A Brief Survey on Semi-supervised Learning with Graph Regularization

Hongyuan You
Department of Computer Science
University of California, Santa Barbara
hyou@cs.ucsb.edu

Abstract

In this survey, we go over a few historical literatures on semi-supervised learning problems which apply graph regularization on both labled and unlabeled data to improve classification performance. These semi-supervised methods usually construct a nearest neighbour graph on instance space under certain measure function, and then work under the smoothness assumption that class labels of samples change quite slowly for connected instances on the graph. Graph Lapalcian of the graph has been used in the graph smoothness regularization since it is a discrete analogous to Beltrami Laplace smoothness operator in the continuous space. A relationship between kernel method and graph Laplacian has also been discovered, and suggests an approach to design specific kernel for classification utilizing the spectrum of Laplacian matrix.

1 Smoothness Regularization via Laplace Operator and Graph Laplacian

1.1 Semi-supervised Learning Problem

Nowadays there are many practical applications of machine learning problem. Though in academic papers we often assume these problems have some nice conditions and huge number of samples for inference and learning, we cannot achieve such datasets in reality. One of the biggest problem is that not all of sample instances have class labels since people can hardly label millions of instances manually. Then we need some learning method that can deal with the problem with both labeld and unlabled data at the same time, utilizing the labels of labeled instances and also the information from unlabeled instances. Let X be the instance space, $\{x_1, x_2, ..., x_k\}$ are the k labeled training samples with labels $\{y_1, y_2, ..., y_k\}$, $y_i \in [-1, +1]$, and $\{x_{k+1}, x_{k+2}, ..., x_n\}$ are the $n-k$ unlabeled samples. Usually we have $k \ll n$ which means only very few samples are labeled. Our purpose is learning a classification function $f : x \to [-1, +1]$ to predict the class of testing samples. We restrict ourselves to the *transductive setting* where the unlabelled data also serve as the test data in our performance analysis.

1.2 Manifold Assumption and Laplace Operator

The information from unlabeled instance is hard to be incorporated into the classification model unless we could raise a reasonable assumption of the unlabeld data. The first assumption is that*the data lies on a low-dimensional manifold within a high-dimensional representation space*. We can easily find many examples to justify this assumption. For example a handwriting digital number can be represented as a matrix of image pixel with very high dimension, however it can also be represented by only a few parameters in the low-dimensional ambient space, and in this space, the structure of similar handwriting digital numbers lie close to each other to form a low-dimensional manifold. Similar examples can be find in document categorization problems, where people discov-

ered documents represented by word identifier vector (remark the counts or frequency) should be in a manifold no matter how high dimension the original documents are. Realizing the existence of the intrinsic structure of the instance space is significant for developing classification methods for such kind of problems.

On such a low-dimensional manifold, we should have the second assumption, which we may already suggest above, that *instances have a small distance on the manifold should have the same labels*. Under this assumption, we are able to use these unlabeled instances as long as they are close to some labeled training samples.

Under the manifold setting, we can use the Laplace Operator for regularization. Let the Laplace operator $\Delta = -\Sigma_i \partial^2 / \partial x_i^2$, and let f be a $\mathcal{L}^2$ function defined on an n-dimensional compact Riemann manifold $\mathcal{M}$. It has been know that, the spectrum of Δ is discrete ($\lambda_0 = 0$) and its eigenfunctions $e_i(\cdot)$ form a basis of $\mathcal{L}^2(\mathcal{M})$, which means any f can be represented as a linear combination of $e_i(\cdot)$:

$$f(x) = \sum_{i=0}^{\infty} \alpha_i e_i(x) \tag{1}$$

where $\Delta e_i(\cdot) = \lambda_i e_i(\cdot)$. Using Laplace operator we can have a natural measure of smoothness for any function f on $\mathcal{M}$, which computes the variation of function values:

$$S(f) := \int_{\mathcal{M}} |\nabla f|^2 d\mu = \int_{\mathcal{M}} f \Delta f d\mu = \langle \Delta f, f \rangle_{\mathcal{L}^2(\mathcal{M})} \tag{2}$$

And the smoothness of eigenfunctions:

$$S(e_i(\cdot)) = \langle \Delta e_i(\cdot), e_i(\cdot) \rangle_{\mathcal{L}^2(\mathcal{M})} = \lambda_i \tag{3}$$

