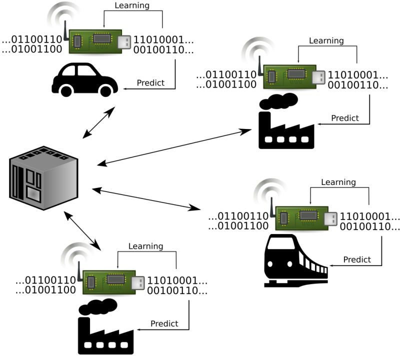
Shrub Ensembles for Online Classification

Sebastian Buschjäger, Sibylle Hess and Katharina Morik

Thirty-Sixth AAAI Conference on Artificial Intelligence 2022

Artificial Intelligence Group@TU Dortmund University - Data Mining Group@ TU Eindhoven TU/e - Collaborative Research Center 876

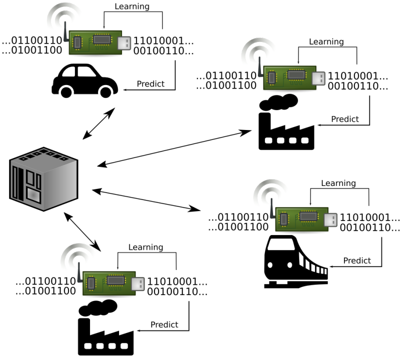
Online Learning on the Edge



In 2018 Roughly 22 Billion IoT Devices world wide

By 2025 $\approx$ 38 Billion IoT Devices world wide[1]

Online Learning on the Edge



In 2018 Roughly 22 Billion IoT Devices world wide
By 2025 ≈ 38 Billion IoT Devices world wide[1]

Clear benefits

+ Privacy and independence of data analysis
+ Reduced communication infrastructure costs
+ Faster response times

[1] https://findstack.com/internet-of-things-statistics/

Computational efficiency

The algorithm must process examples at least as fast as new examples arrive.

Computational efficiency

The algorithm must process examples at least as fast as new examples arrive.

Memory efficiency

The algorithm has only a limited budget of memory and fails if more memory is required.

Computational efficiency

The algorithm must process examples at least as fast as new examples arrive.

Memory efficiency

The algorithm has only a limited budget of memory and fails if more memory is required.

Evolving data streams

The algorithm must adapt to changes in the new data distribution and preserve its performance.

Online Decision Tree Ensemble Learning

Online Decision Tree Ensemble Learning

Gradient-based Learners

+ constant memory consumption
+ joint optimization of all trees
- backpropagation through entire tree

Online Decision Tree Ensemble Learning

Gradient-based Learners

+ constant memory consumption
+ joint optimization of all trees
- backpropagation through entire tree

Incremental Learners

+ fast
+ proven in practice
- nodes are not removed

Online Decision Tree Ensemble Learning

Gradient-based Learners

+ constant memory consumption
+ joint optimization of all trees
- backpropagation through entire tree

Incremental Learners

+ fast
+ proven in practice
- nodes are not removed

Can we design simple <u>and</u> small online ensembles?

$$\ldots, (x_0, y_0), \boxed{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)}, (x_6, ?), (x_7, ?), (x_8, ?) \ldots$$

$$\longrightarrow \text{train} \longrightarrow \text{h(x)} \longrightarrow$$

+ We can use a batch algorithm to train $h(x)$
+ We quickly adapt to concept drift
- No long-term learning possible

Revisiting Sliding Windows

$$\ldots, (x_1, y_1), \boxed{(x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6)}, (x_7, ?), (x_8, ?), (x_9, ?) \ldots$$

$$\rightarrow \text{train} \longrightarrow \text{h(x)}$$

+ We can use a batch algorithm to train $h(x)$
+ We quickly adapt to concept drift
- No long-term learning possible

Revisiting Sliding Windows

$$\ldots, (x_2, y_2), \boxed{(x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6), (x_7, y_7),} (x_8, ?), (x_9, ?), (x_{10}, ?) \ldots$$

