Statistics for Data Science: Project n. 3

Splitting with Confidence in Decision Trees with Application to Stream Mining

Submitted by:

Camilla Andreazzoli
Davide Ricci
Davide Vecchi

Introduction

Stream Mining

A setting in which confidence intervals for splits are extremely useful is **online** or **stream-based learning:** the examples are received incrementally, and a confidence interval can be used to decide how much data should be collected at a certain leaf before a good split can be safely identified.

Challenges

Has enough data accumulated within a leaf node to be sure that the attribute chosen to split the node is really the most advantageous one?
Is now the right time to split the node or is it necessary to wait for other instances to arrive in order to make statistically significant splitting decisions?
Is it possible to obtain meaningful and reliable estimates even with only a few instances?

Proposed solutions

Confidence intervals will be used to make safe decisions in the context described above, analyzing in particular two algorithms that exploit the **Hoeffding Inequality**: **Hoeffding Tree** and **C-tree**.

Hoeffding Bound

Theorem (Hoeffding's Inequality) Let $Y_1, \ldots, Y_n$ *be iid observation such that* $E(Y_i) = \mu$ *and* $a \leq Y_i \leq b$. *Then, for any* $\epsilon > 0$,

$$\mathbb{P}(|\bar{Y}_n - \mu| \geq \epsilon) \leq 2e^{-2n\epsilon^2/(b-a)^2}$$

Corollary *If* $X_1, X_{2,\ldots,} X_n$ *are independent with* $\mathbb{P}(a \leq X_i \leq b) = 1$ and common mean μ, *then, with probability at least* $1 - \delta$,

$$|\bar{x}_n - \mu| \leq \sqrt{\frac{c}{2n}\log\left(\frac{2}{\delta}\right)}$$

where $c = (b-a)^2$

In our case the Hoeffding Bound will be applied to define appropriate confidence intervals to estimate the optimal gain for the splitting decision in a decision tree learning.

MAIN NOTATION

- $\varepsilon = \sqrt{\frac{c}{2n}\log\left(\frac{2}{\delta}\right)}$
- G is the true gain for the available split (it is unknown)
- $\Delta\bar{G}$ is the difference between the gain of first and second best split
- ΔG is the true gain difference
- $\mathcal{F}$ is the class of binary split functions
- m is the number of the examples routed to the node
- $\hat{\phi}_{i|F}$ is the estimate of the impurity obtained by splitting the i^{th} node using the split function F
- $Y_{|i}$ is the label (class) of an instance based in the node i

H-Tree and C-tree

- **H-Tree** algorithm exploits the Hoeffding Bound to decide whether a current decision node should be replaced by a new attribute:
if n examples have been seen at this node and $\Delta\bar{G} > 2\varepsilon$, then the attribute choosed for the split allows the best split with probability 1 − δ. The gain for a split F at node i, written in terms of a generic ϕ impurity measure, is:

$$G_{i|F} = \phi\left(Y_{|i}\right) - \phi\left(Y_{|i}\middle|F\right)$$

where the first term refers to the impurity of the parent node and the second term to the impurity given by the split F. The goal is therefore to choose the split F that allows to obtain the maximum gain. Since the first term $\phi\left(Y_{|i}\right)$ doesn't depend from F, we can ignore it when estimating the gain, focusing only on the children impurity.

- **C-tree** follows the idea of H-tree, moreover in addiction to H-Tree adds a new parameter τ which adds a further condition for finding the best split:
if $\Delta\bar{G} > 2\varepsilon$ or $\varepsilon < \tau$,then the split is performed. This second condition means that the difference between the two gains is so tiny that waiting for more example in order to find out the really best split is not worthwhile.

Theorems

- The Hoeffding Bound provides the maximum distance ε between the impurity estimate given by the split $\hat{\phi}_{i,F}$ and the true impurity, and has been defined in three different ways.

- **Theorem 1**,**2**,**3** define the maximum distances using as splitting criterion Entropy, Gini Index, and a 3rd index called "KM" while **Theorem 4** is about C-tree and provides an upper bound on the probability of getting a τ-suboptimal split.

