Joint Leaf-Refinement and Ensemble Pruning Through L1 Regularization

Sebastian Buschjäger and Katharina Morik

ECML-PKDD 2023 – November 9th

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Idea Report carbon footprint as a (weak) proxy for general resource consumption

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Idea Report carbon footprint as a (weak) proxy for general resource consumption

Apple's Product Environmental Report[https://www.apple.com/environment/]

(excluding end-of-life processing here)

IPhone-14	1 Year [kg]	3 Years [kg]	10 Years [kg]
Production	48.19	48.19	48.19
Transport	1.22	1.22	1.22
Useage	3.66	10.98	36.6

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Idea Report carbon footprint as a (weak) proxy for general resource consumption

Apple's Product Environmental Report[https://www.apple.com/environment/]

(excluding end-of-life processing here)

IPhone-14	1 Year [%]	3 Years [%]	10 Years [%]
Production	90.8	79.0	56.0
Transport	2.3	1.9	1.4
Useage	6.9	18.0	42.5

(Percentages may not total 100 due to rounding.)

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Idea Report carbon footprint as a (weak) proxy for general resource consumption

Apple's Product Environmental Report[https://www.apple.com/environment/]

(excluding end-of-life processing here)

IPhone-14	1 Year [%]	3 Years [%]	10 Years [%]
Production	90.8	79.0	56.0
Transport	2.3	1.9	1.4
Useage	6.9	18.0	42.5

(Percentages may not total 100 due to rounding.)

Clear We must use an IPhone-14 for around ten years to break even with production costs!

But Average life-cycle for an IPhone-14 are 3 to 4 years

Resource consumption of computing hardware

Question How many resources are required by new computing hardware in general?

Idea Report carbon footprint as a (weak) proxy for general resource consumption

Apple's Product Environmental Report[`https://www.apple.com/environment/`]

(excluding end-of-life processing here)

IPhone-14	1 Year [%]	3 Years [%]	10 Years [%]
Production	90.8	79.0	56.0
Transport	2.3	1.9	1.4
Useage	6.9	18.0	42.5

(Percentages may not total 100 due to rounding.)

Clear We must use an IPhone-14 for around ten years to break even with production costs!

But Average life-cycle for an IPhone-14 are 3 to 4 years

Thus We have to run new algorithms on older ($\approx$ smaller) hardware!

A Closer Look at Older / Smaller Hardware

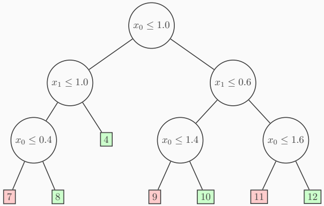
MCU	CPU	Flash	(S)RAM	Power
Arduino Uno (ATMega128P)	16MHz	32KB	2KB	12mA
Arduino Mega (ATMega2560)	16MHz	256KB	8KB	6mA
STM32L0 (Cortex-M0)	32MHz	192KB	20KB	7mA
Arduino MKR1000 (Cortex-M0)	48MHz	256KB	32KB	4mA
STM32F2 (Cortex-M3)	120MHz	1MB	128KB	21mA
STM32F4 (Cortex-M4)	180MHz	2MB	384KB	50mA
RPi A+	700MHz	SD Card	256MB	80mA
RPi Zero	1GHz	SD Card	512MB	80mA
RPi 3B	4@1.2GHz	SD Card	1GB	260mA
Apple A7 (IPhone 5)	2@1.4 Ghz	16-64 GB	1GB	320-485 mA

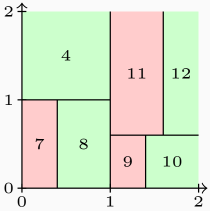
Design ML algorithms for older hardware

($\rightarrow$ fewer computations, less memory)

