Querying Representative Points from a Pool based on Synthesized Queries

Xuelei Hu, Liantao Wang
School of Computer Science and Technology
Nanjing University of Science and Technology
Nanjing 210094, P.R. China
Email: xlhu@njust.edu.cn, ltwang.nust@gmail.com

Bo Yuan
Intelligent Computing Lab
Division of Informatics, Graduate School at Shenzhen
Tsinghua University, Shenzhen 518055, P.R. China
Email: yuanb@sz.tsinghua.edu.cn

***Abstract*—How to build a compact and informative training data set autonomously is crucial for many real-world learning tasks, especially those with large amount of unlabeled data and high cost of labeling. Active learning aims to address this problem by asking queries in a smart way. Two main scenarios of querying considered in the literature are query synthesis and pool-based sampling. Since in many cases synthesized queries are meaningless or difficult for human to label, more efforts have been devoted to pool-based sampling in recent years. However, in pool-based active learning, querying requires evaluating every unlabeled data point in the pool, which is usually very time-consuming. By contrast, query synthesis has clear advantage on querying time, which is independent of the pool size. In this paper, we propose a novel framework combining query synthesis and pool-based sampling to accelerate the learning process and overcome the current limitation of query synthesis. The basic idea is to select the data point nearest to the synthesized query as the query point. We also provide two simple strategies for synthesizing informative queries. Moreover, to further speed up querying, we employ clustering techniques on the whole data set to construct a representative unlabeled data pool based on cluster centers. Experiments on several real-world data sets show that our methods have distinct advantages in time complexity and similar performance compared to pool-based uncertainty sampling methods.**

I. Introduction

Data are being generated at unprecedented speed worldwide. For many large scale real problems, due to the high cost of data labeling, intelligent systems need not only to learn from the given data set but also to autonomously create a compact and informative training data set. One of the solutions is called active learning, which aims to achieve low generalization error using as few labeled instances as possible by intelligently querying the labels of certain unknown instances. In recent years, active learning has gained increasing interests and demonstrated its effectiveness in various applications [1]. There are two main scenarios of querying: query synthesis and pool-based sampling.

In the scenario of query synthesis, a learning system may generate a query in the form of any unlabeled instance in the input space. Early work was carried out by using membership queries [2], which was further studied in [3]. The idea of query synthesis has also been extended to regression tasks [4]. For finite domain problems, synthesizing query may be an efficient way but labeling queries synthesized by learning systems may be difficult or impossible for human experts. For example, query images generated by learning systems for image classification and query sentences created by learning systems for language processing may have no natural meaning and thus may be unrecognizable by human oracles. An unsuccessful example of using query synthesis shows that some of the queries posed by the learning algorithm were too difficult for a person to answer reliably in a handwritten character classification task [5].

Due to this limitation of query synthesis, most efforts have focused on pool-based sampling in recent active learning literature. In the scenario of pool-based sampling, learning systems select query points from a pool consisting of unlabeled data points according to some criterion. Typical pool-based approaches include uncertainty sampling [6], query by committee [7] and active learning for support vector machines [8]. Recently, there have been a variety of pool-based active learning methods proposed in the literature, such as: multiple-instance active learning [9], importance weighted active learning [10], and active learning from crowds [11]. Pool-based active learning methods have been widely used in many real world applications (e.g., text categorization [12], image retrieval [13], and video search [14]). However, pool-based sampling requires evaluating every unlabeled data points (e.g., calculating prediction posterior and computing expected error reduction). The computational cost of querying might be very high when the pool size is large.

To make querying fast and practical, we propose a new framework of active learning that combines query synthesis and pool-based sampling. As illustrated in Fig.1, the basic idea is to select the data point nearest to the synthesized query from a compact representative unlabeled data pool as the query point. To take the speed advantage of query synthesis, we introduce two heuristic strategies for synthesizing queries: i) using the midpoint of two class centers of labeled points as the query; ii) using the midpoints of nearest neighbor pairs that have different class labels as queries. The underlying

This work was supported in part by the National Natural Science Foundation of China (NSFC) (No. 60705020, No. 90820306, No. 60905030), Ministry of Human Resources of China via the Science and Technology Activities Grant for Returned Scholars, and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