Then we can easily compute the smoothness by eigenvalue λ_i's and decomposition factor α_i's:

$$S(f(\cdot)) = \langle \Delta f, f \rangle_{\mathcal{L}^2(\mathcal{M})} = \langle \sum_{i=0}^{\infty} \alpha_i \Delta e_i(\cdot), \sum_{i=0}^{\infty} \alpha_i e_i(\cdot) \rangle_{\mathcal{L}^2(\mathcal{M})} = \sum_{i=0}^{\infty} \lambda_i \alpha_i^2 \tag{4}$$

The last equation tells us, the eigenfunction with large eigenvalues will lead to a high smoothness value and differs a lot on different instances that consists an oscillation component of the original function f. However the eigenfunction with low eigenvalue has a smooth pattern under the measure.

For the learning problem, we can make f be the classifier we want. Then if we want close instances on the low-dimensional manifold have similar classification results, then the smoothness measure $S(f)$ should be small enough. So $S(f)$ become an ideal regularizer.

1.3 Graph Approximation of Manifold and Graph Laplacian

However how to recover the intrinsic structure or the low-dimensional manifold of the input data is still an open problem remained unsolved. What we need here for semi-supervised learning is the second assumption on relationship between similarity distance and the label. We could construct a nearest neighbour graph G to connect similar instances (though distance on graph differs from distance on manifold), and make the graph to be a good approximation of the low-dimensional manifold of the input instance space.

Any instance x_i should adding t nearest neighbours in the graph G. Let A be the adjacency matrix of G, and $A_{ij} = \omega_{ij} = 1$ if instances x_i and x_j are close under some similarity function, such as angle distance (you can also change values on edges into meaningful weights). People have developed many methods on how to compute the similarity distance that we would omit them here. Till now our assumption could be clear, *Two instances connected by one edge should have similar classification results*.

Under the idea of graph approximation of manifold, we can use the graph Laplacian as a natural analogous of Lapalce operator. Let $L = D - A$ be the Laplcian matrix of the adjacency matrix A of graph G, where $D = diag\{d_1, d_2, ..., d_n\}$ and d_i is the degree of instance x_i in the nearest neighbour graph G. We know that L has its eigensystem $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^n$ and $\{\mathbf{v}_i\}_{i=1}^n$ form an orthonormal basis of $\mathbf{R^n}$. Any vector $\mathbf{f} = \{f_1, f_2, ..., f_n\}^T$ can be represented with a set of basis,

$$\mathbf{f} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \qquad L\mathbf{v}_i = \lambda_i \mathbf{v}_i \tag{5}$$

More important, the natural measure of smoothness could keep the same form when we use L to replace Δ:

$$S_L(f) := \sum_{i,j} \omega_{ij}(f_i - f_j)^2 = \mathbf{f}^T L \mathbf{f} = \langle f, Lf \rangle_G \tag{6}$$

From this smoothness definition, we can notice that if ω_{ij} is nonzero and large value, then a difference of their labels will increase the smoothness measure's value, and a same label will make the item zero, which is exactly what we want in regularization. The smoothness of eigenfunctions are the same as their eigenvalues:

$$S_L(\mathbf{v}_i) = \mathbf{v}_i^T L \mathbf{v}_i = \lambda_i \mathbf{v}_i^T \mathbf{v}_i = \lambda_i \tag{7}$$

Then we can easily compute the smoothness by eigenvalue λ_i's and decomposition factor α_i's:

$$S_L(\mathbf{f}) = \langle L\mathbf{f}, f \rangle_G = \langle \sum_{i=1}^{n} \alpha_i L \mathbf{v}_i, \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \rangle_G = \sum_{i=1}^{n} \lambda_i \alpha_i^2 \tag{8}$$

As we can see these properties are all the same with the continuous version.

