Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 5: Kernel Methods

Input and Feature Space

For mining and analysis, it is important to find a suitable data representation. For example, for complex data such as text, sequences, images, and so on, we must typically extract or construct a set of attributes or features, so that we can represent the data instances as multivariate vectors.

Given a data instance $\boldsymbol{x}$ (e.g., a sequence), we need to find a mapping ϕ, so that $\phi(\boldsymbol{x})$ is the vector representation of $\boldsymbol{x}$.

Even when the input data is a numeric data matrix a nonlinear mapping ϕ may be used to discover nonlinear relationships.

The term *input space* refers to the data space for the input data $\boldsymbol{x}$ and *feature space* refers to the space of mapped vectors $\phi(\boldsymbol{x})$.

Sequence-based Features

Consider a dataset of DNA sequences over the alphabet $\Sigma = \{A, C, G, T\}$.

One simple feature space is to represent each sequence in terms of the probability distribution over symbols in Σ. That is, given a sequence $\boldsymbol{x}$ with length $|\boldsymbol{x}| = m$, the mapping into feature space is given as

$$\phi(\boldsymbol{x}) = \{P(A), P(C), P(G), P(T)\}$$

where $P(s) = \frac{n_s}{m}$ is the probability of observing symbol $s \in \Sigma$, and n_s is the number of times s appears in sequence $\boldsymbol{x}$.

For example, if $\boldsymbol{x} = ACAGCAGTA$, with $m = |\boldsymbol{x}| = 9$, since A occurs four times, C and G occur twice, and T occurs once, we have

$$\phi(\boldsymbol{x}) = (4/9, 2/9, 2/9, 1/9) = (0.44, 0.22, 0.22, 0.11)$$

We can compute larger feature spaces by considering, for example, the probability distribution over all substrings or words of size up to k over the alphabet Σ.

Nonlinear Features

Consider the mapping ϕ that takes as input a vector $\boldsymbol{x} = (x_1, x_2)^T \in \mathbb{R}^2$ and maps it to a "quadratic" feature space via the nonlinear mapping

$$\phi(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T \in \mathbb{R}^3$$

For example, the point $\boldsymbol{x} = (5.9, 3)^T$ is mapped to the vector

$$\phi(\boldsymbol{x}) = (5.9^2, 3^2, \sqrt{2} \cdot 5.9 \cdot 3)^T = (34.81, 9, 25.03)^T$$

We can then apply well-known linear analysis methods in the feature space.

Kernel Method

Let $\mathcal{I}$ denote the input space, which can comprise any arbitrary set of objects, and let $\boldsymbol{D} = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathcal{I}$ be a dataset comprising n objects in the input space. Let $\phi: \mathcal{I} \to \mathcal{F}$ be a mapping from the input space $\mathcal{I}$ to the feature space $\mathcal{F}$.

Kernel methods avoid explicitly transforming each point $\boldsymbol{x}$ in the input space into the mapped point $\phi(\boldsymbol{x})$ in the feature space. Instead, the input objects are represented via their pairwise similarity values comprising the $n \times n$ *kernel matrix*, defined as

$$\boldsymbol{K} = \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n, \boldsymbol{x}_1) & K(\boldsymbol{x}_n, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

$K: \mathcal{I} \times \mathcal{I} \to \mathbb{R}$ is a *kernel function* on any two points in input space, which should satisfy the condition

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

Intuitively, we need to be able to compute the value of the dot product using the original input representation $\boldsymbol{x}$, without having recourse to the mapping $\phi(\boldsymbol{x})$.

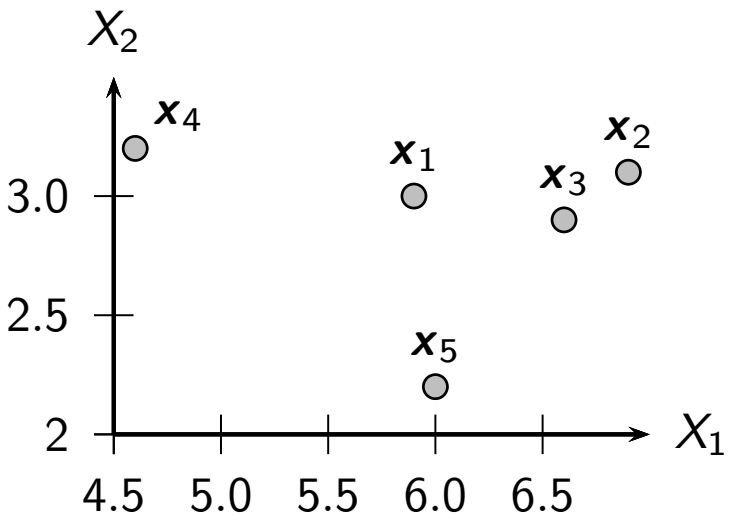
Linear Kernel

Let $\phi(\boldsymbol{x}) \rightarrow \boldsymbol{x}$ be the *identity kernel*. This leads to the *linear kernel*, which is simply the dot product between two input vectors:

$$\phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y} = K(\boldsymbol{x}, \boldsymbol{y})$$

