GSSL PROJECT REPORT (SHOULD CHANGE THIS)

ROHIN GILMAN, ALEX JOHNSON, AND KAITLYNN LILLY

Abstract. To be written last

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. Incorporating kernel methods into GSSL has emerged as a popular approach in machine learning due to its ability to leverage both labeled and unlabeled data, resulting in improved accuracy and robustness. In kernel methods, the unlabeled data is used to construct a similarity graph, where each node represents an instance and the edges indicate the similarity between them. The graph is then used to propagate the labels from the labeled data to the unlabeled data, resulting in a complete labeling of the data set. As a result, graphical semi-supervised learning using kernel methods is a promising approach for tackling real-world problems in various domains such as image recognition, natural language processing, and bioinformatics.

2. Theoretical Background

2.1. Types of Graphs.

2.1.1. *Proximity Graphs.* Given a set of n points $\{x_1, x_2, \ldots, x_n\}$ in a metric space, a proximity graph is a 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 representing the edges connecting pairs of vertices that are "close" to each other, according to some measure of proximity.

Date: June 9, 2023.

To represent the proximity graph G using a weight matrix, we define a kernel function $K\colon X \times X \to \mathbb{R}$ that maps pairs of points in X to a real number, which captures the similarity between the points.

The weight matrix, W, corresponding to the proximity graph G is then defined by $W_{i,j} = K(x_i, x_j)$. The nonzero entries correspond to the similarity between pairs of points that are connected by an edge in G. Diagonal entries, $W_{i,i}$, correspond to the "self-similarity" of each point x_i, and the off-diagonal entries, $W_{i,j}$, correspond to the similarity between pairs of points x_i and x_j. By using a kernel function to define the weights of the edges, proximity graphs can capture underlying geometric relationships between the data points, making them suitable for a wide range of applications in machine learning and data analysis.

2.1.2. *K-Nearest Neighbor.* K-Nearest Neighbor (K-NN) graphs are a type of proximity graph for which edge weights are determined exclusively by the nearest neighbors of a given vertex. Given a set of n vertices $V = \{x_1, x_2, ..., x_n\}$ in a metric space, the K-NN graph $G_k = (V, W)$ is constructed as follows: for each point x_i, its k-nearest neighbors are found according to some distance metric, and edges of weight 1 are added between x_i and each of its k-nearest neighbors. In other words, x_i is adjacent to the k vertices that are closest to it in the data set.

K-NN graphs have several advantages over other types of proximity graphs. One of these advantages is that on average, vertices have relatively low degree, so the weight matrix W is often sparse. This increases the performance of graphical methods. Moreover, K-NN graphs are easy to construct and can be used in a wide range of applications, such as clustering, classification, and anomaly detection.

2.2. **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. The degree of a vertex is the sum of the weights of the edges adjacent to the vertex. We note that if G has M connected components, then $\text{Null}(L) = M$. Thus, we simply need to determine the degree of the zero eigenvalue of L to determine the number of clusters in the data.

One common variant of the graph Laplacian is the normalized Laplacian, which is defined as:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}.$$

The normalized Laplacian satisfies the important property that its eigenvalues are in the range $[0, 2]$, which allows for efficient algorithms and analysis.

2.3. **Types of Kernels.**

2.3.1. *Matérn Family.* The Matérn family is a class of parametric covariance functions that includes a wide range of covariance functions that are characterized by two parameters: a smoothness parameter, ν, and a length scale parameter, ℓ. The smoothness parameter determines how quickly the correlation between two points decays as the distance between them increases, while the length scale parameter controls the range of spatial dependence.

In GSSL, the Matérn family is often used as a graph Laplacian regularization term to enforce smoothness and continuity in the labeling of graph vertices. The graph Laplacian is constructed based on the pairwise similarities or distances between data points, and the Matérn family is used to model the spatial dependence in these pairwise similarities or distances. By including the Matérn family as a regularization term in the graph Laplacian, GSSL algorithms can effectively leverage the spatial structure of the data to improve the quality of the labeling.

2.3.2. *Green's Function of Elliptic Differential Operators.* A Green's function is a solution to the equation:

$$\mathcal{L}G(x, y) = \delta(x - y),$$

where $\mathcal{L}$ is an elliptic differential operator, δ is the Dirac delta function, and $G(x, y)$ is the Green's function. If we have a linear elliptic differential equation of the form

$$\mathcal{L}u(x) = f(x),$$

with boundary conditions, we can solve for the solution $u(x)$ using the Green's function as follows:

$$u(x) = \int G(x, y)f(y)\mathrm{d}y,$$

where the integral is taken over the domain of the equation.

