STATISTICS AND DATA SCIENCE 365 / 665
INTERMEDIATE MACHINE LEARNING

Notes on Mercer Kernels

Mercer kernels and reproducing kernel Hilbert spaces (RKHS) have been an active area in machine learning for many years, with researchers inventing many new uses of this technique that goes back at least to the early 1970s. In these notes we introduce some of the basic theory of reproducing kernel Hilbert spaces, including the construction of the reproducing kernel Hilbert space (RKHS) associated with a kernel.

Note: This material is more advanced than what we present in class. It's offered as a complement and to give further detail to what is presented in class; students are not responsible for the parts that are not discussed in lecture.

1. Introduction

Mercer kernels and Reproducing kernel Hilbert spaces (RKHS) are concepts that arise in a number of places. Roughly speaking, a kernel $K(x, y)$ is a bivariate function that, in some sense, measures the similarity between x and y. An example is $K(x, y) = e^{-||x-y||^2}$. The kernel K can be used to construct a class of functions $\mathcal{H}$, called the RKHS generated by K. This class of functions is typically a class of smooth, well-behaved functions. Here are two motivating examples.

Example 1.1. [Nonparametric regression] We observe $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. One way to estimate the regression function $m(x) = \mathbb{E}(Y|X = x)$ is to find m to minimize $\sum_{i=1}^n (y_i - m(x_i))^2$. To avoid overfitting, we need to restrict m to lie in some class of functions $\mathcal{H}$. Thus we define $\widehat{m}$ to be the function in $\mathcal{H}$ that minimizes $\sum_{i=1}^n (y_i - m(x_i))^2$. If we take $\mathcal{H}$ to be a RKHS generated by some kernel K, then, as we shall see, the estimator $\widehat{m}$ takes on a simple form and has good theoretical properties.

Example 1.2. [Kernelized Classification] We observe $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. A linear classifier has the form $h(x) = \text{sign}(h(x))$ where $h(x) = \langle \beta, x \rangle = \beta^T x$. Many algorithms for finding a good linear classifier are functions of the inner products $\langle x_i, x_j \rangle$. We shall see that if we replace $\langle x_i, x_j \rangle$ with $K(x_i, x_j)$ we get a nonlinear classifier that can be more flexible than the linear one. The kernel K generates a RKHS $\mathcal{H}$ and we can view the new classifier as a linear classifier in the space $\mathcal{H}$.

These two examples, which we will deal with in more detail later, illustrate the two main reasons for introducing kernels and RKHS. First, they are a convenient way to define classes of functions. Second, the process of replacing inner products with kernels—called kernelization— often leads to useful generalizations of existing methods.

2. Basic concepts of reproducing kernel Hilbert spaces

In functional analysis, a reproducing kernel Hilbert space is a Hilbert space of functions in which the *point evaluation functional* is bounded and continuous. Equivalently, they are spaces that can be defined by reproducing kernels. This sounds very abstract, but the basic ideas are simple and are described in detail in this chapter.

2.1. Hilbert spaces and the evaluation functional

A Hilbert space is a complete inner product space. We will see that a *reproducing kernel Hilbert space* (*RKHS*) is a Hilbert space with extra structure that makes it very useful for statistics and machine learning.

An example of a Hilbert space is $\mathcal{L}_2[0,1] = \left\{ f : [0,1] \to \mathbb{R} : \int f^2 < \infty \right\}$ endowed with the inner product $\langle f, g \rangle = \int f(x)g(x)dx$. The corresponding norm is $\|f\| = \sqrt{\langle f, f \rangle} = \sqrt{\int f^2(x)dx}$. We write $f_n \to f$ to mean that $\|f_n - f\| \to 0$ as $n \to \infty$.

The evaluation functional $\delta_x : \mathcal{H} \to \mathbb{R}$ assigns a real number to each function $f \in \mathcal{H}$. It is defined by $\delta_x f = f(x)$. In general, the evaluation functional is not continuous. This means we can have $f_n \to f$ but $\delta_x f_n$ does not converge to $\delta_x f$. For example, let $f(x) = 0$ and $f_n(x) = \sqrt{n} I(x < 1/n^2)$. Then $\|f_n - f\| = 1/\sqrt{n} \to 0$. But $\delta_0 f_n = \sqrt{n}$ which does not converge to $\delta_0 f = 0$. Intuitively, this is because a general Hilbert space can contain very unsmooth functions. We shall see that RKHS are Hilbert spaces where the evaluation functional is bounded and continuous. Intuitively, this means that the functions in the space are well-behaved. More specifically, δ_x is bounded means there exists a constant $M > 0$ such that

$$\forall f \in \mathcal{H}, \ \ |\delta_x(f)| \leq M\|f\|. \tag{1}$$

This definition is quite technical. Its importance comes from the Riesz representation theorem, which states that if ϕ is a bounded linear functional on a Hilbert space $\mathcal{H}$, then there exists a unique element $g \in \mathcal{H}$ such that $\phi(f) = \langle f, g \rangle$ for any $f \in \mathcal{H}$. Applying this theorem on the bounded linear evaluation functional δ_x, we find that for any $f \in \mathcal{H}$, there is a unique element $K_x \in \mathcal{H}$ such that

$$\delta_x(f) = f(x) = \langle f, K_x \rangle. \tag{2}$$

In this way, we can define a bivariate function $K(x, y) = \langle K_x, K_y \rangle$ where K_x and K_y are respectively the unique representatives of δ_x and δ_y. In the next section, we will show that $K(x, y)$ is exactly the reproducing kernel for the Hilbert space $\mathcal{H}$.

2.2. Reproducing kernel Hilbert spaces (RKHS)

A RKHS is defined by a *Mercer kernel*. A Mercer kernel $K(x, y)$ is a function of two variables that is symmetric and positive definite. This means that, for any function f,

$$\int \int K(x, y) f(x) f(y) dx\, dy \geq 0.$$

This is a generalization of the notion of a positive definite matrix: $x^T Ax \geq 0$ for each x). One important Mercer kernel is the Gaussian kernel

$$K(x, y) = e^{-\frac{||x-y||^2}{\sigma^2}}. \tag{3}$$

Let $\mathcal{X}$ be the input space of x. Given a kernel K, let $K_x(\cdot)$ be the function obtained by fixing the first coordinate. That is, $K_x(y) = K(x, y)$. For the Gaussian kernel, K_x is an unnormalized Gaussian density, centered at x. We can create functions by taking finitely linear combinations of the kernels: $f(x) = \sum_{j=1}^{k} \alpha_j K_{x_j}(x)$. Let $\mathcal{H}_0$ denote all such functions:

$$\mathcal{H}_0 = \left\{ f : \sum_{j=1}^{k} \alpha_j K_{x_j} \ \ \forall k \in \mathbb{N}_+ \text{ and } x_j \in \mathcal{X} \right\}.$$

Given two such functions $f(x) = \sum_{j=1}^{k} \alpha_j K_{x_j}(x)$ and $g(x) = \sum_{j=1}^{m} \beta_j K_{y_j}(x)$ we define an inner product

$$\langle f, g \rangle_K = \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_i \beta_j K(x_i, y_j). \tag{4}$$

In general, f (and g) might be representable in more than one way. We can check that $\langle f, g \rangle_K$ is independent of how f (or g) is represented. This is because

$$\langle f, g \rangle_K = \sum_{i=1}^{k} \sum_{j=1}^{m} \alpha_i \beta_j K(x_i, y_j) = \sum_{j=1}^{m} \beta_j\, f(y_j) = \sum_{i=1}^{k} \alpha_i\, g(x_i).$$

Therefore the inner product is independent of the particular way that we represent the functions f and g in $\mathcal{H}_0$.

The inner product defines a norm:

$$\|f\|_K = \sqrt{\langle f, f \rangle_K} = \sqrt{\sum_j \sum_k \alpha_j \alpha_k K(x_j, x_k)} = \sqrt{\alpha^T K \alpha}$$

where $\alpha = (\alpha_1, \ldots, \alpha_k)^T$ and K is the $k \times k$ matrix with $K_{jk} = K(x_j, x_k)$.

