Active Class Discovery and Learning for Networked Data

Meng Fang*† Jie Yin† Xingquan Zhu* Chengqi Zhang*

Abstract

With the recent explosion of social network applications, active learning has increasingly become an important paradigm for classifying networked data. While existing research has shown promising results by exploiting network properties to improve the active learning performance, they are all based on a static setting where the number and the type of classes underlying the networked data remain stable and unchanged. For most social network applications, the dynamic change of users and their evolving relationships, along with the emergence of new social events, often result in new classes that need to be immediately discovered and labeled for classification. This paper proposes a novel approach called ADLNET for active class discovery and learning with networked data. Our proposed method uses the Dirichlet process defined over class distributions to enable active discovery of new classes, and explicitly models label correlations in the utility function of active learning. Experimental results on two real-world networked data sets demonstrate that our proposed approach outperforms other state-of-the-art methods.

Keywords: Active learning, networked data, class discovery

1 Introduction

In most data mining problems, obtaining label information for training classification models is usually an expensive and time consuming process [2]. Instead of labeling all instances, or randomly selecting instances to label, active learning [7, 21] represents a family of methods that selectively choose the most informative instances for a labeler (or an oracle) to label. Active learning aims to significantly reduce the labeling cost, but still maintain similar classification accuracy. It has been extensively studied for almost two decades since [14] and has become a popular tool to reduce labeling costs for many real-world data mining applications.

In recent years, the widespread use of social network systems has enabled the availability of a huge amount of networked data, in which the data exists in graph structure where nodes (V) denote instances[1] (such as users or scientific publications) and edges (E) denote relationships between nodes (*e.g.* users in a social network may share friendship and scientific publication may have citation relationships). Because instances are not independent, but are connected by links between each other to form a network, the labels of neighboring instances are correlated. For example, in a citation network, papers that cite each other often have similar topics. In friendship networks, people that are friends are likely to have similar interests. In this situation, effective active learning algorithms should not just depend on the properties of an instance itself, as does traditional active learning. Instead, the network structure should be taken into consideration to choose the best instances for labeling. Recent research works [3, 4, 11, 15, 22] have shown that selectively querying node labels based on network connectivity and collective classification significantly improves the active learning performance.

Despite much progress in active learning for networked data, two important issues have not been well addressed. First, almost all the algorithms have assumed that the class space, to which the instances are classified, is known and remains stable. This often leads to a stronger assumption that training data consists of samples from all the classes, so the goal of the active learning algorithm is simply to label the most informative samples with respect to the known classes. However, the networked data is complex and dynamic in nature. Some classes may not exist in advance, or the class might exist but samples representing the class are simply not available at the time when data are collected for training. For example, in Twitter, where users constantly produce new tweets, abnormal events, such as terrorist attacks, or natural disasters often appear abruptly and attract significant attention. It is clear that these events have not happened before, so each has its unique class label that does not exist in the existing class space. *The active learning goal is therefore not only to label informative samples, but also to actively discover and update the class space for the learning task*, as shown in Figure 1.

*Centre for Quantum Computation & Intelligent System, Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia.

†Information Engineering Laboratory, CSIRO ICT Centre, Australia.

[1] In this paper, nodes and instances are interchangeable terms

Second, most of the existing research has not explicitly modeled label correlations in the utility calculation of active learning for querying. Previous work has primarily focused on using graph-based metrics to define the informativeness of instances [3, 6, 22], or on combining link information with node-specific features to train a classifier [4, 11]. In the context of networked data, the labels of linked nodes are correlated in the local neighborhood, and such dependencies, if properly modeled in the utility function of active learning, could significantly contribute to classifying the nodes in the network.

In this work, we propose ADLNET (active discovery and learning for network data), which incorporates network structure and collective classification to enable active discovery of new classes and learning. Specifically, we use a Dirichlet process (DP) model defined over class distributions to estimate the probability of an instance belonging to known or unknown classes. The DP model is thereafter coupled with the empirical risk minimization (ERM) approach that minimizes the expected classification error of a collective classifier to identify nodes in the network for labeling. We also introduce a social regularization term to explicitly capture label correlations between neighboring nodes. A new utility function, which integrates new class discovery, classification, and social regularization, is proposed to select the most informative nodes for active learning.

To improve computational efficiency, we also employ graph-based measures to speed up active learning on large-scale networked data. The ERM approach employed by ADLNET, while having good performance, is computationally expensive because it needs to examine classification error reduction with respect to the models trained for each unlabeled instance. This means that ERM needs to induce a classifier for each unlabeled instance and chooses the one with the minimum expected error. Motivated by the observation that the instances picked up by ERM are "important" nodes from network perspectives [15], we propose to use various graph-based measures to limit the search space for ADLNET. That is, ADLNET employs these measures to pick up a small subset of instances as candidates, and then uses ERM to find the best instance to label among these candidates. This hybrid scheme achieves a better tradeoff between classification accuracy and computational efficiency.

