Learning to Propagate Rare Labels

Rakesh Pimplikar
IBM Research
New Delhi, India
rakesh.pimplikar@gmail.com

Dinesh Garg
IBM Research
New Delhi, India
garg.dinesh@in.ibm.com

Deepesh Bharani
IIT Delhi
New Delhi, India
deepeshbharani.iitd@gmail.com

Gyana Parija
IBM Research
New Delhi, India
gyana.parija@in.ibm.com

ABSTRACT

Label propagation is a well-explored family of methods for training a semi-supervise classifier where input data points (both labeled and unlabeled) are connected in the form of a weighted graph. For binary classification, the performance of these methods starts degrading considerably whenever input dataset exhibits following characteristics - (i) *one of the class label is rare label or equivalently, class imbalance (CI) is very high*, and (ii) *degree of supervision (DoS) is very low – defined as fraction of labeled points.* These characteristics are common in many real-world datasets relating to network fraud detection. Moreover, in such applications, the amount of class imbalance is not known a priori. In this paper, we have proposed and justified the use of an alternative formulation for graph label propagation under such extreme behavior of the datasets. In our formulation, objective function is a *difference of convex quadratic functions* and the constraints are box constraints. We solve this program using *Concave-Convex Procedure (CCCP)*. Whenever the problem size becomes too large, we suggest to work with a k-NN subgraph of the given graph which can be sampled by using *Locality Sensitive Hashing (LSH)* technique. We have also discussed various issues that one typically faces while sampling such a k-NN subgraph in practice. Further, we have proposed a novel *label flipping* method on top of the CCCP solution, which improves the result of CCCP further whenever class imbalance information is made available a priori. Our method can be easily adopted for a MapReduce platform, such as Hadoop. We have conducted experiments on 11 datasets comprising a graph size of up to 20K nodes, CI as high as 99.6%, and DoS as low as 0.5%. Our method has resulted up to 19.5-times improvement in F-measure and up to 17.5-times improvement in AUC-PR measure against baseline methods.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661982.

General Terms

Algorithms, Design, Performance, Experimentation

Keywords

Label Propagation; Class Imbalance, Low Supervision

1. INTRODUCTION

The presence of fraudulent users is inevitable in any e-Commerce, m-Commerce, and online social networking applications. For example, the presence of fraudulent users in mobile payment networks [23], fake or malicious user accounts in an online social network [10, 26, 25], and fraudsters in online auction networks [22] are commonly occurring problems in real-life. Such users pose serious threats to these specific services and society at large. Detecting the fraudulent users in such networks is a daunting challenge from the perspective of both modeling as well as computational requirements. Almost any outlier detection technique, which is based on just user centric features, would cease to perform well here because it is quite easy for these users to tweak the values of the features in a way to circumvent the filter. On the other hand, the users in these applications inherently carry out transactions among themselves (e.g. money exchange, tweets, posts, etc.) resulting in a natural graph structure that is quite informative about users' behavior and contain much less noise. Thus, it would be apt to leverage these data in a critical manner for the purpose of identifying fraudulent users. In addition, for these applications, the number of fraudulent users (positive class) is much smaller than the normal users (negative class); and the available labeled dataset is much smaller in the size than the given unlabeled dataset. To summarize, the datasets in such applications comprise of following signature characteristics.

1. Sparse Graph Structure: For the given dataset, there is an underlying (typically explicit) sparse graph structure among the labeled and the unlabeled data points which is quite informative.

2. Rare Labels (aka Extremely High Class Imbalance): The class distribution is highly skewed. That is, positive label is a rare label, say of the order of 1% of total dataset or 99% negative label data points. For such a setting, we define N/n as the *class imbalance (CI)* where N and n denote the number of negative and total data points, respectively. The class imbalance is typically not known a priori.

3. Extremely Low Degree of Supervision: The size of the labeled dataset is quite small relative to the size of unlabeled dataset, say of the order of 0.5% of total dataset. We define the *degree of supervision* as l/n where l and n denote the number of labeled points and the total number of points, respectively.

Very high volume of the dataset is an additional challenge in these applications. Any classification problem comprising of all the above data characteristics naturally calls for *an efficient (both in terms of memory and time) semi-supervised graph label propagation method for rare labels*. Such a method, surprisingly, is not well addressed in the existing literature and is precisely the focus of our paper. In the following section, we begin with summarizing state-of-the-art on graph label propagation methods with a special attention on rare labels.

1.1 Graph Label Propagation - Prior Art

Graph label propagation (aka graph transduction) [4, 1] relates to the problem of learning a classifier where input data points (labeled and unlabeled) are connected in the form of a graph and the edge weights represent similarities between the points. The existing popular methods for label propagation include Gaussian fields and harmonic function (GFHF) based method [30], local and global consistency (LGC) method [28, 29], and graph transduction via alternating minimization (GTAM) method[1] [24]. The key idea behind these methods is to propagate the information about known labels across the whole graph in a smooth manner. This is achieved by striking a right balance between the accuracy on labeled nodes and a regularizer term that encourages smooth propagation. The underlying formulation for most of these methods is some form of a *quadratic convex program*.

An important point to note here is that all these methods exhibit a promising performance when input dataset behave normally, that is moderate values for both class imbalance (CI) and degree of supervision (DoS). However, the performance of these methods get compromised when input dataset behave abnormally – extremely high CI and extremely low DoS. We have not only experimentally validated but also argued analytically about the compromised performance of these methods in a subsequent section.

If the input data points were not connected in the form of a graph, the problem could have been formulated as either *semi-supervised outlier detection* [12] or *supervised classification from imbalanced data* [6, 16, 5, 19, 15, 11] depending on whether degree of supervision is low or high. The existing literature, however, appears to be quite slim when it comes to address the problem of *semi-supervised graph label propagation for rare labels*.

1.2 Contributions

The key contributions of this paper are summarized below.

1. We have experimentally validated and analytically argued that the performance of existing label propagation methods, such as GFHF, LGC, and GTAM, gets compromised when CI and DoS are pushed to the boundary.

2. We have proposed a novel label propagation method for the case when input dataset exhibit such extreme behavior. Our method extends the existing convex quadratic criterion formulation used by GFHF, LGC, and GTAM methods [30, 28, 29, 24] in a non-trivial way to account for limiting behavior of class imbalance and degree of supervision. This results in a *non-convex quadratic constrained program*.

3. In our non-convex quadratic formulation, the objective function turns out to be a *difference of convex quadratic functions* and the constraints are box constraints. We apply *Concave-Convex Procedure (CCCP)* to solve this problem in an approximate manner, where we use *gradient projection method* recursively in intermediate steps. This procedure does not require the knowledge of class imbalance a priori. We call this method as non-convex label propagation (NCLP) method.

4. The space and time complexity of each iteration in gradient projection method is $O(n^2)$, where n is the number of data points. This is one of the most expensive steps in any single iteration of the CCCP. To scale this method for real-world applications, we have proposed generating an approximate k-NN graph of the given dataset by making use of techniques such as *Locality Sensitive Hashing (LSH)*. This brings down the space and time complexity of each iteration in the gradient projection method to $O(kn)$.

5. We have also addressed different issues that one might face while working with kNN graph. In particular, we have discussed a method to fix the problem of label starvation and also choosing the parameters for LSH.

6. We have developed a novel and intuitive *label flipping* method which further improves the result of CCCP for the scenario when *class imbalance information is made available a priori*. If we work with the k-NN subgraph then the space and time complexity of this method turns out to be $O(kn)$.

7. GTAM [24] is one of the recent scheme to address the problem of class imbalance in graph label propagation. We have used GTAM and LGC methods as baseline to compare the performance of the proposed method. Note, Wang et al [24] have already shown that GTAM outperforms GFHF and hence we have not compared our method with GFHF method in our experiments. We have used two robust metrics to compare the performances – F*-measure* and *Area Under Precision-Recall Curve (AUC-PR)*. Our experiments suggest that the proposed method outperforms GTAM and LGC methods significantly resulting in up to 19.5-times improvement in F-measure and 17.5-times improvement in AUC-PR.

