Active Learning for Level Set Estimation

Alkis Gotovos\textsuperscript{1}  Nathalie Casati\textsuperscript{1,2}  Gregory Hitz\textsuperscript{1}  Andreas Krause\textsuperscript{1}

\textsuperscript{1}Department of Computer Science
ETH Zurich

\textsuperscript{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!

“[…]Switzerland’s Lake Zurich […] 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* 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

_Alkis Gotovos et al. (ETH Zurich)_

_Active Learning for Level Set Estimation_
Swimmers of Lake Zurich, beware!

“[…]Switzerland’s Lake Zurich […] an ideal environment for a population explosion of algae including *Planktothrix rubescens* […]”  
— *Scientific American*

“*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!

“[...]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
Take measurements on a vertical transect of the lake
Original algae concentration measurements (∼ 2000)

![Graph showing original algae concentration measurements.](image)

- **Length (m):** 0, 400, 800, 1200, 1600, 2000
- **Depth (m):** −18, −14, −10, −6, −2, 0

**Alkis Gotovos et al. (ETH Zurich)**

**Active Learning for Level Set Estimation**

**IJCAI ’13**
Interpolated algae concentration field

Length (m)  Depth (m)
Focus on accurately estimating regions of “high” concentration (e.g. $\geq 7$)
Classify transect into a **super**- and a **sub**level 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 ($x_t \in D$)
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 \((x_t \in D)\)
- 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 ($x_t \in D$)
- Obtain noisy observation ($y_t = f(x_t) + n_t$)
- 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 $C(x)$
Gaussian processes

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

\[ \mu(x) + \beta \sigma(x) \]
\[ \mu(x) \]
\[ \mu(x) - \beta \sigma(x) \]
Gaussian processes

- Mean **and variance** estimates: construct confidence intervals $C(x)$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- Mean and variance estimates: construct confidence intervals $C(x)$
- Bayesian, yet efficient: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- **Mean and variance** estimates: construct confidence intervals $C(x)$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- Mean and variance estimates: construct confidence intervals $C(x)$
- **Bayesian**, yet efficient: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- Mean and variance estimates: construct confidence intervals $C(x)$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- Mean **and variance** estimates: construct confidence intervals $C(x)$
- **Bayesian**, yet **efficient**: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
Gaussian processes

- **Mean and variance** estimates: construct confidence intervals $C(x)$
- **Bayesian, yet efficient**: suitable for step-by-step updates
- Impose prior “smoothness” assumptions via kernel $k(x, x')$
1. How do we estimate the function?
1. How do we **estimate** the function?
1. How do we **estimate** the function? 

---

Alkis Gotovos et al. (ETH Zurich)  Active Learning for Level Set Estimation  IJCAI '13  9 / 21
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**?

---

C(1)

(Credit: Alkis Gotovos et al. (ETH Zurich))
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...

---

**Active Learning for Level Set Estimation**

Alkis Gotovos et al. (ETH Zurich)
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