Green's functions of elliptic differential operators provide a natural way to define diffusion processes on graphs in GSSL, where the information from the labeled points diffuses through the graph to the unlabeled points. This diffusion process can be described using a diffusion kernel, which is defined in terms of the graph Laplacian, an elliptic differential operator on the graph. The Green's function of the graph Laplacian can be used to construct the diffusion kernel, which in turn can be used to define a diffusion process that spreads the information from the labeled points to the unlabeled points. This diffusion process can then be used to make predictions for the unlabeled data points, based on the information that has diffused from the labeled points.

2.4. **GSSL Algorithms.**

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

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

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.

3. Methods

Write about what it is we are actually going to do

4. Methods

In this section, we describe our process for performing GSSL on several subproblems.

4.1. **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.

4.2. **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 subproblem, we perform GSSL on a disconnected and weakly connected graph. We compute our minimal latent function using probit and regression loss.

5. Results

5.1. **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 centers of the clusters, the number of points in each distribution, and a covariance matrix. We demonstrate our results by choosing 3 clusters centered around $(0, 0)$, $(1, 0)$, and $(1, 1)$, with 100 points in each cluster. The points in the cluster centered around $(0, 0)$ have true label $+1$, while the points in the other two clusters have true label -1. 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 functions, parameters, and type of graph. In this case, we explore two types of graphs: a K-NN approach and a full proximity graph approach. We begin by examining a K-NN approach using a uniform kernel. This allows us to create a disconnected graph with 3 connected clusters. Such a graph constructed using

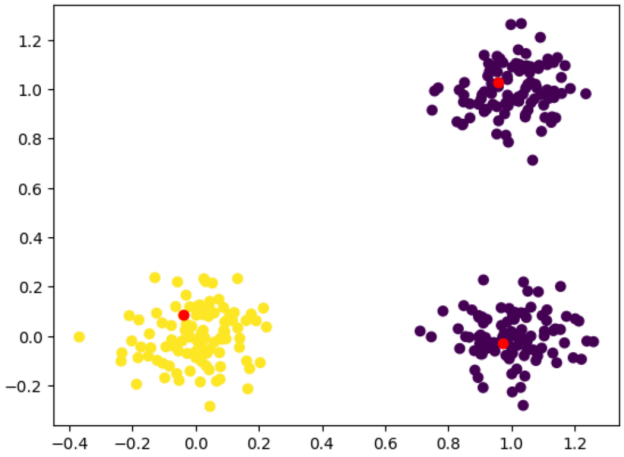


FIGURE 1. Plot of the 3 well-separated clusters; the yellow points represent a label of $+1$, while purple points represent a label of -1. The red points indicate the points for which the label is known.

