Kernel

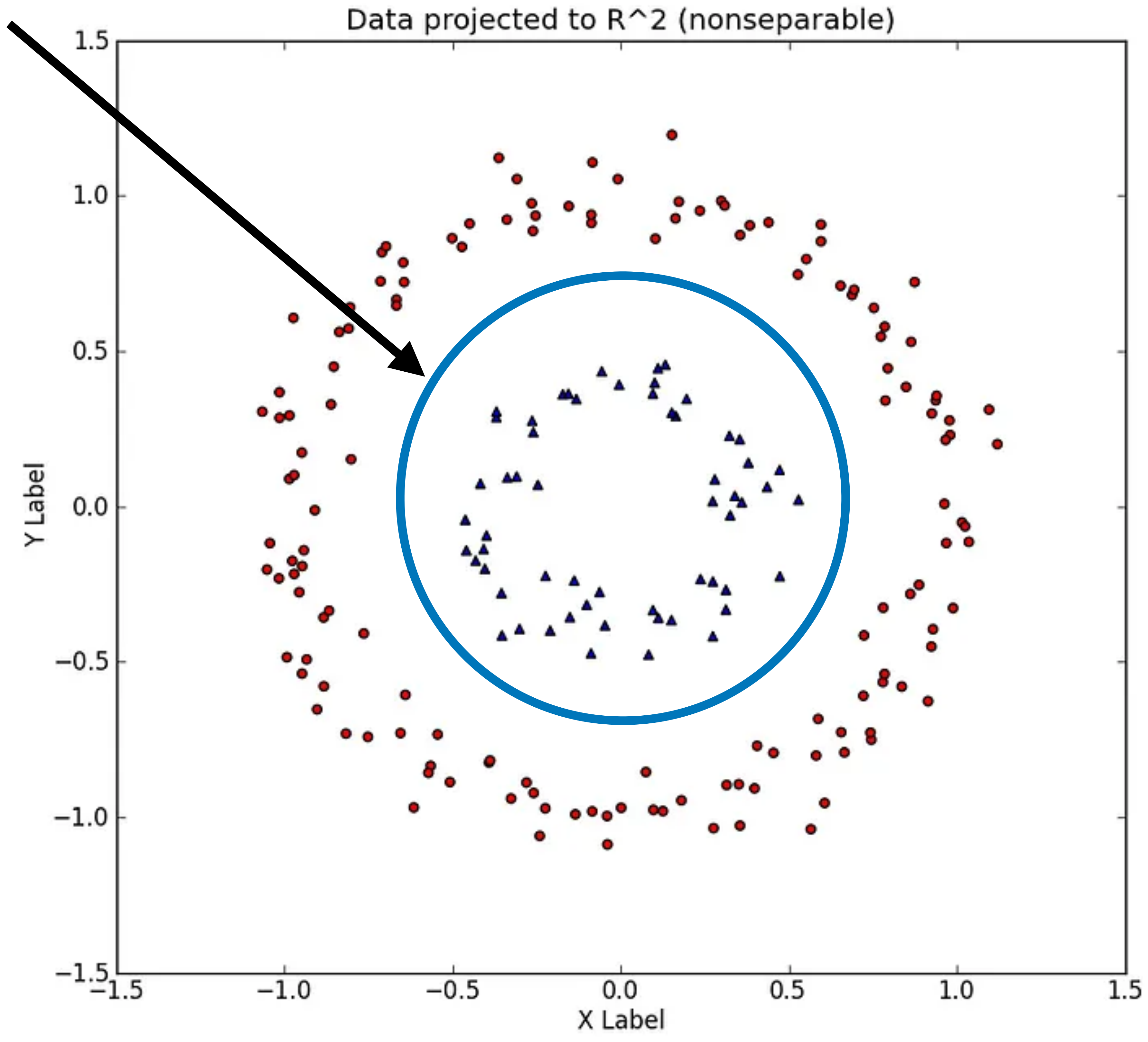
Announcements

HW5 and P5 are released (due in one week)

