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/© 2014 Elsevier B.V. All rights reserved.

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.

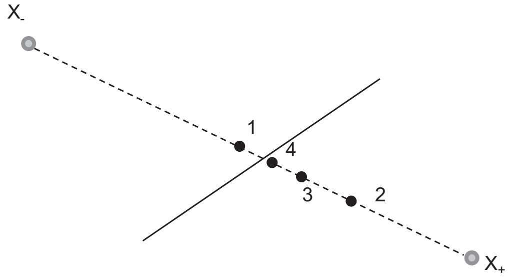


Fig. 1. The binary search process used to find a point very nearly on the separating hyperplane given $\{x_+, x_-\}$. Queries are always generated by the midpoint of the closest *Opposite Pair*. The first point **1** is positive. The second query thus is the midpoint between query **1** and x_-. After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

If we always use binary search to generate queries, it can be guaranteed that the queries are close to the boundary with high precision, but the queries will concentrate in a local neighbourhood. Ideally, we expect the queries to locate close to the separating plane dispersedly. As a result, after we find an *Opposite Pair* $\{x_+, x_-\}$ close to the separating plane, we make the next query by adding their midpoint with a small vector: $x_s = (x_+ + x_-)/2 + x_r$. Since we want the queries always close to the separating plane, the small vector x_r should be orthogonal to the connecting line of the *Opposite Pair*. Similarly, we search its nearest neighbour denoted by x_q. If the label of x_q is positive, we then repeat this process using x_q and x_-. See Fig. 2 for a simple example.

Overall, given a data set with a small labelled set, we initialize the *Opposite Pair* with the centroids of positive labelled set and negative labelled set. Then we find an *Opposite Pair* close enough to the separating hyperplane. After that, we generate queries along the midperpendicular of newly founded *Opposite Pair*. We expect that the queries are distributed in the shape of net on the sides of the separating plane.

We now specify these procedures. Let c_+ and c_- be the centroids of positive and negative labelled instances respectively.

$$c_+ = \frac{1}{|\mathcal{L}_+|}\sum_{x_i \in \mathcal{L}_+} x_i, \quad c_- = \frac{1}{|\mathcal{L}_-|}\sum_{x_i \in \mathcal{L}_-} x_i.$$

We then synthesize the first instance $x_s = \frac{1}{2}(c_+ + c_-)$ and query its label. Since this is a pseudo-instance, it might not be recognized by the human annotator, especially in vision-based tasks. Instead, we search its nearest neighbour (denoted by x_q) from the unlabelled set and query its label. If positive, we then use the midpoint of x_q and c_- as a new synthesized instance. Repeating this process b times, we can find an *Opposite Pair* $\{x_+, x_-\}$ close to the classification boundary. See Algorithm 1.

Algorithm 1. Find *Opposite Pair* close to separating hyperplane. $(\{x_+, x_-\}, \mathcal{L}, \mathcal{U}) = findPair(\{x^o_+, x^o_-\}, \mathcal{L}^o, \mathcal{U}^o, b)$.

```
Require: $\{x^o_+, x^o_-\}$ – Initial Opposite Pair; $\mathcal{L}^o$ – Current labelled
    set; $\mathcal{U}^o$ – Current unlabelled set; $b$ – Number of binary
    search
Ensure: $\{x_+, x_-\}$ – Opposite Pair close to separating
    hyperplane; $\mathcal{L}$ – Updated labelled set; $\mathcal{U}$ – Updated
    unlabelled set;
    for $i = 1$ to $b$ do
        Synthesize an instance $x_s = (x_+ + x_-)/2$
        $x_q \leftarrow nnSearch(x_s, \mathcal{U})$
        Query $x_q$
        $\mathcal{L} = \mathcal{L} \cup x_q$, $\mathcal{U} = \mathcal{U} \backslash x_q$
        if $x_q$ is positive then
            $x_+ \leftarrow x_q$
        else
            $x_- \leftarrow x_q$
        end if
    end for
    return $\{x_+, x_-\}$, $\mathcal{L}$, $\mathcal{U}$
```

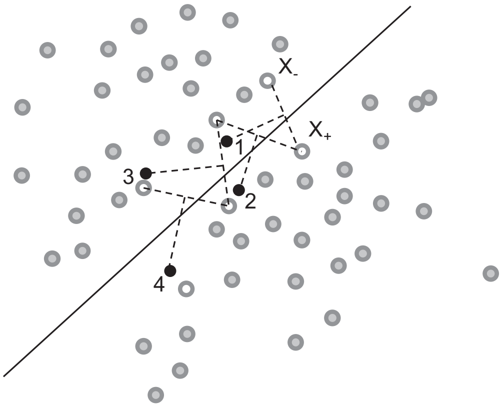


Fig. 2. A simple procedure for rest queries after finding an *Opposite Pair* $\{x_+, x_-\}$ close to the separating hyperplane. We synthesize the next query using a vector starting from the midpoint of their connecting line and orthogonal to the connecting line. Queries are always generated by the latest found *Opposite Pair*.

After we find an *Opposite Pair* close enough to the classification boundary, we generate the next query in the direction of its midperpendicular. This new synthesis can be obtained by adding an orthogonal vector to the midpoint. We can find the vector by orthogonalizing a random vector using Gram–Schmit process and normalize its magnitude to λ. See Algorithm 2 for details. Similarly, we search for its nearest neighbour from the unlabelled set and query the label. If positive, we then use x_q and x_- to generate the next query. The entire algorithm is shown in Algorithm 3.