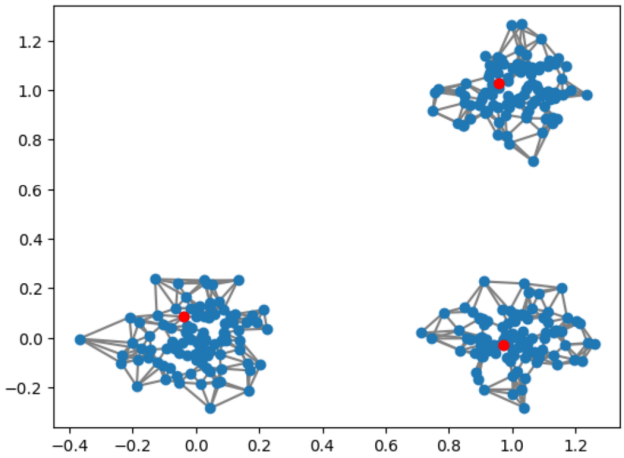


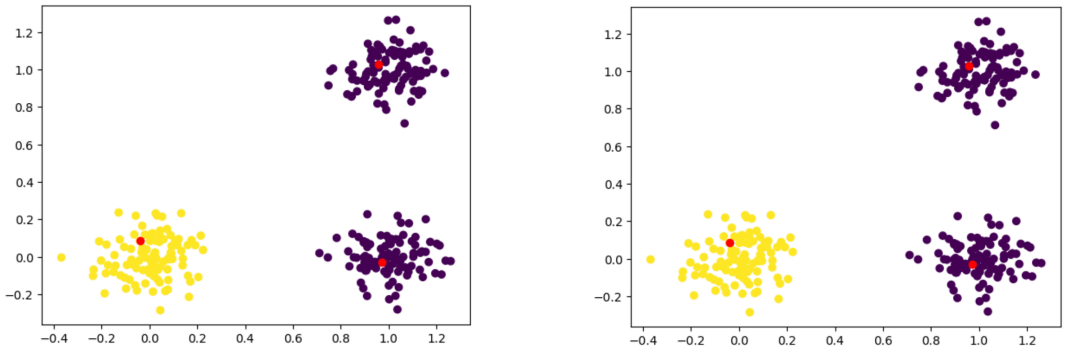
FIGURE 2. Plot of the graph constructed on the unlabelled data from Figure 1 when using a K-NN approach with a uniform kernel. Blue points indicate an unknown label, while red labels indicate known labels.

this approach is shown in Figure 2. Note that blue points indicate an unknown label, while red labels indicate known labels. We then perform our GSSL algorithm with both the Probit loss function as described in Section **??** and the regression loss function, which is given by

$$L(f) = \sum_j (y_j - f_j)^2.$$

We note that 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. From Figure 3, we can see that our predicted labels are



(A) Probit Loss Function (B) Regression Loss Function

FIGURE 3. Plot of the predicted labels for the K-NN graph in Figure 2 and proximity graph in Figure 4 using only the known labels.

exactly those as our known labels, indicating that our GSSL was successful in this case.

We now repeat the process with a full proximity graph approach. In this case, we construct a proximity graph using the RBF kernel. This allows us to create a fully connected graph in which the connections between the clusters are of $O(\epsilon)$. Usage of the RBF kernel requires tuning of the γ parameter in order to obtain the best results. We accomplish this with cross validation on the eigenvalues of the resulting graph Laplacian matrix. Since we expect three distinct clusters, there should be a large gap between the third and fourth eigenvalues. With the RBF kernel tuned, the proximity graph shown in Figure 4 is produced.

We then perform our GSSL algorithm. In this case it was necessary to tune the τ parameter here, so that it is $O(\sqrt{\epsilon})$. Doing this resulted in 100% accuracy for both loss functions and thus produced the same graphs 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 described above. The results of this are shown in Table 1.

5.2. **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

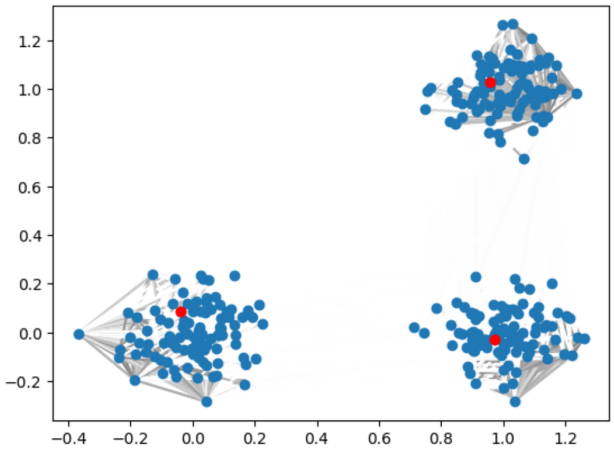


FIGURE 4. Plot of the graph constructed on the unlabelled data from Figure 1 when using a full proximity graph approach with the RBF kernel. Note that the darker the shade of the edge, the higher its weight is. Note that this graph is fully connected with weights between clusters that are difficult to make out due to them having weight of $O(\epsilon)$.

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.

will be defined, a variance for how far the points will vary from the true manifold, and the number of points to be generated. We demonstrate our results by choosing 3 clusters in the t-ranges $[\pi, 5\pi/2]$, $[3\pi, 7\pi/2]$, and $[4\pi, 9\pi/2]$, respectively with a variance of 1. The cluster from $t = [\pi, 5\pi/2]$ has 200 points and have true label $+1$, while the other two clusters each have 100 points and have true label -1. Two points in each cluster are then assigned its true label. We then further assigned two more points in the cluster labelled $+1$ their true label to ensure that each label is assigned equal numbers of true labels. This is done to avoid majority labelling, which resulted in poor accuracy when not accounted for. A plot of the clusters formed using this process is shown in Figure 5.

