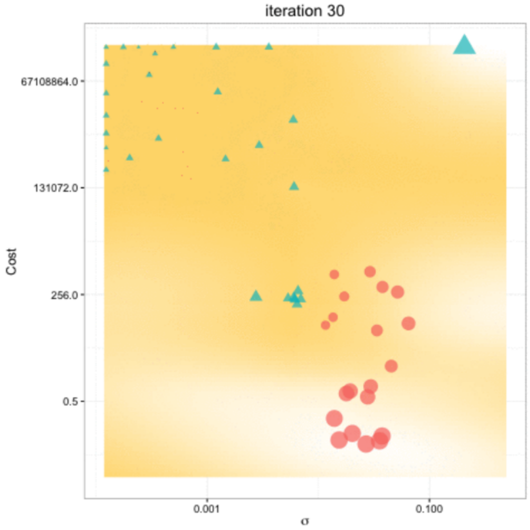
Example: GP predictive variance (based on initial random search)



Example: GP-UCB (based on initial random search)



Example: GP-UCB solution after 30 evaluations



Some extensions

Full Bayesian treatment

Full Bayesian treatment

- GP with ARD would typically have $D+3$ hyperparameters: D length scales, constant mean, noise variance σ_n^2 and signal variance σ_f^2

Full Bayesian treatment

- GP with ARD would typically have $D + 3$ hyperparameters: D length scales, constant mean, noise variance σ_n^2 and signal variance σ_f^2
- Instead of obtaining the point estimates of hyperparameters by optimizing the marginal likelihood, we should take into account uncertainty about GP's parameters when optimizing the acquisition function

Full Bayesian treatment

- GP with ARD would typically have $D + 3$ hyperparameters: D length scales, constant mean, noise variance σ_n^2 and signal variance σ_f^2
- Instead of obtaining the point estimates of hyperparameters by optimizing the marginal likelihood, we should take into account uncertainty about GP's parameters when optimizing the acquisition function
- Feasible with some functions like EI and PI

Full Bayesian treatment

- GP with ARD would typically have $D+3$ hyperparameters: D length scales, constant mean, noise variance σ_n^2 and signal variance σ_f^2
- Instead of obtaining the point estimates of hyperparameters by optimizing the marginal likelihood, we should take into account uncertainty about GP's parameters when optimizing the acquisition function
- Feasible with some functions like EI and PI

- Integrated acquisition function

$$\hat{a}(\mathbf{x}; \mathbf{x}_t, y_t) = \int a(\mathbf{x}; \{\mathbf{x}_t, y_t\}, \theta) P[\theta | \{\mathbf{x}_t, y_t\}^t] d\theta$$

Full Bayesian treatment

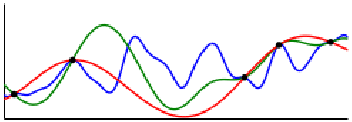
- GP with ARD would typically have $D + 3$ hyperparameters: D length scales, constant mean, noise variance σ_n^2 and signal variance σ_f^2
- Instead of obtaining the point estimates of hyperparameters by optimizing the marginal likelihood, we should take into account uncertainty about GP's parameters when optimizing the acquisition function
- Feasible with some functions like EI and PI

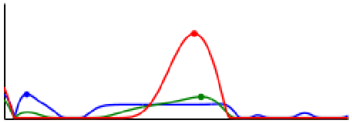
- Integrated acquisition function

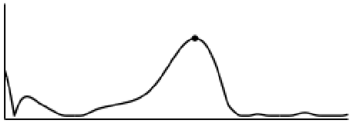
$$\hat{a}(\mathbf{x}; \mathbf{x}_t, y_t) = \int a(\mathbf{x}; \{\mathbf{x}_t, y_t\}, \theta) P[\theta | \{\mathbf{x}_t, y_t\}^t] d\theta$$

- Monte Carlo estimate, can be acquired efficiently with slice sampling (see Murray & Adams, 2010)

Example


(a) Posterior samples under varying hyperparameters


(b) Expected improvement under varying hyperparameters


(c) Integrated expected improvement

Source: Snoek et al (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In Advances in Neural Information Processing Systems (pp. 2951–2959).

