Lecture 5: Contextual multi-armed bandits

Hrvoje Stojic

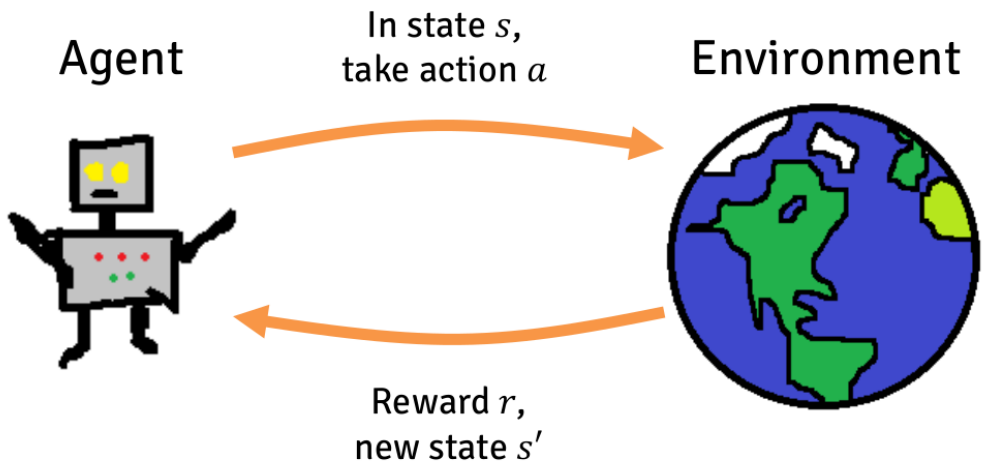
May 10, 2021

Contextual Multi-armed Bandit (CMAB) problem

Contextual Multi-armed Bandit (CMAB) problem

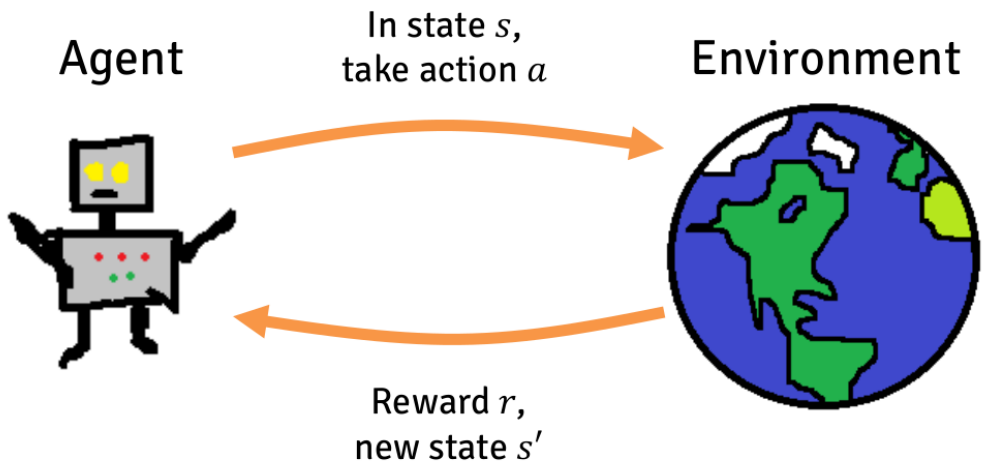
Still a subclass of reinforcement learning problems, but richer

Still a subclass of reinforcement learning problems, but richer



Learning to
- maximize reward
- in a reactive environment
- under partial feedback

Still a subclass of reinforcement learning problems, but richer



Learning to
- maximize reward
- in a reactive environment
- under partial feedback

Two key challenges of RL:

1. Dealing with long-term effects of actions
2. Dealing with uncertainty due to partial feedback

Formulation

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

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 $r_t \sim \mathcal{R}_{s_t}^{a_t}$

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 $r_t \sim \mathcal{R}_{s_t}^{a_t}$

- The goal is still to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$
- By devising a policy that maps context into actions, $\pi(a|s)$

Displaying ads most likely to be clicked

Source: Me, buying a keyboard to alleviate my wrist pain :)

Choosing frontpage stories most likely to be read

Source: theguardian.com, May 6, 2020

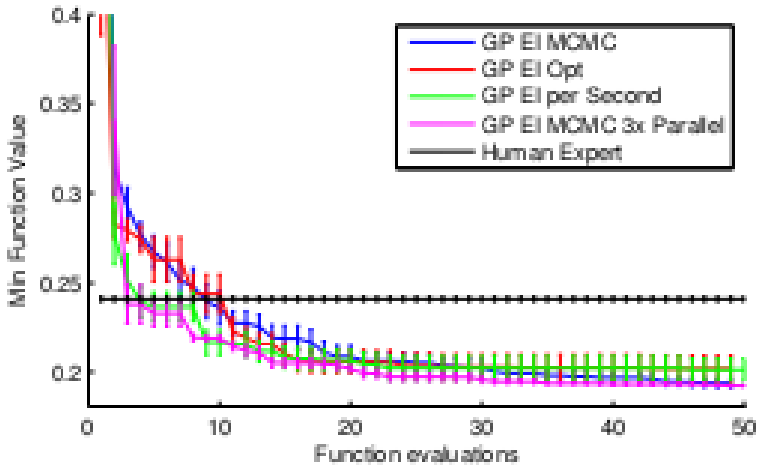
Where to take a bite?

Source: Culinary Backstreets, Federal

Optimizing hyperparameters

Source: Snoek et al 2012; Bergstra et al 2011

Optimizing hyperparameters

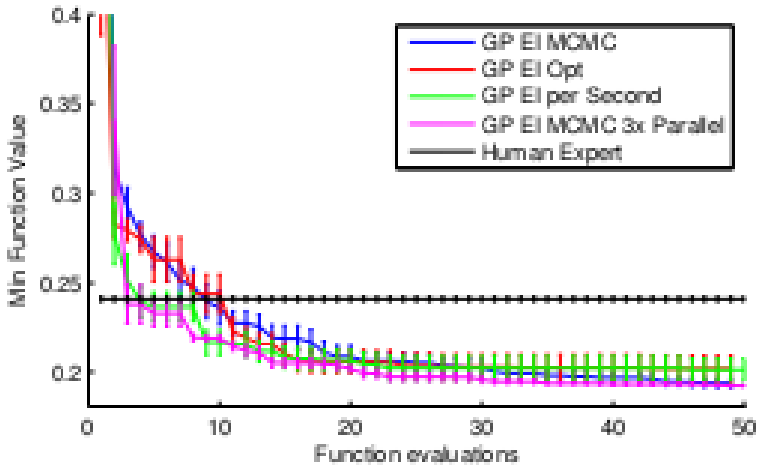


	convex	MRBI
TPE	**14.13** ±0.30 %	**44.55** ±0.44%
GP	16.70 ± 0.32%	47.08 ± 0.44%
Manual	18.63 ± 0.34%	47.39 ± 0.44%
Random	18.97 ± 0.34 %	50.52 ± 0.44%

Table 2: The test set classification error of the best model found by each search algorithm on each problem. Each search algorithm was allowed up to 200 trials. The manual searches used 82 trials for **convex** and 27 trials **MRBI**.

