CSCI567 Machine Learning (Spring 2021)

Sirisha Rambhatla

University of Southern California

Feb 19, 2021

Outline

Outline

Logistics

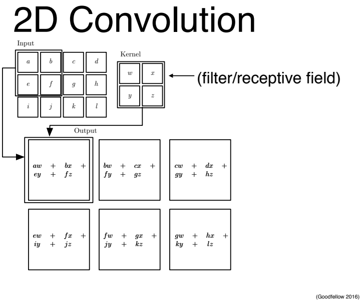
- HW 2 Is due today, and HW 3 will be assigned!

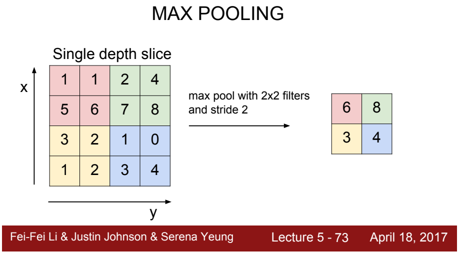
Outline

Convolutional Neural Nets

Typical architecture for CNNs:

Input → [[Conv → ReLU]*N → Pool?]*M → [FC → ReLU]*Q → FC





Outline

Motivation

Recall the question: *how to choose nonlinear basis* $\phi : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$?

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

Motivation

Recall the question: *how to choose nonlinear basis* $\phi : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$?

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

- neural network is one approach: learn ϕ from data

Motivation

Recall the question: *how to choose nonlinear basis* $\phi : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$?

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$$

- neural network is one approach: learn ϕ from data
- **kernel method** is another one: sidestep the issue of choosing ϕ by using *kernel functions*

What are Kernels?

Consider the following example, where the data is not linearly separable in the **ambient** space but is separable **feature** space[1]

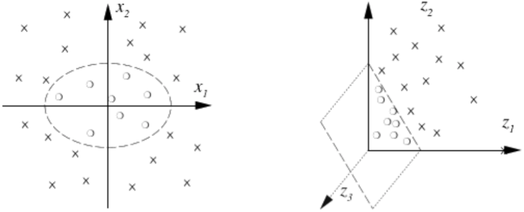


Figure 2.1 Toy example of a binary classification problem mapped into feature space. We assume that the true decision boundary is an ellipse in input space (left panel). The task of the learning process is to estimate this boundary based on empirical data consisting of training points in both classes (crosses and circles, respectively). When mapped into feature space via the nonlinear map $\Phi_2(x) = (z_1, z_2, z_3) = ([x]_1^2, [x]_2^2, \sqrt{2}\,[x]_1[x]_2)$ (right panel), the ellipse becomes a hyperplane (in the present simple case, it is parallel to the z_3 axis, hence all points are plotted in the (z_1, z_2) plane). This is due to the fact that ellipses can be written as linear equations in the entries of (z_1, z_2, z_3). Therefore, in feature space, the problem reduces to that of estimating a hyperplane from the mapped data points. Note that via the polynomial kernel (see (2.12) and (2.13)), the dot product in the three-dimensional space can be computed without computing Φ_2. Later in the book, we shall describe algorithms for constructing hyperplanes which are based on dot products (Chapter 7).

Schölkopf, Bernhard, and Alexander J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

What are Kernels?

We observe that mapping data points to **higher dimension** feature spaces can help with separability, we can then use our favorite linear methods.

What are Kernels?

We observe that mapping data points to **higher dimension** feature spaces can help with separability, we can then use our favorite linear methods.

However, there are following issues:

What are Kernels?

We observe that mapping data points to **higher dimension** feature spaces can help with separability, we can then use our favorite linear methods.

However, there are following issues:

1. Computations in higher dimensions are cumbersome, and the

What are Kernels?

We observe that mapping data points to **higher dimension** feature spaces can help with separability, we can then use our favorite linear methods.

However, there are following issues:

1. Computations in higher dimensions are cumbersome, and the
2. Statistical issue of **curse of dimensionality** kicks-in, which means that as dimension increases we may require exponentially more data samples!

Kernel Trick

Wishlist: It would be great to have an *implicit* way to work in higher dimensions without having to do the computations there.

Kernel Trick

Wishlist: It would be great to have an *implicit* way to work in higher dimensions without having to do the computations there.