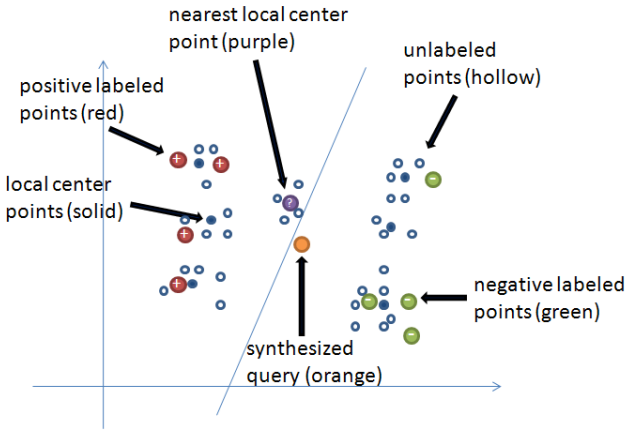


Fig. 1. Illustration of the idea of the proposed framework. The hollow points indicate unlabeled data points and the solid points indicate unlabeled data points located at local high-density areas, also called local center points, which construct a compact unlabeled data pool. The red and green points indicate the data points with positive and negative labels respectively. At each iteration of active learning, the system synthesizes a query (orange point) according to some strategy based on labeled data points and then selects the unlabeled local center point (purple point) nearest to the synthesized query to query.

idea of both strategies is that the midpoint of those points with different labels may have the largest uncertainty and consequently querying it may gain the most information.

However, as mentioned above, the synthesized queries may be difficult or meaningless for a human oracle to label. To address this limitation, we select the nearest neighbor point of the synthesized query from an unlabeled data pool as the final query posted to oracles. Nearest neighbor search algorithms and approximate nearest neighbor search algorithms with sub-linear computational complexity have been well studied over years [15].

Moreover, to query representative points and speedup querying, we exploit clustering techniques on the whole data set as preprocess of active learning to construct a compact and representative unlabeled data pool based on local center points. The effectiveness of using clustering algorithms in active learning has been demonstrated in [16].

The rest of this paper is organized as follows: Section II presents the proposed approach in details; experimental results are reported in Section III; Section IV concludes this work.

II. Method

In this section, we present our framework of active learning that combines query synthesis and pool-based sampling, and strategies to synthesize queries in details.

A. The Framework

Suppose that we have a data set $S = \{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_N\}$ consisting of N observations of a D-dimension random variable x. The initial labeled data and unlabeled data are denoted by L and U respectively. Obviously $L \cup U = S$ and $L \cap U = \emptyset$.

First, we use clustering techniques (e.g., k-means) to process the original data set. In this process, S is partitioned into $K (K < N)$ clusters $\{S_1, S_2, ..., S_K\}$, where $S_i \cap S_j = \emptyset$ for any $i \neq j$ and $S = \uplus_{i=1}^{K} S_i$. Each cluster consists of a group of data points that are similar to each other (low intra-cluster distances and high inter-cluster distances). For cluster $S_i (1 \leq i \leq K)$, its centroid (cluster mean vector) c_i can be calculated by

$$\mathrm{c}_i = \frac{1}{|S_i|} \sum_{\mathrm{x}_j \in S_i} \mathrm{x}_j. \tag{1}$$

Then, the data point x_{ci} nearest to c_i in U can be found by nearest neighbor search strategies. We therefore obtain a set U_c comprising unlabeled data points nearest to centroids

$$U_c = \{\mathrm{x} | \mathrm{x} = nnSearch(\mathrm{c}_i, U), 1 \leq i \leq K\}. \tag{2}$$

Note that $|U_c| = K$ and $U_c \subset U$. With clustering techniques, a set of clusters $\{S_1, S_2, ..., S_K\}$ and their best representation U_c can be obtained to represent the general distribution of S. If the label of centroid c_i is given, it will provide important information for the labels of data points in S_i. We therefore generate queries from U_c in our approach.

After constructing a compact and representative unlabeled data pool U_c by pre-clustering, the system initiates a circle between the oracle and the learner to obtain a hypothesis h. At each iteration, the system synthesizes an example $sExample$ according to some strategy, which will be introduced in the next subsection, and then searches its nearest neighbor $qExample$ in the pool U_c and queries $qExample$ to the oracle. After obtaining the label of the query, the system updates the labeled set and unlabeled pool and continues to the next iteration until some stop criterion is satisfied. Algorithm 1 shows the complete procedure.