Th 1: Bound for Entropy

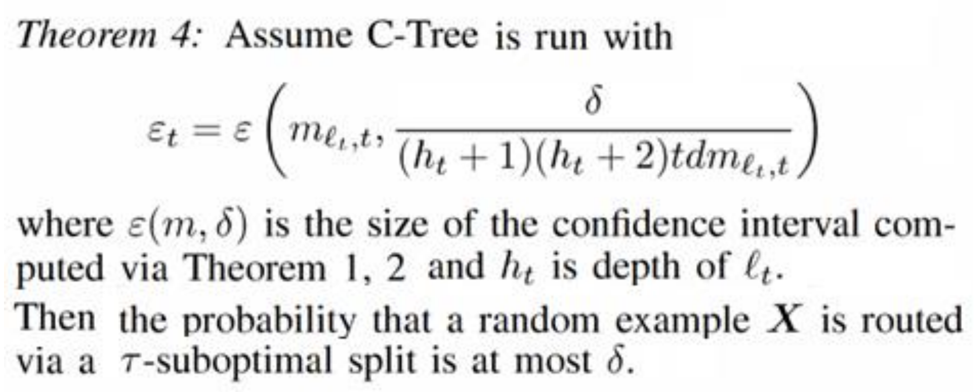
$$\varepsilon_{ent}(m,\delta) = (\ln m)\sqrt{\frac{2}{m}\ln\frac{4}{\delta} + \frac{2}{m}}$$

Th 2: Bound for Gini Index

$$\varepsilon_{Gini}(m,\delta) = \sqrt{\frac{8}{m}\ln\frac{2}{\delta}} + 4\sqrt{\frac{1}{m}}$$

Th 3: Bound for Kernel Mansour Index

$$\varepsilon_{KM}(m,\delta) = 4\sqrt{\frac{1}{m}\ln\frac{8}{\delta}}$$



Theorem 4: Assume C-Tree is run with

$$\varepsilon_t = \varepsilon\left(m_{\ell_t,t}, \frac{\delta}{(h_t+1)(h_t+2)tdm_{\ell_t,t}}\right)$$

where $\varepsilon(m,\delta)$ is the size of the confidence interval computed via Theorem 1, 2 and h_t is depth of ℓ_t.
Then the probability that a random example X is routed via a τ-suboptimal split is at most δ.

Paper Experiments

Two types of experiments were performed:
- on controlled scenarios with synthetic data streams
- on real scenarios using five datasets present on the MOA platform

In both experiments, the online performance of the previously described C-Tree algorithm was compared against two other basic algorithms(H-Tree, CorrH-Tree) which always share the hoeffding bound core in their logics.

In particular, compared to the previous analyses, it was decided in the implementation of the C-Tree to neglect the value of the tie-break parameter τ (because it has been observed that it caused the majority of the splits) and to use the following formula for the definition of the confidence interval:

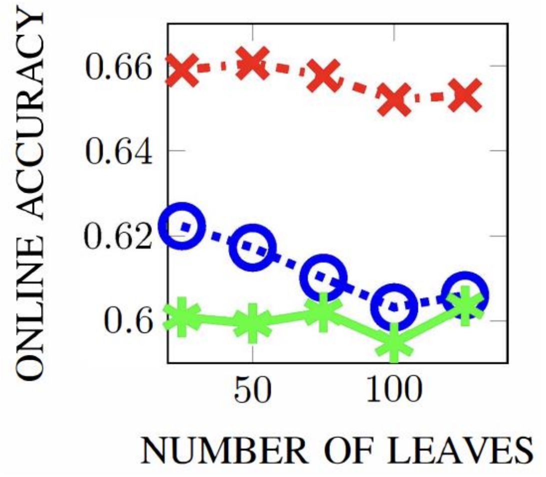
$$\widetilde{\varepsilon}_{\mathrm{KM}} = \widetilde{\varepsilon}_{\mathrm{Gini}} = c\sqrt{\frac{1}{m}\ln\left(m^2h^2td\right)}$$

*parameter c was used to control the number of splits
*Gini Index impurity measure was used for all experiments

Controlled experiments

- In the first type of experiments, the data streams were generated from one of 1000 binary decision trees (with 50 leaves, 5 attributes and numeric values in the range [0,1]), randomly constructed using a suitable generation algorithm.

In the Figure are shown the online performances for the three algorithms averaged over 1000 streams(each generated using a different random binary tree) with respect to the total number of leaves in the tree as the stream was being fed to the algorithm.

