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!

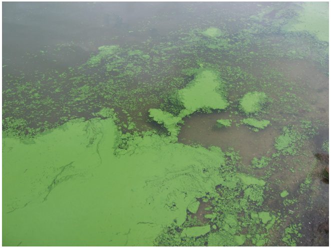
Steffen Schmidt / EPA

Swimmers of Lake Zurich, beware!

> "[...]Switzerland's Lake Zurich [...] an ideal environment for a population explosion of algae including *Planktothrix rubescens* [...]"
>
> — *Scientific American*

Swimmers of Lake Zurich, beware!

www.limnobotics.ch



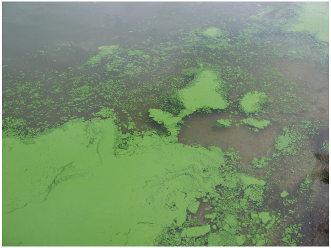
Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

"[...]Switzerland's Lake Zurich [...] an ideal environment for a population explosion of algae including *Planktothrix rubescens* [...]"

— *Scientific American*

Swimmers of Lake Zurich, beware!

www.limnobotics.ch



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

"[...]Switzerland's Lake Zurich [...] an ideal environment for a population explosion of algae including *Planktothrix rubescens* [...]"

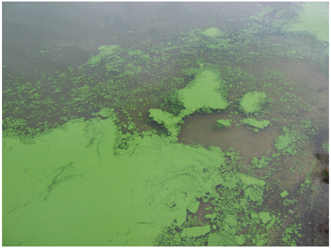
— *Scientific American*

"*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

"[...]Switzerland's Lake Zurich [...] an ideal environment for a population explosion of algae including *Planktothrix rubescens* [...]"

— *Scientific American*

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

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

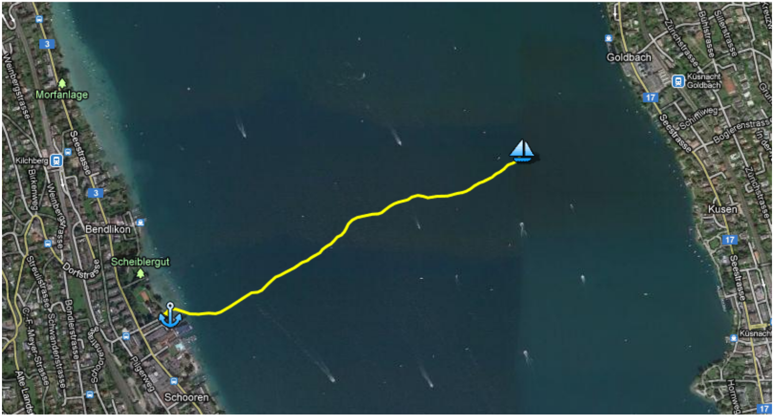
"Microcystins [...] are cyanotoxins and can be very toxic for plants and animals including humans. Their hepatotoxicity may cause serious damage to the liver."

— *Wikipedia*

Autonomous surface vehicle developed by the Autonomous Systems Lab of ETH

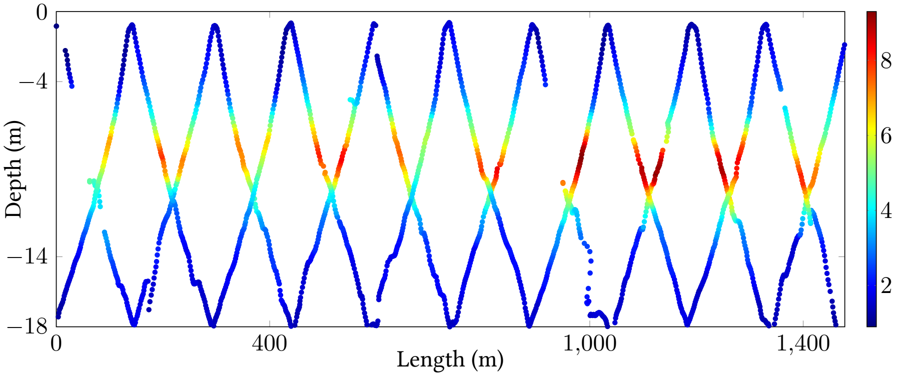
www.limnobotics.ch

Take measurements on a vertical transect of the lake

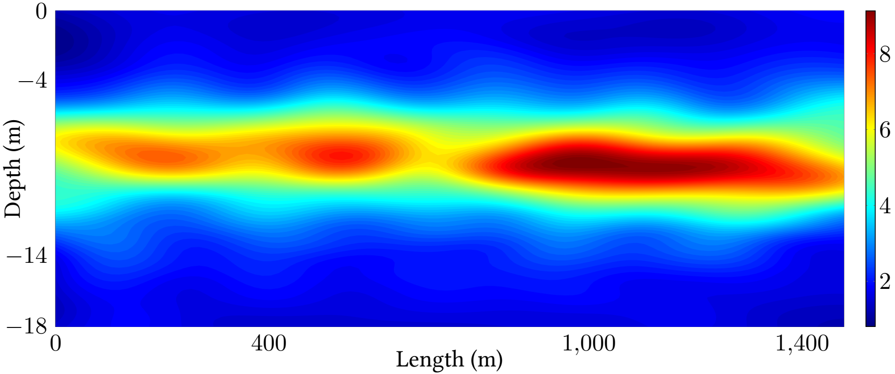


www.limnobotics.ch

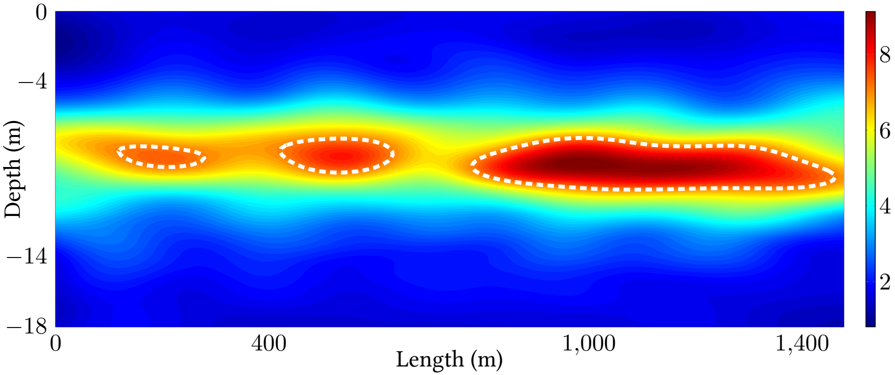
Original algae concentration measurements (~ 2000)



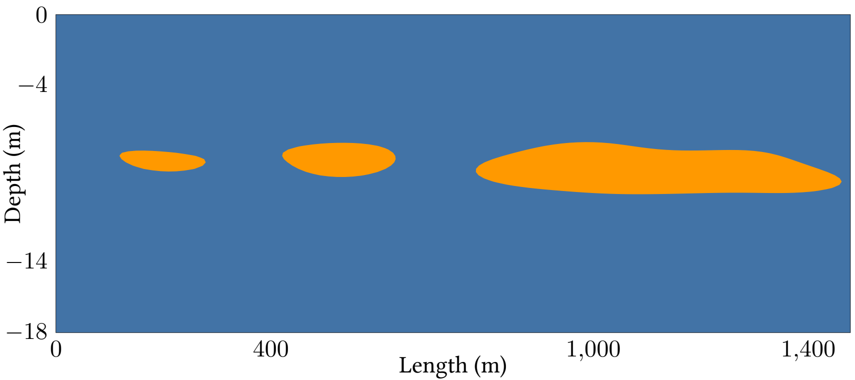
Interpolated algae concentration field



Focus on accurately estimating regions of "high" concentration (e.g. ≥ 7)



Classify transect into a super- and a sublevel set



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

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 classification estimate

1. How do we **estimate** the underlying function from measurements?

1. How do we **estimate** the underlying function from measurements?

2. How do we **classify**?

1. How do we **estimate** the underlying function from measurements?
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 underlying function from measurements?

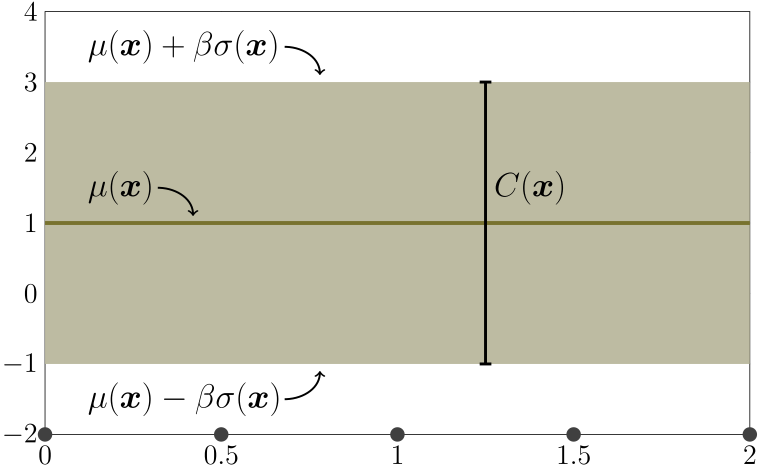
2. How do we **classify**?

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

Gaussian processes to the rescue!

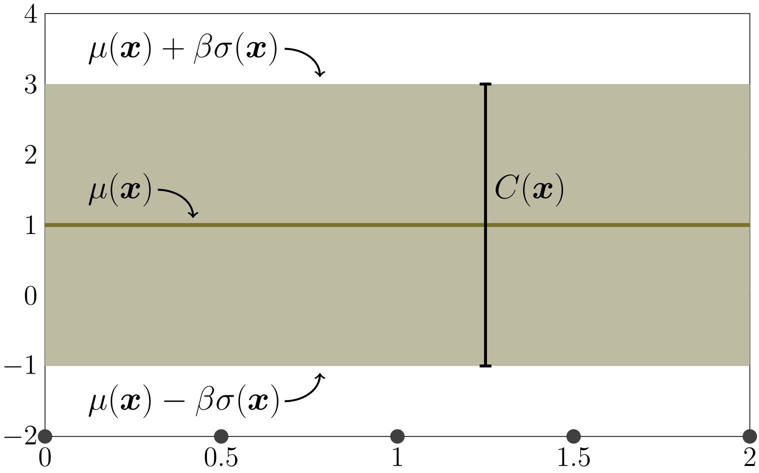
Gaussian processes

- Mean **and variance** estimates: construct confidence intervals $C(\boldsymbol{x})$



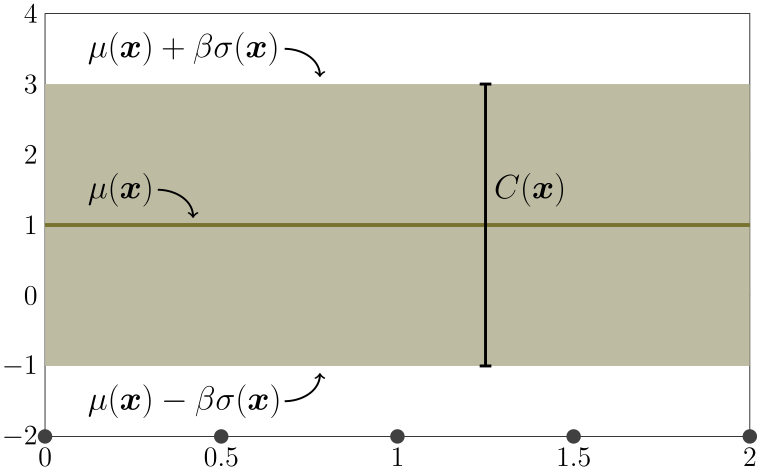
Gaussian processes

- Mean **and variance** estimates: construct confidence intervals $C(\boldsymbol{x})$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates



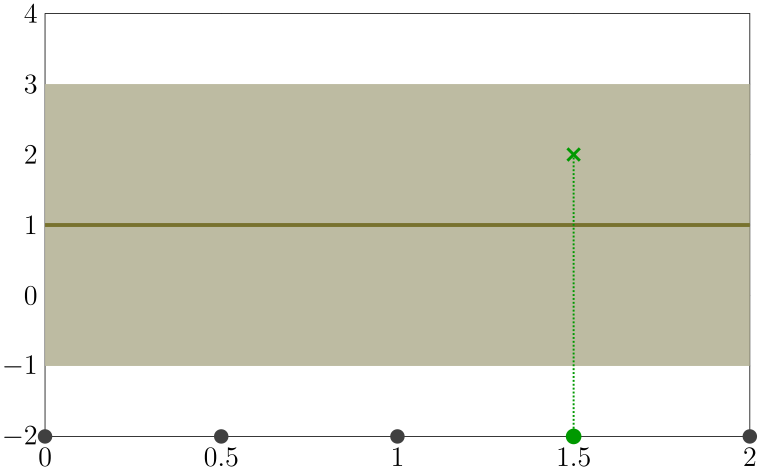
Gaussian processes

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



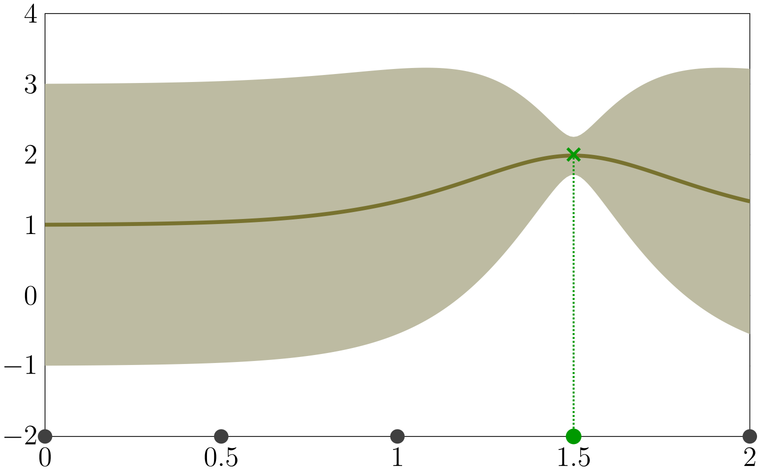
Gaussian processes

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



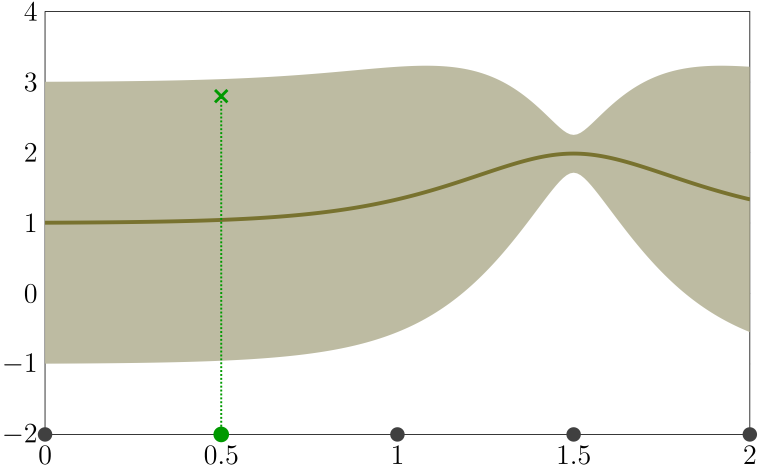
Gaussian processes

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



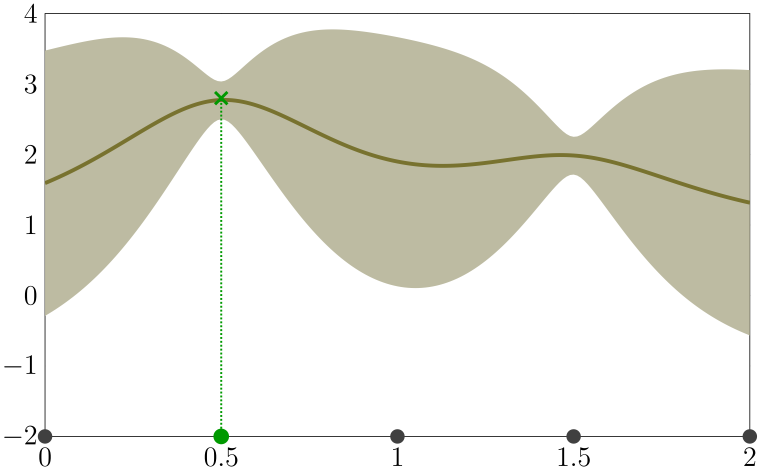
Gaussian processes

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



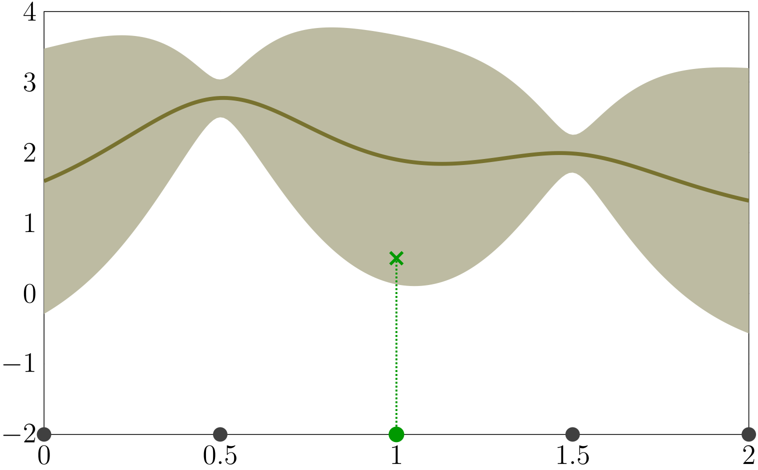
Gaussian processes

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



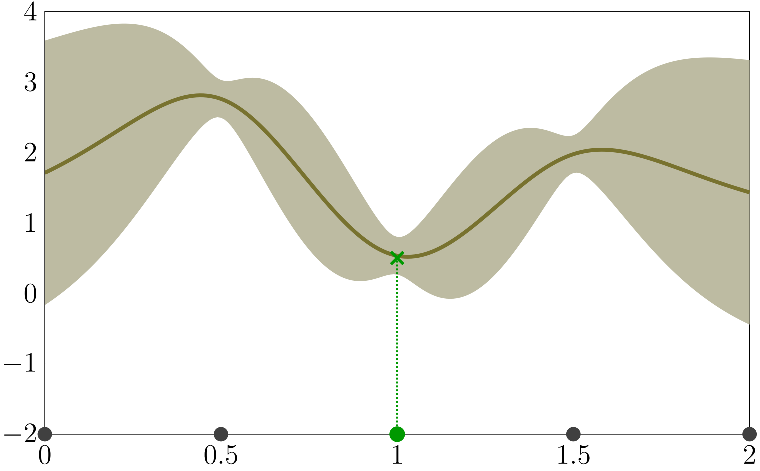
Gaussian processes

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



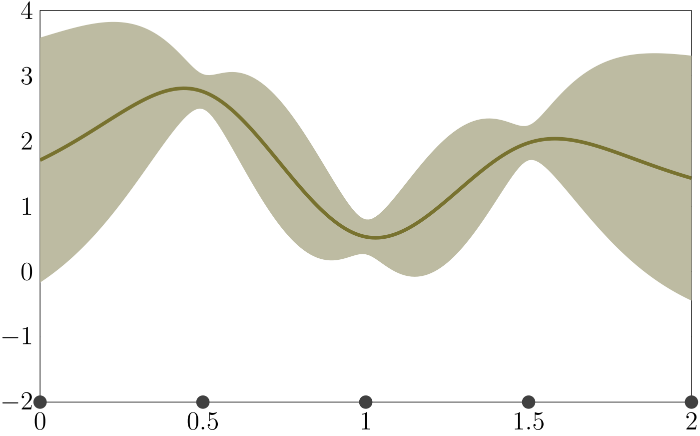
Gaussian processes

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

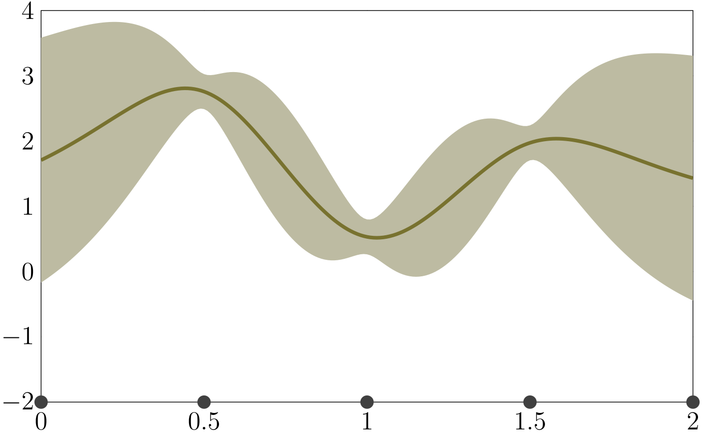


1. How do we **estimate** the function?

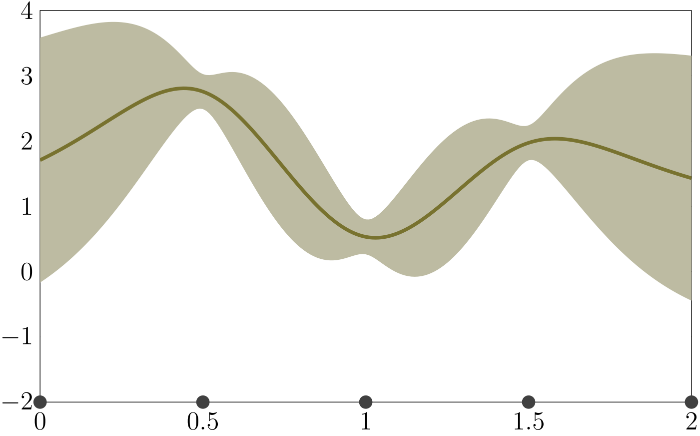
1. How do we **estimate** the function?



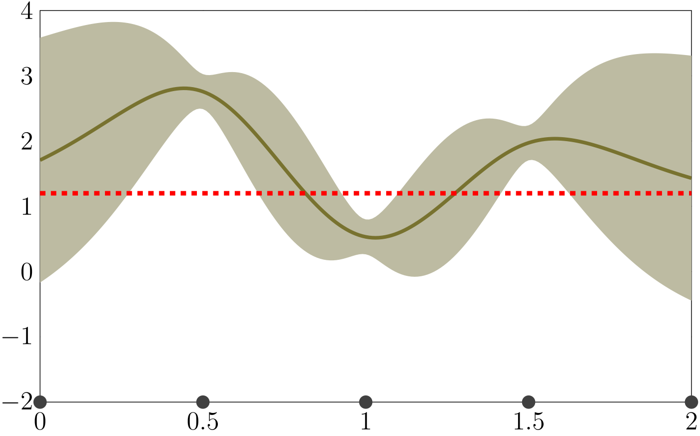
1. How do we **estimate** the function? ✓



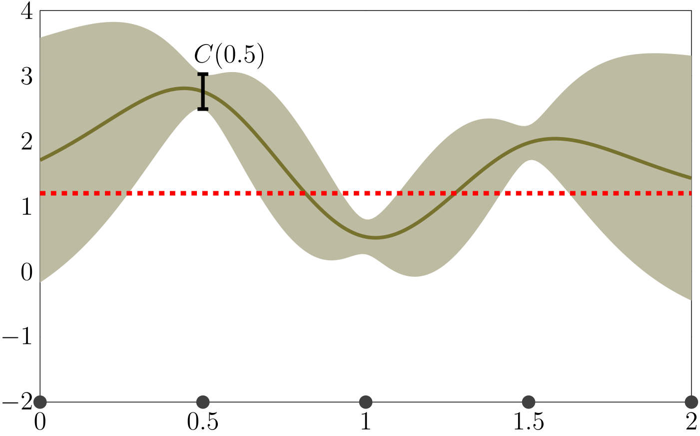
1. How do we **estimate** the function? ✓
2. How do we **classify**?



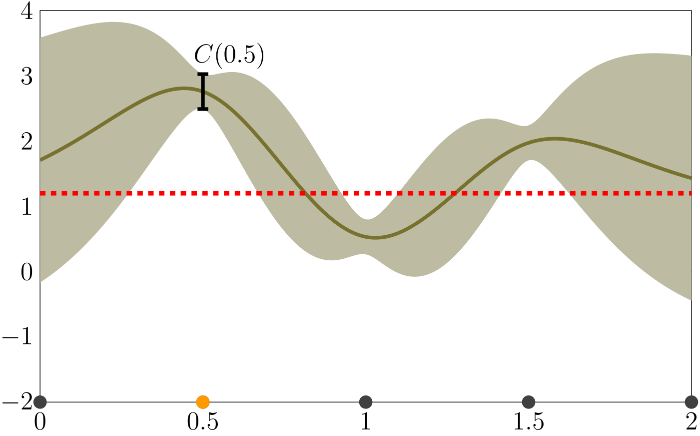
1. How do we **estimate** the function? ✓
2. How do we **classify**?



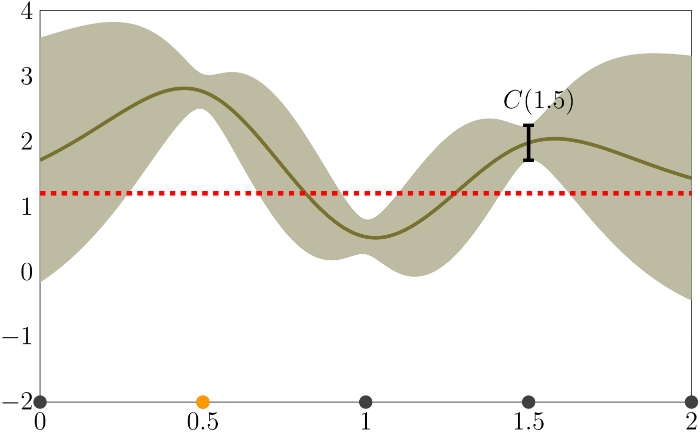
1. How do we **estimate** the function? ✓
2. How do we **classify**?



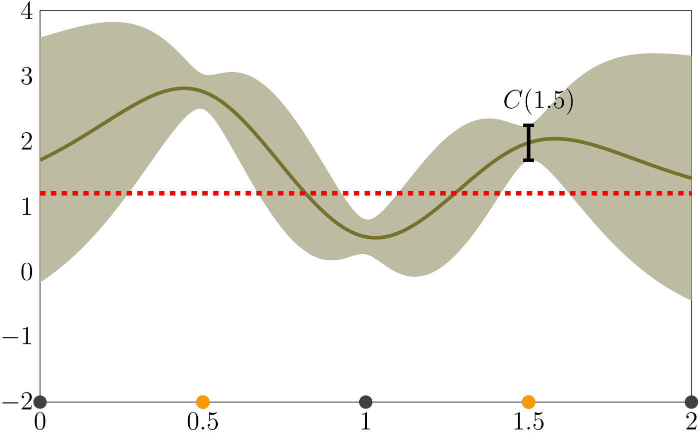
1. How do we **estimate** the function? ✓
2. How do we **classify**?



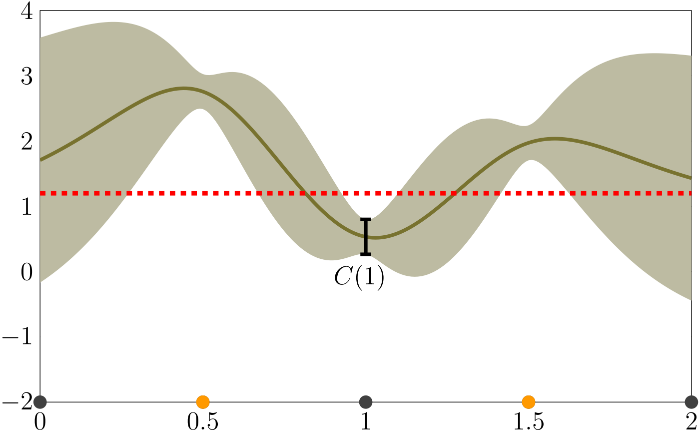
1. How do we **estimate** the function? ✓
2. How do we **classify**?