The proposed approach can overcome the issue of meaningless queries by using their nearest neighbors instead. As to computational complexity, since query synthesis only works on a small set of labeled data, most computational cost of

Algorithm 1 The proposed framework of active learning.

Require:
L - Labeled data set, $L^1 = L$; U - Unlabeled data set;
K - Number of clusters; n - Number of iterations
Ensure:
hypothesis h
$\mathrm{c}_i \leftarrow kmeans(L \cup U), i = 1, ..., K$
$U_c \leftarrow nnSearch(\mathrm{c}_i, U), i = 1, ..., K$
for $i = 1$ to n **do**
  $h = learn(L^i)$
  $sExample = synthesize(strategy, L^i)$
  $qExample = nnSearch(sExample, U_c^i)$
  Query $qExample$
  Label $qExample$
  $L^{i+1} = L^i + qExample$, $U_c^{i+1} = U_c^i - qExample$
end for
return h

querying is on the nearest neighbor search. The computational complexity of the linear search is $O(KD)$, where K is the size of the pool and D is the dimensionality. For large scale data sets, fast approximate nearest neighbor search algorithms can be used. Since the size of the compact pool U_c is smaller than the original data set, the proposed approach is computationally efficient and practical for large scale problems.

B. Strategies of Query Synthesis

We introduce two simple and practical strategies of query synthesis in this subsection. The first one is generating the midpoint of two class centers of labeled points as queries, named centroid generation. The second one is generating the midpoints of nearest neighbor pairs that have different class labels as queries, named neighbor generation.

1) Centroid Generation: Suppose L_+ represents the positive examples in L, and L_- represents the negative examples in L. The centroid of the positive data set is given by

$$L_+ = \{\mathrm{x} | \mathrm{x} \in L, label(\mathrm{x}) > 0\}, \tag{3}$$

$$\mathrm{c}_+ = \frac{1}{|L_+|} \sum_{\mathrm{x}_j \in L_+} \mathrm{x}_j. \tag{4}$$

The centroid of the negative data can be calculated similarly. We then synthesize a new example with the midpoint of c_+ and c_- by

$$sExample = \frac{1}{2}(\mathrm{c}_+ + \mathrm{c}_-). \tag{5}$$

Algorithm 2 specifies the implementation of this strategy. Although this strategy is very simple and easy to implement, it has a potential disadvantage that synthesized queries may be always close to the centroid of the entire labeled data points and the learner may be under the risk of ignoring valuable discriminant information far from the centroid.

2) Neighbor Generation: As to neighbor generation, we first execute the nearest-neighbor search on L, so that a pair of data points is obtained for each data point in L. From these $|L|$ pairs, those having opposite labels are selected to create the set of opposite-neighbor-pairs (O-N-Pairs). Let $\{\mathrm{x}_+, \mathrm{x}_-\}$ be an O-N-Pair, we can synthesize queries by

$$sExample = \frac{1}{2}(\mathrm{x}_+ + \mathrm{x}_-). \tag{6}$$

Algorithm 2 Centroid generation strategy $sExample = synthesize(centroid, L)$.

Require:
L - Labeled data set
Ensure:
$sExample$
1: Obtain c_+ and c_-;
2: $sExample = \frac{1}{2}(\mathrm{c}_+ + \mathrm{c}_-)$
3: **return** $sExample$

We can see, from Algorithm 3, that the time complexity of neighbor generation depends on the size of labeled set. As the size of the labeled set grows, the time cost of O-N-Pairs search will become expensive. In order to overcome this disadvantage, we present an approximate O-N-Pairs Search, which is shown in Algorithm 4. The time complexity of this approximate algorithm depends on the size of the neighborhood of the $(i-1)th$ query, which is a constant denoted by C. Therefore, the time complexity of neighbor generation decreases to $O(C)$.

Algorithm 3 Neighbor generation strategy $sExample = synthesize(neighbor, L)$.

