Active Learning for Level Set Estimation

Alkis Gotovos[1] Nathalie Casati[1,2] Gregory Hitz[1] Andreas Krause[1]

[1]Department of Computer Science
ETH Zurich

[2]IBM Research – Zurich

IJCAI '13

Swimmers of Lake Zurich, beware!

Steffen Schmidt / EPA

Swimmers of Lake Zurich, beware!

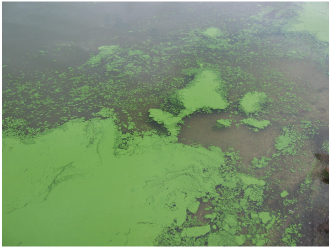
> "The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"
>
> — *Scientific American*

Swimmers of Lake Zurich, beware!

www.limnobotics.ch

“The warming waters of one of central Europe’s most popular holiday destinations, Switzerland’s Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]”

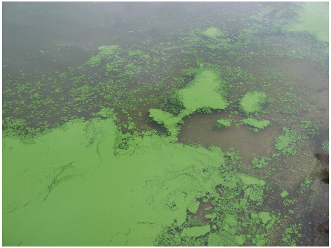
— *Scientific American*



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

Swimmers of Lake Zurich, beware!

www.limnobotics.ch



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

"The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"

— *Scientific American*

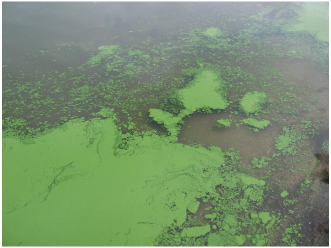
"*Planktotrhix rubescens* can account for half of the total phytoplankton biomass in Lake Zurich in summer [...]"

"*Planktotrhix rubescens* are among the most important producers of hepatotoxic microcystins in freshwaters [...]"

— *Silke Van den Wyngaert* et al., *ASLO, 2011*

Swimmers of Lake Zurich, beware!

www.limnobotics.ch



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

"The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"

— *Scientific American*

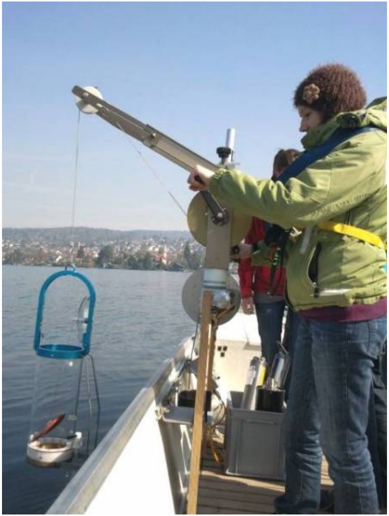
"*Planktotrhix rubescens* can account for half of the total phytoplankton biomass in Lake Zurich in summer [...]"

"*Planktotrhix rubescens* are among the most important producers of hepatotoxic microcystins in freshwaters [...]"

— *Silke Van den Wyngaert* et al., *ASLO, 2011*

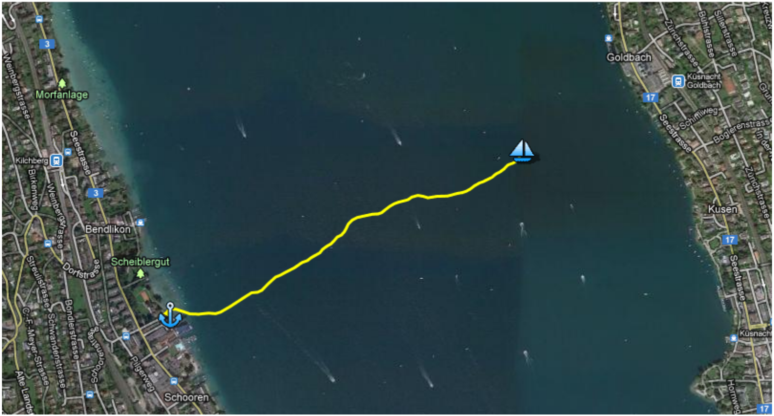
"Microcystins [...] are cyanotoxins and can be very toxic for plants and animals including humans. Their hepatotoxicity may cause serious damage to the liver."

— *Wikipedia*

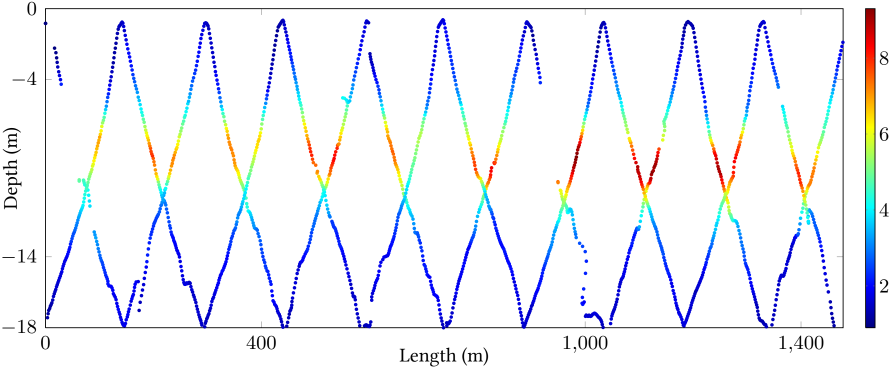


www.limnobotics.ch

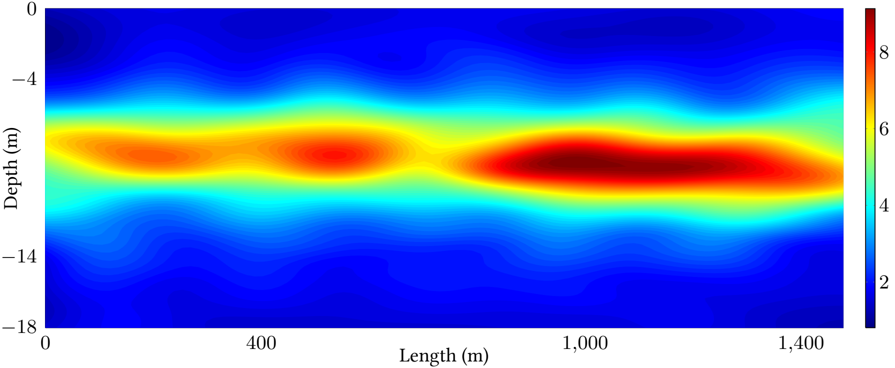
www.limnobotics.ch

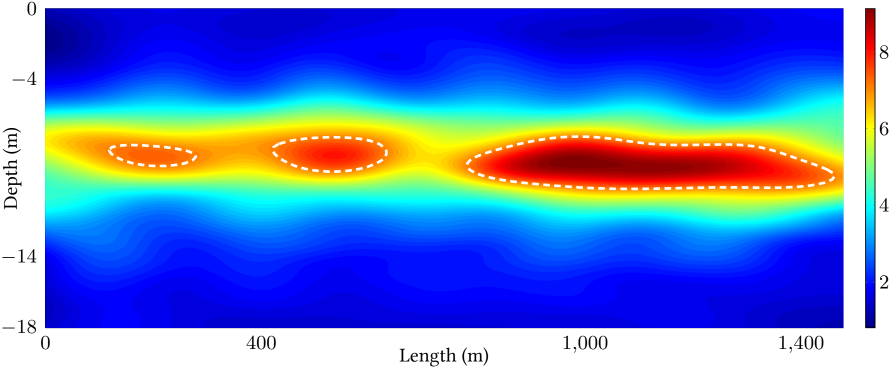


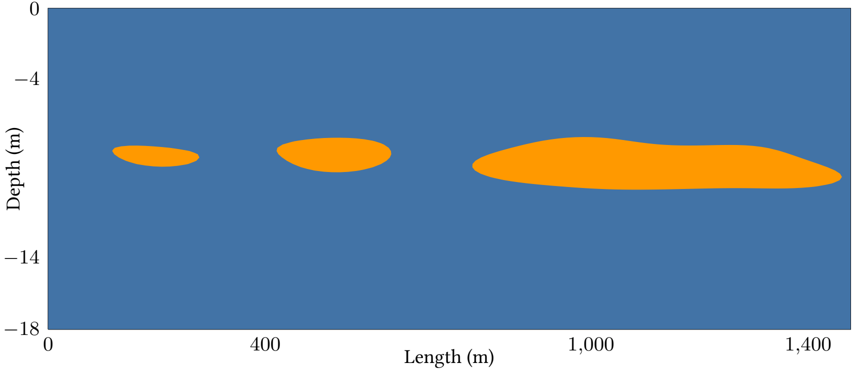
www.limnobotics.ch



0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2



0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

Pose as a sequential decision making problem (*pool-based active learning*):

- No measurements available in advance, just a set (pool) of possible sampling locations (D)

Pose as a sequential decision making problem (*pool-based active learning*):

- No measurements available in advance, just a set (pool) of possible sampling locations (D)

At each iteration $t \geq 1$:

- Decide where to measure next ($\boldsymbol{x}_t \in D$)

Pose as a sequential decision making problem (*pool-based active learning*):

- No measurements available in advance, just a set (pool) of possible sampling locations (D)

At each iteration $t \geq 1$:

- Decide where to measure next ($\boldsymbol{x}_t \in D$)
- Obtain noisy observation ($y_t = f(\boldsymbol{x}_t) + n_t$)

Pose as a sequential decision making problem (*pool-based active learning*):

- No measurements available in advance, just a set (pool) of possible sampling locations (D)

At each iteration $t \geq 1$:

- Decide where to measure next ($\boldsymbol{x}_t \in D$)
- Obtain noisy observation ($y_t = f(\boldsymbol{x}_t) + n_t$)
- Update our estimate of f

1. How do we **estimate** the function?

1. How do we **estimate** the function?
2. How do we **classify**?

1. How do we **estimate** the function?

2. How do we **classify**?

3. Each measurement is expensive (time, battery power).
   How do we **select** "informative" measurements?

1. How do we **estimate** the function?

2. How do we **classify**?

3. Each measurement is expensive (time, battery power).
   How do we **select** “informative” measurements?

Gaussian processes to the rescue!

Gaussian processes

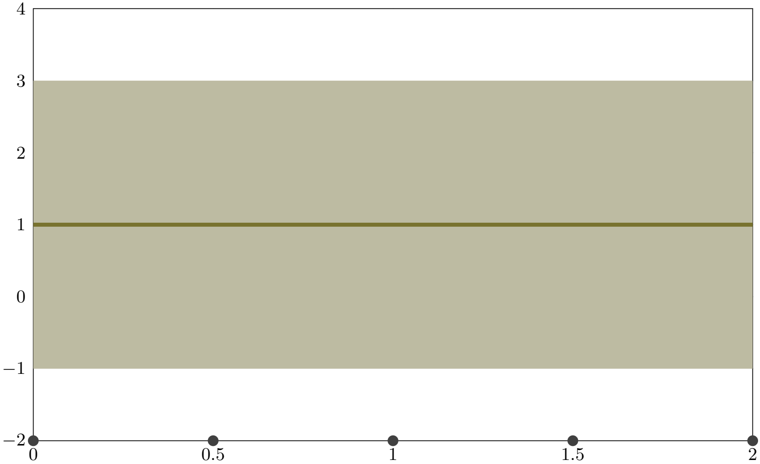
- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$

Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$

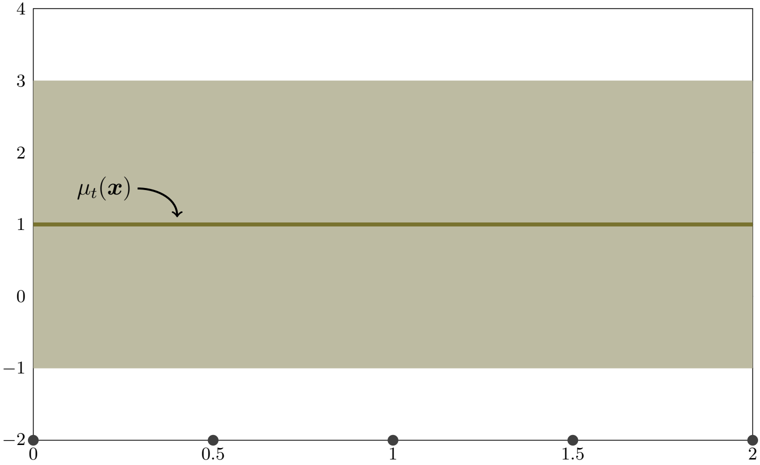
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$



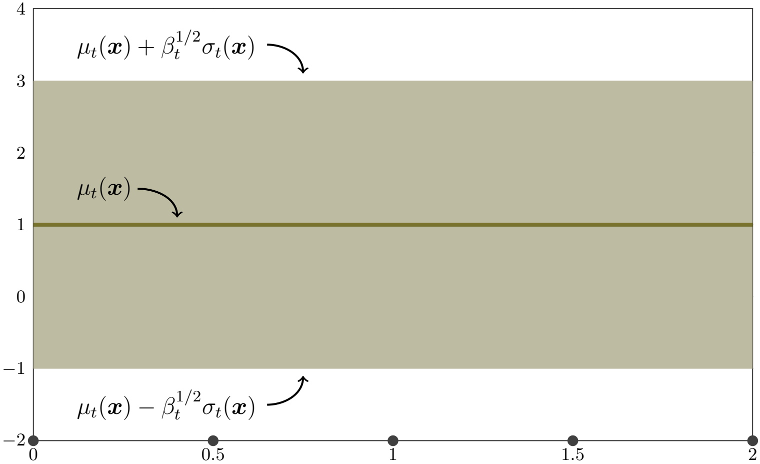
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$



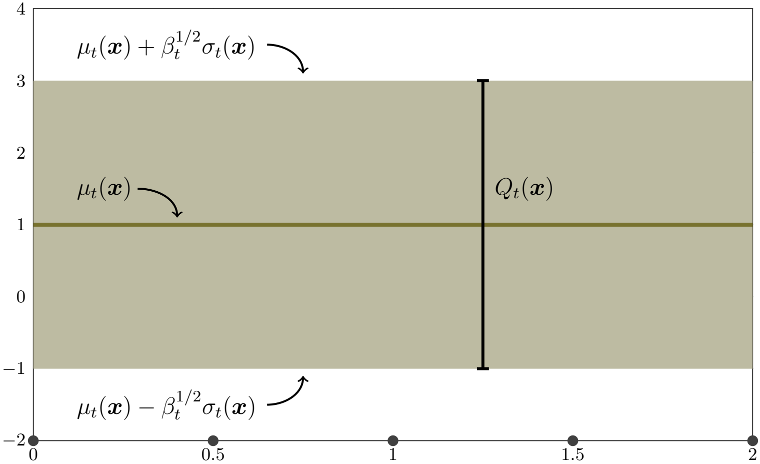
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$



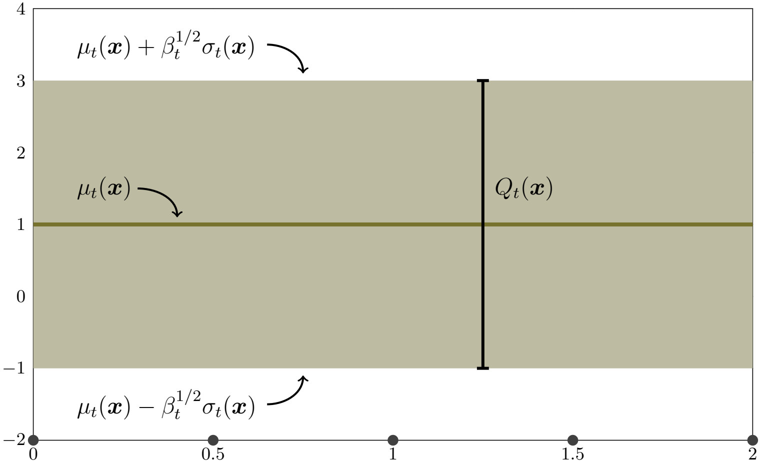
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$