Recap Additive Tree Ensembles

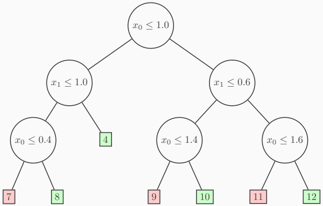
Axis-aligned Decision Trees Split data into groups of increasing label purity

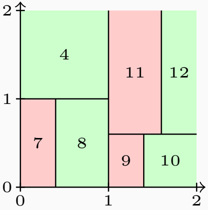




Recap Additive Tree Ensembles

Axis-aligned Decision Trees Split data into groups of increasing label purity

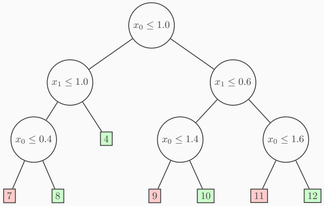


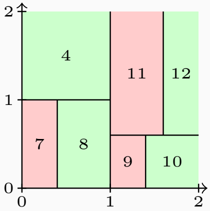


$$h(x) = \sum_{i=1}^{L} y_i \pi_i(x), \;\; \pi_i(x) = 1 \text{ if x in leaf i else } 0$$

Recap Additive Tree Ensembles

Axis-aligned Decision Trees Split data into groups of increasing label purity





$$h(x) = \sum_{i=1}^{L} y_i \pi_i(x), \ \pi_i(x) = 1 \text{ if x in leaf i else } 0$$

Random Forest Train multiple DTs on bootstrap samples and average predictions

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

Training Additive Ensembles *for* Small Devices

Cool RFs require minimal computations

And DTs are simple! RFs is a set of DTs. Hence, aren't Random Forests already small enough?!

Training Additive Ensembles *for* Small Devices

Cool RFs require minimal computations
And DTs are simple! RFs is a set of DTs. Hence, aren't Random Forests already small enough?!

Unfortunately RFs can easily grow in size, even for smaller datasets.

	adult	avila	bank	eeg	elec	mnist
accuracy [%]	86.78	98.58	90.39	93.42	88.98	96.53
model size [MB]	24.99	32.85	24.99	14.95	24.99	56.99

Training Additive Ensembles *for* Small Devices

Cool RFs require minimal computations
And DTs are simple! RFs is a set of DTs. Hence, aren't Random Forests already small enough?!

Unfortunately RFs can easily grow in size, even for smaller datasets.

	adult	avila	bank	eeg	elec	mnist
accuracy [%]	86.78	98.58	90.39	93.42	88.98	96.53
model size [MB]	24.99	32.85	24.99	14.95	24.99	56.99

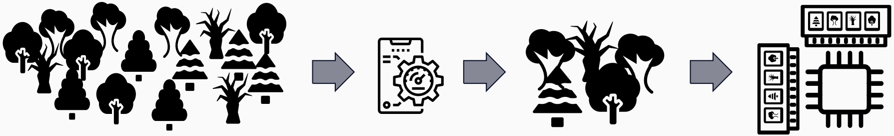
Can we compute a small *and* accurate tree ensemble?

Ensemble Pruning Revisited

Idea 1 Given a large forest with M trees select only a few trees

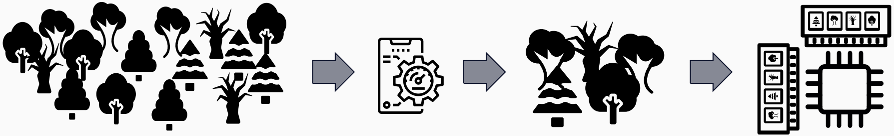
Ensemble Pruning Revisited

Idea 1 Given a large forest with M trees select only a few trees



Ensemble Pruning Revisited

Idea 1 Given a large forest with M trees select only a few trees



Formally

$$f_w(x) = \frac{1}{K}\sum_{i=1}^{M} w_i h_i(x)$$

solve

$$\mathop{\arg\min}_{w\in\{0,1\}^M} \sum_{(x,y)\in\mathcal{S}} \ell\left(f_w(x), y\right) \ \text{s.t.}\ \|w\|_0 = K \ll M$$