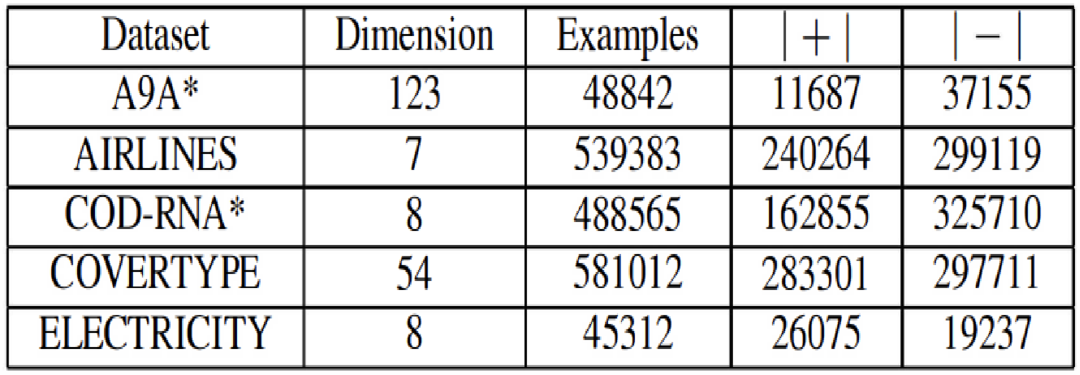


- As can be seen, the C-Tree(red line), performs better than the other two standard algorithm from the beginning.

*C-Tree(red line), H-Tree(blue line), CorrH-Tree(gree line).
*the parameter C for the definition of the Hoeffding Bound in the C-Tree has been tuned on a grid of 200 potential values in the interval (0,2).

Experiments on real-world data

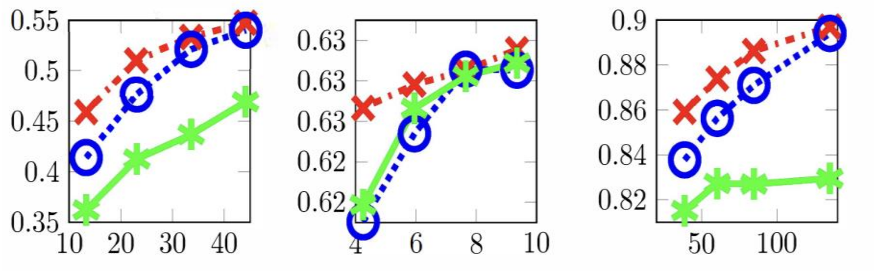
-For these experiments, 10 different streams were generated (realized as permutations of the examples) for each dataset present in the following table.



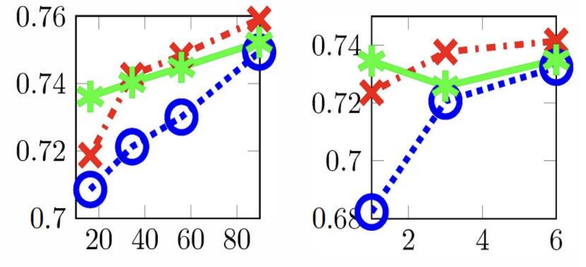
Dataset	Dimension	Examples	|+|	|−|
A9A*	123	48842	11687	37155
AIRLINES	7	539383	240264	299119
COD-RNA*	8	488565	162855	325710
COVERTYPE	54	581012	283301	297711
ELECTRICITY	8	45312	26075	19237

-The procedure for testing the performances of the C-Tree vs the standard ones was the same also in this section.

-Even if the datasets are not particularly large, the plots show that trees generated by the C-Tree algorithm compare favourably with respect to the baselines especially in the first learning phases.



(a) A9A (b) AIRLINES (c) COD-RNA



(d) COVERTYPE (e) ELECTRICITY

Our experiments

Library used:
-"foreign" -> I/O
-"caret" -> performance routine
-"rpart" -> Decision Tree model
-"mlpack" -> Hoeffding Tree model

DATASETS

- Synthetic
- N°rows=from 10^3 to 10^7
- N° columns=10, each from a different distribution

- Real
- Airlines and Electricity
- Both available from MOA platform

In all the experiments:
- ✓ dataset stratified split between training and test set (80%-20%), fixing a seed for the reproducibility of the experiments
- ✓ trained and tested hoeffding_tree from library "mlpack" with default parameters (no parameter tuning), with InfoGain=GiniIndex and Entropy
- ✓ trained and tested a standard decision tree from library "Rpart" with default parameter
- ✓ compared the performances of the classifiers

In the end we defined a pseudo C-tree code with particular focus on the condition of splitting criterion.

