Kernel Ridge Regression and the Kernel Trick

Machine Learning Course - CS-433
28 Oct 2025
Robert West
(Slide credits: Nicolas Flammarion)

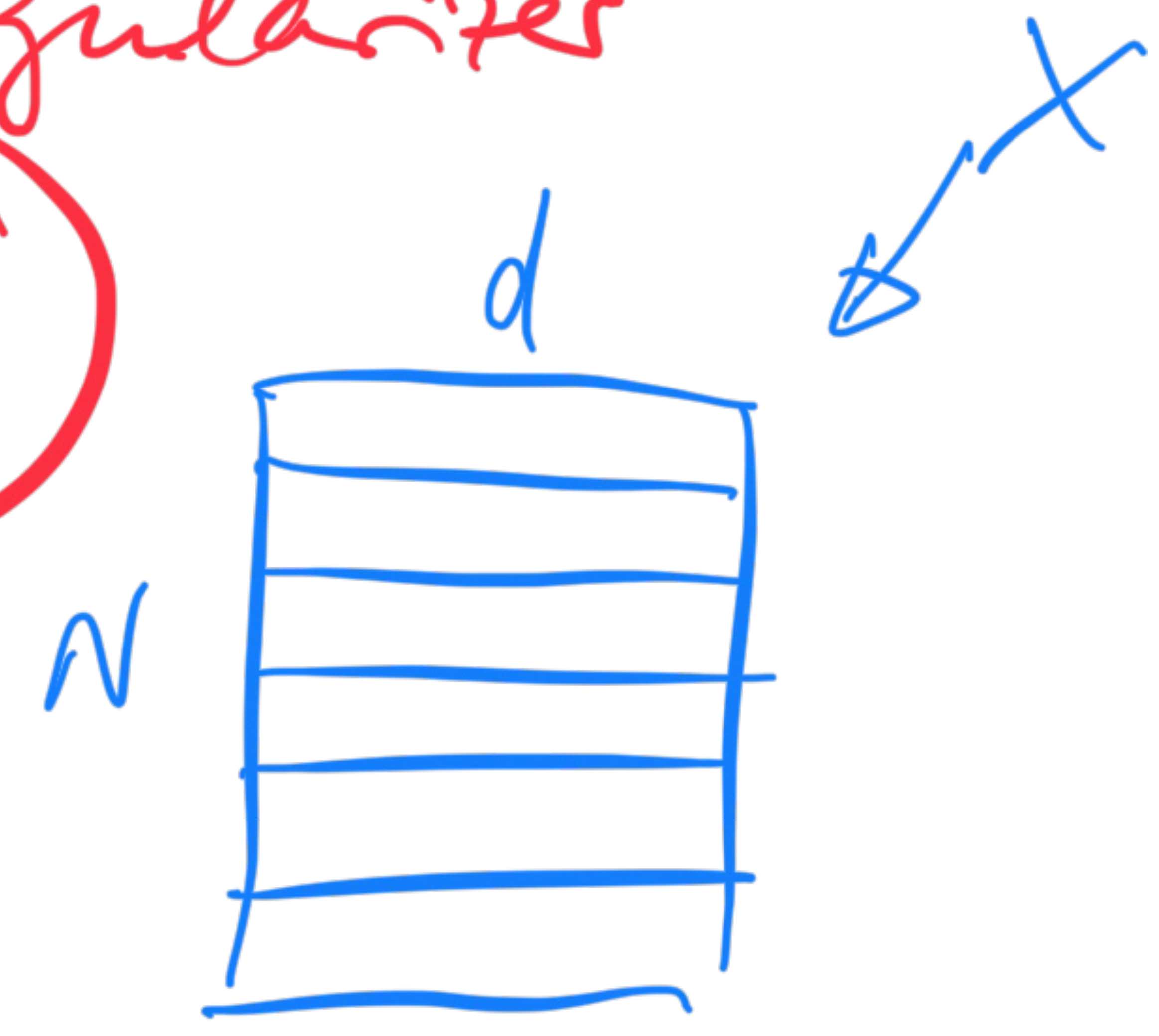
EPFL

Equivalent formulations for ridge regression

Squared loss $\quad$ L_2 regularizer

Objective

$$\min_{w} \overbrace{\frac{1}{2N}\sum_{n=1}^{N}(\underbrace{y_n - w^\top x_n}_{\text{error}})^2}^{\text{squared loss}} + \frac{\lambda}{2}\|w\|^2$$



The solution is given by

covariance matrix

$$w_* = \frac{1}{N}\Big(\underbrace{\frac{1}{N}\mathbf{X}^\top\mathbf{X} + \lambda I_d}\Big)^{-1}\mathbf{X}^\top\mathbf{y}$$

