GRAPHICAL SEMI-SUPERVISED LEARNING

ROHIN GILMAN, ALEX JOHNSON, AND KAITLYNN LILLY

Abstract. We investigate graphical semi-supervised learning approaches and benchmark both Probit and regression models on various manifolds. We construct data on well-separated clusters in the plane, weakly connected clusters on a spiral manifold, and examine data from the MNIST data set. We find that these approaches are generally effective, but can be limited by the underlying data and choices made in model construction.

1. Introduction

Graphical semi-supervised learning (GSSL) is a prominent field of research in machine learning as it serves as a powerful technique for dealing with classification problems in which only a small subset of the data is labeled. In kernel methods for GSSL, the unlabeled data is used to construct a similarity graph, where each node represents an instance and the edges indicate the similarity between them. Clustering this graph results in a complete labeling of the data set. As a result, GSSL using kernel methods is a promising approach for tackling problems in areas such as image recognition, natural language processing, and bioinformatics.

2. Theoretical Background

2.1. Proximity Graphs.

Given a set of points $\{x_1, x_2, \ldots, x_n\}$ in a metric space, we can construct a proximity graph $G = (V, W)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices representing the points, and W is an adjacency matrix whose entries

$$W_{ij} = \begin{cases} \eta(x_i, x_j) & \text{if } d(x_i, x_j) < r, \\ 0 & \text{otherwise,} \end{cases}$$

where $\eta \colon X \times X \to \mathbb{R}$ is a non-increasing function and $r > 0$ is the maximum radius within which vertices can be adjacent, represent the weights of the edges between pairs of vertices.

2.2. **K-Nearest Neighbor Graphs.** K-Nearest Neighbor (K-NN) graphs are a class of graphs within which vertices are adjacent to a fixed number of other vertices. The K-NN graph $G_k = (V, W)$ is constructed as follows: for each vertex v_i, its k-nearest neighbors are found according to some distance metric, and edges of weight $\eta(x_1, x_2)$ are added between v_i and each of its k-nearest neighbors. In other words, v_i is adjacent to the k vertices that are closest to it in the data set. The primary advantage of using K-NN graphs is that on average, vertices have relatively low degree, so the weight matrix W is often sparse, increasing the performance of graphical methods.

2.3. **Graph Laplacian.** Given a graph $G = (V, W)$, we can define a graph Laplacian L as a matrix that captures the pairwise relationships between data points on the graph. Specifically, we can construct the unnormalized graph Laplacian L as follows:

$$L = D - W,$$

where D is the diagonal degree matrix defined by

$$D_{ii} = \sum_{j=1}^{n} W_{ij}.$$

If G has M connected components, then $\dim(\mathrm{Null}(L)) = M$. Thus, we only need to find the multiplicity of the zero-eigenvalue of L to determine the number of clusters in the data.

2.4. **Matérn Family of Kernels.** We define the Matérn kernel as

$$K(x, y) = \kappa(\|x - y\|), \kappa(t) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{t}{\gamma}\right) K_\nu \left(\sqrt{2\nu}\frac{t}{\gamma}\right)$$

where Γ is the Gamma function, K_ν is the modified Bessel function of the second kind, ν is a smoothness index, and γ is the length scale. It has been shown that Matérn kernels on Gaussian Processes satisfy the PDE

$$\left(\frac{2\nu}{\gamma^2} + \Delta\right)^{\frac{\nu}{2}+\frac{d}{4}}$$

where d is the dimension of x, y and Δ is the Laplacian operator. On graphs, obviously in the finite setting, after redefining variables the discrete Matérn kernel is then defined as

$$C = \left(L + \tau^2 I\right)^{-\alpha}$$

where L is the graph Laplacian matrix, and τ, α are refactored terms.

2.5. The Probit Method.

The probit method is a statistical technique used for modeling binary response variables. We assume

$$y_j = \operatorname{sign}(f(x_j) + \epsilon_j),$$

for some latent function f, where $\epsilon_j \overset{\text{iid}}{\sim} \psi$, for some probability density ψ. Assume ψ is symmetric, so that

$$\begin{aligned}
\mathbb{P}(y_j = +1 \mid f) &= \mathbb{P}(f(x_j) + \epsilon_j \geq 0) \\
&= \mathbb{P}(\epsilon_j \geq -f(x_j)) \\
&= \int_{-f(x_j)}^{\infty} \psi(t)\mathrm{d}t \\
&= \int_{-\infty}^{f(x_j)} \psi(t)\mathrm{d}t \\
&= \Psi(f(x_j)) \\
&= \Psi(y_j f(x_j)),
\end{aligned}$$