2 Learning Optimal Labels for Unlabeled Data

2.1 Classification as Interpolating Problem

If we treat the vector **f**'s entries are the function values of classifier $f(\cdot)$ on some instances, and the model of instance space has the smoothness assumption, then it means **f** can be computed as a combination of the prediction $\mathbf{v}_i$ which are the eigenvectors of Laplacian matrix L. The only thing we need to know is the combination factor α_i's. This early work in the semi-supervised learning problem is first done by Mkihail Belkin and Partha Niyogi[2]. The continuous formulation should be

$$f^* = \arg\inf_f \int_{\mathcal{M}} \left(f(x) - \sum_{j=0}^{p} \alpha_j e_j(x) \right)^2 dx \tag{9}$$

However, its hard to get those eigenfunctions on the manifold $\mathcal{M}$, and we usually could access to a limit amount of labeled instances, thus they set the discrete version of formulation below:

$$\mathbf{f}^* = \arg\inf_{\mathbf{f}} \sum_{i=1}^{k} \left(y_i - \sum_{j=1}^{p} \alpha_j \mathbf{v}_j(i) \right)^2 \tag{10}$$

where $\{\mathbf{v}_i\}_{j=1}^{p}$ is the first p eigenvectors of L corresponding to the smallest p eigenvalues, and $\mathbf{f} = \Sigma_j \alpha_j \mathbf{v}_j$. Although they mentioned the smoothness measure of the Graph Laplacian, they did not explicitly use them in this method but choose the smallest p eigenvectors. Looking back to the formula of $S_L(\mathbf{f})$ we could find that such component selection leads to the top p smooth eigenvectors and ensure the smoothness of the learned prediction results.

One can solve the least square problem as following:

$$\alpha = \left(E^T E\right)^{-1} E^T y \tag{11}$$

where $\alpha = (\alpha_1, \alpha_2, ..., \alpha_p)^T$, $y = (y_1, y_2, ..., y_p)^T$, and

$$E = \begin{pmatrix} \mathbf{v}_1(1) & \mathbf{v}_2(1) & \cdots & \mathbf{v}_p(1) \\ \mathbf{v}_1(2) & \mathbf{v}_2(2) & \cdots & \mathbf{v}_p(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}_1(s) & \mathbf{v}_2(s) & \cdots & \mathbf{v}_p(s) \end{pmatrix} \tag{12}$$

The final classifier for unlabeled data in the dataset is

$$f(x_i) = sgn\left(\sum_{j=1}^{p} \alpha_j \mathbf{v}_j(i) \right) \tag{13}$$

One problem might be the difficulty of approximation the Laplace operator decomposition with graph Laplacian decomposition, because the decomposition of eigenfunctions of f should be much more meaningful and generalizable than the decomposition of eigenvectors of **f**.

2.2 Tikhonov Regularization

The optimization problem of this method simply changes a little bit from above that is to explicitly add the smoothness measure into the objective function. But the theoretical result of interest shows the error bound is controlled with the Fiedler number of the graph, which means generalization ability of graph regularization learning method has some relationship with the geometric invariants of the graph [2].

The setting here is a little different: labeled instances could be randomly chosen without replacement, which means each labeled instance may appear more than once with same or different labels. And we preprocess the labels like

$$\tilde{y} = (y_1 - \bar{y}, y_2 - \bar{y}, ..., y_k - \bar{y}), \qquad \bar{y} = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{14}$$

They try to learn the labels by solving following optimization problem:

$$\mathbf{f}^* = \arg\inf_{\mathbf{1}^T\mathbf{f}=0} \frac{1}{k}\sum_{i=1}^{k}(\tilde{y}_i - \mathbf{f}_i)^2 + \gamma S_L(\mathbf{f}) \tag{15}$$

where S_L is the regularization term and S_L could be replaced by S_{L^p} for $p \in \mathbf{N}$ or $S_{\exp(-tL)}$ for $t > 0$. Let n_i be the number of the multiplicity of x_i, and let y_{ij} be the j-th label of x_i where $j = 1, 2, ..., n_i$. Then we can rewrite the objective function into the quadratic programming problem:

$$\begin{aligned}
&\arg\inf_{\mathbf{1}^T\mathbf{f}=0} \quad \frac{1}{k}\left(\sum_{i=1}^{k} n_i\mathbf{f}_i^2 + \sum_{j=1}^{n_i} y_{ij}^2 - 2\mathbf{f}_i\sum_{j=1}^{n_i} y_{ij}\right) + \gamma\mathbf{f}^T L\mathbf{f} \\
&= \arg\inf_{\mathbf{1}^T\mathbf{f}=0} \quad \frac{1}{k}\left(\mathbf{f}^T I_k\mathbf{f} - 2\tilde{y}^T\mathbf{f}\right) + \gamma\mathbf{f}^T L\mathbf{f} \\
&= \arg\inf_{\mathbf{1}^T\mathbf{f}=0} \quad \mathbf{f}^T\left(1 + k\gamma S\right)\mathbf{f} - 2\tilde{y}^T\mathbf{f}
\end{aligned} \tag{16}$$