Algorithm 2. Synthesize midperpendicular query for *Opposite Pair*. $x_s = getPerpQuery(\{x_+, x_-\}, \lambda)$.

```
Require: $\{x_+, x_-\}$ – Opposite Pair; $\lambda$ – Magnitude of
    midperpendicular vector;
Ensure: $x_s$ – Synthesis
    $x_o = x_+ - x_-$
    Generate a random vector $x_r$
    Use Gram–Schmit process to make $x_r$ orthogonal to $x_o$,
    $x_r = x_r - \langle x_r, x_o \rangle / \langle x_o, x_o \rangle * x_o$
    Set the magnitude to $\lambda$, $x_r = \lambda / norm(x_r) * x_r$
    Translate it to the midpoint, $x_s = x_r + (x_+ + x_-)/2$
    return $x_s$
```

Algorithm 3. Proposed framework of active learning.

```
Require: $\mathcal{L}$ – Labelled set; $\mathcal{U}$ – Unlabelled set; $n$ – Number of
    instances toselect; $b$ – Number of binary search;
    $\lambda$ - Magnitude of midperpendicular
        vector; Learner - Learner
Ensure: $h$ – hypothesis
    Obtain $c_+$ and $c_-$
    $(\{x_+, x_-\}, \mathcal{L}, \mathcal{U}) = findPair(\{c_+, c_-\}, \mathcal{L}, \mathcal{U}, b)$
    for $i = b+1$ to $n$ do
        $x_s = getPerpQuery(\{x_+, x_-\}, \lambda)$
        $x_q = nnSearch(x_s, \mathcal{U})$
        Query $x_q$
        $\mathcal{L} = \mathcal{L} \cup x_q$, $\mathcal{U} = \mathcal{U} \backslash x_q$
        if $x_q$ is positive then
            $x_+ \leftarrow x_q$
        else
            $x_- \leftarrow x_q$
        end if
    end for
    $h = Learner(\mathcal{L})$;
    return $h$
```

The proposed approach can overcome the issue of meaningless queries by using their nearest neighbours instead. As to computational complexity, since query synthesis only works on a small set of labelled data, most computational cost of querying is on the nearest neighbour search. The computational complexity of the linear search is $O(kd)$, where k is the size of the pool and d is the dimensionality. Our algorithm can be further sped up by using fast approximate nearest neighbour search.

3.2. Kernel extension

In order to extend our framework to kernel version, we need to address the issues of binary search and midperpendicular synthesis in feature space. Consider the same data set $\mathcal{D}=\mathcal{L}\cup\mathcal{U}$, we project the data into an appropriate feature space by a nonlinear mapping $\phi(\mathrm{x})$.

3.2.1. Binary search in feature space

In Section 3.1, we use an *Opposite Pair* $\{\mathrm{x}_+,\mathrm{x}_-\}$ to obtain instances close to the separating hyperplane with binary search. We can also do this in feature space. Given an *Opposite Pair* $\{\mathrm{x}_+,\mathrm{x}_-\}$, i.e. $\{\phi(\mathrm{x}_+),\phi(\mathrm{x}_-)\}$ in feature space, the unlabelled instance nearest to their midpoint in feature space can be obtained by solving:

$$\min_{\mathrm{x}\in\mathcal{U}}\left\|\phi(\mathrm{x})-\frac{1}{2}[\phi(\mathrm{x}_+)+\phi(\mathrm{x}_-)]\right\|_2^2,$$

which can be reformulated by introducing kernel functions

$$\min_{\mathrm{x}\in\mathcal{U}}(k(\mathrm{x},\mathrm{x})-k(\mathrm{x},\mathrm{x}_+)-k(\mathrm{x},\mathrm{x}_-)).$$

3.2.2. Midperpendicular synthesis in feature space

Suppose we already have an *Opposite Pair* $\{\mathrm{x}_+,\mathrm{x}_-\}$, and in feature space they become $\{\phi(\mathrm{x}_+),\phi(\mathrm{x}_-)\}$. Similarly we should generate a random vector, here we randomly select an instance x_r in the data set, then used Gram–Schmit to orthogonalize it and set its magnitude to λ. After we obtain the synthesis, we then search its nearest neighbour in the unlabelled pool as the actual query. The instance in unlabelled pool closet to the synthesized query can be obtained by solving:

$$\min_{\mathrm{x}\in U}(k(\mathrm{x},\mathrm{x})+m_1k(\mathrm{x},\mathrm{x}_r)+m_2k(\mathrm{x},\mathrm{x}_+)+m_3k(\mathrm{x},\mathrm{x}_-)),$$

where

$$m_1=-2\lambda/\sqrt{k(\mathrm{x}_r,\mathrm{x}_r)-\alpha^2/\beta},$$
$$m_2=-1+2\lambda\alpha/\sqrt{\beta^2k(\mathrm{x}_r,\mathrm{x}_r)-\alpha^2\beta},$$
$$m_3=-1-2\lambda\alpha/\sqrt{\beta^2k(\mathrm{x}_r,\mathrm{x}_r)-\alpha^2\beta}$$

with $\alpha=k(\mathrm{x}_r,\mathrm{x}_+)-k(\mathrm{x}_r,\mathrm{x}_-)$, and $\beta=k(\mathrm{x}_+,\mathrm{x}_+)+k(\mathrm{x}_-,\mathrm{x}_-)-2k(\mathrm{x}_+,\mathrm{x}_-)$.

3.3. Representative and compact pool

To also take into account the representativeness of our queries, we exploit clustering techniques on the whole data set as a preprocessing, and collect the unlabelled instances nearest to the local centres to construct a representative and compact unlabelled data pool.