2 Related work

Active learning is a machine learning strategy that allows a prediction model to construct its training data in interaction with an oracle. It aims to reduce the labeling cost by choosing informative instances to label. In this paper, we focus on pool-based active learning. Depending on what query strategies are used,

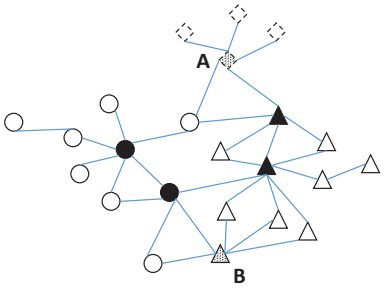


Figure 1: An example of class discovery and active learning for networked data. The shape of each node denotes its class. Solid-filled nodes and empty nodes denote labeled and unlabeled nodes, respectively. Nodes with dashed lines belong to a new class that has not been discovered. Nodes A and B are the most informative nodes from a class discovery and active learning perspective, respectively.

pool-based active learning techniques can be broadly grouped into three categories [21]. The first category is based on uncertainty sampling, in which an active learner selects an instance that it is most uncertain about to label [12, 14, 24]. The second category is the query-by-committee (QBC) approach [8, 17]. The QBC approach maintains a committee of classifiers and the most informative query is considered to be the one that all classifiers most disagree about. The third family of methods aim to query instances where the expected classification error can be reduced as much as possible (*i.e.*, minimize the empirical risk) [15, 19].

Recently, graph-based active learning has been proposed to address the problem of classifying networked data [3, 4, 6, 15, 22]. Some existing research has focused on using graph-based metrics to define the informativeness of instances and then select instances with the highest informative scores [3, 6, 22]. Other research has considered combining link information with node-specific features to train the classifiers, and then using various query strategies for instance selection [4, 11]. Bilgic et al. [4] proposed ALFNET, an active learning method for networked data. ALFNET uses clustering techniques to form an initial labeled set. At each iteration, ALFNET builds three classifiers and computes a local disagreement score for each node. These scores are aggregated for each cluster and thereafter clusters with the highest scores are chosen, from which a set of instances are selected to label. Kuwadekar and Neville [11] proposed a semi-supervised method that models a network structure using relational dependency networks and relies on an ensemble of models to select the best instance to query. However, these methods are based on a restrictive assumption that the initial training set

contains examples from all the possible classes; thus, there is a lack of capability to model and discover new classes.

Early research work on new or novel class detection has been found in active learning and data stream areas [9, 10, 13, 16]. Masud et al. [16] presented an active learning algorithm to discover new classes in data streams, in which a pre-specified threshold is used to differentiate existing classes and new classes. Hospedales et al. [9] proposed an active learning algorithm using both generative and discriminative models. This method relies on calculating the likelihood of each instance being generated by a Gaussian mixture model to detect new classes. Similarly, the work by [10] also depends on the estimated class membership probabilities to identify new classes. None of the methods consider a prior model for class distributions, which results in the decision for discovering a new class to be mainly ad-hoc and data driven. Also, the lack of ability to model network structures makes these approaches inapplicable for our classification problem on networked data.

Another line of work related to active discovery of new classes has been developed in the area of online document clustering with novelty detection [1, 26]. These approaches use a non-parametric Dirichlet process prior to model the growing number of clusters, and to handle the generation of novel clusters. This has inspired us to use a Dirichlet process to model online class distributions in our active learning framework for new class discovery. Different from online cluster modeling which depends on a vague prior distribution, in our work, the Dirichlet process prior distribution can be estimated using examples from corresponding classes. Coupled with the empirical risk minimization approach, our proposed algorithm, to the best of our knowledge, is the first research work to address active class discovery and learning for complex networked data.

3 Problem Definition and Preliminary

This work focuses on pool-based active learning problem. The networked data is represented as a large graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes a set of nodes (vertices) and $\mathcal{E}$ denotes the edges between nodes. Each node $v_i \in \mathcal{V}$ is described by a feature vector $\mathbf{x}_i = \hbar(v_i)$ and a class label $y_i \in \mathcal{Y}$, where $\hbar(v)$ denotes a function that extracts a feature vector for node v and $\mathcal{Y}$ denotes a set of class labels. An edge $e_{ij} \in \mathcal{E}$, connecting nodes v_i and v_j, describes relationships between v_i and v_j. Given the initial class space $\mathcal{Y}$ and a set of labeled nodes $\mathcal{V}^l \in G$ with $\mathcal{V}^l = \{(v_i, y_i)\}_{i=1}^N$ and $y_i \in \mathcal{Y}$, our active learning problem **aims** to: (1) label the most informative samples to build a learner for classifying unlabeled nodes $\mathcal{V}^u \in G$ with maximum accuracy, and (2) discover new classes to include in the class space $\mathcal{Y}$.