2. PROBLEM FORMULATION

Let $\mathscr{D} = \{(x_1, y_1), \ldots, (x_\ell, y_\ell), x_{\ell+1}, \ldots, x_{\ell+u}\}$ be the partially labeled input dataset for the binary classification problem, where ℓ points are labeled, u points are unlabeled, and $y_i \in \{+1, -1\}$ for $i = 1, 2, \ldots, \ell$. Let us assume that the total number of data points is n so that we have $n = \ell + u$. We also assume that labeled set follows the same class distribution as the set of unlabeled samples. Let $G = (V, E)$ be an undirected weighted graph representing the given dataset. This graph contains a vertex set $V = \{v_1, v_2, \ldots, v_n\}$, where each vertex v_i corresponds to a unique data point x_i. Further, each edge $(v_i, v_j) \in E$ of this graph is associated with a non-negative weight $w_{ij} \geq 0$ which measures the similarity between nodes v_i and v_j, and is computed using an appropriate Kernel function. Let $\boldsymbol{W} := (w_{ij})_{i,j=1,2,\ldots,n}$ denotes *weighted adjacency matrix* of graph G. The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. The *degree matrix* $\boldsymbol{D}$ is defined as a diagonal matrix with $d_1, d_2, \ldots, d_n$ as diagonal entries. The *normalized and unnormalized graph Laplacian matrices* [2] are defined as follows.

$$\boldsymbol{L} := \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2};\ \boldsymbol{L}_u := \boldsymbol{D} - \boldsymbol{W}$$

[1] There is another family of methods based on the theory of manifold learning [3]; but they are beyond the scope of this paper.

[2] $\boldsymbol{L} := \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}$ is also known as normalized graph Laplacian matrix [1, 18, 7] but we will avoid using this form here.

2.1 Convex Label Propagation

Given partially labeled dataset $\mathscr{D}$ and similarity matrix $\boldsymbol{W}$, the goal of a typical semi-supervised label propagation method is to learn a vector $\boldsymbol{f} \in \mathbb{R}^n$ and assign $\text{sign}(f_i)$ as the label for the vertex v_i. As shown in [1] and references therein, most of the existing methods for label propagation solve a variant of the following convex quadratic optimization problem:

$$\begin{aligned} \underset{\boldsymbol{f} \in \mathscr{F}}{\text{minimize}} \quad & \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} \\ \text{subject to} \quad & \sum\nolimits_{i=1}^{\ell} (f_i - y_i)^2 \leq \epsilon \end{aligned} \tag{1}$$

where, $\epsilon \geq 0$ is a constant, $\mathscr{F} \subseteq \mathbb{R}^n$, and $\boldsymbol{L}$ is a normalized graph Laplacian. In practice, these methods work with the Lagrangian of the above formulation, which is given by $L(\boldsymbol{f}, \mu) = \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} + \mu \sum_{i=1}^{\ell} (f_i - y_i)^2$ where, $\mu \geq 0$ is the Lagrange multiplier. In this Lagrangian form, the first term is a regularizer and corresponds to the smoothness of the label propagation (aka structural risk) and the second term corresponds to the empirical risk. These methods essentially try to minimize this Lagrangian subject to the constraint that $\boldsymbol{f} \in \mathscr{F}$. In what follows, we show how the existing methods, namely GFHF, LGC, and GTAM, become special cases of the convex quadratic program (1). For more details, one can refer to [1] and references therein.

1. GFHF based Method [30]: It solves a variant of (1) with quantities $(\boldsymbol{L}, \epsilon, \mathscr{F})$ assigned to $(\boldsymbol{L}_u, 0, \mathbb{R}^n)$.

2. LGC Method [28]: It solves yet another variant of (1) with quantities $(\epsilon, \mathscr{F})$ assigned to $(\epsilon/\mu, \mathbb{R}^n)$. The constant μ corresponds to the optimal Lagrange multiplier. Additionally, this method begins with an assumption of $y_i = 0$ for every unlabeled point and the constraint gets modified to $\sum_{i=1}^{n} (f_i - y_i)^2 \leq \epsilon$. Therefore, this method essentially tries to minimize the following Lagrangian.

$$\underset{\boldsymbol{f} \in \mathbb{R}^n}{\text{minimize}} \quad \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} + \mu \sum\nolimits_{i=1}^{n} (f_i - y_i)^2 \tag{2}$$

3. GTAM Method [24]: This method was designed for multi-class label propagation in the presence of class imbalance. The underlying formulation, ideally speaking, is not a variant of (1) but it is a mixed integer augmentation of (1). For the case of binary label propagation, this method essentially solves the following mixed integer program.

$$\begin{aligned} \underset{\substack{\boldsymbol{f}^+, \boldsymbol{f}^- \in \mathbb{R}^n \\ \boldsymbol{y} \in \{0,1\}^n}}{\text{minimize}} \Big[& \boldsymbol{f}^{+\top} \boldsymbol{L} \boldsymbol{f}^+ + \mu \sum\nolimits_{i=1}^{n} (f_i^+ - c_i^+ y_i)^2 + \\ & \boldsymbol{f}^{-\top} \boldsymbol{L} \boldsymbol{f}^- + \mu \sum\nolimits_{i=1}^{n} (f_i^- - c_i^- (1 - y_i))^2 \Big] \end{aligned} \tag{3}$$

where $f_i^+ = f_i y_i$, $f_i^- = f_i (1 - y_i)$, $c_i^+ = d_i / \sum_i d_i y_i$ and $c_i^- = d_i / \sum_i d_i (1 - y_i)$. For every point x_i, it is required that we have $y_i = 1$ if its label is $+1$ and $y_i = 0$, otherwise. The key innovation in this method lies in making $\boldsymbol{y}$ also as the binary decision variable and then solving the optimization problem by fixing the value of one variable (either $\boldsymbol{y}$ or $\boldsymbol{f}^+, \boldsymbol{f}^-$) and minimizing for the other one. Note, for fixed $\boldsymbol{y}$, this formulation becomes a convex quadratic program. At the optimal solution, the vector $\boldsymbol{y}$ itself is treated as the final labels of the vertices.

2.2 Limitations of Convex Methods

Consider the convex quadratic program (1) and its Lagrangian $L(\boldsymbol{f}, \mu) = \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} + \mu \sum_{i=1}^{\ell} (f_i - y_i)^2$. Observe, if the degree of supervision (DoS) is very low, the second term in the Lagrangian would become quite insignificant and the first term would be a dominating term. In such a scenario, the first term would drive the f_i values towards zero for majority of the unlabeled data points in an optimal solution because matrix L is always a positive semi-definite (PSD) matrix. Further, in practice, due to limited machine precision, these f_i values would indeed be equal to zero. Given that GFHF and LGC methods assign labels to the unlabeled points by means of the formula $y_i = \text{sign}(f_i)$, the labeling confusion at optimal solution would prevail at the same level as it was before solving the optimization problem. Further, if the class imbalance also happens to be excessively high then it would worsen the performance in following way – *if one chooses to assign $+1$ label for $f_i = 0$ then it would result in high recall but very bad precision. On the other hand, if -1 label is picked for $f_i = 0$ then it would result in very bad recall and moderate precision*. Note, the Lagrangian of LGC method comprises an additional third term of $\sum_{i=\ell+1}^{n} f_i^2$, but even this term would also support driving the f_i values towards zero for majority of the unlabeled data points in an optimal solution. In what follows, we unearth the ways by which one can fix such an undesirable behavior of convex methods under extreme values of CI and DoS. This discussion paves the way for our proposed non-convex label propagation (NCLP) method.

