Kernel Methods

David S. Rosenberg

New York University

February 26, 2019

Contents

Bring in the feature map to kernelization

A kernelized objective

- Previously, we took the following optimization problem:

$$w^* \in \underset{w \in \mathcal{H}}{\arg\min}\, R\left(\|w\|\right) + L\left(\langle w, x_1\rangle, \ldots, \langle w, x_n\rangle\right)$$

  and kernelized it as

$$\alpha^* \in \underset{\alpha \in \mathbf{R}^n}{\arg\min}\, R\left(\sqrt{\alpha^T K \alpha}\right) + L\left(K\alpha\right),$$

  where

$$K = \begin{pmatrix} \langle x_1, x_1\rangle & \cdots & \langle x_1, x_n\rangle \\ \vdots & \ddots & \ldots \\ \langle x_n, x_1\rangle & \cdots & \langle x_n, x_n\rangle \end{pmatrix} \quad \text{and} \quad k_x = \begin{pmatrix} \langle x_1, x\rangle \\ \vdots \\ \langle x_n, x\rangle \end{pmatrix}$$

- But computing K and k_x can be computationally hard for large feature spaces.
- To address this issue, we'll take a step back, and explicitly talk about feature maps.

The Input Space $\mathcal{X}$

- Our general learning theory setup: no assumptions about $\mathcal{X}$
- But $\mathcal{X} = \mathbf{R}^d$ for the specific methods we've developed:
  - Ridge regression
  - Lasso regression
  - Support Vector Machines
- Our hypothesis space for these was all affine functions on $\mathbf{R}^d$:

$$\mathcal{F} = \left\{ x \mapsto w^T x + b \mid w \in \mathbf{R}^d, b \in \mathbf{R} \right\}.$$

- What if we want to do prediction on inputs not natively in $\mathbf{R}^d$?

Linear Models with Explicit Feature Map

- **Input space**: $\mathcal{X}$
- **Feature space**: $\mathcal{H}$ (a Hilbert space, i.e. an inner product space with projections, e.g. $\mathbf{R}^d$)
- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- Hypothesis space of affine functions on feature space:

$$\mathcal{F} = \{x \mapsto \langle w, \psi(x) \rangle + b \mid w \in \mathcal{H}, b \in \mathbf{R}\}.$$

Kernelized objective with feature map

- Optimization problem with explicit feature map:

$$w^* \in \operatorname*{arg\,min}_{w \in \mathcal{H}} R(\|w\|) + L(\langle w, \psi(x_1) \rangle, \ldots, \langle w, \psi(x_n) \rangle)$$

and kernelized it as

$$\alpha^* \in \operatorname*{arg\,min}_{\alpha \in \mathbf{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L(K\alpha),$$

where

$$K = \begin{pmatrix} \langle \psi(x_1), \psi(x_1) \rangle & \cdots & \langle \psi(x_1), \psi(x_n) \rangle \\ \vdots & \ddots & \cdots \\ \langle \psi(x_n), \psi(x_1) \rangle & \cdots & \langle \psi(x_n), \psi(x_n) \rangle \end{pmatrix} \quad \text{and} \quad k_x = \begin{pmatrix} \langle \psi(x_1), \psi(x) \rangle \\ \vdots \\ \langle \psi(x_n), \psi(x) \rangle \end{pmatrix}$$

The Kernel Function

The Kernel Function

- **Input space**: $\mathcal{X}$
- **Feature space**: $\mathcal{H}$ (a Hilbert space, i.e. an inner product space with projections, e.g. $\mathbf{R}^d$)
- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ in $\mathcal{H}$ is

$$k(x, x') = \langle \psi(x), \psi(x') \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the inner product associated with $\mathcal{H}$.

The Kernel Function: Why do we need this?

- **Feature map**: $\psi : \mathcal{X} \to \mathcal{H}$
- The **kernel function** corresponding to ψ is

$$k(x, x') = \left\langle \psi(x), \psi(x') \right\rangle.$$

- Why introduce this new notation $k(x, x')$?
- We can often evaluate $k(x, x')$ without explicitly computing $\psi(x)$ and $\psi(x')$.
- For large feature spaces, can be much faster.

Kernel Evaluation Can Be Fast

Example

Quadratic feature map for $x = (x_1, \ldots, x_d) \in \mathbf{R}^d$.

$$\psi(x) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d)^T$$

has dimension $O(d^2)$, but for any $x, x' \in \mathbf{R}^d$ and the standard Euclidean dot products,

$$k(x, x') = \langle \psi(x), \psi(x') \rangle = \langle x, x' \rangle + \langle x, x' \rangle^2$$

- Explicit computation of $k(x, x')$: $O(d^2)$
- Implicit computation of $k(x, x')$: $O(d)$

Kernels as Similarity Scores

- Often useful to think of the kernel function as a **similarity score**.
- But this is not a mathematically precise statement.
- There are many ways to design a similarity score.
- We will use kernel functions that correspond to inner products in some feature space.
- These are called **Mercer kernels.**

What are the Benefits of Kernelization?