For example, if $\boldsymbol{x}_1 = \begin{pmatrix} 5.9 & 3 \end{pmatrix}^T$ and $\boldsymbol{x}_2 = \begin{pmatrix} 6.9 & 3.1 \end{pmatrix}^T$, then we have

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1^T \boldsymbol{x}_2 = 5.9 \times 6.9 + 3 \times 3.1 = 40.71 + 9.3 = 50.01$$



$\boldsymbol{K}$	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$
$\boldsymbol{x}_1$	43.81	50.01	47.64	36.74	42.00
$\boldsymbol{x}_2$	50.01	57.22	54.53	41.66	48.22
$\boldsymbol{x}_3$	47.64	54.53	51.97	39.64	45.98
$\boldsymbol{x}_4$	36.74	41.66	39.64	31.40	34.64
$\boldsymbol{x}_5$	42.00	48.22	45.98	34.64	40.84

Kernel Trick

Many data mining methods can be *kernelized* that is, instead of mapping the input points into feature space, the data can be represented via the $n \times n$ kernel matrix $\boldsymbol{K}$, and all relevant analysis can be performed over $\boldsymbol{K}$.

This is done via the *kernel trick*, that is, show that the analysis task requires only dot products $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ in feature space, which can be replaced by the corresponding kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ that can be computed efficiently in input space.

Once the kernel matrix has been computed, we no longer even need the input points $\boldsymbol{x}_i$, as all operations involving only dot products in the feature space can be performed over the $n \times n$ kernel matrix $\boldsymbol{K}$.

Kernel Matrix

A function K is called a **positive semidefinite kernel** if and only if it is symmetric:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

and the corresponding kernel matrix $\boldsymbol{K}$ for any subset $\boldsymbol{D} \subset \mathcal{I}$ is positive semidefinite, that is,

$$\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} \geq 0, \text{ for all vectors } \boldsymbol{a} \in \mathbb{R}^n$$

which implies that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \text{ for all } a_i \in \mathbb{R}, i \in [1, n]$$

Dot Products and Positive Semi-definite Kernels

Positive Semidefinite Kernel

If $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ represents the dot product $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ in some feature space, then K is a positive semidefinite kernel.

First, K is symmetric since the dot product is symmetric, which also implies that $\boldsymbol{K}$ is symmetric.

Second, $\boldsymbol{K}$ is positive semidefinite because

$$\begin{aligned}
\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} &= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \\
&= \left(\sum_{i=1}^{n} a_i \phi(\boldsymbol{x}_i) \right)^T \left(\sum_{j=1}^{n} a_j \phi(\boldsymbol{x}_j) \right) \\
&= \left\| \sum_{i=1}^{n} a_i \phi(\boldsymbol{x}_i) \right\|^2 \geq 0
\end{aligned}$$

Empirical Kernel Map

We now show that if we are given a positive semidefinite kernel $K : \mathcal{I} \times \mathcal{I} \to \mathbb{R}$, then it corresponds to a dot product in some feature space $\mathcal{F}$.

Define the map ϕ as follows:

$$\phi(\boldsymbol{x}) = \Big((K(\boldsymbol{x}_1, \boldsymbol{x}), K(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x})\Big)^T \in \mathbb{R}^n$$