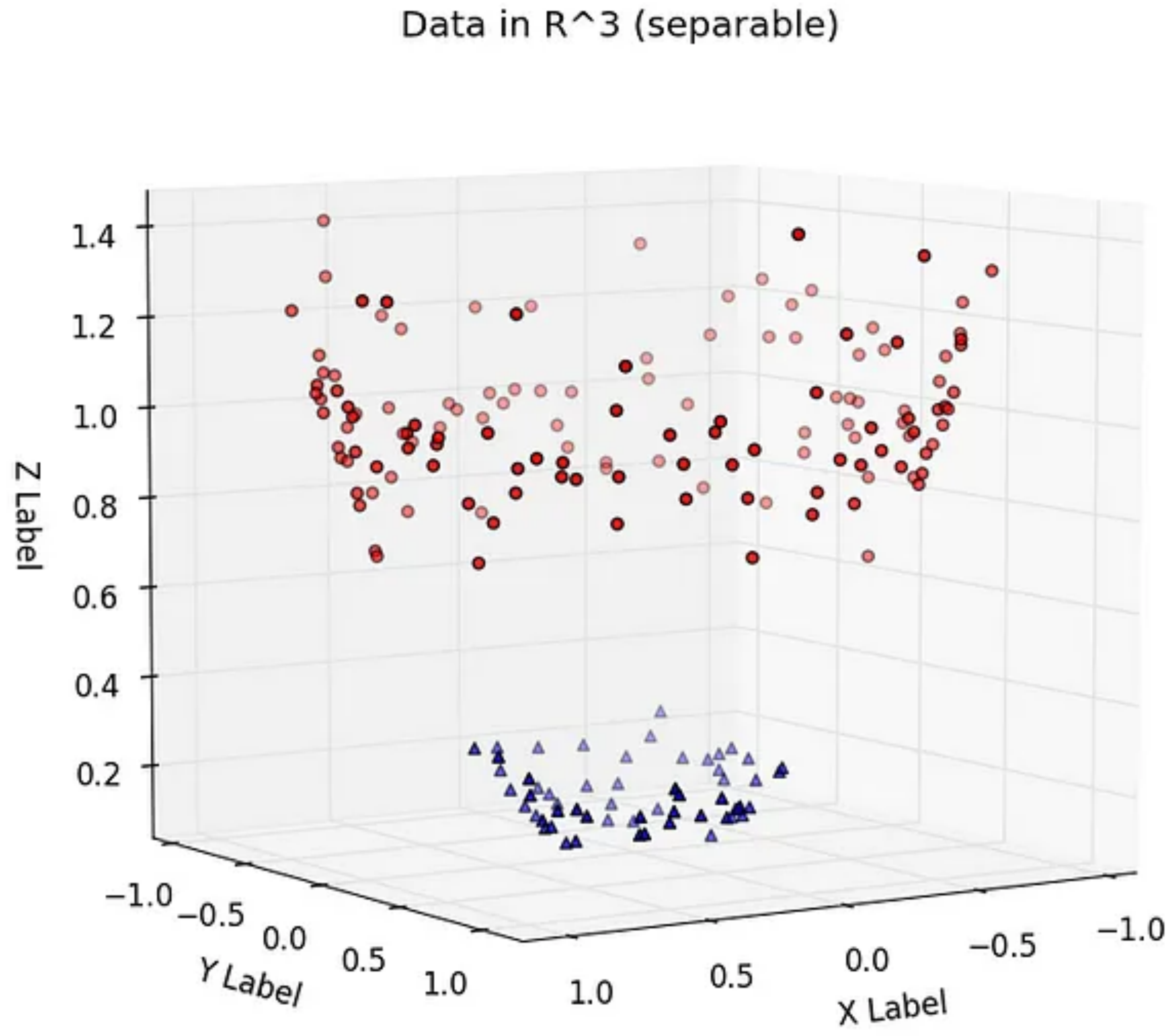
Objective today (and next Tuesday)

Use kernels to design nonlinear ML models (regression & classification)

Goal: Non-linear decision boundary



Our approach



Outline

1. A new perspective on ridge linear regression

2. Feature mapping and Kernel

3. Kernel trick and demo of kernel regression

Linear regression revisited

Dataset $\mathscr{D} = \{\mathbf{x}_i, y_i\}, \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$

Ridge Linear regression solves the following problem:

$$\arg\min_{w} \sum_{i=1}^{n} (w^\top \mathbf{x}_i - y_i)^2 + \lambda \|w\|_2^2$$

Closed-form solution exists, i.e.,

$$\hat{w} = (XX^\top + \lambda I)^{-1} XY$$

Linear regression revisited

Claim: $\hat{w} = (XX^\top + \lambda I)^{-1} XY \in \text{Span}(X)$

An intuitive proof: GD (or SGD)

$$w_0 = \mathbf{0}, w^{t+1} = w^t - \eta \left[\sum_{i=1}^{n} (\mathbf{x}_i^\top w^t - y_i)\mathbf{x}_i + \lambda w^t \right]$$

A new perspective of linear regression

Since we know optimal solution lives in span(X), we can re-parameterize

$$w = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i = X\alpha, \ \ \alpha_i \in \mathbb{R}, \forall i$$

$$\arg\min_{w} \sum_{i=1}^{n} \left\| X^\top w - Y \right\|_2^2 + \lambda \|w\|_2^2 \quad \Rightarrow \quad \arg\min_{\alpha} \left\| X^\top X\alpha - Y \right\|_2^2 + \|X\alpha\|_2^2$$