Reproducing Kernel Hilbert spaces (RKHS)

The completion of $\mathcal{H}_0$ with respect to $\|\cdot\|_K$ is denoted by $\mathcal{H}_K$ and is called the RKHS generated by K. Here K is called the *reproducing kernel* of the Hilbert space $\mathcal{H}_K$.

To verify that this is a well-defined Hilbert space, you should check that the following properties hold for any $f, g \in \mathcal{H}_K$:

$$\begin{aligned} \langle f, g\rangle_K &= \langle g, f\rangle_K \\ \langle cf + dg, h\rangle_K &= c\langle f, h\rangle_K + d\langle g, h\rangle_K \text{ with } c, d \in \mathbb{R} \\ \langle f, f\rangle_K &= 0 \text{ iff } f = 0. \end{aligned}$$

The last one is not obvious so let us verify it here. It is easy to see that $f = 0$ impies that $\langle f, f\rangle_K = 0$. Now we must show that $\langle f, f\rangle_K = 0$ implies that $f(x) = 0$. So suppose that $\langle f, f\rangle_K = 0$. Pick any $x \in \mathcal{X}$. We have

$$0 \leq f^2(x) = \langle f, K_x\rangle_K^2 = \langle f, K_x\rangle_K \langle f, K_x\rangle_K \leq \|f\|_K^2 \|K_x\|_K^2 = \langle f, f\rangle_K^2 \|K_x\|_K^2 = 0$$

where we used the Cauchy-Schwartz inequality $\langle f, K_x\rangle_K \leq \|f\|_K \|K_x\|_K$. So $0 \leq f^2(x) \leq 0$ which means that $f(x) = 0$.

2.3. The reproducing property

Why $\mathcal{H}_K$ is called reproducing kernel Hilbert space? What does the word "reproducing" mean? Recall that $f(x) = \sum_i K_{x_i}(x)$. Note the following crucial property:

$$\langle f, K_x\rangle_K = \sum_{j=1}^{k} \alpha_j K(x_j, x) = f(x). \tag{5}$$

This follows from the definition of $\langle f, g\rangle$ where we take $g = K_x$. This is called the reproducing property. It also implies that K_x is the *unique representer* of the evaluation functional δ_x.

Returning to the evaluation functional, suppose that $f_n \to f$. Then

$$\delta_x(f_n) = \langle f_n, K_x\rangle_K = f_n(x) \to f(x) = \langle f, K_x\rangle_K = \delta_x(f).$$

So the evaluation functional is continuous.

In fact, a Hilbert space is a RKHS if and only if the evaluation functionals are continuous.

3. Properties of reproducing kernel Hilbert spaces

We have introduced the definition of reproducing kernel Hilbert spaces. In this section we explain different properties of RKHS including viewing kernels as feature maps, and the Mercer's theorem.

3.1. Feature maps and kernel methods

Given a RKHS $\mathcal{H}_K$ with reproducing kernel K, recall that for each input $x \in \mathcal{X}$, its evaluation functional δ_x has unique representation $K_x \in \mathcal{H}_K$. We can define $\phi(x) = K_x$, this builds a mapping $J : \mathcal{X} \to \mathcal{H}_K$ that map a point in the input space $\mathcal{X}$ to the RKHS $\mathcal{H}_K$.

$$\forall x \in \mathcal{X}, \;\; J(x) = \phi(x) \equiv K_x \in \mathcal{H}_K. \tag{6}$$

If the RKHS is of finite dimension, functions in the RKHS are exactly the dual space of the Euclidean space of feature projections.

Kernel methods are a class of algorithms for statistical machine learning, the best known example of which is the support vector machine (SVM). The main idea of kernel methods is to map the data into a high dimensional (possibly infinite dimensional) *feature space* $\mathcal{H}_K$, where each coordinate corresponds to one feature of the original input $x \in \mathcal{X}$, effectively transforming the original data into a high dimensional feature $\phi(x) \in \mathcal{H}_K$. Using$\mathcal{H}_K$, a variety of methods can be used to find nonlinear relations in the data. An algorithm that makes use of this mapping without explicitly computing the high dimensional representation is said to perform *the kernel trick*.

3.2. Spectral decomposition and Mercer's theorem

Usually in kernel methods we don't work with the high dimensional (or possibly infinite dimensional) feature representation $\phi(x)$ explicitly. Instead, it is implicit, and the algorithm works with the kernel $K(x, x')$ directly. However, a feature space can always be expressed in terms of the eigenvector decomposition of the kernel.

This is most clear in the case where the data points come from a finite set. In this case we can think of the Gram matrix as completely determining the kernel. Suppose $\mathbb{K} = [K(x_i, x_j)]_{i,j}$ defines a symmetric, positive-definite matrix. Taking the eigenvector decomposition of $\mathbb{K}$ allows us to write

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

where Λ is a diagonal matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$ along the diagonal (the matrix $\mathbb{K}$ has rank n since it is positive definite). Now, define

$$\phi(x_i) = \Lambda^{\frac{1}{2}} U_i$$

where U_i is the $(n \times 1)$ column vector of the orthonormal matrix U. This gives us

$$K(x_i, x_j) = \left(\Lambda^{\frac{1}{2}} U_i\right)^T \Lambda^{\frac{1}{2}} U_j$$

showing that we can think of ϕ as the feature mapping corresponding to the kernel. Note that we require nonnegative spectrum since otherwise we would have

$$\|\phi(x_i)\|^2 = U_i^T \Lambda U_i = \lambda_i < 0$$

and we would not have an inner-product space.

Mercer's theorem is the generalization of this basic linear algebra to the general case—it's just a functional analysis result.

Mercer's theorem

Theorem 3.3. *Suppose that $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is symmetric and satisfies $\sup_{x,y} K(x,y) < \infty$, and let $L_K f = \int_{\mathcal{X}} K(\cdot, y)\, f(y)\, dy$, suppose that $L_K : \mathcal{L}_2(\mathcal{X}) \to \mathcal{L}_2(\mathcal{X})$ is positive semidefinite; thus,*

$$\int_{\mathcal{X}} \int_{\mathcal{X}} K(x,y)\, f(x)\, f(y)\, dx\, dy \geq 0$$

for any $f \in \mathcal{L}_2(\mathcal{X})$.
Then there exist eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots$ (all non-negative) and eigenfunctions $\psi_1, \psi_2, \cdots \in \mathcal{L}_2(\mathcal{X})$ of L_K, with

$$\int_{\mathcal{X}} K(x,y)\psi_i(y)\, dy = \lambda_i \psi_i(x),$$

and $\sum_i \lambda_i < \infty$ and $\sup_x \psi_i(x) < \infty$. Also, $K(x,y)$ takes the form:

$$K(x,y) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\, \psi_i(y), \tag{7}$$

where the convergence is absolute and uniform in x, y.

Such a kernel defines a *Mercer kernel*. This gives the mapping into feature space as

$$x \mapsto \phi(x) = \left(\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots \right)^T.$$

Let $\langle \cdot, \cdot \rangle$ be the inner product of $\mathcal{L}_2(\mathcal{X})$. Since ψ_i and ψ_j are eigenfunctions we have $\langle \psi_i, \psi_j \rangle = I(i = j)$. Using Mercer's theorem, we can expand a function $f \in \mathcal{H}_K$ either in terms of K or in terms of the basis $\psi_1, \psi_2, \ldots$:

$$f(x) = \sum_{i=1}^{n} \alpha_i' K(x_i, x) = \sum_{i=1}^{\infty} \alpha_i \psi_i(x), \tag{8}$$

$$g(x) = \sum_{j=1}^{n} \beta_j' K(x, x_j) = \sum_{j=1}^{\infty} \beta_j \psi_j(x). \tag{9}$$

Note that in terms of the spectral decomposition of a kernel, the inner product is given by

$$\begin{aligned}
\langle f, g\rangle_K &= \left\langle \sum_{i=1}^{\infty} \alpha_i \psi_i, \sum_{j=1}^{\infty} \beta_j \psi_j \right\rangle_K \\
&= \left\langle \sum_{i=1}^{\infty} \frac{\alpha_i}{\sqrt{\lambda_i}} \sqrt{\lambda_i}\psi_i, \sum_{j=1}^{\infty} \frac{\beta_j}{\sqrt{\lambda_j}} \sqrt{\lambda_j}\psi_j \right\rangle_K \\
&= \sum_{i=1}^{\infty} \frac{\alpha_i \beta_j}{\sqrt{\lambda_i \lambda_j}} \langle \sqrt{\lambda_i}\psi_i, \sqrt{\lambda_j}\psi_j \rangle_K \\
&= \sum_{i=1}^{\infty} \frac{\alpha_i\, \beta_i}{\lambda_i}
\end{aligned}$$

This form makes more clear in what sense the RKHS norm favors smooth functions. As the eigenvalues λ_i get smaller, the eigenfunctions ψ_i vary more quickly—they become less smooth. Thus, a function with small RKHS norm must have the weights α_i die off quickly, so that most of the support is on functions that are more smoothly varying over $\mathcal{X}$.