The *empirical kernel map* is defined as

$$\phi(\boldsymbol{x}) = \boldsymbol{K}^{-1/2} \cdot \Big((K(\boldsymbol{x}_1, \boldsymbol{x}), K(\boldsymbol{x}_2, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_n, \boldsymbol{x})\Big)^T \in \mathbb{R}^n$$

so that the dot product yields

$$\begin{aligned}
\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) &= \Big(\boldsymbol{K}^{-1/2}\, \boldsymbol{K}_i\Big)^T \Big(\boldsymbol{K}^{-1/2}\, \boldsymbol{K}_j\Big) \\
&= \boldsymbol{K}_i^T \left(\boldsymbol{K}^{-1/2} \boldsymbol{K}^{-1/2}\right) \boldsymbol{K}_j \\
&= \boldsymbol{K}_i^T\, \boldsymbol{K}^{-1}\, \boldsymbol{K}_j
\end{aligned}$$

where $\boldsymbol{K}_i$ is the ith column of $\boldsymbol{K}$. Over all pairs of mapped points, we have

$$\left\{\boldsymbol{K}_i^T \boldsymbol{K}^{-1}\, \boldsymbol{K}_j\right\}_{i,j=1}^{n} = \boldsymbol{K}\, \boldsymbol{K}^{-1}\, \boldsymbol{K} = \boldsymbol{K}$$

Data-specific Mercer Kernel Map

The Mercer kernel map also corresponds to a dot product in feature space.

Since $\boldsymbol{K}$ is a symmetric positive semidefinite matrix, it has real and non-negative eigenvalues. It can be decomposed as follows:

$$\boldsymbol{K} = \boldsymbol{U}\Lambda\boldsymbol{U}^T$$

where $\boldsymbol{U}$ is the orthonormal matrix of eigenvectors $\boldsymbol{u}_i = (u_{i1}, u_{i2}, \ldots, u_{in})^T \in \mathbb{R}^n$ (for $i = 1, \ldots, n$), and Λ is the diagonal matrix of eigenvalues, with both arranged in non-increasing order of the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$:
The Mercer map ϕ is given as

$$\phi(\boldsymbol{x}_i) = \sqrt{\Lambda}\boldsymbol{U}_i$$

where $\boldsymbol{U}_i$ is the ith row of $\boldsymbol{U}$.
The kernel value is simply the dot product between scaled rows of $\boldsymbol{U}$:

$$\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = \left(\sqrt{\Lambda}\boldsymbol{U}_i\right)^T\left(\sqrt{\Lambda}\boldsymbol{U}_j\right) = \boldsymbol{U}_i^T\Lambda\boldsymbol{U}_j$$

Polynomial Kernel

Polynomial kernels are of two types: homogeneous or inhomogeneous.

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. The (inhomogeneous) *polynomial kernel* is defined as

$$K_q(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^q$$

where q is the degree of the polynomial, and $c \geq 0$ is some constant. When $c = 0$ we obtain the homogeneous kernel, comprising only degree q terms. When $c > 0$, the feature space is spanned by all products of at most q attributes.
This can be seen from the binomial expansion

$$K_q(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^q = \sum_{k=1}^{q} \binom{q}{k} c^{q-k} \left(\boldsymbol{x}^T \boldsymbol{y}\right)^k$$

The most typical cases are the *linear* (with $q = 1$) and *quadratic* (with $q = 2$) kernels, given as

$$K_1(\boldsymbol{x}, \boldsymbol{y}) = c + \boldsymbol{x}^T \boldsymbol{y}$$

$$K_2(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}^T \boldsymbol{y})^2$$

Gaussian Kernel

The Gaussian kernel, also called the Gaussian radial basis function (RBF) kernel, is defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right\}$$

where $\sigma > 0$ is the spread parameter that plays the same role as the standard deviation in a normal density function.

Note that $K(\boldsymbol{x}, \boldsymbol{x}) = 1$, and further that the kernel value is inversely related to the distance between the two points $\boldsymbol{x}$ and $\boldsymbol{y}$.

A feature space for the Gaussian kernel has infinite dimensionality.

Basic Kernel Operations in Feature Space

Basic data analysis tasks that can be performed solely via kernels, without instantiating $\phi(\boldsymbol{x})$.

Norm of a Point: We can compute the norm of a point $\phi(\boldsymbol{x})$ in feature space as follows:

$$\|\phi(\boldsymbol{x})\|^2 = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x})$$

which implies that $\|\phi(\boldsymbol{x})\| = \sqrt{K(\boldsymbol{x}, \boldsymbol{x})}$.

Distance between Points: The distance between $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ is

$$\begin{aligned} \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2 &= \|\phi(\boldsymbol{x}_i)\|^2 + \|\phi(\boldsymbol{x}_j)\|^2 - 2\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \\ &= K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

which implies that

$$\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\| = \sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j)}$$

Basic Kernel Operations in Feature Space

Kernel Value as Similarity: We can rearrange the terms in

$$\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2 = K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

to obtain

$$\frac{1}{2}\left(\|\phi(\boldsymbol{x}_i)\|^2 + \|\phi(\boldsymbol{x}_j)\|^2 - \|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|^2\right) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

The more the distance $\|\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)\|$ between the two points in feature space, the less the kernel value, that is, the less the similarity.

