Distributed Statistical Algorithms

GÉRARD BIAU

UNIVERSITY PIERRE ET MARIE CURIE — INSTITUT UNIVERSITAIRE DE FRANCE

Gatsby, January 2015
Coauthors

Luc Devroye

Ryad Zenine
Outline

1 Need for distributed computing
2 Distributed and asynchronous estimation
3 Implementation and numerical studies
4 Cellular tree classifiers
Outline

1. Need for distributed computing
2. Distributed and asynchronous estimation
3. Implementation and numerical studies
4. Cellular tree classifiers
Parallel and distributed computing

→ Massive data challenges.
Parallel and distributed computing

→ Massive data challenges.

→ An increasing necessity of robustness and fault tolerance.
Parallel and distributed computing

→ Massive data challenges.

→ An increasing necessity of robustness and fault tolerance.

→ Advent of sensor, wireless and peer-to-peer networks, which must process data cooperatively.
The big data era

- Many modern datasets are so large and complex that they are impossible to process using classical tools.
The big data era

• Many modern datasets are so large and complex that they are impossible to process using classical tools.

• Megabytes and gigabytes are old-fashioned.
The big data era

- Many modern datasets are so large and complex that they are impossible to process using classical tools.

- Megabytes and gigabytes are old-fashioned.

- The learning infrastructure must be flexible enough to quickly accommodate gigantic sizes and uneven workloads.
The big data era

• Many modern datasets are so large and complex that they are impossible to process using classical tools.

• Megabytes and gigabytes are old-fashioned.

• The learning infrastructure must be flexible enough to quickly accommodate gigantic sizes and uneven workloads.

• This calls for parallel or distributed solutions.
Advantage of distributed computing

→ Distributing data across nodes in a cluster of computers.
Advantage of distributed computing

→ Distributing data across nodes in a cluster of computers.

→ Breaking work into tasks in parallel by nodes or processors.
Advantage of distributed computing

→ Distributing data across nodes in a cluster of computers.

→ Breaking work into tasks in parallel by nodes or processors.

→ Hardware is increasingly affordable.
Advantage of distributed computing

→ Distributing data across nodes in a cluster of computers.

→ Breaking work into tasks in parallel by nodes or processors.

→ Hardware is increasingly affordable.

→ MapReduce and Hadoop ecosystems.
Advantage of distributed computing

→ Distributing data across nodes in a cluster of computers.

→ Breaking work into tasks in parallel by nodes or processors.

→ Hardware is increasingly affordable.

→ MapReduce and Hadoop ecosystems.
Online learning

- In various applications, data are acquired sequentially.
Online learning

• In various applications, data are acquired **sequentially**.

• They must be efficiently processed in **real-time**.
Online learning

- In various applications, data are acquired **sequentially**.

- They must be efficiently processed in **real-time**.

- A promising way is to deal with **decentralized distributed systems**.
Online learning

• In various applications, data are acquired \textit{sequentially}.

• They must be efficiently processed in \textit{real-time}.

• A promising way is to deal with \textit{decentralized distributed} systems.

• Designing and analyzing distributed online learning algorithms poses several \textit{mathematical} and \textit{computational} challenges.
Distributed and asynchronous computation

- Starting point: Distributed gradient-type optimization algorithms.
Distributed and asynchronous computation

- Starting point: Distributed gradient-type optimization algorithms.
Distributed and asynchronous computation

- Starting point: Distributed gradient-type optimization algorithms.

- We develop a consensus-based asynchronous distributed solution for nonparametric online regression.
Outline

1. Need for distributed computing
2. Distributed and asynchronous estimation
3. Implementation and numerical studies
4. Cellular tree classifiers
Mathematical setting

• A generic pair $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$, with $\mathbb{E}Y^2 < \infty$. 
Mathematical setting

- A generic pair \((X, Y) \in \mathbb{R}^d \times \mathbb{R}\), with \(\mathbb{E} Y^2 < \infty\).

- Goal: Predict \(Y\) by assessing the regression function
  \[ r(x) = \mathbb{E}[Y|X = x]. \]
Mathematical setting

- A generic pair $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$, with $\mathbb{E}Y^2 < \infty$.

- Goal: Predict $Y$ by assessing the regression function

  $$r(x) = \mathbb{E}[Y|X = x].$$

- Architecture: A set $\{1, \ldots, M\}$ of processors.
Mathematical setting

- A generic pair \((X, Y) \in \mathbb{R}^d \times \mathbb{R}\), with \(\mathbb{E}Y^2 < \infty\).

- **Goal:** Predict \(Y\) by assessing the regression function
  \[
  r(x) = \mathbb{E}[Y | X = x].
  \]

- **Architecture:** A set \(\{1, \ldots, M\}\) of processors.

- **Processor** \(i\) sequentially receives the i.i.d. sequence
  \[
  (X_{i1}, Y_{i1}), (X_{i2}, Y_{i2}), \ldots, (X_{it}, Y_{it}), (X_{it+1}, Y_{it+1}), \ldots
  \]
Mathematical setting

• A generic pair \((X, Y) \in \mathbb{R}^d \times \mathbb{R}\), with \(\mathbb{E} Y^2 < \infty\).

