DED SOFTWARE

Active learning and re-training of tree models

AGENDA

- Active Learning
- Query Strategies
- Learning to Sample Framework
- Adaptive tree learning

Active Learning

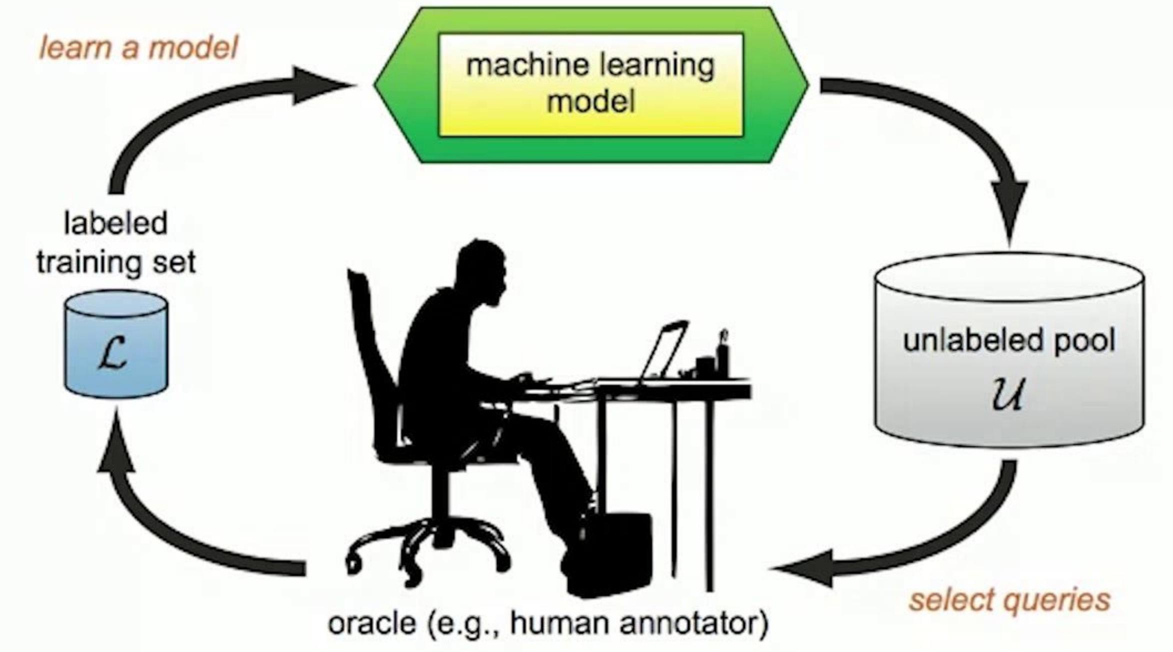


Figure 1: The pool-based active learning cycle.

Query Strategies

Uncertainty Sampling

$$\overline{v}_i = \max_{j=1,\ldots,c} \{v_{ij}\}$$

If sample $\mathbf{x}_i$ get the maximum vote on class p and second maximum vote on class q, namely