Mean in Feature Space: The mean of the points in feature space is given as $\boldsymbol{\mu}_\phi = 1/n \sum_{i=1}^n \phi(\boldsymbol{x}_i)$. Thus, we cannot compute it explicitly. However, the the squared norm of the mean is:

$$\|\boldsymbol{\mu}_\phi\|^2 = \boldsymbol{\mu}_\phi^T \boldsymbol{\mu}_\phi = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

The squared norm of the mean in feature space is simply the average of the values in the kernel matrix $\boldsymbol{K}$.

Basic Kernel Operations in Feature Space

Total Variance in Feature Space: The total variance in feature space is obtained by taking the average squared deviation of points from the mean in feature space:

$$\sigma_\phi^2 = \frac{1}{n}\sum_{i=1}^{n}\|\phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi\|^2 = \frac{1}{n}\sum_{i=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_i) - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

Centering in Feature Space We can center each point in feature space by subtracting the mean from it, as follows:

$$\hat{\phi}(\boldsymbol{x}_i) = \phi(\boldsymbol{x}_i) - \boldsymbol{\mu}_\phi$$

The kernel between centered points is given as

$$\begin{aligned}\hat{K}(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \hat{\phi}(\boldsymbol{x}_i)^T\hat{\phi}(\boldsymbol{x}_j)\\ &= K(\boldsymbol{x}_i,\boldsymbol{x}_j) - \frac{1}{n}\sum_{k=1}^{n}K(\boldsymbol{x}_i,\boldsymbol{x}_k) - \frac{1}{n}\sum_{k=1}^{n}K(\boldsymbol{x}_j,\boldsymbol{x}_k) + \frac{1}{n^2}\sum_{a=1}^{n}\sum_{b=1}^{n}K(\boldsymbol{x}_a,\boldsymbol{x}_b)\end{aligned}$$

More compactly, we have:

$$\hat{\boldsymbol{K}} = \left(\boldsymbol{I} - \frac{1}{n}1_{n\times n}\right)\boldsymbol{K}\left(\boldsymbol{I} - \frac{1}{n}1_{n\times n}\right)$$

where $1_{n\times n}$ is the $n \times n$ matrix of ones.

Basic Kernel Operations in Feature Space

Normalizing in Feature Space: The dot product between normalized points in feature space corresponds to the cosine of the angle between them

$$\phi_n(\boldsymbol{x}_i)^T \phi_n(\boldsymbol{x}_j) = \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i)\| \cdot \|\phi(\boldsymbol{x}_j)\|} = \cos\theta$$

If the mapped points are both centered and normalized, then a dot product corresponds to the correlation between the two points in feature space.

The normalized kernel matrix, $\boldsymbol{K}_n$, can be computed using only the kernel function K, as

$$\boldsymbol{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)}{\|\phi(\boldsymbol{x}_i)\| \cdot \|\phi(\boldsymbol{x}_j)\|} = \frac{K(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{K(\boldsymbol{x}_i, \boldsymbol{x}_i) \cdot K(\boldsymbol{x}_j, \boldsymbol{x}_j)}}$$

$\boldsymbol{K}_n$ has all diagonal elements as 1.

Spectrum Kernel for Strings

Given alphabet Σ, the l-spectrum feature map is the mapping $\phi: \Sigma^* \to \mathbb{R}^{|\Sigma|^l}$ from the set of substrings over Σ to the $|\Sigma|^l$-dimensional space representing the number of occurrences of all possible substrings of length l, defined as

$$\phi(\boldsymbol{x}) = \Big(\cdots, \#(\alpha), \cdots\Big)^T_{\alpha \in \Sigma^l}$$

where $\#(\alpha)$ is the number of occurrences of the l-length string α in $\boldsymbol{x}$.

The (full) spectrum map considers all lengths from $l = 0$ to $l = \infty$, leading to an infinite dimensional feature map $\phi: \Sigma^* \to \mathbb{R}^\infty$:

$$\phi(\boldsymbol{x}) = \Big(\cdots, \#(\alpha), \cdots\Big)^T_{\alpha \in \Sigma^*}$$

where $\#(\alpha)$ is the number of occurrences of the string α in $\boldsymbol{x}$.