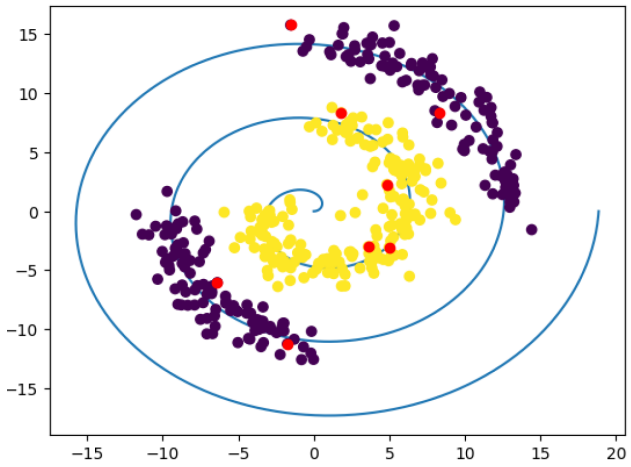


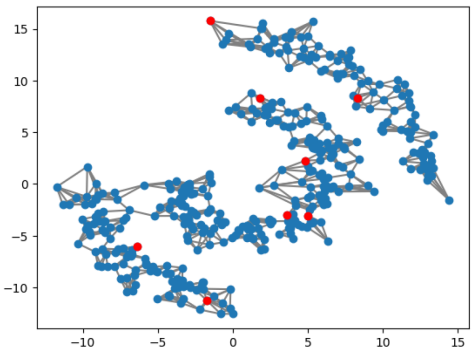
FIGURE 5. Plot of the 3 clusters on the Swiss roll manifold with t-ranges $[\pi, 5\pi/2]$, $[3\pi, 7\pi/2]$, and $[4\pi, 9\pi/2]$. The blue sprial represents the true swiss roll manifold for which our clusters are generated around.

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

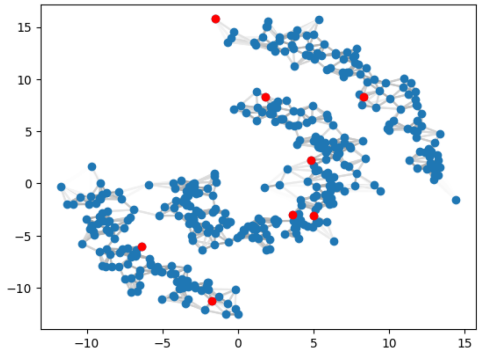
TABLE 2. Accuracy of predicted labels over 1 run and average accuracy over 50 runs for 3 different types of graphs using both probit and regression loss functions. Note that we did not perform 50 trials for the proximity graph with the RBF kernel as this approach performed very poorly in our initial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

We choose to construct three different types of graphs for this example: a K-NN graph with a uniform kernel, a K-NN graph with the RBF kernel, and a proximity graph with the RBF kernel. We tune the parameters for all of these approaches using the method descirbed in the previous section. Graphs formed using each of these techniques is shown in Figure 6.

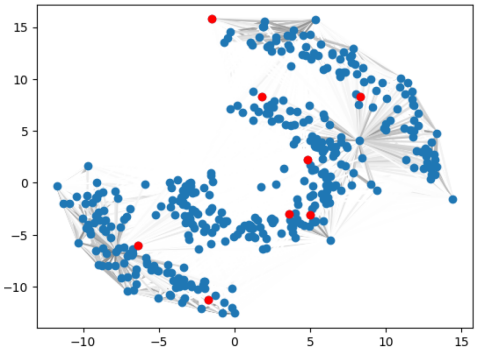
We then perform our GSSL algorithm with both the Probit loss function as described in Section **??** and the regression loss function given in Section 5.1. Plots of the predicted labels using both the probit and regression loss functions for each type of graph are shown in Figure 7. The accuracies of the predicted labels compared to the true labels for each type of graph and each loss function is shown in Table 2. We can see from Figure 7 and Table 2 that the proximity graph with the RBF kernel performs extremely poorly in comparison



(A) K-NN with uniform kernel



(B) K-NN with RBF kernel



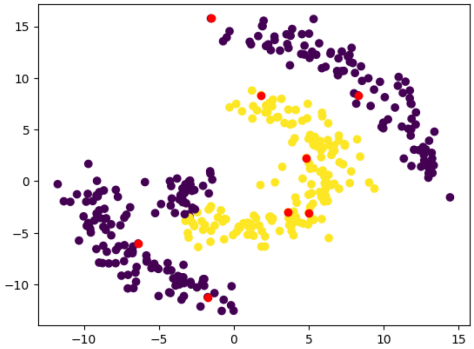
(C) Proximity with RBF kernel

FIGURE 6. Plot of the graphs formed for the data in Figure 5. Each subfigure shows a different graph forming technique.