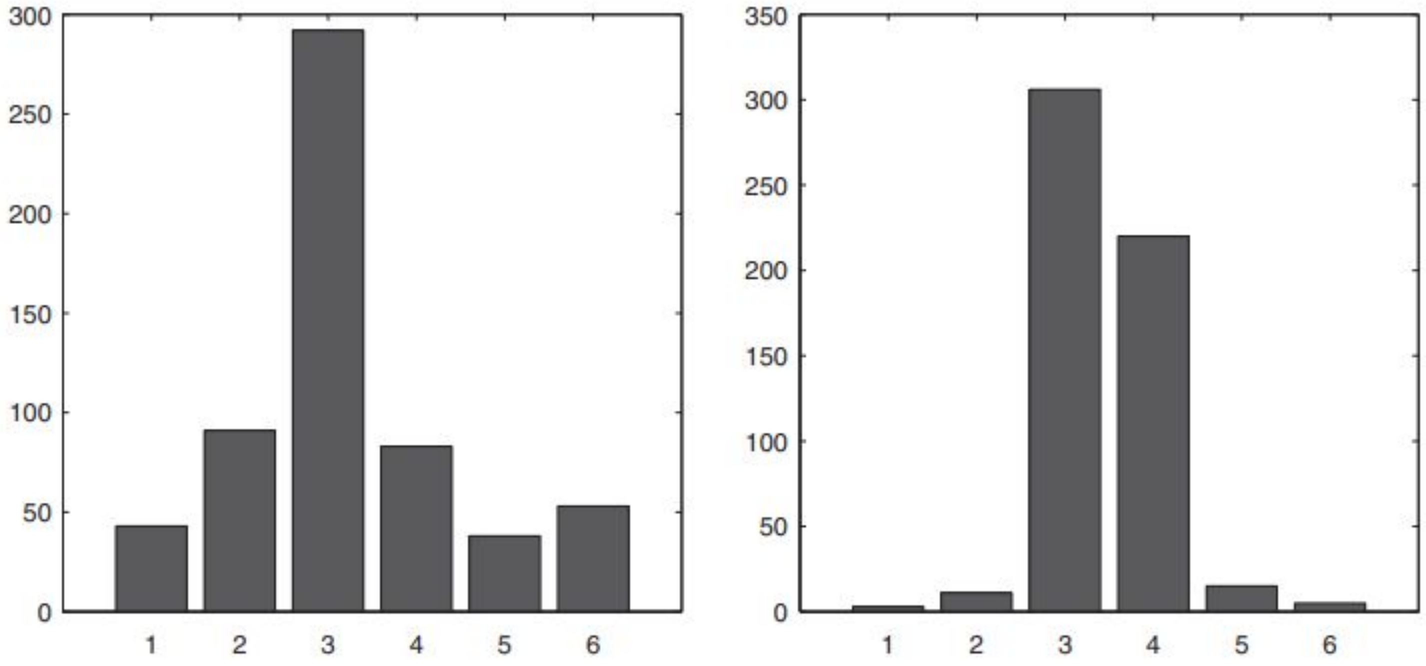
$$p = \arg\max_{j=1,\ldots,c} \{v_{ij}\}$$

$$q = \arg\max_{j=1,\ldots,c,\, j\neq p} \{v_{ij}\}$$

The difference between the maximum vote and the second maximum vote is

$$unc_i = v_{ip} - v_{iq} \quad (2)$$

unc_i is able to measure uncertainty of sample $\mathbf{x}_i$.



Query Strategies

Representative (Density) Sampling

For any $x_i \in \mathcal{U}$, define its k nearest neighbors from $\mathcal{U}$ as $x_{i_j}, j = 1, ..., k, x_{i_j} \in \mathcal{U}$. The average distance from $\mathbf{x}_i$ and its k nearest neighbors can be computed:

$$den_i = \frac{1}{k}\sum_{j=1}^{k}\|x_i - x_{i_j}\|^2 \quad (3)$$

We use den_i to measure the density of sample $\mathbf{x}_i$.

Query Strategies

Diversity sampling

For any $x_i \in \mathcal{U}$, compute the distance between $\mathbf{x}_i$ and its nearest labeled neighbor:

$$div_i = \min_{j=1,2,\ldots,n} \|x_i - x_j\|^2 \quad (4)$$

$$x_i \in \mathcal{U}, x_j \in \mathcal{L}$$

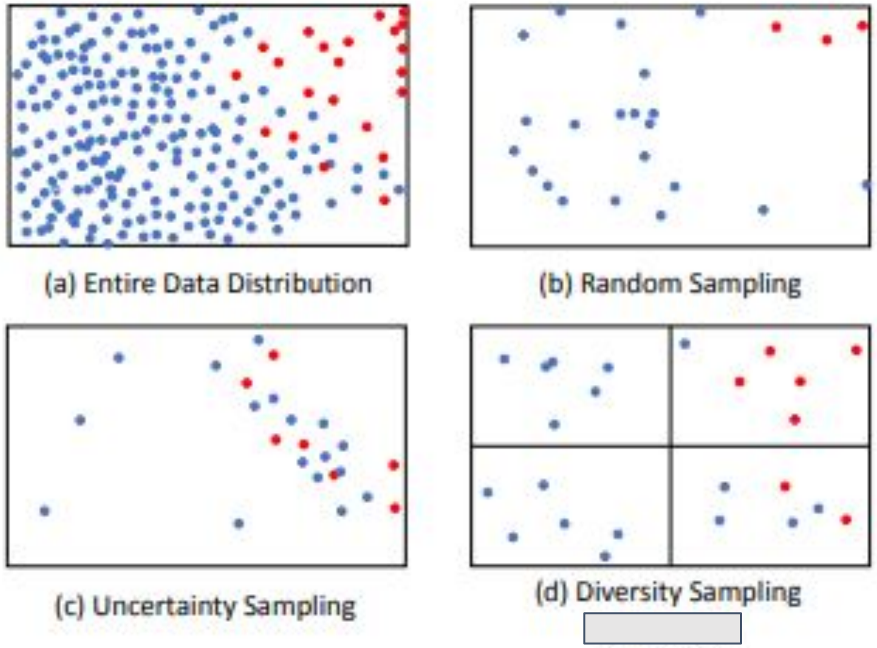


Fig. 3: Comparison of different sampling strategies, where 24 samples are selected in each of (b), (c) and (d).

Learning to Sample: an Active Learning Framework

Jingyu Shao, Qing Wang and Fangbing Liu
Research School of Computer Science Australian National University
2019

Learning to Sample: an Active Learning Framework

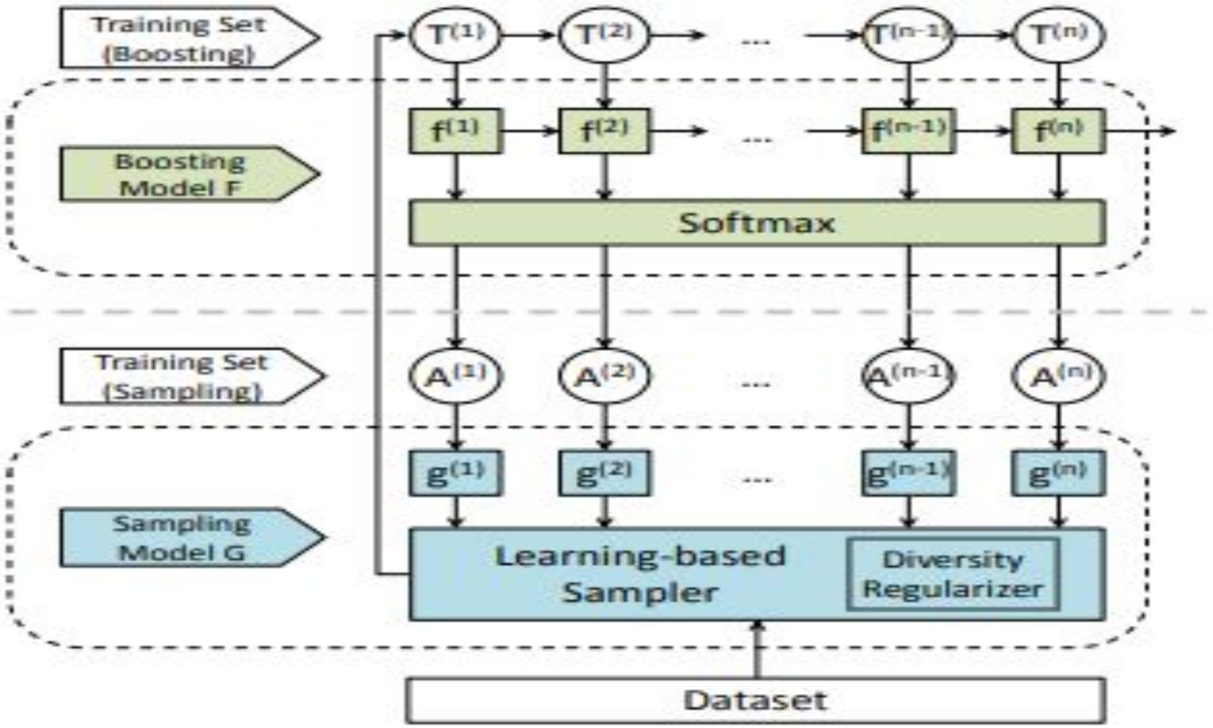


Fig. 2: The overall framework of Learning To Sample (LTS)

