Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2019-4747887

M.Sc. Thesis

Active Semi-Supervised Learning For Diffusions on Graphs

Bishwadeep Das

Abstract

In statistical learning over large data-sets, labeling all points is expensive and time-consuming. Semi-supervised classification allows learning with very few labels. Naturally, selecting a few points to label becomes crucial as the performance relies heavily on the labeled points. The motivation behind active learning is to build an optimal training set keeping the classifier in mind. Random or heuristic-driven selection does not care for the classification process or are trivially defined. We are interested in the graph structure formed by the data, as seen in citation, social and biological networks. Accordingly, active semi-supervised learning on graphs labels nodes to enhance the performance of classification. We propose a new methodology to perform active learning for diffusion-based semi-supervised classifiers. In particular, we focus on a classifier which diffuses probability distributions over the graph through random walks. We postulate the active learning problem as *i*) a linear inverse problem with a sparse starting distribution over the nodes; *ii*) a model output selection problem. For the former, we use sparsity-regularized inverse problems to select nodes. For the latter, we use tools from Compressed Sensing and Sparse Sensing to select the nodes with the relevant model output. We show that we can select all the relevant nodes in a single shot fashion, hence avoiding reliance on multiple training phases. Results on simulated as well as real data-sets show the proposed methods outperform random labeling, thereby proving to be relevant for active semi-supervised learning on graphs.

Active Semi-Supervised Learning For Diffusions on Graphs

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Bishwadeep Das
born in Calcutta, India

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Active Semi-Supervised Learning For Diffusions on Graphs"** by **Bishwadeep Das** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 27th November, 2019

Chairman: _____
prof.dr.ir. G.J.T Leus

Advisor: _____
prof.dr.ir. G.J.T Leus

Committee Members: _____
dr.ir. Elvin Isufi

_____
dr.ir. David Tax

Abstract

In statistical learning over large data-sets, labeling all points is expensive and time-consuming. Semi-supervised classification allows learning with very few labels. Naturally, selecting a few points to label becomes crucial as the performance relies heavily on the labeled points. The motivation behind active learning is to build an optimal training set keeping the classifier in mind. Random or heuristic-driven selection does not care for the classification process or are trivially defined. We are interested in the graph structure formed by the data, as seen in citation, social and biological networks. Accordingly, active semi-supervised learning on graphs labels nodes to enhance the performance of classification. We propose a new methodology to perform active learning for diffusion-based semi-supervised classifiers. In particular, we focus on a classifier which diffuses probability distributions over the graph through random walks. We postulate the active learning problem as *i*) a linear inverse problem with a sparse starting distribution over the nodes; *ii*) a model output selection problem. For the former, we use sparsity-regularized inverse problems to select nodes. For the latter, we use tools from Compressed Sensing and Sparse Sensing to select the nodes with the relevant model output. We show that we can select all the relevant nodes in a single shot fashion, hence avoiding reliance on multiple training phases. Results on simulated as well as real data-sets show the proposed methods outperform random labeling, thereby proving to be relevant for active semi-supervised learning on graphs.

Acknowledgments

First of all, I would like to thank Geert for giving his time and energy to supervise this thesis. Each of my meetings with you has been a stepping stone for this work. I am thankful for the insightful and thought-provoking criticism and also for providing the platform for me to work with my ideas. Next, I would like to thank Elvin for his overall guidance, keeping me on track and patience in teaching me so much about the finer aspects of technical writing and offering different perspectives behind the same. This thesis is a lot more readable due to his feedback. I also acknowledge David Tax for agreeing to be on my committee and adjusting his schedule to make room for my defence. Also, I would like to thank my colleagues with whom I shared the Master's study room for more than a year. Joris, Pranav, Sherine and Bram, I thank you for the enriching discussions on society, culture and politics in addition to technical discussions. I will also like to thank the PhD students on the 17th floor for being helpful and offering their advice whenever I needed it. I would also like to thank my friends in Delft, for supporting me and providing inspiration. I would also like to acknowledge Arséne Wenger, whose vision and stubborn idealism has stuck throughout, and the various musicians whose works have been an inseparable part of me throughout my Masters. Last, but not least, I thank my parents for their support, patience and love. Without their presence, I would not be here in the first place.

Bishwadeep Das
Delft, The Netherlands
27th November, 2019

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Nowadays, we are witnessing an increase in the volume and diversity of data-sets. Examples can be found in social [1], biological [2] [3] and citation networks [4], to name a few. When we think of network data, we consider a network defined through the relationship between data points. The network can also be seen as a graph with nodes (vertices) representing the data points. For example, in a weather monitoring network, each station can be considered to be a node and the data its measurements. Two stations located close to each other are connected through edges, where the importance of each edge depends on the distance between the stations that share it. There are several advantages of this representation. First, the data now resides on a lower-dimensional manifold [5]. Secondly, existing research on spectral graph theory can be leveraged to process data [6]. Third, the graph structure allows for distributed processing [7]. The availability of graph (network) data and the numerous advantages offered by graphs signify these data processing tasks. One such task is making an inference about the entire network while observing information over a very small portion of it. To be more specific, for each node of the network we would like to infer about its properties or predict some value associated with it. But this is a stumbling block for networks often due to their huge size and irregular structure. So we rely on selecting a small set of nodes, whose information makes the aforementioned task easier.

We focus on classifying the nodes of a graph. Given a graph data-set, the nodes are the objects and have their respective labels. We want to predict the labels of all nodes. As an example, we consider a social network such as Facebook, with its users as nodes and edges capturing friendships [1]. We want to classify the users according to their political affiliation (i.e. Republican or Democrat). Classification requires the presence of labeled information (data). For our Facebook graph, this requires the affiliation of all users which is difficult to acquire due to the size of such a network and the difficulty involved in acquiring it. The natural choice is to focus on training with labeled data which is much smaller in number than the unlabeled data. This does not bode well for supervised learning, as it can learn only from labeled data and generally performs better with more labeled data. Semi-supervised learning is the way to proceed [8]. It learns a classifier from both labeled and unlabeled data. The goal is to incorporate the unlabeled data and the structure of the overall data to aid the learner and perform better than using the labeled data alone. For our example network, this means we need the affiliation of only a few users and use that information to predict those of the rest. To ensure a good classification performance, one approach is to build the optimal set of points to be labeled (queried) from the available data. The field of active learning looks

into this particular problem [9]. This means for the Facebook network, we build the optimal set of users using an active learning approach. Next, we obtain their affiliations and then predict the labels of the other users.

1.2 Current Literature

Active learning and experimental design (optimal selection or allocation of objects to facilitate experiments) are well connected, and this is evident from the literature on this topic [10] [11] [12] [13] [14] [15]. Scholars have looked into this for classifiers like Gaussian random fields on graphs [10] [16] [17], graph Laplacian regularized least squares [11] [18], the logistic classifier [13] and ridge regression [12]. In terms of the querying criterion (a function which evaluates how optimal a point or a set of points are), the most informative points can be selected based on uncertainty [19] [20], smoothness [21] [22], experimental design [11] [12], error bounds [18] [14] and node density [23]. The literature also contains methods which consider training over multiple batches [16] or just one batch [10].

However, little is known about active learning for label propagation on graphs, a popular semi-supervised learning technique. The work in [19] combines different criteria but assumes the learning and selection processes to be independent while the work in [20] depends solely on the entropy. The work in [23] uses node densities to select nodes and then uses semi-supervised label propagation to detect communities on graphs. These methods are based on heuristics. Incorporating active learning into the classification model can increase the quality of the selected samples as well as the classification accuracy. There exists a semi-supervised classifier which is adaptive to each class in the graph and uses a finite number of random walks to propagate labels [24]. Some of the works which use experimental design are aimed primarily at regularized regression (which is conceptualized in the Euclidean space) and impose smoothness constraints on the classifier.

1.3 Research Question and Proposed Approach

To bring closer active learning on graphs with adaptive diffusion, this thesis concerns the research question:

Given an unlabeled data-set and a graph built from it, what is the set of nodes that should be queried jointly so that an adaptive diffusion-based semi-supervised classifier can perform optimally?

We will answer the above question for the semi-supervised classifier on graphs discussed in [24]. By accounting for the classifier model (linear label propagation/ diffusion), we formulate the problem of building the optimal set of nodes in two different ways. In the first approach, we assume it to be the solution of a regularized inverse

problem. The second approach is based on model output selection (i.e. selecting the nodes with the most relevant outputs). We will select the nodes jointly instead of multiple batches involving repeated training and we will motivate why that is possible. We will evaluate their performance and relative strengths/ weaknesses.

We adopt two sets of approaches to solve this problem. In the first, we try to solve the inverse problem directly to find the starting distribution of the diffusion by relying on some proxies of the landing probabilities as defined in [24]. These are obtained through a fuzzy clustering of the nodes and are used as observations to solve for the starting densities. Once they are recovered, we query the labels at the non-zero locations and proceed with semi-supervised learning [24]. In the second approach, we do not bother with obtaining proxies or the nature of the starting density. We select the nodes whose observations (i.e. outputs observed at those nodes) are the most important for estimating the starting density. We use tools from Sparse Sensing and Compressed Sensing to come up with two different methods to tackle this problem. For sparse sensing, we rely on experimental design while for compressed sensing we rely on real equiangular frames [25].

Our contributions are as follows: i) we postulate the problem of a single pool-based active semi-supervised learning for adaptive diffusion on graphs as $i.i$) one based on model output selection and $i.ii$) one based on solving a sparse inverse problem with estimated proxies; ii) the model output selection has been tied to Compressed Sensing [26] and Sparse Sensing [27], with each approach imposing different priors on the labeled nodes. Numerical results on simulated and real graphs corroborate our findings and showcase the improved performance compared with the state of the art.

1.4 Notation and Outline

In this thesis, we will adopt the following notation. A scalar variable is represented by a plain alphabet a or A while a vector is represented by a bold alphabet like $\mathbf{a}$. A matrix is presented as a bold upper-case alphabet $\mathbf{A}$. Sets are denoted by calligraphic alphabets like $\mathcal{S}$ and their cardinality is denoted as $|\mathcal{S}|$. Referring the ith element of a vector $\mathbf{a}$ is done as a_i or $[\mathbf{a}]_i$ while an element in the ith row and jth column of a matrix is indicated as A_{ij} or $[\mathbf{A}]_{ij}$. The matrix obtained by extracting the rows and columns of $\mathbf{A}$ indexed by the set $\mathcal{S}$ is $\mathbf{A}_{\mathcal{S}\times\mathcal{S}}$. The diagonal matrix with its diagonal elements $\mathbf{a} = [a_1, \ldots, a_N]$ is denoted by $\text{diag}(\mathbf{a})$. Likewise, $\mathbf{a} = \text{diag}(\mathbf{A})$ is the vector comprised of the diagonal elements of a square matrix $\mathbf{A}$. The trace of a matrix $\mathbf{A}$ is denoted as $\text{tr}(\mathbf{A})$. Numbers in bold case, like $\mathbf{1}_N$ and $\mathbf{0}_N$ denote the N-dimensional vectors of all ones and all zeros, respectively. Matrices $\mathbf{A}^H$ and $\mathbf{A}^T$ represent the Hermitian and transpose of matrix $\mathbf{A}$, respectively. Matrices $\mathbf{A}^{-1}$ and $\mathbf{A}^\dagger$ denote the matrix inverse and pseudo-inverse of $\mathbf{A}$. The expectation operator is $\mathbb{E}(\cdot)$. The symbol $\mathbb{R}^N$ represents the space of N dimensional real vectors, $\mathbb{R}^{M\times N}$ that of $M \times N$ real matrices and $\mathbb{S}^{N\times N}$ represents the space of $N \times N$ symmetric matrices. The p-norm of a vector $\mathbf{a}$ is referred as $||\mathbf{a}||_p$ and $||\mathbf{a}||^2_{\mathbf{A}} = \mathbf{a}^T\mathbf{A}\mathbf{a}$ is the weighted 2-norm of $\mathbf{a}$ with respect to matrix $\mathbf{A}$. The matrix or spectral norm of $\mathbf{A}$ is denoted by $||\mathbf{A}||$. The sign operator is $\text{sign}(\cdot)$. The

updated value of $\mathbf{x}$ after the kth step in an iterative algorithm is $\mathbf{x}^{(k)}$. The parentheses $\{\cdot\}$ are usually used for elements of a set while the bracket symbol $[\cdot]$ is used to denote vectors or matrices.

The outline of this thesis is as follows: Chapter 1 is the introduction; Chapter 2 deals with the background information; Chapter 3 reviews the existing literature; Chapter 4 introduces the diffusion-based adaptive semi-supervised learner; Chapter 5 describes the two sets of approaches we propose in this thesis; Chapter 6 shows the results obtained on some standard synthetic graphs and real-world data-sets and offers a deeper understanding of the methods; Chapter 7 discusses our findings, conclusions, limitations and the scope of future work.

2 Background

This chapter introduces the background material which will be required for understanding the material in the rest of the thesis. The terminology, theory, problems and algorithms presented in this chapter will often be referred to in the upcoming chapters. This chapter is organized as follows: Section 2.1 is a brief introduction on Active Learning and its types; Section 2.2 introduces the Sparse Sensing framework for Statistical Inference; Section 2.3 introduces Graph Signal Processing, the field of processing data observed over graphs; Section 2.4 discusses the representation and clustering of the nodes of a graph; Section 2.5 highlights the important terms, concepts and algorithms of the Compressed Sensing framework; Section 2.6 concludes the chapter by highlighting the key concepts and ideas from each section which will be combined in the thesis.

2.1 Active Learning

Active learning is a branch of machine learning where the learner can select the data it wants to learn from [9]. In supervised learning, the classifier can work only on labeled data. Obtaining labels is often expensive and time-consuming as it usually involves a human expert. For instance, downloading academic papers is practically free but classifying them according to their field of study would need an expert to read and label each paper. In such a situation, we would like to use a learning mechanism that gives a good performance but for a small number of labeled objects. Selecting which data to label becomes the central topic of active learning. Figure 2.1 shows the general active learning framework. This framework is composed of four modules and follows four stages indicated by the numbers over the arrows.

1. There is usually a set of labeled data $\mathcal{L}$ and a set of unlabeled data $\mathcal{U}$ at the start of the process. A machine learning model or classifier is trained over $\mathcal{L}$ and sometimes $\mathcal{L}$ and $\mathcal{U}$ in the case of semi-supervised learning.

2. The learner "searches" over $\mathcal{U}$ and selects objects to query which are optimal for a particular criterion. There are different criteria to select from. Some commonly used criteria include uncertainty, expected model change [28] and expected generalization error [16].

3. The selected objects are sent to an oracle or expert to provide labels. This is called the querying stage.

4. The queried objects now have a label and are incorporated into the labeled set $\mathcal{L}$. The learner can now operate on this set, thereby completing one cycle of active

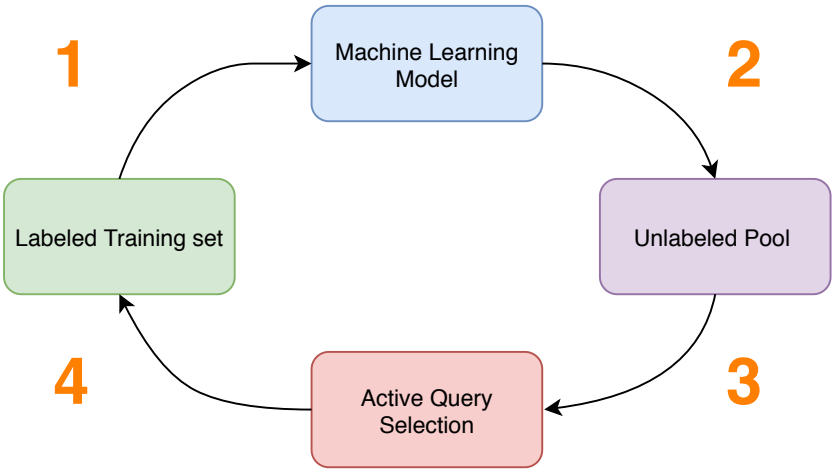


Figure 2.1: General Active Learning Framework. There are four modules: namely the Labeled Training Set, a Machine Learning Model, a pool of Unlabeled data and an Oracle. In step 1, the learner trains itself on the labeled set; in step 2 the optimal set of objects are selected from the unlabeled data; in step 3, these objects are sent to an oracle to be queried; in step 4, after querying, they are added to the labeled set.

learning. This is repeated until the desired performance criterion is met or until the required number of training samples is selected.

Active learning approaches can be classified based on the scenario and/ or the querying criterion. A particular criterion can be combined with a particular scenario depending on the problem.

2.1.1 Classification based on Scenario

The following are the three scenarios used to classify active learning:

2.1.1.1 Membership Query Synthesis

The learner queries any unlabeled data given as an input [29] [30]. The queries are made to identify a concept. As a result, arbitrary and non-informative points can be labelled. This can work in some scenarios but its arbitrariness and tendency to pick non-informative points renders it unappealing for more difficult problems.

2.1.1.2 Stream-based Selective Sampling

As the name suggests, this active learning method deals with a stream of data [31]. There is a source from which all unlabeled instances are generated. The learner gets access to each instance sequentially and has to decide for each. If chosen to be queried, the object is sent to the expert who provides the label and updates the labeled set. This approach is also called sequential active learning. The availability of one point at

a time can be thought of as a uniform sampling over the instance set, which has its underlying distribution.

2.1.1.3 Pool-based Learning

In pool-based active learning, the main assumption is the existence of a large pool of unlabeled data $\tilde{\mathcal{U}} \subseteq \mathcal{U}$. The difference between this and stream-based selective sampling is that in stream-based selective sampling, the learner has access to only one unlabeled object at a time whereas now the learner has access to a pool of unlabeled data. This opens up new approaches to select the desired points. The learner can also query up to several points and send them to the expert. After updating the learner, another batch of unlabeled points can be requested again, and the process repeats. Depending on the feasibility of the scenario, some learners depend only on one batch [18],[22] while others consider more than one batch [16], [19]. Reliance on one batch can be due to a slow training procedure, limited waiting time or the nature of the querying criterion. By having access to more points, we can make use of the advantages provided by unlabeled data and build more sophisticated decision models. This is the go-to scenario for our problem. However, selecting all the points depending on some criterion in one go has its drawbacks. Selecting the K-best points based on informativeness, for instance, may not be the best for classification as there may be correlated information between them. Instead, some diversity between the selected points could also be considered.

2.1.2 Classification based on Querying Criterion

At the heart of active learning approaches is the evaluation of a certain object (or a set of objects) on how optimal it is (they are) for being labeled. This evaluation is carried out by the querying criterion. As explained in [9], some categories used to measure optimality are as follows: Uncertainty Sampling, Query by Committee, Expected Model Change and Estimated Error Reduction.

2.1.2.1 Uncertainty Sampling

In the uncertainty sampling based approach, the learner queries the most informative points [32]. The entropy of a random variable is a measure of its uncertainty [33]; higher entropy implies more uncertainty. For an unlabeled object $\mathbf{x}$ in a C-class classification problem, the posterior probabilities $p(y_1|\mathbf{x};\boldsymbol{\theta}),\ldots,p(y_C|\mathbf{x};\boldsymbol{\theta})$ form a distribution, where $\{y_1,\ldots,y_C\}$ is the set of labels and $\boldsymbol{\theta}$ denotes the parameters of the classifier. The entropy of such a distribution is

$$h(\mathbf{x}) = -\sum_{c=1}^{C} p(y_c|\mathbf{x};\boldsymbol{\theta})\log\big(p(y_c|\mathbf{x};\boldsymbol{\theta})\big). \tag{2.1}$$

A higher value of entropy means that the classifier is unsure as to which class this object belongs and this makes it a difficult point to classify. This measure can be used for classifiers which obtain the exact posterior probabilities like the logistic regression classifier. This is also suitable for classifiers where the posterior probability can be

approximated. A related measure is the least confident criterion [34]. The object $\mathbf{x}$ in the unlabeled pool $\tilde{\mathcal{U}}$, that has the least maximum value of posterior probability for any of the C classes is chosen.

2.1.2.2 Query by Committee

In query by committee, a committee of classifiers is formed and their predictions are used together to select the most informative objects [35]. Let $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ denote the set of K classifiers or experts sampled from the hypothesis space $\mathcal{H}(\boldsymbol{\theta})$ which are consistent with the labeled data $\mathcal{L}$. This set is also called the version space. The aim of Query by Committee is to reduce the version space, by selecting points from the unlabeled set $\mathcal{U}$ which reduces the version space. As Settles [9] puts it,

> *"If we view machine learning as a search for the best model within the version space, then our goal in active learning is to constrain the size of this space as much as possible (so that the search can be more precise) with as few labeled instances as possible."*

One way to reduce the version space is to train each of its members on $\mathcal{L}$, and then predict the labels on $\mathcal{U}$. The vote-based entropy criterion

$$(\mathbf{x}^*) = \underset{\mathbf{x}\in\mathcal{U}}{\operatorname{argmax}} - \sum_{c=1}^{C} \frac{V_c}{K} \log \frac{V_c}{K} \tag{2.2}$$

is used to select the most informative objects [36]. The variable V_c refers to the number of committee members who predicted that object $\mathbf{x}$ belongs to class c. This is a form of the entropy measurement defined for the distribution over committee votes. There are other criteria as well, like the one based on the average Kullback-Leibler Divergence [37] and the one on support vector classifiers [38].

2.1.2.3 Expected Model Change

This querying strategy selects those points that influence the model to change the most. For instance, one way to select an optimal point is the one, which when added to $\mathcal{L}$ increases the length of the gradient of the cost function (associated with the model parameters of the classifier) the most [28]. The selection is represented as

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{U}}{\operatorname{argmax}} \sum_{c=1}^{C} P_{\boldsymbol{\theta}}(y_c|\mathbf{x};\boldsymbol{\theta}) ||\nabla l_{\boldsymbol{\theta}}(\mathcal{L} \cup \{\mathbf{x}, y_c\})|| \tag{2.3}$$

where $\mathbf{x}^*$ is the optimal point chosen. $||\nabla l_{\boldsymbol{\theta}}(\mathcal{L} \cup \{\mathbf{x}, y_c\})||$ is the magnitude of the derivative of some function l taken with respect to the classifier parameter $\boldsymbol{\theta}$ and evaluated for the training set $\{\mathcal{L} \cup \{\mathbf{x}, y_c\}\}$. The length of the gradient is averaged across all possible labels to give the expected gradient length. The point $\mathbf{x}$ which maximizes the change in gradient length, and hence that of the model, is added to $\mathcal{L}$. This is one example of selection through model change.