Source: Snoek et al 2012; Bergstra et al 2011

Optimizing hyperparameters



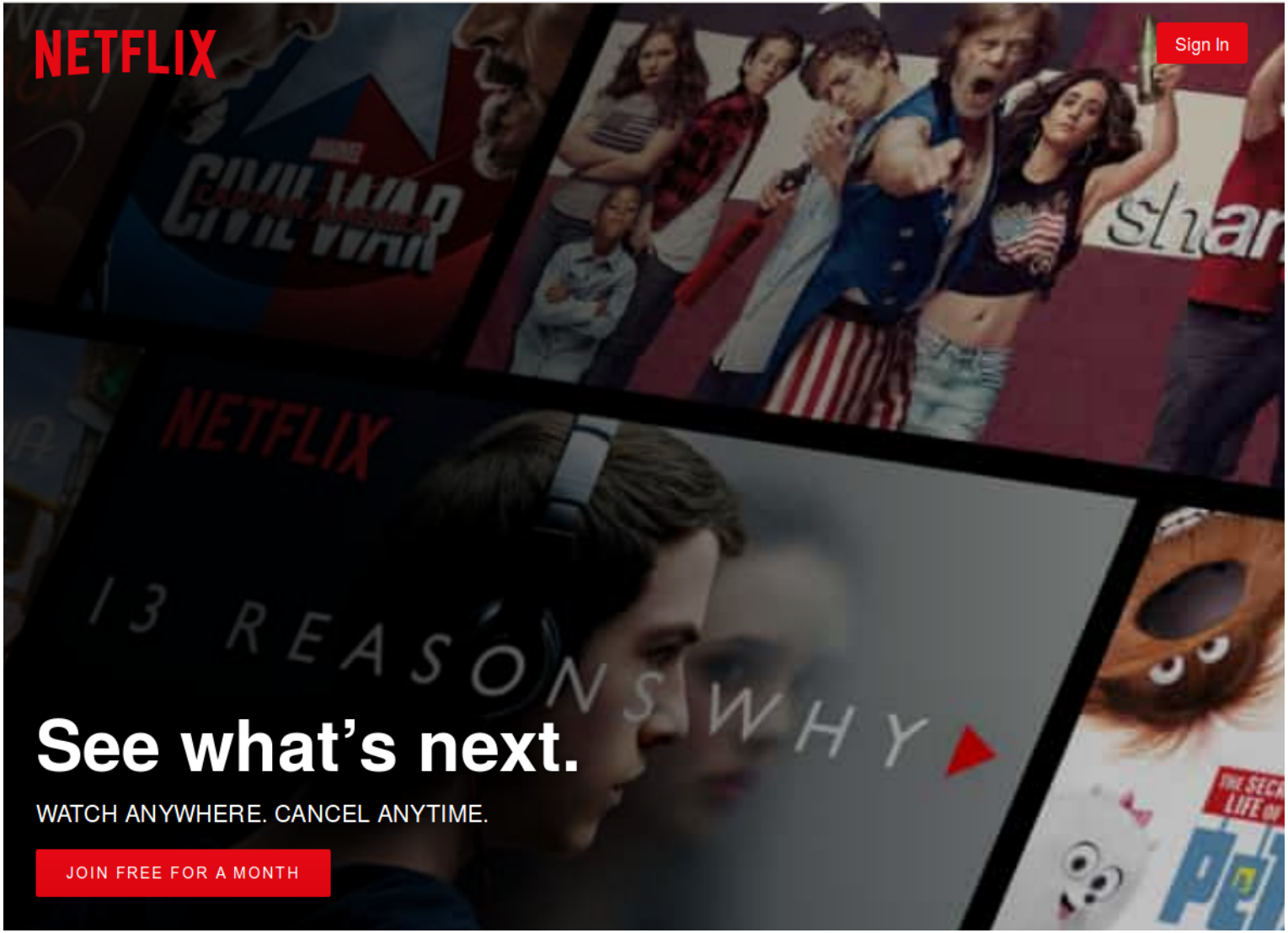
	convex	MRBI
TPE	**14.13** ±0.30 %	**44.55** ±0.44%
GP	16.70 ± 0.32%	47.08 ± 0.44%
Manual	18.63 ± 0.34%	47.39 ± 0.44%
Random	18.97 ± 0.34 %	50.52 ± 0.44%

Table 2: The test set classification error of the best model found by each search algorithm on each problem. Each search algorithm was allowed up to 200 trials. The manual searches used 82 trials for **convex** and 27 trials **MRBI**.

- CIFAR 10: state of the art was test error of 18%, they achieved 14.98%
- MNIST rotated background images

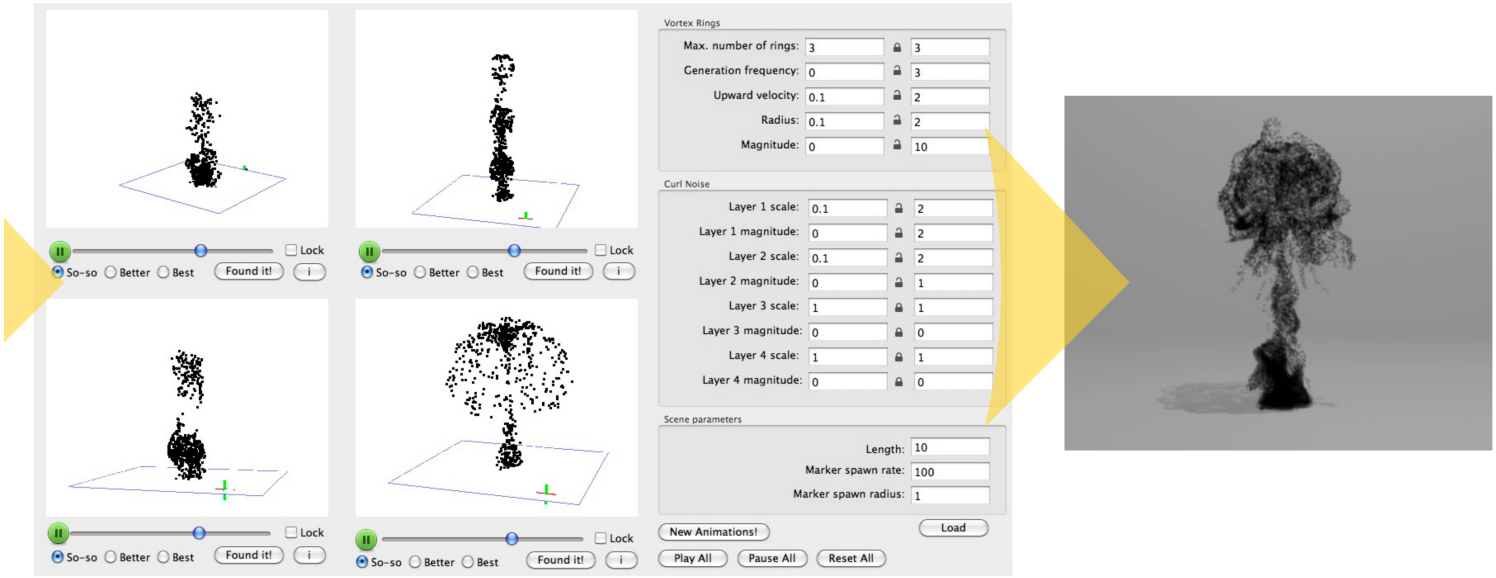
Source: Snoek et al 2012; Bergstra et al 2011

Preference learning



Source: Netflix webpage

Interactive user interfaces



Source: Brochu, Brochu & de Freitas (2010)

Learning with context

Learning with context

- Before any learning happens we (agent designers) choose a *form* of the policy
  - e.g. neural network, decision trees, ridge regression
  - this defines a *policy space*, Π, what policies we can possibly learn

Learning with context

- Before any learning happens we (agent designers) choose a *form* of the policy
  - e.g. neural network, decision trees, ridge regression
  - this defines a *policy space*, Π, what policies we can possibly learn
- Assumption is, $\exists \pi \in \Pi$ that yields high rewards

