Big Active Learning

Er-Chen Huang*, Hsing-Kuo Pao*, Yuh-Jye Lee†
* Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan
Emails: huang455069@yahoo.com.tw, pao@mail.ntust.edu.tw
† Department of Applied Mathematics
National Chiao Tung University, Hsinchu, Taiwan
Email: yuhjye@math.nctu.edu.tw

***Abstract*—Active learning is a common strategy to deal with large-scale data with limited labeling effort. In each iteration of active learning, a query is ready for oracle to answer such as what the label is for a given unlabeled data. Given the method, we can request the labels only for those data that are essential and save the labeling effort from oracle. We focus on pool-based active learning where a set of unlabeled data is selected for querying in each run of active learning. To apply pool-based active learning to massive high-dimensional data, especially when the unlabeled data set is much larger than the labeled set, we propose the APRAL and MLP strategies so that the computation for active learning can be dramatically reduced while keeping the model power more or less the same. In APRAL, we avoid unnecessary data re-ranking given an unlabeled data selection criteria. To further improve the efficiency, with MLP, we organize the unlabeled data in a multi-layer pool based on a dimensionality reduction technique and the most valuable data to know their label information are more likely to store in the top layers. Given the APRAL and MLP strategies, the active learning computation time is reduced by about 83% if compared to the traditional active learning ones; at the same time, the model effectiveness remains.**

***Index Terms*—active learning; high dimensionality; large-scale data; pool-based sampling.**

I. Introduction

A. Motivation

Given massive data for analysis, active learning (AL) is one of the promising solutions where we use as small as possible set of labeled information to save human effort for effective modeling. In active learning, queries that follow a data selecting strategy pop up constantly for *oracle* or human annotator to answer. In each iteration of active learning, the typical query is to label a few unlabeled data that could be hard to decide their label information by premature models. A common set of strategies for unlabeled data selection includes using the criteria of *uncertainty*, *diversity*, or *representativeness* [1] to name a few to decide which data may be crucial where we are eager to know their label information for effective modeling. One issue that could be overlooked by most active learning scenarios is to ignore the unbalanced ratio between labeled and unlabeled data for learning. More specifically, we may have a relatively small set of labeled data for training and a huge set of unlabeled data for *ranking* according to their properties such as the aforementioned criteria in each active learning iteration. That creates incompatible time consumption on the (labeled) data training and the (unlabeled) data ranking even the model training usually owns higher computation complexity than data ranking. In this work, we propose a novel approach that can speed up the data ranking procedure based on approximate criteria computation, hierarchical data arrangement and flexible ranking decision. The experiment results show that the overall active learning computation can be deducted by about 83% using the proposed active learning approach if compared to the traditional active learning ones. At the same time, the model effectiveness more or less remains.

We propose the Anti Re-rank Pool-based Active Learning (ARPAL) and Multi-Layer Pool (MLP) for efficient active learning given massive high-dimensional data. The focused active learning scenario is a so-called *pool-based* active learning where we have a set of unlabeled candidate data available for querying from oracle in each run of active learning. A close analysis reveals that how much the computation we need for active learning is (at least) decided by:

1) the time for model training given the labeled data;
2) the time to compute the ranks of unlabeled data given pre-defined data selection criteria; and
3) the time to rank the unlabeled set according to the selection criteria; and

In our opinion, the time for model training can often be negligible because the size of labeled data is much smaller than the size of unlabeled data in most big-data applications. To speed up the active learning computation, we then focus on the second and the third parts. A naïve thought is to compute the data ranks and operate the sorting according to the ranks when it is really necessary to do so. Also, we can obtain the ranks based on an approximate computation when the exact rank computation from a premature model is not meaningful. It may even shorten the time for active learning. In an extreme case, we can select the next unlabeled data to label based on the purely random strategy. Therefore, the time we need to spend on the second and the third parts is zero. Apparently, the random strategy is far from a good choice because we may need a much larger set of data for training to reach an equally effective model. To work on the trade-off, making the model more effective and trying not to spend too much time on data re-ranking simultaneously is the central goal of the work. In the next subsection, we elaborate the details of the proposed method.

B. The Proposed Approach

In order to overcome the shortcomings of the traditional pool-based active learning, we propose a few strategies to speed up the active learning computation given massive high-dimensional data. In the proposed ARPAL approach, we first investigate how necessary for each data re-ranking in the active learning iterations. Intuitively speaking, when the models between two consecutive iterations do not change much from each other, we may think that the rankings on the unlabeled data based on the two models differ very little. Therefore, we do not compute the data ranks nor perform the re-ranking to save the computation. For high-dimensional data, the time that we can save on computing the data ranks is a lot. Because most of the computation is associated with dimensionality. We shall save more from a second strategy for data arrangement with multi-layers where dimensionality reduction is applied in particular for high-dimensional data. We have to point out that in the early stage of active learning where the model does not yet converge to a stable one, we do not trust much about the prediction of the model nor the criteria computation on e.g., uncertainty that is derived from the model. Although a frequent data re-ranking is expected in this period; however, we can adopt an approximate rank computation to improve the efficiency.