Require:
L - Labeled data set;
$pair\_queue$ is a queue structure storing O-N-Pairs
Ensure:
$sExample$
1: **for** $j = 1$ to $|L|$ **do**
2: $\mathrm{x}_j^n = nnSearch(\mathrm{x}_j, L)$
3: **if** the label of x_j^n is opposite to that of x_j **then**
4: $pushTail(pair\_queue, (\mathrm{x}_j, \mathrm{x}_j^n))$
5: **end if**
6: **end for**
7: **if** $pair\_queue$ is not empty **then**
8: $aPair = popFront(pair\_queue)$
9: $sExample = mid(aPair)$
10: **else**
11: $sExample = synthesize(centroid, L)$
12: **end if**
13: **return** $sExample$

Algorithm 4 Approximate neighbor generation strategy $sExample = synthesize(approxNeighb, L)$.

Require:
L - Labeled data set;
L_i - Neighborhood of the $(i-1)th$ query; $(L_1 = L)$;
$pair\_queue$ is a queue structure storing O-N-Pairs
Ensure:
$sExample$
1: **for** $j = 1$ to $|L_i|$ **do**
2: $\mathrm{x}_j^n = nnSearch(\mathrm{x}_j, L)$
3: **if** x_j^n is opposite to x_j **then**
4: $pushTail(pair\_queue, (\mathrm{x}_j, \mathrm{x}_j^n))$
5: **end if**
6: **end for**
7: **if** $pair\_queue$ is not empty **then**
8: $aPair = popFront(pair\_queue)$
9: $sExample = mid(aPair)$
10: **else**
11: $sExample = synthesize(centroid, L)$
12: **end if**
13: **return** $sExample$

III. EXPERIMENTS

In this section, we present the experimental results to demonstrate the performance of the proposed active learning framework compared with random sampling and pool-based uncertainty sampling on various data sets from UCI machine learning repository [17] as well as MNIST database of handwritten digits [18]. For the sake of convenience, active learning framework with centroid generation and neighbor generation are denoted by C-C and C-N respectively.

We used nearest neighbor classifiers as the common learner for all algorithms and evaluated their performance against the size of labeled set through iterations. In each round of iteration, every algorithm selected one example from the unlabeled set using different strategies. The random sampling method selected data points randomly to ask for label. In uncertainty sampling, five nearest labeled neighbors of each unlabeled data point constructed the voting committee. The uncertainty of each unlabeled point was measured by the vote entropy of the 5 committee members, and the most uncertain example was selected to ask for label.

In the experiments, each data set was randomly partitioned into the initial labeled data set L, the test set T and the unlabeled data pool U, which accounted for 10%, 25% and 65% of the total size N_{total} respectively. This partition process was repeated 200 times to reduce randomness. All the results shown in this section were the average values over 200 repeated experiments. For C-C and C-N, the whole data set was clustered into $\lfloor N_{total}/10 \rfloor$ groups using the k-means method. As mentioned in Section II, examples closest to centroids in U constituted the learning pool U_c for C-C and C-N. A PC with Intel Core 2 Duo CPU (2.93GHz) and 2GB RAM with Windows XP and Matlab 2010b was used as the experimental platform.

A. Experiments on UCI Data Sets

We first investigated the performance of random sampling, uncertainty sampling, and the proposed approaches on 9 UCI data sets, which are described briefly in Table I.

Fig.2 shows the curves of the average classification accuracies of each method on various UCI data sets. As we can see from the plots, our methods were superior to random sampling (their learning curves dominated those of random sampling) and had similar performance as uncertainty sampling. The average CPU time costs of each round of iteration of different methods on 9 UCI data sets are shown in Table II. It is clear that our methods were superior to uncertainty sampling in terms of computational cost.

B. Handwritten Digit Recognition

Baum's query learning failed in a handwritten digit recognition task because the human oracle had difficulty understanding the synthesized image [5]. To address this problem, our algorithms query the real example closest to the synthesized example. In order to demonstrate the competence of our algorithms in vision-based recognition tasks, we conducted handwritten digit recognition experiments on MNIST database. MNIST database is a popular handwritten digit data set containing 60000 images in the training set and 10000 images in the test set (images of size 28×28). Some instances are shown in Fig.3(a).