3.3. The representer theorem

The following elementary but important result shows that minimizing regularized risk functionals for a large family of loss function results in kernel classifiers or regressors of the form

$$f(x) = \sum_i \alpha_i\, K(x_i, x)$$

The result is attributed to Wahba (1990). We'll denote the output space by $\mathcal{Y}$.

Theorem 3.4. *[Representer Theorem] Let K be a Mercer kernel defined on the space $\mathcal{X}$ and $\mathcal{H}_K$ be its corresponding RKHS. Let $x_1, \ldots x_n$ be a finite set of data points of $\mathcal{X}$ and let $L : \mathbb{R}^{n+1} \to \mathbb{R}$ be any function that is strictly increasing with respect to its last argument. Let $g_{\text{reg}} : \mathbb{R} \to [0, \infty)$ be monotonically increasing. Then any f minimizing the regularized empirical risk*

$$L\big(f(x_1), \ldots, f(x_n), \|f\|_K\big) \tag{10}$$

can be represented in the form $f(\cdot) = \sum_{i=1}^{n} \alpha_i\, K(x_i, \cdot)$ for some $\alpha_i \in \mathbb{R}$.

Proof. Let $\phi(x) = K(x, \cdot)$. Given a set of points $\{x_1, x_2, \ldots, x_n\}$ we can decompose any $f \in \mathcal{H}_K$ as $f = \sum_{i=1}^{n} \alpha_i \phi(x_i) + v$, where

$$v \notin \text{span}(\phi(x_1), \ldots, \phi(x_n)),$$

that is, v lies in the orthogonal complement of the finite dimensional vector space spanned by the $\phi(x_i)$. Now, by the reproducing property we have that

$$f(x_j) = \langle f, \phi(x_j)\rangle_K = \Big\langle \sum_{i=1}^n \alpha_i \phi(x_i) + v, \phi(x_j)\Big\rangle_K = \Big\langle \sum_{i=1}^n \alpha_i \phi(x_i), \phi(x_j)\Big\rangle_K.$$

where the last equality follows since v is orthogonal to all of the $\phi(x_j)$. Thus, the first n arguments of $L\big(f(x_1), \ldots, f(x_n), \|f\|_K\big)$ is independent of v.

We further have

$$\|f\|_K^2 = \big\|\sum_{i=1}^n \alpha_i \phi(x_i) + v\big\|_K^2 = \big\|\sum_{i=1}^n \alpha_i \phi(x_i)\big\|_K^2 + \big\|v\big\|_K^2 \geq \big\|\sum_{i=1}^n \alpha_i \phi(x_i)\big\|_K^2.$$

Thus, the regularization term can only be minimized by taking $v = 0$. □

4. Examples of RKHSs

In this section, we show some examples of some reproducing kernel Hilbert spaces and their corresponding reproducing kernels.

Example 4.5. Let $\mathcal{H}$ be all functions f on $\mathbb{R}$ such that the support of the Fourier transform of f is contained in $[-a, a]$. Then

$$K(x, y) = \frac{\sin(a(y-x))}{a(y-x)} \text{ and } \langle f, g\rangle_K = \int_{-a}^{a} f(t)g(t)dt. \tag{11}$$

Example 4.6. Let $\mathcal{H}$ be all functions f on $(0, 1)$ such that

$$\int_0^1 (f^2(x) + (f'(x))^2)x^2 dx < \infty.$$

Then

$$K(x, y) = (xy)^{-1}\left(e^{-x}\sinh(y)I(0 < x \leq y) + e^{-y}\sinh(x)I(0 < y \leq x)\right)$$

and $\|f\|^2 = \int_0^1 (f^2(x) + (f'(x))^2)x^2 dx$.

Example 4.7. The Sobolev space of order m is (roughly speaking) the set of functions f such that $\int_{\mathcal{X}} (f^{(m)})^2 < \infty$. For $m = 2$ and $\mathcal{X} = [0, 1]$ the kernel is

$$K(x, y) = \begin{cases} 1 + xy + \frac{xy^2}{2} - \frac{y^3}{6} & 0 \leq y \leq x \leq 1 \\ 1 + xy + \frac{yx^2}{2} - \frac{x^3}{6} & 0 \leq x \leq y \leq 1 \end{cases}$$

and $\|f\|_K^2 = f^2(0) + f'(0)^2 + \int_0^1 (f''(x))^2 dx$.

5. Constructing kernels for complex data

In this section, we explain practical guidelines that can apply to the selection of a kernel given a dataset of structured objects.

5.1. Kernels from kernels

The positive-definite requirement for Mercer kernels is in general difficult to verify. The following basic results show how one can build up kernels in pieces. If $K_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are Mercer kernels then so are the following:

1. $K(x, y) = K_1(x, y) + K_2(x, y)$
2. $K(x, y) = c\, K_1(x, y) + K_2(x, y)$ for $c \in \mathbb{R}_+$
3. $K(x, y) = K_1(x, y) + c$ for $c \in \mathbb{R}_+$
4. $K(x, y) = K_1(x, y)\, K_2(x, y)$
5. $K(x, y) = f(x)\, f(y)$ for $f : \mathcal{X} \longrightarrow \mathbb{R}$
6. $K(x, y) = (K_1(x, y) + c)^d$ for $\theta_1 \in \mathbb{R}_+$ and $d \in \mathbb{N}$
7. $K(x, y) = \exp\left(K_1(x, y)/\sigma^2\right)$ for $\sigma \in \mathbb{R}$
8. $K(x, y) = \exp\left(-(K_1(x, x) - 2K_1(x, y) + K_1(y, y))/2\sigma^2\right)$
9. $K(x, y) = K_1(x, y)/\sqrt{K_1(x, x)\, K_1(y, y)}$.

This last fact shows that we can implicitly "normalize" kernels without going to the feature representation, since

$$\frac{K_1(x, y)}{\sqrt{K_1(x, x)\, K_1(y, y)}} = \frac{\langle \phi(x), \phi(y) \rangle_K}{\|\phi(x)\|_K \, \|\phi(y)\|_K} = \left\langle \frac{\phi(x)}{\|\phi(x)\|_K}, \frac{\phi(y)}{\|\phi(y)\|_K} \right\rangle_K.$$

5.2. Kernels for discrete data

Many learning problems can be naturally cast in terms of data on graphs; semisupervised learning methods are a key example. In such problems, the key concept is regularization of functions defined over a graph, to penalize functions that are not smooth on the graph. The central quantity in graph-based regularization is the spectral decomposition of the graph Laplacian, a matrix derived from the edge weights. The eigenvectors with small eigenvalues are smooth, and ideally represent large cluster structures within the data. The eigenvectors having large eigenvalues are rugged, and considered noise.

Functions are compared against these eigenvectors to determine their smoothness. But one can also alter the eigenvalues to emphasize different eigenvectors. Such changes, or spectral transforms, produce different regularizers.

5.2.1. The graph Laplacian

We are given a dataset $\{x_1, \ldots, x_n\}$, and form a graph $G = (V, E)$ where the vertices V are in one-to-one correspondence with the data points $x_1, \ldots, x_n$, and the edges E are represented by an $n \times n$ matrix $W = [W_{ij}]$. Entry W_{ij} is the edge weight between nodes i, j, with $W_{ij} = 0$ if i, j are not connected. The entries of W have to be non-negative and symmetric, but it is not necessary for W itself to be positive semidefinite. Let $D = [D_{ij}]$ be the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$ being the total weight on edges connected to node i. For example, the weights could be specified in terms of a smoothing kernel, $W_{ij} = K(x_i, x_j)$, where positive definiteness is not required. The *combinatorial graph Laplacian* is defined as $\mathcal{L} = D - W$, which is also called the unnormalized Laplacian. The *normalized graph Laplacian* is $\mathcal{L}^{\text{norm}} = D^{-1/2}\mathcal{L}D^{-1/2} = I - D^{-1/2}WD^{-1/2}$.

