Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 18 (2005) 790–798
Neural Networks
www.elsevier.com/locate/neunet

2005 Special Issue

# Efficient streaming text clustering☆

Shi Zhong*

*Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA*

**Abstract**

Clustering data streams has been a new research topic, recently emerged from many real data mining applications, and has attracted a lot of research attention. However, there is little work on clustering high-dimensional streaming text data. This paper combines an efficient online spherical k-means (OSKM) algorithm with an existing scalable clustering strategy to achieve fast and adaptive clustering of text streams. The OSKM algorithm modifies the spherical k-means (SPKM) algorithm, using online update (for cluster centroids) based on the well-known Winner-Take-All competitive learning. It has been shown to be as efficient as SPKM, but much superior in clustering quality. The scalable clustering strategy was previously developed to deal with very large databases that cannot fit into a limited memory and that are too expensive to read/scan multiple times. Using the strategy, one keeps only sufficient statistics for history data to retain (part of) the contribution of history data and to accommodate the limited memory. To make the proposed clustering algorithm adaptive to data streams, we introduce a forgetting factor that applies exponential decay to the importance of history data. The older a set of text documents, the less weight they carry. Our experimental results demonstrate the efficiency of the proposed algorithm and reveal an intuitive and an interesting fact for clustering text streams—one needs to forget to be adaptive.


*keywords:* Text clustering; Streaming clustering; Spherical k-means; Competitive learning

## 1. Introduction

In many real data mining applications, data comes in as a continuous stream and presents several challenges to traditional static data mining algorithms. Application examples include topic detection from a news stream, intrusion detection from continuous network traffic, object recognition from video sequences, etc. Challenges lie in several aspects: high algorithm efficiency is required in real time; huge data volume that cannot be kept in memory all at once; multiple scans from secondary storage is not desirable since it causes intolerable delays; and mining algorithms need to be adaptive since data patterns change over time.

Many recent research works tried to address these challenges, yet with different focuses. In general, they can be divided into two categories—*scalable* methods and *adaptive* methods—based on the focus. Works in the first category (Bradley, Fayyad & Reina, 1998; Farnstrom, Lewis & Elkan, 2000; Guha, Meyerson, Mishra, Motwani & O'Callaghan, 2003; O'Callaghan, Mishra, Meyerson, Guha & Motwani, 2002) aim to capture overall data characteristics over a long time period despite the constraint that at any time only a small subset of data can reside in the memory for a limited time frame for once. Adaptive methods (Agrawal, Han, Wang & Yu, 2003; 2004; Hulten, Spencer & Domingos, 2001; Kifer, Ben-David & Gehrke, 2004), on the other hand, focus on following changes in the data characteristics over time. Therefore, the quality of such mining algorithms is usually measured locally at a set of stream points.

Like most of the aforementioned works, we focus on clustering techniques in the paper. More specifically, we investigate a streaming text clustering method that emphasizes on adaptivity.

Text clustering has become an increasingly important technique for unsupervised document organization, automatic topic extraction, and fast information retrieval or filtering. Not surprisingly, there has been a vast literature on text clustering. However, there has been little work on clustering streaming text data such as news streams. Banerjee and Ghosh (2004) extended a frequency-sensitive spherical k-means algorithm to cluster text streams. Their

☆ An abbreviated version of some portions of this article appeared in Zhong (2005), published under the IEEE copyright.

* Tel.: +1 561 297 3168; fax: +1 561 297 2800.
*E-mail address:* zhong@cse.fau.edu.

0893-6080/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2005.06.008

work focused on extracting a set of clusters (of balanced size) over a long period. That is, their method effectively belongs to the *scalable* category and they measured clustering performance using all documents in a stream. Furthermore, their experimental setting on streaming clustering is flawed in that they repeated a set of documents multiple times to create a stream, thus effectively reading each text document into memory multiple times.

In this paper, we extend our previous work on efficient online spherical k-means (Zhong, 2005) to cluster text streams. The extension is essentially a combination of the online spherical k-means (OSKM) algorithm with scalable clustering techniques (Farnstrom et al., 2000). This combination leverages on the efficiency and effectiveness of OSKM, which has been demonstrated in Zhong (2005), as well as the ability of scalable methods to process huge data streams. But there is an improvement on the *scalable* part. Similar to the methods proposed in Agrawal et al. (2003), we focus on the adaptivity of our streaming algorithm by introducing to the scalable approach a decay factor that exponentially reduce the contribution of history data. The decay factor essentially impose a forgetting mechanism on the clustering process, which is commonly used in adaptive neural networks (Widrow & Lehr, 1990).

In this paper, we design experiments to investigate the effect of the forgetting mechanism, and results show that one needs to forget to adapt. That is, clustering quality at a series of stream points is high when the decay parameter is small (or history is quickly forgotten). This indicates that the original scalable approach (Farnstrom et al., 2000) cannot be applied directly to streaming data clustering without any change.

