1 Diffusion Kernels

Risi Kondor
Jean-Philippe Vert

Graphs are one of the simplest types of objects in mathematics. In chapter **??** we saw how to construct kernels between graphs, that is, when the individual examples $\mathbf{x} \in \mathcal{X}$ are graphs. In this chapter we consider the case when the input space $\mathcal{X}$ is itself a graph and the examples are vertices in this graph.

Such a case may arise naturally in diverse contexts. We may be faced with a network, trying to extrapolate known values of some quantity at specific nodes to other nodes. An illustrative example might be the graph of interactions between the proteins of an organism, which can be built from recent high-throughput technologies. Let's say we are trying to learn the localization of proteins in the cell, or their functions. In the absence of other knowledge, a reasonable starting point is to assume that proteins that can interact are likely to have similar localization or functions. Other examples of naturally occurring networks include metabolic and signaling pathways, and also social networks, the World Wide Web, and citation networks.

In other cases we might be faced with something much more intricate that is not itself a network, but can conveniently be modeled as one. Assume we are interested in predicting the physical properties of organic molecules. Clearly, the set of all known organic molecules is very large and it is next to impossible to impose a global metric on it or sensibly fit it into a vector space. On the other hand, it is not so difficult to come up with rules for which molecules are expected to be similar. The saturation of a single bond or the addition of an extra link to a long carbon chain is unlikely to dramatically change the global properties of a molecule. We can ask a human expert to supply us with a set of empirical rules from which we can build a similarity graph and treat our domain as if it were a network.

The challenge is to build learning algorithms that can exploit such graphical structures. The modularity of kernel-based learning suggests that information about the graph should be incorporated in the kernel. Once we have fabricated a good kernel for the graph, we can plug it into our favorite algorithm (support vector machine [SVM], kernel regression, kernel principal component analysis [KPCA], etc.) and expect the algorithm to perform similarly to how it performs in more conventional settings.

The function of the kernel is to provide a *global* similarity metric, whereas graphs incorporate information on *local* similarity. The kernel must be able to express the degree of similarity between any two examples $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ with fine distinctions in the degree to which $\mathbf{x}$ and $\mathbf{x}'$ are distant from each other in the graph. In contrast, the graph itself only expresses whether $\mathbf{x}$ and $\mathbf{x}'$ are neighbors or not. In section 1.1 we show how the physical process of diffusion suggests a natural way of constructing a kernel from such *local* information, and section 1.2 discusses how to compute the diffusion kernel in specific cases. In section 1.3 we highlight the interpretation of the diffusion kernel in the context of regularization operators.

In section 1.4 we then apply these ideas to the network of chemical pathways in the cell and show how this can boost microarray analysis. Finally, section 1.5 recasts the central ideas of this chapter in a slightly more abstract form and provides an outlook on their role in generalizing kernel-based learning to not just graphs but a wide variety of mathematical objects, from finite groups to Riemannian manifolds.

1.1 Random Walks and Diffusion

The role of the kernel k is to provide a similarity metric on the input space $\mathcal{X}$. Let $\mathcal{X}$ be the vertices, labeled from 1 to n, of an unirected graph G. For now, we assume that G is unweighted, that is, any two vertices i and j in G are either neighbors, denoted $i \sim j$, or they are not neighbors, denoted $i \not\sim j$.

Positive definiteness

A kernel must also satisfy the mathematical requirements of being symmetric and positive definite. Recall that positive definiteness means that for any choice of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathcal{X}$ and any choice of coefficients $c_1, c_2, \ldots, c_m \in \mathbb{R}$,

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0,$$

this being the crucial condition for the existence of the feature mapping $\Phi : \mathcal{X} \mapsto \mathcal{F}$ satisfying $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$. For finite graphs, the kernel can equivalently be specified by an $n \times n$ matrix K, with $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Since k and K are essentially the same object, we shall refer to both as "the kernel" and alternate between the two notations depending on whether we want to emphasize the functional or the matrix aspect.

Shortest-path distance

The simplest measure of similarity on G is the shortest-path distance $d(i, j)$, but it is not clear how to construct a positive definite function from d. Furthermore, d is extremely sensitive to the insertion/deletion of individual edges. In many potential applications, the connectivity information is itself derived from data, and as such is subject to noise. A more robust similarlity measure, perhaps involving averaging over many paths, is more desirable.

Random walks

This leads to the idea of random walks. Recall that a random walk is a process generating paths $z_0 z_1 z_2 \dots z_T$. Starting from a given vertex z_0, at each timestep $t = 1, 2, \dots, T$ the new vertex z_t is chosen at random from among the neighbors of z_{t-1}, each neighbor having the same probability of being selected.

A compact representation of this process is provided by the *adjacency matrix* indexgraph!adjacency matrix

$$A_{ij} = \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$

or rather the *normalized adjacency matrix*

$$Q_{ij} = \begin{cases} 1/\gamma_j & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

where γ_j is the degree of vertex j, that is, the number of edges incident at j. It is easy to see that if $p_t = (p_1^{(t)}, p_2^{(t)}, \dots, p_n^{(t)})^\top$ describes the probability of finding the random walker at each vertex at time t, then $p_{t+1} = Q p_t$. We say that Q is the transition matrix of the random walk. Applying this relation recursively shows that raising the normalized adjacency matrix to the power T,

$$P_T = Q^T, \tag{1.2}$$

gives the matrix whose i, j element describes the probability of a random walker starting from j being found at i at time T.

Generally, random walks have a tendency to meander about close to their origin. This is because when i and j are close and T is reasonably large, in most graphs there is a very large number of possible length T paths from i to j. When i and j are far apart the choice of paths is much more restricted and $[P_T]_{i,j}$ will be correspondingly small. Unfortunately, P_T is not suitable as a kernel, for a number of reasons:

1. There is no clear prescription for how to choose T. Any choice less than the diameter of G will lead to pairs of vertices not reachable from one another, and a corresponding absolute cutoff in the kernel. Kernels with such limited horizon do not have a sense of the global structure of G. On the other hand, choosing too large a T might make the peak of $[P_T]_{i,j}$ around i very flat, resulting in a kernel unable to differentiate betweeen nearby vertices.

2. If the graph has symmetries, particular choices of T might make certain vertices unreachable. For example, if G is just a cycle, an even number of steps could never take us from vertex i to either of its neighbors. Similarly, if T is odd, we could not get back to the vertex we started from.

3. Finally, and most seriously, P_T is generally not positive definite; in fact, it is not even guaranteed to be symmetric.

To see how to overcome these difficulties we first need to take a brief detour to continuous spaces. Physical ideas borrowed from the continuous case will help us construct a modified random walk leading to a natural kernel on G.

The Gaussian kernel and diffusion

One of the most popular kernels on $\mathcal{X} = \mathbb{R}^N$ is the Gaussian radial basis function kernel (Gaussian RBF or just Gaussian kernel)

$$k(\mathbf{x},\mathbf{x}') = \frac{1}{(2\pi\sigma^2)^{N/2}}\, e^{-\|\,\mathbf{x}-\mathbf{x}'\,\|^2/(2\sigma^2)} \tag{1.3}$$

with length scale parameter σ. It may not be immediately obvious from its functional form, but the Gaussian kernel is positive definite on $\mathbb{R}^N$.

