

Active Learning for Level Set Estimation

Alkis Gotovos¹ Nathalie Casati^{1,2} Gregory Hitz¹ Andreas Krause¹

¹Department of Computer Science
ETH Zurich

²IBM Research – Zurich

IJCAI 2013

Outline

- Motivation and background
- The LSE algorithm
- Two extensions

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*

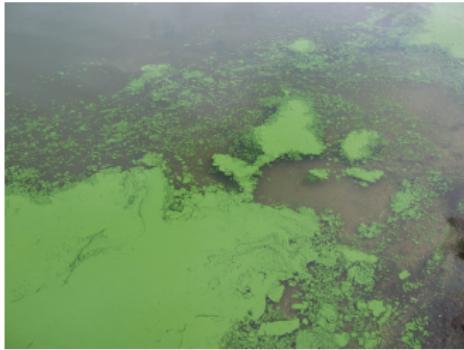
Swimmers of Lake Zurich, beware!



[www.limnobiotics.ch](http://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

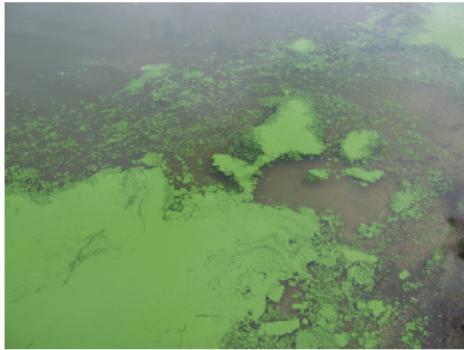
Swimmers of Lake Zurich, beware!



www.limnobotics.ch

"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

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

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

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

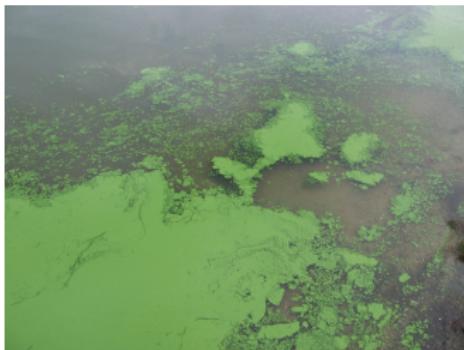
Swimmers of Lake Zurich, beware!



[www.limnobiotics.ch](http://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

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

"*Planktothrix 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."

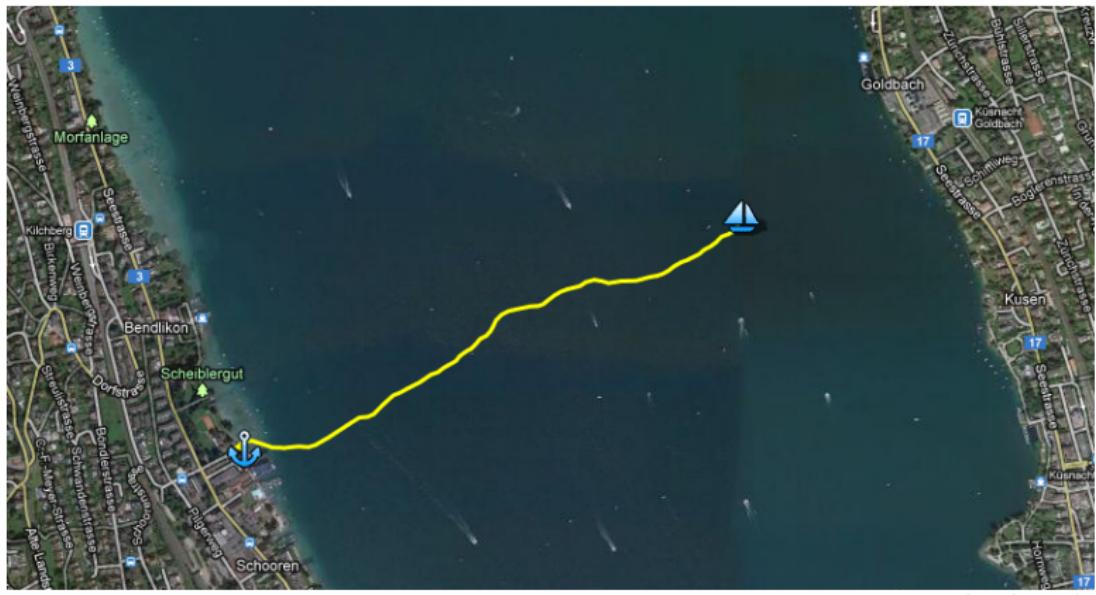
— *Wikipedia*



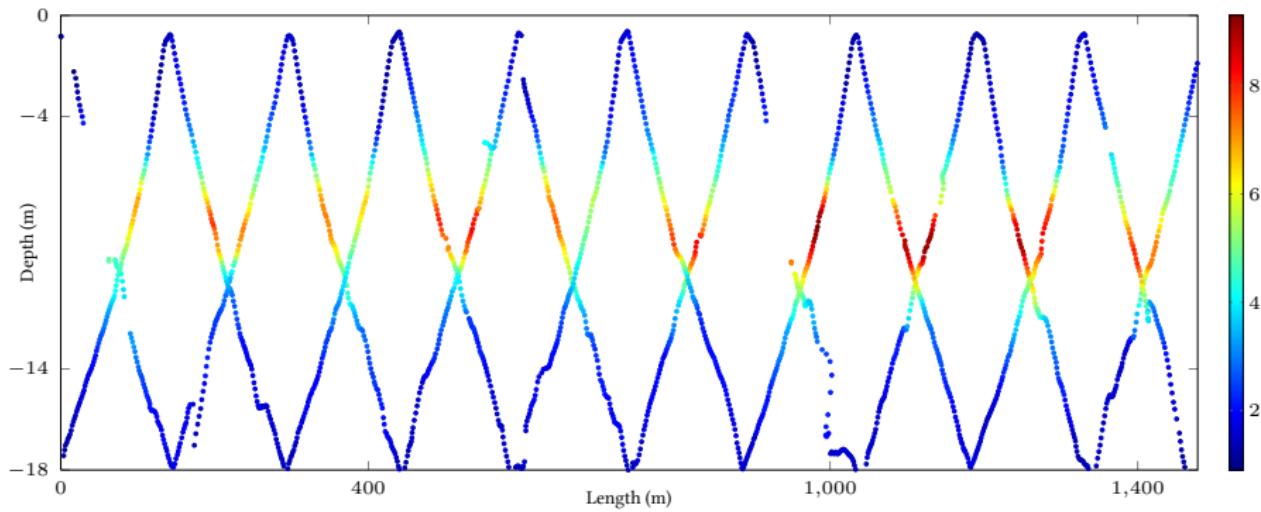
www.limnobotics.ch

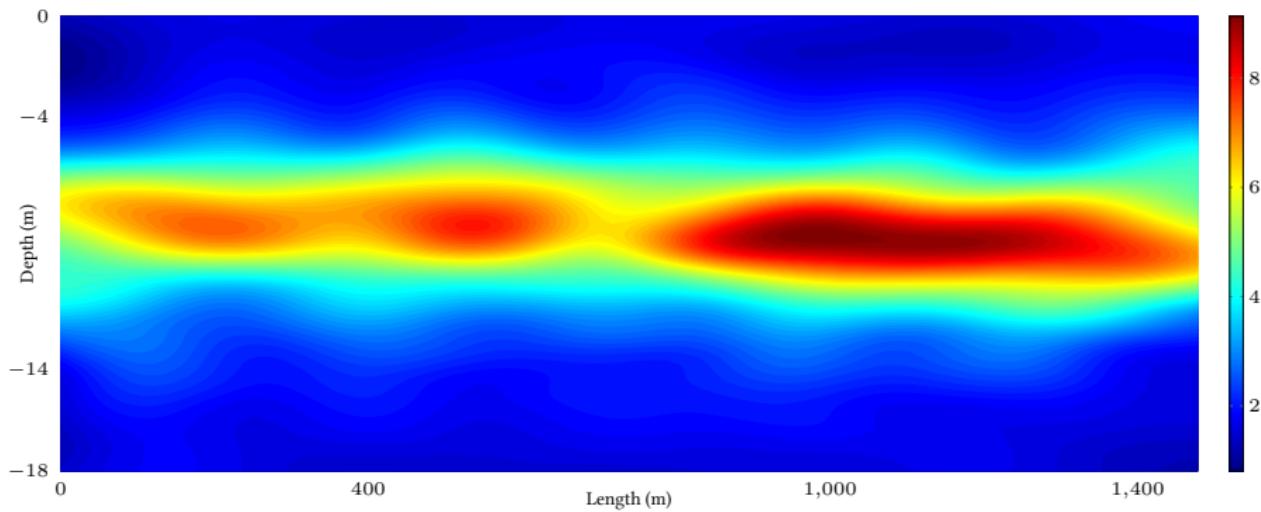


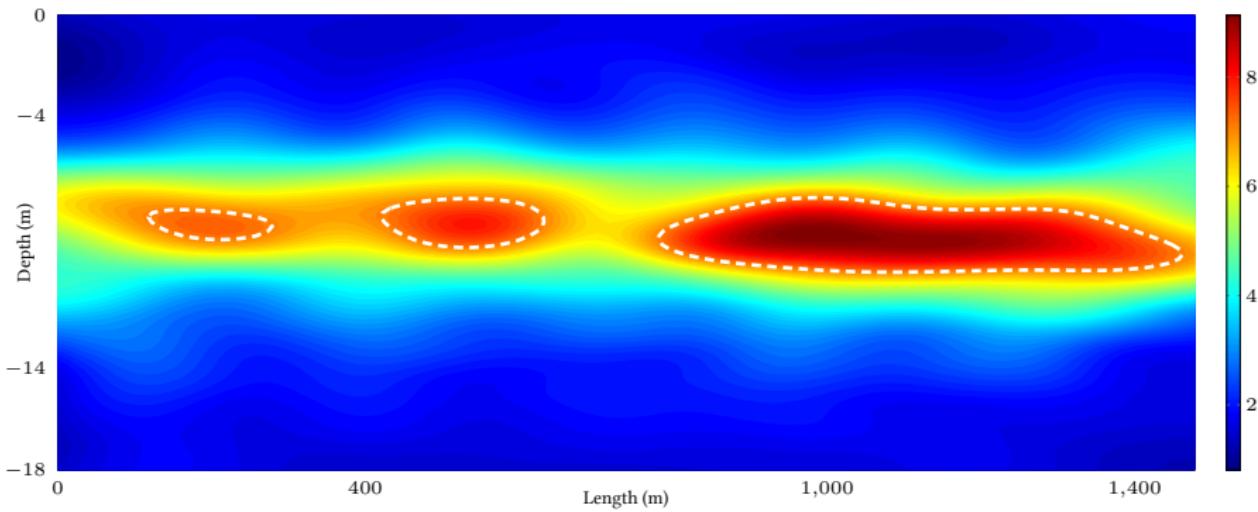
www.limnobotics.ch

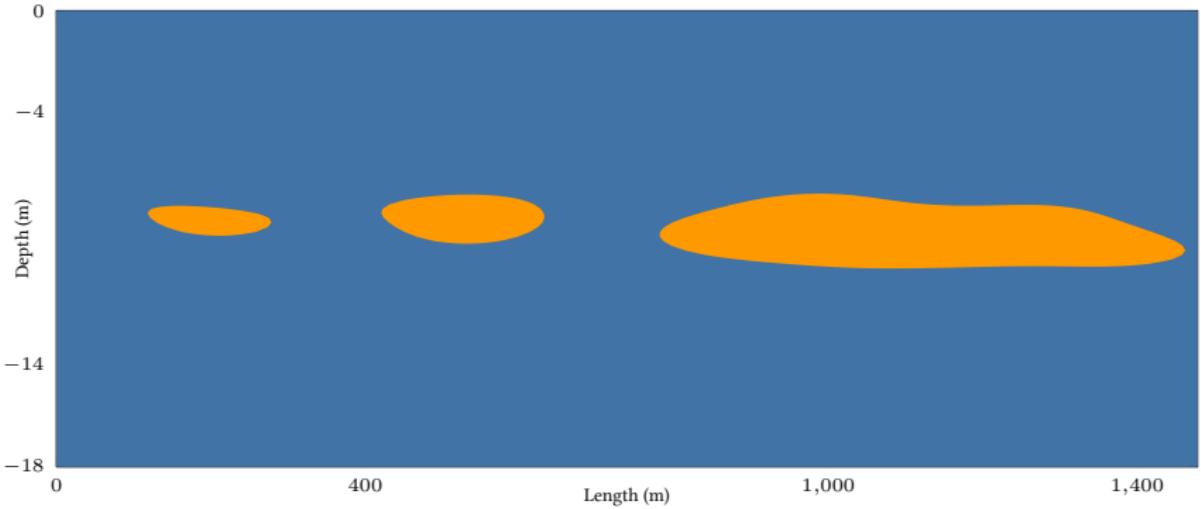


www.limnobotics.ch



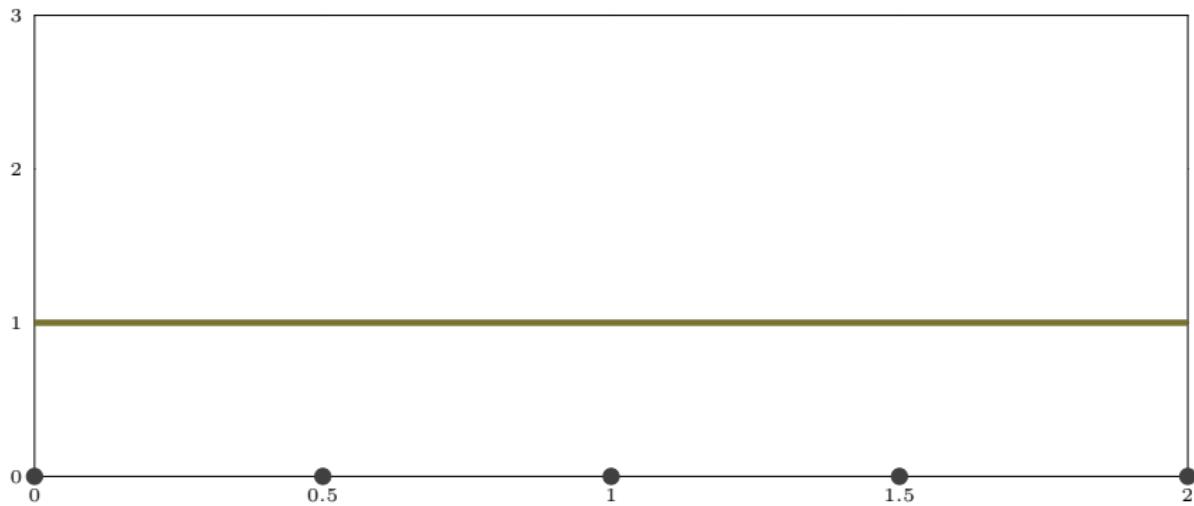






- ▶ Pose as a sequential decision making problem (*pool-based active learning*):

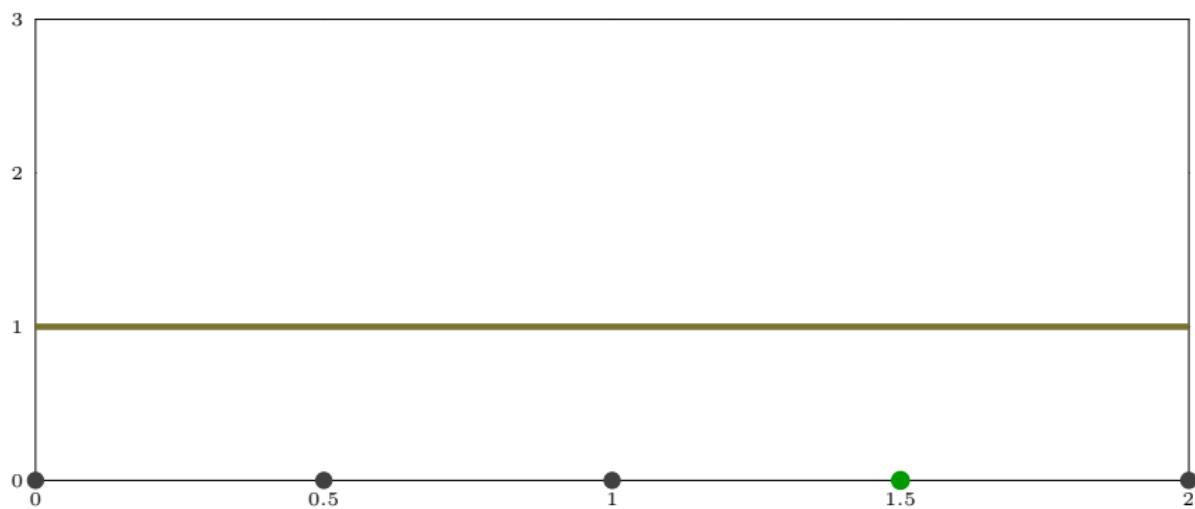
- ▶ Pose as a sequential decision making problem (*pool-based active learning*):
 - ▶ No measurements available in advance, just a set (pool) of possible sampling locations (D)



- ▶ 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 $t \geq 1$:

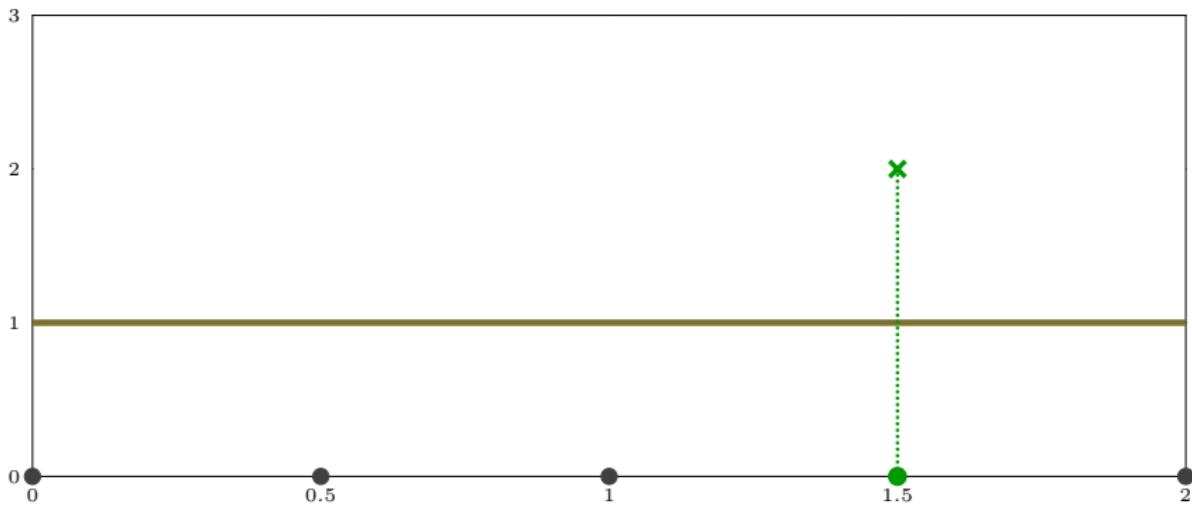
- ▶ Decide where to measure next (x_t)



- ▶ 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 $t \geq 1$:

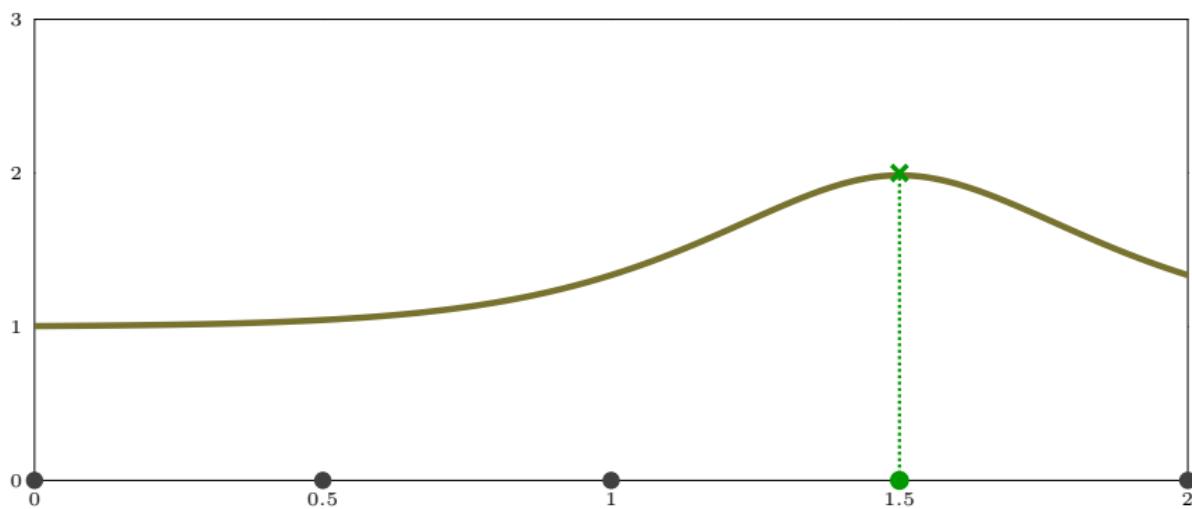
- ▶ Decide where to measure next (x_t)
- ▶ Obtain noisy observation ($y_t = f(x_t) + n_t$)



- ▶ 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 $t \geq 1$:

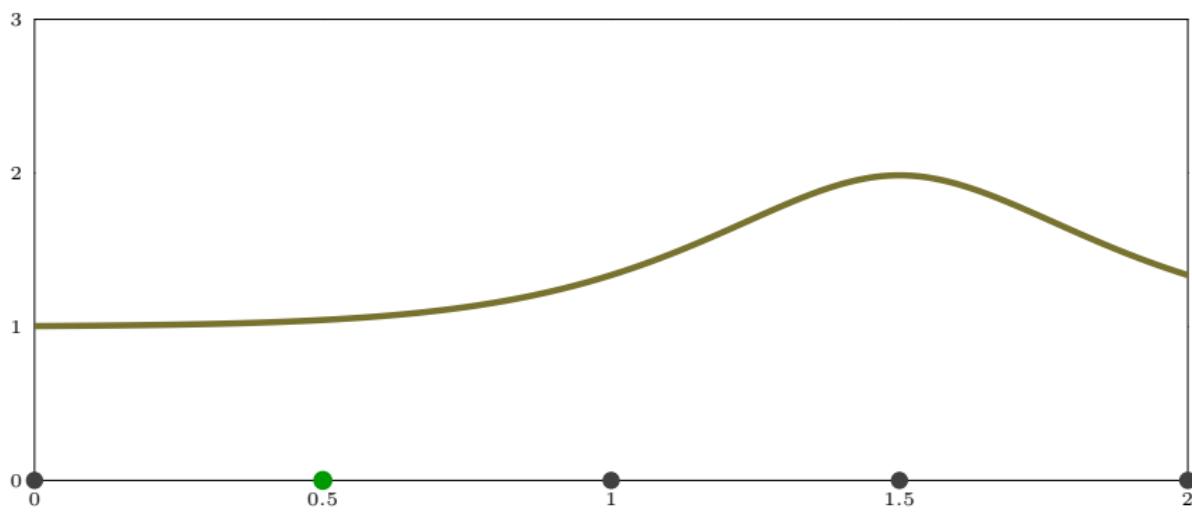
- ▶ Decide where to measure next (x_t)
- ▶ Obtain noisy observation ($y_t = f(x_t) + n_t$)
- ▶ Update our estimate of the underlying function



- ▶ 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 $t \geq 1$:

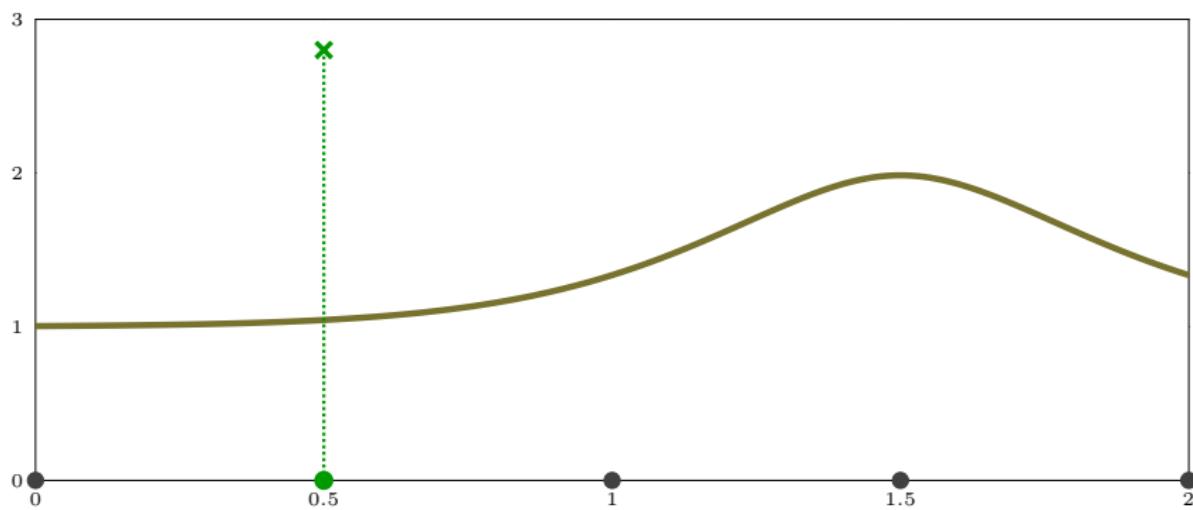
- ▶ Decide where to measure next (x_t)
- ▶ Obtain noisy observation ($y_t = f(x_t) + n_t$)
- ▶ Update our estimate of the underlying function



- ▶ 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 $t \geq 1$:

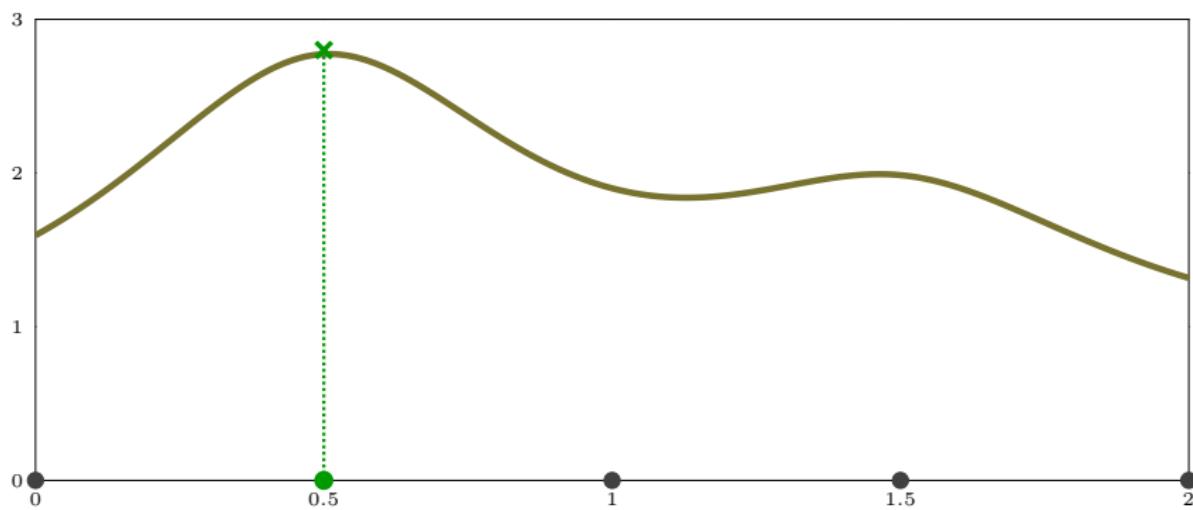
- ▶ Decide where to measure next (x_t)
- ▶ Obtain noisy observation ($y_t = f(x_t) + n_t$)
- ▶ Update our estimate of the underlying function



- ▶ 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 (x_t)
- ▶ Obtain noisy observation ($y_t = f(x_t) + n_t$)
- ▶ Update our estimate of the underlying 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**?
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!

Why GPs are nice

Why GPs are nice

- ▶ **Nonparametric:**

Impose “smoothness” assumptions via kernel $k(x, x')$.

Why GPs are nice

- ▶ **Nonparametric:**

Impose “smoothness” assumptions via kernel $k(x, x')$.

- ▶ **Bayesian, yet efficient:**

$$\mu_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_t$$

$$k_t(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_t(\mathbf{x})$$

$$\sigma_t^2(\mathbf{x}) = k_t(\mathbf{x}, \mathbf{x})$$

Suitable for step-by-step updates.

Why GPs are nice

- ▶ **Nonparametric:**

Impose “smoothness” assumptions via kernel $k(x, x')$.