In graph-based semisupervised learning the Laplacian $\mathcal{L}$ (or $\mathcal{L}^{\text{norm}}$) is a central quantity. Let us denote the eigensystem of $\mathcal{L}$ by $\{\lambda, \phi\}$, $\lambda_1 \leq \ldots \leq \lambda_n$. Therefore the spectral decomposition of the graph Laplacian is given as $\mathcal{L} = \sum_{i=1}^{n} \lambda_i \phi_i \phi_i^T$. We refer to Chung (1997) for a discussion of the mathematical aspects of this decomposition, but here are two relevant properties:

Theorem 5.8. *The Laplacian $\mathcal{L}$ is positive semidefinite, i.e., $\lambda_i \geq 0$.*

Proof. It is not hard to show that for any function $f : 1, \ldots, n \to \mathbb{R}$,

$$f^T \mathcal{L} f = \frac{1}{2} \sum_{i,j} W_{ij} \left(f(i) - f(j)\right)^2 \geq 0 \tag{12}$$

where the inequality holds because W has non-negative entries. □

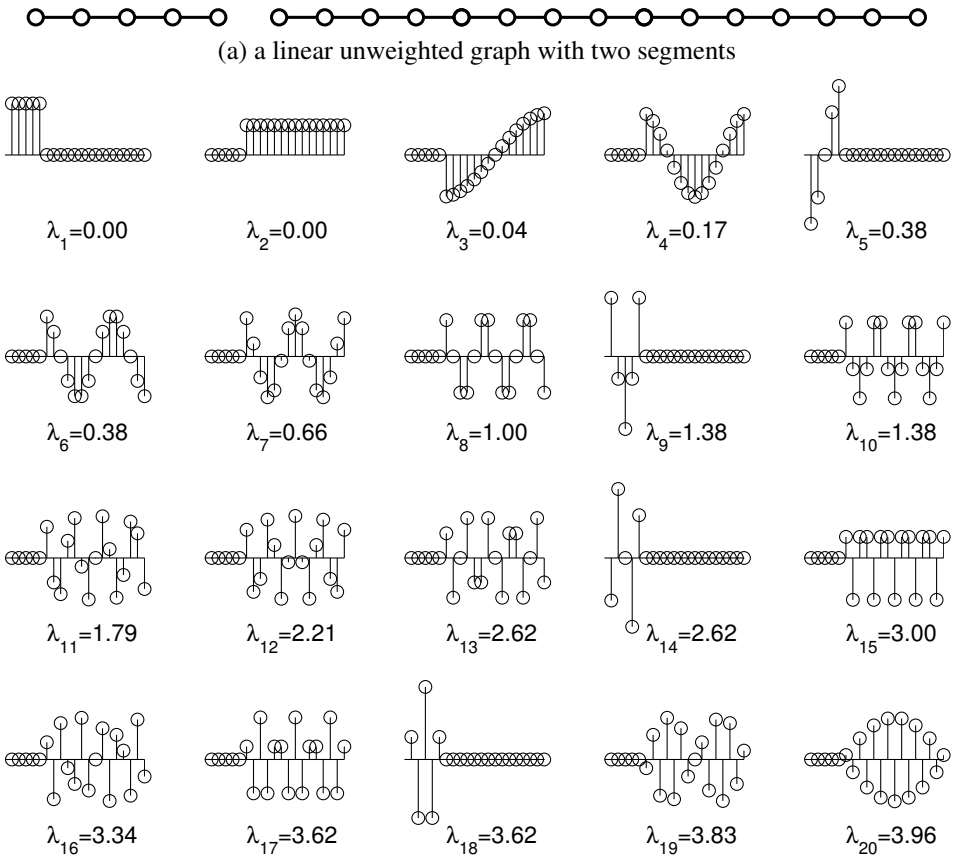
Equation (12) shows that $f^T \mathcal{L} f$ measures the *smoothness* of f on the graph, where a smaller value corresponds to a smoother function. Roughly speaking, f is smooth if $f(i) \approx f(j)$ for those pairs with large W_{ij}. This is sometimes informally expressed by saying that f varies slowly over the graph, or f follows the data manifold. In particular, the smoothness of an eigenvector is

$$\phi_i^T \mathcal{L} \phi_i = \lambda_i. \tag{13}$$

Thus, eigenvectors with smaller eigenvalues are smoother. Since $\{\phi_i\}$ forms a basis on $\mathbb{R}^n$, we can always write any function f as $f = \sum_{i=1}^{n} \alpha_i \phi_i$, $\alpha_i \in \mathbb{R}$ and the smoothness of f can be expressed as

$$f^T \mathcal{L} f = \sum_{i=1}^{n} \alpha_i^2 \lambda_i. \tag{14}$$

Theorem 5.9. *The graph G has k connected components if and only if $\lambda_i = 0$ for $i = 1, 2, \ldots, k$.* The corresponding eigenvectors $\phi_1, \ldots, \phi_k$ of $\mathcal{L}$ are constant on the nodes within the corresponding connected component, and zero elsewhere. Note λ_1 is always 0 for any graph.



(a) a linear unweighted graph with two segments

(b) the eigenvectors and eigenvalues of the Laplacian $\mathcal{L}$

FIG 1. *A simple graph and its Laplacian spectral decomposition. Note the eigenvectors become rougher with larger eigenvalues.*

Example 5.10. As an example, Figure 1(a) shows an unweighted graph ($W_{ij} = 1$ if there is an edge) consisting of two linear segments. The spectral decomposition of its Laplacian $\mathcal{L}$ is shown in (b). Note that the eigenvectors do indeed look smoother for small λ_i, and that the graph has two connected components.

5.3. Kernels by spectral transforms

The graph Laplacian can be used to define new kernels on the graph. To establish a link with the original graph, we can consider K having the form

$$\mathbb{K} = \sum_{i=1}^{n} \mu_i \phi_i \phi_i^T \tag{15}$$

where ϕ are the eigenvectors of the graph Laplacian $\mathcal{L}$, and $\mu_i \geq 0$ are the eigenvalues of $\mathbb{K}$. Since $\mathbb{K}$ is the non-negative sum of outer products, it is positive semidefinite, i.e., a kernel matrix. The

matrix $\mathbb{K}$ defines a reproducing kernel Hilbert space with norm

$$\|f\|_K^2 = \langle f, f \rangle_K = \sum_{i=1}^{n} \frac{\alpha_i^2}{\mu_i} \tag{16}$$

for a function $f = \sum_{i=1}^{n} \alpha_i \phi_i$.

In many learning algorithms, regularization is expressed as an increasing function of $\|f\|_K$; we want f to be penalized if it is not smooth with respect to the graph. Comparing the smoothness of f in equation (14) with equation (16), we find this can be achieved by making μ_i small if the Laplacian eigenvalue λ_i is large, and vice versa.

Smola and Kondor (2003) suggest a simple approach to creating a kernel K from the graph Laplacian. Define a *spectral transformation* function $r : \mathbb{R}_+ \to \mathbb{R}_+$ that is non-negative and decreasing. Set the kernel spectrum by $\mu_i = r(\lambda_i)$ to obtain the kernel

$$\mathbb{K} = \sum_{i=1}^{n} r(\lambda_i) \phi_i \phi_i^T \tag{17}$$

Note that since r is decreasing, a greater penalty is incurred if a function is not smooth with respect to the original graph, as measured by the eigenvectors of the graph Laplacian.

The transform r is often chosen from a parametric family. For example Chapelle et al. (2002) and Smola and Kondor (2003) list the following transformations on $\mathcal{L}^{\text{norm}}$

- regularized Laplacian: $r(\lambda) = \frac{1}{\lambda + \epsilon}$
- diffusion kernel: $r(\lambda) = \exp\left(-\frac{\sigma^2}{2}\lambda\right)$
- one step random walk: $r(\lambda) = (\alpha - \lambda)$ with $\alpha \geq 2$
- p-step random walk: $r(\lambda) = (\alpha - \lambda)^p$ with $\alpha \geq 2$
- inverse cosine: $r(\lambda) = \cos(\lambda \pi / 4)$
- step function: $r(\lambda) = 1$ if $\lambda \leq \lambda_{\text{cut}}$

Each has its own special interpretation. Of course there are many other natural choices for r. Although the general principle of equation (17) are appealing, it does not address the question of which parametric family to use.