2.1.2.4 Estimated Error reduction

In this approach, the future expected generalization error is calculated for an object $\mathbf{x}$ when it is added to the labeled set $\mathcal{L}$. This is done for all $\mathbf{x} \in \mathcal{U}$ and the one that leads to the minimum error is selected. This is a computationally demanding approach because for each point $\mathbf{x}$, the model has to be trained on $\mathcal{L} \cup \{\mathbf{x}\}$ to get the future predictions and this is repeated for all possible labels of $\mathbf{x}$. The work in [16] uses the expected classification error as the querying criterion to actively select unlabeled nodes for a semi-supervised learning task on graphs.

2.2 Sparse Sensing for Statistical Inference

Sparse sensing for inference involves sampling from a set of observations for estimating a parameter of interest. In this section, we will deal with the estimation of a deterministic (non-random) parameter from its observations for a linear model. Section 2.2.1 discusses the model under these assumptions and the performance metrics; Sections 2.2.2 and 2.2.3 discuss the A and D-experimental design approach for sparse sensing, respectively.

2.2.1 Model

In statistical estimation theory, we are interested in estimating parameters, given observations which are linked to the parameters through a model [39]. The linear measurement model is

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{2.4}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the observation vector, $\mathbf{x} \in \mathbb{R}^N$ is the deterministic parameter vector of interest with usually $M < N$ and $\mathbf{A} \in \mathbb{R}^{M \times N}$ is the $M \times N$ system matrix. The vector $\mathbf{n} \in \mathbb{R}^M$ is the additive noise, commonly assumed to be Gaussian and white. Each element of $\mathbf{y}$ is a random variable due to the influence of noise. The estimate of $\mathbf{x}$ is $\hat{\mathbf{x}} = g(\mathbf{y})$. The function $g(\cdot)$ is the estimator. The estimate also is a random variable due to its dependence on $\mathbf{y}$, with mean $\mathbb{E}(\hat{\mathbf{x}})$ and covariance matrix $\mathbf{\Sigma}_{\hat{\mathbf{x}}}$. It is now important to introduce two quantities to evaluate the estimation. The first is the mean square error (MSE) in $\mathbf{x}$, denoted as

$$\text{MSE} = \mathbb{E}||\mathbf{x} - \hat{\mathbf{x}}||^2. \tag{2.5}$$

The other quantity is the trace of the error covariance matrix $\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}}$. It is defined as

$$\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}} = \mathbb{E}\big((\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T\big). \tag{2.6}$$

An important relation linking the two is

$$\begin{aligned}\mathrm{tr}(\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}}) =& \sum_{i=1}^{N} \mathbb{E}(\hat{x}_i - x_i)^2 \\ =& \mathbb{E} \sum_{i=1}^{N} (\hat{x}_i - x_i)^2 \\ =& \mathbb{E}||\hat{\mathbf{x}} - \mathbf{x}||^2 \\ =& \text{MSE}.\end{aligned} \tag{2.7}$$

The trace of the error covariance matrix is the same as the mean square error. In the case of an unbiased estimator [39], we have $\mathbb{E}(\hat{\mathbf{x}}) = \mathbf{x}$ and the error covariance matrix is the same as the covariance matrix of $\hat{\mathbf{x}}$. To minimize the MSE, we have to obtain the error covariance matrix in terms of the system matrix $\mathbf{A}$. The vector Cramer-Rao Lower bound theorem [39] states that for a vector parameter $\mathbf{x}$, its covariance matrix for an unbiased estimator $\hat{\mathbf{x}}$ follows

$$\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}} \geq \mathbf{I}_{\mathbf{x}}^{-1}, \tag{2.8}$$

where $\mathbf{I}_{\mathbf{x}}$ is the Fisher Information Matrix (FIM) of $\mathbf{x}$ [39]. The probability of observing $\mathbf{y}$, parameterized by $\mathbf{x}$ is $p(\mathbf{y}; \mathbf{x})$. Mathematically, the FIM in defined as

$$\mathbf{I}_{\mathbf{x}} = \mathbb{E}\left[\left(\frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial \mathbf{x}} \right) \left(\frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial \mathbf{x}} \right)^T \right], \tag{2.9}$$

where

$$\frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial \mathbf{x}} = \left[\frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial x_1}, \ldots, \frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial x_N} \right]^T,$$

with $\ln(p(\mathbf{y};\mathbf{x}))$ the log-likelihood of the probability. The equality in (2.8) is achieved when the Minimum Variance Unbiased Estimator (MVUE) exists for $\mathbf{x}$. The mathematical condition for it to exist [39] is

$$\frac{\partial \ln(p(\mathbf{y};\mathbf{x}))}{\partial \mathbf{x}} = \mathbf{I}_{\mathbf{x}}\big(g(\mathbf{y}) - \mathbf{x}\big). \tag{2.10}$$

In such an event, g($\mathbf{y}$) is the MVUE of $\mathbf{x}$. When the noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, the MVUE for $\mathbf{x}$ exists and the FIM is given by

$$\mathbf{I}_{\mathbf{x}} = \frac{1}{\sigma^2}\mathbf{A}^T\mathbf{A} = \sum_{i=1}^{M} \frac{1}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T, \tag{2.11}$$

where $\mathbf{a}_i$ denotes the ith row of $\mathbf{A}$. Therefore the MSE for this model is

$$\text{MSE} = \mathrm{tr}(\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}}) = \mathrm{tr}(\mathbf{I}_{\mathbf{x}}^{-1}) = \mathrm{tr}\left(\sum_{i=1}^{M} \frac{1}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T \right)^{-1}. \tag{2.12}$$

Sparse sensing selectively samples a fixed number of observations from $\mathbf{y}$ in order to ensure that the MSE or a related measure (concerning the quality of estimation of $\mathbf{x}$) is optimized. There could be realistic constraints of several kinds (expert involvement, difficulty in labeling, budget constraints) which restrict us with the option to choose only a few observations. Applications of sparse sensing include sensor selection [27], sensor placement [40] and as we will explore in this thesis, active semi-supervised learning on graphs.

Let us assume we need to select R of the M observations. We take the sensing vector $\mathbf{w}$ with $w_i = 1$ if the ith observation y_i is selected and $w_i = 0$ if y_i is not selected. We also associate a selection matrix $\mathbf{\Phi}(\mathbf{w}) \in \{0,1\}^{\{R\times M\}}$ of size $R \times M$. Each row of $\mathbf{\Phi}(\mathbf{w})$ has exactly one element equal to one and the rest are zero. The index corresponding to 1 in the k-th row of $\mathbf{\Phi}(\mathbf{w})$ is the index of $\mathbf{w}$ where the kth 1 appears. The sensed measurement vector is $\mathbf{z} = \mathbf{\Phi}(\mathbf{w})\mathbf{y} = \mathbf{\Phi}(\mathbf{w})(\mathbf{A}\mathbf{x} + \mathbf{n})$.

In the context of the background provided in the previous section, we now look into the estimation problem and MSE in terms of the sensing vector $\mathbf{w}$. The FIM for the model in (2.4) and for $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_M)$ is given as [41]

$$\mathbf{I}_\mathbf{x} = \sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T, \tag{2.13}$$

where w_i is one for y_i being selected and is zero otherwise. In (2.13), the summation is possible because of the white noise; all observations are mutually uncorrelated and the FIM for each observation adds up. If an observation is not selected, it provides no information.

2.2.2 Sparse Sensing with A-experimental Design

We know from Section 2.2.1 that the MSE is equal to the trace of the inverse of the FIM (given the Minimum Variance Unbiased Estimate (MVUE) exists [39]). In the case of additive white Gaussian Noise in a linear observation model, it can be shown that the MVUE does exist [39]. This allows us to work with the MSE directly and to select R observations that leads to the minimum MSE. The l_0 norm of $\mathbf{w}$ should be R. The l_0 norm of a vector calculates the number of non-zero elements it contains. We look to minimize the MSE given we can select only R observations. In terms of an optimization problem, this is expressed as

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \text{tr}\left(\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right)^{-1} \\ \text{subject to} \quad & ||\mathbf{w}||_0 = R, \\ & \mathbf{w} \in \{0,1\}^M. \end{aligned} \tag{2.14}$$

The l_0 norm is not a convex function and is also called a pseudo-norm.. The l_0 norm can be relaxed to the l_1 norm, which is defined as

$$||\mathbf{x}||_1 = \sum_{i=1}^{N} |x_i| \tag{2.15}$$

for an N dimensional real vector $\mathbf{x}$. For the FIM to be invertible, all the w_i's have to be positive and be upper bounded by 1 for the selection constraint to make sense. The relaxed problem is

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \text{tr}\left(\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right)^{-1} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = R, \\ & \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{2.16}$$

This problem is the A-experimental design problem for sparse sensing. The assumption made here is that $\left(\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right)$ is invertible. Problem (2.16) is convex in $\mathbf{w}$ [42], implying that a unique solution exists and can be obtained using readily available solvers [43].

2.2.3 Sparse sensing with D-experimental design

The error covariance matrix $\mathbf{\Sigma}_{\mathbf{x}-\hat{\mathbf{x}}}$ in Section 2.2.1 captures the distribution of the error $\hat{\mathbf{x}}-\mathbf{x}$. The η confidence ellipsoid is the N dimensional ellipsoid having minimum volume which contains the error $\hat{\mathbf{x}} - \mathbf{x}$. The log-volume of this ellipsoid [27] for the model in (2.4) is given as

$$\log \text{vol}(\eta) = \beta - \text{logdet}\left(\frac{1}{2}\sum_{i=1}^{M} \frac{1}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right). \tag{2.17}$$

The log-volume with respect to the sparse sensing vector $\mathbf{w}$ is

$$\log \text{vol}(\eta) = \beta - \text{logdet}\left(\frac{1}{2}\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right). \tag{2.18}$$

One is interested in the ellipsoid having minimum possible volume, which amounts to maximizing the function $\text{logdet}\left(\frac{1}{2}\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right)$. In the sparse sensing context, this is posed as

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & -\text{logdet}\left(\frac{1}{2}\sum_{i=1}^{M} \frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right) \\ \text{subject to} \quad & ||\mathbf{w}||_0 = R, \\ & \mathbf{w} \in \{0,1\}^M. \end{aligned} \tag{2.19}$$

As we saw before, this formulation is not convex because $||\mathbf{w}||_0$ is not a convex function in $\mathbf{w}$ and $\{0,1\}^M$ is not a convex set. So, the relaxed problem is solved as

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & -\log\det\left(\frac{1}{2}\sum_{i=1}^{M}\frac{w_i}{\sigma^2}\mathbf{a}_i\mathbf{a}_i^T\right) \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = R, \\ & \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{2.20}$$

This approach is called the D-experimental design problem for sparse sensing. The objective function is concave in $\mathbf{w}$ [42] and it can be solved by interior point based solvers which adopt a semi-definite programming (SDP) approach [43].

2.3 Graph Signal Processing

Graph signal processing is an emerging field for processing data which resides over the nodes of a graph [44]. Throughout this section we will consider an undirected graph $\mathcal{G}$ with its node set $\mathcal{V} = \{v_1, \ldots, v_N\}$ and edge set $\mathcal{E}$. If nodes v_1 and v_2 are connected through an edge, then $(v_1, v_2) \in \mathcal{E}$. $|\mathcal{V}| = N$ is the number of nodes in the graph and $|\mathcal{E}| = M$ is the number of edges. $\mathcal{G}$ is also assumed to be acyclic, i.e. it has no self-loops. In this section, we highlight the important concepts, operators and data-processing tasks carried out in graph signal processing. We start with some important matrices which capture the graph structure, namely the Adjacency, Degree and the graph Laplacian matrix. Then, we discuss Shift Operators, the Graph Fourier Transform and Graph Filters, which show how data is processed on the graph. We conclude this section by discussing FIR graph filters and their connection to label propagation, a type of semi-supervised learning usually practised on graphs.

2.3.1 Graph Signals

A graph signal $\mathbf{x} \in \mathbb{R}^N$ is a mapping from $\mathcal{V}$ to $\mathbb{R}^N$ [44]. Each node v_i has a value x_i associated to it. In the case of a graph of a sensor network monitoring the temperature over a space, v_i represents a sensor and x_i the temperature recorded at that sensor. Figure 2.2 shows a graph signal observed over a sensor network having 150 nodes.

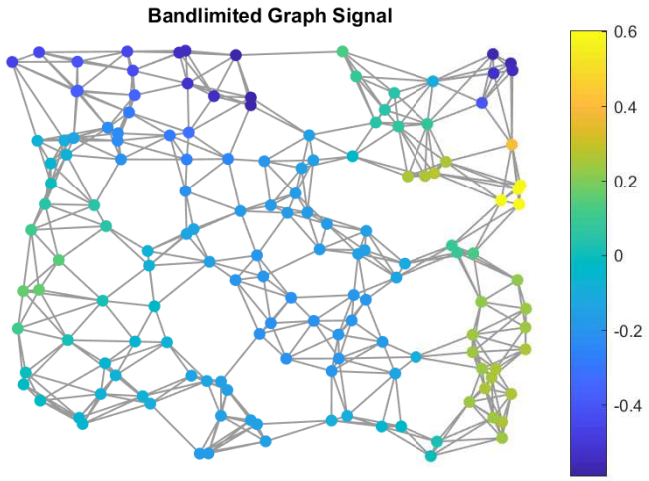


Figure 2.2: A graph signal over a sensor network graph. The circles are the nodes and the lines joining them are the edges. The node colour is the signal value over the graph.

2.3.2 Important Matrices and Operators

An $N \times N$ matrix $\mathbf{S}$ that captures the structure of the graph is called a graph shift operator [45]. For example, the adjacency matrix $\mathbf{A}$ is a shift operator such that A_{ij} represents the similarity between any two nodes v_i and v_j. A common way to define $\mathbf{A}$ is through the Gaussian kernel [46]:

$$A_{ij} = \begin{cases} \frac{1}{c} \exp \frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2} & ||\mathbf{x}_i - \mathbf{x}_j||_2^2 < \delta \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

where $\mathbf{x}_i$, $\mathbf{x}_j$ are the representations or features associated with v_i and v_j respectively, σ controls the decay of the Gaussian and c is a positive scaling factor. The edge (v_i, v_j) is allocated to $\mathcal{E}$ only if $||\mathbf{x}_i - \mathbf{x}_j||_2^2 < \delta$ where δ is a practically set threshold. This ensures that $\mathbf{A}$ captures the structure of the graph. The closer two objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the Euclidean space, the higher the value of $A_{ij} = A_{ji}$. The adjacency matrix $\mathbf{A}$ accepts the eigendecomposition $\mathbf{A} = \mathbf{U}_A \mathbf{\Sigma}_A \mathbf{U}_A^T$ with $\mathbf{U}_A^T \mathbf{U}_A = \mathbf{I}_N$ and $\mathbf{\Sigma}_A$ containing its real eigenvalues. $\mathbf{S}$ can also be non-symmetric like a random walk matrix [24]. The degree matrix $\mathbf{D}$ is a diagonal matrix with its ith diagonal entry D_{ii} representing the degree of node v_i. It is mathematically represented as $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_N)$, where $\mathbf{1}_N$ is the N dimensional vector of all ones. The graph Laplacian, also known as the combinatorial graph Laplacian, is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. $\mathbf{L}$ is symmetric and its eigendecomposition is $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$. The columns of $\mathbf{U}$ contain the eigenvectors of $\mathbf{L}$ and $\mathbf{\Sigma}$ contains the eigenvalues. The eigenvalues of $\mathbf{L}$ are also called the spectrum of the graph [47]. The graph Laplacian is an important matrix associated with graph signal processing and the spectral properties of the graph [6].

2.3.3 Graph Shift Operators

In classical signal processing, linear time-invariant (LTI) systems are based on shift operators. For time signals, shifting a signal is easy to visualize, but it is difficult to imagine shifting a graph signal over its nodes. To understand how shift operators are used on graph signals, consider we have a graph signal $\mathbf{x}$. The adjacency matrix shift operator applied on it gives the 1-shifted signal

$$\mathbf{x}^{(1)} = \mathbf{A}\mathbf{x}$$

with ith entry

$$x_i^{(1)} = \sum_{j=1}^{N} A_{ij}x_j = \sum_{j:v_j \in \mathcal{N}(v_i)} A_{ij}x_j. \tag{2.22}$$

This shows that $x_i^{(1)}$ only depends on the signal values of the nodes it shares an edge with. The shift operation means each node collects values from its neighbors, weighs and combines them, and this spreads the signal at each node throughout the graph over multiple shifts. This operation will be fundamental in defining graph filters in Section 2.3.5 and label propagation on graphs in Section 2.3.6.

2.3.4 Graph Fourier Transform

The Graph Fourier Transform (GFT) of a graph signal $\mathbf{x}$ represents the signal in the Graph spectrum. It is defined as $\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ where $\mathbf{U}$ is the eigenvector matrix of the Laplacian matrix given by $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ [47]. The GFT is the projection of $\mathbf{x}$ on the eigenvectors of $\mathbf{L}$. $\hat{x}_i$ is the GFT coefficient for the ith eigenvalue λ_i of $\mathbf{L}$. The GFT decomposes $\mathbf{x}$ into its spectral components. Similarly, the Inverse Graph Fourier Transform (IGFT) combines these components to synthesize $\mathbf{x}$ through $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. The eigenvalues of $\mathbf{L}$, $\{\lambda_1, \ldots, \lambda_N\}$ are collectively called the spectrum of $\mathcal{G}$.

2.3.5 Graph Filters

In LTI systems, a filter is a linear operator which shapes the spectrum of the signal it operates on by a point-wise multiplication in the spectral domain. Likewise, a graph filter shapes the GFT of a graph signal as follows:

$$\hat{y}_n = h(\lambda_n)\hat{x}_n, \tag{2.23}$$

where $\hat{y}_n$, $\hat{x}_n$ are the GFT coefficients at the output and input for frequency λ_n, respectively. λ_n is the nth eigenvalue of $\mathbf{L}$. The spectral filter response $h(\lambda_n)$ characterizes the graph filter and shapes $\mathbf{x}$ in the graph spectral domain. There are several ways proposed to design graph filters [48] [49]. Two examples are

$$h(\lambda_n) = \begin{cases} c & \lambda_n \leq \lambda_c \\ 0 & \text{otherwise} \end{cases}$$

and

$$h(\lambda_n) = \frac{1}{c}e^{-\frac{\lambda_n^2}{2\sigma^2}}$$

The first is a band-limited graph filter with cut-off frequency λ_c. It cuts all spectral components corresponding to frequencies $\lambda_n > \lambda_c$. The second example is the exponentially decaying filter, where the decay is controlled by the parameter σ. This filter attenuates the input signal more for larger values of λ_n.

The filtered signal in the vertex domain is $\mathbf{y} = \mathbf{U}\hat{\mathbf{y}}$. From (2.23), by taking the IGFT on both sides, we obtain

$$\mathbf{y} = \big(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^H\big)\mathbf{x}, \tag{2.24}$$

where $\mathbf{\Lambda} = \text{diag}([h(\lambda_1), \dots h(\lambda_n)])$. $\mathbf{U}\mathbf{\Lambda}\mathbf{U}^H$ is the graph filter in the vertex domain. We use the eigendecomposition of $\mathbf{L}$ to illustrate this concept but any suitable shift matrix $\mathbf{S}$ can also be used.

2.3.5.1 FIR graph filter

A Finite Impulse Response (FIR) graph filter of order K is of the form $\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K} \alpha_k \mathbf{S}^k$, where α_k denotes the kth graph filter coefficient. The role of α_k is to weigh the k-shifted version of the graph signal. This operation is similar to how a filter behaves in LTI systems. The transfer function is $h(\lambda_n) = \sum_{k=0}^{K-1} \alpha_k \lambda_n^k$, where λ_n is the n-th eigenvalue of $\mathbf{S}$. The FIR filter also has the advantage of being distributed and is easy to operate because of the recursive nature of its computation. This is because $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}$, $\mathbf{x}^{(2)} = \mathbf{S}^2\mathbf{x} = \mathbf{S}\big(\mathbf{S}\mathbf{x}\big) = \mathbf{S}\mathbf{x}^{(1)}$. In general the k-shifted output is related to its previous shifted output as $\mathbf{x}^{(k)} = \mathbf{S}\mathbf{x}^{(k-1)}$ and so on.

2.3.6 Label Propagation and FIR graph filters

Label propagation on graphs is a popular semi-supervised learning technique [50]. For this thesis, we will look at label propagation in terms of graph signals and FIR graph filters [48]. Consider a two-class problem on a graph $\mathcal{G}$ with N nodes. The two classes are ω_1 and ω_2, with labels $+1$ and -1, respectively. Since we are in the semi-supervised setting, the labels are known at only some of the nodes. Based on this assumption, the graph signal can be considered as $\mathbf{x}$ with

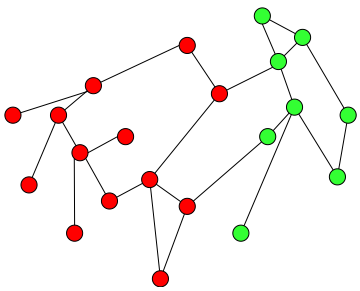
$$x_i = \begin{cases} +1 & v_i \in \omega_1 \\ -1 & v_i \in \omega_2 \\ 0 & \text{otherwise} \end{cases}. \tag{2.25}$$

This graph signal has a value of $+1$ or -1 at each labeled node and zero at each unlabeled node. If a row normalized version of the adjacency matrix, $\mathbf{A}_n$ is taken as the shift operator, $\mathbf{A}_n\mathbf{x}$ gives the spread of the labels over the graph after one step, $\mathbf{A}_n^2\mathbf{x}$ the spread over two steps and so on. After every spread or shift, the value of the shifted signal on the nodes which have labels is usually clamped to the original value to provide a constant source of the original label. This process converges and the unlabeled nodes that have a value greater than zero are classified to ω_1 and the rest to ω_2. We refer readers interested in this class of problems to [50] for an in-depth view. The output of such a process involves inverting a matrix of the order of the size of unlabeled data as shown in [50].

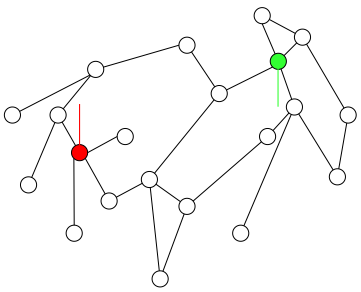
Label propagation can also be seen as graph filtering. Instead of depending only on the kth powers of $\mathbf{A}_n$ for label propagation through $\mathbf{y} = \mathbf{A}_n^K \mathbf{x}$, the labels can depend on all the previous powers to get

$$\mathbf{y} = \sum_{k=0}^{K} \alpha_k \mathbf{A}_n^k \mathbf{x}. \tag{2.26}$$

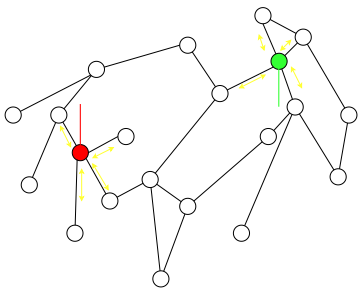
The operator $\sum_{k=0}^{K} \alpha_k \mathbf{A}_n^k$ is an FIR graph filter of order K. Depending on the choice of $\mathbf{A}$ (and therefore $\mathbf{A}_n$) and the nature of the α_k's, we can have more control over label propagation. The α_k's are usually designed for each specific problem [24] [49] [48]. The graph filtering operation is sufficiently fast. We will consider such a filter for our analysis, as presented in [24].