- ▶ **Bayesian, yet efficient:**

$$\mu_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_t$$

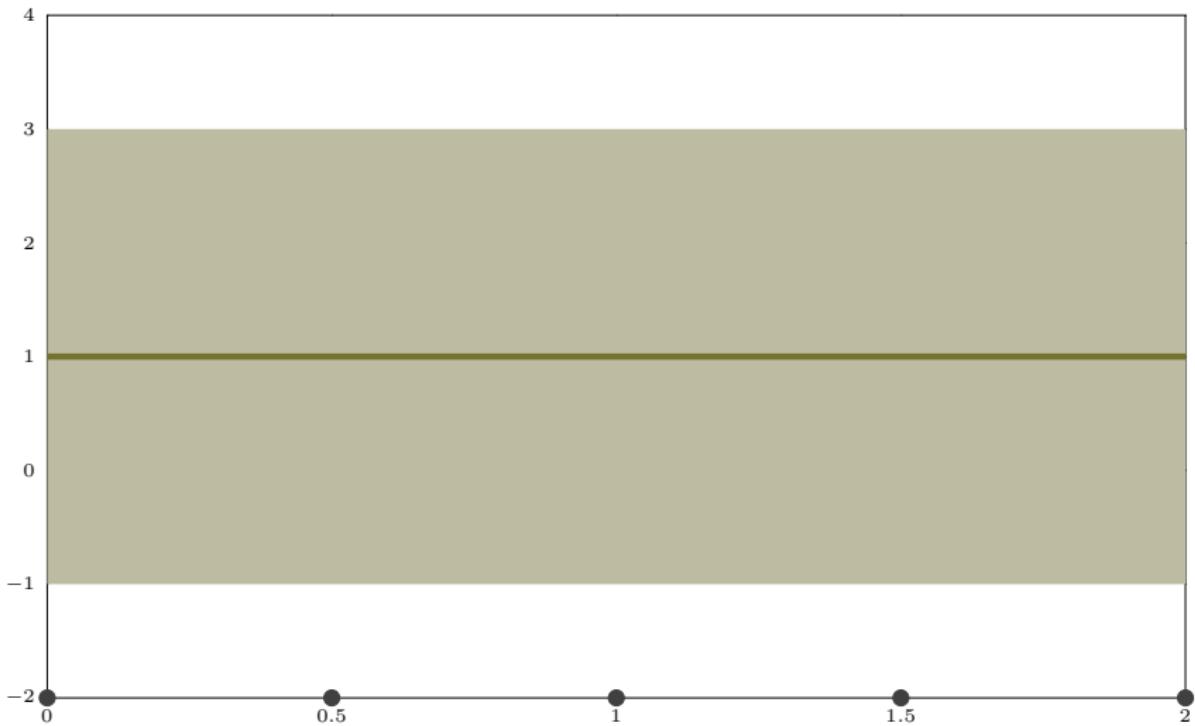
$$k_t(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^T (\mathbf{K}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_t(\mathbf{x})$$

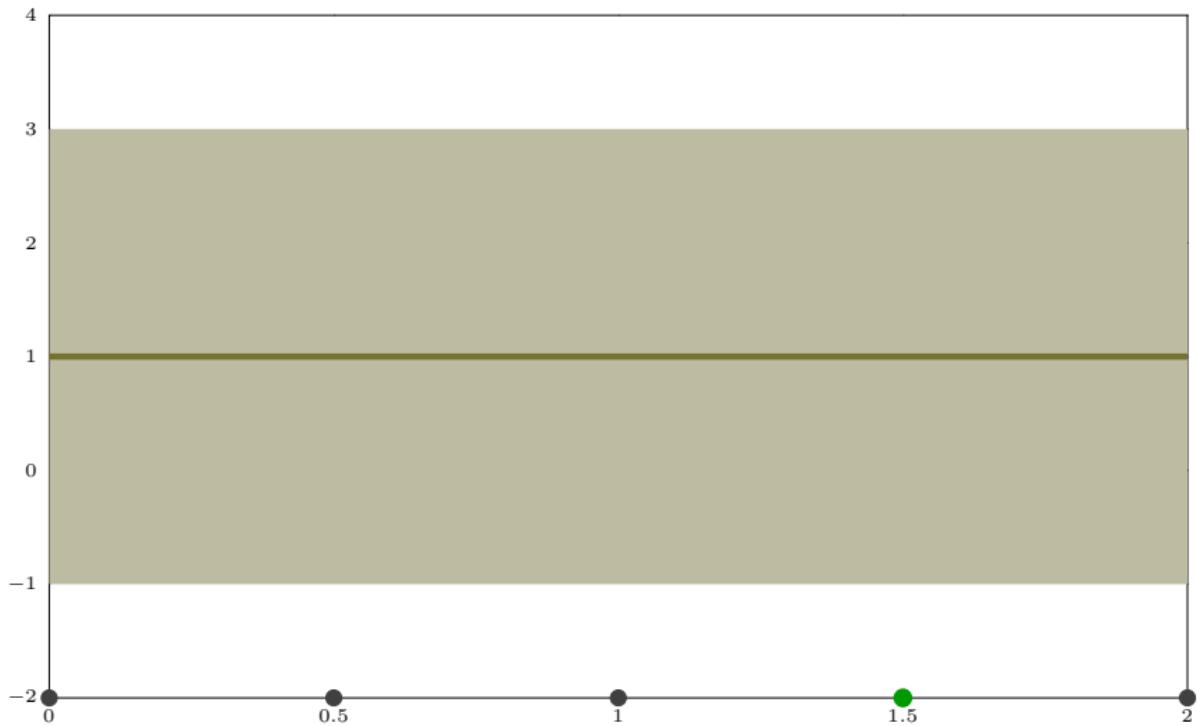
$$\sigma_t^2(\mathbf{x}) = k_t(\mathbf{x}, \mathbf{x})$$

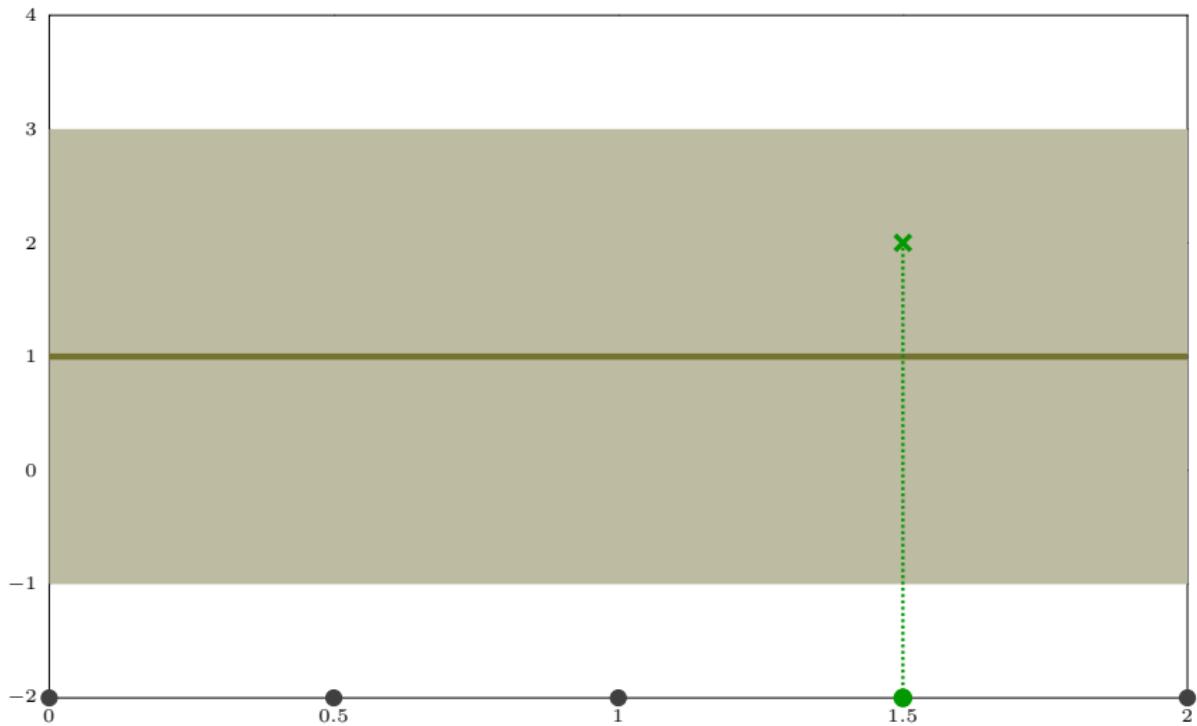
Suitable for step-by-step updates.

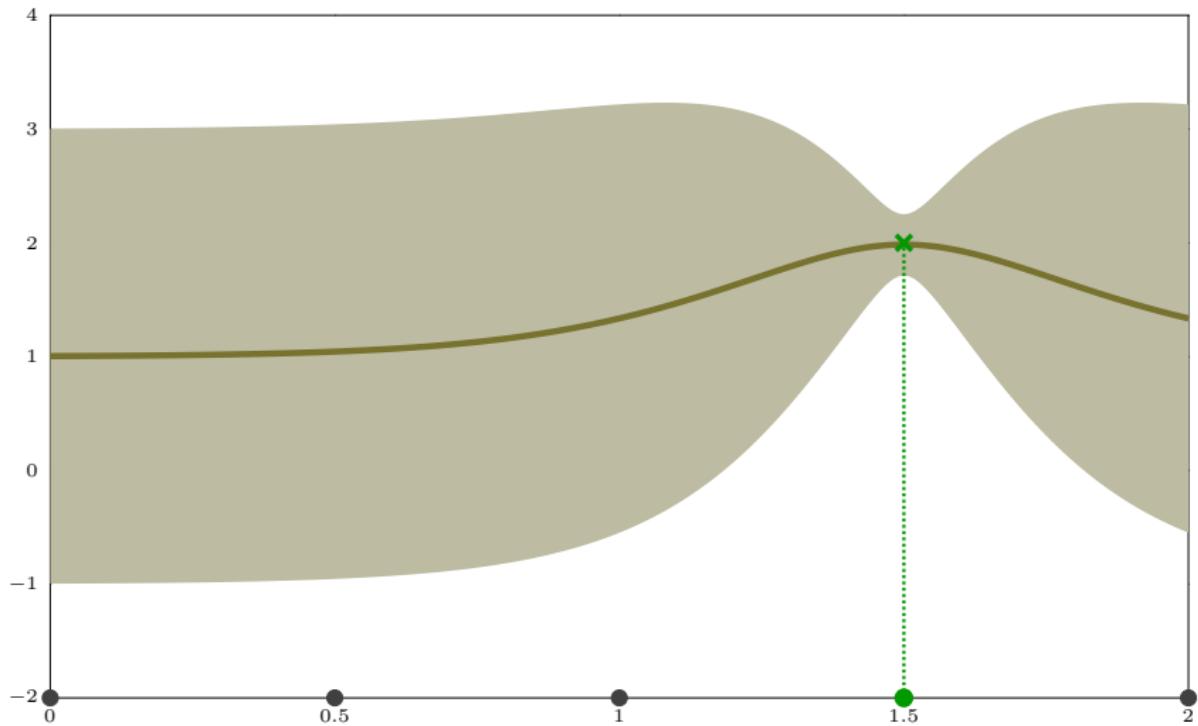
- ▶ They provide **variance estimates!**

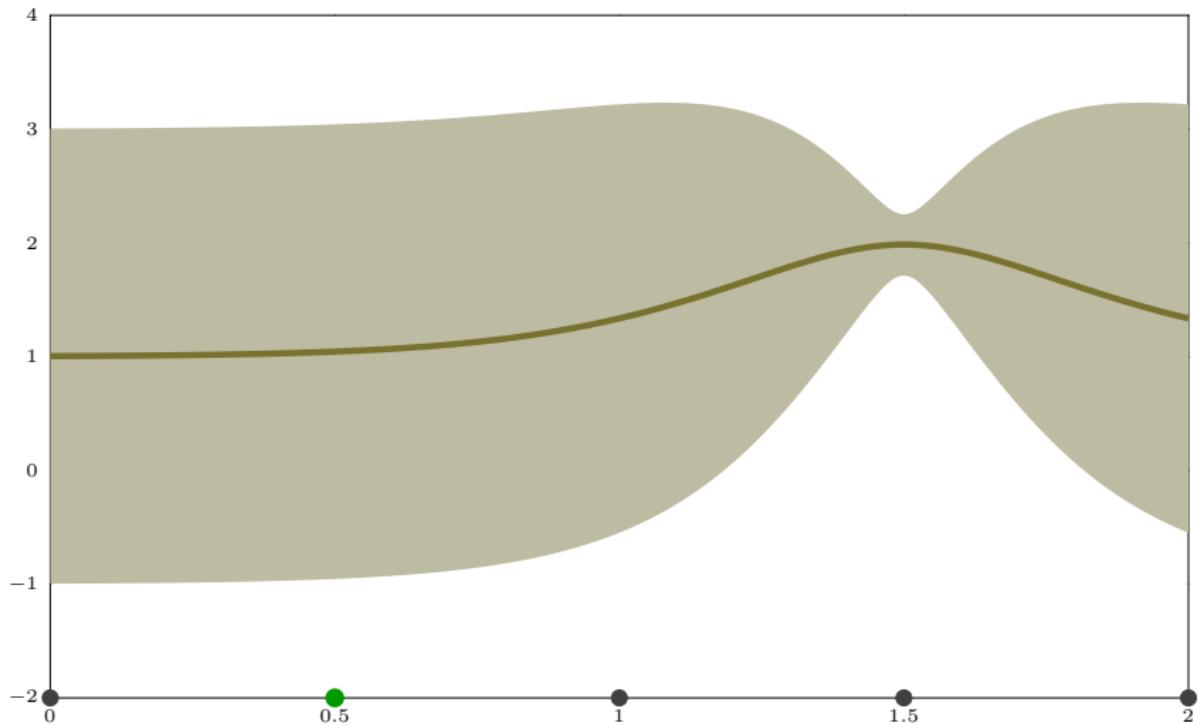
Construct confidence intervals: $Q_t(\mathbf{x}) = [\mu_{t-1}(\mathbf{x}) \pm \beta_t^{1/2} \sigma_{t-1}(\mathbf{x})]$

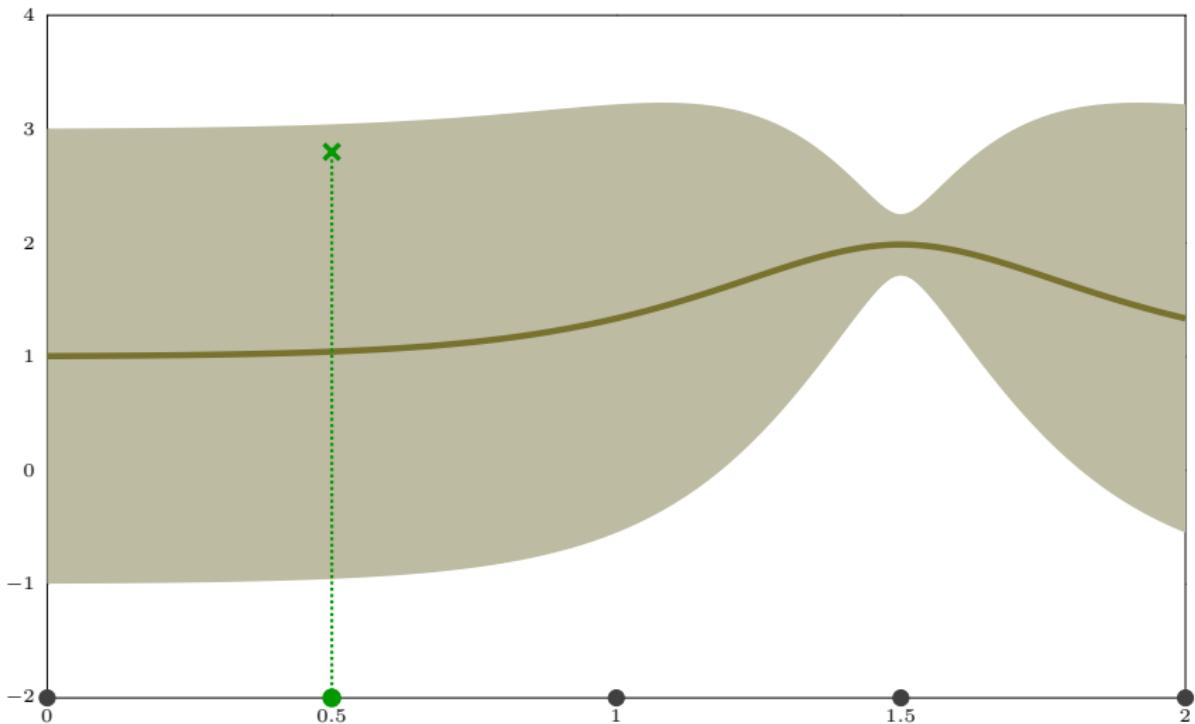


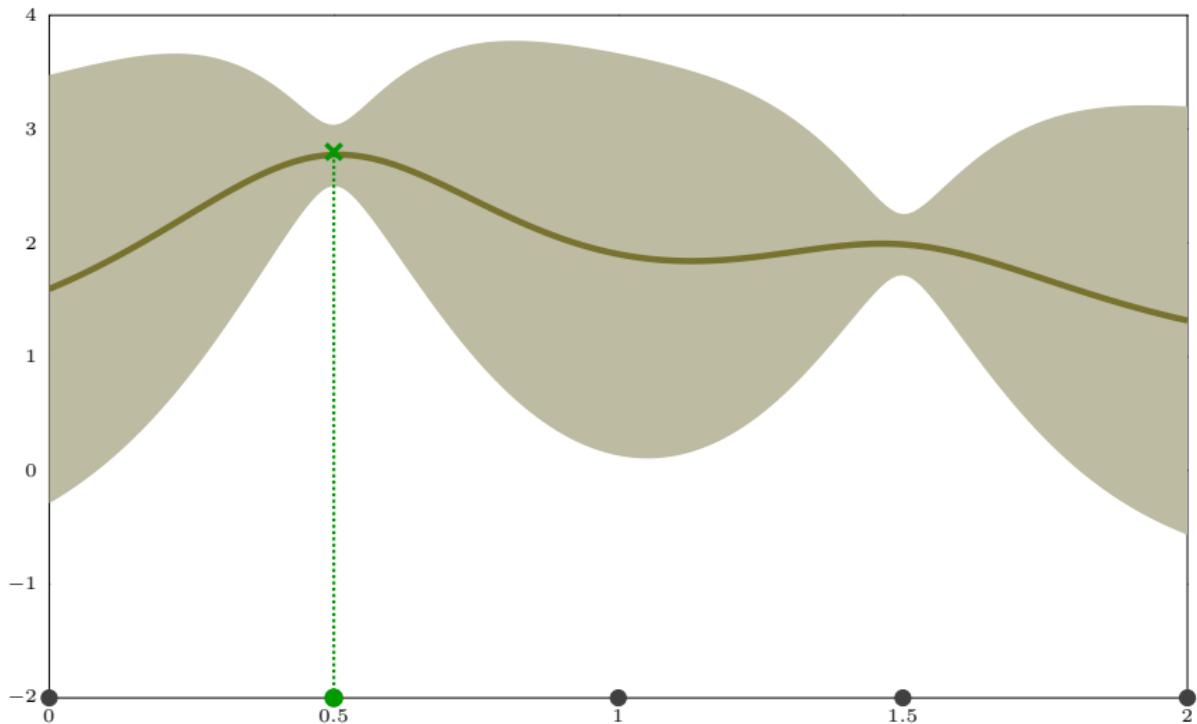


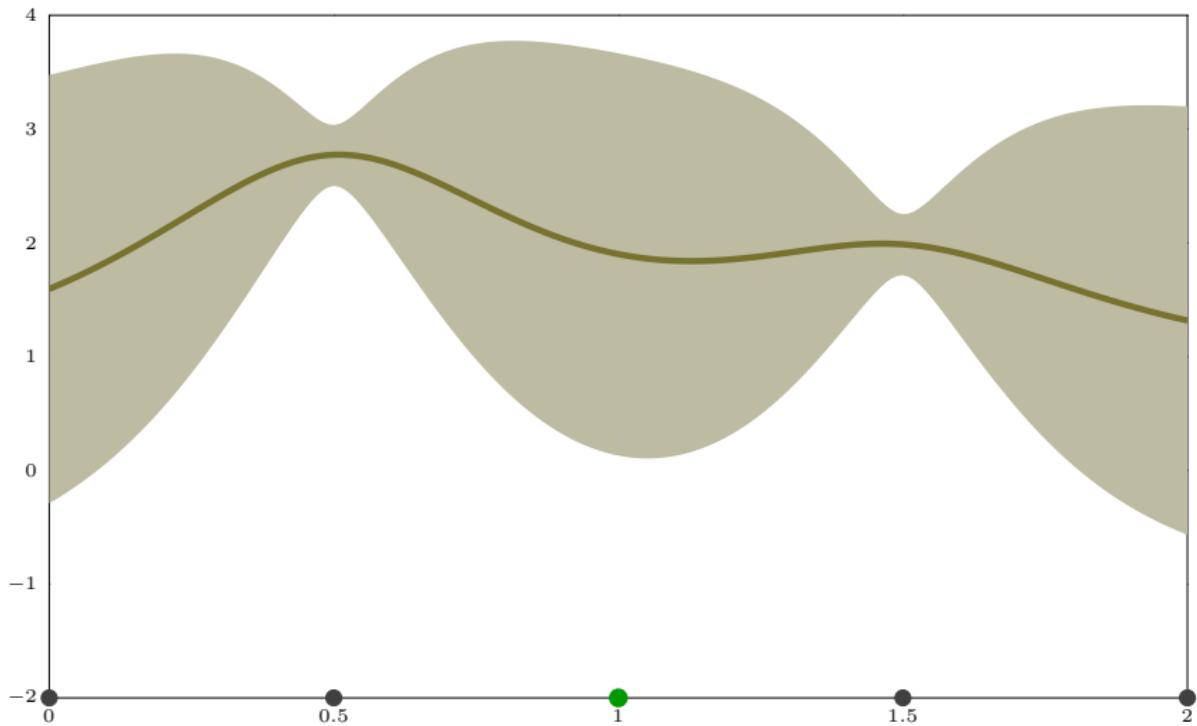


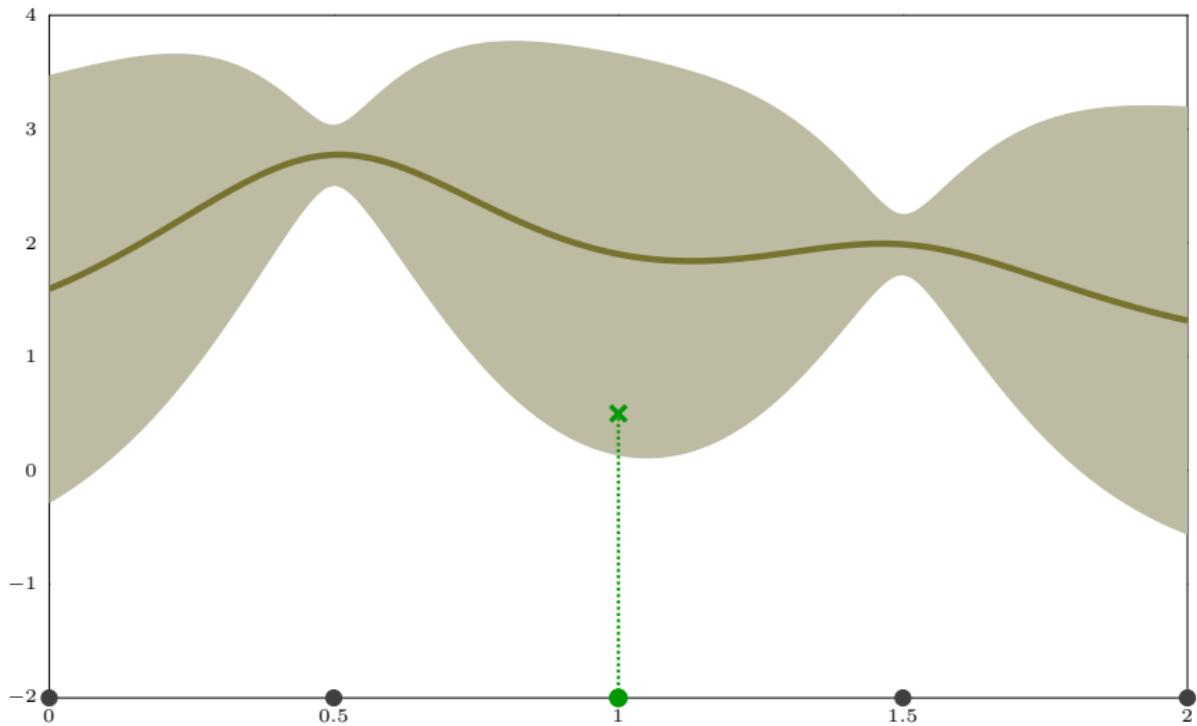


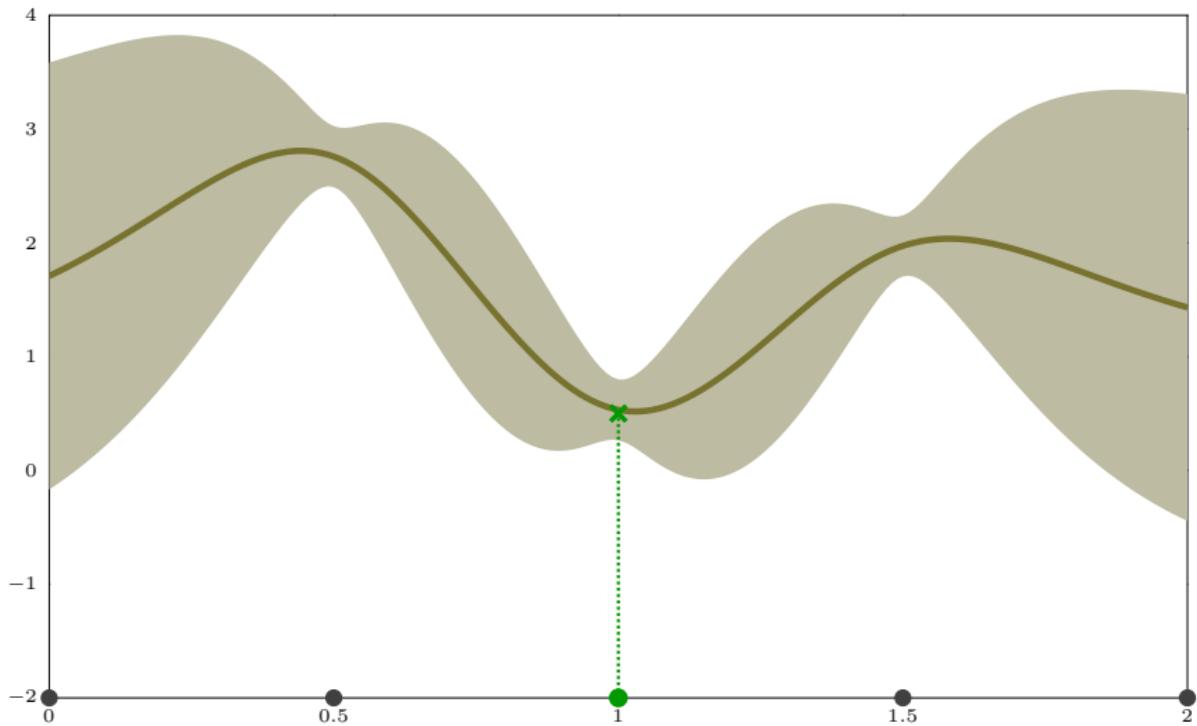






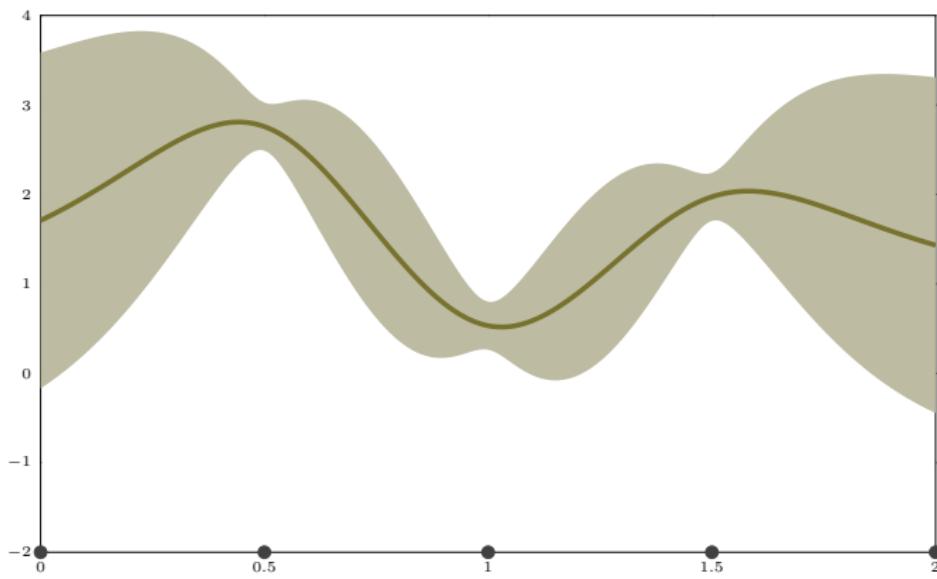




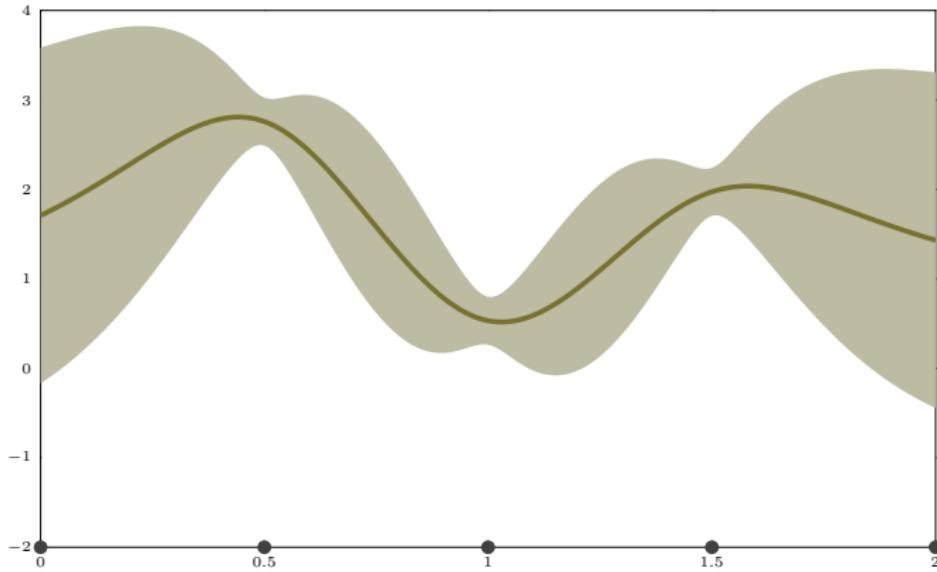


1. How do we **estimate** the function?

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

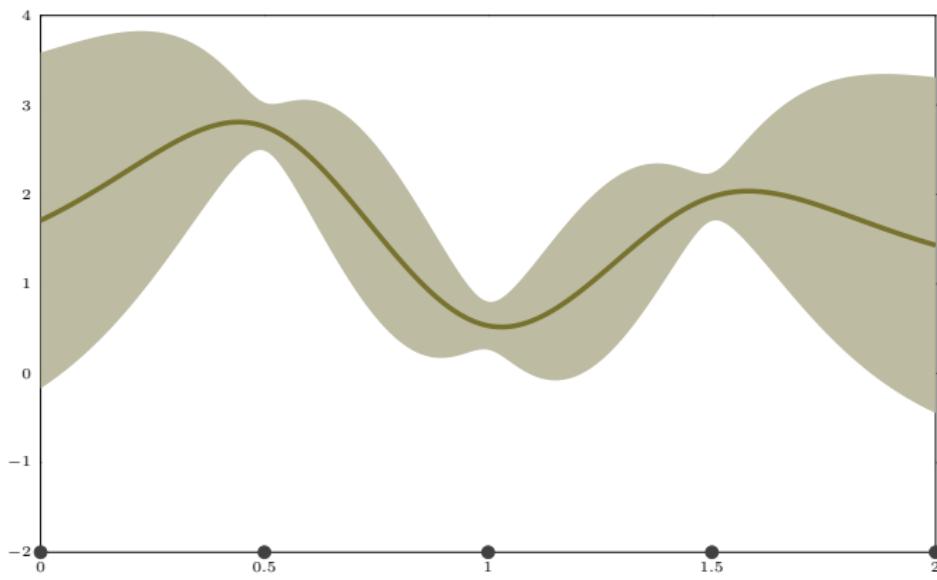


1. How do we **estimate** the function? ✓



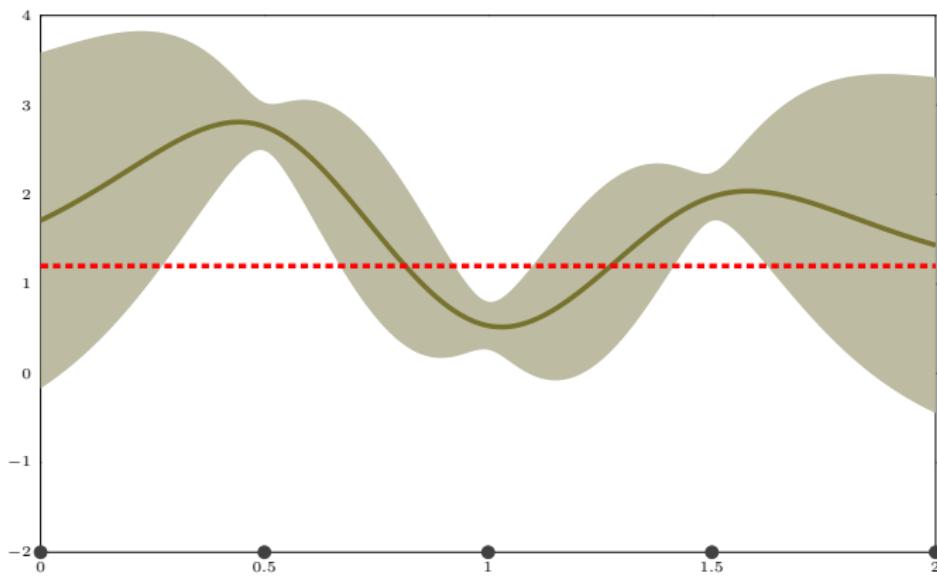
1. How do we **estimate** the function? ✓