where Ψ is the cumulative distribution function (CDF) of ψ. By the same calculation, we obtain $\mathbb{P}(y_j = -1 \mid f) = \Psi(y_j f(x_j))$. Thus, $\mathbb{P}(y_j \mid f) = \Psi(y_j f(x_j))$. Since the ϵ_j are independent and identically distributed, we can write

$$\mathbb{P}(y \mid f) = \prod_{j=1}^{n} \Psi(y_j f(x_j)).$$

For a function f, we want our loss function to be large when the signs of $f(x_j)$ and y_j disagree, so we define our probit loss:

$$L(f) = -\log(\mathbb{P}(y \mid f)) = -\sum_{j=1}^{n} \Psi(y_j f(x_j)).$$

Finally, we add a regularization term based on the RKHS norm to define our optimization problem for the probit method:

$$f^* = \operatorname*{arg\,min}_{f \in \mathbb{R}^m} L(f) + \lambda f^T C^{-1} f.$$

Here, C is as defined in section 2.4. Since C is defined from the graph Laplacian matrix, the optimization problem thus includes spectral information of L, hence embedding geometric cluster information in the assigning of labels when such a minimizing function is found.

2.6. **Regression.** Instead of using the probit model, we can instead use the simpler regression loss:

$$L(f) = \sum_{j=1}^{n} |f_j - y_j|^2.$$

3. Methods

3.1. General Procedure.

(1) Data is obtained on a given manifold. True labels for the data are either extracted or assigned and are ultimately used to compute the accuracy of our approach.

(2) A graph is constructed from the unlabelled data, used to infer structure on the data.

(3) The minimal latent function is determined by solving an optimization problem over the sum of a loss function using only the known labels and regularization term.

(4) The minimal latent function is used to predict the labels on all unlabelled vertices.

(5) The accuracy of the predicted labels using the true labels is computed.

3.2. **Subproblems.** We now perform our general GSSL procedure on three different subproblems: three well separated clusters on the $\mathbb{R}^2$ manifold, three clusters on the swiss roll manifold, and the clusters generated using two digits from the MNIST data set. For each

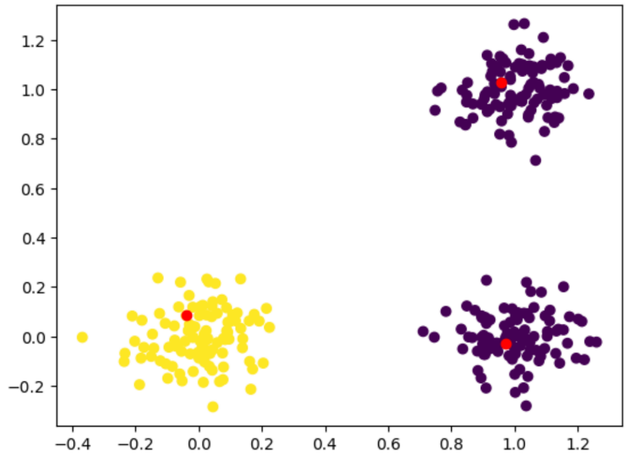


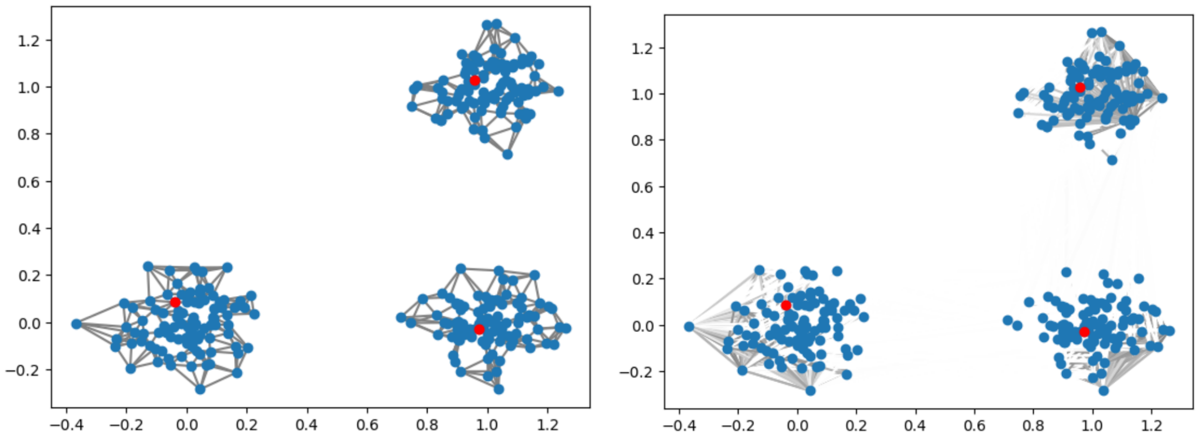
Figure 1. Plot of the 3 well-separated clusters; the yellow points represent a label of $+1$, while purple points represent a label of -1. The red points indicate the points for which the label is known.