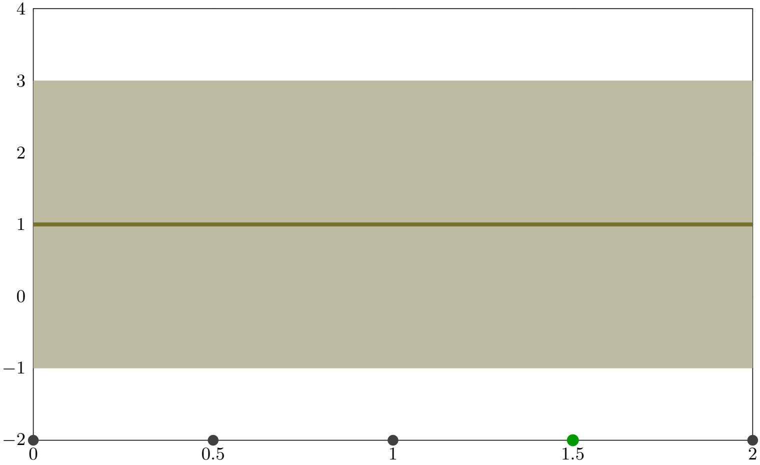
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



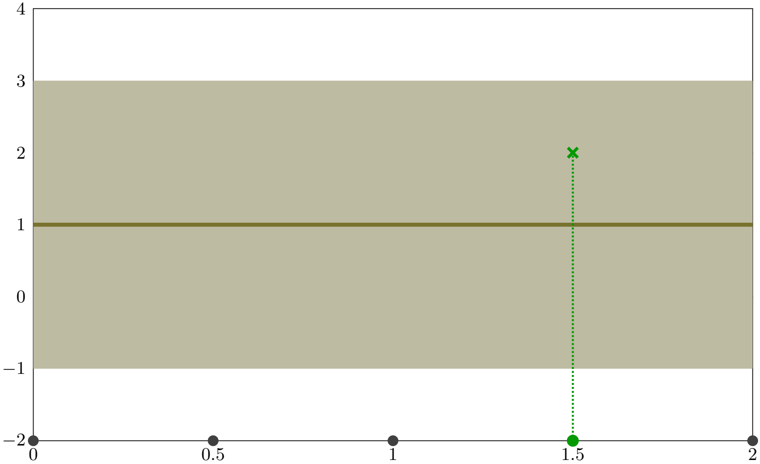
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



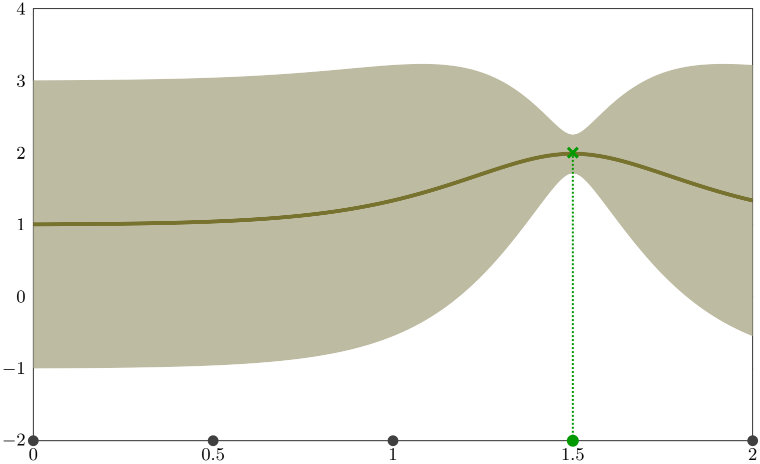
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



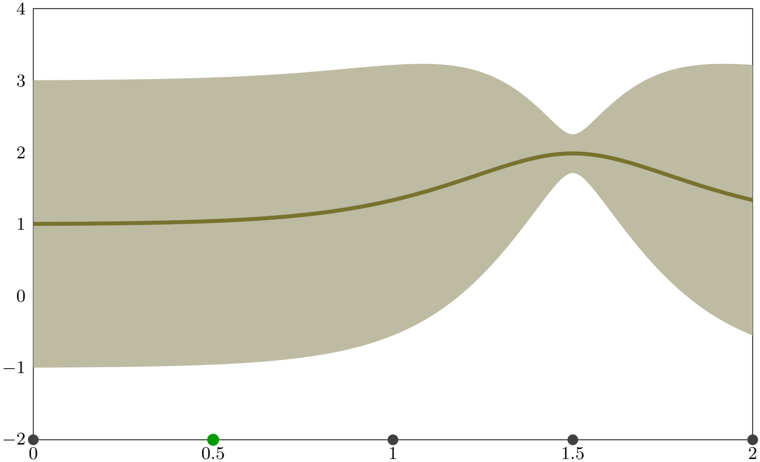
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



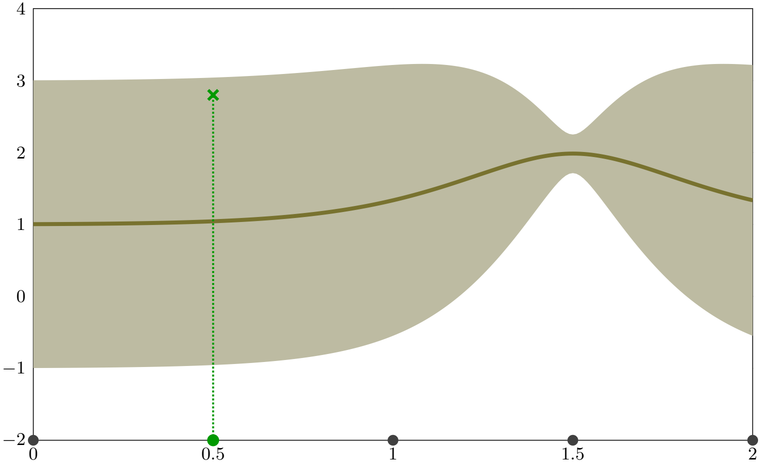
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



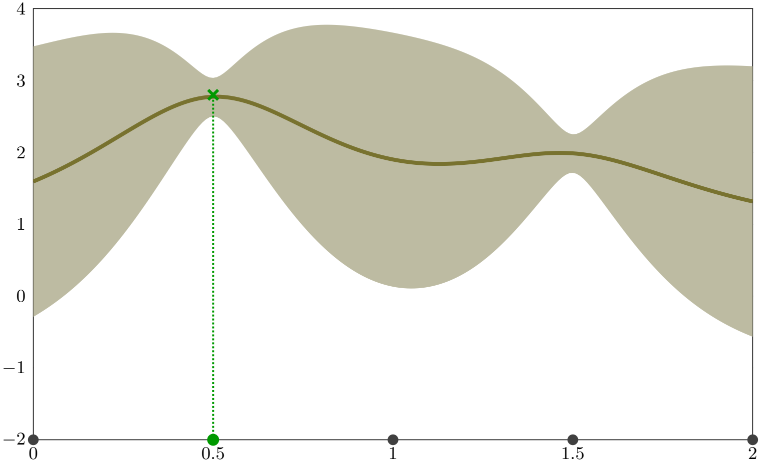
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



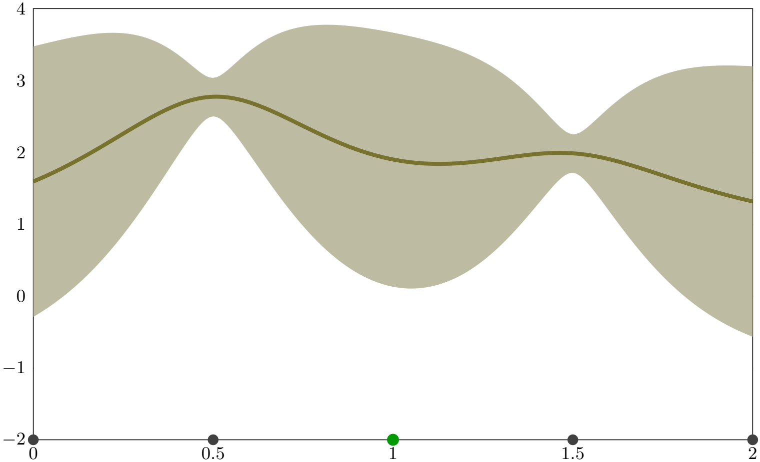
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



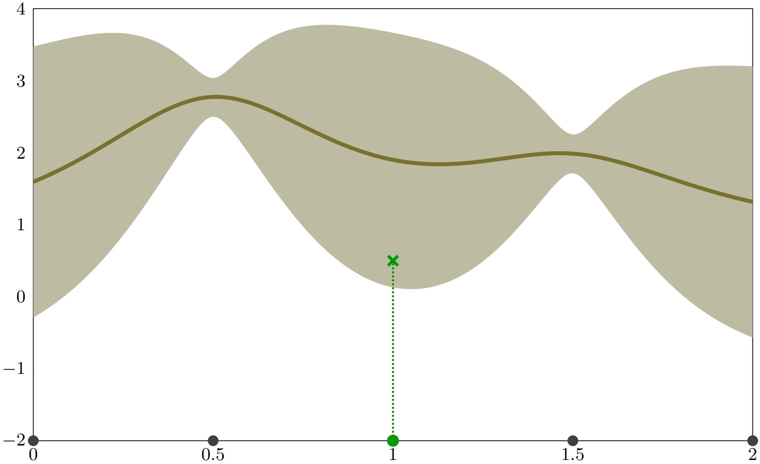
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



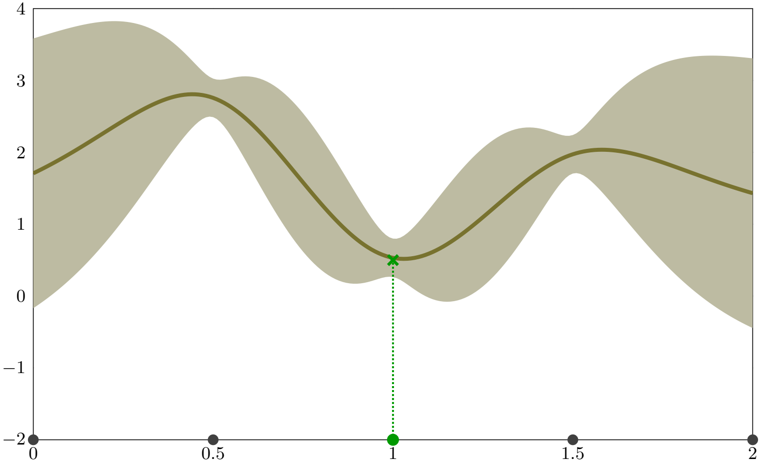
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



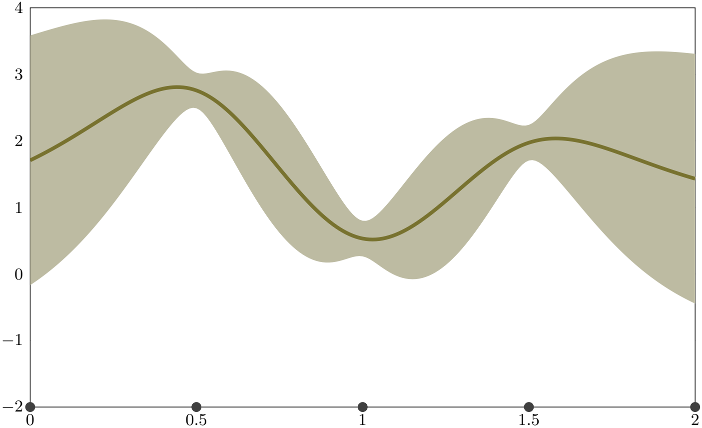
Gaussian processes

- **Nonpaparametric**: impose "smoothness" assumptions via kernel $k(\boldsymbol{x}, \boldsymbol{x}')$
- Mean **and variance** estimates: construct confidence intervals $Q_t(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates

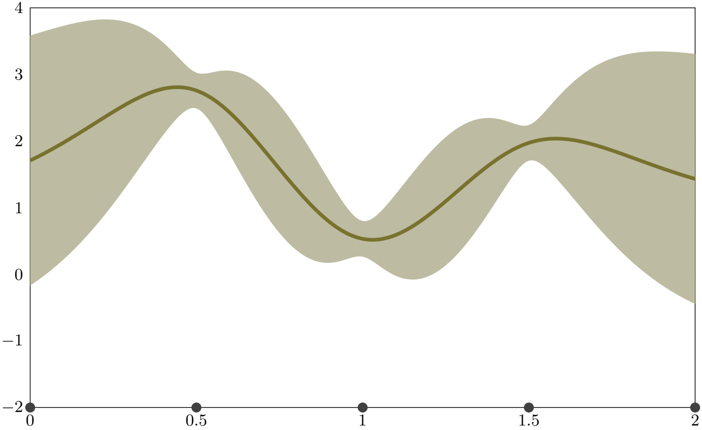


1. How do we **estimate** the function?

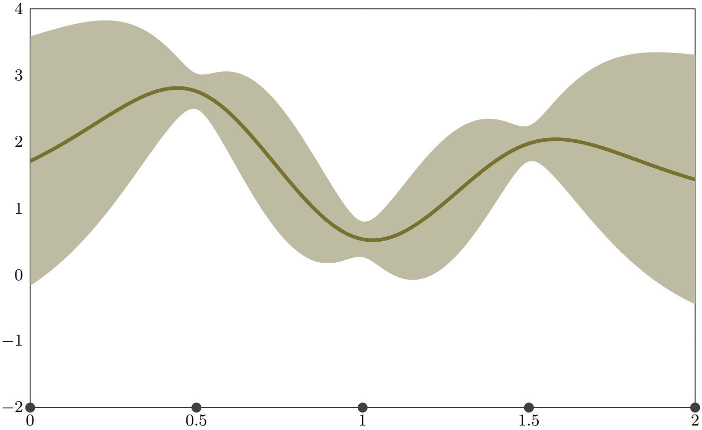
1. How do we **estimate** the function?



1. How do we **estimate** the function? ✓

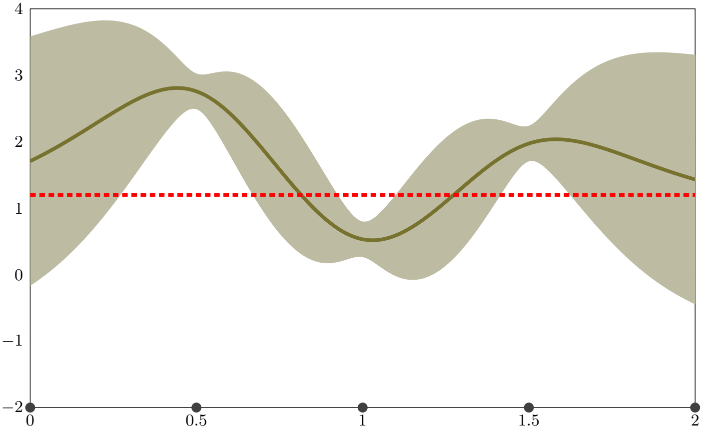


1. How do we **estimate** the function? ✓
2. How do we **classify**?

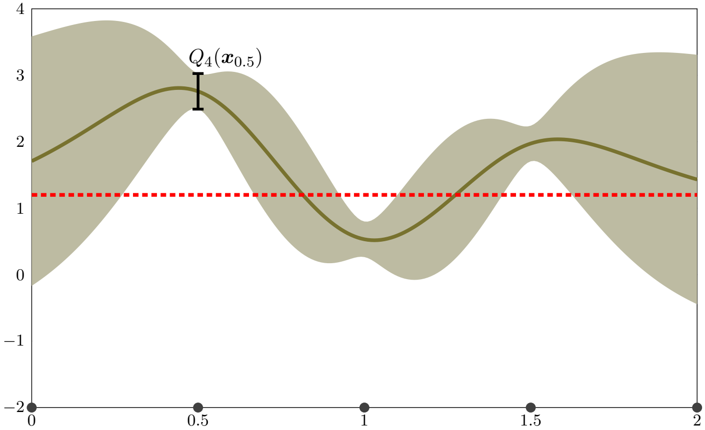


1. How do we **estimate** the function? ✓

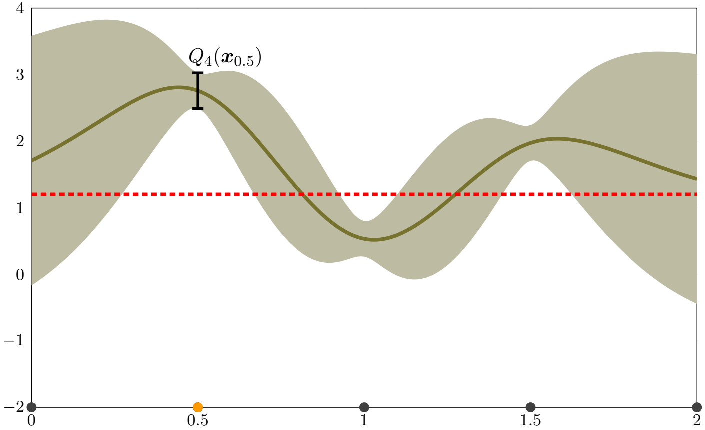
2. How do we **classify**?