Learning with context

- Before any learning happens we (agent designers) choose a *form* of the policy
  - e.g. neural network, decision trees, ridge regression
  - this defines a *policy space*, Π, what policies we can possibly learn
- Assumption is, $\exists \pi \in \Pi$ that yields high rewards
- We want to learn through experimentation to do almost as well as best π
- Typically we want Π to be large, but finite
- This allows us to learn complex behaviors, expressive policies

Learning with context

- Before any learning happens we (agent designers) choose a *form* of the policy
  - e.g. neural network, decision trees, ridge regression
  - this defines a *policy space*, Π, what policies we can possibly learn
- Assumption is, $\exists \pi \in \Pi$ that yields high rewards
- We want to learn through experimentation to do almost as well as best π
- Typically we want Π to be large, but finite
- This allows us to learn complex behaviors, expressive policies
- Challenges:
  - Π can be extremely large
  - When action selected, only observe reward for policies that would have chosen same action
  - While we need to be learning about **all** policies, simultaneously performing as well as the best

What we will not cover

What we will not cover

- Full information setup
  - Agent has access to all rewards
  - Follow-the-leader
  - Hedging

What we will not cover

- Full information setup
  - Agent has access to all rewards
  - Follow-the-leader
  - Hedging
- Important theoretical ideas and development
- But not of practical use
  - Algorithms assume oracles
  - Access to counterfactuals

What we will not cover

- Full information setup
  - Agent has access to all rewards
  - Follow-the-leader
  - Hedging
- Important theoretical ideas and development
- But not of practical use
  - Algorithms assume oracles
  - Access to counterfactuals
- Some topics in contextual bandits with partial feedback
  - Bandits with expert advice
  - Exp4 (Exp3 with experts)

Stochastic linear bandits and LinUCB

Stochastic linear bandits and LinUCB

Stochastic contextual bandits

Stochastic contextual bandits

- Reward is now a combination of arm and context (state)

$$r_t = f(c_t, a_t) + \epsilon_t$$

  - where $f : C \times [K] \to \mathbb{R}$ is (unknown) reward function and ϵ_t is zero-mean IID noise term

Stochastic contextual bandits

- Reward is now a combination of arm and context (state)

$$r_t = f(c_t, a_t) + \epsilon_t$$

  - where $f : C \times [K] \to \mathbb{R}$ is (unknown) reward function and ϵ_t is zero-mean IID noise term

- Regret is $L_T = E[\sum_{t=1}^{T} \max_{a \in K} f(c_t, a) - \sum_{t=1}^{T} r_t]$

Stochastic contextual bandits

- Reward is now a combination of arm and context (state)

$$r_t = f(c_t, a_t) + \epsilon_t$$

  - where $f : C \times [K] \to \mathbb{R}$ is (unknown) reward function and ϵ_t is zero-mean IID noise term

- Regret is $L_T = E[\sum_{t=1}^{T} \max_{a \in K} f(c_t, a) - \sum_{t=1}^{T} r_t]$
- Poor man's contextual bandit algorithm
  - Assume C is finite
  - Assign bandit to each $c \in C$ and learn action values
  - Problem if C (and K) is very large

Stochastic contextual bandits

- Reward is now a combination of arm and context (state)

$$r_t = f(c_t, a_t) + \epsilon_t$$

  - where $f : C \times [K] \to \mathbb{R}$ is (unknown) reward function and ϵ_t is zero-mean IID noise term

- Regret is $L_T = E[\sum_{t=1}^{T} \max_{a \in K} f(c_t, a) - \sum_{t=1}^{T} r_t]$
- Poor man's contextual bandit algorithm
  - Assume C is finite
  - Assign bandit to each $c \in C$ and learn action values
  - Problem if C (and K) is very large
- How to save this? Assume some structure

Stochastic linear bandits

Stochastic linear bandits

- Assume there is a feature map $\psi : C \times [K] \to \mathbb{R}^d$
- And for an unknown vector $\theta_* \in \mathbb{R}^d$ it holds that

$$f(c, a) = \theta_*^T \psi(c, a), \quad \forall (c, a) \in C \times [K]$$

Stochastic linear bandits

- Assume there is a feature map $\psi : C \times [K] \to \mathbb{R}^d$
- And for an unknown vector $\theta_* \in \mathbb{R}^d$ it holds that

$$f(c, a) = \theta_*^T \psi(c, a), \quad \forall (c, a) \in C \times [K]$$

- We can simplify things
  - In each round t agent is given the choice set $A_t \subset \mathbb{R}^d$
  - Agent chooses an action $a_t \in A_t$ and receives reward

$$r_t = \theta_*^T a_t + \epsilon_t$$

Stochastic linear bandits

- Assume there is a feature map $\psi : C \times [K] \to \mathbb{R}^d$
- And for an unknown vector $\theta_* \in \mathbb{R}^d$ it holds that

$$f(c, a) = \theta_*^T \psi(c, a), \quad \forall (c, a) \in C \times [K]$$

- We can simplify things
  - In each round t agent is given the choice set $A_t \subset \mathbb{R}^d$
  - Agent chooses an action $a_t \in A_t$ and receives reward

$$r_t = \theta_*^T a_t + \epsilon_t$$

- UCB approach
  - Construct a confidence set $\mathcal{C}_t \subset \mathbb{R}^d$ that contains the unknown parameter vector θ_* with high probability
  - For any action $a \in \mathbb{R}^d$ let $\mathrm{UCB}_t(a) = \max_{\theta \in \mathcal{C}_t} \theta^T a$ be an upper bound on the expected reward $\theta_*^T a$ of a
  - Then select $a_t = \mathrm{argmax}_{a \in A_t} \mathrm{UCB}_t(a)$

Stochastic linear bandits

- Assume there is a feature map $\psi : C \times [K] \to \mathbb{R}^d$
- And for an unknown vector $\theta_* \in \mathbb{R}^d$ it holds that

$$f(c, a) = \theta_*^T \psi(c, a), \quad \forall (c, a) \in C \times [K]$$

- We can simplify things
  - In each round t agent is given the choice set $A_t \subset \mathbb{R}^d$
  - Agent chooses an action $a_t \in A_t$ and receives reward

