Improving FIMTDD using adaptive filters

Mats Richter, Julius Autz

July 4, 2017

Abstract

The Fast Incremental Model Tree for Drift Detection (FIMTDD) by [?] is one of the more complex attempts on adapting the hoefding tree concept to online regression problems. However, the FIMTDD algorithm suffers from multiple conceptual problems in it's design as a result of it's complexity. This paper attempts to solve some of those problems, by replacing the weak learners in the leaf nodes of the model tree with adaptive filters. Furthermore we will try implement a novel splitting process to avoid overcomplex models when encountering large amounts of data.

1 Overview over the FIMTDD algorithm

In the first section of this paper we will present the basic outlines of the FIMTDD algorithm. We will shortly present the algorithm itself and discuss the core challanges in the design and how [?] resolved these. Then we will discuss the conceptual and practical problems of the algorithm.

1.1 What is FIMTDD

The FIMTDD-algorithm (Fast Incremental Model Tree for Drift Detection) was developed by [?]. FIMTDD and specially it's splitting process are based on the VFDT-Algorithm by [?] and were adapted for online regression problems. FIMTDD uses online learning to process large, possibly infinite amounts of data in real time. The predictions are conducted by gradient decent based weak learners in the leaf nodes. The tree structure also features a drift detection mechanism for replacing sub-tress with alternative alternative sub-trees.

1.2 Online learning

Online Learning is variation of the typical learning process of machine learning algorithms. The trainings process is cyclic and processes one instance, consisting ob data vector and the target label, at a time.
First, the machine learning algorithm will receive a single datapoint x and create a prediction based of the data point and the current model $\hat{y}$. After the prediction, the system will observe the real value of the label y. Now the system will update it's model based on the current Loss $L(y, \hat{y})$ before the cycle repeats itself.
Other than classical learning approaches based on fixed data sets, online learning

can conceptually handle data streams and therefore possible infinite amounts of data, since only a single training instance (x, y) has to be stored at any given time.
Online learning is specially usefull for time dependant data, like most data streams, since it always includes the most recent data in it's model by constantly adapting to the incoming data.

1.3 Concept Drift

A unexspected sudden or gradual change of the ground truth in a regression or classification problem is called a concept drift. Concept Drifts are usally the result influencing factors not included in the data vector and therefore unknown to the machine learning algorithm. Concept Drift often occurs in time dependant data and renders a problem to most machine learning algorithms, since old information has to be discarded and newer, more recent information must be included into the model in order to keep the distance between prediction and observed values small. Dealing with Concept Drift is growing research topic in the scientific comunity and many different attempts were made, like [?] shows in his survey on Concept Drift adaptive learning.

1.4 The Drift Detection

FIMTDD aims to handle concept drift by replacing a subtree of the model where the concept drift has occurred. First, in order to detect a concept drift, each node of FIMTDD keeps track of the cumulative loss that it's subtree is responsible for.
The Page-Hinkley test is used to evaluate whether or not this loss is indicative of a concept drift. For each timestep t, the mean loss $\bar{x}_t = 1/t \sum_{l=1}^{t} x_l$, and the sum of the distance to the mean loss until now (T) $m_t = \sum_{t=1}^{T}(x_t - \bar{x}_t - \alpha)$ is calculated, subtracting the decay rate α. Also, the lowest value of this sum is saved as M_t. If $m_t - M_t$ is smaller than a threshold λ a concept drift is detected.
This manner of drift detection requires two additional parameters, α and λ, which both affect the drift detection decisively.
Whenever a concept drift is detected, the node that detected the drift, an alternative tree begins to grow. The alternative tree starts as a single leaf node with no previous data, and is fed with the same data as the node from that moment on. The cumulative error of both the alternative tree and the original tree are consistently compared. Should the loss of the alternative tree consistently be lower than the loss of the orignal tree, it replaces the original tree and the old one is discarded.