6. Statistical machine learning in a RHKS

Kernel machines consist a family of learning algorithms that utilizes the *kernel trick*. This is a fairly general trick. In many algorithms you can replace $\langle x_i, x_j \rangle$ with $K(x_i, x_j)$ and get a nonlinear version of the algorithm. This is equivalent to replacing x with $\phi(x)$ and replacing $\langle x_i, x_j \rangle$ with $\langle \phi(x_i), \phi(x_j) \rangle_K$. However, $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_K$ and $K(x_i, x_j)$ is much easier to compute. In summary, by replacing $\langle x_i, x_j \rangle$ with $K(x_i, x_j)$ we turn a linear procedure into a nonlinear procedure without adding much computation. In the following we give some popular examples of kernel machines, including kernel ridge regression, kernel logistic regression, kernel principle component analysis kernel canonical correlation analysis, and kernel support vector machines.

6.1. Kernel ridge regression

The penalized empirical risk minimization approach to learning over an RKHS allows kernels to be used in several otherwise standard settings, resulting in simple nonparametric estimators.

Suppose that we consider squared error, with an RKHS norm penalty:

$$\sum_{i=1}^{n}\Big(y_i - f(x_i)\Big)^2 + \lambda\|f\|_K^2 \tag{18}$$

where $y_i \in \mathbb{R}$ and $f \in \mathcal{H}_K$ is in the reproducing kernel Hilbert space for a kernel K. Then the representer theorem tells us that we can consider the dual form $f(x) = \sum_{i=1}^{n}\alpha_i K(x_i, x)$ leading to the objective function

$$\sum_{i=1}^{n}\Big(y_i - \sum_{j=1}^{n}\alpha_j K(x_i, x_j)\Big)^2 + \lambda\sum_{i,j}\alpha_i\alpha_j K(x_i, x_j) = \|y - \mathbb{K}\alpha\|_2^2 + \lambda\alpha^T\mathbb{K}\alpha, \tag{19}$$

where $\mathbb{K}$ is the gram matrix. The solution can be given in closed form. Note that this is a kind of generalized ridge regression problem, since in dual form, the RKHS penalty looks like an ℓ_2 penalty. The normal equations are

$$-\mathbb{K}(y - \mathbb{K}\alpha) + \lambda\mathbb{K}\alpha = 0 \tag{20}$$

using the fact that $\mathbb{K} = \mathbb{K}^T$. Assuming that $\mathbb{K}$ is nonsingular, the solution is then given by

$$\widehat{\alpha} = (\mathbb{K}^2 + \lambda\mathbb{K})^{-1}\mathbb{K}y = (\mathbb{K} + \lambda I)^{-1}y \tag{21}$$

The estimated (smoothed) values $\widehat{f}(x_i)$ are then given by

$$\widehat{f} = \mathbb{K}\widehat{\alpha} = \mathbb{K}(\mathbb{K} + \lambda I)^{-1}y = (I + \lambda\mathbb{K}^{-1})^{-1}y. \tag{22}$$

In fact, smoothing splines are a special case of the kernel ridge regression.

6.2. Kernel logistic regression

A similar approach applies to logistic regression. Here the nonparametric model is

$$\mathbb{P}(Y = 1 \,|\, X) \;=\; \frac{\exp\big(f(x)\big)}{1 + \exp\big(f(x)\big)} \tag{23}$$

where $f \in \mathcal{H}_K$. Using a penalty term $\|f\|_K^2$, we define the penalized empirical risk minimization functional

$$J(f, \lambda) = \sum_{i=1}^{n}\Big[\log\big(1 + \exp\big(f(x_i)\big)\big) - y_i f(x_i)\Big] + \frac{\lambda}{2}\|f\|_K^2 \tag{24}$$

Using the representer theorem, we represent this in dual form leads to

$$J(\alpha, \lambda) = \sum_{i=1}^{n} \Big[\log\left(1 + \exp\left(\mathbb{K}_i^T \alpha\right)\right) - y_i \mathbb{K}_i^T \alpha \Big] + \frac{\lambda}{2} \alpha^T \mathbb{K} \alpha \tag{25}$$

$$= 1^T \log\left(1 + \exp(\mathbb{K}\alpha)\right) - y^T \mathbb{K}\alpha + \frac{\lambda}{2} \alpha^T K \alpha \tag{26}$$

To apply Newton's method, we compute the gradient

$$\mathbb{K}^T (p(\alpha) - y) + \lambda \mathbb{K}\alpha \tag{27}$$

where

$$p(\alpha) = \left(\frac{\exp(\mathbb{K}_i^T \alpha)}{1 + \exp(\mathbb{K}_i^T \alpha)} \right) \tag{28}$$

and the Hessian $\mathbb{K}^T W \mathbb{K} + \lambda \mathbb{K}$ with $W = \text{diag}(p(\alpha)_i (1 - p(\alpha)_i))$. The Newton updates are thus given by

$$\alpha^{\text{new}} = \alpha - \left(\nabla^2 J\right)^{-1} \nabla J \tag{29}$$

$$= \alpha - \left(\mathbb{K}^T W \mathbb{K} + \lambda \mathbb{K}\right)^{-1} \left(\mathbb{K}^T (p(\alpha) - y) + \lambda \mathbb{K}\alpha\right) \tag{30}$$

$$= \left(\mathbb{K}^T W \mathbb{K} + \lambda \mathbb{K}\right)^{-1} \mathbb{K}^T W (W^{-1}(y - p(\alpha)) + \mathbb{K}\alpha) \tag{31}$$

Note that neither the kernel regression nor the kernel logistic regression give us a *sparse* model. Typically the α_is will be nonzero for all of the data.

To get sparsity, it is possible to add an ℓ_1-penalty:

$$\widehat{\alpha} = \arg\min_{\alpha} \Big\{ J(\alpha, \lambda) + \mu \|\alpha\|_1 \Big\}. \tag{32}$$

We'll return to this type of regularization when we discuss sparsity.

6.3. Kernel support vector machines

Recall that the linear SVM was formulated as a penalized hinge-loss regression problem

$$J(\beta, \lambda) = \sum_{i=1}^{n} \left[1 - y_i(\beta^T x_i + \beta_0)\right]_+ + \frac{\lambda}{2} \|\beta\|_2^2. \tag{33}$$

If we assume the discriminant function $f(x)$ lies in a RKHS of a kernel K, we can directly write it into the following optimization:

$$J(f, \lambda) = \sum_{i=1}^{n} \left[1 - y_i f(x_i)\right]_+ + \frac{\lambda}{2} \|f\|_K^2 \tag{34}$$

or in terms of the dual variable representation by the representer theorem,

$$J(\alpha, \lambda) = \sum_{i=1}^{n} \left[1 - y_i(\mathbb{K}_i^T \alpha + \alpha_0)\right]_+ + \frac{\lambda}{2} \alpha^T \mathbb{K} \alpha. \tag{35}$$

Alternatively, recall that the dual problem for linear SVMs is

$$\max_{\alpha} \qquad \alpha^T 1 - \frac{1}{2} \alpha^T y^T \mathbb{G} y \alpha \tag{36}$$

$$\text{such that} \qquad 0 \leq \alpha \leq C. \tag{37}$$

Since $\mathbb{G} = \mathcal{X}\mathcal{X}^T$ only involves inner products, we can kernelize this by simply replacing $\mathbb{G}$ with $\mathbb{K}$.

6.4. Kernel principle component analysis

Principal components analysis is a commonly used technique for forming a low-dimensional representation of high dimensional data. It is typically used to extract important "features" from data such as images, EEGs, etc., which may then be used for visualization or further "downstream" processing. The basic idea is that in an appropriate basis, the largest principal directions have the highest variance; therefore, representing the data by projecting onto these directions can capture the main variation in the data.