Original formulation

New formulation w/ α as our variables

A new perspective of linear regression

$$\arg\min_{\alpha} \left\| X^\top X\alpha - Y \right\|_2^2 + \lambda \|X\alpha\|_2^2$$

Solution:

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

$$X^\top X \in \mathbb{R}^{n\times n},\ (X^\top X)_{i,j} = \mathbf{x}_i^\top \mathbf{x}_j = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

A new perspective of linear regression

When we make prediction on a test example $\mathbf{x} \in \mathbb{R}^d$, we have:

$$\hat{w}^\top \mathbf{x} = (\sum_{i=1}^{n} \alpha_i \mathbf{x}_i)^\top \mathbf{x} = \sum_{i=1}^{n} \alpha_i \cdot \langle \mathbf{x}_i, \mathbf{x} \rangle$$

Notice a theme here:

Linear regression can be done by just using inner product of features

$\langle \mathbf{x}, \mathbf{z} \rangle, \mathbf{x} \in \mathbb{R}^d, \mathbf{z} \in \mathbb{R}^d$

Outline

1. A new perspective on ridge linear regression

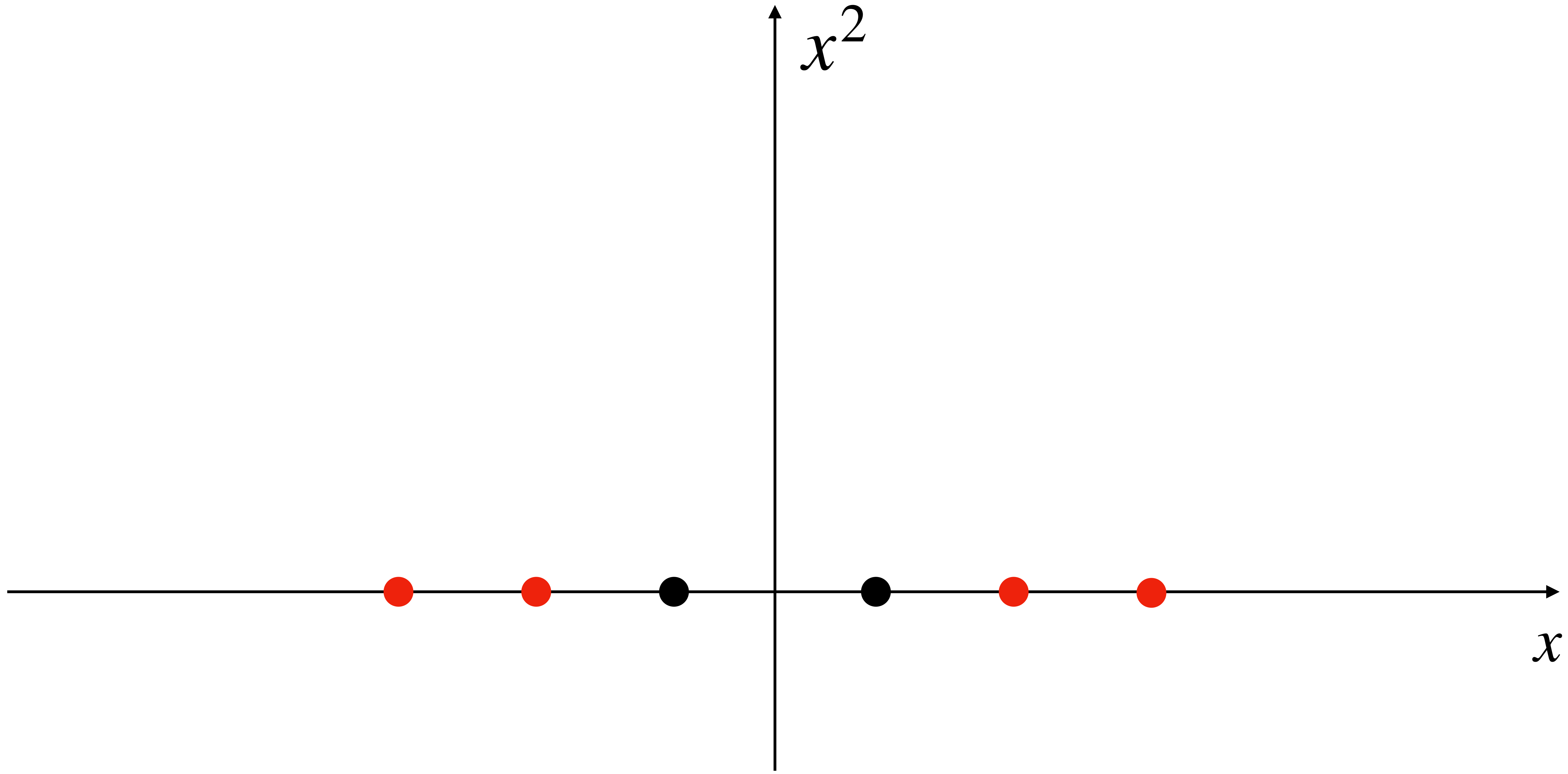
2. Feature mapping and Kernel

3. Kernel trick and demo of kernel regression

Feature mapping

Define $\phi(\mathbf{x}) \in \mathbb{R}^m$ as a feature mapping (often $m > d$)