Kernels are special functions which allow us to accomplish exactly this!

Kernel Trick

Wishlist: It would be great to have an *implicit* way to work in higher dimensions without having to do the computations there.

Kernels are special functions which allow us to accomplish exactly this!

How: Kernel functions allow us to compute inner-products in the **feature space** while operating on the data samples in the **ambient space**.

Kernel Trick

Wishlist: It would be great to have an *implicit* way to work in higher dimensions without having to do the computations there.

Kernels are special functions which allow us to accomplish exactly this!

How: Kernel functions allow us to compute inner-products in the **feature space** while operating on the data samples in the **ambient space**.

Kernel Trick: To *kernelize* any given algorithm, our aim will be to write the computations as inner-products, and then utilize Kernel functions to do the computations.

Kernel Trick

Wishlist: It would be great to have an *implicit* way to work in higher dimensions without having to do the computations there.

Kernels are special functions which allow us to accomplish exactly this!

How: Kernel functions allow us to compute inner-products in the **feature space** while operating on the data samples in the **ambient space**.

Kernel Trick: To *kernelize* any given algorithm, our aim will be to write the computations as inner-products, and then utilize Kernel functions to do the computations.

Don't need to know $\phi(\cdot)$: Since we use **Kernel function** we actually don't need to know the mapping $\phi(\cdot)$. This means that $\phi(\cdot)$ may be infinite dimensional but we can still evaluate the inner-products in an infinite dimensional feature space!!

Example

Let's take a closer look at the example. Here, we consider the following polynomial basis $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Example

Let's take a closer look at the example. Here, we consider the following polynomial basis $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$?

$$\boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}') = {x_1}^2{x_1'}^2 + 2x_1x_2x_1'x_2' + {x_2}^2{x_2'}^2$$

Example

Let's take a closer look at the example. Here, we consider the following polynomial basis $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$?

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}') &= {x_1}^2 {x'_1}^2 + 2x_1x_2x'_1x'_2 + {x_2}^2 {x'_2}^2 \\ &= (x_1x'_1 + x_2x'_2)^2 \end{aligned}$$

Example

Let's take a closer look at the example. Here, we consider the following polynomial basis $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$?

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}') &= {x_1}^2{x_1'}^2 + 2x_1x_2x_1'x_2' + {x_2}^2{x_2'}^2 \\ &= (x_1x_1' + x_2x_2')^2 = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^2 \end{aligned}$$

Example

Let's take a closer look at the example. Here, we consider the following polynomial basis $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^3$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x}')$?

$$\begin{aligned} \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}') &= {x_1}^2{x_1'}^2 + 2x_1x_2x_1'x_2' + {x_2}^2{x_2'}^2 \\ &= (x_1x_1' + x_2x_2')^2 = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}')^2 \end{aligned}$$

Therefore, *the inner product in the new space is simply a function of the inner product in the original space*.

Another example

$\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{2\mathsf{D}}$ is parameterized by θ:

$$\boldsymbol{\phi}_{\theta}(\boldsymbol{x}) = \begin{pmatrix} \cos(\theta x_1) \\ \sin(\theta x_1) \\ \vdots \\ \cos(\theta x_{\mathsf{D}}) \\ \sin(\theta x_{\mathsf{D}}) \end{pmatrix}$$

Another example

$\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{2\mathsf{D}}$ is parameterized by θ:

$$\boldsymbol{\phi}_\theta(\boldsymbol{x}) = \begin{pmatrix} \cos(\theta x_1) \\ \sin(\theta x_1) \\ \vdots \\ \cos(\theta x_{\mathsf{D}}) \\ \sin(\theta x_{\mathsf{D}}) \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}_\theta(\boldsymbol{x})$ and $\boldsymbol{\phi}_\theta(\boldsymbol{x}')$?

$$\boldsymbol{\phi}_\theta(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}_\theta(\boldsymbol{x}') = \sum_{d=1}^{\mathsf{D}} \cos(\theta x_d) \cos(\theta x'_d) + \sin(\theta x_d) \sin(\theta x'_d)$$