$\rightarrow$ train $\longrightarrow$ h(x)

+ We can use a batch algorithm to train $h(x)$
+ We quickly adapt to concept drift
- No long-term learning possible

Long-Term Learning With Sliding Windows

$$\ldots, \boxed{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)}, (x_6, ?) \ldots, \boxed{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)}, (x_6, ?) \ldots,$$

$$\longrightarrow \text{train} \longrightarrow h_1(x) \qquad \longrightarrow \text{train} \longrightarrow h_2(x)$$

Often Local Patterns repeat over long running processes

Hence The two classifiers h_1 and h_2 are redundant. We should use h_1

Long-Term Learning With Sliding Windows

$$\ldots, \boxed{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)}, (x_6, ?) \ldots, \boxed{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)}, (x_6, ?) \ldots,$$

$$\text{train} \longrightarrow h_1(x) \qquad \text{train} \longrightarrow h_2(x)$$

Often Local Patterns repeat over long running processes

Hence The two classifiers h_1 and h_2 are redundant. We should use h_1

Idea Learn an ensemble of local patterns from sliding windows

$$f(x) = \sum_{i=1}^{M} w_i h_i(x)$$

Setting and Notation

Formally Let $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ be the set of trees learned on local patterns:

$$\arg\min_{w \in \mathbb{R}^K} \sum_{t=1}^{T} \ell\left(f_{S[0:t-1]}(x_t), y_t\right) \text{ s.t. } \|w\|_0 \leq M, w_i \geq 0, \sum_{i=1}^{K} w_i = 1$$

with

- $f_{S[0:t-1]} \colon \mathbb{R}^d \to \mathbb{R}^C$ is the model at time t
- $\ell \colon \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ is the loss

Setting and Notation

Formally Let $\mathcal{H} = \{h_1, h_2, \ldots, h_K\}$ be the set of trees learned on local patterns:

$$\arg\min_{w \in \mathbb{R}^K} \sum_{t=1}^{T} \ell\left(f_{S[0:t-1]}(x_t), y_t\right) \text{ s.t. } \|w\|_0 \le M, w_i \ge 0, \sum_{i=1}^{K} w_i = 1$$

with

- $f_{S[0:t-1]} \colon \mathbb{R}^d \to \mathbb{R}^C$ is the model at time t
- $\ell \colon \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ is the loss
- $M \ge 1$ is the maximum number of ensemble members

Setting and Notation

Formally Let $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ be the set of trees learned on local patterns:

$$\arg\min_{w \in \mathbb{R}^K} \sum_{t=1}^{T} \ell\left(f_{S[0:t-1]}(x_t), y_t\right) \text{ s.t. } \|w\|_0 \leq M, w_i \geq 0, \sum_{i=1}^{K} w_i = 1$$

with

- $f_{S[0:t-1]} \colon \mathbb{R}^d \to \mathbb{R}^C$ is the model at time t
- $\ell \colon \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ is the loss
- $M \geq 1$ is the maximum number of ensemble members
- $\|w\|_0 = \sum_{i=1}^{K} \mathbb{1}\{w_i \neq 0\}$ is the 0-norm

Setting and Notation

Formally Let $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ be the set of trees learned on local patterns:

$$\arg\min_{w \in \mathbb{R}^K} \sum_{t=1}^{T} \ell\left(f_{S[0:t-1]}(x_t), y_t\right) \text{ s.t. } \|w\|_0 \leq M, w_i \geq 0, \sum_{i=1}^{K} w_i = 1$$

with

- $f_{S[0:t-1]} \colon \mathbb{R}^d \to \mathbb{R}^C$ is the model at time t
- $\ell \colon \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ is the loss
- $M \geq 1$ is the maximum number of ensemble members
- $\|w\|_0 = \sum_{i=1}^{K} \mathbb{1}\{w_i \neq 0\}$ is the 0-norm