For each query t and its label y_t provided by the labeler, if class y_t does not exist in the current class space $\mathcal{Y}$, *i.e.* $y_t \notin \mathcal{Y}$, we will need to expand the class space to include new label $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_t\}$. For example, in Figure 1, "circle" and "triangle" are two classes that have been discovered. If an algorithm selects A as a query node, the "diamond" will be discovered and included in the class space.

Given the current class space and networked data set that contains labeled and unlabeled data, collective classification [20] is commonly used to predict the label of nodes for networked data, by using the correlations between neighbors as additional features. Assume N_i denotes the labels of the neighbors of node v_i, where $N_i = \{y_j | e_{ij} \in \mathcal{E}\}$, collective classification uses a variety of aggregation operators including `count`, `model` and `proportion`, *etc.* to construct new features $aggr(N_i)$. It then trains a probabilistic classifier based on node v_i's original features $\mathbf{x_i}$, and new features $aggr(N_i)$, represented by $P(y_i|\mathbf{x_i}, aggr(N_i))$.

The iterative classification algorithm (ICA) is a popularly used collective classification method [20], which is based on a local vector-based classifier. For each node v_i represented by its feature vector $\mathbf{x_i}$, ICA calculates the aggregation values as link features to learn $P(y_i|\mathbf{x_i}, aggr(N_i))$. It iteratively updates the predictions of all nodes by using the previous predictions for unknown labels in the neighborhood as new features $aggr(N_i)$, until the algorithm converges.

4 The Proposed Algorithm

Since our objective is to selectively label informative samples and actively discover and update the class space for networked data, we use an objective function in Eq.(4.1), which integrates three individual utility components including utilities for classification ($U_{classification}$), class discovery ($U_{discovery}$), and social regularization (U_{social}) in a networked environment. Parameters α and β in Eq.(4.1) control respective contributions of the three components to the overall query selection strategy, and the question of how they would affect the classification accuracy will be empirically investigated.

$$U = U_{classification} + \alpha U_{discovery} + \beta U_{social} \tag{4.1}$$

Once the utility function defined in Eq.(4.1) is suitably determined, our ADLNET algorithm will follow a typical active learning process to query class labels for selecting instances and expanding the class space, if necessary. Algorithm 1 lists the pseudo-code of the ADLNET algorithm, with detailed steps explained in the following sub-sections.

Algorithm 1 Active class Discovery and Learning for Networked Data

Input: (1) Networked data: $G = (\mathcal{V}, \mathcal{E})$; (2) initial class space: $\mathcal{Y}$; and (3) label budget: *budget*.
Output: Labeled data G and updated class space $\mathcal{Y}$.

```
V^u ← unlabeled nodes in G
x ← feature vector of a node v
Train initial models and ICA classifier
numQueries ← 0
while numQueries ≤ budget do
   for each v_i ∈ V^u do
      U_newclass ← New class probability Eq.(4.3)
      U_classification ← Classification uncertainty Eq.(4.9)
      U_network ← Social regularization Eq.(4.10)
   end for
   v* ← arg max U(v_i) defined in Eq.(4.1)
   y_v* ← Query class label of v*
   if y_v* ∉ Y then
      Y ← Y ∪ y_v*
   end if
   V^u ← V^u \ v*
   Update models and ICA classifier
   numQueries ← unmQueries + 1
end while
```

4.1 Active Class Discovery One important objective of the ADLNET algorithm is to actively discover and update the class space for the active learning task. It is a non-trivial task to calculate the probability of an instance belonging to an unknown class because nothing has been observed yet for the class. To solve this problem, we model class distributions under the Dirichlet process assumption so as to calculate the probability of an instance being to a new class. Dirichlet process is a stochastic process used to handle an infinite number of classes [18]. It has proven useful for non-parametric Bayesian modeling on online document clustering.

In particular, we use a two-parameter Poisson-Dirichlet Process (PDP) [5], which is an extension of the Dirichlet Process. It is defined as

$$G|G_0, a, b \sim Dirichlet(a, b, G_0), \tag{4.2}$$

where G_0 indicates a base distribution over a probability space. The discount parameter a, $0 \leq a \leq 1$, and the strength parameter b, $b \geq -\alpha$, control the amount of variation of G from G_0.

In PDP, the number of classes is assumed to be infinite and we use its marginalized posterior – the Chinese restaurant process – as a prior on the true class distribution. The Chinese restaurant process can be described using the analogy of a restaurant with an infinite number of tables, where K tables are already occupied and n_k is the number of customers already at the kth table. The total number of customers in the restaurant is $n = \sum_{k=1}^{K} n_k$. Suppose that other customers enter the restaurant once at a time and choose a table to sit at. The probability of choosing an occupied table is $(n_k - b)/(n - 1 + a)$, and the probability of choosing a new table is $(a + bK)/(n - 1 + a)$.

