Foundations of Machine Learning
AI2000 and AI5000

FoML-32
Constructing the Kernels

Dr. Konda Reddy Mopuri
Department of AI, IIT Hyderabad
July-Nov 2025

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DiL
Data-driven Intelligence
& Learning Lab

So far in FoML

- Intro to ML and Probability refresher
- MLE, MAP, and fully Bayesian treatment
- Supervised learning
    a. Linear Regression with basis functions
    b. Bias-Variance Decomposition
    c. Decision Theory - three broad classification strategies
    d. Neural Networks
- Unsupervised learning
    a. K-Means, Hierarchical, and GMM for clustering
- Kernelizing linear Models
    a. Dual representation

For today

- Kernel substitution/trick
- Constructing the Kernels

Dual formulation

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}. \qquad y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}$$

- Despite the computational demand, is useful
  - Expressed entirely in terms of the Kernel function
  - Avoids defining the basis functions explicitly
  - Allows us to implicitly use high (even, infinite) dimensional feature spaces

Kernel substitution

If we have an algorithm formulated in such a way that the input vector x enters only in the form of scalar products

↓

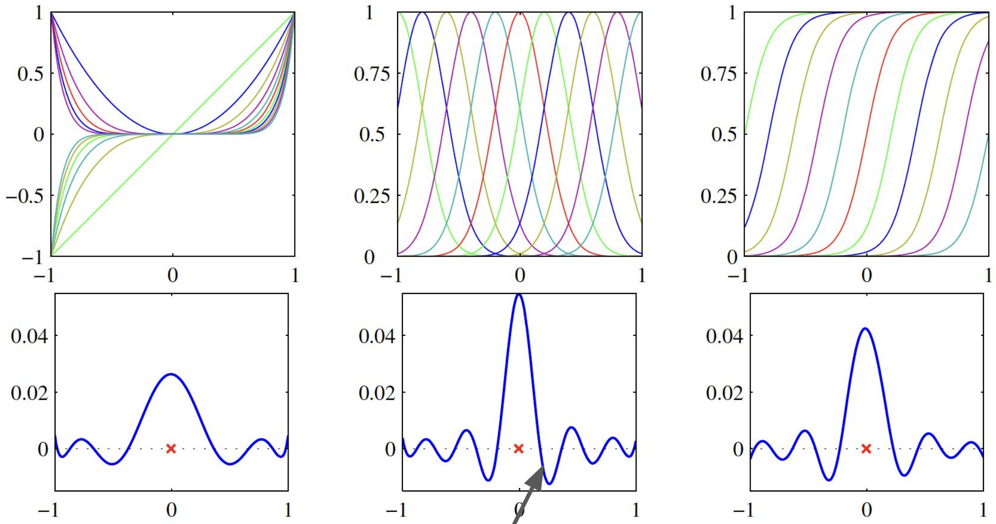
then we can replace that scalar product with a kernel

Constructing Kernels

- One way - choose the feature space
  - Then construct the kernel

$$k(x, x') = \boldsymbol{\phi}(x)^{\mathrm{T}} \boldsymbol{\phi}(x') = \sum_{i=1}^{M} \phi_i(x) \phi_i(x')$$

Constructing Kernels



Likely to be result of using a different kernel based on the Gaussian basis, but not the one shown in the above equation, or, a scaled dot product is used.

Constructing Kernels

- Alternate - construct kernel directly
- We must ensure that it is valid
    - i.e., it corresponds to scalar product in some feature space

Constructing Kernels

- Example
- 2D input: x, z

$$k(\mathbf{x}, \mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2.$$

Constructing Kernels

- Need a simple way to test a function if it is a valid kernel
- Necessary and sufficient condition
  - Gram matrix should be PSD
  - i.e., for every PSD kernel, there exists a feature projection (ϕ)

Example Kernels

- Gaussian $$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$$
- Generalized polynomial $$k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')^M$$
- Radial basis function $$k(\mathbf{x}, \mathbf{x}') = k(||\mathbf{x}^T \mathbf{x}'||^2)$$

Constructing Kernels

- Another way - build them out of simpler kernels as building blocks

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b)
\end{aligned}
$$

Next

- SVM

Rough