By Lagrange multiplier, we set $\nabla_{\mathbf{f}}\mathcal{L}(\mathbf{f}, \mu) = 0$, then get $(I_k + k\gamma S)\mathbf{f} = \tilde{y} + (\mu/2)\mathbf{1}$, redefine $\mu/2 \to \mu$, the optimal solution of label vector $\mathbf{f}$ is

$$\begin{aligned}
\mathbf{f}^* &= \left(1 + k\gamma S\right)^{-1}(\tilde{y} + \mu\mathbf{1}) \\
\mu &= -\frac{\mathbf{1}^T\left(1 + k\gamma S\right)^{-1}\tilde{y}}{\mathbf{1}^T\left(1 + k\gamma S\right)^{-1}\mathbf{1}}
\end{aligned} \tag{17}$$

where the selection of μ makes $\mathbf{1}^T\mathbf{f} = 0$.

By defining the empirical error bound on labeled training samples

$$R_k(\mathbf{f}) = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{f}_i - \tilde{y}_i)^2 \tag{18}$$

and defining the generalization error bound on all labled and unlabeled samples

$$R(\mathbf{f}) = E_{x\sim\mathcal{D}, y\sim\mathbf{R}}(\mathbf{f}(x) - \tilde{y}(x))^2 \tag{19}$$

We can have the following performance bound[2] based on stability performance analysis [3]:

Let γ be the regularization parameter, T be a set of $k > 4$ vertices $x_1, ...x_k$ where each vertex occurs no more than t times, together with values $y_1, ..., y_k$, and $|y_i| \leq M$. Let $\boldsymbol{f}$ be the regularization solution using the smoothness functional L with the second smallest eigenvalue λ_2. Assuming that $|\boldsymbol{f}_i| \leq K$. We have the following bound exists with probability at least $1 - \delta$:

$$|R_k(\boldsymbol{f}) - R(\boldsymbol{f})| \leq \beta + \sqrt{\frac{2log(2/\delta)}{k}}(k\beta + (K + M)) \tag{20}$$

where

$$\beta = \frac{3M\sqrt{tk}}{(k\gamma\lambda_2 - t)^2} + \frac{4M}{k\gamma\lambda_2 - t} \tag{21}$$

Here the Fiedler number λ_2 shows in the generalization error bound. λ_2 gives an estimation of the connectivity of the graph. If λ_2 is small, then it is possible to have a small graph cut, and if λ_2 is large, then the graph is more connected. And we can see from the bound, the larger the λ_2, the better performance on the smoothness regularization, because an over-sparse graph is hard to peform smoothness assumption, and the prediction value of a single instance is influenced only on a very few number of neighbours which may leads to overfitting and worse generalization performance.

One drawback of these two methods is that they can only predict labels of a set of unlabeled instance at one time. In the next time, they have to recompute the minimization problem for a new set of unlabeled data, that is, they do not learn an explicit form of classifier $f(x)$ for the instance which does not show up in the dataset but only the best prediction result $f(x_i)$. To overcome this shortage, we will then look into kernel method and find the relationship between graph Laplaican and the kernel matrix.

3 Kernel Method and Graph Regularization

3.1 Kernel function and kernel matrix

Kernel method is a class of algorithms which contains Support Vector Machine (SVM). The basic idea is utilizing training sample instances to predict testing samples through pairwise similarity. The pairwise similarity matrix is then the kernel matrix K, which the entry $K_{ij} = K(x_i, x_j)$ shows the similarity between x_i and x_j, and $K(\cdot, \cdot)$ here should be a kernel function over $X \times X$, where X be the instance space. Actually, a function $K : X \times X \to \mathbf{R}$ is a kernel function, if:

- (i) K is symmetric, i.e. $K(x, y) = K(y, x)$;
- (ii) K is positive semi-definite, i.e. for any set of $\{x_1, ..., x_n\}$, kernel matrix K (which $K_{ij} = K(x_i, x_j)$) is positive semi-definite.