We first use a clustering technique (e.g., k-means) to process the original data set. In this process, $\mathcal{D}$ is partitioned into K clusters $\{\mathcal{D}_1,\mathcal{D}_2,\ldots,\mathcal{D}_K\}$, where $\mathcal{D}_i\cap\mathcal{D}_j=\varnothing$ for any $i\neq j$ and $\mathcal{D}=\uplus_{i=1}^K\mathcal{D}_i$. Each cluster consists of a group of data points that are similar to each other. For cluster $\mathcal{D}_i$ $(1\le i\le K)$, the centroid (cluster mean vector) c_i is

$$\mathrm{c}_i=\frac{1}{|\mathcal{D}_i|}\sum_{\mathrm{x}_j\in\mathcal{D}_i}\mathrm{x}_j$$

Then, the data point $\mathrm{x}_{\mathrm{c}_i}$ in $\mathcal{U}$ nearest to c_i can be found by nearest neighbour search strategies. We therefore obtain a set $\mathcal{U}_c$ containing nearest neighbours of centroids:

$$\mathcal{U}_c=\left\{\arg\min_{\mathrm{x}\in\mathcal{U}}\|\mathrm{x}-\mathrm{c}_i\|_2^2, i=1,\ldots,K\right\}$$

With clustering techniques, a set of clusters $\{\mathcal{D}_1,\ldots,\mathcal{D}_K\}$ and their best representation $\mathcal{U}_c$ can be obtained to represent the general distribution of $\mathcal{D}$. If the label of $\mathrm{x}_{\mathrm{c}_i}$ is given, it will provide important information for the labels of data points in $\mathcal{D}_i$. We therefore generate queries from $\mathcal{U}_c$ in our approach. Note that $|\mathcal{U}_c|=K$ and $\mathcal{U}_c\subset\mathcal{U}$. The size of the compact pool $\mathcal{U}_c$ is smaller than that of the original unlabelled data set, the proposed approach is computationally efficient and practical for large scale problems.

For the pool construction in feature space, we first project the data set $\mathcal{D}$ into an appropriate feature space by a nonlinear mapping $\phi(\mathrm{x})$. Similarly, kernel clustering technique can be used to partition $\mathcal{D}$ into clusters $\{\mathcal{D}'_1,\mathcal{D}'_2,\ldots,\mathcal{D}'_K\}$. So, we now construct the compact and representative unlabelled data pool by finding the nearest neighbour to the centroid of each cluster. Concretely, for each cluster $\mathcal{D}'_i$ $(1\le i\le K)$, the centroid in feature space can be represented by $\mathrm{c}'_i=(1/|\mathcal{D}'_i|)\sum_{\mathrm{x}_j\in\mathcal{D}'_i}\phi(\mathrm{x}_j)$, where $|\mathcal{D}'_i|$ is the number of instances in cluster $\mathcal{D}'_i$. Then the nearest neighbour can be obtained by

$$\min_{\mathrm{x}\in\mathcal{U}}\|\phi(\mathrm{x})-\mathrm{c}'_i\|_2^2,$$

which is equivalent to

$$\min_{\mathrm{x}\in\mathcal{U}}\left(k(\mathrm{x},\mathrm{x})-\frac{2}{|\mathcal{D}'_i|}\sum_{\mathrm{x}_j\in\mathcal{D}'_i}k(\mathrm{x},\mathrm{x}_j)\right).$$

Then a compact and representative unlabelled data pool can be constructed:

$$\mathcal{U}_c=\left\{\arg\min_{\mathrm{x}\in\mathcal{U}}\left(k(\mathrm{x},\mathrm{x})-\frac{2}{|\mathcal{D}'_i|}\sum_{\mathrm{x}_j\in\mathcal{D}'_i}k(\mathrm{x},\mathrm{x}_j)\right),\ i=1,\ldots,K\right\}.$$

4. Experiments

In this section, we demonstrate the performance of the proposed method on an artificial data set, UCI data sets [36] and MNIST database of handwritten digits [37].

We compared our methods with random sampling, uncertainty-based sampling method and LLR-based active learning [4]. (i) Random sampling: The algorithm randomly selects an instance in each round of iteration. This is actually a passive learning method. (ii) Uncertainty-based sampling: The algorithm selects the most uncertain instance as next query. For Nearest Neighbour learner, the uncertainty of each unlabelled point is measured by the vote entropy of its five nearest neighbours. For the SVM learner, we used libsvm [38] to get a posterior probability for each instance. The most uncertain instance is the one with a posterior probability closest to 0.5. (iii) LLR-based active learning in [4] selects the most representative instance at each iteration based on locally linear reconstruction. We evaluated the performances of different competitors by the curves of classification accuracy against the size of labelled set.

For each data set, we randomly partitioned it into the initial labelled data set $\mathcal{L}$, the test set $\mathcal{T}$ and the unlabelled data pool $\mathcal{U}$, which accounted for 5%, 25% and 70% of the total size N_{total} respectively. This partition process was repeated 200 times to reduce randomness. All the results shown in this section are the average values over 200 repetitions. In addition, as described in Section 3.3, we clustered the whole data set into $\lfloor N_{total}/10\rfloor$ groups using (kernel) k-means. Instances closest to centroids in $\mathcal{U}$ was selected to construct the compact pool $\mathcal{U}_c$ for query generation.

A PC with AMD Athlon 7550 CPU (2.50 GHz) and 2 GB RAM with 64-bit Win7 and Matlab 2013a was used as the experimental platform.

