Multi-Agent Adaptive Sampling

16.412 Advanced Lecture

Juan Jose Garau Luis, Stewart Jamieson, Sarah Keren, Rajat Talak

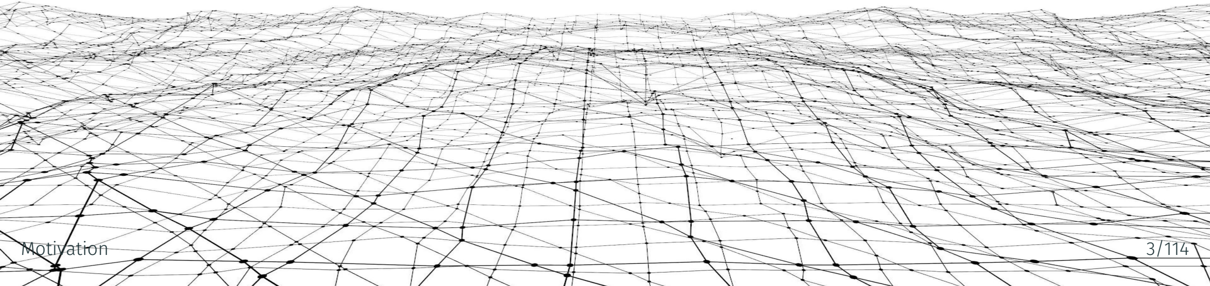
April 10, 2019

Table of Contents

Table of Contents

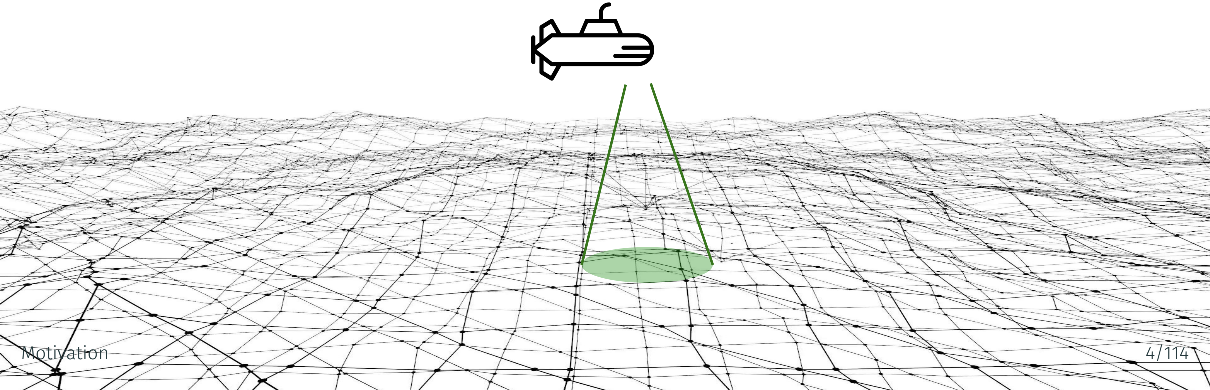
Motivation

- **Goal:** Locating the minima/maxima of a field in an unknown environment (e.g. ocean floor depth, chemical concentration, etc)



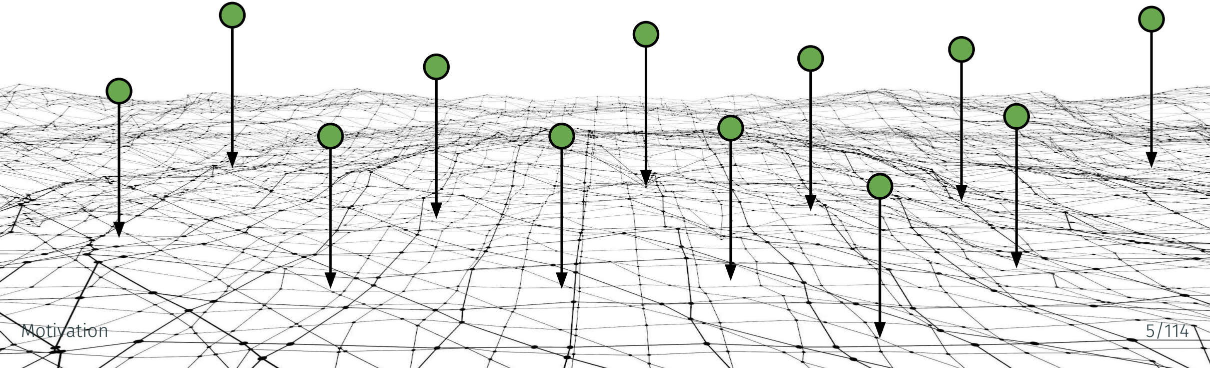
Motivation

- **Goal:** Locating the minima/maxima of a field in an unknown environment (e.g. ocean floor depth, chemical concentration, etc)
- Gliders, by means of different **sensors**, are capable of **sampling** the value of this field at their location



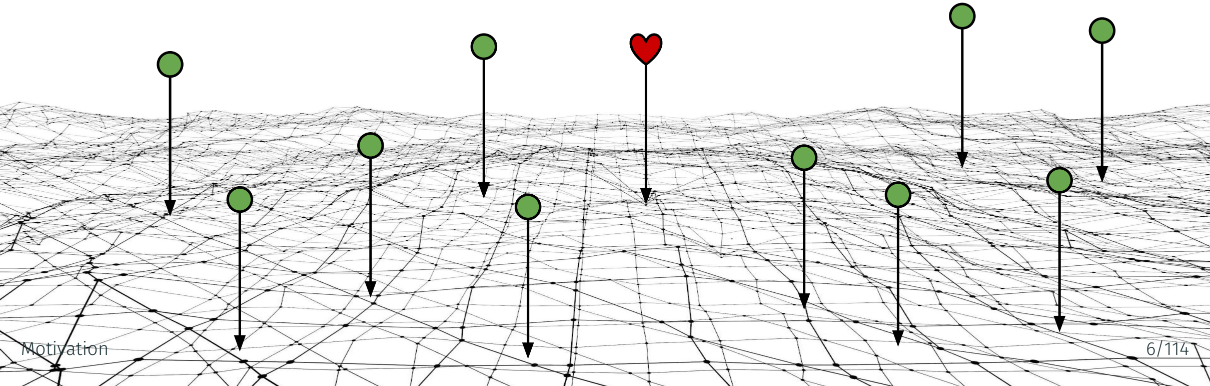
Motivation

- **Goal:** Locating the minima/maxima of a field in an unknown environment (e.g. ocean floor depth, chemical concentration, etc)
- Gliders, by means of different **sensors**, are capable of **sampling** the value of this field at their location
- We usually don't have enough **time** to sample an entire region, sampling may have a **cost,** or there may be a **limit** on the number of samples



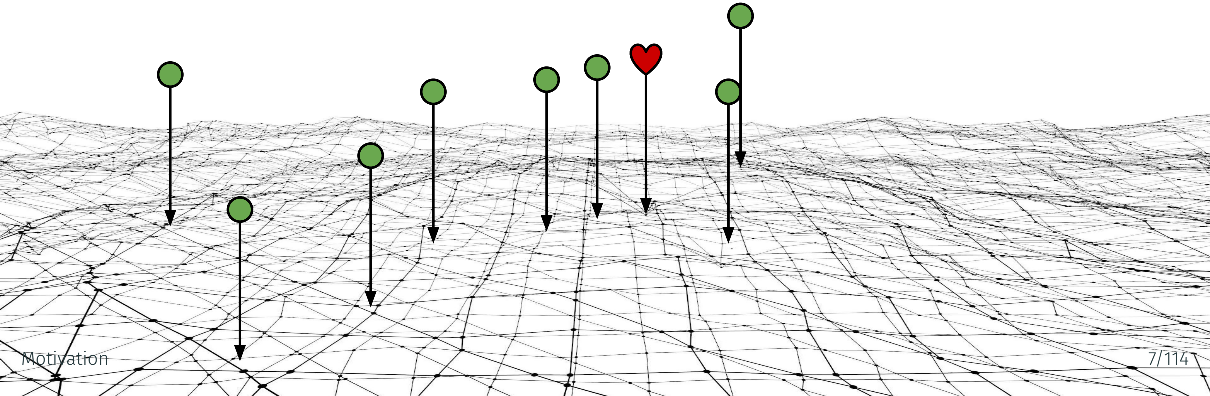
Motivation

- Some sampling policies might take many samples and still get a poor estimate of the optimal value



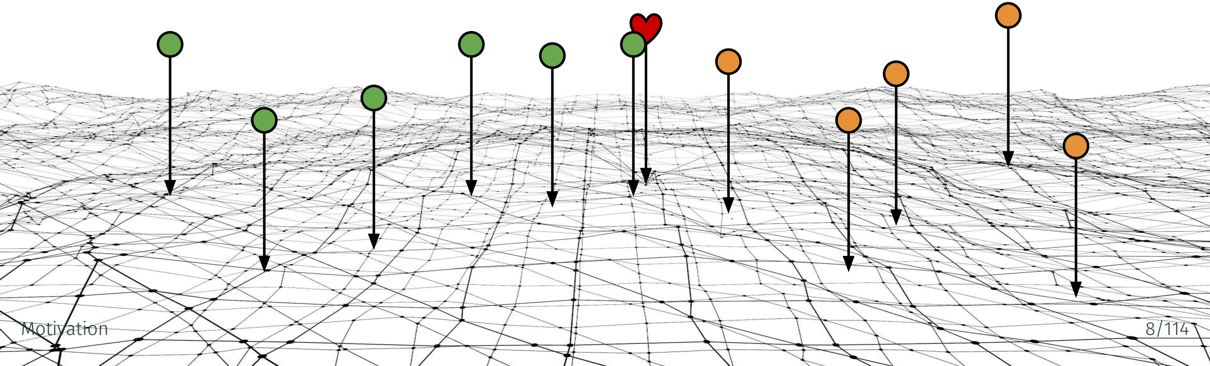
Motivation

- Some sampling policies might take many samples and still get a poor estimate of the optimal value
- How can we design sampling policies to get good estimates?



Motivation

- Some sampling policies might take many samples and still get a poor estimate of the optimal value
- How can we design sampling policies to get good estimates?
- How can we take advantage of using more than one glider?



Motivating Example

Figure 1: We have a team of gliders (blue triangles), and would like to find the deepest part of the caldera quickly so that we have plenty of time to take photos of benthic life.

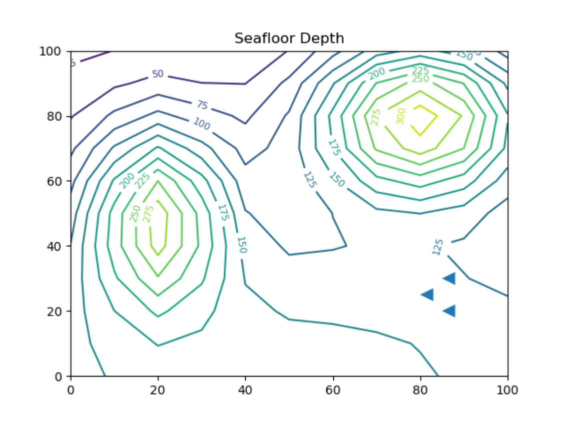


Table of Contents

The Sampling Problem

We wish to **improve** our understanding of the world (some quantity of interest)

- Our example: where is the deepest part of the caldera?

The Sampling Problem

We wish to **improve** our understanding of the world (some quantity of interest)

- Our example: where is the deepest part of the caldera?

What do we need to consider?

The Sampling Problem

We wish to **improve** our understanding of the world (some quantity of interest)

- Our example: where is the deepest part of the caldera?

What do we need to consider?

- The robot has a time-varying **physical state**

The Sampling Problem

We wish to **improve** our understanding of the world (some quantity of interest)

- Our example: where is the deepest part of the caldera?

What do we need to consider?

- The robot has a time-varying **physical state**
- The robot has a *model of the world*, represented by the **information state**

The Sampling Problem

We wish to **improve** our understanding of the world (some quantity of interest)

- Our example: where is the deepest part of the caldera?

What do we need to consider?

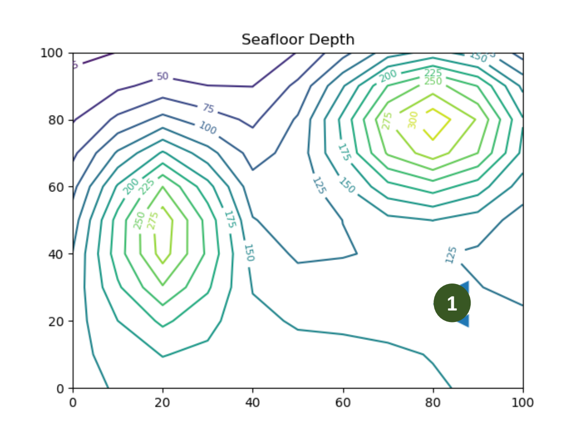
- The robot has a time-varying **physical state**
- The robot has a *model of the world*, represented by the **information state**
- The robot has a program, aka **policy**, that tells it what to do in any state

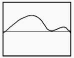
The Sampling Problem

We will explain how to **model the world**, **decide where to sample**, and **improve our model given a new sample**.

Modeling

Caldera example





I

1

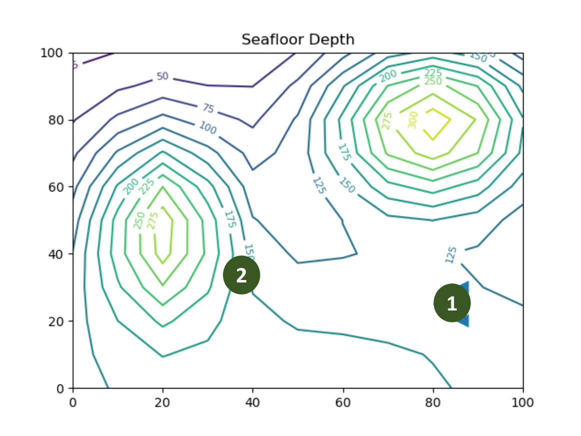
P = (x1, y1)

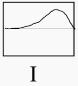
Two states:

1. Physical State P
2. Information State I

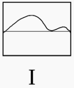
Modeling

Caldera example





2 P = (x2, y2)



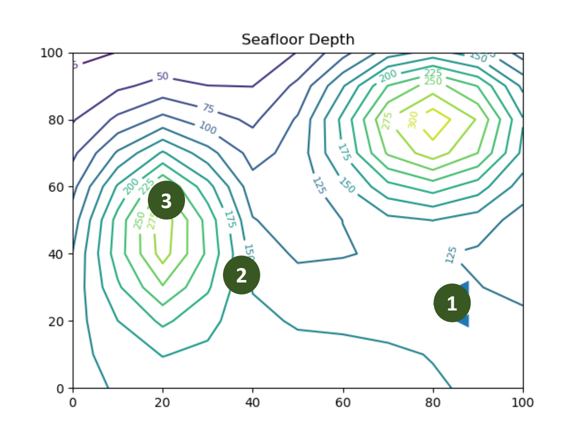
1 P = (x1, y1)

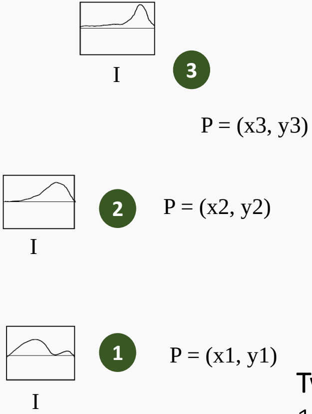
Two states:

1. Physical State P
2. Information State I

Modeling

Caldera example

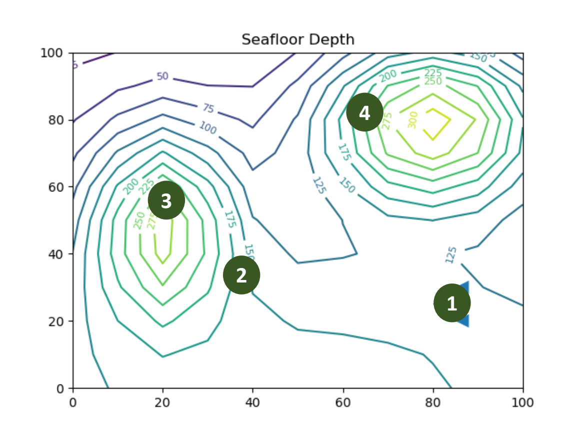


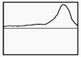


Two states:

1. Physical State P
2. Information State I

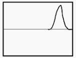
Caldera example




I

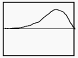
3

P = (x3, y3)

I


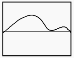
4

P = (x4, y4)


I

2

P = (x2, y2)


I

1

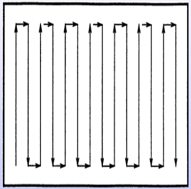
P = (x1, y1)

Two states:

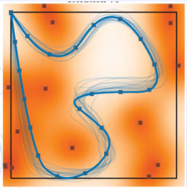
1. Physical State P
2. Information State I

Overview of Sampling Strategies

- Fixed Strategies
  - Standard Paths

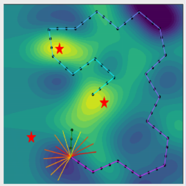
  - Informative Paths

- Adaptive Strategies
  - Dynamic Replanning

  - Bayesian Optimization

Fixed Sampling Strategies

Definition: Fixed Sampling

A type of sampling in which the next sample location does not depend on any previously sampled values.