subproblem, we perform GSSL on a disconnected and weakly connected graph, except for MNIST, on which we only use a weakly connected graph. We compute our minimal latent function using probit and regression loss.

4. Results

4.1. **Clusters on $\mathbb{R}^2$.** We begin by generating the points to form well-separated clusters in $\mathbb{R}^2$. These clusters are formed by specifying the mean and a covariance matrix and the points are distributed normally. We demonstrate our results by choosing 3 clusters centered around $(0, 0)$, $(1, 0)$, and $(1, 1)$, with 100 points in each cluster. One point in each cluster is then assigned its true label. Only these three assigned points will be used in our GSSL loss function. A plot of the clusters formed using this process is shown in Figure 1.

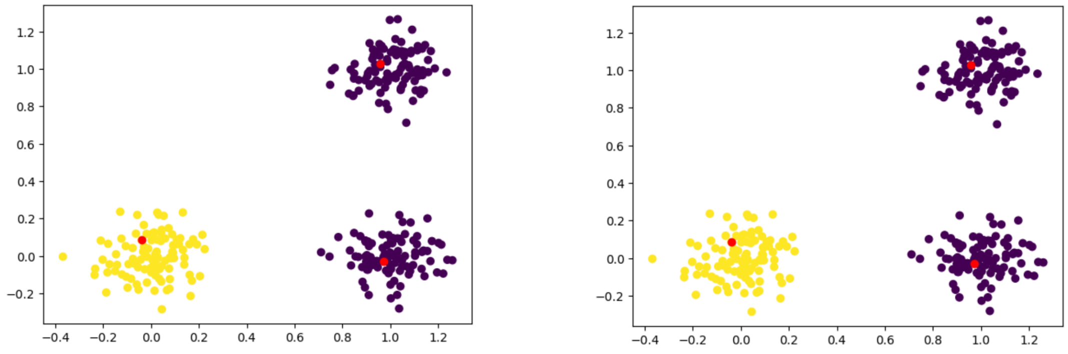
We next use the data to construct a graph on the points. There are several choices to be made here, including the choice of weight function (and its parameters) and the type of graph. In this case, we explore two types of graphs: K-NN with uniform weight function and proximity with Gaussian weights. We begin by examining the K-NN approach. This allows us to create a disconnected graph with 3 connected clusters. Such a graph constructed using this approach is shown in Figure 2a.



(A) K-NN Approach (B) Proximity Approach

Figure 2. Plot of the graphs constructed on the unlabelled data from Figure 1. Blue points indicate an unknown label, while red labels indicate known labels. The darker the shade of the edge, the higher its weight is.

We then perform our GSSL algorithm with both the Probit loss function and the regression loss function. Since the clusters in this case are completely disconnected that it was not necessary to tune any parameters in order to achieve an accuracy of 100% using both loss functions. For each of these loss functions, we choose the following parameters: $\tau = 1$, $\alpha = 2$, and $\lambda = \frac{\tau^{2\alpha}}{2}$. Plots of the predicted labels using both the probit and regression loss functions are shown in Figure 3.



(A) Probit Loss (B) Regression Loss

Figure 3. Plot of the predicted labels for the K-NN graph in Figure 2a and proximity graph in Figure 2b using only the known labels.

