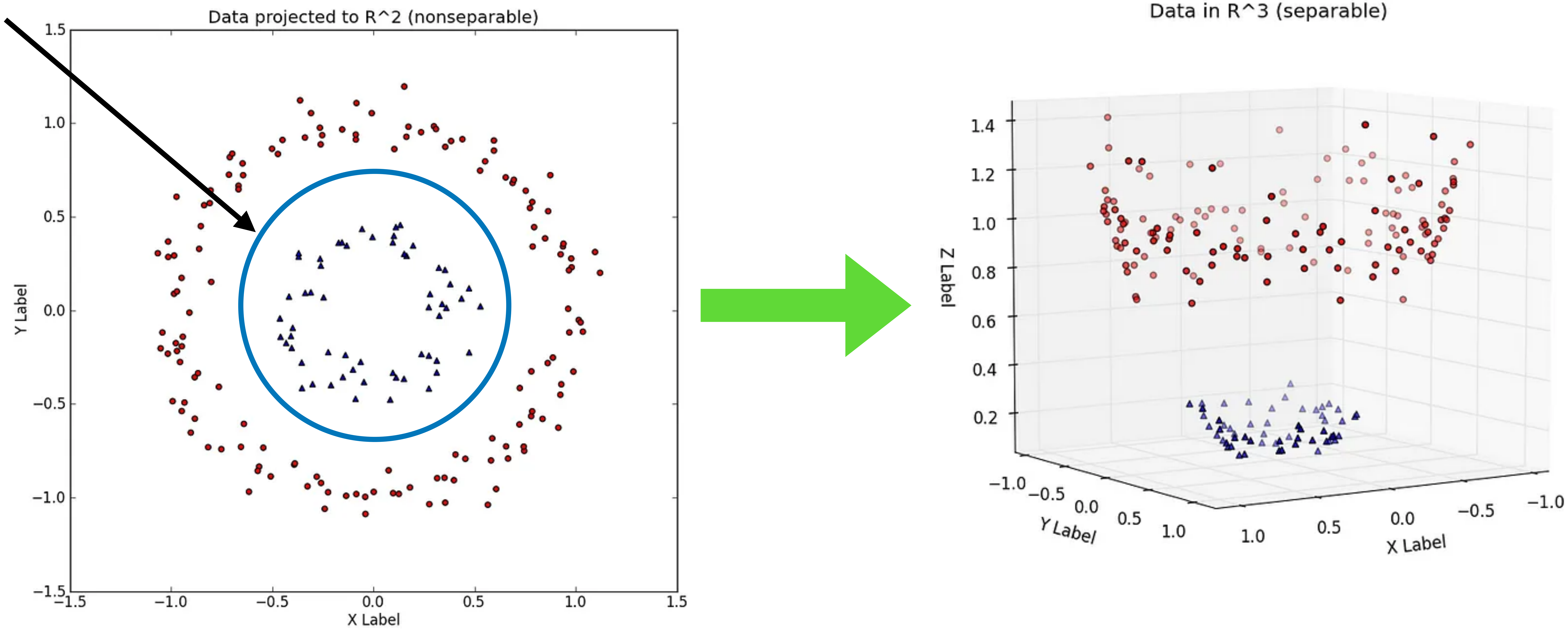
Kernel

Objective today

Use kernels to design nonlinear regression & classification models

Goal: Non-linear decision boundary



Outline

1. Kernel

2. Kernel trick and Kernel regression

3. Kernel SVM

Common Kernels

Linear kernel: $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$

Polynomial kernel: $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^p$

Gaussian kernel (aka RBF):

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\|\mathbf{x} - \mathbf{z}\|_2^2 / \sigma^2\right)$$

Well-defined Kernels

Given any symmetric function $k(\mathbf{x}, \mathbf{z})$, can it be used as a kernel?

$$\exists \phi, \text{ s.t., } k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z}), \forall \mathbf{x}, \mathbf{z}$$

$$\Updownarrow$$

$$\exists \phi, \text{ s.t., } \forall \mathbf{x}_1, \ldots, \mathbf{x}_m, \text{ the kernel matrix } K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_2) & \ldots & k(\mathbf{x_1}, \mathbf{x_m}) \\ k(\mathbf{x}_2, \mathbf{x}_1) & \ldots & k(\mathbf{x_2}, \mathbf{x_m}) \\ \ldots & \ldots & \ldots \\ k(\mathbf{x}_m, \mathbf{x}_1) & \ldots & k(\mathbf{x_m}, \mathbf{x_m}) \end{bmatrix} \text{ is PSD}$$