We assume that the class is not bounded for the K classes observed so far. Node v (and its feature vector $\mathbf{x} = \hbar(v)$) may belong to an existing class $k \in \{1, \ldots, K\}$ or a new class $K + 1$. Using PDP, we define the existing and the new class probabilities for node v as follows.

$$p_{pdp}(y = k|\mathbf{x}) = \begin{cases} \frac{n_k - b}{(n-1+a)} p(\mathbf{x}|y = k) & if\ k \leq K, \\ \frac{a+bk}{(n-1+a)} p(\mathbf{x}) & if\ k = K + 1. \end{cases} \tag{4.3}$$

In Eq.(4.3), parameters a and b are estimated by putting hyper priors over them to infer their values using the sampling strategy described in [23].

To calculate conditional probability $p(\mathbf{x}|y)$ in Eq.(4.3), we can employ any probabilistic models, and in our work, we use a Gaussian mixture model

$$p(\mathbf{x}|y) = \sum_{t=1}^{T_y} w_{y,t} g(\mathbf{x}|\mu_{y,t}, \Sigma_{y,t}), \tag{4.4}$$

where $w_{y,t},\ t = 1, \ldots, T_y$ are the mixture weights, and $g(\mathbf{x}|\mu_{y,t}, \Sigma_{y,t})$ are the component Gaussian densities, defined as a D-variable Gaussian function form

$$g(\mathbf{x}|\mu_{y,t}, \Sigma_{y,t}) = \frac{1}{(2\pi)^{D/2}|\Sigma_{y,t}|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_{y,t})\Sigma_{y,t}^{-1}(\mathbf{x} - \mu_{y,t})\}. \tag{4.5}$$

The density $p(\mathbf{x})$ can be estimated by using Gibbs sampling or a Gaussian model. For simplicity, we construct a Gaussian model from all the instances to calculate $p(\mathbf{x})$.

By substituting $p(\mathbf{x}|y)$ and $p(\mathbf{x})$ in Eq.(4.3), we obtain the probability of an instance belonging to an existing class or a new class. Intuitively, a new class is likely to emerge, if we observe that an instance has a high probability of being generated by the new class. Therefore, we define $U_{discovery} = p(y_{new}|\mathbf{x})$ to aim for the discovery of new classes as early as possible.

4.2 Active Learning In order to improve classification accuracy, ADLNET adopts an empirical risk minimization (ERM) strategy [19] to identify the most informative node in the network to label. Given an unlabeled node $v \in \mathcal{V}^u$ (v is denoted by its feature vector $\mathbf{x} = \hbar(v)$) and a predicted label y using the current ICA classifier, ADLNET trains a new ICA classifier by including $\{\mathbf{x}, y\}$ into the labeled nodes $\mathcal{V}^l$ and estimates

the new classifier's expected classification error. The aim is to select node v, which results in the maximum error reduction, with the expected error of a classifier given as follows

$$\mathcal{E}_p = \int_{\mathbf{x}} \ell(p(y|\mathbf{x}), \hat{p}(y|\mathbf{x}))dp(\mathbf{x}, y), \tag{4.6}$$

where $\ell(\cdot)$ is the loss function that measures the difference between the true distribution $p(y|\mathbf{x})$ and the classifier's prediction $\hat{p}(y|\mathbf{x})$. For the loss function, we use the log loss

$$\ell = \sum_{y \in \mathcal{Y}} p_{\mathcal{V}^l}(y|\mathbf{x}) \log \hat{p}_{\mathcal{V}^l}(y|\mathbf{x}). \tag{4.7}$$

We limit the calculation of expected future error only for labeled data, in our work, because true labels are not known for the rest of the data. We calculate the margin between the true label and the estimated probability for the labeled data based on the current classifier. Thus, we have

$$\mathcal{E}_p = -\frac{1}{|\mathcal{V}^l|} \sum_{v \in \mathcal{V}^l; \mathbf{x}=\hbar(v)} \sum_{y \in \mathcal{Y}} p_{\mathcal{V}^l}(y|\mathbf{x}) \log \hat{p}_{\mathcal{V}^l}(y|\mathbf{x}). \tag{4.8}$$

Given a node v, before we make the query, the true label of v is unknown. However, the current classifier provides an estimate of the distribution from which v's true label would be chosen, and we use it in an expectation calculation to compute the estimated error for each possible label, $y \in \{y_1, \ldots, y_m\}$. Furthermore, we take an average weighted by the current classifier's posterior, $p_{pdp}(y|\mathbf{x})$, which is computed using Eq.(4.3). Therefore, we have

$$\begin{aligned} U_{classification} &= \exp\{-\mathcal{E}_P\} \\ &= \exp\{\frac{1}{|\mathcal{V}^l|} \sum_{y \in \mathcal{Y}} \sum_{v \in \mathcal{V}^l; \mathbf{x}=\hbar(v)} [p_{pdp}(y|\mathbf{x}) \ell(y, \hat{p}(y|\mathbf{x}))]\}. \end{aligned} \tag{4.9}$$

Based on this classification utility, from the unlabeled set $\mathcal{V}^u$ we select the node that maximizes the expected reduction in error for the labeled node set $\mathcal{V}^l$.