Boosting model

More specifically, the individual results of the first t-1 functions are combined to predict the label of an instance at the (t-1)-th iteration such that:

$$\hat{y}_i^{(t-1)} = \sum_{k=1}^{t-1} f^{(k)}(x_i). \quad (1)$$

Then, the t-th function $f^{(t)}$ is trained on the actively selected training subset $T^{(t)}$ by minimizing the following objective function:

$$\sum_{(x_i,y_i)\in T^{(t)}} \ell_1(\hat{y}_i^{(t-1)} + f^{(t)}(x_i), y_i) + \Omega_1(f^{(t)}) \quad (2)$$

Sampling model

$$\begin{aligned} \textbf{maximize} \quad & \sum_{i=1}^{k} v_i g^{(t)}(x_i) + \alpha \times \Gamma(\mathbf{v}) \\ \textbf{subject to} \quad & ||\mathbf{v}||_1 = |\Delta^{(t)}| \end{aligned}$$

where $k = |X_U^{(t)}|$, $\mathbf{v} = (v_1, ..., v_k)^T \in \{0, 1\}^k$, and each v_i is associated with an instance $x_i \in X_U^{(t)}$. When $v_i = 1$, it indicates that x_i is selected as a sample, and conversely, $v_i = 0$ indicates that x_i is not selected. The term $g^{(t)}(x_i)$ indicates the uncertainty score of an instance x_i which is predicated by a regressor $g^{(t)}$, and the regularization term $\Gamma(\mathbf{v})$

Learning to Sample: an Active Learning Framework

Uncertainty Sampling:

$$\sum_{(x_i, z_i^{(t)}) \in A^{(t)}} w_i^{(t)} \ell_2(g^{(t)}(x_i), z_i^{(t)}) + \Omega_2(g^{(t)})$$

Diversity Sampling:

$$\Gamma(\mathbf{v}) = ||\mathbf{v}||_{2,1} = \sum_{j=1}^{b} ||\mathbf{v}_j||_2$$

Algorithm 1: Learning To Sample (LTS)

Input: X with k groups, i.e. $\sum_{i=1}^{k} X_i^{(0)} = X$; label budget ζ;
Balancing parameter α; Number of iterations n;

Output: A boosting model F

Initialize $T^{(0)} = \emptyset$

Select a set of seed samples $\Delta^{(0)}$ from k groups to maximize $\Gamma(\mathbf{v})$, where $|\Delta^{(0)}| = \frac{\zeta}{n}$

for $t = 1, \ldots, n$ **do**

  Update $T^{(t)} = T^{(t-1)} + \Delta^{(t-1)}$

  Train an additive function $f^{(t)}$ by minimizing the objective in Eq. 2 using $T^{(t)}$

  Generate a training set $A^{(t)}$

  Train a regression function $g^{(t)}$ by minimizing the objective in Eq. 5 using $A^{(t)}$

  Update $X_i^{(t)} = \{x \in X_i^{(t-1)} | x \notin \Delta^{(t-1)}\}$, where $i = 1, \ldots, k$

  Select a set of samples $\Delta^{(t)}$ from $\sum_{i=1}^{k} X_i^{(t)}$ by maximizing the objective in Eq. 4, with $|\Delta^{(t)}| = \frac{\zeta}{n}$