And for every valid kernel function K, people has proved in Mercer theorem that there is always a function $\phi : X \to V$ which satisfies $K(x, y) = \langle \phi(x), \phi(y) \rangle_V$. It means the pairwise similarity computed by kernel function actually comes from the inner product in some high dimensional feature space V (usually), and ϕ is a kind of feature transformation. This feature-space formulation will helps in the future.

3.2 Repoducing kernel Hilbert spaces

Let us look deeper into the kernel method and reproducing kernel Hilbert space, which is helpful for solving the semi-supervised learning problem using graph regularization. Let X be the instance space, $K : X \times X \to \mathbf{R}$ be a kernel function described above. Let functions $f : X \to \mathbf{R}$ where the classifier should comes from. $\mathcal{H}_K$ is a Hilbert space of function f's with inner product $\langle \cdot, \cdot \rangle_K$. We say that $\mathcal{H}_K$ is a reproducing kernel Hilbert space (RKHS), if the definition of $\mathcal{H}_K$ and $\langle \cdot, \cdot \rangle_K$ satisfy the following two conditions:

- (i) mapping function $\Phi : x \mapsto K(x, \cdot)$, for any $x \in X$, we have $K(x, \cdot) \in \mathcal{H}_K$;
- (ii) $f(x) = \langle f(\cdot), K(x, \cdot) \rangle_K$, for any $f \in \mathcal{H}_K$ and $x \in X$.

The first condition is a map from x to a function $K(x, \cdot)$. The second condition is the *reproducing kernel property* which implies $K(x, y) = \langle K(x, \cdot), K(y, \cdot) \rangle$. Φ maps a space of instances into a space of functions, and makes the function value of f at point x become the inner product of f and $K(x, \cdot)$

Regularization in an RKHS is usally formulated as following minimization problem which will learn a function $f \in \mathcal{H}_K$:

$$f_* = \arg\inf_f E_\gamma(K) = \arg\inf_f \sum_{i=1}^{k} L(f(x_i), y_i) + \gamma \langle f, f \rangle_K \tag{22}$$

where L is the loss function and $\gamma > 0$ is the function regularization parameter. People has known that the minimizer has the following form [10]

$$f(x) = \Sigma_{i=1}^{k} c_i K(x_i, x), \quad x \in X \tag{23}$$

which is a linear combination of the reproducing mapping function $K(x_i, \cdot)$ of training sample instancs x_i. This form helps a lot in our learning problem because it reduces a learning problem on all possible functions into a problem of learning a set of combination factors. Thus it is possible to generalize previous methods by learning a classifier $f(x)$ for all instances x in space X rather than just learning the labels for unlabled data in the training dataset.

3.3 Construct RKHS and Regularization

A specific setting is that

- Let $\mathcal{H}_K$ be the linear combination of the following kind of function:

$$f(\cdot) = \sum_{i=1}^{l} \alpha_i K(x_i, \cdot) \tag{24}$$

  where $l \in \mathbf{N}$, and $\alpha_i \in \mathbf{R}$, K is the valid kernel function, x_i's are labeled or unlabled training samples.

- Let the inner product of $\mathcal{H}_K$ defined as following

$$\langle f(\cdot), g(\cdot) \rangle_K = \langle \sum_i \alpha_i K(x_i, \cdot), \sum_j \alpha_j K(x_j, \cdot) \rangle = \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \tag{25}$$

Under this setting you can easily check that $\mathcal{H}_K$ is a RKHS with kernel K and inner product $\langle \cdot, \cdot \rangle_K$. Then people proved following equivalence [7][8]: If we have a valid kernel K on all the labled and unlabeled instances, the problem of learning labels of unlabeled instances (focusing on labels) below:

$$\mathbf{f}^* = \arg\inf_{\mathbf{f}} \left(\frac{1}{k} \sum_{i=1}^{k} L\left(\tilde{y}_i - \mathbf{f}_i\right) + \gamma \mathbf{f}^T K^{-1} \mathbf{f} \right) \tag{26}$$

is equivalent to the problem

$$f^*(\cdot) = \arg\inf_{f} \left(\frac{1}{k} \sum_{i=1}^{k} L\left(\tilde{y}_i - f(x_i)\right) + \gamma \langle f, f \rangle_K \right) \tag{27}$$

Note that (i) the second problem is the same as SVM in the feature space formulation (we mentioned earlier) and can be solved in dual space efficiently; (ii) the first problem optimizes a label vector and the second problem optimizes a function; (iii) The equivalence means that $f^*(x_i) = \mathbf{f}^* x_i$ for those unlabled instances in the dataset. This equivalence means that we can learn a classifier to predict new instances without more computational cost.