• Goal: Predict \(Y\) by assessing the regression function

\[ r(x) = \mathbb{E}[Y|X = x]. \]

• Architecture: A set \(\{1, \ldots, M\}\) of processors.

• Processor \(i\) sequentially receives the i.i.d. sequence

\[ (X_i^1, Y_i^1), (X_i^2, Y_i^2), \ldots, (X_i^t, Y_i^t), (X_i^{t+1}, Y_i^{t+1}), \ldots \]

• ... and computes online its estimate \(r_t^i(x)\) of \(r(x)\).
$$r^1_t(x)$$

$$r^2_t(x)$$

$$r^3_t(x)$$

$$r^4_t(x)$$

Time

$t_1$

$t_2$

$t_3$

Data

Data

Data

Data
Révész-type recursive estimate (1977)

- General form:

\[
\begin{align*}
  r_1(x) &= Y_1 \\
  r_{t+1}(x) &= r_t(x) (1 - \varepsilon_{t+1} K_{t+1}(x, X_{t+1})) + \varepsilon_{t+1} Y_{t+1} K_{t+1}(x, X_{t+1}) \quad \text{for } t \geq 1,
\end{align*}
\]
Révész-type recursive estimate (1977)

- General form:

\[
\begin{align*}
 r_1(x) &= Y_1 \\
 r_{t+1}(x) &= r_t(x)(1 - \varepsilon_{t+1} K_{t+1}(x, X_{t+1})) + \varepsilon_{t+1} Y_{t+1} K_{t+1}(x, X_{t+1}) & \text{for } t \geq 1,
\end{align*}
\]

where

- \((K_t(\cdot, \cdot))_{t \geq 1}\) are nonnegative and symmetric functions
Révész-type recursive estimate (1977)

- General form:

\[
\begin{align*}
    r_1(x) &= Y_1 \\
    r_{t+1}(x) &= r_t(x)(1 - \varepsilon_{t+1}K_{t+1}(x, X_{t+1})) + \varepsilon_{t+1}Y_{t+1}K_{t+1}(x, X_{t+1}) \quad \text{for } t \geq 1,
\end{align*}
\]

where

- \((K_t(\cdot, \cdot))_{t \geq 1}\) are nonnegative and symmetric functions
- and \((\varepsilon_t)_{t \geq 1}\) are positive real parameters.
Révész-type recursive estimate (1977)

- General form:

\[
\begin{align*}
    r_1(x) &= Y_1 \\
    r_{t+1}(x) &= r_t(x) (1 - \varepsilon_{t+1} K_{t+1}(x, X_{t+1})) + \varepsilon_{t+1} Y_{t+1} K_{t+1}(x, X_{t+1}) \quad \text{for } t \geq 1,
\end{align*}
\]

where

- \((K_t(\cdot, \cdot))_{t \geq 1}\) are nonnegative and symmetric functions
- and \((\varepsilon_t)_{t \geq 1}\) are positive real parameters.

- Recursiveness is a major computational advantage.
Révész-type recursive estimate

- Compact form:

\[
\begin{align*}
  r_1(x) &= Y_1 \\
  r_{t+1}(x) &= r_t(x) - \varepsilon_{t+1} H(Z_{t+1}, r_t(x)) \quad \text{for } t \geq 1,
\end{align*}
\]
Révész-type recursive estimate

- Compact form:

\[
\begin{align*}
  r_1(x) &= Y_1 \\
  r_{t+1}(x) &= r_t(x) - \varepsilon_{t+1} H(Z_{t+1}, r_t(x)) \quad \text{for } t \geq 1,
\end{align*}
\]

where \( Z_{t+1} = (X_{t+1}, Y_{t+1}) \) and

\[
H(Z_{t+1}, r_t(x)) = r_t(x) K_{t+1}(x, X_{t+1}) - Y_{t+1} K_{t+1}(x, X_{t+1}).
\]
Révész-type recursive estimate

• Compact form:

\[
\begin{align*}
    r_1(x) &= Y_1 \\
    r_{t+1}(x) &= r_t(x) - \varepsilon_{t+1} H(Z_{t+1}, r_t(x)) \quad \text{for } t \geq 1,
\end{align*}
\]

where \( Z_{t+1} = (X_{t+1}, Y_{t+1}) \) and

\[
H(Z_{t+1}, r_t(x)) = r_t(x) K_{t+1}(x, X_{t+1}) - Y_{t+1} K_{t+1}(x, X_{t+1}).
\]

• Typically:

\[
K_t(x, z) = \frac{1}{h_t^d} K \left( \frac{x - z}{h_t} \right), \quad x, z \in \mathbb{R}^d.
\]
Distributed regression

• Computation/combining process:

\[
\begin{cases}
    r^i_1(x) &= Y^i_1 \\
    r^i_{t+1}(x) &= \sum_{j=1}^{M} a^i_{t} r^j(x, \tau^i_{t}) + s^i_t \quad \text{for } t \geq 1,
\end{cases}
\]
Distributed regression