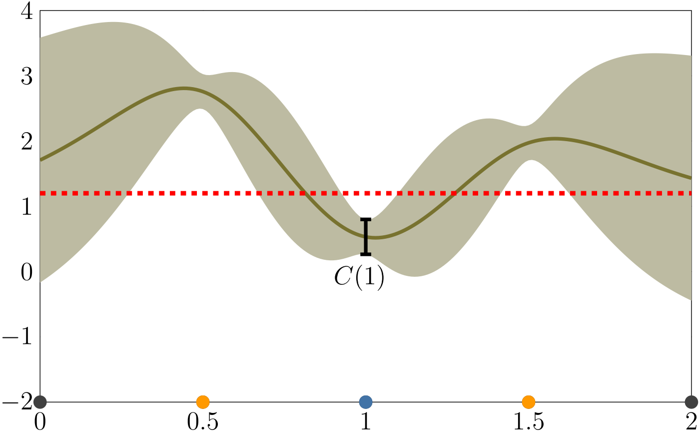
1. How do we **estimate** the function? ✓
2. How do we **classify**?



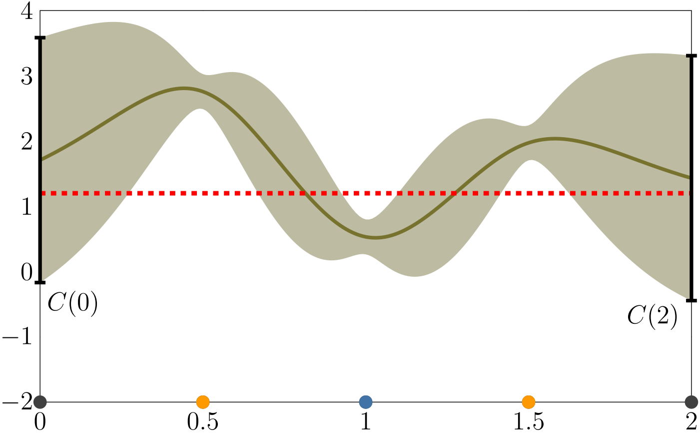
1. How do we **estimate** the function? ✓
2. How do we **classify**?



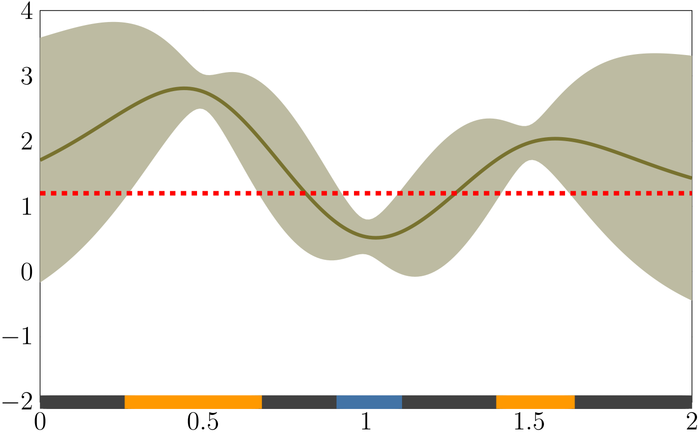
1. How do we **estimate** the function? ✓
2. How do we **classify**?



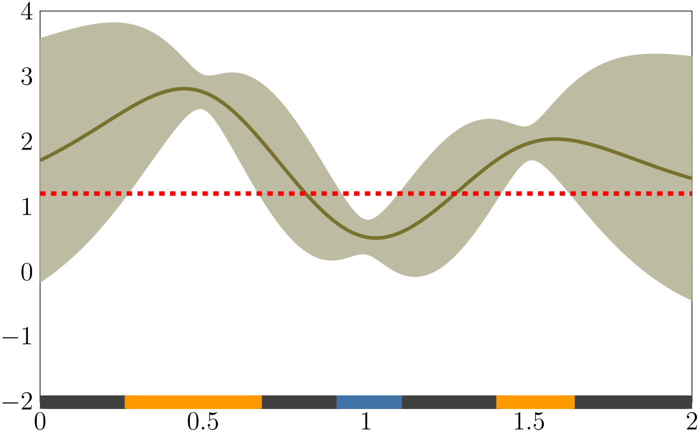
1. How do we **estimate** the function? ✓
2. How do we **classify**?



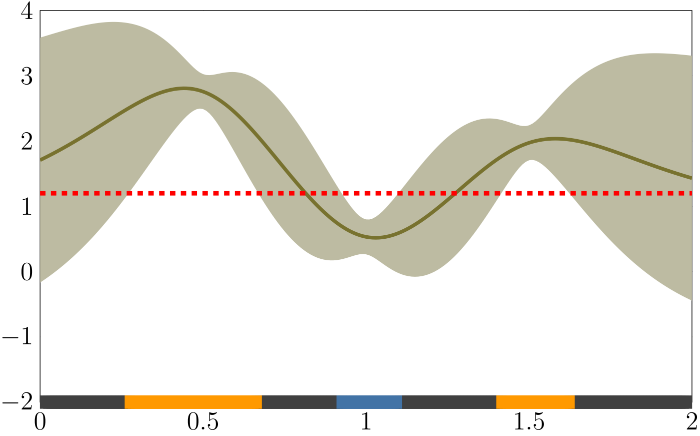
1. How do we **estimate** the function? ✓
2. How do we **classify**?



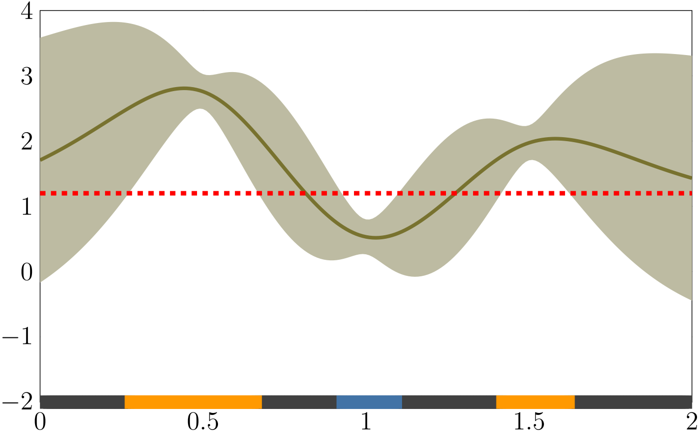
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓



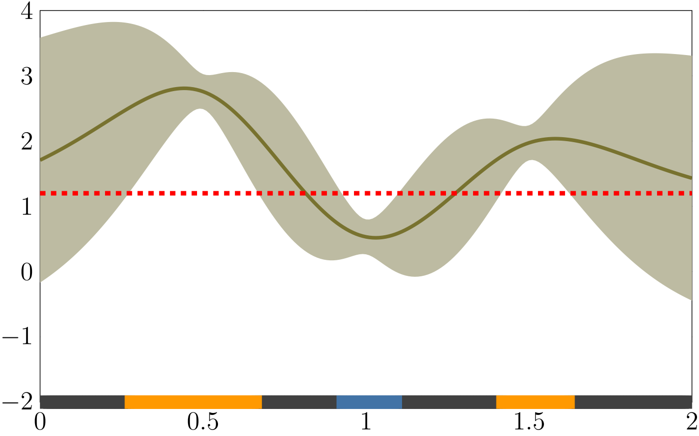
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?



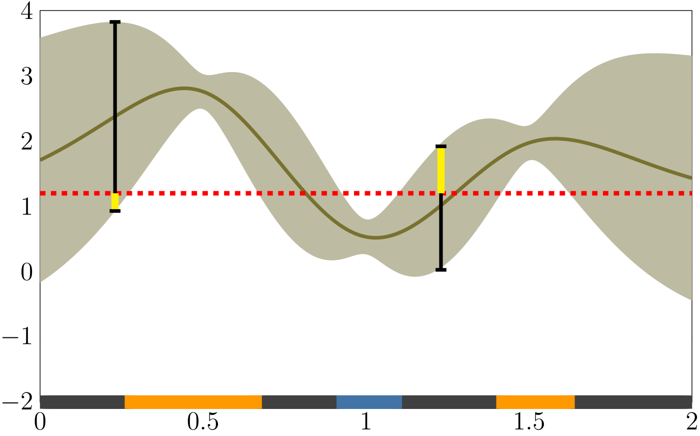
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** "informative" measurements?
   - Pick among the yet unclassified...



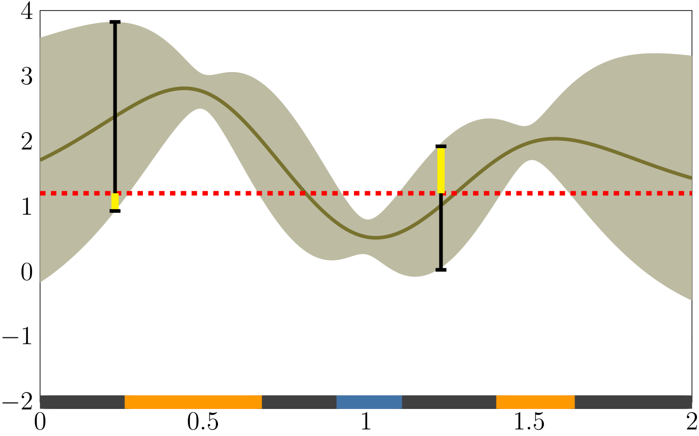
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
    - Pick among the yet unclassified...
    - ...the most “ambiguous” point



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



The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

while $\exists$ unclassified points in D **do**

end while

The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

while $\exists$ unclassified points in D **do**
  for all unclassified points $\boldsymbol{x} \in D$ **do**
    if $C_t(\boldsymbol{x})$ lies above h **then**
      classify $\boldsymbol{x}$ into superlevel set
    else if $C_t(\boldsymbol{x})$ lies below h **then**
      classify $\boldsymbol{x}$ into sublevel set
    else
      leave $\boldsymbol{x}$ unclassified
    end if
  end for
end while

$\leftarrow$ classify

The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

while $\exists$ unclassified points in D **do**
  for all unclassified points $\boldsymbol{x} \in D$ **do**
    if $C_t(\boldsymbol{x})$ lies above h **then**
      classify $\boldsymbol{x}$ into superlevel set
    else if $C_t(\boldsymbol{x})$ lies below h **then**
      classify $\boldsymbol{x}$ into sublevel set
    else
      leave $\boldsymbol{x}$ unclassified
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \text{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$

end while

$\leftarrow$ classify

$\leftarrow$ sample

The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

while $\exists$ unclassified points in D **do**
  for all unclassified points $\boldsymbol{x} \in D$ **do**
    if $C_t(\boldsymbol{x})$ lies above h **then**
      classify $\boldsymbol{x}$ into superlevel set
    else if $C_t(\boldsymbol{x})$ lies below h **then**
      classify $\boldsymbol{x}$ into sublevel set
    else
      leave $\boldsymbol{x}$ unclassified
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  perform GP inference
end while

$\leftarrow$ classify

$\leftarrow$ sample

The Level Set Estimation (LSE) algorithm

Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

while $\exists$ unclassified points in D **do**
  for all unclassified points $\boldsymbol{x} \in D$ **do**
    if $C_t(\boldsymbol{x})$ lies above h **then**
      classify $\boldsymbol{x}$ into superlevel set
    else if $C_t(\boldsymbol{x})$ lies below h **then**
      classify $\boldsymbol{x}$ into sublevel set
    else
      leave $\boldsymbol{x}$ unclassified
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \text{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  perform GP inference
end while

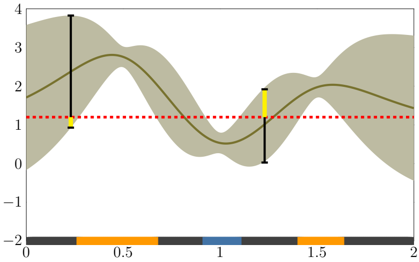
- Monotonicity of
  1. confidence intervals
  2. classification

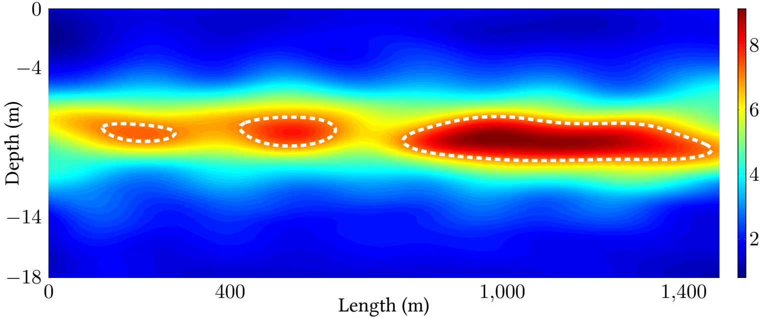
The Level Set Estimation (LSE) algorithm

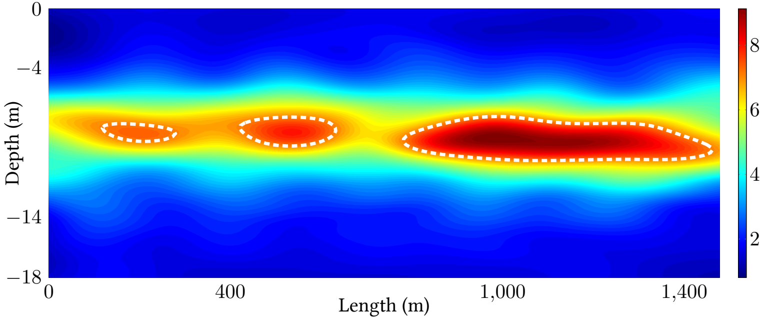
Input: sample space D, threshold level h
Output: predicted super- and sublevel sets