2. How do we **classify**?



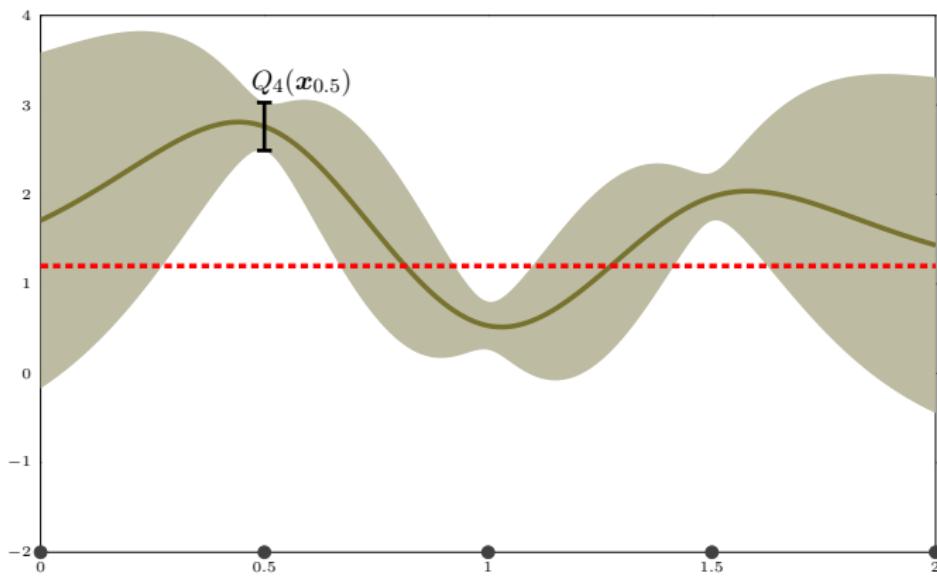
1. How do we **estimate** the function? ✓

2. How do we **classify**?



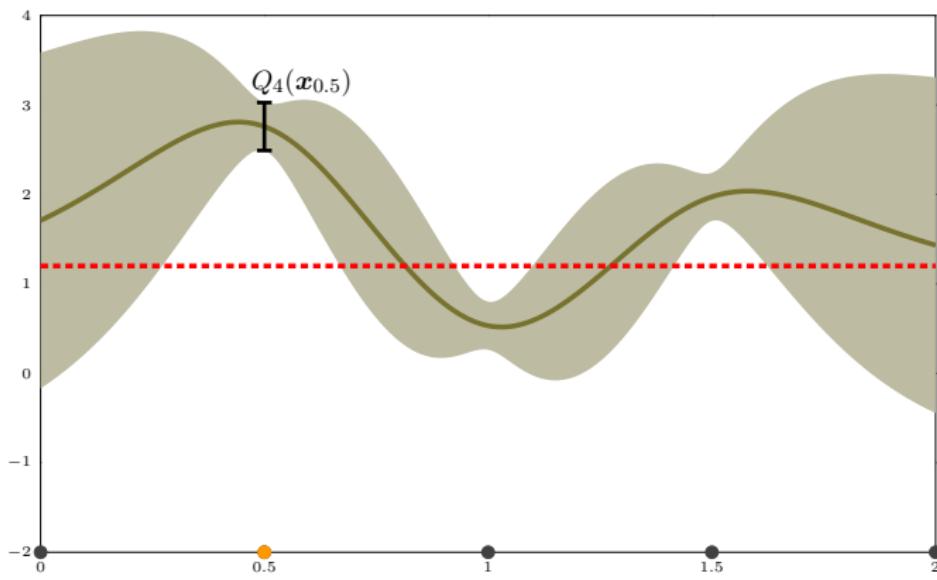
1. How do we **estimate** the function? ✓

2. How do we **classify**?



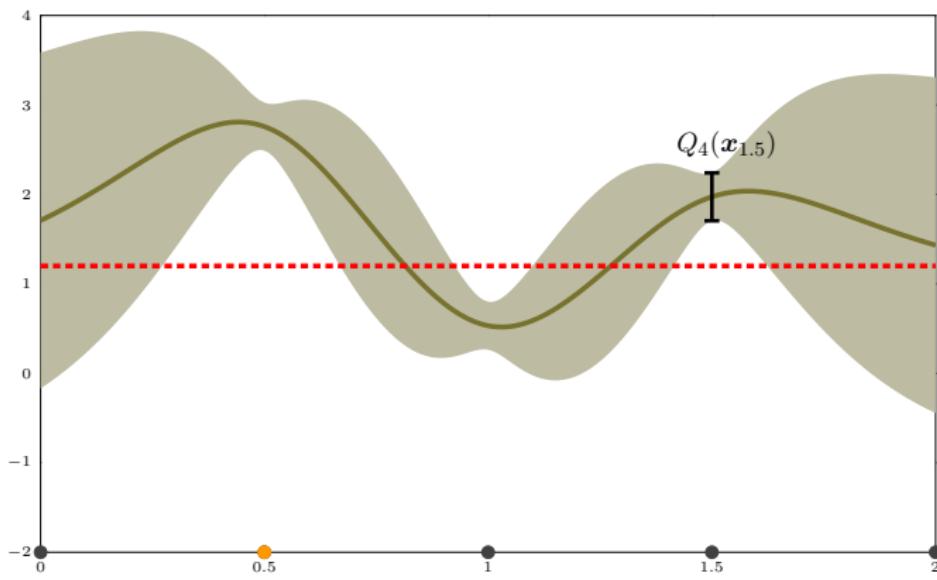
1. How do we **estimate** the function? ✓

2. How do we **classify**?



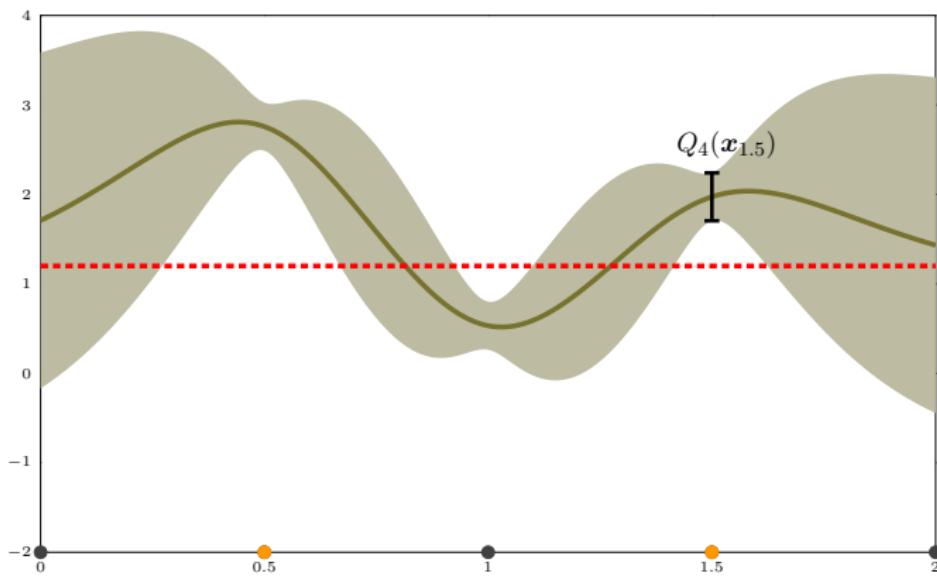
1. How do we **estimate** the function? ✓

2. How do we **classify**?



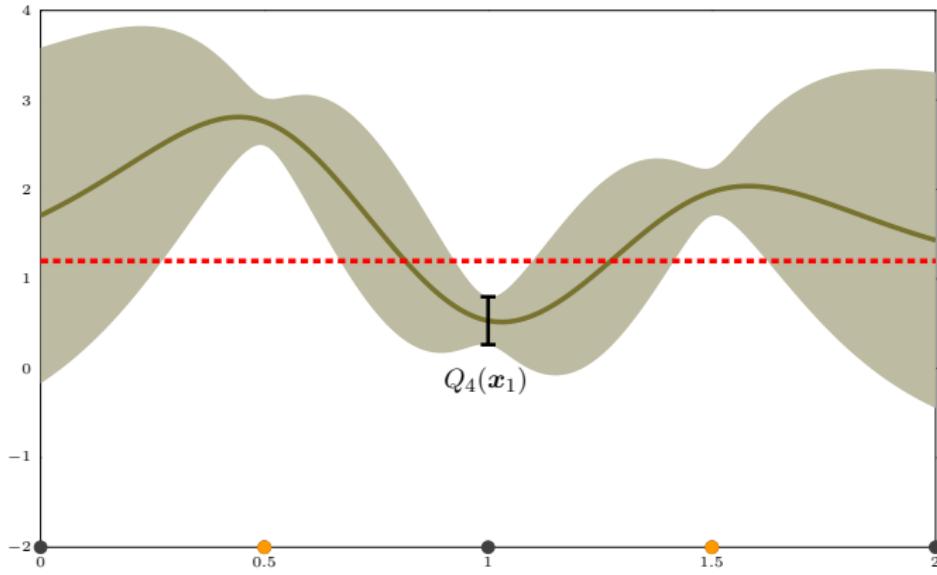
1. How do we **estimate** the function? ✓

2. How do we **classify**?



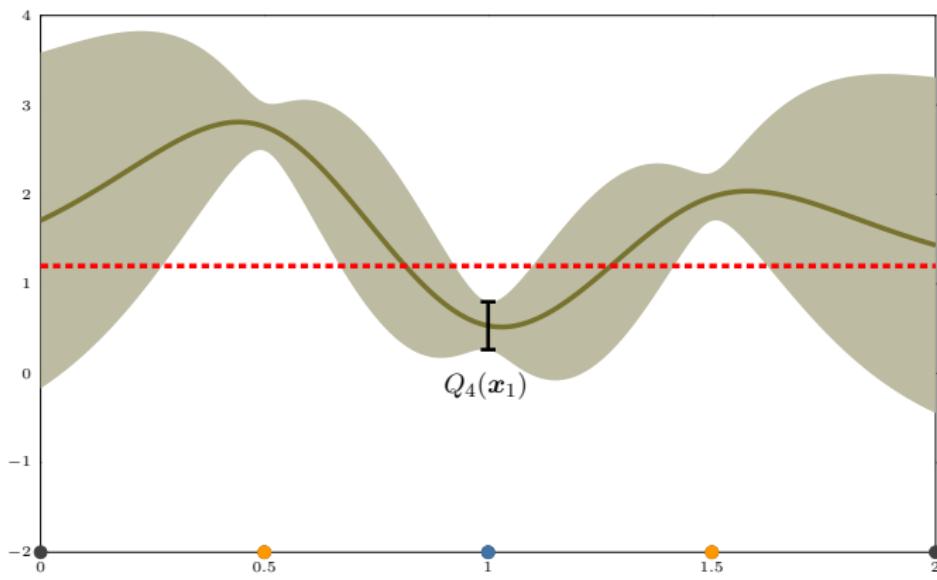
1. How do we **estimate** the function? ✓

2. How do we **classify**?



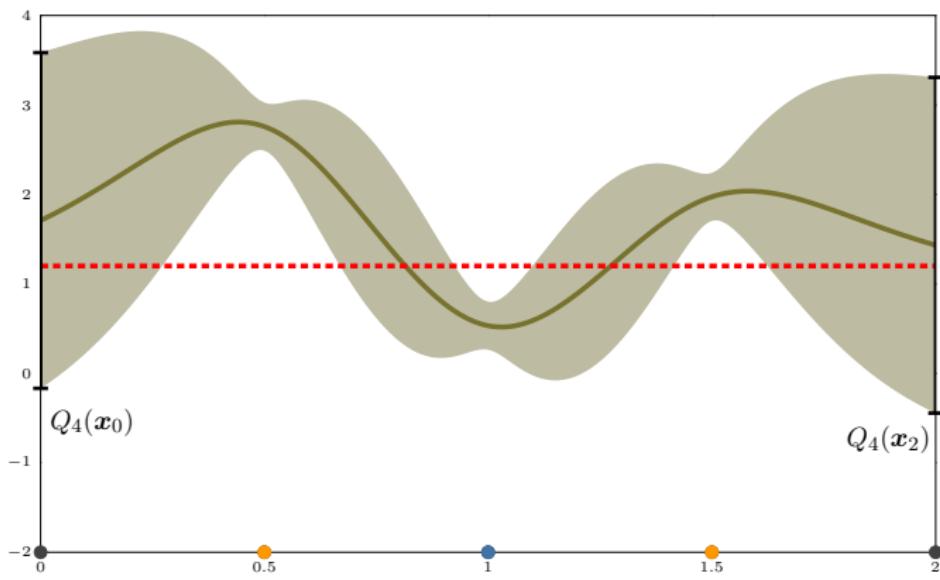
1. How do we **estimate** the function? ✓

2. How do we **classify**?



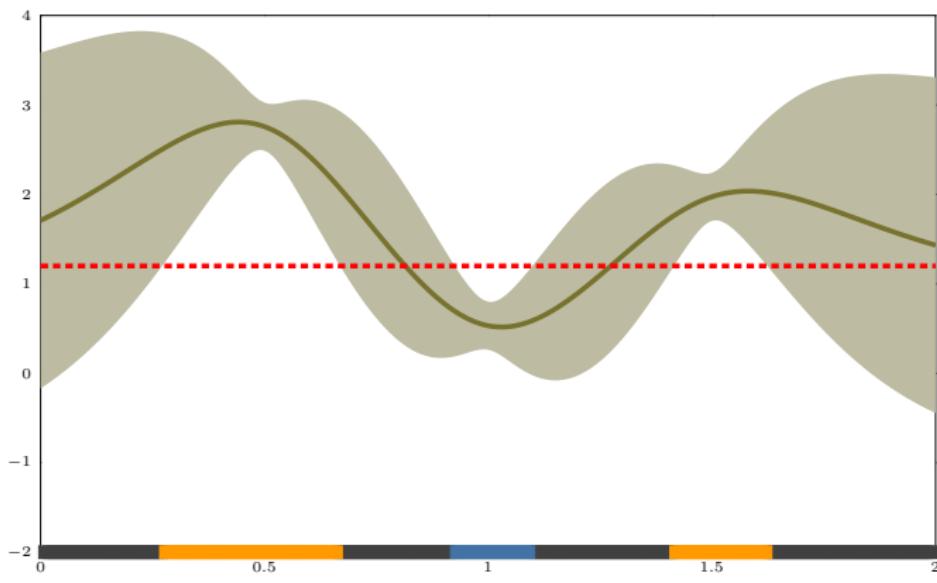
1. How do we **estimate** the function? ✓

2. How do we **classify**?



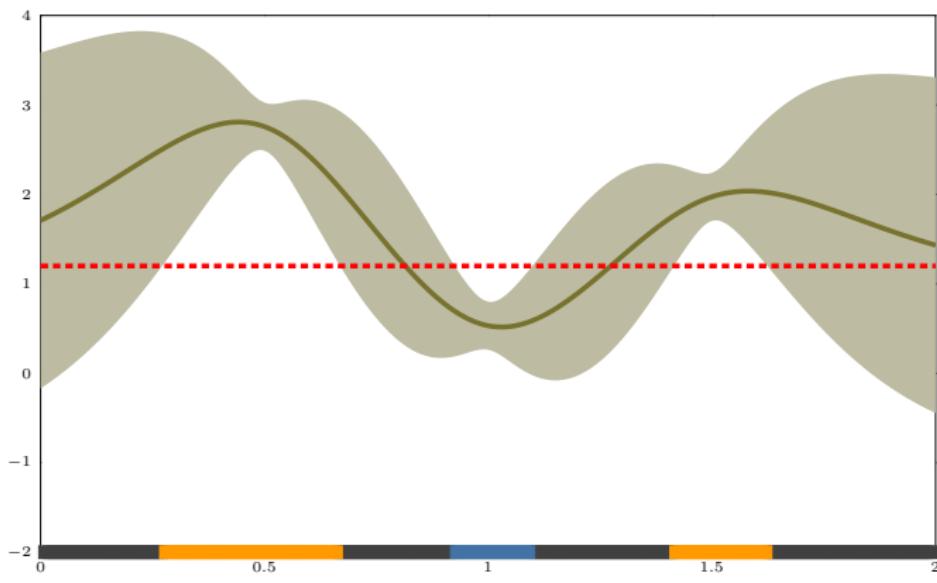
1. How do we **estimate** the function? ✓

2. How do we **classify**?

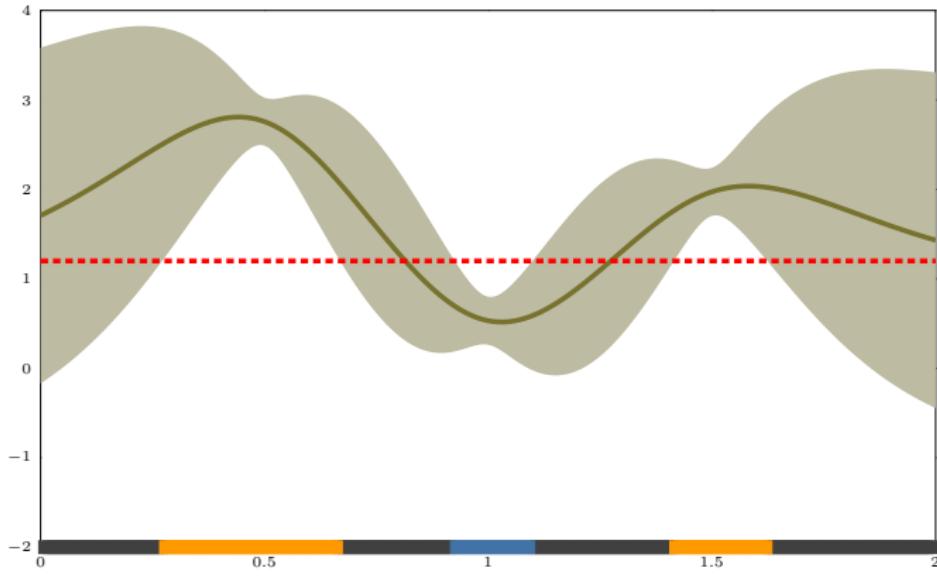


1. How do we **estimate** the function? ✓

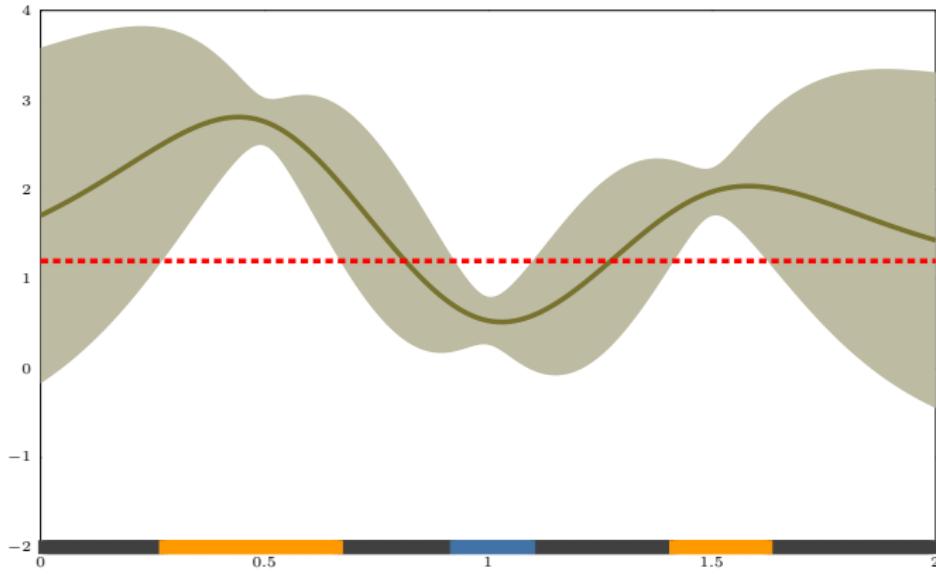
2. How do we **classify**? ✓



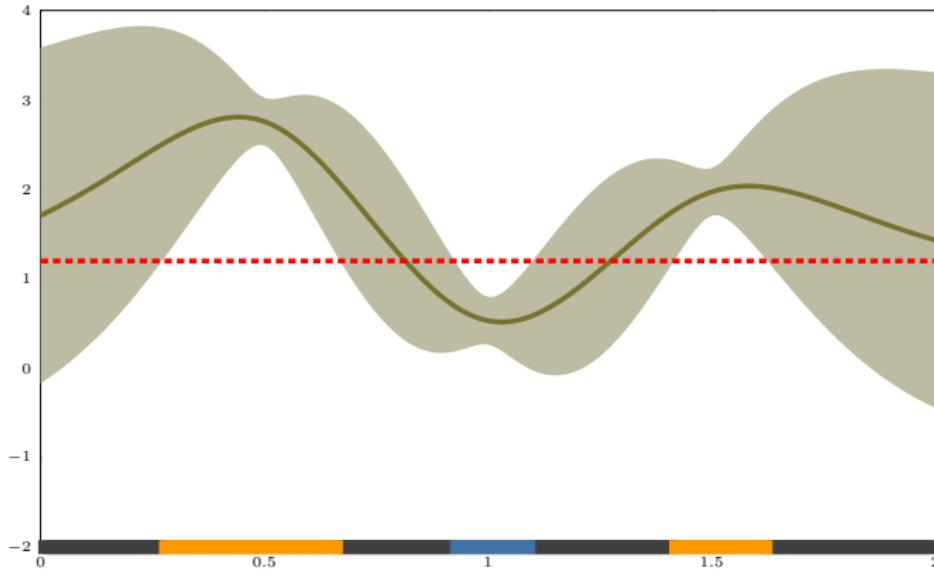
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?



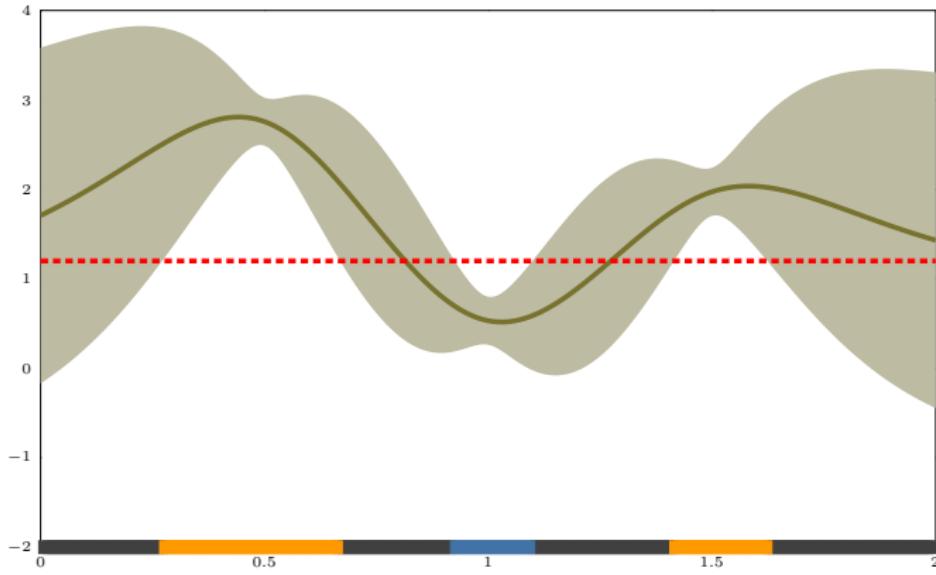
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.



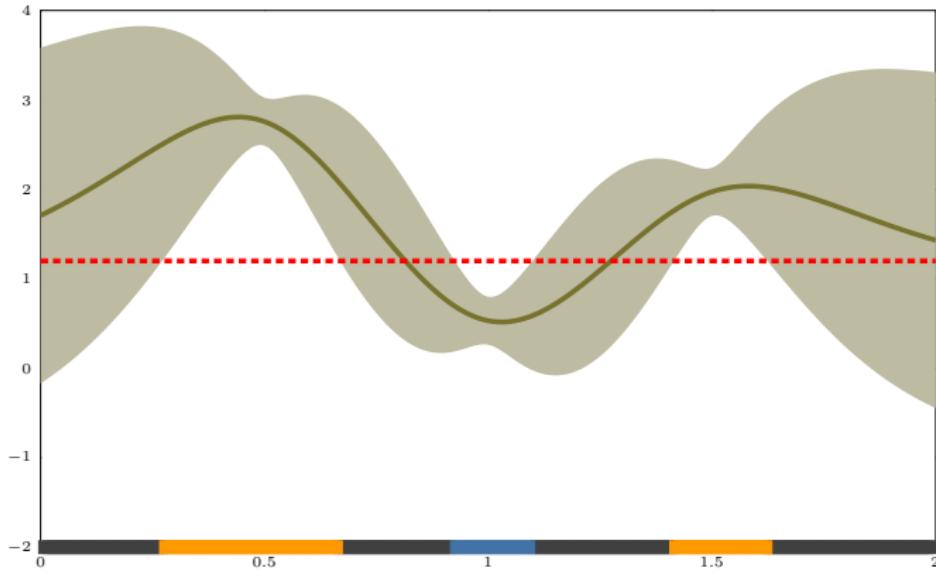
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!



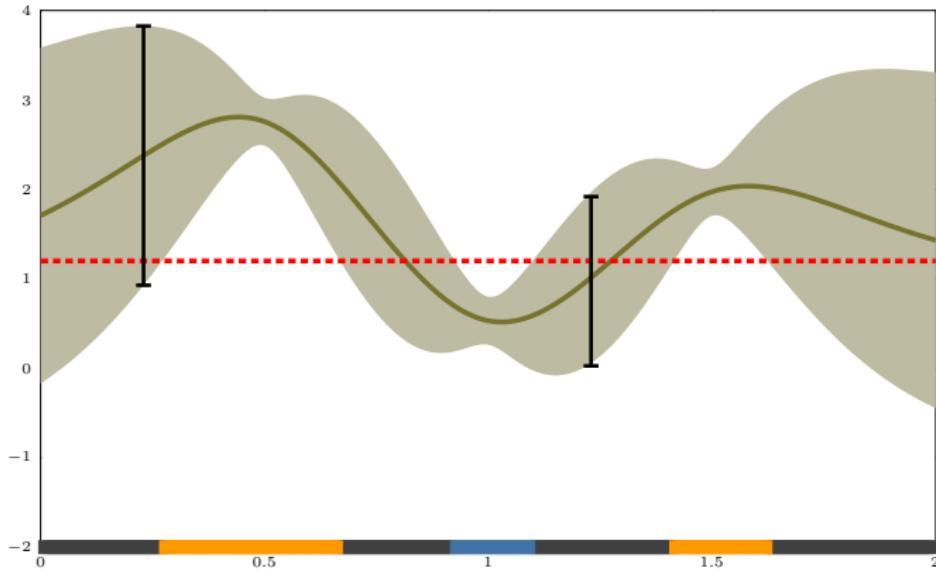
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance,



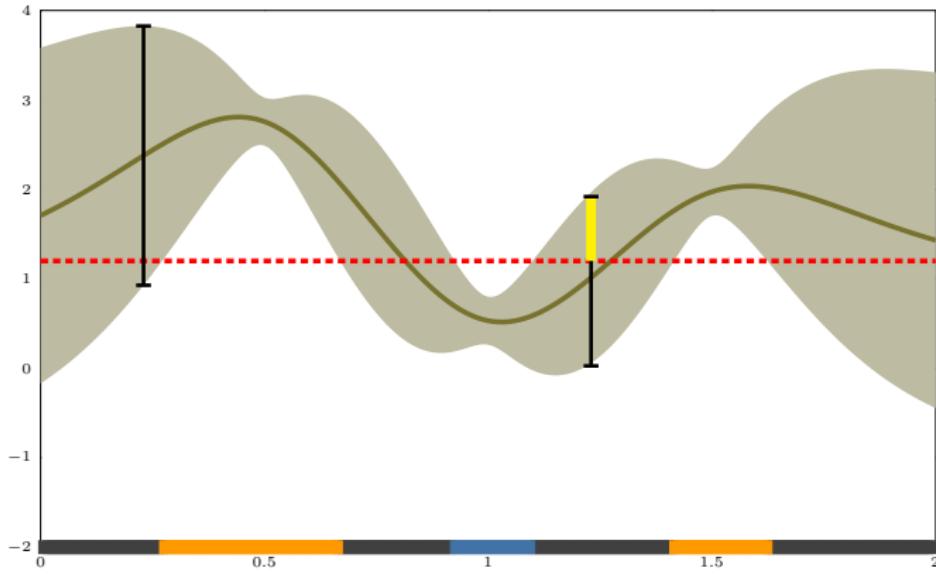
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance, max. ambiguity)



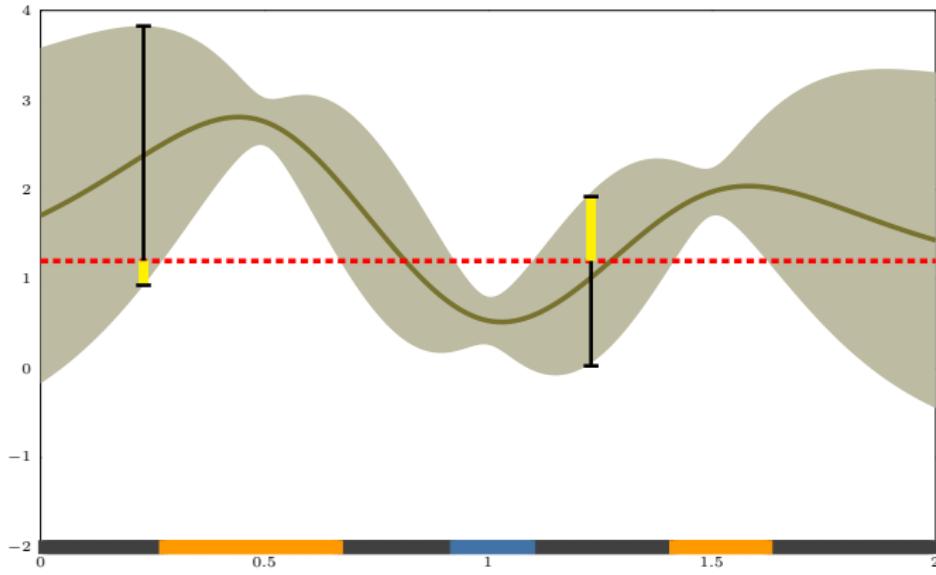
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance, max. ambiguity)



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance, max. ambiguity)



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance, max. ambiguity)