Ex 1: $\mathbf{x} \in \mathbb{R}$, $\phi(\mathbf{x}) = [x, x^2]^\top \in \mathbb{R}^2$



Feature mapping

Define $\phi(\mathbf{x}) \in \mathbb{R}^m$ as a feature mapping (often $m > d$)

Ex 2: quadratic
feature mapping ϕ

$$\mathbf{x} = [x_1, x_2]^\top,$$

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]^\top$$

Feature mapping

Define $\phi(\mathbf{x}) \in \mathbb{R}^m$ as a feature mapping (often $m > d$)

Ex 2: cubic feature mapping ϕ

$$\mathbf{x} = [x_1, x_2]^\top,$$

$$\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, x_1^3, x_2^3, x_1 x_2^2, x_1^2 x_2]^\top$$

Q: in general, for $\mathbf{x} \in \mathbb{R}^d$, and a p-th order polynomial feature ϕ, what's the dimension of $\phi(\mathbf{x})$?

at least $\binom{d}{p}$

Dim of $\phi(\mathbf{x})$ can be very large!

Fit linear functions in the high-dim feature space

The feature mapping $\phi(\mathbf{x}) \in \mathbb{R}^m$ allows us to perform linear regression in the ϕ space

Ex: cubic feature mapping ϕ

$$\mathbf{x} = [x_1, x_2]^\top, \ \phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1x_2^2, x_1^2x_2]^\top$$

$w^\top \phi(\mathbf{x})$ now can represent a 3-order polynomials!

To fit a 3-order polynomial in $\mathbf{x}$, we can instead do linear regression in $\phi(\mathbf{x})$

Fit linear functions in the high-dim feature space

Perform linear regression in ϕ space, i.e.,

$$\min_{w} \sum_{i=1}^{n} \left(w^\top \phi(\mathbf{x}_i) - y_i\right)^2 + \lambda \|w\|_2^2$$

Linear in ϕ, but high-order poly in $\mathbf{x}$

What is the potential problem of doing this?

This is where the new perspective of linear regression and kernels come to rescue!

Kernel

Kernel $k(\mathbf{x}, \mathbf{z})$

A valid kernel is a kernel such that $\exists \phi, k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z}), \forall \mathbf{x}, \mathbf{z}$

Ex: quadratic kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^2$$

$$\phi(\mathbf{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^\top$$

Q: what's the computation of $k(\mathbf{x}, \mathbf{z})$?

Q: what's the computation of $\phi(\mathbf{x})^\top \phi(\mathbf{z})$?

Ex: cubic feature mapping ϕ

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^3$$

Generalizing to p-th order polynomials:

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

Kernel

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

The mapping $\phi(\mathbf{x})$ is infinite-dimensional

Ex: $\mathbf{x} \in \mathbb{R}$, the mapping $\phi(\mathbf{x})$:

$$\phi(\mathbf{x}) = \left[\ldots, \frac{1}{\sqrt{i!}} \exp\left(-\frac{x^2}{2\sigma^2}\right) x^i, \ldots\right]^\top \in \mathbb{R}^\infty$$

Kernel

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

2. Linear function $w^\top \phi(\mathbf{x})$ can model any indefinitely differentiable function f

Why? ϕ contains all polynomials, and f can be written as an infinite Taylor series..

Summary so far

1. Feature mapping $\phi(\mathbf{x})$ lifts $\mathbf{x}$ into high-dimensional space (e.g., high-order polynomials)

2. A kernel $k(\mathbf{x}, \mathbf{z})$ is a symmetric function, such that there exists a ϕ, so that

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

3. Kernel allows us to compute $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ without ever explicitly computing ϕ

($k(\mathbf{x}, \mathbf{z})$ is easy to compute but $\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ is hard to compute)

Outline

1. A new perspective on ridge linear regression

2. Feature mapping and Kernel

3. Kernel trick and demo of kernel regression

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….

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

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

take-home message

Kernel trick enables to do LR in $\phi(\mathbf{x})$ space (possibly infinite dim) **without ever explicitly computing $\phi(\mathbf{x})$!**