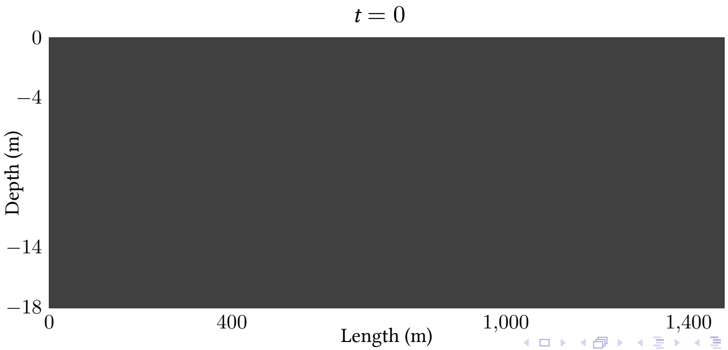
while $\exists$ unclassified points in D **do**
  for all unclassified points $\boldsymbol{x} \in D$ **do**
    if $C_t(\boldsymbol{x})$ lies above h **then**
      classify $\boldsymbol{x}$ into superlevel set
    else if $C_t(\boldsymbol{x})$ lies below h **then**
      classify $\boldsymbol{x}$ into sublevel set
    else
      leave $\boldsymbol{x}$ unclassified
    end if
  end for
  $\boldsymbol{x}_t \leftarrow \operatorname{argmax}\{a_t(\boldsymbol{x}) \mid \boldsymbol{x} \in U_t\}$
  $y_t \leftarrow f(\boldsymbol{x}_t) + n_t$
  perform GP inference
end while

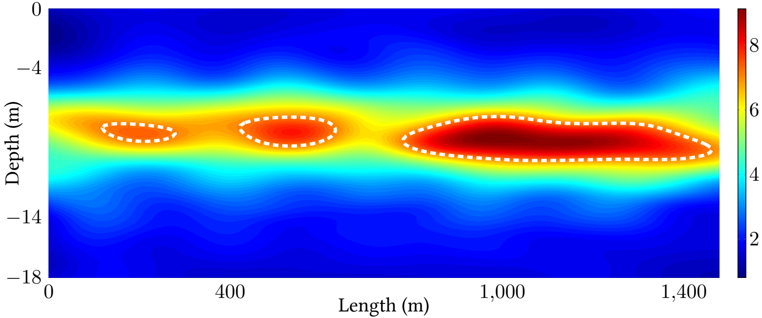
- Monotonicity of
  1. confidence intervals
  2. classification
- Relax classification rules by an accuracy parameter ϵ



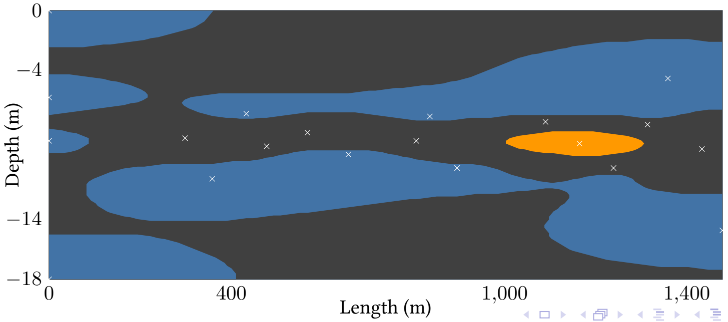

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

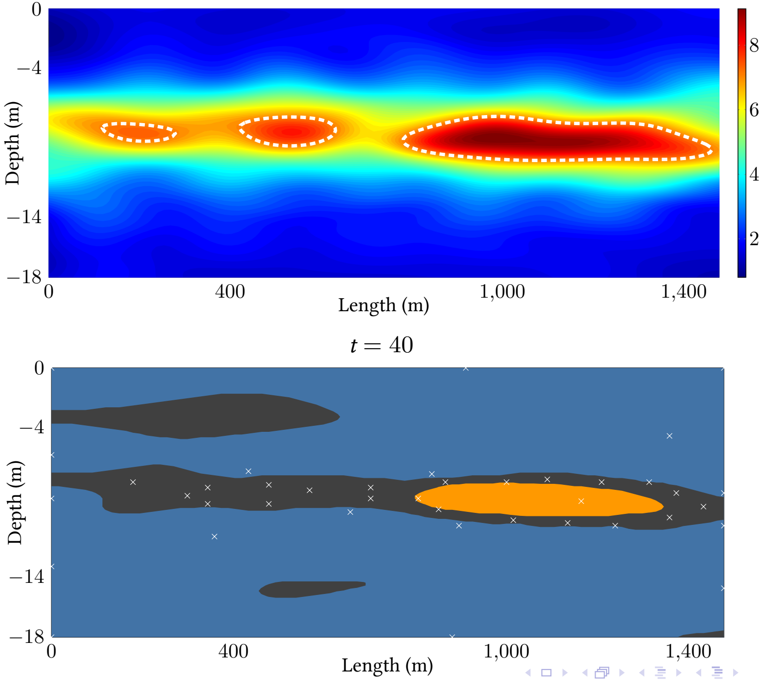

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

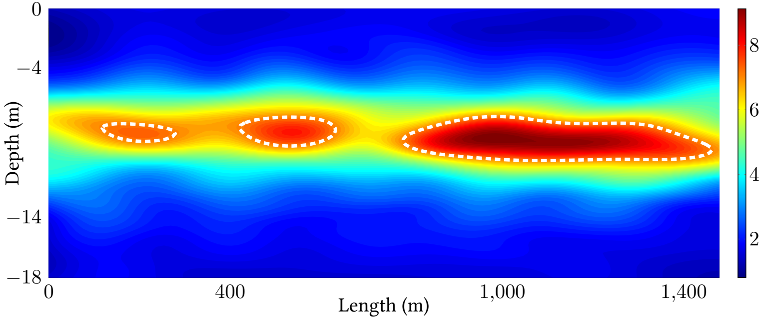

$t = 0$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

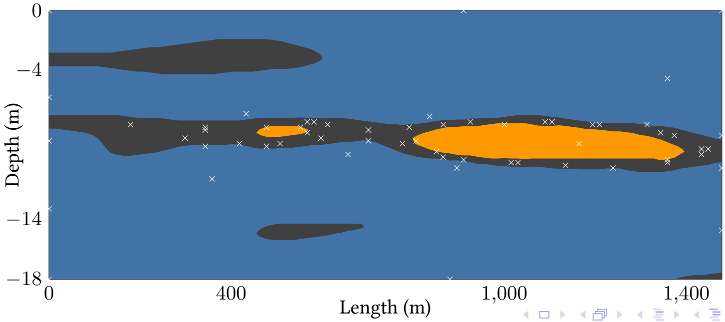
$t = 20$

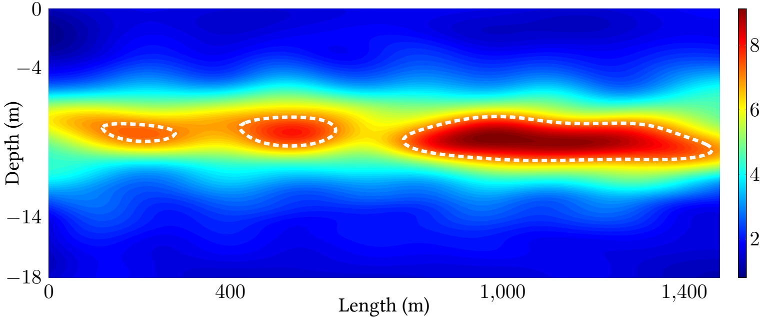

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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)

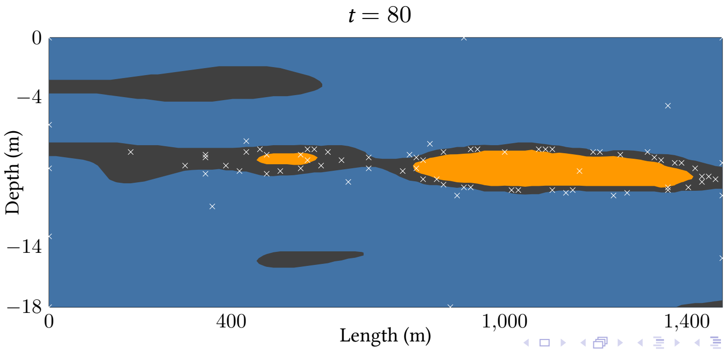

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

$t = 60$

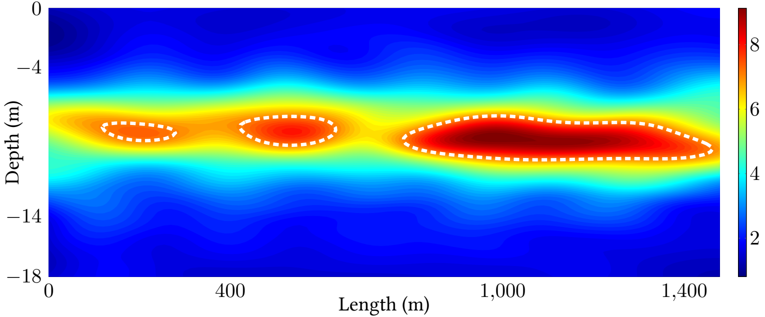

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)



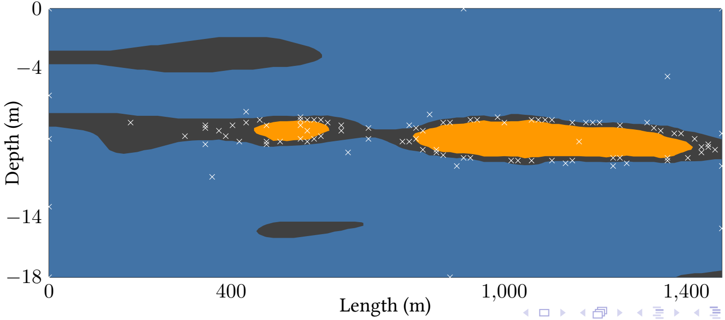
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

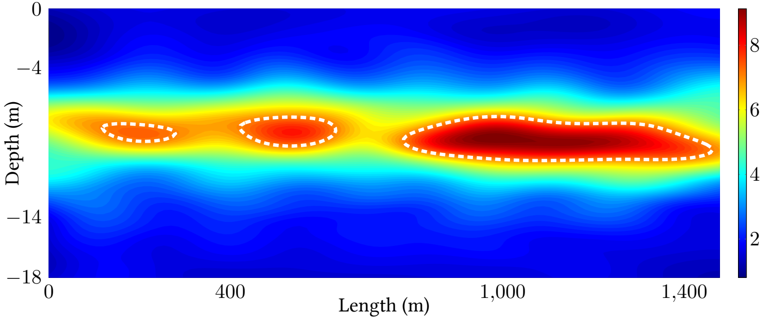


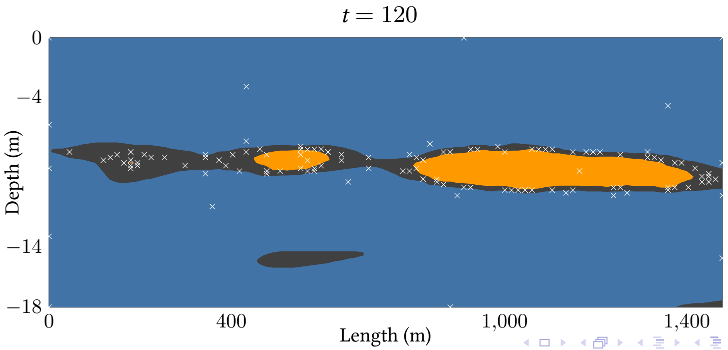
$t = 80$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

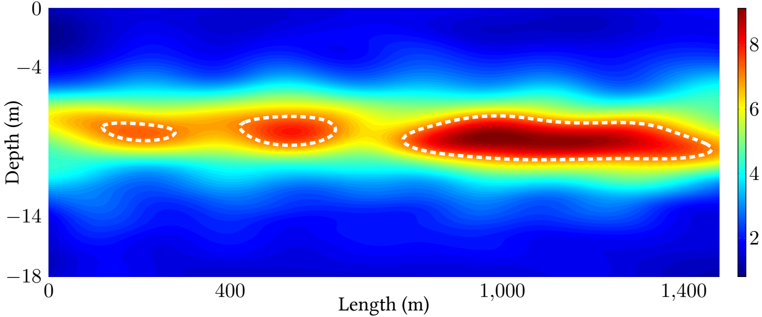

Depth (m)
0
−4
−14
−18
0
400
1,000
1,400
Length (m)
8
6
4
2

$t = 100$

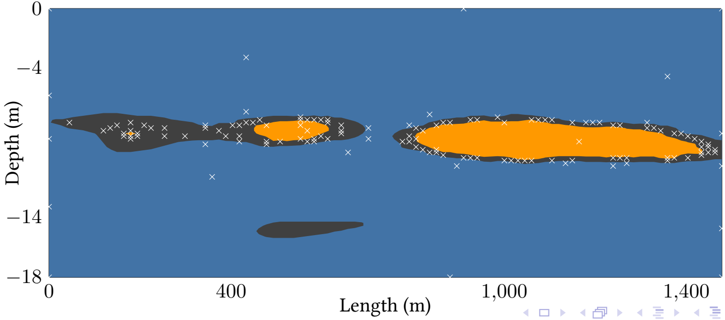

Depth (m)
0
−4
−14
−18
0
400
1,000
1,400
Length (m)