Fixed Sampling Strategies

Definition: Fixed Sampling

A type of sampling in which the next sample location does not depend on any previously sampled values.

Some common fixed sampling strategies:

- Standard Patterns (e.g. lawnmower)
- Informative Paths

Fixed Sampling Examples: Standard Patterns

Features:

- Can guarantee complete coverage
- Efficient for complete searches[1]

When to use it:

- Enough time for an exhaustive search
- No cost or limit to sampling
- Finding the ”needle in a haystack” (e.g. treasure chest)

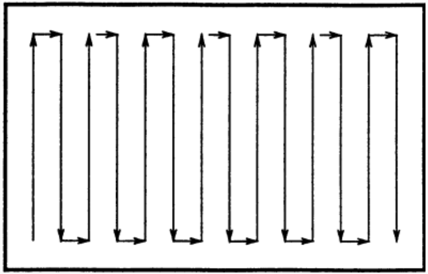


Figure 2: *Boustrophedonic* (lawnmower) patterns[1]: Used since ancient times, the ”ox turning” path for plowing a field.

[1] Choset and Pignon, “Coverage Path Planning: The Boustrophedon Cellular Decomposition”, 1998.

Fixed Sampling Example: Standard Patterns

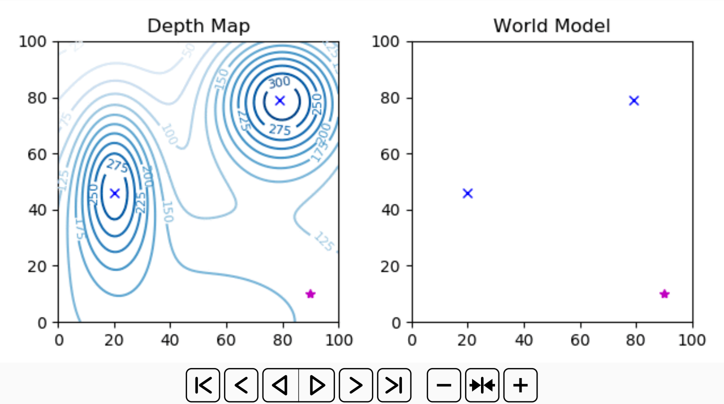


Figure 3: Fixed sampling using a standard "lawnmower" pattern.

Fixed Sampling Examples: Informative Paths

Features:

- Objective function $f(x)$ can incorporate knowledge about the world
- Maximize $\int_{\mathbb{P}} f(x)dx$ along the path $\mathbb{P}$

When to use it:

- There is a good objective function for your application
- Using a few simple robots (i.e. optimization is tractable)

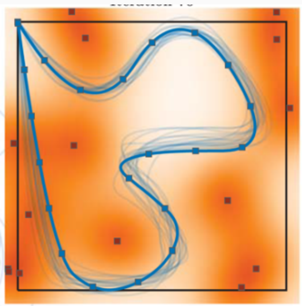


Figure 4: A trajectory formed by an evolutionary process[2] to maximize coverage while avoiding the black squares.

[2] Hitz, Galceran, Garneau, Pomerleau, and Siegwart, "Adaptive continuous-space informative path planning for online environmental monitoring", 2017

Adaptive Sampling

Definition: Adaptive Sampling

Sampling in which sample locations are chosen based on previous samples from the same mission.

Adaptive Sampling

Definition: Adaptive Sampling

Sampling in which sample locations are chosen based on previous samples from the same mission.

Why is it worth the extra computation? Things we observe are often **spatially correlated**:

Adaptive Sampling

Definition: Adaptive Sampling

Sampling in which sample locations are chosen based on previous samples from the same mission.

Why is it worth the extra computation? Things we observe are often **spatially correlated**:

- Many environments are described by **continuous** functions (e.g. seafloor depth)

Adaptive Sampling

Definition: Adaptive Sampling

Sampling in which sample locations are chosen based on previous samples from the same mission.

Why is it worth the extra computation? Things we observe are often **spatially correlated**:

- Many environments are described by **continuous** functions (e.g. seafloor depth)
- Many discrete phenomenon occur in **clusters** (e.g. volcanoes, fish, corals)

Adaptive Sampling

Definition: Adaptive Sampling

Sampling in which sample locations are chosen based on previous samples from the same mission.

Why is it worth the extra computation? Things we observe are often **spatially correlated**:

- Many environments are described by **continuous** functions (e.g. seafloor depth)
- Many discrete phenomenon occur in **clusters** (e.g. volcanoes, fish, corals)

$\implies$ A measurement at one point gives us hints on what we would measure nearby!

Features:

- Refines IPP solution after each sample
- Can limit deviation from base path

When to use it:

- You are already using IPP
- You only have a single robot, or the robots have assigned zones

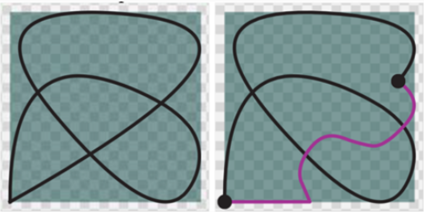


Figure 5: An IPP trajectory (left) that was replanned between two milestones (right) based on samples collected along the way[3].

[3] Hitz, Galceran, Garneau, Pomerleau, and Siegwart, "Adaptive continuous-space informative path planning for online environmental monitoring", 2017

Adaptive Sampling Examples: Bayesian

Features:

- No need for pre-computed base path
- Multiple robots can collaborate easily
- Efficiently finds global optimum (given a good prior world model)

When to use it:

- Multi-agent sampling
- Sufficient on-board computing power

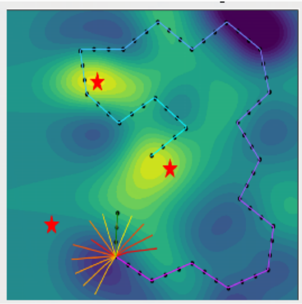


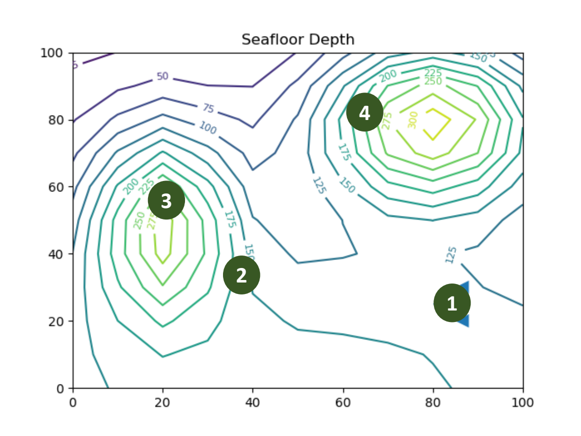
Figure 6: After each timestep, the robot chooses the move that gives it the most information[4].

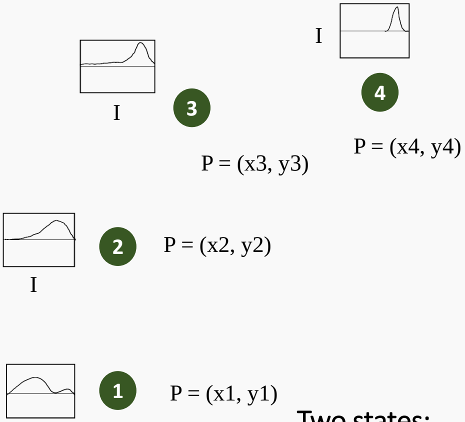
[4] Flaspohler, Preston, Michel, Girdhar, and Roy, "Information-Guided Robotic Maximum Search in Partially Observable Continuous Environments", 2019

Table of Contents

Recall: Modeling

Caldera example





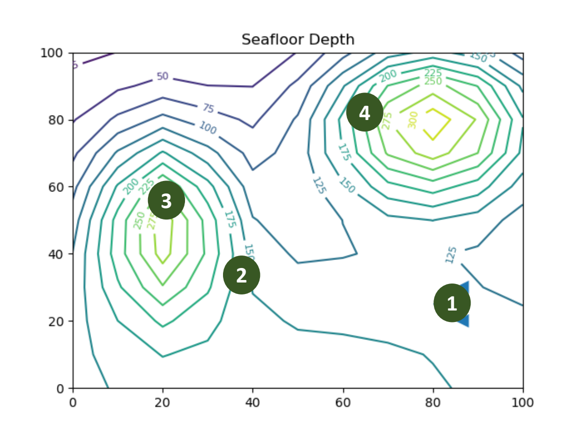
I

Two states:

1. Physical State P
2. Information State I

Recall: Modeling

Caldera example



Physical State:

1. Motion constraints
2. Timing constraints/costs

Information State:

1. State of the world
2. Update after acquiring new information/samples

Two states:

1. Physical State P
2. Information State I

Adaptive Sampling in Information State

- Ignore the physical state, and **focus on information state**
  1. Ignore motion constraints, time constraints …

- **Learn about:**
  1. How to model information state?
  2. How and what to sample, given an information state?
  3. How to update the information state?

  4. Generic adaptive sampling algorithm
  5. When is adaptive sampling better than an offline design?

Add motion constraints later …

Adaptive Sampling in Information State

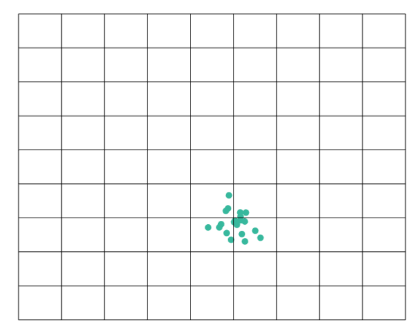
- What is adaptive sampling in information state?

Adaptive Sampling in Information State

- What is adaptive sampling in information state?

  - Have to locate
  - Could be anywhere
  - How do you sample?

Probed regions

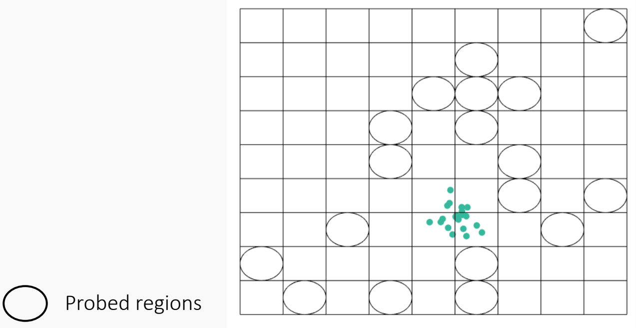


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

A priori design: random samples

Generate N uniformly distributed samples/locations for probing

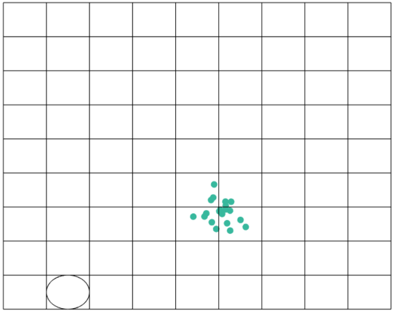


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

Adaptive design:

Generate the next sample, based on the information gathered of the previous sample

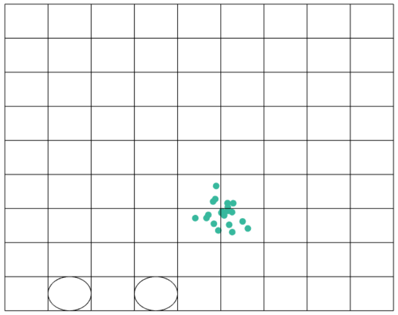


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

Adaptive design:

Generate the next sample, based on the information gathered of the previous sample

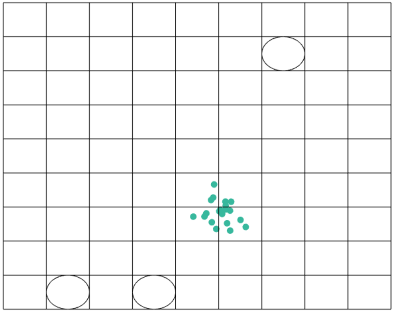


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

Adaptive design:

Generate the next sample, based on the information gathered of the previous sample

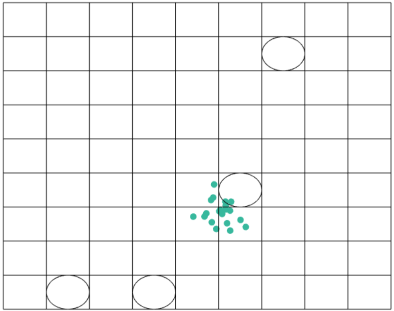


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

Adaptive design:

Generate the next sample, based on the information gathered of the previous sample

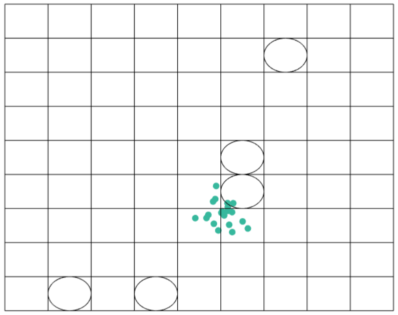


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

Adaptive design:

Generate the next sample, based on the information gathered of the previous sample

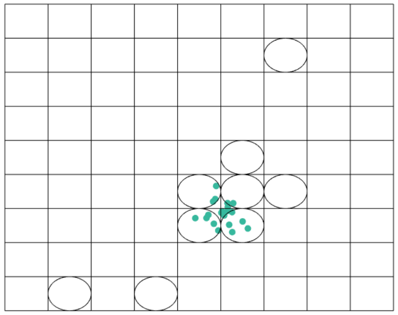


Adaptive Sampling in Information State

- What is adaptive sampling in information state?