Construction of well-defined kernels

Kernels built by recursively applying the following one or more rules are well-defined kernels

Given well-defined k_1, k_2

1. $k(\mathbf{x}, \mathbf{z}) = ck_1(\mathbf{x}, \mathbf{z}), c > 0$
2. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$
3. $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) \cdot k_2(\mathbf{x}, \mathbf{z})$
4. $k(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{z}) f(\mathbf{z})$
5. $k(\mathbf{x}, \mathbf{z}) = \exp(k_1(\mathbf{x}, \mathbf{z}))$

… (see lecture note)

In class exercise:

Given $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$ being well defined,

Prove Gaussian kernel
$\exp\left(-\|\mathbf{x} - \mathbf{z}\|_2^2 / \sigma^2\right)$ is well defined

Hint:

$$\exp(-\mathbf{x}^\top \mathbf{x}/\sigma^2) \cdot \exp(2\mathbf{x}^\top \mathbf{z}/\sigma^2) \cdot \exp(-\mathbf{z}^\top \mathbf{z}/\sigma^2)$$

Outline

1. Kernel

2. Kernel trick and Kernel regression

3. Kernel SVM

Kernel Trick

We wanted to do linear regression in the new features $\phi(\mathbf{x_1}), \ldots, \phi(\mathbf{x_n})$,

BUT, $\phi(\mathbf{x})$ can be very high-dim or even infinite-dim….

Solution: recall linear regression can be done by just using inner product of two features!

The kernel trick

A recipe:

1. Write the learning algorithm in terms of $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$

2. Define a kernel $k(\mathbf{x}, \mathbf{z})$ (e.g., Gaussian kernel, poly kernel)

3. Replace all $\langle \mathbf{x}, \mathbf{z} \rangle$ operation in the Alg by $k(\mathbf{x}, \mathbf{z})$

Kernel ridge regression

1. Recall linear regression can be done via just using inner product:

$$\alpha = \left(X^\top X + \lambda I\right)^{-1} Y \in \mathbb{R}^n$$

2. Define a kernel, e.g., $k(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|_2^2/\sigma^2)$

3. Replace $X^\top X$ by a kernel matrix K

$$K \in \mathbb{R}^{n\times n}, \;\; K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$$

Kernel ridge regression

In test time, recall linear regression makes prediction at $\mathbf{x}$:

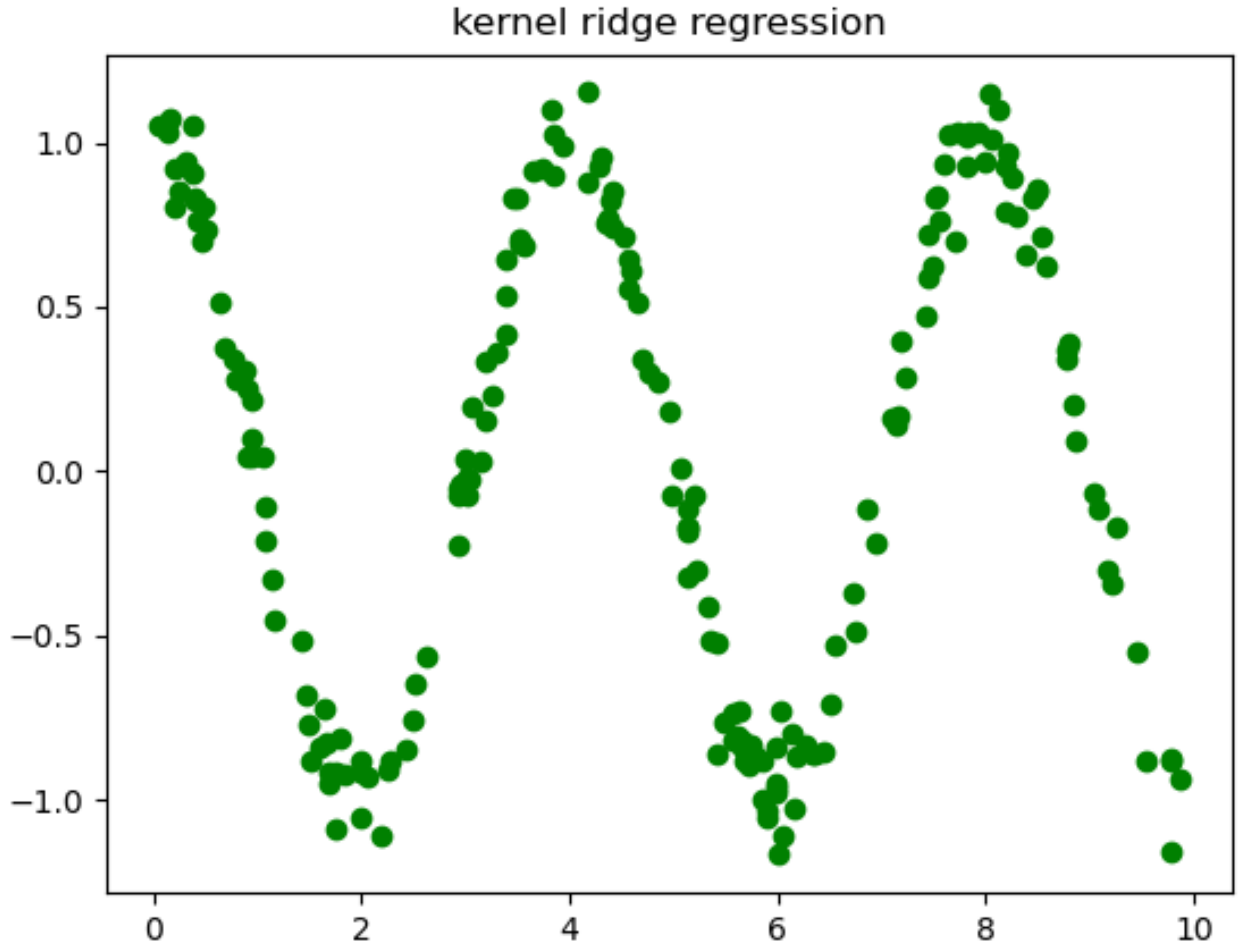
$$\hat{y} = \sum_{i=1}^{n} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