$$k = \lceil \sqrt[d]{\frac{\zeta}{n}} \rceil^{d}$$

Learning to Sample: an Active Learning Framework

Experiments

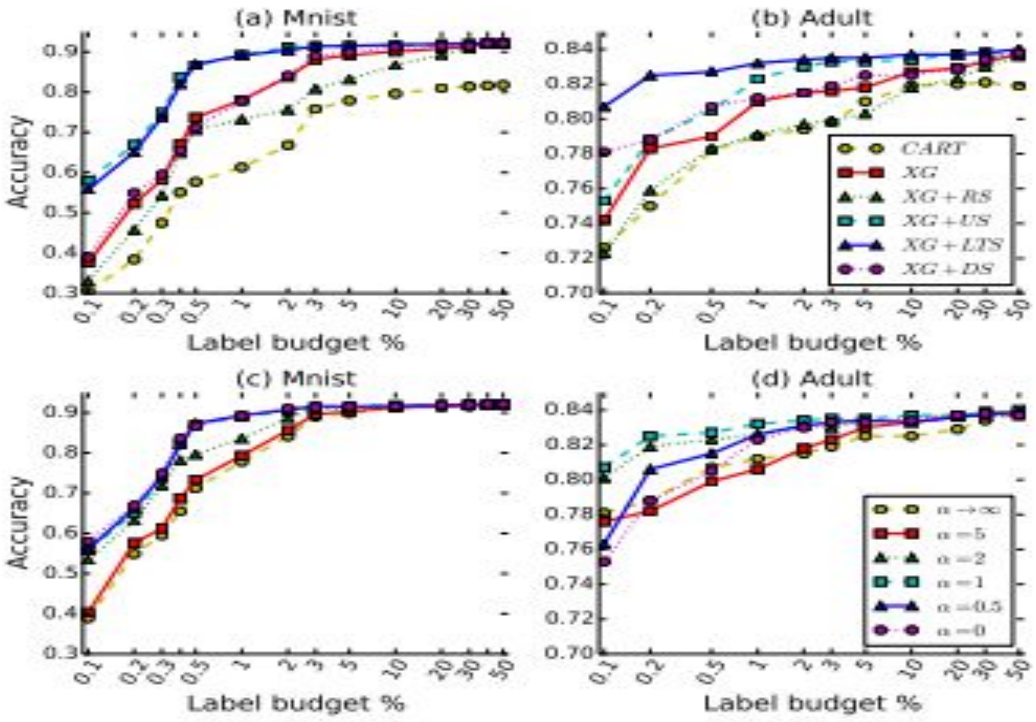


TABLE I: Characteristics of datasets

Classification Tasks	Datasets	# Attributes	# Instances ($\|X\|$)	# Classes	Types of Labels	Class Imbalance Ratio
Image classification	Mnist	28 × 28	60,000	10	10 digits (i.e. 0-9)	N/A
Salary level prediction	Adult	14	48,842	2	{above 50k, not above 50k}	1 : 3
Entity resolution	Cora	12	837,865	2	{match, non-match}	1 : 49
	DBLP-Scholar	4	168,112,008	2	{match, non-match}	1 : 71,233
	DBLP-ACM	4	6,001,104	2	{match, non-match}	1 : 2,698
	NCVoter	18	10M	2	{match, non-match}	1:420

Experiments

TABLE II: Comparison of f-measure results for entity resolution tasks under different label budgets

Dataset	Label Budget ζ (% of $\|X\|$)	CART	XG	XG+RS	XG + US $\alpha = 0$	XG+LTS $\alpha = 0.5$	XG+LTS $\alpha = 1$	XG+LTS $\alpha = 2$	XG+LTS $\alpha = 5$	XG + DS $\alpha \to \infty$	XG + LTS(E) $\alpha = 1$
Cora	0.01	0	0	0	0	0.637	0.857	0.861	0.867	**0.878**	0.862
	0.05	0.741	0.763	0.750	0.827	0.851	0.864	0.870	0.883	**0.885**	0.867
	0.1	0.788	0.796	0.787	0.823	0.863	0.862	0.873	**0.887**	0.886	0.870
	0.5	0.848	0.835	0.835	0.873	0.893	**0.900**	0.895	0.895	0.893	0.890
	1	0.868	0.878	0.880	0.870	0.896	0.902	**0.904**	0.898	0.894	0.896
	5	0.878	0.897	0.892	0.907	0.912	**0.915**	0.913	0.902	0.898	0.904
NCVoter	0.01	0	0	0	0	0.403	0.324	0.403	0.752	**0.875**	0.571
	0.05	0	0	0	0	0.903	0.954	0.989	**0.993**	0.991	0.934
	0.1	0	0	0	0	0.989	**0.994**	0.993	0.993	0.993	0.993
	0.5	0	0	0	0	0.993	**0.994**	0.993	0.993	0.991	**0.994**
	1	0.334	0.379	0.398	0	0.993	0.993	0.993	0.992	**0.994**	0.993
	5	0.993	0.993	0.994	0.993	0.993	**0.997**	0.993	0.994	0.993	0.994
DBLP-ACM	0.1	0	0	0	0	0	0	0	0	**0.397**	0
	0.5	0	0	0	0	0.382	0.702	**0.720**	0.651	0.632	0.679
	1	0.348	0.347	0.279	0	0.813	**0.878**	0.778	0.730	0.721	0.793
	2	0.599	0.767	0.680	0.403	0.851	**0.884**	0.867	0.789	0.783	0.854
	5	0.870	0.850	0.803	0.874	**0.935**	0.931	0.889	0.837	0.833	0.891
	10	0.903	0.911	0.890	0.926	**0.983**	0.981	0.937	0.893	0.899	0.933
DBLP-Scholar	0.1	0	0	0	0	0.586	0.723	0.733	**0.741**	0.731	0.727
	0.5	0.378	0.54	0.498	0.555	0.764	0.773	**0.794**	0.790	0.780	0.781
	1	0.562	0.669	0.659	0.738	0.793	0.804	**0.808**	0.793	0.792	0.794
	2	0.772	0.806	0.771	0.807	0.810	**0.815**	0.813	0.799	0.801	0.811
	5	0.773	0.822	0.803	0.836	**0.838**	0.836	0.831	0.821	0.818	0.828
	10	0.808	0.835	0.830	**0.865**	0.859	0.851	0.844	0.837	0.829	0.853