The Gaussian has many famous properties, but for now we are going to concentrate on the fact that fixing $\mathbf{x}' = \mathbf{x}_0$ and letting $t = \sigma^2/2$,

$$k_{\mathbf{x}_0}(\mathbf{x},t) = \frac{1}{(4\pi t)^{N/2}}\, e^{-\|\,\mathbf{x}-\mathbf{x}_0\,\|^2/(4t)}$$

is the solution of the *diffusion equation*

$$\frac{\partial}{\partial t}\, k_{\mathbf{x}_0}(\mathbf{x},t) = \left[\frac{\partial^2}{\partial \mathbf{x}_{(1)}^2} + \frac{\partial^2}{\partial \mathbf{x}_{(2)}^2} + \ldots + \frac{\partial^2}{\partial \mathbf{x}_{(N)}^2}\right] k_{\mathbf{x}_0}(\mathbf{x},t), \tag{1.4}$$

with Dirac delta initial conditions $k_{\mathbf{x}_0}(\mathbf{x},0) = \delta(\mathbf{x}-\mathbf{x}_0)$. Here we use parenthesized indices to make it clear that we are differentiating with respect to the ith coordinate and not the ith training example. The second-order differential operator in (1.4) is called the *Laplacian* and is often denoted simply by Δ, reducing the diffusion equation to

$$\frac{\partial}{\partial t}\, k_{\mathbf{x}_0}(\mathbf{x},t) = \Delta\, k_{\mathbf{x}_0}(\mathbf{x},t)\,. \tag{1.5}$$

The physical meaning of these equations is clear: (1.5) describes how heat, gases, and so on, introduced at $\mathbf{x}_0$, diffuse with time in a homogeneous, isotropic medium. In learning theory, using the Gaussian kernel amounts to evoking a similar picture of the diffusion of labels $y_1, y_2, \ldots, y_m$ in $\mathcal{X}$. Every learning algorithm must make some sort of assumption about how similarity between inputs $\mathbf{x},\mathbf{x}' \in \mathcal{X}$ will lead to similarity between the corresponding outputs (labels) y, y'. The assumption behind the Gaussian kernel is essentially that $y(x)$ is to be approximated by diffusing the training labels to the rest of $\mathcal{X}$. Using a sophisticated algorithm such as a SVM complicates the picture somewhat, but the underlying idea remains the same.

Now we ask how to transplant these ideas to the discrete setting, in particular, to when $\mathcal{X}$ is a graph. Going back to the idea of random walks, the key modification we make to (1.2) is to make time continuous by taking the limit

$$K_\beta = \lim_{s \to \infty} \left(I + \frac{\beta L}{s} \right)^s \qquad s \in \mathbb{N}, \tag{1.6}$$

Diffusion on graphs

which corresponds to a random walk with an infinite number of infinitesimally small steps (I is the identity matrix and β is a real parameter). The time evolution of this "continuous time random walk" is now governed by the matrix L, which we again define as a close relative of the adjacency matrix,

Graph Laplacian

$$L_{ij} = \begin{cases} 1 & \text{if } i \sim j \\ -\gamma_i & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{1.7}$$

The negative elements on the diagonal serve the same function as dividing by γ_j in (1.1): their presence guarantees that each column of K_β, regarded as a probability distribution over vertices, remains normalized. In spectral graph theory L is known as the *graph Laplacian*, already suggesting that we are on the right track in developing the analogy with the continuous case. The kernel (1.6) we call the *diffusion kernel* or the *heat kernel* on G.

Properties of diffusion kernels

By analogy with the exponentiation of real numbers, the limit in (1.6) is called the *matrix exponential* of L:

$$e^{\beta L} = \lim_{s \to \infty} \left(I + \frac{\beta L}{s} \right)^s \qquad s \in \mathbb{N}. \tag{1.8}$$

Note that $e^{\beta L}$ yields a matrix, but is not equivalent to componentwise exponentiation $J_{ij} = e^{\beta L_{ij}}$. Matrix exponentiation shares many properties with the ordinary exponential function, including the power series expansion

$$e^{\beta L} = I + \beta L + \frac{\beta^2}{2} L^2 + \frac{\beta^3}{6} L^3 + \ldots$$

and the fact that $e^{\beta L}$ satisfies the differential equation

$$\frac{\partial}{\partial \beta} e^{\beta L} = L \, e^{\beta L}. \tag{1.9}$$

One important difference is that in general the identity

$$e^{\beta(A+B)} = e^{\beta A} \, e^{\beta B}$$

does *not* hold in the matrix case.

An important effect of defining kernels in such an exponential form is that we get positive definiteness "for free," as can be seen by writing

$$e^{\beta L} = \lim_{s\to\infty}\left(I + \frac{\beta L}{s}\right)^{s} = \lim_{2s\to\infty}\left(I + \frac{\beta L}{2s}\right)^{2s}$$

and appealing to the well-known fact that any even power of a symmetric matrix is always positive definite. In fact, it is possible to prove that any continuous family of positive definite matrices K_β indexed by a real parameter β in such a way that $K_0 = I$ is of the form $K_\beta = e^{\beta H}$ for some symmetric matrix H.

Analogies

The diffusion kernel will generally increase with increasing shortest-path distance between vertices, but there is no simple one-to-one relationship between d and K_β. Rather, it is helpful to think of K in terms of actual physical processes of diffusion. The function of the parameter β is to control the extent of the diffusion, or to specify the length scale, similarly to σ in the Gaussian kernel.

The correspondence between diffusion kernels and the Gaussian kernel is spelled out even more clearly by considering an infinite regular square grid on $\mathbb{R}^N$. Restricting ourselves to two dimensions for notational simplicity and labeling the vertices with their integer coordinates, the Laplacian becomes

$$L_{(i_1,i_2),(j_1,j_2)} = \begin{cases} 1 & \text{if } i_1 = j_1 \text{ and } i_2 = j_2 \pm 1 \\ 1 & \text{if } i_2 = j_2 \text{ and } i_1 = j_1 \pm 1 \\ -4 & \text{if } i_1 = j_1 \text{ and } i_2 = j_2 \\ 0 & \text{otherwise}. \end{cases}$$

Applied to a function $f : \mathcal{X} \mapsto \mathbb{R}$ [regarded as a vector indexed by (i_1, i_2) pairs], this gives

$$(Lf)_{i_1,i_2} = f_{i_1,i_2+1} + f_{i_1,i_2-1} + f_{i_1+1,i_2} + f_{i_1-1,i_2} - 4f_{i_1,i_2},$$

which is a well-known formula for the finite differences discretization of the continous Laplace operator Δ, commonly used in the numerical solution of partial differential equations. Hence, L can be regarded as just the discretized counterpart of Δ, and, correspondingly, K can be regarded as the discretized Gaussian RBF. The correspondence can be made exact by proving that in the limit of the grid spacing going to zero, K will indeed approach the Gaussian kernel (1.3).