- Computation/combining process:

\[
\begin{align*}
    r_1^i(x) &= Y_1^i \\
    r_{t+1}^i(x) &= \sum_{j=1}^{M} a_{t}^{ij} r_{t}^j(x, \tau_{t}^{ij}) + s_t^i \quad \text{for } t \geq 1,
\end{align*}
\]

where

\[\sum_{j=1}^{M} a_{t}^{ij} = 1\]
Distributed regression

- Computation/combining process:

\[
\begin{align*}
 r_1^i(x) &= Y_1^i \\
 r_{t+1}^i(x) &= \sum_{j=1}^{M} a_{t}^{ij} r^j(x, \tau_{t}^{ij}) + s_t^i \quad \text{for } t \geq 1,
\end{align*}
\]

where

\(\sum_{j=1}^{M} a_{t}^{ij} = 1\)

\(\text{The time instants } (\tau_{t}^{ij})_{t \geq 1} \text{ satisfy } 1 \leq \tau_{t}^{ij} \leq t\)
Distributed regression

- Computation/combining process:

\[
\begin{aligned}
    r_1^i(x) &= Y_1^i \\
    r_{t+1}^i(x) &= \sum_{j=1}^{M} a_{t}^{ij} r_j(x, \tau_{ij}^t) + s_t^i \quad \text{for } t \geq 1,
\end{aligned}
\]

where

\[\sum_{j=1}^{M} a_{t}^{ij} = 1\]

\[\text{The time instants } (\tau_{ij}^t)_{t \geq 1} \text{ satisfy } 1 \leq \tau_{ij}^t \leq t\]

\[\text{The term } s_t^i \text{ is a Révész-type computation step:}\]

\[
s_t^i = \begin{cases} 
    -\varepsilon_{t+1}^i H(Z_{t+1}^i, r_t^i(x)) & \text{if } t \in T^i \\
    0 & \text{otherwise.}
\end{cases}
\]
Main features

- Distributed $\rightarrow$ Large datasets processing.
Main features

- **Distributed** → Large datasets processing.

- **Online** → Time-varying data loads.
Main features

- **Distributed** → Large datasets processing.
- **Online** → Time-varying data loads.
- **Message passing + asynchronism** →
Main features

- **Distributed** → Large datasets processing.
- **Online** → Time-varying data loads.
- **Message passing + asynchronism** →
  - Major speed advantage over synchronous executions.
Main features

• Distributed → Large datasets processing.

• Online → Time-varying data loads.

• Message passing + asynchronism →
  ▶ Major speed advantage over synchronous executions.
  ▶ High degree of flexibility and tolerance to system failures.
Assumptions 1

**Convex combinations**

1. $\sum_{j=1}^{M} a_{ij}^{t} = 1$.

2. $a_{ii}^{t} \geq \alpha$.

3. $a_{ij}^{t} \in \{0\} \cup [\alpha, 1]$. 

Example: $a_{ij}^{t} = \frac{1}{\# N_i^{t}}$ if $j \in N_i^{t}$, otherwise 0, where $N_i^{t} = \{j \in \{1, \ldots, M\} : a_{ij}^{t} > 0\}$. 

Assumptions 1

Convex combinations

1. \( \sum_{j=1}^{M} a_{ij}^t = 1. \)

2. \( a_{ii}^t \geq \alpha. \)

3. \( a_{ij}^t \in \{0\} \cup [\alpha, 1]. \)

Example:

\[
\begin{align*}
a_{ij}^t &= \begin{cases} 
1/\#N_i^t & \text{if } j \in N_i^t \\
0 & \text{otherwise,}
\end{cases} \\
\text{where} \\
N_i^t &= \left\{ j \in \{1, \ldots, M\} : a_{ij}^t > 0 \right\}.
\end{align*}
\]
Assumptions 2

Bounded delays

1. One has $a_{t}^{ij} = 1_{[i\neq j]}$ for all $t \in T^i$.

2. If $a_{t}^{ij} = 0$, then $\tau_{t}^{jj} = t$.

3. One has $\tau_{t}^{ii} = t$.

4. There exists some constant $B_1 \geq 0$ such that

$$t - B_1 \leq \tau_{t}^{ij} \leq t.$$
Assumptions 3-4

• The network communication topology can be described in terms of a directed graph \((\mathcal{M}, E_t)\).
Assumptions 3-4

- The network communication topology can be described in terms of a directed graph \((\mathcal{M}, E_t)\).

- The edge \((j, i) \in E_t\) if and only if \(a_{ij}^t > 0\).
Assumptions 3-4

- The network communication topology can be described in terms of a directed graph $(\mathcal{M}, E_t)$.

- The edge $(j, i) \in E_t$ if and only if $a_{ij}^t > 0$.

Connectivity

The graph $(\mathcal{M}, \bigcup_{s \geq t} E_s)$ is strongly connected for all $t \geq 1$. 
**Assumptions 3-4**

- The network communication topology can be described in terms of a directed graph \((\mathcal{M}, E_t)\).

- The edge \((j, i) \in E_t\) if and only if \(a_{ij}^t > 0\).

