

Kernel Ridge Regression and the Kernel Trick

Machine Learning Course - CS-433

Nov 4, 2021

Nicolas Flammarion

EPFL

Equivalent formulation for Ridge regression

$$\min_w \sum_{i=1}^n (y_i - w^\top x_i)^2 + \frac{\lambda}{2} \|w\|^2$$

The solution is given by

$$w_* = \underbrace{(\mathbf{X}^\top \mathbf{X} + \lambda I_d)}_{\substack{\mathbf{X}^\top \in \mathbb{R}^{d \times n} \rightarrow d \times d}}^{-1} \mathbf{X}^\top \mathbf{y}$$

But it can be alternatively written as

$$w_* = \mathbf{X}^\top \underbrace{(\mathbf{X} \mathbf{X}^\top + \lambda I_n)}_{\substack{\mathbf{X} \in \mathbb{R}^{n \times d} \rightarrow n \times n}}^{-1} \mathbf{y}$$

Proof: let $P \in \mathbb{R}^{m \times n}$ and $Q \in \mathbb{R}^{n \times m}$

$$P(QP + I_n) = PQP + P = (PQ + I_m)P$$

Assume that $QP + I_n$ and $PQ + I_m$ are invertible

$$(PQ + I_m)^{-1}P = P(QP + I_n)^{-1}$$

We get the result with $P = \mathbf{X}^\top$ and $Q = \frac{1}{\lambda}\mathbf{X}$

$$w_* = \underbrace{(\mathbf{X}^\top \mathbf{X} + \lambda I_d)}_{\mathbf{X}^\top \in \mathbb{R}^{d \times n} \rightarrow d \times d}^{-1} \mathbf{X}^\top \mathbf{y}$$

But can be alternatively written as

$$w_* = \mathbf{X}^\top \underbrace{(\mathbf{X} \mathbf{X}^\top + \lambda I_n)}_{\mathbf{X} \in \mathbb{R}^{n \times d} \rightarrow n \times n}^{-1} \mathbf{y}$$

Usefulness of the alternative form

$$w_* = \underbrace{\mathbf{X}^\top}_{d \times n} \underbrace{(\mathbf{X}\mathbf{X}^\top + \lambda I_n)}_{n \times n}^{-1} \mathbf{y}$$

1. Computational complexity:

- For the original formulation $(\mathbf{X}^\top \mathbf{X} + \lambda I_d)^{-1} \mathbf{X}^\top \mathbf{y}$, $O(d^3 + nd^2)$
- For the new formulation $\mathbf{X}^\top (\mathbf{X}\mathbf{X}^\top + \lambda I_n)^{-1} \mathbf{y}$, $O(n^3 + dn^2)$

➡ Depending on d, n one can be more efficient than the other

2. Structural difference:

$$w_* = \mathbf{X}^\top \alpha_* \text{ where } \alpha_* = (\mathbf{X}\mathbf{X}^\top + \lambda I_n)^{-1} \mathbf{y}$$

➡ $w_* \in \text{span}\{x_1, \dots, x_n\}$

These two points are the crucial ingredients of the **kernel trick**

Representer Theorem

Claim: For any loss function ℓ , if $w_* = \arg \min_w \sum_{i=1}^n \ell(x_i^\top w, y_i) + \frac{\lambda}{2} \|w\|^2$

then there exists $\alpha_* \in \mathbb{R}^n$ such that

$$w_* = \mathbf{X}^\top \alpha_*$$

Meaning: There exists an optimal solution that lies in $\text{span}\{x_1, \dots, x_n\}$

Consequence: It is far more general than LS and we will be able to use the kernel tricks for various problems such as: Kernel SVM, Kernel LS, Kernel Principal Component Analysis

Proof of the representer theorem

We can always rewrite w_* as $w_* = \sum_{i=1}^n \alpha_i x_i + u$ where $u^\top x_i = 0$ for all i

Let's denote by $w = w_* - u$

- $\|w_*\|^2 = \|w\|^2 + \|u\|^2$, thus $\|w\|^2 \leq \|w_*\|^2$
- For all i , $w^\top x_i = (w_* - u)^\top x_i = w_*^\top x_i$, thus $\ell(x_i^\top w, y_i) = \ell(x_i^\top w_*, y_i)$

