Kernel Methods

David S. Rosenberg

New York University

February 20, 2018

Contents

Big Feature Spaces for Linear Models

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 $\mathbf{R}^d$:

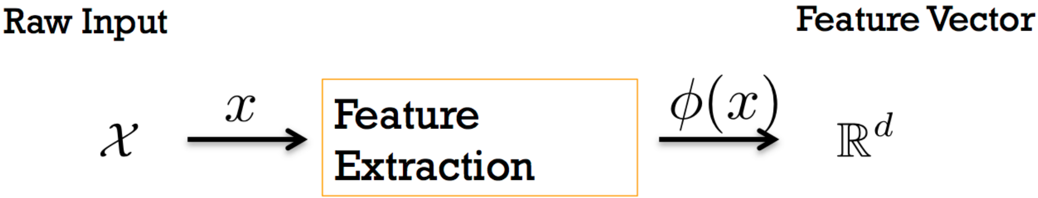
$$\mathcal{F} = \left\{ x \mapsto w^T x + b \mid w \in \mathbf{R}^d, b \in \mathbf{R} \right\}.$$

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

Feature Extraction

Definition

Mapping an input from $\mathcal{X}$ to a vector in $\mathbf{R}^d$ is called **feature extraction** or **featurization**.

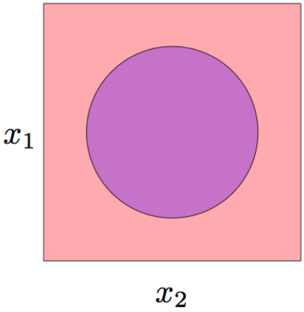


Linear Models with Explicit Feature Map

- Input space: $\mathcal{X}$ (no assumptions)
- Introduce **feature map** $\psi : \mathcal{X} \to \mathbf{R}^d$
- The feature map maps into the **feature space** $\mathbf{R}^d$.
- Hypothesis space of affine functions on feature space:

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

Geometric Example: Two class problem, nonlinear boundary



- With linear feature map $\psi(x) = (x_1, x_2)$ and linear models, can't separate regions
- With appropriate featurization $\psi(x) = \left(x_1, x_2, x_1^2 + x_2^2\right)$, becomes linearly separable .
- Video: http://youtu.be/3liCbRZPrZA

From Percy Liang's "Lecture 3" slides from Stanford's CS221, Autumn 2014.

Expressivity of Hypothesis Space

- For linear models, to grow the hypothesis spaces, we must add features.
- Sometimes we say a larger hypothesis is "more expressive".
  - (can fit more relationships between input and action)
- The previous lecture on "Features" suggests many ways to create new features.

Example: Monomial Interaction Terms

- Suppose we start with $x = (1, x_1, \ldots, x_d) \in \mathbf{R}^{d+1} = \mathcal{X}$.
- To get a more expressive hypothesis space, we want to add interaction terms.
- Consider adding add all monomials up to degree M: $x_1^{p_1} \cdots x_d^{p_d}$, with $p_1 + \cdots + p_d \leqslant M$.
- How many features will we end up with?
- $\binom{M+d}{M}$ ("flower shop problem" from combinatorics)
- For $d = 40$ and $M = 8$, we get 377348994 features.
- That will make some extremely large data matrices...

Big Feature Spaces

Very large feature spaces have two potential issues:

1. Overfitting
2. Memory and computational costs

- Overfitting we handle with regularization.
- "**Kernel methods**" can (sometimes) help with memory and computational costs.

Kernel Methods: Motivation

SVM with Explicit Feature Map

- Let $\psi : \mathcal{X} \to \mathbf{R}^d$ be a feature map.
- The SVM optimization problem (with explicit feature map):

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T \psi(x_i)\right).$$

- Last time we mentioned an equivalent optimization problem from Lagrangian duality...

