FINAL PROJECT FOR THE COURSE ADVANCED FOUNDATIONS OF MACHINE LEARNING: ONLINE LEARNING OVER GRAPHS*

PAUL SOPHER LINTILHAC† AND THOMAS NANFENG LI‡

MAY 9TH, 2017

1 Introduction to Graph

1.1 Concepts and Definitions

We first summarise some key concepts of graph theory, for more detailed knowledge, we refer to Bapat (2014). A ***simple graph***, that is, graph without loops and parallel edges, $G(V, E)$ consists of a finite set of ***vertices*** $V(G)$ and a set of ***edges*** $E(G)$ consisting of distinct, unordered pairs of vertices. $V(G) = \{v_1, v_2, \cdots, v_n\}$ is called the vertex set with $n = |V(G|$, $E(G) = \{e_{ij}\}$ is called the edge set with $m = |E(G)|$. An edge e_{ij} connects vertices v_i and v_j if they are ***adjacent*** or neighbours, which is denoted by $v_i \sim v_j$. The number of neighbours of a vertex v is called the ***degree*** of v and is denoted by $d(v)$, therefore, for each vertex, $d(v_i) = \sum_{v_i \sim v_j} e_{ij}$. If all the vertices of a graph have the same degree, the graph is ***regula***r, the vertices of an ***Eulerian Graph*** have even degree. A graph is ***complete*** if there is an edge between every pair of vertices.

$H(G)$ is a $\boldsymbol{sub-graph}$ of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A sub-graph $H(G)$ is an $\boldsymbol{induced\ sub-graph}$ of G if two vertices of $V(H)$ are adjacent if and only if they are adjacent in G. A ***clique*** is a complete sub-graph of a graph. A ***path*** of k vertices is a sequence of k distinct vertices such that consecutive vertices are adjacent. A ***cycle*** is a connected sub-graph where every vertex has exactly two neighbours. A graph containing no cycles is a ***forest***. A connected forest is a ***tree***.

We define incidence matrix of graph. Let $G(V, E)$ be a graph with $V(G) = \{v_1, v_2, \cdots, v_n\}$ and $E(G) = \{e_1, e_2, \cdots, e_m\}$. Suppose each edge of $G(V, E)$ is assigned an orientation, which is arbitrary but fixed. The vertex-edge $\boldsymbol{incidence\ matrix}$ of $G(V, E)$, denoted by $\boldsymbol{Q}(G)$, is the $n \times m$ matrix defined as follows. The rows and the columns of $\boldsymbol{Q}(G)$ are indexed by $V(G)$ and $E(G)$, respectively. The (i, j) entry of $\boldsymbol{Q}(G)$ is 0 if vertex i and edge e_j are not incident, and otherwise it is -1 or 1 according as e_j originates or terminates at i, respectively. For instance, the incidence matrix $\boldsymbol{Q}(G)$ of the graph that is shown in figure 1.1 is

$$\boldsymbol{Q}(G) = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} . \tag{1.1}$$

We introduce adjacency matrix of graph. Let $G(V, E)$ be a graph with $V(G) = \{v_1, v_2, \cdots, v_n\}$ and $E(G) = \{e_1, e_2, \cdots, e_m\}$. The $\boldsymbol{adjacency\ matrix}$ of $G(V, E)$, denoted by $\boldsymbol{A}(G)$, is the $n \times n$ matrix defined as follows. The rows and the columns of $\boldsymbol{A}(G)$ are indexed by $V(G)$. If $i \neq j$ then the (i, j) entry of $\boldsymbol{A}(G)$ is 0 for vertices i and j non-adjacent, and the (i, j) entry is 1 for i and j adjacent. The (i, j) entry of $\boldsymbol{A}(G)$ is 0 for $i = j = 1, \cdots, n$. For instance, the adjacency matrix $\boldsymbol{A}(G)$ of the graph that is shown in

*New York University Courant Institute of Mathematical Sciences. Spring 2017. Professor Mehryar Mohri, Ph.D.

†New York University School of Engineering. psl274@nyu.edu
‡New York University School of Engineering. nl747@nyu.edu

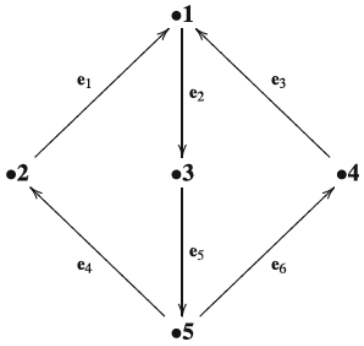

Figure 1.1: Example of Incidence Matrix of Graph

figure 1.2 is

$$\boldsymbol{A}(G) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} . \tag{1.2}$$

Clearly $\boldsymbol{A}$ is a symmetric matrix with zeros on the diagonal. The (i, j) entry of $\boldsymbol{A}^k$ is the number of walks of length k from i to j.