We conducted binary classification experiments on '3' vs. '5', '6' vs. '8' and '7' vs. '9'. In these three 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 C-C is given in Fig.3(b). Data points are plotted in the subspace specified by the first two principal components. Some O-N-Pairs and their synthesized queries according to our approach C-N are shown in Fig.3(c). Similar to the work by Baum, the synthesized examples were a bit tricky to understand. However, instead of using the synthesized examples directly, we queried their nearest neighbors, which were normal examples and easily recognizable.

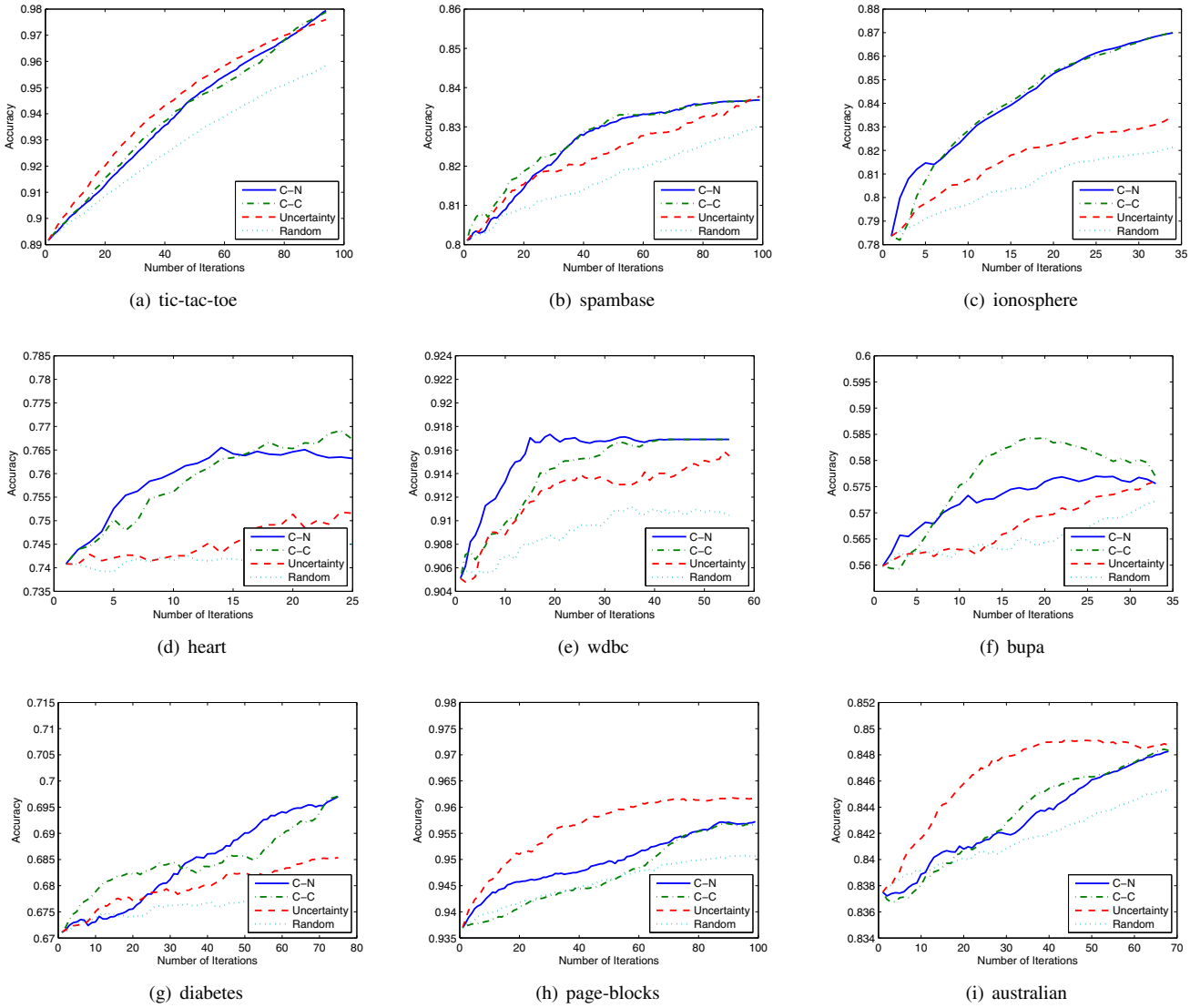
The comparison of accuracy is shown in Fig.4. Time consumption is shown in Table III. In the classification tasks of '3' vs. '5' and '7' vs. '9', the accuracies of C-C and C-N both outperformed uncertainty sampling, while the time cost was much smaller. Experimental results show that our algorithms are not only practical but also competitive against pool-based uncertainty sampling methods, which have been widely used in practice.