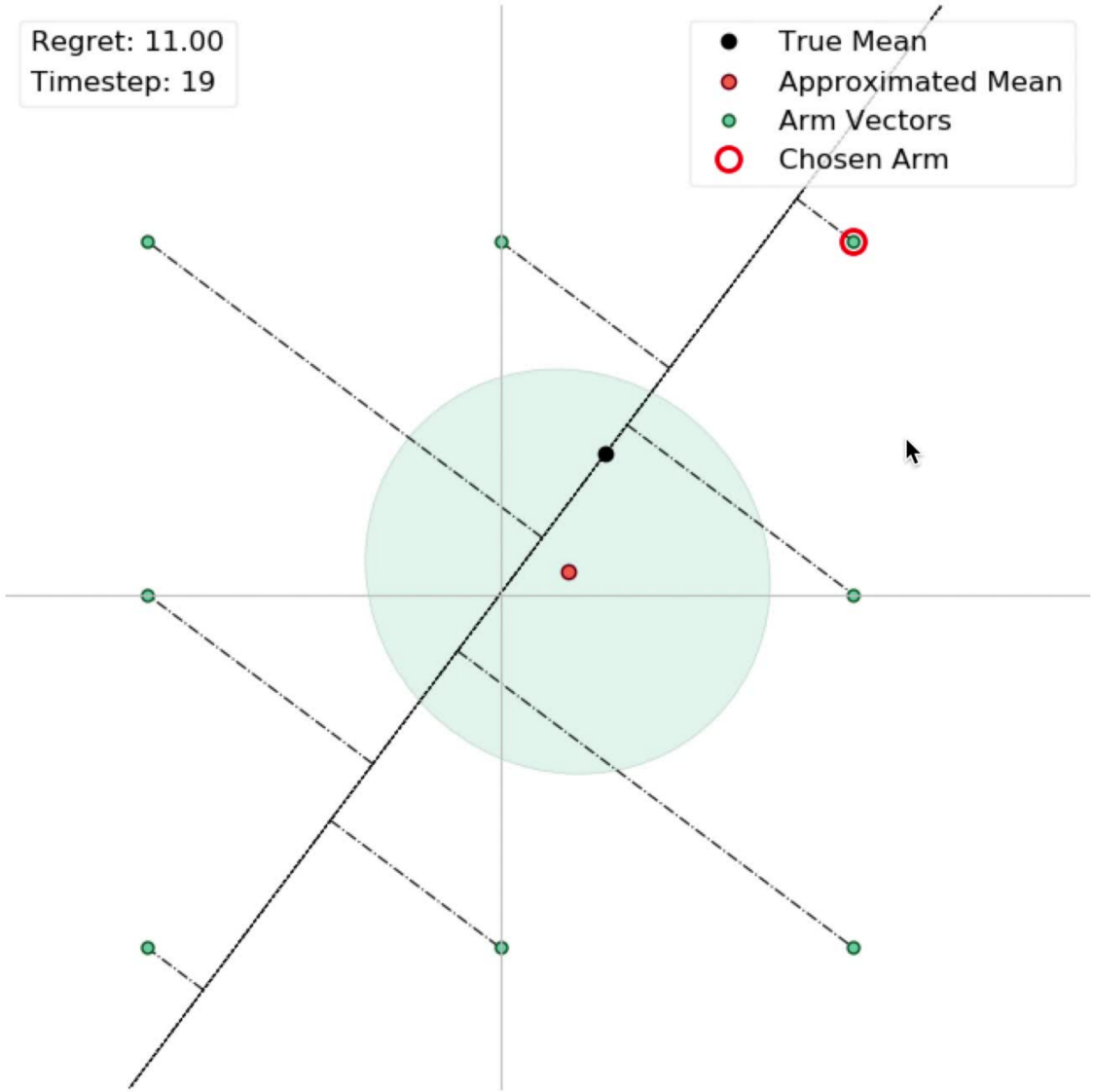
$$r_t = \theta_*^T a_t + \epsilon_t$$

- UCB approach
  - Construct a confidence set $\mathcal{C}_t \subset \mathbb{R}^d$ that contains the unknown parameter vector θ_* with high probability
  - For any action $a \in \mathbb{R}^d$ let $\mathrm{UCB}_t(a) = \max_{\theta \in \mathcal{C}_t} \theta^T a$ be an upper bound on the expected reward $\theta_*^T a$ of a
  - Then select $a_t = \mathrm{argmax}_{a \in A_t} \mathrm{UCB}_t(a)$
- Introduced by Abe et al. (2003) and developed by Auer (2002)

Illustration of the UCB approach

Source: banditalgs.com

Illustration of the UCB approach



Source: banditalgs.com

LinUCB

LinUCB

- Inputs: $\alpha \in \mathbb{R}_+, K, d \in \mathbb{N}$
- Set $A \leftarrow I_d$ and $b \leftarrow \mathbf{0}_d$
- **for** $t = 1, 2, \ldots, T$ **do**
  - $\theta_t \leftarrow A^{-1}b$
  - Observe K features $x_{t,1}, \ldots, x_{t,K} \in \mathbb{R}^d$
  - **for** $a = 1, 2, \ldots, K$ **do**
    - $p_{t,a} \leftarrow \theta_t^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A^{-1} x_{t,a}}$
  - **end for**
  - Choose action $a_t = \mathrm{argmax}_a p_{t,a}$
  - Observe reward $r_t \in 0, 1$
  - $A \leftarrow A + x_{t,a}x_{t,a}^T$
  - $b \leftarrow b + x_{t,a}r_t$
- **end for**

LinUCB

- Inputs: $\alpha \in \mathbb{R}_+, K, d \in \mathbb{N}$
- Set $A \leftarrow I_d$ and $b \leftarrow \mathbf{0}_d$
- **for** $t = 1, 2, \ldots, T$ **do**
  - $\theta_t \leftarrow A^{-1}b$
  - Observe K features $x_{t,1}, \ldots, x_{t,K} \in \mathbb{R}^d$
  - **for** $a = 1, 2, \ldots, K$ **do**
    - $p_{t,a} \leftarrow \theta_t^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A^{-1} x_{t,a}}$
  - **end for**
  - Choose action $a_t = \text{argmax}_a p_{t,a}$
  - Observe reward $r_t \in 0, 1$
  - $A \leftarrow A + x_{t,a}x_{t,a}^T$
  - $b \leftarrow b + x_{t,a}r_t$
- **end for**

- Theoretical guarantees in Chu et al. (2011) and empirical performance in Li et al. (2010)

Performance

Source: Lattimore & Czsepesvari (2020), Fig 19.1; Li et al. (2010) Fig 1 and 4

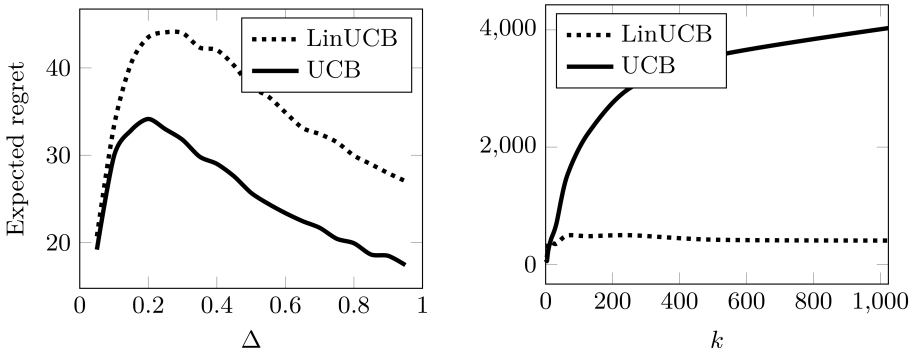
Performance

- Regret bound: $O(\sqrt{Td\ln^3(KT\ln(T)/\delta)})$

Source: Lattimore & Czsepesvari (2020), Fig 19.1; Li et al. (2010) Fig 1 and 4

Performance

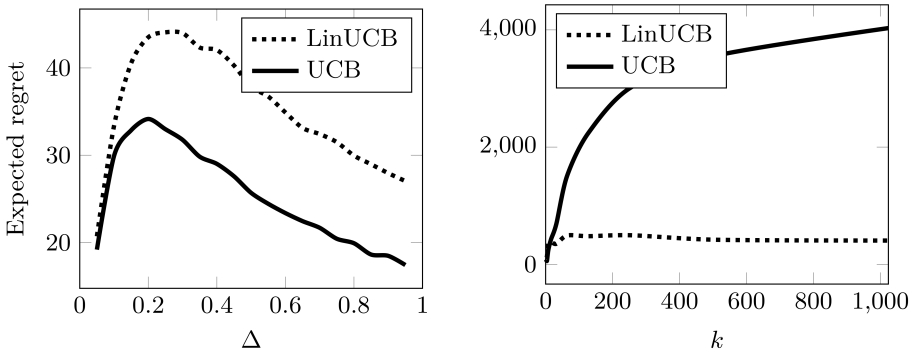
- Regret bound: $O(\sqrt{Td\ln^3(KT\ln(T)/\delta)})$
- Comparison to "poor man's" solution