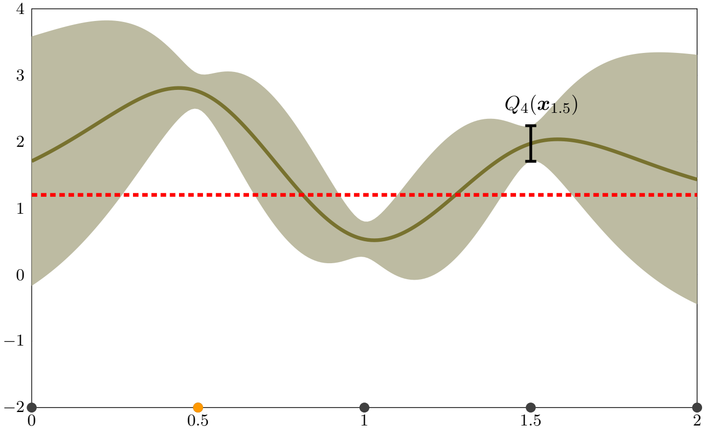
1. How do we **estimate** the function? ✓
2. How do we **classify**?



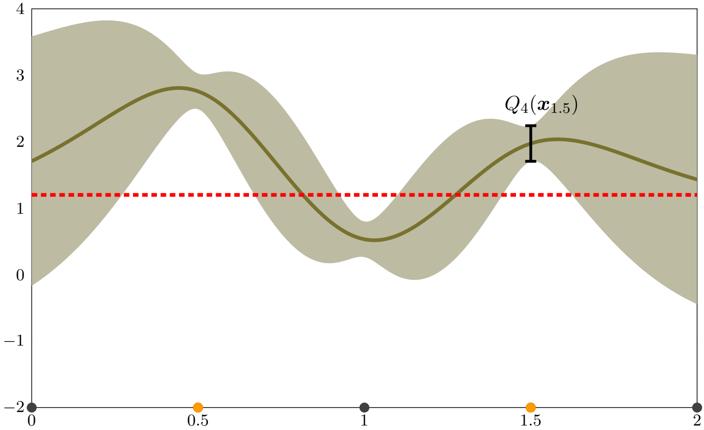
1. How do we **estimate** the function? ✓
2. How do we **classify**?



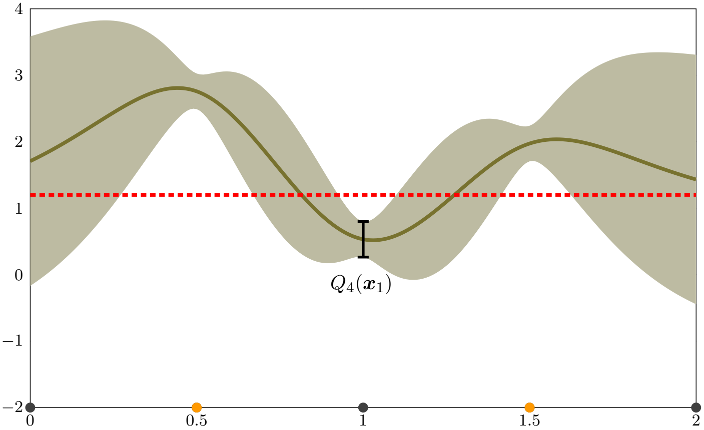
1. How do we **estimate** the function? ✓
2. How do we **classify**?



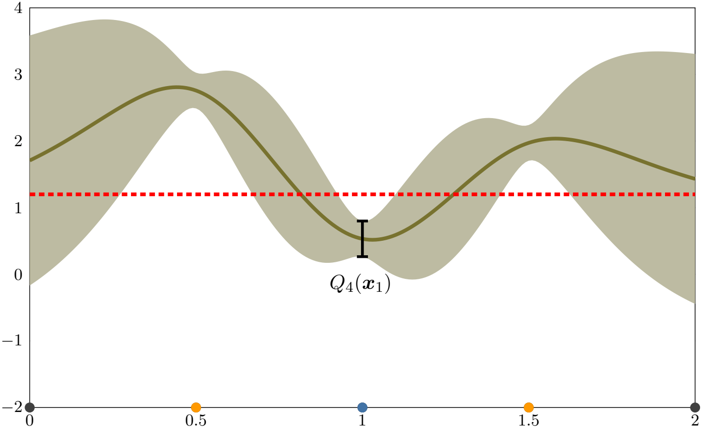
1. How do we **estimate** the function? ✓
2. How do we **classify**?



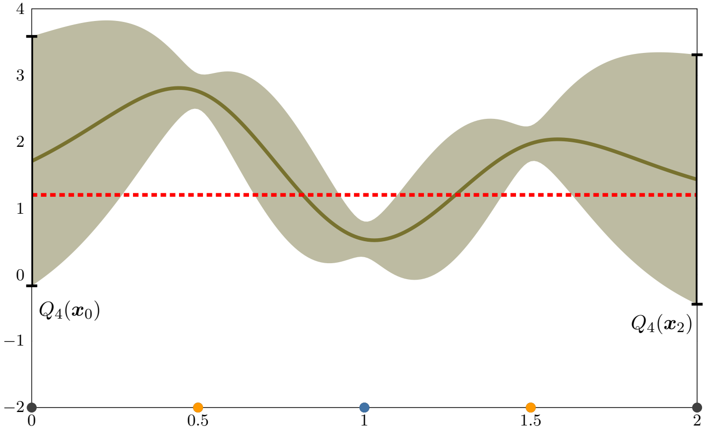
1. How do we **estimate** the function? ✓
2. How do we **classify**?



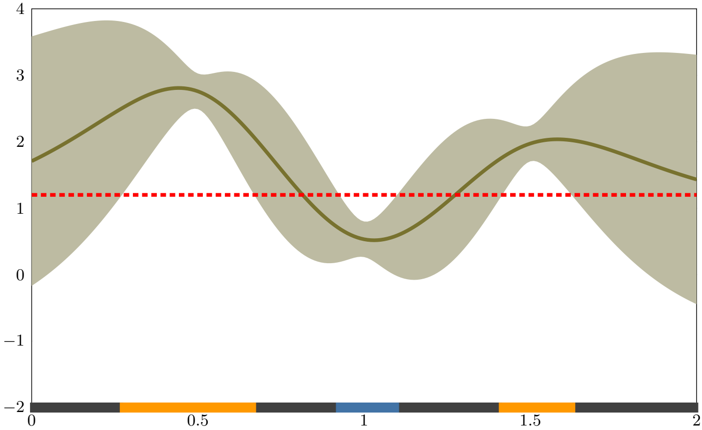
1. How do we **estimate** the function? ✓
2. How do we **classify**?



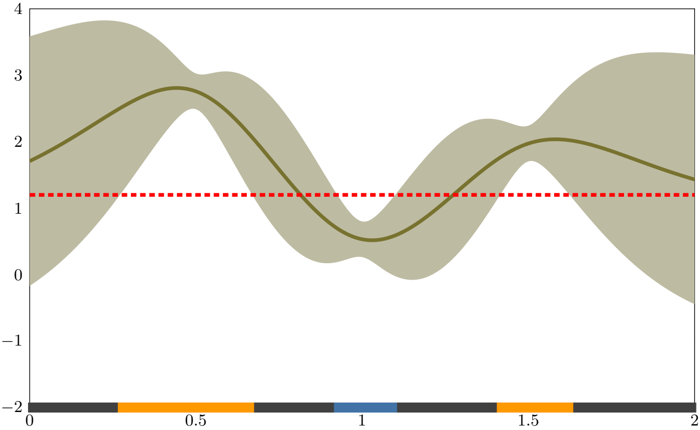
1. How do we **estimate** the function? ✓
2. How do we **classify**?



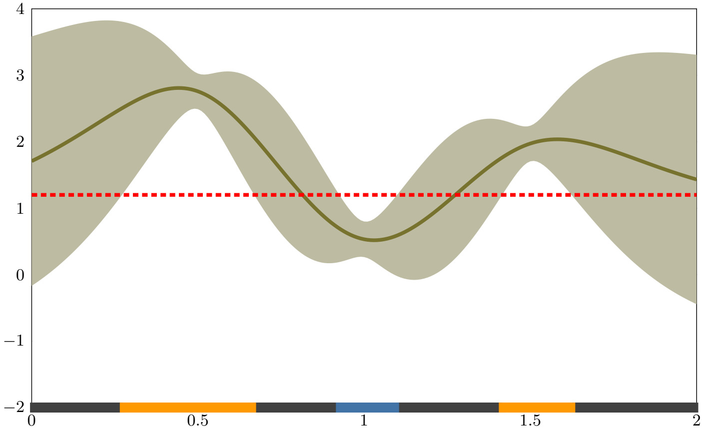
1. How do we **estimate** the function? ✓
2. How do we **classify**?



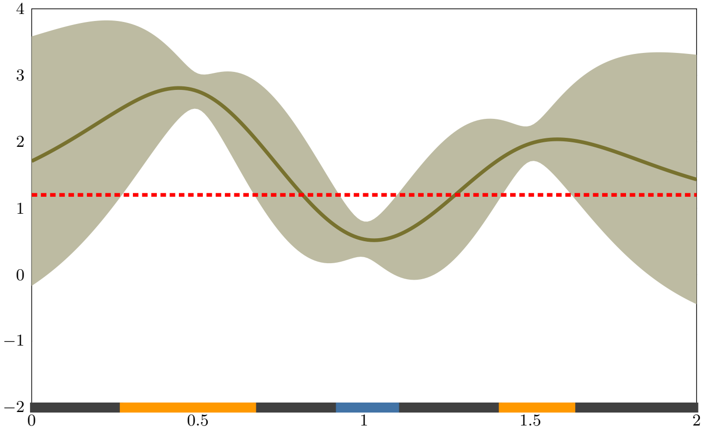
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓



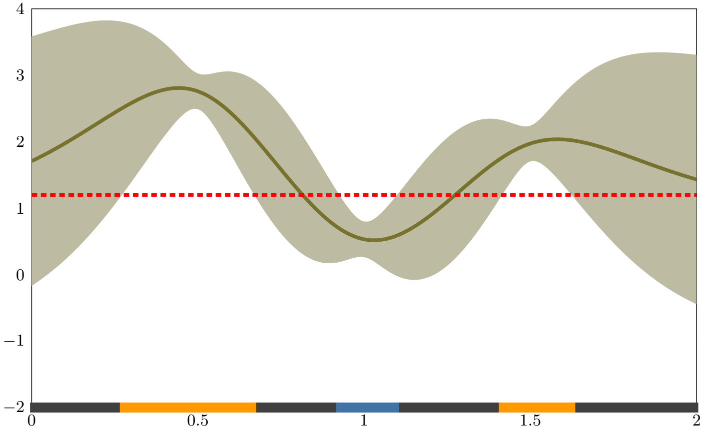
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?



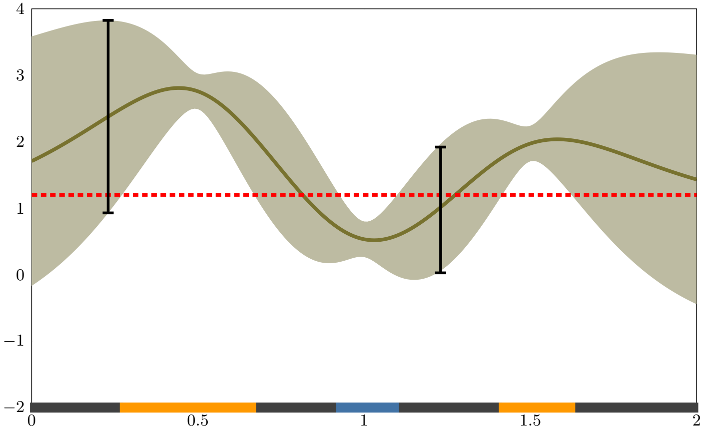
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** "informative" measurements?
   - Pick among the yet unclassified...



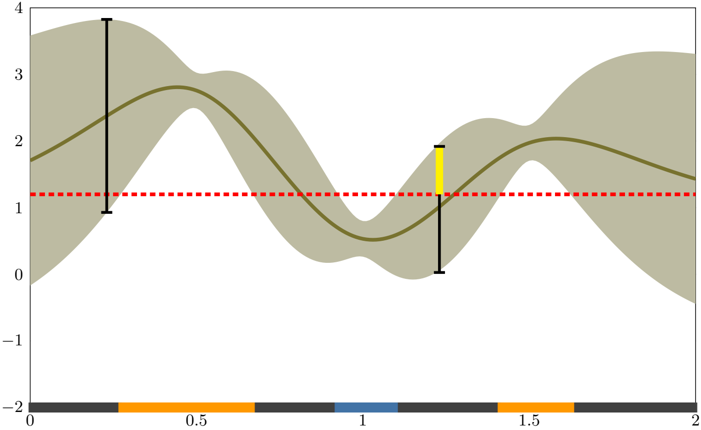
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
   - Pick among the yet unclassified...
   - ...the most “ambiguous” point



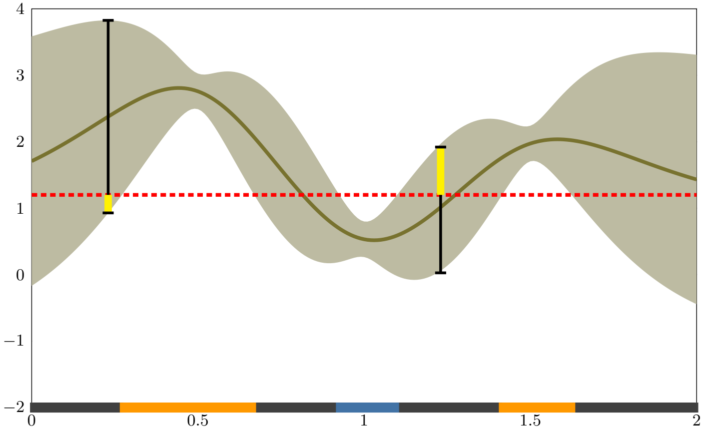
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** "informative" measurements?
   - Pick among the yet unclassified...
   - ...the most "ambiguous" point



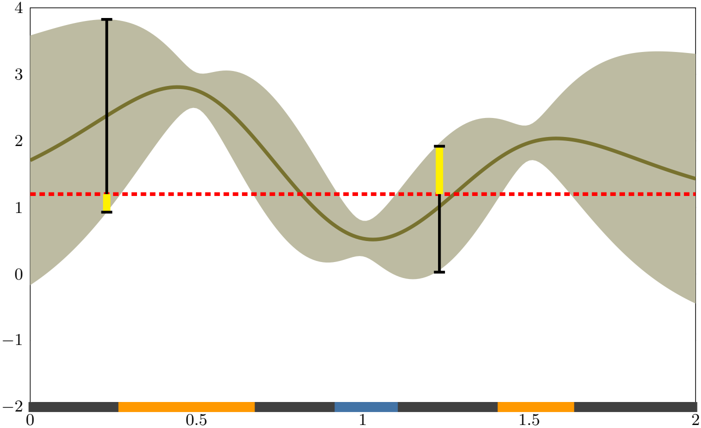
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** "informative" measurements?
   - Pick among the yet unclassified...
   - ...the most "ambiguous" point



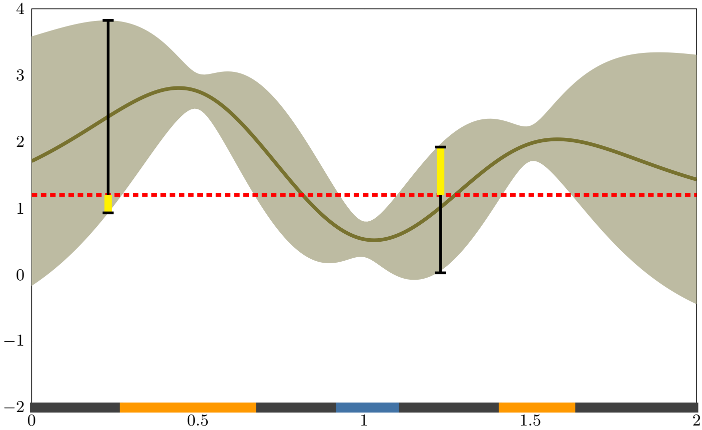
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
   - Pick among the yet unclassified...
   - ...the most “ambiguous” point