Replace it w/ $k(\mathbf{x}_i, \mathbf{x})$:

$$\hat{y} = \sum_{i=1}^{n} \alpha_i \cdot k(\mathbf{x}_i, \mathbf{x})$$

Demo

Training data is generated as follows: $x \sim$ uniform$[0,10]$, $y = \sin(x\pi/2) + \epsilon, \epsilon \sim \mathcal{N}(0,0.1)$



Outline

1. Kernel

2. Kernel trick and Kernel regression

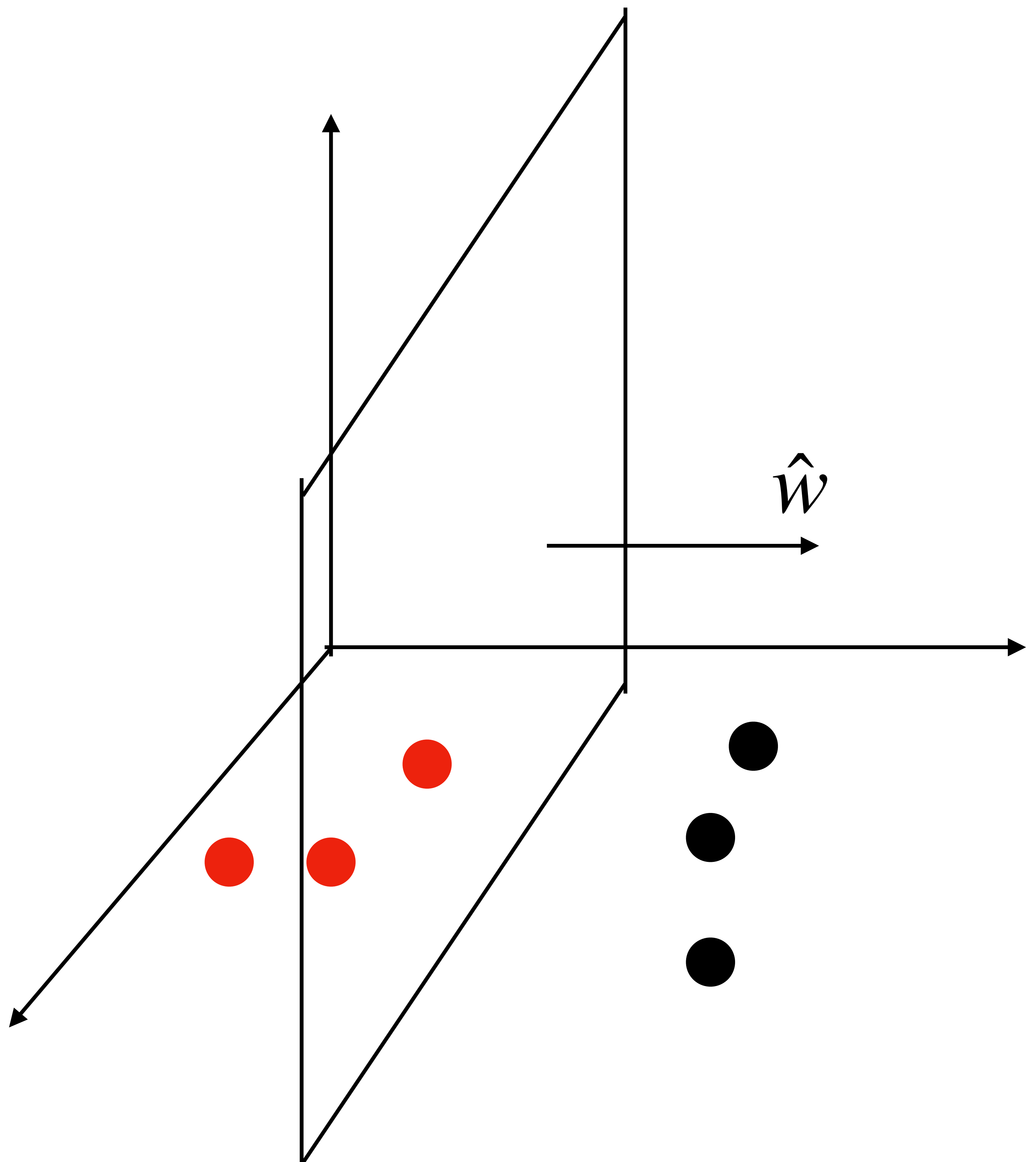
3. Kernel SVM

Recall the soft-margin SVM formulation

$$\min_{w} \|w\|_2^2/2 + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i(w^\top \mathbf{x}_i)\right\}$$

Claim: the optimal solution $\hat{w}$ is also in span(X)

Intuitive proof:



A new formulation of soft-margin SVM formulation

Re-parameterize $w = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = X\alpha$

$$\min_{\alpha} \|X\alpha\|_2^2/2 + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i(\mathbf{x}_i^\top X\alpha)\right\}$$

Alg: gradient descent to optimize $\alpha \in \mathbb{R}^n$

$$\nabla_\alpha \ell(\alpha) = 2X^\top X\alpha + C\sum_{i=1}^{n} \mathbf{1}\left\{y_i(x_i^\top X\alpha) \leq 1\right\}(-y_i X^\top x_i)$$

$$\alpha' = \alpha - \eta \nabla_\alpha \ell(\alpha)$$

Q: Can we apply kernel trick??

Kernelized GD for SVM

$$\min_{\alpha} \|X\alpha\|_2^2 + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i(\mathbf{x}_i^\top X\alpha)\right\}$$