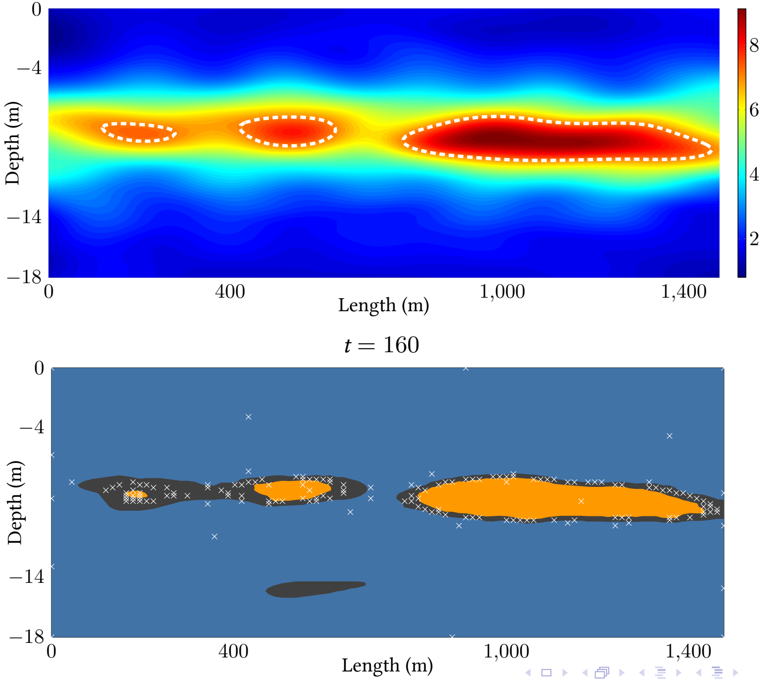

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2


$t = 120$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

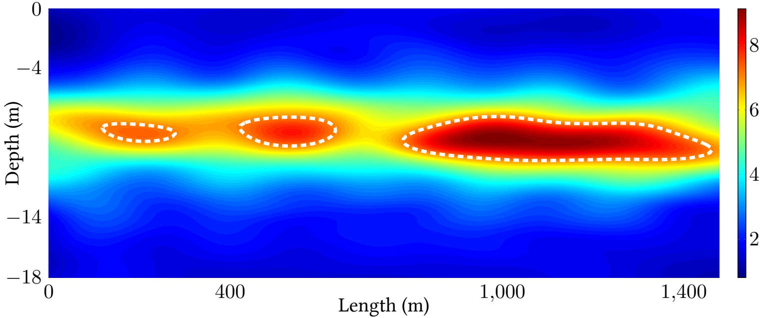

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

$t = 140$

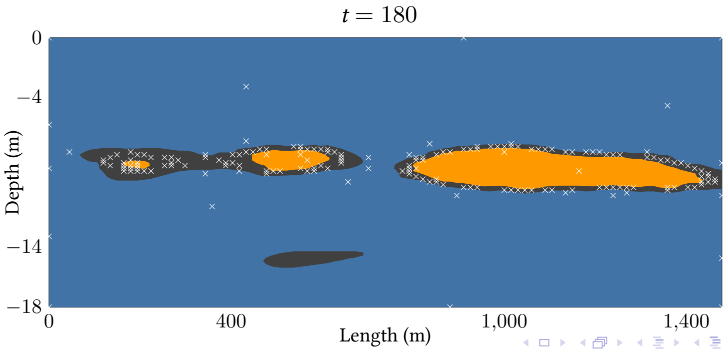

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)



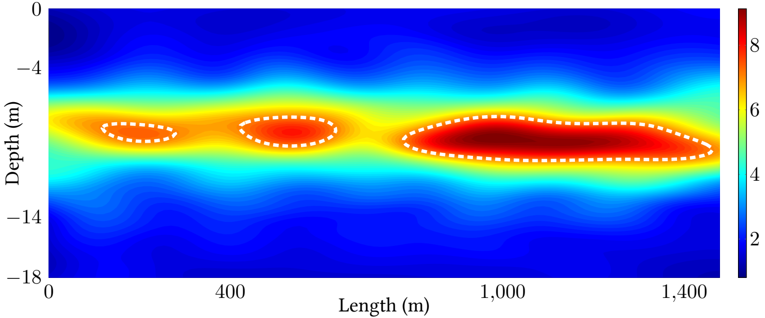
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2
t = 160
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

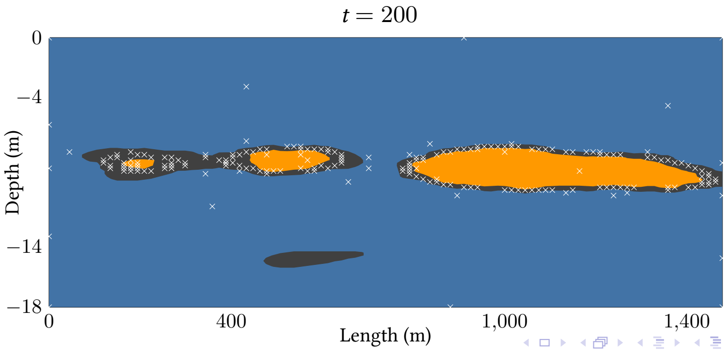


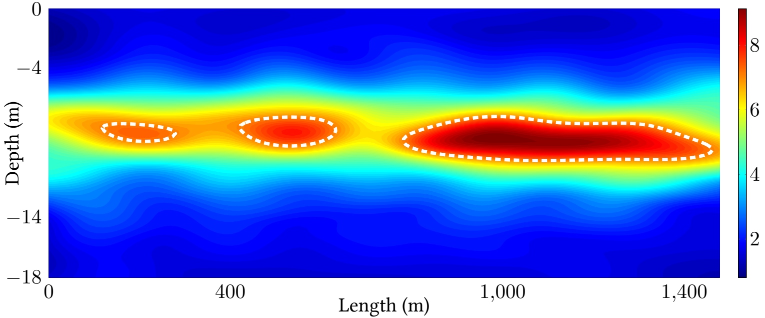
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

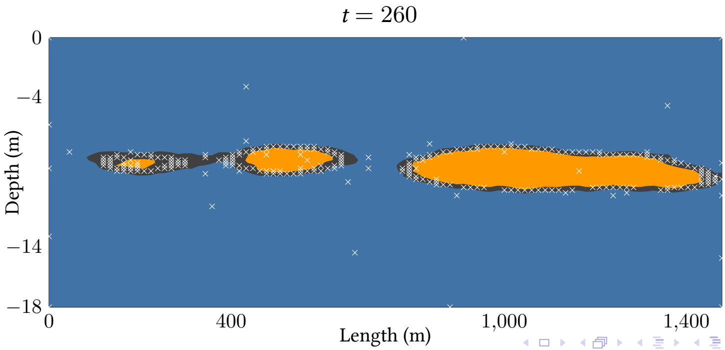


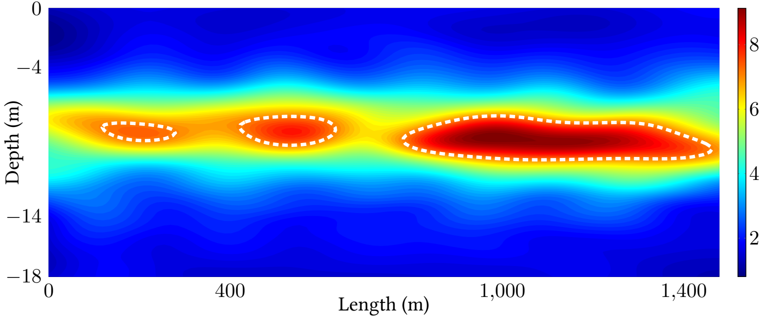
$t = 180$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

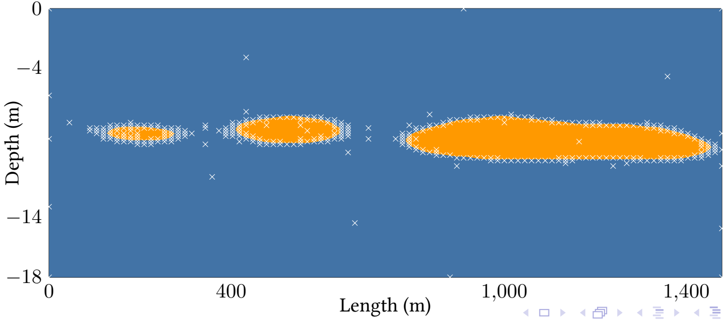

t = 200
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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 *appropriately (large enough), then:*

Theorem (Simplified)

If we choose β appropriately (large enough), then:

- *LSE terminates after a number of iterations T*

Theorem (Simplified)

If we choose β appropriately (large enough), then:

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

Theorem (Simplified)

If we choose β appropriately (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 appropriately (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 appropriately (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 appropriately (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:

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

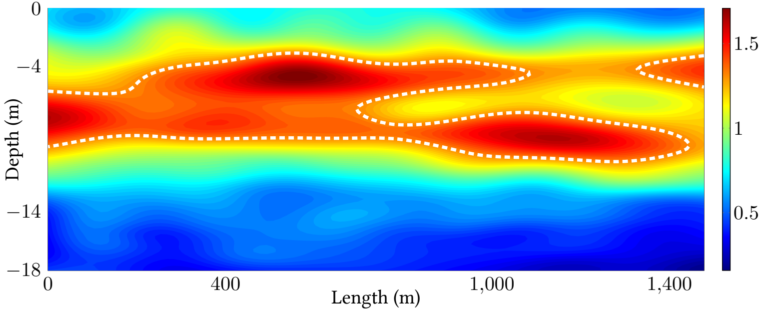
Experiments

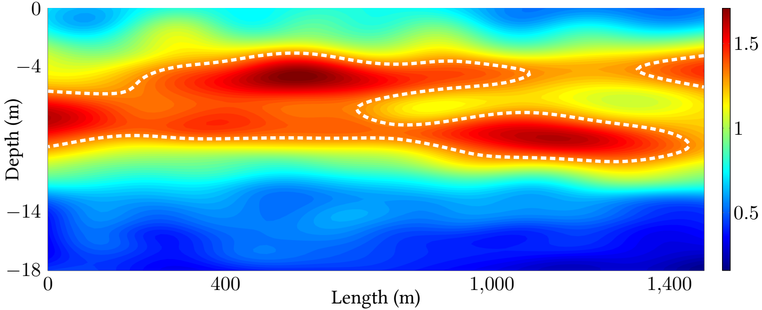
1. LSE
2. Maximum variance sampling:

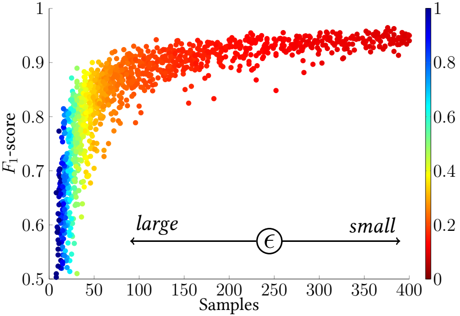
$$\boldsymbol{x}_t = \operatorname*{argmax}_{\boldsymbol{x} \in D} \sigma_{t-1}(\boldsymbol{x})$$

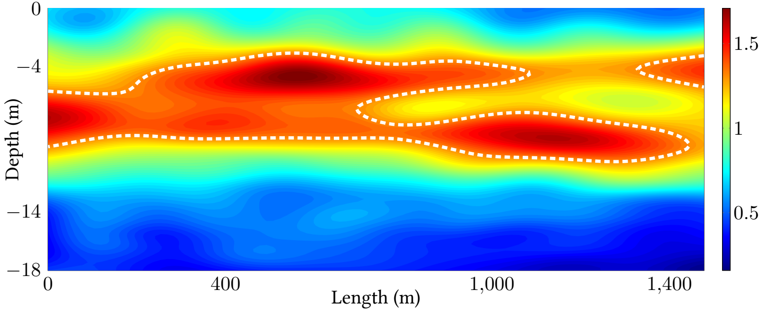
3. State of the art "straddle" heuristic (Bryan *et al.*, 2005):

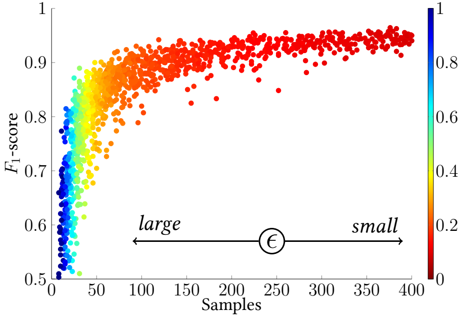
$$\boldsymbol{x}_t \approx \operatorname*{argmax}_{\boldsymbol{x} \in D} a_{t-1}(\boldsymbol{x}) \qquad \text{(for } \beta_t^{1/2} = 1.96\text{)}$$

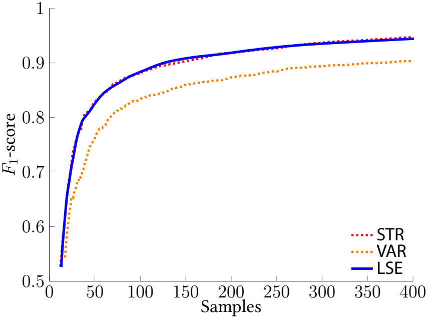

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
1.5
1
0.5

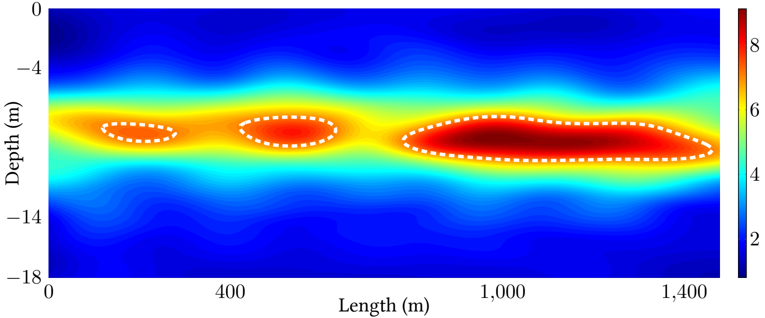

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
1.5
1
0.5

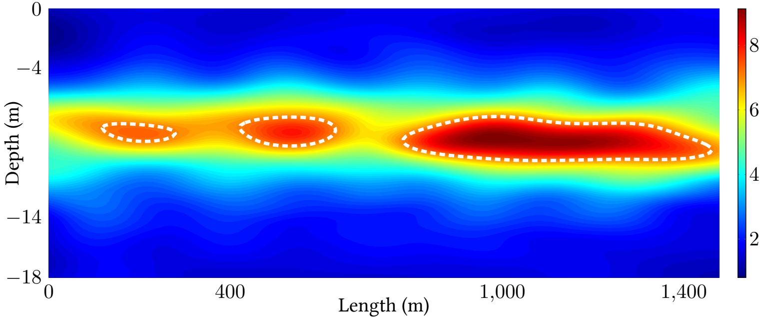

1
0.9
0.8
0.7
0.6
0.5
F_1-score
0
50
100
150
200
250
300
350
400
Samples
large
ϵ
small
1
0.8
0.6
0.4
0.2
0