The (l-)spectrum kernel between two strings $\boldsymbol{x}_i, \boldsymbol{x}_j$ is simply the dot product between their (l-)spectrum maps:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

The (full) spectrum kernel can be computed efficiently via suffix trees in $O(n+m)$ time for two strings of length n and m.

Diffusion Kernels on Graph Nodes

Let $\boldsymbol{S}$ be some symmetric similarity matrix between nodes of a graph $G = (V, E)$. For instance, $\boldsymbol{S}$ can be the (weighted) adjacency matrix $\boldsymbol{A}$ or the Laplacian matrix $\boldsymbol{L} = \boldsymbol{A} - \Delta$ (or its negation), where Δ is the degree matrix for an undirected graph G, defined as $\Delta(i,i) = d_i$ and $\Delta(i,j) = 0$ for all $i \neq j$, and d_i is the degree of node i.

Power Kernels: Summing up the product of the base similarities over all l-length paths between two nodes, we obtain the l-length similarity matrix $\boldsymbol{S}^{(l)}$, which is simply the lth power of $\boldsymbol{S}$, that is,

$$\boldsymbol{S}^{(l)} = \boldsymbol{S}^l$$

Even path lengths lead to positive semidefinite kernels, but odd path lengths are not guaranteed to do so, unless the base matrix $\boldsymbol{S}$ is itself a positive semidefinite matrix.

Power kernel $\boldsymbol{K}$ can be obtained via the eigen-decomposition of $\boldsymbol{S}^l$:

$$\boldsymbol{K} = \boldsymbol{S}^l = \left(\boldsymbol{U}\Lambda\boldsymbol{U}^T\right)^l = \boldsymbol{U}\left(\Lambda^l\right)\boldsymbol{U}^T$$

Exponential Diffusion Kernel

The exponential diffusion kernel we can obtain a new kernel between nodes of a graph by paths of all possible lengths, but damps the contribution of longer paths

$$\begin{aligned}\boldsymbol{K} &= \sum_{l=0}^{\infty} \frac{1}{l!}\beta^l \boldsymbol{S}^l \\ &= \boldsymbol{I} + \beta\boldsymbol{S} + \frac{1}{2!}\beta^2\boldsymbol{S}^2 + \frac{1}{3!}\beta^3\boldsymbol{S}^3 + \cdots \\ &= \exp\{\beta\boldsymbol{S}\}\end{aligned}$$

where β is a damping factor, and $\exp\{\beta\boldsymbol{S}\}$ is the matrix exponential. The series on the right hand side above converges for all $\beta \geq 0$.

Substituting $\boldsymbol{S} = \boldsymbol{U}\Lambda\boldsymbol{U}^T$ the kernel can be computed as

$$\begin{aligned}\boldsymbol{K} &= \boldsymbol{I} + \beta\boldsymbol{S} + \frac{1}{2!}\beta^2\boldsymbol{S}^2 + \cdots \\ &= \boldsymbol{U}\begin{pmatrix} \exp\{\beta\lambda_1\} & 0 & \cdots & 0 \\ 0 & \exp\{\beta\lambda_2\} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \exp\{\beta\lambda_n\} \end{pmatrix}\boldsymbol{U}^T\end{aligned}$$

where λ_i is an eigenvalue of $\boldsymbol{S}$.

Von Neumann Diffusion Kernel

The *von Neumann diffusion kernel* is defined as

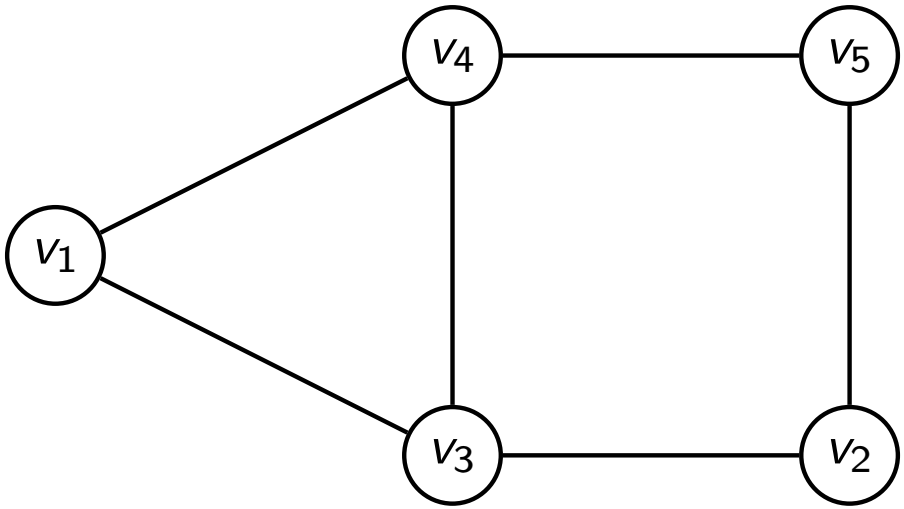
$$\boldsymbol{K} = \sum_{l=0}^{\infty} \beta^l \boldsymbol{S}^l$$