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

Alternatively, the solution can be written as kernel matrix

$$w_* = \frac{1}{N}\mathbf{X}^\top\Big(\underbrace{\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N}\Big)^{-1}\mathbf{y}$$

$$\mathbf{X} \in \mathbb{R}^{N\times d} \quad \rightarrow \quad N \times N$$

e regression

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

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

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

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

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

$\|^2$

$$w_* = \frac{1}{N}\Big(\underbrace{\frac{1}{N}\mathbf{X}^\top\mathbf{X} + \lambda I_d}\Big)^{-1}\mathbf{X}^\top\mathbf{y}$$

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

But it can be alternatively written as

$$w_* = \frac{1}{N}\mathbf{X}^\top\Big(\underbrace{\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N}\Big)^{-1}\mathbf{y}$$

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

Usefulness of the alternative form

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

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

     $N\times N$

   ➡ Depending on d, N one formulation may be more efficient than the other

   $\alpha_* \in \mathbb{R}^N$

2. Structural difference:

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

   ➡ $w_* \in \text{span}\{x_1, \cdots, x_N\}$

These two insights are fundamental to understanding the **kernel trick**

Representer Theorem

previous slide:
$\ell(x^\top w, y) = (y - x^\top w)^2$
$|y - x^\top w|$

Claim: For any loss function ℓ, and any minimizer

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

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

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

Meaning: There exists an optimal solution within span$\{x_1, \cdots, x_N\}$

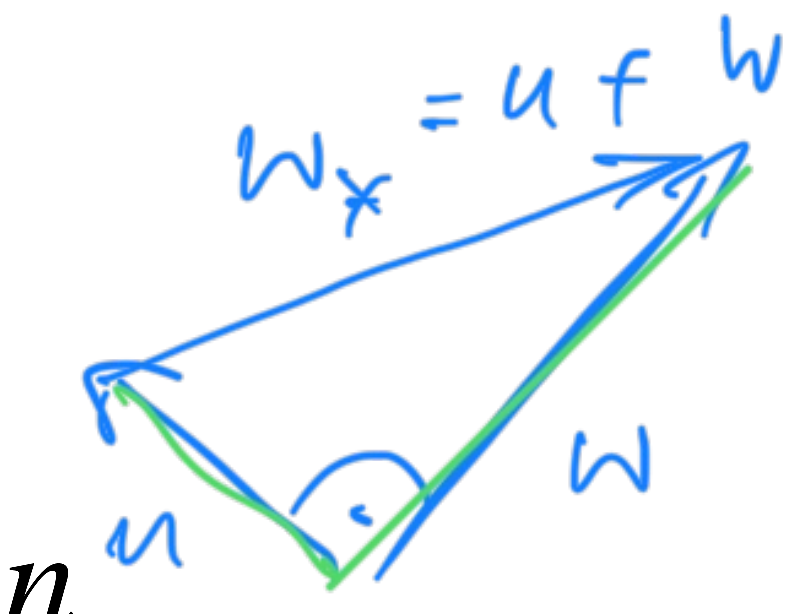
Consequence: This is more general than LS, enabling the kernel trick to various problems, including Kernel SVM, Kernel LS, and Kernel Principal Component Analysis

Wahba, G. *Spline models for observational data*. SIAM. 1990
Schölkopf, B., Herbrich, R., & Smola, A. J. *A generalized representer theorem.* COLT*. 2001*

Proof of the representer theorem

Let w_* be an optimal solution of $\min_w \frac{1}{N}\sum_{n=1}^N \ell(x_n^\top w, y_n) + \frac{\lambda}{2}\|w\|^2$

We can always rewrite w_* as $w_* = \underbrace{\sum_{n=1}^N \alpha_n x_n}_{w} + u$ where $u^\top x_n = 0$ for all n



$\|w_*\| \leq \|w\|_1 + \|u\|_1$

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 $\ell(x_n^\top w, y_n) = \ell(x_n^\top w_*, y_n)$

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

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

Hence $\|w_*\|^2 = \|w\|^2$ and $u = 0$. Thus $w_* = w = \sum_{n=1}^N \alpha_n x_n$

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:

$$\alpha_* = \arg\min_\alpha \overbrace{\frac{1}{2}\alpha^\top(\underbrace{\tfrac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N}_{M})\alpha}^{\in \mathbb{R}} - \frac{1}{N}\alpha^\top\mathbf{y}$$

Claim: These two formulations are equivalent

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

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

$$= \frac{1}{N}X^\top \underbrace{\left(\frac{1}{N}XX^\top + \lambda I_N\right)^{-1} y}_{\alpha_*}$$

$$M\alpha - \frac{1}{N}y \stackrel{!}{=} 0$$

$$M\alpha = \frac{1}{N}y$$

$$\alpha = \frac{1}{N}M^{-1}y$$

Key takeaways:

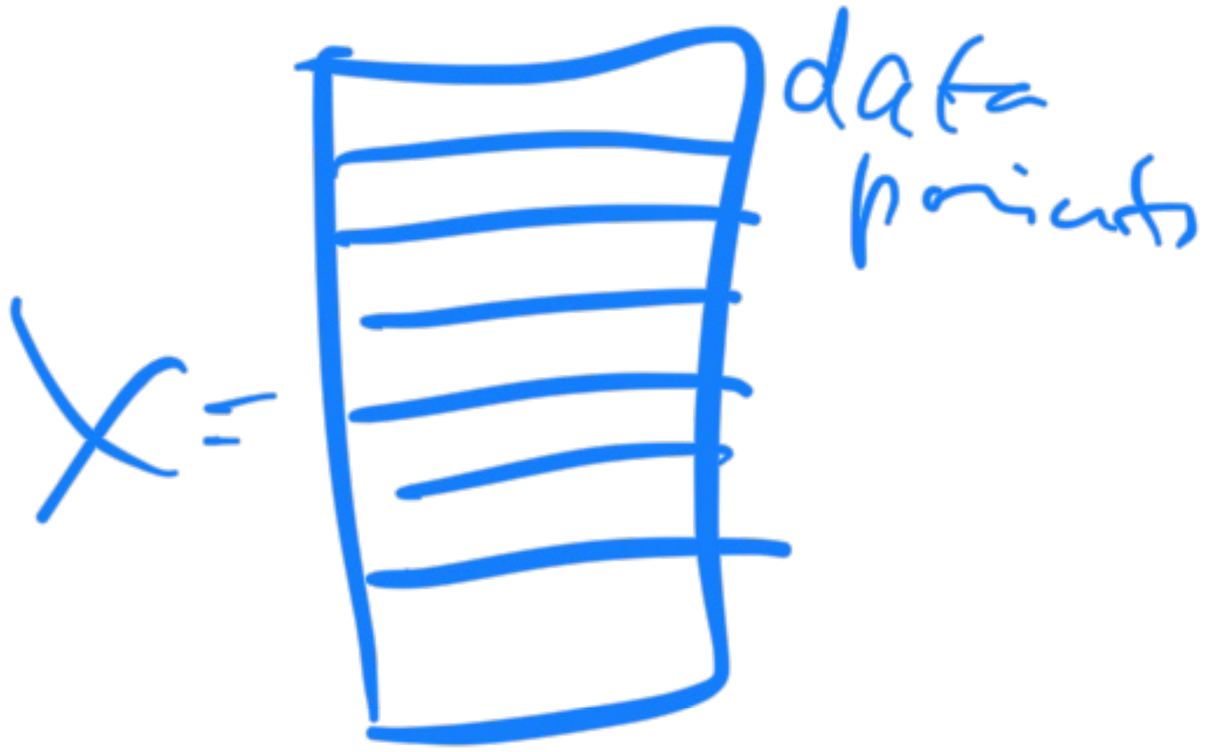
- Computational complexity - depending on d, N
- The dual formulation only uses $\mathbf{X}$ through the kernel matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^\top$

Kernel matrix

$x_i^\top x_j = x_j^\top x_i$

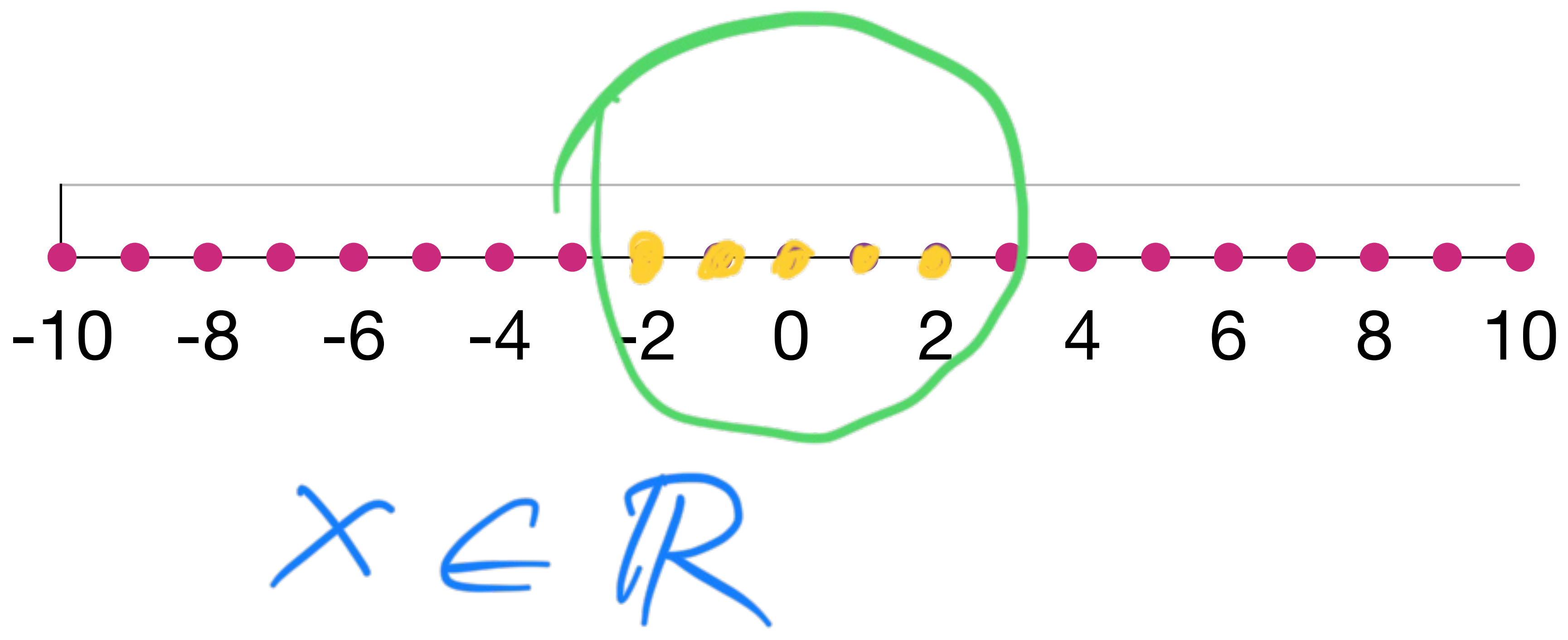
$u^\top v = \|u\| \cdot \|v\| \cdot \cos(\alpha)$