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
   - Pick among the yet unclassified...
   - ...the most “ambiguous” point (max. variance or random also work)



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements? ✓
   - Pick among the yet unclassified...
   - ...the most “ambiguous” point (max. variance or random also work)



The Level Set Estimation (LSE) algorithm

Input: sample set D, GP prior (μ_0, k, σ_0), thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

Input: sample set D, GP prior $(\mu_0,\ k,\ \sigma_0)$, thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$

The Level Set Estimation (LSE) algorithm

$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

The Level Set Estimation (LSE) algorithm

Input: sample set D, GP prior $(\mu_0,\ k,\ \sigma_0)$,
thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \varnothing$ **do**
  $H_t \leftarrow H_{t-1},\ L_t \leftarrow L_{t-1},\ U_t \leftarrow U_{t-1}$

  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

$\leftarrow$ loop until all points have been classified

The Level Set Estimation (LSE) algorithm

Input: sample set D, GP prior (μ_0, k, σ_0), thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \varnothing$ **do** ← loop until all points have been classified
  $H_t \leftarrow H_{t-1},\ L_t \leftarrow L_{t-1},\ U_t \leftarrow U_{t-1}$
  for all $\boldsymbol{x} \in U_{t-1}$ **do**
    $C_t(\boldsymbol{x}) \leftarrow C_{t-1}(\boldsymbol{x}) \cap Q_t(\boldsymbol{x})$
    if $\min(C_t(\boldsymbol{x})) + \epsilon > h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$ ← classify
      $H_t \leftarrow H_t \cup \{\boldsymbol{x}\}$
    else if $\max(C_t(\boldsymbol{x})) - \epsilon \leq h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $L_t \leftarrow L_t \cup \{\boldsymbol{x}\}$
    end if
  end for
  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

The Level Set Estimation (LSE) algorithm

Input: sample set D, GP prior $(\mu_0,\ k,\ \sigma_0)$, thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \varnothing$ **do** ← loop until all points have been classified
  $H_t \leftarrow H_{t-1},\ L_t \leftarrow L_{t-1},\ U_t \leftarrow U_{t-1}$
  for all $\boldsymbol{x} \in U_{t-1}$ **do**
    $C_t(\boldsymbol{x}) \leftarrow C_{t-1}(\boldsymbol{x}) \cap Q_t(\boldsymbol{x})$
    if $\min(C_t(\boldsymbol{x})) + \epsilon > h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $H_t \leftarrow H_t \cup \{\boldsymbol{x}\}$ ← classify
    else if $\max(C_t(\boldsymbol{x})) - \epsilon \leq h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $L_t \leftarrow L_t \cup \{\boldsymbol{x}\}$
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$ ← select max. ambiguity point
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$

  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

The Level Set Estimation (LSE) algorithm

Input: sample set D, GP prior (μ_0, k, σ_0), thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \varnothing$ **do** ← loop until all points have been classified
  $H_t \leftarrow H_{t-1},\ L_t \leftarrow L_{t-1},\ U_t \leftarrow U_{t-1}$
  for all $\boldsymbol{x} \in U_{t-1}$ **do**
    $C_t(\boldsymbol{x}) \leftarrow C_{t-1}(\boldsymbol{x}) \cap Q_t(\boldsymbol{x})$
    if $\min(C_t(\boldsymbol{x})) + \epsilon > h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $H_t \leftarrow H_t \cup \{\boldsymbol{x}\}$ ← classify
    else if $\max(C_t(\boldsymbol{x})) - \epsilon \leq h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $L_t \leftarrow L_t \cup \{\boldsymbol{x}\}$
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$ ← select max. ambiguity point
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  Compute $\mu_t(\boldsymbol{x})$ and $\sigma_t(\boldsymbol{x})$, $\forall \boldsymbol{x} \in U_t$ ← update GP estimate
  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

Input: sample set D, GP prior (μ_0, k, σ_0), thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

$H_0 \leftarrow \varnothing,\ L_0 \leftarrow \varnothing,\ U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \varnothing$ **do**
  $H_t \leftarrow H_{t-1},\ L_t \leftarrow L_{t-1},\ U_t \leftarrow U_{t-1}$
  for all $\boldsymbol{x} \in U_{t-1}$ **do**
    $C_t(\boldsymbol{x}) \leftarrow C_{t-1}(\boldsymbol{x}) \cap Q_t(\boldsymbol{x})$
    if $\min(C_t(\boldsymbol{x})) + \epsilon > h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $H_t \leftarrow H_t \cup \{\boldsymbol{x}\}$
    else if $\max(C_t(\boldsymbol{x})) - \epsilon \leq h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $L_t \leftarrow L_t \cup \{\boldsymbol{x}\}$
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  Compute $\mu_t(\boldsymbol{x})$ and $\sigma_t(\boldsymbol{x})$, $\forall \boldsymbol{x} \in U_t$
  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

The Level Set Estimation (LSE) algorithm

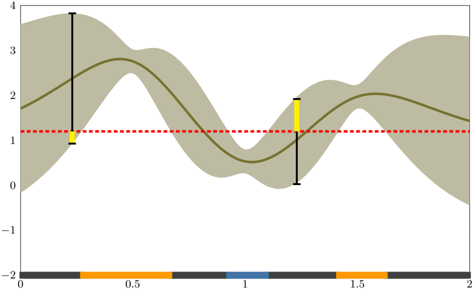
- Monotonicity of
  1. confidence intervals
  2. classification

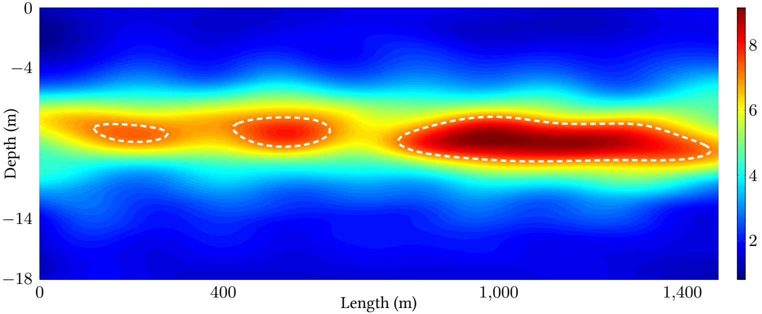
Input: sample set D, GP prior (μ_0, k, σ_0), thr. value h, accuracy parameter ϵ
Output: predicted sets $\hat{H}$, $\hat{L}$

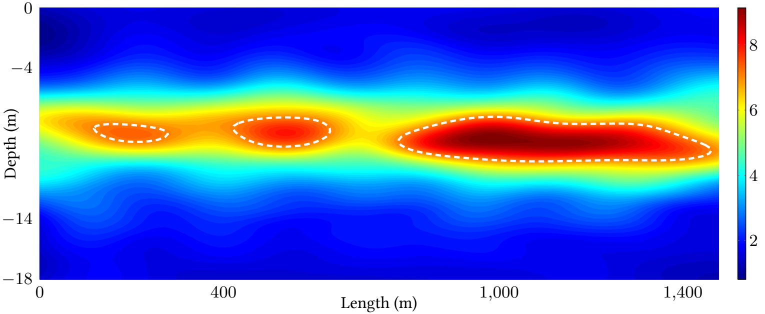
$H_0 \leftarrow \emptyset$, $L_0 \leftarrow \emptyset$, $U_0 \leftarrow D$
$C_0(\boldsymbol{x}) \leftarrow \mathbb{R}$, for all $\boldsymbol{x} \in D$
$t \leftarrow 1$
while $U_{t-1} \neq \emptyset$ **do**
  $H_t \leftarrow H_{t-1}$, $L_t \leftarrow L_{t-1}$, $U_t \leftarrow U_{t-1}$
  for all $\boldsymbol{x} \in U_{t-1}$ **do**
    $C_t(\boldsymbol{x}) \leftarrow C_{t-1}(\boldsymbol{x}) \cap Q_t(\boldsymbol{x})$
    if $\min(C_t(\boldsymbol{x})) + \epsilon > h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $H_t \leftarrow H_t \cup \{\boldsymbol{x}\}$
    else if $\max(C_t(\boldsymbol{x})) - \epsilon \leq h$ **then**
      $U_t \leftarrow U_t \setminus \{\boldsymbol{x}\}$
      $L_t \leftarrow L_t \cup \{\boldsymbol{x}\}$
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  Compute $\mu_t(\boldsymbol{x})$ and $\sigma_t(\boldsymbol{x})$, $\forall \boldsymbol{x} \in U_t$
  $t \leftarrow t + 1$
end while
$\hat{H} \leftarrow H_{t-1}$
$\hat{L} \leftarrow L_{t-1}$

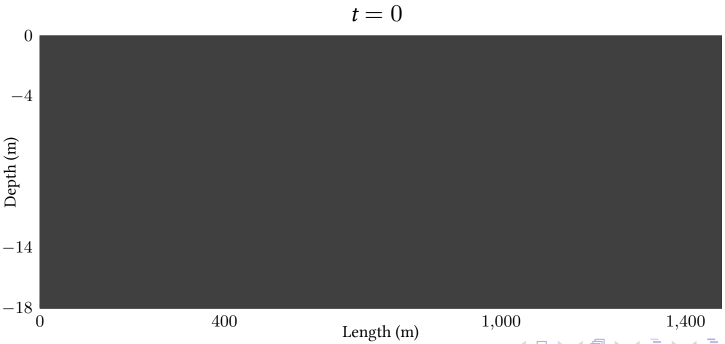
The Level Set Estimation (LSE) algorithm

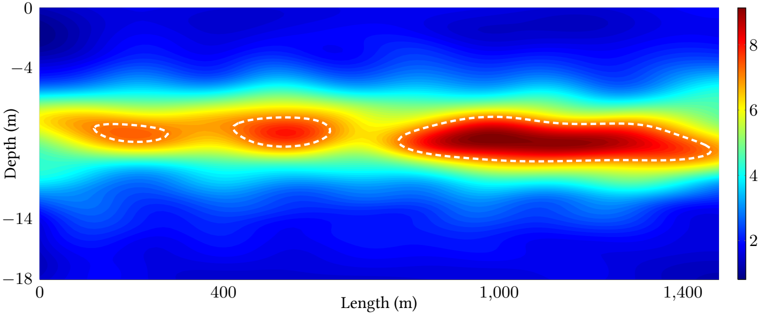
- Monotonicity of
  1. confidence intervals
  2. classification
- Relaxed classification rules by an accuracy parameter ϵ (trades off sampling cost for accuracy)




0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

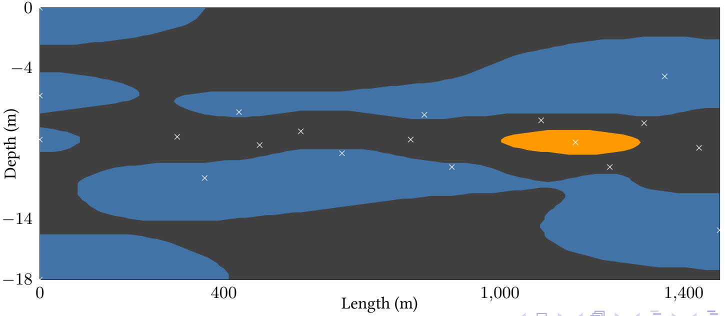

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


$t = 0$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

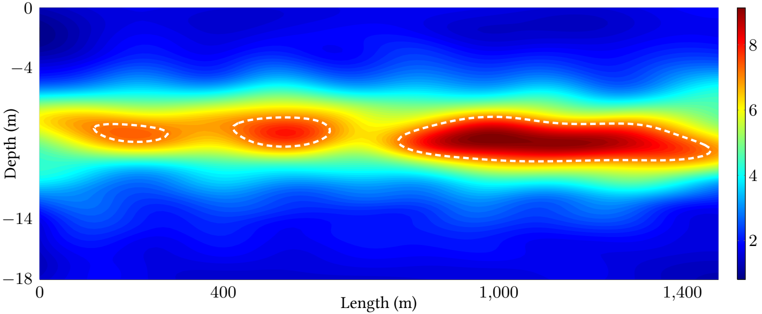


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

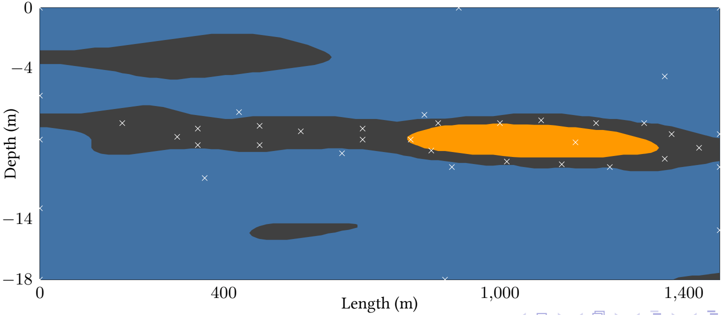
$t = 20$

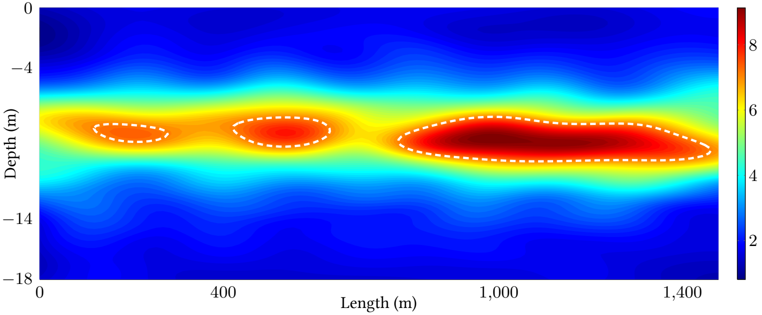


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

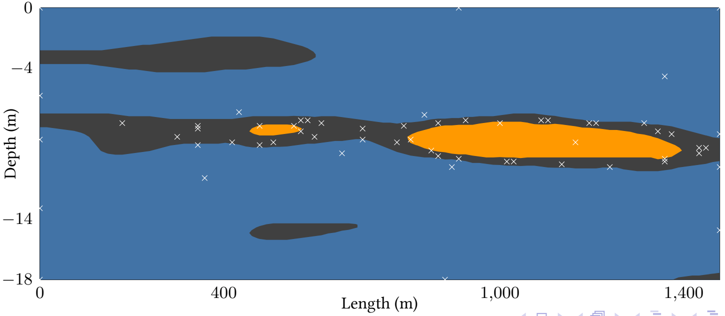

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

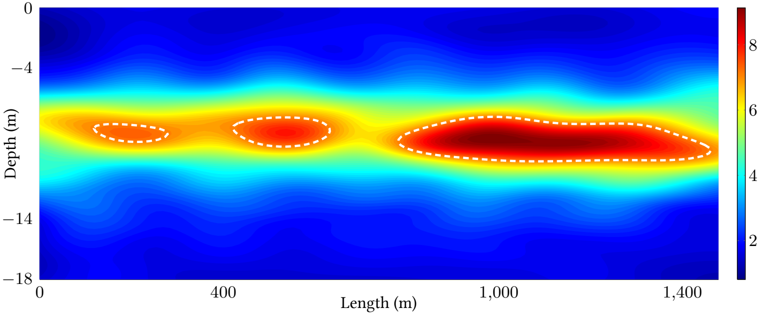
$t = 40$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

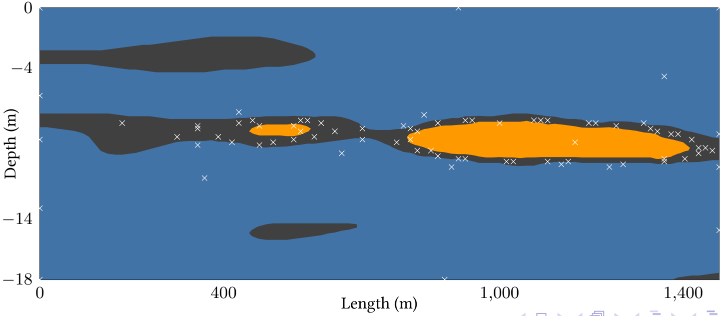
$t = 60$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