Therefore

$$\sum_{i=1}^n \ell(x_i^\top w, y_i) + \frac{\lambda}{2} \|w\|^2 \leq \sum_{i=1}^n \ell(x_i^\top w_*, y_i) + \frac{\lambda}{2} \|w_*\|^2$$

And w is an optimal solution for this problem. Since the objective is strongly convex, there is unicity of the solution and $w_* = w$

Kernelized ridge regression

Classical formulation in w :

$$w_* = \arg \min_w \frac{1}{2} \|\mathbf{y} - \mathbf{X}w\|^2 + \frac{\lambda}{2} \|w\|^2$$

Alternative formulation in α :

$$\alpha_* = \arg \min_{\alpha} \frac{1}{2} \alpha^\top (\mathbf{X}\mathbf{X}^\top + \lambda I_n) \alpha - \alpha^\top \mathbf{y}$$

Claim: These two formulations are equivalent

Proof: Set the gradient to 0, to obtain $\alpha_* = (\mathbf{X}\mathbf{X}^\top + \lambda I_n)^{-1} \mathbf{y}$, and $w_* = \mathbf{X}^\top \alpha_*$

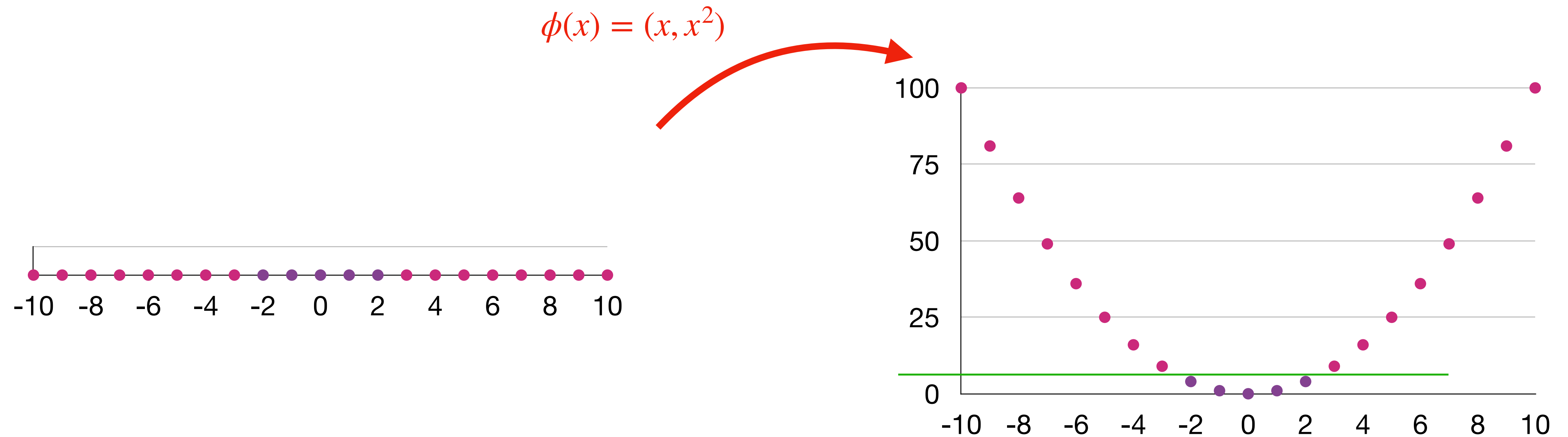
Interest:

- Computational complexity - depending on d, n
- The dual formulation only uses \mathbf{X} through the kernel matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$

Kernel matrix

$$\mathbf{K} = \mathbf{X}\mathbf{X}^\top = \begin{pmatrix} x_1^\top x_1 & x_1^\top x_2 & \cdots & x_1^\top x_n \\ x_2^\top x_1 & x_2^\top x_2 & \cdots & x_2^\top x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n^\top x_1 & x_n^\top x_2 & \cdots & x_n^\top x_n \end{pmatrix} = (x_i^\top x_j)_{i,j} \in \mathbb{R}^{n \times n}$$