1. Computational (when optimizing over $\mathbf{R}^n$ is better than over $\mathbf{R}^d$)).
2. Can sometimes avoid any $O(d)$ operations
   - allows access to **infinite-dimensional feature spaces**.
3. Allows thinking in terms of "similarity" rather than features.

The Kernel Matrix

Definition

The **kernel matrix** for a kernel k on $x_1,\ldots,x_n \in \mathcal{X}$ is

$$K = \left(k(x_i,x_j)\right)_{i,j} = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \cdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{pmatrix} \in \mathbf{R}^{n\times n}.$$

- In ML this is also called a **Gram matrix**, but traditionally (in linear algebra),
  - Gram matrices are defined without reference to a kernel or feature map.

The Kernel Matrix

- The kernel matrix summarizes all the information we need about the training inputs $x_1, \ldots, x_n$ to solve a kernelized optimization problem.
- e.g. in the kernelized SVM, we can replace $\psi(x_i)^T \psi(x_j)$ with K_{ij}:

$$
\begin{array}{lll}
\sup_{\alpha} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ij} & \\
\text{s.t.} & \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{c}{n}\right] \; i = 1, \ldots, n. &
\end{array}
$$

The "Kernel Trick"

1. Given a kernelized ML algorithm (i.e. all $\psi(x)$'s show up as $\langle\psi(x),\psi(x')\rangle$).
2. Can swap out the inner product for a new kernel function.
3. New kernel may correspond to a very high-dimensional feature space.
4. Once the kernel matrix is computed, the computational cost depends on number of data points, rather than the dimension of feature space.

The **trick** is that once you've implemented your method in terms of a kernel matrix, you can go from a kernel corresponding to a very small feature vector to a kernel corresponding to a very large (even infinite dimensional) feature vector, without changing your code, just by swapping one kernel matrix for another. Runtime is unaffected, after the kernel matrix is computed.

The Kernel Trick(s)

1. Given a kernelized ML algorithm (i.e. all $\psi(x)$'s show up as $\langle \psi(x), \psi(x') \rangle$).
2. Can write training function to depend on x's only through kernel matrix:
   - `kernelized_train(kernel_matrix K, training_labels y)`
3. One "kernel trick" is that whether the kernel corresponds to a small feature vector, a very large feature vector, or even an infinite dimensional feature vector (to be discussed), the training code and training time remain the same (**once the kernel matrix is computed**).
4. Another "kernel trick" is that often we can compute the entries of the kernel matrix without the computational cost of explicilty working in the feature space.

Kernels

Choose a kernel instead of choosing a feature map

- Once a method is kernelized, we can effectively change the feature map by changing the kernel function.
- This will allow us to run linear methods with kernels that correspond to infinite dimensional feature maps.
- Below we discuss the most commonly used kernels.

Linear Kernel

- Input space: $\mathcal{X} = \mathbf{R}^d$
- Feature space: $\mathcal{H} = \mathbf{R}^d$, with standard inner product
- Feature map

$$\psi(x) = x$$

- Kernel:

$$k(x, x') = x^T x'$$

Quadratic Kernel in $\mathbf{R}^d$

- Input space $\mathcal{X} = \mathbf{R}^d$
- Feature space: $\mathcal{H} = \mathbf{R}^D$, where $D = d + \binom{d}{2} \approx d^2/2$.
- Feature map:

$$\psi(x) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_ix_j, \ldots \sqrt{2}x_{d-1}x_d)^T$$

- Then for $\forall x, x' \in \mathbf{R}^d$

$$\begin{aligned} k(x,x') &= \langle \psi(x), \psi(x') \rangle \\ &= \langle x, x' \rangle + \langle x, x' \rangle^2 \end{aligned}$$

- Computation for inner product with explicit mapping: $O(d^2)$
- Computation for implicit kernel calculation: $O(d)$.

Based on Guillaume Obozinski's Statistical Machine Learning course at Louvain, Feb 2014.

Polynomial Kernel in $\mathbf{R}^d$

- Input space $\mathcal{X} = \mathbf{R}^d$
- Kernel function:

$$k(x, x') = \left(1 + \langle x, x' \rangle\right)^M$$

- Corresponds to a feature map with all monomials up to degree M.
- For any M, computing the kernel has same computational cost
- Cost of explicit inner product computation grows rapidly in M.

The RBF Kernel

Radial Basis Function (RBF) / Gaussian Kernel

- Input space $\mathcal{X} = \mathbf{R}^d$

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right),$$

  where σ^2 is known as the bandwidth parameter.