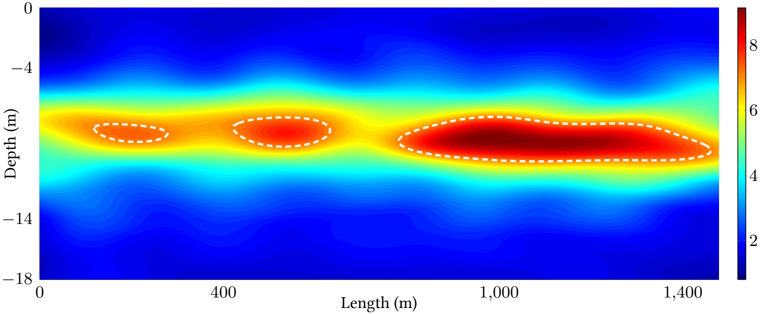


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

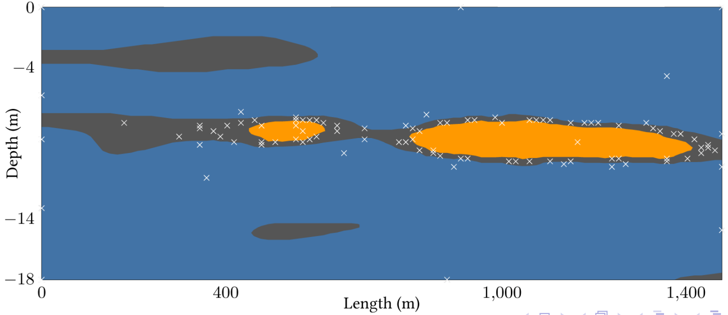
$t = 80$

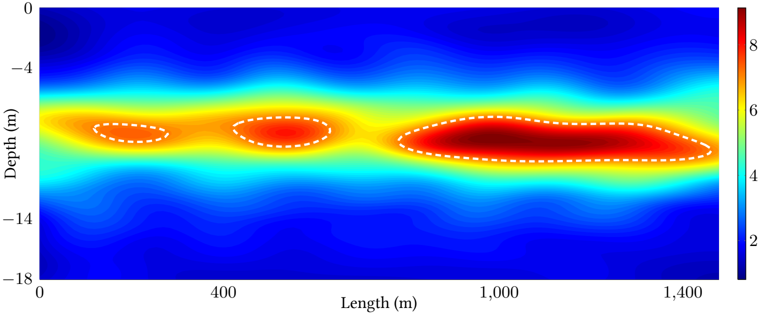


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

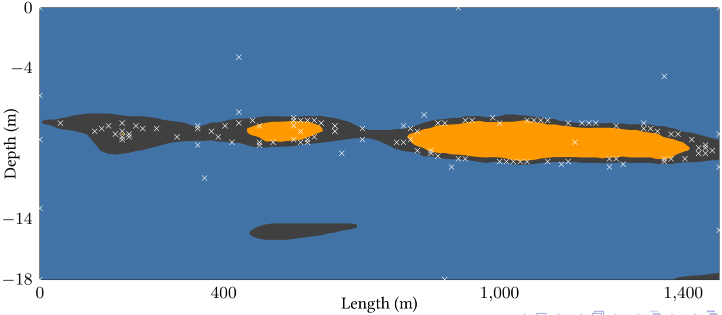

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

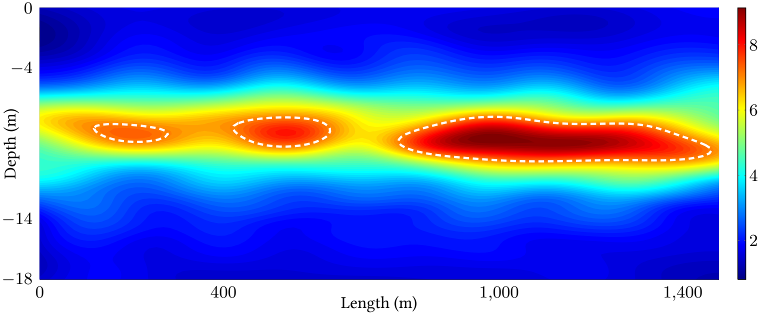
$t = 100$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

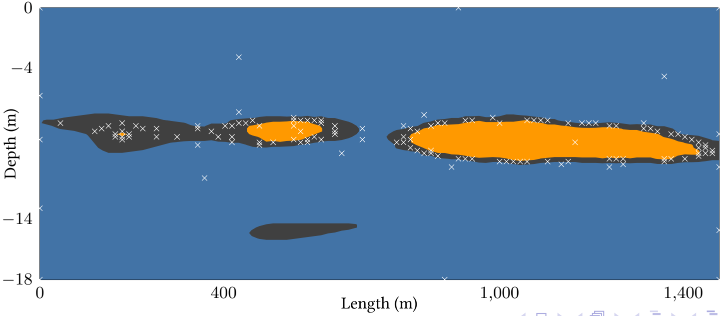

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

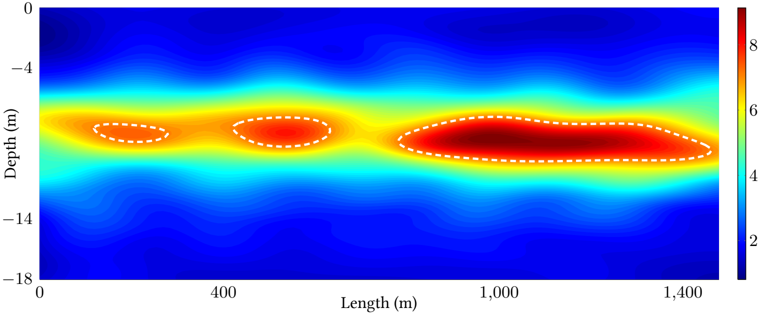
$t = 120$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

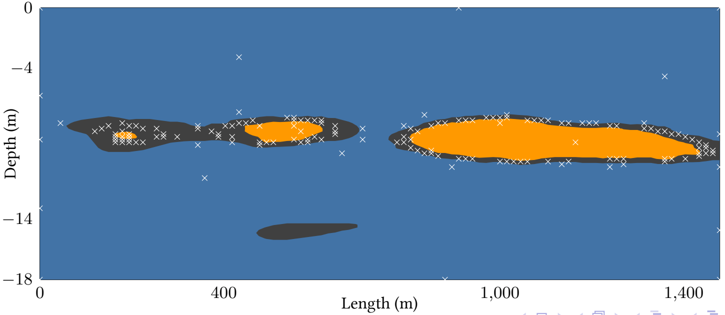
$t = 140$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

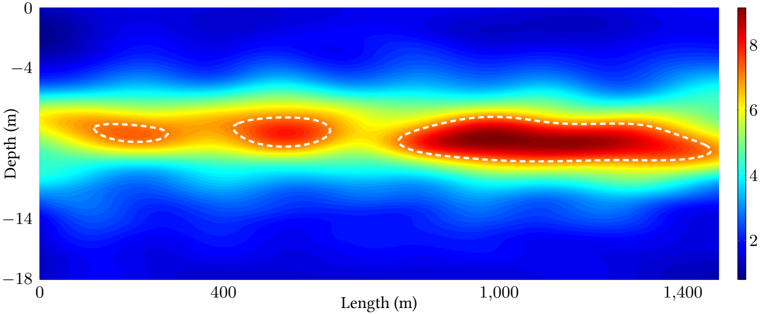


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

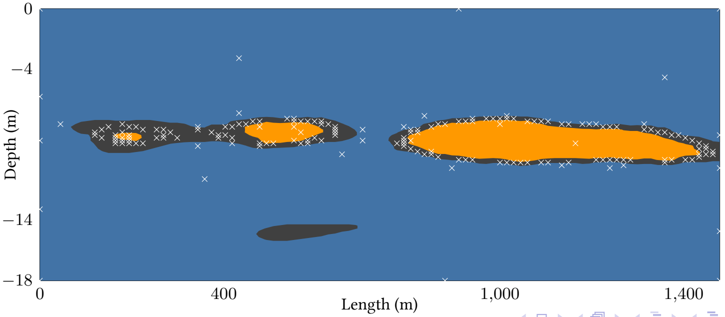
$t = 160$

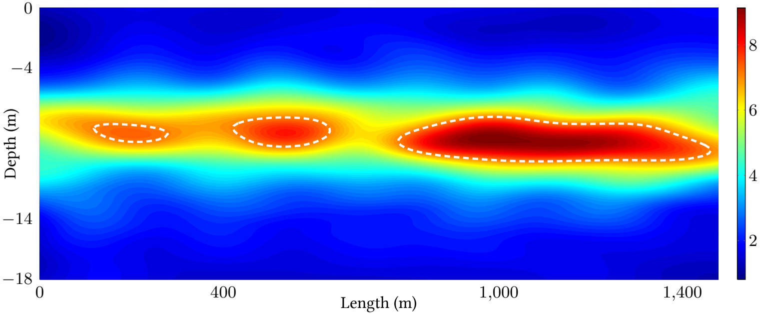


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

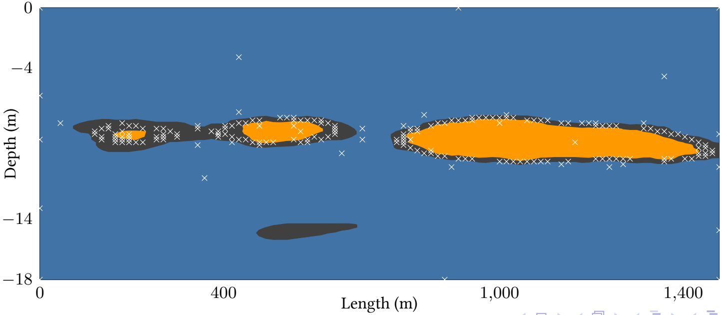

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

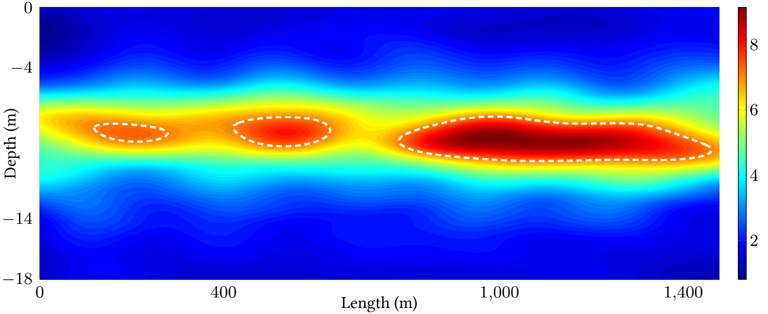
$t = 180$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

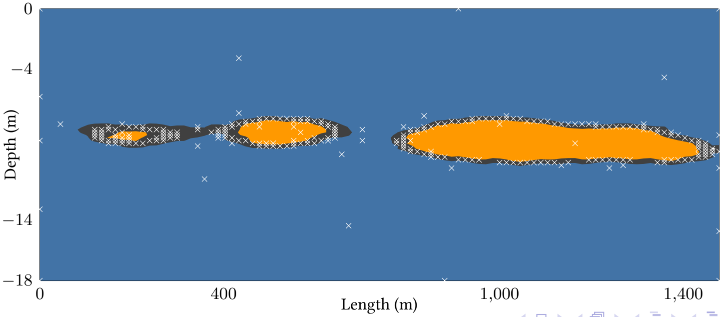
$t = 200$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

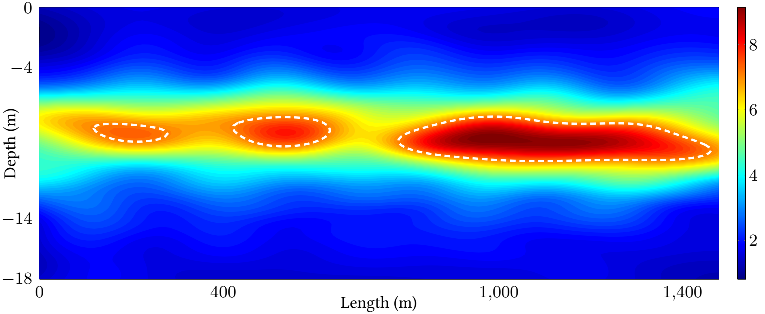


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

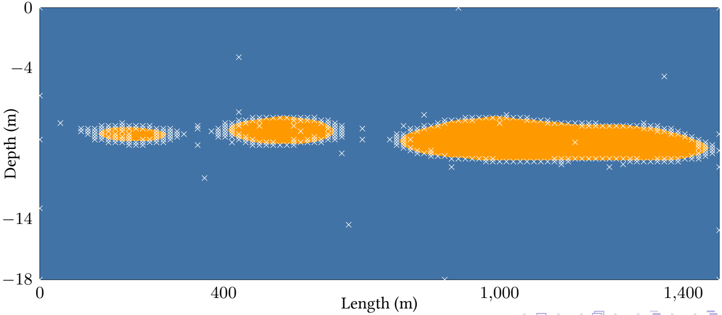
$t = 260$



0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

$t = 354$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

Theorem (Convergence of LSE)

For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x}\in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1-\delta.$$

Theorem (Simplified)

If we choose β_t large enough, then:

Theorem (Simplified)

If we choose β_t large enough, then:

- *LSE terminates after a number of iterations T*

Theorem (Simplified)

If we choose β_t *large enough, then:*

- *LSE terminates after a number of iterations* T
    1. *smoother kernel* $\Rightarrow T \downarrow$

Theorem (Simplified)

If we choose β_t large enough, then:

- *LSE terminates after a number of iterations T*
  1. *smoother kernel* $\Rightarrow T\downarrow$
  2. $\sigma\uparrow \Rightarrow T\uparrow$

Theorem (Simplified)

If we choose β_t large enough, then:

- *LSE terminates after a number of iterations* T
  1. *smoother kernel* $\Rightarrow T\downarrow$
  2. $\sigma\uparrow \Rightarrow T\uparrow$
  3. $\epsilon\uparrow \Rightarrow T\downarrow$

Theorem (Simplified)

If we choose β_t large enough, then:

- *LSE terminates after a number of iterations T*
    1. *smoother kernel* $\Rightarrow T\downarrow$
    2. $\sigma\uparrow\Rightarrow T\uparrow$
    3. $\epsilon\uparrow\Rightarrow T\downarrow$
- *The solution returned is ϵ-accurate with high probability*

Theorem (Simplified)

If we choose β_t large enough, then:

- *LSE terminates after a number of iterations T*
  1. *smoother kernel* $\Rightarrow T\downarrow$
  2. $\sigma\uparrow \Rightarrow T\uparrow$
  3. $\epsilon\uparrow \Rightarrow T\downarrow$
- *The solution returned is ϵ-accurate with high probability*

Experiments

1. LSE

Experiments

1. LSE
2. Maximum variance sampling:

$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x}\in D} \sigma_{t-1}(\boldsymbol{x})$$

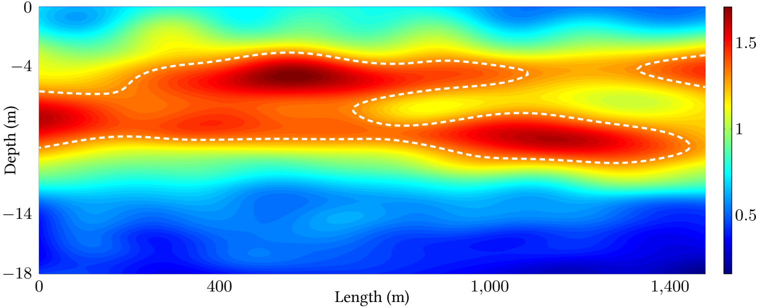
Experiments

1. LSE
2. Maximum variance sampling:

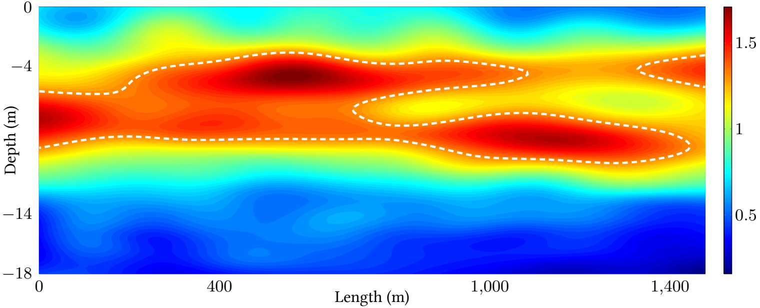
$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \sigma_{t-1}(\boldsymbol{x})$$