2.2.1 Integrality Gap

Ideally speaking, one should go about solving an integer version of the convex quadratic program (1) where we have $\boldsymbol{f} \in \{+1, -1\}^n$. This is because methods such as GFHF and LGC eventually use $\text{sign}(f_i)$ as the labels. However, due to the computational complexity, these methods implicitly seek to solve a relaxed version, where they relax the integer constraints through either $\boldsymbol{f} \in \mathbb{R}^n$ or $\boldsymbol{f} \in \mathscr{F}$. Under the extreme behavior of the dataset, an optimal value of such a relaxed integer program is quite far from the corresponding integer optimal value, that is $f_i = 0$ for pretty much all the points as discussed before. The presence of integrality constraints avoids such degenerate point as being the optimal solution. Thus, it is this large *integrality gap* that plays a lead role behind poor classification performance of the convex methods in scenarios where DoS is extremely low and CI is excessively high. For such scenarios, the performance of the convex methods degrades in the sense that either precision would take a hit or recall would take a hit and thereby resulting in low value of F-measure.

2.2.2 Choice of Loss Function

Any semi-supervised classification method, which uses a classification score f_i to classify unlabeled data point x_i, tends to strike a balance across three quantities by means of underlying optimization formulation – *empirical loss for labeled data, loss for unlabeled data*, and *generalization error*. For vast majority of graph label propagation methods, the generalization error remains the same and is given by the graph regularizer term $\boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f}$. It is the loss function for unlabeled data which plays a crucial role in determining the performance of the method under extreme behavior of the dataset. Figure 1 shows the unlabeled loss function used by different methods. Note, GFHF function uses no loss function for unlabeled data points, whereas LGC method uses convex quadratic loss function. The loss function used by LGC method encourages the classification scores f_i to be as close to 0 as possible, and thereby resulting in poor performance under extreme behavior of the data. This figure also shows the loss function used by a popular non-graphical semi-supervised classification method, namely *Transductive SVM (TSVM)* [13]. This loss function encourages classification scores f_i to be as close to ± 1 as possible. We are inspired by this loss function to modify the convex quadratic criterion and obtain a non-convex quadratic criterion (as described in the next section). Figure 1 also captures the unlabeled loss function used by our proposed *non-convex label propagation (NCLP) method* which we described in next section.

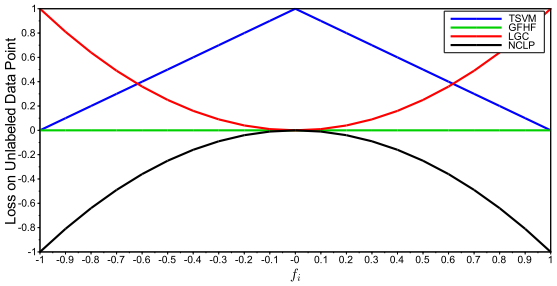


Figure 1: Different Loss Functions for Unlabeled Data Points

3. NON-CONVEX LABEL PROPAGATION

The idea behind the non-convex label propagation (NCLP) is to choose a middle ground between two extreme scenarios, namely $\boldsymbol{f} \in \{+1, -1\}^n$ and $\boldsymbol{f} \in \mathbb{R}^n$. Our non-convex criterion basically solves the following non-convex quadratic program with box constraints and then chooses sign(f_i) as the label of x_i.

$$\begin{aligned} \underset{\boldsymbol{f}}{\text{minimize}} \quad & \boldsymbol{f}^\top \boldsymbol{L}\boldsymbol{f} + \alpha \sum\nolimits_{i=1}^{l} (f_i - y_i)^2 - \beta \sum\nolimits_{i=(l+1)}^{n} f_i^2 \\ \text{subject to} \quad & -1 \leq f_i \leq +1 \quad \forall i = 1, \ldots, n \end{aligned} \tag{4}$$

where, α and β are hyper-parameters. Notice, there are two key differences between the formulation used by any of the convex quadratic criterion based methods and the proposed non-convex criterion based formulation: (i) we have restricted the values of f_i in the interval $[-1, +1]$ for non-convex criterion, and (ii) the last term captures the loss for unlabeled points which is a **concave** function. This makes the objective function as a difference of two convex functions. The above modifications yield better results for the extreme scenario of very low supervision and very high class imbalance. To gain an understanding and appreciation behind why above trick works, in what follows, we highlight some important properties of this non-convex program.

3.1 Properties of Non-Convex Criterion

3.1.1 *Reduced Integrality Gap*

Consider any integer solution point, say $\boldsymbol{f} \in \{+1, -1\}^n$. Observe, non-convex objective (4) value for this integer solution would be equal to $\boldsymbol{f}^\top \boldsymbol{L}\boldsymbol{f} + \alpha \sum_{i=1}^{l} (f_i - y_i)^2 - \beta(n - l)$. Similarly convex objective (2) value would be equal to $\boldsymbol{f}^\top \boldsymbol{L}\boldsymbol{f} + \alpha \sum_{i=1}^{l} (f_i - y_i)^2 + \alpha(n - l)$, if we choose α as the hyper-parameter instead of μ. For any such integer solution point, following relation holds true:

$$\begin{aligned} \text{convex objective value} = \text{non-convex objective value} \\ + (\alpha + \beta)(n - l) \end{aligned} \tag{5}$$

Given that α, β, l and n are constants, this relation implies that if we restrict our attention to only integer solutions, that is $\boldsymbol{f} \in \{+1, -1\}^n$, then the optimal solution would be the same for all the three methods – namely, GFHF, LGC, and NCLP. However, both convex and the non-convex criterion based methods solve relaxed versions of the problems, and their fractional optimal solutions are very different from each other. As discussed before, for the extreme scenario of low DoS and high CI, the fractional solution of the convex method has almost all its entries as zero. On the other hand, this is not the case for the non-convex method. The fractional solution of the non-convex method is closer to the integer optimal solution than the fractional solution of the convex method; primarily due the presence of the last term in its objective function. This last term, that is $\beta \sum_{i=(l+1)}^{n} f_i^2$, pushes more and more entries of the solution vector towards integer boundaries ($+1$ or -1) and thereby resulting in a better quality solution. A better quality fractional solution is precisely the reason behind non-convex criterion based method outperforming the convex criterion based method in our experiments (Section 5) under extreme conditions. In what follows, we justify why the presence of the last term in non-convex objective function pushes the f_i values towards integer boundaries.

3.1.2 *Pushing f_i Scores to the Integer Boundary*

The last term in the objective function of (4) forces more f_i values to be closer to the integer boundary, that is either -1 or $+1$. This means that despite the second term in (4) being insignificant under low DoS, the first term, unlike convex methods, can't drive all the f_i values to zero. This alleviates the limitations of the convex quadratic criterion based methods as described in section 2.2. In order to understand this effect more closely, let us consider a hypothetical graph as shown in Figure 2. The red and the green labeled nodes correspond to the $+1$ and -1 labeled data points, respectively. Labels of all the other points (nodes) need to be predicted. Remember, we always assume $+1$ as the rare class label. For illustration purpose, we have assumed a true class separation boundary as shown in this figure. The true label for each point on the left side of this boundary is $+1$ and that on the right side is -1. The aim of any label propagation method is to predict labels for all the unlabeled points.

