Lecture Notes: PRML Chapter 6 – Section 6.2 Constructing Kernels

Prerequisites

- Inner products and Mercer's theorem (positive-semidefiniteness)
- Feature-space mappings $\phi(x)$ and basis functions $\phi_i(x)$
- Familiarity with common kernels (linear, polynomial, RBF)

Key Terminology

- **Feature map** $\phi(x)$: Transforms input x into a (possibly infinite-dimensional) feature space.
- **Basis functions** $\phi_i(x)$: Coordinates of $\phi(x)$, used to build kernels via

$$k(x, x') = \sum_{i=1}^{M} \phi_i(x)\, \phi_i(x')\,. \tag{6.10}$$

- **Valid kernel (Mercer kernel)**: A function k for which every Gram matrix $K_{nm} = k(x_n, x_m)$ is positive-semidefinite.
- **Closure properties**: Operations (sum, product, scaling, composition) that preserve kernel validity (Equations 6.13–6.22).
- **Stationary kernel**: Depends only on $x - x'$.
- **Homogeneous/RBF kernel**: Depends only on $\|x - x'\|$, e.g. Gaussian kernel (6.23).

Why It Matters

- **Custom similarity measures**: By constructing kernels directly, we can tailor similarity functions to data types (vectors, sets, strings) without explicit $\phi(x)$.
- **Infinite-dimensional features**: Valid kernels let us work implicitly in very high or infinite feature spaces.
- **Modular design**: Closure properties (sums, products, exponentials) enable building complex kernels from simpler ones.

Key Ideas

1. **Feature-map construction**
    - Specify $\phi(x)$ (e.g. monomials, Gaussians, sigmoids) and compute

$$k(x, x') = \sum_i \phi_i(x)\, \phi_i(x')\,. \tag{6.10}$$

    - Example: Quadratic kernel

$$k(x, z) = (x^\top z)^2$$

      expands to $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$, yielding Equation (6.12).
2. **Mercer's condition**
    - A function k is a valid kernel if and only if every finite Gram matrix is positive-semidefinite.
    - No need to find $\phi(x)$ explicitly—test via eigenvalues of K.
3. **Closure properties for building kernels**
    - If k_1, k_2 are valid, then so are (for c>0, f any function, q polynomial ≥ 0 coefficients):
        - ck_1 (6.13)
        - $f(x)\, k_1(x, x')\, f(x')$ (6.14)
        - $q(k_1(x, x'))$ (6.15)
        - $\exp(k_1(x, x'))$ (6.16)
        - $k_1 + k_2$ (6.17)
        - $k_1\, k_2$ (6.18)
        - plus block-structured and composite forms (6.19–6.22).
4. **Common kernels**
    - **Polynomial**: $(x^\top x' + c)^M$ includes all monomials up to degree M.
    - **Gaussian RBF**:

$$k(x, x') = \exp\bigl(-\|x - x'\|^2/(2\sigma^2)\bigr)\,, \tag{6.23}$$

      constructed via exponentials and scaling (6.24–6.26).
    - **Set kernels**: e.g. $k(A_1, A_2) = 2^{|A_1 \cap A_2|}$ for subsets.

- **Fisher kernel**:

$$k(x, x') = g(\theta, x)^\top F^{-1} g(\theta, x')\,, \quad g = \nabla_\theta \ln p(x|\theta)\,.$$

Relevant Figures from PRML

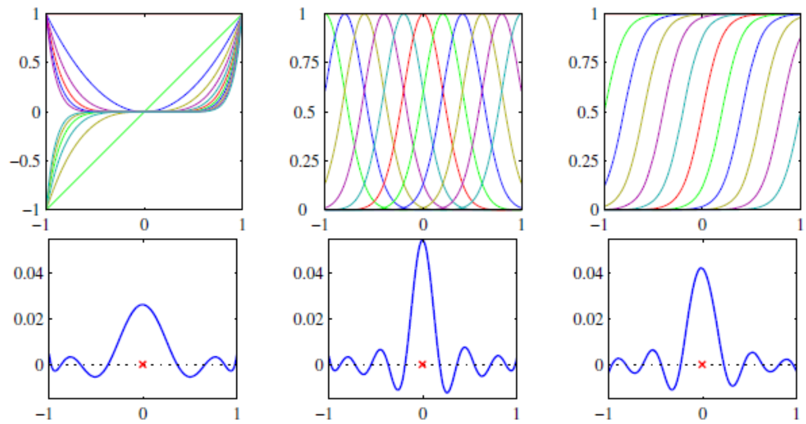
- 

  Figure 6.1 (p. 295): Shows three columns of basis functions (polynomials, Gaussians, sigmoids) paired with their induced kernels $k(x, x')$ plotted versus x for fixed $x' = 0$. Demonstrates how different ϕ-choices produce different similarity measures.

Learning Outcomes

After studying Section 6.2 “Constructing Kernels,” students will be able to:

1. **Explain the feature-map definition of a kernel** via

$$k(x, x') = \sum_{i=1}^{M} \phi_i(x)\,\phi_i(x').$$

2. **State Mercer’s condition**: a kernel is valid if every Gram matrix is positive-semidefinite.

3. **Apply closure properties** (scaling, sums, products, exponentials, function–kernel–function) to build new kernels.

4. **Identify common kernel families**: polynomial, Gaussian RBF, set kernels, Fisher kernels.

5. **Derive explicit feature maps** for simple polynomial kernels (e.g. $(x^\top z)^2$ or $(x^\top z + c)^2$).

6. **Test kernel validity** by reasoning about Gram-matrix positive-semidefiniteness without explicit $\phi(x)$.

7. **Construct kernels from generative models**, including mixture-based and Fisher kernels.

8. **Distinguish stationary kernels** (depend on $x - x'$) from general kernels.

9. **Combine kernels** modularly using the listed construction rules (Equations 6.13–6.22).

10. **Design simple domain-specific kernels**, such as set-intersection kernels.