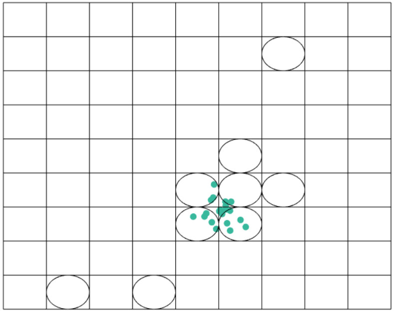
Adaptive design:

Generate the next sample, based on the information gathered of the previous sample



Adaptive Sampling in Information State

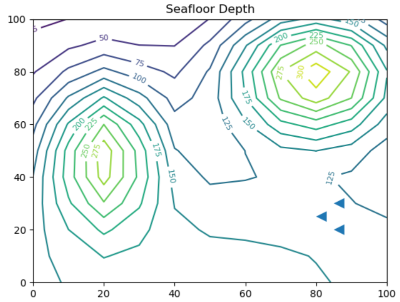
- Advantages of adaptive sampling
  - A priori design may be wasteful of resources
  - Take too many samples to achieve a certain accuracy
  - Sequential design (adaptive sampling) makes use of the information acquired
- Next:
  - Mathematical formulation
  - Modeling the Information State



Adaptive Sampling in Information State

- Advantages of adaptive sampling
  - A priori design may be wasteful of resources
  - Take too many samples to achieve a certain accuracy
  - Sequential design (adaptive sampling) makes use of the information acquired
- Next:
  - Mathematical formulation
  - Modeling the Information State

Caldera example



We are interested in depth

But, it could be any thing else ....

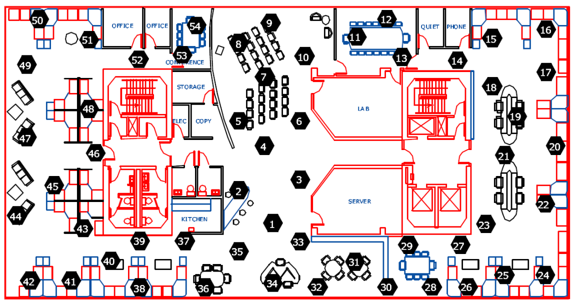
- Concentration, Temperature, ... any spatial field

Modeling the Information State

- Need a joint distribution over measurements at 54 locations

Any suggestions?

Why?



54 sensor node deployment to measure temperature

[Krause, Singh, Guestrin "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms, and Empirical Studies" J ML Research 2008]

Modeling the Information State

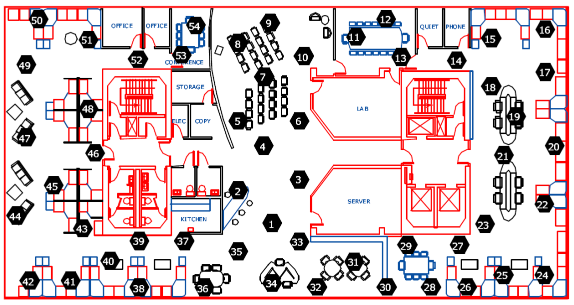
- Need a joint distribution over measurements at 54 locations
- Simple, effective approach:
  - Joint (multi-variate) Gaussian distribution

$$P(\mathcal{X}_{\mathcal{V}} = \mathbf{x}_{\mathcal{V}}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{\mathcal{V}\mathcal{V}}|} e^{-\frac{1}{2}(\mathbf{x}_{\mathcal{V}}-\mu_{\mathcal{V}})^T \Sigma_{\mathcal{V}\mathcal{V}}^{-1}(\mathbf{x}_{\mathcal{V}}-\mu_{\mathcal{V}})}$$

$\mu_{\mathcal{V}}$ -- mean vector

$\Sigma_{\mathcal{V}\mathcal{V}}$ -- covariance matrix

- Analytically tractable



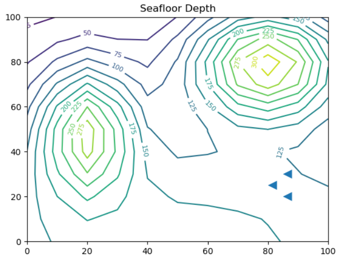
54 sensor node deployment to measure temperature

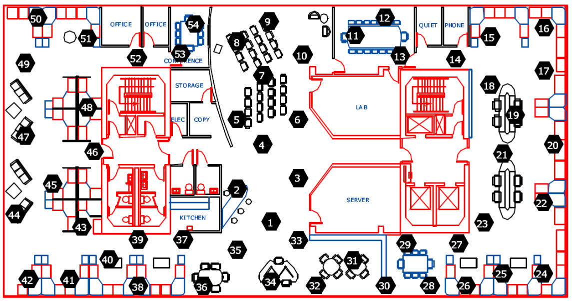
[Krause, Singh, Guestrin “Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms, and Empirical Studies” J ML Research 2008]

Modeling the Information State

- Interested in locations where no sensor is placed (yet).
- Need a model for measurements at infinitely many locations.
  - Infinitely many random variables.

Caldera example





54 sensor node deployment to measure temperature

[Krause, Singh, Guestrin "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms, and Empirical Studies" J ML Research 2008]

- **Gaussian Process:** natural extension
- Used to model various spatial fields
  - Temperature, pH, depth, …

Gaussian Processes

[Rasmussen & Williams, GP for Machine Learning, 2006]

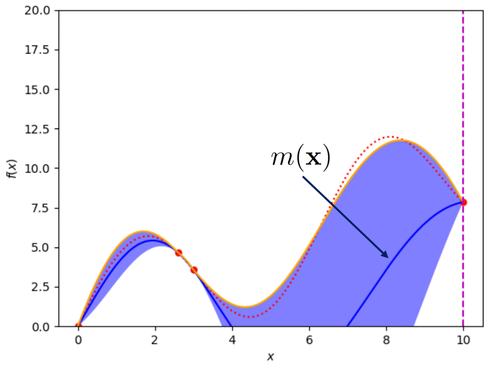
Definition:

Is a collection of random variables, any finite number of which have a joint Gaussian distribution

Notation: $f(\mathbf{x})$ is a random variable for each $\mathbf{x}$

Mean and covariance functions:

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$



In the picture we see only variance, but covariance is also defined

Gaussian Processes

[Rasmussen & Williams, GP for Machine Learning, 2006]

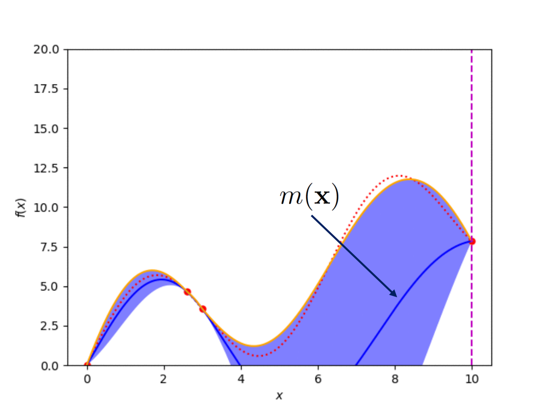
Our information state will be modeled as a Gaussian process

- Randomness indicates our uncertainty in knowing the actual state

Notation: $f(\mathbf{x})$ is a random variable for each $\mathbf{x}$

Mean and covariance functions:

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$

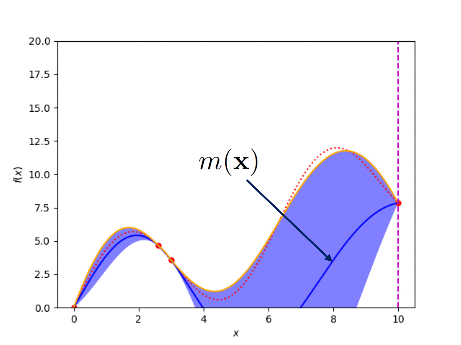


In the picture we see only variance, but covariance is also defined

Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

- When a new sample is obtained, the Gaussian process (Information state) is updated:



Gaussian Processes: Update?

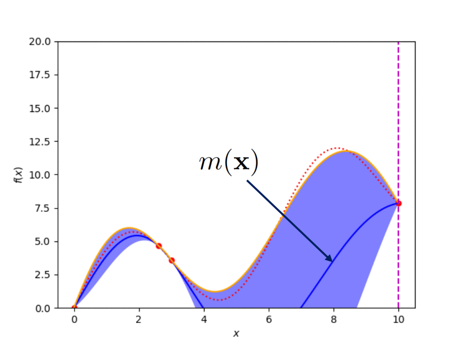
[Rasmussen & Williams, GP for Machine Learning, 2006]

- When a new sample is obtained, the Gaussian process (Information state) is updated:

$\mathbf{X}$ sampling location
$\mathbf{f}$ sample value

$$\mathbf{f}_*|X_*, X, \mathbf{f}$$

$$\sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f},\\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big)$$



Joint Gaussian distribution

Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

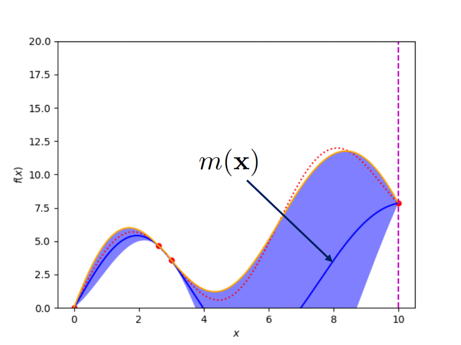
- When a new sample is obtained, the Gaussian process (Information state) is updated:

$\mathbf{X}$ sampling location

$\mathbf{f}$ sample value

$$\mathbf{f}_* | \overbrace{X_*}^{\text{new points}}, \overbrace{X, \mathbf{f}}^{\text{sampled data}}$$

$$\sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f},$$
$$K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big)$$



Joint Gaussian distribution

Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

- When a new sample is obtained, the Gaussian process (Information state) is updated:

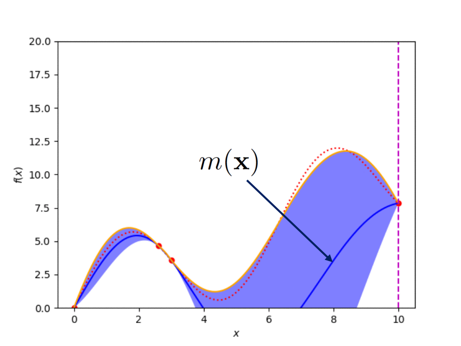
$\mathbf{X}$ sampling location

$\mathbf{f}$ sample value

$$\mathbf{f}_* | \overbrace{X_*}^{\text{new points}}, \overbrace{X, \mathbf{f}}^{\text{sampled data}}$$

$$\sim \mathcal{N}\Big(\overbrace{K(X_*, X)K(X, X)^{-1}\mathbf{f}}^{\text{Mean}},\\ \underbrace{K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)}_{\text{Covariance}}\Big)$$

Joint Gaussian distribution



Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

- When a new sample is obtained, the Gaussian process (Information state) is updated:

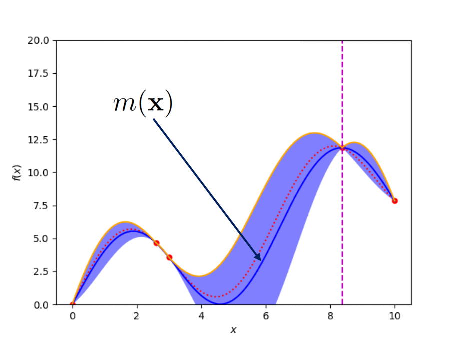
$$\mathbf{f}_* | \underbrace{X_*}_{\text{new points}}, \underbrace{X, \mathbf{f}}_{\text{sampled data}}$$

$$\sim \mathcal{N}\big(\underbrace{K(X_*, X)K(X,X)^{-1}\mathbf{f}}_{\text{Mean}},\ \underbrace{K(X_*, X_*) - K(X_*, X)K(X,X)^{-1}K(X, X_*)}_{\text{Covariance}}\big)$$

Joint Gaussian distribution

$\mathbf{X}$ sampling location

$\mathbf{f}$ sample value



Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

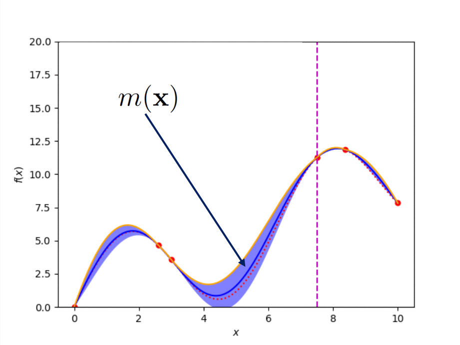
- When a new sample is obtained, the Gaussian process (Information state) is updated:

$\mathbf{X}$ sampling location
$\mathbf{f}$ sample value

$$\mathbf{f}_*|\overbrace{X_*}^{\text{new points}}, \overbrace{X, \mathbf{f}}^{\text{sampled data}}$$

$$\sim \mathcal{N}\Big(\overbrace{K(X_*,X)K(X,X)^{-1}\mathbf{f}}^{\text{Mean}},\ \underbrace{K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)}_{\text{Covariance}}\Big)$$

Joint Gaussian distribution

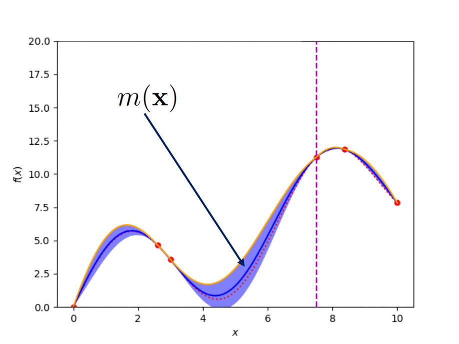


Gaussian Processes: Update?

[Rasmussen & Williams, GP for Machine Learning, 2006]

- When a new sample is obtained, the Gaussian process (Information state) is updated:

$\mathbf{X}$ sampling location
$\mathbf{f}$ sample value



$$\mathbf{f}_* | \overbrace{X_*}^{\text{new points}}, \overbrace{X, \mathbf{f}}^{\text{sampled data}}$$

$$\sim \mathcal{N}\Big(\overbrace{K(X_*, X)K(X, X)^{-1}\mathbf{f}}^{\text{Mean}},\; \underbrace{K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)}_{\text{Covariance}}\Big)$$

Joint Gaussian distribution

<u>Note</u>: the variance update does not depend on what you sample, but only where you sample.

Gaussian Process: Mean and Covariance Functions

[Rasmussen & Williams, GP for Machine Learning, 2006]

Specifying mean and covariance is enough to define a Gaussian Process

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$

Examples of mean and covariance functions:

- Mean function is assumed to be $\mathbf{0}$:
- Various covariance functions:

Squared Exponential Function

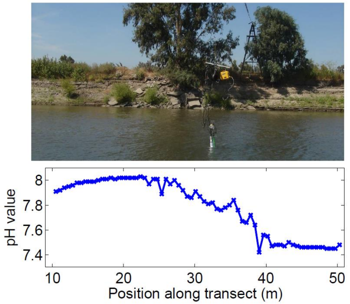
$$
k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^\top M (\mathbf{x}_p - \mathbf{x}_q)\right) + \sigma_n^2 \delta_{pq}
$$

Gaussian Process: Mean and Covariance Functions

[Krause, Guestrin "Nonmyopic Active Learning of Gaussian Processes: An Exploration-Exploitation Approach" ICML 2007]

Specifying mean and covariance is enough to define a Gaussian Process

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$



Samples of pH acquired along horizontal transect [Harmon et al., 2006]

Examples of mean and covariance functions:

- Mean function is assumed to be 0:
- Various covariance functions:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-|\mathbf{x}_p - \mathbf{x}_q|/\theta) \qquad k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-\left(\mathbf{x}_p - \mathbf{x}_q\right)^2/\theta)$$

Gaussian Process: Mean and Covariance Functions

[Rasmussen & Williams, GP for Machine Learning, 2006]

Specifying mean and covariance is enough to define a Gaussian Process

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$

Examples of mean and covariance functions:

- Mean function is assumed to be $\mathbf{0}$:
- Various covariance functions:

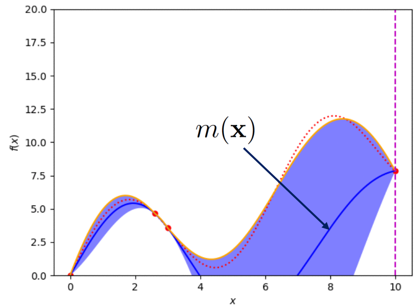
For atmospheric concentration of CO_2

$$
k(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{2\theta_2^2}\right) + \theta_3^2 \exp\left(-\frac{(x - x')^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x - x'))}{\theta_5^2}\right) + \theta_6^2\left(1 + \frac{(x - x')^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}
$$

Gaussian Process: Mean and Covariance Functions

Specifying mean and covariance is enough to define a Gaussian Process

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$



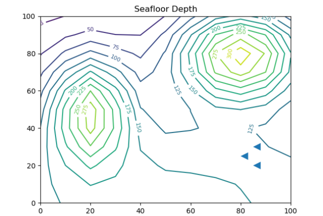
Information State as a Gaussian Process

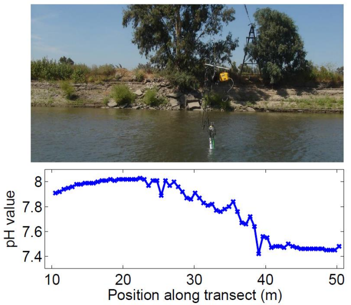
Specifying mean and covariance is enough to define a Gaussian Process

$$
\begin{aligned}
m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\
k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]
\end{aligned}
$$

Is it enough for modeling the Information state?