<table>
<thead>
<tr>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>The graph ((\mathcal{M}, \bigcup_{s \geq t} E_s)) is strongly connected for all (t \geq 1).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bounded intercommunication intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is some constant (B_2 \geq 0) such that if ((i, j) \in E_t) infinitely often, then, for all (t \geq 1), ((i, j) \in E_t \cup E_{t+1} \cup \cdots \cup E_{t+B_2}).</td>
</tr>
</tbody>
</table>


Assumptions 5

Idle processors and learning rate

1. For all $t \geq 1$, one has $\sum_{j=1}^{M} 1_{[t \in T^j]} \geq 1$.

2. There exist two constants $C_1 > 0$ and $C_2 > 0$ such that

$$\frac{C_1}{t} \leq \varepsilon_i^t \leq \frac{C_2}{t}.$$
Main result

Theorem

- Assumptions 1-5 are satisfied.
Main result

**Theorem**

- Assumptions 1-5 are satisfied.
- One has
  \[ h_t^d K_t(x, z) \leq L \left( \frac{\|x - z\|}{h_t} \right), \]
  with \( h_t \to 0 \) and \( \sum_{t \geq 1} \frac{1}{t^2 h_t^{2\alpha}} < \infty. \)
Main result

**Theorem**

- **Assumptions 1-5** are satisfied.
- One has
  \[
  h_t^d K_t(x, z) \leq L \left( \frac{\|x - z\|}{h_t} \right),
  \]
  with \( h_t \to 0 \) and \( \sum_{t \geq 1} \frac{1}{t^2 h_t^{2d}} < \infty \).
- \( Y \) is bounded + technical assumptions on \( K_t \).
Main result

Theorem

- Assumptions 1-5 are satisfied.
- One has
  \[ h_t^d K_t(x, z) \leq L \left( \frac{\|x - z\|}{h_t} \right), \]
  with \( h_t \to 0 \) and \( \sum_{t \geq 1} \frac{1}{t^2 h_t^{2d}} < \infty. \)
- \( Y \) is bounded + technical assumptions on \( K_t \).

Then, for all \( i \in \{1, \ldots, M\} \),
Main result

Theorem

- Assumptions 1-5 are satisfied.

- One has

\[
h_t^d K_t(x, z) \leq L \left( \frac{\|x - z\|}{h_t} \right),
\]

with \( h_t \to 0 \) and \( \sum_{t \geq 1} \frac{1}{t^2 h_t^{2d}} < \infty \).

- \( Y \) is bounded + technical assumptions on \( K_t \).

Then, for all \( i \in \{1, \ldots, M\} \),

\[
\mathbb{E} \left[ \int_{\mathbb{R}^d} |r^i_t(x) - r(x)|^2 \mu(dx) \right] \to 0 \quad \text{as} \quad t \to \infty.
\]
Proof’s architecture

A general model:

\[ z_{t+1}^i = \sum_{j=1}^{M} a_{t}^{ij} z_{t}^j (\tau_{t}^{ij}) + s_{t}^i. \]
Proof’s architecture

1 A general model:

\[ z_{t+1}^i = \sum_{j=1}^{M} a_{t}^{ij} z_j^j(\tau_t^{ij}) + s_t^i. \]

2 Exploiting linearity:

\[ z_t^i = \sum_{j=1}^{M} \phi^{ij}(t, 0) z_1^j + \sum_{\tau=1}^{t-1} \sum_{j=1}^{M} \phi^{ij}(t, \tau) s_\tau^j. \]
Proof’s architecture

1. A general model:

\[ z_{t+1}^i = \sum_{j=1}^{M} a_{t}^{ij} z_j^j(\tau_t^j) + s_t^i. \]

2. Exploiting linearity:

\[ z_t^i = \sum_{j=1}^{M} \phi_t^{ij}(t, 0) z_1^j + \sum_{\tau=1}^{t-1} \sum_{j=1}^{M} \phi_t^{ij}(t, \tau) s_{\tau}^j. \]

3. Assume that the processors stop computing after time \( t_0 \). Then, as \( t \to \infty \),

\[ z_{t_0}^* = \sum_{j=1}^{M} \phi_0^{j} z_1^j + \sum_{\tau=1}^{t_0-1} \sum_{j=1}^{M} \phi_{\tau}^{j} s_{\tau}^j. \]
Regression case:

\[
s_t^i = \begin{cases} 
-\varepsilon_{t+1}^i H(Z_{t+1}, r_t^i(x)) & \text{if } t \in T^i \\
0 & \text{otherwise.}
\end{cases}
\]
1 Regression case:

\[ s_t^i = \begin{cases} 
-\varepsilon_{t+1} H (Z_{t+1}^i, r_t^i(x)) & \text{if } t \in T^i \\
0 & \text{otherwise}
\end{cases} \]