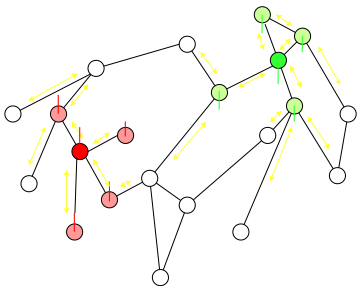
(a) Label Propagation in action.



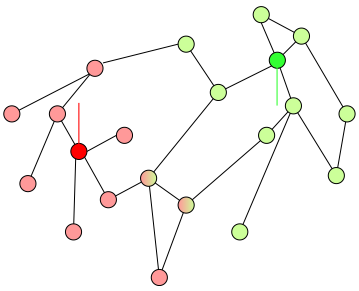
(b) Label Propagation in action.



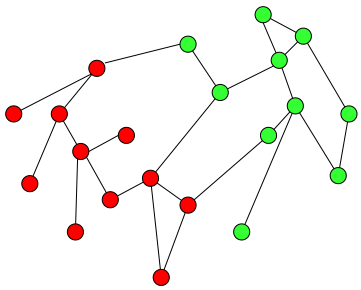
(c) Label Propagation in action.



(d) Label propagation first iteration.



(e) Label propagation after convergence.



(f) Predicted Labels.

Figure 2.3: Label propagation phenomenon illustrated: Figure 2.3a (top left) shows the original labels of each node (red and green); Figure 2.3b (top middle) shows one labeled node selected from each class (i.e. training set), red has positive label while green negative; Figure 2.3c (top right) shows the first diffusion step where the labeled nodes spread their values to their neighbours; Figure 2.3d (bottom left) shows the subsequent diffusion step; Figure 2.3e (bottom middle) shows the values at each node at the end of propagation; Figure 2.3f (bottom right) shows the predicted label at each node.

Figure 2.3 illustrates a label propagation process. Figure 2.3a shows the entire dataset, which is the ground truth. Figure 2.3b shows the semi-supervised stage with one node selected from each class (red and green). The label attached to the red node

is assumed to be positive (bar up) while the one with the green is assumed negative (bar down). Figure 2.3c shows the label propagation in action with the arrows (in yellow) along the edges denoting the spread of labeled information through connected nodes. Figure 2.3e shows the spread of the original labels from two nodes over the entire graph after convergence (i.e. no more spreading necessary). A threshold operation then predicts the labels for the unlabeled nodes, as shown in Figure 2.3f.

2.4 Clustering on Graphs

Clustering divides objects into groups. This section discusses clustering the nodes of a graph. Two popular clustering techniques, namely K-means and Fuzzy-C-means are discussed in Section 2.4.1. Section 2.4.2 discusses an appropriate node embedding for carrying out the clustering operation.

2.4.1 K-means and Fuzzy C-means clustering

In this section, we will briefly explain the two clustering methods, one of which we will use in this thesis: K-means and Fuzzy C-Means clustering. The K-means algorithm is one of the most popular hard clustering techniques [51]. Given a D-dimensional data-set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i$ the ith data point, K-means minimizes the cost

$$J(\mathbf{U}, \mathbf{C}) = \sum_{i=1}^{K} \sum_{j=1}^{N} U_{ij} ||\mathbf{x}_j - \mathbf{c}_i||^2, \tag{2.27}$$

where the column $\mathbf{c}_i$ of matrix $\mathbf{C}$ is the cluster centre of the ith cluster, U_{ij} is the similarity between data point $\mathbf{x}_j$ and cluster centre $\mathbf{c}_i$, and N and K denote the number of objects and the number of clusters, respectively. The task is to jointly find the optimal U_{ij}'s and the $\mathbf{c}_i$'s that minimize J. In K-means, for each $\mathbf{x}_j$, the corresponding $\mathbf{u}_j = [u_{1j}, u_{2j}, \ldots, u_{Kj}]^T$ has only one element which is one and the rest are zero. This is known as hard clustering. The element which is equal to one corresponds to the cluster to which $\mathbf{x}_j$ is assigned. Each step in K-means comprises two updates: the first step in K-means assigns each point to its nearest cluster in a hard fashion; the second step updates the cluster centres based on the objects assigned to them. Then the objects are assigned to the clusters again and this goes on until convergence.

Fuzzy C-Means clustering assigns a membership value to each point. The membership function defined between a point and a cluster returns a value between zero and one which quantifies the probability of that point belonging to that cluster. One represents maximum membership and zero denotes no membership. Membership can also be thought of as belongingness. For FCM [52], the cost function $J(\cdot)$ is similar to K-Means but the assignment of an object to a cluster is no longer hard. Two constraints are imposed on $\mathbf{u}_j$ for each object $\mathbf{x}_j$. As the membership values for each point are probabilities, they must add up to one over the clusters and they must be individually non-negative. These values will be generally more spread out compared to K-means.

More specifically, FCM solves

$$\begin{aligned} \underset{\mathbf{U}\in\mathbb{R}^{K\times N},\mathbf{C}\in\mathbb{R}^{D\times K}}{\text{minimize}} \quad & \sum_{i=1}^{K}\sum_{j=1}^{N} U_{ij}^{m}||\mathbf{x}_j - \mathbf{c}_i||^2 \\ \text{subject to} \quad & \mathbf{1}_K^T\mathbf{U} = \mathbf{1}_N^T, U_{ij} > 0, i = 1,\dots K, j = 1,\dots N. \end{aligned} \tag{2.28}$$

The exponent m controls the degree of fuzziness and its default value is two.

2.4.2 Laplacian Eigenmap Embedding

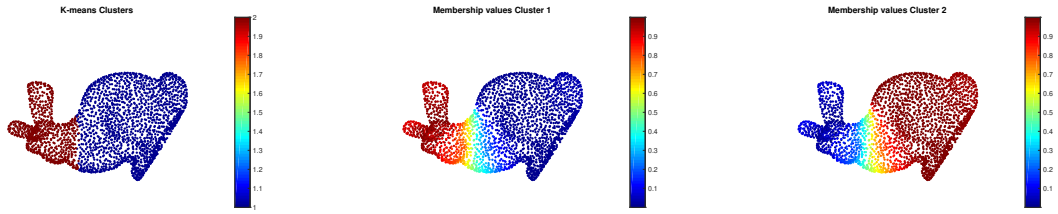
To cluster the nodes of a graph, we need a representation or feature vector for each node. A representation or embedding maps every node to a feature vector. If the graph is built from data, then clustering with the original representation may not be desired because we need to use the underlying graph structure. One representation useful for clustering nodes is the Laplacian eigenmap embedding [46]. This reduces the dimension of the original feature vectors such that if two points are similar in the Euclidean space, they will be similar in this lower-dimensional space concerning the graph.

The Laplacian eigenmap problem is defined as follows: Given a dataset of size $N \times D$, with objects stacked row-wise, we want to learn the embedding in a P-dimensional space and generate the matrix $\mathbf{P}$ of size $N \times P$ with $P \ll D$. The latter can be recast as solving the optimization problem

$$\begin{aligned} \underset{\mathbf{P}\in\mathbb{R}^{N\times P}}{\text{minimize}} \quad & \text{tr}(\mathbf{PLP}^T) \\ \text{subject to} \quad & \mathbf{P}^T\mathbf{P} = \mathbf{I}_P, \end{aligned} \tag{2.29}$$

where $\mathbf{I}_P$ is the $P \times P$ identity matrix and $\mathbf{L}$ the graph Laplacian learned from the original data-set. The solution to this problem contains the first P eigenvectors of $\mathbf{L}$ stacked column-wise. These are the eigenvectors corresponding to the lowest P eigenvalues of $\mathbf{L}$. However, for feature representation, the eigenvector corresponding to the zero eigenvalue is ignored as it is a constant vector and does not help in providing information about any node. If we want to cluster the nodes into 2 clusters, for instance, we need the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$, $\mathbf{u}_2$. If we need K clusters we need the first K eigenvectors barring the first to construct the embedding $\mathbf{P} = [\mathbf{u}_2, \dots, \mathbf{u}_K]$. Upon obtaining $\mathbf{P}$ for the data-set, we can use it to cluster the nodes of the graph.

Figure 2.4 illustrates the key difference between K-means and FCM clustering using the bunny graph. The Laplacian embedding is used for a two cluster problem. We choose the second feature for each node as the first feature is a constant; there is one feature per node. Figure 2.4a shows the hard clustering nature of K-means. Each node is assigned a value of either one or two. In terms of membership values, this means each node belongs either to the first or the second class with a membership of one. Figure 2.4b shows the membership values obtained by FCM for each node and the first cluster and Figure 2.4c the second cluster. We can see that the values vary from zero to one for each cluster smoothly with a transition at the border between the two clusters. We will use the combination of Laplacian eigenmap embedding and Fuzzy C-means clustering to perform active learning.



(a) K-Means membership values for two clusters.

(b) FCM Membership values for the first cluster.

(c) FCM Membership values for second cluster.

Figure 2.4: K-means and Fuzzy C-means Clustering on the Bunny graph. Fig 2.4a (left) shows the K-means clustering of the nodes of the Bunny graph for 2 clusters. Each node has a value of either one or two. Figures 2.4b (middle) and 2.4c (right) show the membership functions obtained through FCM clustering for the first and second cluster, respectively. Each node has two membership values (one for each cluster) that add up to one.

2.5 Compressive Sensing

Compressive (Compressed) sensing deals with the reconstruction of signals that are sparse in some basis from sub-sampled measurements. It is a rich research area and has led to advances in signal processing. This section will provide a brief introduction to the main functional aspects of compressed sensing. Section 2.5.1 describes the notations and framework of such a system; Section 2.5.2 describes two popular methods to recover signals.

2.5.1 Compressed Sensing Framework

A signal $\mathbf{x} \in \mathbb{R}^N$ can be expressed as the combination of N basis vectors in $\mathbb{R}^N$ as

$$\mathbf{x} = \mathbf{\Psi s}. \tag{2.30}$$

$\mathbf{\Psi} = [\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N]$ is the basis matrix. $\{\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_N\}$ are the basis vectors and $\mathrm{s} \in \mathbb{R}^N$ contains the coefficients which represent $\mathbf{x}$ in this basis. The signal $\mathbf{s}$ is said to be S-sparse if only $S \ll N$ of the entries in $\mathbf{s}$ are non-zero. $\mathbf{x}$ is then said to be S-sparse in the basis $\mathbf{\Psi}$. A popular example of a basis $\mathbf{\Psi}$ is the Discrete Fourier Basis [53]. Compressive sensing concerns itself with the theoretical framework of reconstructing $\mathbf{s}$ (hence $\mathbf{x}$) exactly from $M \ll N$ measurements without knowing the locations of the non-zero elements in $\mathbf{s}$. The measurements are obtained through the following sensing mechanism:

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s}, \tag{2.31}$$

where $\mathbf{\Phi}$ is an $M \times N$ sensing matrix. The vector $\mathbf{y} \in \mathbb{R}^M$ can also be viewed as the projection of $\mathbf{x}$ on the M rows of $\mathbf{\Phi}$.

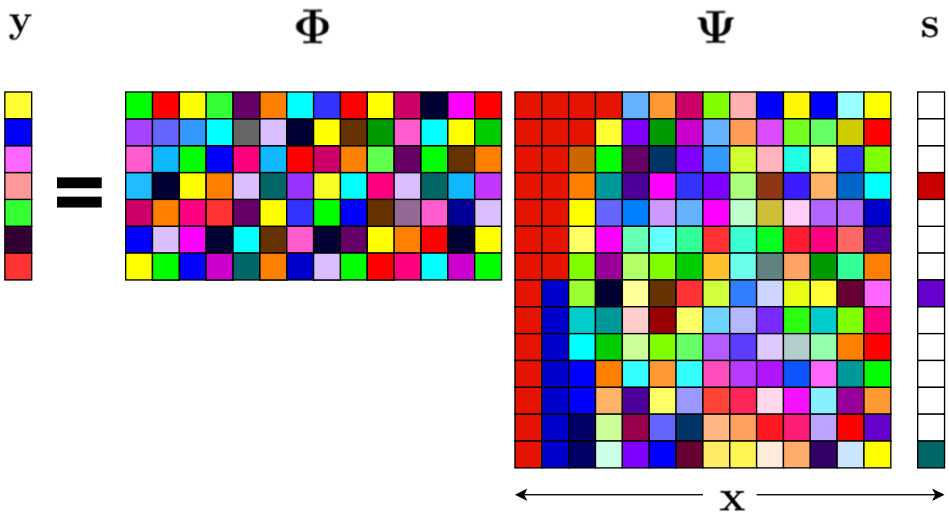


Figure 2.5: A Compressed Sensing system. Vector $\mathbf{y}$ is the $M \times 1$ observed vector with $M = 7$. The sparse vector $\mathbf{s}$ is 3-sparse with non-zero entries in 3 locations: s_4, s_8 and s_{14}. The basis $\mathbf{\Psi}$ is of size $N \times N$ with $N = 14$. $\mathbf{\Phi}$ is a random sensing matrix (i.e. sensing matrix filled with random values drawn from an i.i.d. Gaussian distribution for example).

The main questions associated with the reconstruction of $\mathbf{x}$ given $\mathbf{\Psi}$ are:

1. What does $\mathbf{\Phi}$ tell us about reconstructing $\mathbf{x}$?
2. If a good $\mathbf{\Phi}$ exists, how does one reconstruct $\mathbf{x}$?

From equation (2.31), $\mathbf{\Phi\Psi}$ can be written as a matrix $\mathbf{A}$ of size $M \times N$, which is also called a dictionary. One criterion which allows the successful reconstruction of $\mathbf{x}$ with high probability is the Restricted Isometry Property (RIP) [54]. $\mathbf{A}$ satisfies the RIP if there exists a $\delta_K \in (0, 1)$ such that

$$||(1 - \delta_K)\mathbf{s}||_2^2 \leq ||\mathbf{As}||_2^2 \leq ||(1 + \delta_K)\mathbf{s}||_2^2 \tag{2.32}$$

where $\mathbf{s}$ belongs to the set of all signals having l_0 norm less than or equal to K. Expression (2.32) says that $\mathbf{A}$ with RIP approximately preserves the norm of $\mathbf{s}$, when reducing it from an N to an M-dimensional space. Since $\mathbf{A} = \mathbf{\Phi\Psi}$ and $\mathbf{\Psi}$ is usually fixed, designing $\mathbf{A}$ is the same as designing $\mathbf{\Phi}$. Constructing $\mathbf{\Phi}$ such that $\mathbf{A}$ satisfies the RIP is an NP-hard problem. Instead, if we select every element of $\mathbf{\Phi}$, $\phi_{i,j}$ such that $\phi_{i,j} \sim \mathcal{N}(0, \frac{1}{N})$, then $\mathbf{A}$ obeys the RIP with high probability when

$$M \geq cK\log(\frac{N}{k}), \tag{2.33}$$

with c being a constant [54][55][56]. The inequality (2.33) gives a lower bound on the number of measurements M one needs in such a case.

Another important parameter which can determine the success of the reconstruction of $\mathbf{x}$ is the mutual coherence of the matrix $\mathbf{A}$, denoted by $\mu_{\mathbf{A}}$. $\mu_{\mathbf{A}}$ is defined as

$$\mu_{\mathbf{A}} = \max_{1<=i,j<=N;i\neq j} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{||\mathbf{a}_i||_2 ||\mathbf{a}_j||_2} \tag{2.34}$$

where $\mathbf{a}_i$ represents the ith column of $\mathbf{A}$. The mutual coherence qualitatively gives us a picture of what is the highest similarity between any two different columns of $\mathbf{A}$. A lower value of mutual coherence helps algorithms like Orthogonal Matching Pursuit [57] and Basis Pursuit [58] to reconstruct $\mathbf{s}$ (and $\mathbf{x}$) perfectly with high probability in the noiseless case. In the noiseless case, the exact sparse solution is guaranteed when $S \leq \frac{1}{2}\left(1 + \frac{1}{\mu_{\mathbf{A}}}\right)$.

2.5.2 Recovering the Sparse signal

Given a matrix $\mathbf{\Phi}$ which satisfies the RIP condition, the problem for finding the sparsest $\mathbf{s}$ is

$$\begin{aligned} \underset{\mathbf{s}\in\mathbb{R}^N}{\text{minimize}} \quad & ||\mathbf{s}||_0 \\ \text{subject to} \quad & \mathbf{\Phi\Psi s} = \mathbf{y}. \end{aligned} \tag{2.35}$$

The Problem (2.35) is non convex, due to the l_0 norm. Since the non-zero positions are not known, solving (2.35) is difficult and is NP-hard. There are two sets of approaches for solving it. One is based on greedy methods like Orthogonal Matching Pursuit (OMP), Matching Pursuit (MP), thresholding and the other is based on convex relaxations of the l_0 norm like Basis Pursuit (BP). Approximate methods like Orthogonal Matching Pursuit (OMP) [57], Basis Pursuit (BP) [58] obtain the true solution to (**??**) conditionally [54] [55] [56].

2.5.2.1 Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) is one of the most important methods which falls into the category of greedy approximations to solve problem (2.35) [57]. It selects the desired support for the sparse $\mathbf{s}$ from the columns from $\mathbf{A}$, one at a time.

Algorithm 1 Orthogonal matching pursuit (OMP) [57]

Require: sparse $\mathbf{s}$, $||\mathbf{s}||_0 = S$.
Input: $\mathbf{A} \in \mathbb{R}^{M\times N}$, $\mathbf{y} \in \mathbb{R}^{M\times 1}$, S.
Output: $\mathbf{s}$.

1: **Initialization** $\mathbf{s}_0 = 0$, $\mathbf{r}_0 = \mathbf{y}$, $\mathcal{S} = \{\}$, $\mathbf{a}_i = \frac{\mathbf{a}_i}{|\mathbf{a}_i|}$; $i = 1, \ldots, N$.
2: **while** $|\mathcal{S}| \leq S$ **do**
3: $\quad k = k + 1$,
4: $\quad j = \underset{i\in\{1,\ldots,N\}}{\text{argmax}} \ |\mathbf{a}_i^T \mathbf{r}_{k-1}|_1$,
5: $\quad \mathcal{S} = \mathcal{S} \cup \{j\}$,
6: $\quad \mathbf{s}_k = \underset{\mathbf{s}\in\mathbb{R}^{|\mathcal{S}|}}{\text{argmin}} ||\mathbf{A}_{\mathcal{S}}\mathbf{s} - \mathbf{y}||_2^2$,
7: $\quad \mathbf{r}_k = \mathbf{A}_{\mathcal{S}}\mathbf{s}_k - \mathbf{y}$,
8: **end while**

The algorithmic description of OMP is shown in Algorithm 1. A brief overview of OMP is as follows: The columns of $\mathbf{A}$ are normalized to have unit norm in line 1. The set $\mathcal{S}$ contains the support of $\mathbf{s}$. $\mathcal{S}$ is initialized to the null set. The residue $\mathbf{r} = \mathbf{As} - \mathbf{y}$ is a measure of the approximation of the system and is initialized to $\mathbf{y}$. At each step,

the columns of $\mathbf{A}$ are projected onto the residual and the column which has the highest absolute component along the residual is selected in line 4. The index of this column is then added to the support in line 5. The solution of $\mathbf{s}$ corresponding to the support $\mathcal{S}$ is obtained through the least squares solution of $\mathbf{A}_{\mathcal{S}}\mathbf{s} = \mathbf{y}$ in line 6. $\mathbf{A}_{\mathcal{S}}$ contains the columns of $\mathbf{A}$ indexed by the elements of $\mathcal{S}$. The residue obtained in line 7 is the least square error of the approximation. This residue is used in the next iteration. It is important to note that the residue is perpendicular to the current support because of the nature of the least squares solution. In the next step, the current support will have no components along it and the search is conducted in a direction orthogonal to the current support. The steps are repeated until the desired cardinality of $\mathcal{S}$ is reached or the norm of the residue is below a certain limit.

2.5.2.2 Basis Pursuit

The $l1$ convex relaxation of problem (2.35) is

$$\begin{aligned} \underset{\mathbf{s}\in\mathbb{R}^N}{\text{minimize}} \quad & ||\mathbf{s}||_1 \\ \text{subject to} \quad & \mathbf{As} = \mathbf{y} \end{aligned} \tag{2.36}$$

This is solved by the basis pursuit algorithm [58]. It is a convex problem and can be solved by converting it into a linear program or using any off-the-shelf solver [43].

2.6 Conclusion

This concludes the background material. In this thesis, we focus on active learning for semi-supervised classifiers on graphs. The following concepts will be important for the rest of this thesis from each section: from Section 2.1, we will focus on a variant of Pool-based active learning; from Section 2.2, we use both A and D-experimental design as a querying criterion for active learning in Chapter 5; from Section 2.3, we focus on shift operators, graph filters and label propagation; from Section 2.4, we use the Laplacian eigenmap embedding problem along with Fuzzy C-Means clustering for one set of approaches in Chapter 5; from section 2.5, we use OMP, basis pursuit and the concept of mutual coherence and projection matrices in Chapter 5.

Literature Review

3

This chapter discusses the relevant literature on active semi-supervised learning on graphs. The material and concepts touched upon in Chapter 2 will be leveraged to discuss these methods. This review will also cast the differences between these methods. We identify one scenario, that of pool-based semi-supervised learning to categorize the literature. The chapter is organized as follows: Section 3.1 introduces the Pool-based semi-supervised active learning scenario, which is classified into two categories: multiple batch training and single batch training. Section 3.2 summarizes the literature and highlights some gaps.

3.1 Pool-based Active Semi-supervised Learning on Graphs

The topic of active semi-supervised learning contains a multitude of approaches, as shown in [9]. The methods considered in this literature review consider the graph structure of the data, while using it as a regularizer [5], or as a platform for label propagation [59]. However, an exception is made for the method in [12]. Although not being graph-based, we discuss this method because it considers transductive learning, which is a common theme for semi-supervised learning on graphs. The methods included in this survey fall into the pool-based active learning scenario as described in Section 2.1. Pool-based active learning deals with selecting one or more points from a pool of unlabeled data. There are two types of pool-based active learning: Multiple training phases and Single training phase. Active learning can also be classified based on the objective function chosen for selecting querying points, or the semi-supervised method used.

