Concept Drift Handling of Imbalanced Data Streams: a Literature Review

R. Anderson, K. Chen and S. Jin

September 16, 2014

Abstract

We review the literature available around classification of imbalanced data in a streaming environment. First, we introduce the particular problems faced within streaming environments and with imbalanced classes. Through examining current research in the field, we explore the facets of classification in this environment: sampling techniques; approaches to detecting concept drift; base learners for building models to classify unseen data; and methods of utilising ensembles to improve classification accuracy. Finally, we examine evaluation of these methods, and the pertinent measures used.

1 Introduction

In recent year, as computing power and hardware technologies become more powerful and robust, its possible to store and process large volume of data. Such data sets which continuously and rapidly grow over time are referred to as data streams. Mining Data streams brings challenges to traditional data mining techniques, as most of the traditional data mining techniques cannot simply apply to data streams due to: 1. Traditional data mining techniques need process the data more than once or multiple pass. 2. Data streams temporal locality or concept drift, which is important because the class distribution changes as the data stream evolving over time in the context of classification. Furthermore, its more challenging to improve the overall accuracy of classification if the classes distribution are imbalanced, which is especially important for many applications in some fields, such as fraud detection and network intrusion detection.

This literature survey gives an overall picture on handling concept drift in imbalanced data streams from different angles such as sampling techniques, ensemble techniques, change detection algorithms and adaptive learning algorithms,and evaluation methodologies of adaptive learning algorithms. The survey is organized as follows. In Section 1, we introduce the problem of concept drift and imbalance problem in data stream. Section 2 presents several sampling methods. Section 3 presents several common drift detection algorithms.

Section 4 discusses tree learning algorithms. Section 5 discusses ensemble techniques. Section 6 discusses the evaluation methodologies of adaptive learning algorithms. Section 7 concludes the survey.

When data comes in the form of streams, it is impossible to fit the entire data into machines memory for processing, hence only online processing is feasible for mining data streams. Online processing refers to the predictive models are trained incrementally. However, this only solves the computational issue. In a dynamically changing environment, concept drift refers to the underlying class distribution of data stream can change over time. For example, online shopping customers interest can easily change over time. To solve the problem of concept drift, predictive models needs to be updated online, which is also called online adaptive learning. There are two categories for handling concept drift: 1. update the model at regular intervals without considering whether there is a change occurred; 2. detect a concept change before updating the model. In this survey, we mainly discuss the second approach. Well known change detection algorithm are: Statistical Process Control (SPC), ADaptative WINDdowing (ADWIN) , Fixed Cumulative Windows Model (FCWM) and Page Hinkley Test (PHT), flora,winnowdetailed summarization of these algorithms are in in section 4.

Imbalanced data set has a skewed class distribution, usually one majority class distribution, one or more minority class distribution. The problem has been well researched since 2000, but more recently researches start applying some approaches to data streams. Three different approaches are: 1. data level approach, also called sampling technique, which simply modifies the data set in order to rebalance the class distribution; 2. algorithm level approach, which adapted to deal with minority class to improve performance; 3. cost sensitive approach has a cost matrix to describe the cost of misclassifying a class instance. Among them, data level approach is mostly used one in the context of data streams. Sample Ensemble Framework is a representative one in this approach. Most of the data level techniques for handling concept drift combined with ensemble techniques to improve accuracy. Data level approach has advantages of not changing the algorithm. However, the data level approach suffers from the problem of resource and computation. On the other hand, algorithm level approach doesnt need sampling as a pre-processing step, thus doesnt have the problem of resource and computation.

2 Sampling methods

In this paper, the sampling techniques specifically focus on the imbalanced data set sampling, but not the general sampling techniques in data stream, such as Reservoir Sampling and Sliding Window Sampling. Balancing training data is the most straight-forward approach for imbalanced concept-drifting data stream mining. Many sampling algorithms improved accuracy by under-sampling the majority class or over-sampling the minority class. Synthetic Minority Over-sampling Technique (SMOTE), the minority class is over-sampled by taking

each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors, rather than by over-sampling with replacement. SMOTE forces decision region of minority class to become more general. SMOTE gives better classification accuracy in terms of ROC. Neighborhood Cleaning Rule (NCL) uses the Wilsons Edited Nearest Neighbor Rule (ENN) to remove majority class instances. Sampling techniques is used at data level, which may keep the algorithm unchanged, however it could bring computational and memory issues when applying to data stream mining.

3 Concept drift detection

4 Base learners

Supervised learning requires models to be trained from labelled data. When model-building, we need an approach that can use explanatory variables and classes in a training set of data to create a set of rules for classifying unseen instances. This approach is our base learner: our tool that will build our classifier to apply to unseen instances of our data [**?**].