On more general graphs the key to understanding diffusion kernels is the differential equation (1.9). This expresses that starting from the identity matrix $K_0 = I$, the kernel K_β is generated by successive infinitesimal applications of L. Note that the Laplacian encodes strictly local information about the graph. However, through this continuous process expressed by the differential equation, L is lifted to a smooth global similarity measure on G, which is exactly what a kernel is supposed to be.

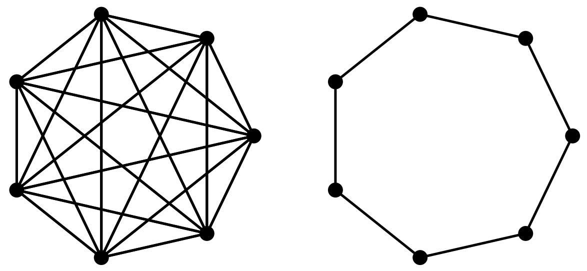

Figure 1.1 Two examples of elementary graphs for which the diffusion kernel can be found in closed form: the complete graph and the closed chain over seven vertices.

1.2 Computing the Diffusion Kernel

At this point, the reader might be wondering how he or she might compute the limit (1.8) on a computer in a finite amount of time. In general, the way to proceed is to compute the normalized eigenvectors $v_1, v_2, \ldots, v_n$ and corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of L and exploit orthogonality to write

$$L^s = \left(\sum_{i=1}^{n} v_i \, \lambda_i \, v_i^\top \right)^s = \sum_{i=1}^{n} v_i \, \lambda_i^s \, v_i^\top ,$$

from which

$$e^{\beta L} = I + \left(\sum_{i=1}^{n} v_i \, \beta \lambda_i \, v_i^\top \right) + \left(\sum_{i=1}^{n} v_i \, \frac{(\beta \lambda_i)^2}{2} \, v_i^\top \right) + \ldots = \sum_{i=1}^{n} v_i \, e^{\beta \lambda_i} \, v_i^\top , \quad (1.10)$$

reducing matrix exponentiation to real exponentiation. Unfortunately, the complexity of diagonalizing the Laplacian is of order n^3, which threatens to be computationally more expensive than the learning itself for most learning algorithms. However, there are a few special graphs for which the diffusion kernel can be computed in closed form.

Complete graphs

In the complete graph of order n every vertex is linked to every other vertex, so the Laplacian is $L_{ij} = 1 - n \, \delta_{ij}$. In this case (1.9) can be solved explicitly giving

$$k(i,j) = K_{i,j} = \begin{cases} \dfrac{1 + (n-1) \, e^{-n\beta}}{n} & \text{for } \; i = j \\[2ex] \dfrac{1 - e^{-n\beta}}{n} & \text{for } \; i \neq j , \end{cases}$$

showing that with increasing β, the kernel relaxes exponentially to $k(i,j) = 1/n$. The asymptotically exponential character of this solution converging to the uniform kernel is a direct consequence of the form of the diffusion equation. We shall see this type of behavior recur in other examples.

Closed chains

When G is a single closed chain, $k(i,j)$ can clearly only depend on the distance $d(i,j)$ along the chain between i and j. Labeling the vertices consecutively from 0 to $n-1$ and fixing i (without loss of generality, taking $i=0$), $k_0(j)=k(0,j)$ can be expressed in terms of its discrete Fourier transform

$$k(0,j) = k_0(j) = \frac{1}{\sqrt{n}} \sum_{\nu=0}^{n-1} \widehat{k}_0(j) \, \cos \frac{2\pi\nu j}{n} \; .$$

The heat equation implies

$$\frac{d}{d\beta}\, k_0(j) \;=\; k_0((j+1) \bmod n) - 2\,k_0(j) + k_0((j-1) \bmod n),$$

which after some trigonometry translates into

$$\frac{d}{d\beta}\, \widehat{k}_0(\nu) = -\,2\left(1 - \cos\frac{2\pi\nu}{n}\right) \widehat{k}_0(\nu),$$

showing that each Fourier coefficient decays independently. Now applying the inverse Fourier transform, the solution corresponding to the initial condition $k_0(i)=\delta_{i,0}$ at $\beta=0$ can be expressed as

$$k_0(j) = \frac{1}{n} \sum_{\nu=0}^{n-1} e^{-\omega_\nu \beta} \cos \frac{2\pi\nu j}{n},$$

where $\omega_\nu = 2\left(1 - \cos\frac{2\pi\nu}{n}\right)$. Hence the full kernel is

$$k(i,j) = \frac{1}{n} \sum_{\nu=0}^{n-1} e^{-\omega_\nu \beta} \cos \frac{2\pi\nu(i-j)}{n} \; .$$

p-regular trees

There also exist special solutions for tree-shaped graphs, albeit infinite trees with no root. A p-regular tree is an infinite graph with no cycles in which every vertex has exactly p neighbors (figure 1.2). Clearly, such a tree looks the same from every vertex, so $k(i,j)$ can only depend on the shortest-path distance $d(i,j)$. Even for such simple, highly symmetric graphs, the formula for the diffusion kernel is not trivial and has only recently been derived (Chung and Yau, 1999):

$$k(i,j) = \frac{2}{\pi(k-1)} \int_0^\pi \frac{e^{-\beta\left(1-\frac{2\sqrt{k-1}}{k}\cos x\right)} \sin x\,[\,(k-1)\sin(d+1)\,x - \sin(d-1)\,x\,]}{k^2 - 4\,(k-1)\,\cos^2 x}\, dx$$

for $d = d(i,j) \geq 1$, and

$$k(i,i) \;=\; \frac{2k(k-1)}{\pi} \int_0^\pi \frac{\exp\left(-\beta\left(1 - \frac{2\sqrt{k-1}}{k}\cos x\right)\right)\, \sin^2 x}{k^2 - 4\,(k-1)\,\cos^2 x}\, dx$$

for the diagonal elements. The case of infinite rooted trees can be reduced to p-regular trees using a trick from physics called the "method of images," as described in Kondor and Lafferty (2002).

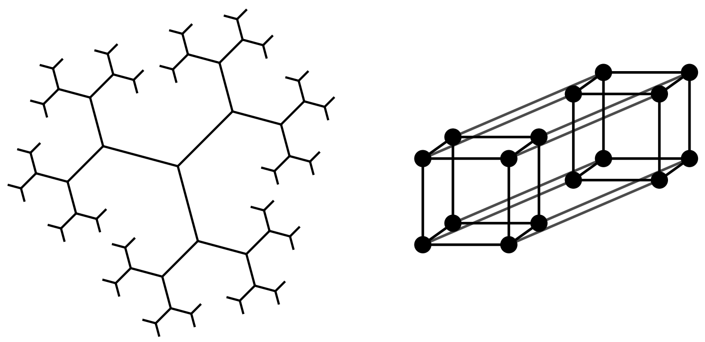

Figure 1.2 *Left,* A few edges of the three-regular tree. The tree extends to infinity in all directions. *Right,* The four-dimensional hypercube can be regarded as the complete graph of order two (two vertices joined by a single edge) "cubed."