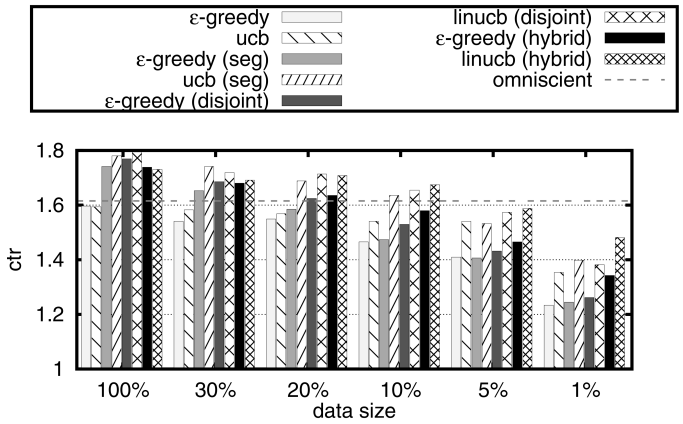
Source: Lattimore & Czsepesvari (2020), Fig 19.1; Li et al. (2010) Fig 1 and 4

Performance

- Regret bound: $O(\sqrt{Td\ln^3(KT\ln(T)/\delta)})$
- Comparison to "poor man's" solution



- Offline evaluation in Li et al. (2010)



Source: Lattimore & Czsepesvari (2020), Fig 19.1; Li et al. (2010) Fig 1 and 4

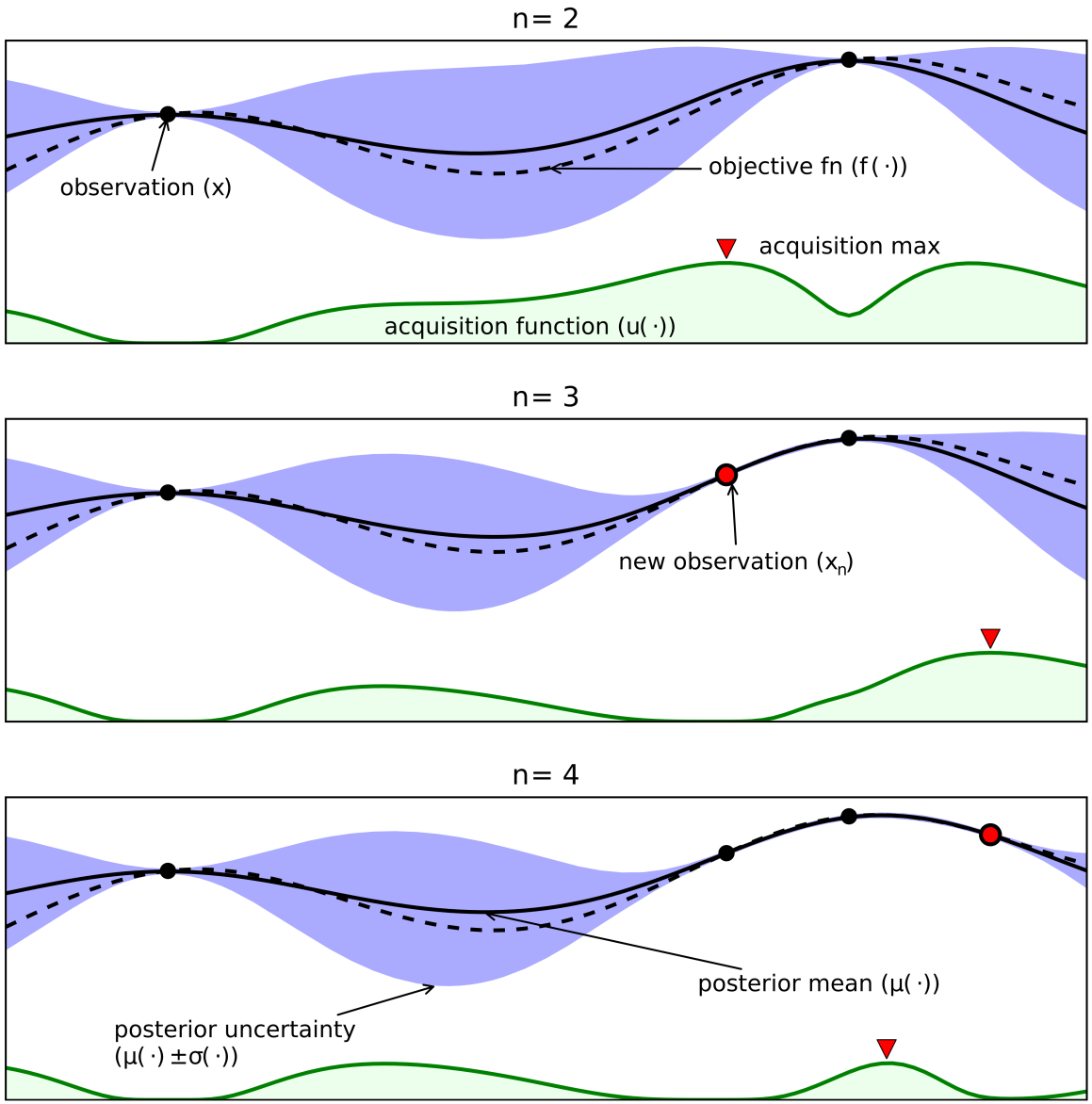
Beyond linear models with Bayesian optimization

Beyond linear models with Bayesian optimization

Bayesian optimization roadmap

Source: Shahriari et al 2016

Bayesian optimization roadmap



Source: Shahriari et al 2016

Refresher: Multivariate Gaussian distributions

Source: Wikimedia

Refresher: Multivariate Gaussian distributions

- Generalization of a one-dimensional Gaussian distribution
- Defined by a mean vector μ and covariance matrix Σ, $X \sim \mathcal{N}(\mu, \Sigma)$
- Distribution is centered about μ, while Σ determines its shape

Source: Wikimedia

Refresher: Multivariate Gaussian distributions

- Generalization of a one-dimensional Gaussian distribution
- Defined by a mean vector μ and covariance matrix Σ, $X \sim \mathcal{N}(\mu, \Sigma)$
- Distribution is centered about μ, while Σ determines its shape
- Decomposition on X and Y subsets is useful:

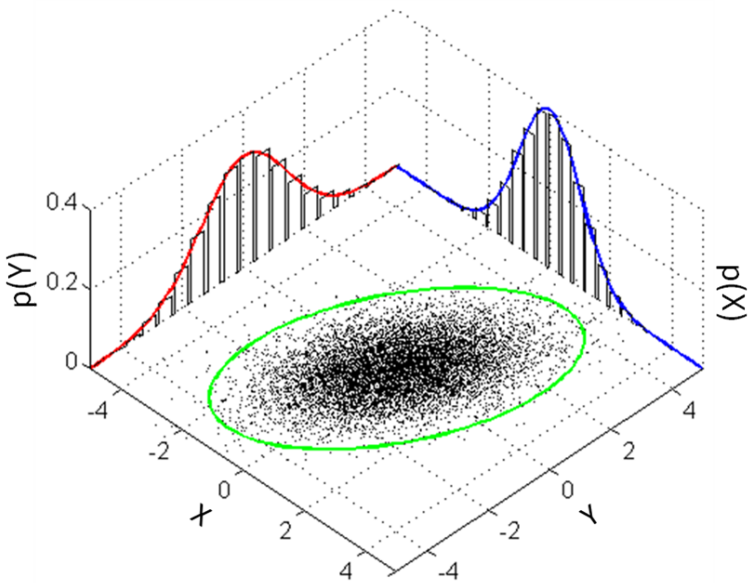
$$P_{X,Y} \sim \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$

Source: Wikimedia

Refresher: Multivariate Gaussian distributions

- Generalization of a one-dimensional Gaussian distribution
- Defined by a mean vector μ and covariance matrix Σ, $X \sim \mathcal{N}(\mu, \Sigma)$
- Distribution is centered about μ, while Σ determines its shape
- Decomposition on X and Y subsets is useful:

$$P_{X,Y} \sim \mathcal{N}\left(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix}\right)$$