Standard PCA takes data vectors $x_1, x_2, \ldots, x_n$ in $\mathbb{R}^d$ and finds the directions of maximal variance. It is assumed that the data are "centered" to have mean zero

$$\sum_{i=1}^{n} x_i = 0\,. \tag{38}$$

The matrix $\widehat{\Sigma} \in \mathbb{R}^{d \times d}$

$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} x_i x^T \tag{39}$$

can thus be thought of as the empirical covariance matrix of the data. The PCA technique is to diagonalize this matrix, obtaining eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \tag{40}$$

and then using the top eigenvectors to represent the data. The eigenvector v_1 with eigenvalue λ_1 is the first principal component, corresponding to the axis along which the data have greatest

variance. We have then for any eigenvector/eigenvector pair (v, λ) that

$$\begin{aligned} \lambda v &= \widehat{\Sigma} v && (41) \\ &= \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T v && (42) \\ &= \frac{1}{n} \sum_{i=1}^{n} \langle x_i, v \rangle x_i && (43) \\ &= \frac{1}{n\lambda} \sum_{i=1}^{n} \left\langle x_i, \widehat{\Sigma} v \right\rangle x_i && (44) \\ &= \sum_{i=1}^{n} \alpha_i x_i && (45) \end{aligned}$$

where $\alpha_i = x_i^T \widehat{\Sigma} v / (n\lambda)$. Since the principal components are linear combinations of the data, they are typically not sparse. To reduce variance and for consistency, it may be beneficial to obtain sparse vectors by thresholding or other means (These techniques will be discussed in the sparsity part of this book). In this chapter, we first discuss kernel PCA.

Suppose now that we have a "feature map"

$$x \mapsto \phi(x) \tag{46}$$

and want to carry out PCA in this new feature space. If we assume that the feature vectors are centered, then the empirical covariance matrix is formed:

$$\mathbb{C}_\phi = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \phi(x_i)^T \tag{47}$$

$$\tag{48}$$

After diagonalizing, we have as before that

$$\begin{aligned} \lambda v &= \mathbb{C}_\phi v && (49) \\ &= \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \phi(x_i)^T v && (50) \\ &= \frac{1}{n} \sum_{i=1}^{n} \langle \phi(x_i), v \rangle \phi(x_i) && (51) \\ &= \sum_{i=1}^{n} \alpha_i \phi(x_i) && (52) \end{aligned}$$

where

$$\alpha_i = \langle \phi(x_i), v \rangle = \frac{1}{\lambda} \langle \phi(x_i), \mathbb{C}_\phi v \rangle \tag{53}$$

Now, if ϕ is very high dimensional, diagonalizing directly to get the spectrum would be very costly. However, note that we can write

$$\begin{aligned}
\lambda \langle \phi(x_k), v \rangle &= \lambda \sum_{i=1}^{n} \alpha_i \langle \Phi(x_k), \phi(x_i) \rangle && (54)\\
&= \left\langle \phi(x_k), \mathbb{C}_\phi v \right\rangle && (55)\\
&= \left\langle \phi(x_k), \frac{1}{n} \sum_{j=1}^{n} \phi(x_j)\phi(x_j)^T v \right\rangle && (56)\\
&= \left\langle \phi(x_k), \frac{1}{n} \sum_{j=1}^{n} \phi(x_j)\phi(x_j)^T \sum_{i=1}^{n} \alpha_i \phi(x_i) \right\rangle && (57)\\
&= \frac{1}{n} \sum_{i=1}^{n} \alpha_i \left\langle \Phi(x_k), \sum_{j=1}^{n} \langle \phi(x_j), \phi(x_i) \rangle \, \phi(x_j) \right\rangle && (58)
\end{aligned}$$

In terms of the Gram matrix $\mathbb{K}_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$, this can be written in matrix form as

$$\lambda n \mathbb{K} \alpha = \mathbb{K}^2 \alpha. \quad (59)$$

Thus, we need to solve the kernel eigenvalue problem

$$n\lambda\alpha = \mathbb{K}\alpha \quad (60)$$

which requires diagonalizing only an $n \times n$ system. Normalizing the eigenvectors, $\langle v, v \rangle = 1$ leads to the condition $\lambda \langle \alpha, \alpha \rangle = 1$.

In order to compute the kernel PCA projection of a new test point x, it is necessary to project the feature vector $\phi(x)$ onto the principal direction v. This requires the evaluation of

$$\langle v, \phi(x) \rangle = \sum_{i=1}^{n} \alpha_i \langle \phi(x_i), \phi(x) \rangle = \sum_{i=1}^{n} \alpha_i K(x_i, x). \quad (61)$$

Thus, the entire procedure uses only the kernel evaluations $K(x, x_i)$ and never requires actual manipulation of feature vectors, which could be infinite dimensional. This is an instance of the *kernel trick*.

To complete the description of the algorithm, it is necessary to explain how to center the data in feature space using only kernel operations. This is accomplished by transforming the kernel according to

$$\widetilde{\mathbb{K}}_{ij} = \left(\mathbb{K} - 1_n\mathbb{K} - \mathbb{K}1_n + 1_n\mathbb{K}1_n\right)_{ij} \quad (62)$$

where

$$1_n = \frac{1}{n} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} = \frac{1}{n} 1 1^T \quad (63)$$

where 1 denotes the vector of all ones.

Kernel PCA

Given a Mercer kernel K and data $x_1, x_2, \ldots, x_n$

1. Center the kernel
2. Compute $\mathbb{K}_{ij} = K(x_i, x_j)$
3. Diagonalize $\mathbb{K}$
4. Normalize eigenvectors $\alpha^{(j)}$ so that $\left\langle \alpha^{(j)}, \alpha^{(j)} \right\rangle = \frac{1}{\lambda_j}$
5. Compute the projection of a test point x onto an eigenvector v_j by

$$\langle v_j, \phi(x) \rangle = \sum_{i=1}^{n} \alpha_i^{(j)} K(x_i, x) \tag{64}$$

Just as for standard PCA, this selects components of high variance, but in the feature space of the kernel. In addition, the "feature functions"

$$f_j(x) = \sum_{i=1}^{n} \alpha_i^{(j)} K(x_i, x) \tag{65}$$

are orthogonal and act as representative feature functions in the reproducing kernel Hilbert space of the kernel, with respect to the given data. Intuitively, these functions are smooth respect to the RKHS norm $\| \cdot \|_K$ among all those supported on the data.

6.5. Kernel canonical correlation analysis

Here's a use of Mercer kernels that is different from the classification and regression examples we have considered so far. Suppose that we want to assess whether two random variables X_1 and X_2 are independent. It's not sufficient that

$$\text{Cor}(X_1, X_2) = \frac{\text{Cov}(X_1, X_2)}{\sqrt{\text{Var}(X_1)}\sqrt{\text{Var}(X_2)}} = 0 \tag{66}$$

On the other hand, $X_1 \perp\!\!\!\perp X_2$ if and only if $\rho = \sup_{h_1, h_2 \in \mathcal{H}} \text{Cor}(h_1(X_1), h_2(X_2)) = 0$ for a suitably rich family of functions $\mathcal{H}$. In particular, this holds in case $h_1, h_2 \in \mathcal{H}_K$ for an appropriate RKHS $\mathcal{H}_K$. By the reproducing property, this can then be written as

$$\rho = \sup_{h_1, h_2 \in \mathcal{H}_K} \text{Cor}\big(\langle h_1, K(\cdot, X_1) \rangle_K , \langle h_2, K(\cdot, X_2) \rangle_K \big) = 0. \tag{67}$$

Recall the idea behind principal components. Given a random vector X, PCA tries to find a linear transformation under which the components of the transformed vector are uncorrelated. In canonical correlation analysis, given two random vectors X_1 and X_2, CCA attempts to find a *pair* of linear

transformations $\mathcal{T}_1$ and $\mathcal{T}_2$ so that under these transformations, each coordinate of $\mathcal{T}_1 X_1$ is correlated only with a single component of $\mathcal{T}_2 X_2$. CCA can be carried out sequentially, one component at a time, just as PCA.