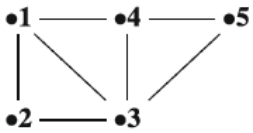

Figure 1.2: Example of Adjacency Matrix of Graph

We define degree matrix of graph. Let $G(V, E)$ be a graph with $V(G) = \{v_1, v_2, \cdots, v_n\}$ and $E(G) = \{e_1, e_2, \cdots, e_m\}$. The ***degree matrix*** $\boldsymbol{D}(G)$ for $G(V, E)$ is a $n \times n$ diagonal matrix defined as

$$\boldsymbol{D}(G)_{i,j} \coloneqq \begin{cases} d(v_i) & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

According to this definition, the degree matrix of figure 1.2 is

$$\boldsymbol{A}(G) = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} . \tag{1.3}$$

Weighted graph $G(V, E, W)$ is a graph with real edge weights given by $w : E \to \mathbb{R}$. Here, the weight $w(e)$ of an edge e indicates the similarity of the incident vertices, and a missing edge corresponds to zero similarity. The ***weighted adjacency matrix*** $\boldsymbol{W}(G)$ of the graph $G(V, E, W)$ is defined by

$$\boldsymbol{W}_{ij} \coloneqq \begin{cases} w(e) & \text{if } e = (i, j) \in E \\ 0 & \text{otherwise.} \end{cases} . \tag{1.4}$$

The weight matrix $\boldsymbol{W}(G)$ can be, for instance, the k-nearest neighbour matrix $\boldsymbol{W}(G)_{ij} = 1$ if and only if vertex v_i is among the k-nearest neighbours of v_j or vice versa, and is 0 otherwise. Another typical weight matrix is given by the Gaussian kernel of width σ

$$\boldsymbol{W}(G)_{ij} = e^{-\frac{\|v_i - v_j\|^2}{2\sigma^2}}. \tag{1.5}$$

Then the ***degree matrix*** $\boldsymbol{D}(G)$ for weighted graph $G(V, E, W)$ is defined by

$$\boldsymbol{D}(G)_{i,i} \coloneqq \sum_j \boldsymbol{W}(G)_{ij} \tag{1.6}$$

1.2 Graph Laplacian

The graph Laplacian $\boldsymbol{L}(G)$ is defined in two different ways. The ***normalized graph Laplacian*** is

$$\boldsymbol{L}(G) \coloneqq \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}}, \tag{1.7}$$

and the ***unnormalized graph Laplacian*** is

$$\boldsymbol{L}(G) \coloneqq \boldsymbol{D} - \boldsymbol{W}. \tag{1.8}$$

Let us consider an example to understand the graph Laplacian of the graph that is shown in figure 1.3. Suppose $\boldsymbol{f} : V \to \mathbb{R}$ is a real-valued function on the set of the vertices of graph $G(V, E)$ such that it

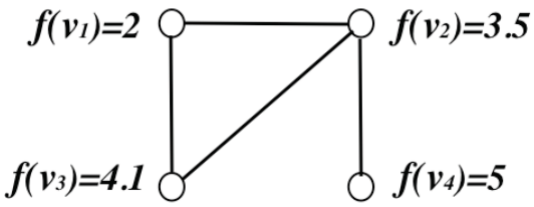

Figure 1.3: Real-Valued Functions on a Graph

assigns a real number to each graph vertex. Therefore, $\boldsymbol{f} = (f(v_1), f(v_2), \cdots, f(v_n))^T \in \mathbb{R}^n$ is a vector indexed by the vertices of graph. Its adjacency matrix is

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{1.9}$$

Hence, the eigenvectors of the adjacency matrix, $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$, can be viewed as eigenfunctions $\boldsymbol{A}\boldsymbol{f} = \lambda \boldsymbol{f}$. The adjacency matrix can be viewed as an operator

$$\begin{aligned} \boldsymbol{g} &= \boldsymbol{A}\boldsymbol{f} \\ g(i) &= \sum_{i \sim j} f(j), \end{aligned} \tag{1.10}$$

and it can also be viewed as a quadratic form

$$\boldsymbol{f}^T \boldsymbol{A} \boldsymbol{f} = \sum_{e_{ij}} f(i) f(j). \tag{1.11}$$

Assume that each edge in the graph have an arbitrary but fixed orientation, which is shown in figure 1.4. Then the incidence matrix of the graph is

$$\boldsymbol{Q} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} . \tag{1.12}$$