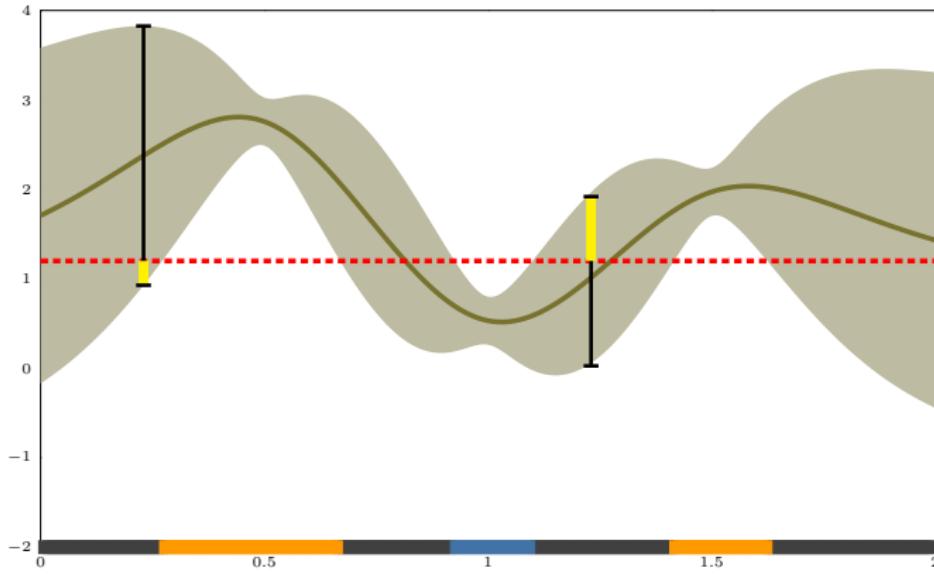


1. How do we **estimate** the function? ✓

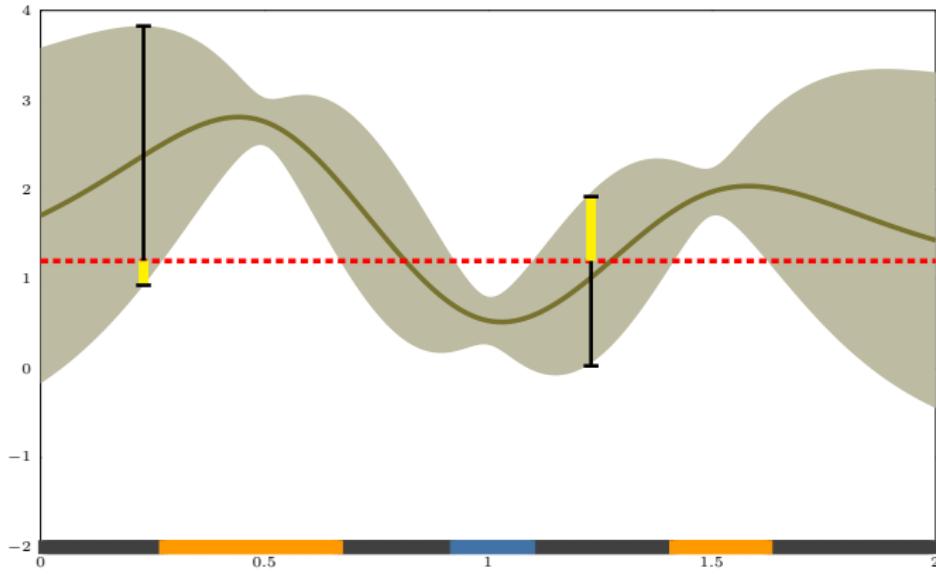
2. How do we **classify**? ✓

3. How do we **select** “informative” measurements?

- ▶ Pick among the yet unclassified.
- ▶ Doesn't really matter which!
(e.g. max. variance, max. ambiguity, or even random)



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements? ✓
 - ▶ Pick among the yet unclassified.
 - ▶ Doesn’t really matter which!
(e.g. max. variance, max. ambiguity, or even random)



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

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}

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

The Level Set Estimation (LSE) algorithm

$$\begin{aligned}\hat{H} &\leftarrow H_{t-1} \\ \hat{L} &\leftarrow L_{t-1}\end{aligned}$$

Input: sample set D , GP prior (μ_0, k, σ_0) ,
thr. value h , accuracy parameter ϵ

Output: predicted sets \hat{H}, \hat{L}

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$, for all $x \in D$

$t \leftarrow 1$

while $U_{t-1} \neq \emptyset$ **do**

$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

$t \leftarrow t + 1$

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}

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$, for all $x \in D$

$t \leftarrow 1$

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 $x \in U_{t-1}$ **do**

$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$

if $\min(C_t(x)) + \epsilon > h$ **then**

$U_t \leftarrow U_t \setminus \{x\}$

$H_t \leftarrow H_t \cup \{x\}$

else if $\max(C_t(x)) - \epsilon \leq h$ **then**

$U_t \leftarrow U_t \setminus \{x\}$

$L_t \leftarrow L_t \cup \{x\}$

end if

end for

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

← classify

$t \leftarrow t + 1$

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}

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$, for all $x \in D$

$t \leftarrow 1$

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 $x \in U_{t-1}$ **do**

$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$

if $\min(C_t(x)) + \epsilon > h$ **then**

$U_t \leftarrow U_t \setminus \{x\}$

$H_t \leftarrow H_t \cup \{x\}$

else if $\max(C_t(x)) - \epsilon \leq h$ **then**

$U_t \leftarrow U_t \setminus \{x\}$

$L_t \leftarrow L_t \cup \{x\}$

end if

end for

$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$

$y_t \leftarrow f(x_t) + n_t$

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

← classify

← select max. ambiguity point

$t \leftarrow t + 1$

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}

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

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 $x \in U_{t-1}$ **do**

$$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$$

if $\min(C_t(x)) + \epsilon > h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$H_t \leftarrow H_t \cup \{x\}$$

else if $\max(C_t(x)) - \epsilon \leq h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$L_t \leftarrow L_t \cup \{x\}$$

end if

end for

$$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$$

$$y_t \leftarrow f(x_t) + n_t$$

Compute $\mu_t(x)$ and $\sigma_t(x)$, $\forall x \in U_t$

$$t \leftarrow t + 1$$

end while

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

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

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

← classify

← select max. ambiguity point

← update GP estimate

Input: sample set D , GP prior (μ_0, k, σ_0) ,
thr. value h , accuracy parameter ϵ

Output: predicted sets \hat{H}, \hat{L}

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

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 $x \in U_{t-1}$ **do**

$$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$$

if $\min(C_t(x)) + \epsilon > h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$H_t \leftarrow H_t \cup \{x\}$$

else if $\max(C_t(x)) - \epsilon \leq h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$L_t \leftarrow L_t \cup \{x\}$$

end if

end for

$$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$$

$$y_t \leftarrow f(x_t) + n_t$$

Compute $\mu_t(x)$ and $\sigma_t(x)$, $\forall x \in U_t$

$$t \leftarrow t + 1$$

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}

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

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 $x \in U_{t-1}$ **do**

$$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$$

if $\min(C_t(x)) + \epsilon > h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$H_t \leftarrow H_t \cup \{x\}$$

else if $\max(C_t(x)) - \epsilon \leq h$ **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$L_t \leftarrow L_t \cup \{x\}$$

end if

end for

$$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$$

$$y_t \leftarrow f(x_t) + n_t$$

Compute $\mu_t(x)$ and $\sigma_t(x)$, $\forall x \in U_t$

$$t \leftarrow t + 1$$

end while

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

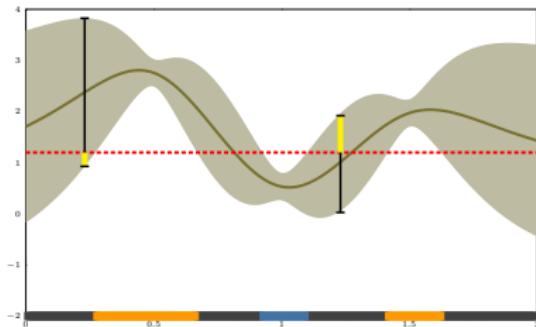
$$\hat{L} \leftarrow L_{t-1}$$

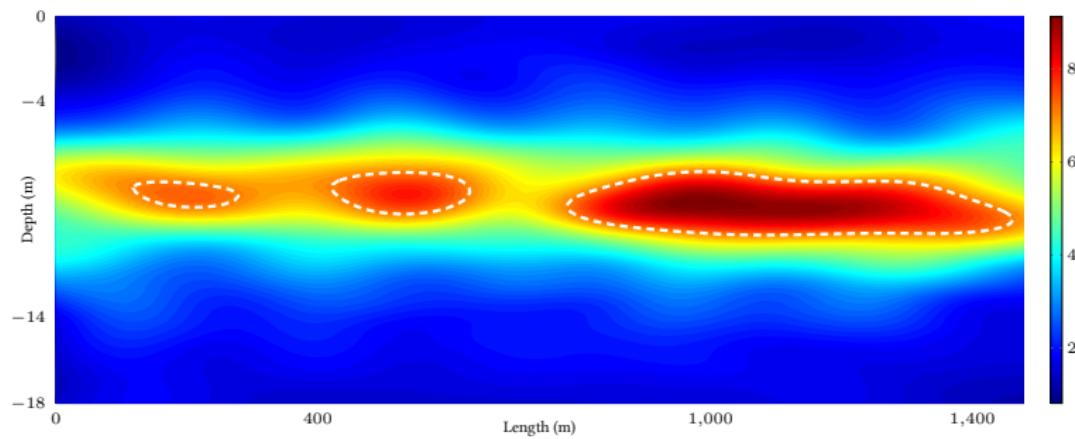
The Level Set Estimation (lse) algorithm

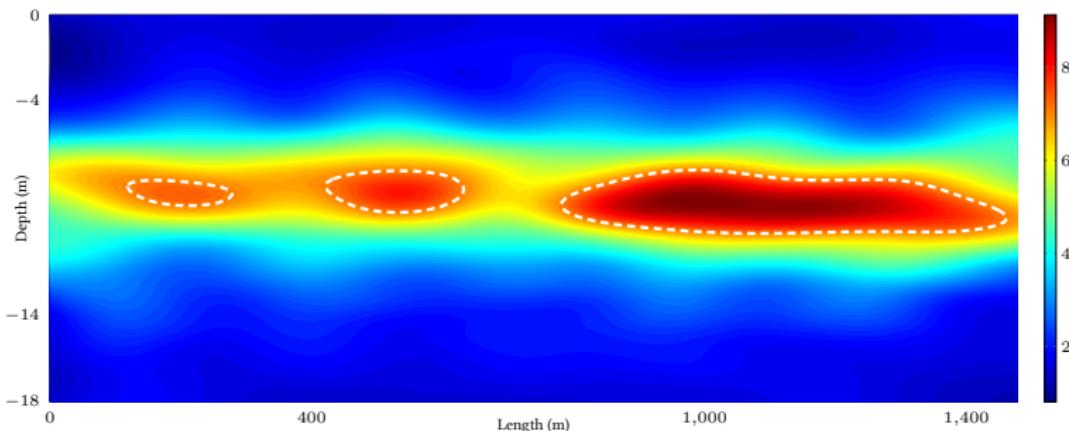
► Monotonicity of

1. confidence intervals
2. classification

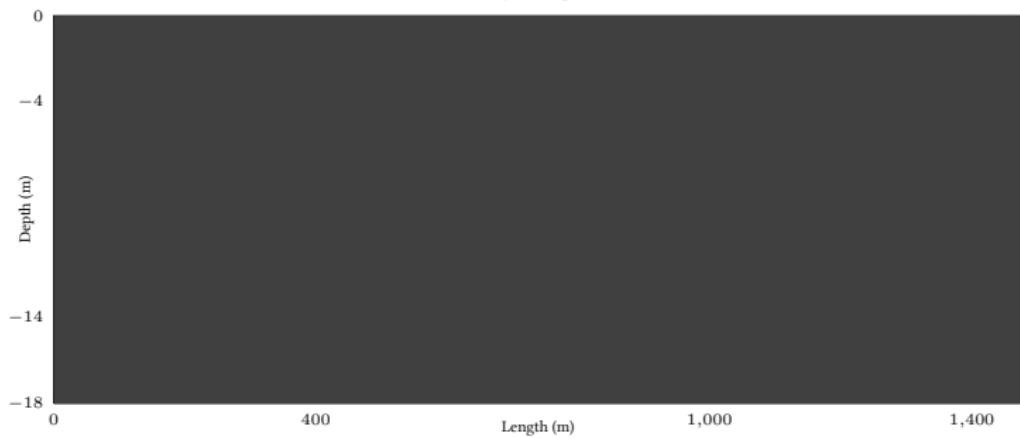
► Relaxed classification rules by an accuracy parameter ϵ
(trades off sampling cost for accuracy)