2 Agreement sequence:

\[
\begin{align*}
 r_1^*(x) &= \sum_{j=1}^{M} \phi_0^j Y_1^j \\
 r_{t+1}^*(x) &= r_t^*(x) - \sum_{j=1}^{M} 1_{[t \in T^j]} \phi_t^j \varepsilon_{t+1}^j H (Z_{t+1}^j, r_t^j(x)) & \text{for } t \geq 1.
\end{align*}
\]
1 Regression case:

\[
{s}_t^i = \begin{cases} 
-\varepsilon_{t+1}^i H \left( Z_{t+1}^i, r_t^i(x) \right) & \text{if } t \in T^i \\
0 & \text{otherwise}
\end{cases}
\]

2 Agreement sequence:

\[
\begin{align*}
    r^*_1(x) &= \sum_{j=1}^M \phi_0^j Y_1^j \\
    r^*_{t+1}(x) &= r_t^*(x) - \sum_{j=1}^M 1_{[t \in T^j]} \phi_t^j \varepsilon_t^j H \left( Z_{t+1}^j, r_t^j(x) \right) \quad \text{for } t \geq 1.
\end{align*}
\]

3 Idea:

\[
\begin{align*}
    r^*_1(x) &= \sum_{j=1}^M \phi_0^j Y_1^j \\
    r^*_{t+1}(x) &= r_t^*(x) - \sum_{j=1}^M 1_{[t \in T^j]} \phi_t^j \varepsilon_t^j H \left( Z_{t+1}^j, r_t^j(x) \right) + \Delta_{t+1}(x) \quad \text{for } t \geq 1.
\end{align*}
\]
Consistency of $r^*$:

$$\mathbb{E} \left[ \int_{\mathbb{R}^d} \left| r_t^*(x) - r(x) \right|^2 \mu(dx) \right] \to 0.$$
Consistency of $r^*$:

$$\mathbb{E} \left[ \int_{\mathbb{R}^d} |r_t^*(x) - r(x)|^2 \mu(dx) \right] \to 0.$$ 

Agreement:

$$\sup_{x \in \mathbb{R}^d} |r_t^i(x) - r_t^*(x)| \to 0.$$
Consistency of $r^*$:

$$\mathbb{E} \left[ \int_{\mathbb{R}^d} |r^*_t(x) - r(x)|^2 \mu(dx) \right] \to 0.$$ 

Agreement:

$$\sup_{x \in \mathbb{R}^d} |r^*_i(x) - r^*_t(x)| \to 0.$$ 

Conclusion.
Outline

1 Need for distributed computing

2 Distributed and asynchronous estimation

3 Implementation and numerical studies

4 Cellular tree classifiers
Implementation

- **Dolphin**: A software implemented in Go.
Implementation

- **Dolphin**: A software implemented in Go.

- **Open source**: [https://github.com/ryadzenine/dolphin](https://github.com/ryadzenine/dolphin).
Implementation

- **Dolphin**: A software implemented in Go.


- **Good scaling**: Managing the communication overhead.
Implementation

- **Dolphin**: A software implemented in Go.


- **Good scaling**: Managing the communication overhead.

- Careful when choosing the shape of the graph \((M, \cup_{s \geq t} E_s)\).
Implementation

- **Dolphin**: A software implemented in Go.


- **Good scaling**: Managing the communication overhead.

- Careful when choosing the shape of the graph \((\mathcal{M}, \bigcup_{s \geq t} E_s)\).

- **Asynchronism** forbids the use of any centralized mechanism.
Parameter setting

- Metronome: \( T^i = \{ k \in \mathbb{N}^* : k \equiv 0 \pmod{\tau} \}^c \).
### Parameter setting

- **Metronome:** 
  \[ T^i = \{ k \in \mathbb{N}^* : k \equiv 0 \pmod{\tau} \}\,^c. \]

- **Kernel:**
  \[ K_t(x, z) = \frac{1}{h_t^d} e^{-\|x-z\|^2/h_t^2}. \]

- **Smoothing:** 
  \[ t^{-\frac{d}{d+4}}. \]
Parameter setting

• **Metronome:** \( T^i = \{ k \in \mathbb{N}^* : k \equiv 0 \pmod{\tau} \}^c \).

• **Kernel:**
\[
K_t(x, z) = \frac{1}{h_t^d} e^{-\|x-z\|^2/h_t^2}.
\]

• **Smoothing:** \( t^{-\frac{d}{d+4}} \).

• **Calibration:** \( \varepsilon_t = 1/t \).
Models

Model 1: \( Y = X_1^2 + \exp(-X_2^2). \)

Model 2: \( Y = X_1 X_2 + X_3^2 - X_4 + \mathcal{N}(0, 0.05). \)

Model 3: \( Y = 1_{[X_1 > 0]} + 1_{[X_4 - X_2 > 1 + X_3]} + X_2^3 + \exp(-X_2^2) + \mathcal{N}(0, 0.05). \)

- **Designs:** Uniform over \((0, 1)^d\) and Gaussian with mean 0 and covariance matrix \( \Sigma_{ij} = 2^{-|i-j|} \).
Models

Model 1: \( Y = X_1^2 + \exp(-X_2^2) \).

Model 2: \( Y = X_1 X_2 + X_3^2 - X_4 + \mathcal{N}(0, 0.05) \).