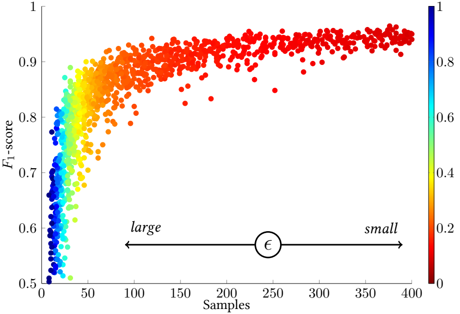
3. State of the art "straddle" heuristic (Bryan *et al.*, 2005):

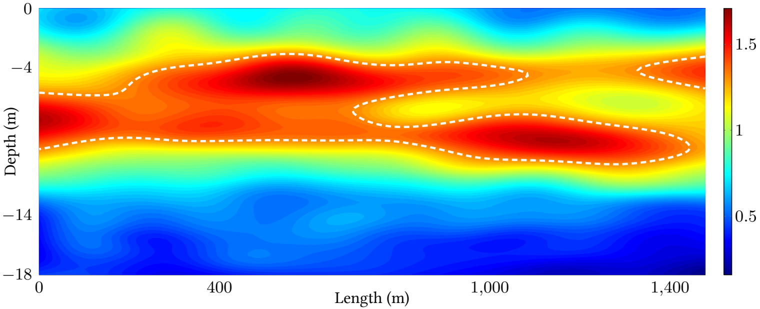
$$\boldsymbol{x}_t \approx \operatorname*{argmax}_{\boldsymbol{x} \in D} a_{t-1}(\boldsymbol{x}) \qquad \text{(for } \beta_t^{1/2} = 1.96\text{)}$$



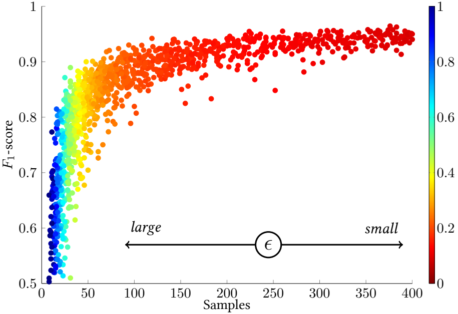
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
1.5
1
0.5


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
1.5
1
0.5

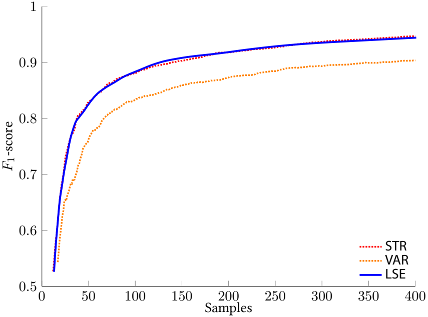

F_1-score
Samples
large
small
ϵ



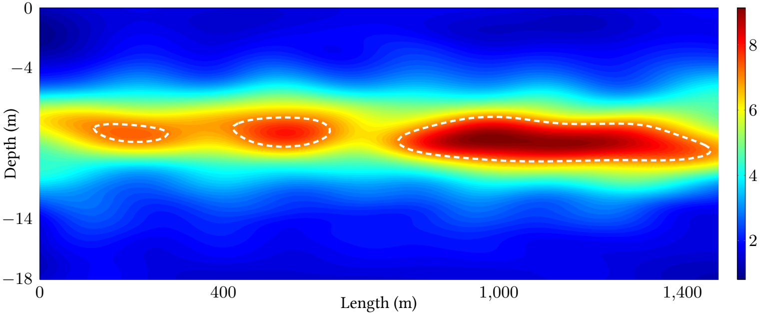
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
1.5
1
0.5

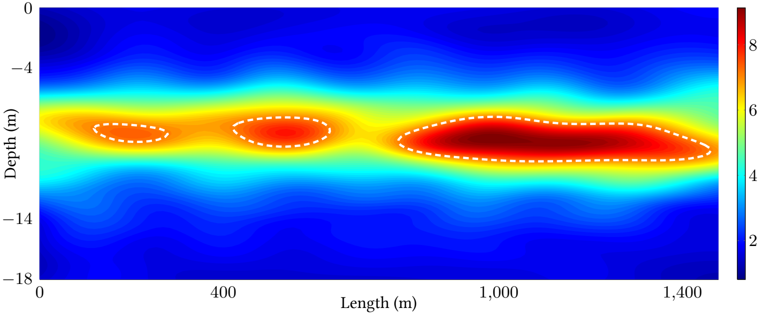


F_1-score
Samples
large
ϵ
small

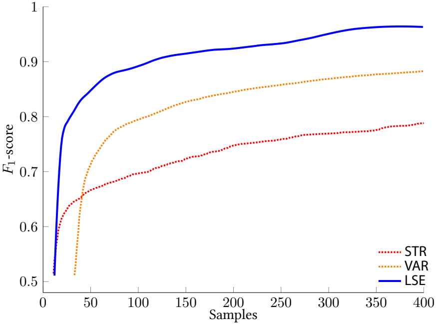


F_1-score
Samples
STR
VAR
LSE


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2



0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2



1
0.9
0.8
0.7
0.6
0.5
F_1-score
0
50
100
150
200
250
300
350
400
Samples
STR
VAR
LSE

- What if we don't know the threshold level h?

- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$

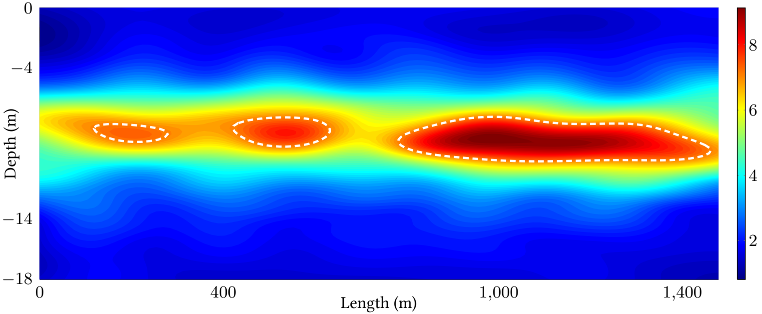
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)

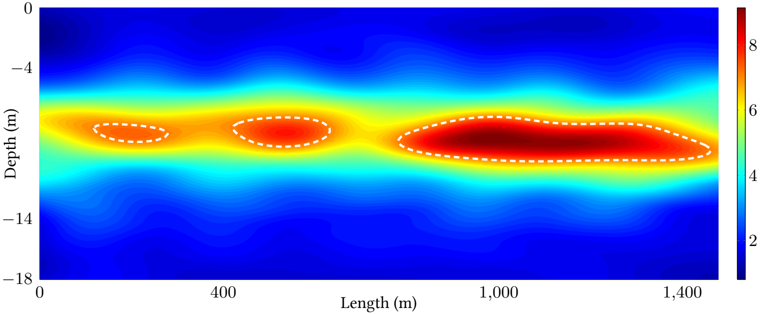
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting

- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- LSE_{imp} algorithm:

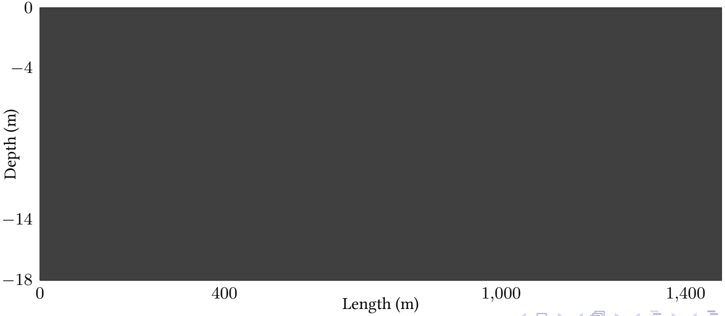
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- LSE_{imp} algorithm:
  - Need to accurately estimate the maximum → modified selection rule

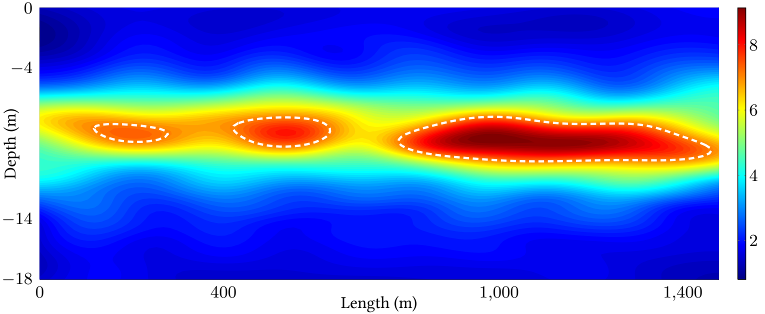
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- LSE_{imp} algorithm:
  - Need to accurately estimate the maximum $\rightarrow$ modified selection rule
  - h is now an estimated quantity $\rightarrow$ modified classification rules


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

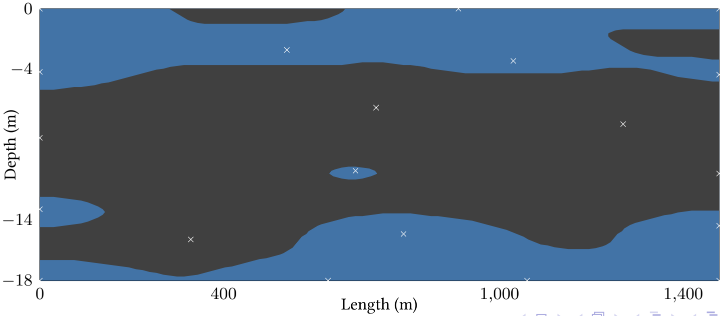
$t = 0$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

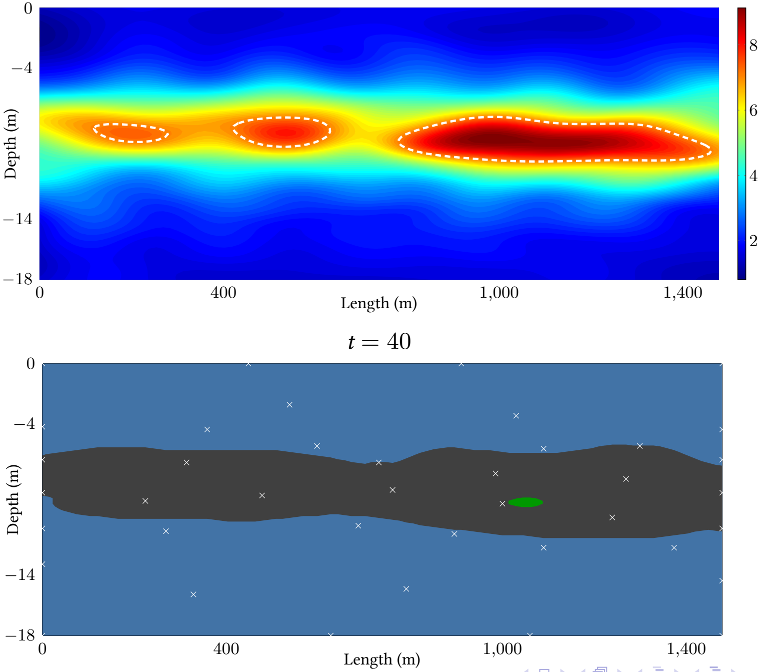


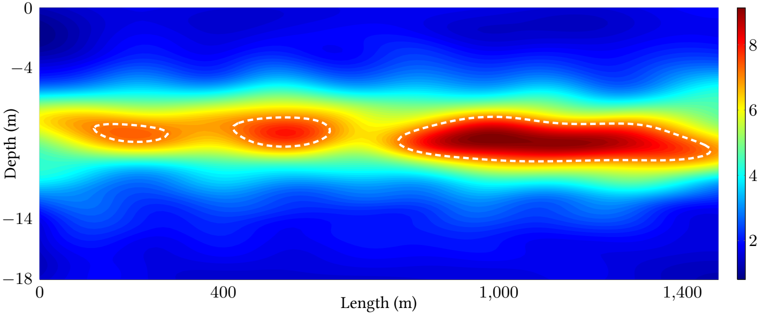
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

$t = 20$



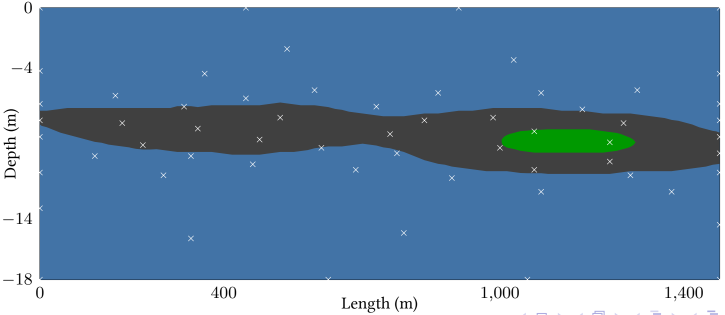
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2
t = 40
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

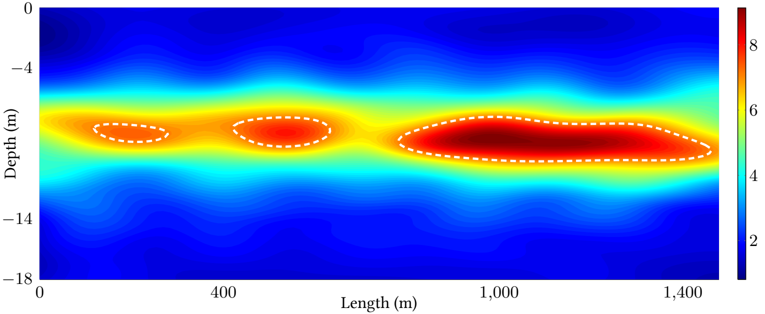


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

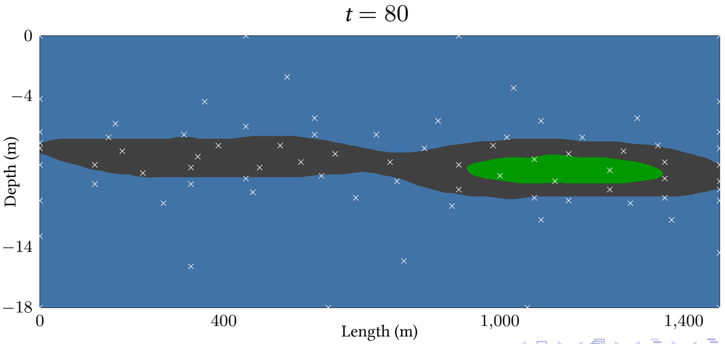
$t = 60$



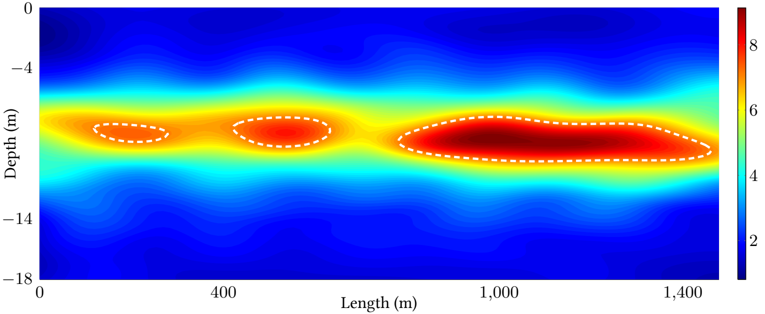
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

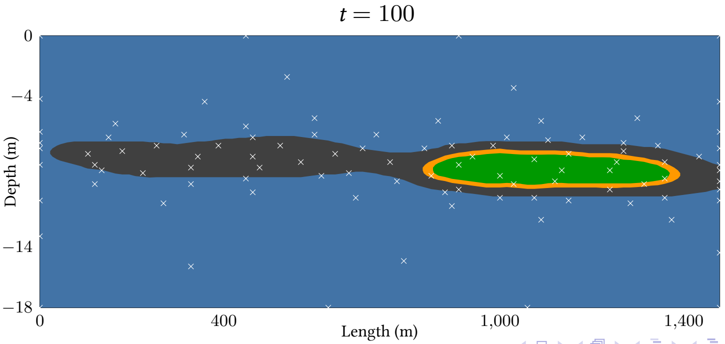


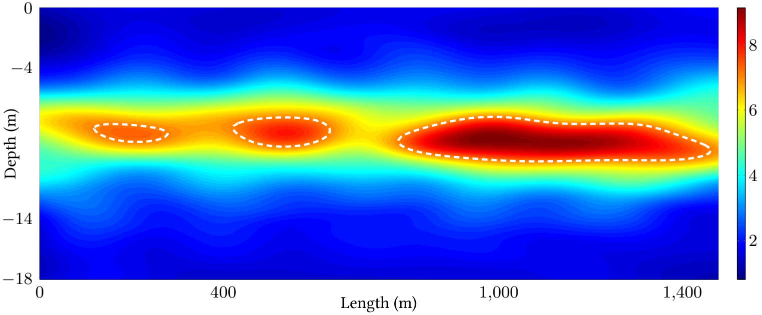
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2