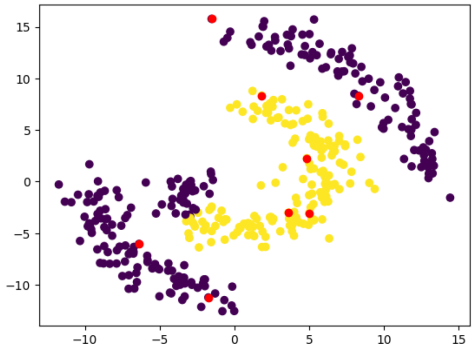
with both of the K-NN kernels that both have over 90% accuracy. As such, we proceed only with the two K-NN graphs. To ensure the consistency of the success of the K-NN kernels, we computed the average accuracy over 50 trials for the two K-NN graphs with both the probit and regression loss functions. The average accuracy of over these 50 trials are shown in Table 2.

5.3. **MNIST Data.**

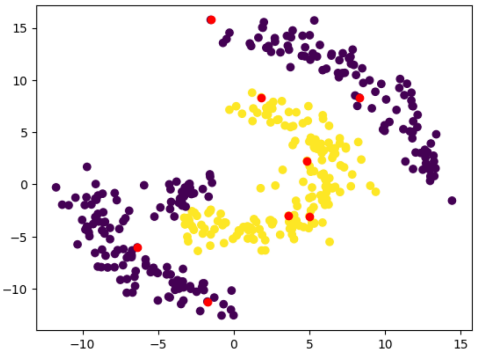
6. CONCLUSION



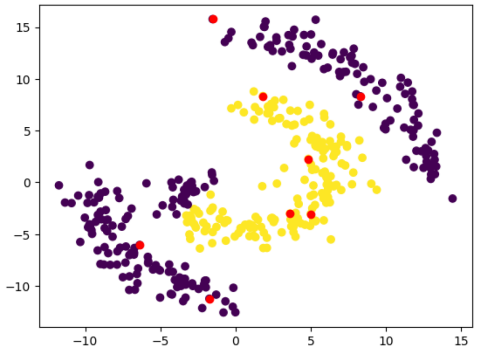
(A) K-NN with uniform kernel: Probit



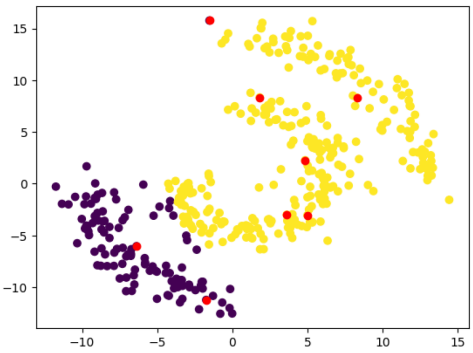
(B) K-NN with uniform kernel: Regression



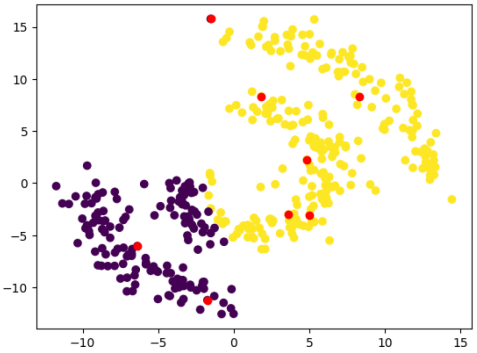
(C) K-NN with RBF kernel: Probit



(D) K-NN with RBF kernel: Regression



(E) Proximity with RBF kernel: Probit



(F) Proximity with RBF kernel: Regression

FIGURE 7. Plot of the predicted labels for the clusters generated on the Swiss Roll manifold using the graphs generated in Figure 6.