Caldera example



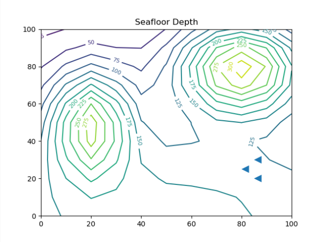


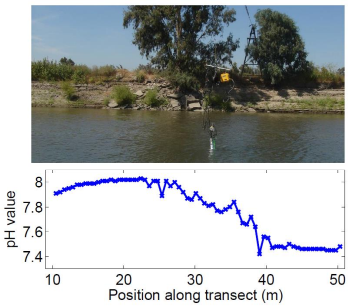
Samples of pH acquired along horizontal transect [Harmon et al., 2006]

Information State as a Gaussian Process

- We may not know the mean and covariance function
  - what is the mean depth?
  - how is the concentration/depth/field correlated across different locations?

Caldera example





Samples of pH acquired along horizontal transect [Harmon et al., 2006]

Information State as a Gaussian Process

- Way out:
- Impose a model on the covariance itself!
- Done with hyper-parameters

- Parameterize the covariance function

Hyper-parameters

1. $k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^\top M(\mathbf{x}_p - \mathbf{x}_q)\right) + \sigma_n^2 \delta_{pq}$ $\quad (\{M\}, \sigma_f^2, \sigma_n^2)$

2. $k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-|\mathbf{x}_p - \mathbf{x}_q|/\theta)$ $\quad k(\mathbf{x}_p, \mathbf{x}_q) = \exp(-(\mathbf{x}_p - \mathbf{x}_q)^2/\theta)$ $\quad \theta$

Information State as a Gaussian Process

- Way out:
  - Impose a model on the covariance itself!
  - Done with hyper-parameters

- Parameterize the covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = k_\theta(\mathbf{x}_p, \mathbf{x}_q)$$

more generally for hyper-parameter θ

- **Model**: Add a distribution on the hyper-parameters

Information State as a Gaussian Process

- Way out:
- Impose a model on the covariance itself!
- Done with hyper-parameters

- Parameterize the covariance function

$$k(\mathbf{x}_p, \mathbf{x}_q) = k_\theta(\mathbf{x}_p, \mathbf{x}_q)$$

more generally for hyper-parameter θ

- **Model**: Add a distribution on the hyper-parameters

θ Hyper-parameter

$m(\mathbf{x})$ mean function

$k(\mathbf{x}, \mathbf{x}')$ covariance function

Information state as a Gaussian Process

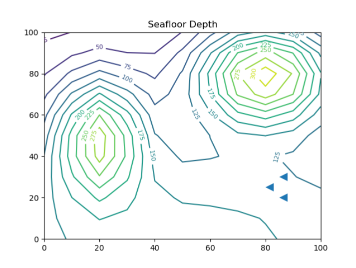
$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big)$$

Pause and Recall ...

- Adaptive Sampling
- Information State and Physical State
  - Ignore physical state for now ...
- Adaptive Sampling on Information State
  - How to model the Information State?
- Information State as a Gaussian Process
  - Mean and covariance function
  - Update
  - Hyper-parameters

Next

- Generic Adaptive Sampling Algorithm
- Adaptive Sampling for Depth
  - Caldera Example

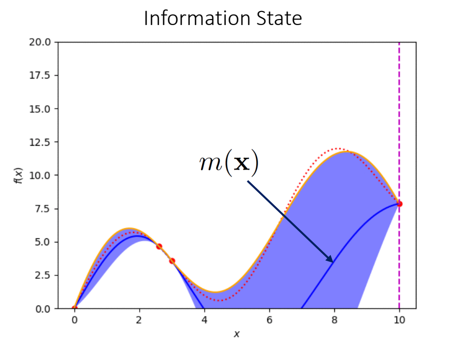


Adaptive Sampling Algorithm: Acquisition Function

Where to sample next?

Acquisition function:

- Defined over the space of interest
- Quantifies how good a sample at that location would be



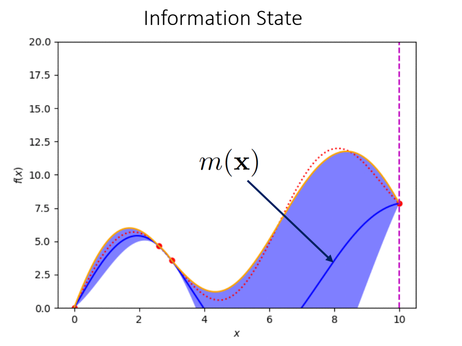
Adaptive Sampling Algorithm: Acquisition Function

Where to sample next?

Acquisition function:

- Defined over the space of interest
- Quantifies how good a sample at that location would be

Sample at the point, where the acquisition function is maximized

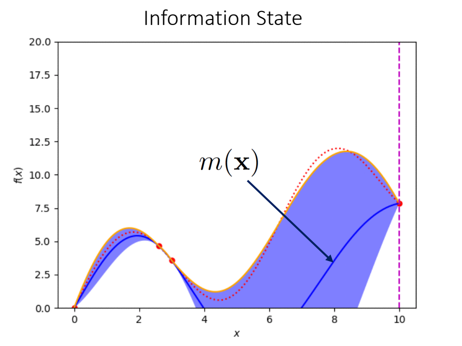


Adaptive Sampling Algorithm: Acquisition Function

Acquisition function:

- Defined over the space of interest
- Quantifies how good a sample at that location would be

Sample at the point, where the acquisition function is maximized

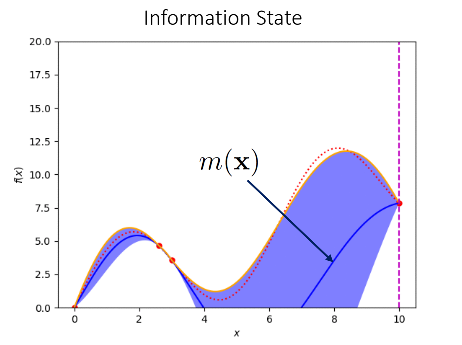


Adaptive Sampling Algorithm: Acquisition Function

Acquisition function:

- Defined over the space of interest
- Quantifies how good a sample at that location would be

Sample at the point, where the acquisition function is maximized



Examples of acquisition functions:

1. Entropy or Variance $h(\mathbf{x}) = \sqrt{k(\mathbf{x}, \mathbf{x})}$
   - Used when we want to reduce uncertainty in our information state
2. Mutual Information
3. UCB: Mean + Variance $h(\mathbf{x}) = m(\mathbf{x}) + \alpha\sqrt{k(\mathbf{x}, \mathbf{x})}$
   - Used when we are interested in the largest mean/deepest point.
   - Caldera example.

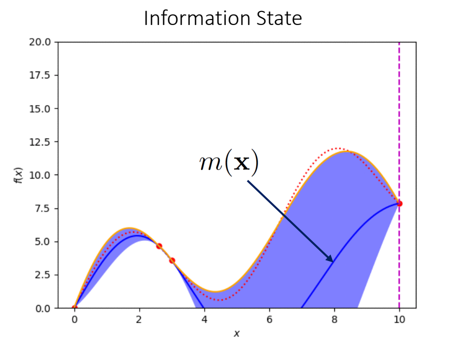
Generic Adaptive Sampling Algorithm

Start with:

1. A prior information state (Gaussian Process)
2. A prior distribution on hyper-parameters
3. A prior acquisition function h()

Iterate:

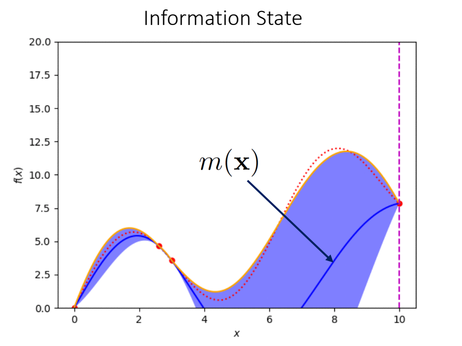
1. Find the sampling location x* = argmax h(x)
2. y* = f(x*) sampling
3. Update the Information State (Gaussian Process)
4. Bayesian update on hyper-parameters
5. Update the acquisition function h()



Generic Adaptive Sampling Algorithm

Start with:

1. A prior information state (Gaussian Process)
2. A prior distribution on hyper-parameters
3. A prior acquisition function h()



Iterate:

1. Find the sampling location x* = argmax h(x)
2. y* = f(x*) sampling
3. Update the Information State (Gaussian Process)
4. Bayesian update on hyper-parameters
5. Update the acquisition function h()

Stopping Criteria:

1. Peaked hyper-parameter distribution
2. Exhausted max number of samples
3. Variance of GP reaches within tolerance bound

Adaptive Sampling Algorithm for Exploring Deepest Point

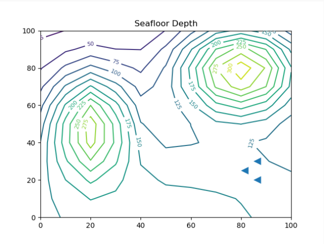
Start with:

1. A prior information state (Gaussian Process)
2. ~~A prior distribution on hyper-parameters~~
3. A prior acquisition function h()

Iterate:

1. Find the sampling location x* = argmax h(x)
2. y* = f(x*) sampling
3. Update the Information State (Gaussian Process)
4. ~~Bayesian update on hyper-parameters~~
5. Update the acquisition function h()

Caldera example



Stopping Criteria:

1. ~~Peaked hyper-parameter distribution~~
2. Exhausted max number of samples
3. ~~Variance of GP reaches within tolerance bound~~

Adaptive Sampling Algorithm for Exploring Deepest Point

Start with:

1. A prior information state (Gaussian Process)
2. A prior acquisition function h($\mathbf{x}$) = $m(\mathbf{x}) + \alpha\sqrt{k(\mathbf{x},\mathbf{x})}$

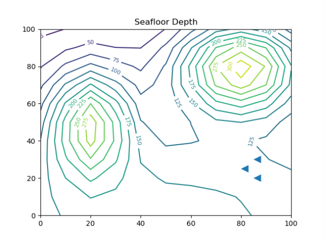
Iterate:

1. Find the sampling location x^* = argmax h(x)
2. y* = f(x^*) sampling
3. Update the Information State (Gaussian Process)
4. Update the acquisition function h()
5. Update the deepest point (with smallest f)

Stopping & Output:

1. When max number of samples used
2. Output the deepest point obtained thus far

Caldera example



Adaptive Sampling Algorithm for Exploring Deepest Point

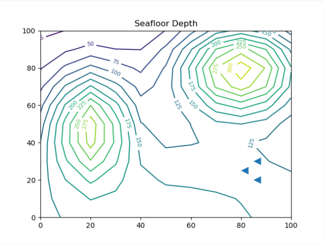
Start with:
1. A prior information state (Gaussian Process)
2. A prior acquisition function h($\mathbf{x}$) = $m(\mathbf{x}) + \alpha\sqrt{k(\mathbf{x},\mathbf{x})}$

UCB

Iterate:
1. Find the sampling location x^* = argmax h(x)
2. y* = f(x^*) sampling
3. Update the Information State (Gaussian Process)
4. Update the acquisition function h()
5. Update the deepest point (with smallest f)

Stopping & Output:
1. When max number of samples used
2. Output the deepest point obtained thus far

Caldera example



Bayesian Optimization

Information State: Fixed vs Adaptive Sampling

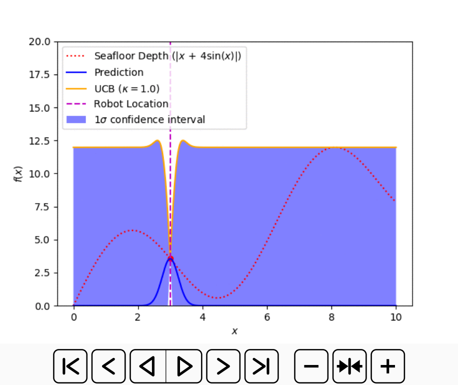


Figure 7: 1D example of fixed sampling.

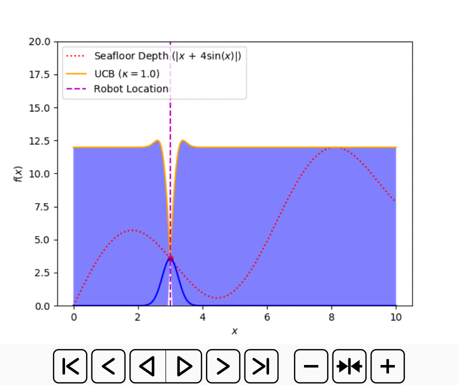


Figure 8: 1D example of Bayesian optimization.

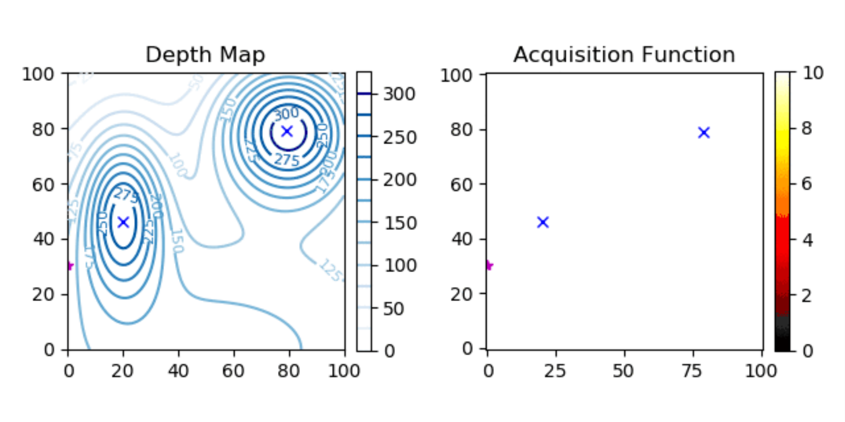


Figure 9: Bayesian optimization solution for low $\kappa = 10$.