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

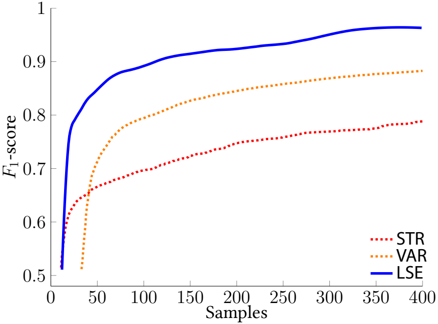

1
0.9
0.8
0.7
0.6
0.5
F_1-score
0
50
100
150
200
250
300
350
400
Samples
large
ϵ
small
1
0.8
0.6
0.4
0.2
0


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

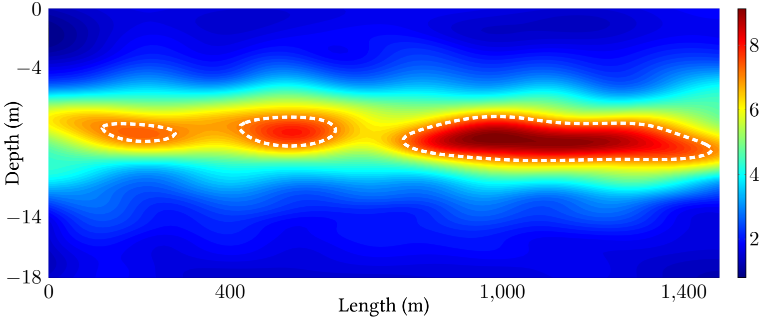

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

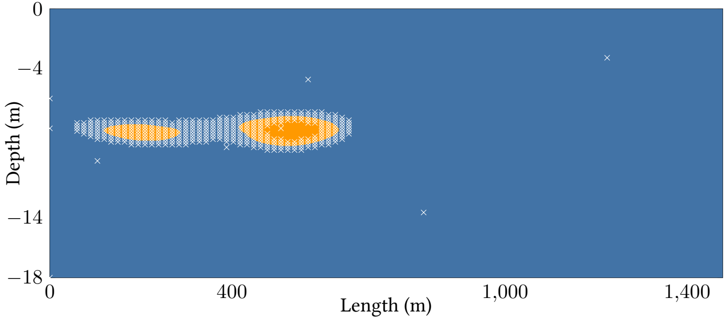


0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2



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


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)

Implicit threshold level

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

Implicit threshold level

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

Implicit threshold level

- 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)

Implicit threshold level

- 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

Implicit threshold level

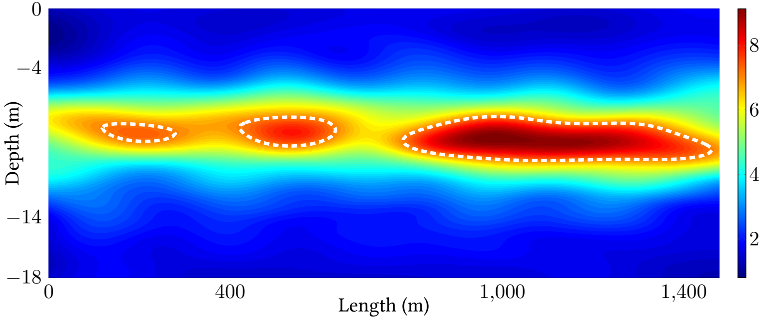
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- $\mathbf{LSE_{imp}}$ algorithm:

Implicit threshold level

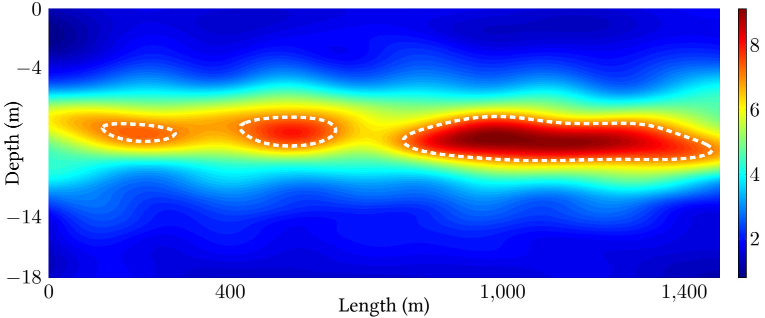
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- **LSE$_{\text{imp}}$** algorithm:
  - h is now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

Implicit threshold level

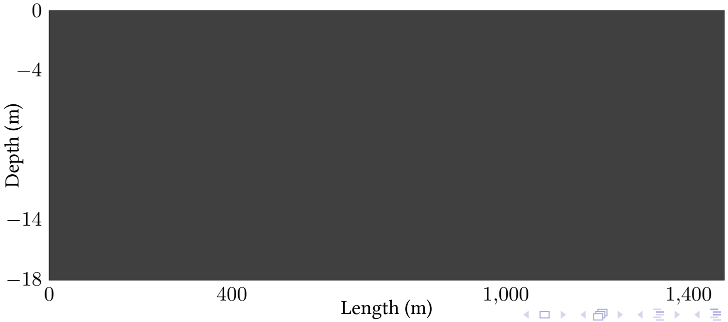
- What if we don't know the threshold level h?
- Implicitly define $h = \omega \max_{\boldsymbol{x} \in D} f(\boldsymbol{x}),\ 0 < \omega < 1$
- No existing algorithms for this problem (to our knowledge)
- We extend LSE (and its theory!) to this setting
- **LSE$_{\textbf{imp}}$** algorithm:
  - h is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

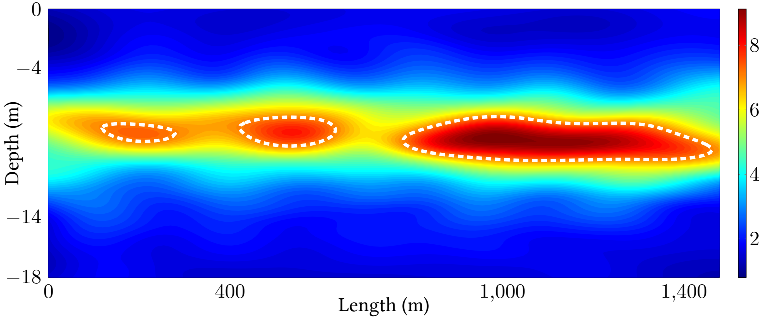


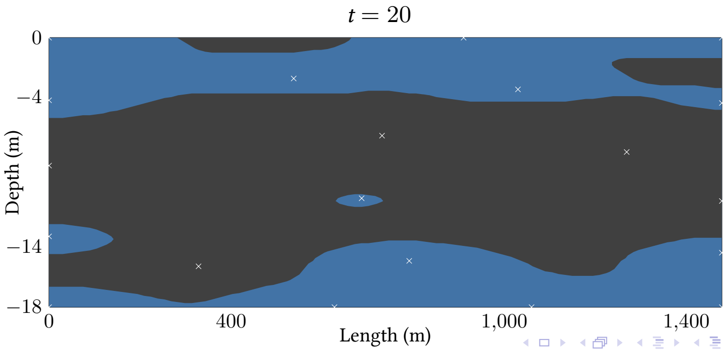
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

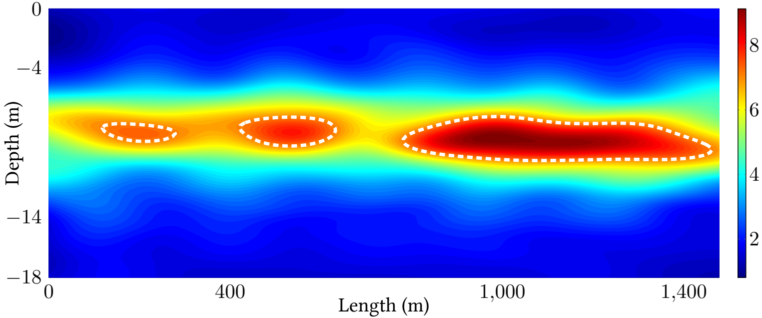

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

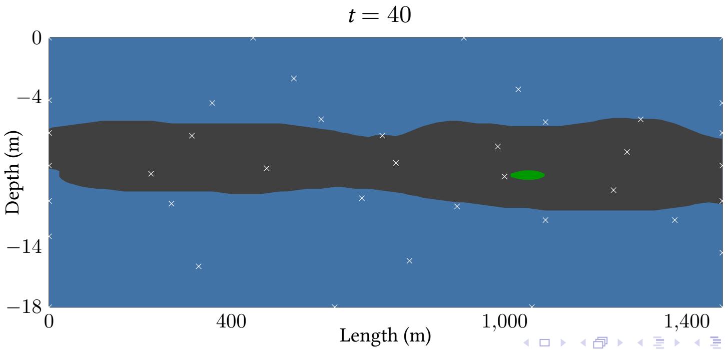
$t = 0$

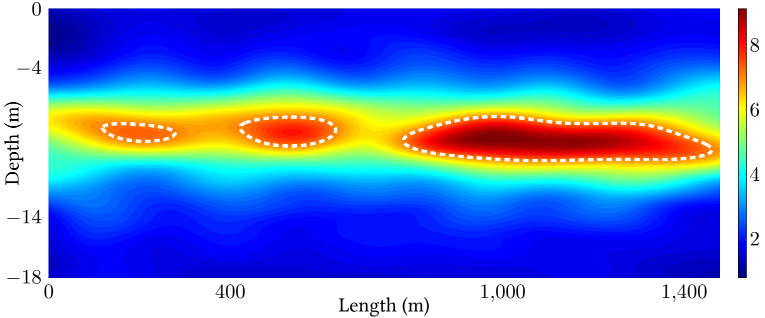

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

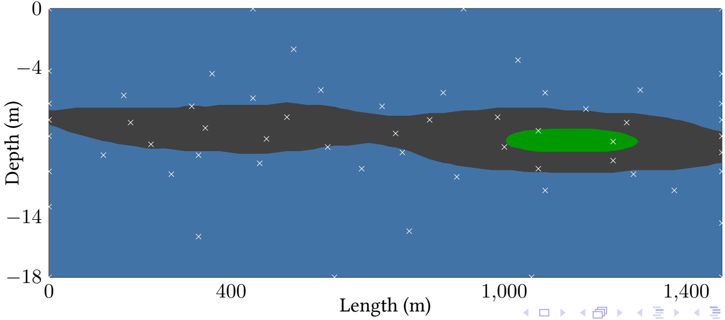

$t = 20$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

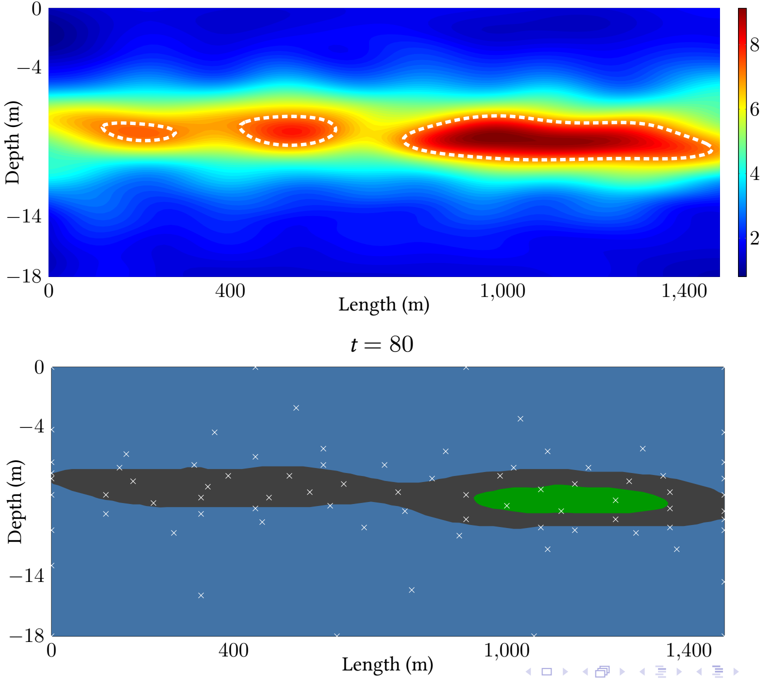

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

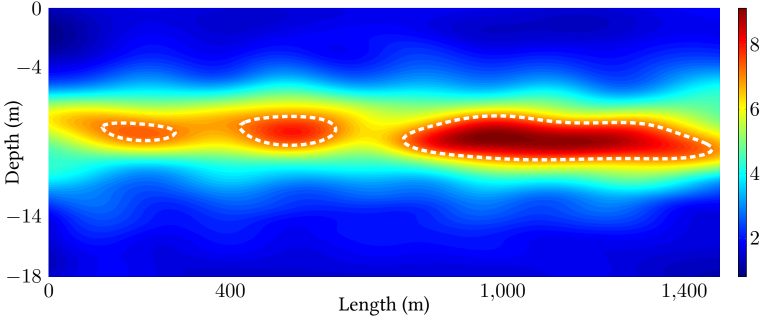

$t = 40$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

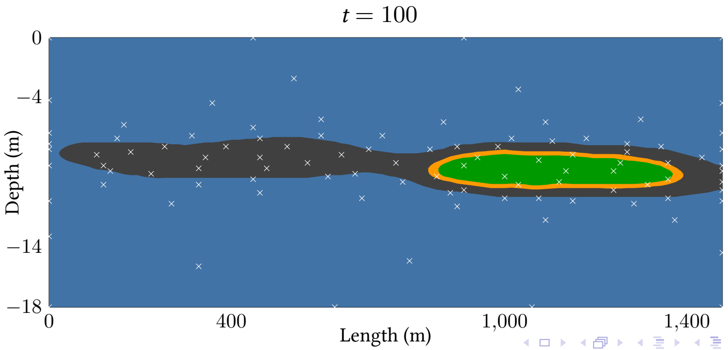
$t = 60$


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

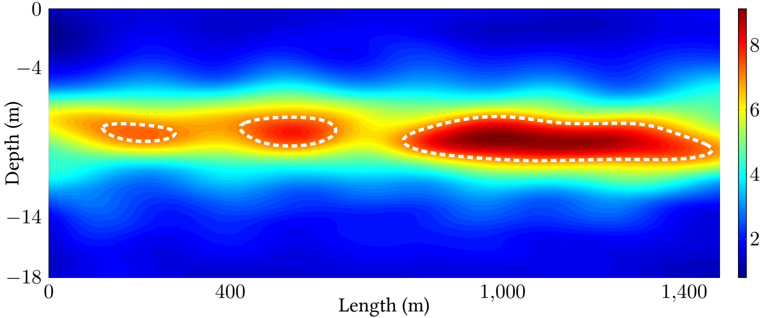

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2
$t = 80$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)



