Kernel Ridge Regression and the Kernel Trick

Machine Learning Course - CS-433
Oct 31, 2023
Nicolas Flammarion
EPFL

Equivalent formulations for ridge regression

Objective

$$\min_{w} \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - w^\top x_n\right)^2 + \frac{\lambda}{2} \|w\|^2$$

The solution is given by

$$\mathcal{W}_* = \frac{1}{N} \left(\frac{1}{N} \mathbf{X}^\top \mathbf{X} + \lambda I_d\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

$$\mathbf{X}^\top \in \mathbb{R}^{d \times N} \to d \times d$$

Alternatively, the solution can be written as

$$\mathcal{W}_* = \frac{1}{N} \mathbf{X}^\top \underbrace{(\frac{1}{N} \mathbf{X}\mathbf{X}^\top + \lambda I_N}_{\mathbf{X} \in \mathbb{R}^{N \times d}})^{-1} \mathbf{y}$$

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

$$P\left(QP + I_n\right) = PQP + P = \left(PQ + I_m\right) P$$

Assuming that both $QP + I_n$ and $PQ + I_m$ are invertible

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

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

$$\mathcal{W}_* = \frac{1}{N} \underbrace{(\frac{1}{N} \mathbf{X}^\top \mathbf{X} + \lambda I_d\Big)^{-1} \mathbf{X}^\top \mathbf{y}}_{\to d \times d}$$

$$\mathbf{X}^\top \in \mathbb{R}^{d \times N} \to d^{-1}$$

But it can be alternatively written as

$$\mathcal{W}_* = \frac{1}{N}\mathbf{X}^\top \underbrace{(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N)}_{\mathbf{X}\in\mathbb{R}^{N\times d}}^{-1}\mathbf{y}$$

Usefulness of the alternative form

$$\mathcal{W}_* = \underbrace{\frac{1}{N}\mathbf{X}^\top}_{d\times N} \underbrace{(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N)}_{N\times N}^{-1}\mathbf{y}$$

1. Computational complexity:

- For the original formulation $\frac{1}{N}\left(\frac{1}{N}\mathbf{X}^\top\mathbf{X} + \lambda I_d\right)^{-1}\mathbf{X}^\top\mathbf{y}, O\left(d^3 + Nd^2\right)$
- For the new formulation $\frac{1}{N}\mathbf{X}^\top\left(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N\right)^{-1}\mathbf{y}, O\left(N^3 + dN^2\right)$

$\Rightarrow$ Depending on d, N one formulation may be more efficient than the other

2. Structural difference:

$$w_* = \mathbf{X}^\top\alpha_* \text{ where } \alpha_* = \frac{1}{N}\left(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N\right)^{-1}\mathbf{y}$$

$\Rightarrow w_* \in \operatorname{span}\{x_1, \cdots, x_N\}$
These two insights are fundamental to understanding the kernel trick

Representer Theorem

Claim: For any loss function ℓ, there exists $\alpha_* \in \mathbb{R}^N$ such that

$$w_* := \mathbf{X}^\top\alpha_* \in \arg\min_w \frac{1}{N}\sum_{n=1}^N \ell\left(x_n^\top w, y_n\right) + \frac{\lambda}{2}\|w\|^2$$

Meaning: There exists an optimal solution within $\operatorname{span}\{x_1, \cdots, x_N\}$
Consequence: This is more general than LS, enabling the kernel trick to various problems, including Kernel SVM, Kernel LS, and Kernel Principal Component Analysis

Proof of the representer theorem

Let w_* be an optimal solution of $\min_w \frac{1}{N}\sum_{n=1}^N \ell\left(x_n^\top w, y_n\right) + \frac{\lambda}{2}\|w\|^2$
We can always rewrite w_* as $w_* = \sum_{n=1}^N \alpha_n x_n + u$ where $u^\top x_n = 0$ for all n
Let's define $w = w_* - u$

- $\|w_*\|^2 = \|w\|^2 + \|u\|^2$, thus $\|w\|^2 \leq \|w_*\|^2$
- For all $n, w^\top x_n = (w_* - u)^\top x_n = w_*^\top x_n$, thus $\ell\left(x_n^\top w, y_n\right) = \ell\left(x_n^\top w_*, y_n\right)$

Therefore

$$\frac{1}{N} \sum_{n=1}^{N} \ell\left(x_n^\top w, y_n\right) + \frac{\lambda}{2}\|w\|^2 \leq \frac{1}{N} \sum_{n=1}^{N} \ell\left(x_n^\top w_*, y_n\right) + \frac{\lambda}{2}\|w_*\|^2$$

And w is an optimal solution for this problem.