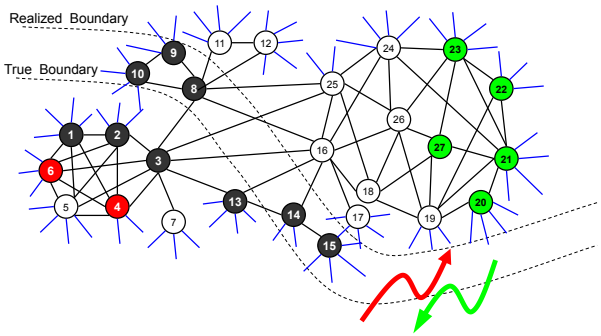


Figure 2: Illustration of High CI and Low DoS Effect

The general idea behind any label propagation can be understood by means of following influence propagation phenomenon on this graph – *the red nodes and the green nodes simultaneously start influencing their adjoining nodes in a manner similar to disease or influence spread. This is shown by red and green arrows in Figure 2. The quantum of influence depends on the weight of the edge joining two nodes. At the end, the class label/color of an unlabeled node gets determined depending on which color's overall influence is stronger. Further, by design, it is ensured that such an influence propagation is as smooth as possible.*

In view of this metaphor, it is not hard to believe that both convex and non-convex quadratic criterion based methods would do a good job of predicting labels for the points that lie far from the separation boundary. It is the points closer to the separation boundary which make all the difference. In Figure 2, points corresponding to the node ids $1, 2, 3, 8, 9, 10, 13, 14, 15$ are closer to the separation boundary and are denoted as black nodes. When a convex quadratic criterion based method, such as GFHF or LGC, is applied on this graph, it tries to regularize f_i scores and hence these scores gradually shift from positive values to negative values as we move from the left side of the boundary to the right side of the boundary. This results in $f_1, f_2, f_3, f_8, f_9, f_{10}, f_{13}, f_{14}$, and f_{15} taking tiny positive values whenever CI is very high and DoS is very low. The reason being the

following – *under very high CI and low DoS, influence of the red nodes dominate the green nodes at these boundary points*. Therefore, all these boundary points get mapped to $+1$ labels. The end result is separation boundary getting pushed further towards right as shown in Figure 2. On the contrary, if we let labels propagate under the non-convex quadratic criterion, there would be a tendency for any two points to acquire same signed f scores only when they are connected with very high weight edges for otherwise, it will push the objective function value high (due to the presence of last term). This results in f_8, f_9, f_{10}, f_{14} and f_{15} taking negative values as the weights of the edges connecting these nodes to their +Ve class side neighbors are outweigh by the weights of the edges connecting them to their -Ve class side neighbors.

3.1.3 Parallelization for Connected Components

A little thought would suggest that whenever given graph G has multiple connected components, the formulation (4) can be decomposed into multiple independent formulations – one for each connected component. This feature adds to the parallelization prospects of any solution methodology. However, one must keep in mind that each of these connected components must have labeled data points within itself for otherwise this formulation will recommend the same label (either +1 or -1) to all the data points.

3.2 Choosing Hyper-Parameters

The k-fold cross-validation coupled with grid search is a common trick to choose the values of hyper-parameters. In our context, however, such an approach would be infeasible due to its time complexity for large size problems. Therefore, in what follows, we suggest a rule of thumb to choose hyper-parameters. We have found these rules working extremely well in our experiments.

For choosing α, we advocate that the empirical loss arising out of labeled data should never be dominated by the structural risk. The reason being that empirical loss will anyways be quite small when degree of supervision is very low. For any graph, if we assume an integer solution $\boldsymbol{f} \in \{+1, -1\}^n$, the structural risk can be bounded as follows:

$$\sum_{(i,j)} w_{ij}/2 \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 \quad \leq \quad \sum_{i,j} (w_{ij}/2)\,(4/d_i) = 2n$$

The empirical loss for the labeled points can be at most $4\alpha l$, where l corresponds to the number of labeled points. Therefore, we suggest $4\alpha l \geq 2n$. In our experiments, we have chosen $\alpha = n/2l$.

For choosing β, we again advocate that the highest possible (absolute) value of the loss for unlabeled points should never dominate the structural risk. The reason being that this loss remains the same (equal to βu) across all possible integer solutions and hence we don't want this term to dictate our final solution. For this, we need to compute a lower bound on $\boldsymbol{f}^\top \boldsymbol{L}\boldsymbol{f}$ where $f_i \in \{+1, -1\}$. We get an approximate lower bound by solving the convex program corresponding to LGC method using an iterative scheme proposed by [28]. We use the α value as suggested before while applying this iterative method. Let $\boldsymbol{f} \in \mathbb{R}^n$ be the solution of this iterative method. We define $g_i = \text{sign}(f_i)$ and assign $\beta = [\boldsymbol{g}^\top \boldsymbol{L}\boldsymbol{g}]/u$.

4. ALGORITHMS

As discussed earlier, the formulation (4) is a difference of convex (DC) program and hence we use the well known *Concave-Convex Procedure (CCCP)* [27, 14] to solve this optimization problem in an approximate manner. CCCP is an iterative method which takes the linear approximation of the second function in each iteration and thereby reducing the overall problem into a convex quadratic approximation. Algorithm 1 depicts the pseudo code for the CCCP based label propagation method assuming **there is one single connected component (CC)**. If there are multiple CC then we need to run this method independently on each of the CC. The most critical step of this method involves solving a convex quadratic program using gradient projection method which is discussed in the next section.

Algorithm 1: Non-Convex Label Propagation (NCLP)

input : $\boldsymbol{L}$, $\{y_i\}_{i=1}^{l}$, initial guess $\boldsymbol{f}^0 \in [-1, 1]^n$
output : Final labels $\boldsymbol{y}^* \in \{-1, 1\}^n$

1 Choose an appropriate value for α and β;
2 **begin** Initialize
3 $\quad \boldsymbol{f}^{new} \leftarrow \boldsymbol{f}^0$;
4 **repeat**
5 $\quad \boldsymbol{f}^{old} \leftarrow \boldsymbol{f}^{new}$;
6 $\quad$ Solve following convex quadratic program using *gradient projection method* and call its solution $\boldsymbol{f}^{new}$.

$$\underset{\boldsymbol{f}}{\text{argmin}} \quad \boldsymbol{f}^\top L \boldsymbol{f} + \alpha \sum_{i=1}^{l} (f_i - y_i)^2 - 2\beta \sum_{i=(l+1)}^{n} f_i f_i^{old}$$
$$\text{s. t.} \quad -1 \leq f_i \leq +1 \;\; \forall i = 1, \ldots, n$$

7 **until** $\|\boldsymbol{f}^{new} - \boldsymbol{f}^{old}\| \leq$ *threshold*;
8 $y_i^* \leftarrow \text{sign}(f_i^{new})\ \forall i = 1, \ldots, n$

We set the initial guess $\boldsymbol{f}^0$ by setting f_i^0 as 0 for all the unlabeled points and given label y_i for all the labeled points.

4.1 Gradient Projection Method

The gradient projection method is one of the popular methods for solving a convex quadratic program having simple constraints such as box constraints [20, 17]. This method, in spirit, is very much similar to the method of steepest descent direction for unconstrained optimization except that the negative gradient is projected onto the surface of the box constraints in order to define the direction of movement. Exact details about this method can be found in [20, 17] but in what follows, we present a simplified form of this method when applied to our problem context.

The Algorithm 2 depicts the pseudo code for the gradient projection method applied to the convex quadratic program appearing in Algorithm 1. This method takes $\boldsymbol{f}^{old}$ as the input which is the current approximate solution of the overall problem and outputs an optimal solution $\boldsymbol{f}^{new}$ for the convex quadratic program appearing in the Algorithm 1. The matrix $\boldsymbol{S}$ in this algorithm is a $0/1$ diagonal matrix of size $n \times n$, having non-zero diagonal entry corresponding to each training data point. The gradient projection method is an iterative procedure where we maintain the set Q of active constraints' indices. In every iteration, we find a descent direction $\boldsymbol{d}$ in the space orthogonal to the space of active constraints. This ensures that the active constraints remain active when we move in this descent direction. If this direction is zero then we test whether the optimality has been achieved (or KKT conditions have been satisfied). If not then we drop the constraint from active set whose lagrange multiplier is the most negative and continue for the next iteration. If the descent direction $\boldsymbol{d}$ is nonzero then we first compute the range of the step size, denoted by $\overline{\eta}$, such that by taking this much step size in the direction $\boldsymbol{d}$, we still remain in the feasible region. Next, we find out the optimal step size η^* within this feasible range $\overline{\eta}$ which would minimize the objective function as a function of step size η^*.