TABLE I
DESCRIPTION OF DATA SETS

Name	#cls	#atts	N_{total}	L	U	T
tic-tac-toe	2	9	958	95	623	240
spambase	2	57	4601	460	2991	1150
ionosphere	2	34	351	35	228	88
heart	2	13	270	27	175	68
wdbc	2	31	569	57	370	142
bupa	2	6	345	35	224	86
diabetes	2	8	768	77	499	192
page-blocks	5	10	5242(5473)	524	3408	1310
australian	2	14	690	69	449	172

TABLE II
TIME CONSUMPTION ON UCI DATA SETS (IN SECONDS)

Algorithm	C-N	C-C	Uncertainty	Random
tic-tac-toe	2.8864e-004	1.0187e-004	0.0245	5.2742e-005
spambase	0.0054	0.0040	1.8608	0.0011
ionosphere	2.9236e-004	1.0014e-004	0.0079	5.8227e-005
heart	2.6325e-004	9.0542e-005	0.0047	3.4428e-005
wdbc	2.8509e-004	1.0520e-004	0.0156	1.0276e-004
bupa	3.8028e-004	8.7021e-005	0.0059	2.8058e-005
diabetes	3.8235e-004	9.4360e-005	0.0166	4.2761e-005
page-blocks	0.0022	9.2707e-004	0.8909	4.3953e-004
australian	3.2126e-004	1.0506e-004	0.0177	5.8930e-005

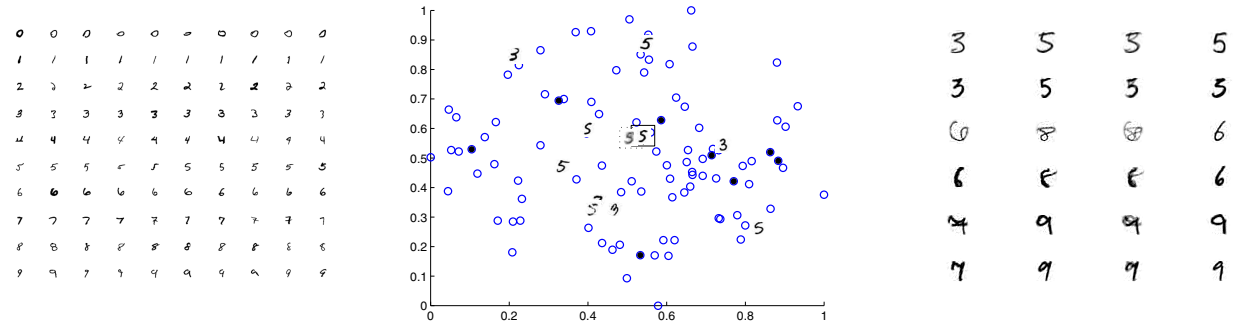


(a) tic-tac-toe (b) spambase (c) ionosphere

(d) heart (e) wdbc (f) bupa

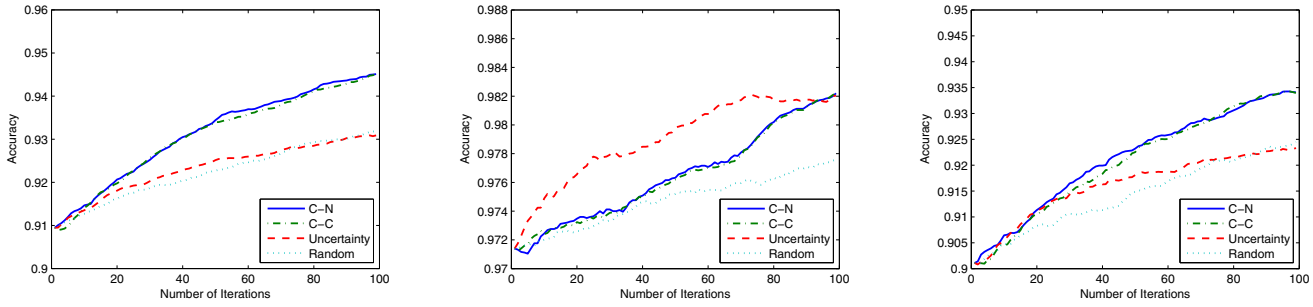
(g) diabetes (h) page-blocks (i) australian

Fig. 2. Plots of mean accuracies against the number of iterations on UCI data sets.