Therefore the co-boundary mapping of the graph $\boldsymbol{f} \to \boldsymbol{Q}\boldsymbol{f}$ implies $(\boldsymbol{Q}\boldsymbol{f})(e_{ij}) = f(v_j) - f(v_i)$ is

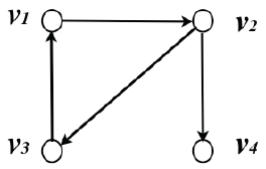

Figure 1.4: Orientation of the Graph

$$\begin{bmatrix} f(2) - f(1) \\ f(1) - f(3) \\ f(3) - f(2) \\ f(4) - f(2) \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \end{bmatrix} . \tag{1.13}$$

If we let

$$\boldsymbol{L} = \boldsymbol{Q}^T \boldsymbol{Q}, \tag{1.14}$$

then we have

$$(\boldsymbol{L}\boldsymbol{f})(v_i) = \sum_{v_i \sim v_j} [f(v_i) - f(v_j)] . \tag{1.15}$$

Hence, the connection between the Laplacian and the adjacency matrices is

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} , \tag{1.16}$$

where the degree matrix $\boldsymbol{D}$ is

$$\boldsymbol{D} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{1.17}$$

If we consider undirected weighted graphs, which is each edge e_{ij} is weighted by w_{ij}, then the Laplacian as an operator is

$$(\boldsymbol{L}\boldsymbol{f})(v_i) = \sum_{v_i \sim v_j} w_{ij} [f(v_i) - f(v_j)] . \tag{1.18}$$

Its quadratic form is

$$\boldsymbol{f}^T \boldsymbol{L}\boldsymbol{f} = \frac{1}{2} \sum_{e_{ij}} w_{ij} [f(v_i) - f(v_j)]^2 . \tag{1.19}$$

The intuition behind a Laplacian matrix is the following. If, for instance, we apply the Laplacian operator of formula 1.16 to the real-valued functions $\boldsymbol{f} = (f(v_1), f(v_2), f(v_3), f(v_4))^T$ of the set of the vertices of

graph $G(V, E)$, we have

$$(\boldsymbol{L}\boldsymbol{f})(v_i) = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix}. \tag{1.20}$$

For simplicity, let us only look at the first element

$$\begin{aligned} (\boldsymbol{L}\boldsymbol{f})(v_i)_1 &= \begin{bmatrix} 2 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix} \\ &= 2f(v_1) - f(v_2) - f(v_3) \\ &= -[f(v_2) - 2f(v_1) + f(v_3)] \\ &= -[f(v_2) - f(v_1) - f(v_1) + f(v_3)] \end{aligned} \tag{1.21}$$

If we label $f(v_1) = f_k$, $f(v_2) = f_{k+1}$, and $f(v_3) = f_{k-1}$, then we have

$$(\boldsymbol{L}\boldsymbol{f})(v_i)_1 = -[f_{k+1} - 2f_k + f_{k-1}]. \tag{1.22}$$

We recall that the second order derivative can be approximated by

$$\begin{aligned} f''(x) &= \frac{\frac{f(x+\Delta x)-f(x)}{\Delta x} - \frac{f(x)-f(x-\Delta x)}{\Delta x}}{\Delta x} \\ &= \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{(\Delta x)^2}. \end{aligned} \tag{1.23}$$

Hence, we observe that the graph Laplacian is the negative numerator of the finite difference approximation of the second derivative. The Laplacian matrix $\boldsymbol{L}$ is symmetric and positive semi-definite, and it has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. The number of 0 eigenvalues of the Laplacian matrix $\boldsymbol{L}$ is the number of connected components, because each connected component forms a block in the Laplacian matrix that only has edges within itself, and each block is the Laplacian for a small connected component and it has one zero eigenvalue, so the number of zeros is the number of blocks is the number of connected components.

2 Semi-Supervised Learning and Regularization over Graphs

Semi − supervised learning is halfway between supervised and unsupervised learning. In addition to unlabelled data, the algorithm is provided with some supervision information, but not necessarily for all examples. Often, this information will be the targets associated with some of the examples. In this case, the data set $\boldsymbol{X} = \{x_1, x_2, \cdots, x_n\}$ can be divided into two parts: the points $\boldsymbol{X}_l = \{x_1, x_2, \cdots, x_l\}$, for which labels $\boldsymbol{Y}_l = (y_1, y_2, \cdots, y_l)$ are provided, and the points $\boldsymbol{X}_u = \{x_{l+1}, x_{l+2}, \cdots, x_{l+u}\}$, the labels of which are not known.

The common denominator of semi-supervised learning algorithms over graphs is that the data are represented by the vertices of a graph, the edges of which are labelled with the pairwise distances of the incident vertices, and a missing edge corresponds to infinite distance. Most graph methods refer to the graph by utilizing the graph Laplacian.

