Neurocomputing 147 (2015) 426–434

Contents lists available at ScienceDirect

Neurocomputing

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

Active learning via query synthesis and nearest neighbour search

Liantao Wang [a], Xuelei Hu [a,c,*], Bo Yuan [d], Jianfeng Lu [a,b]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[b] Jiangsu Key Laboratory of Image and Video Understanding for Social Safety, Nanjing University of Science and Technology, Nanjing 210094, China
[c] School of Information Technology and Electrical Engineering, University of Queensland, Brisbane QLD 4072, Australia
[d] Intelligent Computing Lab, Division of Informatics, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China

ARTICLE INFO

Article history:
Received 22 August 2013
Received in revised form
26 May 2014
Accepted 19 June 2014
Communicated by Steven Hoi
Available online 19 July 2014

Keywords:
Active learning
Query synthesis
Pool-based sampling
Kernel function

ABSTRACT

Active learning has received great interests from researchers due to its ability to reduce the amount of supervision required for effective learning. As the core component of active learning algorithms, query synthesis and pool-based sampling are two main scenarios of querying considered in the literature. Query synthesis features low querying time, but only has limited applications as the synthesized query might be unrecognizable to human oracle. As a result, most efforts have focused on pool-based sampling in recent years, although it is much more time-consuming. In this paper, we propose new strategies for a novel querying framework that combines query synthesis and pool-based sampling. It overcomes the limitation of query synthesis, and has the advantage of fast querying. The basic idea is to synthesize an instance close to the decision boundary using labelled data, and then select the real instance closest to the synthesized one as a query. For this purpose, we propose a synthesis strategy, which can synthesize instances close to the decision boundary and spreading along the decision boundary. Since the synthesis only depends on the relatively small labelled set, instead of evaluating the entire unlabelled set as many other active learning algorithms do, our method has the advantage of efficiency. In order to handle more complicated data and make our framework compatible with powerful kernel-based learners, we also extend our method to kernel version. Experiments on several real-world data sets show that our method has significant advantage on time complexity and similar performance compared to pool-based uncertainty sampling methods.

1. Introduction

Active learning is an important approach to constructing a high performance classifier while keeping the amount of supervision to a minimum by actively selecting the most valuable training instances. As an effective way to reduce the cost of human labelling, it has been successfully applied to various applications [1,2], especially when labelling is difficult or time-consuming. In a typical active learning cycle, the algorithm selects the most valuable (informative [3] or representative [4]) instance and requests its label. Then the new labelled instance is added to the training set, and the classifier is retrained. Note that how to form a query plays a key role in an active learning algorithm. In terms of query formation, there are two scenarios of active learning in the literature: query synthesis [5–7] and sampling, which can be further divided into stream-based sampling [8,9] and pool-based sampling [3,10–12].

In the scenario of query synthesis, the learner may generate a query in the form of any unlabelled instance in the input space, i.e. the query can be fictitious. By contrast, active learning based on sampling selects real instances from the unlabelled set. Note that stream-based sampling and pool-based sampling can share the same criterion of value measure (e.g. uncertainty [3], query by committee [13]), and the only difference is the way they access the unlabelled data. Concretely, stream-based sampling selects one instance at a time and decides whether to query it or not. Pool-based sampling, however, maintains a pool consisting of unlabelled data. At each iteration, it evaluates and ranks the entire collection of unlabelled data before selecting the most valuable one. Since pool-based sampling generates queries in a greedy fashion, it is more effective and has attracted most of the research interests, while stream-based sampling is only appropriate in some special situations where memory or processing power is limited.

Since query synthesis generates a query using a small amount of labelled data, it is therefore very efficient. However sometimes synthesized queries are unrecognizable to human oracle [14]. Pool-based sampling is effective as it generates a query by evaluating the entire unlabelled set, but it is very time-consuming. To make the

* Corresponding author.
E-mail addresses: ltwang.nust@gmail.com (L. Wang), xlhu@njust.edu.cn, xuelei.hu@uq.edu.au (X. Hu), yuanb@sz.tsinghua.edu.cn (B. Yuan), lujf@njust.edu.cn (J. Lu).

http://dx.doi.org/10.1016/j.neucom.2014.06.042
0925-2312/

querying both fast and effective, we have proposed a framework that combines query synthesis and pool-based sampling in [15]. In this paper, we propose a new strategy for query synthesis and extend the framework to kernel version. The main idea is that, at each iteration, we synthesize an instance close to the current classification boundary and search for its nearest neighbour among the unlabelled instances as the actual query. Specifically, from the initially labelled instances, we can obtain one positive instance and one negative instance. We call these two instances with opposite labels an *Opposite Pair*. According to the initial *Opposite Pair*, we first use an efficient method to find another *Opposite Pair* close to the classification boundary. After that, we iteratively synthesize a query along the midperpendicular of the previously found *Opposite Pair*. This can guarantee that the queries are close to the classification boundary and well dispersed.

