The Journal of Systems and Software 106 (2015) 1–8

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Learning to detect representative data for large scale instance selection

Wei-Chao Lin [a], Chih-Fong Tsai [b,*], Shih-Wen Ke [c], Chia-Wen Hung [a], William Eberle [d]

[a] Department of Computer Science and Information Engineering, Hwa Hsia University of Technology, Taiwan
[b] Department of Information Management, National Central University, Taiwan
[c] Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan
[d] Department of Computer Science, Tennessee Technological University, USA

ARTICLE INFO

Article history:
Received 27 September 2014
Revised 19 February 2015
Accepted 9 April 2015
Available online 14 April 2015

Keywords:
Instance selection
Data reduction
Data mining

ABSTRACT

Instance selection is an important data pre-processing step in the knowledge discovery process. However, the dataset sizes of various domain problems are usually very large, and some are even non-stationary, composed of both old data and a large amount of new data samples. Current algorithms for solving this type of scalability problem have certain limitations, meaning they require a very high computational cost over very large scale datasets during instance selection. To this end, we introduce the ReDD (**Re**presentative **D**ata **D**etection) approach, which is based on outlier pattern analysis and prediction. First, a machine learning model, or detector, is used to learn the patterns of (un)representative data selected by a specific instance selection method from a small amount of training data. Then, the detector can be used to detect the rest of the large amount of training data, or newly added data. We empirically evaluate ReDD over 50 domain datasets to examine the effectiveness of the learned detector, using four very large scale datasets for validation. The experimental results show that ReDD not only reduces the computational cost nearly two or three times by three baselines, but also maintains the final classification accuracy.

1. Introduction

1.1. Background

The large size of today's data collections often makes them very difficult for the current data mining algorithms to handle properly. As a consequence, data pre-processing has become one of the most important steps in KDD (knowledge discovery in databases) for good quality data mining. In other words, if the chosen dataset contains too many instances (i.e., data samples), it can result in large memory requirements, slow execution speed, and over-sensitivity to noise. Another problem with using the original data points is that there may not be any located at the precise points that would make for the most accurate and concise concept description (Pyle, 1999).

Data pre-processing is often implemented using instance selection, or data reduction. The aim of instance selection is to reduce the dataset size by filtering out data from a given dataset that are noisy, redundant or both, and so likely to degrade the mining performance (Wilson and Martinez, 2000; Li and Jacob, 2008). More specifically, instance selection is used to shrink the amount of data, after which data mining algorithms can be applied to the reduced dataset. Sufficient results are achieved if the selection strategy is appropriate (Reinartz, 2002).

This task is similar to outlier detection (Hodge and Austin, 2004) or anomaly detection (Chandola et al., 2009) where the aim is to discover observations that lie an abnormal distance from other values in a population. Simply, outliers are the unusual observations (or bad data points) that are far removed from the mass of data. In other words, they are further away from the sample mean than what is deemed reasonable. Consequently, outliers could lead to significant performance degradation (Aggarwal and Yu, 2001; Barnett and Lewis, 1994).

Filtering out the detected outliers is very useful for discovering the normative patterns in the data (Knorr et al., 2000). Therefore, from the data mining perspective, the aim of instance selection can be thought of as the same as outlier detection (Liu and Motoda, 2001). In other words, performing instance selection and outlier detection over a given dataset can reduce the size of datasets and ensure that they contain higher proportions of representative data.

1.2. Motivation

Defining whether outliers are lying an abnormal distance from other samples or not is a subjective process and defining what constitutes an outlier or determining whether or not an observation is an outlier is a difficult problem. Many instance selection and outlier

* Corresponding author. Tel.: +886 3 422 7151; fax: +886 3 4254604.
E-mail address: cftsai@mgt.ncu.edu.tw (C.-F. Tsai).

http://dx.doi.org/10.1016/j.jss.2015.04.038
0164-1212/

detection methods have been proposed to detect and remove unrepresentative data from a given dataset, and they have shown some promising results (García et al., 2012; Hodge and Austin, 2004; Chandola et al., 2009).

However, since we live in a non-stationary environment, datasets in many domains do not always contain fixed numbers of data samples. In other words, new data samples are continually being added to the database for data mining, which causes the dataset size to become larger and larger. As a consequence, wrong decisions could be made if mining results are discovered from 'out of date' datasets.

For example, instance selection or outlier detection performed over a given dataset D_1 containing 10,000 examples collected at a specific time T_1, results in a reduced dataset $D_{1\_reduced}$ for the later mining stage (where the size of $D_{1\_reduced}$ is smaller than D_1). However, after some time, the size of D_1 becomes larger as new data samples, D_{new}, are stored. As a result, a new larger dataset D_2, composed of D_1 and D_{new}, is created at time T_2. At this point, there are two possible strategies for performing instance selection or outlier detection. The first one, the common strategy, is usually employed over D_2. This can be regarded as the static environment problem without considering the growing dataset. The second one involves performing instance selection over D_{new} resulting in $D_{new\_reduced}$ where the reduced dataset of D_2 is the combination of $D_{1\_reduced}$ and $D_{new\_reduced}$.