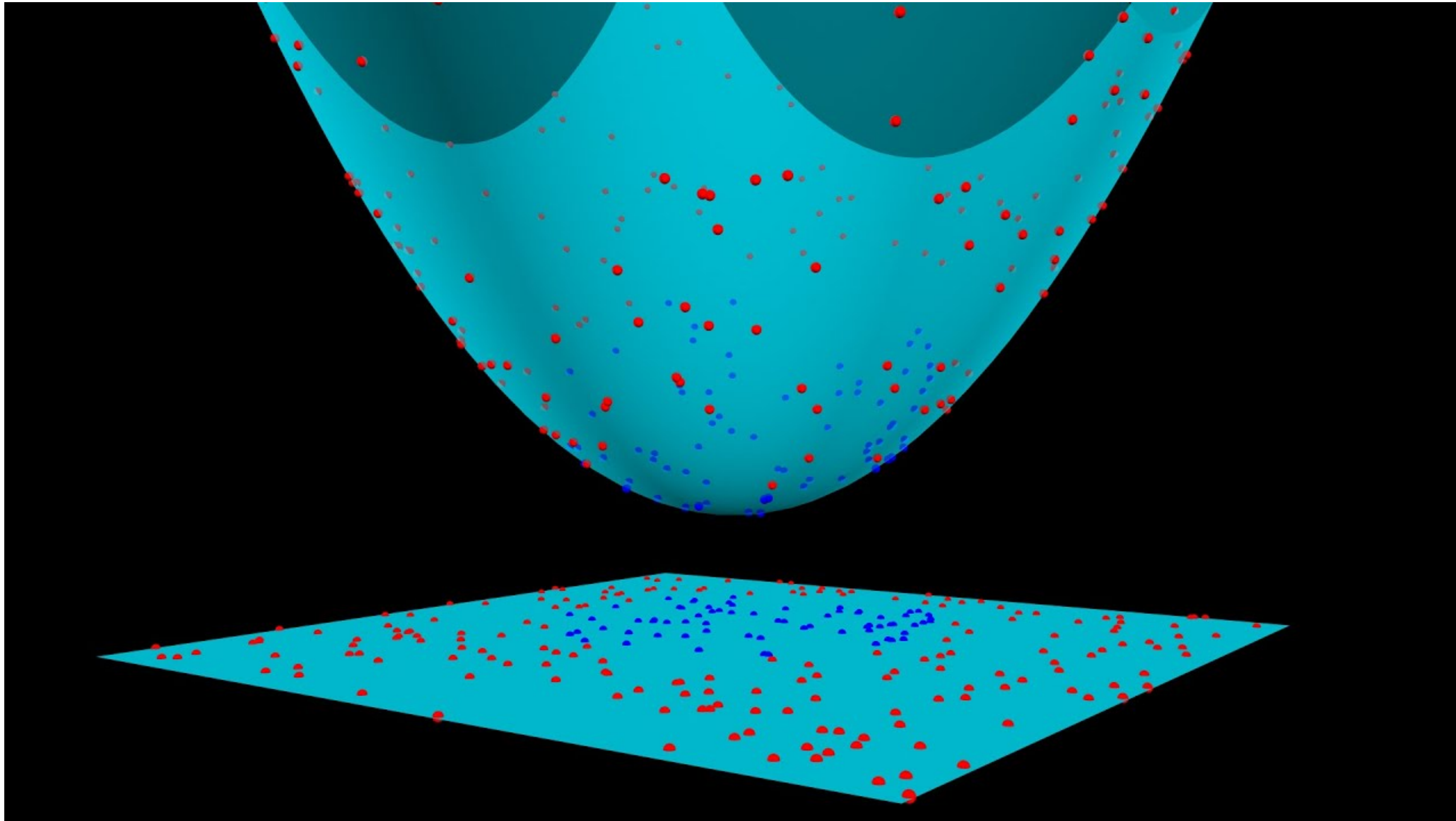
Embedding into feature spaces



Not separable by a half space

Separable by a half space

Usefulness of feature spaces



Kernel matrix with feature spaces

When a feature map $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{\tilde{d}}$ is used,

$$(x_i)_{i=1}^n \hookrightarrow (\phi(x_i))_{i=1}^n$$

The associated kernel matrix is

$$\mathbf{K} = \mathbf{\Phi} \mathbf{\Phi}^\top = \begin{pmatrix} \phi(x_1)^\top \phi(x_1) & \phi(x_1)^\top \phi(x_2) & \cdots & \phi(x_1)^\top \phi(x_n) \\ \phi(x_2)^\top \phi(x_1) & \phi(x_2)^\top \phi(x_2) & \cdots & \phi(x_2)^\top \phi(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_n)^\top \phi(x_1) & \phi(x_n)^\top \phi(x_2) & \cdots & \phi(x_n)^\top \phi(x_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Problem: when $d \ll \tilde{d}$ computing $\phi(x)^\top \phi(x')$ costs $O(\tilde{d})$ - too expensive