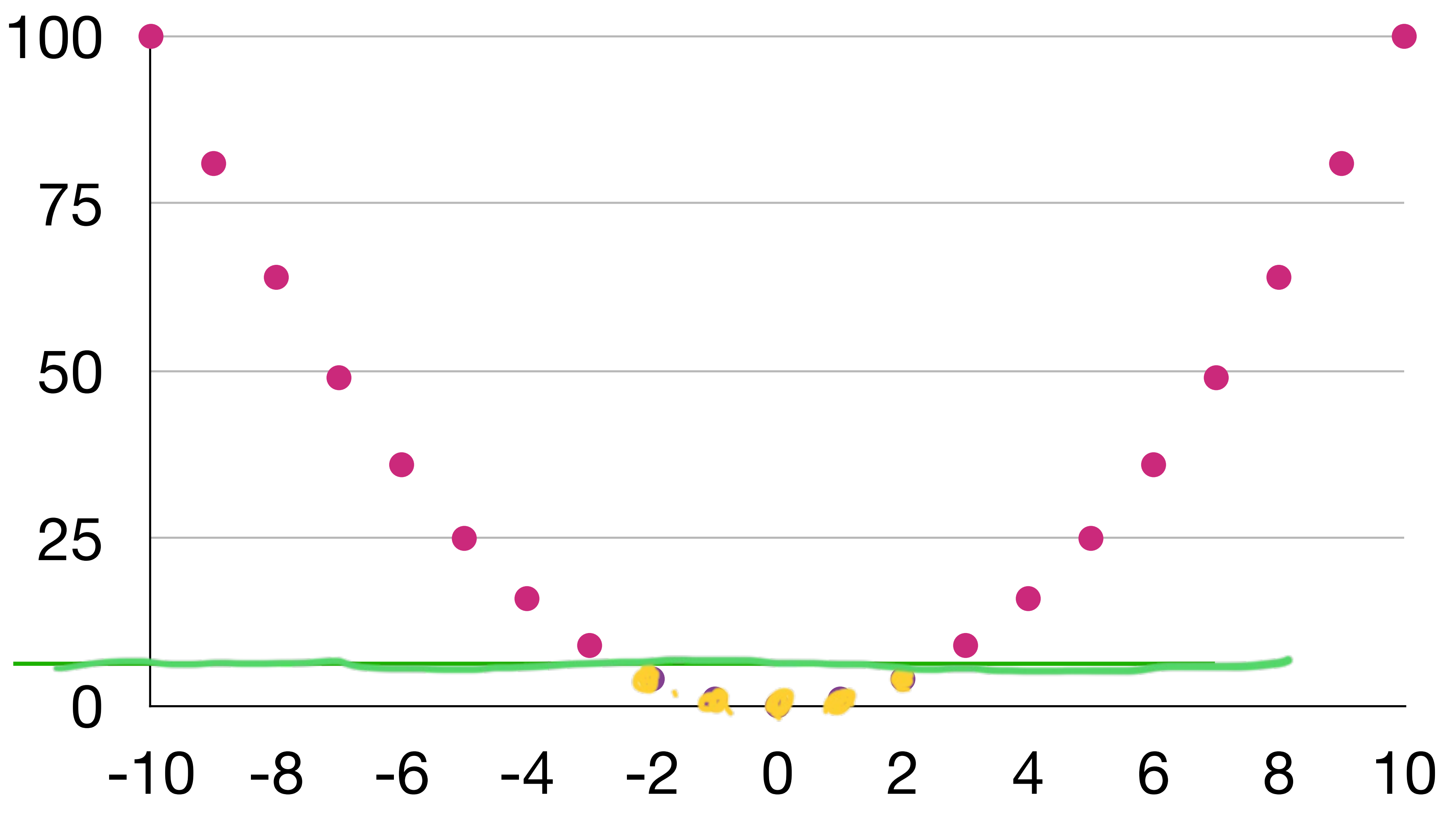
$$\mathbf{K} = \mathbf{X}\mathbf{X}^\top = \begin{pmatrix} x_1^\top x_1 & x_1^\top x_2 & \cdots & x_1^\top x_N \\ x_2^\top x_1 & x_2^\top x_2 & \cdots & x_2^\top x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_N^\top x_1 & x_N^\top x_2 & \cdots & x_N^\top x_N \end{pmatrix} = (x_i^\top x_j)_{i,j} \in \mathbb{R}^{N\times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