Product graphs

Another way to compute diffusion kernels is to reduce complex graphs to elementary ones. Specifically, let G_1 and G_2 be two undirected, unweighted graphs with n_1 and n_2 vertices, respectively. The direct product graph $G = G_1 \times G_2$ will then have $n_1 \cdot n_2$ vertices labeled by pairs of integers (i_1, i_2), with $1 \leq i_1 \leq n_1$ and $1 \leq i_2 \leq n_2$. Two vertices (i_1, i_2) and (j_1, j_2) will then be neighbors either if $i_1 = j_1$ and $i_2 \sim j_2$ or if $i_2 = j_2$ and $i_1 \sim j_1$. The infinite square grid that we encountered at the end of section 1.1 is an example of such a structure. More generally, also encompasssing weighted graphs, the adjacency matrix of the product graph will be related to the adjacency matrices of the factor graphs by

$$A = A_1 \otimes I_2 + I_1 \otimes A_2,$$

where I_1 and I_2 are the $n_1 \times n_1$ and $n_2 \times n_2$ unit matrices, respectively. The Laplacians are similarly related:

$$L = L_1 \otimes I_2 + I_1 \otimes L_2$$

and the corresponding diffusion kernel will be the direct product of the diffusion kernels on the factor graphs:

$$K_\beta = K_\beta^{(1)} \otimes K_\beta^{(2)},$$

as can easily be seen by plugging this into the diffusion equation

$$\frac{\partial}{\partial \beta} K_\beta = L K_\beta$$

and invoking the product rule for differentiation.

The product graph construction makes it possible to compute diffusion kernels on the D-dimensional hypercube, regarded as the D-fold power of a complete graph of order 2:

Hypercubes

$$k(\mathbf{x},\mathbf{x}') \propto \left(\frac{1-e^{-2\beta}}{1+e^{-2\beta}}\right)^{d(\mathbf{x},\mathbf{x}')} = (\tanh\beta)^{d(\mathbf{x},\mathbf{x}')},$$

where $d(\mathbf{x},\mathbf{x}')$ is the Hamming distance betweeen vertices $\mathbf{x}$ and $\mathbf{x}'$. More generally, we may consider products of complete graphs of orders $g_1, g_2, \ldots g_D$, and compute the kernel

$$k(\mathbf{x},\mathbf{x}') \propto \prod_{i=1}^{D} \left[\frac{1-e^{-\beta g_i}}{1+(g_i-1)e^{-\beta g_i}}\right]^{d_i(\mathbf{x},\mathbf{x}')},$$

where $d_i(\mathbf{x},\mathbf{x}') = 0$ if $\mathbf{x}$ and $\mathbf{x}'$ match in the ith index and 1 otherwise. This construction can be used to apply diffusion kernels to learning problems involving categorical data (Kondor and Lafferty, 2002), assuming D distinct attributes with $g_1, g_2, \ldots g_D$ possible values.

1.3 Regularization Theory

The correspondence between kernel-based algorithms and regularization theory is now widely recognized in the machine learning community (Smola et al., 1998; Girosi, 1998; Girosi et al., 1995; Tikhonov and Arsenin, 1977). SVMs, Gaussian processes, and so on, can all be cast in the form of searching some linear space of functions $\mathcal{H}$ to find $f: \mathcal{X} \mapsto \mathcal{Y}$ minimizing the *regularized risk*

$$R_{\text{reg}}[f] = \sum_{i=1}^{m} \mathcal{L}(f(\mathbf{x}_i), y_i) + \Omega[f], \tag{1.11}$$

where $\Omega[f]$ is expressed in terms of a *regularization operator* $P: \mathcal{H} \mapsto \mathcal{H}$ as

$$\Omega[f] = \int_{\mathcal{X}} [(Pf)(\mathbf{x})]^2 \, d\mathbf{x}.$$

Without going into detail, we note that (1.11) expresses a tradeoff between fitting the training data and generalizing well to future examples. Given a loss function $\mathcal{L}$, the first term tries to minimize the error of f on the training set, while the second term stands for the competing objective of restricting f to "desirable" functions according to some criterion embodied in P. The choice of algorithm corresponds to choosing $\mathcal{L}$ and the choice of kernel to choosing the regularization operator.

When $\mathcal{X}$ is a finite graph, f is just a vector, P is a matrix, and the regularization term becomes

$$\Omega[f] = \|\, Pf \,\|^2 = f^\top (P^\top P)\, f. \tag{1.12}$$

Since $P^\top P$ is symmetric, its normalized eigenvectors $v_1, v_2, \ldots, v_n$ form an orthonormal basis, and expressing f as $f = \sum_{i=1}^n c_i v_i$, (1.12) becomes

$$\Omega[f] = \sum_{i=1}^{n} \lambda_i c_i^2,$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues corresponding to $v_1, v_2, \ldots, v_n$. The effect of the regularization term in (1.11) will be to force most of the energy in f to the low eigenvalue modes of $P^\top P$. From the learning theory point of view, our goal is to make f as smooth as possible, in the sense of making few sudden jumps between neighboring vertices. Jagged functions are likely to overfit the training data and not generalize well to future examples. Hence, we are interested in regularization operators whose eigenfunctions form a progression from the smoothest to the roughest functions on our graph.

Connection between K and P

To see how diffusion kernels fare from the regularization theory point of view, it remains to find the connection between the kernel and the regularization operator already alluded to. For concreteness, consider support vector regression, which aims to fit to the data a function f of the form $f(\mathbf{x}) = \langle \boldsymbol{w}, \Phi(\mathbf{x}) \rangle$ for some feature space vector $\boldsymbol{w}$ by minimizing

$$\frac{1}{2} \| \boldsymbol{w} \| + C \sum_{i=1}^{m} | f(\mathbf{x}_i) - y_i |_\epsilon ,$$

where $| f(\mathbf{x}) - y |_\epsilon$ is the ϵ-insensitive loss function $\max \{ 0, | f(\mathbf{x}) - y | - \epsilon \}$ (Vapnik, 1995). As in other kernel methods, the solution of this minimization problem will be of the form

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{1.13}$$

reducing it to finding the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$ minimizing

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{m} \sum_{j=1}^{m} | \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) - y_j |_\epsilon . \tag{1.14}$$

When $\mathcal{X}$ is discrete, if we collect the coefficients in a vector, the first term becomes just $\alpha^\top K \alpha$. Similarly, the regularization term (1.12) can be written as $\Omega[f] = \alpha^\top K (P^\top P) K \alpha$. Then comparing (1.14) with (1.11) we see that the two can be made equivalent by setting

$$P^\top P = K^{-1} \quad \text{and} \quad \mathcal{L}(f(\mathbf{x}), y) = 2C \, | f(\mathbf{x}) - y |_\epsilon .$$

Since K is positive definite, we may just take $P = K^{-1/2}$. The relationship between kernel and regularization is rather direct.