Kernel trick

Kernel function: $\kappa(x, x')$ such that

$$\kappa(x, x') = \phi(x)^\top \phi(x')$$

Similarity between x_i and x_j

Similarity realized as inner products in the feature space

It is equivalent to

- Directly compute $\kappa(x, x')$
- First augment the features to $\phi(x)$, then compute $\phi(x)^\top \phi(x')$

Interest: enable to compute linear classifier in high-dimensional space without having to do computation in this high-dimensional space.

Examples of kernel (easy)

1. Linear kernel: $\kappa(x, x') = x^\top x'$
 - ➡ Feature map is $\phi(x) = x$
2. Quadratic kernel: $\kappa(x, x') = (xx')^2$ for $x, x' \in \mathbb{R}$
 - ➡ Feature map is $\phi(x) = x^2$

3. Polynomial kernel

Let $x, x' \in \mathbb{R}^3$

$$\kappa(x, x') = (x_1x'_1 + x_2x'_2 + x_3x'_3)^2$$

Feature map:

$$\phi(x) = [x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3] \in \mathbb{R}^6$$

Proof:

$$\kappa(x, x') = \phi(x)^\top \phi(x')$$

$$\begin{aligned}\kappa(x, x') &= (x_1x'_1 + x_2x'_2 + x_3x'_3)^2 \\ &= (x_1x'_1)^2 + (x_2x'_2)^2 + (x_3x'_3)^2 + 2x_1x_2x'_1x'_2 + 2x_1x_3x'_1x'_3 + 2x_2x_3x'_2x'_3 \\ &= (x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)^\top (x_1'^2, x_2'^2, x_3'^2, \sqrt{2}x'_1x'_2, \sqrt{2}x'_1x'_3, \sqrt{2}x'_2x'_3)\end{aligned}$$

We obtain ϕ by identification

4. Radial basis function (RBF) kernel

Let $x, x' \in \mathbb{R}^d$

$$\kappa(x, x') = e^{-(x-x')^\top (x-x')}$$

For $x, x' \in \mathbb{R}$

$$\kappa(x, x') = e^{-(x-x')^2}$$

Feature map:

$$\phi(x) = e^{-x^2} \left(\dots, \frac{2^{k/2} x^k}{\sqrt{k!}} \dots \right) \longleftarrow \text{Infinite dimensional vector}$$

Proof: $\kappa(x, x') = e^{-x^2 - x'^2 + 2xx'}$
 $= e^{-x^2} e^{-x'^2} \sum_{k=0}^{\infty} \frac{2^k x^k x'^k}{k!}$ by the Taylor expansion of exp

$$\phi(x) = e^{-x^2} \left(\dots, \frac{2^{k/2} x^k}{\sqrt{k!}} \dots \right) \implies \phi(x)^\top \phi(x') = \kappa(x, x')$$

Interest: it cannot be represented as an inner product in a finite-dimensional space

Building new kernels from old kernels

Let κ_1, κ_2 be two kernel functions and ϕ_1, ϕ_2 the corresponding feature maps

Claim 1: Positive linear combinations of kernel are kernels

$$\kappa(x, x') = \alpha\kappa_1(x, x') + \beta\kappa_2(x, x') \text{ for } \alpha, \beta \geq 0$$