2.1 Label Propagation

Given the graph $G(V, E, W)$, a simple idea for semi-supervised learning label propagation is to propagate labels on the graph. Starting with vertices $1, 2, \cdots, l$ labelled l with their known label1 or -1 and nodes $l+1, \cdots, n$ labelled with 0, each vertex starts to propagate its label to its neighbours, and the process is repeated until convergence. Zhou et al. (2004) proposed a label propagation algorithm, see algorithm 2.1, that uses graph Laplacian. Estimated labels on both labelled and unlabelled data are denoted by $\hat{\boldsymbol{Y}} = \left(\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u\right)$, where where $\hat{\boldsymbol{Y}}_l$ may be allowed to differ from the given labels $\boldsymbol{Y}_l = (y_1, y_2, \cdots, y_l)$. At each step a vertex i receives a contribution from its neighbours j and an additional small contribution given by its initial value.

Algorithm 2.1 Graph Laplacian Label Propagation Algorithm

Compute weight matrix $\boldsymbol{W}$ from formula 1.5 for $i \neq j$ and $\boldsymbol{W}_{ii} = 0$
Compute the diagonal degree matrix $\boldsymbol{D}$ by $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$
Compute the normalized graph Laplacian $\boldsymbol{L} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}^{-\frac{1}{2}}$
Initialize $\hat{\boldsymbol{Y}}^{(0)} = (y_1, y_2, \cdots, y_l, 0, 0, \cdots, 0)$
Choose a parameter $\alpha \in [0, 1]$
Iterate
$\quad \hat{\boldsymbol{Y}}^{(t+1)} = \alpha \boldsymbol{L} \hat{\boldsymbol{Y}}^{(t)} + (1-\alpha)\, \hat{\boldsymbol{Y}}^{(0)}$
until convergence to $\hat{\boldsymbol{Y}}^{(\infty)}$
Label point v_i by the sign of $\hat{y}_i^{(\infty)}$

Bengio, Delalleau and Roux (2006) proposed a similer label propagation scheme based on the Jacobi iterative method for linear systems. It uses an additional regularization term ϵ for better numerical stability, which is shown in algorithm 2.2.

Algorithm 2.2 Jacobi Iterative Label Propagation Algorithm

Compute weight matrix $\boldsymbol{W}$ from formula 1.5 such that $\boldsymbol{W}_{ii} = 0$
Compute the diagonal degree matrix $\boldsymbol{D}$ by $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{W}_{ij}$
Choose a parameter $\alpha \in (0, 1)$ and a small $\epsilon > 0$
$\mu = \frac{\alpha}{1-\alpha} \in (0, +\infty)$
Compute the diagonal matrix $\boldsymbol{A}$ by $\boldsymbol{A}_{ii} = \boldsymbol{I}_l(i) + \mu \boldsymbol{D}_{ii} + \mu\epsilon$
Initialize $\hat{\boldsymbol{Y}}^{(0)} = (y_1, y_2, \cdots, y_l, 0, 0, \cdots, 0)$
Iterate
$\quad \hat{\boldsymbol{Y}}^{(t+1)} = \boldsymbol{A}^{-1}\left(\mu \boldsymbol{W} \hat{\boldsymbol{Y}}^{(t)} + \hat{\boldsymbol{Y}}^{(0)}\right)$
until convergence to $\hat{\boldsymbol{Y}}^{(\infty)}$
Label point v_i by the sign of $\hat{y}_i^{(\infty)}$

2.2 Minimization of Cost Function

A method that is equivalent to label propagation algorithms is minimising a cost function derived from the graph $G(V, E, W)$. Given a labelling $\hat{\boldsymbol{Y}} = \left(\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u\right)$, consistency with the initial labelling can be measured by

$$\sum_{i=1}^{l} (\hat{y}_i - y_i)^2 = \left\| \hat{\boldsymbol{Y}}_l - \boldsymbol{Y}_l \right\|^2. \tag{2.1}$$

By following the smoothness assumption, if two points x_1 and x_2 in a high-density region are close, then so should be the corresponding outputs y_1 and y_2, we consider a penalty term of the form

$$\begin{aligned} \frac{1}{2} \sum_{i,j=1}^{n} \boldsymbol{W}_{ij} (\hat{y}_i - \hat{y}_j)^2 &= \frac{1}{2} \left(2 \sum_{i=1}^{n} \hat{y}_i^2 \sum_{j=1}^{n} \boldsymbol{W}_{ij} - 2 \sum_{i,j=1}^{n} \boldsymbol{W}_{ij} \hat{y}_i \hat{y}_j \right) \\ &= \hat{\boldsymbol{Y}}^T (\boldsymbol{D} - \boldsymbol{W}) \hat{\boldsymbol{Y}} \\ &= \hat{\boldsymbol{Y}}^T \boldsymbol{L} \hat{\boldsymbol{Y}}. \end{aligned} \tag{2.2}$$

