Let there be \( l \) labeled points \((x_1, y_1) \ldots (x_l, y_l)\), and \( u \) unlabeled points \( x_{l+1} \ldots x_{l+u} \). Let \( n = l + u \). Assume \( y \in \{0, 1\} \). Assume we are given a \( n \times n \) symmetric weight matrix \( W \) between all points. Consider the graph \( G \) with \( n \) nodes and weights \( W \) on edges. Nodes \( 1 \ldots l \) are labeled as \( y_1 \ldots y_l \), and the task is to assign labels to nodes \( l + 1 \ldots n \).

\section{Label propagation}

Our strategy is to first compute a real function \( f \) on \( G \) with certain nice properties, and then assign labels based on \( f \). Intuitively we want close points to have similar labels. One way is to let the labels \( y_1 \ldots y_l \) propagate through the graph structure \( G \). In the continuous case, \( G \) becomes the data manifold and propagation becomes diffusion. We do require, however, that labeled data remain fixed during the process. Let \( D \) be the diagonal matrix s.t. \( D_{ii} = \sum_{j=1}^{n} w_{ij} \). \( D_{ii} \) is known as the volume of node \( i \). Define a probabilistic transition matrix \( P = D^{-1}W \). We have the following algorithm.

\textit{Algorithm (Label propagation)} Repeat until convergence: 1. Clamp \( f(i) = y_i \), \( i = 1 \ldots l \). 2. Propagate \( f = Pf \).

We showed in \cite{1} that the label propagation algorithm converges to a simple solution regardless of the initial \( f \). We split \( W \) matrix into 4 blocks after the \( l \)th row / column (and \( D, P \) similarly):

\[
W = \begin{bmatrix}
W_{ll} & W_{lu} \\
W_{ul} & W_{uu}
\end{bmatrix}
\]  
\( (1) \)

Let \( f = \begin{bmatrix} f_l \\ f_u \end{bmatrix} \) where \( f_l \) are clamped to the labeled data and \( f_u \) are on unlabeled points. Then the solution is

\[
f_u = (I - P_{uu})^{-1}P_{ul}f_l \]  
\( (2) \)

\[
= (I - D_{uu}^{-1}W_{uu})^{-1}D_{uu}^{-1}W_{ul}f_l \]  
\( (3) \)

\[
= (D_{uu} - W_{uu})^{-1}W_{uu}f_l \]  
\( (4) \)

The algorithm is closely related to the random walk work by Szummer and Jaakkola \cite{2}, with two major differences: 1. we fix the labeled points; 2. our solution is an equilibrium state while in \cite{2} it is dynamic w.r.t a time parameter \( t \). We will come back to this point when discussing heat kernels later.

\footnote{The transition matrix was slightly different in \cite{1} but it doesn’t affect the analysis}
2 Random walk and electric networks

Solution of this form is not new. It is not hard to see for any unlabeled node \( i \), \( f(i) \) is the weighted average of all its neighboring \( f \) values. Such \( f \) is called the harmonic function over \( G \) with Dirichlet boundary condition \( f_\pi \), and uniquely exists. Doyle and Snell discussed it in [3]. Using it for semi-supervised learning, as far as we know, has not been proposed before.

There is a random walk interpretation to the solution. Imagine a particle walking on \( G \). If it is on an unlabeled node \( i \), it will walk to a node \( j \) with probability \( P_{ij} \) after one step. Labeled nodes are absorbing: once the particle reaches a labeled node, it stays there forever. Then \( f(i) \) is the probability that the particle, starting from node \( i \), reaches an absorbing state with label 1 before reaching an absorbing state with label 0.

An electric networks interpretation is also given in [3]. Imagine the edges of \( G \) being resistors with conductance \( W \). We connect \( y = 1 \) labeled nodes to +1 voltage source, and \( y = 0 \) labeled nodes to the ground. Then \( f_u \) is the voltage on unlabeled nodes. Furthermore \( f_u \) minimizes the energy dissipation of the electric network \( G \), for the given \( f_i \):

\[
\sum_{i,j} \frac{1}{2} (f(i) - f(j))^2 w_{ij}
\]  

3 Integrated heat kernel

(hmm... in general \( L = D - W \) has no invert... what’s wrong?)