As the second MLP strategy, we transform the traditional pool in active learning to a multi-layer pool where different dimensionality reduction is applied in different layers for an approximate to exact data selection criteria computation from the bottom to the top layers. More specifically, we gradually select the unlabeled data for querying from the bottom layers to the top layers. In bottom layers where we have a relatively huge set, we compute data ranks approximately. For instance, we may project the data into a low-dimensional space to decide the data ranks in that space to save the time that we need to spend for high-dimensional data. As we go from the bottom layers to top layers, we compute data ranks based on a relatively higher and higher dimensional inputs until in the top layer, where we attempt to compute the exact data ranks given the full set of features, but on a small set of data. We expect that the most informative data for AL query can be found in the top layer set in most cases. We shall mainly use Principal Component Analysis (PCA) to find the low-dimensional structures for data. In summary, we trade the model effectiveness for an efficient result. As we test the proposed approach, we find out that the model effectiveness that we give up may not be much if a careful treatment is adopted.

We use the prediction accuracy and the execution time to evaluate the two proposed strategies. As a result, we observe that the accuracy from the traditional AL and the proposed ARPAL and MLP combined AL remain to be similar while the proposed method perform much more efficient than the traditional approach. In the rest of the paper, we first review the past work on active learning in Section II and introduce the proposed method for active learning given massive high-dimensional data in Section III, which is followed by the experiment results in Section IV. After that, we conclude the work in Section V.

II. Related Work

Various active learning methods have been proposed in the last decades to improve the model effectiveness with a reduced effort on data labeling. Especially, new approaches were developed in recent years to echo the big data trend for real applications. In traditional active learning, the learner wants to find out a set of unlabeled data which is the most important part to be included in training. In other words, if the model has the least confidence on labeling a particular set of data, it is preferred to ask oracle for the information and the model based on the newly acquired labeled data and previously labeled data can achieve better accuracy than the one without the newly acquired data. The aforementioned strategy is the so-called *uncertainty sampling*, first introduced by Lewis *et al.* [2]. Other criteria for unlabeled data selection include diversity and representativeness. Du *et al.* [3] proposed an integrated criterion which combines representativeness and informativeness to choose the most suitable instances to be included in modeling. Huang *et al.* [4] considered a set of similar information based on a min-max view of active learning. There is another interesting query strategy, called query-by-committee (QBC) [5], which uses many subsets of training data to build sub-models. After that, all of the sub-models predict the label of instances and vote to find the consensus. The set of instances that receive little consensus should be the set to ask for help from oracles.

To speak of incorporating learning models in active learning, Tong *et al.* [6] applied an uncertainty sampling algorithm in SVM training. They provided a theoretical motivation for the algorithm using the notion of version space. Kremer *et al.* [7] discussed the advantages of adopting SVMs in active learning. Joshi *et al.* [8] presented a margin uncertainty measure in multi-class cases. The uncertainty measure is based on the difference between the probability of two most probable classes where it is hard for the learner to determine the data's class. Fu *et al.* [9] surveyed existing works on active learning from an instance-selection perspective. Settles summarized the active learning methods in a well-known book [1].

Many active learning methods are applied to the real world. Demir *et al.* [10] and Tuia *et al.* [11] applied active learning in image recognition. Donmez *et al.* [12] introduced the proactive learning to relax unrealistic assumptions, and it relies on a decision-theoretic approach to jointly select the optimal oracles and instances for annotation. The work can be applied to multi-oracle applications such as using active learning in smart factories. To deal with some special active learning applications, Persello *et al.* [13] studied a cost-sensitive active learning where the cost of labeling an unlabeled instance depends on the past labeling effort. One possible application of the work is in a smart factory where labeling cost for a new instance may be associated with the past labeling efforts, if the labeling implies walking from one place to another.

Kapoor *et al.* [14] proposed a method that can balance the cost of misclassification and the cost of annotation in the active learning procedure.

Overall, most past active learning algorithms were focused on developing a good strategy for unlabeled data sampling or designing a good scenario for applying active learning in real cases. However, very few addressed the computation issue of active learning; more specifically, the computation that is needed for large-scale data or data with high dimensionality. That is the main motivation for this work.

III. Methodology

In this section, we introduce the background and scenarios of active learning that are related to this work. Following that, we discuss the issues that we need to pay attention to for active learning. That explains why we develop the Anti Re-rank Pool-based Active Learning (ARPAL) and Multi-Layer Pool (MLP) to improve the performance of active learning systems. ARPAL can avoid the unnecessary re-ranking of the unlabeled data in the pool and MLP can reduce the computation time for each re-ranking. The details of the two methods are discussed in the following two subsections. Overall, the active learning system that utilizes ARPAL and MLP can gain much improvement over the computation time while maintain the similar model prediction power.