4.3 Social Regularization To capture network structure, we also introduce a social regularization term in the utility measure to explicitly model label correlations in the local neighborhood. Intuitively, the most valuable unlabeled node should lie in high-density regions in the network and their predictions should disagree the least with their local neighbors. So we aim to select the node with low variance, and high agreement between its neighbors. Given a node v and its neighbor set N, we define

$$U_{social} = \sum_{j \in N} exp(-D_{KL}(y_i||y_j)), \tag{4.10}$$

where D_{KL} is Kullback-Leibler divergence between the prediction for node i and its neighbors. This utility measures the agreement between $\mathbf{x}$ and its neighbors with respect to their predictions. It implicitly indicates that a node would be favored if its neighbors have similar predictions.

4.4 Efficiency Improvement The ADLNET algorithm finds the most informative node v^* by iterating over all unlabeled nodes $\mathcal{V}^u$ and selects the one with the largest utility value, as shown in Algorithm 1. The empirical risk minimization (ERM) strategy employed in Section 4.2 is the most computationally expensive part that takes $O(Kn^2)$, where n is the number of nodes and K is the number of classes in $\mathcal{Y}$. For large-scale networked data, the algorithm, referred to as ADLNET(ERM), is computationally inefficient.

To improve computational efficiency, we focus on speeding up the ERM process. Similar to [15], we use three graph-based metrics (betweenness centrality, closeness centrality, and clustering coefficients) [25] to limit the search space for ERM. Specifically, betweenness centrality measures the degree of brokerage for each node, that is, how much information is propagated through the node. Closeness centrality measures how close a node is to all other nodes in the network, as defined by the shortest path from the source node to the destination node. Clustering coefficients measure the ratio of the number of links between a node's neighbors to the total possible number of links between the node's neighbors. From a communication flow perspective, nodes having high scores for these measures are important or central nodes in the network. We consider such nodes to also be good candidates to label for active learning. Therefore, instead of iterating over all the unlabeled nodes $\mathcal{V}^u$, ADLNET employs each of these measures to pick up top m instances to form a small candidate set (Δ), and then uses ERM to find the best node among all candidates in Δ (where $\Delta \in \mathcal{V}^u$ and $|\Delta| \ll |\mathcal{V}^u|$). The three graph-based metrics are global measures of the underlying network connectivity that can be computed once before making the query. At each iteration of active learning, it just takes $O(1)$ to retrieve the top m instances with respect to each measure, because the sorted list of nodes do not change. As a result, the computational complexity of ADLNET is reduced to $O(Kmn)$ where $m \ll n$.

5 Experiments

In order to validate the performance of our proposed algorithm, we conducted extensive experiments on two real-world data sets – CiteSeer and Cora [20]. In both data sets, instances correspond to documents, which

Data Set	CiteSeer	Cora
# of Instances	3312	2708
# of Classes	6	7
# of Links	4732	5429
# of Instances of largest Class	701	818
# of Instances of smallest Class	249	180

Table 1: Description of CiteSeer and Cora

are represented by a set of features vectors, and the network structure is provided by the citations between different documents. We ignore the documents' self-citations and the direction of links, thus considering two documents as connected if either of them has cited the other. The detailed description of the two data sets is given in Table 1. Our proposed algorithm is referred to as **ADLNET** in our experiments, which uses the speedup strategy to implement ERM and selects the top three instances for each metric. For comparison, we compare ADLNET with three baseline methods:

- **RAND** uses the same collective classifier as the proposed ADLNET algorithm, but randomly selects instances from the unlabeled set $\mathcal{V}^u$ to label.
- **ALFNET** is a state-of-the-art method to active learning on networked data proposed in [4]. It uses clustering and committee ensemble to choose instances in the network to label. To make fair comparisons, we select only one instance, instead of k instances at each iteration.
- **ADLNET(ERM)** uses the same proposed utility measure in Eq.(4.1) to select nodes for labeling. It differs from ADLNET in that, while computing classification error reduction, it iterates over all the unlabeled instances rather than using the speedup strategy to select instances. We will particularly compare the time efficiency of these two methods.