In a streaming environment, however, we can only make one pass through our data. Concept drift requires a learner that can be run at regular intervals, so as to understand the current distribution that our instances fall into. This requires a trade-off between time, accuracy and memory. A successful learner will be sufficiently accurate while running quickly and with manageable memory overhead. These requirements are much more important than in a static environment, where we have a finite amount of data and time to process it.

Decision trees have particular features which make them a good fit for classifying streams with concept drift. The C4.5 decision tree, and its predecessor, the ID3 tree, are very popular within the research community [**?**]. Their appeal lies in their simplicity at every node, a binary decision is made related to one of the explanatory variables, resulting in a divide-and-conquer approach which results in fast classification and low overhead for the model. This leads to some clear drawbacks only rectangular regions can be used to discriminate between classes in the feature space for instance. Splits are made through information gain measures, seeking to maximise the change in impurity of the nodes with each split. These trees will naturally overfit, if fully grown, but can be pruned to achieve a desired level of fit to the data. Their adaptability and speed make them a natural fit in classification of streams.

Domingos and Hulten [**?**] highlight a major concern with C4.5, ID3 and CART they assume that all training examples can be held in memory, and are limited in the examples they can consider. Other trees such as SPRINT and SLIQ have attempted to resolve this issue by using a window to scan large datasets sequentially, but this is very slow when building complex trees, and still requires all data to reside in disk-space. They propose the Very Fast Decision Tree (VFDT), which requires that data items are only read once, in a small

and constant time. They utilise Hoeffding Bounds [**?**] to guarantee that the statistically best attribute to divide a node on can very commonly be found using a small number of examples of instances as could be with infinite. They propose the Hoeffding tree, which regularly checks whether splitting a node on the best attribute will lead to an improvement in a selected measure (information gain or Gini) beyond a set threshold. This allows it to continue to adapt, even with infinite data, without sacrificing significant quality nor having burdensome memory requirements. Parameters need to be set by the user: the minimum number of examples before reviewing whether to split a node; the threshold to allow a split; and whether the algorithm can rescan prior examples. Even if these parameters are well selected by the user, there is a danger that they will lead to poor performance in environments featuring significant concept drift.

Recognising the issues posed by concept drift, Hulten et al. [**?**] proposed the Concept-Adapting Very Fast Decision Tree (CVFDT), an adaptation on the VFDT to give it the adaptability to handle changing data. Through adopting a sliding window approach, the CVFDT increments counts of recently seen data while reducing counts of older data. This should have no effect when there is no concept drift, but where there is drift, prior splits will no longer pass the Hoeffding tests that deemed them worthwhile. At this point, an alternate subtree is grown by the algorithm from that node. If it accurately classifies new data better than the existing subtree, the existing subtree is replaced. Overall, this requires additional memory to keep alternate subtrees in memory, and summary statistics of split quality at each node. However, these memory requirements are still constant with the data items received. CVFDTs cannot utilise old sub-trees which are discarded, which could provide potential performance improvements where concepts drifts can revert back to prior probability distributions.

Decision trees have trouble accurately classifying rare classes. Consider a binary class problem: if a very large majority belongs to one class, often a classifier achieves best performance by classifying all instances as that class. Single classifier often needs to combine in a ensemble to achieve good performance for imbalanced data set, however recent researches has made it possible for some classifiers perform as good as ensemble. Lyon et al [**?**] propose an improvement on the VFDT by introducing a new criterion for splitting: Hellinger distance. This measure seeks the attributes which are most disjoint in the data, and seeks to create tree splits based on these features. Instead of prioritising information gain, this tree prioritises class discrimination. This will often lead to a smaller portion correctly classified, but should amplify the number of the rare class successfully classified. Calculating Hellinger distance can be costly, as it scales to the number of classes and needs to discretize continuous classes into bins, leading to further multiplicative scaling. Lyon et al propose the HD-VFDT which uses the splitting criterion above, and also the Gaussian Hellinger Very Fast Decision Tree, which assumes normality of the distribution to simplify calculation of the Hellinger Distance. However, instances in a data stream are not independent, as time affects their distribution, and so this could lead to flawed results in some circumstances. Experimental analysis of this approach showed statistically significant improvements using these methods over simple Hoeffd-

ing Trees where the minority class makes up 1% or less of the total data. They do not evaluate the approach thoroughly with regard to time nor memory use. Specifically, it would be interesting to measure the difference in accuracy, speed and time required between the two variants of the Hellinger tree proposed.

Liechtenwalter and Chawla [**?**] have found potential in using Hellinger trees in environments with concept drift. By weighting the Hellinger distance with Information Gain to measure distance between datasets, a learner can decide whether model learnt on previous data is valid on current data. Pazzolo et al [**?**] showed, using the HDDT proposed by Cieslak and Chawla [**?**] and C4.5 tree, that trees with this weighting generally outperformed trees without the weighting in class-imbalanced environments with drift. These papers do not address the multi-class problem, which would be a natural extension of this research.

5 Ensemble learners

6 Evaluating approaches

7 Conclusion