Embedding into feature spaces

$\phi(x) = (x, x^2)$

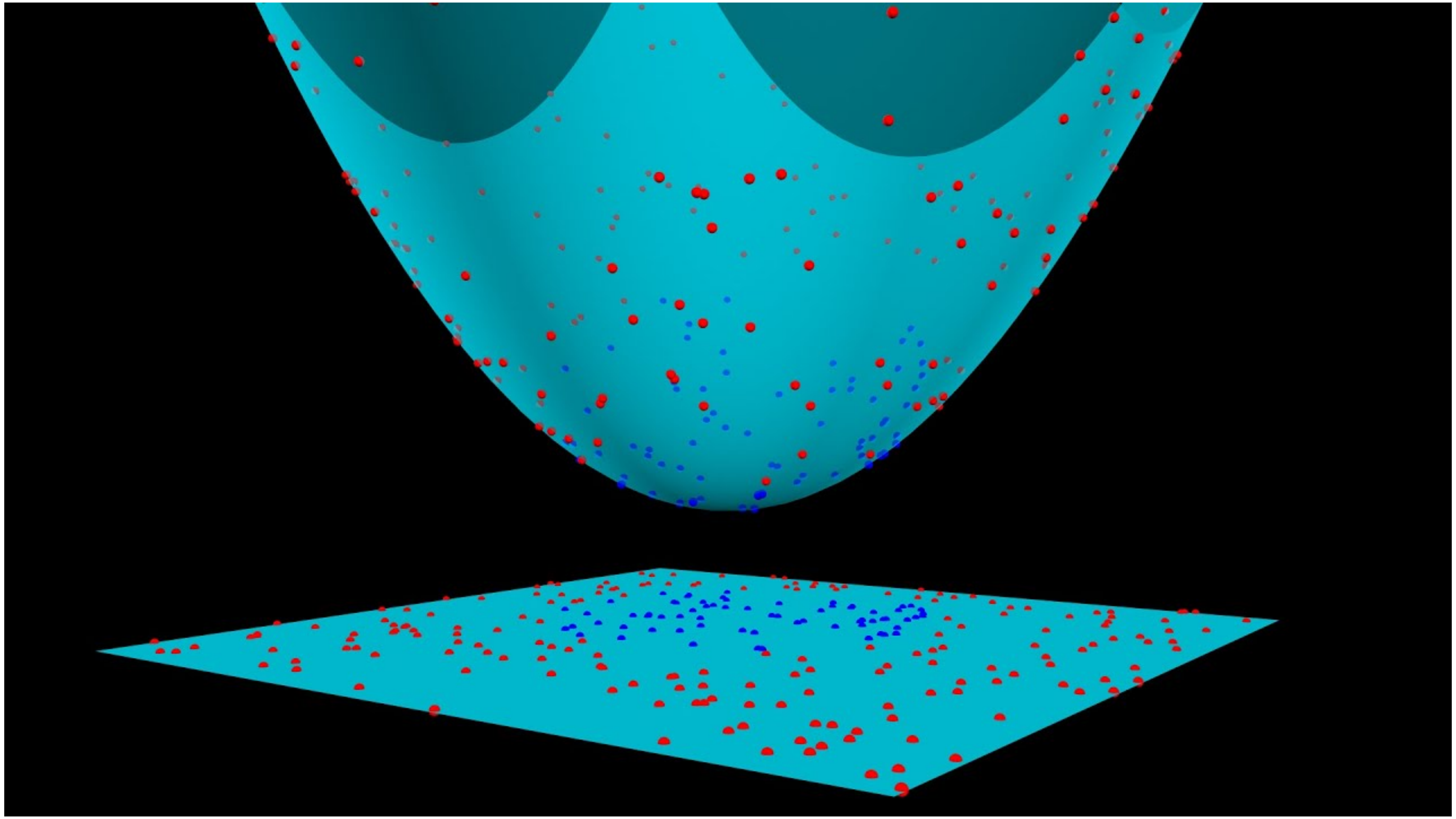




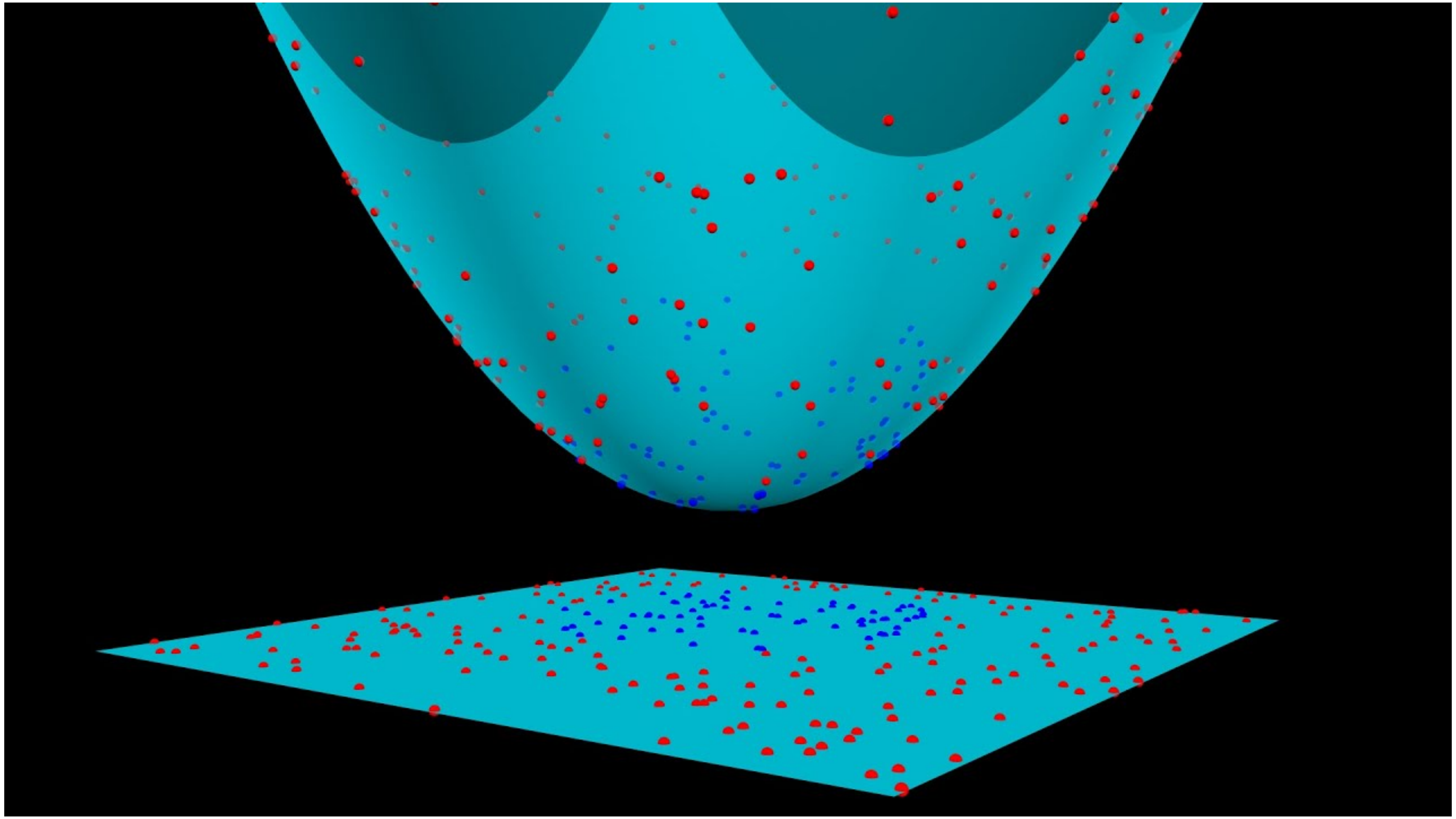
Not separable by a half space

Separable by a half space

Usefulness of 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,

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

The associated kernel matrix is

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

<u>Problem</u>: 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 $\phi(x)^\top \phi(x')$

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

$\phi(x) =$

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 $\phi(x)$?