Classifiers Comparison

ht_{Gini} -> hoeffding tree with InfoGain=GiniIndex
ht_{Entr} -> hoeffding tree with InfoGain=Entropy
dt -> standard decision tree

SYNTHETIC DATASET (N° rows=10^7)

Classifier	Accuracy	Precision	Recall	F1-score
ht_{Gini}	0.5006323	0.5004539	0.4765451	0.4882069
ht_{Entr}	0.5000093	0.4998077	0.4966442	0.4982209

AIRLINES

Classifier	Accuracy	Precision	Recall	F1-score
ht_{Gini}	0.5974359	0.5408910	0.6410539	0.5867283
ht_{Entr}	0.6122956	0.5621084	0.5894610	0.5754598
dt	0.6290927	0.6762395	0.3222010	0.4364507

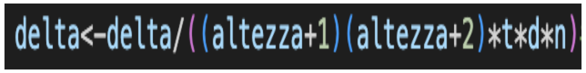
ELECTRICITY

Classifier	Accuracy	Precision	Recall	F1-score
ht_{Gini}	0.7631869	0.7619875	0.8496707	0.8034439
ht_{Entr}	0.7612006	0.7520175	0.8665246	0.8052205
dt	0.7543589	0.7323520	0.8963580	0.8060976

- Since no tuning of the hyperparameters has been done, the Decision Tree model trained on the synthetic dataset is in overfitting and, as a result, performance was poor.
-Typically dt are more effective than ht, as they immediately have the entire dataset available for training, but in presence of a large amount of data they are computationally inefficient.

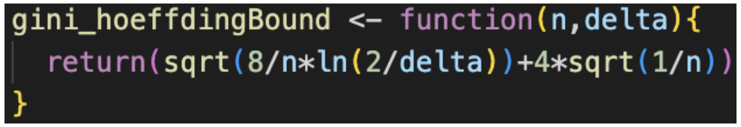
C-Tree in R

As a last experiment, we tried to define some of the logics analyzed in the paper for the C-Tree algorithm within the R environment. Below are the main routines:

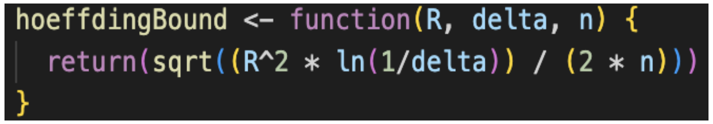
- Following Theorem 4 the redefinition of delta value: 

```
delta<-delta/((altezza+1)(altezza+2)*t*d*n)
```

Where altezza=height of the node in the tree. T=time. D=nm.attributes considered for the potential split. N=nm. examples in the node.

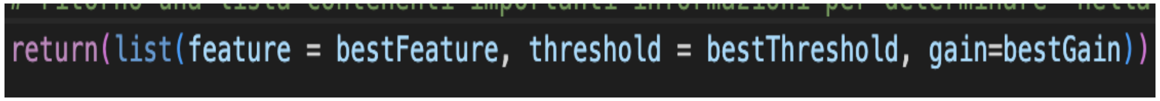
- Following Theorem 2 the redefinition of the confidence intervals based on the Gini Index measure of impurity(compared with the standard of the H-Tree): 

```
gini_hoeffdingBound <- function(n,delta){
  return(sqrt(8/n*ln(2/delta))+4*sqrt(1/n))
}
```



```
hoeffdingBound <- function(R, delta, n) {
  return(sqrt((R^2 * ln(1/delta)) / (2 * n)))
}
```