5.1 Experimental settings All experimental results are based on five times five-fold cross validation. The labels of instances in the training set are unknown until queried. The links of the training data to the test data are removed to avoid test data being queried during training. On both data sets, the feature vectors (keywords) representing the documents are very sparse, so we use principal component analysis (PCA) to transform the original feature space into a lower-dimensional space. For classification, we use logistic regression as the learning method to train the ICA classifier and we use the `proportion`, which is the proportion of each class in the neighbors of `x`, as link features. The ICA classifier is trained on the combined features, including node features (keywords) and link features.

At the beginning of active learning, we randomly choose nodes (less than 1% of training data) from two majority classes to form an initial labeled set. Therefore, the initial class space $\mathcal{Y}$ includes two classes, and during the succeeding active iterations, class space $\mathcal{Y}$ is continuously updated to include new classes. At each iteration, each method selects one node to query its class label by using different query strategies. Once the label is obtained, the labeled set $\mathcal{V}^l$ is updated by including the new labeled node, with a new ICA classifier being retrained and validated on the same test set. When testing the algorithm, the links between the test nodes and the rest of the data are restored. The ICA classifiers generated using different active learning algorithms are applied to the whole training and test data. The classification accuracy is calculated based on the algorithm performance on the held-out test nodes.

Our experiments compare the classification accuracy of different methods with respect to a fixed number of queries. Suppose that each query is subject to the same labeling cost, the number of queries indicates the total cost of an active learning process. A better active learning algorithm is expected to achieve a higher classification accuracy given a same number of queries. In order to compare the performance of different algorithms for new class discovery, we also report the number of classes found during the active learning process for all algorithms.

5.2 Results on CiteSeer In Figure 2, we report average classification accuracy of different algorithms with respect to the number of queries on the CiteSeer data set. We set the two parameters α and β in Eq.(4.1) to 2 and 3, respectively. The results show that ADLNET performs best, and Random is inferior to both ADLNET and ALFNET. At the beginning of the active learning process, ADLNET achieves a much higher accuracy than ALFNET because it discovers new classes earlier than ALFNET, which leads to a significant performance gain in classification accuracy. After about 80 queries, the accuracy of the two algorithms gets very close because the class space has been fully explored. Overall, ADLNET can be observed to outperform all baseline methods.

Figure 3 compares different active learning algorithms in terms of their ability to discovery new classes. We can see that, ADLNET performs most effectively to discover new classes and it is the first to find all the classes from all three algorithms.

We also compare the performance of two proposed algorithms, ADLNET and ADLNET(ERM). ADL-

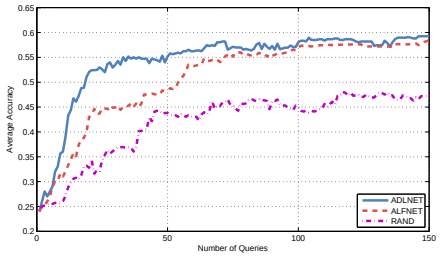


Figure 2: Average classification accuracy on CiteSeer

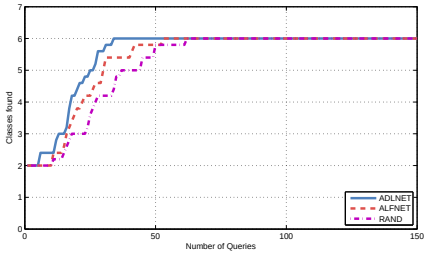
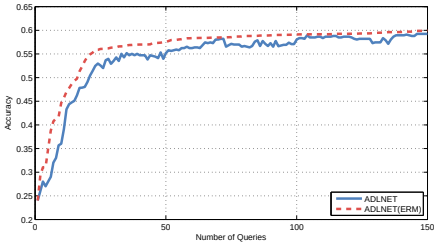
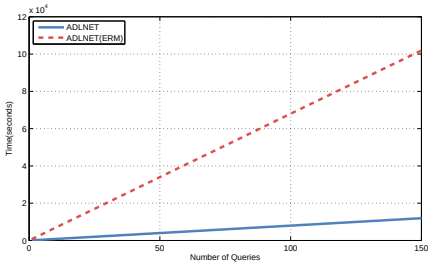


Figure 3: Average number of classes found on CiteSeer



(a) Classification accuracy



(b) Time efficiency

Figure 4: Comparison of ADLNET and ADLNET(ERM) on CiteSeer

NET(ERM) needs to iterate over all the unlabeled data, while ADLNET uses three graph-based metrics to speed up the ERM process. Figure 4(a) and Figure 4(b) compare the classification accuracy and time efficiency of the two algorithms. We observe that both methods have comparable accuracy with ADLNET(ERM) having a slightly higher accuracy than ADLNET. However, using the speedup strategy, ADLNET significantly improves the time efficiency of ADLNET(ERM).