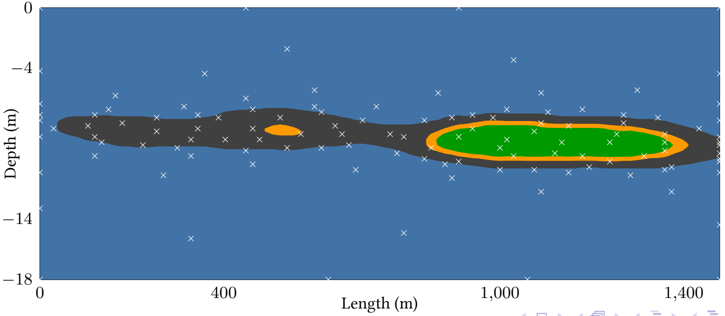
$t = 80$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

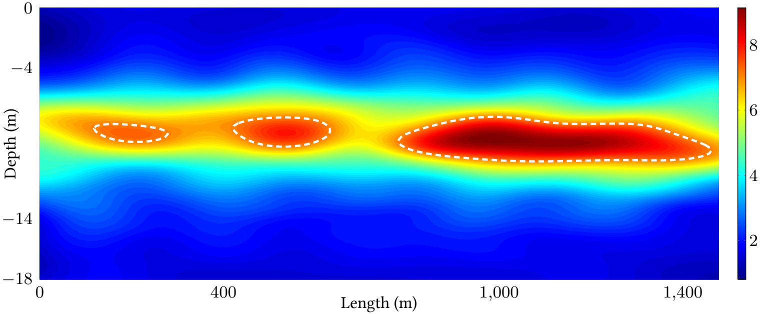

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


$t = 100$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

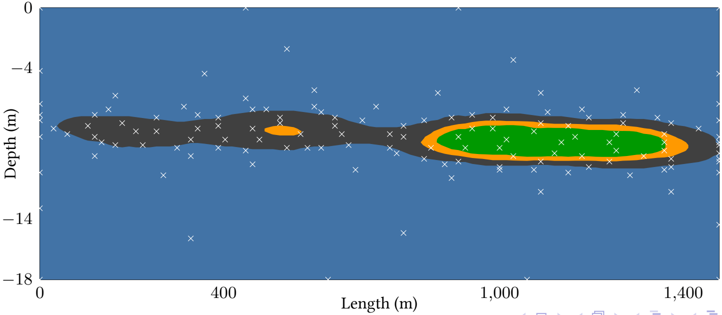

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

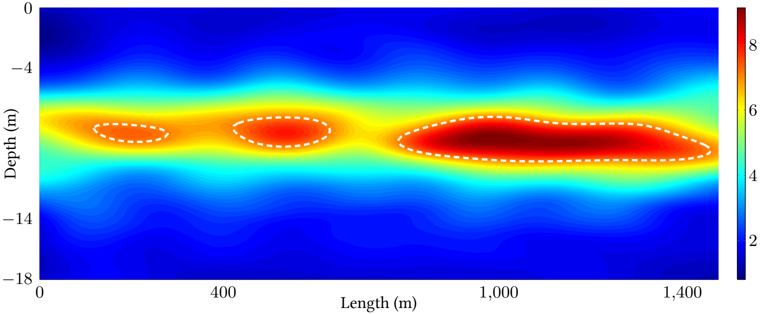
$t = 120$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

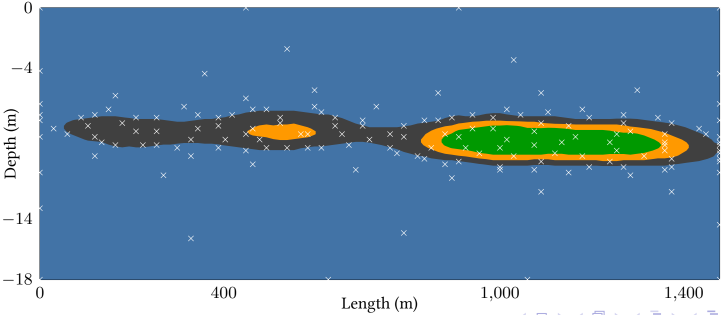

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

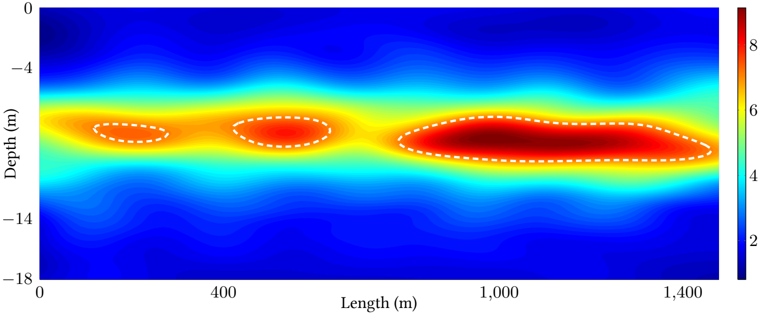
$t = 140$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

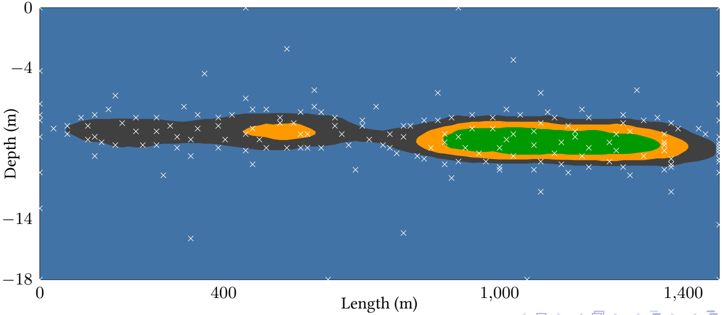

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

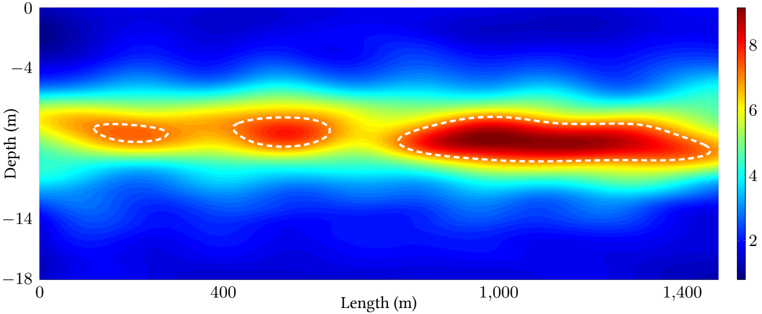
$t = 160$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

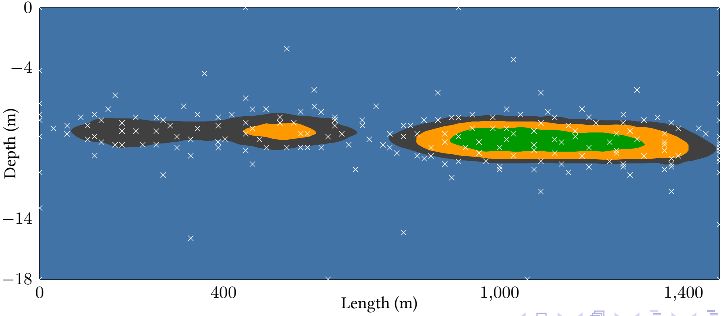

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

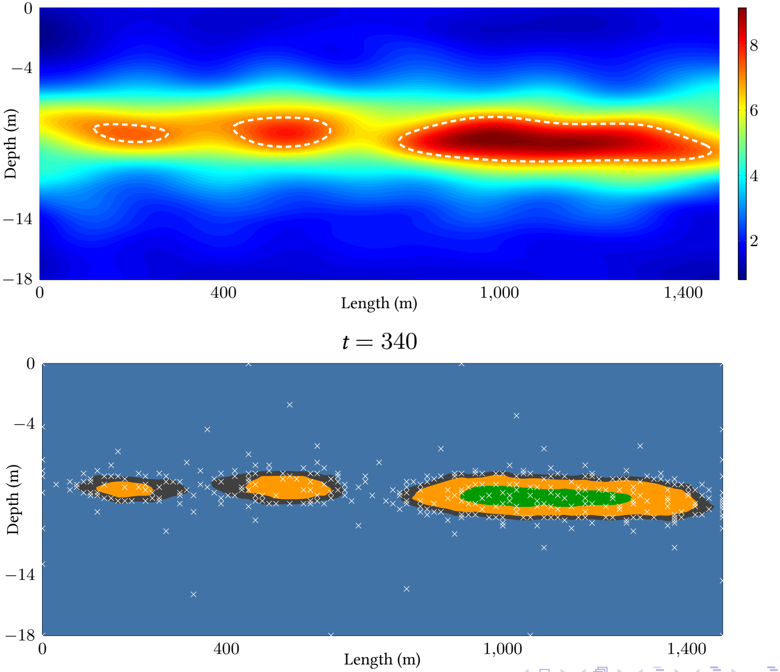
$t = 180$

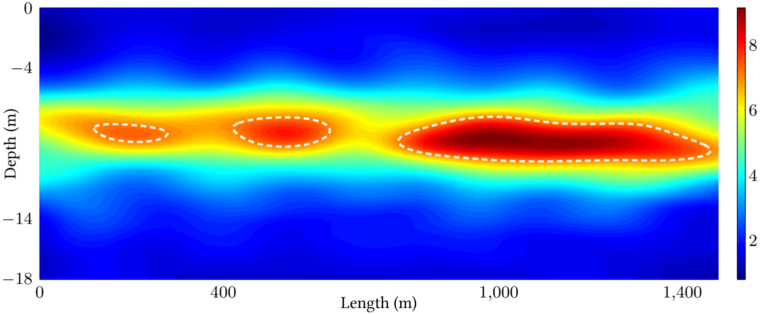

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

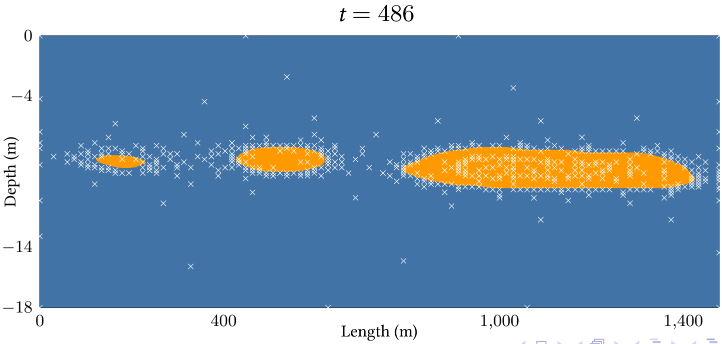

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

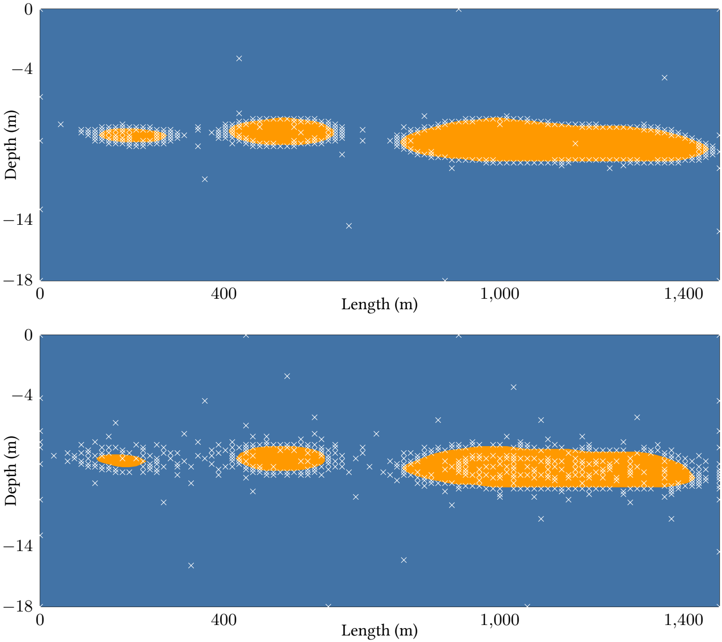
$t = 200$


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2
$t = 340$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


$t = 486$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

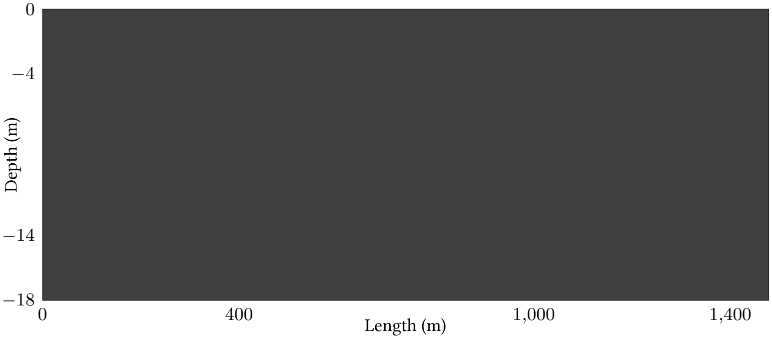

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

- Up to this point we have assumed a fixed cost per sample

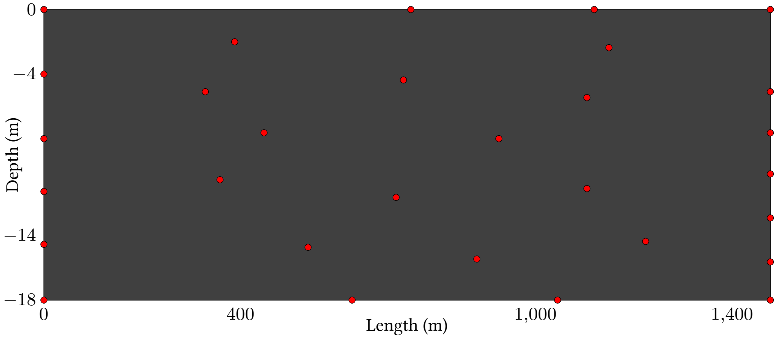
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?

- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

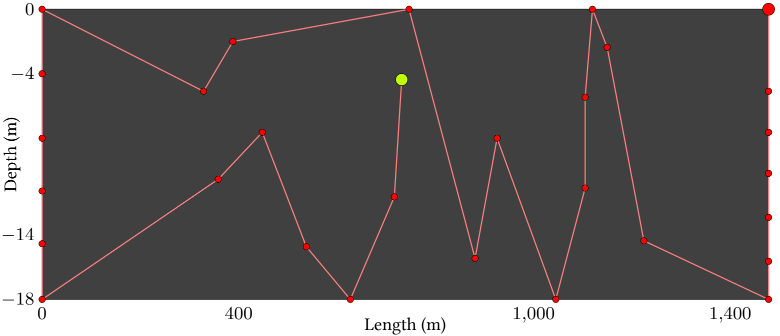
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



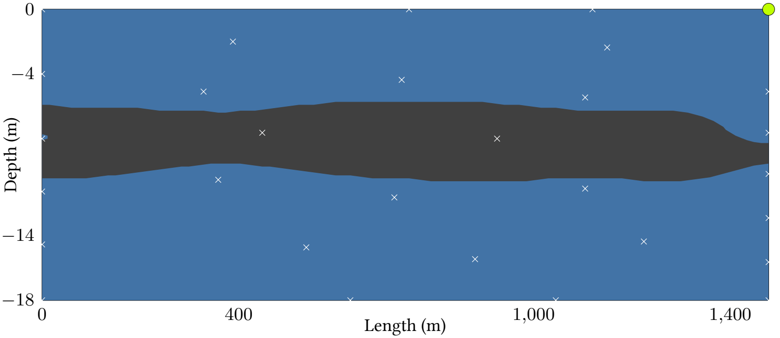
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



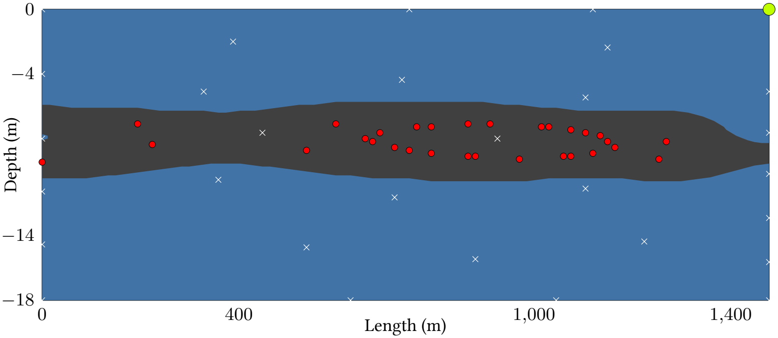
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



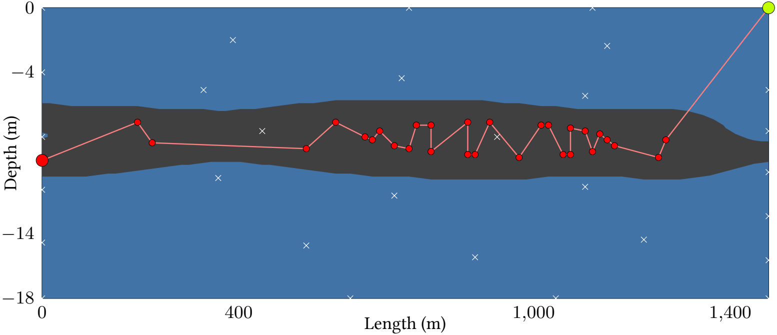
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



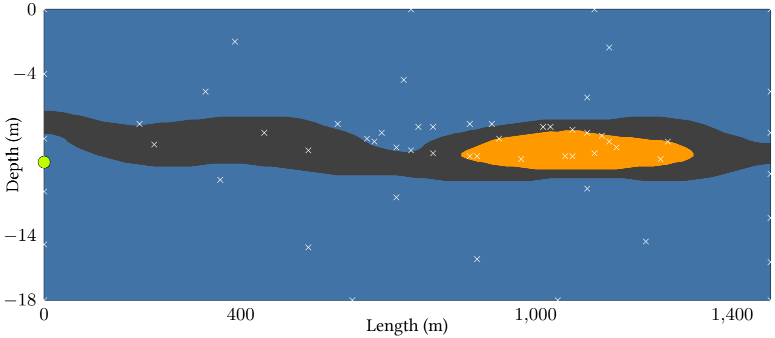
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\mathrm{LSE}_{\mathrm{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



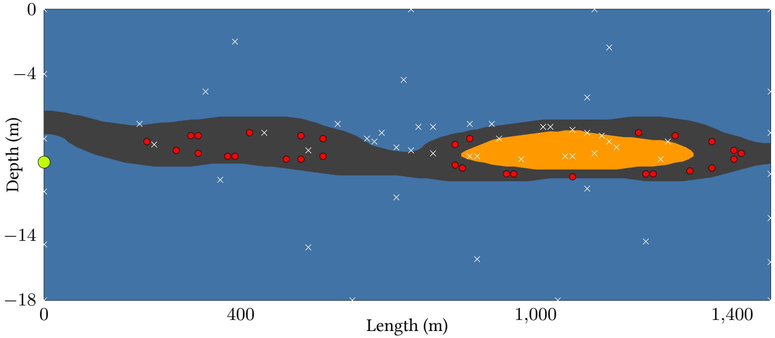
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



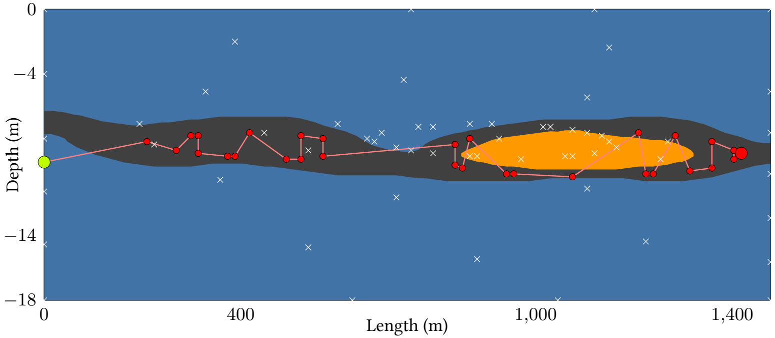
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\mathrm{LSE}_{\mathrm{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



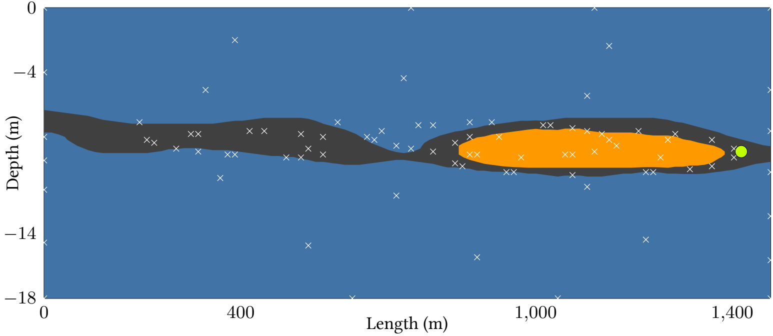
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



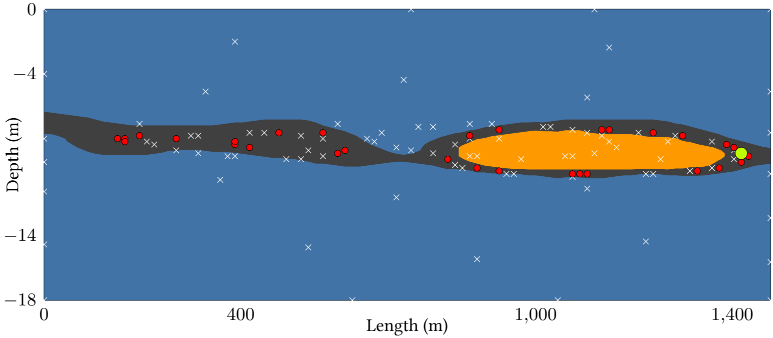
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



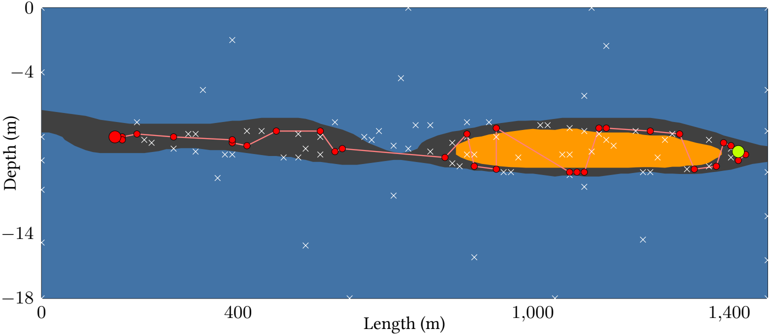
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



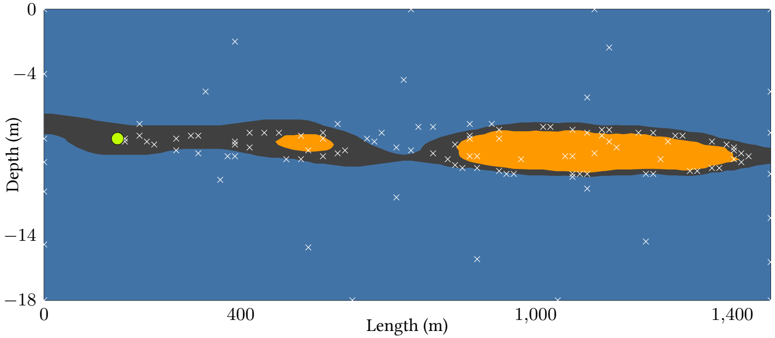
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\mathrm{LSE}_{\mathrm{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



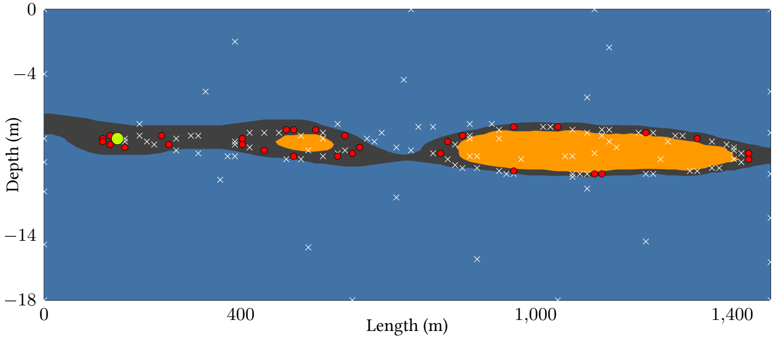
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



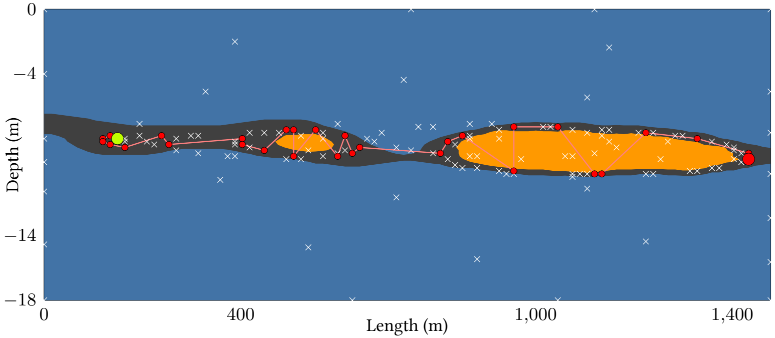
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
    - Select a batch of sampling locations
    - Connect them using a Euclidean TSP path
    - Traverse path and collect measurements



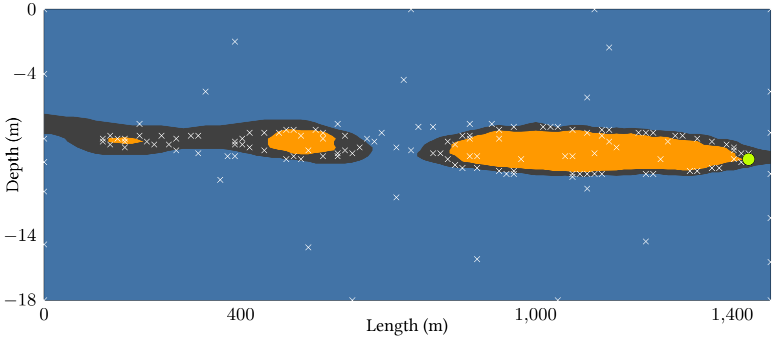
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



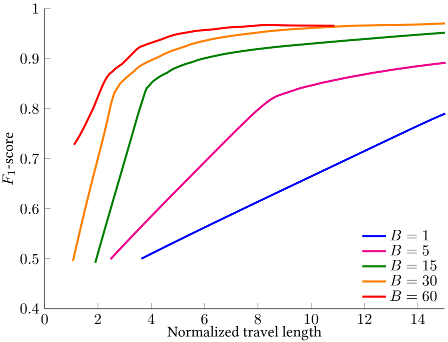
- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\text{LSE}_{\text{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



- Up to this point we have assumed a fixed cost per sample
- What about the traveling distance between measurements?
- We extend LSE to select a *batch* of sampling locations at each step ($\mathrm{LSE}_{\mathrm{batch}}$)
- Use this to plan ahead:
  - Select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



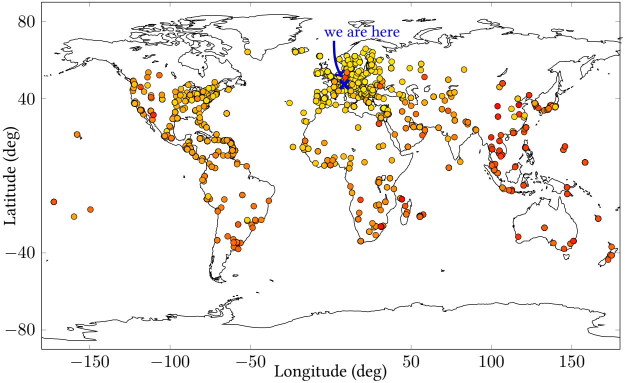
Come to the poster session for more...

Come to the poster session for more...

- ...theory: sample complexity bounds in more detail

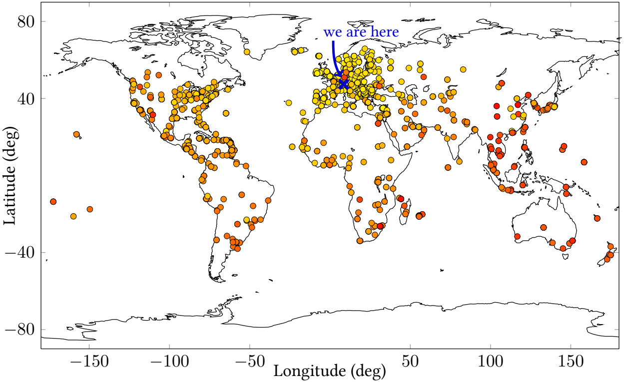
Come to the poster session for more...

- ...theory: sample complexity bounds in more detail
- ...applications: estimate world regions of low internet latency



Come to the poster session for more...

- ...theory: sample complexity bounds in more detail
- ...applications: estimate world regions of low internet latency



- ...results: why does LSE greatly outperform “straddle” in some cases?

Summary

- LSE algorithm:

Summary

- LSE algorithm:
  - Theoretical guarantees

Theorem (Convergence of LSE)

For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.
Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x} \in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1 - \delta.$$

Summary

- LSE algorithm:
  - Theoretical guarantees
  - Competitive with the state-of-the-art in practice (sometimes considerably superior)

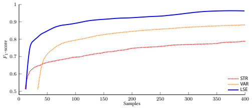
Theorem (Convergence of LSE)

For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x}\in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1-\delta.$$



Summary

- LSE algorithm:
  - Theoretical guarantees
  - Competitive with the state-of-the-art in practice (sometimes considerably superior)

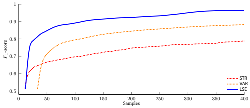
Theorem (Convergence of LSE)

For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x} \in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1 - \delta.$$



- Two useful extensions:

Summary

- LSE algorithm:
  - Theoretical guarantees
  - Competitive with the state-of-the-art in practice (sometimes considerably superior)
- Two useful extensions:
  - Implicit threshold level

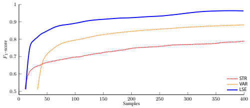
Theorem (Convergence of LSE)

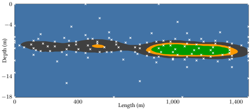
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x} \in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1-\delta.$$





Summary

- LSE algorithm:
  - Theoretical guarantees
  - Competitive with the state-of-the-art in practice (sometimes considerably superior)
- Two useful extensions:
  - Implicit threshold level
  - Batch sampling

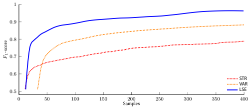
Theorem (Convergence of LSE)

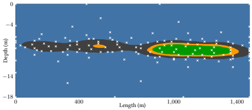
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

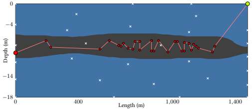
$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{x \in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1-\delta.$$







Summary

- LSE algorithm:
  - Theoretical guarantees

Theorem (Convergence of LSE)

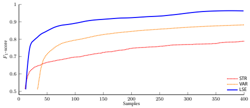
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

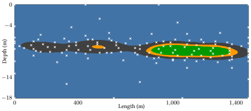
where $C_1 = 8/\log(1+\sigma^{-2})$.
Furthermore, with probability at least $1-\delta$, the algorithm returns an ϵ-accurate solution, that is

$$\Pr\left\{\max_{\boldsymbol{x}\in D} \ell_h(\boldsymbol{x}) \leq \epsilon\right\} \geq 1-\delta.$$

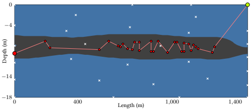
  - Competitive with the state-of-the-art in practice (sometimes considerably superior)



- Two useful extensions:
  - Implicit threshold level



  - Batch sampling



- Look out for algae when swimming in Lake Zurich! :)