hi-RF: Incremental Learning Random Forest for large-scale multi-class Data Classification

Tingting Xie, Yuxing Peng, Changjian Wang
National Lab for Parallel and Distributed Processing, School of Computer, National University of Defense Technology
2016

hi-RF: Incremental Learning Random Forest for large-scale multi-class Data Classification

Heterogeneous incremental nearest class mean random forest

Algorithm 1 Heterogeneous incremental Nearest Class Mean Random Forest (hi-RF)

Input:
- Previous model, m;
- Number of decision trees in m, s;
- The set of out-of-bag error for each tree, O;
- Old training data, D_o;
- New training data, D_n;

Output:
- New model, M;

```
for each time new data arriving do
    threshold ← OOB_estimation(O)
    for each tree T_i in m do
        if T_i does not reach the threshold then
            T_i ← Retraining(D^o, D^n)
        else
            T_i ← Updating(D^o, D^n)
        end if
    end for
    O ← OOB_boosting(O)
    M ← Bagging(T_1, T_2, ..., T_s)
end for
return M
```

OOB estimation

Algorithm 2 OOB estimation

Input:

The whole RF, T;

Bootstrap samples for each tree, D_i $(i \in 1, 2, ..., s)$;

Old training data, D_o;

New training data, D_n;

Output:

The threshold, δ;

for each tree T_i in T **do**
  $D = D^o + D^n$
  $D^l = D - D_i$ // D^l:left-out sample set
  for (x, y) in D^l **do**
    $\hat{y} \leftarrow T_i(x)$
    if $\hat{y} = y$ **then**
      $I\{\hat{y} = y\} = 1$ // $I\{\hat{y} = y\}$:loss function
    else
      $I\{\hat{y} = y\} = 0$
    end if
  end for
  $o_i = \frac{\sum_{(x,y) \in D^l} I\{\hat{y}=y\}}{|D^l|}$ // calculate the out-of-bag error for T_i
end for
$O \sim N(\mu, \sigma^2)$
$(\mu, \sigma^2) \leftarrow MaxLikelihoodEstimation(O, \mu, \sigma^2)$
$\delta = \mu$
return δ

$$L(o_1, o_2, ..., o_s; \mu, \sigma) = \prod_{i=1}^{s} p(o_i) = \prod_{i=1}^{s} \frac{1}{\sqrt{2\sigma^2\pi}} \cdot e^{-\frac{(o_i-\mu)^2}{2\sigma^2}}$$

$$o = o + \alpha * tanh(o)$$

hi-RF: Incremental Learning Random Forest for large-scale multi-class Data Classification

Rolling release NCM decision trees (RRN)

Algorithm 3 Rolling release NCM decision tree(RRN)