3.4 From Graph Laplacian to Kernel

Now the only problem is how to choose a valid kernel from the graph Laplacian matrix L we know.

Assume that we have the graph Laplacian matrix L over n instances which k of them are labeled training samples. Let us define a semi-norm on $\mathcal{R}^n$ space with L:

$$\langle \mathbf{f}, \mathbf{g} \rangle := \mathbf{f}^T L \mathbf{g}, \quad \mathbf{f}, \mathbf{g} \in \mathcal{R}^n \tag{28}$$

It is a semi-norm because that $\langle \mathbf{f}, \mathbf{f} \rangle = 0$ if $\mathbf{f}$ is a constant vector. However if we define the space $\mathcal{H}_0$ to be the space of real value functions defined on instances in the graph, and let $\mathcal{H}$ be a subspace of $\mathcal{H}_0$ which any function f cannot have the same value on all instances. Then it is clear that $\langle \mathbf{f}, \mathbf{g} \rangle$ is a well defined norm on $\mathcal{H}_0$ [6].

We can claim that the pseudoinverse L^+ is a valid kernel matrix. It is because we know the following equation

$$\mathbf{f}^T L L^+ = L L^+ \mathbf{f} = \mathbf{f} \tag{29}$$

which can be interpreted as following

$$\mathbf{f}_i = \langle \mathbf{f}, L_i^+ \rangle = \mathbf{f}^T L L_i^+ \tag{30}$$

where L_i^+ is the i-th column of $L+$. It shows a *reproducing property* of the inner product defined with L^+ on $\mathcal{H}$. Thus $L+$ is a valid reproducing kernel on $\mathcal{H}$. If the graph Laplacian matrix is invertible, we simply use L^{-1} other than the expression of L^+. And we can use such $K = L^+$ in the minimization problem in the last subsection.

This is a simple justification of design kernel from graph Laplacian matrix. And further, we can hope that the kernel matrix has the following spectral form:

$$K = \sum_{i=1}^{n} \frac{1}{\lambda_i} \mathbf{v}_i \mathbf{v}_i^T \tag{31}$$

where $\mathbf{v}_i$'s are eigenvectors of L.

4 Spectral Transform for Graph Kernel

4.1 Parametric Spectral Transform for Graph Kernel

However, we can further improve our kernel for better performance, where the basic idea is to encourage the spectrum with small eigenvalue and penalize the spectrum with high eigenvalues. Here we add a function $r(\cdot)$ defined on eigenvalues and perform a parametric spectral transform of K_0 to design a better kernel K. Then we have the following form of kernel K.

$$K = \sum_{i=1}^{n} \frac{1}{r(\lambda_i)} \mathbf{v}_i \mathbf{v}_i^T \tag{32}$$

As shown in [4], under such spectral grap kernel framework, many kernels can be designed by transforming the

- Regularized Laplacian Kernel $\quad K = (I + \sigma^2 \tilde{L})^{-1}$

$$r(\lambda) = 1 + \sigma^2 \lambda \tag{33}$$

- Diffusion Process Kernel $\quad K = exp(-\sigma^2/2\tilde{L})$

$$r(\lambda) = exp(\sigma^2/2\lambda) \tag{34}$$

- p-step Random Walk Kernel $\quad K = (\alpha I - \tilde{L})^p, \quad \alpha \geq 2$

$$r(\lambda) = exp(\sigma^2/2\lambda) \tag{35}$$

- Inverse Cosine Kernel $\quad K = cos(\tilde{L}\pi/4)$

$$r(\lambda) = exp(\sigma^2/2\lambda) \tag{36}$$

4.2 Nonparametric Spectral Transform for Graph Kernel

Many methods use the framework above to design kernels from graph Laplacian matrix L. But the spectral transformation does not address question of which parameter family to use for r. The number of degrees of freedom of r (usually one or two) might be too few for modelling the learning problem and leaves the kernel overly constrained. Then people are trying to optimize a nonparametric spectral transformation which only applies the ordering constraint (see the last constraint) [5].