SVM Dual Problem

- By Lagrangian duality, it is equivalent to solve following optimization problem:

$$
\begin{aligned}
\sup_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \psi(x_j)^T \psi(x_i) \\
\text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \qquad \alpha_i \in \left[0, \frac{c}{n}\right] \; i = 1, \ldots, n.
\end{aligned}
$$

- If α^* is an optimal value, then

$$
w^* = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i) \qquad \text{and} \qquad \hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i \psi(x_i)^T \psi(x).
$$

- Notice: $\psi(x)$ only shows up in an inner products with another $\psi(x')$.

Some Methods Can Be "Kernelized"

Definition

A method is **kernelized** if every feature vector $\psi(x)$ only appears inside an inner product with another feature vector $\psi(x')$. In particular, this applies to both the optimization problem and the prediction function.

- The SVM Dual is a kernelization of the original SVM formulation.

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 is

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

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

The Kernel Function: Why?

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

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

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

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

- Naively 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 kernel functions which correspond to inner products in some feature space.
    - These are called **Mercer kernels.**

What are the Benefits of Kernelization?

1. Computational (e.g. when feature space dimension d larger than sample size n).
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**[a] for a kernel k on $x_1,\ldots,x_n \in \mathcal{X}$ is

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

[a]This is also called a **Gram matrix,** though usually Gram matrices are defined without reference to a kernel function or a feature map.

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

$$\sup_{\alpha} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K_{ij}$$

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 all inner products are computed, 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.

Our Plan

- Present our principal tool for kernelization: the **representer theorem**
- To keep things clean, we'll drop the explicit feature map until we need it.
- Discuss specific cases of kernel ridge regression and kernel SVM
- Discuss several kernels, including the famous RBF kernel.
- Discuss how to create a kernel without an explicit feature map.

The Representer Theorem to Kernelize

The Representer Theorem

Theorem (Representer Theorem)

Let

$$J(w) = R\left(\|w\|\right) + L\left(\langle w, x_1\rangle, \ldots, \langle w, x_n\rangle\right),$$

where

- $w, x_1, \ldots, x_n \in \mathcal{H}$ *for some Hilbert space* $\mathcal{H}$. *(We typically have* $\mathcal{H} = \mathbf{R}^d$.*)*
- $\|\cdot\|$ *is the norm corresponding to the inner product of* $\mathcal{H}$. *(i.e.* $\|w\| = \sqrt{\langle w, w\rangle}$*)*
- $R : [0, \infty) \to \mathbf{R}$ *is nondecreasing (**Regularization term**), and*
- $L : \mathbf{R}^n \to \mathbf{R}$ *is arbitrary (**Loss term**).*

If $J(w)$ *has a minimizer, then it **has a minimizer of the form*** $w^* = \sum_{i=1}^n \alpha_i x_i$.
[If R *is strictly increasing, then all minimizers have this form. (Proof in homework.)]*

Rewriting the Objective Function

- Define the training score function $s : \mathbf{R}^d \to \mathbf{R}^n$ by

$$s(w) = \begin{pmatrix} \langle w, x_1 \rangle \\ \vdots \\ \langle w, x_n \rangle \end{pmatrix},$$

  which gives the **training score vector** for any w.

- We can then rewrite the objective function as

$$J(w) = R(\|w\|) + L(s(w)),$$

  where now $L : \mathbf{R}^{n \times 1} \to \mathbf{R}$ takes a column vector as input.

- This will allow us to have a slick reparametrized version...

Reparametrize the Generalized Objective

- By the Representer Theorem, it's sufficient to minimize $J(w)$ for w of the form $\sum_{i=1}^{n} \alpha_i x_i$.
- Plugging this form into $J(w)$, we see we can just minimize

$$J_0(\alpha) = R\left(\left\|\sum_{i=1}^{n} \alpha_i x_i\right\|\right) + L\left(s\left(\sum_{i=1}^{n} \alpha_i x_i\right)\right)$$

  over $\alpha = (\alpha_1, \ldots, \alpha_n)^T \in \mathbf{R}^{n \times 1}$.

- With some new notation, we can substantially simplify
  - the norm piece $\|w\| = \|\sum_{i=1}^{n} \alpha_i x_i\|$, and
  - the score piece $s(w) = s\left(\sum_{i=1}^{n} \alpha_i x_i\right)$.