$w_* = \phi(X)^\top \alpha_*$

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:

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

Linear prediction in the feature space

Non linear prediction in the $\mathcal{X}$ 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 $\phi(x) = x^2$

$$\phi(x)^T \phi(x') = x^2 \cdot x'^2 = (x \cdot x')^2 = \kappa(x, x')$$

3. Polynomial kernel

$(x^T x')^d$

$(x^T x' + 1)^d$

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

$$\kappa(x, x') = (\underbrace{x_1x'_1 + x_2x'_2 + x_3x'_3}_{x^T x'})^{2 \to d}$$

<u>Feature map</u>:

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

<u>Proof</u>:

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

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

$$= (x_1x'_1)^2 + (x_2x'_2)^2 + (x_3x'_3)^2 + 2x_1x_2x'_1x'_2 + 2x_1x_3x'_1x'_3 + 2x_2x_3x'_2x'_3$$

$$= \left(x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3\right)^\top \left(x_1'^2, x_2'^2, x_3'^2, \sqrt{2}x'_1x'_2, \sqrt{2}x'_1x'_3, \sqrt{2}x'_2x'_3\right)$$

We obtain ϕ by identification

4. Radial basis function (RBF) kernel

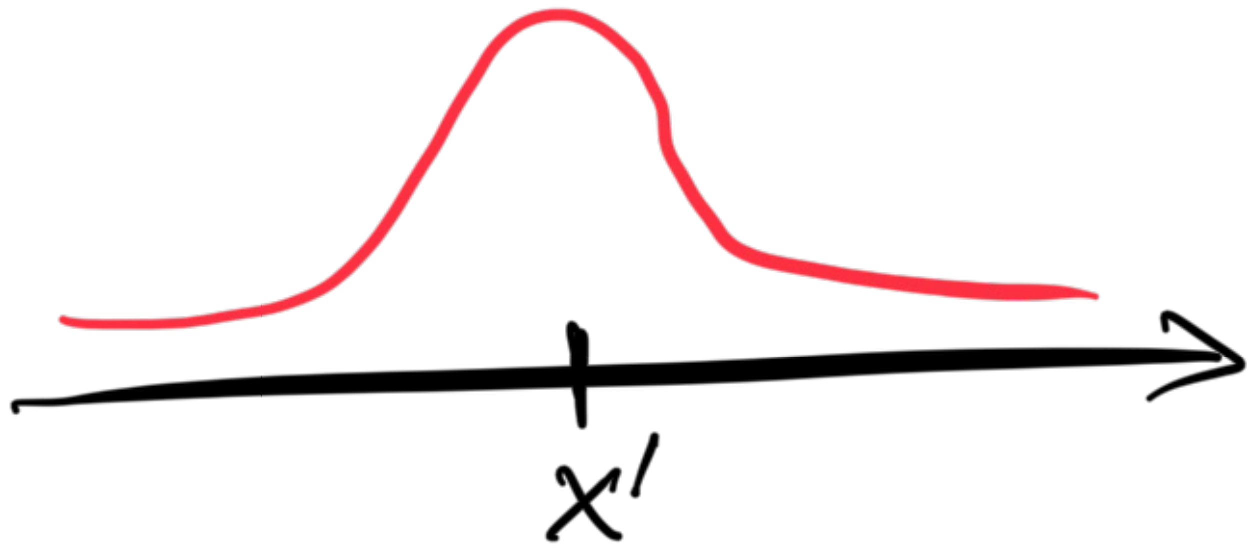
$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

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

$$\kappa(x, x') = e^{-\underbrace{(x-x')^\top (x-x')}_{\|x-x'\|^2}}$$

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

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



<u>Feature map</u>:

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

Infinite dimensional vector

$$e^{2xx'} = \sum_{k=0}^{\infty} \frac{2^k x^k x'^k}{k!} = \sum_{k=0}^{\infty} \frac{2^{\frac{k}{2}} x^k}{\sqrt{k!}} \cdot \frac{2^{\frac{k}{2}} x'^k}{\sqrt{k!}}$$