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible
- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

- Snoek et al (2012): *expected improvement per second*

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

- Snoek et al (2012): *expected improvement per second*
  - Duration function is also not known

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

- Snoek et al (2012): *expected improvement per second*
  - Duration function is also not known
  - $c(\mathbf{x}) : \mathcal{X} \rightarrow R^+$

Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

- Snoek et al (2012): *expected improvement per second*
  - Duration function is also not known
  - $c(\mathbf{x}) : \mathcal{X} \to R^+$
  - We can use the GP machinery to estimate $c()$ as well

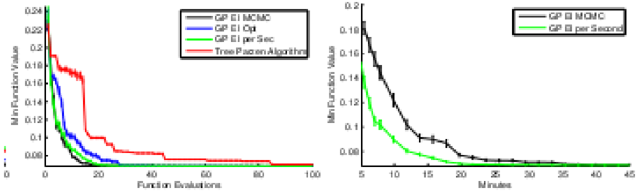
Taking into account modeling costs

- So far we have been concerned with finding good hyperparameters in fewest steps/evaluations possible

- Sometimes a better goal is to minimize duration, wallclock time, not the number of function evaluations
- Different combinations of hyperparameters can lead to very different evaluation times

- Snoek et al (2012): *expected improvement per second*
  - Duration function is also not known
  - $c(\mathbf{x}) : \mathcal{X} \to R^+$
  - We can use the GP machinery to estimate $c()$ as well
  - Combine the predicted objective cost and duration

Example



Source: Snoek et al (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In Advances in Neural Information Processing Systems (pp. 2951–2959).

Parallelization

Parallelization

- if we are concerned with wallclock time, then a natural question is

Parallelization

- if we are concerned with wallclock time, then a natural question is
- what $\mathbf{x}$ should be evaluated next, even while a set of points is being evaluated?

Parallelization

- if we are concerned with wallclock time, then a natural question is
- what $\mathbf{x}$ should be evaluated next, even while a set of points is being evaluated?
- Ideally, we would do some planning (information state space search ala Gittins indices), but they are usually intractable

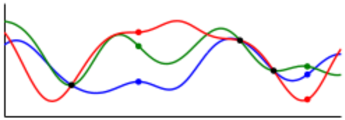
Parallelization

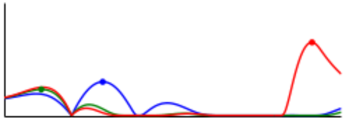
- if we are concerned with wallclock time, then a natural question is
- what $\mathbf{x}$ should be evaluated next, even while a set of points is being evaluated?
- Ideally, we would do some planning (information state space search ala Gittins indices), but they are usually intractable
- Snoek et al (2012) propose to compute MC estimates of the acquisition function under different possible results from pending function evaluations

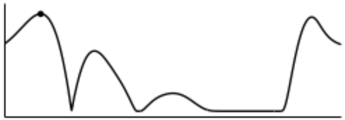
Parallelization

- if we are concerned with wallclock time, then a natural question is
- what $\mathbf{x}$ should be evaluated next, even while a set of points is being evaluated?
- Ideally, we would do some planning (information state space search ala Gittins indices), but they are usually intractable
- Snoek et al (2012) propose to compute MC estimates of the acquisition function under different possible results from pending function evaluations
- With function like EI we can leverage Gaussian integration property

Example


(a) Posterior samples after three data


(b) Expected improvement under three fantasies


(c) Expected improvement across fantasies

Source: Snoek et al (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In Advances in Neural Information Processing Systems (pp. 2951–2959).

Other applications of BO

Tackling any other (C)MAB problem

- Ad placement:
  - Placing an ad that has been shown to attract the most clicks (exploiting)
  - Or placing a different ad that we know less about, that might attract more clicks (exploring)
- Recommender system
  - People are interested in true exploration, not only seeing what other similar people have looked for
  - In some setups, like news recommendation, choice sets are changing all the time, RL approach is more suitable
- Learning user preferences
  - Market research, finding an optimal combination of features in a new product
  - E.g. tuning the recommender system so it is customized for each user