A. The Background and Issues of Active Learning

1) Active Learning Scenarios: There are three traditional approaches of active learning [1]: (1) query synthesis, (2) stream-based selective sampling, and (3) pool-based sampling. In the query synthesis, the assumption is that the learner has a definition of the input space (also, the feature dimensions). The learner may query the oracle about the labels of *any* (real or synthesized) unlabeled data in the input space. Query synthesis has good results in some special applications. However, the learner generates arbitrary synthesis data which may be meaningless and irrelevant for an oracle to recognize and make judgment.

An alternative to synthesizing queries is selective sampling, also called stream-based active learning. The assumption is that obtaining an unlabeled data (may be from an unknown distribution) is cheap and continuous, and then the learner can decide whether or not to request its label. Some common strategy is to select a threshold and request the label information when the unlabeled data reaches the threshold passively. There are issues and drawbacks for the stream-based sampling approach. Some unlabeled data may not reveal its importance when it arrives at a particular moment, and can become valuable later when the data is no longer available. To solve the problem, we can keep a buffer for active learning to save unlabeled data in that buffer, so-called the *pool* and selection criteria can then be used to find the most valuable data from the pool to ask their label information. It is the idea of *pool-based* active learning.

Pool-based active learning, the most popular scenario for active learning, assumes that there is a small set of labeled data L and a large pool of unlabeled data U available. Initially, a learner is built from L, and it continuously selects a set of unlabeled data from the pool, according to a pre-defined selection criterion for querying and adds them into the training to find the next-run learners. The procedure goes on until we feel satisfied with the result. For many real-world situations, a large amount of unlabeled data can be easily collected at once and pool-based active learning is appropriate to apply. In this work, we assume $|L| \ll |U|$, the size of labeled data is much smaller than the size of unlabeled data. The pool should reserve as many unlabeled data as possible if the memory is enough to store them and the learner is not sure about the data's potential value in some later stage of active learning. In this work, we focus on the pool-based active learning where we have a massive high dimensional data in the pool.

Fig. 1 elaborates the detail procedure of the focused pool-based scenario. Initially, we use a small training set to build a preliminary model, a model likely to be far from perfect for prediction. The preliminary model is used to estimate the label information of the unlabeled pool and then we select a set of data for querying with a pre-defined criterion, such as uncertainty sampling. The active learning system shall rank the pool data based on the pre-defined uncertainty measure and then choose a small set of unlabeled data with the top ranks for an oracle to label. After we add the newly labeled data to the training set, the model should be re-trained, and the information value in the pool will be modified again and the iteration goes on until we have a satisfiable model.

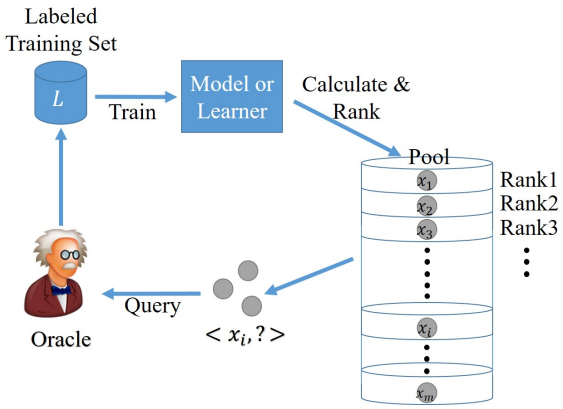

Fig. 1: Pool-Based Active Learning

We have to emphasize that the traditional pool-based AL has to calculate the data information for ranking and rank the pool in each iteration. It creates a bottleneck in the active learning computation. The difficulty increases when we have high dimensional data. We propose ARPAL and MLP strategies to solve the problem. Before we go on to discuss the details of the two strategies, we discuss the uncertainty sampling that we adopt in this work.

2) Uncertainty Sampling: The active learning system needs to have query strategies to measure the utility of unlabeled data. There are many proposed query strategies and one

of the most common and effective strategies is uncertainty sampling [15]. Others include checking the data diversity or representativeness. What is different between the three is that data diversity and representativeness can be estimated without knowing any label information, therefore can be computed offline as long as the data features are available. In other words, we should focus on the uncertainty computation which needs some online effort to obtain the result.

To speak of the uncertainty computation, the basic idea is that the learner should focus on the unlabeled data that has low confidence rather than high confidence on its label information. Probabilistically, we can estimate $P(y \mid \mathbf{x})$ to judge the confidence level on labeling the unlabeled data with features $\mathbf{x}$. The judgement should adjust according to either a binary classification or multi-class classification problems. For the binary case, a low label confidence on a given unlabeled data $\mathbf{x}$ may be indicated by a probability of $P(y \mid \mathbf{x})$ close to 0.5 for $y \in \{1, -1\}$. It is quite possible that $\mathbf{x}$ is located near the decision boundary in this case. There are three different uncertainty strategies to measure the utility of instances: (1) the least confidence computation, (2) distance to margin estimation, and (3) entropy estimation. In this work, we follow Joshi et al. [8] to choose the margin-based uncertainty sampling for the data selection strategy.