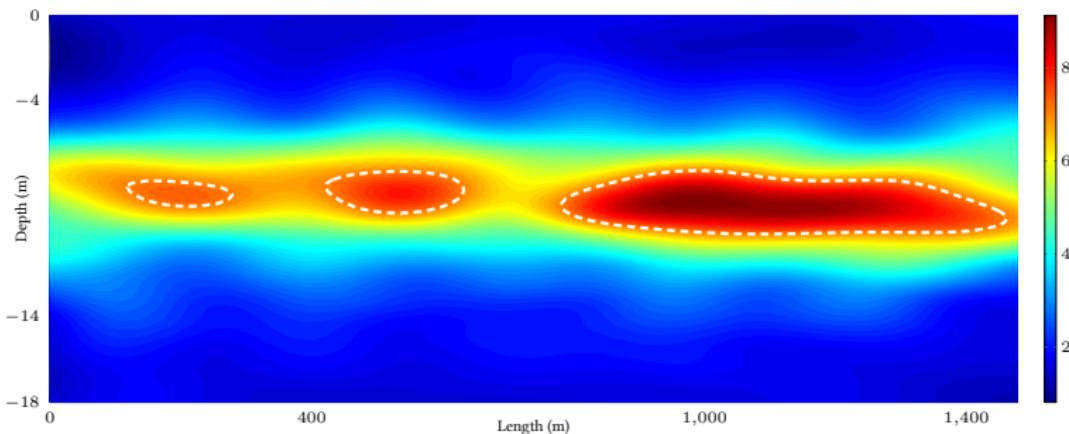




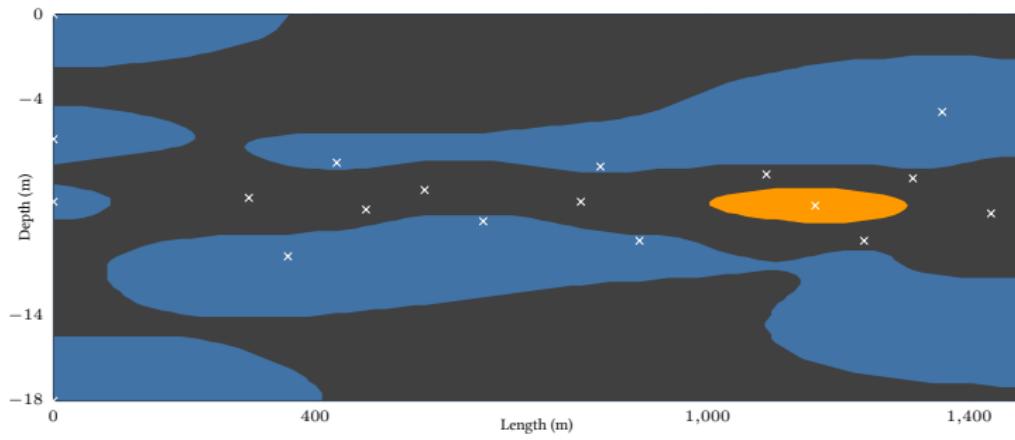


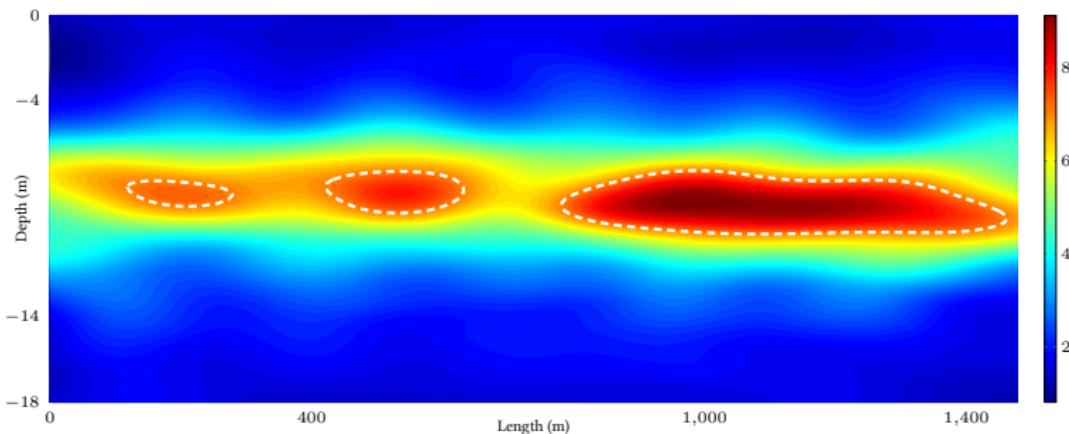
$t = 0$



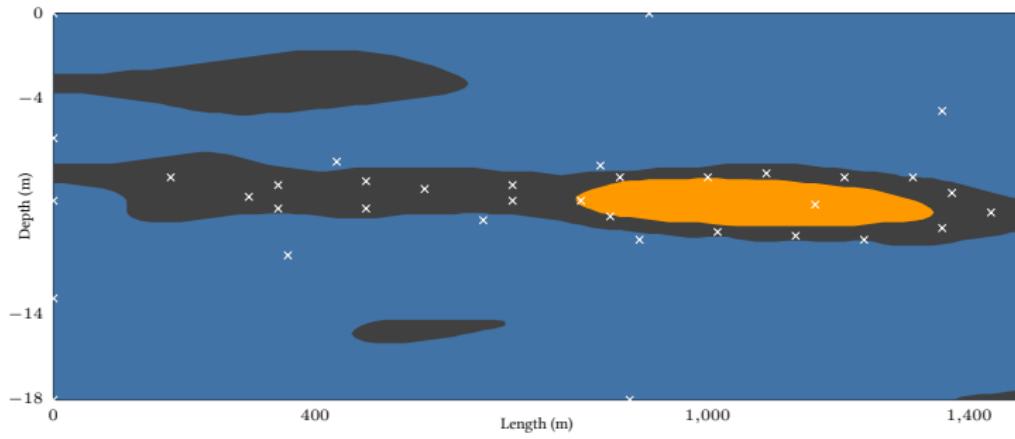


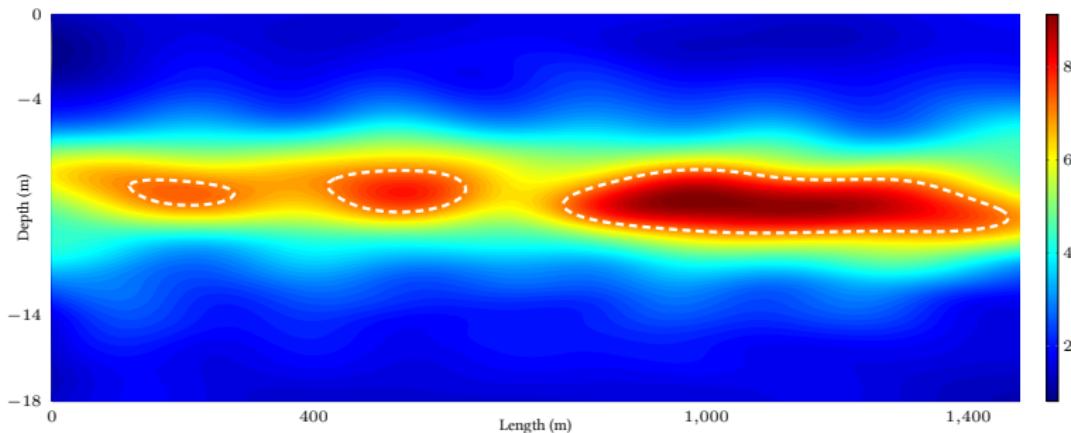
$t = 20$



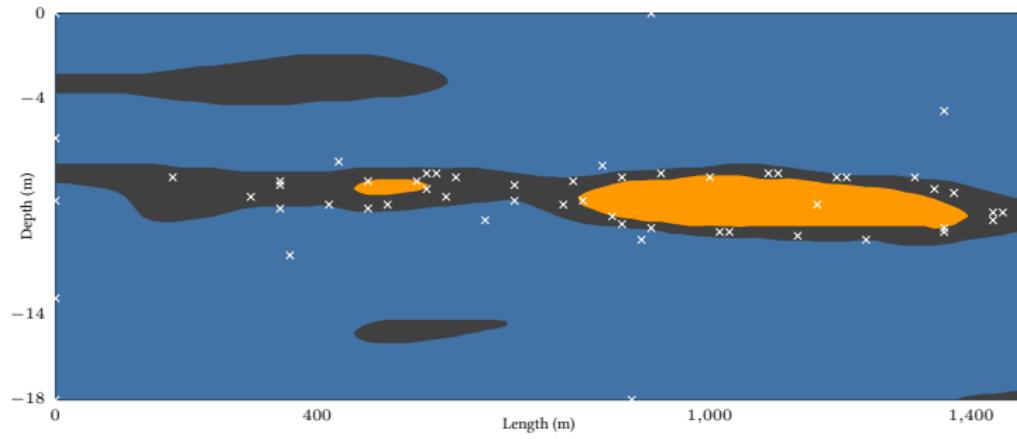


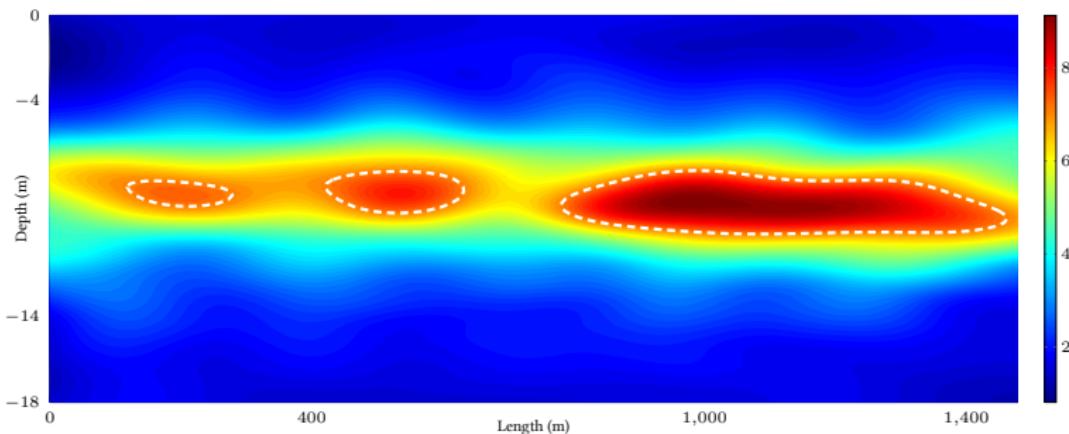
$t = 40$



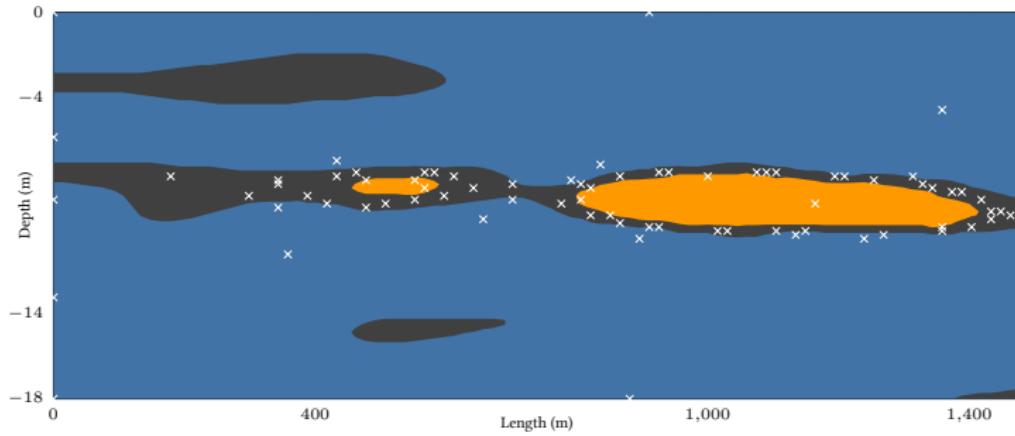


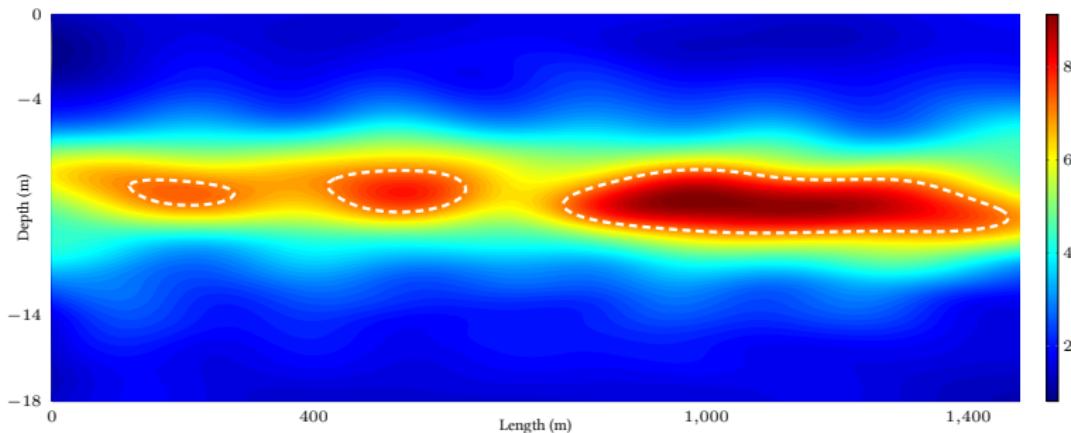
$t = 60$



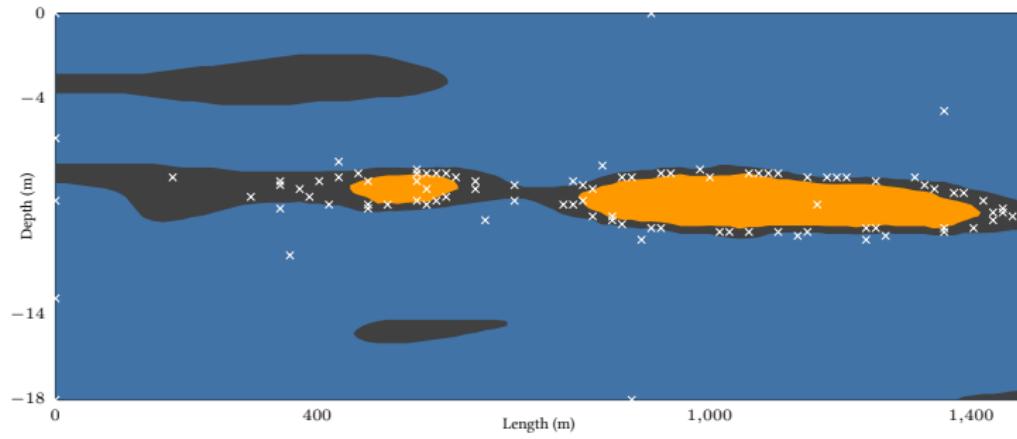


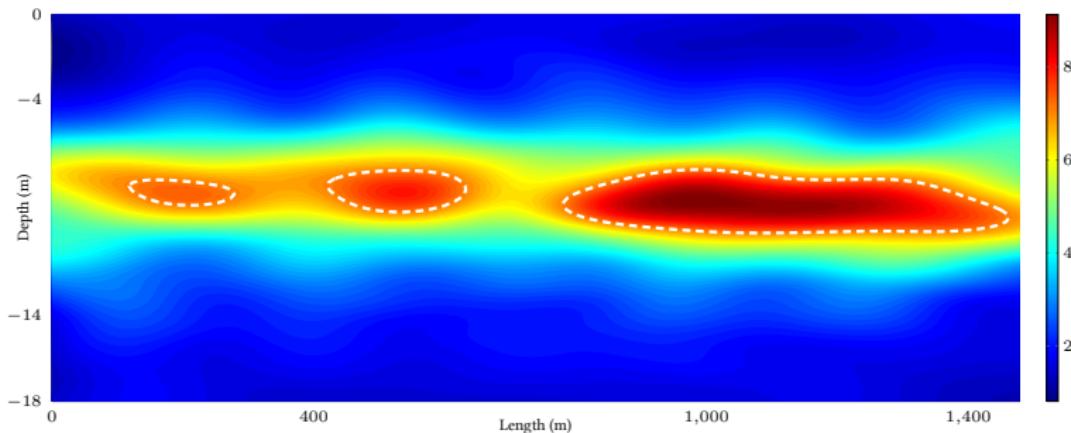
$t = 80$



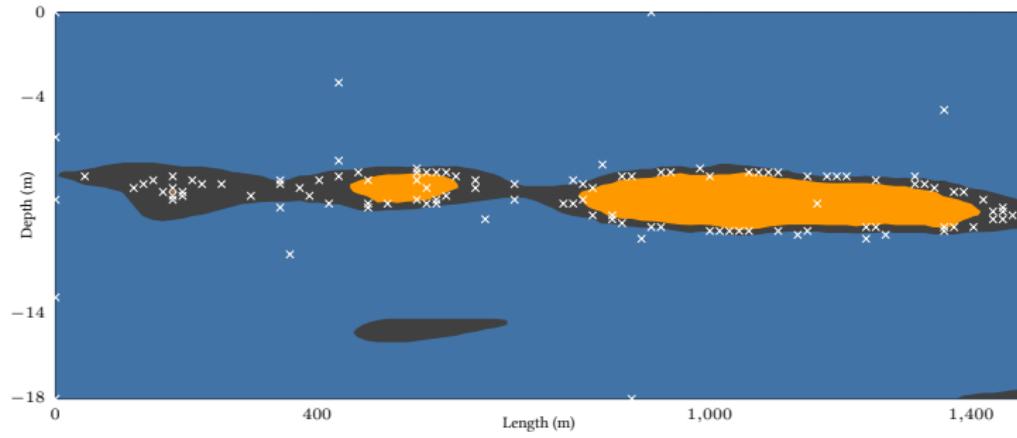


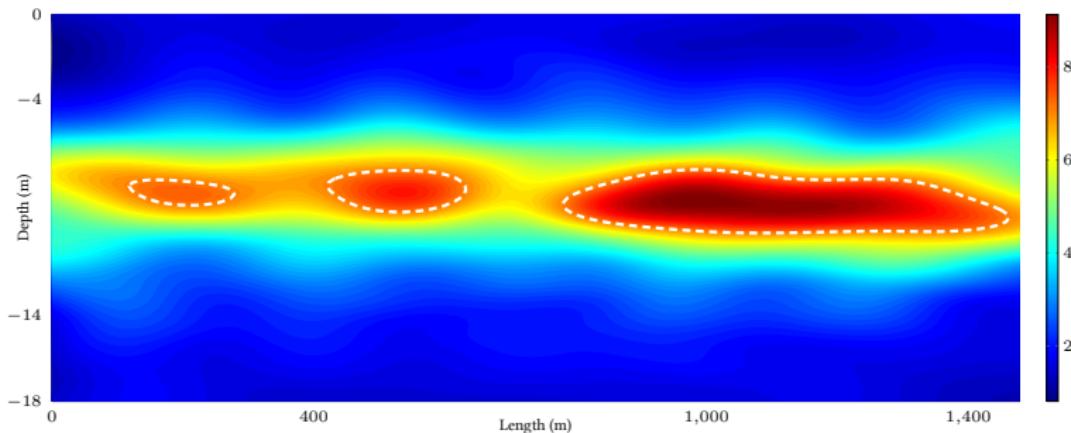
$t = 100$



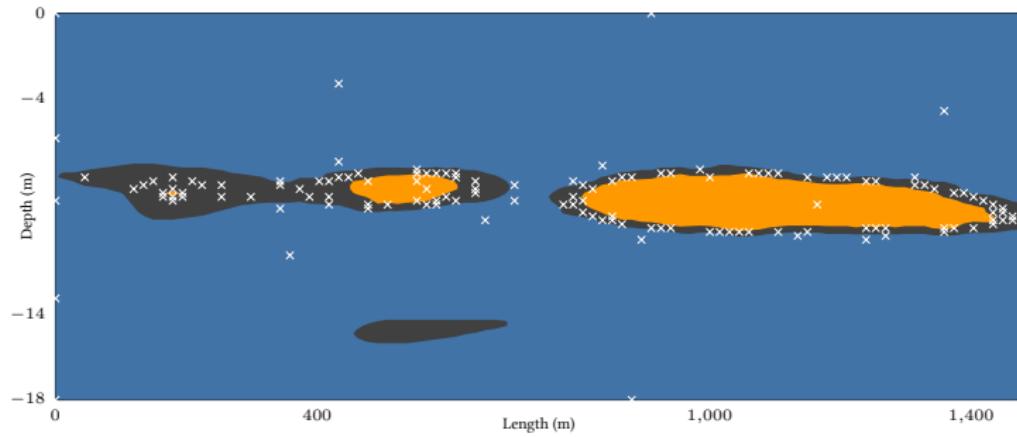


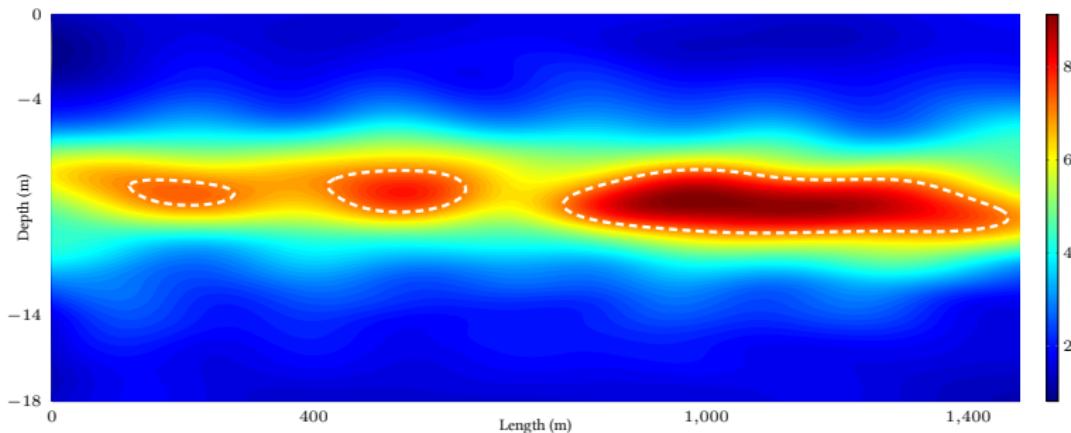
$t = 120$



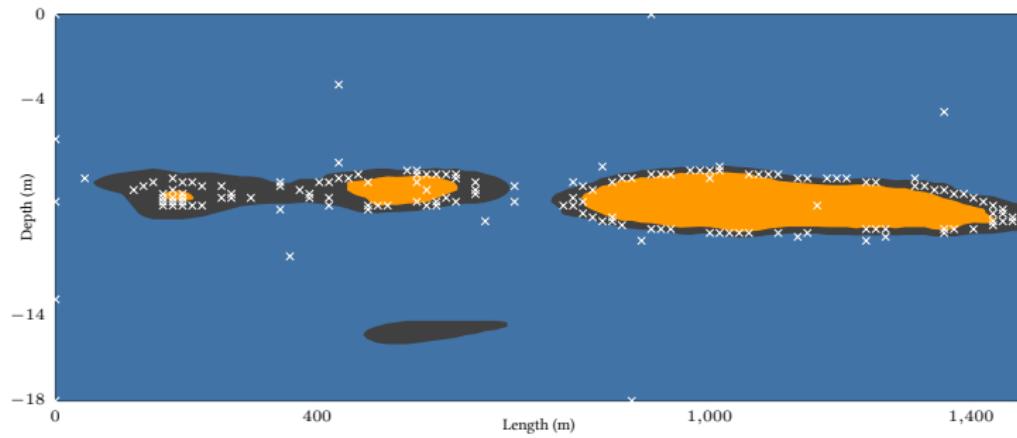


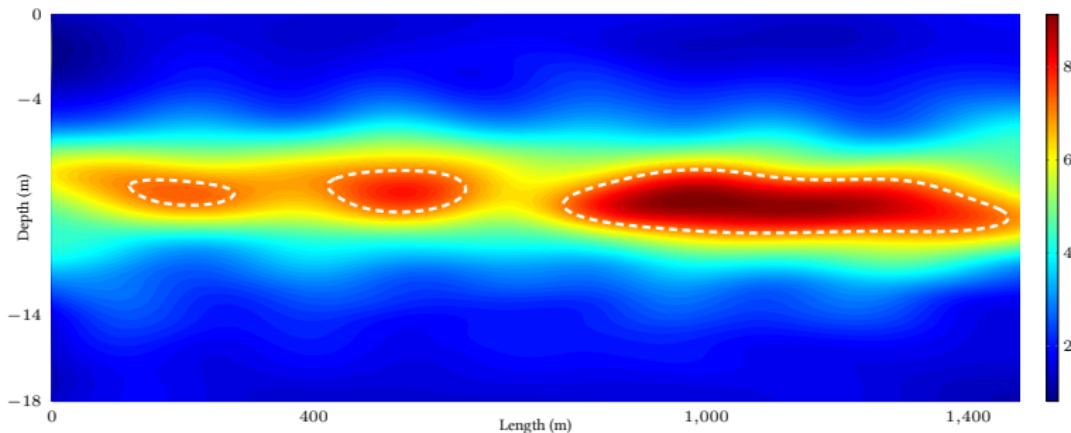
$t = 140$



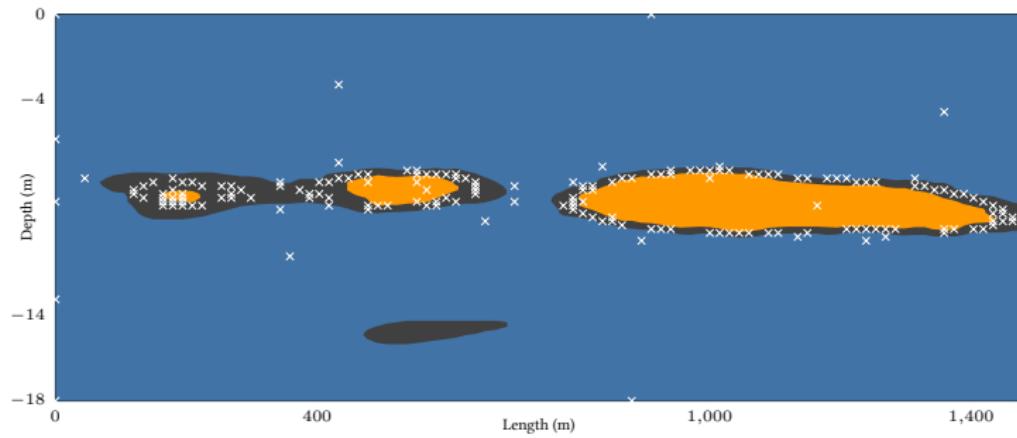


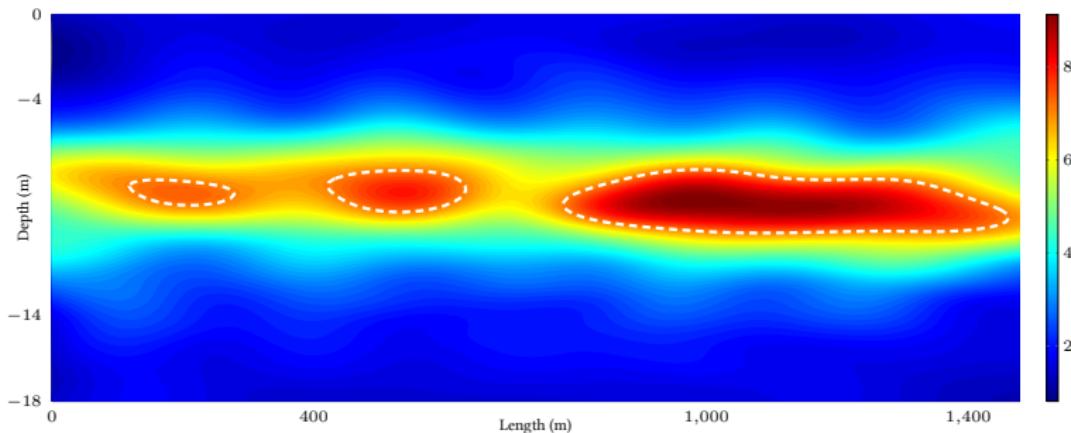
$t = 160$



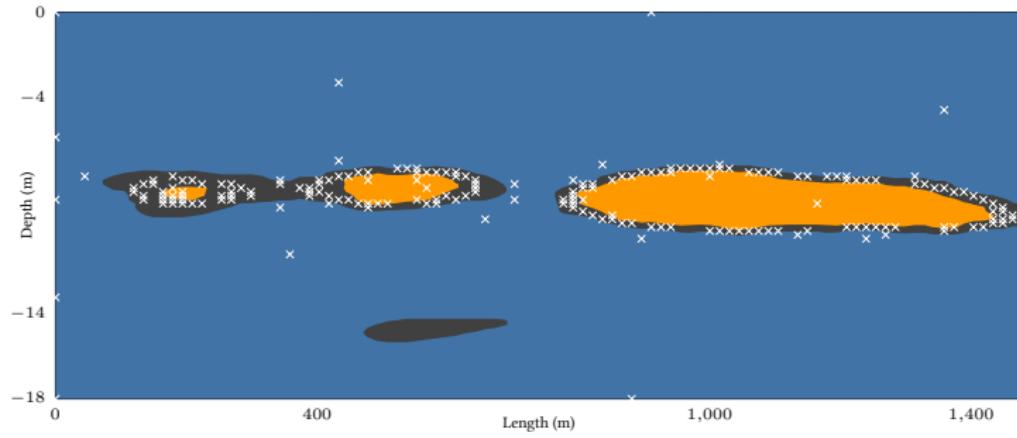


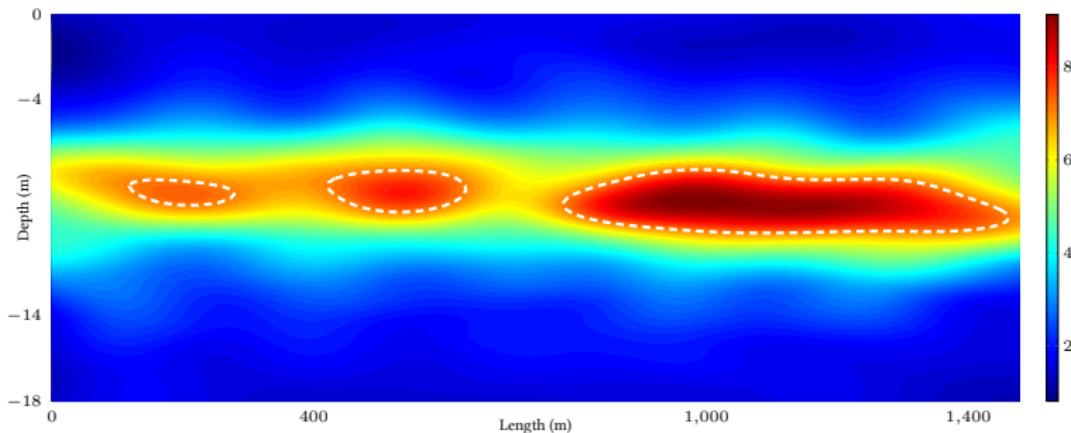
$t = 180$



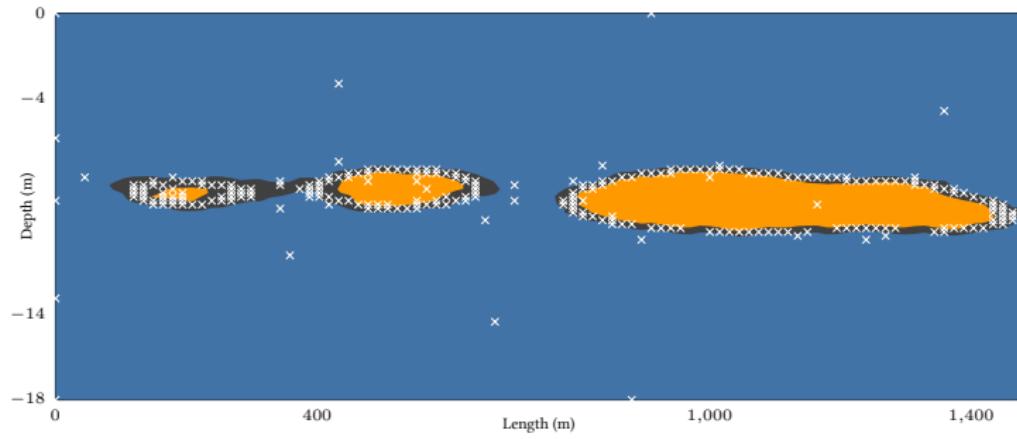


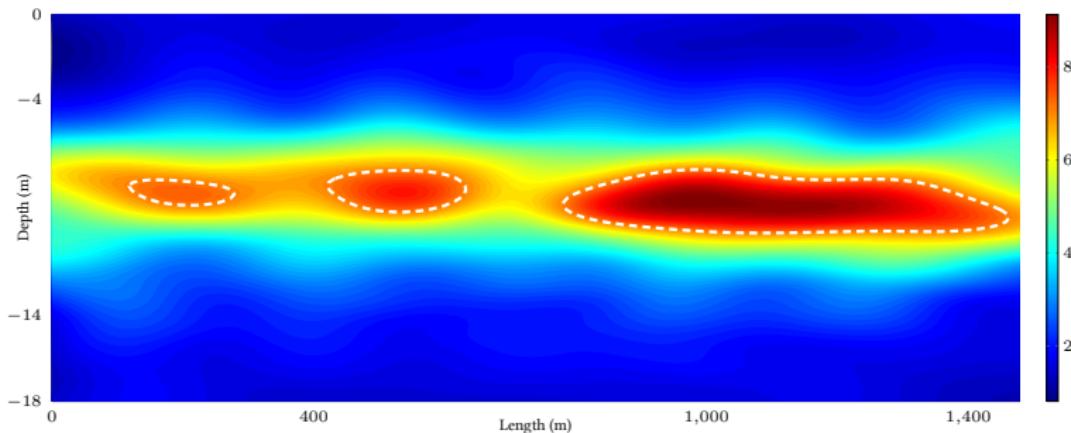
$t = 200$



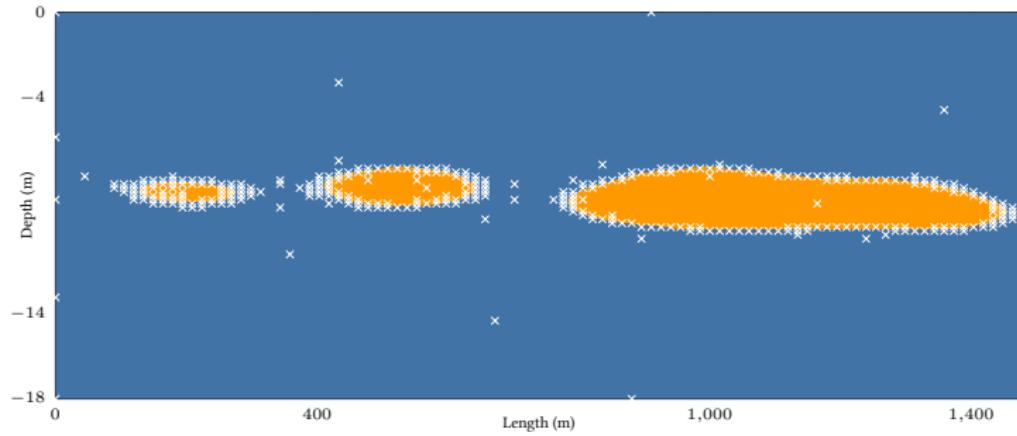


$t = 260$





$t = 354$



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

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

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:

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

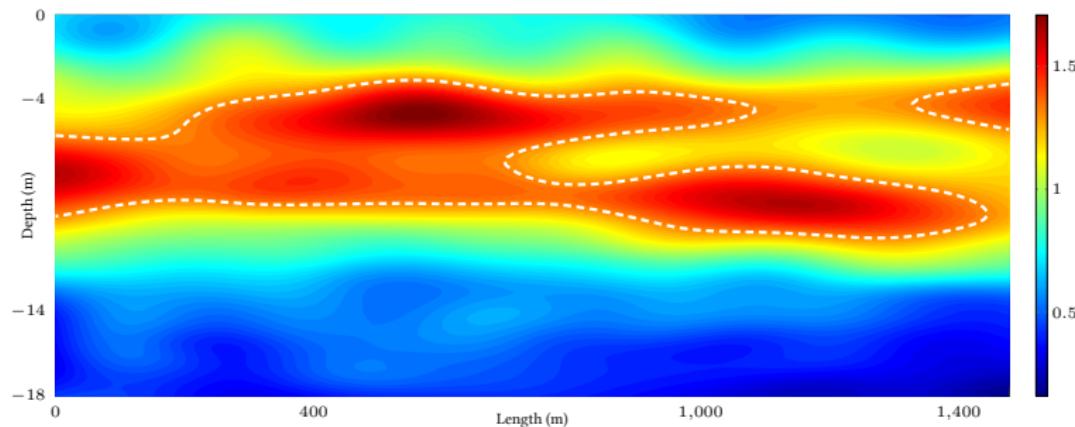
Experiments

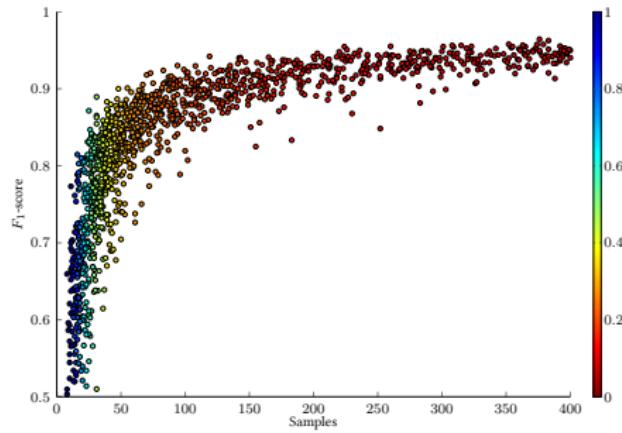
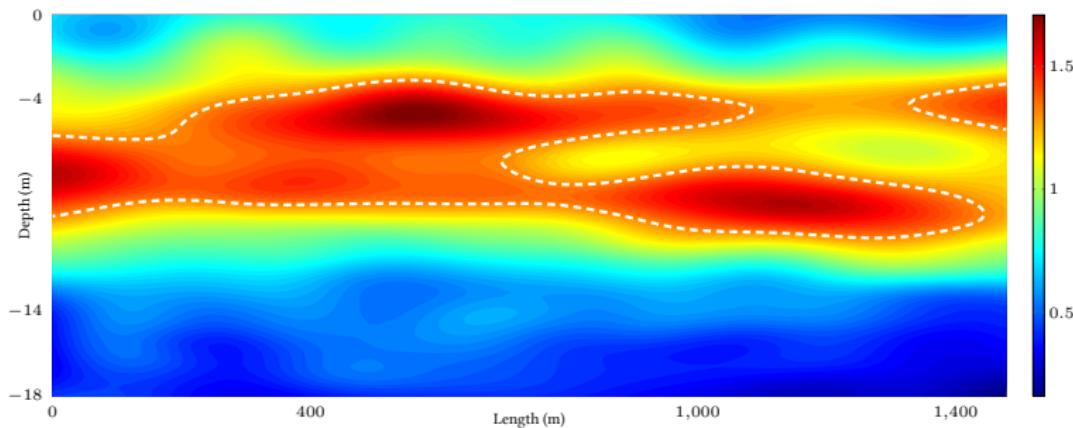
1. LSE
2. Maximum variance sampling:

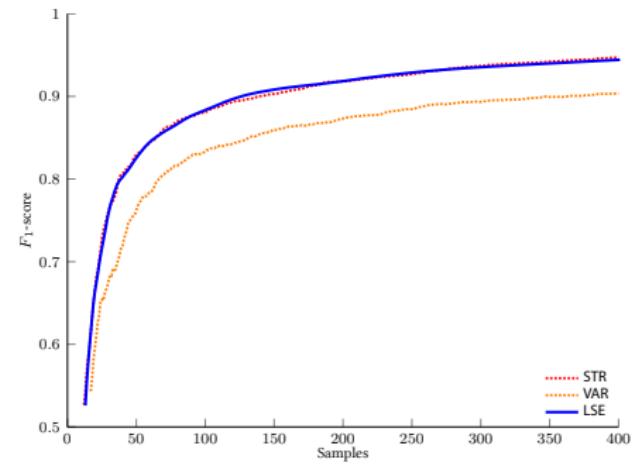
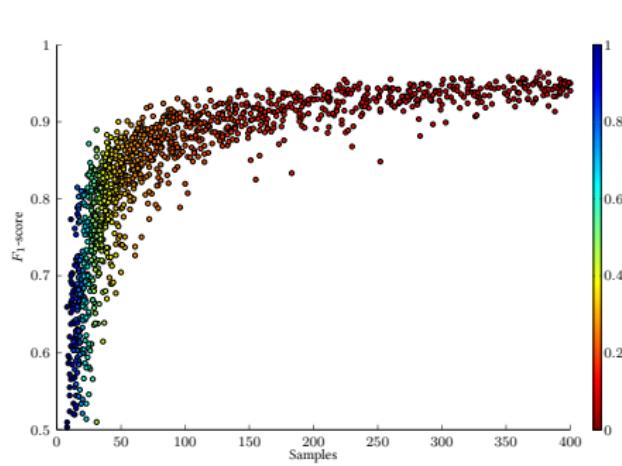
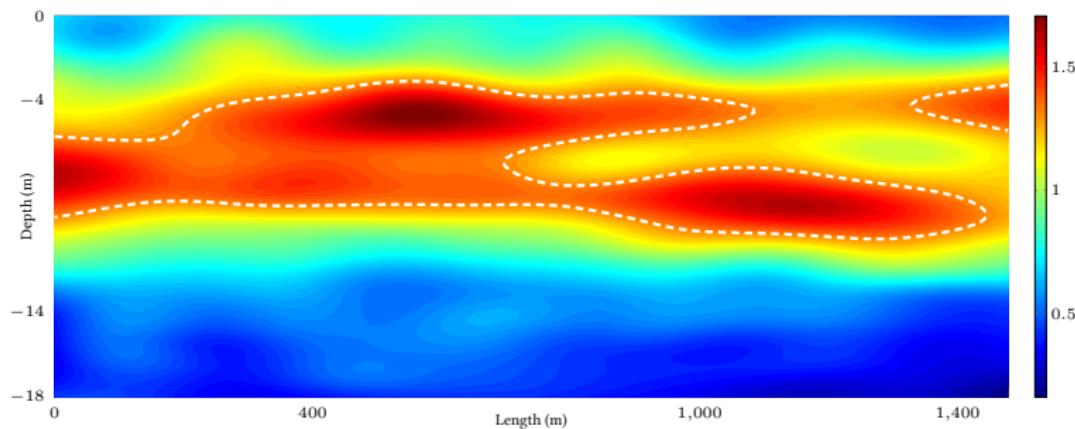
$$\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x} \in D} \sigma_{t-1}(\mathbf{x})$$

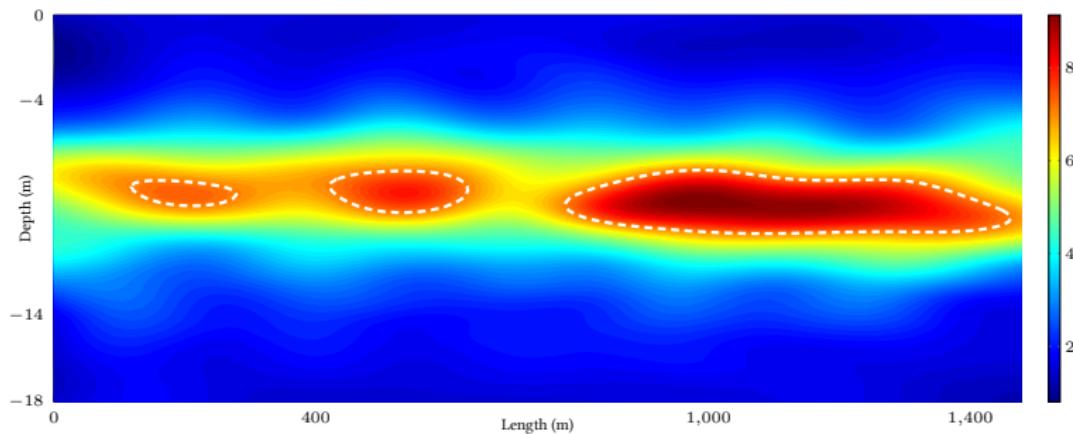
3. “Straddle” heuristic (Bryan *et al.*, 2005):

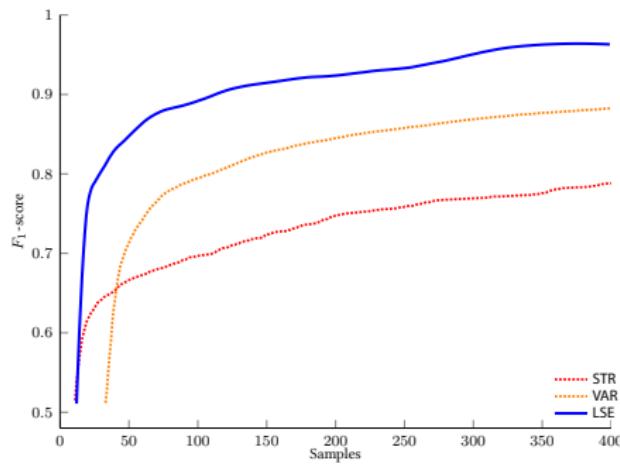
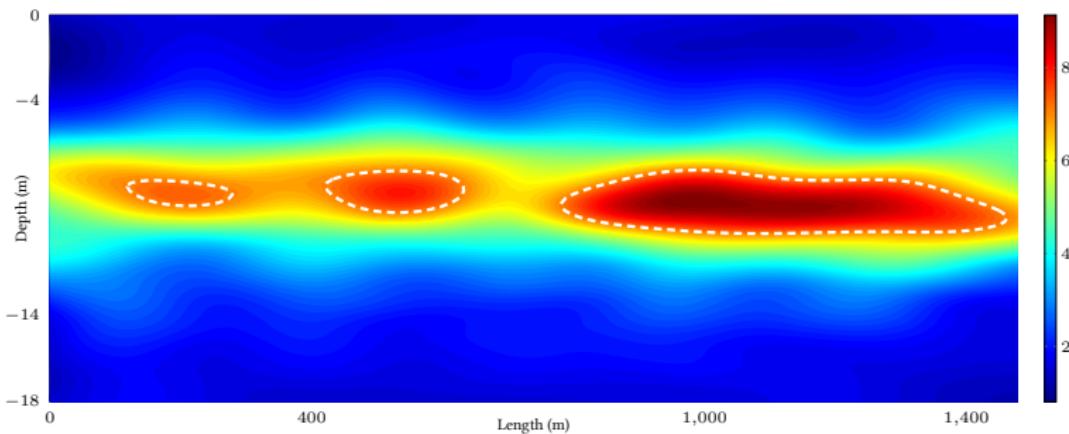
$$\mathbf{x}_t \approx \operatorname{argmax}_{\mathbf{x} \in D} a_{t-1}(\mathbf{x}) \quad (\text{for } \beta_t^{1/2} = 1.96)$$

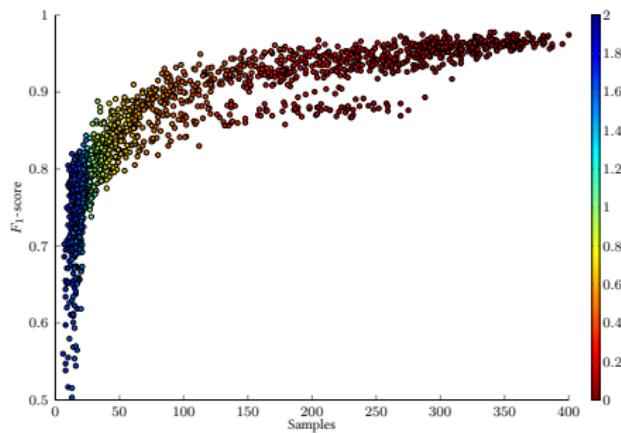
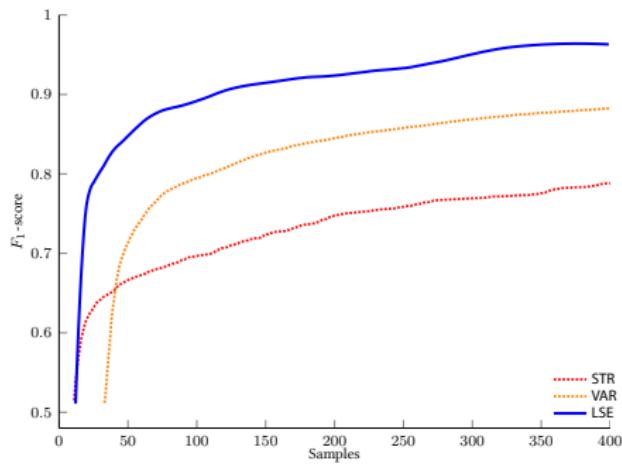
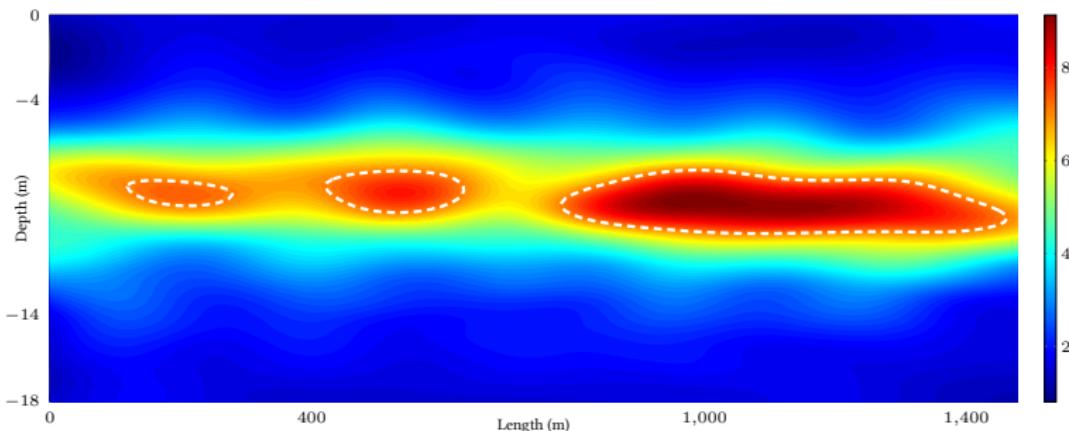