We repeat the process with the proximity approach. We create a fully connected graph in which the connections between the clusters are of $O(\epsilon)$. Usage of the RBF kernel requires

Type of Graph	Loss Function	Average Accuracy
Disconnected (K-NN)	Probit	100%
Disconnected (K-NN)	Regression	100%
Weakly Connected (Proximity)	Probit	100%
Weakly Connected (Proximity)	Regression	100%

Table 1. Average Accuracy of predicted labels over 50 runs for both a fully disconnected and a weakly connected graph using both probit and regression loss functions.

tuning the (length scale). We accomplish this by starting with the first quartile of pairwise distances between points and using cross validation on the eigenvalues of L to further tune. We expect three distinct clusters, so there should be a gap between the third and fourth eigenvalues. With the RBF kernel tuned, the proximity graph is shown in Figure 2b is produced. Here, it was necessary to tune τ, so that it is $O(\sqrt{\epsilon})$. Doing this resulted in 100% accuracy for both loss functions and thus produced the same plots as shown in Figure 3. To ensure that our accuracy results were consistent, we computed the average accuracy over 50 trials of each of the cases. These results are shown in Table 1.

4.2. **Clusters on the Swiss Roll.** Consider the Swiss roll manifold described by $(x, y) = (t\cos(t), t\sin(t))$. We begin by generating points to form well-separated clusters on this manifold. These clusters are formed by specifying a range of t-values for which each cluster will be defined, a variance for how far the points will vary from the true manifold, and the number of points to be generated. To avoid majority labelling we ensure that there are the same number of points labeled for each class. A plot of the clusters formed using this process is shown in Figure 4.

We choose to construct three different types of graphs for this example: a K-NN graph with uniform weights, a K-NN graph with the Gaussian weights, and a proximity graph with Gaussian weights. We tune the parameters for all of these approaches using the method described previously. Graphs formed using each of these techniques is shown in Figure 5. We then perform our GSSL algorithm with both the Probit loss and regression loss. Plots of the predicted labels using both the Probit and regression loss functions for each type of

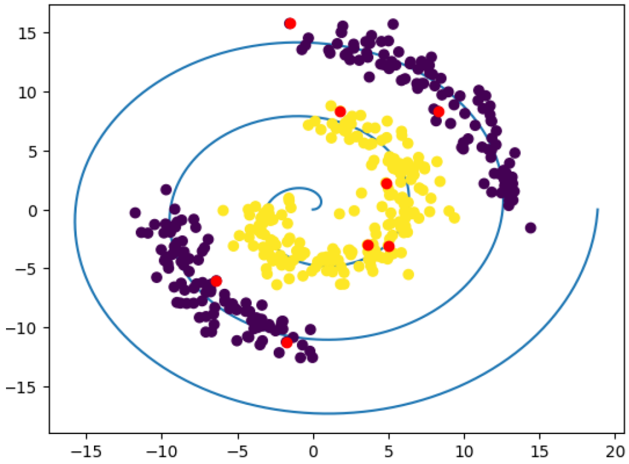
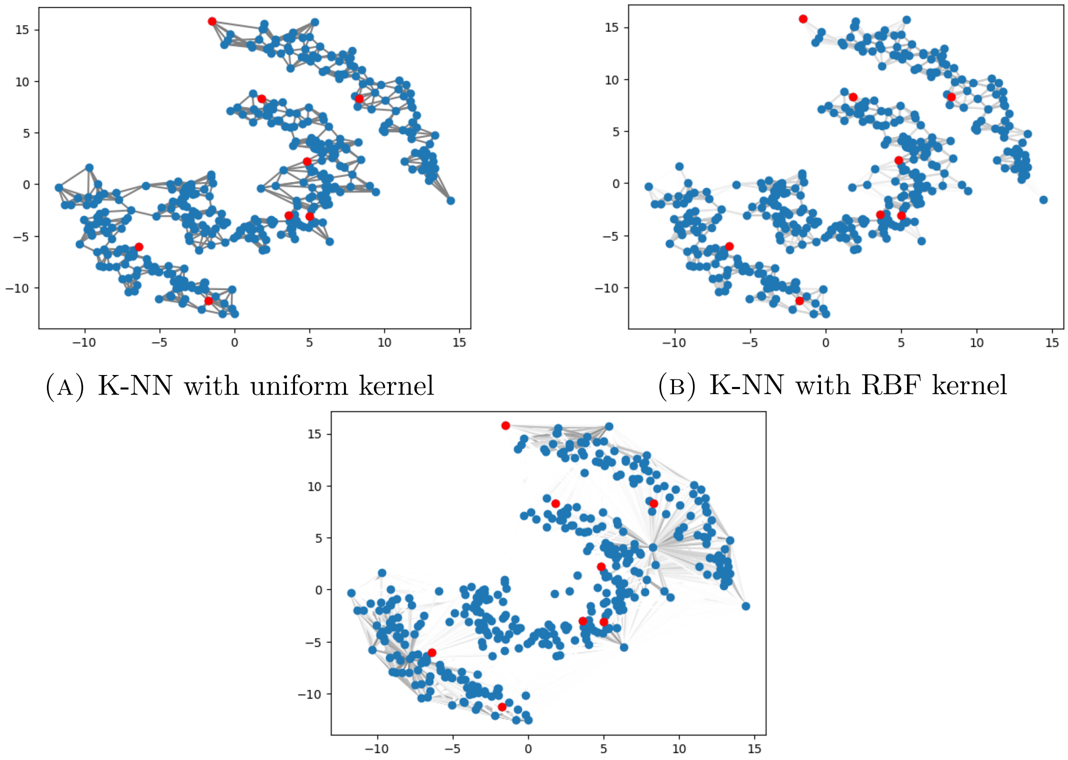