Simplifying the Reparametrized Norm

- For the norm piece $\|w\| = \|\sum_{i=1}^{n} \alpha_i x_i\|$, we have

$$
\begin{aligned}
\|w\|^2 &= \langle w, w \rangle \\
&= \left\langle \sum_{i=1}^{n} \alpha_i x_i, \sum_{j=1}^{n} \alpha_j x_j \right\rangle \\
&= \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle .
\end{aligned}
$$

- This expression involves the n^2 inner products between all pairs of input vectors.
- We often put those values together into a matrix...

The Gram Matrix

Definition

The **Gram matrix** of a set of points $x_1, \ldots, x_n$ in an inner product space is defined as

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

- This is the traditional definition from linear algebra.
- The Gram matrix is a special case of a **kernel matrix** with an identity feature map.
- That's why we write K for the Gram matrix instead of G, as done elsewhere.
- NOTE: In ML, we often use Gram matrix and kernel matrix to mean the same thing. Don't get too hung up on the definitions.

Example: Gram Matrix for the Dot Product

- Consider $x_1, \ldots, x_n \in \mathbf{R}^{d \times 1}$ with the standard inner product $\langle x, x' \rangle = x^T x'$.
- Let $X \in \mathbf{R}^{n \times d}$ be the **design matrix**, which has each input vector as a row:

$$X = \begin{pmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{pmatrix}.$$

- Then the Gram matrix is

$$\begin{aligned} K &= \begin{pmatrix} x_1^T x_1 & \cdots & x_1^T x_n \\ \vdots & \ddots & \cdots \\ x_n^T x_1 & \cdots & x_n^T x_n \end{pmatrix} = \begin{pmatrix} -x_1^T- \\ \vdots \\ -x_n^T- \end{pmatrix} \begin{pmatrix} | & \cdots & | \\ x_1 & \cdots & x_n \\ | & \cdots & | \end{pmatrix} \\ &= XX^T \end{aligned}$$

Simplifying the Reparametrized Norm

- With $w=\sum_{i=1}^{n}\alpha_i x_i$, we have

$$
\begin{aligned}
\|w\|^2 &= \langle w,w\rangle \\
&= \left\langle \sum_{i=1}^{n}\alpha_i x_i, \sum_{j=1}^{n}\alpha_j x_j \right\rangle \\
&= \sum_{i,j=1}^{n}\alpha_i\alpha_j \left\langle x_i, x_j\right\rangle \\
&= \alpha^T K\alpha.
\end{aligned}
$$

Simplifying the Training Score Vector

- The score for x_j for $w = \sum_{i=1}^n \alpha_i x_i$ is

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

- The training score vector is

$$\begin{aligned} s\left(\sum_{i=1}^n \alpha_i x_i\right) &= \begin{pmatrix} \sum_{i=1}^n \alpha_i \langle x_i, x_1 \rangle \\ \vdots \\ \sum_{i=1}^n \alpha_i \langle x_i, x_n \rangle \end{pmatrix} = \begin{pmatrix} \alpha_1 \langle x_1, x_1 \rangle + \cdots + \alpha_n \langle x_n, x_1 \rangle \\ \vdots \\ \alpha_1 \langle x_1, x_n \rangle + \cdots + \alpha_n \langle x_n, x_n \rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle x_1, x_1 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \ddots & \cdots \\ \langle x_n, x_1 \rangle & \cdots & \langle x_n, x_n \rangle \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} \\ &= K\alpha \end{aligned}$$

Reparametrized Objective

- Putting it all together, our reparametrized objective function can be written as

$$\begin{aligned} J_0(\alpha) &= R\left(\left\|\sum_{i=1}^{n}\alpha_i x_i\right\|\right) + L\left(s\left(\sum_{i=1}^{n}\alpha_i x_i\right)\right) \\ &= R\left(\sqrt{\alpha^T K\alpha}\right) + L(K\alpha), \end{aligned}$$

  which we minimize over $\alpha \in \mathbf{R}^n$.
- All information needed about $x_1, \ldots, x_n$ is summarized in the Gram matrix K.
- We're now minimizing over $\mathbf{R}^n$ rather than $\mathbf{R}^d$.
- If $d \gg n$, this can be a big win computationally (at least once K is computed).