3.1.1 Multiple Training Phase

In this mode, active learning is carried out over more than one training phase. After each phase, the learner queries the desired points and they are added to the labeled set. The learner trains itself on this set, and this continues until the desired criteria are met. Listed below are examples of such methods found in the literature.

- **Gaussian Fields and Harmonic Functions in Active Learning**

  Zhu et.al. in [16] combines pool-based active learning over multiple batches with semi-supervised learning on graphs. The classifier is based on Gaussian random fields and harmonic functions [60]. The node labels together form a graph signal. It has its own distribution. The classifier solves for a graph signal which promotes the smoothness of the labels over the graph (for both labeled and unlabeled nodes) while the labeled nodes retain their values. The classifier minimizes an energy function which is indicative of the smoothness of the labels on the graph. There

are multiple interpretations to the solution of this problem which link it to random-walk matrices and Gaussian processes [61]. Once the classifier output is obtained, a greedy search is done over the unlabeled nodes and the one that minimizes the empirical risk (i.e. expected classification error) is added to the labeled set $\mathcal{L}$. This is pool based learning wherein each batch, one node is selected. The classifier is trained on the updated $\mathcal{L}$ to select another node until the desired cardinality on $\mathcal{L}$ is met. The main contribution of this method is in using the semi-supervised learner and formulating an empirical risk for selecting the optimal nodes. Several extensions to this approach have been made, like [17] which works with the estimation of the expected generalization error. Another method that shares the same classifier but opts for one training phase and selects the nodes together is shown in [10].

- **Active Learning Networked Data**

  Bilgic et al developed ALFNET, which stands for Active Learning for Networked Data. ALFNET uses two different representations for each data point [62]. The *content-only* representation contains the original features while the *collective* representation takes each data point's neighborhood in the graph into account. The graph is initially divided into several clusters using modularity clustering [63]. From some of these clusters, a point is added to the labeled set at random. Then the logistic classifier is trained on both the *content-only* and *collective* data. Each cluster is evaluated on how much the classifier disagrees over the points contained within it. Nodes from the most disagreeable clusters are selected at random and added to the labeled set and the process continues. This is an example of multiple batch pool-based active learning with multiple nodes being selected per phase. The work in [19] considers label propagation on graphs during the learning phase. For selecting points, they introduce a new cost function comprising three criteria: the uncertainty, impact and redundancy. The uncertainty can be calculated after each training phase through the entropy. This method does not select nodes solely on one criterion as the K most uncertain or informative points may not result in the best classification. The impact measures the influence of the selected point on the remaining unlabeled nodes. Redundancy tends to select points that do not overlap in terms of informativeness. The overall cost function is sub-modular and allows greedy selection. The method in [20] uses entropy as the criterion to select points after each training phase.

3.1.2 Single Training Phase

In the single training mode of pool-based active learning, the training is carried out only once, after the set of optimal points is selected by the learner in one go.

- **Variance Minimization Criterion**

  The work in [10] assumes the labels of the nodes will be smooth with respect to the graph structure; i.e., it is a smooth graph signal. This means if two nodes are connected with high similarity, they will have similar labels. Such signals

are described by a Gaussian multivariate probability distribution. In the semi-supervised scenario, the labels of the unlabeled points are determined by the conditional distribution over the unlabeled nodes, given the labeled nodes. This conditional distribution is also a multivariate Gaussian [64] and its mean is the classifier output. The trace of the covariance matrix is the mean square error of the prediction (cf. Section 2.2.1). The trace does not depend on the labels and this allows the selection of nodes that minimize it greedily. Once the labels are obtained, the classifier is run on each class and the unlabeled nodes are classified. The highlights of this method are: it selects all points in one go, so it can be done offline; it uses the Gaussian Process-based classifier and minimizes the variance of prediction. It should also be noted that this classifier does not need multiple batches. The work in [15] is related to [10] but they consider the $\mathbf{\Sigma}$ optimality criterion for active surveying, which is the sum of all elements of the inverse of the Laplacian sub-matrix indexed by the unlabeled nodes. The authors inspect the sub-modularity of this criterion and select nodes greedily with improved results over the trace-based criterion.

- **Error Bound Minimization**

  In paper [18], the authors use a Laplacian regularized least squares classifier [5]. They take into account the structure formed by the data through the graph Laplacian. The authors derive an upper bound on the mean square predicted error. The goal is to select the samples that minimize this upper bound. This problem is originally NP-hard, so a reformulation is introduced to select rows from the data matrix. Projected gradient descent is used to find such an optimal matrix. This approach shares similarity with optimal experimental design as it solves the A-experimental design problem under a specific condition. The work in [11] looks at the D-experimental design aspect of the same classifier but the focus is to minimize the MSE of the classifier and not its prediction. The authors also extend their work to account for kernels. The work in [13] uses a similar approach by using the logistic classifier. It should, however, be noted that [13] solves the problem without considering the graph structure. The work in [21] also proposes an active labeling scheme on graphs for one batch training by minimizing an error bound which depends on the prediction error in terms of label smoothness. They use a graph min-cut based algorithm to partition the graph and predict the remaining nodes [65].

- **Transductive Experimental Design**

  The paper in [12] introduces the framework of selecting data points for pool-based active learning for linear and non-linear regression problems. It focuses on the variance of the classifier prediction on test data. If all the data (i.e. training and testing) is used together, which is the case with transductive learning, the necessary error function can include the entire data. The authors show that the solution to the transductive A-experimental design error function is the same as that of a representation problem that involves all the data. Selecting the most informative points that minimize the variance of prediction on the test set is the same as selecting the set of data points which capture the most information about

the entire dataset. This is a strong result and motivates the use of transductive A-experimental design. Another advantage of a linear model is that the A(or D) experimental design cost function is independent of labeled information. The authors also extend their method to kernels and propose an alternating optimization routine to solve the selection problem (an alternative to the slow semi-definite programming solvers).

- **Band-limited Graph Signal Sampling**

  Gadde et.al [22] take a different approach to the graph active learning problem. They assert that the membership functions for all classes should be smooth graph signals for graphs which are built on the similarity between data points. The membership function of each class is treated to be a bandlimited graph signal as such signals are smooth [66]. However, in practice such signals are not bandlimited; hence, there is a need for a bandlimited approximation of such graph signals for each class, given a set of nodes which belong to that class. They leverage the theory of sampling and reconstruction of bandlimited signals [67], [68] and organize the active learning algorithm as follows: In the first stage, the method greedily selects the nodes from which a graph signal of the highest cut-off frequency can be perfectly reconstructed; next, these nodes are labeled and the band-limited approximations of all class membership graph signals are obtained; finally, these functions are compared at each unlabeled node to classify them.

3.2 Discussion

Table 3.1 categorizes the literature relevant to this thesis. This organization suggests that such methods can be classified based on multiple categories. We opt for the scenario criterion (i.e. multiple batches and one batch) shown in the third column of the table to do so since it is the most intuitive. Looking at the table, we see that methods which utilize label propagation do not unify the classifier and the active learning process. Usually, the propagation is carried out first, then the points are queried according to some known criterion. This is a gap that will be bridged in this thesis. Besides, the label propagation methods reviewed belong to the multiple training category of pool-based learning. Training multiple times is affordable when the training phase is fast and when the querying function is dependent on the current state of the classifier. When the function does not depend on the labels or the state of the classifier, all the points can be selected in one go and multiple training stages are not required. In [10], for instance, the objective function depends on the matrix obtained by selecting the rows and columns of the graph Laplacian corresponding to the unlabeled nodes. This is independent of the classifier or the labeled information; hence, all points can be selected together. The same can be said for some of the methods based on experimental design which are based on a linear classifier model [12], [18]. Column four shows that optimal experimental design, error bound minimization and information criteria are commonly used querying functions. The selection is done using greedy approximations [22], relaxed methods [18], or a mixture of both [12]. Experimental design has been linked to active learning on classifiers which are not conceptualized

Paper	Classifier	Batch	Query	Selection
Zhou et.al. [16]	Gaussian Process harmonic function	Multiple	Expected generalization error	Greedy
Bilgic et.al. [62]	Logistic Regression	Multiple	Disagreement within cluster	Random
Shi et.al. [19]	Label Propagation	Multiple	Entropy, Influence, Redundancy	Greedy
Gu et.al. [13]	Logistic regression	Multiple batches	Variance of classifier	Non-greedy
Long. et.al. [20]	Label Propagation	Multiple	Entropy	Non-greedy
Yu et.al. [12]	Ridge Regression	Single	Transductive A-experimental design	Alternating optimization, Greedy search
Ji et.al. [10]	Gaussian process harmonic function	Single	Variance of prediction	Greedy
Ma et.al. [15]	Gaussian process harmonic function	Single	Σ criterion	Greedy
Gu et.al. [18]	Laplacian regularized least squares	Single	Prediction MSE error bound	Non-greedy
He et.al. [11]	Laplacian regularized least squares	Single	D-experimental design	Non-greedy
Gadde et.al. [22]	Band-limited signal reconstruction	Single	Bandwidth increment	Greedy
Guillory et.al. [21]	Graph Mincuts	Single	Generalization error	Greedy

Table 3.1: Review of relevant literature. The first column mentions the authors along with the citation; the second column shows the classifier used; the third column specifies the pool-based approach being adopted (i.e. multiple or single batch); the fourth column shows the querying criterion; the fifth column shows the nature of selection of the points (i.e greedy or non-greedy selection).

in the vertex domain but on regression problems with smoothness constraints. For a label propagation scenario, the same assumptions do not hold. In this thesis, we analyze how experimental design applies to this setting. We also want to look at other querying methods that consider the graph structure.

In summary, current works have not explored label propagation from the classifier's

perspective for one training batch. We aim to analyze the impact of experimental design on this classifier and provide interdisciplinary querying techniques.

Semi-supervised Classifier

4

In this chapter, we introduce the semi-supervised classifier chosen for this thesis [24]. Section 4.1 introduces random walks and the semi-supervised classifier. This classifier works on the diffusion of labeled information (probabilities) of each class over the graph following the principle of random walks. Section 4.3 summarizes this method, the assumptions and the problems arising from it.

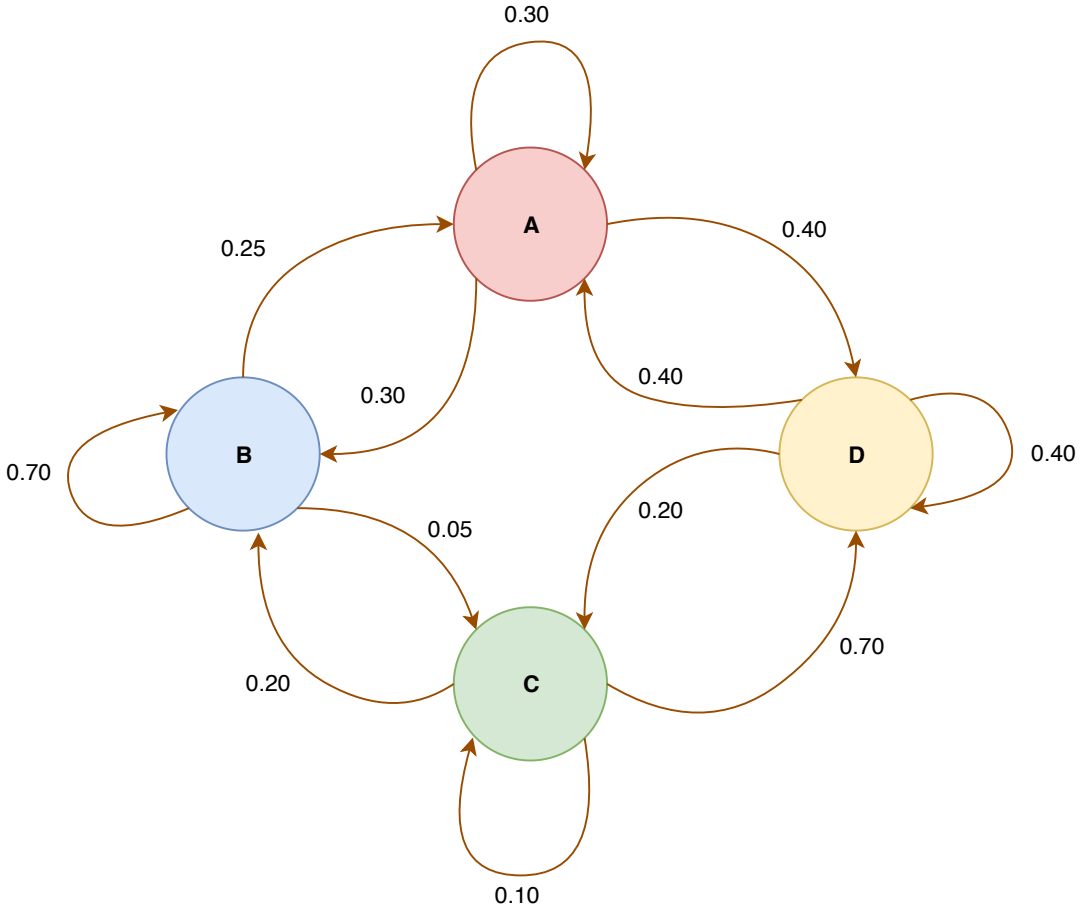


Figure 4.1: A probability state transition diagram: the circles A,B,C and D represent the states; the arrows indicate the transition between states; the arrow numbers indicate the conditional probabilities of each state transition.

4.1 Random Walk-based Diffusions on Graphs

We start this section by introducing the concept of a state and state landing probabilities. Figure 4.1 shows four different circles with a set of arrows: the circles are the states; the arrows indicate the transition between states and the numbers indicate the conditional probabilities of each state transition. For instance, the probability of transitioning from state A at any time instant k to B at time $k+1$ is 0.30. The movement from one state to the next is also called a step or a hop. There are also probabilities associated with no change of state, e.g. being in state A and remaining at that state at any time instance k has a probability of 0.30.

A walk is a series of steps or hops. To describe a walk mathematically, we first define the probability state vector $\mathbf{p}_0 = [p_{0,A}, p_{0,B}, p_{0,C}, p_{0,D}]^T$. Each entry indicates the probability of being present at that particular state at the start of the walk (hop=0). The transition probabilities are conditional only on the current state (first order Markov property) and not on the previous ones. We place the state transition probabilities in the matrix

$$\mathbf{S} = \begin{bmatrix} 0.3 & 0.25 & 0 & 0.4 \\ 0.3 & 0.7 & 0.2 & 0 \\ 0 & 0.05 & 0.1 & 0.2 \\ 0.4 & 0 & 0.7 & 0.4 \end{bmatrix},$$

where S_{ij} indicates the transition probability from the current state j to next state i. When $\mathbf{S}$ is multiplied with $\mathbf{p}_0$, we get the probability state vector $\mathbf{p}_1$ as

$$\mathbf{p}_1 = \mathbf{S}\mathbf{p}_0. \tag{4.1}$$

The ith element of $\mathbf{p}_1$, $[\mathbf{p}_1]_i$ denotes the probability of being at state i after one transition from the starting state $\mathbf{p}_0$. This is the one-hop probability vector. The k-hop probability vector starting from $\mathbf{p}_0$ is

$$\mathbf{p}_k = \mathbf{S}^k\mathbf{p}_0. \tag{4.2}$$

This model is central to the diffusion semi-supervised classifier we discuss next. The state transition matrix for this classifier is a weighted combination of the powers of the random walk matrix, which is also a graph shift operator (Section 2.2). Each state is now a node.

Let $\mathbf{W}$ be the weighted adjacency matrix of a graph $\mathcal{G}$ with N nodes and C classes. The matrix $\mathbf{D} = \text{diag}(\mathbf{W}\mathbf{1}_N)$ is the degree matrix, with $\mathbf{1}_N$ the N-dimensional vector of all ones. The random walk model over a graph consists of a discrete Markov chain. $\mathbf{p}_k$ is the state probability vector defined over the nodes at hop k, whose entries sum up to one. The entry $[\mathbf{p}_k]_i$ is the probability of the walk being present at node i at hop k. The matrix $\mathbf{S} = \mathbf{W}\mathbf{D}^{-1}$ is the state transition matrix with $[\mathbf{W}\mathbf{D}^{-1}]_{ij}$ the probability of making a hop from node j (at hop k) to node i (at hop $k+1$) for all k. The sum of all S_{ij}'s for a fixed j is equal to one, which is expected, since the sum of the jth column of $\mathbf{W}$ equals d_j. So,

$$S_{ij} = \Pr(X_k = i | X_{k-1} = j) = W_{ij}/d_j = [\mathbf{W}\mathbf{D}^{-1}]_{ij} \tag{4.3}$$

denotes the probability of landing at node i from node j and X_k denotes the state of the walk at time k. The vector $\mathbf{p}_0$ denotes the probability state vector, which is a distribution over the nodes at the start (i.e. at hop $k = 0$). The vector $\mathbf{p}_1 = \mathbf{S}\mathbf{p}_0$ is the probability state vector at hop $k = 1$, $\mathbf{p}_2 = \mathbf{S}^2\mathbf{p}_0$ the state vector at hop $k = 2$, and so on. In general, $\mathbf{S}^k\mathbf{p}_0$ is the probability state vector after hop k. The entry $[\mathbf{p}_k]_i = [\mathbf{S}^k\mathbf{p}_0]_i$ denotes the probability of a walk landing at node i, having started from the initial probability state vector $\mathbf{p}_0$. We introduce a $K \times 1$ parameter vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_K]^T$, that combines the landing probability vectors of the first K hops as

$$\mathbf{f}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \mathbf{p}_k, \tag{4.4}$$

where $\boldsymbol{\theta}$ belongs to the probability simplex $\mathcal{S}^K$ (i.e., $\boldsymbol{\theta} \geq \mathbf{0}$ and $\boldsymbol{\theta}^T\mathbf{1}_K = 1$). The entry θ_j is the weight for the jth hop probability state vector. The parameter $\boldsymbol{\theta}$ characterizes the diffusion process and is unique for each class. We exploit this to find this parameter for each class.

4.2 Adaptive Random Walk Diffusion-based Semi-supervised learning on Graphs

We are given $\mathcal{G}$ with N nodes and C classes. In the semi-supervised setting, only a few labeled nodes are known for each class [24]. For a class c, $\mathcal{L}_c \subset \mathcal{V}$ is the index set of the labled nodes belonging to class c. For class c, the starting probability vector $\mathbf{v}_c$ (equivalent to $\mathbf{p}_0$ above) is

$$[\mathbf{v}_c]_i = \begin{cases} \frac{1}{|\mathcal{L}_c|}, & \text{if } i \epsilon \mathcal{L}_c \\ 0, & \text{otherwise} \end{cases}, \tag{4.5}$$

where $|\mathcal{V}_c|$ is the number of objects in class c. This walk starts uniformly at random from any of the labeled nodes belonging to that class c. After K hops, the probability vector for class c is

$$\mathbf{f}_c(\boldsymbol{\theta}_c) = \sum_{k=1}^{K} \theta_{c,k} \mathbf{p}_{c,k} = \mathbf{P}_{c,K}\boldsymbol{\theta}_c, \tag{4.6}$$

where $\mathbf{P}_{c,K} = [\mathbf{p}_{c,K}, \mathbf{p}_{c,K-1}, \ldots, \mathbf{p}_{c,1}]$ contains the state vector distributions for the K hops stacked column-wise and $\mathbf{p}_{c,k}$ is the probability state vector for the diffusion of class c after k hops from the starting state. The term $\theta_{c,k}$ is the diffusion coefficient for the kth hop for class c. The vector $\boldsymbol{\theta}_c$ contains the coefficients which characterizes the diffusion for class c. $\boldsymbol{\theta}_c$ is the parameter of interest and is solved for each class [24]; hence, making the method adaptive to each class. The target vector for each class, $\mathbf{y}_{\mathcal{L}_c}$ is

$$[\mathbf{y}_{\mathcal{L}_c}]_i = \begin{cases} \frac{1}{|\mathcal{L}|}, & \text{if } i \epsilon \mathcal{L}_c \\ 0, & \text{otherwise} \end{cases}. \tag{4.7}$$

The set $\mathcal{L} = \bigcup_{c=1}^{C} \mathcal{L}_c$ contains all the labeled nodes. The target vector contains $\frac{1}{|\mathcal{L}|}$ as an entry for all nodes labeled as belonging to class c and zero otherwise. The matrix $\mathbf{D}_{\mathcal{L}}^{-1}$ is an $N \times N$ diagonal matrix defined as

$$[\mathbf{D}_{\mathcal{L}}^{-1}]_{ii} = \begin{cases} \frac{1}{d_i}, & \text{if } i \epsilon \mathcal{L} \\ 0, & \text{otherwise} \end{cases}. \tag{4.8}$$

To solve for the parameter $\boldsymbol{\theta}_c$ for each class, a problem (cost function) has to be formulated with $\boldsymbol{\theta}_c$ as the variable. The function $l(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta}_c))$ measures the normalized square error loss between the target vector $[\mathbf{y}_{\mathcal{L}_c}]_i$ and $[\mathbf{f}_c(\boldsymbol{\theta}_c)]_i$ if the node i is labeled. When i belongs to class c, a low error will be achieved when $[\mathbf{f}_c(\boldsymbol{\theta}_c)]_i$ is close to $\frac{1}{|\mathcal{L}|}$. This error is similar to the labeled error loss usually seen in semi-supervised learning.