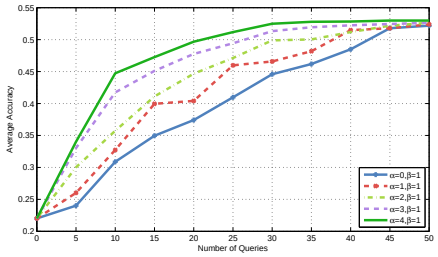
Experiments were also performed to investigate how the parameters α and β (as defined in Eq.(4.1)) would affect classification accuracy of our ADLNET algorithm. Figure 5(a) shows classification accuracy by varying the values of α. We can observe that, at the beginning, as the value of α increases, ADLNET achieves higher accuracy. This is because using a larger value of α places more emphasis on discovering new classes, which results in better classification performance. Figure 5(b) shows classification accuracy using different values of β given a fixed value of α. As we can see, ADLNET achieves the highest accuracy when β is set to be 3. Overall, when β is set to be 2, 3, or 4, the accuracy of ADLNET is very close. This indicates that, social regularization helps to improve the classification accuracy when its value is properly set.

5.3 Results on Cora In the second set of experiments, we compared the performance of different algorithms on the Cora data set. Figure 6 shows average classification accuracy with respect to the number of queries. The two parameters α and β were set to 2 and 3, respectively. We can see that ADLNET outperforms the other two baselines, in particular at the beginning of the query process. Again, it indicates that, ADLNET has the advantage of discovering new classes in the early stage of the active learning process.

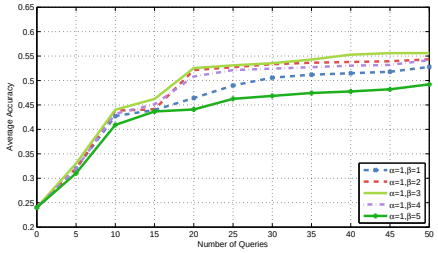
Figure 7 compares the average number of classes found by different algorithms. As we can see, ADLNET outperforms other baselines to quickly find all the classes, while the performance of ALFNET and RAND are very close to each other in terms of their ability for new class discovery.

In Figure 8(a) and Figure 8(b), we compare the performance of ADLNET and ADLNET(ERM) on the Cora data set. The results show that ADLNET and ADLNET(ERM) have comparable classification accuracy. However, ADLNET takes eight times less processing time than ADLNET(ERM).

In Figure 9(a) and Figure 9(b), we study the impact of ADLNET's two parameters α and β on the classification accuracy. The results confirm that a larger α value is more effective to discover new classes at the early stage of the active learning process. Moreover, when β is set to 3, ADLNET achieves superior accuracy by using social regularization to capture label correlations in the network.



(a) Comparison between different α values



(b) Comparison between different β values

Figure 5: Average accuracy for different α and β values

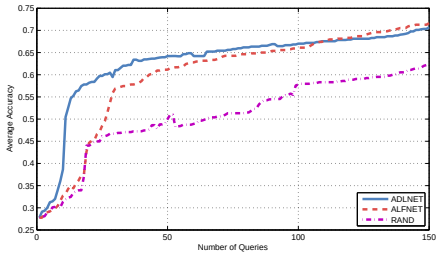


Figure 6: Average classification accuracy on Cora

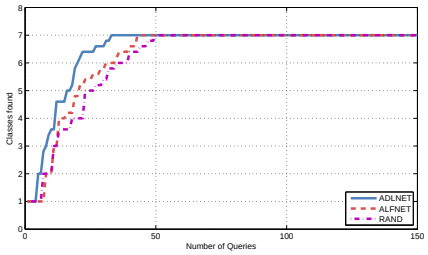
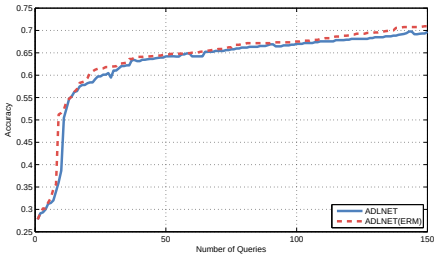
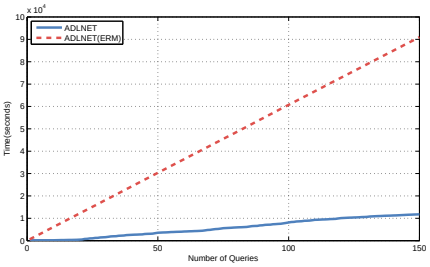


Figure 7: Average number of classes found on Cora



(a) Classification accuracy



(b) Time efficiency

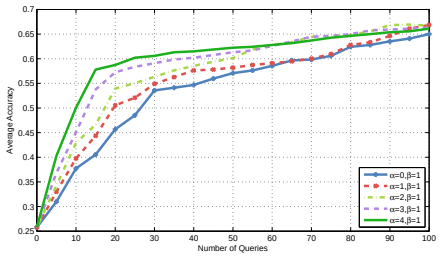
Figure 8: Comparison of ADLNET and ADLNET(ERM) on Cora