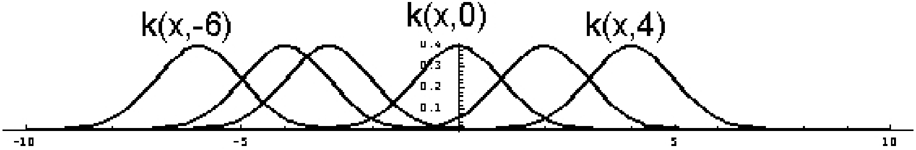
- Does it act like a similarity score?
- Why "radial"?
- Have we departed from our "inner product of feature vector" recipe?
  - Yes and no: corresponds to an infinite dimensional feature vector
- Probably the most common nonlinear kernel.

RBF Basis

- Input space $\mathcal{X} = \mathbf{R}$
- Output space: $\mathcal{Y} = \mathbf{R}$
- RBF kernel $k(w,x) = \exp\left(-(w-x)^2\right)$.
- Suppose we have 6 training examples: $x_i \in \{-6,-4,-3,0,2,4\}$.
- If representer theorem applies, then

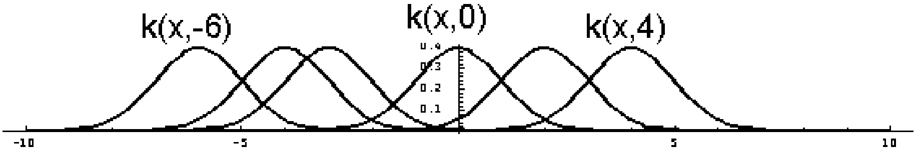
$$f(x) = \sum_{i=1}^{6} \alpha_i k(x_i, x).$$

- f is a linear combination of 6 basis functions of form $k(x_i,\cdot)$:

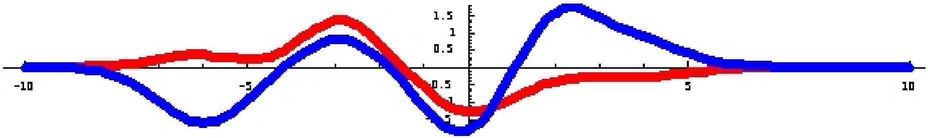


RBF Predictions

- Basis functions



- Predictions of the form $f(x) = \sum_{i=1}^{6} \alpha_i k(x_i, x)$:



- When kernelizing with RBF kernel, prediction functions always look this way.
- (Whether we get w from SVM, ridge regression, etc...)

RBF Feature Space: The Sequence Space ℓ_2

- To work with infinite dimensional feature vectors, we need a space with certain properties.
  - an inner product
  - a norm related to the inner product
  - projection theorem: $x = x_\perp + x_{||}$ where $x_{||} \in S = \text{span}(w_1, \ldots, w_n)$ and $\langle x_\perp, s \rangle = 0 \quad \forall s \in S$.
- Basically, we need a Hilbert space.

Definition

ℓ_2 is the space of all real-valued sequences: $(x_0, x_1, x_2, x_3, \ldots)$ with $\sum_{i=0}^{\infty} x_i^2 < \infty$.

Theorem

*With the the inner product $\langle x, x' \rangle = \sum_{i=0}^{\infty} x_i x_i'$, ℓ_2 is a **Hilbert space**.*

The Infinite Dimensional Feature Vector for RBF

- Consider RBF kernel (1-dim): $k(x,x') = \exp\left(-(x-x')^2/2\right)$
- We claim that $\psi : \mathbf{R} \to \ell_2$, defined by

$$[\psi(x)]_j = \frac{1}{\sqrt{j!}} e^{-x^2/2} x^j$$

gives the **"infinite-dimensional feature vector" corresponding to RBF kernel**.

- Is this mapping even well-defined? Is $\psi(x)$ even an element of ℓ_2?
- Yes:

$$\sum_{j=0}^{\infty} \frac{1}{j!} e^{-x^2} x^{2j} = e^{-x^2} \sum_{j=0}^{\infty} \frac{\left(x^2\right)^j}{j!} = 1 < \infty$$

.

The Infinite Dimensional Feature Vector for RBF

- Does feature vector $[\psi(x)]_n = \frac{1}{\sqrt{j!}} e^{-x^2/2} x^j$ actually correspond to the RBF kernel?
- Yes! Proof:

$$
\begin{aligned}
\langle \psi(x), \psi(x') \rangle &= \sum_{j=0}^{\infty} \frac{1}{j!} e^{-\left(x^2+(x')^2\right)/2} x^j \left(x'\right)^j \\
&= e^{-\left(x^2+(x')^2\right)/2} \sum_{j=0}^{\infty} \frac{(xx')^j}{j!} \\
&= \exp\left(-\left[x^2+\left(x'\right)^2\right]/2\right) \exp\left(xx'\right) \\
&= \exp\left(-\left[(x-x')^2/2\right]\right)
\end{aligned}
$$

QED

When is $k(x, x')$ a kernel function? (Mercer's Theorem)

How to Get Kernels?