Setting and Notation

Formally Let $\mathcal{H} = \{h_1, h_2, \dots, h_K\}$ be the set of trees learned on local patterns:

$$\arg\min_{w \in \mathbb{R}^K} \sum_{t=1}^{T} \ell\left(f_{S[0:t-1]}(x_t), y_t\right) \text{ s.t. } \|w\|_0 \le M, w_i \ge 0, \sum_{i=1}^{K} w_i = 1$$

with

- $f_{S[0:t-1]}\colon \mathbb{R}^d \to \mathbb{R}^C$ is the model at time t
- $\ell\colon \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}_+$ is the loss
- $M \ge 1$ is the maximum number of ensemble members
- $\|w\|_0 = \sum_{i=1}^{K} \mathbb{1}\{w_i \neq 0\}$ is the 0-norm

For this talk Focus on the MSE loss

$$\ell(f_{S[0:t-1]}(x_t), y_t) = \frac{1}{C}\left\|f_{S[0:t-1]}(x_t) - y_t\right\|^2$$

Proximal Gradient Descent

$$w \leftarrow \mathcal{P}\left(w - \alpha \nabla_w L_{\mathcal{B}}(w)\right),$$

where

- $\mathcal{B}$ is the current window with $|\mathcal{B}| = B$ examples

Proximal Gradient Descent

$$w \leftarrow \mathcal{P}\left(w - \alpha \nabla_w L_{\mathcal{B}}(w)\right),$$

where

- $\mathcal{B}$ is the current window with $|\mathcal{B}| = B$ examples
- $\nabla_w L_{\mathcal{B}}(w)$ is the gradient on the current window

Proximal Gradient Descent

$$w \leftarrow \mathcal{P}\left(w - \alpha \nabla_w L_{\mathcal{B}}(w)\right),$$

where

- $\mathcal{B}$ is the current window with $|\mathcal{B}| = B$ examples
- $\nabla_w L_{\mathcal{B}}(w)$ is the gradient on the current window
- $\alpha \in \mathbb{R}_+$ is the step-size

Proximal Gradient Descent

$$w \leftarrow \mathcal{P}\left(w - \alpha \nabla_w L_{\mathcal{B}}(w)\right),$$

where

- $\mathcal{B}$ is the current window with $|\mathcal{B}| = B$ examples
- $\nabla_w L_{\mathcal{B}}(w)$ is the gradient on the current window
- $\alpha \in \mathbb{R}_+$ is the step-size
- $\mathcal{P}(w)$ is the prox-operator for the feasible set $\Delta = \left\{ w \in \mathbb{R}_+^K \middle| \sum_{i=1}^K w_i = 1, \|w\|_0 = M \right\}$

Proximal Gradient Descent

$$w \leftarrow \mathcal{P}\left(w - \alpha \nabla_w L_{\mathcal{B}}(w)\right),$$

where

- $\mathcal{B}$ is the current window with $|\mathcal{B}| = B$ examples
- $\nabla_w L_{\mathcal{B}}(w)$ is the gradient on the current window
- $\alpha \in \mathbb{R}_+$ is the step-size
- $\mathcal{P}(w)$ is the prox-operator for the feasible set $\Delta = \left\{ w \in \mathbb{R}_+^K \middle| \sum_{i=1}^K w_i = 1, \|w\|_0 = M \right\}$

Our paper / Kyrillidis et al. 2013 Details for prox-operator

Putting it all together: Shrub Ensembles (SE)

Idea If a tree is helpful to the ensemble, then it should have a large weight

Algorithm 1: Shrub Ensembles.