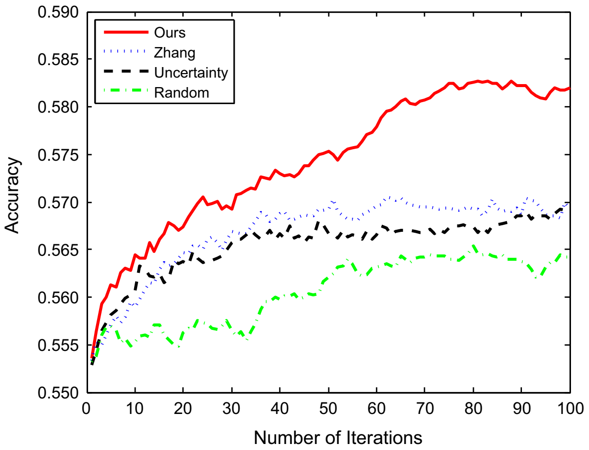


Fig. 3. Results on artificial data set by Nearest Neighbour learner without pre-clustering.

4.1. Artificial data set

The artificial data set consists of two classes of data that are linearly separable in $\mathbb{R}^{100}$. We first got a hyperplane in the space by randomly generating its normal vector $w \in \mathbb{R}^{100}$. Then we randomly sampled in the space, and collected 250 data points with $w^T x > 0$ and 250 data points with $w^T x < 0$. To validate the effectiveness of our synthesis strategy, we compared our algorithm without pre-clustering with the baselines. The results are shown in Fig. 3. It can be seen that our synthesis-based query strategy outperforms the baselines significantly, which demonstrates the effectiveness of our synthesis query even without pre-clustering.

4.2. UCI data sets

We investigate the performance of the competitors on 6 UCI data sets, which are described briefly in Table 1.

We first consider the comparison using Nearest Neighbour learner. Since Nearest Neighbour learner has shown good applicability to nonlinear problems, we do not use nonlinear mapping, i.e., these querying proceeds in the original space. As we can see from the plots in Fig. 4, our methods outperform all the others in terms of average

Table 1
Description of data sets.

Datasets	#cls	#attr	#inst	$\lvert L\rvert$	$\lvert U\rvert$	$\lvert T\rvert$
bupa	2	6	345	35	224	86
diabetes	2	8	768	77	499	192
heart	2	13	270	27	175	68
ionosphere	2	34	351	35	228	88
spambase	2	57	4601	460	2991	1150
wdbc	2	31	569	57	370	142

Table 2
Time of querying using Nearest Neighbour learner ($\times 10^{-4}$ s).

Datasets	Ours	Zhang	Uncertainty	Random
bupa	5+1	3904	96	1
diabetes	7+5	106 893	231	2
heart	4+2	1999	78	1
ionosphere	6+3	5125	111	2
spambase	12+177	1 072 720	8236	3
wdbc	7+6	45 540	194	2

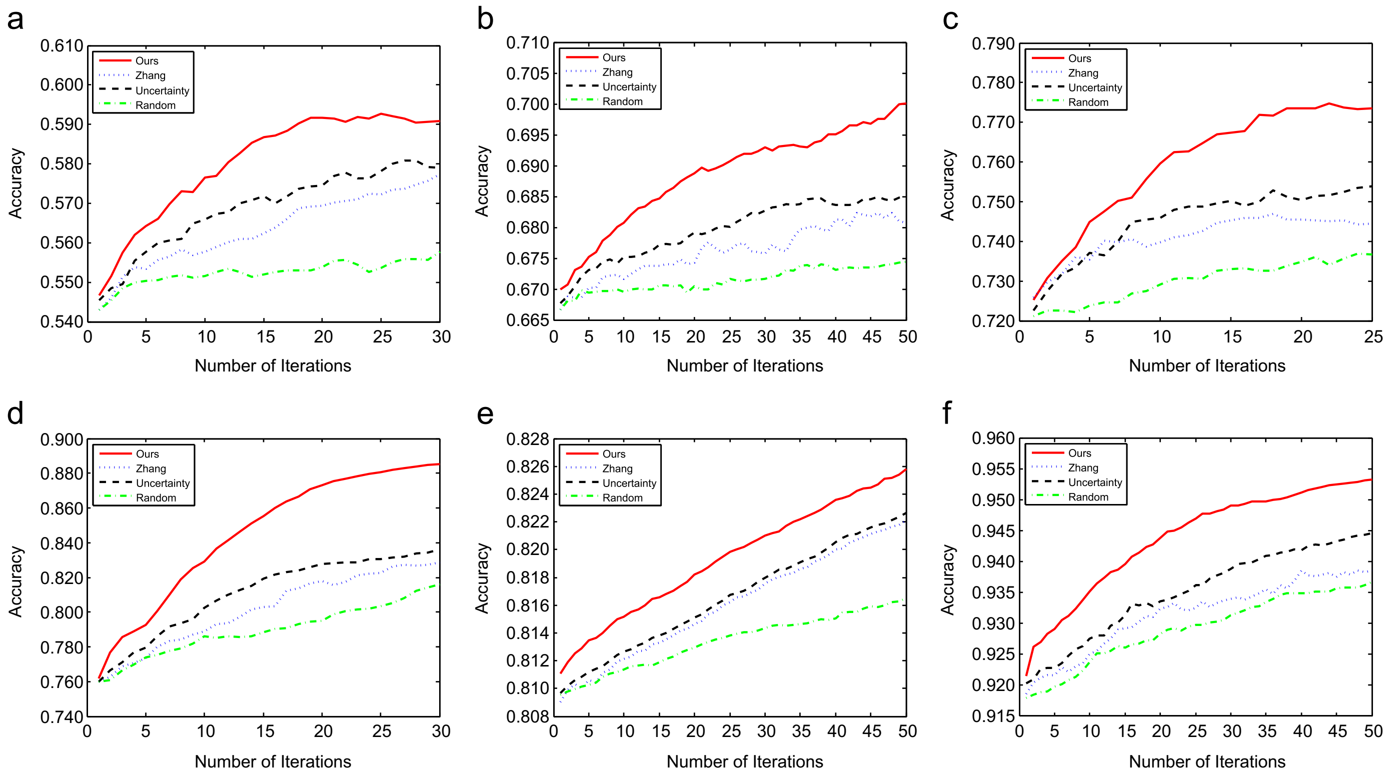


Fig. 4. Plots of mean accuracy with respect to the number of iterations on UCI data sets using Nearest Neighbour learner. (a) bupa, (b) diabetes, (c) heart, (d) ionosphere, (e) spambase, (f) wdbc.