Specifically, referring back to (1.10), L, K, and $P^\top P$ share the same orthonormal system of eigenvectors $v_1, v_2, \ldots, v_n$ and their eigenvalues are related by

$$\lambda_i^{(K)} = \exp(\beta\,\lambda_i^{(L)}) \qquad \lambda_i^{(P^\top P)} = \exp(-\beta\,\lambda_i^{(L)}). \tag{1.15}$$

Furthermore, from the definition of the graph Laplacian (1.7),

$$-\lambda_i^{(L)} = -\,v_i^\top L\, v_i = \sum_{\mathbf{x}\sim\mathbf{x}'} \left(v_i(\mathbf{x}) - v_i(\mathbf{x}')\right)^2,$$

which can be interpreted as how many edges v_i "violates." Ordering $v_1, v_2, \ldots, v_n$ so that $-\lambda_1 \leq -\lambda_2 \leq \ldots \leq -\lambda_n$, the first eigenvector will always be the constant function on G with eigenvalue 0. The second eigenvector will tend to be positive on roughly one half of G and negative on the other half with a smooth transition in between. The third, fourth, and so on, eigenvectors correspond to successive subdivisions, all the way to the last eigenvectors, which oscillate rapidly, changing sign between most pairs of neigboring vertices. Together with (1.15) this shows that the regularization operator corresponding to the diffusion kernel does indeed establish a basis of functions on G sorted by smoothness.

The natural analogy is the Fourier basis of sine and cosine functions on $\mathbb{R}^N$. In general, K acts as a smoothing operator, since it dampens the "high frequency" modes of the Laplacian, while P is a coarsening operator, because it exaggerates them.

This type of analysis involving operators and their eigensystems (albeit in a somewhat more rigorous form) is at the center of spectral graph theory (Chung, 1997), some of the cornerstones of which are the Laplacian and the heat kernel. In fact, we could have motivated this whole chapter purely by regularization ideas, and derived the diffusion kernel that way, instead of talking about the actual physical process of diffusion, or appealing to the analogy with the Gaussian kernel.

1.4 Applications

In this section we present an application of the diffusion kernel idea to the analysis of gene expression data and metabolic pathways. We first present in subsection 1.4.1 a graph of gene, called the metabolic gene network, which encodes our current knowledge of metabolic pathways. We then show how the regularization operator associated with the diffusion kernel on this graph can be useful for extracting pathway activity from gene expression data, and illustrate this approach by a short analysis of gene expression during two cell cycles. More details on this approach can be found in Vert and Kanehisa (2003b), and more data analysis is presented in Vert and Kanehisa (2003a).

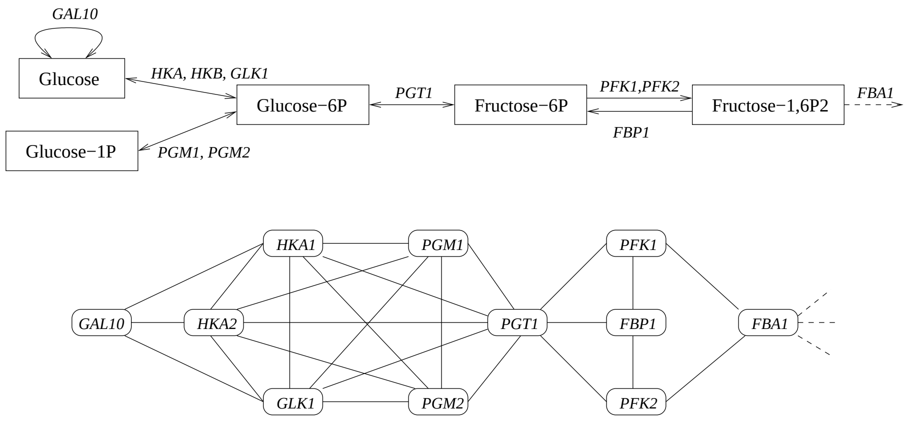

Figure 1.3 *Top*, The first three reactions of the glycolysis pathway, together with the catalyzing enzymes in the yeast *S. cerevisiae. Bottom*, The metabolic gene network derived from these reactions by linking two genes whenever they can catalyze two successive reactions.

1.4.1 The Metabolic Gene Network

At the molecular level life can be described as an continual flow of chemical reactions that degrade, synthesize, or transform various molecules for various purposes. In particular, metabolism encompasses the processes by which energy is provided for vital processes and activities, and by which new material is assimilated. Metabolic processes are usually organized into series of chemical reactions, called *metabolic pathways*, that take place sequentially in the cell. As an example, glycolysis is the metabolic pathway in charge of degrading glucose into pyruvate with the concomitant production of adenosine triphosphate (ATP) molecules. Figure 1.3 (top) shows the first three reactions of glycolysis: addition of a phosphate group to the glucose molecule to obtain glucose-6P, followed by an izomerization of glucose-6P into fructose-6P, and by the addition of a second phosphate group to obtain fructose-1,6P2.

Metabolic pathways

Each reaction in a pathway usually requires the presence of a particular molecule, called an enzyme, to occur. Enzymes catalyze reactions, that is, they facilitate the reaction usually by placing the various substrates in a precise spatial configuration. Most enzymes in biochemistry are proteins synthesized by the cell itself, which are encoded in the genome of the organism. In Figure 1.3 (top) the enzymes are characterized by the name of the genes that encode them. For example, the izomerization from glucose-6P to fructose-6P is catalyzed by the protein encoded by the gene PGT1 in yeast. When several genes are associated with a single reaction, it is either because the proteins they encode form a complex, or because several different proteins can catalyze the same reaction.

Metabolic gene network

The *metabolic gene network* is derived from the set of metabolic pathways as follows. It is an undirected graph whose vertices are the genes of a given organism, and with an edge between two genes whenever the proteins they encode can participate in the catalysis of two successive reactions, that is, two reactions such that the main product of the first one is the main substrate of the second one. As an example, the local metabolic graph network for the yeast derived from the first three reactions of glycolysis is shown in Figure 1.3 (bottom).

When all known metabolic pathways are considered, the resulting metabolic gene network is complex for at least two reasons. First, the same chemical compound (such as glucose) can be present in different metabolic pathways, and therefore edges can link genes which catalyze reactions in different pathways. Second, a given gene can usually catalyze more than one reaction.

In the following we use the metabolic gene network for the yeast *Sacchromyces cerevisiae* derived from the metabolic pathways available in the LIGAND database of chemical compounds of reactions in biological pathways (Goto et al., 2002). The resulting graph contains 774 nodes linked through 16,650 edges.

1.4.2 Gene Expression

Reactions in a metabolic pathway occur when both the necessary subtrates and the necessary enzymes are present. As a result, a cell can control its metabolism by controlling the quantity of each enzyme. Because enzymes are proteins, the first level of control of their concentrations is the control of gene expression. For example, in the bacterium *Escherichia coli*, the presence of tryptophan inhibits the expression of the genes that encode the enzymes which catalyze the reactions of the tryptophan synthesis pathway.

DNA microarrays

DNA microarray technology enables the monitoring of gene expression for all genes of an organism simultaneously. It is therefore tempting to try to make sense of gene expression data in terms of pathways, at least for the genes that encode enzymes. More precisely, by following the expression of enzyme genes through various experiments, one can hope to detect activated or inhibited pathways, suggest new pathways, or detect mistakes in the current pathway databases. As a first step toward these ambitious goals we now present a method for automatically detecting activity levels of known pathways by confronting gene expression data with the metabolic gene network.