We define

$$\begin{aligned}\rho(X_1, X_2) &= \sup_{\xi_1,\xi_2} \text{Cor}(\xi_1^T X_1, \xi_2^T X_2) && (68)\\ &= \sup_{\xi_1,\xi_2} \frac{\text{Cov}(\xi_1^T X_1, \xi_2^T X_2)}{\sqrt{\text{Var}(\xi_1^T X_1)}\sqrt{\text{Var}(\xi_2^T X_2)}} && (69)\\ &= \sup_{\xi_1,\xi_2} \frac{\xi_1^T \Sigma_{12} \xi_2^T}{\sqrt{\xi_1^T \Sigma_{11} \xi_1}\sqrt{\xi_2^T \Sigma_{22} \xi_2}} && (70)\end{aligned}$$

where

$$\Sigma = \text{Cov}(X_1, X_2) = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \tag{71}$$

is the covariance matrix. Normalizing so that

$$\xi_1^T \Sigma_{11} \xi_1 = \xi_2^T \Sigma_{22} \xi_2 = 1, \tag{72}$$

Using Lagrange multipliers, ξ_1 and ξ_2 can be solved through the following generalized eigenvalue problem:

$$\begin{pmatrix} 0 & \Sigma_{12} \\ \Sigma_{21} & 0 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \rho \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{22} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}. \tag{73}$$

Bach and Jordan (2003) kernelize this, considering two kernels K_1 and K_2 with features maps ϕ_1 and ϕ_2. We want to estimate

$$\rho = \max_{f_1, f_2 \in \mathcal{F}} \text{Cor}\Big(\langle \phi_1(X_1), f_1 \rangle_K , \langle \phi_2(X_2), f_s \rangle_K \Big) \tag{74}$$

Now, consider the sample version of this, where $x_{11}, x_{21}, \ldots, x_{n1}$ are sampled from the distribution of X_1 and $x_{12}, x_{22}, \ldots, x_{n2}$ are sampled from the distribution of X_2. Then, suppose that

$$f_1(x) = \sum_{i=1}^n \alpha_{i1} K(x_{i1}, x) + v_1 \;\text{ and }\; f_2(x) = \sum_{i=1}^n \alpha_{i2} K(x_{i2}, x) + v_2 \tag{75}$$

where $v_1, v_1 \in (\text{span}\{K(x_{1j}, \cdot), \ldots, K(x_{nj}, \cdot)\})^\perp$.

Then

$$\widehat{\text{Cov}}\left(f_1(X_1), f_2(X_2)\right) = \frac{1}{n} \alpha_1^T \mathbb{K}_1 \mathbb{K}_2 \alpha_2 \tag{76}$$

where $\mathbb{K}_1 = [K(x_{i1}, x_{j1})]$ and $\mathbb{K}_2 = [K(x_{i2}, x_{j2})]$ are the appropriate gram matrices. Similarly,

$$\widehat{\text{Var}}\left(f_1(X_1)\right) = \frac{1}{n} \alpha_1^T \mathbb{K}_1 \mathbb{K}_1 \alpha_1 \;\text{ and }\; \widehat{\text{Var}}\left(f_2(X_2)\right) = \frac{1}{n} \alpha_2^T \mathbb{K}_2 \mathbb{K}_2 \alpha_2. \tag{77}$$

Thus, the estimate of the kernel canonical correlation is

$$\widehat{\rho} = \max_{\alpha_1,\alpha_2} \frac{\alpha_1^T \mathbb{K}_1 \mathbb{K}_2 \alpha_2}{\sqrt{\alpha_1^T \mathbb{K}_1^2 \alpha_1}\sqrt{\alpha_2^T \mathbb{K}_2^2 \alpha_2}} \tag{78}$$

which is equivalent to solving a generalized eigenvector problem:

$$\begin{pmatrix} 0 & \mathbb{K}_1\mathbb{K}_2 \\ \mathbb{K}_2\mathbb{K}_1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \rho \begin{pmatrix} \mathbb{K}_1^2 & 0 \\ 0 & \mathbb{K}_2^2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}. \tag{79}$$

Unfortunately, this estimate will overfit. It is necessary to regularize it, leading to the modified generalized eigenvalue problem

$$\begin{pmatrix} 0 & \mathbb{K}_1\mathbb{K}_2 \\ \mathbb{K}_2\mathbb{K}_1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \rho \begin{pmatrix} (\mathbb{K}_1 + \lambda_n I)^2 & 0 \\ 0 & (\mathbb{K}_2 + \lambda_n I)^2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \tag{80}$$

for a suitable $\lambda_n \to 0$. This can be shown to lead to a consistent estimate of the kernel correlation coefficient.

7. Hidden tuning parameters

There are hidden tuning parameters in the RKHS. Consider the Gaussian kernel

$$K(x,y) = e^{-\frac{||x-y||^2}{\sigma^2}}.$$

For nonparametric regression we minimize $\sum_i (Y_i - m(X_i))^2$ subject to $||m||_K \leq L$. We control the bias variance tradeoff by doing cross-validation over L. But what about σ?

This parameter seems to get mostly ignored. Suppose we have a uniform distribution on a circle. The eigenfunctions of $K(x,y)$ are the sines and cosines. The eigenvalues λ_k die off like $(1/\sigma)^{2k}$. So σ affects the bias-variance tradeoff since it weights things towards lower order Fourier functions. In principle we can compensate for this by varying L. But clearly there is some interaction between L and σ. The practical effect is not well understood.

Now consider the polynomial kernel $K(x,y) = (1 + \langle x, y\rangle)^d$. This kernel has the same eigenfunctions but the eigenvalues decay at a polynomial rate depending on d. So there is an interaction between L, d and, the choice of kernel itself.

8. Kernel methods and the perceptron algorithm

The perceptron is a simple and elegant linear classification algorithm. The voted perceptron variant has stronger theoretical guarantees, and is often competitive with support vector machines.

Let $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^{d+1} \times \mathbb{R}$ be training data for a binary classification problem, where the outputs y_i are encoded as $y_i \in \{-1, +1\}$. A *linear classifier* in x is a classifier whose decision rule is of the form

$$f_w(x) = \text{sign}\big(\langle x, w\rangle\big). \tag{81}$$

The hyperplane decision surface is characterized by the normal vector $w \in \mathbb{R}^{d+1}$. As usual, we allow affine hyperplanes as decision surfaces by taking one component of each covariate x to be the constant 1.

The perceptron algorithm makes multiple passes through the data, testing whether the current data point is correctly classified. If not, the hyperplane is adjusted in the direction of that data point.

The Perceptron Algorithm

Initialize $w_0 \leftarrow 0$, $t \leftarrow 0$. Proceed by making (possibly) multiple passes through the training data:

- For the next example (x_i, y_i), if $y_i \langle w, x_i \rangle \leq 0$, update

$$t \leftarrow t + 1 \tag{82}$$
$$w_t = w_{t-1} + y_i x_i \tag{83}$$

- Stop after each example is correctly classified

Output: hyperplane with normal vector w_t.

It is simple to show that in case the data are linearly separable, the perceptron algorithm will output a separating hyperplane in finite time. Let $M = \max_i \|x_i\|^2$ (Euclidean norm) and suppose that there is a unit vector w^* with $y_i \langle w^*, x_i \rangle \geq \varepsilon$ for each i. Then

$$\begin{aligned} \langle w^*, w_t \rangle &= \langle w^*, w_{t-1} \rangle + y_i \langle w^*, x_i \rangle \\ &\geq \langle w^*, w_{t-1} \rangle + \varepsilon \\ &\geq \cdots \\ &\geq t\varepsilon \end{aligned}$$

Now, note that the norm of w_t is bounded from above as

$$\begin{aligned} \|w_t\|^2 &= \langle w_{t-1} + y_i x_i, w_{t-1} + y_i x_i \rangle && (84) \\ &= \|w_{t-1}\|^2 + 2y_i \langle x_i, w_{t-1} \rangle + \|x_i\|^2 && (85) \\ &\leq \|w_{t-1}\|^2 + M && (86) \\ &\leq \cdots && (87) \\ &\leq tM. && (88) \end{aligned}$$

Thus, using the Cauchy-Schwarz inequality,

$$t\varepsilon \leq \langle w^*, w_t \rangle \leq \|w^*\| \|w_t\| \leq \sqrt{t}\sqrt{M}$$

implying that $t \leq M/\varepsilon^2$. This is the bound on the number of mistakes made by the algorithm before the data is correctly separated.