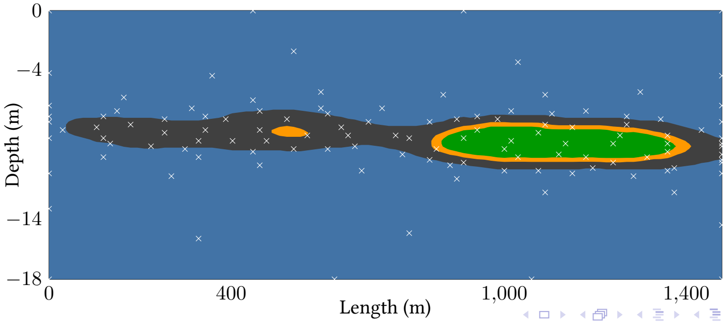
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

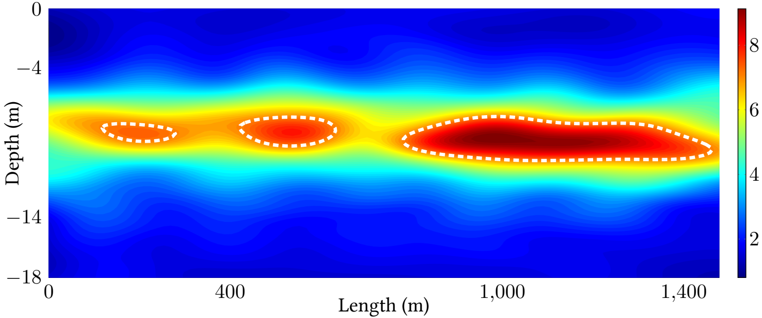


$t = 100$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

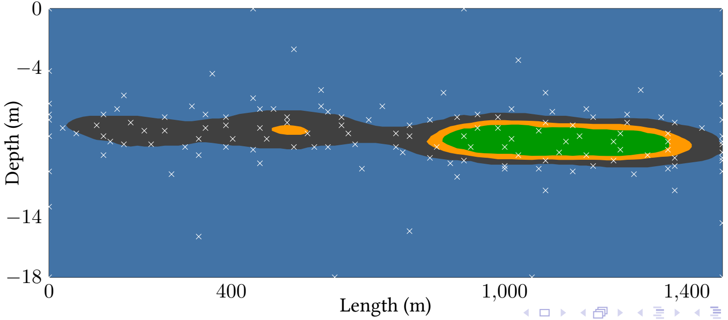
$t = 120$


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

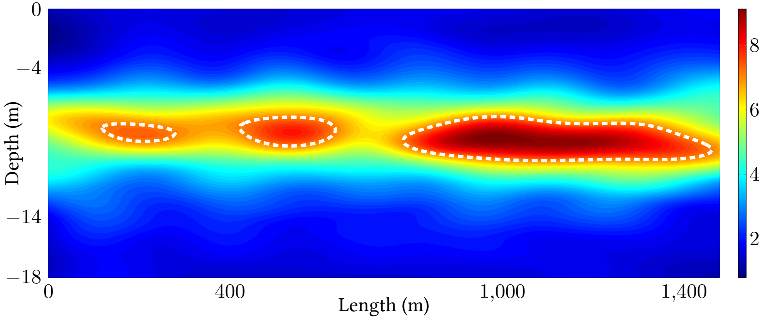


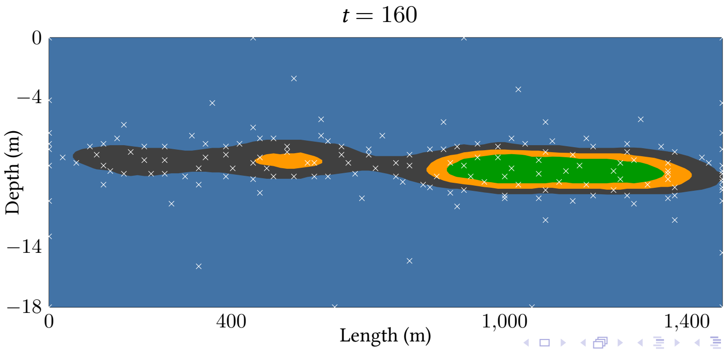
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

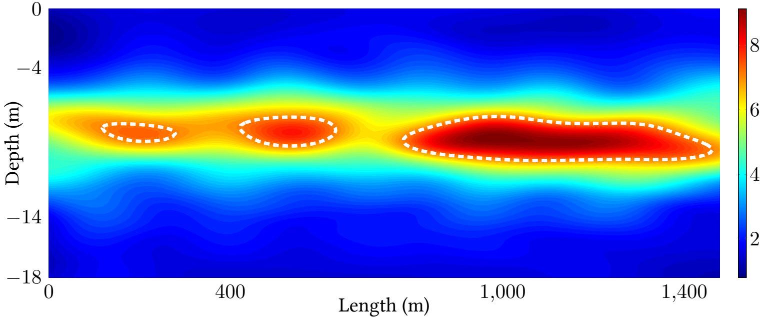
$t = 140$



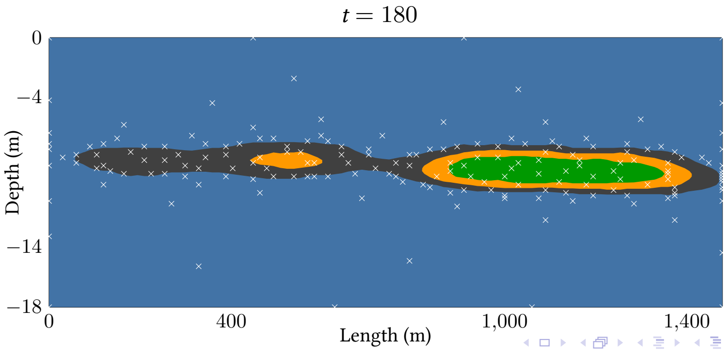
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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

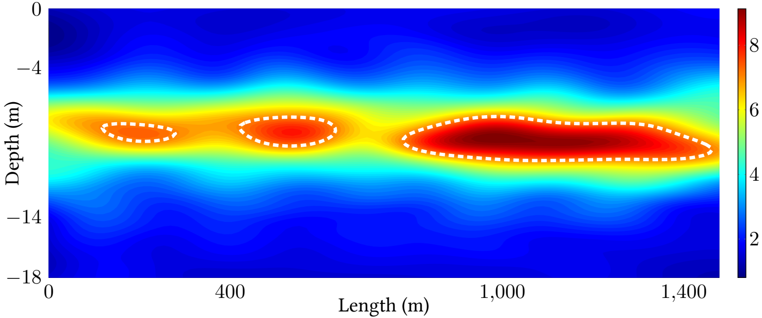

$t = 160$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

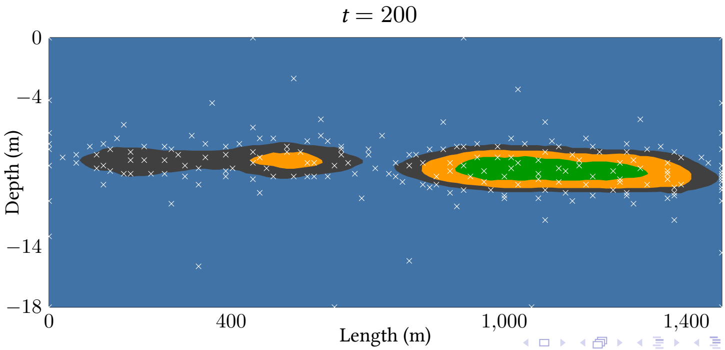


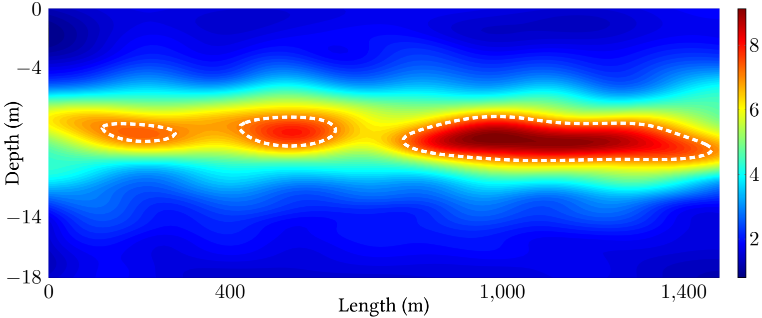
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

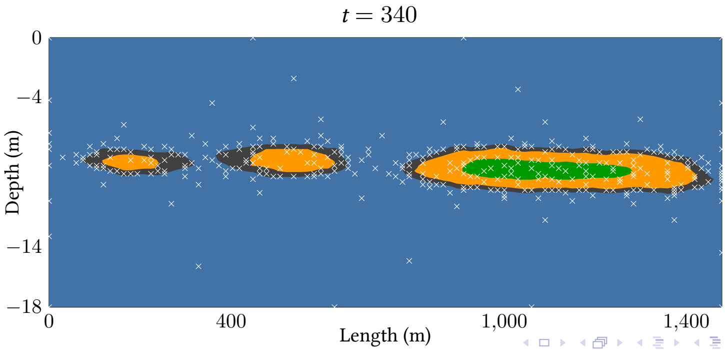


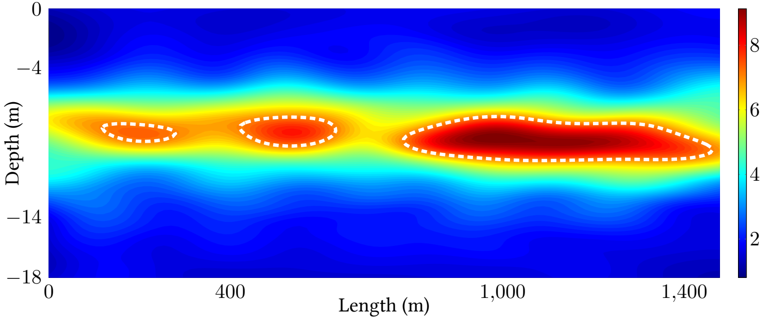
t = 180
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)

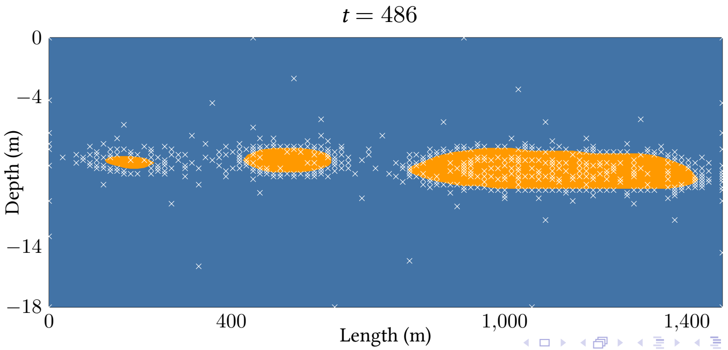

0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)
8
6
4
2

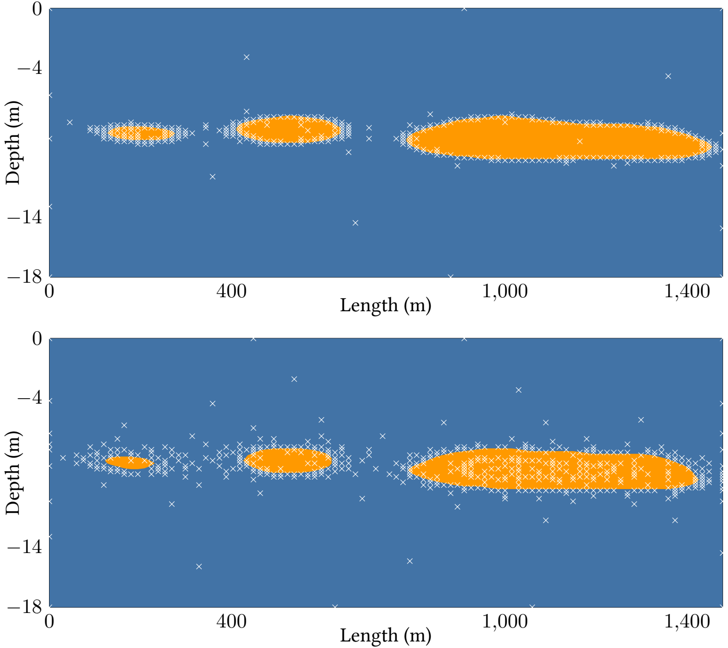

$t = 200$
0
−4
−14
−18
Depth (m)
0
400
1,000
1,400
Length (m)


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)

Batch sampling

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

Batch sampling

- 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

Batch sampling

- 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

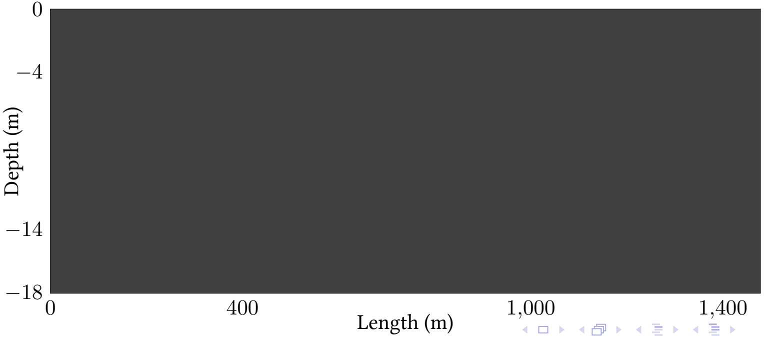
- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

Batch sampling

- 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

- Plan ahead:
  - Use $\textbf{LSE}_{\textbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