For class c, the following problem is solved to find $\boldsymbol{\theta}_c$:

$$\begin{aligned} \underset{\boldsymbol{\theta}_c}{\text{minimize}} \quad & l(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta}_c)) + \mu \mathcal{R}(\mathbf{f}_c(\boldsymbol{\theta}_c)) \\ \text{subject to} \quad & \boldsymbol{\theta}_c \geq \mathbf{0}, \ \ \boldsymbol{\theta}_c^T \mathbf{1}_K = 1. \end{aligned} \tag{4.9}$$

Here, $l(\mathbf{y}_{\mathcal{L}_c}, \mathbf{f}_c(\boldsymbol{\theta}_c)) = (\mathbf{y}_{\mathcal{L}_c} - \mathbf{f}_c(\boldsymbol{\theta}_c))^T \mathbf{D}_{\mathcal{L}}^{-1} (\mathbf{y}_{\mathcal{L}_c} - \mathbf{f}_c(\boldsymbol{\theta}_c))$ is the weighted least squares error. $\mathbf{D}_{\mathcal{L}}^{-1}$ ensures that the error is calculated only at the labeled nodes. The second error term $\mu \mathcal{R}(\mathbf{f}_c(\boldsymbol{\theta}_c))$ is a regularization term on $\mathbf{f}_c(\boldsymbol{\theta}_c)$. By enforcing the cluster assumption [8], nodes belonging to the same class should be in the same cluster; i.e. $[\mathbf{f}_c(\boldsymbol{\theta}_c)]_i$ is similar for nodes in the same class. We use a regularization that penalizes $\mathbf{f}_c(\boldsymbol{\theta}_c)$'s that are not smooth over the graph. Normalized Tikhonov regularization is used with $\mathcal{R}(\mathbf{f}_c(\boldsymbol{\theta}_c)) = \mathbf{f}_c(\boldsymbol{\theta}_c))^T \mathbf{D}^{-1} \mathbf{L} \mathbf{D}^{-1} \mathbf{f}_c(\boldsymbol{\theta}_c)$. μ is the regularization parameter which influences how much a non-smooth $\mathbf{f}_c(\boldsymbol{\theta}_c)$ should be penalized. By substituting these two terms in the cost function, Problem (4.9) becomes

$$\begin{aligned} \underset{\boldsymbol{\theta}_c}{\text{minimize}} \quad & ||\mathbf{y}_{\mathcal{L}_c} - \mathbf{f}_c(\boldsymbol{\theta}_c)||^2_{\mathbf{D}_{\mathcal{L}}^{-1}} + \mu \mathbf{f}_c(\boldsymbol{\theta}_c)^T \mathbf{D}^{-1} \mathbf{L} \mathbf{D}^{-1} \mathbf{f}_c(\boldsymbol{\theta}_c) \\ \text{subject to} \quad & \boldsymbol{\theta}_c \geq \mathbf{0}, \ \ \boldsymbol{\theta}_c^T \mathbf{1}_K = 1. \end{aligned} \tag{4.10}$$

where $||\mathbf{x}||^2_{\mathbf{A}} = \mathbf{x}^T \mathbf{A} \mathbf{x}$ is the norm of vector $\mathbf{x}$ with respect to matrix $\mathbf{A}$. Note that the problem (4.10) is solved for each class c in a one-vs-all fashion and the corresponding $\boldsymbol{\theta}_c$'s are obtained. The unlabeled nodes are classified only after all the $\boldsymbol{\theta}_c$'s are obtained. For each unlabeled node v_j, the value of $[\mathbf{f}_c(\boldsymbol{\theta}_c)]_j$ is compared for all classes $c = 1, \ldots, C$. The class for which $[\mathbf{f}_c(\boldsymbol{\theta}_c)]_j$ is the largest is assigned to v_j as follows:

$$\hat{c}_j = \underset{c=\{1,\ldots,C\}}{\text{argmax}} \ [\mathbf{f}_c((\theta)_c)]_j \tag{4.11}$$

where $\hat{c}_j$ is the class assigned to the unlabeled node v_j.

4.3 Discussion

The assumptions and noteworthy points about this approach are as follows:

1. The knowledge of the target $\mathbf{y}_{\mathcal{L}_c}$ depends on the knowledge of $\mathbf{v}_c$; both of them have non-zero elements in the same indices. This assumes zero landing probabilities for nodes which are not in the same class.

2. The non-zero elements in $\mathbf{y}_{\mathcal{L}_c}$ are $\frac{1}{|\mathcal{L}|}$ at the nodes which are labeled. The target K-step probability vector says that no matter which labeled node of class c one starts from, it is equally probable to land on any of the labeled nodes of that class.

3. In equation (4.4), the value of k runs from $k = 1$ to $k = K$. The value $k = 0$ is ignored.

4. The diffusion parameters of each class are learnt separately.

This concludes our discussion of the semi-supervised classifier. We recall that the cost function in (4.10) has two terms, an error as per the model and a constraint term. $\mathbf{f}_c(\boldsymbol{\theta}_c)$ is a linear function of both $\boldsymbol{\theta}_c$ and $\mathbf{v}_c$, written as

$$\mathbf{y}_{\mathcal{L}_c} = \mathbf{f}_c(\boldsymbol{\theta}_c) = \sum_{k=1}^{K} \theta_{c,k} \mathbf{S}^k \mathbf{v}_c \tag{4.12}$$

For the semi-supervised classifier, from equation (4.12) it is clear that the starting density $\mathbf{v}_c$ must be known. For single-batch pool-based active learning, no knowledge of $\mathbf{v}_c$, and hence $\mathbf{y}_{\mathcal{L}_c}$ is available. To be able to learn, $\mathbf{v}_c$ needs to be estimated, given $\mathbf{S}$ and $\boldsymbol{\theta}$. The one-batch pool scenario has no labeled data as input, so $\mathbf{y}_{\mathcal{L}_c}$ is also unknown. Finding $\mathbf{v}_c$ thus becomes a challenge.

This concludes the discussion on the random walk model and adaptive semi-supervised classifier which utilizes this model. We see that the classifier follows a linear model and is adaptive for each class. In the next chapter, we will focus on developing a methodology to solve the pool-based single batch active node selection problem for this classifier. We will look at methods to select nodes to query which will result in good classification performance.

5 Proposed Active Sensing

This chapter discusses the main contribution of this thesis and introduces the methods adopted to solve the single-batch pool-based active semi-supervised learning for random walk diffusions on graphs. We divide our methods into two broad categories. Methods in the first category solve a regularized inverse problem with sparse constraints — Section 5.1. Methods in the second category do not focus on the inverse problem but solve for the most informative nodes using compressed sensing and sparse sensing — Section 5.2. Section 5.3 mentions approaches based on heuristics and random sampling which we use as a benchmark. Section 5.4 concludes the chapter. Table 5.1 presents

Method	Solves for	Class Name
Proxy-Iterative reweighted l_1 Proxy-Orthogonal matching pursuit	Sparse $\mathbf{v}_c$	Proxy-based deterministic estimation
Sparse Sensing - Active Learning(A design) Sparse Sensing - Active Learning(D design) Compressed Sensing - Active learning	Model Output	Active Sensing
Random Sampling	Nothing	Other approaches

Table 5.1: Overview of approaches considered in this chapter. The first column names the approach used; the second column specifies the underlying objective; the third column categorizes them.

the approaches considered in this thesis. The first column names the method; the second column mentions the objective behind each method; the third column categorizes them on the basis of their objective. There are three categories, namely, Proxy-based deterministic estimation, Active Sensing and Other approaches respectively, which we will discuss in the sequel.

5.1 Proxy-based deterministic estimation.

As discussed in Section 4.2 of Chapter 4, the semi-supervised classifier follows the model [24]:

$$\mathbf{f}_c(\boldsymbol{\theta}_c) = \sum_{k=1}^{K} \theta_{c,k} \mathbf{S}^k \mathbf{v}_c, \tag{5.1}$$

where $\mathbf{f}_c(\boldsymbol{\theta}_c)$ is the classifier output of the K combined landing probabilities for class c, and $\boldsymbol{\theta}_c = [\theta_{c,1}, \ldots, \theta_{c,K}]^T$ is the vector containing the K diffusion coefficients for the cth class. In the one-batch scenario, we jointly select all the relevant nodes from scratch (i.e., without labeled information). From (5.1), we see that the classifier output is linear in $\mathbf{v}_c$.

In this class, we assume the starting node distribution $\mathbf{v}_c$ (for class c) holds the key to understanding which nodes could be important to start the diffusion from. More specifically, we assume that only a few nodes from each class play the most influential role in the diffusion in (5.1); this means $\mathbf{v}_c$ is sparse. For each class, $\mathbf{v}_c$ can be obtained by solving the inverse problem posed by (5.1). To solve this, the landing probabilities $\mathbf{f}_c(\boldsymbol{\theta}_c)$ and labels of all nodes need to be known beforehand, which is indeed what we seek to obtain. The methods in this section operate in two steps: in the first step, the proxies of the landing probabilities are obtained for each class c; in the second step, they are used as observations to solve the inverse problem. Figure 5.1 shows the flowchart of the methods to be discussed in this section.

5.1.1 Obtaining Proxies

Consider the N node classification problem with C classes. We perform Fuzzy-C-Means clustering [cf. Section 2.4.1] on these nodes for C clusters. The number of clusters is assumed to be the number of classes [8]. The nodes are represented by the Laplacian Eigenmap Embedding [cf. Section 2.4.1] up to dimension $C-1$. For each node v_j, t_{jc} denotes the membership value (posterior probability) of node j in cluster c. For each cluster c, the membership values of all nodes are in the vector $\mathbf{t}_c = [t_{1c}, t_{2c}, \ldots, t_{Nc}]^T$. $\mathbf{t}_c$ is a graph signal for cluster (class) c. The normalized version of $\mathbf{t}_c$, $\tilde{\mathbf{t}}_c$ is obtained by dividing it with the sum of its elements. $\tilde{\mathbf{t}}_c$ is a probability mass function defined over the nodes and is used as a proxy for the landing probabilities of class c. (5.1) then rewrites as

$$\tilde{\mathbf{t}}_c \approx \mathbf{f}_c(\boldsymbol{\theta}_c) = \sum_{k=1}^{K} \theta_{c,k} \mathbf{S}^k \mathbf{v}_c, \quad c = 1, \ldots, C. \tag{5.2}$$

Vector $\boldsymbol{\theta}_c$ that weighs each of the K-hop probabilities over the graph. In [24], the assumption is there is a class-specific $\boldsymbol{\theta}_c$. In our scenario, $\boldsymbol{\theta}_c$ is not known for any c. Solving for $\mathbf{v}_c$ and $\boldsymbol{\theta}_c$ simultaneously is challenging. To avoid this, we have taken a fixed $\boldsymbol{\theta}$ for all classes. For a K-hop process, $\boldsymbol{\theta} = (1-\alpha)[\alpha^0 \alpha^1 \ldots \alpha^K]$ with $0 < \alpha < 1$, which is inspired by Personalized Page Rank diffusion [69] [70]. The working model is now

$$\tilde{\mathbf{t}}_c = \sum_{k=1}^{K} \theta_k \mathbf{S}^k \mathbf{v}_c, \quad c = 1, \ldots, C \tag{5.3}$$

We now want to solve for $\mathbf{v}_c$ given $\tilde{\mathbf{t}}_c$ and $\boldsymbol{\theta}$. The value of α can be determined via cross-validation.

5.1.2 Iterative re-weighted l_1 norm minimization

Iterative re-weighted l_1 norm minimization minimizes the weighted l_1 norm of the variable for which sparsity is desired. It solves a series of weighted l_1 minimization problems. In our scenario, this problem becomes

$$\begin{aligned} \underset{\mathbf{v}_c^l}{\text{minimize}} \quad & ||\mathbf{W}^{l-1}\mathbf{v}_c^l||_1 \\ \text{subject to} \quad & ||\tilde{\mathbf{t}}_c - \sum_{k=1}^{K} \theta_k \mathbf{S}^k \mathbf{v}_c^l||_{\mathbf{D}^{-1}}^2 \leq \delta \end{aligned} \tag{5.4}$$

where $\mathbf{v}_c^l$ is the solution obtained at the lth iteration and $\mathbf{W}^{l-1} = \text{diag}(w_1^{l-1}, \ldots, w_N^{l-1})$ contains the weights updated during the $(l-1)$th iteration along its diagonals. The w's are initialized to 1, which makes $||\mathbf{W}^1\mathbf{v}_c^l||_1$ the l_1 norm. Upon solving Problem (5.4), the w's are updated to enhance the sparsity of $\mathbf{v}_c^{l+1}$ and (5.4) is solved again with the updated w's. A detailed analysis of the method is provided in [71]. δ is the permissible tolerance for the error in approximating the landing probability, which is indicated as $||\tilde{\mathbf{t}}_c - \sum_{k=1}^{K} \theta_k \mathbf{S}^k \mathbf{v}_c||_{\mathbf{D}^{-1}}^2 = (\tilde{\mathbf{t}}_c - \sum_{k=1}^{K} \theta_k \mathbf{S}^k \mathbf{v}_c)^T \mathbf{D}^{-1} (\tilde{\mathbf{t}}_c - \sum_{k=1}^{K} \theta_k \mathbf{S}^k \mathbf{v}_c)$, the squared error normalized with the degree matrix $\mathbf{D}$.

The clustering, even though built on reasonable assumptions, is not guaranteed to group nodes which belong to the same class to the same cluster. The term δ specifies how much we want to rely on the clustering and allows us to have some control over the solution. After the minimization has run enough times, the nodes corresponding to the maximum vales of the final $\mathbf{v}_c$ are identified. They are the nodes that contribute the most to the diffusion process for the given proxies and will be queried.

5.1.3 Orthogonal Matching Pursuit (OMP)

Orthogonal Matching Pursuit [57], as seen in Section 2.5.2.1 recovers the sparse solution to a linear under-determined system of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$. In our context, this implies using OMP for solving each $\mathbf{v}_c$, as follows:

$$\begin{aligned} \underset{\mathbf{v}_c}{\text{minimize}} \quad & ||\mathbf{v}_c||_0 \\ \text{subject to} \quad & \tilde{\mathbf{t}}_c = \big(\sum_{k=1}^{K} \theta_k \mathbf{S}^k\big)\mathbf{v}_c, \end{aligned} \tag{5.5}$$

for each $c = 1, \ldots, C$. The difference between this and the l_1 minimization approach is that the OMP algorithm minimizes the l_0 norm while the l_1 minimization minimizes a relaxed version of the original cost. Also, the OMP is faster in execution than SDP-based solvers, which is one of the ways used to solve the iterative minimization problem.

5.1.4 Remarks

This section discusses the assumptions behind this type of approaches.

- The clustering is carried out on one graph. If the graph is naturally defined (through connections, citations), not much can be done about it. However, when the graph is to be built from the data, the question "What is a good enough graph?" comes into the picture. For instance, if we have a data-set and its objects can be part of two two-class classification problems. Then, the same graph may work for one problem but not for the other. In such a case, the graph should be constructed carefully.
- The assumption behind clustering is that nodes which are in the same class should belong to the same cluster in the graph. This is not always guaranteed to happen.
- In each of the methods in Sections 5.1.2 (Iterative re-weighted l_1) and 5.1.3 (OMP), the key idea is that solving C inverse problems independently will give us the desired points.

5.2 Active sensing

In this section, we introduce and motivate the second type of approaches. In the first type (Section 5.1), we were primarily focusing on recovering a sparse $\mathbf{v}_c$ for each class c, given proxies obtained by a fuzzy clustering of the nodes. The main difference here is that we do not care about estimating $\mathbf{v}_c$ directly, nor do we care about finding proxies for the landing probabilities. Instead, we focus on a common diffusion which operates on the labels of all the labeled nodes from all classes together. We propose approaches which select the nodes on which the output of such a diffusion are important for estimating $\mathbf{v}_c$ for all classes together. This simplifies the linear system (5.2) to

$$\mathbf{y} = \mathbf{H}\mathbf{v} + \mathbf{n}, \tag{5.6}$$

where $\mathbf{v}$ contains the information regarding all the important nodes, irrespective of the class and $\mathbf{n}$ is the additive noise. The design of $\mathbf{H} = \sum_{k=1}^{K} \theta_k \mathbf{S}^k$ incorporates the common diffusion assumption with a fixed diffusion parameter vector $\boldsymbol{\theta} = (1-\alpha)[\alpha^0\alpha^1 \dots \alpha^K]$ with $0 < \alpha < 1$, as shown in Section 5.1.1. With this assumption, the active learning approach is not adaptive, unlike the classifier. Each observation $y_i = \mathbf{h}_i^T\mathbf{v} + n_i$ is related to the ith row of $\mathbf{H}$ and node v_i of the graph.

We do not always make assumptions about the nature of $\mathbf{v}$. We propose two methods in this section: the first one based on Compressive Sensing [26] and the second based on Sparse Sensing [27]. Figure 5.2 provides an overview of the active sensing approach.

5.2.1 Compressed Sensing Active Learning (CS-AL)

This section introduces the approach which takes inspiration from compressive sensing. The background material needed has been covered in Chapter 2. Section 5.2.1.1 introduces the mutual coherence concept and its importance; Section 5.13 describes the approach used; Section 5.2.3 comments on the assumptions and remarks on this approach.

5.2.1.1 Mutual Coherence

Compressive sensing deals with recovering signals $\mathbf{x} \in \mathbb{R}^N$ which are sparse with respect to a basis $\mathbf{\Psi} \in \mathbb{R}^{N \times N}$ from subsampled measurements. The vector $\mathbf{x}$ is expressed as $\mathbf{x} = \mathbf{\Psi s}$ where $\mathbf{s}$ is a sparse vector. The measurements $\mathbf{y}$ are obtained as

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s}, \tag{5.7}$$

as shown in [55] [26]. The matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ denotes the sensing matrix with usually $M \ll N$. The system can also be written as $\mathbf{y} = \mathbf{Px}$ with the dictionary $\mathbf{P} = \mathbf{\Phi \Psi} \in \mathbb{R}^{M \times N}$.

The mutual coherence of $\mathbf{P}$, $\mu_{\mathbf{P}}$ is defined as

$$\mu_{\mathbf{P}} = \max_{1<=i,j<=N,\ i \neq j} \frac{|\mathbf{p}_i^T \mathbf{p}_j|}{||\mathbf{p}_i||_2 ||\mathbf{p}_j||_2}, \tag{5.8}$$

with $\mathbf{p}_i$ being the ith column (atom) of $\mathbf{P}$. The mutual coherence quantifies the highest similarity between all pairs of columns (atoms) of $\mathbf{P}$. A high value of $\mu_{\mathbf{P}}$ means that there are at least two columns in $\mathbf{P}$ that are aligned closely. A small value of $\mu_{\mathbf{P}}$ indicates that the two most similar columns are not so much aligned and the remaining pair-wise alignments will be even less aligned. This also implies that the columns of matrix $\mathbf{P}$ are not as redundant (for $M \ll N$, the columns are linearly dependent already). The reason we elaborate on this interpretation of the mutual coherence is because the solution to Problem (2.35) depends on it. If the cardinality of a candidate solution to Problem (2.35) satisfies

$$||s^{\text{candidate}}||_0 \leq \frac{1}{2}\big(1 + \frac{1}{\mu_{\mathbf{P}}}\big), \tag{5.9}$$

then it is necessarily the optimal solution [55].

The highest possible value of mutual coherence is 1. In such a case, the cardinality of the optimal solution would satisfy $||s^{\text{candidate}}||_0 \leq \frac{1}{2}\big(1 + \frac{1}{1}\big) = 1$. This is not informative as we know the sparsest $\mathbf{s}$ for a non trivial $\mathbf{y}$ will always have a cardinality of at least 1. This motivates the use of dictionaries which have low mutual coherence. As a popular example, a $\mathbf{P}$ with each of its elements drawn from an i.i.d Gaussian distribution will have low mutual coherence [56]. In our case, $\mathbf{H}$ in (5.6) plays the same role as $\mathbf{\Psi}$ in (5.7). This leads to a discussion on $\mathbf{\Phi}$ and the following section discusses how $\mathbf{\Phi}$ can be designed, given $\mathbf{\Psi}$, so as to reduce some measure associated with the mutual coherence (reducing the mutual coherence directly is a difficult problem).

5.2.1.2 Projection Matrix Design for Improved Compressive Sensing Performance

This section discusses the idea behind designing a projection matrix $\mathbf{\Phi}$ such that the recovery performance of a Compressive Sensing system as shown in Section 2.5 is enhanced. It first introduces equiangular frames, the Welch bound and some projection matrix design approaches with equiangular frames in mind. We then introduce our method to select rows from the basis matrix which is motivated by the same.

A frame [25] is a set of vectors $\{\mathbf{f}_i\}, i = 1, \ldots, N$, such that there exists constants $0 \leq P, Q \leq \infty$ such that

$$P||\mathbf{x}||_2^2 \leq \sum_{i=1}^{N} | < \mathbf{x}, \mathbf{f}_i > |^2 \leq Q||\mathbf{x}||_2^2,$$

$\forall\ \mathbf{x}$ in a Hilbert space $\mathcal{H}$ where $< \mathbf{a}, \mathbf{b} >$ represents the dot-product between vectors $\mathbf{a}$ and $\mathbf{b}$. When $||\mathbf{f}_i||_2 = ||\mathbf{f}_j||_2\ \forall\ \{i, j\}$, the frame is called an equal norm frame. A frame can also be thought of as a matrix, with the elements $\mathbf{f}_i, i = 1, \ldots, N$ stacked column-wise. An Equiangular Frame (EF) $\mathbf{E}_{M,N}$ of size $M \times N$ is an equal norm frame with norm equal to one and where each element (atom) has the same alignment in magnitude with every other atom. The Gram matrix of such a frame $\mathbf{G}_{M,N} = \mathbf{E}_{M,N}^T \mathbf{E}_{M,N}$ has absolute value entries

$$|G_{i,j}| = \begin{cases} \sqrt{\frac{N-M}{M(N-1)}} & i \neq j \\ 1 & i = j \end{cases}. \tag{5.10}$$

Every pair of non-identical atoms of $\mathbf{E}_{M,N}$ has the same angle between them in magnitude. The term $\sqrt{\frac{N-M}{M(N-1)}}$ is also known as the Welch bound. The columns of $\mathbf{H}$ are each in the M dimensional positive orthant as $H_{ij} \geq 0\ \forall\ \{i, j\}$. Hence, all elements of the gram matrix will be greater than or equal to zero. The ambiguity surrounding the absolute value sign for G_{ij} is lost and the Gram matrix, if it exists, is determined uniquely by

$$G_{i,j} = \begin{cases} \sqrt{\frac{N-M}{M(N-1)}} & i \neq j \\ 1 & i = j \end{cases}. \tag{5.11}$$

We now discuss some popular methods which have aimed at designing $\mathbf{\Phi}$ for a fixed $\mathbf{\Psi}$ so that the resultant dictionary $\mathbf{\Phi\Psi}$ is close to such equiangular frames.

Elad, in his paper [72] finds the optimal $\mathbf{\Phi}$ through an iterative process. $\mathbf{\Phi}$ is initialized with random values. At any particular iteration, the following happens: the Gram matrix is built; next, it shrinks the values of the mutual coherence that are greater than a set threshold in absolute value and retains the smaller values; next, the rank of the modified Gram Matrix is then reduced and the optimal $\mathbf{\Phi}$ is solved for in the least squares sense through the dictionary decomposition. Elad considers the reduction of the t-averaged mutual coherence through this method, instead of the mutual coherence. In [73], the general theory behind designing equiangular tight frames is discussed. To design such a frame, an alternate projection is used. One projection involves projecting upon the space of tight frames, whereas the other projects the Gram matrix on the space of equiangular frames (all values in the Gram matrix which have absolute value more than the Welch bound are clamped to that bound). The method in [74] considers the joint design of $\mathbf{\Phi}$ and $\mathbf{\Psi}$ by assuming the equivalent Gram matrix to be identity (i.e. all columns are orthogonal to each other, which is a special case of an EF).