Algorithm 2: Gradient Projection Method

input : $\boldsymbol{f}^{old}$, $\boldsymbol{L}$, α, β
output : $\boldsymbol{f}^{new} \in [-1, 1]^n$

1 $\boldsymbol{f} \leftarrow \boldsymbol{f}^{old}$;
2 $\bigtriangledown \boldsymbol{f} = 2\boldsymbol{L}\boldsymbol{f} + 2\alpha\boldsymbol{S}(\boldsymbol{f} - \boldsymbol{y}) - 2\beta(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{f}^{old}$;
3 $Q \leftarrow \{i \mid$ constraint f_i is tight$\}$;
4 Define an indicator vector $\boldsymbol{\theta} \in [-1, 1]^n$ as follows

$$\theta_i = \begin{cases} 0 & \text{if } i \notin Q \\ f_i & \text{o/w} \end{cases} ;$$

5 Define a decent direction $\boldsymbol{d} \in \mathbb{R}^n$ as follows

$$d_i = \begin{cases} 0 & \text{if } i \in Q \\ -\bigtriangledown f_i & \text{o/w} \end{cases} ;$$

6 **if** $\boldsymbol{d} = \boldsymbol{0}$ **then**
7 $\quad i^* \leftarrow \underset{i=1,\ldots,n}{\text{argmin}} \; [\lambda_i = -\theta_i \bigtriangledown f_i]$;
8 $\quad$ **if** $\lambda_{i^*} \geq 0$ **then**
9 $\quad\quad \boldsymbol{f}^{new} \leftarrow \boldsymbol{f}$ and Stop ; } KKT conditions satisfied
10 $\quad$ **else**
11 $\quad\quad Q \leftarrow Q \setminus \{i^*\}$ and go to Step 4
12 **else**
13 $\quad$ Compute feasible step size $\overline{\eta}$ along direction $\boldsymbol{d}$ as follows: $\overline{\eta} \leftarrow \min_i \eta_i$, where $\forall i$, we have

$$\eta_i = \begin{cases} \frac{1 - f_i}{d_i} & \text{if } d_i > 0 \\ \frac{-1 - f_i}{d_i} & \text{if } d_i < 0 \\ \infty & \text{o/w} \end{cases}$$

η_i is the maximum allowable step size along d_i to remain in feasible region

$\quad \tilde{\eta} \leftarrow -\boldsymbol{d}^\top \bigtriangledown \boldsymbol{f} / 2\boldsymbol{d}^\top(\boldsymbol{L} + \alpha\boldsymbol{S})\boldsymbol{d}$;
14 $\quad$ Compute the optimal step size η^* as follows
$\quad \eta^* \leftarrow \max\{0, \min\{\tilde{\eta}, \overline{\eta}\}\}$;
15 $\quad \boldsymbol{f} \leftarrow \boldsymbol{f} + \eta^* \boldsymbol{d}$ and go to Step 2

4.2 Scaling Issues and k-NN Graph Sampling

The Steps 2 and 13 of Algorithm 2 are the most expensive, $O(n^2)$, steps in terms of both memory and time requirements. All other operations are of the order of $O(n)$. In what follows, we discuss ways to deal with this problem.

Recall, if the application has an explicit graph, e.g. social network or mobile payment network, then typically such graphs are quite sparse and one can cope with the scale by employing the sparse representations of the matrices $\boldsymbol{L}$ and $\boldsymbol{S}$. One can also consider using MapReduce platform where we already have efficient algorithms available for large matrix multiplication [21]. However, the challenge arises when there is no explicit graph given among the data points. In such a scenario, one can always construct a complete graph among the data points by means of defining an appropriate similarity (or distance) function between a pair of data points. As dealing with such a huge graph is quite expensive, we advocate to work with a k-NN sub-graph instead. That is, we suggest to generate a k-NN graph of the data points. The use of k-NN graph brings down the space and time complexities of the steps 2 and 13 to $O(kn)$. The generation of a k-NN graph, however, takes $O(n^2)$ in naive manner. To deal with this problem, we further suggest to work with an approximate k-NN graph by employing a hashing technique such as Locality Sensitive Hashing (LSH) [2] or more recent technique proposed by [9]. In LSH based approach, it is required to first hash (aka project) each of the given n data points – having d-dimension – into a p-dimensional space, where $p << d$. To boost the approximation quality, LSH performs T different projections in a parallel and independent manner. Once this data structure is ready, we can generate an approximate k-NN efficiently. The space and time required to hash n data points is of the order of $O(pTn)$ where, p and T are known as LSH parameters. Once this hashing is completed, the time required to generate approximate k-NN graph reduces to $O(pTn^{1+\epsilon})$ where $0 < \epsilon < 1$. LSH technique has an advantage that one can control the tradeoff between the quality of a k-NN graph and the resource requirement (memory and time) by means of adjusting the tunable parameters.

4.3 Tuning LSH Parameters

As discussed above, in order to cope with the size of a problem in real world settings, it would be necessary to generate an approximate k-NN graph of the given dataset before even applying NCLP method. LSH is an efficient data structure for such purpose. Typically, the implementation of LSH involves setting up three tunable parameters – (i) the bucket width w for a hash function, (ii) the lower dimension p into which the data are projected under each hash table, and (iii) number of hash tables T. While, the choice of parameters p and T directly impact the tradeoff between resource requirements and the quality of k-NN, there is no rule of thumb or guidelines when it comes to choosing them. Depending on one's appetite for resources (memory and time), the values of p and T need to be chosen in pretty much hit and trail manner. On the other hand, for choosing the bucket width w, one can be little more systematic as discussed next.

Suppose, the original data points are having d-dimensional features. Then, by normalizing them in a range of $[0, 1]$, the distance between any two points in the d-dimensional space can be made at max $\sqrt{d}$. This means when we project such data points into a p-dimensional space under LSH, the maximum value for each of these p-coordinates would be $\sqrt{d}$. This further means that there would be $\sqrt{d}/w$ number of hash bucket along each projection dimension if we decide to choose w as the bucket width for each hash function. Assuming, given n data points are uniformly distributed, the projected coordinates of these points are also likely to be nearly uniformly distributed along any dimension. Thus, along each projection dimension, each of these hash bucket should comprise $nw/\sqrt{d}$ number of points on an average. Given that we want to generate a k-NN graph, we should make sure that for any point, there are at least k-NN points in the same bucket along each projection dimension. This would mean that we should set $nw/\sqrt{d} \geq k$. In our experiments, we opt to set $nw/\sqrt{d} = 10k$ to be on safe side. This also gives us decent performance in terms of scaling.

4.4 Label Starvation Problem in k-NN Graph

We would also like to highlight a difficulty that one typically runs into while using k-NN graph to cope with the scale of the problem. Note, as we decrease k value, we are likely to get more number of connected components in the graph. In such a case, as described earlier, we need to run NCLP method independently on each of these connected components. For this to be feasible, however, each of these connected components should contain both +Ve and -Ve labeled data points for otherwise NCLP method will output the same label across all data points in that component. We call this problem as *label starvation problem in k-NN graph*. If all the connected components in the generated k-NN graph satisfy this feasibility

requirement then we can simply proceed by applying NCLP method independently on each of these components. On the contrary, if this is not the case then we would require to add some extra edges in the k-NN graph so that we reconnect all these components and get one single connected component having both types of labeled points within it. While there could be many ways to accomplish it, we have adopted a *minimum spanning tree (MST)* based strategy for this problem while conducting experiments. This strategy is summarized in the form of Algorithm 3. At high level, the idea behind this strategy is to add minimum possible edges that are most effective in terms of connecting label starved components with their nearby components who have labeled nodes within them.