The rest of this paper is organized as follows. We review the OSKM algorithm (Zhong, 2005) in the next section and scalable clustering techniques (Bradley et al., 1998; Farnstrom et al., 2000) in Section 3. The proposed streaming OSKM algorithm is presented in Section 4 and experimental results are shown in Section 5. Finally, Section 6 concludes this paper.

For notation in this paper, we use bold face variables (e.g. $\mathbf{x}$, $\mu$) to represent vectors, upper case calligraphic alphabets (e.g. $\mathcal{X}$, $\mathcal{Y}$) to represent sets, $\|\cdot\|$ to denote the $L_2$ norm, and $|\cdot|$ to denote the number of elements in a set.

## 2. Online spherical k-means

### 2.1. Spherical k-means (SPKM)

For high-dimensional data such as text documents (represented as Term Frequency-Inverse Document Frequency vectors), cosine similarity has been shown to be a superior measure to Euclidean distance (Strehl, Ghosh & Mooney, 2000). The implication is that the direction of a document vector is more important than the magnitude. This leads to a unit-vector representation, i.e. each document vector is normalized to be of unit length ($\|x\|=1$). Let $\{\mu_1,\ldots,\mu_K\}$ be a set of unit-length centroid vectors, the spherical k-means algorithm (Dhillon & Modha, 2001) aims to maximize the average cosine similarity objective

$$L = \sum_x \mathbf{x}^T \mu_k(\mathbf{x}), \tag{1}$$

where $k(\mathbf{x})=\arg\max_k x^T\mu_k$. The batch version of SPKM again iterates between a data assignment step and a centroid estimation step (Zhong & Ghosh, 2003). It can also be derived from mixture of von Mises–Fisher distributions (Banerjee & Ghosh, 2004). The main difference from standard k-means (MacQueen, 1967) is that the re-estimated mean vectors $\{\mu\}$ need to be normalized to unit-length.

### 2.2. Online spherical k-means (OSKM)

The 'Winner-Take-All' mechanism is one of the simplest competitive learning strategies (Ahalt, Krishnamurthy, Chen & Melton, 1990) and allows only one winner per input. For each input, only the winning neuron learns (i.e. gets updated). When applied to clustering (with cluster centroids as cells), WTA leads to online k-means clustering. Soft competitive learning methods (Kohonen, 1990; Martinetz, Berkovich & Schulten, 1993) have been employed to adjust multiple neurons per input, with the rate of adjustment inversely proportional to the distance (or some function of the distance) between each neuron and the given input.

We use WTA learning in online spherical k-means clustering to maximize the objective (1). Given a data point $\mathbf{x}$, we incrementally update the closest cluster center $\mu_{k(\mathbf{x})}$ as:

$$\mu_{k(\mathbf{x})}^{(\text{new})} = \frac{\mu_{k(\mathbf{x})} + \eta \frac{\partial L_{\mathbf{x}}}{\partial \mu_{k(\mathbf{x})}}}{Z} = \frac{\mu_{k(\mathbf{x})} + \eta \mathbf{x}}{Z}, \tag{2}$$

where $\eta$ is a learning rate, $L_{\mathbf{x}}=\mathbf{x}^T\mu_k(\mathbf{x})$, and $Z$ is a normalization term to keep the updated $\mu$ on the unit hypersphere. The derivative term $\partial L_{\mathbf{x}}/\partial\mu_{k(\mathbf{x})}$ is basically the gradient of the $x$-related derivative term part of the objective function with respect to $\mu_{k(\mathbf{x})}$. Thus this is essentially an incremental gradient ascent algorithm.

**Algorithm:** online spherical k-means (OSKM)
**Input:** A set of $N$ *unit-length* data vectors $\mathcal{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ in $\mathbb{R}^d$, number of clusters $K$, and number of batch iterations $M$.
**Output:** A partition of the data vectors given by the cluster identity vector $\mathcal{Y}=\{y_1,\ldots y_N\}$, $y_n\in\{1,\ldots,K\}$ .
**Steps:**

1. Initialization: initialize the *unit-length* cluster centroid vectors $\{\mu_1,\ldots,\mu_K\}$, $i=0$ ;
2. For $m$ = 1 to $M$
   For $n$ = 1 to $N$
   (a) For each data vector $\mathbf{x}_n$, find the closest centroid $y_n=\arg\max_k \mathbf{x}_n^T\mu_k$ ;
   (b) Estimate each centroid as $\mu_{y_n}\leftarrow\frac{\mu_{y_n}+\eta^{(i)}\mathbf{x}_n}{\|\mu_{y_n}+\eta^{(i)}\mathbf{x}_n\|}$ ;
   (c) $i\leftarrow i+1$;

Fig. 1. Online spherical k-means algorithm.