6 Conclusion and Future Work

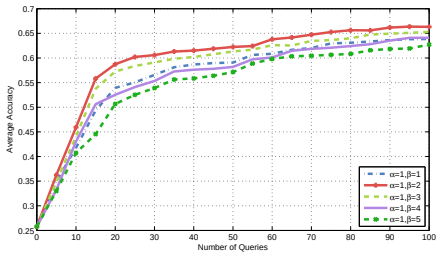
This paper has proposed a new framework that combines new classes discovery and active learning for networked data. Our proposed ADLNET algorithm uses a Dirichlet process prior to model class distributions for both known and unknown classes, through which ADLNET has the ability to actively discover and update the class space. To select important nodes for labeling, ADLNET chooses the node that minimizes the expected error of the collective classifiers trained from the benchmark set. Label correlations are also modeled as social regularization in the overall utility calculation for effective active learning. Experimental results demonstrate that, as compared to other state-of-the-art methods, the proposed ADLNET algorithm discovers new classes more effectively and results in higher accuracy for classifying networked data.

This research can be extended in several directions. First, we will investigate how to improve the computational efficiency of the ADLNET algorithm in large-scale network data. Second, we will explore different strategies to model social regularization as a way to capture label correlations in the networked data. Finally, we will extend this work to stream-based data environments, where nodes and links in the network may dynamically change over time.

Acknowledgement This research is supported by an Australian Research Council (ARC) Future Fellowship under Grant No. FT100100971 and a supplementary postgraduate scholarship from CSIRO.



(a) Comparison between different α values



(b) Comparison between different β values

Figure 9: Average accuracy for different α and β values

References

[1] A. Ahmed, Q. Ho, C. Teo, J. Eisenstein, A. Smola, and E. Xing. Online inference for the infinite topic-cluster model: Storylines from streaming text. In *Proc. of AISTATS*, pages 101–109, 2011.

[2] J. Baldridge and M. Osborne. Active learning and the total cost of annotation. In *Proc. of EMNLP*, pages 9–16, 2004.

[3] M. Bilgic and L. Getoor. Effective label acquisition for collective classification. In *Proc. of SIGKDD*, pages 43–51, 2008.

[4] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *Proc. of ICML*, pages 79–86, 2010.

[5] W. Buntine and M. Hutter. A Bayesian view of the poisson-dirichlet process. *Computing Research Repository*, abs/1007.0296, 2010.

[6] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Active learning on trees and graphs. In *Proc. of COLT*, pages 320–332, 2010.

[7] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *J. ARTIF. INTELL. RES.*, 4:129–145, 1996.

[8] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *MACH. LEARN.*, 28:133–168, 1997.

[9] T. Hospedales, S. Gong, and T. Xiang. Finding rare classes: active learning with generative and discriminative models. *TKDE*, 25(2):374–386, 2013.

[10] A. Joshi, F. Porikli, and N. Papanikolopoulos. Scalable active learning for multi-class image classification. *IEEE PAMI*, 34(11):2259–2273, 2012.

[11] A. Kuwadekar and J. Neville. Relational active learning for joint collective classification models. In *Proc. of ICML*, pages 385–392, 2011.

[12] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. of SIGIR*, pages 3–12, 1994.

[13] C. Loy, T. Hospedales, T. Xiang, and S. Gong. Stream-based joint exploration-exploitation active learning. In *Proc. of CVPR*, pages 1560–1567, 2012.

[14] D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4:590–604, 1992.

[15] S. Macskassy. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Proc. of SIGKDD*, pages 597–606, 2009.

[16] M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Classification and novel class detection in data streams with active mining. In *Proc. of PAKDD*, pages 311–324, 2010.

[17] P. Melville and R. Mooney. Diverse ensembles for active learning. In *Proc. of ICML*, pages 584–591, 2004.

[18] C. Rasmussen. The infinite gaussian mixture model. In *Proc. of NIPS*, pages 554–560, 2000.

[19] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. of ICML*, pages 441–448, 2001.

[20] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[21] B. Settles. Active learning literature survey. *University of Wisconsin–Madison*, 2010.

[22] L. Shi, Y. Zhao, and J. Tang. Batch mode active learning for networked data. *ACM TIST*, 3(2):1–25, 2012.

[23] Y. Teh. A Bayesian interpretation of interpolated kneser-ney. 2006.

[24] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, 2002.

[25] S. Wasserman and K. Faust. *Social Network Analysis*. Cambridge University Press, 1995.

[26] J. Zhang, Z. Ghalramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *NIPS*, volume 17, 2004.