accuracy. To explore the efficiency of our strategy, we also listed the average CPU time costs of each round of iteration of different methods in Table 2. The time cost of our framework included that for pre-processing (clustering and kernel matrix computation for SVM), which was averaged into every iteration. Note that this cost can be reduced when the number of iteration is increased. So we listed the cost in smaller fonts. It is clear that the time costs of our methods are similar to those of random sampling, which are much smaller than those of uncertainty sampling and [4].

In order to show the ability to be compatible with kernel learners and get higher accuracies, we evaluate the performance of our framework combined with SVM. For every competitor, all of the data were mapped into a high-dimension space using RBF kernel. Fig. 5 shows the curves of the average classification accuracies of each competitor using SVM learner. As we can see from the plots, our methods are superior to method in [4] and random sampling and have slightly worse performance than uncertainty sampling. As we have developed kernel version, our framework can be combined with more powerful kernel SVM learner, the accuracies of every competitor are higher than that obtained using Nearest Neighbour learner. The average CPU time costs of each round of iteration of different methods are shown in Table 3. It is clear that the time costs of our methods are similar to those of random sampling, which are much smaller than those of uncertainty sampling and [4].

4.3. Handwritten digit recognition

Lang and Baum's query learning failed in a handwritten digit recognition task because the human oracle could not understand the image synthesized by the algorithm [14]. To address this problem, our algorithms query the real instance closest to the synthesized one. In order to demonstrate the competence of our algorithms in vision-based recognition tasks, we conducted hand-written digit recognition experiments on MNIST database. MNIST database is a popular handwritten digit data set containing 60 000

Table 3
Time of querying using SVM learner ($\times 10^{-4}$ s).

Datasets	Ours	Zhang	Uncertainty	Random
bupa	9+3	3904	103	2
diabetes	12+10	106 893	340	2
heart	4+4	1999	30	2
ionosphere	6+5	5125	57	2
spambase	174+382	1 072 720	5173	39
wdbc	4+9	45 540	77	2

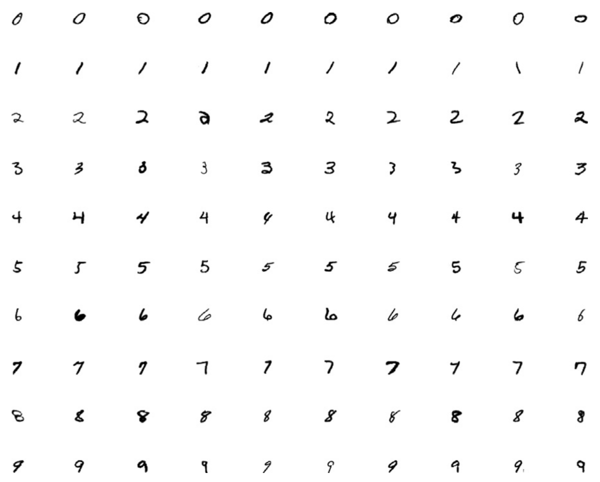


Fig. 6. Examples of MNIST data.