The ‘integrated heat kernel’ is an integration of heat kernels on unlabeled points. The weight matrix on unlabeled points is \( W_{UU} \). Let \( D_U \) be the \( U \times U \) diagonal matrix s.t. \( D_{ii} = \Sigma_{j=L+1}^n W_{ij} \). Note unlike \( D_{UU+} \), \( D_U \) does not include weights to labeled points.

The graph Laplacian on unlabeled data is \( L = D_U - W_{UU} \). The standard heat kernel is [4]

\[
K_t = e^{-tL}
\]  

The integrated heat kernel is

\[
K = \int_0^\infty K_t dt
\]  

\[
= \int_0^\infty e^{-tL} dt
\]
\[ f_x \propto \sum_{z \in \mathcal{U}} \sum_{y \in \mathcal{L}} K(x, z) P(z \rightarrow y) f_y \]

\[ \propto \sum_{z \in \mathcal{U}} \sum_{y \in \mathcal{L}} K(x, z) \frac{W_{zy}}{D_{zz}} f_y \]

### 4 Graph Laplacian and spectral clustering

Remember the solution \( f \) minimizes the energy (5) given the constraint \( f_t \). The well studied spectral graph segmentation problem minimizes the same energy function without the constraint. Belkin and Niyogi [5] showed that the energy function

\[ \sum_{i,j} \frac{1}{2} (f(i) - f(j))^2 w_{ij} = \langle f, L f \rangle \]

where \( L = D - W \) is the graph Laplacian of \( G \). When \( f \) is not constrained (except normalized), the energy is minimized by the minimum eigenvalue solution to the generalized eigenvalue problem \( L f = \lambda D f \). It can be shown that \( f = 1 \) is a trivial solution with minimum eigenvalue \( \lambda = 0 \). This corresponds to an electric network with the same voltage on all nodes, thus no energy dissipation. The next interesting solution is the eigenvector corresponding to the second smallest (non zero) eigenvalue \( \lambda \). This is exact the normalized cut algorithm by Shi and Malik [6]. In addition when \( W \) is close to block diagonal, it can be shown that the image of data points are tightly clustered in the eigenspace spanned by the first few eigenvectors of \( L \) [7] [8]. This leads to spectral clustering algorithms.

In semi-supervised learning, we have labeled data \( f_t \). A naive algorithm is to cluster the whole data set first with spectral clustering, then assign labels to clusters. This works well when \( W \) is block diagonal, but we believe the constrained solution \( f \) to be more robust when \( W \) is less well formed.
A related algorithm is proposed by Belkin and Niyogi in [9]. Note the normalized eigenvectors of $L$ form a basis on $G$. The energy (or ‘smoothness’ as it is called in [9]) of any normalized eigenvector is its eigenvalue. Belkin et al. proposed to learn a function (called $g$ here) on $G$ that is regularized for small energy and aims to fit the labeled data. In particular, they pick $p$ normalized eigenvectors of $L$ with the smallest eigenvalues, and $g$ is the linear combination of the $p$ eigenvectors which best fits $f_i$ in least squares sense. We remark that $f$ by definition fits labeled data exactly, while $g$ may not; $f$ is the minimum energy solution, while $g$, although made up of low energy eigenvector basis, may be far from minimum energy.

5 From $f$ function to classification

The seemingly obvious classification method is to assign node $i$ label 1 if $f(i) > 0.5$, and label 0 otherwise. This works well when classes are well separated. However in real datasets classes are often not ideally separated, and this 0.5-thresholding tends to produce severely unbalanced classes. We experiment with three ways to control class balance:

- Constraining the mean of $f$
- Add two virtual nodes to $G$
- Heuristic class mass normalization

References