Reparametrizing Predictions

- Suppose we've found

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

- Then we know $w^* = \sum_{i=1}^n \alpha^* x_i$ satisfies

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

- The prediction on a new point $x \in \mathcal{H}$ is

$$\hat{f}(x) = \langle w^*, x \rangle = \sum_{i=1}^n \alpha_i^* \langle x_i, x \rangle.$$

- To make a new prediction, we may need to touch all the training inputs $x_1, \dots, x_n$.

More Notation

- It will be convenient to define the following column vector for any $x \in \mathcal{H}$:

$$k_x = \begin{pmatrix} \langle x_1, x \rangle \\ \vdots \\ \langle x_n, x \rangle \end{pmatrix}$$

- Then we can write our predictions on a new point x as

$$\hat{f}(x) = k_x^T \alpha^*$$

Summary So Far

- Original plan:
  - Find $w^* \in \arg\min_{w \in \mathcal{H}} R\left(\|w\|\right) + L\left(\langle w, x_1 \rangle, \ldots, \langle w, x_n \rangle\right)$
  - Predict with $\hat{f}(x) = \langle w^*, x \rangle$.
- We showed that the following is equivalent:
  - Find $\alpha^* \in \arg\min_{\alpha \in \mathbf{R}^n} R\left(\sqrt{\alpha^T K \alpha}\right) + L\left(K\alpha\right)$
  - Predict with $\hat{f}(x) = k_x^T \alpha^*$, where

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

  - Every element $x \in \mathcal{H}$ occurs inside an inner products with a training input $x_i \in \mathcal{H}$.

Kernelization

Definition

A method is **kernelized** if every input $x \in \mathcal{H}$ only appears inside an inner products with another $x' \in \mathcal{H}$. In particular, this applies to both the optimization problem and the prediction function.

- Thus finding

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

and making predictions with $\hat{f}(x) = k_x^T \alpha^*$ is a **kernelization** of finding

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

and making predictions with $\hat{f}(x) = \langle w^*, x \rangle$.

- Will give some more explanation for the terms "kernelized" and "kernelization" later on.

Kernel Ridge Regression

Kernelizing Ridge Regression

- Ridge Regression:
$$\min_{w \in \mathbf{R}^d} \frac{1}{n}\|Xw - y\|^2 + \lambda\|w\|^2$$
- Plugging in $w = \sum_{i=1}^{n} \alpha_i x_i$, we get the kernelized ridge regression objective function:
$$\min_{\alpha \in \mathbf{R}^n} \frac{1}{n}\|K\alpha - y\|^2 + \lambda\alpha^T K\alpha$$
- This is usually just called **kernel ridge regression**.

Kernel Ridge Regression Solutions

- For $\lambda > 0$, the **ridge regression solution** is

$$w^* = (X^T X + \lambda I)^{-1} X^T y$$

- and the **kernel ridge regression solution** is

$$\begin{aligned} \alpha^* &= (XX^T + \lambda I)^{-1} y \\ &= (K + \lambda I)^{-1} y \end{aligned}$$

- (Shown in homework.)
- For ridge regression we're dealing with a $d \times d$ matrix.
- For kernel ridge regression we're dealing an $n \times n$ matix.

Predictions

- Predictions in terms of w^*:

$$\hat{f}(x) = x^T w^*$$

- Predictions in terms of α^*:

$$\hat{f}(x) = k_x^T \alpha^* = \sum_{i=1}^{n} \alpha_i^* x_i^T x$$

- For kernel ridge regression, need to access all training inputs $x_1, \ldots, x_n$ to predict.
- For SVM, we may not...

