Fundamentals of Analyzing and Mining Data Streams

Graham Cormode

AT&T Labs–Research, 180 Park Avenue, Florham Park, NJ 07932, USA

Abstract. Many scenarios, such as network analysis, utility monitoring, and financial applications, generate massive streams of data. These streams consist of millions or billions of simple updates every hour, and must be processed to extract the information described in tiny pieces. This survey provides an introduction the problems of *data stream monitoring*, and some of the techniques that have been developed over recent years to help mine the data while avoiding drowning in these massive flows of information. In particular, this tutorial introduces the fundamental techniques used to create compact summaries of data streams: sampling, sketching, and other synopsis techniques. It describes how to extract features such as clusters and association rules. Lastly, we see methods to detect when and how the process generating the stream is evolving, indicating some important change has occurred.
Keywords: data streams, sampling, sketches, association rules, clustering, change detection.

1 Introduction

In recent years there has been growing interest in the study and analysis of *data streams*: flows of data that are so large that it is usually impractical to store them completely data. Instead, they must be analyzed as they are produced, and high quality results guaranteed, no matter what outcomes are observed as the stream progresses. This tutorial surveys some of the key ideas and techniques that have been developed to analyze and mine such massive data streams. See [13] for a longer survey from an algorithmic perspective.

Motivation for studying data streams comes from a variety of areas: scientific data generation, from satellite observation to experiments on subatomic particles can generate terabytes of data in short amounts of time; sensor networks may have many hundreds or even thousands of nodes, each taking readings at a high rate; and communications networks generate huge quantities of *meta-data* about the traffic passing across them. In all cases, this information must be processed and analyzed for a variety of reasons: to monitor a system, analyze an experiment, or to ensure that a service is running correctly. However, given the massive size of the input, it is typically not feasible to store it all for convenient access. Instead, we must operate with resources much smaller than the size of the input ("sublinear"), and still guarantee a good quality answer for particular computations over the data.

From these disparate settings we can abstract a general framework within which to study them: the streaming model. In fact, there are several variations of this model, depending on what form the input may take and how an algorithm must respond.

Models: Arrivals only, or Arrivals and Departures. The basic model of data streams is an arrivals-only one. Here, the stream consists of a quantity of tuples, or items, which describe the input. Typically each tuple is a simple, small object, which might indicate, for example, the identity of a particular object of interest, and a weight or value associated with this arrival. In a network, the observation of a packet could be interpreted as a tuple indicating the intended destination of the packet, and the size of the packet payload in bytes. For another application, the same packet could be interpreted as a tuple whose identity is the concatenation of the source and destination of the packet, with a weight of 1, indicating that it is a single packet. Typically, we can interpret these streams as defining massive implicit vectors, indexed by item names, and whose entries are (usually) the sum of the associated counts (although many other interpretations may be possible). A richer model allows departures: in additional to positive updates to entries in this implicit vector, they may be negative. This captures more general situations in which earlier updates might be revoked, or observations for which negative values are feasible. In either case, the assumption is that each tuple in the input stream must be processed as it is seen, and cannot be revisited later unless it is stored explicitly by the stream algorithm within its limited internal memory.

Randomization and Approximation. Within these models, many natural and fundamental questions can be shown to require space linear in the input to answer exactly. For example, to test whether two separate