In these two cases, the computational cost of performing this data-processing task becomes higher and higher as the new dataset becomes larger and larger in size. This creates the problem of a very high computational cost which is required for performing instance selection over D_2 or D_{new}.

To this end, we introduce a novel process, namely ReDD (**Re**presentative **D**ata **D**etection) which is based on analyzing (or learning) the patterns of unrepresentative data that are identified in the instance selection step. These patterns are then used as guidelines to predict whether a new data sample is representative or not. This prediction task can be accomplished by training a supervised machine learning model. The hypothesis behind ReDD is that if (un)representative data can be well predicted over a set of new data samples, there is no need to perform instance selection again over a new, larger dataset. For the previous example, we only need to train a specific classifier over a two-class training set composed of the representative group (i.e., $D_{1\_reduced}$) and the unrepresentative group (i.e., $D_1 - D_{1\_reduced}$). The classifier can then be used to distinguish between representative and unrepresentative data over D_{new}.

Consequently, the time cost of ReDD over D_{new} can be much smaller than that of performing instance selection over D_2 or D_{new} since the time for performing on-line classification as testing is usually much shorter than off-line learning as training (Chang et al., 2010; Edakunni and Vijayakumar, 2009). In our case, the total time for training a classifier over D_1, and testing the classifier to perform representative data detection as the on-line classification task over D_{new}, is smaller than directly performing instance selection over D_2 or D_{new}, especially when D_2 or D_{new} is certainly larger than D_1.

Note that detecting (un)representative data using ReDD is different from the existing outlier detection methodology used to detect (non)outliers. First, outlier detection aims to detect whether a new exemplar lies in a region of normality, but ReDD focuses on training a classifier to classify a new exemplar into one of two pre-defined classes (i.e., representative and unrepresentative classes) without considering the 'normal' and 'abnormal' data distributions. Second, for supervised learning based upon outlier detection approaches, the number of outliers in the training dataset is usually very small. In addition, the training dataset is typically based on manually labeling normal and abnormal data. On the other hand, the generation of the training dataset in ReDD is based on instance selection, which usually contains a large number of unrepresentative data and a small number of representative data,[1] with the labeling for the two groups of data being fully automatic.

The rest of this paper is organized as follows. Section 2 briefly reviews related literature of instance selection and outlier detection. Section 3 introduces the proposed ReDD process for (un)representative data analysis and prediction. Section 4 presents the experimental results and the conclusion is provided in Section 5.

2. Literature review

2.1. Instance selection

Instance selection can be defined as follows. Given a dataset D composed of training set T and testing set U, let X_i be the ith instance in D, where $X_i = (X_1, X_2, \ldots, X_m)$ which contains m different features. Let $S \subset T$ be the subset of selected instances that result from the execution of an instance selection algorithm. Then, U is used to test a classification technique trained by S (Cano et al., 2003; García et al., 2012).

In the literature, there are a number of related studies proposing instance selection methods for obtaining better mining quality. Specifically, Pradhan and Wu (1999) and Jankowski and Grochowski (2004) surveyed several relevant selection techniques, which can be divided into three application-type groups: noise filters, condensation algorithms, and prototype searching algorithms. In addition, extensive comparative experiments were conducted by Wilson and Martinez (2000), García-Pedrajas et al. (2010), and García et al. (2012). Some cutting-edge instance selection algorithms have been identified, such as Decremental Reduction Optimization Procedure 3 (DROP3), and Genetic Algorithms (GA), which make the k-NN classifiers provide better performance over other instance selection methods.

The noise-filtering algorithms are usually based on the nearest neighbor principle to remove data points which do not agree with the majority of its k nearest neighbor. For condensation algorithms, IB3 (Aha et al., 1991) and DROP3 (Wilson and Martinez, 2000) are two representative algorithms. In IB3, instance x from the training set T is added to a new set S if the nearest acceptable instance in S has different class than x, in which acceptability is defined by a confidence interval

$$\frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(p-1)}{n} + \frac{z^2}{2n^2}}}{1 + \frac{z^2}{n}} \quad (1)$$

where z is a confidence factor, p is the classification accuracy of a given instance (while added to S), and n is equal to the number of classification-trials for the given instance (while added to S).

On the other hand, the Decremental Reduction Optimization Procedure 1 (DROP1) uses the following basic rule to decide if it is safe to remove an instance from the instance set S (where $S = T$ originally):

> Remove P if at least as many of its associates in S would be classified correctly without P. (2)

DROP2 starts the process from sorting instances according to their distances from the nearest opposite class instance. The DROP3 algorithm additionally performs the noise filtering approach before starting the DROP2 algorithm.

Finally, the genetic algorithm (GA) (Cano et al., 2003) is one type of prototype searching algorithm. In general, it uses a population of strings (called chromosomes), which encode candidate solutions

[1] In García-Pedrajas et al. (2010) and García et al. (2012), the reduction rates for instance selection by state-of-the-art algorithms over various domain datasets are very high, i.e. about 80% on average. This means that a large amount of data in each dataset is filtered out.