Model 3: \( Y = 1_{[X_1 > 0]} + 1_{[X_4 - X_2 > 1 + X_3]} + X_2^3 + \exp(-X_2^2) + \mathcal{N}(0, 0.05) \).

- Designs: Uniform over \((0, 1)^d\) and Gaussian with mean 0 and covariance matrix \( \Sigma_{ij} = 2^{-|i-j|} \).

- Number of workers: 1 to 28.
Models

**Model 1:** \( Y = X_1^2 + \exp(-X_2^2). \)

**Model 2:** \( Y = X_1X_2 + X_3^2 - X_4 + \mathcal{N}(0, 0.05). \)

**Model 3:** \( Y = 1_{[X_1>0]} + 1_{[X_4-X_2>1+X_3]} + X_2^3 + \exp(-X_2^2) + \mathcal{N}(0, 0.05). \)

- **Designs:** Uniform over \((0, 1)^d\) and Gaussian with mean 0 and covariance matrix \( \Sigma_{ij} = 2^{-|i-j|}. \)

- **Number of workers:** 1 to 28.

- **Dataset:** \( n = 10^6 \), 20% for test.
Model 1, uniform design
Model 1, uniform design
Outline

1. Need for distributed computing
2. Distributed and asynchronous estimation
3. Implementation and numerical studies
4. Cellular tree classifiers
Greedy algorithms

- **Greedy algorithms** build solutions incrementally, usually with little effort.
Greedy algorithms

• **Greedy algorithms** build solutions incrementally, usually with little effort.

• Such procedures form a result **piece by piece**.
Greedy algorithms

- Greedy algorithms build solutions incrementally, usually with little effort.

- Such procedures form a result piece by piece.

- Greedy methods have an autonomy that makes them ideally suited for distributed or parallel computation.
Classification
Classification
Classification
Mathematical setting

- A generic pair $(X, Y) \in \mathbb{R}^d \times \{0, 1\}$. 

- Goal: Design a classifier $g : \mathbb{R}^d \to \{0, 1\}$.

- The probability of error is $L(g) = P\{g(X) \neq Y\}$.

- The Bayes classifier $g^\star(x) = \begin{cases} 
1 & \text{if } P\{Y = 1 | X = x\} > 1/2 \\
0 & \text{otherwise}
\end{cases}$ has the smallest probability of error, that is $L^\star = L(g^\star) = \inf_{g : \mathbb{R}^d \to \{0, 1\}} P\{g(X) \neq Y\}$. 
Mathematical setting

- A generic pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\).

- Goal: Design a classifier \(g : \mathbb{R}^d \to \{0, 1\}\).
Mathematical setting

- A generic pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\).

- **Goal:** Design a classifier \(g : \mathbb{R}^d \to \{0, 1\}\).

- The probability of error is \(L(g) = \mathbb{P}\{g(X) \neq Y\}\).
Mathematical setting

• A generic pair \((X, Y) \in \mathbb{R}^d \times \{0, 1\}\).

• Goal: Design a classifier \(g : \mathbb{R}^d \rightarrow \{0, 1\}\).

• The probability of error is \(L(g) = \mathbb{P}\{g(X) \neq Y\}\).

• The Bayes classifier

\[
g^*(x) = \begin{cases} 
1 & \text{if } \mathbb{P}\{Y = 1|X = x\} > 1/2 \\
0 & \text{otherwise}
\end{cases}
\]

has the smallest probability of error, that is

\[
L^* = L(g^*) = \inf_{g: \mathbb{R}^d \rightarrow \{0, 1\}} \mathbb{P}\{g(X) \neq Y\}.
\]
Basics of classification

• The data: $D_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, i.i.d. copies of $(X, Y)$. 
Basics of classification

- The data: $D_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, i.i.d. copies of $(X, Y)$.

- A classifier $g_n(x)$ is a function of $x$ and $D_n$. 
Basics of classification

- The data: \( \mathcal{D}_n = (X_1, Y_1), \ldots, (X_n, Y_n) \), i.i.d. copies of \((X, Y)\).

- A classifier \( g_n(x) \) is a function of \( x \) and \( \mathcal{D}_n \).

- The probability of error is

\[
L(g_n) = \mathbb{P}\{g_n(X) \neq Y | \mathcal{D}_n\}.
\]
Basics of classification

- The data: $D_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, i.i.d. copies of $(X, Y)$.

- A classifier $g_n(x)$ is a function of $x$ and $D_n$.

- The probability of error is

$$L(g_n) = \mathbb{P}\{g_n(X) \neq Y | D_n\}.$$

- It is consistent if

$$\mathbb{E}L(g_n) \rightarrow L^* \quad \text{as} \quad n \rightarrow \infty.$$
Basics of classification

- **The data**: $\mathcal{D}_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, i.i.d. copies of $(X, Y)$.

- A **classifier** $g_n(x)$ is a function of $x$ and $\mathcal{D}_n$.