This means we penalize rapid changes in $\hat{\boldsymbol{Y}}$ between points that are close, which is given by the similarity matrix $\boldsymbol{W}$. However, if there is noise in the available labels, it may be beneficial to allow the algorithm to relabel the labelled data. This could also help generalization in a noise-free setting where, for instance, a positive sample had been drawn from a region of space mainly filled with negative samples.

Based on this observation, Belkin, Matveeva and Niyogi (2004) proposed a more general cost criterion involving a trade-off between 2.1 and 2.2, which is shown in algorithm 2.3 and algorithm 2.4. The paper assumed that $G(V, E, W)$ is connected and that the vertices of the graph are numbered, and only partial information, $f(\boldsymbol{x}_i) = y_i$, $1 \leq i \leq k$, is given. The labels can potentially be noisy. They also allow data points to have multiplicities, i.e. each vertex of the graph may appear more than once with same or different y value. They precondition the data by mean subtracting first. That is we take

$$\tilde{\boldsymbol{y}} = (y_1 - \bar{y}, y_2 - \bar{y}, \cdots, y_k - \bar{y}), \tag{2.3}$$

where $\bar{y} = \frac{1}{k}\sum y_i$.

In the algorithm 2.3, $\boldsymbol{S}$ is a smoothness matrix, e.g. $S = L$ or $S = L^p$, $p \in \mathbb{N}$. The condition $\sum f_i = 0$ is needed to make the algorithm stable. The solution to the quadratic problem above is not hard to obtain by standard linear algebra considerations. We have the objective function $\frac{1}{k}\sum_i (f_i - \tilde{y}_i)^2 + \gamma \boldsymbol{f}^T \boldsymbol{S} \boldsymbol{f}$ and constraint $\sum f_i - 0$. Therefore, the Lagrangian is

$$\begin{aligned}
\mathcal{L} &= \frac{1}{k}\sum_i (f_i - \tilde{y}_i)^2 + \gamma \boldsymbol{f}^T \boldsymbol{S} \boldsymbol{f} - \mu\left(\sum f_i - 0\right) \\
&= \frac{1}{k}\boldsymbol{f}^T \boldsymbol{f} - \frac{2}{k}\boldsymbol{f}^T \tilde{\boldsymbol{y}} + \frac{1}{k}\tilde{\boldsymbol{y}}^T \tilde{\boldsymbol{y}} + \gamma \boldsymbol{f}^T \boldsymbol{S} \boldsymbol{f} - \mu \boldsymbol{f}^T \boldsymbol{1} \\
&= \boldsymbol{f}^T \left(\frac{1}{k}\boldsymbol{I} + \gamma \boldsymbol{S}\right) \boldsymbol{f} - \frac{2}{k}\boldsymbol{f}^T (\tilde{\boldsymbol{y}} + \mu \boldsymbol{1}) + \frac{1}{k}\tilde{\boldsymbol{y}}^T \tilde{\boldsymbol{y}}.
\end{aligned} \tag{2.4}$$

Then by taking $\frac{\partial \mathcal{L}}{\partial \boldsymbol{f}} = 0$, we have

$$\tilde{\boldsymbol{f}} = (\boldsymbol{I} + k\gamma \boldsymbol{S})^{-1} (\tilde{\boldsymbol{y}} + \mu \boldsymbol{1}). \tag{2.5}$$

Here, μ is chosen so that the resulting vector $\boldsymbol{f}$ is orthogonal to $\boldsymbol{1}$. Denote by $s(\boldsymbol{f})$ the functional

$$s : \boldsymbol{f} \to \sum_i f_i. \tag{2.6}$$

Since s is linear, we obtain

$$0 = s\left(\tilde{\boldsymbol{f}}\right) = s\left((\boldsymbol{I} + k\gamma \boldsymbol{S})^{-1} \tilde{\boldsymbol{y}}\right) + s\left((\boldsymbol{I} + k\gamma \boldsymbol{S})^{-1} \boldsymbol{1}\right). \tag{2.7}$$

Therefore we can write

$$\mu = -\frac{s\left((\boldsymbol{I} + k\gamma \boldsymbol{S})^{-1} \tilde{\boldsymbol{y}}\right)}{s\left((\boldsymbol{I} + k\gamma \boldsymbol{S})^{-1} \boldsymbol{1}\right)} \tag{2.8}$$