B. Anti Re-Rank Pool-Based Active Learning (ARPAL)

In traditional pool-based active learning, a drawback of the method is that it has to rank the unlabeled data pool all the times and it slows down the system's performance. We observe that many of the data ranking and data rank computation may not be necessary. If the models are stabilized in the active learning procedure, the models from consecutive iterations as well as their predictions may be similar. Therefore, the data ranks remain to be similar and we do not need to re-calculate the data ranks again. It is more likely to be in this case if we run into the late stages of the AL procedure where we may have enough data to understand the whole distribution. On the other hand, if the models are not reliable, we may need to add more labeled data for a better training. In this case, the data rank computation based on the model may not be reliable either. For the first situation, we try to avoid unnecessary data re-ranking and for the second case, we may focus on the data rank computation for a small subset of the whole unlabeled pool. We propose ARPAL to deal with the first case and Multi-Layer Pool (MLP) for the second case.

According to the above, we only need to check the difference between the current model and the previous model to make the decision on re-ranking. If the difference is large or the model changes a lot in a moment, we perform re-ranking in the pool, otherwise do nothing if the difference is small or the model changes very little. All we need is a criterion to judge whether or not the model changes more or less than a pre-defined threshold. Let us use the perceptron learning algorithm for binary classification as an example to illustrate the idea. In the perceptron learning, we compute the inner product $\sum_j w_j x_j$ and use it as the decision boundary for binary classification for a given input $\mathbf{x} = (x_1, x_2, \ldots)$ and a model weight $\mathbf{w} = (w_1, w_2, \ldots,)$. Similarly, there is also a weight vector $\mathbf{w}$ to specify a model for linear SVM. In either case, we can compute the well-known cosine similarity to estimate the similarity between the current model $\mathbf{w}_t$ and the previous model $\mathbf{w}_{t-t'}$ which is the model a few steps back. (In between the model does not change at all.) The similarity measure ranged from zero to one, is defined as:

$$\text{Model Similarity (MS)} = \frac{\langle \mathbf{w}_{t-t'}, \mathbf{w}_t \rangle}{\|\mathbf{w}_{t-t'}\| \cdot \|\mathbf{w}_t\|} \tag{1}$$

In ARPAL, we calculate the similarity between the models in two moments. Also, we set a threshold $0 \leq \sigma \leq 1$ to decide whether or not to perform the re-ranking. That is, if the model similarity is less than the pre-defined threshold σ, we should re-calculate the data ranks. This approach reduces the frequency of unnecessary re-ranking and therefore can be more efficient than the traditional AL. The complete algorithm of ARPAL is shown in Algorithm 1.

Input : σ: similarity threshold ;
q: number of samples selected by the query at once ;
L: labeled data for training ;
U: unlabeled data in the pool.
Output: X: a set of labeled samples to be added to the training set
repeat
  Use L to train a model ;
  if *MS (Eq. 1)* $< \sigma$ **then**
    Compute margin for each $\mathbf{x} \in U$ in the pool ;
    Rank the pool according to the margin criterion ;
  end
  `// Keep the previous ranking otherwise.`
  Initialize X to the empty set $(X = \varnothing)$;
  Add the q most uncertain samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_q\}$ (highest ranks from the pool) to X ;
  Request oracle for X's labels ;
  Add $X = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_q, y_q)\}$ to training set L
until *Model reaches its pre-defined accuracy*;
Algorithm 1: Anti Re-rank Pool-Based Active Learning

ARPAL overcomes the inefficiency of traditional active learning. We compute the model similarity and bypass the re-ranking if the similarity between the model now and the model in a previous moment is high.

Let us also discuss the similarity measure when a nonlinear SVM is chosen to be the classifier. In a nonlinear SVM, instead of dealing with the weight vector $\mathbf{w}$, we store the support vectors $\{\text{SV}_{t_n}\}$ and the corresponding Lagrange multipliers $\{\alpha_{t_n}\}$ where t_n is the number of support vectors at the t-th iteration. The following formula describes how to compute the similarity between the current model and the previous model:

$$\text{MS}_{\text{nonlinear}} = \frac{\left\langle \sum_{i=1}^{t_{n'}} \alpha_i \text{SV}_i, \sum_{i=1}^{t_n} \alpha_i \text{SV}_i \right\rangle}{\left\| \sum_{i=1}^{t_{n'}} \alpha_i \text{SV}_i \right\| \cdot \left\| \sum_{i=1}^{t_n} \alpha_i \text{SV}_i \right\|}, \tag{2}$$

where the $t_{n'}$ is the number of support vectors for the previous model. We use Eq. 2 to decide whether or not we need another re-ranking in the active learning procedure.