Caldera Example

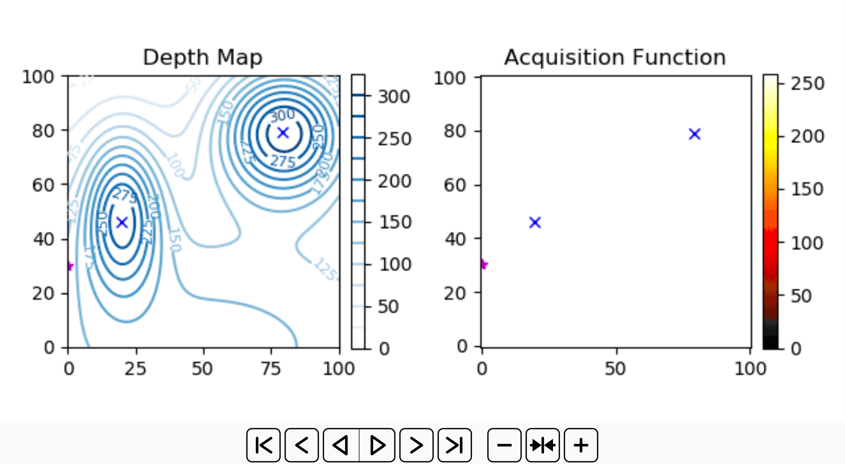
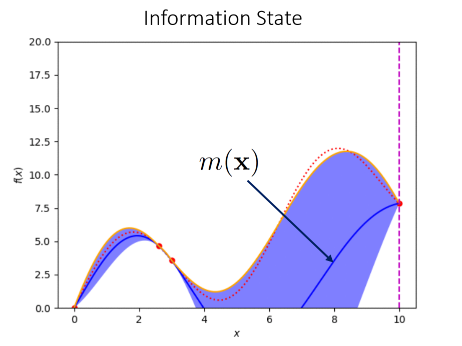


Figure 9: Bayesian optimization solution for high $\kappa = 257$.

Adaptive vs Offline Design

- Is adaptive sampling better than an offline design?
- Recall:
  1. Uncertainty in the information is characterized by the covariance function
  2. The variance update does not depend on what was observed (the sample value)!!



Mean

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}\Big(K(X_*, X)K(X, X)^{-1}\mathbf{f},\ \underbrace{K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)}_{\text{Covariance}}\Big)$$

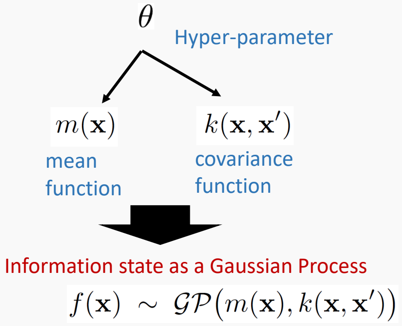
$\mathbf{X}$ sampling location

$\mathbf{f}$ sample value

- Would an offline design work as well?

Adaptive vs Offline Design

- It will (in certain cases) if the hyper-parameters are fixed
    - Our goal is to reduce uncertainty/entropy (variance)



Adaptive vs Offline Design

- It will (in certain cases) if the hyper-parameters are fixed
  - Our goal is to reduce uncertainty/entropy (variance)

- Criteria: sample to maximize entropy

θ Hyper-parameter

$m(\mathbf{x})$ mean function

$k(\mathbf{x}, \mathbf{x}')$ covariance function

Information state as a Gaussian Process

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

Adaptive vs Offline Design

- It will (in certain cases) if the hyper-parameters are fixed
  - Our goal is to reduce uncertainty/entropy (variance)

- Criteria: sample to maximize entropy

θ Hyper-parameter

$m(\mathbf{x})$ mean function

$k(\mathbf{x}, \mathbf{x}')$ covariance function

Information state as a Gaussian Process

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

$$\max_{(\mathbf{x}_1 \ldots \mathbf{x}_m)} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m)) \leq \max_{\pi} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m)) \leq \sum_{\theta} P(\theta) \max_{(\mathbf{x}_1 \ldots \mathbf{x}_m)} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m) | \theta) + H(\theta)$$

optimal offline sampling — optimal adaptive design — optimal offline sampling, given hyper-parameters

Adaptive vs Offline Design

- It will (in certain cases) if the hyper-parameters are fixed
  - Our goal is to reduce uncertainty/entropy (variance)

- <u>Criteria</u>: sample to maximize entropy

If distribution on hyper-parameters is less uncertain, then upper and lower bounds are close

θ Hyper-parameter

$m(\mathbf{x})$ mean function

$k(\mathbf{x}, \mathbf{x}')$ covariance function

Information state as a Gaussian Process

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

$$\max_{(\mathbf{x}_1 \ldots \mathbf{x}_m)} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m)) \leq \max_{\pi} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m)) \leq \sum_{\theta} P(\theta) \max_{(\mathbf{x}_1 \ldots \mathbf{x}_m)} H(f(\mathbf{x}_1), \ldots f(\mathbf{x}_m) | \theta) + H(\theta)$$

optimal offline sampling — optimal adaptive design — optimal offline sampling, given hyper-parameters

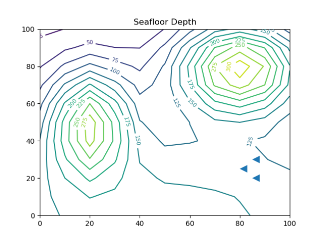
Adaptive vs Offline Design

- It will (in certain cases) if the hyper-parameters are fixed
  - Our goal is to reduce uncertainty/entropy (variance)

- Criteria: sample to maximize entropy

If distribution on hyper-parameters is less uncertain, then upper and lower bounds are close

Caldera example



- Applies to Caldera, if we are interested in learning the depth profile
- Not when we are searching for the deepest point
  - Not the entropy criteria

Pause and Recall …

- Adaptive Sampling
- Information State and Physical State
  - Ignore physical state for now …
- Adaptive Sampling on Information State
  - How to model the Information State?
- Information State as a Gaussian Process
  - Mean and covariance function
  - Update
  - Hyper-parameters

- Generic Adaptive Sampling Algorithm
- Adaptive Sampling for Depth
  - Caldera Example
- Adaptive Sampling vs Offline Design
  - Criteria: Maximize uncertainty
  - Depends on the Entropy of the hyper-parameter distribution

Pause and Recall ...

- Adaptive Sampling
- Information State and Physical State
  - Ignore physical state for now ...
- Adaptive Sampling on Information State
  - How to model the Information State?
- Information State as a Gaussian Process
  - Mean and covariance function
  - Update
  - Hyper-parameters

- Generic Adaptive Sampling Algorithm
- Adaptive Sampling for Depth
  - Caldera Example
- Adaptive Sampling vs Offline Design
  - Criteria: Maximize uncertainty
  - Depends on the Entropy of the hyper-parameter distribution

Next

- Motion Constraints
- Multi-Agent Adaptive Sampling Designs

Table of Contents

Bayesian Optimization for Sampling

1. Find the maxima of the acquisition function $x^* = \arg\max_x h(x)$
2. Plan a path to x^* that avoids obstacles (e.g. using A*)
3. Move to and sample at x^*
4. Update information state using the sample
5. Repeat from #1 until any time/sample constraints are reached

Caldera Example

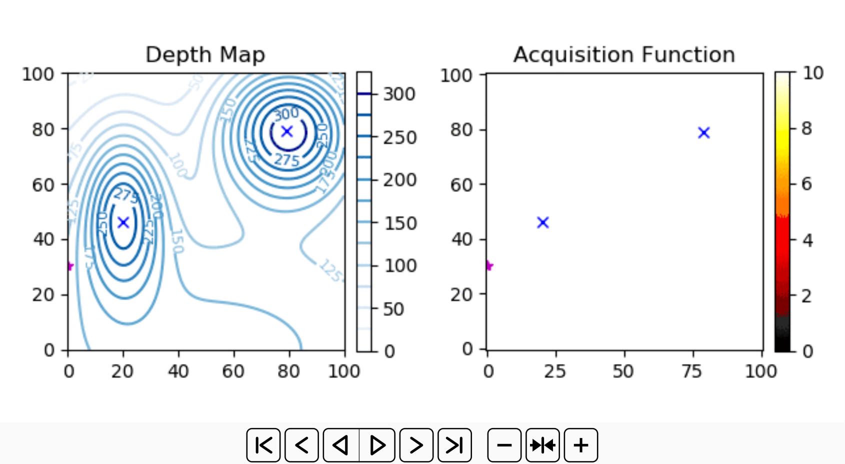


Figure 9: Bayesian optimization solution for low $\kappa = 10$.

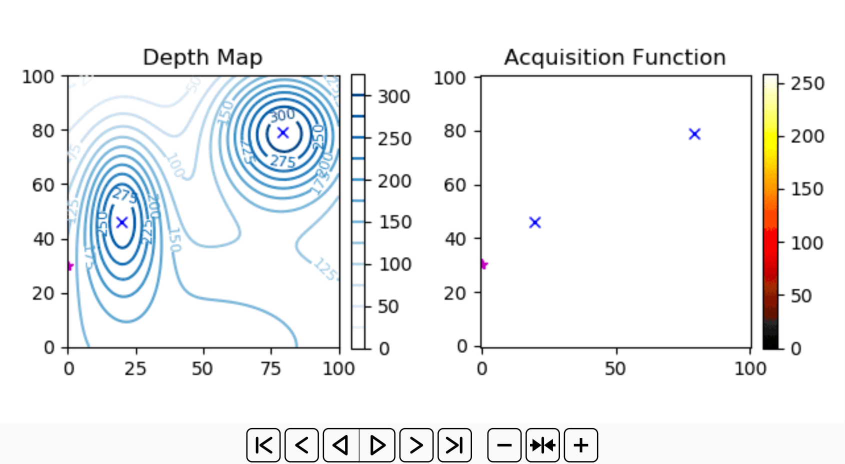


Figure 9: Bayesian optimization solution for high $\kappa = 257$.

Robot Path Planning

- Comments on the quality of the previous paths?
- When are those paths good? When are they bad?
- What more should we take into account?

Physical and Temporal Constraints

- Unlimited time and samples:

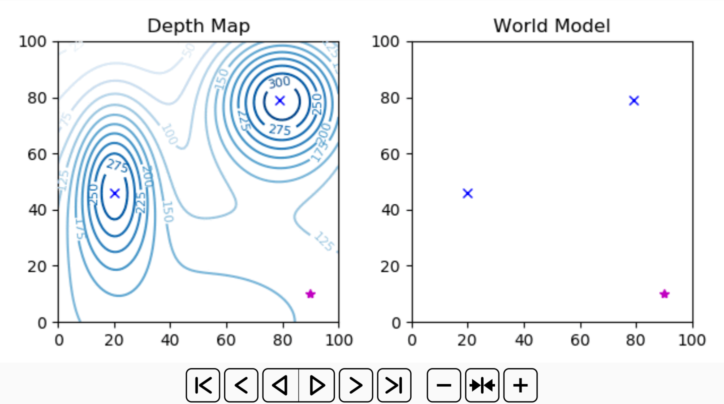


Figure 10: The lawnmower pattern is best if time and number of samples aren't limited.

- Unlimited time, but limited/expensive samples:

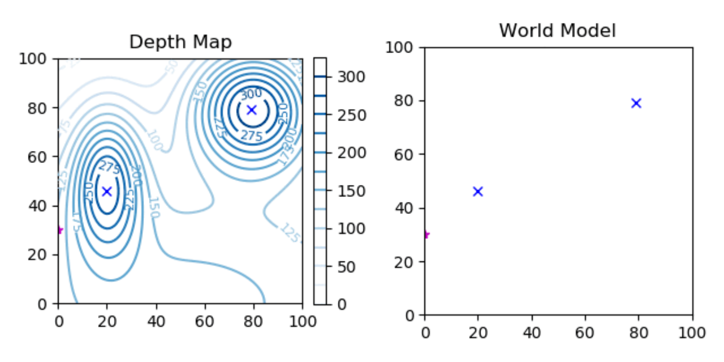


Figure 10: BO only takes samples at locations that maximize our acquisition function.

Physical and Temporal Constraints

- Unlimited free samples, but limited time?

Physical and Temporal Constraints

- Unlimited free samples, but limited time?
- Some combination of constraints (e.g. limited time and samples)?

Physical and Temporal Constraints

- $\boxed{\text{Unlimited free samples, but limited time?}}$
- Some combination of constraints (e.g. limited time and samples)?

Robot Modelling

The robot state $X(t)$ contains the physical state of the robot at time t, such as:

- Robot location
- Robot velocity
- Battery level
- Water current directions and flow rate
- ...

Robot Modelling

The robot state $X(t)$ contains the physical state of the robot at time t, such as:

- Robot location
- Robot velocity
- Battery level
- Water current directions and flow rate
- ...

In general, $X(t)$ can be partially or fully observable.

Simplified Robot Model

- We use a simple 2D and **fully observable** robot state:

$$X_t = \langle x_t, y_t \rangle$$

Simplified Robot Model

- We use a simple 2D and **fully observable** robot state:

$$X_t = \langle x_t, y_t \rangle$$

- We discretize time and space:

$$x_t, y_t \in \mathbb{Z}$$
$$t = 1, \ldots, T$$

Simplified Robot Model

- We use a simple 2D and **fully observable** robot state:

$$X_t = \langle x_t, y_t \rangle$$

- We discretize time and space:

$$x_t, y_t \in \mathbb{Z}$$
$$t = 1, \ldots, T$$

- Our information state is the GP depth model based on our samples:

$$\mathcal{I}_t = \mathrm{GP}(S_t)$$
$$S_t = \{\langle x_i, y_i, f_{\mathrm{obs}}(x_i, y_i)\rangle\}_{i=1}^{k_t}$$

Robot Action Model

Our robots have a variety of **actions**:

- Moving

Robot Action Model

Our robots have a variety of **actions**:

- Moving
  - Limited speed

Robot Action Model

Our robots have a variety of **actions**:

- Moving
    - Limited speed
    - Motion constraints

Robot Action Model

Our robots have a variety of **actions**:

- Moving
  - Limited speed
  - Motion constraints
- Using a sensor
  - Free: Taking a depth measurement

Robot Action Model

Our robots have a variety of **actions**:

- Moving
  - Limited speed
  - Motion constraints
- Using a sensor
  - Free: Taking a depth measurement
  - Limited uses: Taking a water sample

Robot Action Model

Our robots have a variety of **actions**:

- Moving
  - Limited speed
  - Motion constraints
- Using a sensor
  - Free: Taking a depth measurement
  - Limited uses: Taking a water sample
  - Costly: Picking up a rock sample

Simplified Action Model

- Move 1 unit in one of {`N, W, S, E`} in unit time
- Sample depth immediately after every move

Graph-Based Problem Formulation

$$\text{Robot State: } X_t = \langle x_t, y_t \rangle$$

$$\text{Information State: } \mathcal{I}_t = \mathrm{GP}(S_t),\ S_t = \{\langle x_i, y_i, f_{\text{obs}}(x_i, y_i)\rangle\}_{i=1}^{k_t}$$

Graph-Based Problem Formulation

$$\text{Robot State: } X_t = \langle x_t, y_t \rangle$$

$$\text{Information State: } \mathcal{I}_t = \mathrm{GP}(S_t),\ S_t = \{\langle x_i, y_i, f_{\mathrm{obs}}(x_i, y_i)\rangle\}_{i=1}^{k_t}$$

$$\text{Robot Actions: } a_t \in A = \{\mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$$

Graph-Based Problem Formulation

$$\text{Robot State: } X_t = \langle x_t, y_t \rangle$$

$$\text{Information State: } \mathcal{I}_t = \text{GP}(S_t),\ S_t = \{\langle x_i, y_i, f_{\text{obs}}(x_i, y_i)\rangle\}_{i=1}^{k_t}$$

$$\text{Robot Actions: } a_t \in A = \{\texttt{N}, \texttt{E}, \texttt{S}, \texttt{W}\}$$

$$\text{Physical Update: } X_{t+1} = \begin{cases} \langle x_t, y_t + 1 \rangle, & a_t = \texttt{N} \\ \vdots & \vdots \end{cases}$$

