Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Bayesian topic model approaches to online and time-dependent clustering

M. Kharratzadeh, B. Renard, M.J. Coates [*]

Department of Electrical and Computer Engineering, McGill University, 3480 University St, Montreal, Quebec, H3A 0E9, Canada

ARTICLE INFO

Article history:
Available online 19 March 2015

Keywords:
Online clustering
Probabilistic topic models
Dirichlet process mixture models
Streaming data
Sequential Monte Carlo sampling

ABSTRACT

Clustering algorithms strive to organize data into meaningful groups in an unsupervised fashion. For some datasets, these algorithms can provide important insights into the structure of the data and the relationships between the constituent items. Clustering analysis is applied in numerous fields, e.g., biology, economics, and computer vision. If the structure of the data changes over time, we need models and algorithms that can capture the time-varying characteristics and permit evolution of the clustering. Additional complications arise when we do not have the entire dataset but instead receive elements one-by-one. In the case of data streams, we would like to process the data online, sequentially maintaining an up-to-date clustering. In this paper, we focus on Bayesian topic models; although these were originally derived for processing collections of documents, they can be adapted to many kinds of data. The main purpose of the paper is to provide a tutorial description and survey of dynamic topic models that are suitable for online clustering algorithms, but we illustrate the modeling approach by introducing a novel algorithm that addresses the challenges of time-dependent clustering of streaming data.

1. Introduction

Clustering is an unsupervised learning technique, with the goal of finding a structure or pattern in a collection of unlabeled samples. It strives to identify groups, or clusters, of similar objects. The clusters may be distinct, in the sense that each object belongs to a single cluster, or they may overlap. In an unfortunate terminology overload in the literature, the word "clustering" is used to describe both the act of identifying the clusters (the algorithm) and the set of clusters identified by the algorithm. Most clustering algorithms employ some notion of distance between objects, and also have an explicit or implicit criterion defining what constitutes a "good" clustering. The algorithms then optimize (often heuristically) this criterion to determine the clustering.

In this paper, we focus on the task of online clustering, which involves clustering a series of data items that arrive sequentially. The goal is to provide a new clustering after the arrival of each data item; generally we also want to ensure that the identified clusterings evolve smoothly over time. We require that the clustering algorithm is capable of learning the number of clusters automatically, and that the use of computational and memory resources remains bounded over time. The algorithm must be capable of taking into account the order of the data arrivals; preferably the clustering should be dependent on the actual generation or arrival times associated with each data item.

In mathematical form, the input is a sequence of data items $\{x_1, x_2, \ldots\}$, which can either be a data stream or a sequence of limited size, as long as items are received one-by-one. Each data item x_i is associated with a timestamp t_i, whose value represents the time when the item was generated or received. In most cases, we will assume that each data item x_i is a set of a discrete elements that are members of a predefined "vocabulary", $\mathcal{V}$, and we assume that each element of x_i is essential to the meaning of the item. Our goal is to infer a clustering label z_i for each data item; we need to provide the label for x_i before data item x_{i+1} arrives.

We concentrate on clustering algorithms built upon probabilistic topic models. Although these have some limitations in terms of the types of data they can represent efficiently, they have advantages over many other clustering approaches. In particular, they specify a generative probabilistic model for the clustering, which permits application of principled inference procedures, including Bayesian methods. The primary purpose of this paper is to provide a tutorial description and survey of dynamic topic models that are suitable for online clustering algorithms, but we illustrate the modeling approach by introducing a novel algorithm that employs sequential Monte Carlo sampling to address the challenges of time-dependent clustering of streaming data.

[*] Corresponding author.
E-mail addresses: milad.kharratzadeh@mail.mcgill.ca (M. Kharratzadeh), benjamin.renard@melix.net (B. Renard), mark.coates@mcgill.ca (M.J. Coates).

http://dx.doi.org/10.1016/j.dsp.2015.03.010
1051-2004/© 2015 Elsevier Inc. All rights reserved.

1.1. Probabilistic topic models

Probabilistic topic models were developed for the analysis of large collections of documents, with the goal of identifying common themes and topics. Excellent introductions are provided in [1,2]. One of the earliest topic models was *latent Dirichlet allocation* (LDA) [3]. The key idea in LDA is that each document addresses multiple topics, and the words comprising the document can thus be considered as samples from common words employed when discussing these topics. Mathematically, each topic is defined as a distribution over a fixed vocabulary. In the probabilistic LDA model, to generate each document, a distribution over the topics is drawn from a Dirichlet distribution. To generate each of the words that comprise the document, we first draw a topic from the topic distribution, and then draw a word from the topic. The LDA generative model assumes a fixed number of topics and fixed word probabilities within each topic.