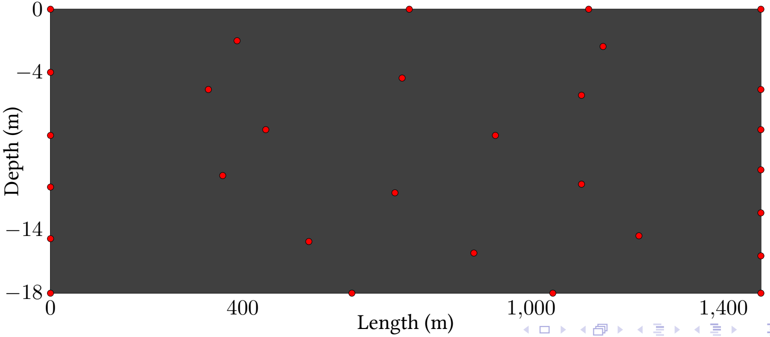


Batch sampling

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

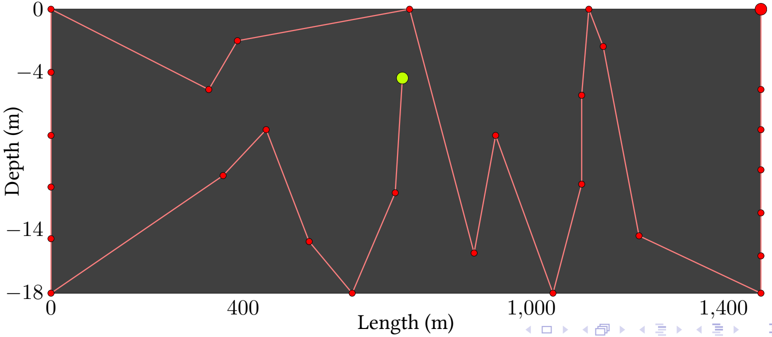
- We extend LSE to select a *batch* of sampling locations at each step

- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



Batch sampling

- 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
- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

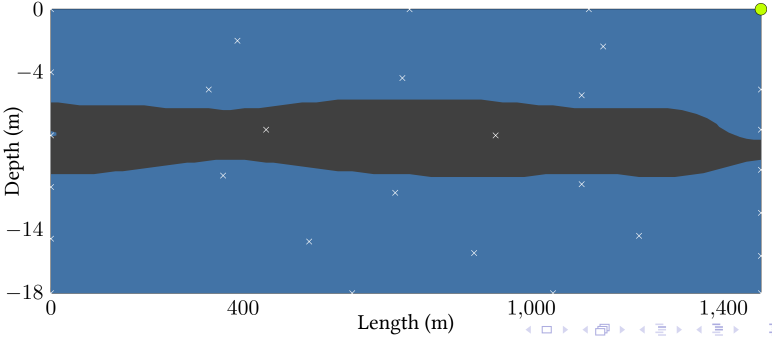


Batch sampling

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

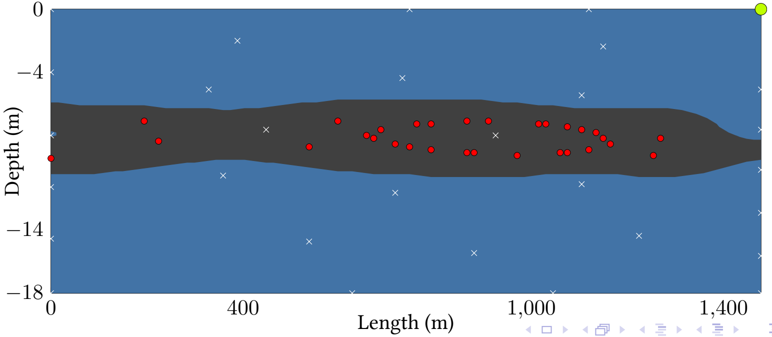
- We extend LSE to select a *batch* of sampling locations at each step

- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



Batch sampling

- 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
- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

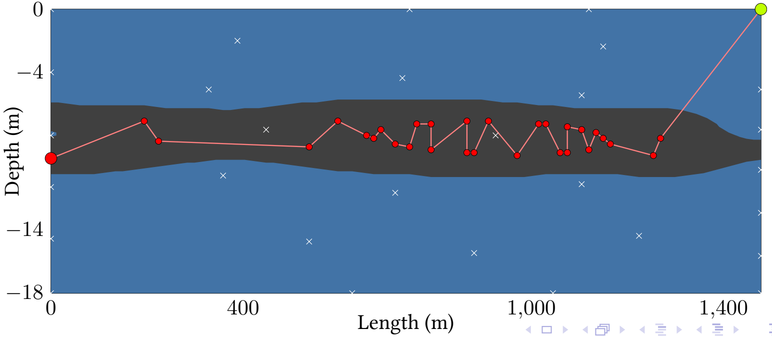


Batch sampling

- 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

- Plan ahead:
  - Use $\mathbf{LSE}_{\mathbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

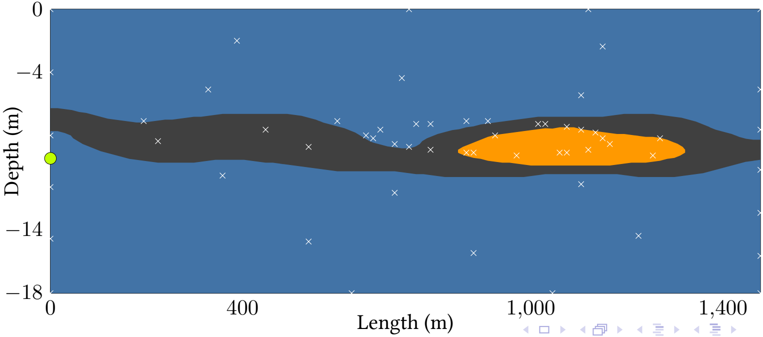


Batch sampling

- 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

- Plan ahead:
  - Use $\textbf{LSE}_{\textbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

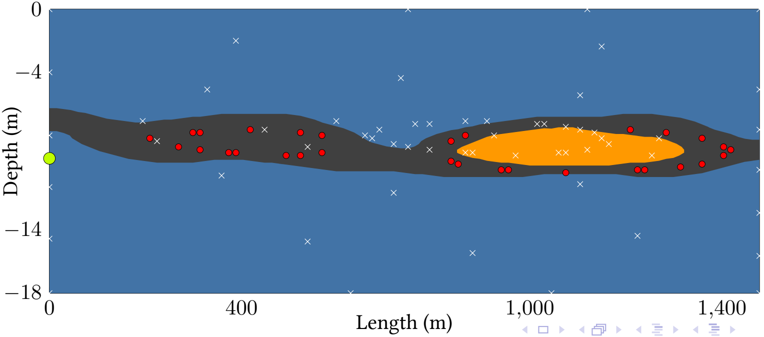


Batch sampling

- 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

- Plan ahead:
  - Use $\textbf{LSE}_{\textbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

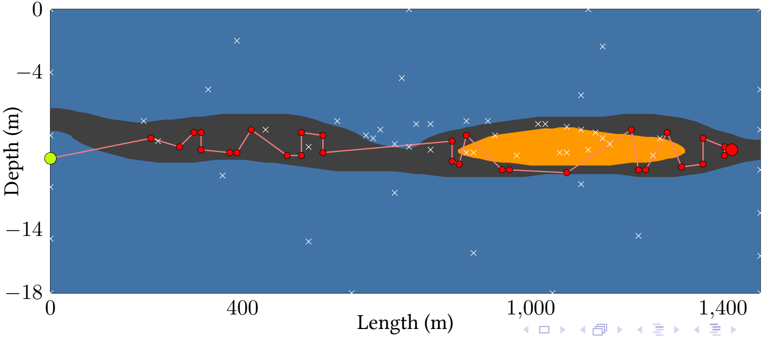


Batch sampling

- 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

- Plan ahead:
  - Use $\mathbf{LSE_{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

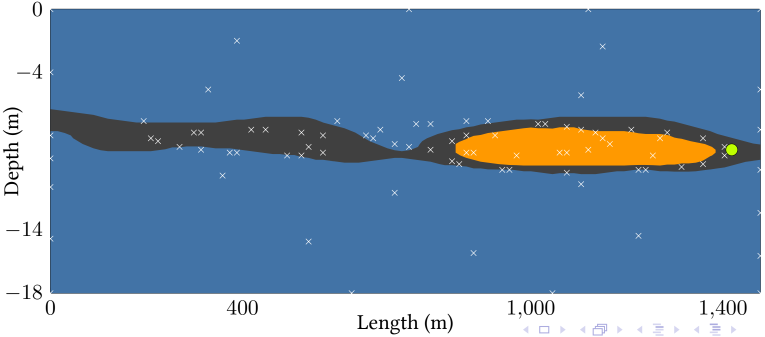


Batch sampling

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

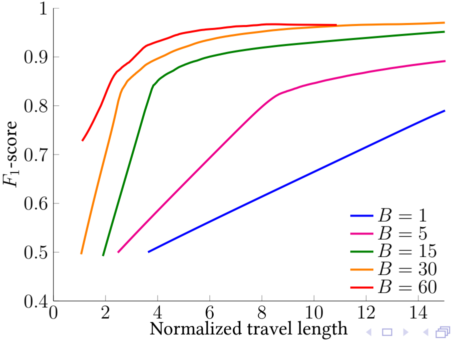
- We extend LSE to select a *batch* of sampling locations at each step

- Plan ahead:
  - Use $\textbf{LSE}_{\textbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



Batch sampling

- 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
- Plan ahead:
  - Use $\mathbf{LSE}_{\mathbf{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements



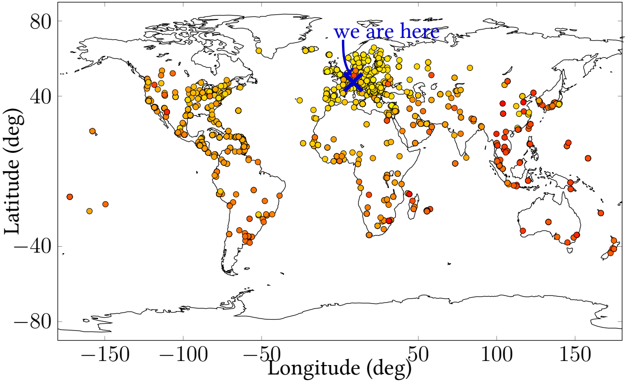
In our paper, more...

In our paper, more...

- ...theory: sample complexity bounds and their proofs in more detail

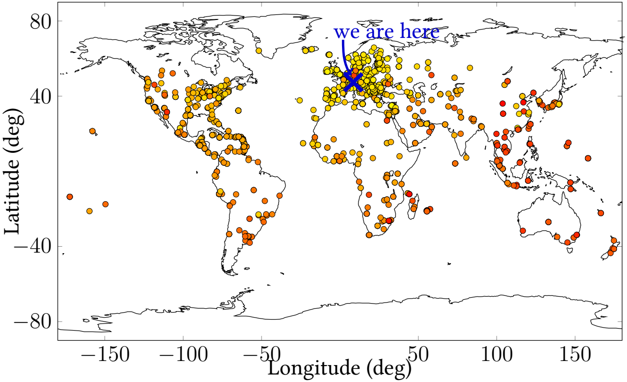
In our paper, more...

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



In our paper, more...

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



- ...experimental results

Summary

Summary

- LSE algorithm:

Theoretical guarantees

Theorem (Convergence of LSE)

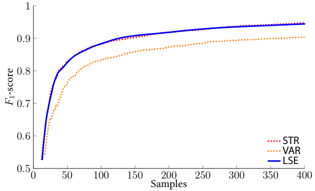
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

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

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



Summary

- LSE algorithm:

Theoretical guarantees

Theorem (Convergence of LSE)

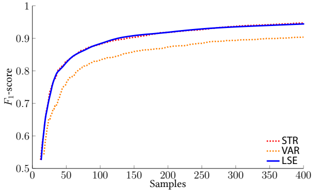
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

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

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

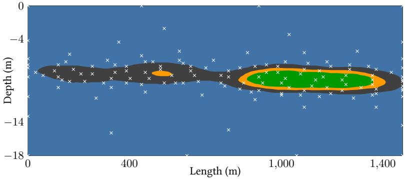
$$\Pr\left\{\max_{x \in D} \ell_h(x) \leq \epsilon\right\} \geq 1 - \delta.$$

Competitive with the state of the art

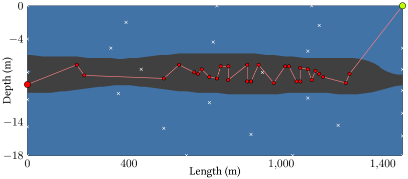


- Two useful extensions:

Implicit threshold level (LSE_{imp})



Batch sampling ($\text{LSE}_{\text{batch}}$)



Summary

- LSE algorithm:

Theoretical guarantees

Theorem (Convergence of LSE)

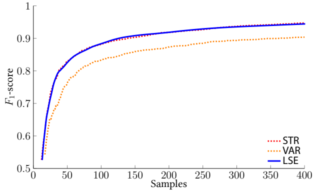
For any $h \in \mathbb{R}$, $\delta \in (0,1)$, and $\epsilon > 0$, if $\beta_t = 2\log(|D|\pi^2 t^2/(6\delta))$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

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

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

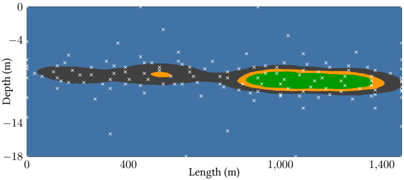
$$\Pr\left\{ \max_{x \in D} \ell_h(x) \leq \epsilon \right\} \geq 1 - \delta.$$

Competitive with the state of the art

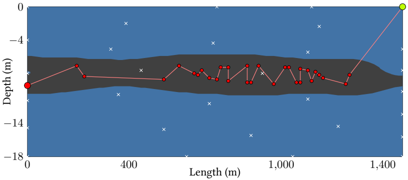


- Two useful extensions:

Implicit threshold level (LSE_{imp})



Batch sampling ($\text{LSE}_{\text{batch}}$)



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