Graph-Based Problem Formulation

$$\text{Robot State: } X_t = \langle x_t, y_t \rangle$$

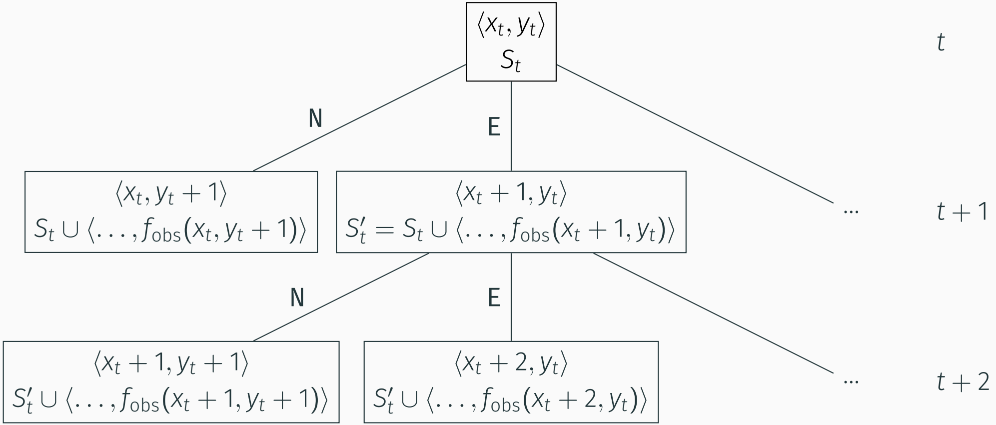
$$\text{Information State: } \mathcal{I}_t = \mathrm{GP}(S_t),\ S_t = \{\langle x_i, y_i, f_{\mathrm{obs}}(x_i, y_i)\rangle\}_{i=1}^{k_t}$$

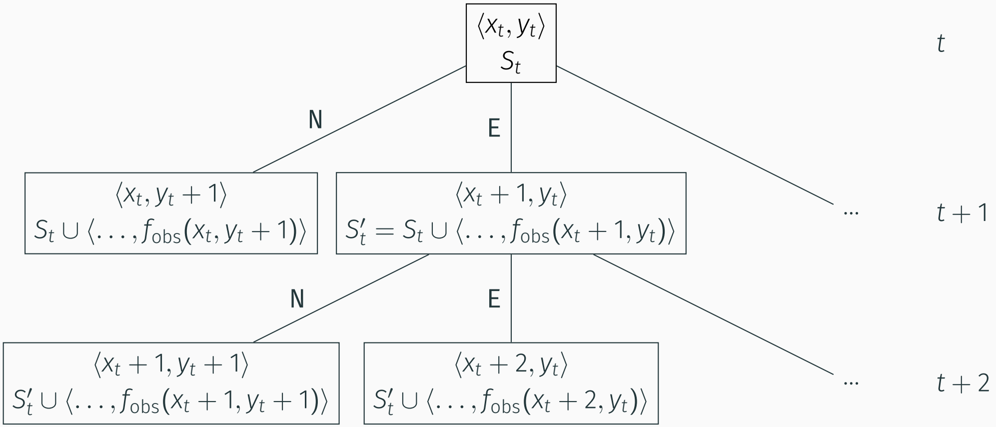
$$\text{Robot Actions: } a_t \in A = \{\mathtt{N}, \mathtt{E}, \mathtt{S}, \mathtt{W}\}$$

$$\text{Physical Update: } X_{t+1} = \begin{cases} \langle x_t, y_t + 1 \rangle\,, & a_t = \mathtt{N} \\ \vdots & \vdots \end{cases}$$

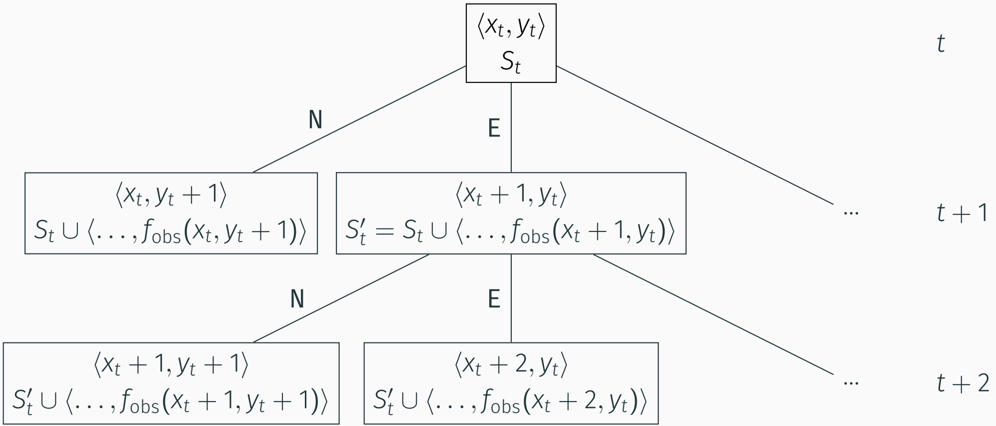
$$\text{Information Update: } S_{t+1} = S_t \cup \langle x_{t+1}, y_{t+1}, f_{\mathrm{obs}}(x_{t+1}, y_{t+1})\rangle$$

Graph-Based Problem Formulation

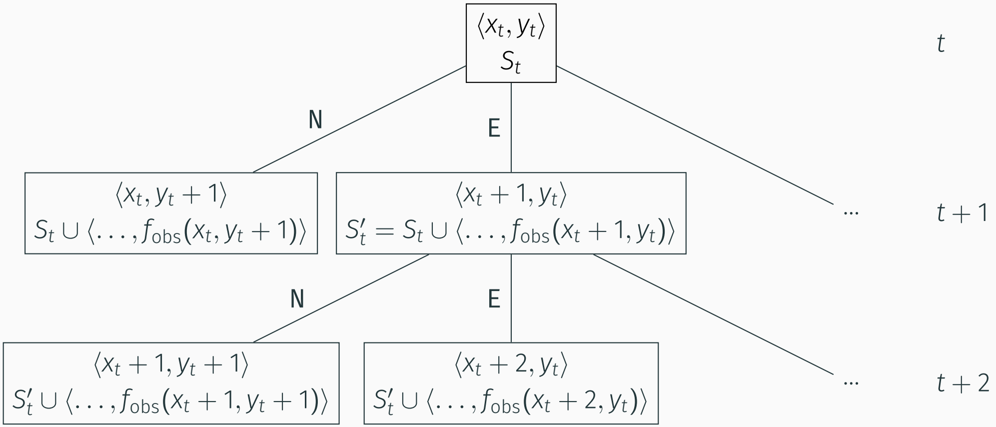




Is this a graph or a tree?

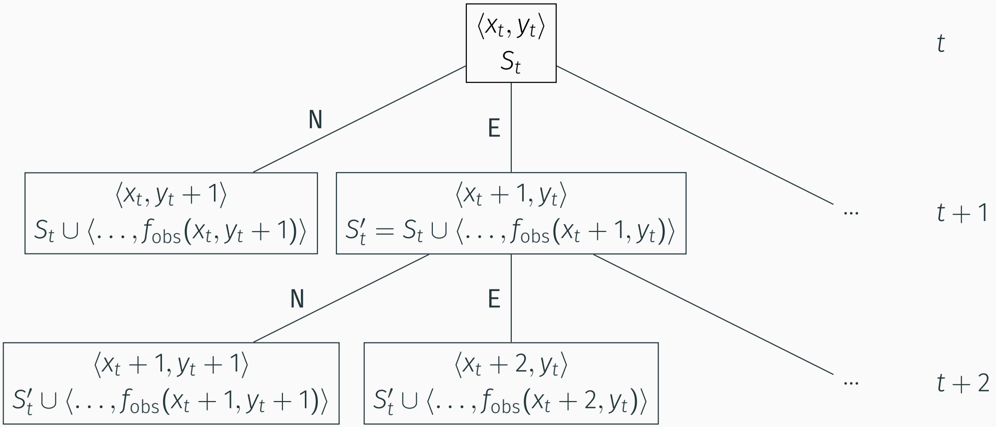


Are there any terminal states?



Do we have a goal state?

Graph-Based Problem Formulation



How should we pick an action at each time step?

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$
- Set of Actions $a_t \in \mathcal{A}$

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$
- Set of Actions $a_t \in \mathcal{A}$
- Transition Relation $\delta : (s_1, a) \mapsto (s_2)$

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$
- Set of Actions $a_t \in \mathcal{A}$
- Transition Relation $\delta : (s_1, a) \mapsto (s_2)$
- Reward Function $R : (s, a) \mapsto r \in \mathbb{R}$

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$
- Set of Actions $a_t \in \mathcal{A}$
- Transition Relation $\delta : (s_1, a) \mapsto (s_2)$
- Reward Function $R : (s, a) \mapsto r \in \mathbb{R}$

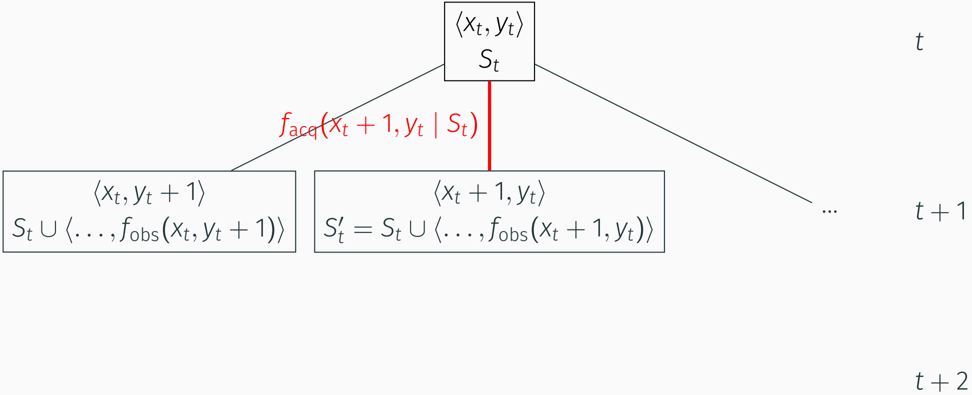
$$R(X_t, a_t) = ?$$

Evaluating Actions

Recall the features of a Markov Decision Process (MDP):

- Set of States $\langle X_t, \mathcal{I}_t \rangle$
- Set of Actions $a_t \in \mathcal{A}$
- Transition Relation $\delta : (s_1, a) \mapsto (s_2)$
- Reward Function $R : (s, a) \mapsto r \in \mathbb{R}$

$$R(X_t, a_t) = f_{\mathrm{acq}}(X_t' \mid S_t)$$
$$f_{\mathrm{acq}}(X_t \mid S_t) = \mu(X_t \mid S_t) + \kappa\sigma(X_t \mid S_t)$$


$\langle x_t, y_t \rangle$
S_t
$f_{\text{acq}}(x_t + 1, y_t \mid S_t)$
$\langle x_t, y_t + 1 \rangle$
$S_t \cup \langle \ldots, f_{\text{obs}}(x_t, y_t + 1) \rangle$
$\langle x_t + 1, y_t \rangle$
$S'_t = S_t \cup \langle \ldots, f_{\text{obs}}(x_t + 1, y_t) \rangle$
...
t
$t + 1$
$t + 2$

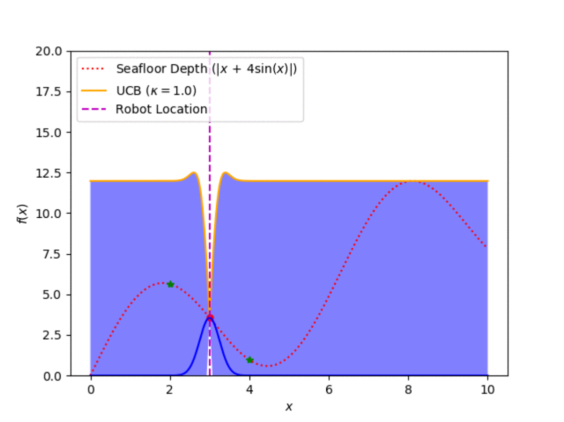
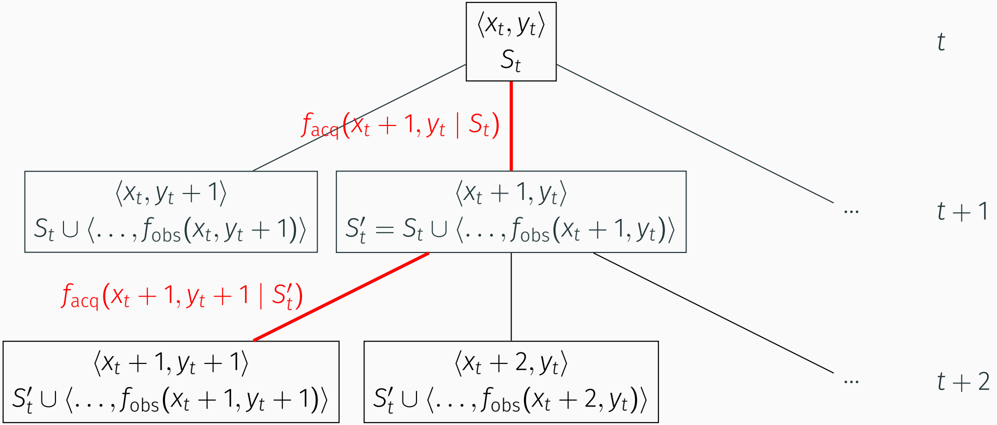


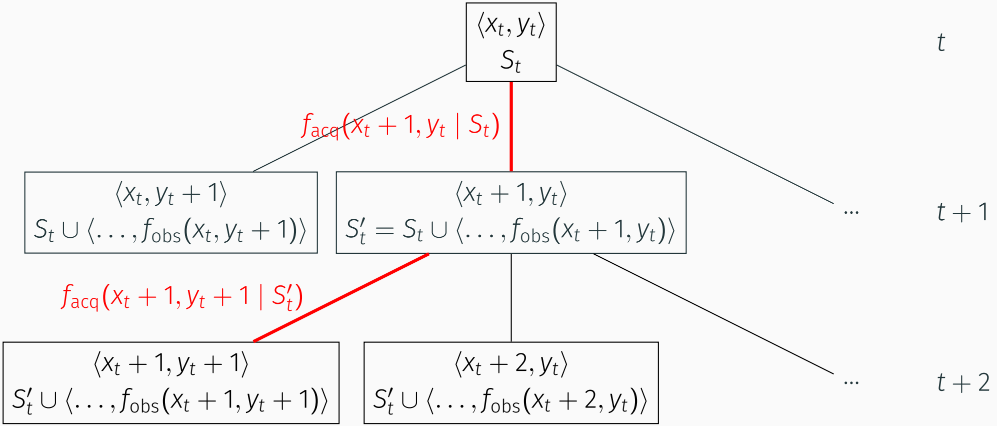
Figure 10: Result of choosing the best action (green marks) based on the acquisition function.