1.4.3 Mathematical Formulation

Let us represent the set of genes by the finite set $\mathcal{X}$. The metabolic gene network is a simple graph $\Gamma = (\mathcal{X}, \mathcal{E})$ with the genes as vertices. The set of expression profiles measured by DNA microarray for a gene $\mathbf{x} \in \mathcal{X}$ is a vector $e(\mathbf{x}) \in \mathbb{R}^p$, where p is the number of microarray measurements. By subtracting the mean profile from all genes, we suppose below that the set of profiles is centered, that is, $\sum_{\mathbf{x}\in\mathcal{X}} e(\mathbf{x}) = 0$.

We formulate our problem as the problem of automatically finding profiles which exhibit some coherence with respect to the graph topology. Formally speaking, a profile is a vector $v \in \mathbb{R}^p$. We don't require v to be any actual gene expression profile, but rather use it to represent some more abstract or hidden information, such as the quantity of some substance in the cell, or the activity of a pathway. Intuitively, if v represents the evolution of such a biological quantity, then expression profiles of genes participating in or affected by this event should exhibit some form of correlation with v.

For a zero-mean candidate profile $v \in \mathbb{R}^p$ (i.e., $\sum_{i=1}^p v_i = 0$), let us therefore call $f_1(\mathbf{x}) := v^T e(\mathbf{x})$ the correlation between the profile v and the gene expression profile $e(\mathbf{x})$. Typically, if v represents the activity level of a pathway where gene $\mathbf{x}$ plays a regulatory role, then $f_1(\mathbf{x})$ is likely to be either strongly positive or strongly negative.

These remarks lead to a key observation. If v represents the activity of a pathway, then $f_1(\mathbf{x})$ is likely to be particularly positive or negative for several of the enzymes $\mathbf{x}$ that catalyze the corresponding reactions. Observed on the metabolic gene graph, this suggests that f_1 should have less variations on average between adjacent nodes if v corresponds to a true biological process than otherwise. Indeed, we can at least expect some local regularities in the regions of the graph that correspond to the pathways related to the process.

Smoothness on the network

This is where the diffusion kernel becomes useful. Recall from section 1.3 that the diffusion kernel K_1 on the metabolic gene network defines a regularization operator $\Omega_1[f]$ on functions $f : \mathcal{X} \to \mathbb{R}$ that is small when f has little high-frequency energy. This suggests looking for candidate profiles v such that $\Omega_1[f_1]$ be as small as possible. Writing f_1 in a dual form $f_1 = K_1\alpha$, this means requiring $\alpha^T K_1 \alpha / \alpha^T \alpha$ to be as small as possible.

On the other hand, minimizing $\Omega_1[f_1]$ over v is likely to be an ill-posed or at least an ill-conditioned problem. First observe that any component of v orthogonal to the linear span of $\{e(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$ does not modify f_1. This suggests restricting v to this subspace, that is, writing v as $v = \sum_{\mathbf{x}\in\mathcal{X}} \beta(\mathbf{x}) e(\mathbf{x})$, and therefore $f_1 = K_2\beta$ where K_2 is the Gram matrix of the linear kernel $K_2(\mathbf{x}, \mathbf{y}) = e(\mathbf{x})^T e(\mathbf{y})$ and $\beta : \mathcal{X} \to \mathbb{R}$ is a dual form of f_1. Second, when the linear span of $\{e(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$ is large, eventually the whole space of centered profiles, then a perfectly smooth function might be found whether or not a "true" biological correlation exists between the profiles and the graph. This suggests imposing some form of regularity on v, such as being close to the directions of natural variations between profiles. This is achieved by requiring $\Omega_2[f_1] = \beta' K_2 \beta / \beta' \beta$ to be as small as possible. In order to end up with a computationally feasible formulation, Vert and Kanehisa (2003b) proposed decoupling the problem as follows: find two different functions $f_1 = K_1\alpha$ and $f_2 = K_2\beta$ such that $\Omega_1[f_1]$ and $\Omega_2[f_2]$ are both small, while f_1 and f_2 are as similar as possible. A possible measure of similarity between f_1 and f_2 being their correlations $f_1^\top, f_2$, the different goals can be fulfilled simultaneously by maximizing the following functional:

Natural variations

$$\gamma(\alpha, \beta) := \frac{\alpha^T K_1 K_2 \beta}{\left(\alpha^T \left(K_1^2 + \delta K_1\right) \alpha\right)^{\frac{1}{2}} \left(\beta^T \left(K_2^2 + \delta K_2\right) \beta\right)^{\frac{1}{2}}}, \tag{1.16}$$

where δ is a parameter that controls the tradeoff between regularities of f_1 and f_2 on the one hand, and similarity of f_1 and f_2 on the other hand. It turns out that (1.16) can be seen as a regularized form of canonical component analysis (Bach and Jordan, 2002), equivalent to the following generalized eigenvalue problem:

Kernel CCA

$$\begin{pmatrix} 0 & K_1K_2 \\ K_2K_1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \rho \begin{pmatrix} K_1^2 + \delta K_1 & 0 \\ 0 & K_2^2 + \delta K_2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{1.17}$$

As pointed out in Bach and Jordan (2002) and Vert and Kanehisa (2003b) this problem can be solved efficiently and results in a series of pairs of features $\{(\alpha_i, \beta_i), i = 1, \ldots, n\}$ with decreasing values of $\gamma(\alpha_i, \beta_i)$. The corresponding profiles can be recovered by $v_i = \sum_{\mathbf{x} \in \mathcal{X}} \beta_i(\mathbf{x}) e(\mathbf{x})$.

1.4.4 Analysis of α Factor Release

In order to illustrate this method on a real-world example we compared the metabolic gene network with a collection of 18 expression measurements for 6198 yeast genes, collected every 7 minutes after cells were synchronized in G_1 by addition of α factor (Spellman et al., 1998). The original goal of Spellman et al. (1998) was to detect genes whose expression exhibits periodicity related to the cell cycle.

The analysis that follows is restricted to the 756 genes of the metabolic gene netwok with an expression profile in this set. The profiles contain 18 points, hence 17 pairs or features with dual coordinates $(\alpha_i, \beta_i)_{i=1,\ldots,17}$ were extracted. We solved (1.17) using the free and publicly available program Octave.[1] Following experiments detailed in Vert and Kanehisa (2003b) the regularization parameter δ of (1.16) was set to 0.01.

Figure 1.4 shows the first two profiles extracted, and table 1.1 contains a list representative of the genes with highest or lowest correlation with each profile, as well as the pathways they participate in in the KEGG database.

The first extracted profile is essentially a strong signal immediately following the beginning of the experiment. Several pathways positively correlated with this pattern are involved in energy metabolism (oxidative phosphorylation, tricarboxylic acid cycle, glycerolipid metabolism), while pathways negatively correlated are mainly involved in protein synthesis (aminoacyl-tRNA biosynthesis, RNA polymerase, pyrimidine metabolism). Hence this profile clearly detects the sudden change of environment, and the priority fueling the start of the cell cycle with fresh energetic molecules rather than synthesizing proteins. This result highlights the experimental bias in the data: while the goal of the experiment was to study the evolution of gene expression during the cell cycle, the strongest signal we detected is related to the need to synchronize the cells by addition of α factor.