- The **probability of error** is
  
  $$L(g_n) = \mathbb{P}\{g_n(X) \neq Y | \mathcal{D}_n\}.$$

- It is **consistent** if
  
  $$\mathbb{E}L(g_n) \to L^* \text{ as } n \to \infty.$$

- It is **universally consistent** if it is consistent for all possible distributions of $(X, Y)$.
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Trees
Tree classifiers

- Many popular classifiers are universally consistent.
Tree classifiers

- Many popular classifiers are universally consistent.

- These include various brands of histogram rules, $k$-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.
Tree classifiers

• Many popular classifiers are universally consistent.

• These include various brands of histogram rules, $k$-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.

• Tree methods *loom large* for several reasons:
Tree classifiers

- Many popular classifiers are universally consistent.

- These include various brands of histogram rules, \( k \)-nearest neighbor rules, kernel rules, neural networks, and tree classifiers.

- Tree methods *loom large* for several reasons:
  - All procedures that *partition space* can be viewed as special cases of partitions generated by trees.
  - Tree classifiers are *conceptually simple*, and explain the data very well.
Trees

- The tree structure is usually data dependent.
Trees

- The tree structure is usually data dependent.

- There are virtually infinitely many possible strategies to build classification trees.
Trees

• The tree structure is usually data dependent.

• There are virtually infinitely many possible strategies to build classification trees.

• All tree species end up with two fundamental questions:

① Should the node be split?
② In the affirmative, what are its children?
The cellular spirit

• Cellular trees proceed from a different philosophy.
The cellular spirit

• Cellular trees proceed from a different philosophy.

• A cellular tree should be able to answer questions ₁ and ₂ using local information only.
Cellular recursive procedure

▷ If $\theta(D_A) = 0$, the cell is final.
Cellular recursive procedure

- If $\theta(\mathcal{D}_A) = 0$, the cell is final.

- Otherwise, $\mathbb{R}^d$ is split into

$$A = \{x : f(x, \sigma(\mathcal{D}_A)) \geq 0\} \quad \text{and} \quad B = \{x : f(x, \sigma(\mathcal{D}_A)) < 0\}.$$
Cellular recursive procedure

- If $\theta(\mathcal{D}_A) = 0$, the cell is final.

- Otherwise, $\mathbb{R}^d$ is split into

  $A = \{x : f(x, \sigma(\mathcal{D}_A)) \geq 0\}$ and $B = \{x : f(x, \sigma(\mathcal{D}_A)) < 0\}$.

- The data are partitioned into two groups.
Cellular recursive procedure

- If \( \theta(D_A) = 0 \), the cell is final.
- Otherwise, \( \mathbb{R}^d \) is split into
  \[ A = \{ x : f(x, \sigma(D_A)) \geq 0 \} \text{ and } B = \{ x : f(x, \sigma(D_A)) < 0 \} . \]
- The data are partitioned into two groups.
- The groups are sent to child cells, and the process is repeated.
The $k$-median tree

- When $d = 1$, split by finding the median element among the $X_i$'s.
The $k$-median tree

- When $d = 1$, split by finding the median element among the $X_i$'s.
- Keep doing this for $k$ rounds.
The $k$-median tree

- When $d = 1$, split by finding the median element among the $X_i$’s.
- Keep doing this for $k$ rounds.
- In $d$ dimensions, rotate through the coordinates.
The $k$-median tree

- When $d = 1$, split by finding the median element among the $X_i$’s.

- Keep doing this for $k$ rounds.

- In $d$ dimensions, rotate through the coordinates.

This rule is consistent, provided $k \to \infty$ and $k2^k/n \to 0$. 
The $k$-median tree

- When $d = 1$, split by finding the median element among the $X_i$'s.

- Keep doing this for $k$ rounds.

- In $d$ dimensions, rotate through the coordinates.

This rule is consistent, provided $k \to \infty$ and $k2^k/n \to 0$.

This is not cellular.
A randomized solution

- Consider a nonincreasing function $\varphi : \mathbb{N} \rightarrow (0, 1]$. \\

Theorem
Let $\beta$ be a real number in $(0, 1)$. Define $\varphi(n) = \begin{cases} 1 & \text{if } n < 3 \\ \frac{1}{\log \beta} n & \text{if } n \geq 3. \end{cases}$ Then $E[L(g_n)] \rightarrow L^\star$ as $n \rightarrow \infty$. 
A randomized solution

• Consider a nonincreasing function $\varphi : \mathbb{N} \rightarrow (0, 1]$.

• Then, if $U$ is the uniform $[0, 1]$ random variable associated with node $A$,

$$\theta = 1_{[U > \varphi(N(A))]}.$$
A randomized solution

- Consider a nonincreasing function $\varphi : \mathbb{N} \to (0, 1]$.