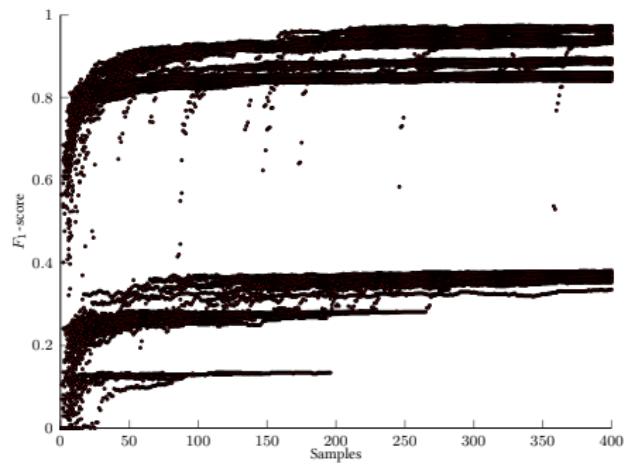
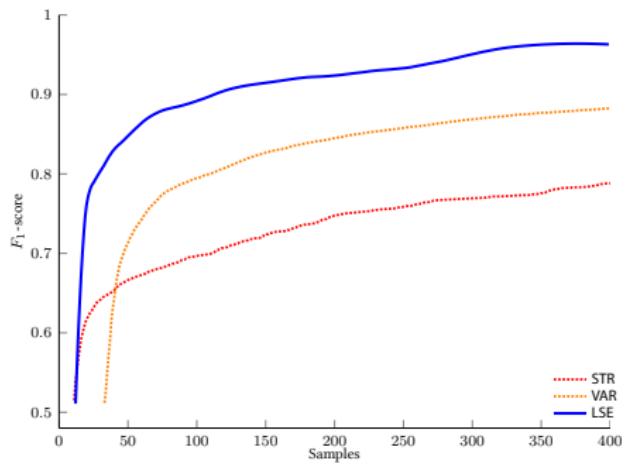
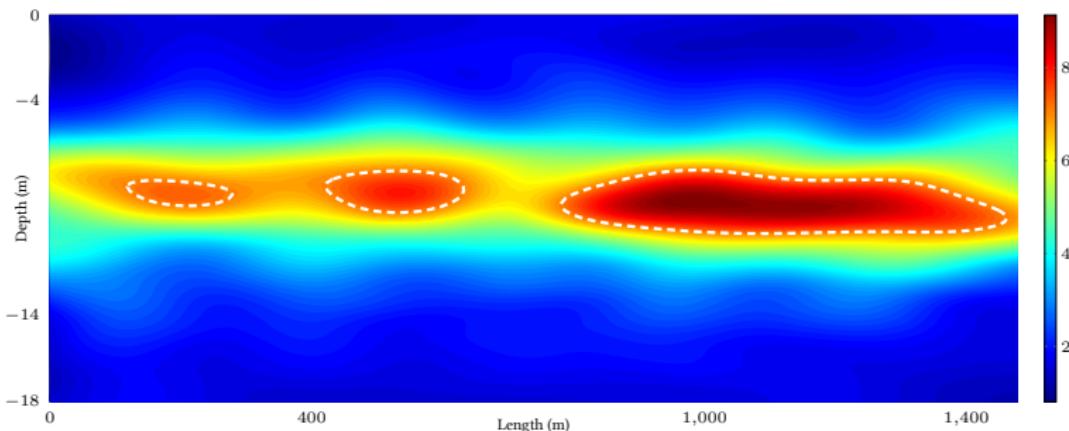


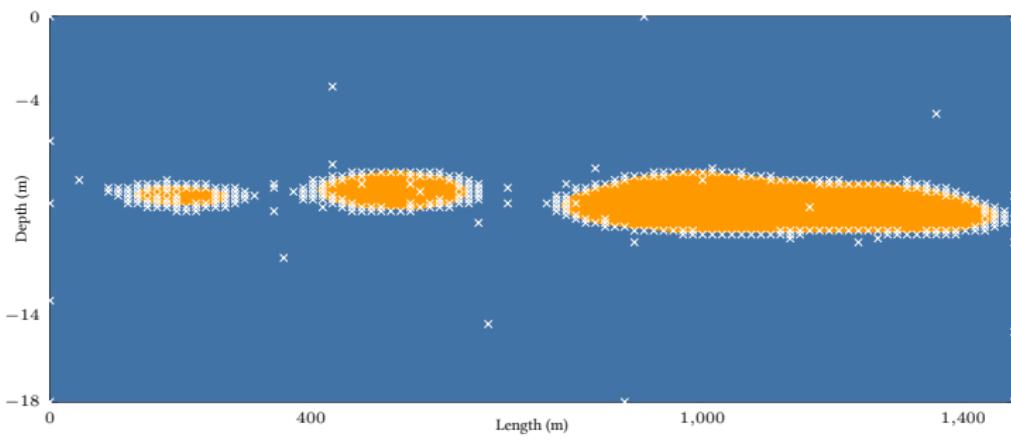
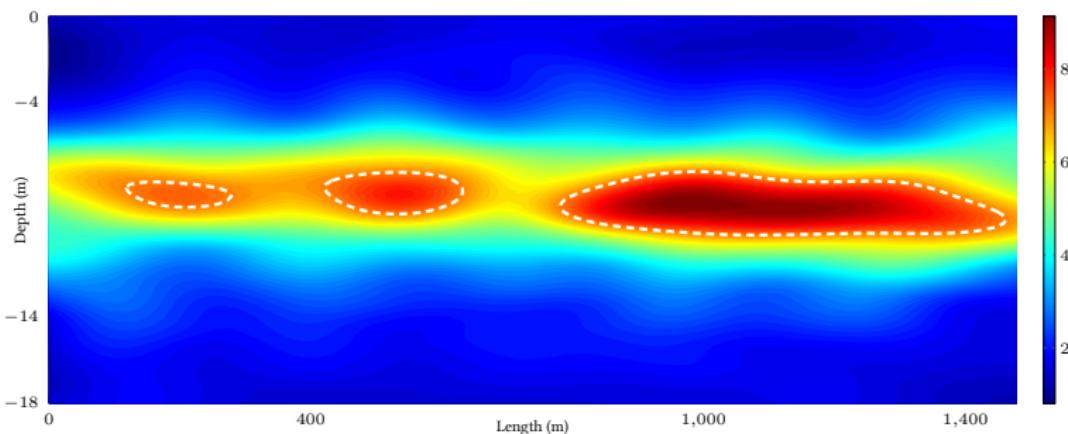


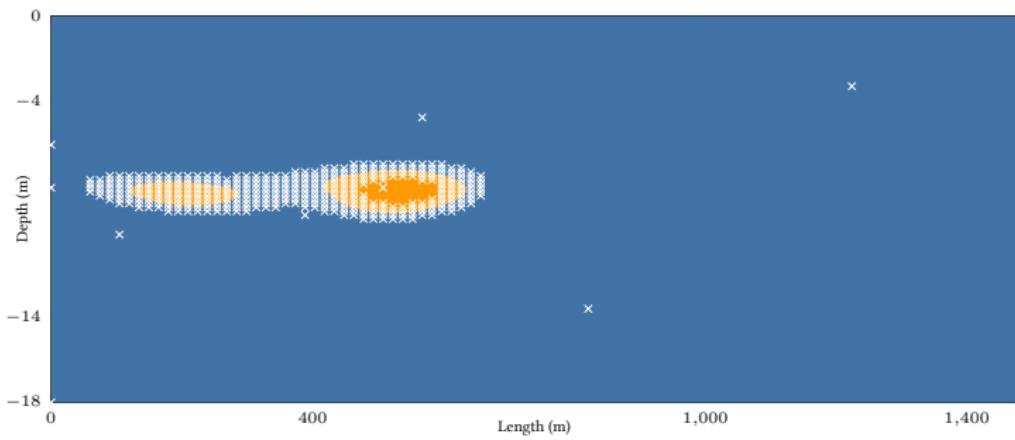
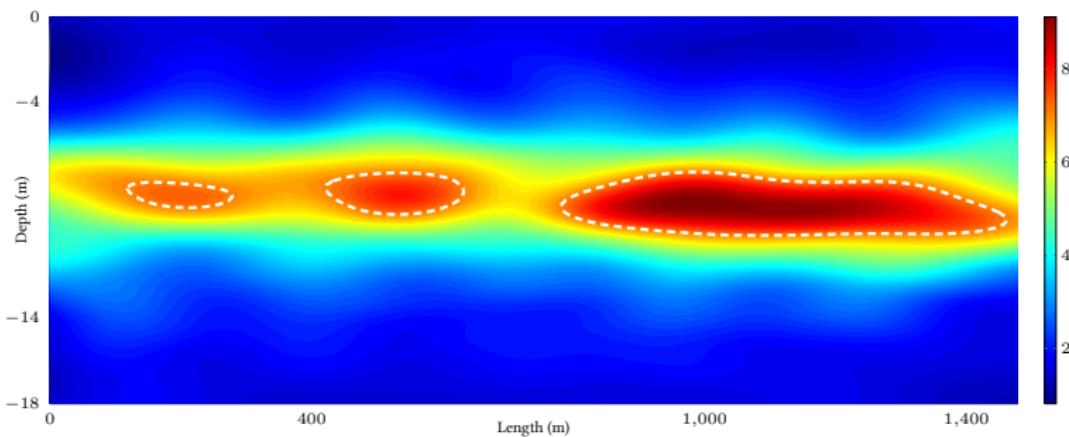


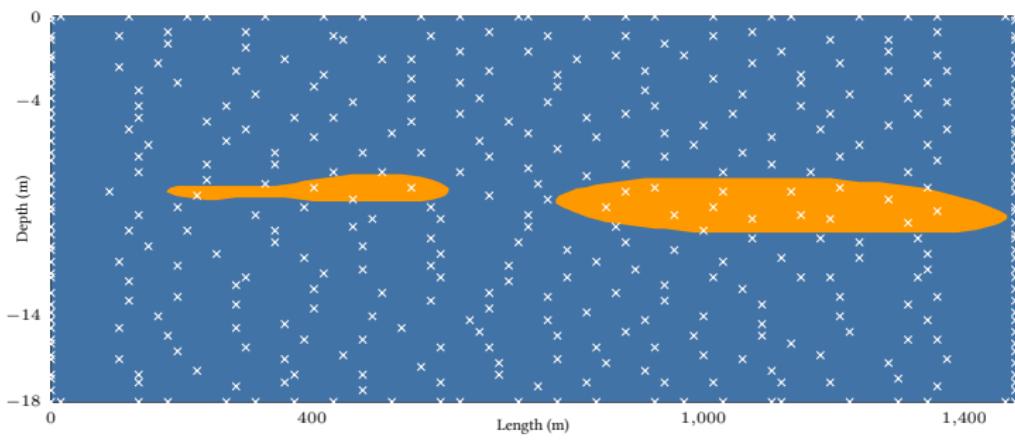
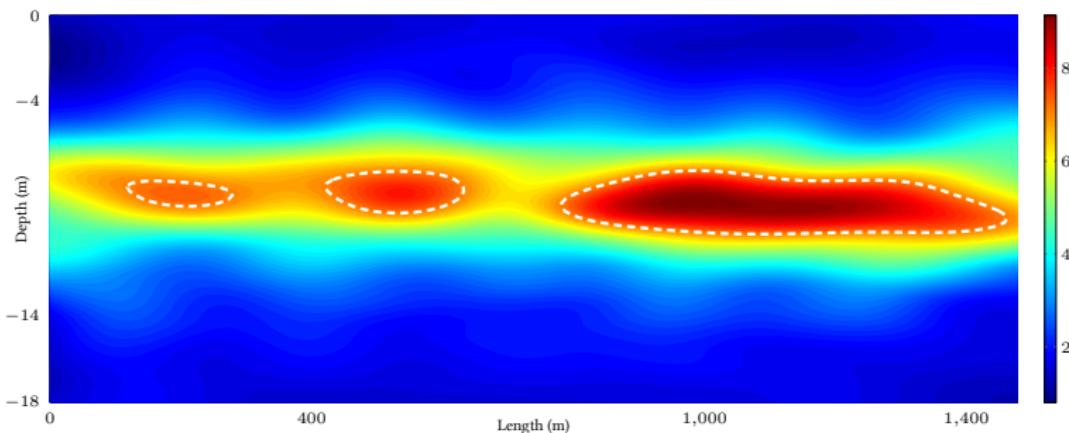












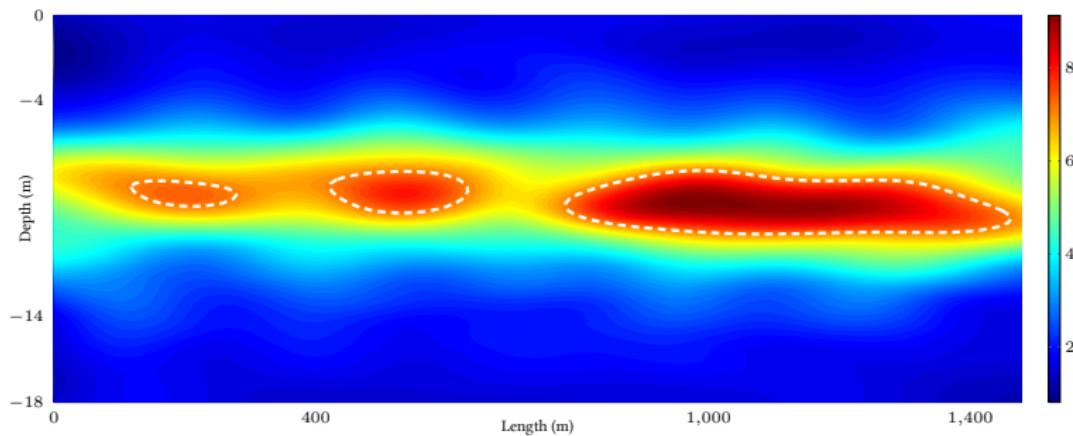
- ▶ What if we don't know h ?

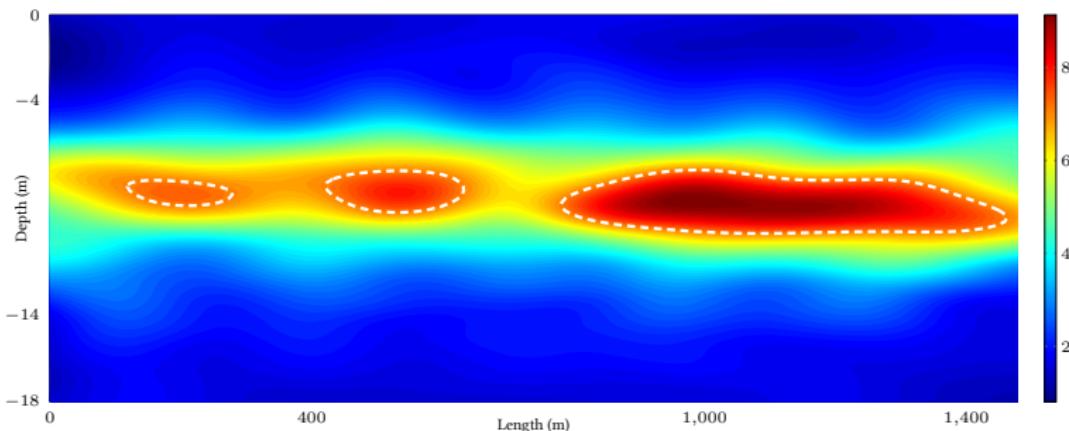
- ▶ What if we don't know h ?
- ▶ Implicitly define the threshold level: $h = \omega \max_{\mathbf{x} \in D} f(\mathbf{x})$, $0 < \omega < 1$

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

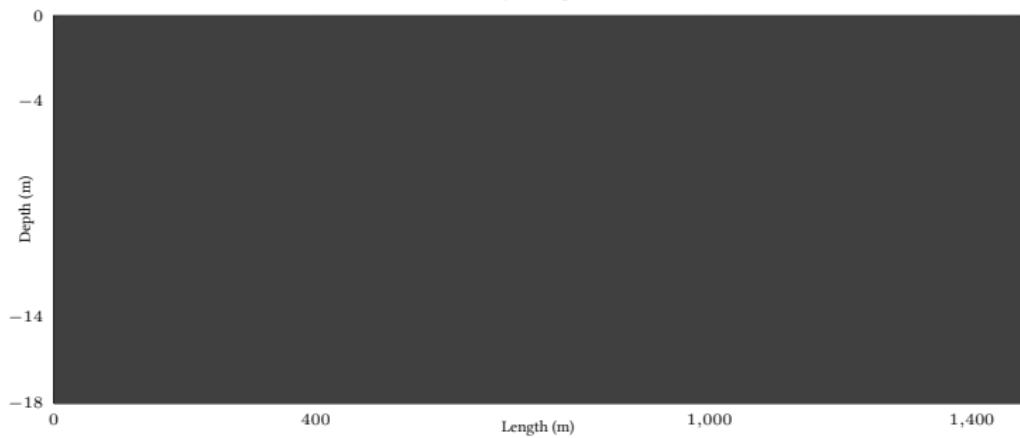
- ▶ What if we don't know h ?
- ▶ Implicitly define the threshold level: $h = \omega \max_{\mathbf{x} \in D} f(\mathbf{x})$, $0 < \omega < 1$
- ▶ LSE_{imp} :
 - ▶ Need to also estimate the maximum

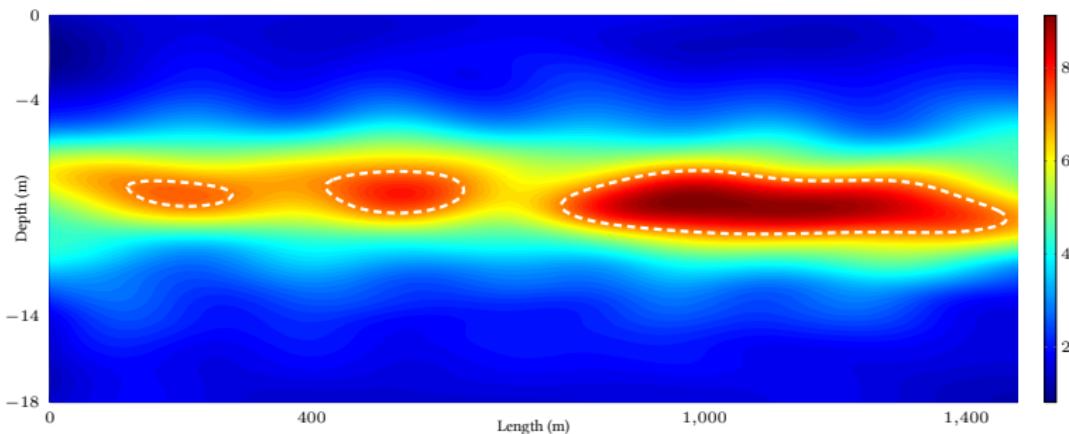
- ▶ What if we don't know h ?
- ▶ Implicitly define the threshold level: $h = \omega \max_{\mathbf{x} \in D} f(\mathbf{x})$, $0 < \omega < 1$
- ▶ LSE_{imp} :
 - ▶ Need to also estimate the maximum
 - ▶ Slower classification due to uncertainty about h