Detection of experimental bias

The second extracted profile exhibits a strong sinusoidal shape corresponding to the progression in the cell cycle experiment, but the first one is more visible than

1. Available at `http://www.octave.org`.

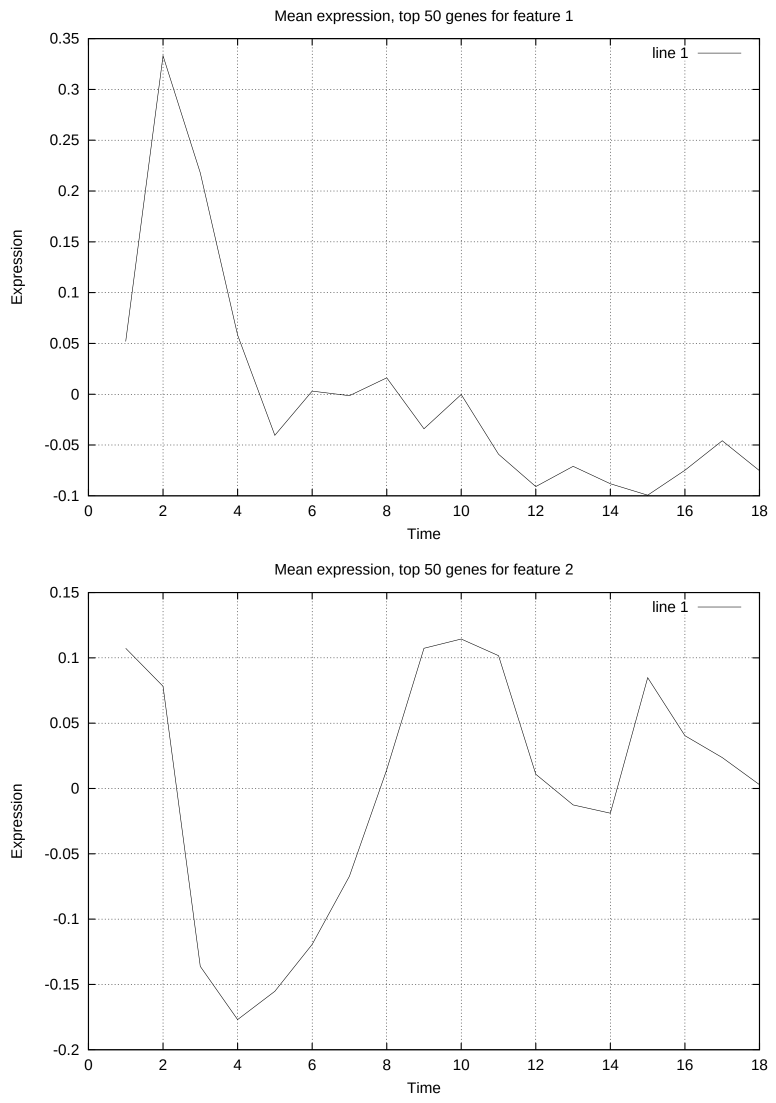

Figure 1.4 First 2 profiles extracted (α factor data set).

Table 1.1 Pathways and genes with highest and lowest scores on the first 2 features extracted.

Feature	Correlation	Main pathways and genes
1	+	Glycolysis/gluconeogenesis (*PGK1, GPM2, ALD4,6*), TCA cycle (*CIT2, MDH1,2, SDH1, LSC1*), pentose phosphate pathway (*RBK1, SOL4, ZWF1, YGR043C*), glycerolipid metabolism (*GPD1,2,3, ALD4,6*), glyoxylate and dicarboxylate metabolism (*MDH1,2, CIT2, ICL2*), sulfur metabolism (*MET2,14,16,17*).
1	-	Pyrimidine metabolism (*RPA12,34,49,190, RPB2,5, RPC53, DUT1, TRR1, POL5, URK1, MIP1, PUS1*), purine metabolism (*RPA12,34,49,190, RPB2,5, RPC53, CDC19, APT2, POL5, MIP1*), aminoacyl-tRNA biosynthesis (*ILS1, FRS2, MES1, YHR020W, GLN4, ALA1, CDC60*), starch and sucrose metabolism (*MPS1, HPR5, SWE1, HSL1, EXG1*).
2	+	Pyrimidine metabolism (*DEG1, PUS1,3,4, URA1,2, CPA1,2,FCY1*), folate biosynthesis (*ENA1,5, BRR2, HPR5, FOL1*), starch and sucrose metabolism (*ENA1,5, BRR2, HPR5, PGU1*), phenylalanine, tyrosine, and tryptophan biosynthesis (*TRP2,3,4, ARO2,7*), sterol biosynthesis (*ERG7,12, HGM1,2*).
2	-	Starch and sucrose metabolism (*CDC7, ENA1, GIN4, HXK2, HPR5, SWE1, UGP1, HSL1, FKS1, MEK1*), purine and pyrimidine metabolism (*POL12, ADK2, DUT1, RNR2, HYS2, YNK1, CDC21*), fructose and mannose metabolism (*MNN1, PMI40, SEC53, HXK2*), cell cycle (*CDC7, GIN4, SWE1, HSL1*).

Detection of the cell cycle

the second one because the synchronization in the yeast colony decreased while the experiment progressed. Several genes directly involved in DNA synthesis . Two cell cycles took place during the (*YNK1*, *RNR2*, *POL12*) can be recognized in the list of genes anticorrelated with the second feature (corresponding to maximum expression in the S phase). Some pathways such as the starch metabolism have genes which exhibit either strong correlation or strong anticorrelation with the second profile, corresponding to the various regimens in the normal cell cycle (e.g., periods of energy storage alternate with periods of energy consumption).

1.5 Extensions

Weighted graphs

In weighted graphs each edge has an associated weight $w_{ij} = w_{ji} > 0$ or $w_{ij} = 0$ if i and j are not connected. Weighted graphs can naturally be incorporated in the diffusion kernel framework by defining the Laplacian as

$$L_{ij} = \begin{cases} w_{ij} & \text{if } i \neq j \\ -\sum_{l=1}^{n} w_{il} & \text{if } i = j. \end{cases}$$

The rest of the development follows exactly as in the unweighted case. Unfortunately, no similarly straightforward solution suggests itself for directed graphs ($w_{ij} \neq w_{ji}$), since the symmetry of L is essential for the positive definiteness of k.

Other kernels

A different line of generalizing diffusion kernels, expounded in Smola and Kondor (2003), focuses on replacing $e^{\beta L}$ with a general expansion in terms of the eigenvalues and eigenvectors of L,

$$k = \sum_{i=1}^{n} v_i \, \frac{1}{r(\lambda_i)} \, v_i^\top$$

for some function $r : \mathbb{R} \mapsto \mathbb{R}^+$. Choosing $r(\lambda) = \exp(\sigma^2 \lambda / 2)$ gives the diffusion kernel, but other choices lead to interesting new kernels such as

$$\begin{aligned} r(\lambda) &= 1 + \sigma^2 \lambda && \text{(regularized Laplacian kernel)} \\ r(\lambda) &= (aI - \lambda)^{-p} && \text{(p-step random walk kernel)} \\ r(\lambda) &= \cos(\pi\lambda/4) && \text{(inverse cosine kernel)}\ , \end{aligned}$$

although these generally cannot boast a similarly rich collection of interpretations.