FIGURE 4. Plot of the 3 clusters on the Swiss roll manifold with t-ranges $[\pi, 5\pi/2]$, $[3\pi, 7\pi/2]$, and $[4\pi, 9\pi/2]$ with variance 1. The blue sprial represents the true swiss roll manifold for which our clusters are generated around.



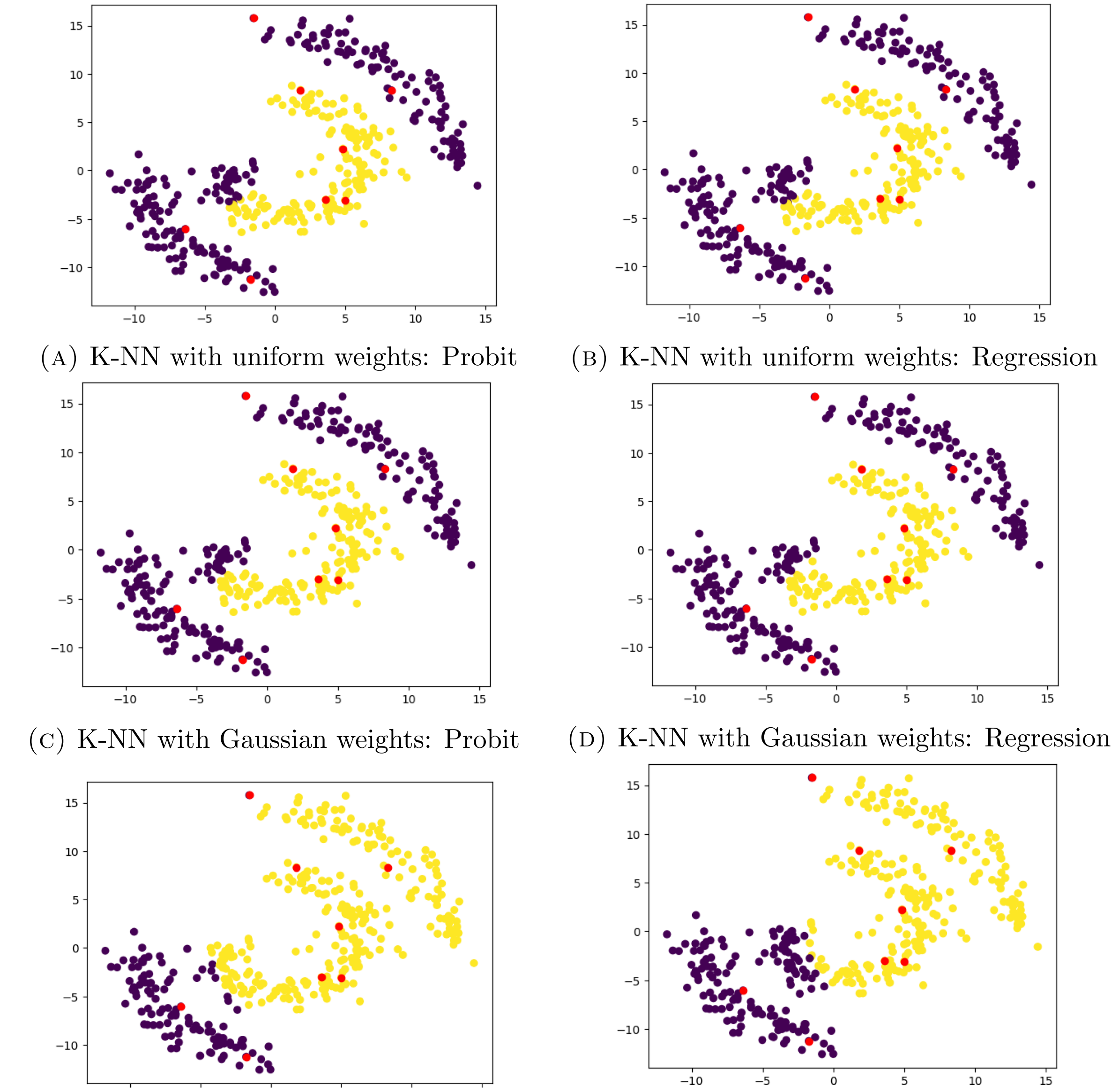
(A) K-NN with uniform kernel

(B) K-NN with RBF kernel

(C) Proximity with RBF kernel

FIGURE 5. Plot of the graphs formed for the data in Figure 4. Each subfigure shows a different graph forming technique.

graph are shown in Figure 6. The accuracies of the predicted labels compared to the true labels for each type of graph and each loss function is shown in Table 2.



(A) K-NN with uniform weights: Probit

(B) K-NN with uniform weights: Regression

(C) K-NN with Gaussian weights: Probit

(D) K-NN with Gaussian weights: Regression

(E) Proximity with Gaussian weights: Probit

(F) Proximity with Gaussian weights: Regression

Figure 6. Plot of the predicted labels for the clusters generated on the Swiss roll manifold using the graphs generated in Figure 5.

We can see from Figure 6 and Table 2 that the proximity graph with the RBF kernel performs poorly in comparison with both of the K-NN graphs that both have over 90% accuracy. Thus, we proceed only with the two K-NN graphs. To ensure the consistency of the success of the K-NN kernels, we computed the average accuracy over 50 trials for the two K-NN graphs with both the probit and regression loss functions. The average accuracy of over these 50 trials are shown in Table 2.

Type of Graph	Loss Function	1 Trial Accuracy	50 Trial Avg. Accuracy
K-NN with uniform kernel	Probit	91.00%	94.80%
K-NN with uniform kernel	Regression	91.00%	93.51%
K-NN with RBF kernel	Probit	92.25%	96.30%
K-NN with RBF kernel	Regression	92.50%	95.76%
Proximity with RBF kernel	Probit	73.00%	N/A
Proximity with RBF kernel	Regression	63.00%	N/A