- Then, if $U$ is the uniform $[0, 1]$ random variable associated with node $A$,

$$\theta = 1[U > \varphi(N(A))] .$$

Theorem

Let $\beta$ be a real number in $(0, 1)$. Define

$$\varphi(n) = \begin{cases} 
  1 & \text{if } n < 3 \\
  1/\log^\beta n & \text{if } n \geq 3.
\end{cases}$$

Then

$$\mathbb{E}L(g_n) \to L^* \text{ as } n \to \infty.$$
A non-randomized solution

- At the root, we find the median in direction 1.
A non-randomized solution

- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.
A non-randomized solution

- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.
- Then on each of the four subsets, we find the median in direction 3, and so forth.
A non-randomized solution

- At the root, we find the median in direction 1.
- Then on each of the two subsets, we find the median in direction 2.
- Then on each of the four subsets, we find the median in direction 3, and so forth.
- Repeating this for $k$ levels of nodes leads to $2^{dk}$ leaf regions.
The stopping rule $\theta$

- The quality of the classifier at node $A$ is assessed by

$$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, Y_i=1], \sum_{i=1}^{n} 1[x_i \in A, Y_i=0] \right).$$
The stopping rule \( \theta \)

- The quality of the classifier at node \( A \) is assessed by

\[
\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, Y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, Y_i = 0] \right).
\]

- Define the nonnegative integer \( k^+ \) by

\[
k^+ = \lceil \alpha \log_2(N(A) + 1) \rceil.
\]
The stopping rule $\theta$

- The quality of the classifier at node $A$ is assessed by

$$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, Y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, Y_i = 0] \right).$$

- Define the nonnegative integer $k^+$ by

$$k^+ = \lfloor \alpha \log_2(N(A) + 1) \rfloor.$$

- Set

$$\hat{L}_n(A, k^+) = \sum_{A_j \in \mathcal{P}_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.$$
The stopping rule $\theta$

- The quality of the classifier at node $A$ is assessed by
  $$\hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, y_i = 0] \right).$$

- Define the nonnegative integer $k^+$ by
  $$k^+ = \lfloor \alpha \log_2 (N(A) + 1) \rfloor.$$

- Set
  $$\hat{L}_n(A, k^+) = \sum_{A_j \in P_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.$$

- Both $\hat{L}_n(A)$ and $\hat{L}_n(A, k^+)$ may be evaluated on the basis of the data points falling in $A$ only.
The stopping rule $\theta$

- The **quality** of the classifier at node $A$ is assessed by
  \[
  \hat{L}_n(A) = \frac{1}{N(A)} \min \left( \sum_{i=1}^{n} 1[x_i \in A, Y_i = 1], \sum_{i=1}^{n} 1[x_i \in A, Y_i = 0] \right).
  \]

- Define the **nonnegative integer** $k^+$ by
  \[
  k^+ = \lfloor \alpha \log_2(N(A) + 1) \rfloor.
  \]

- Set
  \[
  \hat{L}_n(A, k^+) = \sum_{A_j \in P_{k^+}(A)} \hat{L}_n(A_j) \frac{N(A_j)}{N(A)}.
  \]

- Both $\hat{L}_n(A)$ and $\hat{L}_n(A, k^+)$ may be evaluated on the basis of the data points falling in $A$ only.

This is **cellular**.
Result

Put $\theta = 0$ if

$$\left| \hat{L}_n(A) - \hat{L}_n(A, k^+) \right| \leq \left( \frac{1}{N(A) + 1} \right)^\beta.$$
Result

Put \( \theta = 0 \) if

\[
\left| \hat{L}_n(A) - \hat{L}_n(A, k^+) \right| \leq \left( \frac{1}{N(A) + 1} \right)^\beta.
\]

---

Theorem

Take \( 1 - d\alpha - 2\beta > 0 \). Then

\[
\mathbb{E} L(g_n) \to L^* \quad \text{as} \quad n \to \infty.
\]
Result

\[ \text{Put } \theta = 0 \text{ if } \left| \hat{L}_n(A) - \hat{L}_n(A, k^+) \right| \leq \left( \frac{1}{N(A) + 1} \right)^\beta. \]

Theorem

Take \( 1 - d\alpha - 2\beta > 0 \). Then

\[ \mathbb{E}L(g_n) \to L^* \text{ as } n \to \infty. \]
MapReduce

- A general distributed programming model due to Google (2004).
MapReduce

• A general distributed programming model due to Google (2004).

• A MapReduce program is composed of:
MapReduce

- A general distributed programming model due to Google (2004).

- A MapReduce program is composed of:
  - A Map procedure that performs filtering and sorting
MapReduce

- A general distributed programming model due to Google (2004).

- A MapReduce program is composed of:
  - A Map procedure that performs filtering and sorting
  - A Reduce procedure that performs a summary operation.
MapReduce

- A general **distributed programming model** due to Google (2004).

- A MapReduce program is composed of:
  - A **Map procedure** that performs filtering and sorting
  - A **Reduce procedure** that performs a summary operation.

- The MapReduce system **orchestrates the processing** by marshaling the distributed servers and running the various tasks in parallel.
Hadoop

- **Hadoop** is an open-source software framework designed to abstract away much of the complexity of MapReduce.
Hadoop

• **Hadoop** is an open-source software framework designed to abstract away much of the complexity of MapReduce.

• Hadoop = HDFS + MapReduce.
Hadoop

- Hadoop is an open-source software framework designed to abstract away much of the complexity of MapReduce.

- Hadoop = HDFS + MapReduce.