Kernelized ridge regression

Classic formulation in w :

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

Alternative formulation in α :

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

Claim: These two formulations are equivalent

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

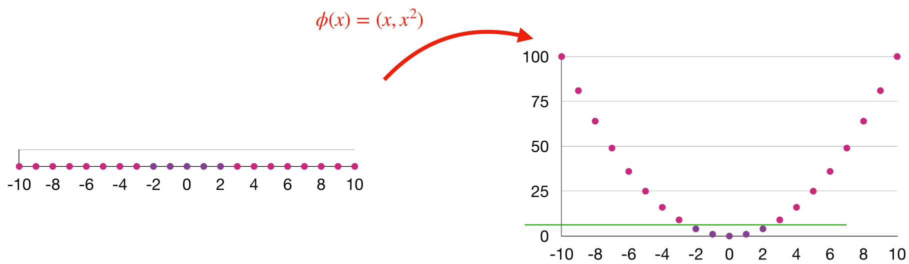
Key takeaways:

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

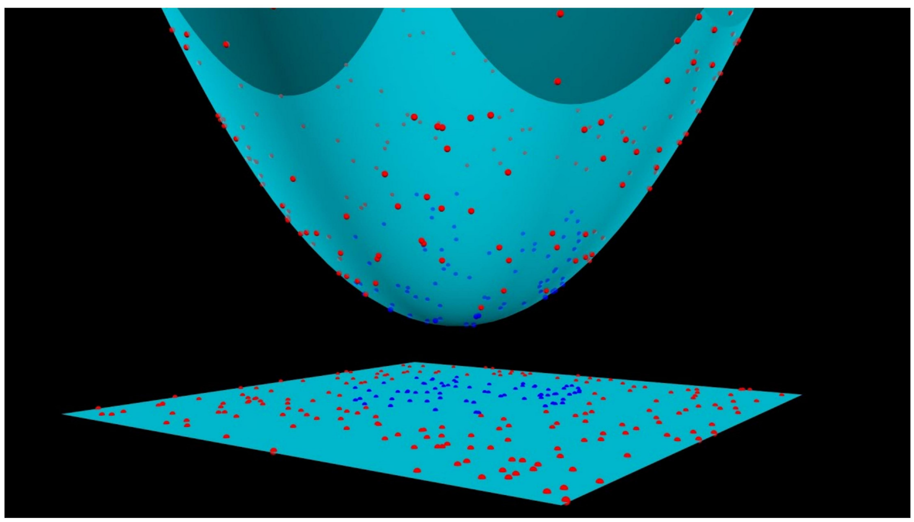
Kernel matrix

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

Embedding into feature spaces



Usefulness of feature spaces



Kernel matrix with feature spaces

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

$$(x_n)_{n=1}^N \hookrightarrow (\phi(x_n))_{n=1}^N$$

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 $d \lll \tilde{d}$ 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 and x'

Similarity realized as an inner product in the feature space

It is equivalent to

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

Purpose: enable computation of linear classifiers in high-dimensional space without performing computations in this high-dimensional space directly.

Predicting with kernels

Problem: The prediction is $y = \phi(x)^{\top} w_*$ but computing $\phi(x)$ can be expensive

Question: How can we make predictions using only the kernel function, without the need to compute $\phi(x)$?

Answer: $\phi(x)^{\top} w_* = \phi(x)^{\top} \phi(\mathbf{X})^{\top} \alpha_* = \sum_{n=1}^{N} \kappa(x, x_n) \alpha_{*n}$ We can do a prediction only using the kernel function

Important remark:

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

Linear prediction
in the feature space Non linear prediction
in the $\mathscr{X}$ space

Examples of kernel (easy)

1. Linear kernel: $\kappa(x, x') = x^{\top} x'$

$\rightarrow$ Feature map is $\phi(x) = x$

2. Quadratic kernel: $\kappa(x, x') = (xx')^2$ for $x, x' \in \mathbb{R}$

$\Rightarrow$ Feature map is $\phi(x) = x^2$

3. Polynomial kernel

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

$$\kappa(x, x') = (x_1 x_1' + x_2 x_2' + x_3 x_3')^2$$

Feature map:
Proof:

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

$$\begin{aligned}
\kappa(x, x') &= \phi(x)^\top \phi(x') \\
\kappa(x, x') &= (x_1 x_1' + x_2 x_2' + x_3 x_3')^2 \\
&= (x_1 x_1')^2 + (x_2 x_2')^2 + (x_3 x_3')^2 + 2 x_1 x_2 x_1' x_2' + 2 x_1 x_3 x_1' x_3' + 2 x_2 x_3 x_2' x_3' \\
&= \left(x_1^2, x_2^2, x_3^2, \sqrt{2} x_1 x_2, \sqrt{2} x_1 x_3, \sqrt{2} x_2 x_3\right)^\top \left(x_1'^2, x_2'^2, x_3'^2, \sqrt{2} x_1' x_2', \sqrt{2} x_1' x_3', \sqrt{2} x_2' x_3'\right)
\end{aligned}$$

We obtain ϕ by identification

4. Radial basis function (RBF) kernel

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

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

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

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

Feature map:

$$\phi(x) = e^{-x^2} \left(\cdots, \frac{2^{k/2} x^k}{\sqrt{k!}} \cdots\right)$$

Proof: $\quad \kappa(x, x') = e^{-x^2 - x^2 + 2xx'}$

$$= e^{-x^2} e^{-x'^2} \sum_{k=0}^{\infty} \frac{2^k x^k x'^k}{k!} \text{ by the Taylor expansion of } \exp$$

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

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

Building new kernels from existing 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: Products of kernels are kernels

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

Objective: To provide building blocks for deriving new kernels

Proof 1:

$$\begin{aligned}
&\kappa(x, x') = \alpha\kappa_1(x, x') + \beta\kappa_2(x, x') \\
&= \alpha\phi_1(x)^\top\phi_1(x') + \beta\phi_2(x)^\top\phi_2(x') \\
&= \phi(x)^\top\phi(x') \\
&\text{where } \phi(x) = \begin{pmatrix} \sqrt{\alpha}\phi_1(x) \\ \sqrt{\beta}\phi_2(x) \end{pmatrix} \in \mathbb{R}^{d_1+d_2}
\end{aligned}$$

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: Products of kernels are kernel

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

Proof 2:

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

$$= \phi_1(x)^\top\phi_1(x')\,\phi_2(x)^\top\phi_2(x')$$

Let

$\phi(x)^\top = \left((\phi_1(x))_1(\phi_2(x))_1, \cdots, (\phi_1(x))_1(\phi_2(x))_{d_2}, \cdots, (\phi_1(x))_{d_1}(\phi_2(x))_1, \cdots, (\phi_1(x))_{d_1}(\phi_2(x))_{d_2}\right) \in \mathbb{R}^{d_1 d_2}$ then

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

Claim 2: Products of kernels are kernel

$$\kappa\left(x, x^{\prime}\right)=\kappa_{1}\left(x, x^{\prime}\right) \kappa_{2}\left(x, x^{\prime}\right)$$

Mercer's condition

Question: Given a kernel function κ, how can we ensure the existence of a feature map ϕ such that

$$\kappa\left(x, x^{\prime}\right)=\phi(x)^{\top} \phi\left(x^{\prime}\right)$$

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

- The kernel function is symmetric:

$$\forall x, x^{\prime}, \kappa\left(x, x^{\prime}\right)=\kappa\left(x^{\prime}, x\right)$$

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

$$\forall N \geq 0, \forall\left(x_{n}\right)_{n=1}^{N}, \quad \mathbf{K}=\left(\kappa\left(x_{i}, x_{j}\right)\right)_{i, j=1}^{N} \geqslant 0$$

Recap

- Many algorithms (SVM, Least Squares, PCA, etc) can be rewritten so that they rely only on inner products between data points $\left(\mathbf{X X}^{\top}\right)$
- This motivates to generalize the inner products with kernels to make the model non-linear in the input space
- This can improve efficiency by avoiding a direct computation of feature maps $\phi(x)$ of a potentially high-dimensional space
- To predict with kernels, you need to compute the similarity $k\left(x, x_{i}\right)$ of a new point x with every training point x_{i}
- You can derive new kernels using a certain set of properties

Bonus: proof of Mercer theorem

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

$$v^{\top} \mathrm{K} v=\sum_{i, j} v_{i} v_{j} \phi\left(x_{i}\right)^{\top} \phi\left(x_{j}\right)=\left\|\sum_{i} v_{i} \phi\left(x_{i}\right)\right\|^{2}$$

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

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

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

Consequently

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