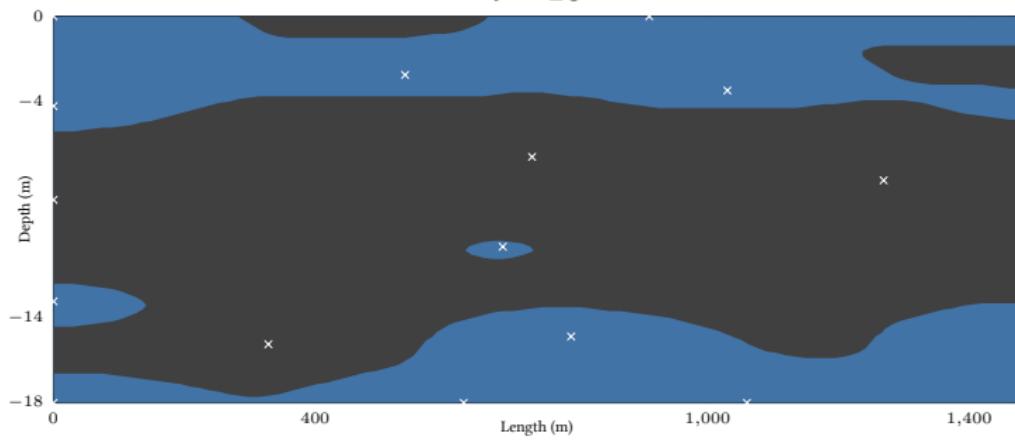


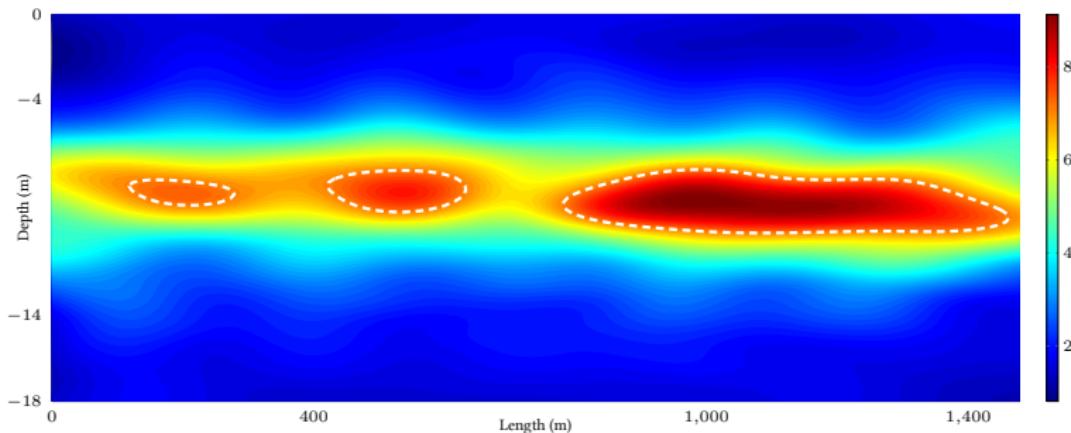
$t = 0$



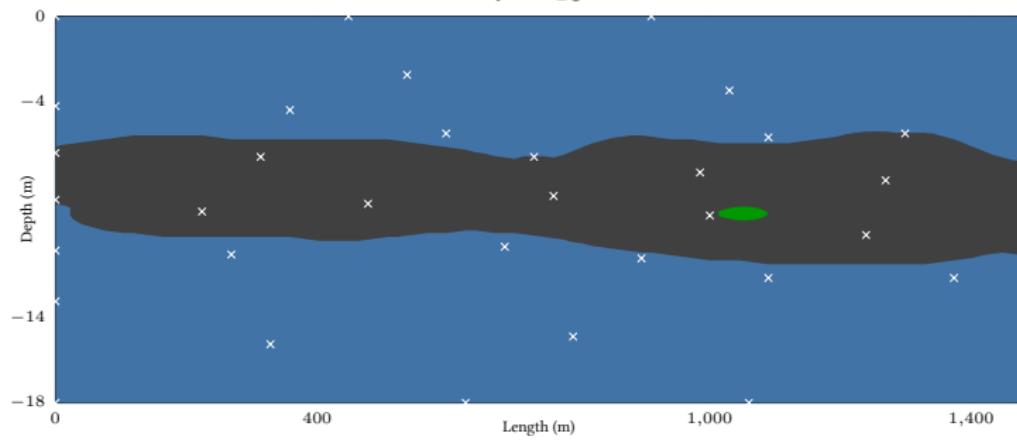


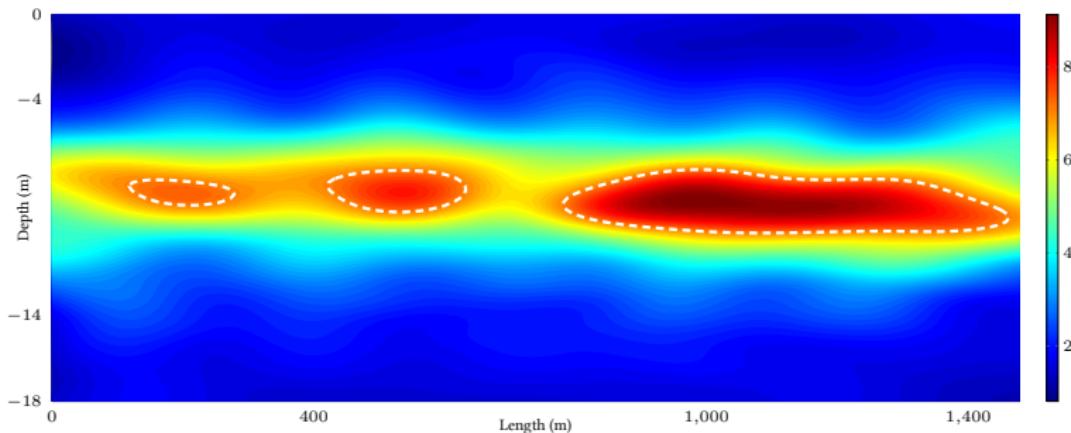
$t = 20$



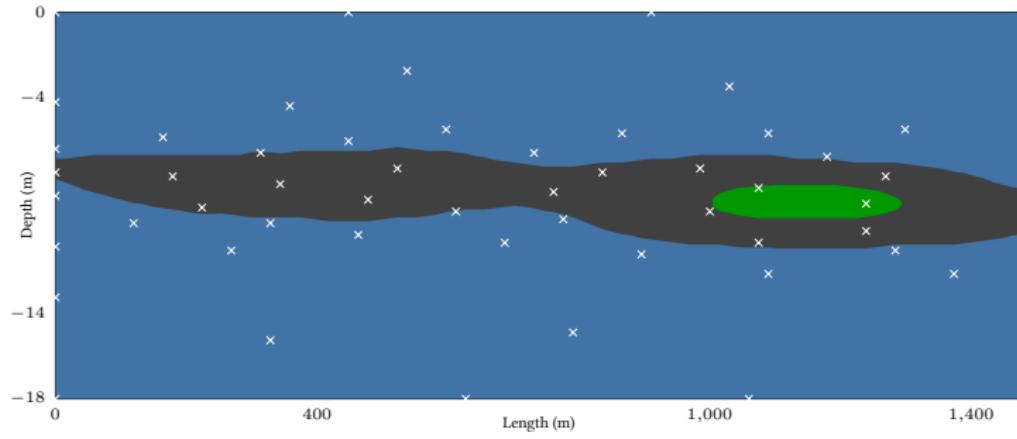


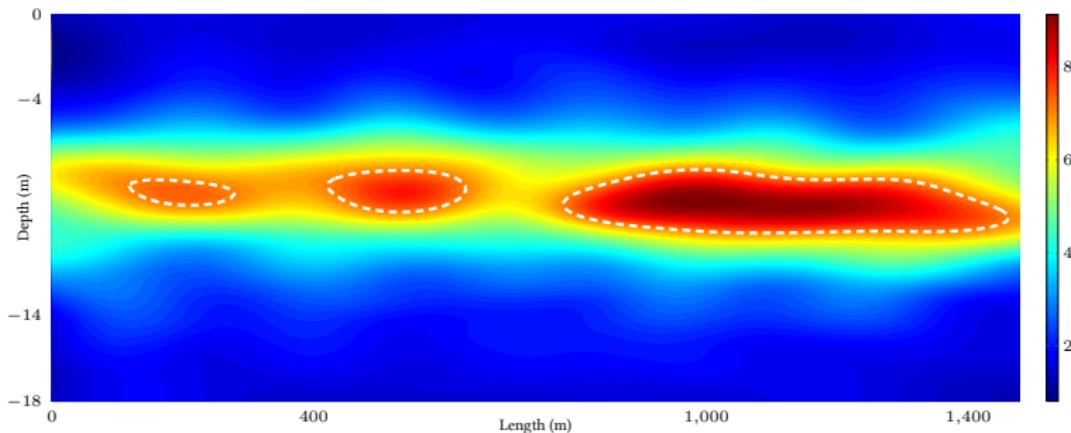
$t = 40$



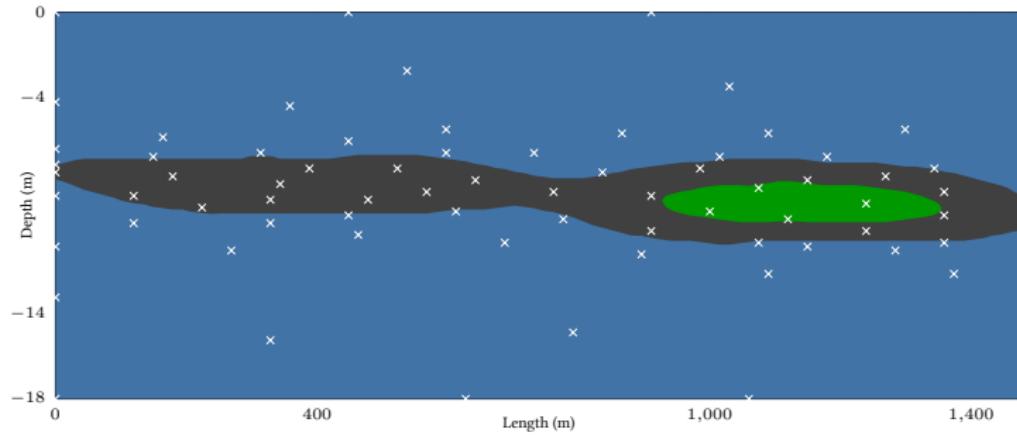


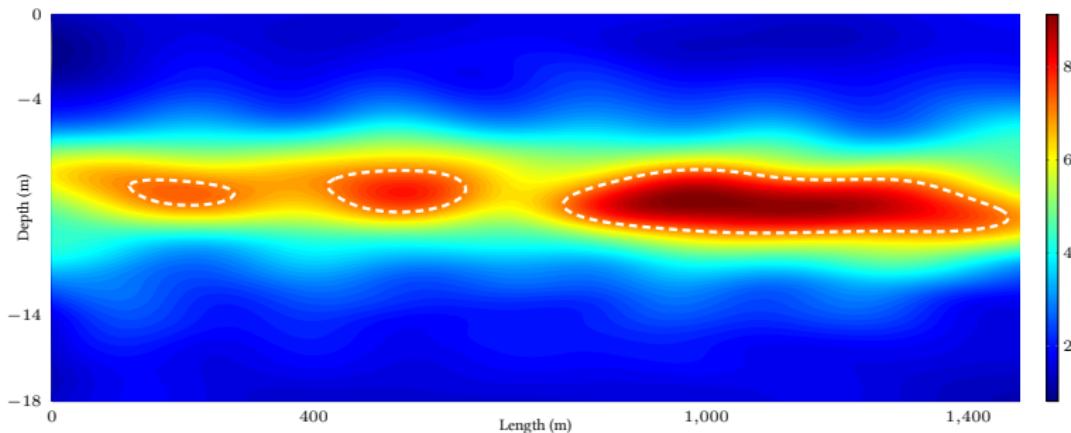
$t = 60$



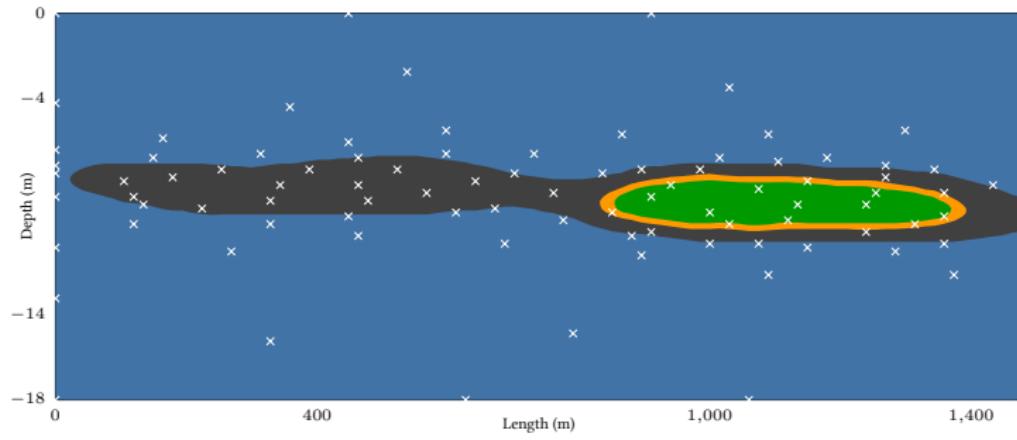


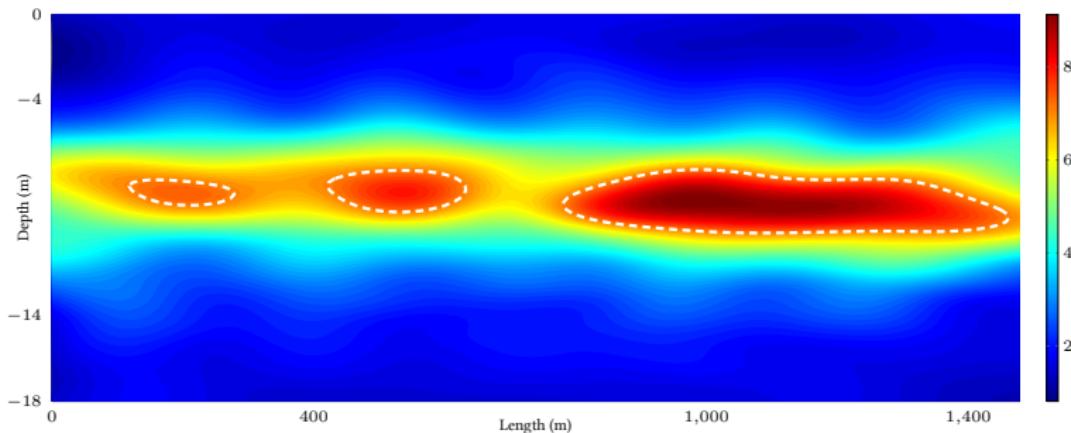
$t = 80$



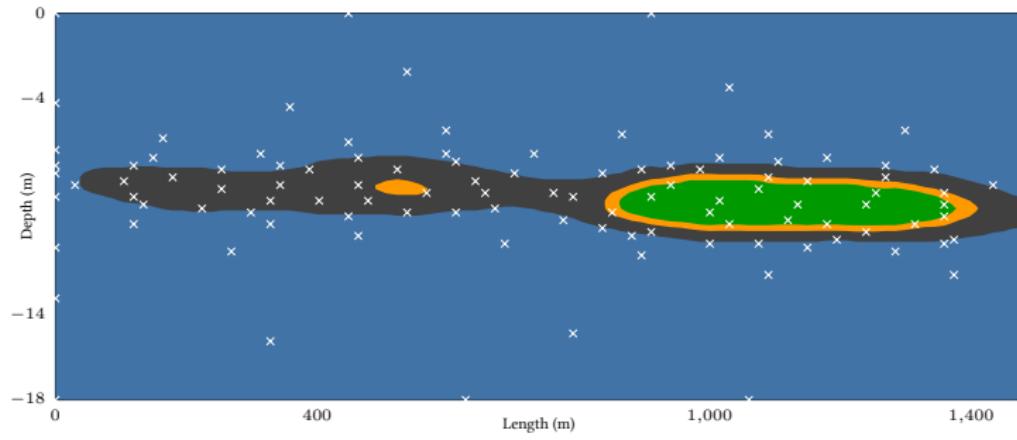


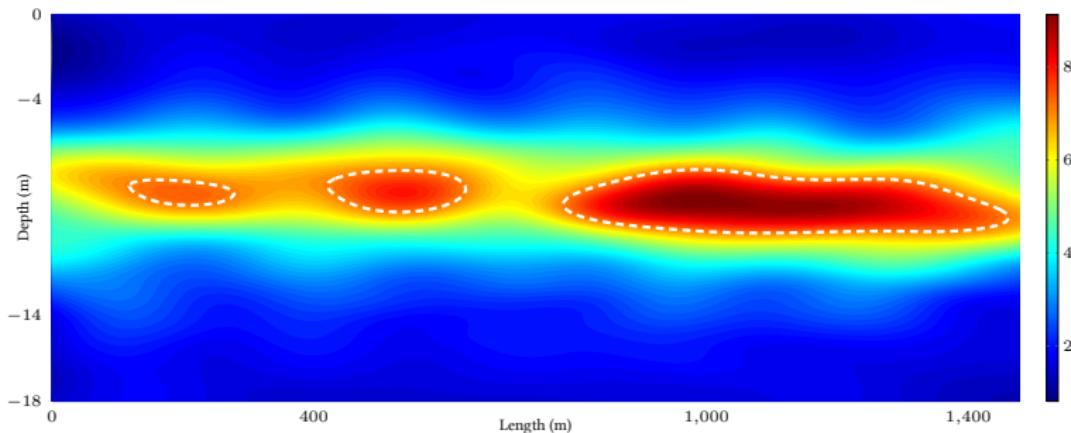
$t = 100$



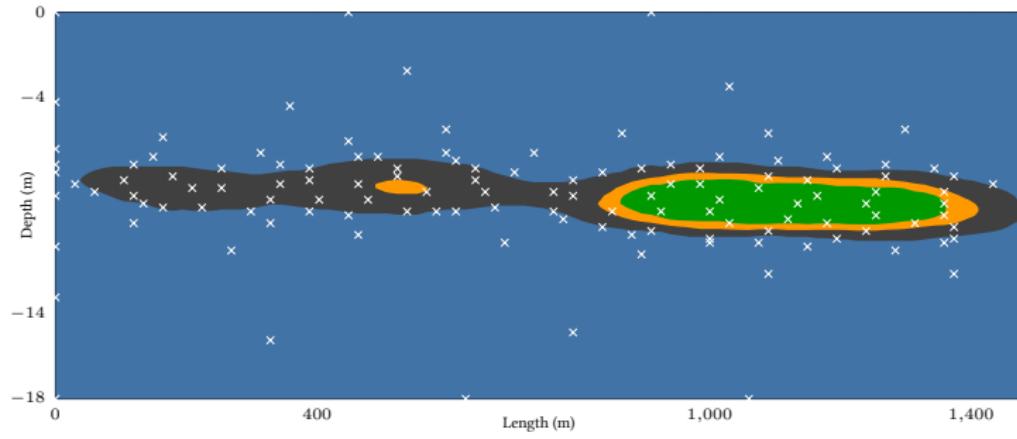


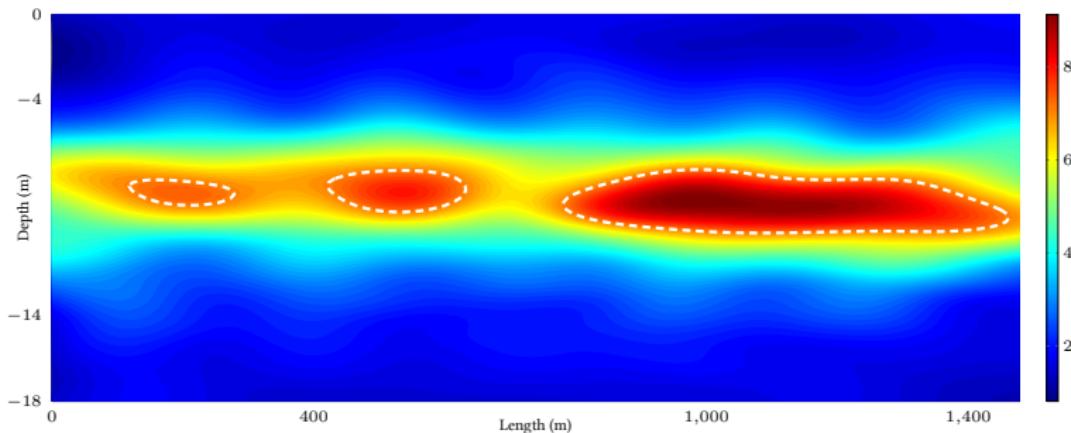
$t = 120$



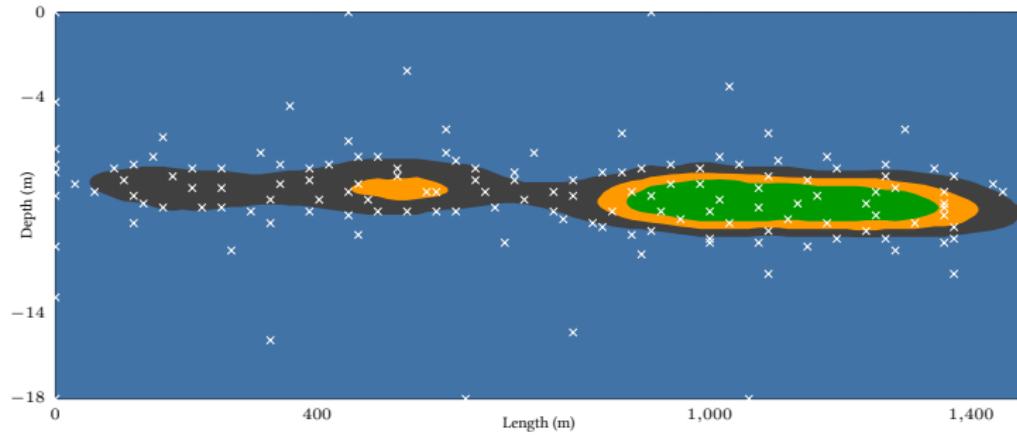


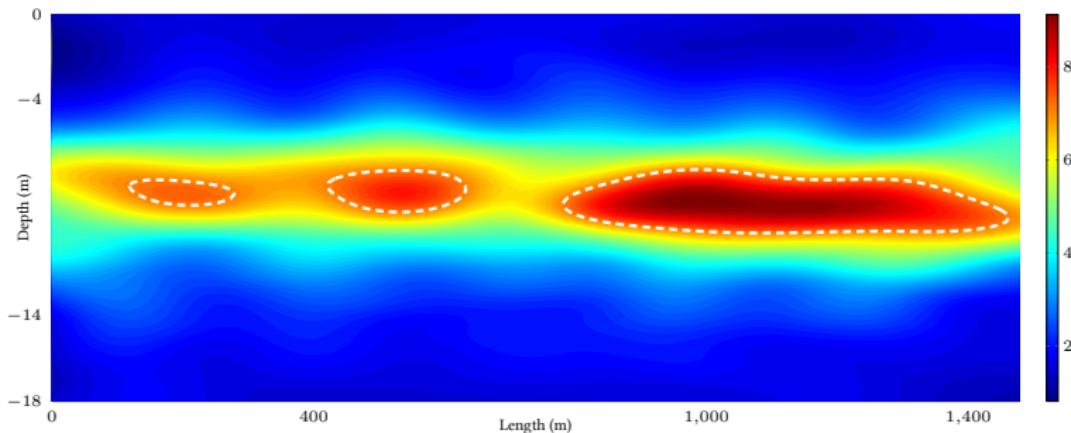
$t = 140$



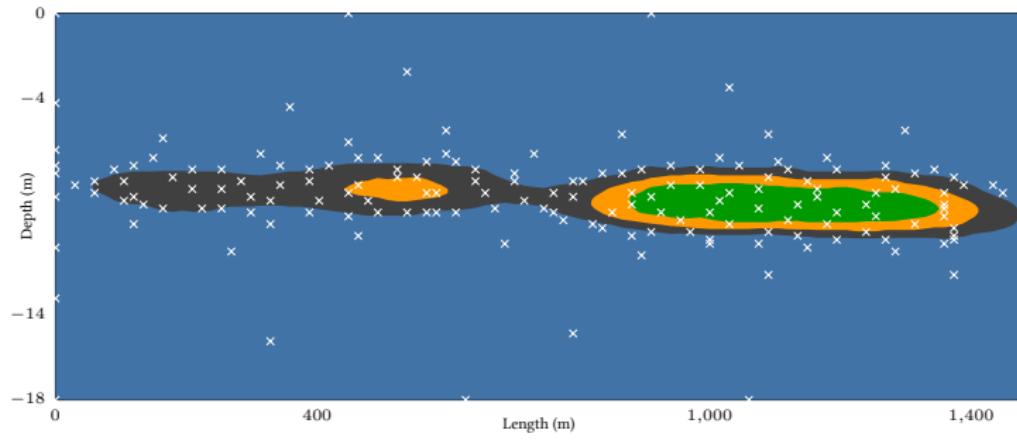


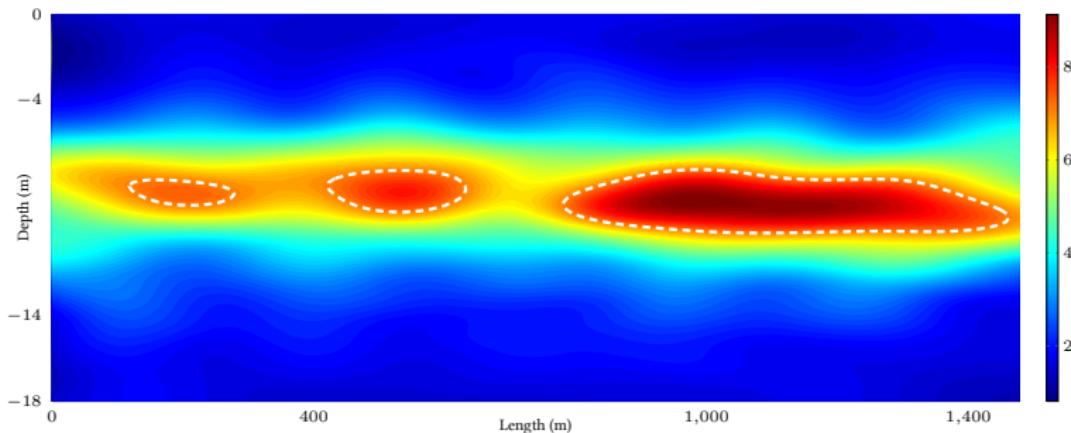
$t = 160$



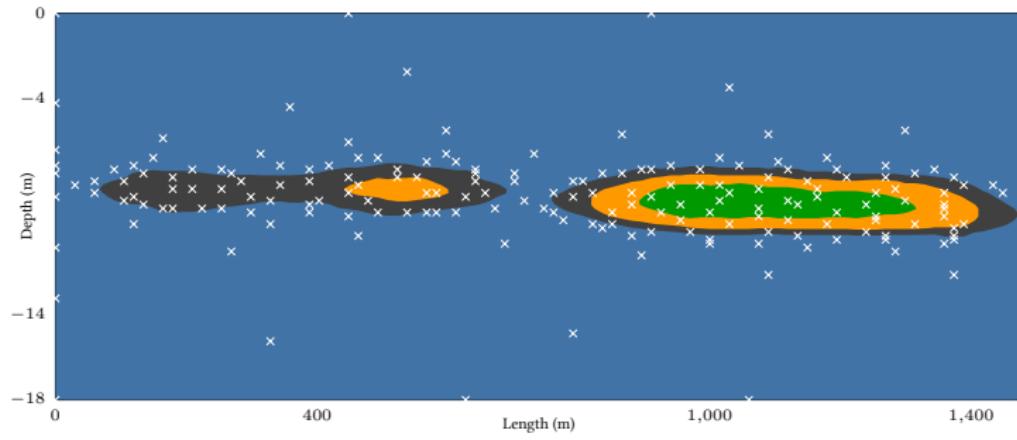


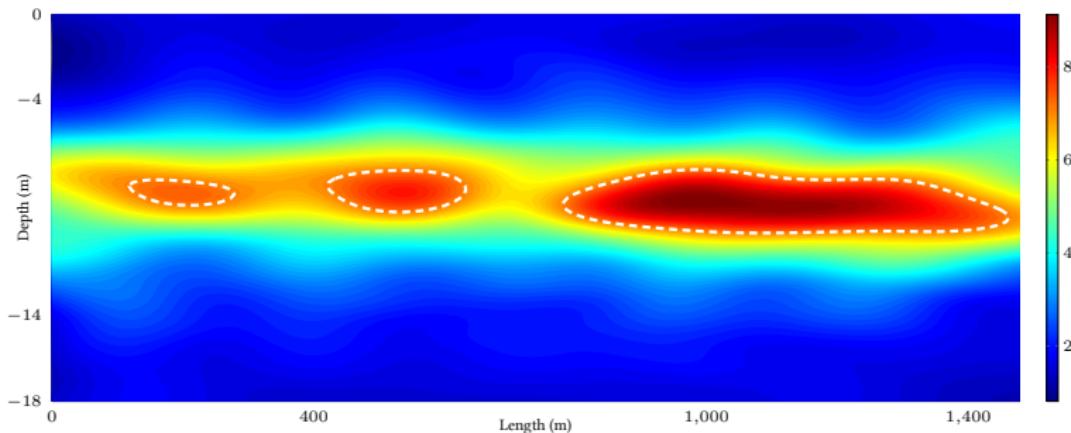
$t = 180$



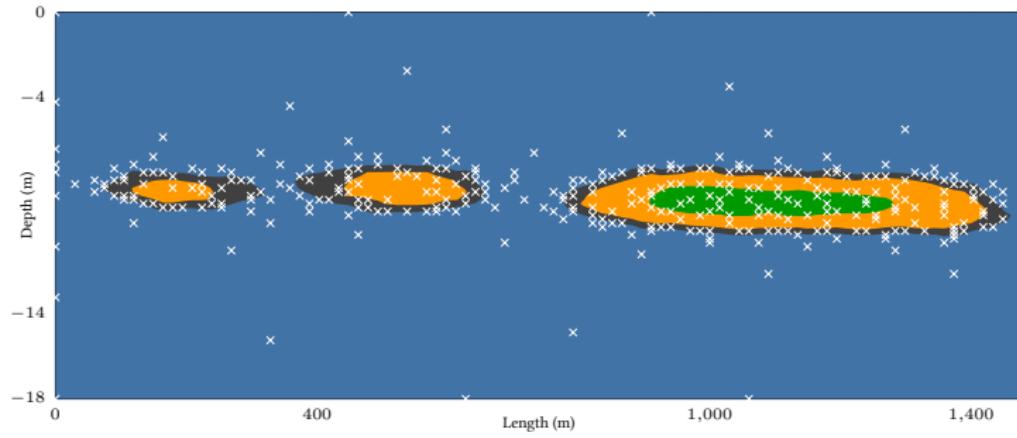


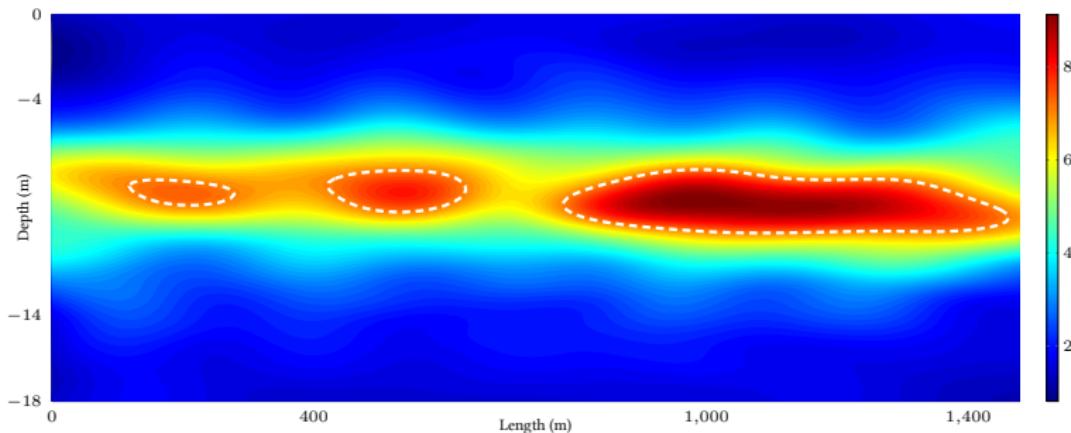
$t = 200$



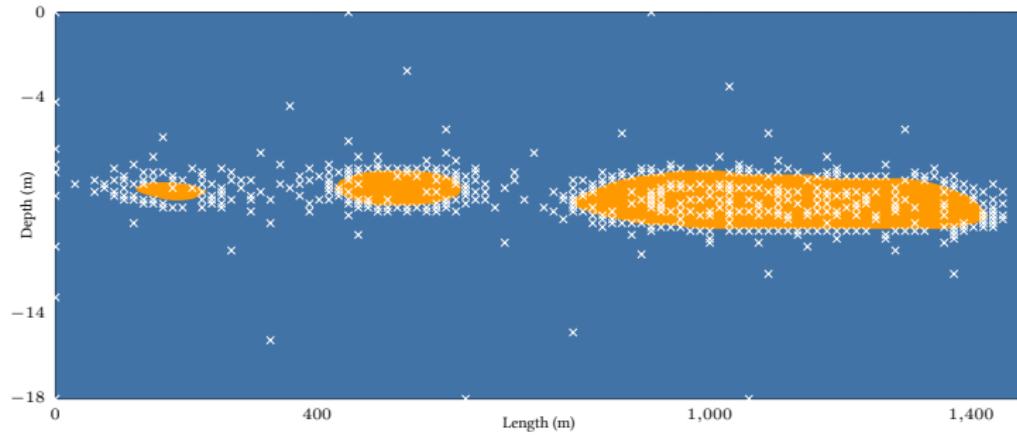


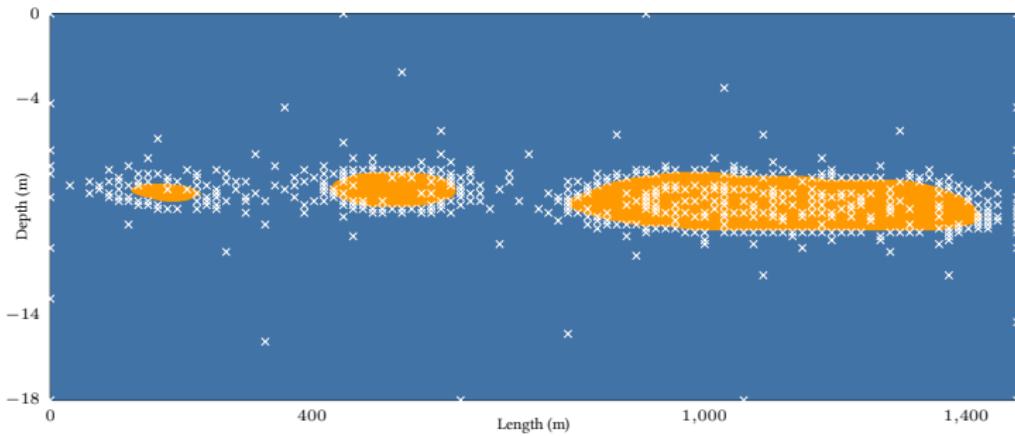
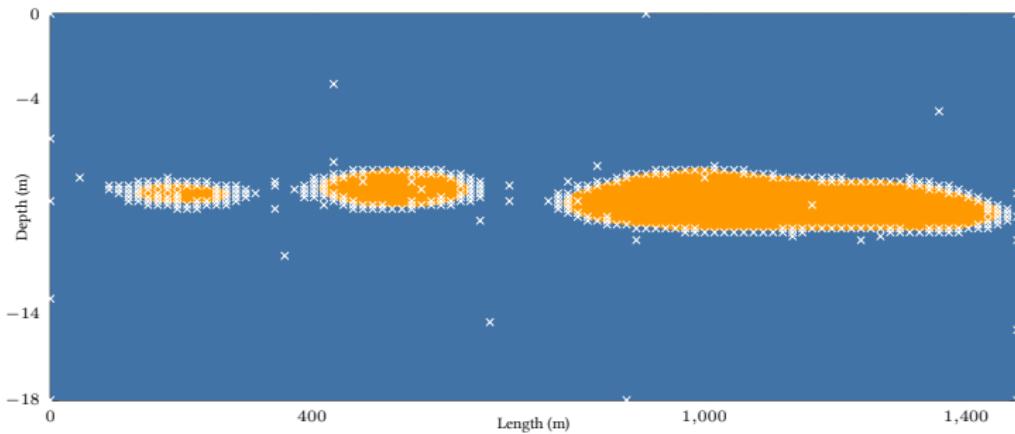
$t = 340$





$t = 486$



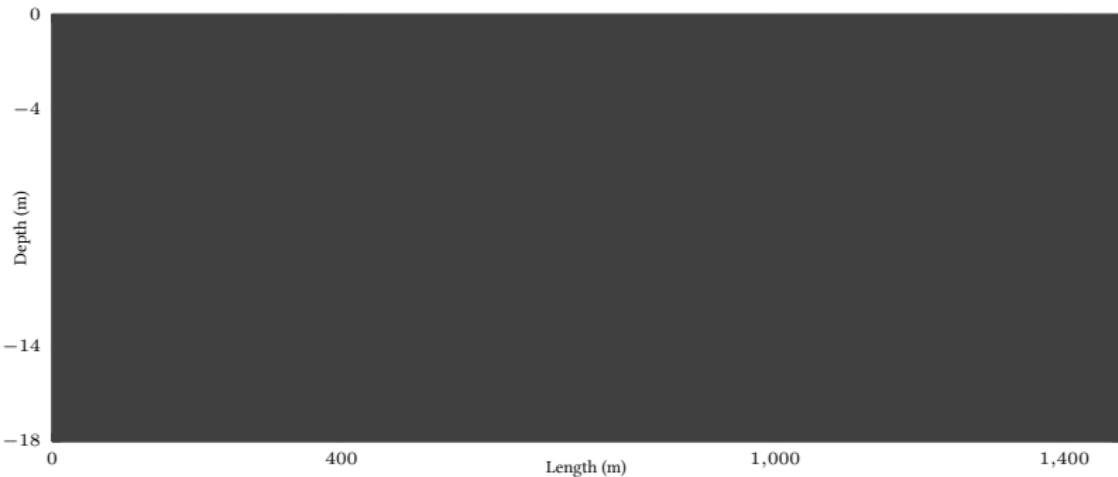


- ▶ Up to this point we have assumed a fixed cost per sample

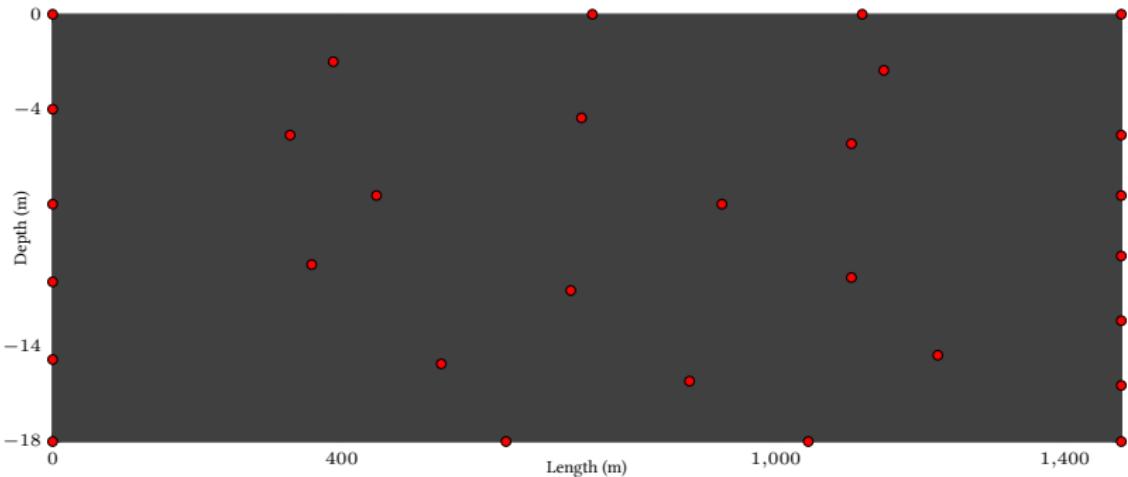
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?