(a) Examples of MNIST data (b) Illustration of one iteration of C-C. (c) Examples of queries by C-N.

Fig. 3. Illustration of experiments on MNIST data. (a) Examples of original data. (b) Illustration of a single iteration in the proposed method C-C. The hollow points indicate unlabeled data points and the solid points indicate local center points. The digit images without frame indicate the labeled data points. The digit images with dotted frame indicate the synthesized query using Centroid generation strategy. The digit images with solid frame indicate the nearest neighbor of the synthesized query among local center points. (c) Examples of queries by the proposed method C-N. The first two columns are the O-N-Pairs found in the experimental process. Their midpoints (the synthesized queries) are shown in the third column. The neighbors nearest to the synthesized queries are shown in the last column.



(a) 3 and 5 (b) 6 and 8 (c) 7 and 9

Fig. 4. Plots of mean accuracies against the number of iterations on MNIST data sets.

TABLE III
TIME CONSUMPTION ON MNIST DATABASE (IN SECONDS)

Algorithm	C-N	C-C	Uncertainty	Random
3 vs 5	9.1245e-004	8.5285e-004	0.0854	2.6743e-004
6 vs 8	9.6648e-004	9.1324e-004	0.0887	2.8376e-004
7 vs 9	7.8783e-004	7.5162e-004	0.0797	2.4253e-004

IV. CONCLUSION

We proposed a new framework of active learning that combines query synthesis and pool-based sampling. The basic idea is to select the data point nearest to the synthesized query from a compact representative unlabeled data pool as the query point. The proposed approaches not only enjoy the speed advantage of query synthesis but also make the queries reasonable and meaningful for human oracles to label. Furthermore, we provided two efficient strategies for synthesizing queries and employed clustering techniques to construct a compact representative unlabeled data pool. Experimental results on several realistic data sets indicate that the proposed methods outperformed the random sampling method and had distinct speed advantage and similar performance compared to pool-based uncertainty sampling methods.

REFERENCES

[1] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010.

[2] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, pp. 319–342, 1988.

[3] ——, "Queries revisited," in *Proceedings of International Conference on Algorithmic Learning Theory*, 2001, pp. 12–31.

[4] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[5] K. Lang and E. Baum, "Query learning can work poorly when a human oracle is used," in *Proceedings of IEEE International Joint Conference on Neural Networks*, 1992, pp. 335–340.

[6] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers: Corrigendum and additional data," in *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 3–12.

[7] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of ACM Workshop on Computational Learning Theory*, 1992, pp. 287–294.

[8] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 3, pp. 45–66, 2002.

[9] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Advances in Neural Information Processing Systems*. MIT Press, 2008, pp. 1289–1296.

[10] A. Beygelzimer, S. Dasgupta, and J. Langford, "Importance weighted active learning," in *Proceedings of the International Conference on Machine Learning*, 2009, pp. 49–56.

[11] Y. Yan, R. Rosales, G. Fung, and J. Dy, "Active learning from crowds," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 1161–1168.

[12] S. C. H. Hoi, R. Jin, and M. R. Lyu, "Batch mode active learning with applications to text categorization and image retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1233–1247, 2009.

[13] D. Zhang, F. Wang, Z. Shi, and C. Zhang, "Interactive localized content based image retrieval with multiple-instance active learning," *Pattern Recognition*, vol. 43, pp. 478–484, 2010.

[14] X.-Y. Wei and Z.-Q. Yang, "Coached active learning for interactive video search," in *Proceedings of the ACM International Conference on Multimedia*, 2011, pp. 443–452.

[15] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.

[16] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the 21st International Conference on Machine Learning*. ACM Press, 2004, pp. 623–630.

[17] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[18] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of learning algorithms for handwritten digit recognition," in *Proceedings of International Conference on Artificial Neural Networks*, 1995, pp. 53–60.