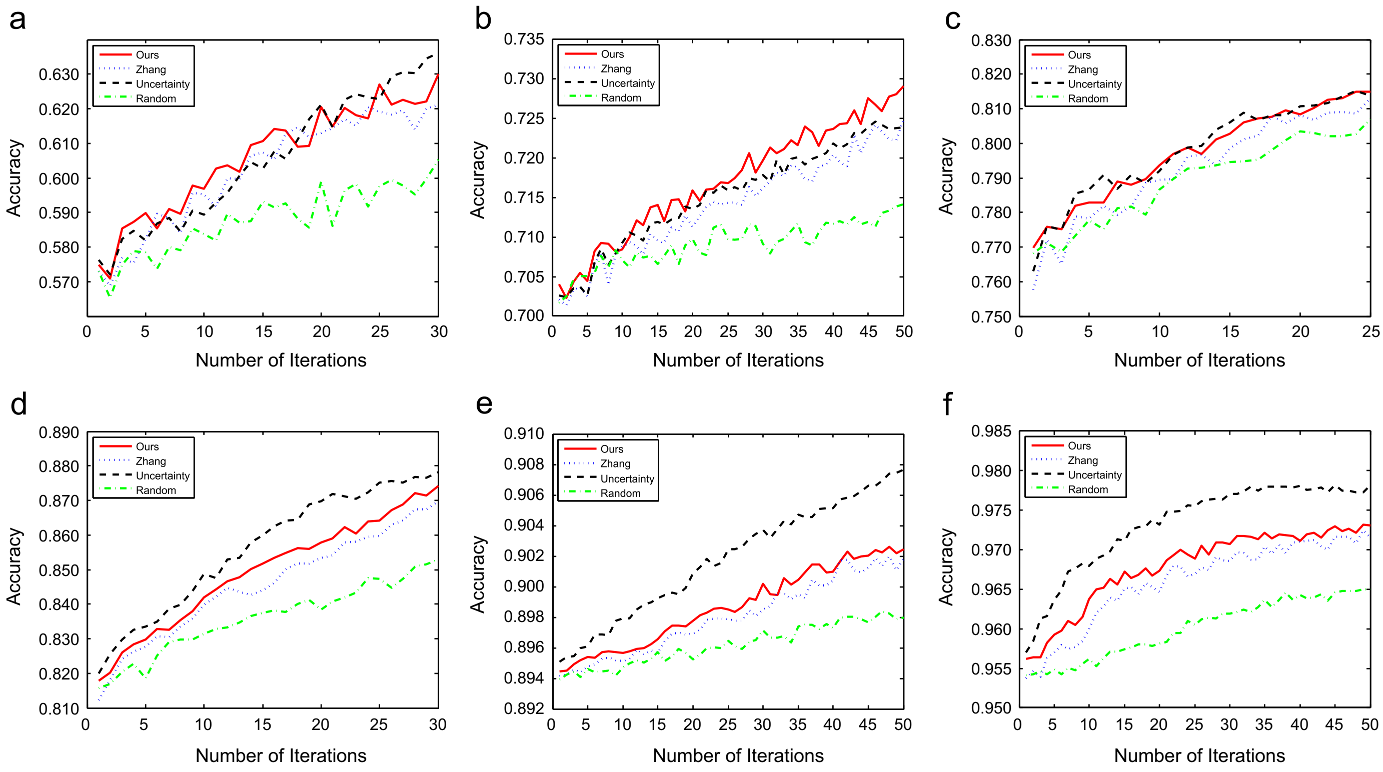


Fig. 5. Plots of mean accuracy with respect to the number of iterations on UCI data sets using SVM learner. (a) bupa, (b) diabetes, (c) heart, (d) ionosphere, (e) spambase, (f) wdbc.

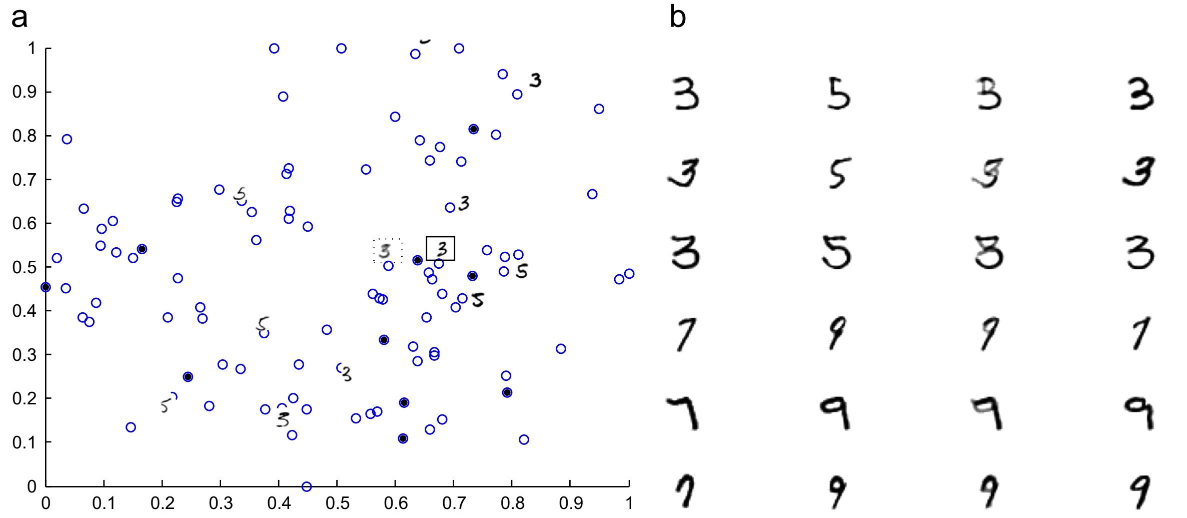


Fig. 7. Illustration of experiments on MNIST data. (a) Illustration of a single iteration in the proposed method. The hollow points indicate unlabelled data points and the solid points indicate local centre points. The digit images without frame indicate the labelled data points. The digit image with dotted frame indicates the synthesized query using binary search. The digit image with solid frame indicates the nearest neighbor of the synthesized query among local centre points. (b) Examples of queries by the proposed method. The first two columns are *Opposite Pairs* found in the experimental process. Their midpoints (the synthesized queries) are shown in the third column. The unlabelled instances nearest to the synthesized queries are shown in the last column.

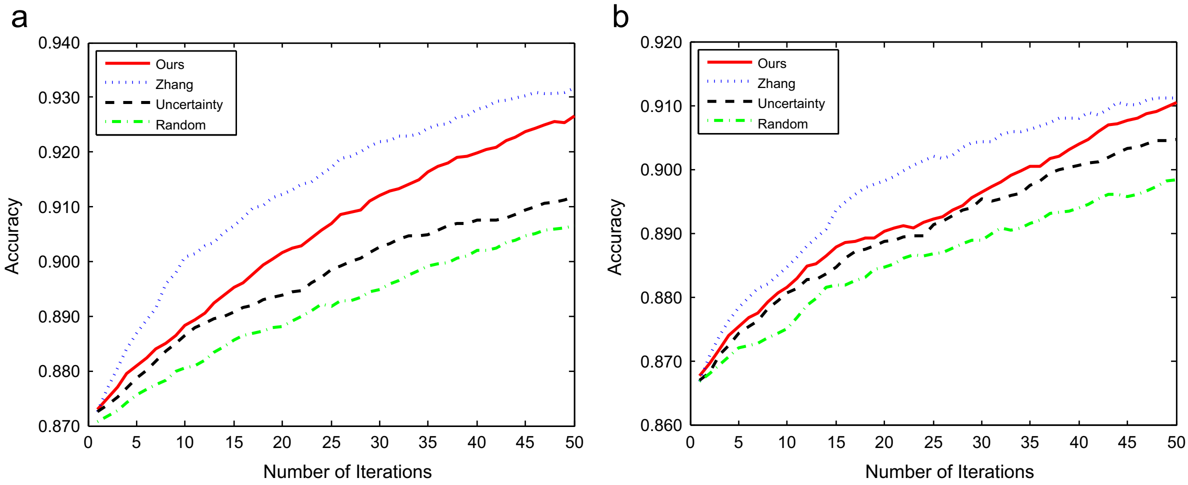


Fig. 8. Plots of mean accuracy with respect to the number of iterations on MNIST data sets. (a) 3 vs 5, (b) 7 vs 9.