where $\beta \geq 0$. Expanding and rearranging the terms, we obtain

$$\boldsymbol{K} = (\boldsymbol{I} - \beta \boldsymbol{S})^{-1}$$

The kernel is guaranteed to be positive semidefinite if $|\beta| < 1/\rho(\boldsymbol{S})$, where $\rho(\boldsymbol{S}) = \max_i\{|\lambda_i|\}$ is called the *spectral radius* of $\boldsymbol{S}$, defined as the largest eigenvalue of $\boldsymbol{S}$ in absolute value.

Graph Diffusion Kernel: Example



Adjacency and degree matrices are given as

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \qquad \Delta = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Graph Diffusion Kernel: Example

Let the base similarity matrix $\boldsymbol{S}$ be the negated Laplacian matrix

$$\boldsymbol{S} = -\boldsymbol{L} = \boldsymbol{A} - \boldsymbol{D} = \begin{pmatrix} -2 & 0 & 1 & 1 & 0 \\ 0 & -2 & 1 & 0 & 1 \\ 1 & 1 & -3 & 1 & 0 \\ 1 & 0 & 1 & -3 & 1 \\ 0 & 1 & 0 & 1 & -2 \end{pmatrix}$$

The eigenvalues of $\boldsymbol{S}$ are as follows:

$$\lambda_1 = 0 \qquad \lambda_2 = -1.38 \qquad \lambda_3 = -2.38 \qquad \lambda_4 = -3.62 \qquad \lambda_5 = -4.62$$

and the eigenvectors of $\boldsymbol{S}$ are

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \boldsymbol{u}_4 & \boldsymbol{u}_5 \\ \hline 0.45 & -0.63 & 0.00 & 0.63 & 0.00 \\ 0.45 & 0.51 & -0.60 & 0.20 & -0.37 \\ 0.45 & -0.20 & -0.37 & -0.51 & 0.60 \\ 0.45 & -0.20 & 0.37 & -0.51 & -0.60 \\ 0.45 & 0.51 & 0.60 & 0.20 & 0.37 \end{pmatrix}$$

Graph Diffusion Kernel: Example

Assuming $\beta = 0.2$, the exponential diffusion kernel matrix is given as

$$\boldsymbol{K} = \exp\{0.2\boldsymbol{S}\} = \boldsymbol{U}\begin{pmatrix} \exp\{0.2\lambda_1\} & 0 & \cdots & 0 \\ 0 & \exp\{0.2\lambda_2\} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \exp\{0.2\lambda_n\} \end{pmatrix}\boldsymbol{U}^T$$

$$= \begin{pmatrix} 0.70 & 0.01 & 0.14 & 0.14 & 0.01 \\ 0.01 & 0.70 & 0.13 & 0.03 & 0.14 \\ 0.14 & 0.13 & 0.59 & 0.13 & 0.03 \\ 0.14 & 0.03 & 0.13 & 0.59 & 0.13 \\ 0.01 & 0.14 & 0.03 & 0.13 & 0.70 \end{pmatrix}$$

Assuming $\beta = 0.2$, the von Neumann kernel is given as

$$\boldsymbol{K} = \boldsymbol{U}(\boldsymbol{I} - 0.2\Lambda)^{-1}\boldsymbol{U}^T = \begin{pmatrix} 0.75 & 0.02 & 0.11 & 0.11 & 0.02 \\ 0.02 & 0.74 & 0.10 & 0.03 & 0.11 \\ 0.11 & 0.10 & 0.66 & 0.10 & 0.03 \\ 0.11 & 0.03 & 0.10 & 0.66 & 0.10 \\ 0.02 & 0.11 & 0.03 & 0.10 & 0.74 \end{pmatrix}$$

Data Mining and Machine Learning: Fundamental Concepts and Algorithms

dataminingbook.info

Mohammed J. Zaki[1] Wagner Meira Jr.[2]

[1]Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

[2]Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 5: Kernel Methods