The common theme across these methods is that $\mathbf{G}_{M,N}$ is not known because of the ambiguity in the sign, which does not allow for a definite problem formulation and encourages iterative approaches based on projections. Also, in [72], Elad evaluates the method on $\mathbf{\Psi}$ of size 80×120 with random entries, while reporting that the reconstruction performance for OMP and Basis Pursuit did not improve for structured bases. In our case, the system matrix $\mathbf{H} = \sum_{k=1}^{K} \mathbf{S}^k \theta_k$ is structured through the shift operator $\mathbf{S}$ and its powers.

We design a sensing mechanism on the system output $\mathbf{y}$ (i.e., model output selection), by selecting the rows of $\mathbf{H}$ through a matrix $\mathbf{\Phi}$ such that the resultant Gram matrix is as close to $\mathbf{G}_{M,N}$ as possible. Each row of $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ has one element equal to one to indicate the selected row and the rest zero. The sensing vector is $\mathbf{w} = \{0,1\}^N$ with $w_i = 1$ if row i is selected and zero otherwise. The matrix $\mathbf{\Phi}$ has two properties: $\mathbf{\Phi}^T\mathbf{\Phi} = \text{diag}(\mathbf{w})$; $\mathbf{\Phi}\mathbf{\Phi}^T = \mathbf{I}_M$. If M rows of $\mathbf{H}$ are to be selected through $\mathbf{\Phi}$, the Gram matrix of the row selected matrix $\mathbf{\Phi H}$ is $\mathbf{H}^T\mathbf{\Phi}^T\mathbf{\Phi H} = \mathbf{H}^T\text{diag}\big(\mathbf{w}\big)\mathbf{H}$ with $||\mathbf{w}_0|| = M$. The selection problem can be written as

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & ||\mathbf{H}^T\text{diag}\big(\mathbf{w}\big)\mathbf{H} - \mathbf{G}_{M,N}||_F^2 \\ \text{subject to} \quad & ||\mathbf{w}||_0 = M, \quad \mathbf{w} \in \{0,1\}^N. \end{aligned} \tag{5.12}$$

Problem (5.12) is a combinatorial NP-hard problem. We can solve it efficiently by substituting the l_0 pseudo-norm $\|\mathbf{w}\|_0 = M$ with the l_1 norm surrogate $\|\mathbf{w}\|_1 = M$ and the Boolean constraint $\mathbf{w} \in \{0,1\}^N$ with the box one $\mathbf{w} \in [0,1]^N$. This transforms (5.12) into a convex problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & ||\mathbf{H}^T\text{diag}\big(\mathbf{w}\big)\mathbf{H} - \mathbf{G}_{M,N}||_F^2 \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = M, \quad \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{5.13}$$

In addition, a different version of Problem (5.13) can also be solved, where we add a regularization cost to the original function as follows:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & ||\mathbf{H}^T\text{diag}\big(\mathbf{w}\big)\mathbf{H} - \mathbf{G}_{M,N}||_F^2 + \gamma_{M,N}||\mathbf{Hw}||_{\mathbf{L}}^2 \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = M, \quad \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}, \end{aligned} \tag{5.14}$$

with $\gamma_{M,N}$ weighing the regularization term which enforces smoothness with respect to the graph on the K-hop diffused density. The optimization constraints ensure that $\mathbf{\Phi}$ is a selection matrix. Once $\mathbf{w}$ is obtained the rows corresponding to the location of the top M values of $\mathbf{w}$ are selected. Relaxing the problem leads often to solutions that are far from the optimal one. We have found instead that solving (5.12) with greedy methods, i.e., starting with the set $\mathcal{V}$ and removing one node at a time that decreases the cost the least until M nodes are left, leads often to better results. As far as we know, it has not been proven to be sub-modular. The regularization controls the diffusion of the selected labels over the graph; e.g., $||\mathbf{Hw}||_{\mathbf{L}}^2 = \mathbf{w}^T\mathbf{H}^T\mathbf{LHw}$ imposes that the diffused labels of the nodes in $\mathbf{w}$ are smooth over the graph. We will investigate the role of $\gamma_{M,N}$ in Section 6.1.5.3.

5.2.2 Sparse Sensing Active Learning (SS-AL)

In this section, we solve our active learning problem using experimental design. The preliminaries on experimental design have been covered in Section 2.2 of Chapter 2. Here, we focus on a sparse sensing approach that provides which observations or nodes to select such that the estimation of $\mathbf{v}$ in (5.6) is optimal in the mean square error and the confidence ellipsoid volume [27]. Additionally, we look at selecting nodes which allow optimal estimation of the classifier prediction, as that is also important from a classification perspective. We do not make assumptions about the nature of $\mathbf{v}$ (i.e. smooth or sparse).

5.2.2.1 A-Design (SS-AL(A))

For sparse sensing, A-experimental design is equivalent to minimizing the mean square error (MSE) of predicting the parameter of interest, which in this case is $\mathbf{v}$ from (5.6). The sensing vector $\mathbf{w}$ is such that $w_i = 1$ if node i is selected and zero otherwise. The noise in (5.6) is white, with each element having the same variance σ^2. The problem for selecting M nodes under these assumptions, as shown in the background chapter is

$$\begin{aligned} \underset{\mathbf{w}\in\{0,1\}^N}{\text{minimize}} \quad & \text{tr}\Big(\sum_{i=1}^{N} w_i \mathbf{h}_i \mathbf{h}_i^T\Big)^{-1} \\ \text{subject to} \quad & ||\mathbf{w}||_0 = M, \end{aligned} \tag{5.15}$$

where $\mathbf{h}_i$ is the ith row of $\mathbf{H}$. Problem (5.15) is not convex. Instead the relaxed version

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \text{tr}\Big(\sum_{i=1}^{N} w_i \mathbf{h}_i \mathbf{h}_i^T\Big)^{-1} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = M \\ & \mathbf{0} \leq \mathbf{w} \leq \mathbf{1}. \end{aligned} \tag{5.16}$$

is a convex problem that can be solved using widely used solvers such as CVX [43]

5.2.2.2 D-Design (SS-AL(D))

The D-experimental design maximizes the log-determinant of the Fisher information matrix [39] of the parameter $\mathbf{v}$. This is equivalent to minimizing the volume of the confidence ellipsoid associated with estimating $\mathbf{v}$ [27]. The sensing variable $\mathbf{w}$ is such that $w_i = 1$ if node i is selected and zero otherwise. The problem for selecting M nodes under these assumptions is stated as

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \log\det\Big(\sum_{i=1}^{N} w_i \mathbf{h}_i \mathbf{h}_i^T + \epsilon \mathbf{I}_N\Big) \\ \text{subject to} \quad & \mathbf{w} \in \{0,1\}^N, \\ & ||\mathbf{w}||_0 = M, \end{aligned} \tag{5.17}$$

where $\mathbf{h}_i$ is the ith row of $\mathbf{H}$. Problem (5.17) is not convex. Its relaxed version is convex [27], but slow to solve with semi-definite programming solvers [43]. However, the log determinant of a positive semi-definite matrix is a sub-modular function [75] and can be maximized greedily i.e. starting with an empty set and by adding one node at a time to this set which increases the cost function the most. To ensure the determinant does not go to zero, the cost function is modified as $\log\det\big(\sum_{i=1}^{N} w_i\mathbf{h}_i\mathbf{h}_i^T + \epsilon\mathbf{I}_N\big)$, where ϵ is chosen to ensure the existence of the determinant. Algorithm 2 summarizes the selection process. The general assumption throughout literature is that the rows of the system matrix $\mathbf{H}$ span $\mathbb{R}^n$, which is not always true. In these situations, when adding the nodes greedily, we consider only those nodes whose corresponding row in $\mathbf{H}$ increases the rank with respect to the already selected rows.

5.2.2.3 Minimizing Variance of Prediction through Sparse Sensing

So far, we have focused sparse sensing tools on the estimation of $\mathbf{v}$ in (5.6). However, while estimating $\mathbf{v}$ is important, the classification depends on the diffused version of $\mathbf{v}$, $\mathbf{Hv}$, where $\mathbf{H}$ represents the $N \times N$ diffusion operator or graph filter. Here, we focus on estimating the classifier output and the equivalent sparse sensing problem. This is similar to transductive active learning [12]. Under the same model assumptions (i.e., additive white noise), the error covariance matrix will be

$$\begin{aligned}\mathbb{E}\big((\mathbf{Hv}-\mathbf{H}\hat{\mathbf{v}})(\mathbf{Hv}-\mathbf{H}\hat{\mathbf{v}})^T &= \mathbb{E}(\mathbf{H}(\mathbf{v}-\hat{\mathbf{v}})(\mathbf{Hv}-\hat{\mathbf{v}})^T\big)\\ &= \mathbf{H}\mathbb{E}\big((\mathbf{v}-\hat{\mathbf{v}})(\mathbf{v}-\hat{\mathbf{v}})^T\big)\mathbf{H}^T\\ &= \mathbf{H}\big(\sum_{i=1}^{N}\frac{w_i}{\sigma_i^2}\mathbf{h}_i\mathbf{h}_i^T\big)^{-1}\mathbf{H}^T\\ &= \mathbf{H}\big(\frac{1}{\sigma^2}\mathbf{H}^T\mathrm{diag}(\mathbf{w})\mathbf{H}\big)^{-1}\mathbf{H}^T,\end{aligned} \tag{5.18}$$

where $\mathbf{w}$ is the sensing vector. We are interested in selecting the nodes which will help estimate the classifier output optimally. The problem of selecting these nodes has been shown to be [27]

$$\begin{aligned}\underset{\mathbf{w}}{\text{minimize}} \quad & \log\det\ \mathbf{H}\big(\mathbf{H}^T\mathrm{diag}(\mathbf{w})\mathbf{H}\big)^{-1}\mathbf{H}^T\\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = M,\\ & \mathbf{0} \le \mathbf{w} \le \mathbf{1}.\end{aligned} \tag{5.19}$$

When $\mathbf{H}$ is invertible, the outer $\mathbf{H}$ and $\mathbf{H}^T$ in (5.19) can be ignored and we can focus on the term $\big(\mathbf{H}^T\mathrm{diag}(\mathbf{w})\mathbf{H}\big)^{-1}$, given the sparse sensing constraints. Minimizing the log determinant of this term is same as maximizing the log determinant of $\big(\mathbf{H}^T\mathrm{diag}(\mathbf{w})\mathbf{H}\big)$. This is the problem we solve in SS-AL(D) in Section 5.2.2.2. It appears that sensing for optimal estimation of the starting density is equivalent to sensing for estimating the optimal classification output when the system matrix $\mathbf{H} = \sum_{k=1}^{K}\theta_k\mathbf{S}^k$ is invertible. Under this assumption, solving for D-optimality is enough. However, there are no guarantees that $\mathbf{H}$ is full rank. During simulations, we did notice that in most cases, $\mathbf{H}$ was full rank, especially for real data.

Algorithm 2 Greedy log-determinant maximization

Require: $\mathcal{L}^*$,$|\mathcal{L}^*| = M$
Input: Graph $\mathcal{G}$, node set $\mathcal{V}$, matrix $\mathbf{H}$, number of nodes M
Output: Set of selected nodes $\mathcal{L}^*$
1: **Initialization** $\mathcal{L}^* = \emptyset$, $i = 0$
2: **while** $i \leq M$ **do**
3: j=$\underset{k \in \mathcal{V}-\mathcal{L}^*}{\text{argmax}} \log\det\big(\sum_{i\in\mathcal{L}^*} \frac{1}{\sigma_i^2}\mathbf{h}_i\mathbf{h}_i^T + \frac{1}{\sigma_k^2}\mathbf{h}_k\mathbf{h}_k^T + \epsilon\mathbf{I}_N\big)$,
4: $\mathcal{L}^*$=$\mathcal{L}^* \cup j$,
5: $i = i + 1$,
6: **end while**

5.2.3 Remarks

We are now in a position to critically remark on the CS-AL (Section 5.2.1) and SS-AL (Section 5.2.2) approaches.

1. For CS-AL, the existence of an EF is not guaranteed for any combination of (M, N). Finding N equally aligned points in $\mathbb{R}^M$ is a difficult problem. There are some conditions for which a real $M \times N$ frame is equiangular [25]. One such condition is $N \leq \frac{M(M+1)}{2}$.

2. The sensing vector $\mathbf{w}$ is used at the output of the system to sense the node outputs that enable reconstruction of the sparse starting state vector. Let $\mathcal{W}$ denote the set containing the indices of the selected rows. The row selected version of $\mathbf{H}$, $\mathbf{H}_\mathcal{W}$, when used with the observations $\mathbf{y}_\mathcal{W}$ to estimate a sparse $\mathbf{v}$ (i.e. through OMP) is not guaranteed to give a $\mathbf{v}$ which is element-wise non-negative and sums to one. The assumption requires it to be a distribution, but the model-based compressive sensing approach is not guaranteed to give a distribution. This may introduce some mismatch and could influence the results. In that case, we do not rely on the landing probabilities but the filtered version of a sparse graph signal, which has positive or negative signal values at the desired nodes. In case of SS-AL, when $\mathbf{H}_\mathcal{W}$ is used to estimate $\mathbf{v}$, it is not guaranteed to give a $\mathbf{v}$ which sums to one over all nodes.

3. Through different backgrounds, both CS-AL and SS-AL arrive at a row selection procedure.

5.3 Other approaches

This section contains some methods which will be used as a reference to compare the methods proposed in the previous two sections. One basic approach is to sample nodes throughout the graph at random and feed them to the classifier. This selects nodes without taking the structure or the diffusion phenomenon into account.

We also considered the heuristic which involved selecting the nodes with the highest degree, as it would seem that they would have more influence on the propagation of the probabilities. However, this approach leads to a poor performance and we did

not include it in this report. We also sampled according to the membership functions returned for each cluster in Section 5.1. For each cluster, we selected the nodes having maximum membership. This is a naive way of output selection and did not work for reasons we shall see in the next chapter.

5.4 Conclusion

In this chapter, we have proposed two sets of approaches to the active learning problem. The first tries to find a small set of nodes per class which are important for matching proxies of the landing probabilities per cluster on the graph. The second tries to find nodes on which to sample the output of the system (ideally landing probabilities) so that the initial starting density can be estimated optimally. We are now in a position to evaluate these methods on synthetic and real datasets, which we will do in the next chapter.

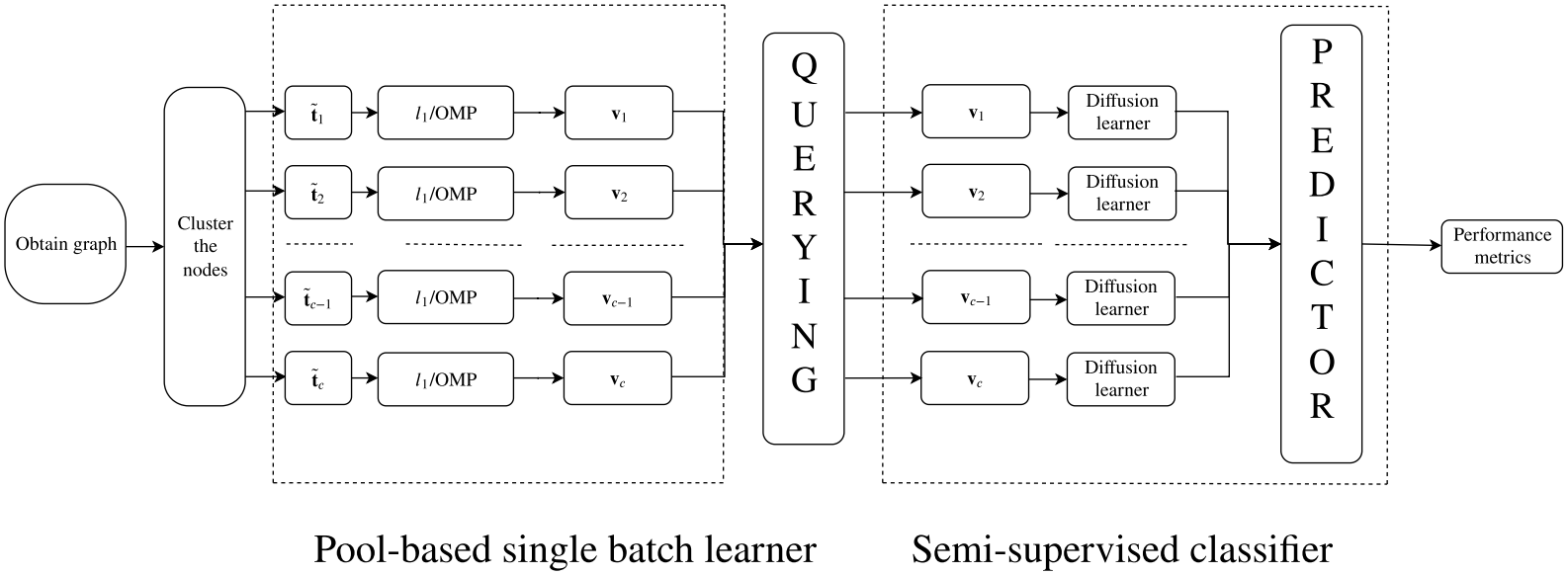


Figure 5.1: Flowchart of the proxy-based method: The pool-based single batch learner consists of a clustering phase, which generates the proxies of landing probabilities $\tilde{\mathbf{t}}_c$; these are then used to solve for a sparse starting density $\mathbf{v}_c$ through a regularized inverse solver; the nodes obtained from the starting density are then sent through a querying step where their labels are extracted; the labeled data is then used as the training set for the adaptive diffusion-based semi-supervised classifier (i.e. the diffusion coefficients are learned for each class and then used to classify the unlabeled nodes).

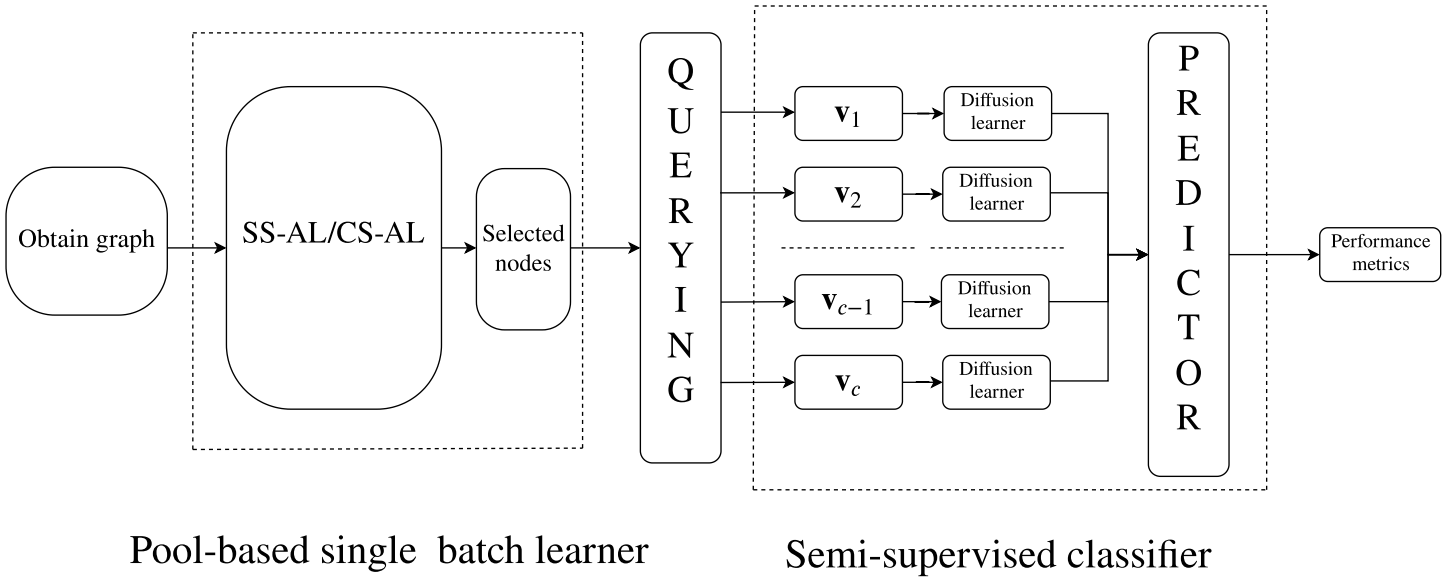


Figure 5.2: Flowchart of the active sensing method: The pool-based single batch learner consists of either the SS-AL (Sparse Sensing-Active Learning) approach or the CS-AL (Compressed Sensing- Active Learning) approach for selecting the nodes with the most relevant model output; the most relevant nodes obtained are then sent through a querying step where their labels are extracted; the labeled data is then used as the training set for the adaptive diffusion-based semi-supervised classifier (i.e. the diffusion coefficients are learnt for each class and then used to classify the unlabeled nodes).

6 Results

In this section, we evaluate the approaches developed in the previous chapter on two synthetic graphs and two graphs from real data. Section 6.1 introduces each graph and the performance of the methods on them; Section 6.1.5 showcases some properties of selected methods; Section 6.2 contains the results obtained for real data-sets.

6.1 Simulated data

We test the active learning approaches on two simulated networks: a stochastic block model and a random sensor network.

6.1.1 Stochastic Block Model

The Stochastic Block Model (SBM) generates graphs which have some community structure. A community is a well-clustered set of nodes in a graph which unites nodes having some degree of similarity. A stochastic block model is generated using the following parameters:

1. Number of nodes N,
2. The community set $\mathcal{C}_1, \ldots, \mathcal{C}_R$ with R denoting the number of communities,
3. The $R \times R$ edge probability matrix $\mathbf{P}_e$ with $[\mathbf{P}_e]_{ij}$ denoting the probability of an edge existing between a node from community $\mathcal{C}_i$ and a node from community $\mathcal{C}_j$. There are two probabilities involved: the inter and intra-class probabilities.

Given the parameters, each edge is sampled at random to generate an instance of the stochastic block model. We generated such graphs using the graph signal processing toolbox [76]. We assume the inter and intra-cluster probabilities to be fixed (i.e. $\mathbf{P}_e$ has equal diagonal and off-diagonal elements). For our simulations, we consider such a network with $N = 200$ nodes and $C = 4$ communities with inter and intra-cluster edge probabilities of 0.01 and 0.8 respectively is shown in the following figure. These parameters will be used in the evaluation process. A realization of this is shown in Figure 6.1.

6.1.2 Random Sensor Network

The random sensor network comprises sensors nodes arranged at random in a rectangular area. We considered a connected random sensor network for with $N = 200$ nodes and $C = 4$ classes. A realization of this is shown in Figure 6.2.

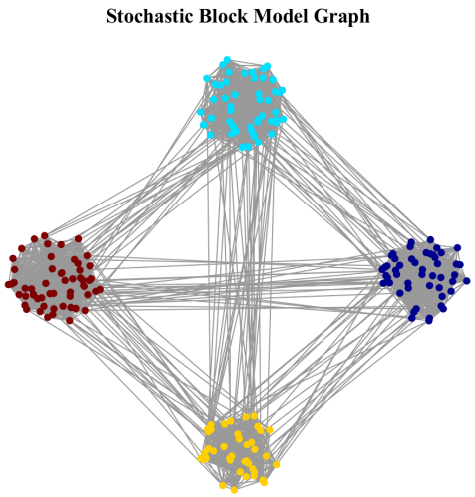


Figure 6.1: A stochastic block model network of $N = 200$ nodes with $C = 4$ classes.