Algorithm 2.3 Tickhonov Regularization with Parameter $\gamma \in \mathbb{R}$

The objective is to minimize the square loss function plus the smoothness penalty.
$\tilde{\boldsymbol{f}} = \arg\min_{\substack{\boldsymbol{f}=(f_1, \cdots, f_n) \\ \sum f_i = 0}} \frac{1}{k}\sum_i (f_i - \tilde{y}_i)^2 + \gamma \boldsymbol{f}^T \boldsymbol{S} \boldsymbol{f}$

In the algorithm 2.4, $\boldsymbol{S}$ is a smoothness matrix, e.g. $S = L$ or $S = L^p$, $p \in \mathbb{N}$. However, here we are not allowing multiple vertices in the sample. We partition $\boldsymbol{S}$ as

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{S}_2 \\ \boldsymbol{S}_2^T & \boldsymbol{S}_3 \end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{S}_1$ is a $k \times k$ matrix, $\boldsymbol{S}_2$ is a $k \times (n-k)$ matrix, and $\boldsymbol{S}_3$ is a $(n-k) \times (n-k)$ matrix. Let $\tilde{\boldsymbol{f}}$ be the values of $\boldsymbol{f}$, where the function is unknown, $\tilde{\boldsymbol{f}} = (f_{k+1}, \cdots, f_n)$. By applying the similar Lagrangian

multiplier method, we have

$$\tilde{\boldsymbol{f}} = \boldsymbol{S}_3^{-1}\boldsymbol{S}_2^T\left(\left(\tilde{y}_1, \cdots, \tilde{y}_k\right)^T + \mu\mathbf{1}\right), \tag{2.10}$$

and

$$\mu = -\frac{s\left(\boldsymbol{S}_3^{-1}\boldsymbol{S}_2^T\tilde{\boldsymbol{y}}\right)}{s\left(\boldsymbol{S}_3^{-1}\boldsymbol{S}_2^T\mathbf{1}\right)} \tag{2.11}$$

Algorithm 2.4 Interpolated Regularization without Parameter

Here we assume that the values $y_1, y_2, \cdots, y_k$ have no noise.
Thus the optimization problem is to find a function of maximum smoothness satisfying $f(\boldsymbol{x}_i) = \tilde{y}_i$, $1 \leq i \leq k$
$\tilde{\boldsymbol{f}} = \arg\min_{\substack{\boldsymbol{f}=\left(\tilde{y}_1, \tilde{y}_2, \cdots, \tilde{y_k}, f_{k+1}, \cdots, f_n\right) \\ \sum f_i = 0}} \boldsymbol{f}^T\boldsymbol{S}\boldsymbol{f}$

We now investigate generalization bounds for this graph semi-supervised learning regularization through algorithmic stability. Belkin, Matveeva and Niyogi (2004) defined the the empirical error $R_k(f)$, which is a measure of how well we do on the training set, and the generalization error $R(f)$, which is the expectation of how well we do on all labelled or unlabelled points, in the following way

$$\begin{aligned} R_k(f) &= \frac{1}{k}\sum_1^k \left(f(\boldsymbol{x}_i) - y_i\right)^2 \\ R(f) &= \mathbb{E}_\mu\left[\left(f(\boldsymbol{x}) - y(\boldsymbol{x})\right)^2\right], \end{aligned} \tag{2.12}$$

where f_T maps a given set of examples T to $\mathbb{R}$, i.e. $f_T : V \to \mathbb{R}$, the expectation is taken over an underlying distribution μ. In theorem 5, they showed that for data samples of size $k \geq 4$ with multiplicity of at most t, γ-regularization using the smoothness functional S is a $\left(\frac{3M\sqrt{tk}}{(k\gamma\lambda_1 - t)^2} + \frac{4M}{k\gamma\lambda_1 - t}\right)$-stable algorithm, assuming that the denominator $k\gamma\lambda_1 - t$ is positive. Bousquet and Elisseeff (2001) showed that for a β-stable algorithm $T \to f_T$ we have

$$\mathbb{P}\left(\left|R_k(f_T) - R(f_T)\right| > \epsilon + \beta\right) \leq 2\exp\left(-\frac{k\epsilon^2}{2\left(k\beta + K + M\right)^2}\right), \qquad \forall\epsilon > 0. \tag{2.13}$$