Source: Wikimedia

Refresher: Multivariate Gaussian distributions

Refresher: Multivariate Gaussian distributions

- They have a great property of being closed under marginalization and conditioning (see Rasmussen & Williams, 2006 for derivations)

Refresher: Multivariate Gaussian distributions

- They have a great property of being closed under marginalization and conditioning (see Rasmussen & Williams, 2006 for derivations)
- **Marginalization**

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX})$$

$$Y \sim \mathcal{N}(\mu_Y, \Sigma_{YY})$$

Refresher: Multivariate Gaussian distributions

- They have a great property of being closed under marginalization and conditioning (see Rasmussen & Williams, 2006 for derivations)
- **Marginalization**

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX})$$

$$Y \sim \mathcal{N}(\mu_Y, \Sigma_{YY})$$

- **Conditioning**

$$X|Y \sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX})$$

$$Y|X \sim \mathcal{N}(\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_x), \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY})$$

Gaussian Processes (GP) - an overview

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*
  - We model the distribution of X_* together with X as a multivariate Gaussian distribution, where joint distribution ($|X|+|X_*|$ dimensional) spans the space of possible function values for the function we want to predict

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*
  - We model the distribution of X_* together with X as a multivariate Gaussian distribution, where joint distribution ($|X|+|X_*|$ dimensional) spans the space of possible function values for the function we want to predict
  - Bayesian inference requires conditional probability X_* given X - this is given immediately by conditioning property above!

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*
  - We model the distribution of X_* together with X as a multivariate Gaussian distribution, where joint distribution ($|X|+|X_*|$ dimensional) spans the space of possible function values for the function we want to predict
  - Bayesian inference requires conditional probability X_* given X - this is given immediately by conditioning property above!
  - Mean and covariance of conditional distribution defines the function for X_*

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*
  - We model the distribution of X_* together with X as a multivariate Gaussian distribution, where joint distribution ($|X|+|X_*|$ dimensional) spans the space of possible function values for the function we want to predict
  - Bayesian inference requires conditional probability X_* given X - this is given immediately by conditioning property above!
  - Mean and covariance of conditional distribution defines the function for X_*

Gaussian Processes (GP) - an overview

- GP is a stochastic process (a collection of random variables), such that every finite collection of those random variables has a multivariate normal distribution
- Goal of machine learning is to generalize to new data, given some training data and a model
  - We will assume that training points X are a finite collection of random variables, each following a Gaussian distribution
  - We want to predict function values for test points X_*
  - We model the distribution of X_* together with X as a multivariate Gaussian distribution, where joint distribution ($|X|+|X_*|$ dimensional) spans the space of possible function values for the function we want to predict
  - Bayesian inference requires conditional probability X_* given X - this is given immediately by conditioning property above!
  - Mean and covariance of conditional distribution defines the function for X_*
- Note that this makes GP a Bayesian nonparametric model

Mean and kernel function - key ingredients

Mean and kernel function - key ingredients

- Behavior of the GP is completely specified by its mean function and kernel (or covariance) function.

Mean and kernel function - key ingredients

- Behavior of the GP is completely specified by its mean function and kernel (or covariance) function.
- For a real process $f(\mathbf{x})$ we define $m(\mathbf{x})$, a mean function modeling the expected output of the function, and $k(\mathbf{x}, \mathbf{x}')$, a kernel function modeling the covariance between different points.

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

and

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

Mean and kernel function - key ingredients

- Behavior of the GP is completely specified by its mean function and kernel (or covariance) function.
- For a real process $f(\mathbf{x})$ we define $m(\mathbf{x})$, a mean function modeling the expected output of the function, and $k(\mathbf{x}, \mathbf{x}')$, a kernel function modeling the covariance between different points.

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

and

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

- We write the Gaussian process as

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

- The random variables represent the value of the function $f(\mathbf{x})$ at location $\mathbf{x}$

Mean and kernel function - key ingredients

- Behavior of the GP is completely specified by its mean function and kernel (or covariance) function.
- For a real process $f(\mathbf{x})$ we define $m(\mathbf{x})$, a mean function modeling the expected output of the function, and $k(\mathbf{x}, \mathbf{x}')$, a kernel function modeling the covariance between different points.

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

  and

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

- We write the Gaussian process as

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

- The random variables represent the value of the function $f(\mathbf{x})$ at location $\mathbf{x}$
- Mean and kernel function are your priors!

Kernel in more concrete terms

Source: Distill

Kernel in more concrete terms

- Kernel function k defines similarity between points, e.g. points in training set X and test set X_*
- $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

Source: Distill

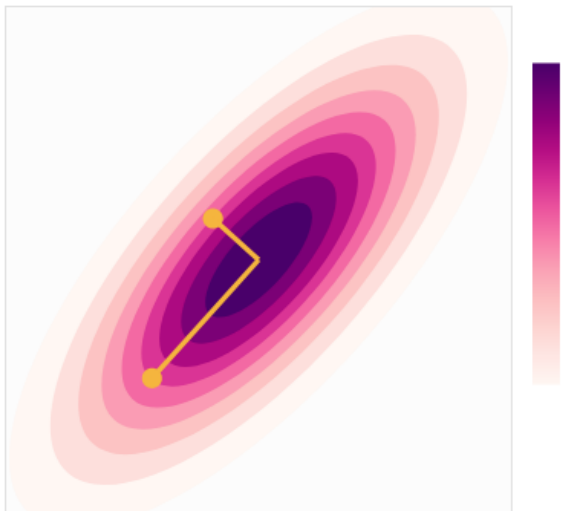
Kernel in more concrete terms

- Kernel function k defines similarity between points, e.g. points in training set X and test set X_*
- $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$
- This allows us to compute entries in the covariance matrix, $\Sigma = \mathrm{Cov}(X, X_*)$, by iteratively applying function k on all pairwise combinations of points in X and X_*

Source: Distill

Kernel in more concrete terms

- Kernel function k defines similarity between points, e.g. points in training set X and test set X_*
- $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$
- This allows us to compute entries in the covariance matrix, $\Sigma = \mathrm{Cov}(X, X_*)$, by iteratively applying function k on all pairwise combinations of points in X and X_*



Covariance matrix (Σ)

0.85	0.67
0.67	0.99

Source: Distill

Radial basis, Gaussian or Squared exponential kernel

Radial basis, Gaussian or Squared exponential kernel

- A popular choice:

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

  - We will focus on this one in Bayesian optimization

Radial basis, Gaussian or Squared exponential kernel

- A popular choice:

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

  - We will focus on this one in Bayesian optimization

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

Radial basis, Gaussian or Squared exponential kernel

- A popular choice:

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

  - We will focus on this one in Bayesian optimization

- Correlation between two points decays according to a power function in dependency of the distance between the two points, governed by λ
- Covariance is symmetric, that is, only the distance between two points matters, but not the direction.
- Good for smooth functions, hyperparameters λ (called the length-scale) and σ^2 (a scale factor) are normally optimized by using the marginal likelihood

Distribution over functions

Distribution over functions

- Specifying a mean and kernel function implies a **distribution over functions**

Distribution over functions

- Specifying a mean and kernel function implies a **distribution over functions**
- You can draw function samples from this distribution, evaluated at some points (e.g. test points X_*) and computing the associated covariance matrix K using the kernel function
- Then you generate a random Gaussian vector using this covariance matrix.

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

Distribution over functions

- Specifying a mean and kernel function implies a **distribution over functions**
- You can draw function samples from this distribution, evaluated at some points (e.g. test points X_*) and computing the associated covariance matrix K using the kernel function
- Then you generate a random Gaussian vector using this covariance matrix.