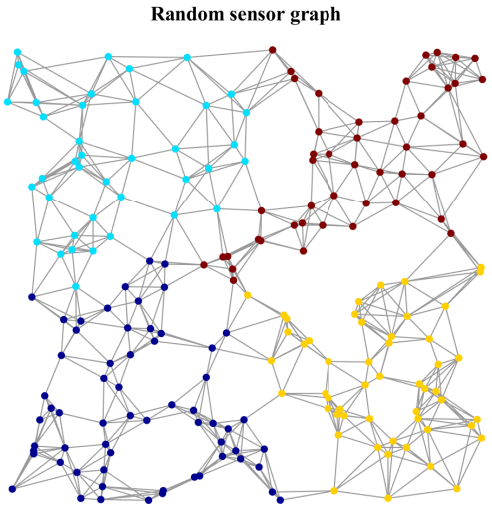


Figure 6.2: A random sensor network of $N = 200$ nodes with $C = 4$ classes.

6.1.3 Experimental Setup

Each simulated graph has $N = 200$ with $\mathcal{C} = 4$. We consider two values for the filter or diffusion order K. We take K to be the diameter of the graph or its half, to illustrate its effect. The value of K is 4 for Stochastic block model and is around 15 for the random sensor network. We consider μ for the semi-supervised classifier in (4.9) to be 1. We take the diffusion weights of the form $\boldsymbol{\theta} = (1 - \alpha)[\alpha^0 \alpha^1 \dots \alpha^K]$ for all

graphs with $\alpha = 0.9$. We consider 10 realizations for each graph to obtain the average performance. To assign labels we clustered the nodes as per fuzzy C-means clustering on the Laplacian embedding and then label the points according to the cluster to which they belong. In this way, we obtained labels that are smooth over the graph. For the CS-AL approach (5.2.1), we take $\gamma_{M,N} = 0$. We did not try to optimise for gamma in terms of classification error for these plots. The solution to the iterative re-weighted l_1 (5.1.2) and SS-AL using A-experimental design (5.2.2.1) was carried out using CVX [43]. For iterative re-weighted l_1, we iterate 10 times and take $\delta = 0.1$. We average the results for random selection over 200 iterations.

6.1.4 Results

Figures 6.3 and 6.4 show the classification error plots for the stochastic block model, while Figures 6.5 and 6.6 show the same for the random sensor network.

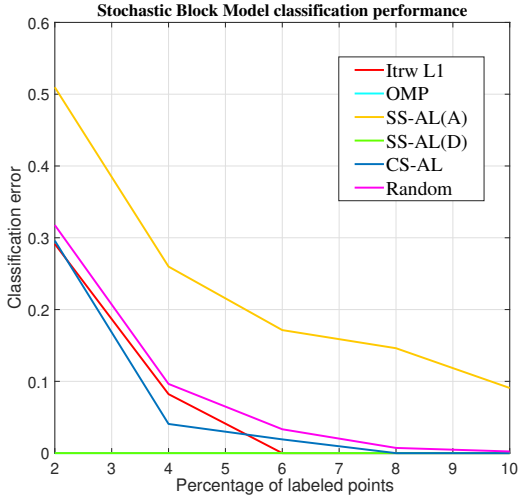


Figure 6.3: Classification error vs. percentage of labeled points for six active learning methods on the Stochastic block model graph of 200 nodes and four classes. The diffusion order is the graph diameter.

Figures 6.3 and 6.4 show the mean classification error as a function of the percentages of labeled points for two scenarios of the SBM graph. Figure 6.3 considers the diffusion order K to be the graph diameter while Figure 6.4 considers K to be half of the diameter. There are six curves, five of those for the active learning methods proposed in Chapter 5 and one for the random selection. We observe the method based on proxies of landing probabilities and OMP (5.1.3) and D-experimental design (5.2.2.2) obtain perfect classification, even for as less as 2% of the data, which equals 4 points. This is possible as the graph has four well-structured clusters (Figure 6.1) and sampling one node per cluster is good enough. SS-AL with D experimental design manages to do this without relying on any proxies. The CS-AL approach does not perform as well as the previous two. We believe this is because it suffers from the condition which requires equiangular frames to exist (5.2.3). The equiangular frame condition is satisfied when

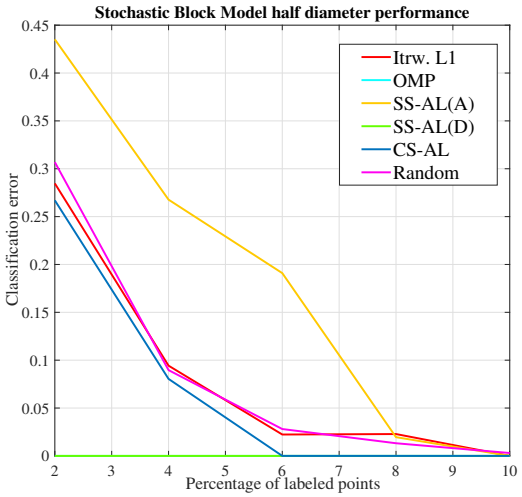


Figure 6.4: Classification error vs. percentage of labeled points for six active learning methods on the Stochastic block model graph of 200 nodes and four classes. The diffusion order is half of the diameter of the graph.

20 or more labels are queried (i.e. 5% of the data). From the figures, the performance does move a lot closer to the optimal classification performance (zero error) after the 5% mark. The performance can be further improved with incorporating $\lambda_{M,N}$ in (5.14) but here we show how the approach holds up. The proxy-based l_1 solution does not do as good but still better than random selection. The SS-AL method on A-experimental design (MSE) performs the worst.

However, we do note an increase in classification accuracy for all the methods apart from the optimal ones when K is half the diameter i.e. $K = 2$. A hop of 2 prevents the unlabeled nodes from being affected by diffusion from other classes. The effect of diameter is more prevalent on this graph, but it is not so easy to conclude for other graphs, as we shall see later. This graph was chosen for its obvious structure and to test how the different methods are capable of picking up nodes from each cluster.

Figures 6.5 and 6.6 display the same curves of classification error as a function of the percentage of labeled points for the random sensor network. The sensor network has four classes which are not well-separated from each other (see Figure 6.2). It is interesting to see how the methods behave for such a graph as getting optimal classification (zero error) is challenging. The proxy-based OMP method outperforms the rest when the number of nodes to be queried is limited but the error does not reduce much when more samples are collected. This could point us to a possible drawback of this approach, which we discuss more into detail in Section 6.1.5.2. The CS-AL and SS-AL(D) plots do better overall than the rest, especially when the diffusion order is half the diameter. This throws some light upon the role of K for active learning. K influences the process in two ways: i) by influencing the contents of matrix $\mathbf{H} = \sum_{k=1}^{K} \mathbf{S}^k \theta_k$ and ii) determining the extent of spread with respect to the graph structure. It is not obvious if increasing K will lead to a poorer performance for any graph. Moreover, each class has its own sub-graph with its own intrinsic diameter

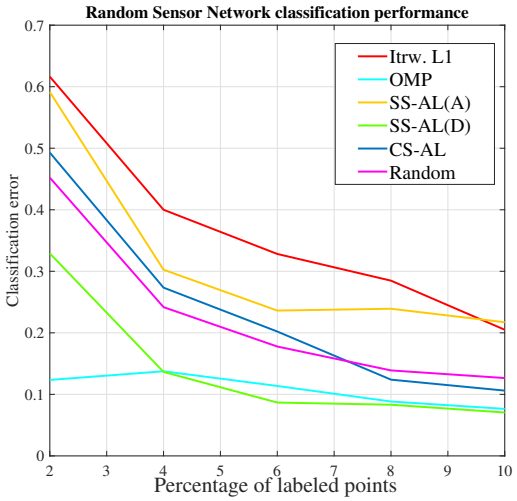


Figure 6.5: Percentage of labeled points vs. Classification error for six active learning methods on the Sensor network graph of 200 nodes and four classes. The diffusion order is the diameter of the graph.

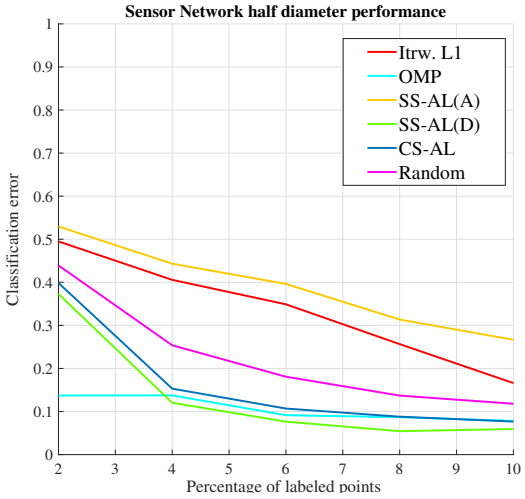


Figure 6.6: Percentage of labeled points vs. Classification error for six active learning methods on the sensor network graph of 200 nodes and four classes. The diffusion order is half of the diameter of the graph.

and ideally each class should diffuse to an extent which covers only its members. This classifier assumes a fixed order of diffusion for each class. The fixed number will be more than the internal diameter for some classes and less for the rest. So it is difficult to predict how a change in K will hamper the performance unless there is some clear structure in the graph and the class-specific diameters are known.

Having examined the performance on these two graphs, we see a trend of three methods (proxy-based OMP, SS-AL(D) and CS-AL) performing better than the random

selection which is the purpose behind active learning. We will continue to look into the strengthens and weaknesses of these methods in more detail in the next few subsections.

6.1.5 Experiments for particular algorithms

In this section, we elaborate further on different aspects of the methods which performed well in the previous section: namely, the CS-AL, SS-AL(D), and proxy-based OMP solution.

6.1.5.1 Greedy selection

The SS-AL(D) approach selects points exploits the log determinant function's submodularity and greedily selects points. In this portion of this chapter, we look at the nature of this selection. We perform active learning to select nodes equalling the number of classes, which is 4 in all cases so far. We show the sequence of selection of the first four points for the SBM network, a modified version of it and the sensor network as seen before.

Figure 6.7 shows the first four nodes selected by SS-AL(D) sequentially. Each of these nodes is from one cluster. This illustrates the effectiveness of this method. The SBM graph has close to an equal number of nodes per class. Next, we change the community sizes to see if it changes the nature of selection. We generate a community graph with 4 communities of size 10, 70, 70 and 100 respectively. The results for this is shown in Figure **??**. It is seen that the learner selects one node from each class for the first four selected nodes.

6.1.5.2 OMP plots

In the results of simulated graphs, we noticed that the proxy-based OMP method was among the best performers. It achieved optimal classification for the SBM and outperformed all methods for a very low percentage of queried points on the sensor network (see Figs 6.5,6.6). The method depends on the proxies of landing probabilities, which are described in Section 5.1.3. The proxies were obtained in such a way that each proxy would give higher membership values around one cluster and low for the rest. This is possible for the stochastic block model graph and the sensor network for 4 classes. We suspect that with an increase in the number of classes, the membership functions would appear more similar and this would limit the potential of the OMP algorithm in terms of the number of unique points it can generate from each proxy. To test this, we plot the difference between the number of requested queries and the number of unique queries generated by this method for the same graph and observe the trend in increasing the number of classes. For example, in a graph of 200 nodes and 4 classes, if 2 queries are requested per class, we would require 8 labeled nodes and the algorithm should ideally be able to provide the same number.

Figures 6.9a and 6.9b showcase the results on the sensor network. Figure 6.9a plots the difference between the number of labels requested and the number of unique labeled objects returned by the method, averaged over 10 iterations. The plot captures the effect of increasing the number of classes. It is clear that with an increase in

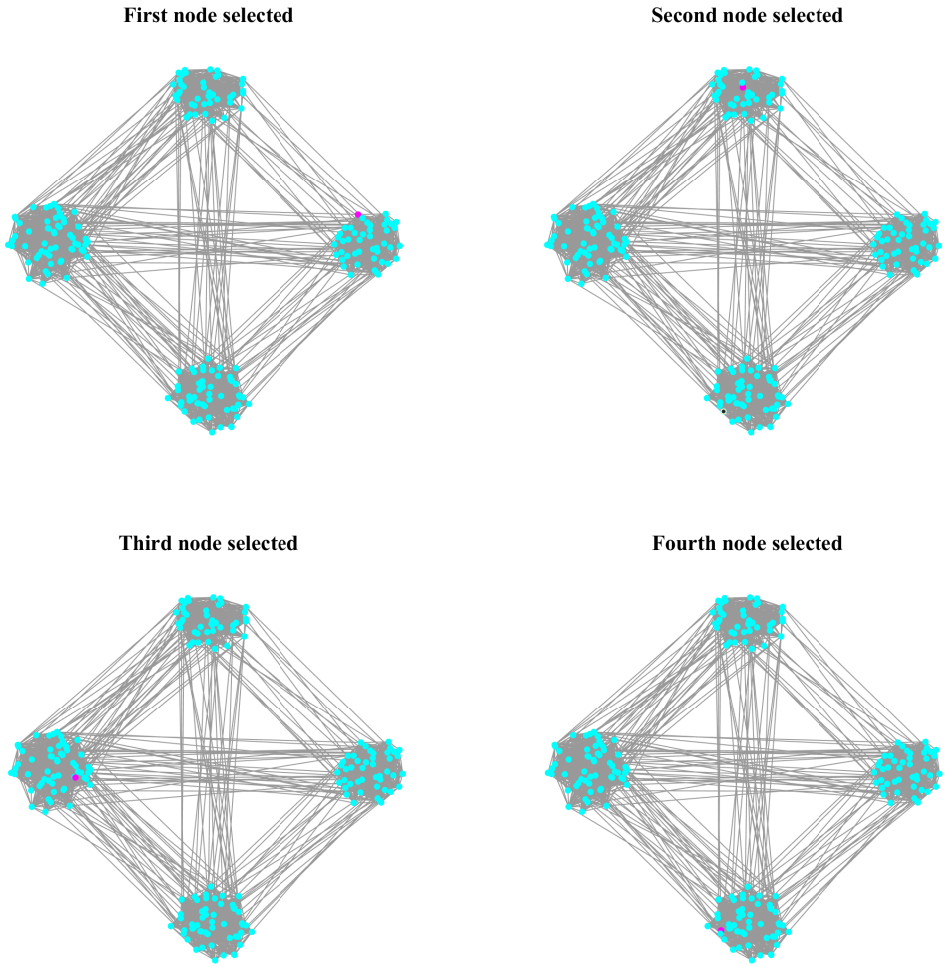


Figure 6.7: The first (top left), second (top right), third (bottom left) and fourth(bottom right) nodes selected by SS-AL(D) for a Stochastic Block Model graph. The selected nodes are denoted in magenta and the rest in cyan.

the number of classes, this difference increases. A high difference means fewer points being selected, and this will lead to an inferior performance compared to methods which guarantee the selection of the same number of unique labels as requested, like the SS-AL(D) approach. Figure 6.9b showcases the effect of this difference on the mean classification error. When the number of desired queries is higher, the error for 5 and 6 classes is higher than for 3 or 4 classes. The error curves do not reduce in the same manner with the increase in desired queried labels. Figures 6.10a and 6.10b showcase the difference in performance between the proxy-based OMP methods and SS-AL(D) (i.e. a method which returns the same number of points as requested) for a higher number of classes than shown in Section 6.1.4. We take the same graphs i.e. the stochastic block model and the random sensor network with 200 nodes but this time with 6 classes. The mean classification error performance is compared between

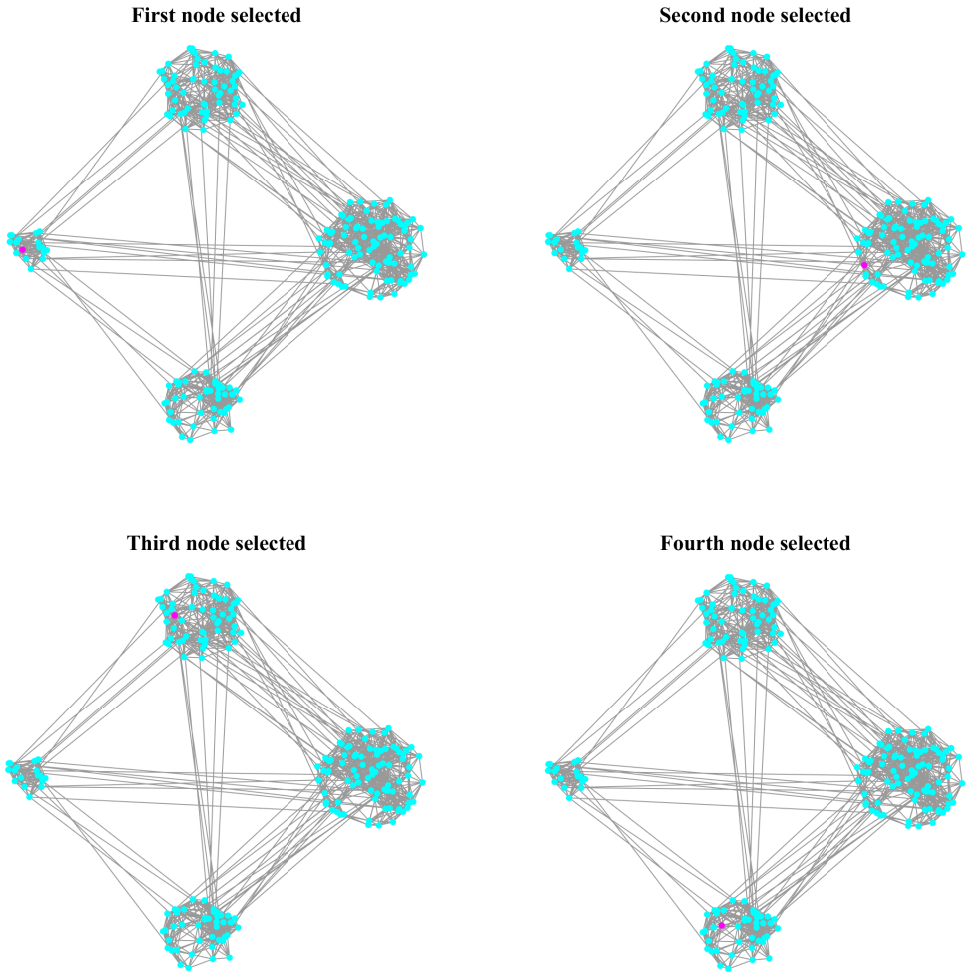
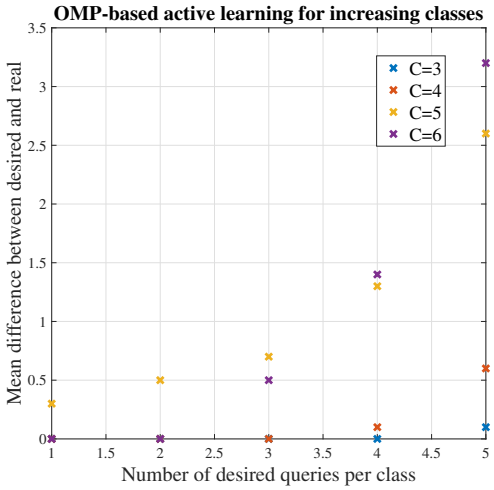
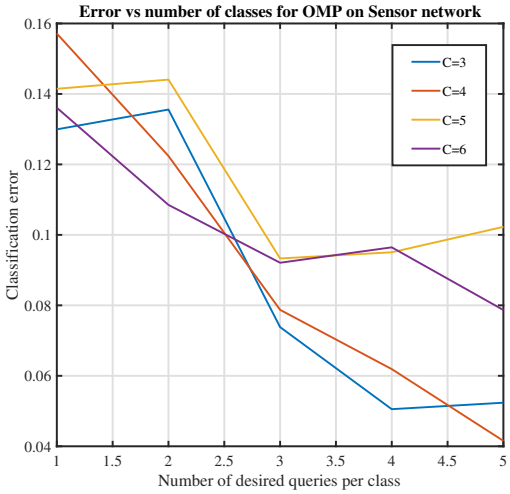


Figure 6.8: The first (top left), second (top right), third (bottom left) and fourth(bottom right) nodes selected by SS-AL(D) for a skewed Stochastic Block Model graph of community size 10, 70, 70 and 100. The selected nodes are denoted in magenta and the rest in cyan.

the two methods. In Figure 6.10a, SS-AL(D) can still obtain optimal performance for all sets of queries while proxy-based OMP struggles to pick up nodes from each community for a small number of requested queries. Figure 6.10b shows a similar trend for sensor networks. Due to the generated proxies being similar, the OMP method cannot distinguish between them and generate sparse signals which are unique for each proxy. As a result, the corresponding error curve does not go down with an increase in the number of labeled points requested, whereas the SS-AL(D) method performs better in the long run.



(a) Mean difference in number of points requested and number of points returned.

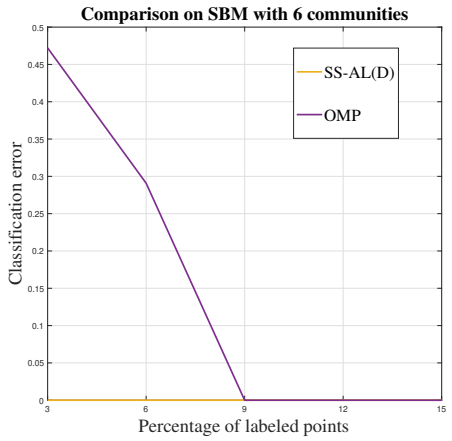


(b) Classification error vs. number of desired points per class for increasing number of classes.

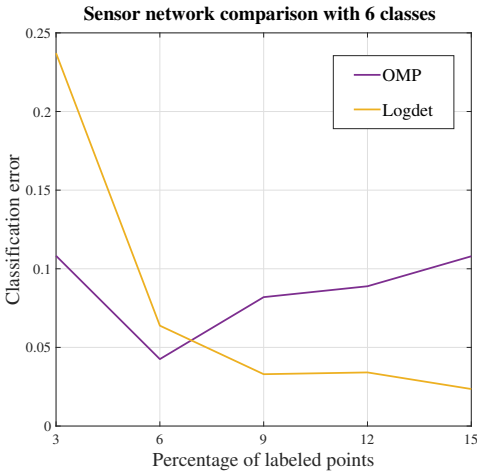
Figure 6.9: Label Propagation. 2.3a shows the original labels, 2.3b shows one selected fro each class with their label value, 2.3c, 2.3d show how the propagation takes place through colours, 2.3e shows the final values after propagation concludes, 2.3f shows the predicted labels after propagation.