While not converged:

$$g = X^\top X\alpha + C\sum_{i=1}^{n} \mathbf{1}\left\{y_i(x_i^\top X\alpha) \leq 1\right\}(-y_i X^\top x_i)$$

$$\alpha' = \alpha - \eta g$$

Pick a well-defined kernel k;

Replace $X^\top X$ by kernel matrix K

Replace $X^\top \mathbf{x}_i$ by $\mathbf{k}_i = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x_i}) \\ k(\mathbf{x}_2, \mathbf{x_i}) \\ \cdots \\ k(\mathbf{x}_n, \mathbf{x_i}) \end{bmatrix}$

Replace

$$g = K\alpha + C\sum_{i=1}^{n} \mathbf{1}\left\{y_i(\mathbf{k}_i^\top \alpha) \leq 1\right\}(-y_i \mathbf{k}_i)$$

Summary for kernel SVM so far

1. Ideally, want to do the SVM in the lifted high-dim feature space, i.e.,

$$\min_{\alpha} \|w\|_2^2 + C\sum_{i=1}^{n} \max\left\{0, 1 - y_i(w^\top \phi(x_i))\right\}$$

But ϕ can be high-dim (e.g., infinite-dim in Gaussian kernel case)..

2. Via the re-parameterization step, we see GD can be implemented ***via just using*** $\langle \mathbf{x}, \mathbf{z} \rangle$

3. We apply kernel trick, i.e., replace all $\langle \mathbf{x}, \mathbf{z} \rangle$ by $k(\mathbf{x}, \mathbf{z})$

Take-home message today

Kernel trick allows us to do regression / classification in $\phi(\mathbf{x})$ space (possibly infinite dim) **without ever explicitly computing $\phi(\mathbf{x})$!**