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

- We are usually not interested in drawing random functions from the prior
- Joint distribution of the training outputs $\mathbf{f}$ and test $\mathbf{f}_*$, according to the prior is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

Predictions

Predictions

- To get a posterior we need to restrict the joint prior distribution to contain only those functions which agree with observed training points
- We achieve this by conditioning

$$\mathbf{f}_*|X_*, X, \mathbf{f} = \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f},$$
$$K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*))$$

Predictions

- To get a posterior we need to restrict the joint prior distribution to contain only those functions which agree with observed training points
- We achieve this by conditioning

$$\mathbf{f}_*|X_*, X, \mathbf{f} = \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*))$$

- Note that this assumed noise-free observations, $\mathbf{f} = f(\mathbf{x})$
- In practice, we observe function values with some noise, $\mathbf{f} = f(\mathbf{x}) + \epsilon$
- Assuming noise is IID and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the noisy observations becomes $\mathrm{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I$

Predictions

- To get a posterior we need to restrict the joint prior distribution to contain only those functions which agree with observed training points
- We achieve this by conditioning

$$\mathbf{f}_*|X_*, X, \mathbf{f} = \mathcal{N}(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*))$$

- Note that this assumed noise-free observations, $\mathbf{f} = f(\mathbf{x})$
- In practice, we observe function values with some noise, $\mathbf{f} = f(\mathbf{x}) + \epsilon$
- Assuming noise is IID and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, the prior on the noisy observations becomes $\mathrm{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I$
- Assuming noisy observations predictions are given by

$$\mathbf{f}_*|X_*, X, \mathbf{y} = \mathcal{N}(K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \\ K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*))$$

Marginal likelihood

Marginal likelihood

- I.e. model evidence, $p(\mathbf{y}|X)$, is the integral of the likelihood times the prior

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}$$

Marginal likelihood

- I.e. model evidence, $p(\mathbf{y}|X)$, is the integral of the likelihood times the prior

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}$$

- The term *marginal* refers to the marginalization over the function values $\mathbf{f}$

Marginal likelihood

- I.e. model evidence, $p(\mathbf{y}|X)$, is the integral of the likelihood times the prior

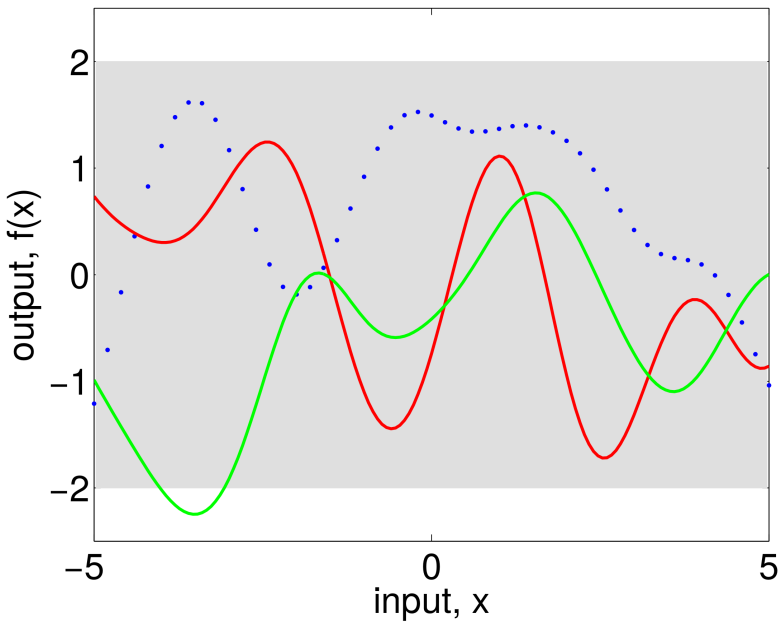
$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}$$

- The term *marginal* refers to the marginalization over the function values $\mathbf{f}$
- This integral can be computed as

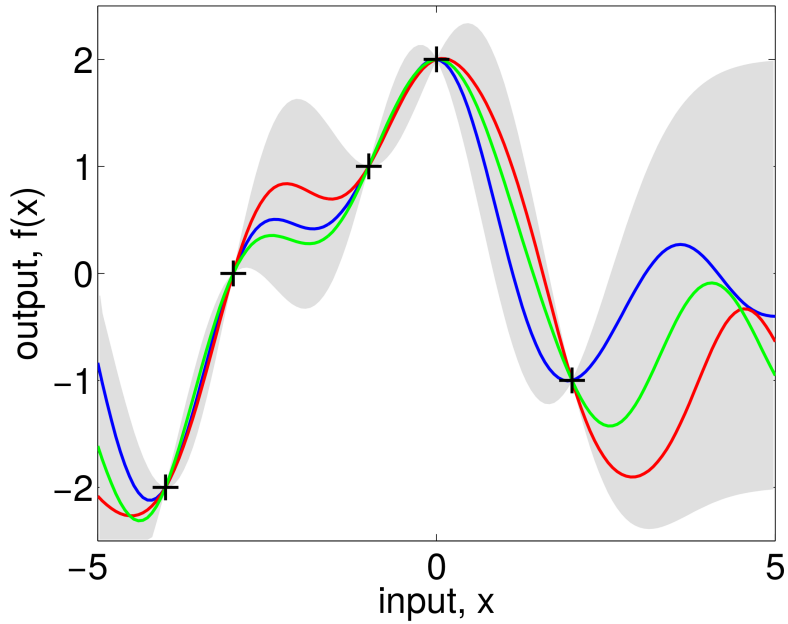
$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T[K + \sigma_n^2 I]^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi$$

- Log probability is usually used for optimization

Drawing from the GP prior and posterior



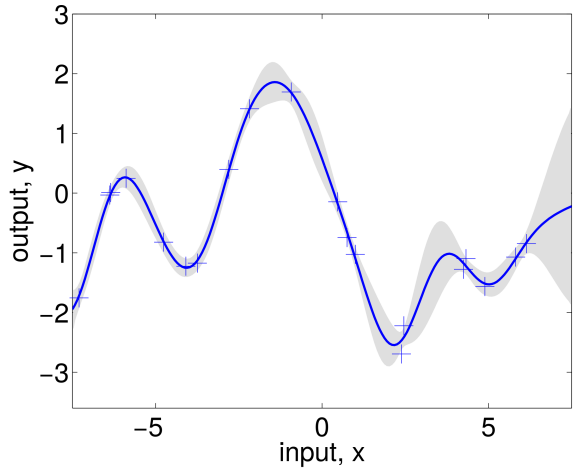
(a), prior



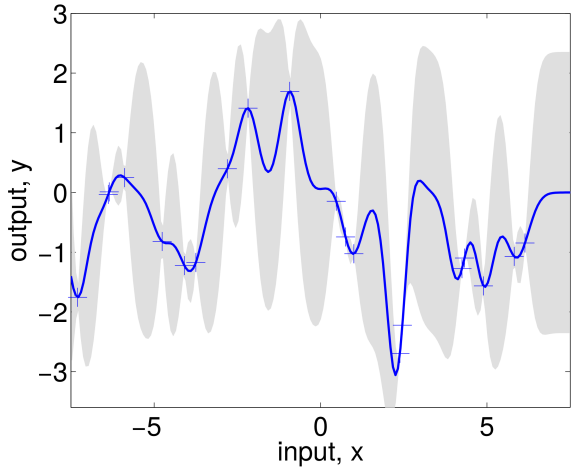
(b), posterior

Source: Rasmussen, Williams (2006)