C. Multi-Layer Pool (MLP)

We propose another strategy for efficient active learning. In traditional pool-based active learning, the leaner may take a while to compute the data ranks given a huge unlabeled set to work with. After all, there are only a few queries that are associated with high ranks and those shall be sent to oracle for labeling information. We believe that such large-scale screening should be avoided. It is even more serious when we deal with high-dimensional data. At the same time, we also observe that the model in the early stage of active learning procedure may not be reliable. Therefore, a wise data rank computation is important.

To deal with the situation, we consider applying approximate computation and exact computation for likely-to-be low-rank data and high-rank data respectively to save the computation. More specifically, we utilize dimensionality reduction to find the low-dimensional projection for the unlabeled data to have a quick understanding of the data. After that, the data that are associated with high ranks can be checked again and again for a more precise computation until we have the final set of querying data for oracle. Fig. 2 shows such idea. The data structure underneath is a hierarchical structure where we organize the whole unlabeled data in the structure from the bottom to the top with fewer and fewer number of data. At the same time, we consider the dimensionality reduction with more and more dimensionality from the bottom to the top. By having that, we have an approximate to an exact computation from the bottom to the top.

1) Principal Components Analysis: We simply choose Principal Components Analysis (PCA) [16], one of the most well-known dimensionality reduction methods to realize the proposed idea. The main idea of PCA is to find a low-dimensional projection of the original data where the data relation can be kept as much as possible. More rigorous, we find the projection that can maximize the variance between data points or the direction that is associated with the largest eigenvalue to be the principal component and we continue finding a series of dimensions to be the ones that we keep the data distribution as much as we can in a low-dimensional space. The first few principal components will retain most of the variance present in the original attribute set. In this paper, we use the eigenvalues to decide how many principal components we should keep:

$$r = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{p} \lambda_j}, \tag{3}$$

where $\sum_{j=1}^{p} \lambda_j$ is the sum of all eigenvalues, and r is the ratio accounted for the first k eigenvalues. We set a threshold r^* somewhere between 65% and 95%, and find the k which is the smallest integer for which $r > r^*$ where we retain k PCs.

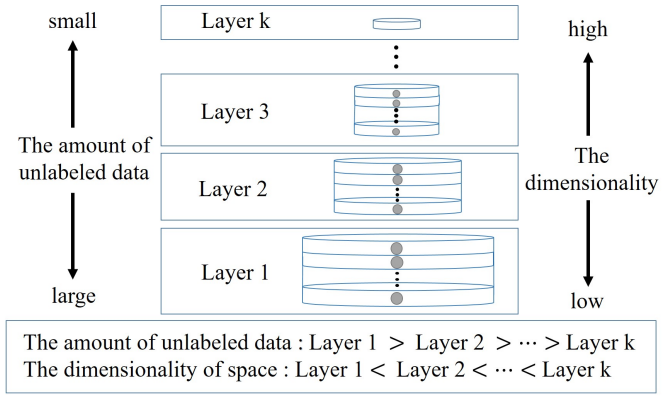

Fig. 2: The Multi-Layer Pool structure. From the bottom to the top layers, we have a large to small set of data. On the other hand, we perform an approximate (applied in low dimensionality) to an exact (applied in high dimensionality) data rank computation from the bottom to the top.

2) Multi-Layer Pool Building: Assuming we want to build a k layers pool, we use a dimensionality reduction method such as PCA to transform the data set into the sets in different dimensional space and create k models via training sets on those different dimensional space. More specifically, the model with the lowest dimensionality will measure and rank the uncertainty value of all unlabeled data based on the limited information from what we can observe from the low-dimensional space to build the first layer of the pool. After that, part of the data with higher ranks in the first pool shall be sent to higher layers for finer measurement. The model in the second layer will re-train the model, re-calculate and re-rank those unlabeled data to have a more accurate ranking to build the second layer of the pool, and we repeat the procedure until we reach the k-th layer. The structure of the multi-layer pool is displayed in Fig. 2.

To decide how many layers that we need, we compute:

$$\text{Layer}_{\text{total}} = \left\lceil \frac{1}{3} \ln(m) \right\rceil, \tag{4}$$

to find the number of layers, where $m = |U|$ is the amount of all unlabeled data. Intuitively, the number of layers should be correlated with the number of (unlabeled) data for ranking. Taking logarithm makes sense due to the pyramid-like structure of MLP. On the other hand, we also need to consider not too many layers because we perform training, rank computation and sorting at each layer which may create extra cost.

After determining the number of layers, we have to decide the number of data in each layer. In Eq. 5, we have m_i data in the i-th layer, which is decided by the number of data and the dimensionality for the data. To go from the i-th layer to higher layers, we should select more data if the PCA suggests a low ratio r_i (Eq. 3), which implies a projection to a low-dimensional space still keeps important information from the original data.