1.5 The splitting process

Splitting a leaf node in the tree structure is always attempted after N_{min} instances reached this leaf node. The splitting process of FIMTDD is a 3 stage process. First, the best split for each axis is calculated, based on standard devitation reduction (SDR) as a quality measure. The exact process is based on exhaustive search through EBST-structures to calculate all necessary statistics in real time, the [?] for greater details.

Secondly the best overall split is chosen by picking the highest SDR split from the best axis aligned splits, calculated in the first step.
Then, in the third and final step, the splitting criterion is checked for the best overall split, only if the criterion is satisfied the split is conducted.
The criterion is based on the Hoeffding-Bound

$$\epsilon = \sqrt{\frac{R^2 \log(\delta)}{2N}}$$

with R being the range of the observed variable, $\delta \in [0, 1]$ adjustable parameter and N the number of observed values. In order to make the Hoeffding Bound criterion applicable for regression problems, not the quality or information gain of the split is used, since the range of the SDR is unknown. Instead, the ratio of the best split and the second best from the second stage of the splitting process is used:

$$r = SDR(h_2)/SDR(h_1)$$

with $SDR(h_1)$ beeing the SDR of the best split of the second stage of the splitting process and $SDR(h_2)$ the second best. Since

$$SDR(h_1) > SDR(h_2)$$

we can assume $r \in [0, 1]$ and therefore $R = 1$. The splitting criterion is satisfied, if $r < 1 - \epsilon$, since this means that we can say with confidence $1 - \delta$ that the best split is in fact the best overall split. However, explaing the Hoeffding Bound in all detail would be to much for this paper, for more information please see [**?**] and [**?**].

1.6 Conceptual Problems of the FIMTDD algorithm

2 Overlap of competence

2.1 Adaptive Filters

2.2 FIMTDD-LS

2.3 Results

3 Splitting bias

Both FIMTDD and FIMTDDLS have a strong bias to splitting their leaves into further subtrees. As the hoeffding bound is automatically satisfied after a certain amount of data points, whenever a best split is better than a second best split, the leaf will split, causing the tree to overgrow and overfit. An overgrown tree is unable to supply all his leaf nodes with enough datapoints so that they may accurately predict the data. Interestingly, this is one of the main triggers for the concept drift detection, as this leads exactly to the pattern of systematic error it looks for.

3.1 Hoeffdingbound and nMin

Consider again the formula of the Hoeffdingbound:

$$\epsilon = \sqrt{\frac{R^2 \log(\delta)}{2N}}$$

Given a static R and a static δ the formula actually only depends on N. As such, setting delta as its own parameter, additionally to N_{min}, seems redundant. Simply using either one should be sufficient, albeit requiring more care.

3.2 greedyFIMTDD-LS

greedyFIMTDD-LS is a further reduced version of the FIMTDD-LS. By discarding the Hoeffdingbound and thus δ, the number of parameters has further shrunk. Furthermore, by reworking the splitting process, it is more conservative, leading to smaller trees.
In greedyFIMTDD-LS, instead of growing the tree whenever the splitting criterion is satisfied, the original tree only grows if an alternative subtree is proven to be better than the leaf node.
We achieved this by repurposing the tree replacement system used in the concept drift adaptation. The original tree may never split its leaf nodes, however, all of its leaf nodes contain an alternative tree from the moment of their creation. Within these alternative trees, leaf nodes start growing alternative trees as soon as N_{min} data points have reached them. By using the same system as in the concept drift adaptation, the tree now only makes use of these alternative trees if they consistently improve compared to a single leaf node, guaranteeing a minimal tree.

3.3 Results

greedyFIMTDD-LS performs in many respects as expected. If the underlying ground truth can be approximated by a simple model (i.e. a singular sinus wave, or a linear approximation) it outperforms both FIMTDD and FIMTDD-LS. However, in case of a more complex model, is is unable to adapt to the complexity in a speedy manner, and is outperformed by both, the more complex, the greater the difference.

4 Conclusion & Future improvements