Metric spaces

The most farreaching avenue of generalizing the ideas in this chapter involves applying the same framework to other mathematical objects, not just graphs. The ideas of diffusion and corresponding regularization are natural and well-studied concepts on a whole spectrum of different metric spaces. Indeed, the Laplacian and the spectral theory induced by it are two of the great unifying concepts of mathematics. For want of space here we can only sketch the wealth of possibilities this leads to.

For $\mathcal{X} = \mathbb{R}^N$ we have already encountered the Laplacian

$$\Delta = \frac{\partial^2}{\partial \mathbf{x}_{(1)}^2} + \frac{\partial^2}{\partial \mathbf{x}_{(2)}^2} + \ldots + \frac{\partial^2}{\partial \mathbf{x}_{(N)}^2}$$

and we have seen that the explicit solution of the diffusion equation is the Gaussian kernel

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{(4\pi\beta)^{N/2}} \, e^{-\| \mathbf{x} - \mathbf{x}' \| / (4\beta)}.$$

It should not come as a surprise that the eigenfunctions of K in this case are the harmonic eigenfunctions

$$\sin(2\pi k \cdot \mathbf{x}) \quad \text{and} \quad \cos(2\pi k \cdot \mathbf{x}), \qquad k \in \mathbb{R}^N$$

with corresponding eigenvalues $e^{-\| k \|^2 / (4\beta)}$.

Riemannian manifolds

The generalization of the Laplacian to curved spaces (Riemannian manifolds) is

$$\Delta = \frac{1}{\sqrt{\det g}} \sum_{ij} \partial_i \left(\sqrt{\det g} \, g^{ij} \partial_j \right),$$

where g is the metric tensor and ∂_i denotes differentiation with respect to the ith coordinate. Plugging this operator into the diffusion equation (1.5) we can solve for k. Locally the diffusion kernel will be very similar to the Gaussian kernel, but not so further away, due to the curvature and possibly nontrivial topology. Unfortunately, there are very few manifolds on which k can be computed in closed form and one has to resort to using asymptotic expansions.

In various domains, most notably image analysis, data often fall on curved manifolds embedded in a higher-dimensional space. Constraining the kernel, and hence the whole learning problem, to this manifold can improve the performance of learning algorithms. Unfortunately, in most cases the manifold is not known and may have very complicated geometry. Belkin and Niyogi (2002) and Belkin and Niyogi (2003) have proposed approximating the manifold by a mesh obtained by connecting data points, for example, according to a k-nearest neighbor rule. The diffusion kernel on the manifold can then be approximated by the diffusion kernel on this mesh, treated as a graph. What is especially attractive in this procedure is that it provides a natural use for unlabeled data. Unlabeled data points will not, of course, feature in a support vector expansion such as (1.13), but they can still play an important role in the learning process as vertices of the mesh, helping to form a good kernel. This can make learning possible in scenarios where only a very small fraction of data points are labeled.

The statistical manifold

Another context in which manifolds appear in learning theory is information geometry (Amari and Nagaoka, 2000). Consider a family of probability distributions $\{p_\theta(\mathbf{x})\}$ parameterized by $\theta \in \mathbb{R}^d$. The natural metric on this space of distributions, for a variety of reasons that we do not have space to go into here, is the Fisher metric

$$g_{ij} = \mathrm{E}_\theta\left[(\partial_i \ell_\theta)(\partial_j \ell_\theta)\right] = \int \left(\partial_i \log p(\mathbf{x}|\theta)\right)\left(\partial_j \log p(\mathbf{x}|\theta)\right) p(\mathbf{x}|\theta)\, d\mathbf{x},$$

where $\ell_\theta(\mathbf{x}) = \log p(\mathbf{x}|\theta)$. The Riemannian manifold this metric gives rise to is called the statistical manifold. The geometry of such manifolds can get rather involved and explicit calculations on them are almost never possible. However, a few celebrated special cases do exist. Lafferty and Lebanon (2003) have shown how the diffusion kernel can be computed in closed form on the statistical manifold of the spherical normal family and how it can be approximated for the multinomial family, in which case the geometry happens to be identical to that of a quadrant of a hypersphere. Assuming one of these two models generate the data, the kernel between two data points can be defined as the information diffusion kernel between the model fit to one and the other. Combined with SVMs, the authors successfully employ this technique to text classification with impressive results.

Acknowledgments

At about the same time that diffusion kernels were introduced by R.K. and John Lafferty (Kondor and Lafferty, 2002), Mikhail Belkin and Partha Niyogi were independently applying similar ideas from spectral graph theory to learning problems in a slightly different context (Belkin and Niyogi, 2002). The problem of partially labeled data is treated in Belkin and Niyogi (2003). Alex Smola has contributed by extending Laplacian-based kernels on graphs beyond diffusion kernels (Smola and Kondor, 2003) and working out the corresponding regularization theory. The application of diffusion kernels to bioinformatics was pioneered by J.-P.V. and Minoru Kanehisa (Vert and Kanehisa, 2003b) and information diffusion kernels were introduced by Lafferty and Lebanon (2003). In mathematics, the ideas of spectral geometry go back a little bit further, at least to Marquis Pierre Simon de Laplace (1749-1827).

The authors thank all of the above for many exchanges of ideas and for their role in bringing the concepts sketched in this chapter into the mainstream of machine learning. This work was partially supported by NSF grant CCR-0312690.

References

S.-I. Amari and H. Nagaoka. *Methods of Information Geometry.* Oxford, UK, Oxford University Press, 2000.

F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 585–591. Cambridge, MA, MIT Press, 2002.

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. Cambridge, MA, MIT Press, 2003.

Fan R. K. Chung. *Spectral Graph Theory.* Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.

Fan R. K. Chung and Shing-Tung Yau. Coverings, heat kernels and spanning trees. *Electronic Journal of Combinatorics*, 6, 1999.

F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

S. Goto, Y. Okuno, M. Hattori, T. Nishioka, and M. Kanehisa. LIGAND: Database of chemical compounds and reactions in biological pathways. *Nucleic Acids Research*, 30:402–404, 2002.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the 19th International Conference (ICML 2002)*, pages 315–322. San Francisco, Morgan Kaufmann, 2002.

J. Lafferty and G. Lebanon. Information diffusion kernels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. Cambridge, MA, MIT Press, 2003.

A. J. Smola and R. Kondor. Kernels and regularization on graphs. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the 16th Annual Conference on Learning Theory (COLT) and 7th Annual Workshop on Kernel Machines*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 144–158. Heidelberg, Springer Verlag, 2003.

A. J. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill–Posed Problems.* Washington, DC, Winston, 1977.

V. N. Vapnik. *The Nature of Statistical Learning Theory.* New York, Springer Verlag, 1995. ISBN 0-387-94559-8.

J.-P. Vert and M. Kanehisa. Extracting active pathways from gene expression data. *Bioinformatics*, 19:238ii–234ii, 2003a.

J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 1425–1432. Cambridge, MA, MIT Press, 2003b.