Evaluating Paths



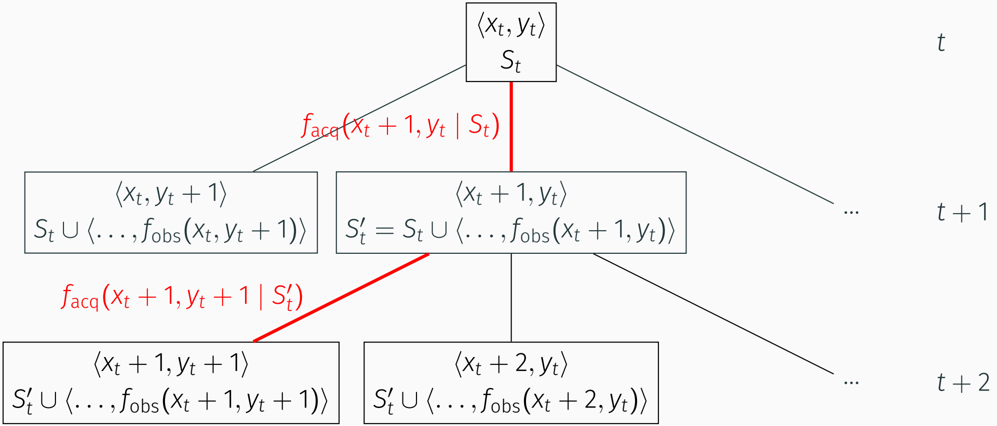
Evaluating Paths

What is S_t' in this example?



Evaluating Paths

What is S'_t in this example? $\mathbb{E}(S'_t) = S_t \cup \langle \ldots, \mu(x_t + 1, y_t \mid S_t) \rangle$



Planning based on Simulation

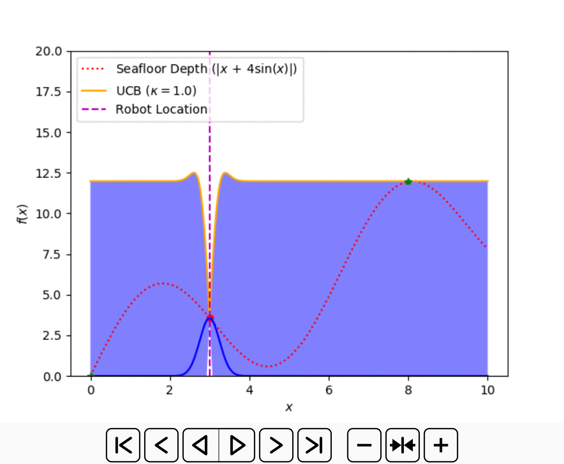


Figure 11: Result of choosing the best action based on highest scoring path of depth 5.

Sequential Bayesian Optimization

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$

[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$
2. Increase accumulated reward by $R(X_{t+1}) = f_{\text{acq}}(x_{t+1}, y_{t+1} \mid S_t)$

[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$
2. Increase accumulated reward by $R(X_{t+1}) = f_{\text{acq}}(x_{t+1}, y_{t+1} \mid S_t)$
3. Simulate a sample: $S_{t+1} = S_t \cup \{x_{t+1}, y_{t+1}, \mu(x_{t+1}, y_{t+1} \mid S_t)\}$

[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$
2. Increase accumulated reward by $R(X_{t+1}) = f_{\text{acq}}(x_{t+1}, y_{t+1} \mid S_t)$
3. Simulate a sample: $S_{t+1} = S_t \cup \{x_{t+1}, y_{t+1}, \mu(x_{t+1}, y_{t+1} \mid S_t)\}$
4. Repeat from #1 for every move action $a_{t+1} \in A$ at the new state

[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$
2. Increase accumulated reward by $R(X_{t+1}) = f_{\text{acq}}(x_{t+1}, y_{t+1} \mid S_t)$
3. Simulate a sample: $S_{t+1} = S_t \cup \{x_{t+1}, y_{t+1}, \mu(x_{t+1}, y_{t+1} \mid S_t)\}$
4. Repeat from #1 for every move action $a_{t+1} \in A$ at the new state
5. Increase the accumulated reward:

$$R(X_t, a_t) \leftarrow R(X_t, a_t) + \max_{a_{t+1} \in A} R(\delta(X_{t+1}, a_{t+1}))$$

[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization

Sequential Bayesian Optimization Algorithm:[2]

Initial state $X_t = \langle x_t, y_t, S_t \rangle$. For each move action $a_t \in A$:

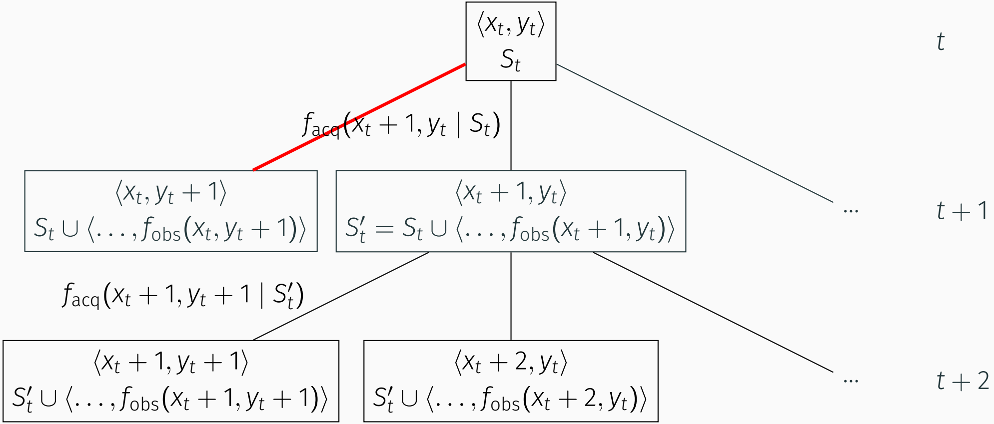
1. Simulate the move: $X_{t+1} = \delta(X_t, a_t)$
2. Increase accumulated reward by $R(X_{t+1}) = f_{\text{acq}}(x_{t+1}, y_{t+1} \mid S_t)$
3. Simulate a sample: $S_{t+1} = S_t \cup \{x_{t+1}, y_{t+1}, \mu(x_{t+1}, y_{t+1} \mid S_t)\}$
4. Repeat from #1 for every move action $a_{t+1} \in A$ at the new state
5. Increase the accumulated reward:

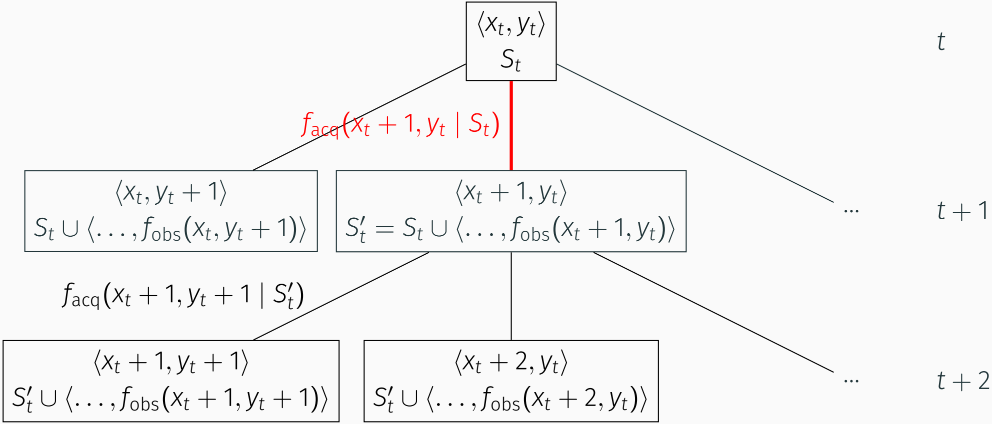
$$R(X_t, a_t) \leftarrow R(X_t, a_t) + \max_{a_{t+1} \in A} R(\delta(X_{t+1}, a_{t+1}))$$

6. Choose the action a_t with the largest accumulated reward

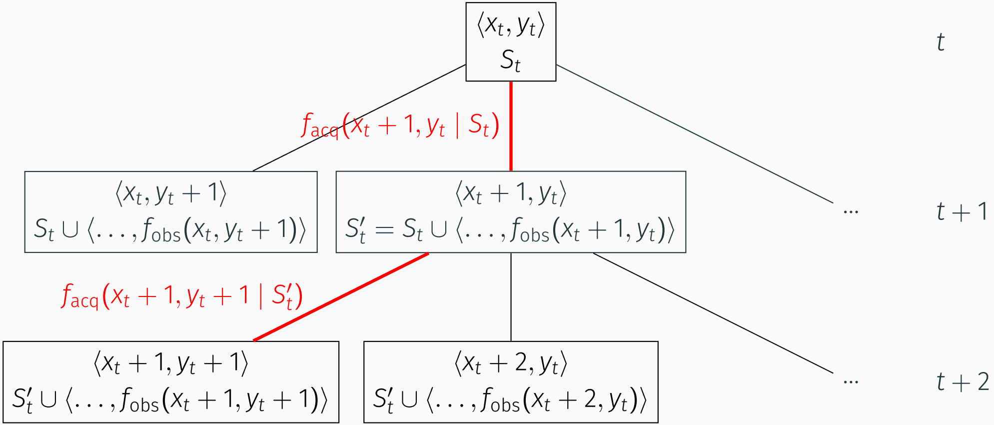
[2] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Sequential Bayesian Optimization





$\langle x_t, y_t \rangle$
S_t
t
$f_{\text{acq}}(x_t + 1, y_t \mid S_t)$
$\langle x_t, y_t + 1 \rangle$
$S_t \cup \langle \ldots, f_{\text{obs}}(x_t, y_t + 1) \rangle$
$\langle x_t + 1, y_t \rangle$
$S'_t = S_t \cup \langle \ldots, f_{\text{obs}}(x_t + 1, y_t) \rangle$
...
$t + 1$
$f_{\text{acq}}(x_t + 1, y_t + 1 \mid S'_t)$
$\langle x_t + 1, y_t + 1 \rangle$
$S'_t \cup \langle \ldots, f_{\text{obs}}(x_t + 1, y_t + 1) \rangle$
$\langle x_t + 2, y_t \rangle$
$S'_t \cup \langle \ldots, f_{\text{obs}}(x_t + 2, y_t) \rangle$
...
$t + 2$


$\langle x_t, y_t \rangle$
S_t
t
$f_{\text{acq}}(x_t + 1, y_t \mid S_t)$
$\langle x_t, y_t + 1 \rangle$
$S_t \cup \langle \ldots, f_{\text{obs}}(x_t, y_t + 1) \rangle$
$\langle x_t + 1, y_t \rangle$
$S_t' = S_t \cup \langle \ldots, f_{\text{obs}}(x_t + 1, y_t) \rangle$
...
$t + 1$
$f_{\text{acq}}(x_t + 1, y_t + 1 \mid S_t')$
$\langle x_t + 1, y_t + 1 \rangle$
$S_t' \cup \langle \ldots, f_{\text{obs}}(x_t + 1, y_t + 1) \rangle$
$\langle x_t + 2, y_t \rangle$
$S_t' \cup \langle \ldots, f_{\text{obs}}(x_t + 2, y_t) \rangle$
...
$t + 2$

Advantages of Sequential BO

- Easily incorporate constraints:
  - Remain in safe region: $x_t \in [a, b]\,, y_t \in [c, d]$

Advantages of Sequential BO

- Easily incorporate constraints:
  - Remain in safe region: $x_t \in [a, b]\,, y_t \in [c, d]$
  - Limited mission duration: restrict $t \leq T$

Advantages of Sequential BO

- Easily incorporate constraints:
  - Remain in safe region: $x_t \in [a, b]\,, y_t \in [c, d]$
  - Limited mission duration: restrict $t \leq T$
  - Limited number of samples: add `sample` action, restrict $k \leq K$

Advantages of Sequential BO

- Easily incorporate constraints:
  - Remain in safe region: $x_t \in [a, b]$, $y_t \in [c, d]$
  - Limited mission duration: restrict $t \leq T$
  - Limited number of samples: add `sample` action, restrict $k \leq K$
  - Cost of taking a sample: $R(X, \texttt{sample}) = \ldots$
- For continuous time planning, give each action a duration Δt and make time an element of the state

Advantages of Sequential BO

- Easily incorporate constraints:
  - Remain in safe region: $x_t \in [a, b]$, $y_t \in [c, d]$
  - Limited mission duration: restrict $t \leq T$
  - Limited number of samples: add `sample` action, restrict $k \leq K$
  - Cost of taking a sample: $R(X, \texttt{sample}) = \dots$
- For continuous time planning, give each action a duration Δt and make time an element of the state
- Easily extended to a richer set of actions (e.g. motion primitives[3])

[3] Marchant, Ramos, and Sanner, "Sequential Bayesian Optimisation for Spatial-Temporal Monitoring", 2014.

Challenges of Sequential BO

- $|A|^T$ actions to consider

Challenges of Sequential BO

- $|A|^T$ actions to consider
    - 4 actions, 10 seconds per action, plan next 5 minutes: $4^{30} \approx 10^{20}$ states to consider

Challenges of Sequential BO

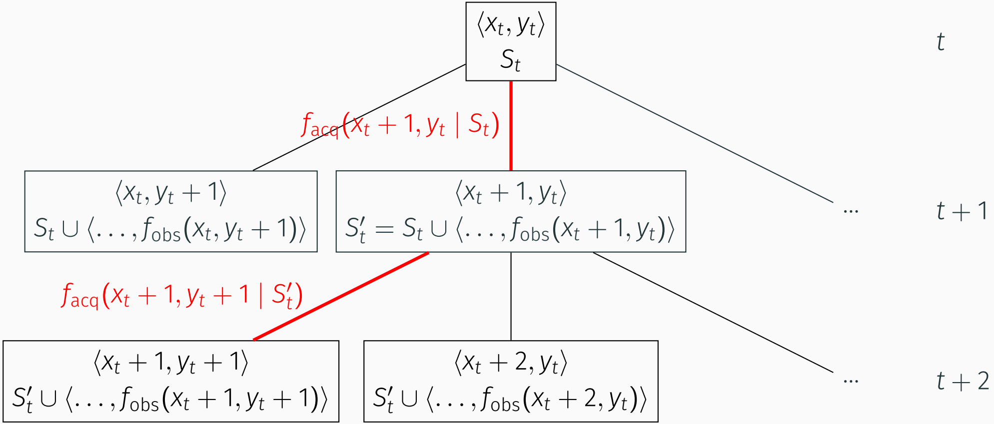
- $|A|^T$ actions to consider
  - 4 actions, 10 seconds per action, plan next 5 minutes: $4^{30} \approx 10^{20}$ states to consider
  - Still guaranteed to find optimal solution if we truncate? What if we used a different acquisition function (not UCB)?

Challenges of Sequential BO

- $|A|^T$ actions to consider
  - 4 actions, 10 seconds per action, plan next 5 minutes: $4^{30} \approx 10^{20}$ states to consider
  - Still guaranteed to find optimal solution if we truncate? What if we used a different acquisition function (not UCB)?
- The right reward function for your application may be non-obvious

Monte Carlo Tree Search

One way to search the tree to a deeper depth is to use MCTS. In MCTS, we randomly sample paths, and the probability of choosing an action proportional to the average reward it has led to in past iterations.