Ensemble Pruning Revisited (2)

Ensemble Pruning Standard method to select fewer trees in a forest

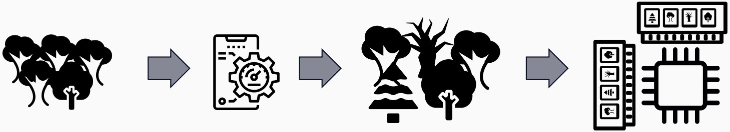
- **Ranking**[Martínez-Muñoz and Suárez 2004, Li et al. 2012, Margineantu and Diettereich 1997]
  Assign a score to each tree and select the top-k trees
- **Clustering**[Giacinto et al. 2000, Bakker and Heskes 2003, Lazarevic and Obradovic 2001, ...]
  Cluster trees and then select a representative from each cluster
- **MQIP**[Cavalcanti et al. 2016, Zhang et al. 2006]
  Construct Mixed Quadratic Integer Program to select trees
- **Ordering**[Jiang et al. 2017, Lu et al. 2010, Margineantu and Dietterich 1997, ...]
  Order the trees according to their overall contribution and select the first K trees

Leaf-Refinement

Idea 2 Use a small forest from the beginning and refine it[Ren et al. 2015, Buschjäger and Morik 2021]

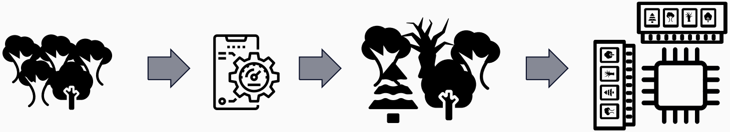
Leaf-Refinement

Idea 2 Use a small forest from the beginning and refine it[Ren et al. 2015, Buschjäger and Morik 2021]



Leaf-Refinement

Idea 2 Use a small forest from the beginning and refine it[Ren et al. 2015, Buschjäger and Morik 2021]



Formally Perform SGD on the leaf nodes $\theta_i = (y_{i,1}, \ldots, y_{i,L_i}), \quad \theta = [\theta_1, \ldots, \theta_M]$

$$\arg\min_{\theta \in \mathbb{R}^{M \cdot L_1 \cdots L_M}} \sum_{(x,y) \in \mathcal{S}} \ell\left(f_\theta(x), y\right)$$

Leaf-Refinement and Pruning combined

Why not combine both approaches?

Leaf-Refinement and Pruning combined

Why not combine both approaches?

$$\underset{\substack{w \in [0,1]^M \\ \theta \in \mathbb{R}^{M \cdot L_1 \cdots L_M}}}{\arg\min} \sum_{(x,y) \in \mathcal{S}} \ell\left(f_{w,\theta}(x), y\right) + \lambda \|w\|_1$$

Leaf-Refinement and Pruning combined

Why not combine both approaches?

$$\underset{\boxed{\substack{w\in[0,1]^M \\ \theta\in\mathbb{R}^{M\cdot L_1\dots L_M}}}}{\arg\min} \sum_{(x,y)\in\mathcal{S}} \ell\left(f_{w,\theta}(x), y\right) + \lambda\|w\|_1$$

Relaxed Constraints

Leaf-Refinement and Pruning combined

Why not combine both approaches?

$$\underset{\boxed{\substack{w \in [0,1]^M \\ \theta \in \mathbb{R}^{M \cdot L_1 \ldots L_M}}}}{\arg\min} \sum_{(x,y) \in \mathcal{S}} \ell\left(f_{\boxed{w,\theta}}(x), y\right) + \lambda \|w\|_1$$

Relaxed Constraints

Optimization over both parameters

Leaf-Refinement and Pruning combined

Why not combine both approaches?