Algorithm 3: Fixing Label Starvation Problem

input : Connected components $C_1, C_2, \ldots, C_z$
output : Connected graph among data points

1 **for** *every pair of* (C_i, C_j) *where* C_i *has at least one labeled point* **do**
2 **for** *every labeled point* $x^i \in C_i$ **do**
3 $\{x_1^j, x_2^j, \ldots, x_k^j\} \leftarrow k$-NN points of x^i in C_j;
4 $z_{x^i} \leftarrow \min_{t=1,\ldots,k} dist(x^i, x_t^j)$;
5 $dist(C_i, C_j) \leftarrow \min_t z_t$;
6 **for** *every pair of* (C_i, C_j) *where* C_i *or* C_j *has at least one labeled point* **do**
7 $d_{ij} \leftarrow \min\{dist(C_i, C_j), dist(C_j, C_i)\}$
8 Build an MST by treating these connected components as nodes of a graph and d_{ij} as the edge weight between nodes C_i and C_j;
9 **for** *each edge* (i, j) *of this MST* **do**
10 Include all the edges between each labeled point of C_i (and C_j) and its k-NN points in C_j (and C_i)

4.5 A Fast and Greedy Label Flipping Method

In this section, we describe a label flipping method which can improve the quality of any given labeling, including output of NCLP method, whenever class imbalance information is also known a priori. This method is simple, fast, and greedy in nature. It starts with a given labeling of the data points, say $\boldsymbol{y}^{old}$, and selectively flips the labels of some points until a desired number of +Ve labels, say P^*, is achieved. At any given point in time, this method compares the class imbalance of the current solution in hand with the target imbalance. If the current class imbalance is higher than the target one then it picks one -Ve labeled point and flips its label. The other case is symmetric. While picking a -Ve labeled point, it picks the point whose label flipping would result in the least increase in value of objective function (4). For this, we define a notion called *potential* of a data point. The potential of a data point measures the increase in the value of the objective function (4) in case its label is flipped. Therefore, this method chooses a -Ve labeled point having minimum potential and flips its label. This iterative process continues until desired level of class imbalance is achieved.

The pseudo code for this method is given in Algorithm 4. Observe, the label flipping method can start with any initial labeling and it outputs an improved labeling with a desired level of class imbalance. The quality of the output, however, may vary depending on the starting point as this approach is greedy in nature. One would see a huge difference in quality of the output if the label flipping is triggered with the output of NCLP method versus any other random label assignment. Finally, we would like to highlight the use of two heaps H_+ and H_- as the data structure in our implementation for this method. This data structure in conjunction with the idea of node potential adds in lots of computational benefits and makes this method really fast in practice. For a k-NN graph, the space and time complexity of this method turns out to be $O(kn)$ assuming the class imbalance for starting solution is not too far from the target class imbalance.

Algorithm 4: Greedy Label Flipping

input : Initial labels $\boldsymbol{y}^{old} \in \{-1, 1\}^n$, $\boldsymbol{W}$, target class imbalance r^*
output : Revised labels $\boldsymbol{y}^{new} \in \{-1, 1\}^n$

1 $\boldsymbol{y} \leftarrow \boldsymbol{y}^{old}$;
2 $P \leftarrow \{i \mid y_i = +1\}$;
3 $N \leftarrow \{i \mid y_i = -1\}$;
4 For every point x_i compute its potential as $\phi_i \leftarrow \sum_{j \in N(i)} 2w_{ij}y_iy_j/\sqrt{d_id_j}$;
5 Using potentials ϕ_i, construct a heap H_+ for the set P and H_- for the set N;
6 Compute the target number of positive points as $P^* = \lceil n(1 - r^*) \rceil$;
7 **if** $|P| < P^*$ **then**
8 $i_- \leftarrow$ Root node of heap H_- ;
9 Flip the label y_{i_-} and update set P accordingly;
10 Flip the sign of ϕ_{i_-} ;
11 Move i_- from H_- into H_+;
12 **for** *each* $j \in$ *Nbr*(i_-) **do**
13 $\phi_j \leftarrow \phi_j + 4w_{i_-j}y_j/\sqrt{d_{i_-}d_j}$;
14 Update node j in its heap;
15 Go to Step 7
16 **else**
17 **if** $|P| > P^*$ **then**
18 Perform similar operations as in Steps 8 through 15 except that swap the +ve and -ve signs everywhere.
19 **else**
20 $\boldsymbol{y}^{new} \leftarrow \boldsymbol{y}$;

5. EXPERIMENTS

We evaluated the performance of our approach over existing methods by conducting a set of experiments on eleven datasets, including KDD Cup'99 dataset[3] related to intrusion detection problem and ten other datasets from the UCI repository[4]. KDD Cup'99 dataset has 494K data points with 38 features each. We randomly sampled 20K points out of these 494K points, including 1K points from positive class. Here positive class represents intrusion. This way, we could get 95% class imbalance. For other datasets, we marked the smallest size class as +Ve class, and all other classes as -Ve class. The summary statistics – *number of instances, number of attributes*, and *percentage of negative instances (class imbalance)* – of these datasets are captured in the first four columns of Table 1.

As described in sections 4.2, we generated an approximate k-NN graph for every dataset using LSH technique. In our experiments, we used $k = 5, p = 7$, and $T = 20$ for all the datasets. The bucket width w for every dataset was computed using formula mentioned in section 4.3. We mainly conducted two types of experiments. The

[3]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[4]http://archive.ics.uci.edu/ml

first set of experiments was conducted to observe the effect of degree of supervision, and the second set of experiments was performed to observe the effect of class imbalance. In what follows, we describe each of these two sets of experiments in detail.

5.1 Effect of Degree of Supervision

In this set of experiments, for every dataset, we kept the class imbalance fixed and varied degree of supervision across a range. For each value within this range, we randomly sampled an appropriate number of points whose labeled were retained and all other labels were masked. While sampling, we ensured the class imbalance across the labeled data is roughly the same as the class imbalance across the whole dataset, so as to mimic the realistic scenario. For every dataset and each value of degree of supervision, we repeated the experiment 50 times – each time with a different sample of labeled points – so as to get an idea behind average behavior of different methods. For each such experiment, we compared the performance of following four methods – out of which the first two serve as the baseline and the last two are the proposed ones – (i) LGC [28, 29], (ii) GTAM [24], (iii) NCLP, and (iv) NCLP followed by label flipping (call it as NCLF).

Tables 1 and 2 below show the average value of *F-measure* and *Area Under Precision-Recall Curve (AUC-PR)*, respectively, for each experiment in this setting. Note, AUC-PR values can't be computed for GTAM and NCLF methods as these methods directly assign labels to the data points without making use of classification score f_i. Further, we prefer to use AUC-PR instead of *Area Under ROC curve (AUC-ROC)* because AUC-PR is known to be a better metric than AUC-ROC while comparing different classifiers – especially under high class imbalance [8]. It has been shown that a classifier which performs good in terms of AUC-PR would definitely perform good in terms of AUC-ROC, but not vice-a-versa.

While actual numbers are listed in Tables 1 and 2, we have sketched heatmap plots for these tables (Figures 3 and 4, respectively) to serve as visual aid while interpreting the results. Figures 3 shows the heatmap for F-measure based performance gain in NCLP method over GTAM method. Similarly, Figure 4 shows the heatmap for AUC-PR based performance gain in NCLP method over LGC method. In each of these heatmaps, the numeric score inside a cell indicates the superiority (or inferiority) level of the proposed method as compared to the baseline method. The score in each cell is equal to the *ratio of superior method's performance to the inferior method's performance*. We assign a green label to the cell and a positive sign to its score if the proposed method performs superior in that cell. Otherwise, we assign a red label to the cell and a negative sign to its score. A cell's color sweeps from red through green as its score swings from negative value to positive value. For our purposes, greener a cell means better the performance of the proposed method as compared to the baseline method. In what follows, we have summarized some interesting insights from these results.