images in the training set and 10 000 images in the test set (images with size of 28 × 28). Some instances are shown in Fig. 6.

We tested our framework with Nearest Neighbour learner on ‘3’ vs ‘5’, ‘7’ vs ‘9’. In these two groups of experiments, we randomly chose 500 examples from each class to create the experimental data set. PCA was employed to reduce the dimension from 784 to 122. The partition of the data set was exactly the same as described above.

An illustration of a single iteration of our approach is given in Fig. 7(a). Data points are plotted in the subspace specified by the first two principal components. Some *Opposite Pairs* and their synthesized queries according to our approach are shown in Fig. 7 (b). Similar to the work by Lang and Baum, the synthesized instances are tricky to understand. However, instead of using the synthesized instances directly, we query their nearest neighbours, which are normal instances and easily recognizable.

The comparison of accuracy is shown in Fig. 8. Time consumption is shown in Table 4. In the classification of ‘3’ vs ‘5’ and ‘7’ vs ‘9’, the accuracy of our method outperforms the uncertainty sampling, while the time cost is much smaller. Method in [4] yields best accuracy, but the time consumption is tremendous.

Table 4
Time consumption on MNIST database ($\times 10^{-4}$ s).

Datasets	Ours	Zhang	Uncertainty	Random
3 vs 5	7+25	168 680	1494	2
7 vs 9	6+22	145 649	1393	2

Experimental results show that our algorithms are not only practical but also competitive compared to the existed pool-based uncertainty sampling methods, which have been widely used in practice.

5. Conclusion

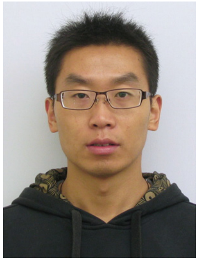
We proposed a novel framework of active learning that combines query synthesis and pool-based sampling. The basic idea is to synthesize an instance on the classification boundary according to the current labelled data in an efficient way, and then select the real instance nearest to the synthesized query from a compact representative unlabelled data pool as the query point. The proposed approaches not only enjoy the efficiency of query synthesis but also make the queries reasonable and meaningful for human oracles to label. Furthermore, we extended this novel technique into feature space, to better cope with non-linearly separable data and be compatible with more powerful kernel learners such as SVM. Experimental results on several real-world data sets indicate that the proposed method outperforms the random sampling in terms of accuracy significantly and has distinct speed advantage and similar performance compared to pool-based uncertainty sampling methods.

Acknowledgement

The work in this paper was supported in part by National Natural Science Foundation of China (NSFC) (Nos. 61233011, 91220301), Jiangsu Natural Science Foundation (No. BK20131351), Ministry of Human Resources of China via the Science and Technology Activities Grant for Returned Scholars, the Jiangsu Key Laboratory of Image and Video Understanding for Social Safety (Nanjing University of Science and Technology) (Nos. JSKL201304, 30920140122007), the Programme of Introducing Talents of Discipline to Universities (No. B13022), the Fundamental Research Funds for the Central Universities (Nos. 30920130122004, 30920130122005, 30920130121006) and the China Scholarship Council (No. 201206840034).