Table 2. Accuracy of predicted labels over 1 run and average accuracy over 50 runs for 3 different types of graphs using both probit and regression loss functions. Note that we did not perform 50 trials for the proximity graph with the Gaussian weights as this approach performed poorly in our initial tests.

4.3. MNIST Data.

We now consider applying our GSSL algorithm to real data. In this case, we consider a subset of the MNIST data set. In particular, we look at 500 randomly chosen images from the complete set of 0s and 1s. We assigned each 0 a true label of -1 and each 1 a true label of 1. Thirty points in the data set are then assigned their true label. We next construct a K-NN graph with Gaussian weights. We then perform our GSSL algorithm with both the Probit and regression loss functions. After cross-validation, we found that the ideal parameters were given by: $\gamma = 0.04$, $\tau = 1$, $\alpha = 2$, and $\lambda = \tau^{2\alpha}/2$. These parameters resulted in the formation of two clusters, as desired. Our approach predicted labels with 89.6% accuracy for the Probit loss function and 90.8% accuracy for the regression loss function, showing that our approach is effective on real data.

5. Conclusions

Graphical semi-supervised learning is a useful tool that combines the strengths of both supervised and unsupervised learning methods. Using information from both labeled and unlabeled data allows for the development of robust machine learning models that require less information than traditional approaches. However, this approach is not without its limitations. The quality of the graph construction, the choice of kernel, and the selection of labeled points all have an impact on the final performance of the model.

References

[1] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Peter Deisenroth, and Nicolas Durrande. Matérn Gaussian processes on graphs, April 2021.

[2] Franca Hoffmann, Bamdad Hosseini, Zhi Ren, and Andrew M. Stuart. Consistency of semi-supervised learning algorithms on graphs: Probit and one-hot methods, March 2020.

[3] Daniel Sanz-Alonso and Ruiyi Yang. The SPDE approach to Matérn fields: Graph representations, April 2021.