$$\underset{\boxed{\substack{w\in[0,1]^M \\ \theta\in\mathbb{R}^{M\cdot L_1\cdots L_M}}}}{\arg\min} \sum_{(x,y)\in\mathcal{S}} \ell\left(f_{\boxed{w,\theta}}(x), y\right) + \boxed{\lambda\|w\|_1}$$

Relaxed Constraints

Optimization over both parameters

Regularization to enforce pruning

Challenge Constraint optimization

Proximal Gradient Descent

Goal

$$\operatorname*{arg\,min}_{\substack{w\in[0,1]^M \\ \theta\in\mathbb{R}^{M\cdot L_1\cdots L_M}}} \sum_{(x,y)\in\mathcal{S}} \ell\left(f_{w,\theta}(x), y\right) + \lambda\|w\|_1$$

Proximal Gradient Descent

Goal

$$\underset{\substack{w \in [0,1]^M \\ \theta \in \mathbb{R}^{M \cdot L_1 \cdots L_M}}}{\arg\min} \; g(w, \theta) + \lambda \|w\|_1$$

Proximal Gradient Descent

Goal

$$\underset{\substack{w \in [0,1]^M \\ \theta \in \mathbb{R}^{M \cdot L_1 \cdots L_M}}}{\arg\min} \; g(w, \theta) + \lambda R(w, \theta)$$

Proximal Gradient Descent

Goal

$$\arg\min_{\beta} g(\beta) + \lambda R(\beta)$$

Proximal Gradient Descent

Goal

$$\arg\min_{\beta} g(\beta) + \lambda R(\beta)$$

where

- $g(\beta)$ is the differentiable objective
- $R(\beta)$ is a potentially non-differentiable and non-smooth regularizer

Proximal Gradient Descent

Goal

$$\arg\min_{\beta} g(\beta) + \lambda R(\beta)$$

where

- $g(\beta)$ is the differentiable objective
- $R(\beta)$ is a potentially non-differentiable and non-smooth regularizer

then we perform an SGD-like algorithm

$$\beta_{t+1} \leftarrow \mathcal{P}_{R,\lambda}\left(\beta_t - \alpha_t \frac{1}{\|\nabla_{\beta_t} g_{\mathcal{B}}(\beta_t)\|} \nabla_{\beta_t} g_{\mathcal{B}}(x_t)\right)$$

$$\mathcal{P}_{R,\lambda}(\beta) = \arg\min_{z \in \mathbb{R}^K} R(z) + \frac{1}{2\lambda}\|z - \beta\|_2^2$$

Proximal Gradient Descent (2)

Solve

$$\mathcal{P}_R(\beta, \lambda) = \arg\min_{z \in \mathbb{R}^K} R(z) + \frac{1}{2\lambda}\|z - \beta\|_2^2$$

For example

$$R(\beta) = \|\beta\|_0 : P_{R,\lambda}(\beta)_i = \begin{cases} \beta_i & if |\beta_i| \geq \sqrt{2\lambda} \\ 0 & else \end{cases}$$

$$R(\beta) = \|\beta\|_1 : P_{R,\lambda}(\beta)_i = \mathrm{sgn}(\beta_i) max(0, |\beta_i| - \lambda)$$