TABLE I: The number of data in different subsets of active learning procedure.

	# of instances
Initial training set	100
Unlabeled query set	50000
Test set	2000
The number of data in a query	10

$$m_i = \begin{cases} m & \text{if} \quad i = 1 \\ (1 - r_i) \ \dfrac{m}{\ln(m)} & \text{if} \quad i = 2, \ldots, \text{Layer}_{\text{total}} \end{cases} \tag{5}$$

IV. Experiment Results

In this section, we demonstrate the experiment results as well as their evaluation according to different methods. Before go on to present each experiment result, we first introduce the dataset for evaluation in Subsection IV-A, which is followed by the elaboration of the experimental settings (Subsection IV-B) in this work. The rest of the section is spent on all the experiment results and the discussion.

A. Dataset and Preprocessing

We evaluated the proposed method on the well-known MNIST hand-written digits dataset which consists of 10-class digit images from '0', '1', to '9'. The images were centered in a 28×28 image, with each pixel as a feature a total of 784 features for our training. The total size of the dataset includes $60,000$ examples, and a test set of $10,000$ examples.

Given the MNIST dataset, we first apply min-max normalization to scale all numeric values to be within the range of $[0, 1]$. Afterwards, we choose PCA (Section III-C1) to reduce the data dimensionality.

B. Experimental Settings

In pool-based AL, we have a small labeled data to train a preliminary model initially. Starting from the first step, a large set of unlabeled data should be used as the possible candidates for querying oracles. In each step, we ask oracles for about 10 queries and add the ten newly labeled data to the training set and hope to have an improved model after the re-training. On the side, we use a separate test set to continuously evaluate the model effectiveness. The number of data for different parts of active learning is presented in Table I. After all, we perform 250 iterations in all experiments and record the accuracy and computation time. We compare several AL approaches, the traditional AL, the AL with either ARPAL, MLP or both, as well as the random sampling for our evaluation. In the experiments, we chose SVM [17], [18] to be the base learner. Both of the linear and nonlinear models shall be adopted for evaluation.

C. Anti Re-rank Pool-based Active Learning

1) Linear Model: We take turns to evaluate the proposed ARPAL and MLP strategies for active learning with massive high-dimensional data. In the first series of experiments, we start with the simple linear SVMs as the base model for evaluation. We compare the linear SVM-based active leaner with or without the ARPAL strategy for the evaluation. More specifically, we compare between the traditional pool-based AL, the AL with random sampling and the AL with ARPAL, as shown in Fig. 3.

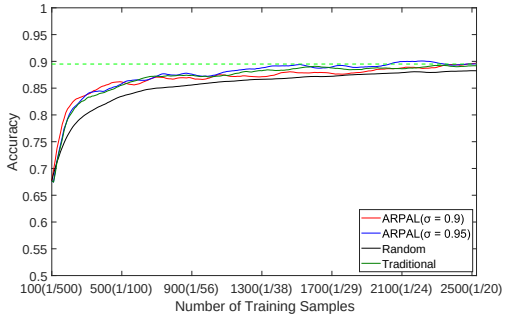

Fig. 3: Active learning with ARPAL from linear SVM.

TABLE II: Active learning with ARPAL from linear SVM.

	Computation Time	# of Re-ranking
ARPAL ($\sigma = 0.9$)	0 hr 57 min 20 sec	17 times
ARPAL ($\sigma = 0.95$)	1 hr 14 min 52 sec	34 times
Random	0 hr 23 min 13 sec	0 times
Traditional	6 hr 25 min 48 sec	250 times

In Fig. 3, x-asis indicates the number of data that is manually labeled by oracles along the AL procedure (which is equal to the training set size) and y-axis shows the model accuracy. The threshold to deciding the similarity of models in consecutive steps is set to be either $\sigma = 0.9$ or $\sigma = 0.95$ from Eq. 1. In general, we have the AL with ARPAL perform similarly to the traditional AL without ARPAL in terms of model accuracy. On the other hand, all methods with active learning are indeed perform better than the labeling with random strategy. We performed ten times and compute the average for the random strategy result.

To speak of computation time, we just need to spend a little more than the random strategy for the active learning with ARPAL strategy. On the other hand, we enjoy a significant computation time reduction, about 85% if compared to traditional active learning, as shown in Table II. Note that choosing $\sigma = 0.95$ doubles the times of re-ranking if compared to the case of $\sigma = 0.9$.

2) Non-Linear Model: In this part, we consider using nonlinear model or nonlinear SVM to be more specific as the base learner for the evaluation. Extending to the more powerful

TABLE III: Active learning with ARPAL from nonlinear SVM.