$w \leftarrow (0); \mathcal{B} \leftarrow [\,]; \mathcal{H} \leftarrow [\,]$ ▷ Init.
for next item (x, y) **do**
    if $|\mathcal{B}| = B$ **then** ▷ Update batch
        $\mathcal{B}$.`pop_first()`
    $\mathcal{B}$.`append`$((x, y))$
    $h_{new} \leftarrow \texttt{train}(\mathcal{B})$ ▷ Add new classifier
    $\mathcal{H}$.`append`(h_{new})
    $w \leftarrow (w_1, \ldots, w_M, 0)$ ▷ Initialize weight
    $w \leftarrow w - \alpha \nabla_w L_{\mathcal{B}}(w)$ ▷ Gradient step
    $w, \mathcal{H} \leftarrow \texttt{sorted}(w, \mathcal{H})$ ▷ Sort decreasing order
    $w \leftarrow \mathcal{P}(w)$ ▷ Project on feasible set
    $w, \mathcal{H} \leftarrow \texttt{prune}(w, \mathcal{H})$ ▷ Remove zero weights

Theoretical Insights

Runtime $\mathcal{O}\left(dB^2 \log B + \log M\right)$ per example with d features

Theoretical Insights

Runtime $\mathcal{O}\left(\boxed{dB^2 \log B} + \log M\right)$ per example with d features

CART

Runtime $\mathcal{O}\left(\underbrace{\boxed{dB^2 \log B}}_{\text{CART}} + \underbrace{\boxed{\log M}}_{\text{prox operator}}\right)$ per example with d features

Theoretical Insights

Runtime $\mathcal{O}\left(\underbrace{\boxed{dB^2 \log B}}_{\text{CART}} + \underbrace{\boxed{\log M}}_{\text{prox operator}}\right)$ per example with d features

Memory $\mathcal{O}(dB + 2 \cdot B \cdot (M + 1))$ per example with d features

Theoretical Insights

Runtime $\mathcal{O}\left(\boxed{dB^2 \log B} + \boxed{\log M}\right)$ per example with d features

CART

prox operator

Memory $\mathcal{O}(\boxed{dB} + 2 \cdot B \cdot (M+1))$ per example with d features

window

Theoretical Insights

Runtime $\mathcal{O}\left(\underbrace{\boxed{dB^2 \log B}}_{\text{CART}} + \underbrace{\boxed{\log M}}_{\text{prox operator}}\right)$ per example with d features

Memory $\mathcal{O}(\underbrace{\boxed{dB}}_{\text{window}} + \underbrace{\boxed{2 \cdot B \cdot (M+1)}}_{\text{trees} \rightarrow \text{shrubs}})$ per example with d features

Theoretical Insights

Runtime $\mathcal{O}\left(\underbrace{\boxed{dB^2 \log B}}_{\text{CART}} + \underbrace{\boxed{\log M}}_{\text{prox operator}}\right)$ per example with d features

Memory $\mathcal{O}(\underbrace{\boxed{dB}}_{\text{window}} + \underbrace{\boxed{2 \cdot B \cdot (M+1)}}_{\text{trees} \rightarrow \text{shrubs}})$ per example with d features

Behaviour Let $m \leq M$ be the number of models in the ensemble and let $\forall j = 1, \ldots, m \colon h_j(x_B) \neq y_B$. If SE trains fully-grown trees such that $\forall i = 1, \ldots, B \colon h(x_i) = y_i$ it holds for $\alpha > \frac{BC}{4m}$ that:

- (1) If $m < M$, then h is added to the ensemble
- (2) If $m = M$, then h replaces the tree with the smallest weight from the ensemble

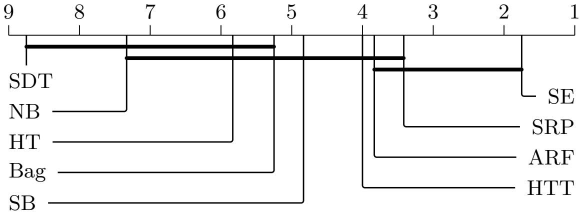
Experimental Insights: Quantitative Analysis

Goal Compare Accuracy-Memory Trade-off of different configurations

1) Plot Pareto Front of best performing configurations against accuracy
2) Compute Area Under the Pareto Front to measure accuracy-memory trade-off
3) Rank each method according to its trade-off and plot a CD-diagram