Since our method synthesizes a query directly, instead of evaluating every instance in the unlabelled data pool, it has the advantage of efficiency. Also by using the real instance nearest to the synthesized one, it can make sure that the query is recognizable to a human oracle. Our algorithm can be further accelerated by using various approximate nearest neighbour search techniques [16].

This strategy can select the instances closest to the decision boundary, which are most informative. Moreover, instead of only considering the informativeness of the query, we also take into account the representativeness, and introduce pre-clustering in our method to exploit the local structure of the data and construct a compact and representative unlabelled pool based on local centre points.

In order to handle more complicated data and make our framework compatible with kernel-based learners such as support vector machine (SVM), we further extend this framework with the query strategy to kernel version. Queries can be synthesized in the feature space without knowing the explicit non-linear mapping function by kernel trick, which has been exploited in many machine learning methods [17–20].

The rest of this paper is organized as follows: In Section 2, we review the work related to our approach. Section 3 introduces our approach in detail. Experimental results are reported in Section 4. Section 5 concludes this work.

2. Related work

2.1. Query synthesis

Query synthesis was first proposed in [5], and further studied in [21]. In this setting, a membership query is generated in the form of any unlabelled instance in the input space. Baum [6] used interpolation to synthesize queries to find separating hyperplane efficiently, but later demonstrated that this kind of query cannot work properly in vision-based task [14], because the human oracle cannot recognize the query synthesized by the algorithm. After that, although King et al. [22,7] found a promising real-world application of query synthesis, few efforts have focused on synthesis query since an arbitrary query might be meaningless and difficult for human to label.

2.2. Pool-based sampling

Pool-based sampling has been the most prosperous branch of active learning, due to its effectiveness. It has been widely used in many real world applications (e.g., text categorization [23], video search [24], image classification [25] and action retrieval [26]). Pool-based active learning was first introduced by Lewis and Gale [10]. The algorithm maintains a pool consisting of unlabelled instances and selects the most informative one at each iteration. The main issue with active learning in this scenario is how to measure the informativeness. The most commonly used strategies are uncertainty sampling [10,27,28,11] and query by committee [29,30,13]. There are also methods aiming at expected error reduction [31,32]. Strategies such as uncertainty sampling and query by committee can select the instances closest to the decision boundary, which is most informative. However they only measure the value of a single instance, as a result they may suffer from querying similar instances repeatedly. To overcome this limitation, the local structure of the data can be considered while selecting queries. For example, clustering was introduced into active learning in [33,34] to select the most representative instances. Representativeness is also taken into account in batch mode active learning [35], where the authors considered an instance's similarity to the remaining unlabelled instances. Huang et al. [12] extended this min–max view of active learning to take into account both the cluster structure of unlabelled instances and the class assignments of the labelled instances. More recently, Zhang et al. [4] used locally linear reconstruction to exploit the intrinsic geometrical structure of the data, so as to select the most representative instances.

3. Methodology

Suppose we have a training data set denoted by $\mathcal{D}=\mathcal{L}\cup\mathcal{U}$, where $\mathcal{L}$ is the labelled set with very small size and $\mathcal{U}$ consists of large amount of unlabelled instances. The goal is to select the most valuable instances for the classifier training from the unlabelled set.

3.1. Algorithm

Similar to many other active learning algorithms, we assume that the instances close to the classification boundary are generally more ambiguous and their labels will provide more information to the classifiers. As a result, we aim to find instances close to the classification boundary.

Suppose $\{x_+, x_-\}$ is an *Opposite Pair*. We can find instances on a separating plane with high precision by interpolating iteratively similar to binary search: We always query the point located in the middle of the closest *Opposite Pair*. Concretely, let $x_1=(x_++x_-)/2$ and query its label. If x_1 is positive (negative), we then query the midpoint of x_1 and $x_-(x_+)$. Repeating this process b times, we can guarantee that x_b is on a separating plane with b bits of precision [6]. An illustration of this process is shown in Fig. 1. Since the synthesized query may not be recognized by the human oracle, in practice we instead query its nearest neighbour rather than itself.

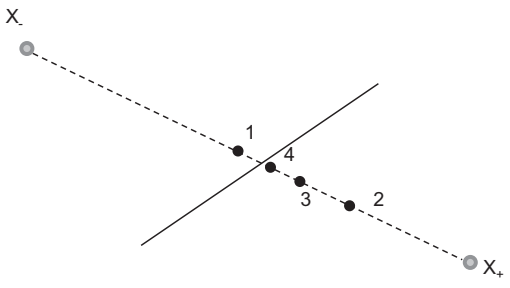

Fig. 1. The binary search process used to find a point very nearly on the separating hyperplane given $\{x_+, x_-\}$. Queries are always generated by the midpoint of the closest *Opposite Pair*. The first point **1** is positive. The second query thus is the midpoint between query **1** and x_-. After b queries, we have a point on the hyperplane with accuracy $2^{-b}d$, where d is the distance between x_+ and x_-.