	Computation Time	# of Re-ranking
ARPAL ($\sigma = 0.9$)	1 hr 43 min 09 sec	4 times
ARPAL ($\sigma = 0.95$)	1 hr 52 min 11 sec	8 times
ARPAL (fixed)	2 hr 24 min 12 sec	25 times
Random	1 hr 24 min 55 sec	0 times
Traditional	9 hr 37 min 03 sec	250 times

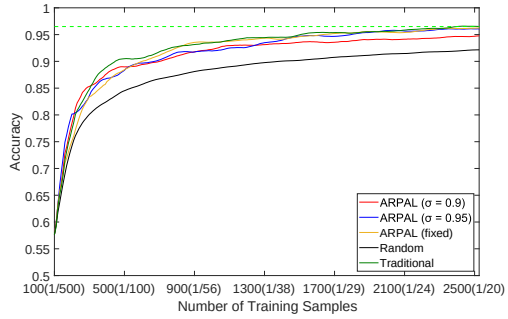

Fig. 4: Active learning with ARPAL from nonlinear SVM.

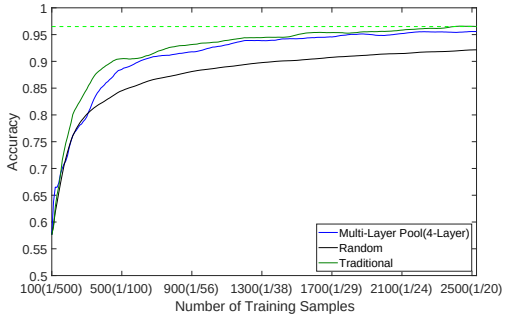

Fig. 5: The MLP result with non-linear SVM.

TABLE IV: The time index of re-ranking in the active learning with ARPAL from nonlinear SVM.

ARPAL ($\sigma = 0.9$)	ARPAL ($\sigma = 0.95$)	ARPAL (fixed)	Random	Traditional
1	1	1	None	1
8	4	11	·	2
26	8	21	·	3
72	16	·	·	·
	33	·	·	·
	61	·	·	·
	107	·	·	249
	225	241	None	250

model than before, we would like to see the advantage of applying ARPAL strategy to active learning. In Fig. 4, we observe that AL with ARPAL performed similarly to the traditional AL, while both of them performed better than the random sampling approach. Compared to the linear case, we can see a larger gap between the AL with or without ARPAL and the random sampling. That is, the nonlinear AL with or without ARPAL can have about 5% higher accuracy than the one from linear AL, especially in some later stage of training. Regarding to the computation, we again see the advantage of applying ARPAL to AL, as shown in Table III. Adopting the ARPAL strategy with $\sigma = 0.9$, we have about 82% deduction in time, almost eight hours faster than the AL without using ARPAL. Note that ARPAL (fixed) is a strategy that we perform re-ranking for every ten iterations. In general, we can argue that ARPAL (fixed) performs some unnecessary re-ranking in the late stage of active learning which makes its computation time longer than that from the proposed ARPAL, as shown in Table III. We should also point out that using ARPAL (fixed) needs to decide the number of steps for each re-ranking, which could be hard to find the best choice.

Let us check some more details for the proposed ARPAL method to see when and how it performs re-ranking. In Table IV, we observe that ARPAL tends to do re-ranking in some early stages of the active learning procedure. In the early stage of the active learning procedure, the model does not have enough information to estimate the full distribution of the data, therefore frequent re-ranking could be necessary given newly acquired labeled data.

TABLE V: The MLP result with non-linear SVM.

	Computation Time	# of Re-ranking
MLP (4-Layer)	3 hr 17 min 21 sec	250 times
Random	1 hr 24 min 55 sec	0 times
Traditional	9 hr 37 min 03 sec	250 times

D. Multi-Layer Pool

In this subsection, we demonstrate how the proposed MLP strategy can further enhance the efficiency. In the experiments, we need to choose a few out of $50,000$ unlabeled candidate data for an oracle to label. In our design, the $50,000$ unlabeled pool was organized in four layers with the threshold r^* set to be 65%, 75%, 85%, 95% and use Eq. 4 to determine the dimensionality. Also, we use Eq. 5 to determine the data size in each layer (m_i).

In Fig. 5, we compare the active learning with the MLP strategy to the traditional active learning and random sampling. Again, we observe a huge gap between the active learning with or without the MLP strategy and the random sampling while the difference between the MLP-based active learning and the traditional active learning is relatively small.

To speak of the computation between different methods, we find out a clear advantage of applying MLP to active learning. According to Table V, given that both the MLP and traditional active learning re-ranked the pool 250 times in the experiment, the MLP-based active learning is faster than the traditional AL for about six and half hours. That is, the MLP-based method introduced about 66% time deduction if compared to the traditional AL. It implies that sacrificing a little accuracy for a huge computation time deduction can be a good choice for active learning developers.

TABLE VI: The ARPAL and MLP combined active learning with non-linear SVM.