Another example

$\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{2\mathsf{D}}$ is parameterized by θ:

$$\boldsymbol{\phi}_\theta(\boldsymbol{x}) = \begin{pmatrix} \cos(\theta x_1) \\ \sin(\theta x_1) \\ \vdots \\ \cos(\theta x_{\mathsf{D}}) \\ \sin(\theta x_{\mathsf{D}}) \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}_\theta(\boldsymbol{x})$ and $\boldsymbol{\phi}_\theta(\boldsymbol{x}')$?

$$\begin{aligned} \boldsymbol{\phi}_\theta(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}_\theta(\boldsymbol{x}') &= \sum_{d=1}^{\mathsf{D}} \cos(\theta x_d) \cos(\theta x'_d) + \sin(\theta x_d) \sin(\theta x'_d) \\ &= \sum_{d=1}^{\mathsf{D}} \cos(\theta(x_d - x'_d)) \end{aligned}$$

Another example

$\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{2\mathsf{D}}$ is parameterized by θ:

$$\boldsymbol{\phi}_\theta(\boldsymbol{x}) = \begin{pmatrix} \cos(\theta x_1) \\ \sin(\theta x_1) \\ \vdots \\ \cos(\theta x_{\mathsf{D}}) \\ \sin(\theta x_{\mathsf{D}}) \end{pmatrix}$$

What is the inner product between $\boldsymbol{\phi}_\theta(\boldsymbol{x})$ and $\boldsymbol{\phi}_\theta(\boldsymbol{x}')$?

$$\begin{aligned}\boldsymbol{\phi}_\theta(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}_\theta(\boldsymbol{x}') &= \sum_{d=1}^{\mathsf{D}} \cos(\theta x_d)\cos(\theta x'_d) + \sin(\theta x_d)\sin(\theta x'_d) \\ &= \sum_{d=1}^{\mathsf{D}} \cos(\theta(x_d - x'_d))\end{aligned}$$

Once again, *the inner product in the new space is a simple function of the features in the original space*.

More complicated example

Based on ϕ_θ, define $\phi_L : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{2\mathsf{D}(L+1)}$ for some integer L:

$$\phi_L(\boldsymbol{x}) = \begin{pmatrix} \phi_0(\boldsymbol{x}) \\ \phi_{\frac{2\pi}{L}}(\boldsymbol{x}) \\ \phi_{2\frac{2\pi}{L}}(\boldsymbol{x}) \\ \vdots \\ \phi_{L\frac{2\pi}{L}}(\boldsymbol{x}) \end{pmatrix}$$

More complicated example

Based on ϕ_θ, define $\phi_L : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{2\mathsf{D}(L+1)}$ for some integer L:

$$\phi_L(\boldsymbol{x}) = \begin{pmatrix} \phi_0(\boldsymbol{x}) \\ \phi_{\frac{2\pi}{L}}(\boldsymbol{x}) \\ \phi_{2\frac{2\pi}{L}}(\boldsymbol{x}) \\ \vdots \\ \phi_{L\frac{2\pi}{L}}(\boldsymbol{x}) \end{pmatrix}$$

What is the inner product between $\phi_L(\boldsymbol{x})$ and $\phi_L(\boldsymbol{x}')$?

$$\begin{aligned} \phi_L(\boldsymbol{x})^{\mathrm{T}}\phi_L(\boldsymbol{x}') &= \sum_{\ell=0}^{L} \phi_{\frac{2\pi\ell}{L}}(\boldsymbol{x})^{\mathrm{T}}\phi_{\frac{2\pi\ell}{L}}(\boldsymbol{x}') \\ &= \sum_{\ell=0}^{L}\sum_{d=1}^{\mathsf{D}} \cos\left(\frac{2\pi\ell}{L}(x_d - x'_d)\right) \end{aligned}$$

Infinite dimensional mapping

When $L \to \infty$, even if we cannot compute $\phi(x)$, a vector of *infinite dimension*, we can still compute inner product:

Infinite dimensional mapping

When $L \to \infty$, even if we cannot compute $\boldsymbol{\phi}(x)$, a vector of *infinite dimension*, we can still compute inner product:

$$\boldsymbol{\phi}_\infty(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}_\infty(\boldsymbol{x}') = \int_0^{2\pi} \sum_{d=1}^{\mathsf{D}} \cos(\theta(x_d - x'_d))\, d\theta$$

$$= \sum_{d=1}^{\mathsf{D}} \frac{\sin(2\pi(x_d - x'_d))}{x_d - x'_d}$$

Again, a simple function of the original features.