There have been many extensions of the topic model employed in LDA. Of most interest to us are (i) the extension to Dirichlet process mixture models, which allow the number of topics to be learned from the data rather than requiring specification in the prior; and (ii) the incorporation of time-dependency in the models. In Section 2 of the paper we provide an introduction to Dirichlet distributions and processes, and Dirichlet process mixture models. In Section 3 we review some of the techniques that have been proposed for injecting temporal dependency into probabilistic topic models.

1.2. Dynamic/static and online/offline distinctions

Dynamic (as opposed to static) clustering incorporates the notion of time in the dataset. Data items can either have a timestamp associated with their arrival in the dataset (e.g., a data stream), or they can evolve dynamically (e.g., geographic position of mobile users over time). A dynamic clustering algorithm then identifies clusterings that change over time.

A dynamic clustering algorithm can be either online or offline. Online clustering means that the algorithm must provide a clustering for the data associated with timestamp t before seeing any data with timestamp $t' > t$ [4]. There are two main uses for online algorithms. The first case corresponds to data streams: we receive data items sequentially and we cannot afford to wait until we have all the items to perform processing. The second arises when we have access to the entire dataset, but the dataset is too big to be processed by offline methods, motivating sequential processing of elements or batches of elements. In offline clustering, the algorithm takes as an input the entire data stream or the complete history of the dataset.

When considering datasets where each data item is associated with a timestamp, we can ask ourselves whether the temporal distance between two consecutive data items (i.e., the difference between their timestamps) is of any importance for the analysis of the dataset. If not, then we can replace the timestamp by the index of the item in the ordered dataset. This setting is useful when the time difference between two consecutive items is always the same (for example when considering articles published in a yearly journal). An algorithm that considers only the order of the data items, rather than the actual times, is called *order-dependent*. If the algorithm explicitly takes into account the time difference between data item arrivals, we say that it is *time-dependent*.

Let us consider the example of clustering marathon runners by their performance in a race. An order-dependent algorithm would only consider the order in which the runners finished. A time-dependent algorithm, however, would process the actual completion times. If we wanted to identify the top-10 finishers, the order-dependent algorithm would suffice; if our goal were to identify a group of racers who all finished within 2 minutes of each other, a time-dependent algorithm is required.

1.3. Inference

Although topic models are well-matched to many data sets, and the prior is constructed so that there is conjugacy with the commonly-assumed likelihood function for the data, exact inference is in general infeasible. We must therefore turn our attention to approximate Bayesian inference approaches; the main candidates are Markov chain Monte Carlo (MCMC) [5], variational inference [6] and Sequential Monte Carlo (SMC) samplers [7].

One of the main challenges of a data stream setting is to keep the computational resources bounded as the number of processed items increases. For online clustering, we generally do not know the number of data items we will need to process ahead of time. Section 4 reviews methods that can be used to perform online posterior inference for the dynamic topic models. We focus on sequential Monte Carlo samplers, because they are naturally suited to online processing. We also highlight some of the recent work in streaming variational Bayes [8], which adapts variational approximation methods to make them more amenable to the online dynamic clustering task.

In Section 5 we present a sequential Monte Carlo (SMC) sampling approach for performing inference for a time-dependent dynamic topic model. We build upon the work in [9,10], where Ülker et al. developed an SMC sampling approach for *static* Dirichlet process mixture models. In the static models, the probability of assignment to a cluster does not depend on the time of arrival or generation of a data item. Our focus is on the streaming, online setting, where the importance of a data item is very much associated with when it arrived, so we incorporate explicit dependence on time, and we consider how to ensure that the memory requirements of the algorithm remain bounded. We introduce a novel sampling procedure that focuses on elements whose clustering labels are more uncertain; our numerical experiments and analysis of news data indicate that this procedure allows us to either improve inference performance or reduce computational overhead.

1.4. Example application

Privacy concerns have always existed for popular social networks such as Facebook, Twitter or Google+, due to their reliance on targeted advertising. These concerns led to the creation of several privacy-focused social networks such as Diaspora [11] and Friendica [12]. These alternative social networks are peer-to-peer networks of servers that distribute data throughout the network in an attempt to maintain a high level of privacy. Each user can remain in control of his/her data by selecting which server stores it. Although this distributed architecture protects the users' privacy, it generates several problems that centralized social networks do not face. Control of the network is more limited and performance problems can arise (primarily slow response time), because nodes of the network can be self-hosted web servers with limited computational resources. Search is more challenging, because each node has access only to a limited portion of the network.

Often users want to search for other users that share a similar interest, usually by providing a set of keywords related to that interest. Centralized networks have direct access to all users' data and hence can directly determine the users whose interests match the query. On the other hand, nodes in a distributed network have only access to the data corresponding to their own users and they only know a subset of entire network, composed of their neighbors in the peer-to-peer graph. To find users on other nodes, they need to forward the query in an efficient way.