6.1.5.3 CS-AL plots

In this section, we examine the role of $\gamma_{M,N}$ as introduced in Section 5.2.1. The term $\gamma_{M,N}$ weighs the regularization terms which penalizes the smoothness of the signal obtained by diffusion of the queried labels. We vary $\gamma_{M,N}$ over four values, 0, 1, 10 and 20. $\gamma_{M,N} = 0$ corresponds to no regularization. The results are obtained for the SBM and sensor graphs with 200 nodes and 4 classes. We want to check if adding this constraint helps in selecting a better-labelled set. Figure 6.11a shows the comparison between different values of $\gamma_{M,N}$ for different values of M. For M varying from 2% to 4% and M beyond 8% of the total nodes, the higher values of $\gamma_{M,N}$ gives better mean classification error. Between 4% to 8%, lower $\gamma_{M,N}$ performs slightly better. This means when the number of selected nodes is very low, a harsher penalty needs to be imposed on its diffused version. The nature of this plot is not revealing much about the existence of an optimal $\gamma_{M,N}$. Moreover from the nature of problem (5.14), for a set (M, N), there should ideally be one $\gamma_{M,N}$. So, the optimal gamma can depend on M. We cannot draw any clear conclusion about the optimal $\gamma_{M,N}$ from this graph. Figure 6.11b, however, provides more clarity on this. Here, $\gamma_{M,N}$ is optimal for all M when it is zero (i.e. no smoothing is preferred). This is due to the graph structure (Figure



(a)



(b)

Figure 6.10: Label Propagation. 2.3a shows the original labels, 2.3b shows one selected fro each class with their label value, 2.3c, 2.3d show how the propagation takes place through colours, 2.3e shows the final values after propagation concludes, 2.3f shows the predicted labels after propagation.

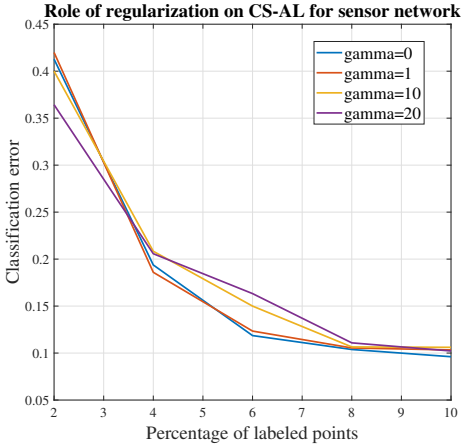
6.1). The diffused version of the graph signal for the selected nodes up to $M = 6\%$ is already smooth with respect to the graph structure and penalizing it is not useful. We think the structure of the graph has a say on the role of $\gamma_{M,N}$.

6.2 Real Data

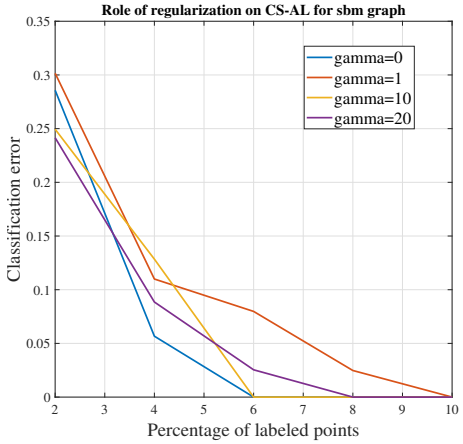
6.2.1 Facebook egonet subnetwork

The Facebook egonet subnetwork is a small portion of the ego network found in [77]. The ego network consists of 4039 nodes and is made up of circles and friend-lists of people on Facebook. The data-set contains anonymous features (i.e. values only) for each user, circles and ego networks. We choose a sub-network of $N = 234$ nodes, with two communities and a diameter of 8. There are no labels available *per se* as the graph is usually analysed for communities and circles [1]. To provide some labels, we label the graph the same way we labeled the graphs in the simulated section (c.f. 6.1). Upon doing that we obtain two classes with strength 219 and 15 respectively. We then use the methods which performed well in the previous section, namely proxy-OMP, SS-AL(D) and CS-AL to select relevant nodes.

Figure 6.12 shows the comparison between the proxy-OMP, SS-AL(D) and the CS-AL approaches on a section of the Facebook ego network. The value of $\gamma_{M,N}$ for CS-AL is taken 0 to evaluate the performance of the idea and not necessarily any improvements brought upon by regularization. Also, The $\mathbf{G}_e$ used for CS-AL in this example is the identity matrix $\mathbf{I}_N$, which is also an equiangular frame and the nodes are selected greedily. The random selection was run 200 times. Figure 6.12a and 6.12b



(a) Role of γ of CS-AL on classification error for sensor network.



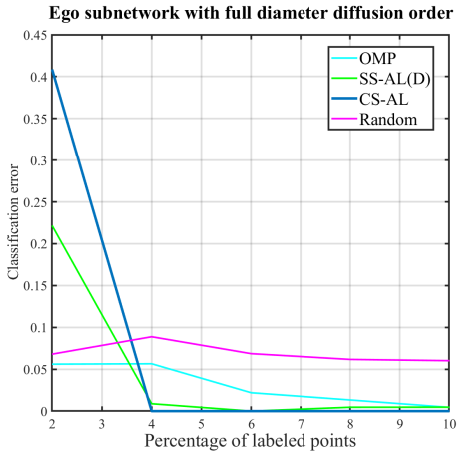
(b) Role of γ of CS-AL on classification error for stochastic block model network.

Figure 6.11: Figures 6.11a and 6.11b showcase the effect of regularization on the node selection and ultimately, the classification error. In 6.11a there is no gamma which is optimal throughout. For the stochastic block model in Fig 6.11b, the results indicate $\gamma = 0$ is overall a good choice (i.e. regularization is not desired).

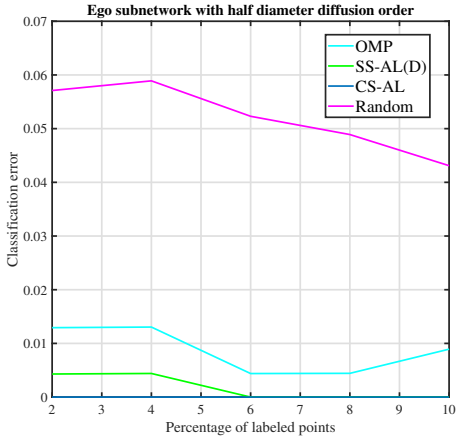
	Class 1				Class 2			
	CS-AL	SS-AL(D)	Random	proxy-OMP	CS-AL	SS-AL(D)	Random	proxy-OMP
Class 1	214	214	208.6	215	0	0	5	1
Class 2	0	0	7.5	0	14	14	6.9	12

Table 6.1: Confusion matrix for the proposed CS-AL and SS-AL, random labeling, and degree-based labeling on the Facebook subnetwork for $|\bar{\mathcal{V}}| = 6$ and filter order $K = 4$. Each row shows how the different algorithms classify the nodes belonging to that class. .

show the classification error plots obtained for K equal to the graph diameter and its half, respectively. For K half of the diameter, CS-AL performs optimal classification, followed by SS-AL(D) and proxy-OMP. All methods outperform random selection For K equal to the graph diameter, the CS-AL and SS-AL(D) method does poorly for $M = 2\%$ of the graph, but outperforms the rest for $M \geq 4\%$. Also, the performance for K equaling half the diameter is better. This is attributed to one class being larger in size, so a higher value of K would mean many points from this class would be classified into the smaller class.



(a) Comparison of error performance of the active learning methods on the Facebook egonet graph with diffusion order equal to the graph diameter.



(b) Comparison of error performance of the active learning methods on the Facebook egonet graph with diffusion order equal to half of the graph diameter.

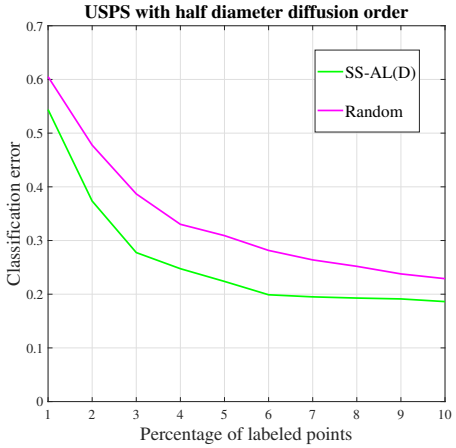
Figure 6.12: Comparison between the proxy-OMP, SS-AL(D), CS-AL and the random approaches on a section of the Facebook ego network with $N = 234$ nodes and $C = 2$ classes defined as per the structure. Figure 6.12a and 6.12b show the classification error plots obtained for K equal to the graph diameter and its half, respectively. The $\mathbf{G}_e$ used for CS-AL in this example is the identity matrix $\mathbf{I}_N$. For K half of the diameter, CS-AL performs optimal classification, followed by SS-AL(D) and proxy-OMP. For K equal to the graph diameter all methods outperform random selection except when the labeled set is 2% of the graph.

Table 6.1 shows the confusion matrix for each method for the ego sub-network. The data corresponds to the case where $K = 4$ (i.e. half of the diameter). For each method,the cell (i, j) denotes the number of nodes belonging to class i that are classified to class j. These results confirm those in Figure 6.12b. CS-AL and SS-AL(D) classify all points correctly in this scenario, as the off-diagonal elements in their confusion matrix are zero.

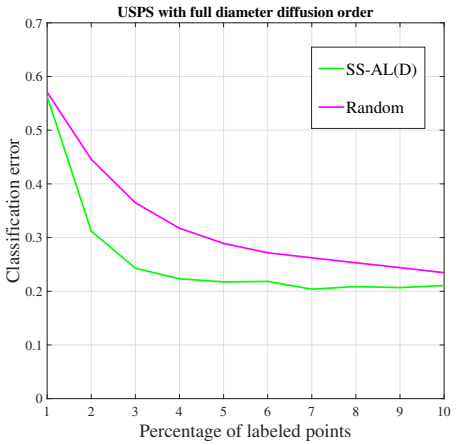
6.2.2 USPS Dataset

The USPS data-set comprises images of size 16×16 representing hand-written digits from 0 to 9 as its objects. It has 1100 objects per class. We sample 100 objects per class at random and build a graph of $N = 1000$ nodes. To build the graph, we obtain a similarity measure $W_{ij} = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||_2^2}{\sigma^2}\right)$, where $\mathbf{x}_i$ is the 256-dimensional feature vector for node i, formed of the pixel intensity values of the corresponding image. The parameter σ is taken to be one-third of the average distance to the P-th nearest neighbor for all objects. The approach behind selecting σ this way is shown in [8]. We

take P to be 5. We obtain our results over 5 realizations. We observe that the diameter of the graph is around 10. We repeat the results for random selection 50 times. Figures



(a) Comparison of error performance of the active learning methods on USPS data with diffusion order $K = 5$



(b) Comparison of error performance of the active learning methods on USPS data with diffusion order $K = 10$.

Figure 6.13: Comparison between the SS-AL(D) and the random selection approaches on the USPS data with $N = 1000$ nodes and $C = 10$ classes. Figure 6.13a and 6.13b show the classification error plots obtained for $K = 10$ and $K = 20$, respectively. In both plots, SS-AL(D) outperforms random labeling. There is not much of a difference in performance for the two values of K.

6.13a and 6.13b showcase the improvement in performance achieved by SS-AL(D) over random labeling in the range of 1% to 10% of labeled data. We omit the results due to proxy-OMP because they were not good compared to both of these methods. The performance for both values of K is similar. This shows that changing K does not always predict the shift in the performance of the classifier.

6.3 Conclusion

In this chapter, we evaluate the proposed active learning methods on two simulated graphs, namely, the stochastic block model and the random sensor network. The results showcase the superiority of some approaches. For the said approaches, we conduct some more experiments to obtain insight into how they work. Next, we test these methods on two graphs related to real data, namely, a sub-network of a social network and a graph built from image data. Improvement in achieved over random labeling, which achieves the basic objective behind active learning.

7 Conclusion

In this thesis, we proposed an active learning methodology for an adaptive semi-supervised classifier on graphs which diffuses probabilities through random walks over it. Acquiring labels for all nodes is an expensive affair and semi-supervised learning is a natural solution. Diffusion-based semi-supervised classifiers spread labeled information over the graph, the source of the diffusion being the labeled nodes. Pool-based active learning with one batch relies on the learner to label all nodes together before training, instead of relying on multiple phases, which is faster. We made the following contributions:

We postulated the active learning problem as finding the sparse solution to a linear system which characterizes the diffusion, given the landing probabilities for each class across all nodes. These probabilities depend upon the starting distribution and hence are unavailable, which prompts us to use proxies based on clustering and smoothness. In our numerical experiments, we saw the performance of these methods degrades with an increase in the number of classes and they depend strongly on the clustering assumption.

We also presented the active learning problem as a model output selection, i.e., selecting nodes on which the landing probabilities are important for estimating the starting distribution. We developed two methods to accomplish this: the first method, Compressed Sensing-Active Learning, assumes that sensing the output at a small fraction of the nodes is enough to recover the starting sparse distribution; the second method, Sparse Sensing-Active Learning, senses those nodes which are important for estimating any starting distribution optimally. These methods, while not being free from their drawbacks, are more flexible, generalize better for multi-class problems and perform better than random selection, which is the basic goal behind active learning. A discussion on the possibilities of future work is now in order. The aim behind these suggestions is to make the approaches more realistic or cater more to some of the assumptions made in this thesis.

1. While obtaining proxies of landing probabilities, the dimensionality of the node embedding was taken to be $(C-1)$, with C being the number of classes. As this method suffered for multi-class problems, the dimensionality of the embedding should be increased to have more separation in the feature space, so that the membership values are more distinct.

2. The active learning carried out is not adaptive, unlike the semi-supervised classifier [cf. [24]] which relies on it. The classifier uses a fixed diffusion order K for all classes but K could be class-specific. For each class, the spread of the diffusion should be contained within it ideally. One way to do this would be to identify clusters or communities given the graph, treat each community as an isolated graph and solve the active learning problem for each sub-graph as we did in this

thesis. This could be faster than solving for the entire graph. The number of nodes from each sub-graph could also be a variable.

3. Almost all the graphs (expect the USPS graph) considered for evaluation are either defined naturally (i.e. through connections, citations or bounded by structure). If the graph is to be inferred from data, it should be well suited for adaptive diffusion and classification with multiple labels. One can also envisage the possibility of forming a consensus over multiple graphs (for example, different N neighbourhood graphs), each governed by its own diffusion.

4. The active sensing approaches, CS-AL and SS-AL are rooted in different theories but both arrive at a similar formulation, based on row selection. If the SS-AL(D) method can be solved with greedy selection, it would be interesting to see if the same applies to the cost of CS-AL. In this thesis, we tried greedy selection on CS-AL and saw it performs reasonably well, but we are not aware of any sub-modular property.

5. We solve for a sensing variable $\mathbf{w}$ for the cost $||\mathbf{H}^T \mathrm{diag}\big(\mathbf{w}\big)\mathbf{H} - \mathbf{G}_{M,N}||_F^2$ in (5.16). Throughout, we assumed that the diffusion parameters are fixed during active learning to simplify the problem. However, the relation $\mathbf{H} = \sum_{k=1}^{K} \mathbf{S}^k \theta_k$ can be utilized to minimize the cost for both $\mathbf{w}$ and $\boldsymbol{\theta}$ simultaneously. For CS-AL, it means there exists some underlying optimal diffusion which improves the sensing system. The problem could be formulated as

$$\begin{aligned} \underset{\mathbf{w}\in\{0,1\}^N, \boldsymbol{\theta}\in\mathbb{R}^K}{\text{minimize}} \quad & ||\big(\sum_{k=1}^{K} \mathbf{S}^k\theta_k\big)^T \mathrm{diag}\big(\mathbf{w}\big)\big(\sum_{k=1}^{K} \mathbf{S}^k\theta_k\big) - \mathbf{G}_{M,N}||_F^2 \\ \text{subject to} \quad & \mathbf{0}_K \leq \boldsymbol{\theta} \leq \mathbf{1}_K, \quad \sum_{k=1}^{K} \theta_k = 1. \end{aligned}$$

Bibliography

[1] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[2] Jüri Reimand, Laur Tooming, Hedi Peterson, Priit Adler, and Jaak Vilo. Graphweb: mining heterogeneous biological networks for gene modules with functional significance. *Nucleic acids research*, 36(suppl_2):W452–W459, 2008.

[3] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[4] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, pages 89–98, New York, NY, USA, 1998. ACM.

[5] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[6] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[7] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

[8] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[9] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[10] Ming Ji and Jiawei Han. A variance minimization criterion to active learning on graphs. In *Artificial Intelligence and Statistics*, pages 556–564, 2012.

[11] Xiaofei He, Ming Ji, and Hujun Bao. A unified active and semi-supervised learning framework for image compression. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72. IEEE, 2009.

[12] Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. In *Proceedings of the 23rd international conference on Machine learning*, pages 1081–1088. ACM, 2006.

[13] Quanquan Gu, Tong Zhang, and Jiawei Han. Batch-mode active learning via error bound minimization. In *UAI*, pages 300–309, 2014.

[14] Andrew Guillory and Jeff A Bilmes. Active semi-supervised learning using submodular functions. *arXiv preprint arXiv:1202.3726*, 2012.

[15] Yifei Ma, Roman Garnett, and Jeff Schneider. σ-optimality for active learning on gaussian random fields. In *Advances in Neural Information Processing Systems*, pages 2751–2759, 2013.

[16] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions.

[17] Kwang-Sung Jun and Robert Nowak. Graph-based active learning: A new look at expected error minimization. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1325–1329. IEEE, 2016.

[18] Quanquan Gu, Tong Zhang, Jiawei Han, and Chris H Ding. Selective labeling via error bound minimization. In *Advances in neural information processing systems*, pages 323–331, 2012.

[19] Lixin Shi, Yuhang Zhao, and Jie Tang. Batch mode active learning for networked data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(2):33, 2012.

[20] Jun Long, Jianping Yin, Wentao Zhao, and En Zhu. Graph-based active learning based on label propagation. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 179–190. Springer, 2008.

[21] Andrew Guillory and Jeff A Bilmes. Label selection on graphs. In *Advances in Neural Information Processing Systems*, pages 691–699, 2009.

[22] Akshay Gadde, Aamir Anis, and Antonio Ortega. Active semi-supervised learning using sampling theory for graph signals. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 492–501. ACM, 2014.

[23] Mingwei Leng, Yukai Yao, Jianjun Cheng, Weiming Lv, and Xiaoyun Chen. Active semi-supervised community detection algorithm with label propagation. In *DASFAA*, 2013.

[24] Dimitris Berberidis, Athanasios N Nikolakopoulos, and Georgios B Giannakis. Adaptive diffusions for scalable learning over graphs. *IEEE Transactions on Signal Processing*, 67(5):1307–1321, 2018.

[25] Peter G Casazza, Dan Redmond, and Janet C Tremain. Real equiangular frames. In *2008 42nd annual conference on information sciences and systems*, pages 715–720. IEEE, 2008.

[26] Richard G Baraniuk. Compressive sensing. *IEEE signal processing magazine*, 24(4), 2007.

[27] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2008.

[28] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296, 2008.

[29] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[30] K Lang and E Baum. Query learning can work poorly when a human oracle is used. ieee intl. In *Joint Conference on Neural Networks*, 1992.

[31] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[32] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR94*, pages 3–12. Springer, 1994.

[33] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[34] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[35] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[36] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[37] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[38] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[39] Steven M Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[40] Sundeep Prabhakar Chepuri and Geert Leus. Continuous sensor placement. *IEEE Signal Processing Letters*, 22(5):544–548, 2014.

[41] Sundeep Prabhakar Chepuri and Geert Leus. Sparsity-promoting sensor selection for non-linear measurement models. *IEEE Transactions on Signal Processing*, 63(3):684–698, 2014.

[42] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[43] Michael Grant and Stephen Boyd. Cvx: Matlab software for disciplined convex programming, version 2.1, 2014.

[44] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[45] Aliaksei Sandryhaila and Jose MF Moura. Big data analysis with signal processing on graphs. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[46] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[47] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[48] Mario Coutino, Elvin Isufi, and Geert Leus. Advances in distributed graph filtering. *IEEE Transactions on Signal Processing*, 67(9):2320–2333, 2019.

[49] Santiago Segarra, Antonio G Marques, and Alejandro Ribeiro. Optimal graph-filter design and applications to distributed linear network operators. *IEEE Transactions on Signal Processing*, 65(15):4117–4131, 2017.

[50] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation.

[51] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[52] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[53] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

[54] Emmanuel Candes, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *arXiv preprint math/0409186*, 2004.

[55] David L Donoho et al. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[56] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. The johnson-lindenstrauss lemma meets compressed sensing. *preprint*, 100(1):0, 2006.

[57] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

[58] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[59] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation.

[60] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[61] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[62] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 79–86, 2010.

[63] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[64] Theodore Wilbur Anderson. An introduction to multivariate statistical analysis. 1962.

[65] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[66] Antonio G Marques, Santiago Segarra, Geert Leus, and Alejandro Ribeiro. Sampling of graph signals with successive local aggregations. *IEEE Transactions on Signal Processing*, 64(7):1832–1843, 2015.

[67] Aamir Anis, Akshay Gadde, and Antonio Ortega. Towards a sampling theorem for signals on arbitrary graphs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3864–3868. IEEE, 2014.

[68] Sunil K Narang, Akshay Gadde, Eduard Sanou, and Antonio Ortega. Localized iterative methods for interpolation in graph structured data. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 491–494. IEEE, 2013.

[69] Isabel M Kloumann, Johan Ugander, and Jon Kleinberg. Block models and personalized pagerank. *Proceedings of the National Academy of Sciences*, 114(1):33–38, 2017.

[70] Konstantin Avrachenkov, Alexey Mishenin, Paulo Gonçalves, and Marina Sokol. Generalized optimization framework for graph-based semi-supervised learning. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 966–974. SIAM, 2012.

[71] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted 1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

[72] Michael Elad. Optimized projections for compressed sensing. *IEEE Transactions on Signal Processing*, 55(12):5695–5702, 2007.

[73] Joel A Tropp, Inderjit S Dhillon, Robert W Heath, and Thomas Strohmer. Designing structured tight frames via an alternating projection method. *IEEE Transactions on information theory*, 51(1):188–209, 2005.

[74] Julio Martin Duarte-Carvajalino and Guillermo Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Transactions on Image Processing*, 18(7):1395–1408, 2009.

[75] Andreas Krause and Daniel Golovin. Submodular function maximization., 2014.

[76] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K Hammond. Gspbox: A toolbox for signal processing on graphs. *arXiv preprint arXiv:1408.5781*, 2014.

[77] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.