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

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

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

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

← classify

← sample
The Level Set Estimation (LSE) algorithm

**Input:** sample space $D$, threshold level $h$

**Output:** predicted super- and sublevel sets

```plaintext
while ∃ unclassified points in $D$ do
  for all unclassified points $x \in D$ do
    if $C_t(x)$ lies above $h$ then
      classify $x$ into superlevel set
    else if $C_t(x)$ lies below $h$ then
      classify $x$ into sublevel set
    else
      leave $x$ unclassified
    end if
  end for

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

  $y_t \leftarrow f(x_t) + n_t$

  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

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

- **Monotonicity of**
  1. confidence intervals
  2. classification

- **Relax classification rules by an accuracy parameter $\epsilon$**
$t = 40$
Active Learning for Level Set Estimation

$t = 60$
$t = 160$
$t = 200$
Theorem (Convergence of LSE)

For any \( h \in \mathbb{R}, \delta \in (0, 1), \) and \( \epsilon > 0, \) if \( \beta_t = 2 \log(|D| \tau^2 \sigma^2 / (6\delta)) \), 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 \( \epsilon \)-accurate solution, that is

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

If we choose $\beta$ appropriately (large enough), then:

$I$ LSE terminates after a number of iterations $T$.

$T$ smoother kernel $T$

$\sigma$ " $T$

$\epsilon$ " $T$

The solution returned is $\epsilon$-accurate with high probability.
Theorem (Simplified)

If we choose $\beta$ appropriately (large enough), then:

- LSE terminates after a number of iterations $T$
Theorem (Simplified)

If we choose $\beta$ appropriately (large enough), then:

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

If we choose $\beta$ 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 $\beta$ 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. $\epsilon \uparrow \Rightarrow T \downarrow$

Alkis Gotovos et al. (ETH Zurich)  
Active Learning for Level Set Estimation  
IJCAI ’13 13 / 21
Theorem (Simplified)

If we choose $\beta$ appropriately (large enough), then:

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

- \textit{The solution returned is $\epsilon$-accurate with high probability}
Theorem (Simplified)

If we choose $\beta$ 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 $\epsilon$-accurate with high probability
Experiments

1. LSE
Experiments

1. LSE
2. Maximum variance sampling:
   \[ x_t = \arg\max_{x \in D} \sigma_{t-1}(x) \]
Experiments

1. LSE
2. Maximum variance sampling:
   \[ x_t = \arg\max_{x \in D} \sigma_{t-1}(x) \]
3. State of the art "straddle" heuristic (Bryan et al., 2005):
   \[ x_t \approx \arg\max_{x \in D} a_{t-1}(x) \quad \text{(for } \beta_t^{1/2} = 1.96) \]
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 $h = \omega \max_{x \in D} f(x), \ 0 < \omega < 1$
Implicit threshold level

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

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

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

- Implicitly define $h = \omega \max_{x \in D} f(x)$, $0 < \omega < 1$

- No existing algorithms for this problem (to our knowledge)

- We extend LSE (and its theory!) to this setting

- $\text{LSE}_{\text{imp}}$ 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 $h = \omega \max_{x \in D} f(x)$, $0 < \omega < 1$

- No existing algorithms for this problem (to our knowledge)

- We extend LSE (and its theory!) to this setting

- $\text{LSE}_{\text{imp}}$ algorithm:
  - $h$ is now an estimated quantity $\rightarrow$ modified classification rules
  - Need to accurately estimate the maximum $\rightarrow$ modified selection rule
Active Learning for Level Set Estimation

IJCAI '13 18 / 21
Alkis Gotovos et al. (ETH Zurich)  
Active Learning for Level Set Estimation

$t = 0$

- Length (m) vs. Depth (m) diagram showing the initial state of a level set estimation problem.
$t = 20$
Alkis Gotovos et al. (ETH Zurich)  Active Learning for Level Set Estimation  IJCAI ’13 18 / 21

$t = 60$
$t = 80$
$t = 100$
$t = 140$
$t = 180$
$t = 200$
$t = 340$
Active Learning for Level Set Estimation

$t = 486$
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 \textit{batch} of sampling locations at each step.

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

![Graph showing batch sampling locations and paths]

---

**Active Learning for Level Set Estimation**

*Alkis Gotovos et al. (ETH Zurich)*
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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ to select a batch of sampling locations
  - Connect them using a Euclidean TSP path
  - Traverse path and collect measurements

![Diagram of sampling locations and measurements](image)
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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ 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 $\text{LSE}_{\text{batch}}$ 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 $\varepsilon > 0$, if $\beta = 2 \log(1 + \varepsilon^2 / \delta)$, LSE terminates after at most $T$ iterations, where $T$ is the smallest positive integer satisfying

  \[
  \frac{T}{\beta_{TT}} \geq \frac{C_1}{4\varepsilon^2},
  \]

  where $C_1 = \frac{8}{\log(1 + \varepsilon^2)}$.

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

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

  **Competitive with the state of the art**

Look out for algae when swimming in Lake Zurich!
Summary

- **LSE algorithm:**
  - Theoretical guarantees
  - Competitive with the state of the art

  **Theorem (Convergence of LSE):**
  
  For any $h \in \mathbb{R}$, $\delta \in (0, 1)$, and $\epsilon > 0$, if $\beta_1 = 2 \log(\sqrt{D} \pi^2 \beta^2 / (6 \delta))$, LSE terminates after at most $T$ iterations, where $T$ is the smallest positive integer satisfying
  
  $$\frac{T}{\beta_1^{1/2}} \geq \frac{C_1}{4e^2},$$
  
  where $C_1 = 8 / \log(1 + \sigma^{-2})$.
  
  Furthermore, with probability at least $1 - \delta$, the algorithm returns an $\epsilon$-accurate solution, that is
  
  $$\Pr\left\{ \max_{x \in D} \epsilon(x) \leq \epsilon \right\} \geq 1 - \delta.$$

- **Two useful extensions:**
  - Implicit threshold level (LSE$_{imp}$)
  - Batch sampling (LSE$_{batch}$)

Look out for algae when swimming in Lake Zurich!
Summary

- **LSE algorithm:**

  **Theoretical guarantees**

  **Competitive with the state of the art**

  \[ \text{Theorem (Convergence of LSE)} \]

  For any \( h \in \mathbb{R}, \delta \in (0, 1), \text{ and } \epsilon > 0, \text{ if } \beta_1 = 2 \log(D \pi^2 \delta^2 / (6\delta)), \text{ LSE terminates after at most } T \text{ iterations, where } T = \frac{C_1}{\beta_1 + \epsilon}. \]

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

  Furthermore, with probability at least \( 1 - \delta, \) the algorithm returns an \( \epsilon \)-accurate solution, that is

  \[ \Pr \left\{ \max_{x \in D} \hat{f}_n(x) \leq \epsilon \right\} \geq 1 - \delta. \]

- **Two useful extensions:**

  - Implicit threshold level (LSE\textsubscript{imp})
  - Batch sampling (LSE\textsubscript{batch})

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