1. Explicitly construct $\psi(x): \mathcal{X} \to \mathbf{R}^d$ and define $k(x, x') = \psi(x)^T \psi(x')$.
2. Directly define the kernel function $k(x, x')$, and verify it corresponds to $\langle \psi(x), \psi(x') \rangle$ for some ψ.

There are many theorems to help us with the second approach

Positive Semidefinite Matrices

Definition

A real, symmetric matrix $M \in \mathbf{R}^{n \times n}$ is **positive semidefinite (psd)** if for any $x \in \mathbf{R}^n$,

$$x^T M x \geqslant 0.$$

Theorem

The following conditions are each necessary and sufficient for a symmetric matrix M to be positive semidefinite:

- *M has can be factorized as $M = R^T R$, for some matrix R.*
- *All eigenvalues of M are greater than or equal to 0.*

Positive Semidefinite Function

Definition

A symmetric kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$ is **positive semidefinite (psd)** if for any finite set $\{x_1, \ldots, x_n\} \in \mathcal{X}$, the kernel matrix on this set

$$K = \left(k(x_i, x_j)\right)_{i,j} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \cdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

is a positive semidefinite matrix.

Mercer's Theorem

Theorem

A symmetric function $k(x,x')$ can be expressed as an inner product

$$k(x,x') = \langle \psi(x), \psi(x') \rangle$$

for some ψ if and only if $k(x,x')$ is ***positive semidefinite.***

Generating New Kernels from Old

- Suppose $k, k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$ are psd kernels. Then so are the following:

$$
\begin{aligned}
k_{\text{new}}(x, x') &= k_1(x, x') + k_2(x, x') \\
k_{\text{new}}(x, x') &= \alpha k(x, x') \\
k_{\text{new}}(x, x') &= f(x) f(x') \text{ for any function } f(\cdot) \\
k_{\text{new}}(x, x') &= k_1(x, x') k_2(x, x')
\end{aligned}
$$

- See Appendix for details.
- Lots more theorems to help you construct new kernels from old...

Details on New Kernels from Old [Optional]

Additive Closure

- Suppose k_1 and k_2 are psd kernels with feature maps ϕ_1 and ϕ_2, respectively.
- Then
$$k_1(x,x') + k_2(x,x')$$
is a psd kernel.
- Proof: Concatenate the feature vectors to get
$$\phi(x) = (\phi_1(x), \phi_2(x)).$$
Then ϕ is a feature map for $k_1 + k_2$.

Closure under Positive Scaling

- Suppose k is a psd kernel with feature maps ϕ.
- Then for any $\alpha > 0$,
$$\alpha k$$
is a psd kernel.
- Proof: Note that
$$\phi(x) = \sqrt{\alpha}\phi(x)$$
is a feature map for αk.

Scalar Function Gives a Kernel

- For any function $f(x)$,
$$k(x,x') = f(x)f(x')$$
is a kernel.
- Proof: Let $f(x)$ be the feature mapping. (It maps into a 1-dimensional feature space.)
$$\langle f(x), f(x') \rangle = f(x)f(x') = k(x,x').$$

Closure under Hadamard Products

- Suppose k_1 and k_2 are psd kernels with feature maps ϕ_1 and ϕ_2, respectively.
- Then

$$k_1(x,x')k_2(x,x')$$

  is a psd kernel.
- Proof: Take the outer product of the feature vectors:

$$\phi(x) = \phi_1(x)\left[\phi_2(x)\right]^T.$$

  Note that $\phi(x)$ is a matrix.
- Continued...

Closure under Hadamard Products

- Then

$$\begin{aligned}
\langle \phi(x), \phi(x') \rangle &= \sum_{i,j} \phi(x)\phi(x') \\
&= \sum_{i,j} \left[\phi_1(x)\left[\phi_2(x)\right]^T\right]_{ij} \left[\phi_1(x')\left[\phi_2(x')\right]^T\right]_{ij} \\
&= \sum_{i,j} \left[\phi_1(x)\right]_i \left[\phi_2(x)\right]_j \left[\phi_1(x')\right]_i \left[\phi_2(x')\right]_j \\
&= \left(\sum_i \left[\phi_1(x)\right]_i \left[\phi_1(x')\right]_i\right)\left(\sum_j \left[\phi_2(x)\right]_j \left[\phi_2(x')\right]_j\right) \\
&= k_1(x,x')k_2(x,x')
\end{aligned}$$