Kernel SVM

Kernelized SVM (From Representer Theorem)

- The SVM objective:

$$\min_{w \in \mathbf{R}^d} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i w^T x_i\right).$$

- Plugging in $w = \sum_{i=1}^{n} \alpha_i x_i$, we get

$$\min_{\alpha \in \mathbf{R}^n} \frac{1}{2}\alpha^T K \alpha + \frac{c}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i (K\alpha)_i\right)$$

- Predictions with

$$\hat{f}(x) = x^T w^* = \sum_{i=1}^{n} \alpha_i^* x_i^T x.$$

- This is one way to kernelize SVM...

Kernelized SVM (From Lagrangian Duality)

- Kernelized SVM from computing the Lagrangian Dual Problem:

$$\begin{aligned} \max_{\alpha \in \mathbf{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_j^T x_i \\ \text{s.t.} \quad & \sum_{i=1}^n \alpha_i y_i = 0 \\ & \alpha_i \in \left[0, \frac{c}{n}\right] \quad i = 1, \ldots, n. \end{aligned}$$

- If α^* is an optimal value, then

$$w^* = \sum_{i=1}^n \alpha_i^* y_i x_i \qquad \text{and} \qquad \hat{f}(x) = \sum_{i=1}^n \alpha_i^* y_i x_i^T x.$$

- Note that the prediction function is also kernelized.

Sparsity in the Data from Complementary Slackness

- Kernelized predictions given by

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i^* y_i x_i^T x.$$

- By a Lagrangian duality analysis (specifically from complementary slackness), we find

$$\begin{aligned} y_i \hat{f}(x_i) < 1 \quad &\Longrightarrow \quad \alpha_i^* = \frac{c}{n} \\ y_i \hat{f}(x_i) = 1 \quad &\Longrightarrow \quad \alpha_i^* \in \left[0, \frac{c}{n}\right] \\ y_i \hat{f}(x_i) > 1 \quad &\Longrightarrow \quad \alpha_i^* = 0 \end{aligned}$$

- So we can leave out any x_i "on the good side of the margin" ($y_i \hat{f}(x_i) > 1$).
- x_i's that we must keep, because $\alpha_i^* \neq 0$, are called **support vectors**.

Kernelization

Kernelization

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 $\mathcal{X} = \mathbf{R}^d$

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

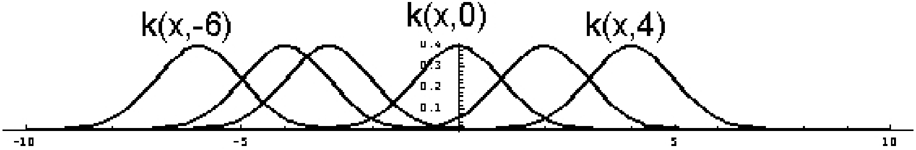
  where σ^2 is known as the bandwidth parameter.
- Does it act like a similarity score?
- Why "radial"?
- Have we departed from our "inner product of feature vector" recipe?
    - Yes and no: corresponds to an infinite dimensional feature vector
- Probably the most common nonlinear kernel.

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

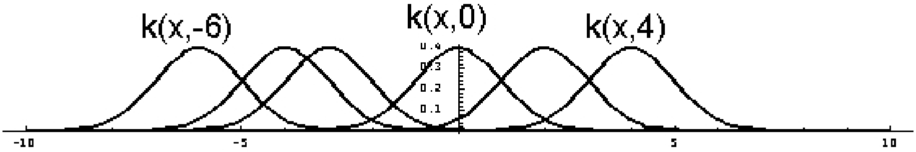
$$f(x) = \sum_{i=1}^{6} \alpha_i k(x_i, x).$$

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

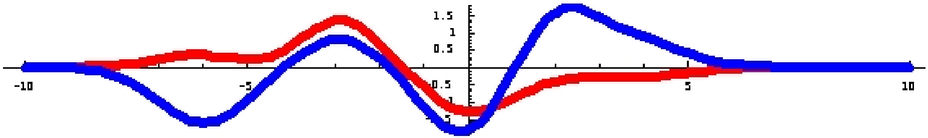


RBF Predictions

- 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.

Theorem

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

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

*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

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