Therefore, by following the derivation in the theorem 11.1 in Mohri, Rostamizadeh and Talwalkar (2012), we have with probability $1 - \delta$

$$\left|R_k(f_T) - R(f_T)\right| < \epsilon + \beta \tag{2.14}$$

where $\delta = 2\exp\left(-\frac{k\epsilon^2}{2(k\beta+K+M)^2}\right)$. We then solve for ϵ in this expression for δ and get

$$\epsilon = \sqrt{\frac{2\ln\frac{2}{\delta}}{k}}\left(k\beta + K + M\right), \tag{2.15}$$

then plug into inequality 2.14 and rearrange terms, then, with probability $1 - \delta$, we have

$$\left|R_k(f_T) - R(f_T)\right| \leq \beta + \sqrt{\frac{2\ln\frac{2}{\delta}}{k}}\left(k\beta + K + M\right), \tag{2.16}$$

where

$$\beta = \left(\frac{3M\sqrt{tk}}{(k\gamma\lambda_1 - t)^2} + \frac{4M}{k\gamma\lambda_1 - t}\right). \tag{2.17}$$

This is the theorem 1 of Belkin, Matveeva and Niyogi (2004).

3 Projection Algorithm for Graph Online Learning

A different approach was proposed by Herbster, Pontil and Wainer (2005) using projection and kernel methods. The method uses minimal norm interpolation, which I will later explain can be thought of as finding

the distribution over the vertices that minimizes the total flux induced over the graph, under the constraint that the labelled points are approximately correct.

3.1 Pseudo-Inverse of Graph Laplacian and Kernel of Hilbert

A key object in this approach is the pseudo-inverse of the graph Laplacian $\boldsymbol{L}^+$ can be thought of as the reproducing kernel of a particular Hilbert space H of real-valued functions over the vertices of the graph

$$\boldsymbol{f} : V \to \mathbb{R}^n \tag{3.1}$$

equipped with the inner product

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{g}. \tag{3.2}$$

Over the entire Vector space of real-valued functions on the graph, $\langle \boldsymbol{f}, \boldsymbol{g} \rangle$ is only a semi-inner product, and therefore induces the semi-norm $\|\boldsymbol{g}\|$. If $\boldsymbol{f}^*$ is an eigenvector of $\boldsymbol{L}$ corresponding to an eigenvalue of 0, then $\langle \boldsymbol{f}^*, \boldsymbol{f}^* \rangle = 0$. These eigenvectors with eigenvalues of zero are precisely the eigenvectors that are piecewise constant. We can see this by noting that

$$\|\boldsymbol{g}\|^2 = \sum_{(i,j) \in E(G)} (\boldsymbol{g}_i - \boldsymbol{g}_j)^2 \quad . \tag{3.3}$$

This is because

$$\begin{aligned}
\|\boldsymbol{g}\|^2 &= \langle \boldsymbol{g}, \boldsymbol{g} \rangle \\
&= \sum_{i,j=1}^{n} \boldsymbol{g}_i \boldsymbol{L}_{ij} \boldsymbol{g}_j \\
&= \sum_{i=j}^{n} \boldsymbol{g}_i \boldsymbol{L}_{ij} \boldsymbol{g}_j + \sum_{i \neq j}^{n} \boldsymbol{g}_i \boldsymbol{L}_{ij} \boldsymbol{g}_j \\
&= \sum_{i=1}^{n} \boldsymbol{g}_i^2 \boldsymbol{D}_{ii} + \sum_{i \neq j}^{n} \boldsymbol{g}_i \boldsymbol{A}_{ij} \boldsymbol{g}_j \\
&= \sum_{i,j=1}^{n} \boldsymbol{g}_i{}^2 \boldsymbol{A}_{ij} + \sum_{i \neq j}^{n} \boldsymbol{g}_i \boldsymbol{L}_{ij} \boldsymbol{g}_j \\
&= \sum_{i,j \in E(G)}^{n} \boldsymbol{g}_i^2 - 2 \boldsymbol{g}_i \boldsymbol{g}_j \\
&= \sum_{(i,j) \in E(G)} (\boldsymbol{g}_i - \boldsymbol{g}_j)^2.
\end{aligned} \tag{3.4}$$

Please notice that the diagonal component of $\boldsymbol{L}$ is $\boldsymbol{D}$, the off-diagonal component of $\boldsymbol{L}$ is $-\boldsymbol{A}$, $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{A}_{ij}$, and in the proof we pick up a factor of 2 from the second summation since E is the set of unordered pairs in the edge set, and therefore each edge gets counted twice.