- It is apparent from Tables 1 that GTAM method outperforms LGC method for almost all the cases. Further, NCLF method outperforms NCLP method for a large number of cases. This observation, however, is not too surprising as NCLF method demands for an additional information, namely class imbalance. Therefore, we have explicitly compared F-measure performance of just NCLP and GTAM methods in Figure 3.

- As per Figure 3, NCLP method beats GTAM method (in terms of F-measure) for all datasets except for MagicGamma, ILPD, and some range of Inosphere dataset. However, for all these three datasets class imbalance is not so high and, hence, this hardly affects the merit of the proposed method.

- For Optdigits dataset, NCLP method performs 5.23-times superior to GTAM method (in terms of F-measure) for 0.5% supervision. This number blows up to 5.32-times if class imbalance information is known and label flipping is applied. On an absolute scale, NCLP method achieves 94% F-measure with just 0.5% supervision. This number blows up to 99.5% when supervision is increased to 5%.

- From Figure 4, we can say that NCLP method beats LGC method in terms of AUC-PR for all datasets except for ILPD, WineQual, Amazon, and a small range of Optdigits, Synthetic, Waveform, and MagicGamma datasets. However, in all such cases, NCLP method is never more than 3.9-times inferior to LGC method. On the flip side, for Optdigits and Synthetic datasets, NCLP is 10 to 17-times superior to LGC method under 0.5% supervision.

- The best (green) and the worst (red) performances of the proposed methods are highlighted in Tables 1 and 2.

5.2 Effect of Class Imbalance

In the second set of experiments, we evaluated the performance of NCLP and NCLF methods over GTAM and LGC methods by varying class imbalance in datasets. For this, we worked with KDD Cup'99 dataset. We sampled 20K points from this dataset in a way to ensure 95% class imbalance. It formed our first dataset for this set of experiments. Similarly, we created nine more datasets by sampling 20K points with $90\%, 85\%, \ldots, 50\%$ class imbalance. For each of these ten different class imbalance valued datasets, we further varied DoS in the range of 0.5% to 10% and computed F-measures using NCLP, NCLF, GTAM, and LGC methods.

We have plotted 3D graphs to capture the variation of F-measures as a function of class imbalance and DoS. Figure 5 shows such 3D graphs for LGC, GTAM, NCLP, and NCLF methods.

It follows from Figure 5 that LGC and GTAM methods are very sensitive to class imbalance as these methods see a steep decline in their performance with an increase in class imbalance. Contrary to this, NCLP method remains almost unaffected by the class imbalance. More precisely, for any given DoS, the F-measure of NCLP method remains pretty much the constant over a range of class imbalance. The numbers show that NCLP method outperforms LGC method, especially under the scenario when class imbalance is extremely high. Finally, NCLF method's performance remains flat for a substantial range of high class imbalance (75%–95%).

6. CONCLUSIONS AND FUTURE WORK

We have highlighted the limitations of the existing convex criterion based graph label propagation methods such as GFHF, LGC, and GTAM under the scenario when class imbalance is quite high and degree of supervision is quite low. We have suggested a non-convex extension to this criterion and proposed an efficient scheme to propagate labels under this new criterion. The questions that are still unanswered include: (i) how to generalize NCLP and NCLF methods to multi-class classification, (ii) can we develop a label flipping type fully combinatorial algorithm to solve the non-convex program approximately, (iii) a sound theoretical understanding behind this shift in the behavior of label propagation method especially when class imbalance is high and degree of supervision is low, and (iv) a rigorous analysis behind what values of hyper-parameters α, β would make the objective function as convex and whether those values are good choices.

Dataset	# Inst	# Attr	CI (%)	Degree of Supervision (%)											
				0.5				1				2			
				LGC	GTAM	NCLP	NCLF	LGC	GTAM	NCLP	NCLF	LGC	GTAM	NCLP	NCLF
Optdigits	5620	64	90.2	.0032	.1802	.9426	.9596	.0036	.1810	.9609	.9605	.0056	.1824	.9888	.9872
Synthetic	600	60	83.3	.0064	.2869	.5275	.8050	.0070	.2878	.5807	.8840	.0098	.2899	.8250	.9320
ImageSeg	2310	19	85.7	.1134	.2502	.3779	.3657	.0609	.2512	.4556	.4673	.0382	.2531	.6209	.5697
Isolet	7797	617	96.2	.0461	.0744	.1041	.1967	.0341	.0748	.1899	.2320	.0364	.0755	.2744	.2900
KDDCup'99	20000	38	95	.0296	.0957	.2119	.1614	.0275	.0961	.3013	.2082	.0281	.0970	.3461	.2520
Waveform	5000	40	66.9	.2172	.4986	.6541	.6164	.1731	.4998	.7224	.6872	.2143	.5024	.7004	.6815
WineQual	4898	11	99.6	.1450	.0101	.0747	.0500	.0692	.0101	.1100	.0700	.0632	.0103	.0809	.0700
Amazon	1500	10000	98	.0321	.0399	.0870	.0667	.0324	.0401	.1163	.0444	.0400	.0405	.0963	.0533
MagicGamma	19020	10	64.8	.0236	.5172	.0295	.5343	.0425	.5185	.0518	.5965	.0777	.5210	.0932	.6333
Ionosphere	350	34	64.3	.4475	.5274	.3209	.4112	.3910	.5285	.3391	.4888	.4918	.5319	.2391	.5984
ILPD	579	10	71.5	.0325	.4463	.0180	.3012	.0569	.4475	.0414	.2407	.1082	.4500	.1303	.2802

Dataset	# Inst	# Attr	CI (%)	Degree of Supervision (%)											
				5				10				20			
				LGC	GTAM	NCLP	NCLF	LGC	GTAM	NCLP	NCLF	LGC	GTAM	NCLP	NCLF
Optdigits	5620	64	90.2	.0111	.1871	.9956	.9948	.0221	.1955	.9954	.9942	.0477	.2147	.3338	.9968
Synthetic	600	60	83.3	.0216	.2963	.8891	.9070	.0387	.3077	.9598	.9720	.0779	.3333	.3333	.9920
ImageSeg	2310	19	85.7	.0498	.2589	.7037	.6599	.0733	.2695	.8348	.8539	.1068	.2933	.8778	.9219
Isolet	7797	617	96.2	.0451	.0777	.3566	.4147	.0546	.0817	.4914	.5920	.0628	.0909	.5238	.7193
KDDCup'99	20000	38	95	.0320	.0998	.4281	.3386	.0499	.1047	.5968	.6046	.0669	.1163	.7298	.8444
Waveform	5000	40	66.9	.2155	.5102	.7305	.7205	.2519	.5237	.7403	.7329	.3008	.5530	.3333	.7794
WineQual	4898	11	99.6	.0307	.0106	.1037	.0700	.0191	.0112	.1802	.1300	.0228	.0125	.2452	.2300
Amazon	1500	10000	98	.0419	.0418	.1318	.0733	.0443	.0440	.1777	.1133	.0487	.0492	.3358	.2267
MagicGamma	19020	10	64.8	.1714	.5288	.2229	.6895	.2784	.5422	.3667	.7111	.4106	.5713	.5552	.7352
Ionosphere	350	34	64.3	.4002	.5388	.4891	.7408	.3413	.5531	.7041	.7912	.3577	.5814	.7825	.8440
ILPD	579	10	71.5	.2276	.4576	.2229	.3358	.3057	.4709	.3064	.3519	.4289	.5008	.4649	.4815

Table 1: An F-measure Based Comparison of the Proposed Methods with Baseline Methods for Varying Degree of Supervision