$$\begin{aligned} \arg\max_{\mu} \quad & \widehat{A}(K_k, T) = \frac{\langle K_k, T \rangle_F}{\sqrt{\langle K_k, K_k \rangle_F \langle T, T \rangle_F}} \\ subject \quad to \quad & K = \Sigma_{i=1}^{n} \mu_i K_i \\ & \mu_i \geq 0 \\ & trace(K) = 1 \\ & \mu_i \geq \mu_{i+1}, i = 2, .., n \end{aligned} \tag{37}$$

The objective function is called the empirical kernel alignment which evaluates the fitness of a kernel K to the labels of labled training samples. It has many good properties which have been shown in [9]. $\langle \cdot, \cdot \rangle_F$ is the Frobenius product, defined as $\langle M, N \rangle_F = \Sigma_{i,j} = M_{ij} N(ij)$. K_k in the objective function is the kernel matrix K restricted in the first k labeled training samples, and T is the outer product of y, that is, $T = yy^T$. And $\mu = (\mu_1, \mu_2, ..., \mu_n)^T$ is the combination factor of kernel's spectrum.

Such an ordering constraint is reasonable. As we mentioned before, considering the smoothness measure S_G, the smaller the eigenvalue λ_i, the smoother its corresponding eigenvector $\mathbf{v}_i$ over the graph G. Then the ordering constraint maintain a decreasing significance order of the transformed spectral vector of the kernel, which implies it encourages smooth functions in the end. Note that we do not constrain the constant vector $\mathbf{v}_1$ with the zero eigenvalue λ_1 (assume connected graph G), which do make sense for it is simply a bias. Constraint $trace(K) = 1$ are both trying to fix the scale invariance of the kernel alignment.

The optimization problem above has an equivalent form:

$$\begin{aligned} &\arg\max_{\mu} \langle K_k, T \rangle_F \\ &subject \quad to \quad \langle K_k, K_k \rangle_F \leq 1 \\ &\qquad\qquad\quad \mu_i \geq 0 \\ &\qquad\qquad\quad \mu_i \geq \mu_{i+1}, i = 2, .., n \end{aligned} \tag{38}$$

Under such subtle configuration of the optimization problem, it can be further transformed into an equivalent quadratically constrained quadratic programming problem (QCQP) below which has feasible and fast optimization method.

$$\begin{aligned} \arg\max_{\mu} \quad & vec(T)^T M \mu \\ subject \quad to \quad & \|M\mu\| \leq 1 \\ & \mu_i \geq 0 \\ & \mu_i \geq \mu_{i+1}, i = 2, .., n \end{aligned} \tag{39}$$

References

[1] Belkin, M., & Niyogi, P. (2002). Semi-supervised learning on manifolds. *Advances in Neural Information Processing Systems (NIPS)*.

[2] Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. *Learning theory*, 624-638. Springer Berlin Heidelberg.

[3] Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *The Journal of Machine Learning Research, 2*, 499-526.

[4] Smola, A. J., & Kondor, R. (2003). Kernels and regularization on graphs. *In Learning theory and kernel machines* (pp. 144-158). Springer Berlin Heidelberg.

[5] Zhu, X., Kandola, J. S., Ghahramani, Z., & Lafferty, J. D. (2004). Nonparametric Transforms of Graph Kernels for Semi-Supervised Learning. *Advances in Neural Information Processing Systems (NIPS), Vol. 17*, 1641-1648.

[6] Argyriou, A., Herbster, M., & Pontil, M. (2005, August). Combining graph Laplacians for semi-supervised learning. *Advances in Neural Information Processing Systems (NIPS)*.

[7] Johnson, R., & Zhang, T. (2008). Graph-based semi-supervised learning and spectral kernel design. *Information Theory, IEEE Transactions on, 54(1)*, 275-288.

[8] Zhang, T., & Ando, R. (2006). Analysis of spectral kernel design based semi-supervised learning. *Advances in neural information processing systems, 18*, 1601.

[9] Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. (2001, December). On kernel target alignment. *Advances in neural information processing systems, 18, Vol. 2*, 4.

[10] Herbster, M., Pontil, M., & Wainer, L. (2005, August). Online learning over graphs. *Proceedings of the 22nd international conference on Machine learning*, 305-312.