The *voted perceptron* (Freund and Schapire, 1999) is a variant that has better theoretical guarantees and practical effectiveness. In this algorithm a weight or "vote" is tallied for every hyperplane—the

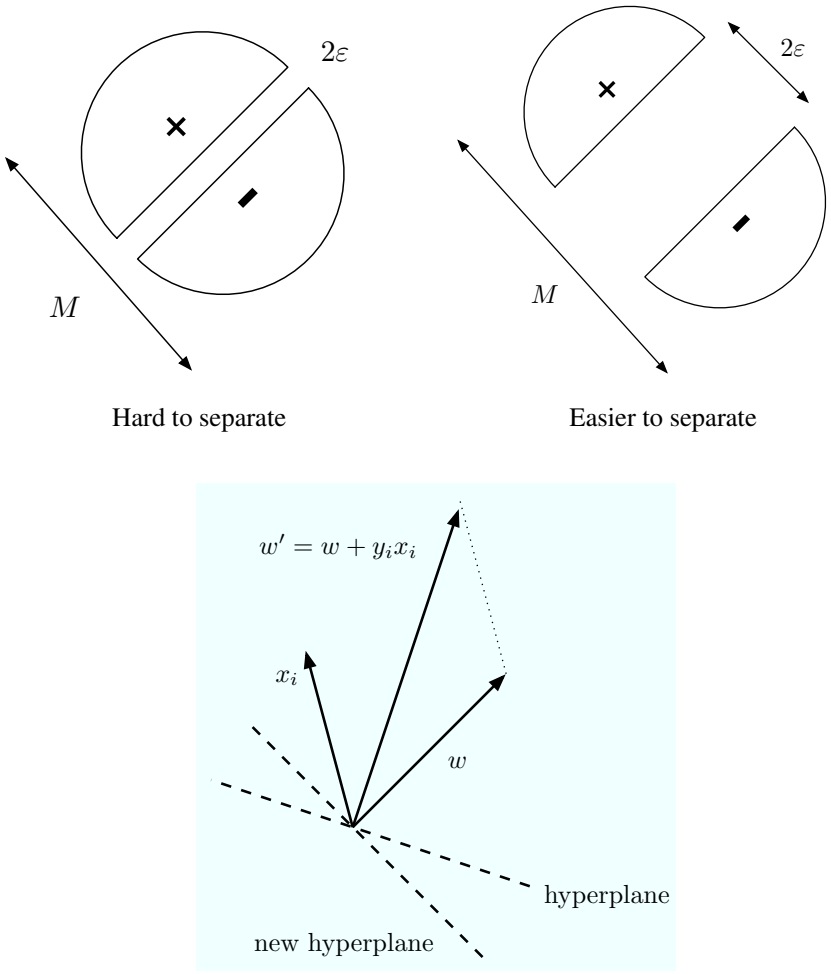


FIG 2. *Geometry of the perceptron algorithm.*

weight is large for those hyperplanes that correctly classify a large number of examples. In the end, the hyperplanes are summed up with the appropriate weights.

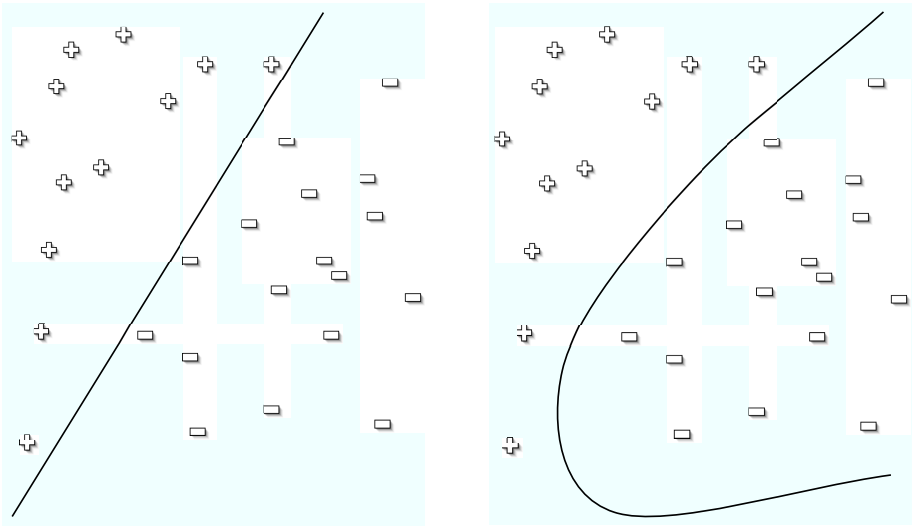


FIG 3. *Kernel decision boundaries are nonlinear in the input space.*

The Voted Perceptron Algorithm

Initialize $w_0 \leftarrow 0$, $t \leftarrow 0$. Proceed by making (possibly) multiple passes through the training data:

- For next example (x_i, y_i)
  - if $y_i \langle w, x_i \rangle \leq 0$, update

$$\begin{aligned} t &\leftarrow t+1 \\ w_t &= w_{t-1} + y_i x_i \\ v_t &= 1 \end{aligned}$$

  - if $y_i \langle w, x_i \rangle \geq 0$, increase vote $v_t \leftarrow v_t + 1$
- Stop after each example is correctly classified

Output: hyperplane with normal vector $w = \sum_t v_t \, w_t$.

As suggested by the figure below, in the first view of kernels we obtain nonlinear decision boundaries by replacing the inputs x with a "feature vector" $\phi(x)$. For example, if $x \in \mathbb{R}^2$ and $\phi(x) = (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \in \mathbb{R}^5$ then the decision surfaces are certain quadratic curves in $\mathbb{R}^2$.

However, a key observation is that the feature vectors need not be computed explicitly. Returning to the perceptron algorithm, note that the algorithm only requires computation of the inner products

$$\langle w, x \rangle = \left\langle \sum_{i=1}^{n} \alpha_i x_i, x \right\rangle$$

for certain weights α_i. This can be rewritten as

$$\langle w, x \rangle = \sum_{i=1}^{n} \alpha_i \langle x_i, x \rangle$$

If we use a feature mapping then the inner products become

$$\langle w, \phi(x) \rangle = \sum_{i=1}^{n} \alpha_i \langle \phi(x_i), \phi(x) \rangle$$

Therefore, we only need to evaluate the inner products

$$K(x_i, x) = \langle \phi(x_i), \phi(x) \rangle$$

and keep track of the weight α_i on the ith example. This is an instance of *the kernel trick*. To implement the kernel algorithm sparsely, we need to know the example that each mistake is made on: we'll use the notation $ind(s)$ to indicate the example used in iteration s. The resulting kernelized version of the perceptron algorithm is described below.

The Kernel Perceptron Algorithm

Initialize $t \leftarrow 1$. Proceed by making (possibly) multiple passes through the data:

- For next example (x_i, y_i), if

$$\text{sign}\left(\sum_{s=1}^{t} y_{ind(s)} K(x_{ind(s)}, x_i)\right) \neq y_i$$

  then set $t \leftarrow t + 1$ and $ind(t) \leftarrow i$.
- Stop after each example is correctly classified

Output: Decision rule $f(x) = \text{sign}\left(\sum_{s=1}^{t} y_{ind(s)} K(x_{ind(s)}, x)\right)$.

In the voted version of this algorithm, we also keep track of the vote v_t for iteration t. In either case, we can express the algorithm as forming a decision rule

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i K(x_i, x)\right)$$

where α_i is a weight assigned to the ith example. The weights are non-zero only on those examples for which mistakes were made; these are referred to as the *support vectors*.

The matrix

$$\mathbb{K} = [K(x_i, x_j)] = [\langle \phi(x_i), \phi(x_j) \rangle]$$

is called the Gram matrix—the kernel evaluations on the data.

References

Bach, F. R. and Jordan, M. I. (2003). Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48.

Chapelle, O., Weston, J., and Schölkopf, B. (2002). Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems, 15*, volume 15.

Chung, F. R. K. (1997). *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Herbich, R. (2002). *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press.

Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Smola, A. and Kondor, R. (2003). Kernels and regularization on graphs. In *Conference on Learning Theory, COLT/KW*.

Wahba, G. (1990). *Spline models for observational data*. SIAM. New York, NY.