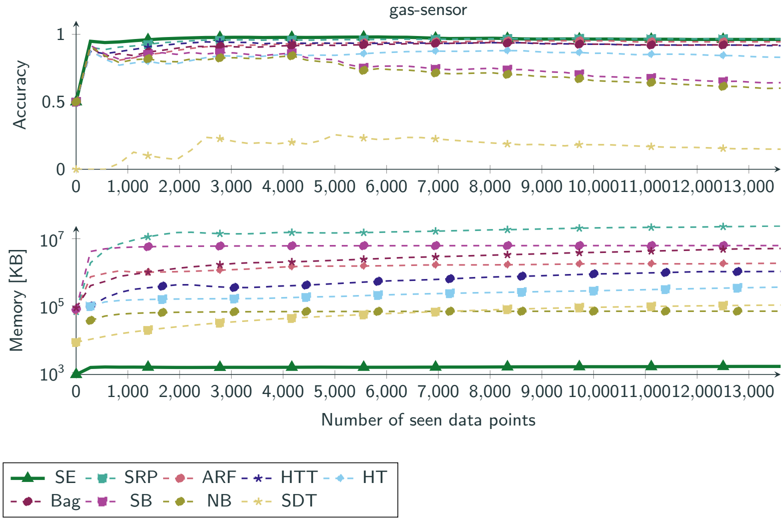
Goal Compare Accuracy-Memory Trade-off of different configurations

1) Plot Pareto Front of best performing configurations against accuracy
2) Compute Area Under the Pareto Front to measure accuracy-memory trade-off
3) Rank each method according to its trade-off and plot a CD-diagram

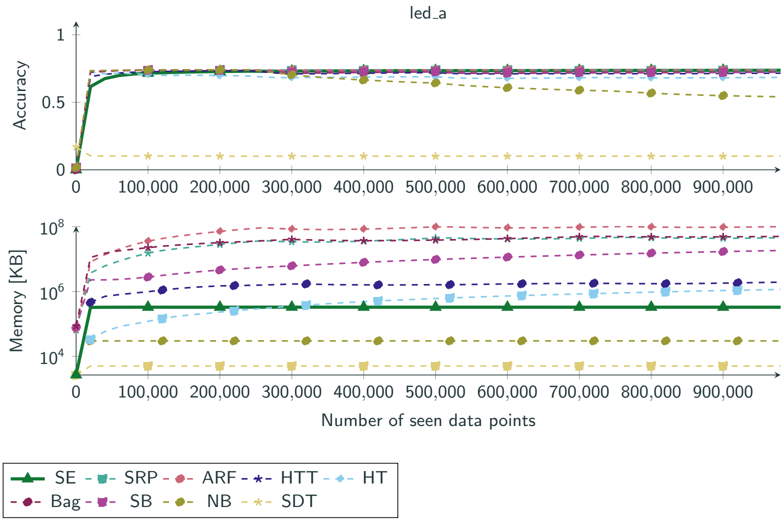


- SE: Shrub Ensembles
- SRP: Streaming Random Patches
- ARP: Adaptive Random Forest
- HTT: HoeffdingAnyTree
- SB: (Online) Smooth Boost
- Bag: (Online) Bagging
- HT: HoeffdingTrees
- NB: (Online) NaiveBayes
- SDT: Soft Decision Trees

Experimental Insights: Qualitative Analysis (1)



Experimental Insights: Qualitative Analysis (2)



Edge learning becomes more important every year

- Computational efficiency
- Memory efficiency
- Evolving data streams

Edge learning becomes more important every year

- Computational efficiency
- Memory efficiency
- Evolving data streams

Current approaches Unbounded memory or costly gradients

Edge learning becomes more important every year

- Computational efficiency
- Memory efficiency
- Evolving data streams

Current approaches Unbounded memory or costly gradients

Shrub Ensembles Bounded memory and simple gradients

- Train small trees ($\rightarrow$ Shrubs) on sliding window
- Maintain tree weights via proximal gradient descent
- Prune unimportant trees

Results Better accuracy-memory trade-off than existing methods!