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

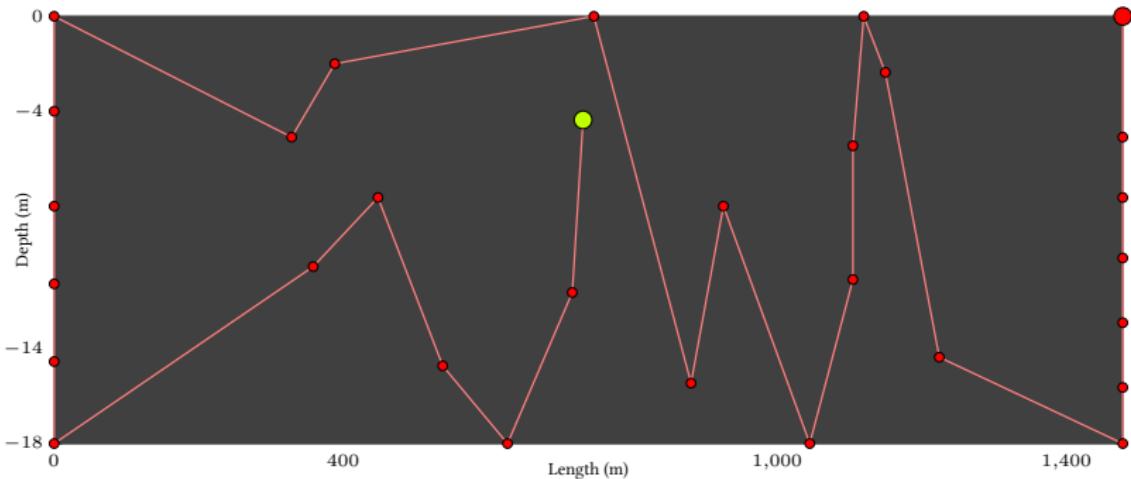
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



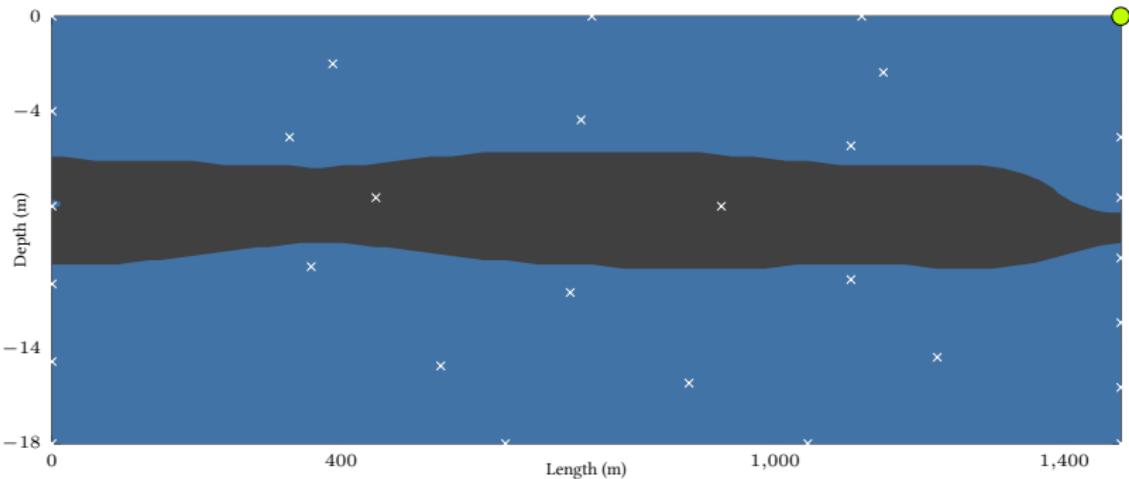
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



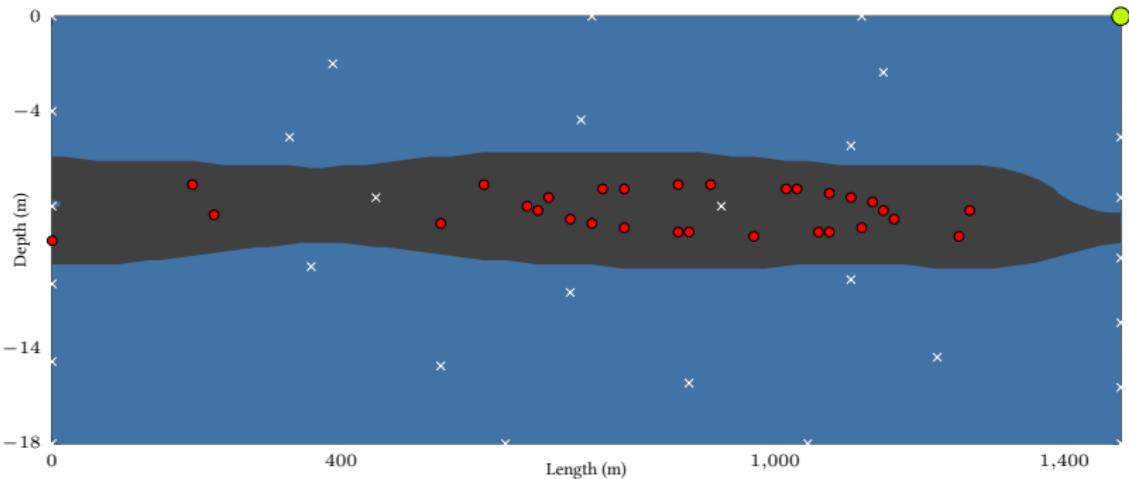
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



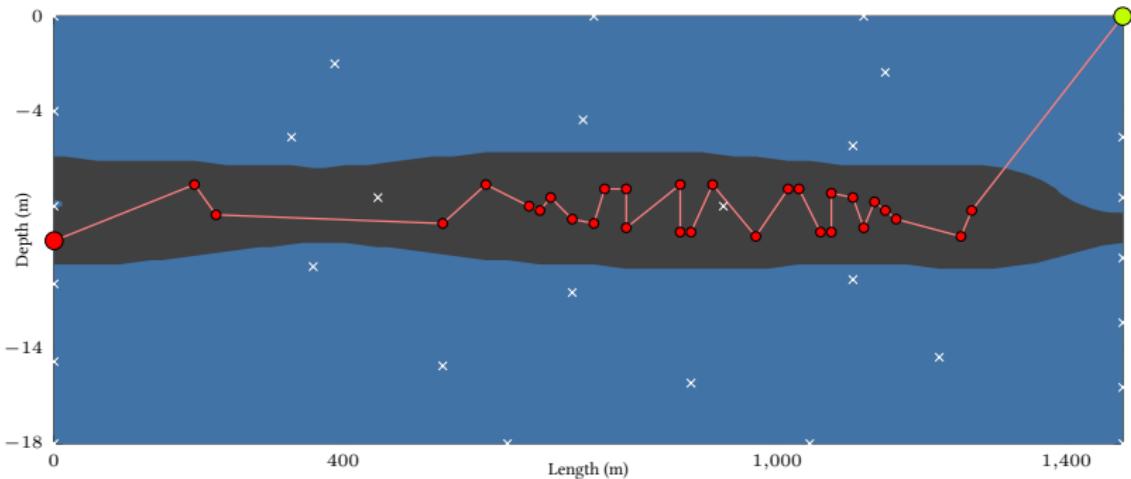
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



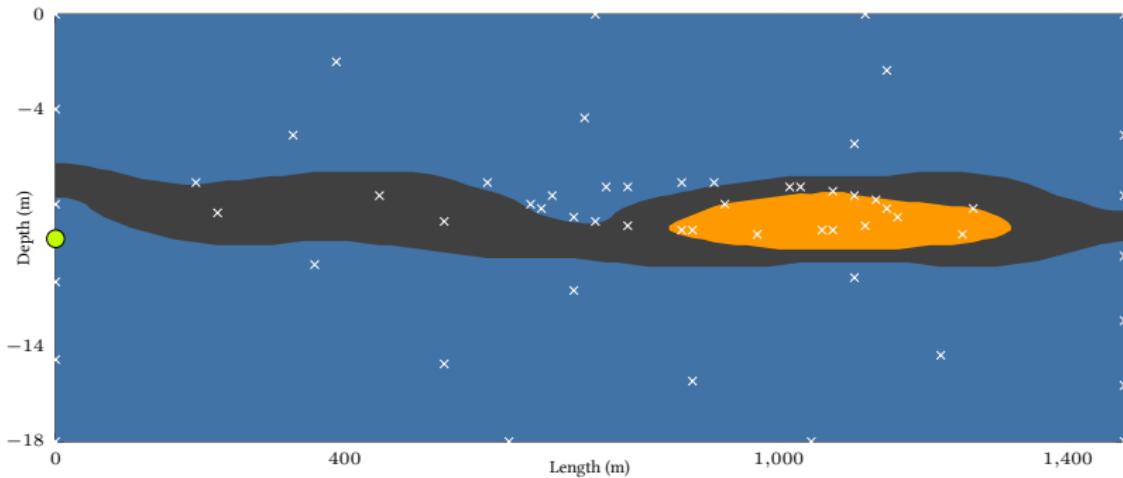
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



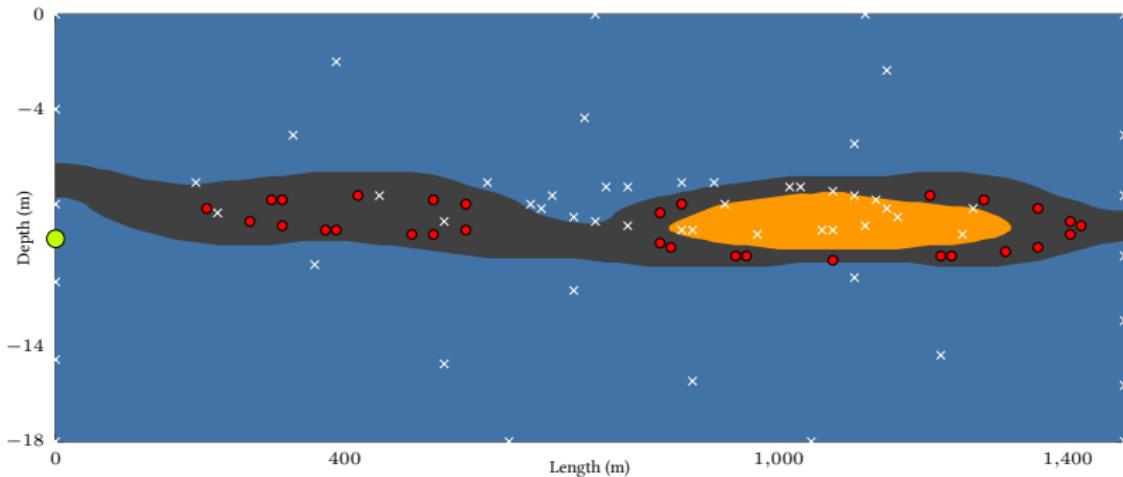
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



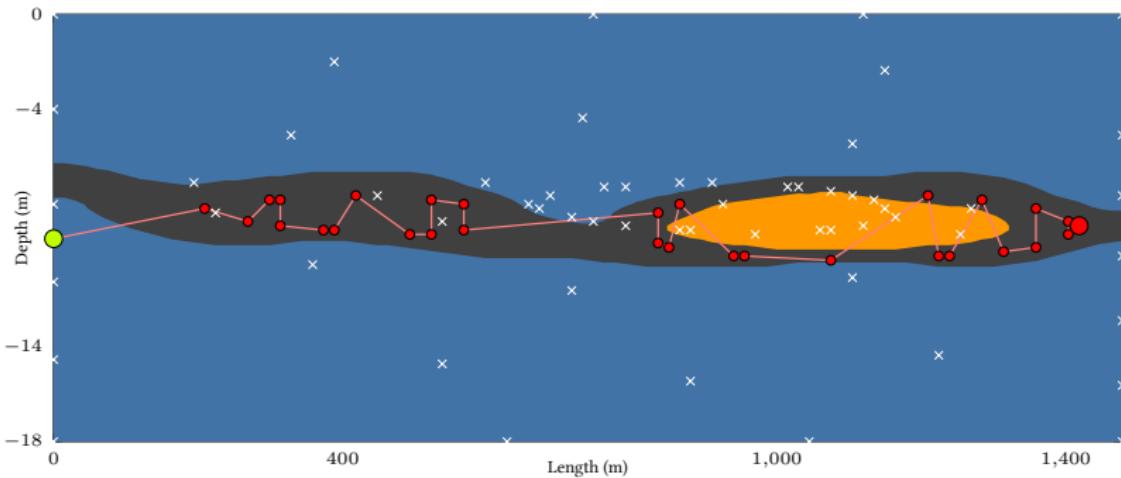
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



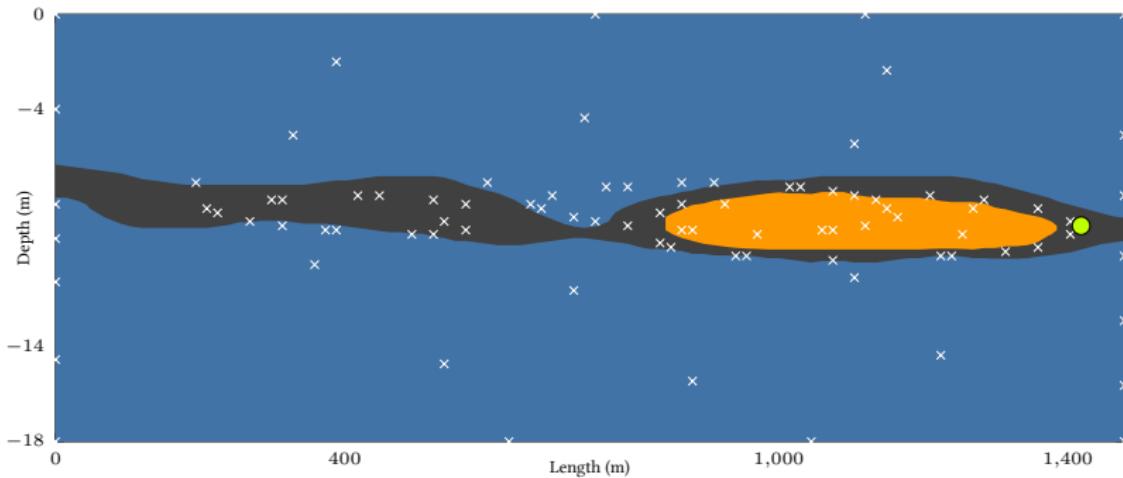
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



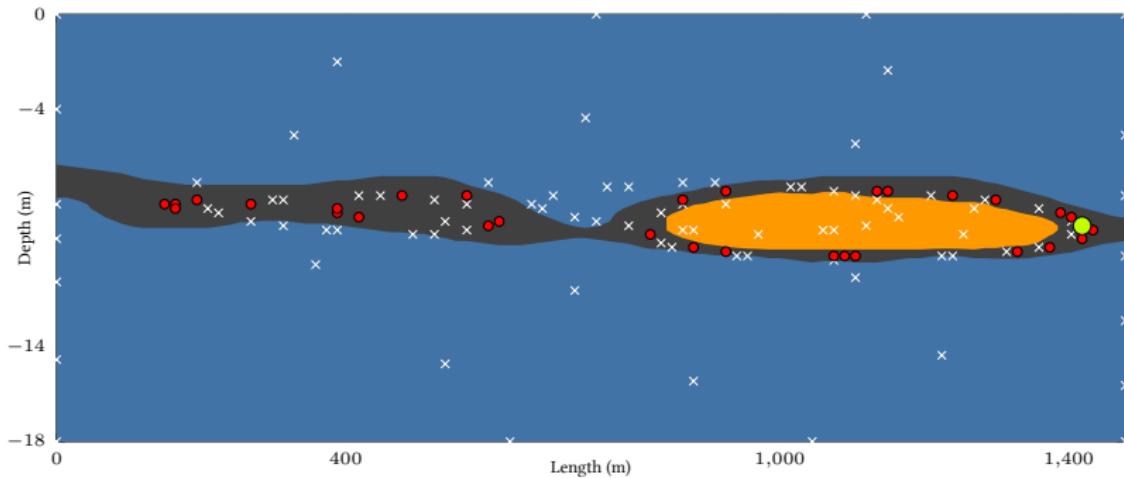
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



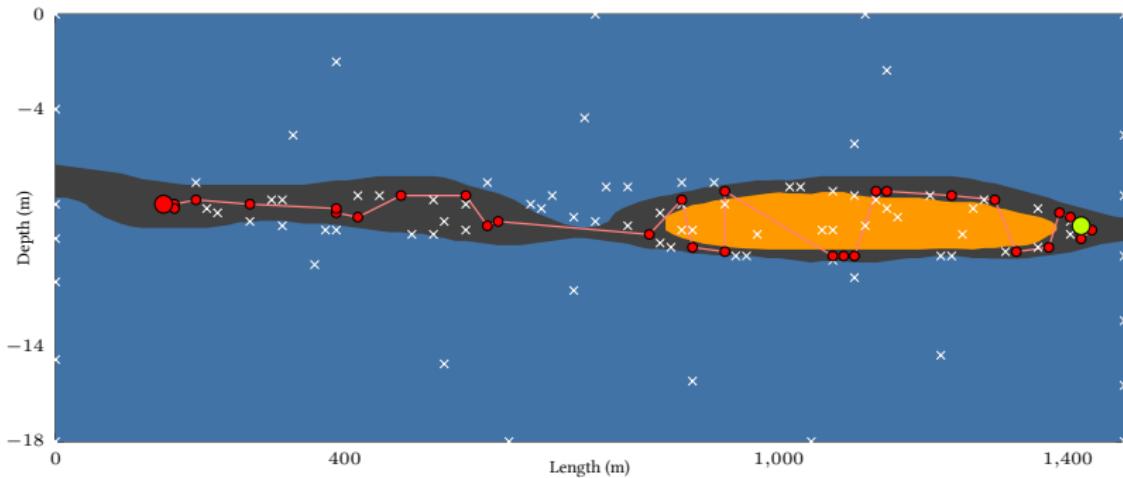
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



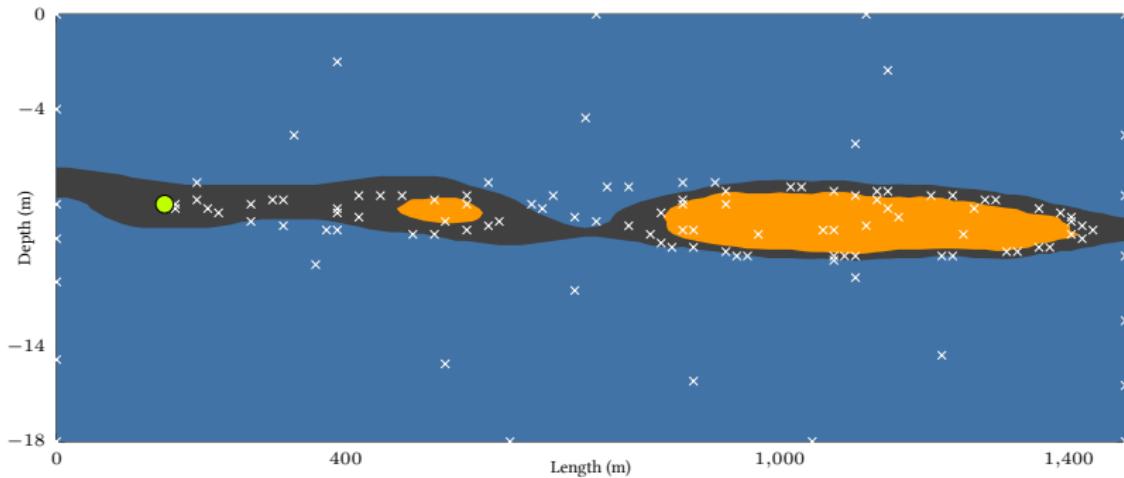
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



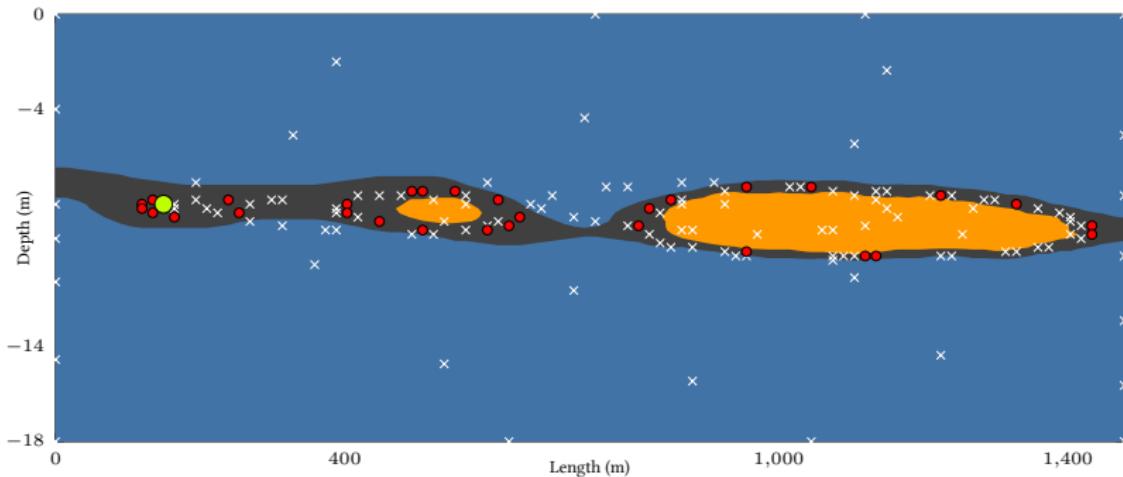
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



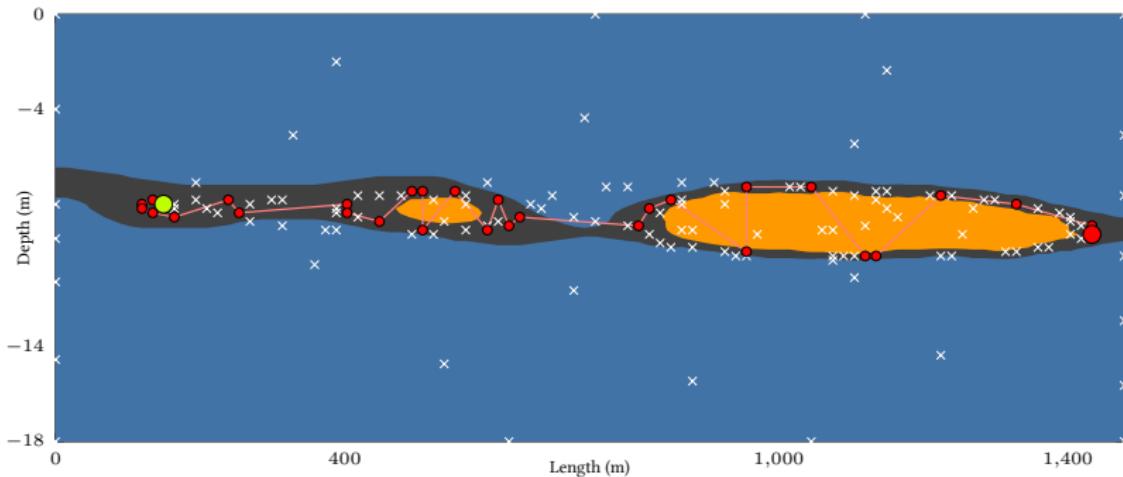
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



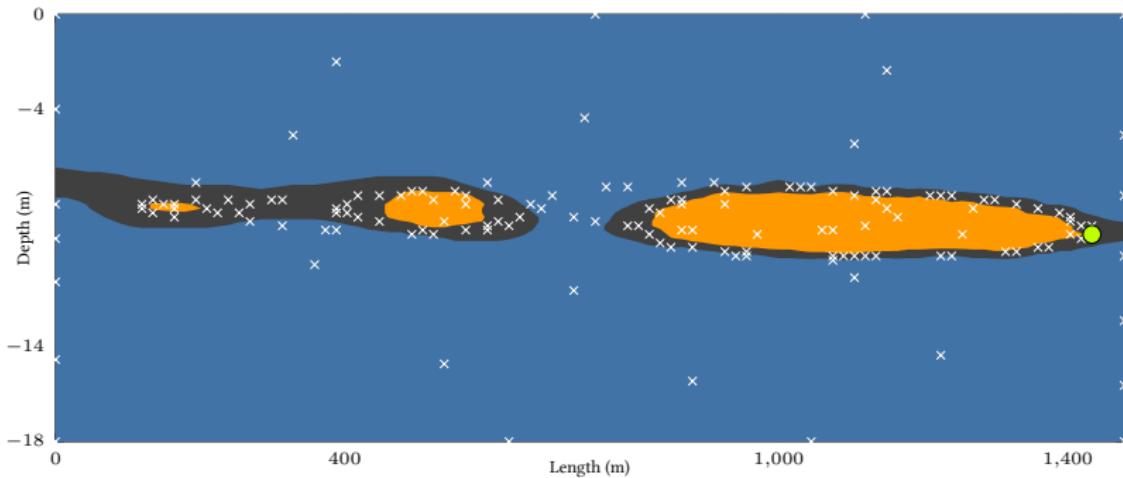
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



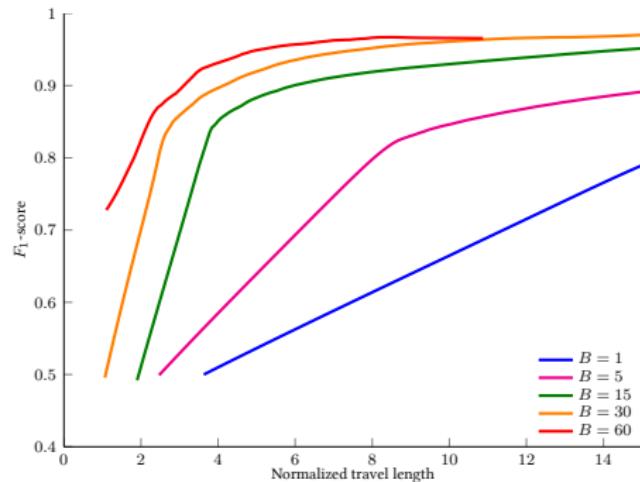
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ Plan ahead:
 - ▶ Select a *batch* of sampling locations ($\text{LSE}_{\text{batch}}$)
 - ▶ Connect them using a Euclidean TSP path
 - ▶ Traverse path and collect measurements



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



Geolocating internet latency

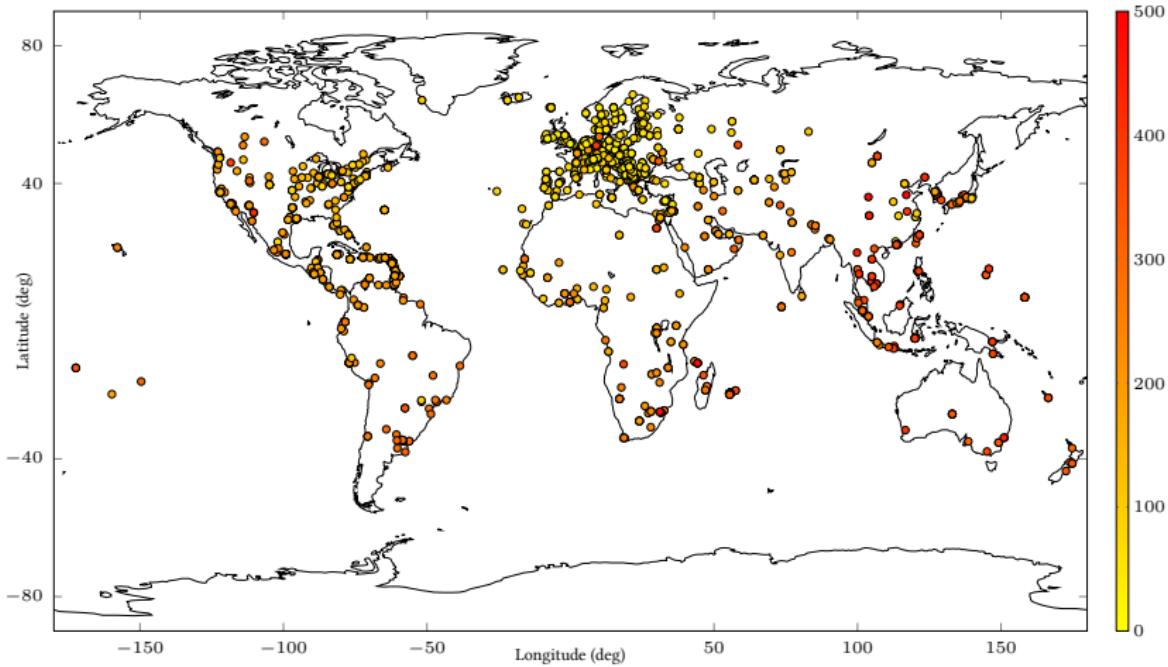
- ▶ Which world regions have low enough latency (e.g. < 150 ms) for video chatting, online gaming, etc.?

Geolocating internet latency

- ▶ Which world regions have low enough latency (e.g. < 150 ms) for video chatting, online gaming, etc.?
- ▶ Ideal setting for batch sampling → multiple pings in parallel

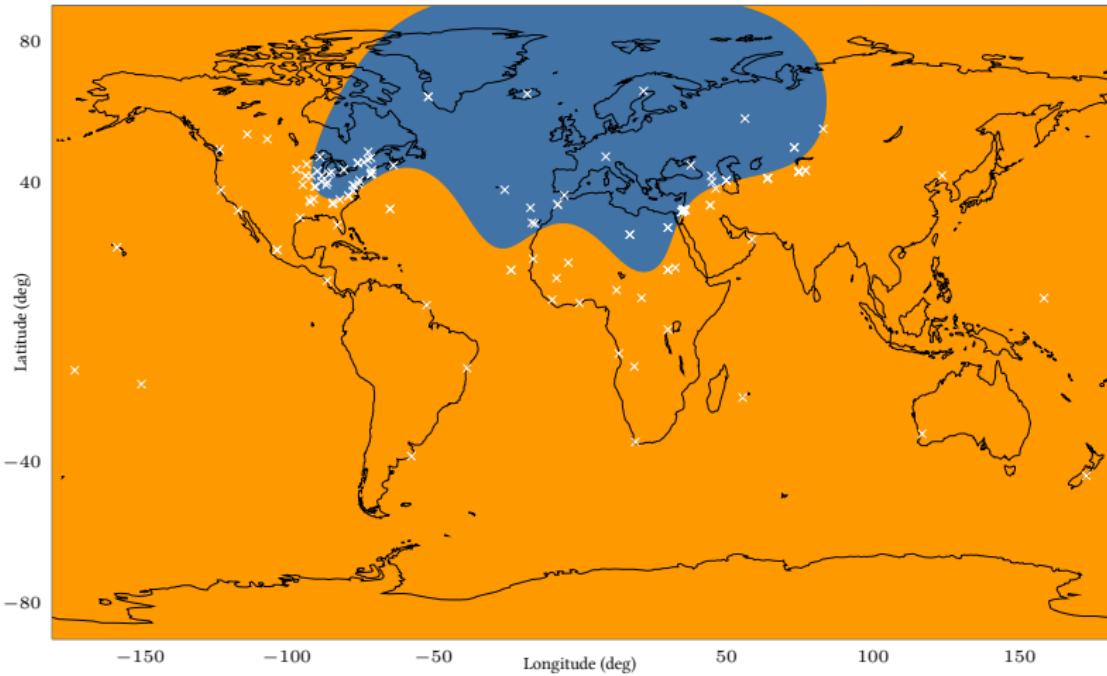
Geolocating internet latency

- ▶ Which world regions have low enough latency (e.g. < 150 ms) for video chatting, online gaming, etc.?
- ▶ Ideal setting for batch sampling → multiple pings in parallel



Geolocating internet latency

- ▶ Which world regions have low enough latency (e.g. < 150 ms) for video chatting, online gaming, etc.?
- ▶ Ideal setting for batch sampling → multiple pings in parallel



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 $\beta_0 = 2 \log(D\pi^2)^2/(6\delta)$, 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

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

Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

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

Theorem (Convergence of LSE)

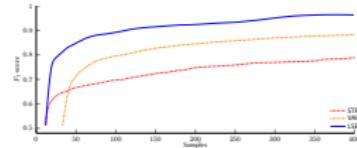
For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_0 = 2 \log(D\pi^2)^2/(6\delta)$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma \tau} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ -accurate solution, that is

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



Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

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

- ▶ Two useful extensions:

Theorem (Convergence of LSE)

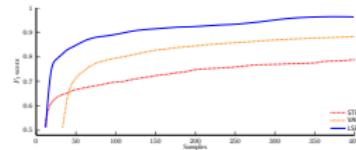
For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_0 = 2 \log(D\pi^2)^2 / (\delta h)$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma \tau} \geq \frac{C_1}{4e^2},$$

where $C_1 = 8/\log(1 + e^{-2})$.

Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ -accurate solution, that is

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



Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

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

- ▶ Two useful extensions:

- ▶ Implicit threshold level

Theorem (Convergence of LSE)

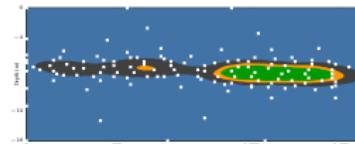
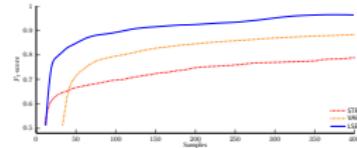
For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_0 = 2 \log(D\pi^2)^2/(6\delta)$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma \tau} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ -accurate solution, that is

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



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)

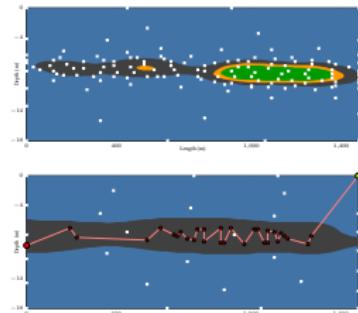
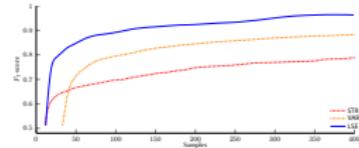
For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_0 = 2 \log(D\pi^2)^2/(6\delta)$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma \tau} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ -accurate solution, that is

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



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

- ▶ Look out for algae when swimming in Lake Zurich! 😊

Theorem (Convergence of LSE)

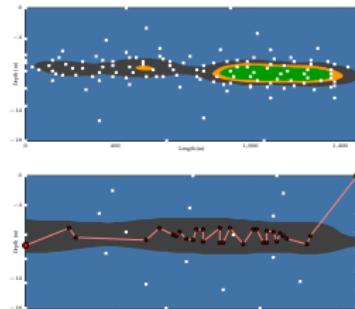
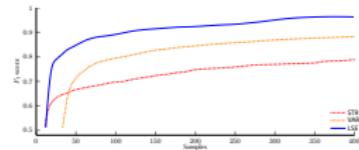
For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_0 = 2 \log(D\pi^2)^2 / (\delta h)$, LSE terminates after at most T iterations, where T is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Furthermore, with probability at least $1 - \delta$, the algorithm returns an ϵ -accurate solution, that is

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



Questions?