Claim 2: Product of kernels are kernels

$$\kappa(x, x') = \kappa_1(x, x')\kappa_2(x, x')$$

Interest: Building blocks to derive new kernels

Proof 1:

$$\begin{aligned}\kappa(x, x') &= \alpha\kappa_1(x, x') + \beta\kappa_2(x, x') \\ &= \alpha\phi_1(x)^\top \phi_1(x') + \beta\phi_2(x)^\top \phi_2(x') \\ &= \phi(x)^\top \phi(x')\end{aligned}$$

$$\text{where } \phi(x) = \begin{pmatrix} \sqrt{\alpha}\phi_1(x) \\ \sqrt{\beta}\phi_2(x) \end{pmatrix} \in \mathbb{R}^{d_1+d_2}$$

kernels from old kernel

s and ϕ_1, ϕ_2 the corresponding feature maps

Claim 1: Positive linear combinations of kernel are kernel

$$\kappa(x, x') = \alpha\kappa_1(x, x') + \beta\kappa_2(x, x')$$

Claim 2: Product of kernels are kernel

$$\kappa(x, x') = \kappa_1(x, x')\kappa_2(x, x')$$

old kernel

Proof 2:

$$\begin{aligned}\kappa(x, x') &= \kappa_1(x, x')\kappa_2(x, x') \\ &= \phi_1(x)^\top \phi_1(x')\phi_2(x)^\top \phi_2(x')\end{aligned}$$

Let $\phi(x) = \begin{pmatrix} (\phi_1(x))_1\phi_2(x) \\ \vdots \\ (\phi_1(x))_{d_1}\phi_2(x) \end{pmatrix} \in \mathbb{R}^{d_1 \times d_2}$, then

ding feature maps

Cl

$$\begin{aligned}\phi(x)^\top \phi(x') &= \sum_{i,j} (\phi_1(x))_i(\phi_2(x))_j(\phi_1(x'))_i(\phi_2(x'))_j \\ &= \sum_i (\phi_1(x))_i(\phi_1(x'))_i \sum_j (\phi_2(x))_j(\phi_2(x'))_j \\ &= \phi_1(x)^\top \phi_1(x')\phi_2(x)^\top \phi_2(x') = \kappa(x, x')\end{aligned}$$

el

Cl

$$\kappa(x, x') = \kappa_1(x, x')\kappa_2(x, x')$$

Mercer's condition

Question: Given a kernel function κ , how can we ensure that there exists a feature map ϕ such that

$$\kappa(x, x') = \phi(x)^\top \phi(x')$$

Answer: It is true if and only if the following Mercer's conditions are fulfilled:

- The kernel function is symmetric:

$$\forall x, x', \kappa(x, x') = \kappa(x', x)$$

- The kernel matrix is psd for all possible input sets:

$$\forall n \geq 0, \forall (x_i)_{i=1}^n, \mathbf{K} = (\kappa(x_i, x_j))_{i,j=1}^n \succeq 0$$

Predicting with kernels

Problem: we predict with $y = \phi(x)^\top w_*$ whereas $\phi(x)$ can be expensive to compute

Question: How to do a prediction only using the kernel function, without computing $\phi(x)$?

Answer: $\phi(x)^\top w_* = \phi(x)^\top \phi(\mathbf{X})^\top \alpha_* = \sum_{i=1}^n \kappa(x, x_i) \alpha_{*i}$

← We can do a prediction only using the kernel function

Important remark:

$y = \phi(x)^\top w_* = f_{w_*}(x)$

Linear prediction
in the feature space

Non linear prediction
in the \mathcal{X} space

Bonus: proof of Mercer theorem

- If κ implements an inner product then it is symmetric and the kernel matrix is psd:

$$v^\top K v = \sum_{i,j} v_i v_j \phi(x_i)^\top \phi(x_j) = (\sum_i v_i \phi(x_i))^2$$

- Define $\phi(x) = \kappa(\cdot, x)$. Define a vector space of function by taking all linear combinations $\{ \sum_i \alpha_i \kappa(\cdot, x_i) \}$. Define an inner product on this vector space by

$$\langle \sum_i \alpha_i \kappa(\cdot, x_i), \sum_j \beta_j \kappa(\cdot, x'_j) \rangle = \sum_{i,j} \alpha_i \beta_j \kappa(x_i, x'_j)$$

This is a valid inner product (symmetric, bilinear and positive definite, with equality only $\phi(x)$ is the zero function)

We have

$$\langle \phi(x), \phi(x') \rangle = \langle \kappa(\cdot, x), \kappa(\cdot, x') \rangle = \kappa(x, x')$$