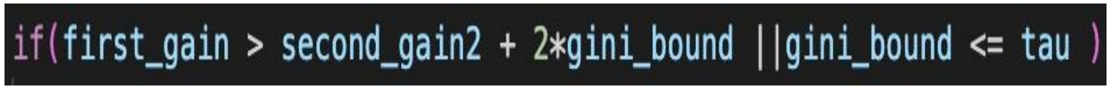
- BestSplit() function that calculate the best attribute and threshold based on Gini Index and return its main information in a list:



```
return(list(feature = bestFeature, threshold = bestThreshold, gain=bestGain))
```

C-Tree in R(part 2)

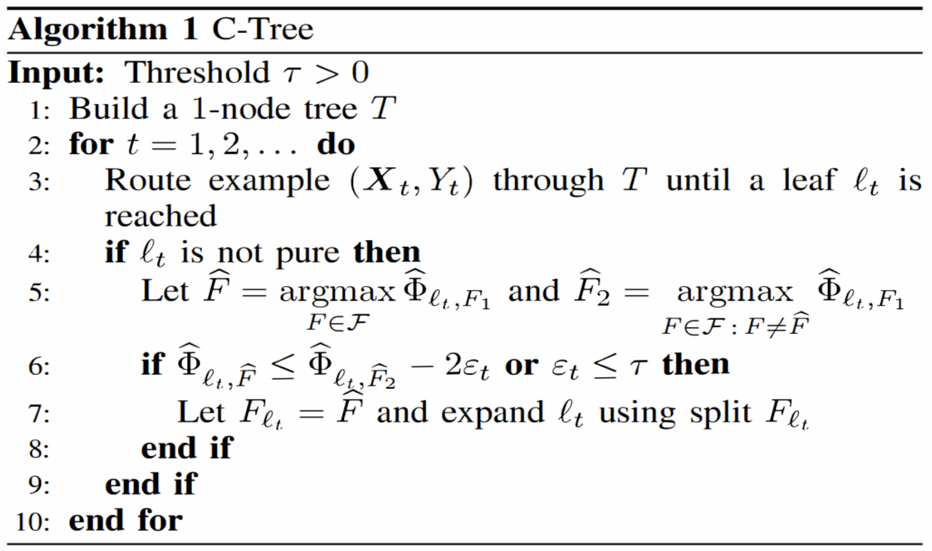
- The core condition of the algorithm, which if verified allows the subdivision of the data according to the first attribute with the appropriate threshold:



```
if(first_gain > second_gain2 + 2*gini_bound ||gini_bound <= tau )
```

Where first_gain and second_gain are the gain quantity relative to the first and second best attribute find by the bestSplit() function called two times; in the second call it is necessary not to consider the attribute and the related values identified by the first call.

- On the right, the pseudocode of the main phase of the C-Tree.



Algorithm 1 C-Tree

```
Input: Threshold $\tau > 0$
 1: Build a 1-node tree $T$
 2: for $t = 1, 2, \ldots$ do
 3:     Route example $(\boldsymbol{X}_t, Y_t)$ through $T$ until a leaf $\ell_t$ is reached
 4:     if $\ell_t$ is not pure then
 5:         Let $\widehat{F} = \operatorname{argmax}_{F \in \mathcal{F}} \widehat{\Phi}_{\ell_t, F_1}$ and $\widehat{F}_2 = \operatorname{argmax}_{F \in \mathcal{F} \,:\, F \neq \widehat{F}} \widehat{\Phi}_{\ell_t, F_1}$
 6:         if $\widehat{\Phi}_{\ell_t, \widehat{F}} \leq \widehat{\Phi}_{\ell_t, \widehat{F}_2} - 2\varepsilon_t$ or $\varepsilon_t \leq \tau$ then
 7:             Let $F_{\ell_t} = \widehat{F}$ and expand $\ell_t$ using split $F_{\ell_t}$
 8:         end if
 9:     end if
10: end for
```

Conclusions

- The main problems encountered are mainly related to the attempt of the reimplementation of the C-Tree, starting from the pseudo-code with the changes analyzed in the paper, in the R development environment. It was therefore decided to propose only some implementations of the logics that calculate the main measures.

- In the end the goal of this work was to provide a more rigorous statistical analysis of confidence intervals for splitting leaves in decision trees learning, estimating the gain related to each split. This study allows to define very efficient methods both in terms of compactness of the tree created and in terms of performance as we can see in the previous experiments. In particular the Hoefdding algorithms performed a lot better respect to the standard Decision Trees in terms of training time fundamental in the field of online stream.