Input:
The previous random forest, T^o;
The out-of-bag error for each tree, o_i;
The threshold, δ;
All training data, D;

Output:
The new random forest, T^n;

```
for each tree T_i in T^o do
  if o_i > δ then
    Discarding(T_i)
    D_i ← Bootstrap(D)
    // The growing a new NCM decision tree
    T_i ← Growing(D_i) :
    if all the (x, y) in D_i has the same label k or reach the max Depth then
      return  classes probabilities P
    else
      // K is a random subset of the classes observed in D_n
      K ← ClassesSubset(D_i)
      class centroids θ_n ← CalClassCentroids(D_n)
      K_left, K_right ← ChooseBestFeature(D, θ_n) according to Information gain
      D_left, D_right ← SplitDataSet(D, K_left, K_right, θ_n)
      build subtree:T_left = Growing(D_left), T_right = Growing(D_right)
      T ← K_left : T_left + K_right : T_right
    end if
    return  T
  end if
end for
T^n ← {T_1, T_2, ..., T_s}
```

$$\theta_n^k = \frac{1}{|D_n^k|} \cdot \sum_{i \in D_n^k} x_i$$

$$Gain(D_n, f) = E_n - \sum_{i \in \{left, right\}} E_i$$

$$\hat{y} = \underset{k \in K}{\operatorname{argmin}} ||x - \theta_n^k||^2$$

ReGenerate leaves probabilities (RLP)

Algorithm 4 ReGenerate leaves probabilities (RLP)

Input:
The previous random forest, T^o;
The out-of-bag error for each tree, o_i;
The threshold, δ;
All training data, D;

Output:
The new random forest, T^n;

for each tree T_i in T^o **do**
  if $o_i <= \delta$ **then**
    $Dismiss(T_i, leavesprobabilities)$
    $D_i \leftarrow Bootstrap(D)$
    $K \leftarrow ClassesSet(D_i)$
    // The updating of a decision tree
    Define $T_i \leftarrow Updating(T_i, D_i)$:
    $n\_node \leftarrow NumberOfLeavesNode(T_i)$
    $\{D_i^1, D_i^2, ..., D_i^{n\_node}\} \leftarrow Fall(T_i, D_i)$
    for $D_i^j \in \left\{D_i^1, D_i^2, ..., D_i^{|K|}\right\}$ **do**
      for k in K **do**
        $D_i^j(k) \leftarrow DatasetOfClassK(D_i^j, k)$
        $P_i^j(k) = \frac{|D_i^j(k)|}{|D_i^j|}$
      end for
      $P_i^j = \left\{P_i^j(1), P_i^j(2), ..., P_i^j(|K|)\right\}$
    end for
    $P_i = \{P_i^1, P_i^2, ..., P_i^{n\_node}\}^{\mathrm{T}}$
    update P_i to T_i
    EndDefine
  end if
  $T^n \leftarrow \{T_1, T_2, ..., T_s\}$
end for