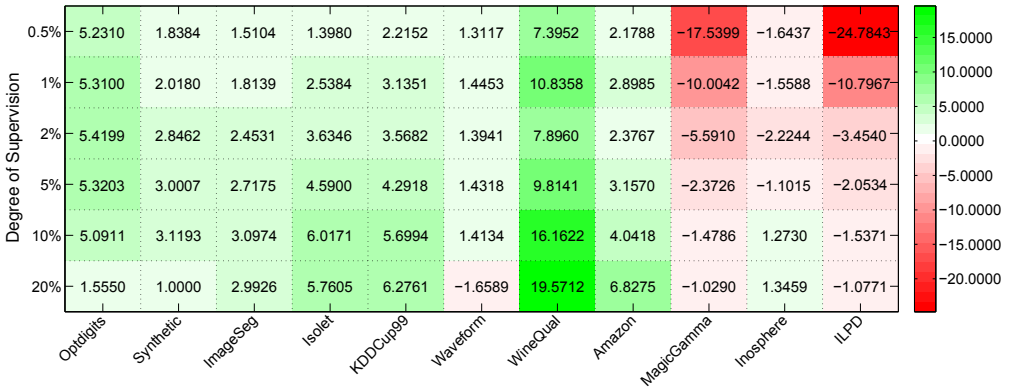


Figure 3: A Heatmap for F-Measure Based Comparison between NCLP and GTAM

7. REFERENCES

[1] A. Agovic and A. Banerjee. A unified view of graph-based semi-supervised learning: Label propagation, graph-cuts, and embeddings. Technical Report TR 09-012, University of Minnesota, 2009.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

[4] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[6] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

[7] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[8] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *ICML*, pages 233–240, 2006.

[9] W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, pages 577–586, 2011.

Dataset	# Inst	# Attr	CI (%)	Degree of Supervision (%)											
				0.5		1		2		5		10		20	
				LGC	NCLP	LGC	NCLP	LGC	NCLP	LGC	NCLP	LGC	NCLP	LGC	NCLP
Optdigits	5620	64	90.2	.0549	.9652	.0606	.9502	.0701	.9860	.1022	.9892	.1532	.9913	.2588	.2474
Synthetic	600	60	83.3	.0940	.9741	.0943	.9707	.1007	.9217	.1330	.9103	.1860	.9674	.2926	.2745
ImageSeg	2310	19	85.7	.0972	.3388	.0926	.4180	.0970	.5805	.1284	.6805	.1812	.8272	.2862	.8129
Isolet	7797	617	96.2	.0282	.0841	.0298	.1811	.0398	.2523	.0712	.2606	.1235	.3814	.2258	.3175
KDDCup'99	20000	38	95	.0321	.1488	.0369	.1978	.0477	.2208	.0793	.2967	.1332	.4475	.2368	.5775
Waveform	5000	40	66.9	.2347	.5823	.2341	.6403	.2521	.6670	.2869	.7014	.3499	.7192	.4613	.3777
WineQual	4898	11	99.6	.0484	.0276	.0358	.0383	.0503	.0277	.0310	.0359	.0557	.0718	.1751	.0452
Amazon	1500	10000	98	.0464	.0563	.0434	.1542	.0525	.0416	.0710	.0544	.1085	.0821	.2119	.1231
MagicGamma	19020	10	64.8	.2086	.2142	.2181	.2266	.2362	.2527	.2915	.3354	.3724	.4488	.5076	.4719
Ionosphere	350	34	64.3	.3130	.4855	.3071	.5509	.2803	.6434	.2929	.7649	.3630	.8226	.4698	.6464
ILPD	579	10	71.5	.1688	.1685	.1785	.1764	.1949	.2160	.2560	.2878	.3476	.3765	.4857	.3056

Table 2: An AUC-PR Based Comparison of the Proposed Methods with Baseline Methods for Varying Degree of Supervision

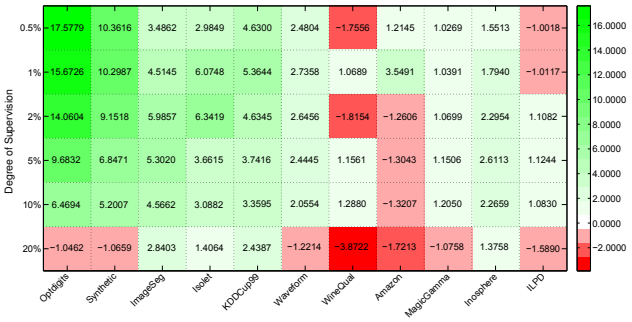
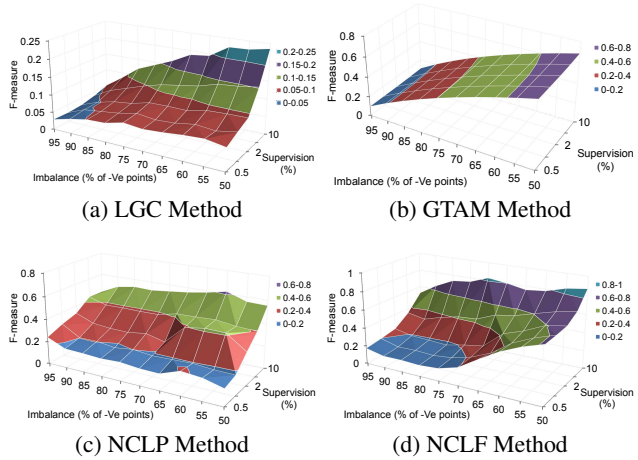


Figure 4: A Heatmap for AUC-PR Comparison of NCLP with LGC



(a) LGC Method (b) GTAM Method

(c) NCLP Method (d) NCLF Method

Figure 5: 3D Plots for F-Measure Based Performance Comparison of Baseline Methods and Proposed Method

[10] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. COMPA: Detecting compromised accounts on social networks. In *NDSS*, 2013.

[11] W. Fithian and T. Hastie. Local case-control sampling: Efficient subsampling in imbalanced data sets. *arXiv:1306.3706*, 2013.

[12] J. Gao, H. Cheng, and P.-N. Tan. Semi-supervised outlier detection. In *Symposium on Applied Computing*, pages 635–636, 2006.

[13] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[14] B. K. Sriperumbudur and G. R. G. Lanckriet. On the convergence of the concave-convex procedure. In *NIPS*, 2009.

[15] S. Li and I. W. Tsang. Maximum margin/volume outlier detection. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 385–392, 2011.

[16] W. Liu and S. Chawla. A quadratic mean based supervised learning model for managing data skewness. In *SDM*, pages 188–198, 2011.

[17] D. G. Luenberger. *Linear and Nonlinear Programming*. Springer, 2003.

[18] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[19] A. K. Menon, H. Narasimhan, S. Agarwal, and S. Chawla. On the statistical consistency of algorithms for binary classification under class imbalance. In *ICML*, 2013.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1997.

[21] J. Norstad. A MapReduce algorithm for matrix multiplication, 2009. http://www.norstad.org/matrix-multiply/index.html.

[22] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: A fast and scalable system for fraud detection in online auction networks. In *WWW*, 2007.

[23] I. N. C. S. Report. Mobile payments–a growing threat. Technical report, Bureau of International Narcotics and Law Enforcement Affairs, U.S. Department of State, 2008, URL: http://www.test.org/doe/.

[24] J. Wang, T. Jebara, and S.-F. Chang. Graph transduction via alternating minimization. In *ICML*, pages 1144–1151, 2008.

[25] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17, May 2008.

[26] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. Sybilguard: Defending against sybil attacks via social networks. *Networking, IEEE/ACM Transactions on*, 16(3):576–589, June 2008.

[27] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 12:915–936, 2003.

[28] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2004.

[29] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. In *ICML Workshop on Statistical Relational Learning*, pages 132–137, 2004.

[30] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.