Cache Hierarchy Inspired Compression: a Novel Architecture for Data Streams

Geoffrey Holmes, Bernhard Pfahringer and Richard Kirkby
Computer Science Department
University of Waikato
Private Bag 3105, Hamilton, New Zealand
goff, bernhard, rkirkby}@cs.waikato.ac.nz.

Abstract - We present an architecture for data streams based on structures typically found in web cache hierarchies. The main idea is to build a meta level analyser from a number of levels constructed over time from a data stream. We present the general architecture for such a system and an application to classification. This architecture is an instance of the general wrapper idea allowing us to reuse standard batch learning algorithms in an inherently incremental learning environment. By artificially generating data sources we demonstrate that a hierarchy containing a mixture of models is able to adapt over time to the source of the data. In these experiments the hierarchies use an elementary performance based replacement policy and unweighted voting for making classification decisions.

Keywords: Data streams, classification, cache hierarchy.

1 Introduction

Conventional data mining algorithms operate in an environment where a model (for example, a set of rules) is induced from a training set of data instances. This training set is available in its entirety from the outset. The learning algorithms that produce a model from this data typically load all of the data into main memory and then access each instance one at a time. This methodology is often referred to as batch learning. A consequence of this approach is that very few incremental algorithms have been developed.

The data stream model treats the training data as a (possibly infinite) stream. In this context there is no possibility of fitting all the data into memory. The model inevitably places constraints on what can be achieved and scaling an algorithm to conform to a data stream is a significant challenge. The requirements are: the need to process instances one at a time with only a glance at the instance (instances cannot be stored); only use available memory (we cannot swap to disk as this would take too long); process an instance in a limited amount of time (we cannot afford to fall behind the rate at which instances arrive), and finally, be ready to make predictions at any time.

In such a context it is extremely difficult to fully process a single instance at a time. One approach to staying in touch with the stream is to accumulate statistics and then make decisions only when those statistics reach critical points, as in Hoeffding trees [1]. An alternative approach is to slowly build up a knowledge of the stream by constructing models from accumulations of instances [2]. Both strategies can be accommodated in the architecture described here.

In the field of Data Communications, web proxy caches are used to enhance the scalability and performance of the web by reducing bandwidth demand and increasing the response time for popular documents. The proxies are often organised into hierarchies as in the squid system [3]. In squid the hierarchy is organised in a parent/child and sibling arrangement. When a cache requests an object from its parent, and the parent does not have the object in its cache, the parent fetches the object, caches it, and delivers it to the child. In addition to the parent-child relationships there are siblings which are caches at the same level in the hierarchy, provided to distribute cache server load.

In a streaming context it would be advantageous to distribute algorithmic load (classification, regression, clustering or association rule learning) through some form of sibling structure. The notion of parent/child caches is harder to reconcile as the objectives of web caching and data mining are somewhat different. A parent could be used to contain older and wiser models, those that have successfully survived some form of culling process over time.

Alternatively, parents could be viewed as the agents in a system that pass on the best models they find for use by their children.

In data streaming and web caching, training and testing have to occur simultaneously. The web cache hierarchy fills up by fetching objects not present in the cache until it is full. Once full, an object replacement policy is needed to keep the hierarchy up to date with the needs of its user.