Experiments

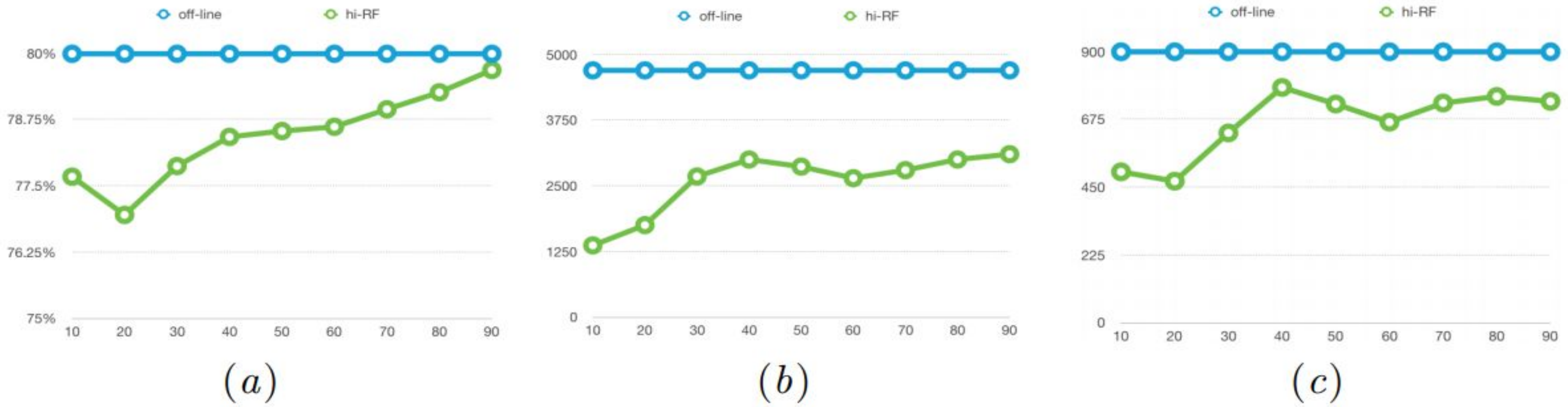


Figure 2: Comparison between baseline and hi-RF, while the original class number is range from 10 to 90, and the added data range from 90 to 10, and the final data class number is 100 a) Accuracy b) Training time c) Testing time

Experiments

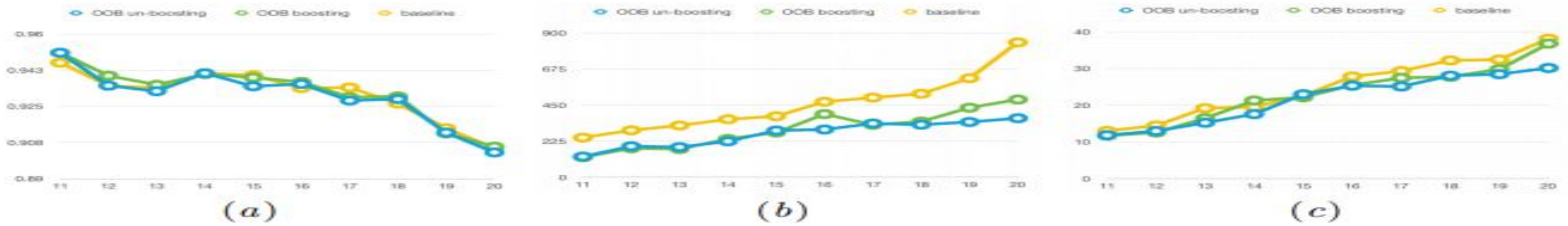


Figure 3: Comparison among baseline, hi-RF with OOB boosting and OOB unboosting, while the original class number is range from 10 to 20, and the step size is 1 a) Accuracy b) Training time c) Testing time

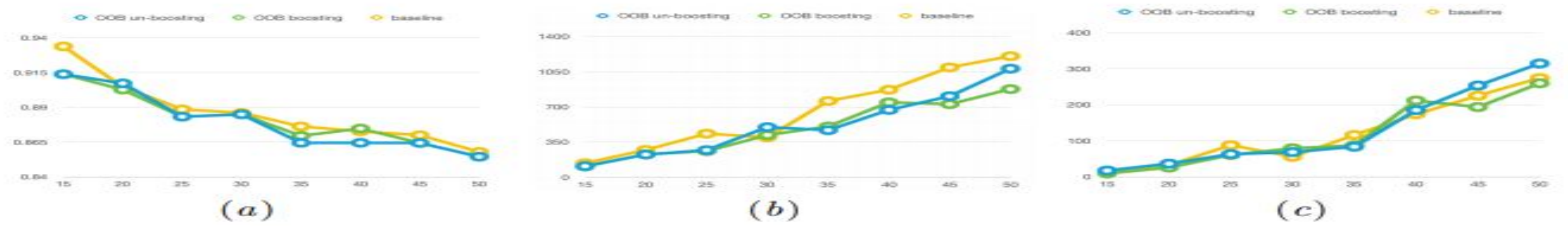


Figure 4: Comparison among baseline, hi-RF with OOB boosting and OOB unboosting, while the original class number is range from 10 to 50, and the step size is 5 a) Accuracy b) Training time c) Testing time

Summary

Q & A

Dziękuję

LiTL

Karol Woźniak
karol.wozniak@qed.pl