Putting it all together

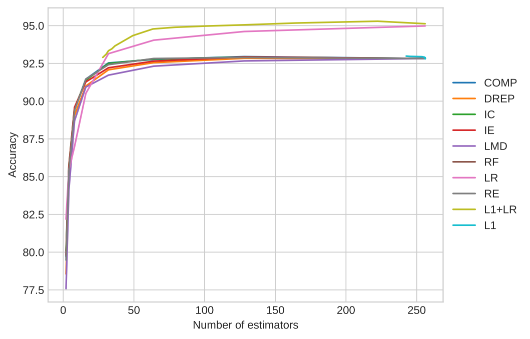
function PRUNE_AND_REFINE($\mathcal{T}, h_1, \ldots, h_M$)
    $\theta_1, \ldots, \theta_M \leftarrow$ ***get_leafs***$(h_1, \ldots, h_M)$ ▷ Load leafs
    $w_1, \ldots, w_M \leftarrow$ ***get_weights***$(h_1, \ldots, h_M)$ ▷ Load weights
    for epoch $1, \ldots, E$ **do** ▷ Perform PSGD for E epochs
        for next batch $\mathcal{B}$ in epoch **do**
            $w \leftarrow w - \alpha g_{\mathcal{B}}(w)$ ▷ Update weights
            $\theta \leftarrow \theta - \alpha g_{\mathcal{B}}(\theta)$ ▷ Update leafs
            $w \leftarrow \mathcal{P}_\lambda(w)$ ▷ Apply the ***prox*** operator
    $H \leftarrow \emptyset, W \leftarrow \emptyset$
    for $i = 1, \ldots, M$ **do**
        if $w_i \neq 0$ **then**
            h_i.***update_leafs***(θ_i) ▷ Copy new leafs into original trees
            $H \leftarrow H \cup \{h_i\}$
            $W \leftarrow W \cup \{w_i\}$
    return H, W

Experiment 1: Compare with Vanilla Random Forest

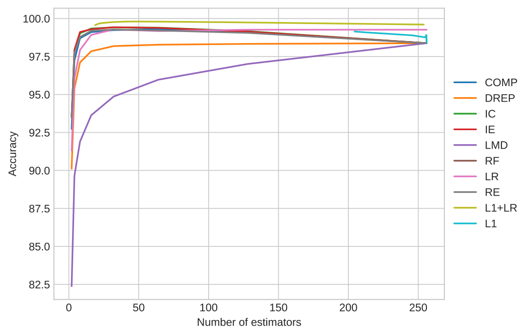
		adult	avila	bank	eeg	elec	mnist
RF	accuracy [%]	86.78	98.58	90.39	93.42	88.98	96.53
	model size [MB]	24.99	32.85	24.99	14.95	24.99	56.99
LR+L1	accuracy [%]	87.25	99.78	90.5	95.55	92.49	98.05
	model size [MB]	0.06	3.52	0.07	5.88	14.37	28.49

Experiments 2: Compare against Ensemble Pruning

EEG Dataset

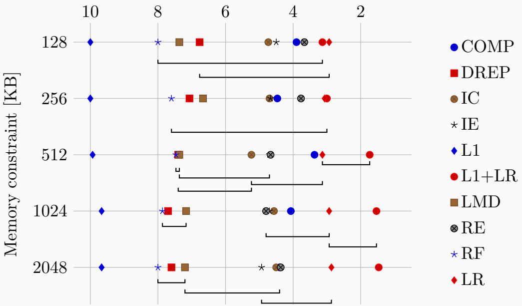


Avila



Experiments 3: Perform systematic experiments on more datasets

Comparison with more algorithms on more datasets 15 datasets, 10 methods, 920 hyperparameter configs per datasets ⇒ 13 800 models cross-validated



Conclusion (1)

We should use smaller hardware / use existing hardware longer

- 80% of the CO_2 procured during the life-cycle of an IPhone 14 are due to its production
- To break even between manufacturing and usage, we need to use an IPhone for 13 years

Tree ensembles are a perfect fit for older devices, but still too large

- Ensemble Pruning removes redundant members, making ensembles smaller and better
- Leaf-Refinement refines prob. estimates in the leaves, making small ensembles better

Conclusion (2)

Leaf-Refinement and Ensemble Pruning combined

- We can combine Leaf-Refinement and Ensemble Pruning via an L_1 regularization term
- Proximal Gradient Descent is the ideal algorithm for refinement and pruning
- Our novel method outperforms existing methods on a variety of datasets

Check out our software

https://github.com/sbuschjaeger/Pypruning/

https://github.com/sbuschjaeger/leaf-refinement-experiments