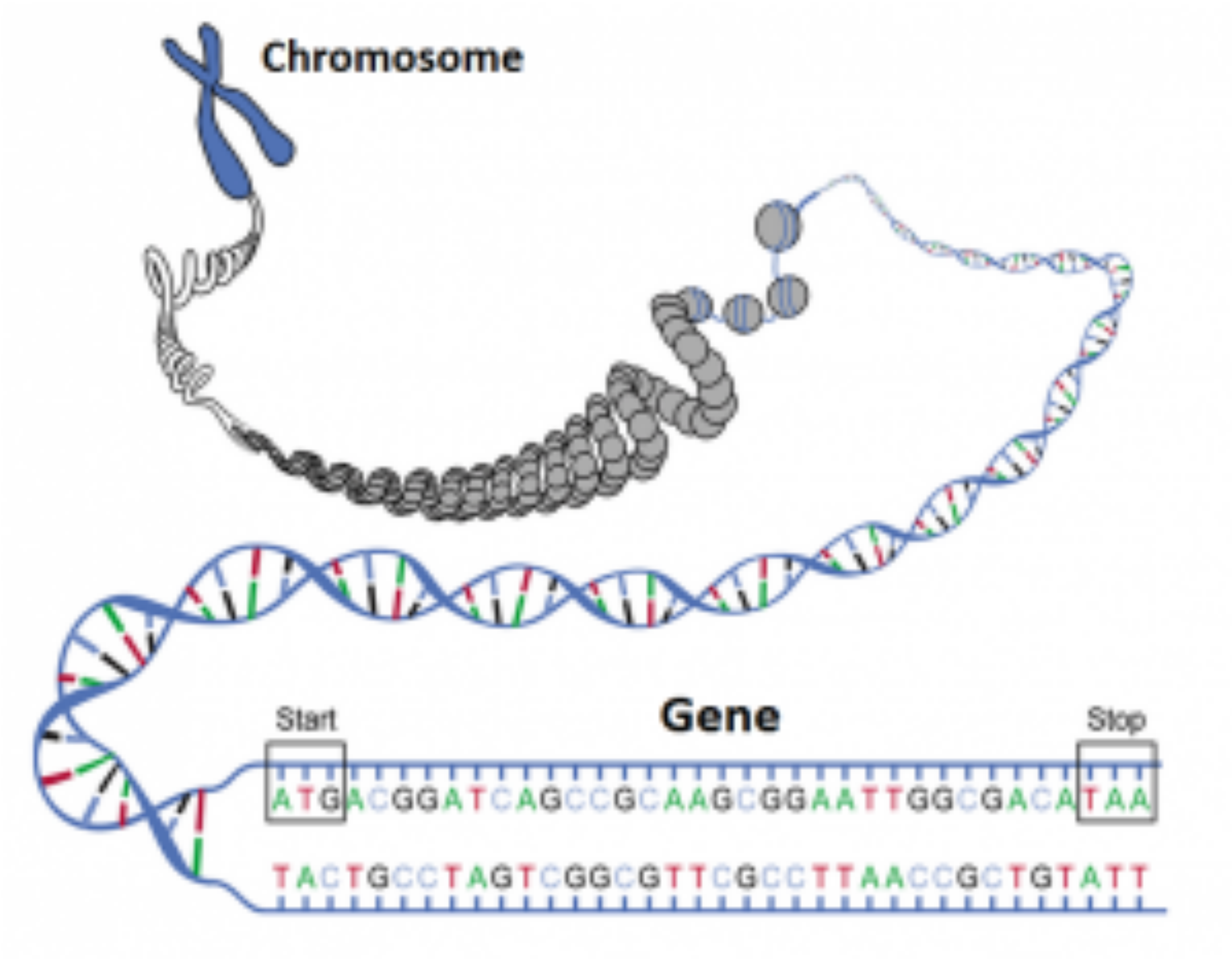
<u>Proof</u>: $\kappa(x, x') = e^{-x^2 - x'^2 + 2xx'}$

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

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

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

4. String kernels



$$\varphi_u : \begin{cases} \Sigma^n \to \mathbb{R}^{\Sigma^n} \\ s \mapsto \sum_{\mathbf{i}:u=s_{\mathbf{i}}} \lambda^{l(\mathbf{i})} \end{cases}$$

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:

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

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: 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

$$\phi(x) = \big((\phi_1(x))_i(\phi_2(x))_j\big)_{1\le i\le d_1, 1\le j\le d_2} \in \mathbb{R}^{d_1 d_2}$$
$$= \big((\phi_1(x))_1(\phi_2(x))_1, \cdots, (\phi_1(x))_1(\phi_2(x))_{d_2}, \cdots, (\phi_1(x))_{d_1}(\phi_2(x))_1, \cdots, (\phi_1(x))_{d_1}(\phi_2(x))_{d_2}\big)^\top$$

then

$$\phi(x)^\top\phi(x') = \textstyle\sum_l (\phi(x))_l(\phi(x'))_l$$
$$= \textstyle\sum_{i,j} (\phi_1(x))_i(\phi_2(x))_j(\phi_1(x'))_i(\phi_2(x'))_j$$
$$= \textstyle\sum_i (\phi_1(x))_i(\phi_1(x'))_i \sum_j (\phi_2(x))_j(\phi_2(x'))_j$$
$$= \phi_1(x)^\top\phi_1(x')\phi_2(x)^\top\phi_2(x') = \kappa(x,x')$$

Claim 2: Products of kernels are kernel

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

Mercer's condition

Question: Given a kernel function κ, how can we ensure the existence of 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:

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

Recap

- Many algorithms (SVM, Least Squares, PCA, etc) can be rewritten so that they rely only on inner products between data points ($\mathbf{X}\mathbf{X}^\top$)
- 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 $k(x, x_i)$ 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:
$$v^\top \mathbf{K} v = \sum_{i,j} v_i v_j \phi(x_i)^\top \phi(x_j) = \| \sum_i v_i \phi(x_i) \|^2$$
- Define $\phi(x) = \kappa(\cdot, x)$. Define a vector space of functions by considering 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 holding only if $\phi(x)$ is the zero function)

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