As we can see, the sum vanishes if $\boldsymbol{g}_i = \boldsymbol{g}_j$ for all vertices connected by an edge, i.e. when g is piecewise constant on each connected component. Therefore we must restrict our functions to be in the subspace of H spanned by the eigenvectors that have non-zero eigenvalues. Restricted to this subspace, $\langle \boldsymbol{f}, \boldsymbol{g} \rangle$ is indeed an inner product, and therefore induces a true norm $\|\boldsymbol{g}\|$. Since H is a Hilbert space, we can use the Riesz Representation theorem to conclude that the evaluation functional of $\boldsymbol{f}$ at any vertex, $\boldsymbol{f}(v_i)$, being a linear functional $(\boldsymbol{f} + \boldsymbol{g})(v_i) = (\boldsymbol{f} + \boldsymbol{g})_i(v) = \boldsymbol{f}_i(v) + \boldsymbol{g}_i(v) = \boldsymbol{f}(v_i) + \boldsymbol{g}(v_i)$, can be represented as an inner product

$$\boldsymbol{f}(v_i) = \langle \boldsymbol{K}_i, \boldsymbol{f} \rangle = \boldsymbol{K}_i \boldsymbol{L} \boldsymbol{f}. \tag{3.5}$$

The pseudo-inverse $\boldsymbol{L}^+$ satisfies this property, which can be verified by noting that

$$\boldsymbol{f} = \boldsymbol{f}_i = \boldsymbol{L}_i^+ \boldsymbol{L} \boldsymbol{f}_i. \tag{3.6}$$

Thus $\boldsymbol{L}^+$ is the reproducing kernel of H.

3.2 Graph Laplacian and Heat Equation

We can get more intuition about the basic objects such as $\|\boldsymbol{g}\|$ and $\boldsymbol{L}^+$ by using the concept of conductance (it is no coincidence that the continuous version of the Laplacian appears in the heat equation). If we think of f as some distribution over the nodes, we can think of $\boldsymbol{Lf}$ roughly as the flux induced at each of the nodes by that distribution over the graph. Then the form $\boldsymbol{g}^T\boldsymbol{Lf}$ represents some weighted measurement of this flux with the weights given by $\boldsymbol{g}$. For example, if f is a binary vector representing the boundary of some open set on the graph $\langle \boldsymbol{f}, \boldsymbol{g} \rangle$ would be the flux produced by $\boldsymbol{g}$ as measured on the boundary represented by $\boldsymbol{f}$. We can even use this formalism to quickly prove a version of Stokes' Theorem on graphs.

Bibliography

Bapat, Ravindra B. (2014). *Graphs and Matrices*. London, UK: Springer-Verlag. 193 pp. ISBN: 9781447165682 (cit. on p. 1).

Belkin, Mikhail, Irina Matveeva and Partha Niyogi (2004). *Regularization and Semi-supervised Learning on Large Graphs*. Ed. by John Shawe-Taylor and Yoram Singer. Learning Theory 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004. Proceedings. Berlin, Deutschland: Springer Science+Business Media. 654 pp. ISBN: 9783540222828 (cit. on pp. 7, 8).

Bengio, Yoshua, Olivier Delalleau and Nicolas Le Roux (2006). *Semi-Supervised Learning*. Cambridge, Massachusetts, USA: The MIT Press. 528 pp. ISBN: 9780262033589 (cit. on p. 6).

Bousquet, Olivier and André Elisseeff (2001). *Algorithmic Stability and Generalization Performance*. Ed. by Todd K. Leen, Thomas G. Dietterich and Volker Tresp. Advances in Neural Information Processing Systems 13 Proceedings of the 2000 Conference. Cambridge, MA, USA: The MIT Press. 1128 pp. ISBN: 9780262122412 (cit. on p. 8).

Herbster, Mark, Massimiliano Pontil and Lisa Wainer (2005). *Online Learning over Graphs*. Ed. by Luc De Raedt and Stefan Wrobel. ICML '05 Proceedings of the 22nd International Conference on Machine Learning. New York, NY, USA: Association for Computing Machinery. 1113 pp. ISBN: 1595931805 (cit. on p. 8).

Mohri, Mehryar, Afshin Rostamizadeh and Ameet Talwalkar (2012). *Foundations of Machine Learning*. Cambridge, Massachusetts, USA: The MIT Press. 432 pp. ISBN: 9780262018258 (cit. on p. 8).

Zhou, Dengyong, Olivier Bousquet, Thomas Navin Lal, Jason Weston and Bernhard Schölkopf (2004). *Learning with Local and Global Consistency*. Ed. by Sebastian Thrun, Lawrence K. Saul and Bernhard Schölkopf. Advances in Neural Information Processing Systems 16 Proceedings of the 2003 Conference. Cambridge, Massachusetts, USA: The MIT Press. 1728 pp. ISBN: 9780262201520 (cit. on p. 6).

Zhu, Xiaojin and Zoubin Ghahramani (2002). "Learning from Labelled and Unlabelled Data with Label Propagation". In: *Technical Report CMU-CALD-02-107, Carnegie Mellon University*.