	Computation Time	# of Re-ranking
MLP (4-Layer)	3 hr 17 min 21 sec	250 times
ARPAL($\sigma = 0.9$)	1 hr 43 min 09 sec	4 times
Combined($\sigma = 0.9$)	1 hr 36 min 25 sec	4 times
Random	1 hr 24 min 55 sec	0 times
Traditional	9 hr 37 min 03 sec	250 times

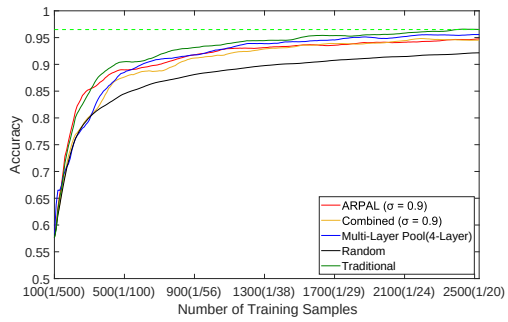

Fig. 6: The ARPAL and MLP combined active learning with non-linear SVM.

E. ARPAL + MLP

The ARPAL and MLP strategies are two more or less independent strategies that we can apply to active learning. Combining them is a good choice to save more time even we may have a small loss on its accuracy. Fig. 6 and Table VI show the combined result. Even the combined method does output the worst result from all active learning approaches in terms of effectiveness, we do observe that the difference between them is quite small. On the other hand, the difference between the computation time from various active learning methods is not that small or about 83% deduction if comparing between the ARPAL and MLP combined approach and the traditional active learning. Clearly, we observed that adopting ARPAL and MLP together can be a good choice for active learning given large scale unlabeled data. It is also not very difficult to imagine an even larger advantage for a larger unlabeled data.

V. Conclusions

We proposed ARPAL and MLP strategies to improve the efficiency of applying active learning for large-scale and high-dimensional data. The ARPAL strategy is used to judge whether or not we should re-rank the unlabeled data in each active learning iteration. If the models change little between this and the next moment, we choose to avoid the re-ranking. In a more precise treatment we considered the difference between the learning in the early stage and in the late stage because the model is expected to be stabilized in the late stage of learning. On the other hand, we further improved the ARPAL-based active learning efficiency by arranging unlabeled data for ranking in a multi-layer pool with the MLP strategy. As shown in experiments, the proposed ARPAL and MLP combined approach can speed up the active learning procedure by about 83% if compared to the traditional active learning; while at the same time, the model effectiveness remains. We believe that the proposed strategies and developed framework for active learning can have an impact for the active learning research in the big data era.

Acknowledgements

This research was partially supported by the Ministry of Science and Technology Grants MOST 105-2221-E-011-131, MOST 106-2221-E-011-159 and also supported in part by the Ministry of Science and Technology, National Taiwan University, and Intel Corporation under Grants MOST 105-2633-E-002-001, 106-2633-E-002-001 and NTU-105R104045, NTU-106R104045. The authors also acknowledge anonymous referees for their constructive criticisms.

References

[1] B. Settles, *Active Learning*. Morgan Claypool, 2012.

[2] D. D. Lewis and 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*, ser. SIGIR '94. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 3–12.

[3] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao, "Exploring representativeness and informativeness for active learning," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 14–26, 2017.

[4] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Advances in neural information processing systems*, 2010, pp. 892–900.

[5] P. Melville and R. J. Mooney, "Diverse ensembles for active learning," in *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, Banff, Canada, July 2004, pp. 584–591.

[6] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Mar. 2002.

[7] J. Kremer, K. Steenstrup Pedersen, and C. Igel, "Active learning with support vector machines," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 4, pp. 313–326, 2014.

[8] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 2372–2379.

[9] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and Information Systems*, vol. 35, no. 2, pp. 249–283, 2013.

[10] B. Demir, C. Persello, and L. Bruzzone, "Batch-mode active-learning methods for the interactive classification of remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 3, pp. 1014–1031, March 2011.

[11] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218–2232, July 2009.

[12] P. Donmez and J. G. Carbonell, "Proactive learning: Cost-sensitive active learning with multiple imperfect oracles," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 619–628.

[13] C. Persello, A. Boularias, M. Dalponte, T. Gobakken, E. Nsset, and B. Schlkopf, "Cost-sensitive active learning with lookahead: Optimizing field surveys for remote sensing data classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 10, pp. 6652–6664, Oct 2014.

[14] A. Kapoor, E. Horvitz, and S. Basu, "Selective supervision: Guiding supervised learning with decision-theoretic active learning." in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 877–882.

[15] D. D. Lewis and J. Catlett., "Heterogeneous uncertainty sampling for supervised learning," in *Machine Learning Proceedings 1994Proceedings of the Eighth International Conference*, W. W. Cohen, Ed. Morgan Kaufmann, 1994, pp. 148–156.

[16] I. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[17] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. Springer, November 1999.

[18] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199