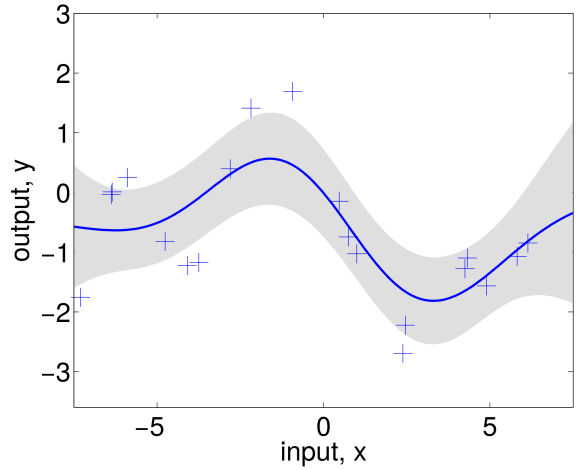
Dependence on hyperparameters



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

Source: Rasmussen, Williams (2006)

More on kernels

More on kernels

- Many choices for a kernel
  - constant: $k(x, x') = C$
  - white noise: $k(x, x') = \delta I_n$
  - linear: $k(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c)$
  - squared exponential: $k(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$
  - periodic: $k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)^2}{\lambda^2}\right)$
  - rational quadratic: $k(x, x') = \sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha\lambda^2}\right)^{-\alpha}$
  - and many more

More on kernels

- Many choices for a kernel
  - constant: $k(x, x') = C$
  - white noise: $k(x, x') = \delta I_n$
  - linear: $k(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c)$
  - squared exponential: $k(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$
  - periodic: $k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)^2}{\lambda^2}\right)$
  - rational quadratic: $k(x, x') = \sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha\lambda^2}\right)^{-\alpha}$
  - and many more
- A valid kernel has to result in a positive definite Σ matrix

More on kernels

- Many choices for a kernel
  - constant: $k(x, x') = C$
  - white noise: $k(x, x') = \delta I_n$
  - linear: $k(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c)$
  - squared exponential: $k(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$
  - periodic: $k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)^2}{\lambda^2}\right)$
  - rational quadratic: $k(x, x') = \sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha\lambda^2}\right)^{-\alpha}$
  - and many more
- A valid kernel has to result in a positive definite Σ matrix
- Algebra with kernels, e.g.:
  - **Sum** of valid kernels is a valid kernel
  - **Product** of valid kernels is a valid kernel

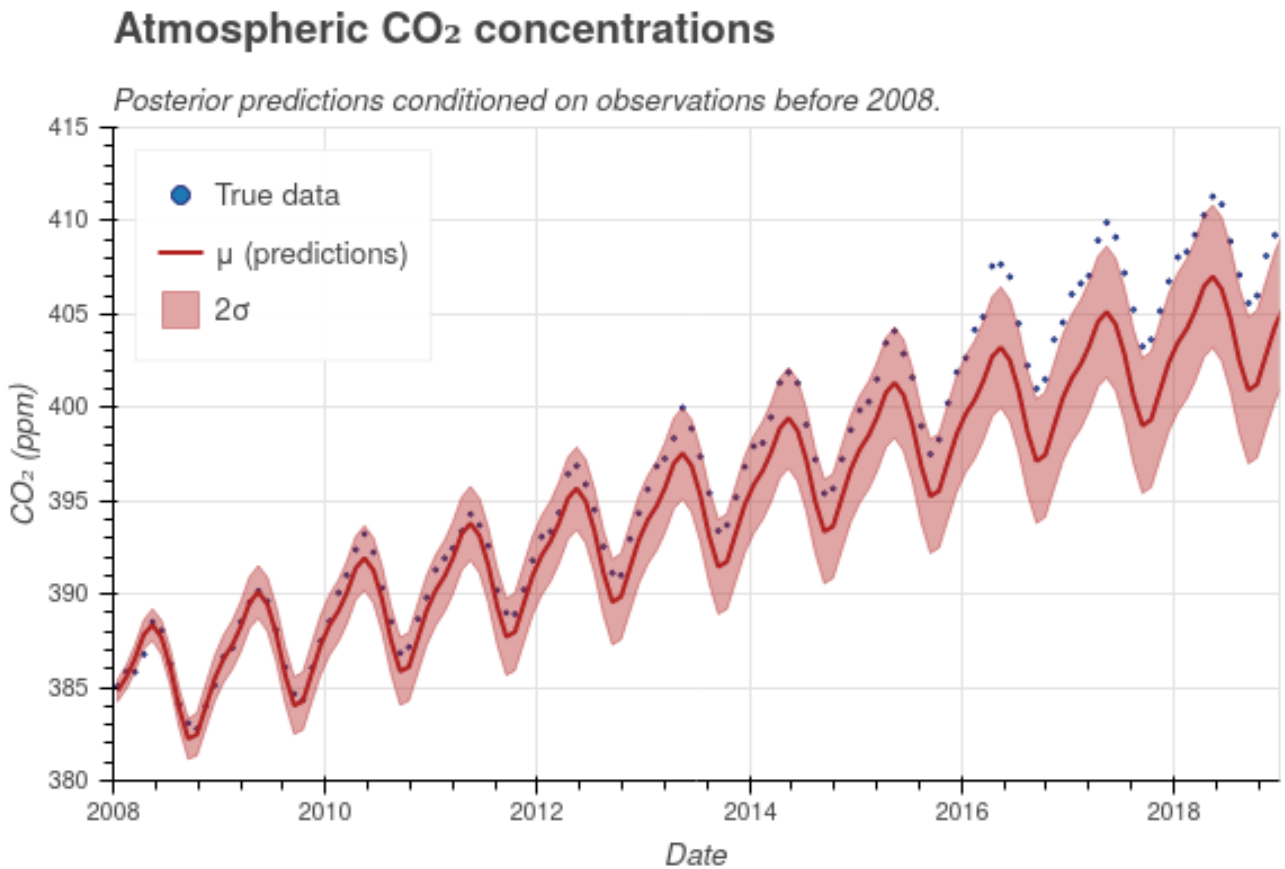
More on kernels

- Many choices for a kernel
  - constant: $k(x, x') = C$
  - white noise: $k(x, x') = \delta I_n$
  - linear: $k(x, x') = \sigma_b^2 + \sigma_v^2 (x - c)(x' - c)$
  - squared exponential: $k(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$
  - periodic: $k(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)^2}{\lambda^2}\right)$
  - rational quadratic: $k(x, x') = \sigma^2 \left(1 + \frac{(x-x')^2}{2\alpha\lambda^2}\right)^{-\alpha}$
  - and many more
- A valid kernel has to result in a positive definite Σ matrix
- Algebra with kernels, e.g.:
  - **Sum** of valid kernels is a valid kernel
  - **Product** of valid kernels is a valid kernel
- The choice is normally based on assumptions such as smoothness and likely patterns to be you expect in the data.

Compositional kernel example

Source: A blog post on GPs

Compositional kernel example



Source: A blog post on GPs

GP algorithm in practice

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_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

Computational considerations

- 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
- Using Cholesky decomposition reduces the cost, but not much

Computational considerations

- 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
- Using Cholesky decomposition reduces the cost, but not much
- Hence, not a good choice for big data
- But, there are attempts at scaling GP's
  - Sparse GP using pseudo-inputs is a popular variant
  - Approximate GP (e.g. with Fourier features)

Computational considerations

- 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
- Using Cholesky decomposition reduces the cost, but not much

- Hence, not a good choice for big data
- But, there are attempts at scaling GP's
  - Sparse GP using pseudo-inputs is a popular variant
  - Approximate GP (e.g. with Fourier features)

- In Bayesian optimization we usually want to repeat it after each new observation, updating our hyperparameters as we go
- With large budgets the cost might be prohibitive