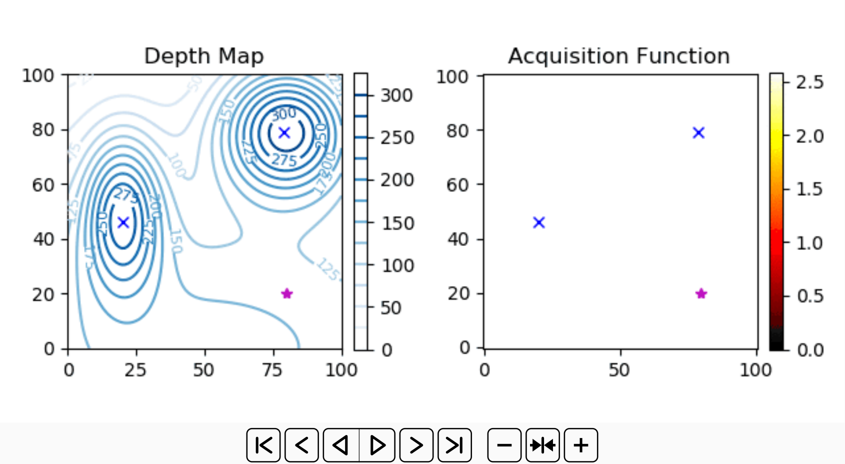


Figure 12: SBO with $\kappa = 2.576$

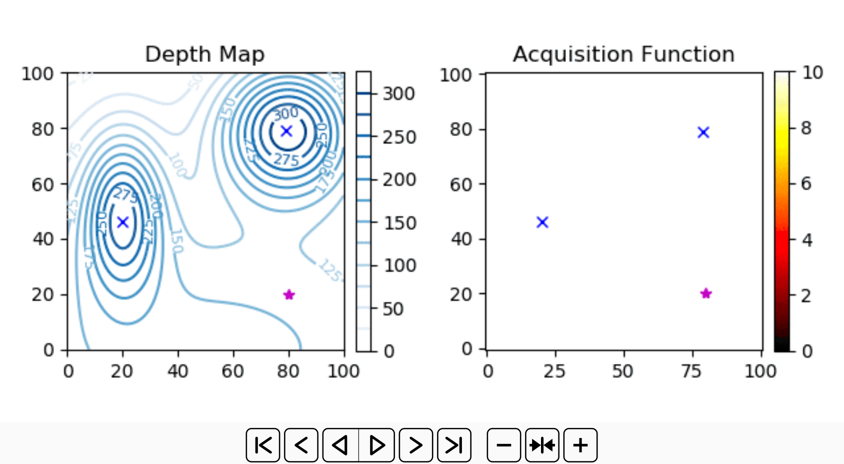


Figure 12: More explorative SBO with $\kappa = 2.576$

Table of Contents

Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots

Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots
- But, if agents don't know what each other are up to (or what samples have been collected) then they may collect redundant samples

Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots
- But, if agents don't know what each other are up to (or what samples have been collected) then they may collect redundant samples
  - Loses the main benefit of adaptive sampling

Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots
- But, if agents don't know what each other are up to (or what samples have been collected) then they may collect redundant samples
  - Loses the main benefit of adaptive sampling
- Multi-agent sampling policies are based on avoiding this, using minimal communication (bandwidth)

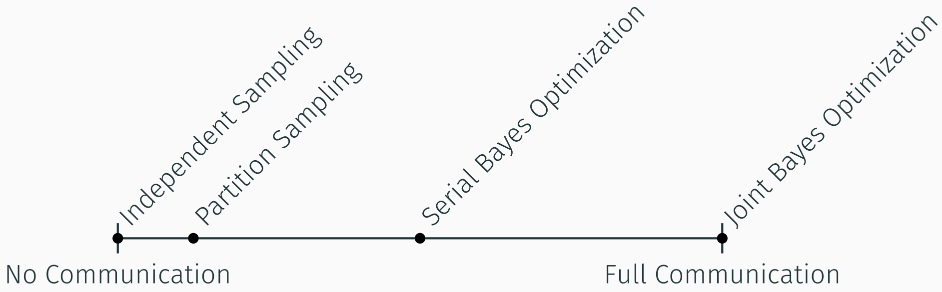
Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots
- But, if agents don't know what each other are up to (or what samples have been collected) then they may collect redundant samples
  - Loses the main benefit of adaptive sampling
- Multi-agent sampling policies are based on avoiding this, using minimal communication (bandwidth)
  - Why? Some environments are highly bandwidth constrained (e.g. underwater)

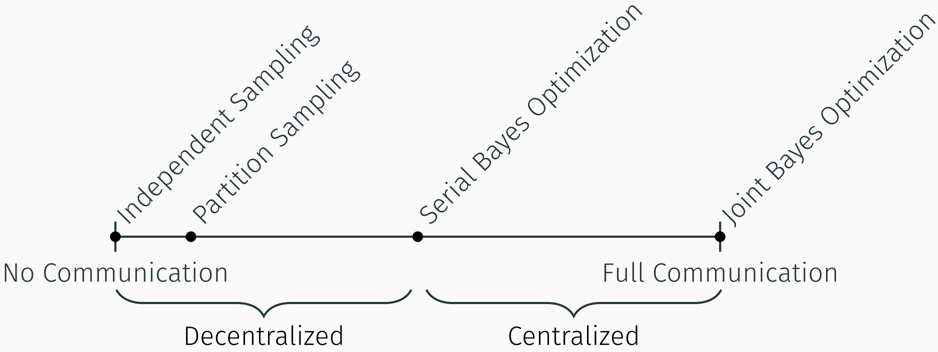
Multi-Agent State Space Search

- We can collect more samples in the same amount of time using multiple robots
- But, if agents don't know what each other are up to (or what samples have been collected) then they may collect redundant samples
    - Loses the main benefit of adaptive sampling
- Multi-agent sampling policies are based on avoiding this, using minimal communication (bandwidth)
    - Why? Some environments are highly bandwidth constrained (e.g. underwater)
    - Bandwidth consumed scales at least linearly with the number of agents

Multi-Agent Sampling Policies



Multi-Agent Sampling Policies



Independent Sampling

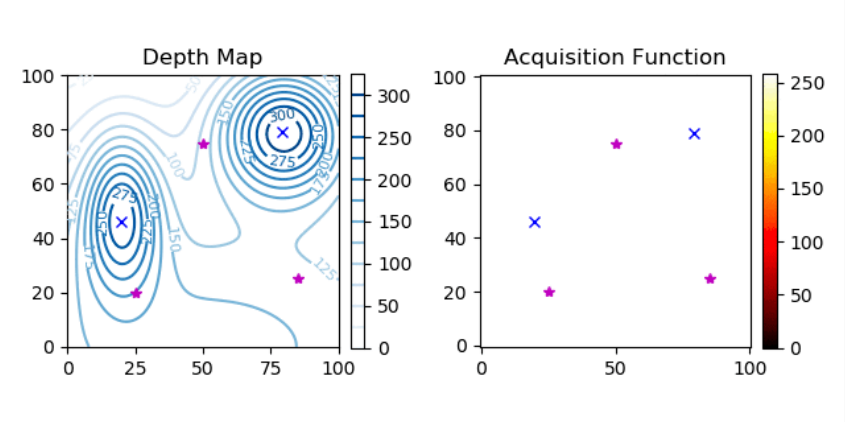


Figure 13: Independent sampling for 3 agents.

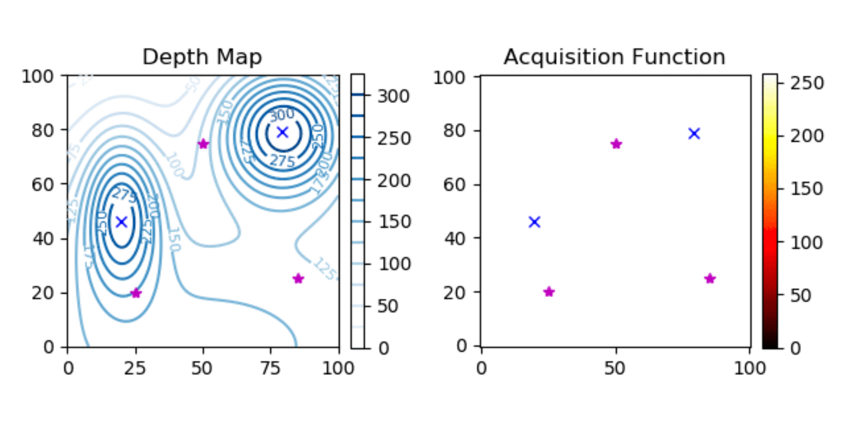
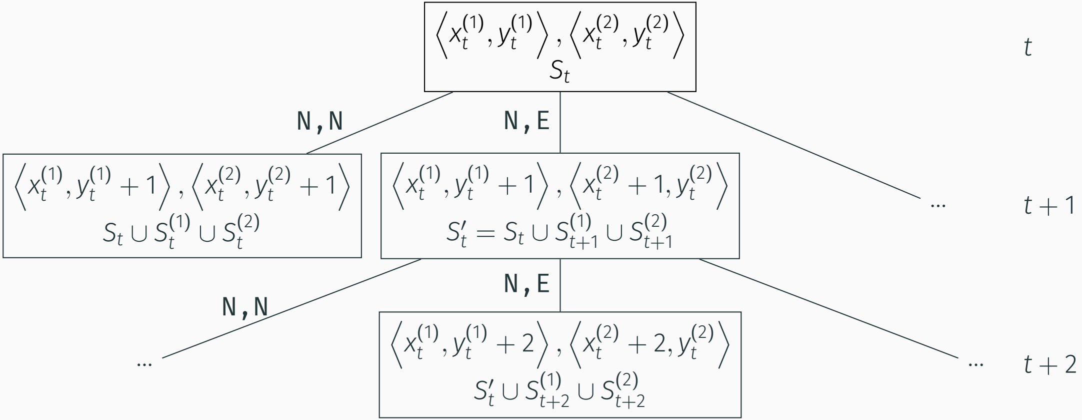


Figure 14: Partition sampling for 3 agents.

Joint Bayesian Optimization



Branching factor is now $|A|^N$ so tree size is $\left(|A|^N\right)^T$; not generally practical.

Serial Bayesian Optimization

Pick each robot's action one by one:

1. Once a robot (#1) has picked its next move with SBO, it tells nearby robots

Serial Bayesian Optimization

Pick each robot's action one by one:

1. Once a robot (#1) has picked its next move with SBO, it tells nearby robots
2. The nearby robots add simulated samples to their models (information states) based on what they expect robot #1's sample to be: $\mu\left(X^{(1)} \mid S^{(2)}\right)$

Serial Bayesian Optimization

Pick each robot's action one by one:

1. Once a robot (#1) has picked its next move with SBO, it tells nearby robots
2. The nearby robots add simulated samples to their models (information states) based on what they expect robot #1's sample to be: $\mu\left(X^{(1)} \mid S^{(2)}\right)$
    - Then, if any of these robots need to pick their next action in the meantime, they avoid sampling the same places as robot #1

Serial Bayesian Optimization

Pick each robot's action one by one:

1. Once a robot (#1) has picked its next move with SBO, it tells nearby robots
2. The nearby robots add simulated samples to their models (information states) based on what they expect robot #1's sample to be: $\mu\left(X^{(1)} \mid S^{(2)}\right)$
   - Then, if any of these robots need to pick their next action in the meantime, they avoid sampling the same places as robot #1
3. Once robot #1 has collected the sample, it tells the nearby robots

Serial Bayesian Optimization

Pick each robot's action one by one:

1. Once a robot (#1) has picked its next move with SBO, it tells nearby robots
2. The nearby robots add simulated samples to their models (information states) based on what they expect robot #1's sample to be: $\mu\left(X^{(1)} \mid S^{(2)}\right)$
    - Then, if any of these robots need to pick their next action in the meantime, they avoid sampling the same places as robot #1
3. Once robot #1 has collected the sample, it tells the nearby robots
    - They replace the simulated sample value with the real one

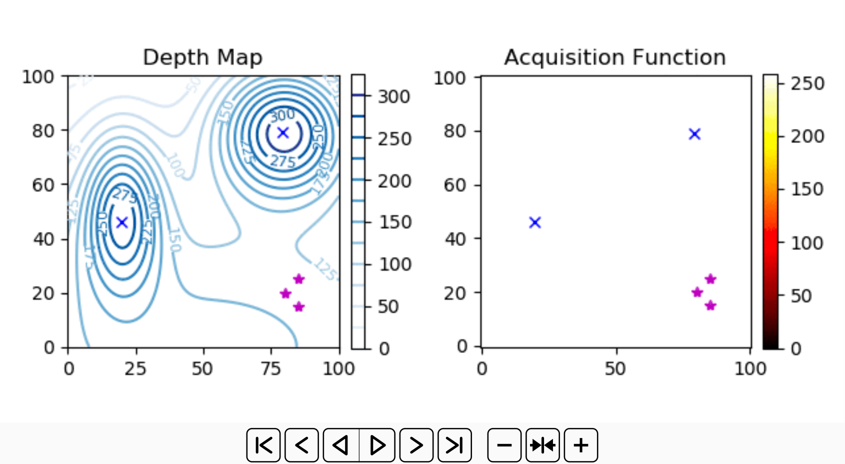
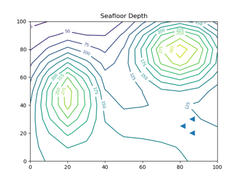


Figure 15: Serial Bayesian Optimization with 3 agents.

Table of Contents

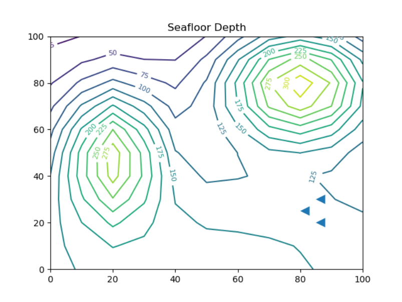
Summary

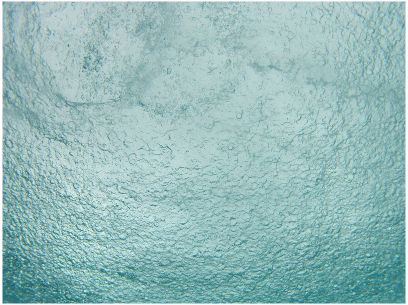
- What is Sampling and when is it useful ?
- Fixed vs. Adaptive sampling
  - The more uncertainty we have about the model parameters, the more effective adaptive sampling is (the exploration vs. exploitation balance)
- Accounting for both the information state and physical state
  - The implications of the robot's physical constraints on the sampling process
- Multiple agent adaptive sampling
  - The challenges of coordinating a multi-agent setting for safe and effective sampling
  - Looking at various levels of communication



Tutorial

The tutorial will allow you to explore our example caldera. We don't know the model parameters, but assume it is a GP



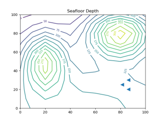


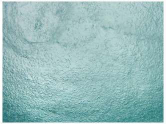
Since we are not actually exploring the caldera, We will provide a simulator.

Tutorial

The tutorial will allow you to:

- experience the results acquired a fixed random sampling
- get a chance to see what happens when you apply an adaptive approach
- account for the physical constraints (and introduce an activity model)
- see what happens when many agents participate in the sampling task





Problem Set

In the problem set you will:

- Test a Single-Agent Adaptive Sampling implementation to explore the performance of adaptive sampling
- Implement a Multi-Agent Adaptive Sampling algorithm based on a Decentralized approach
- Implement a Multi-Agent Adaptive Sampling algorithm based on a Centralized approach

Questions?

References i

- H. Choset and P. Pignon, “Coverage Path Planning: The Boustrophedon Cellular Decomposition”, *Field and Service Robotics*, pp. 203–209, 1998, ISSN: 09295593. DOI: 10.1007/978-1-4471-1273-0{_}32.
- G. Hitz, E. Galceran, M. È. Garneau, F. Pomerleau, and R. Siegwart, “Adaptive continuous-space informative path planning for online environmental monitoring”, *Journal of Field Robotics*, vol. 34, no. 8, pp. 1427–1449, Dec. 2017, ISSN: 15564967. DOI: 10.1002/rob.21722. [Online]. Available: http://doi.wiley.com/10.1002/rob.21722.
- G. Flaspohler, V. Preston, A. P. Michel, Y. Girdhar, and N. Roy, “Information-Guided Robotic Maximum Search in Partially Observable Continuous Environments”, *[Submitted to] Intelligent Robots and Systems (IROS)*, 2019.

R. Marchant, F. Ramos, and S. Sanner, “Sequential Bayesian Optimisation for Spatial-Temporal Monitoring”, *International Conference on Uncertainty in Artificial Intelligence*, pp. 553–562, 2014.