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 and clustering 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, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices representing the points, and E is a set of edges connecting pairs of vertices that are "close" to each other according to some proximity measure.

Date: June 9, 2023.

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

The weight matrix, W, corresponding to the proximity graph G is then defined as $W_{i,j} = K(x_i, x_j)$, where $W(i,j)$ is a matrix where 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 nonlinear 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-NN).* K-Nearest Neighbor (K-NN) graphs are a type of proximity graph that is defined based on the k-nearest neighbors of each point in a given data set. Given a set of n points $\{x_1, x_2, ..., x_n\}$ in a metric space, the K-NN graph $G_k = (V, E)$ is constructed as follows: for each point x_i, its k-nearest neighbors are found according to some distance metric, and edges are added between x_i and each of its k-nearest neighbors. In other words, x_i is connected to the k points that are closest to it in the data set.

K-NN graphs have several advantages over other types of proximity graphs, as they capture the local structure of the data more accurately, because they only connect points that are truly close to each other, while excluding points that are far away, making the weight matrix, W, sparse. 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 $G = (V, E)$, 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, with $D_{ij} = \sum_k W_{ik}$ for $i = j$ and $D_{ij} = 0$ for $i \neq j$, and W is the weight matrix. We note that if G has M-connected components then $\text{Nullity}(L) = M$. Thus, we simply need to count the number of zero eigenvalues 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.

In GSSL, we can use the Laplacian to propagate labels from a small set of labeled data points to the entire graph, by solving the following optimization problem:

$$\min_f \sum_i L_{ii}(f_i - y_i)^2 + \lambda \sum_{i,j} W_{ij}(f_i - f_j)^2,$$

where y_i is the label for the i-th labeled point, f_i is the predicted label for the i-th point, and λ is a regularization parameter that controls the smoothness of the solution.

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. In the kernel-based formulation of the probit method, the goal is to find the optimal values of the model coefficients, or dual variables, by maximizing the likelihood function. The likelihood function is defined as the product of the conditional probabilities of the response variable given the explanatory variables, which are modeled as a probit link function of a linear combination of kernel functions evaluated at the explanatory variables:

$$\mathbb{P}_r(y_i = 1|\mathbf{x}_i) = \Phi(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i)),$$

where y_i is the response variable for the i-th observation, $\mathbf{x}_i$ is the vector of explanatory variables, $\boldsymbol{\phi}(\cdot)$ is a nonlinear mapping of the input space to a high-dimensional feature space, $\mathbf{w}$ is a vector of model coefficients, and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. The model coefficients are then estimated by solving an optimization problem of the form:

$$\min_{\mathbf{w}} \left[-\sum_{i=1}^{n} y_i \log \Phi(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i)) + (1 - y_i) \log(1 - \Phi(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \right],$$

where the first two terms correspond to the negative log-likelihood function, the third term is a regularization term that controls the complexity of the model, and λ is the regularization parameter. This optimization problem can be solved using kernel ridge regression, which involves minimizing a regularized empirical risk function of the form:

$$\min_{\boldsymbol{\alpha}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\mathbf{x}_i)) + \frac{\lambda}{2} \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha},$$

where $\boldsymbol{\alpha}$ are the dual variables, $f(\mathbf{x}_i) = \sum_{j=1}^{n} \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$ is the predicted probability for the i-th observation, $\ell(y_i, f(\mathbf{x}_i))$ is the loss function that measures the discrepancy between the observed response y_i and the predicted probability $f(\mathbf{x}_i)$, λ is the regularization parameter that controls the trade-off between fitting the data and the complexity of the model, and $\mathbf{K}$ is the kernel matrix whose entries are given by $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j)$.

2.4.2. *Spectral Clustering.* Spectral clustering is a clustering technique that uses the eigenvectors of a matrix derived from the data to cluster the data points. One common approach is to use the Laplacian matrix of the graph defined by the data points. Once we have the

Laplacian matrix, we can compute its eigenvectors and eigenvalues. The eigenvectors corresponding to the smallest eigenvalues (excluding the first eigenvector, which is constant) are used as the embedding of the data points. The embedding maps the data points to a low-dimensional space, where the pairwise distances reflect the underlying structure of the data. Finally, we can use a clustering algorithm (e.g., k-means) to cluster the data points in the embedding space. The number of clusters can be determined using the number of zero eigenvalues of the graph Laplacian.

3. Methods

Write about what it is we are actually going to do

4. Results

5. Conclusion