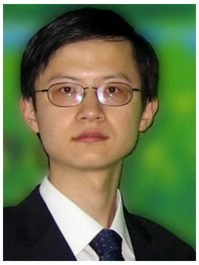
References

[1] B. Settles, Active learning literature survey, Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2010.
[2] B. Settles, Active Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, San Rafael, CA, USA, 2012.
[3] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, in: ICML, 2000, pp. 999–1006.
[4] L. Zhang, C. Chen, J. Bu, D. Cai, X. He, T.S. Huang, Active learning based on locally linear reconstruction, IEEE Trans. Pattern Anal. Mach. Intell. 33 (10) (2011) 2026–2038.
[5] D. Angluin, Queries and concept learning, Mach. Learn. 2 (4) (1988) 319–342.
[6] E.B. Baum, Neural net algorithms that learn in polynomial time from examples and queries, Trans. Neural Netw. 2 (1) (1991) 5–19.
[7] R.D. King, J. Rowland, S.G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L.N. Soldatova, A. Sparkes, K.E. Whelan, A. Clare, The automation of science, Science 324 (2009) 85–89.
[8] L.E. Atlas, D.A. Cohn, R.E. Ladner, Training connectionist networks with queries and selective sampling, in: NIPS, 1989, pp. 566–573.
[9] S. Dasgupta, D. Hsu, C. Monteleoni, A general agnostic active learning algorithm, in: ISAIM, 2008.
[10] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Springer-Verlag New York, Inc., New York, NY, USA, 1994, pp. 3–12.
[11] B. Settles, M. Craven, An analysis of active learning strategies for sequence labeling tasks, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2008, pp. 1070–1079.
[12] S.-J. Huang, R. Jin, Z.-H. Zhou, Active learning by querying informative and representative examples, in: NIPS, 2010, pp. 892–900.
[13] P. Melville, R.J. Mooney, Diverse ensembles for active learning, in: Proceedings of the 21st International Conference on Machine Learning, ACM, New York, NY, USA, 2004, pp. 74–81.
[14] K. Lang, E. Baum, Query learning can work poorly when a human oracle is used, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 1992, pp. 335–340.
[15] X. Hu, L. Wang, B. Yuan, Querying representative points from a pool based on synthesized queries, in: Proceedings of the International Joint Conference on Neural Networks, 2012, pp. 1734–1739.
[16] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the 13th Annual ACM Symposium on Theory of Computing, 1998, pp. 604–613.
[17] B. Schölkopf, A.J. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[18] O. Dekel, C. Gentile, K. Sridharan, Selective sampling and active learning from single and multiple teachers, J. Mach. Learn. Res. 13 (9) (2012) 2655–2697.
[19] K. Yu, L. Ji, X. Zhang, Kernel nearest-neighbor algorithm, Neural Process. Lett. 15 (2) (2002) 147–156.
[20] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.
[21] D. Angluin, Queries revisited, in: Proceedings of the International Conference on Algorithmic Learning Theory, 2001, pp. 12–31.
[22] R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, S. Oliver, Functional genomic hypothesis generation and experimentation by a robot scientist, Nature 427 (5) (2004) 247–252.
[23] S.C.H. Hoi, R. Jin, M.R. Lyu, Batch mode active learning with applications to text categorization and image retrieval, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1233–1247.
[24] X.-Y. Wei, Z.-Q. Yang, Coached active learning for interactive video search, in: Proceedings of the ACM International Conference on Multimedia, 2011, pp. 443–452.
[25] X. Li, Y. Guo, Adaptive active learning for image classification, in: Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1–8.
[26] S. Jones, L. Shao, K. Du, Active learning for human action retrieval using query pool selection, Neurocomputing 124 (2014) 89–96.
[27] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, J. Mach. Learn. Res. 3 (2001) 45–66.
[28] M. Lindenbaum, S. Markovitch, D. Rusakov, Selective sampling for nearest neighbor classifiers, Mach. Learn. 54 (2) (2004) 125–152.
[29] H.S. Seung, M. Opper, H. Sompolinsky, Query by committee, in: Proceedings of the ACM Workshop on Computational Learning Theory, 1992, pp. 287–294.
[30] N. Abe, H. Mamitsuka, Query learning strategies using boosting and bagging, in: Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 1–9.
[31] N. Roy, A. Mccallum, Toward optimal active learning through sampling estimation of error reduction, in: Proceedings of the 18th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, USA, 2001, pp. 441–448.
[32] Y. Guo, R. Greiner, Optimistic active learning using mutual information, in: International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 823–829.
[33] H.T. Nguyen, A. Smeulders, Active learning using pre-clustering, in: Proceedings of the 21st International Conference on Machine Learning, ACM Press, New York, NY, USA, 2004, pp. 623–630.
[34] S. Dasgupta, D. Hsu, Hierarchical sampling for active learning, in: Proceedings of the 25th International Conference on Machine Learning, ACM, New York, NY, USA, 2008, pp. 208–215.
[35] S.C. Hoi, R. Jin, J. Zhu, M.R. Lyu, Semi-supervised svm batch mode active learning for image retrieval, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008, IEEE, Piscataway, NJ, United States, 2008, pp. 1–7.
[36] A. Frank, A. Asuncion, UCI machine learning repository, 2010. URL ⟨http://archive.ics.uci.edu/ml⟩.
[37] Y. LeCun, et al., Comparison of learning algorithms for handwritten digit recognition, in: Proceedings of the International Conference on Artificial Neural Networks, 1995, pp. 53–60.
[38] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 1–27.



Liantao Wang received his B.S. degree in Mechanical Engineering from Nanjing University of Science and Technology, China, in 2004. He is currently working toward the Ph.D. degree in Pattern recognition and intelligent system at Nanjing University of Science and Technology. His research interests include machine learning and computer vision.

Xuelei Hu is an Associate Professor in the School of Computer Science and Engineering at Nanjing University of Science and Technology where she has been a faculty member since 2005 and a Postdoctoral Research Fellow in the School of Information Technology and Electrical Engineering at The University of Queensland. Xuelei Hu received her Ph.D. in Computer Science and Engineering from the Chinese University of Hong Kong, M.Phil. in Pattern Recognition and Intelligence System and B.E. in Computer Science and Engineering from Nanjing University of Science and Technology. From February 2009 to February 2010 she worked as a Visiting Post-doctoral Fellow in the Robotics Institute at Carnegie Mellon University. Her research interests lie in the area of machine learning, ranging from theory to application, with a recent focus on developing novel machine learning and data mining methods for real-world problems.



Dr. Bo Yuan is mostly interested in Data Mining, Evolutionary Computation and Parallel Computing. He received the B.E. degree from Nanjing University of Science and Technology, P.R. China, in 1998, and the M.Sc. and Ph.D. degrees from The University of Queensland, Australia, in 2002 and 2006, respectively. From 2006 to 2007, he was a Research Officer on a project funded by the Australian Research Council at The University of Queensland. He is currently an Associate Professor in the Division of Informatics, Graduate School at Shenzhen, Tsinghua University, P.R. China, and a member of the IEEE and the IEEE Computational Intelligence Society.

Jianfeng Lu received his B.S. degree in Computer Software, M.S. degree and Ph.D. degree in Pattern Recognition and Intelligent System all from Nanjing University of Science and Technology (NUST) respectively. He is currently a Professor in the same school. His research interests include image processing, pattern recognition and data mining.