Infinite dimensional mapping

When $L \to \infty$, even if we cannot compute $\boldsymbol{\phi}(x)$, a vector of *infinite dimension*, we can still compute inner product:

$$\boldsymbol{\phi}_\infty(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}_\infty(\boldsymbol{x}') = \int_0^{2\pi} \sum_{d=1}^{\mathsf{D}} \cos(\theta(x_d - x'_d))\, d\theta$$
$$= \sum_{d=1}^{\mathsf{D}} \frac{\sin(2\pi(x_d - x'_d))}{x_d - x'_d}$$

Again, a simple function of the original features.

Note that using this mapping in linear regression, we are *learning a weight $\boldsymbol{w}^*$ with infinite dimension!*

Kernel functions

Definition: a function $k : \mathbb{R}^{\mathsf{D}} \times \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$ is called a *(positive semidefinite) kernel function* if there exists a function $\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{\mathsf{M}}$ so that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{\mathsf{D}}$,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}')$$

Kernel functions

Definition: a function $k : \mathbb{R}^\mathsf{D} \times \mathbb{R}^\mathsf{D} \rightarrow \mathbb{R}$ is called a *(positive semidefinite) kernel function* if there exists a function $\boldsymbol{\phi} : \mathbb{R}^\mathsf{D} \rightarrow \mathbb{R}^\mathsf{M}$ so that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^\mathsf{D}$,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}')$$

Examples we have seen

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}')^2$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{\mathsf{D}} \frac{\sin(2\pi(x_d - x'_d))}{x_d - x'_d}$$

Using kernel functions

Choosing a nonlinear basis ϕ becomes choosing a kernel function.

Using kernel functions

Choosing a nonlinear basis ϕ becomes choosing a kernel function.

As long as computing the kernel function is more efficient, we should apply the kernel trick.

Using kernel functions

Choosing a nonlinear basis ϕ becomes choosing a kernel function.

As long as computing the kernel function is more efficient, we should apply the kernel trick.

Gram/kernel matrix needs to be *positive semi-definite* and *symmetric*

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

Using kernel functions

Choosing a nonlinear basis ϕ becomes choosing a kernel function.

As long as computing the kernel function is more efficient, we should apply the kernel trick.

Gram/kernel matrix needs to be *positive semi-definite* and *symmetric*

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

In fact, k is a kernel if and only if $\boldsymbol{K}$ is positive semidefinite for *any N and any $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, …, $\boldsymbol{x}_N$* (formalized by the **Mercer theorem**).

Using kernel functions

Choosing a nonlinear basis ϕ becomes choosing a kernel function.

As long as computing the kernel function is more efficient, we should apply the kernel trick.

Gram/kernel matrix needs to be *positive semi-definite* and *symmetric*

$$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

In fact, k is a kernel if and only if $\boldsymbol{K}$ is positive semidefinite for *any N and any $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_N$* (formalized by the **Mercer theorem**).

- useful for proving that a function is not a kernel

Examples that are not kernels

Function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$$

is *not a kernel*, why?

Examples that are not kernels

Function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$$

is *not a kernel*, why?

If it is a kernel, the kernel matrix for two data points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$\boldsymbol{K} = \begin{pmatrix} 0 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 & 0 \end{pmatrix}$$

must be positive semidefinite,

Examples that are not kernels

Function

$$k(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$$

is *not a kernel*, why?

If it is a kernel, the kernel matrix for two data points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$:

$$\boldsymbol{K} = \begin{pmatrix} 0 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \\ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 & 0 \end{pmatrix}$$

must be positive semidefinite, *but is it?*

More examples of kernel functions

Two most commonly used kernel functions in practice:

Polynomial kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + c)^d$$

for $c \geq 0$ and d is a positive integer.

More examples of kernel functions

Two most commonly used kernel functions in practice:

Polynomial kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}' + c)^d$$

for $c \geq 0$ and d is a positive integer.

Gaussian kernel or Radial basis function (RBF) kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2}{2\sigma^2}}$$

for some $\sigma > 0$.

Composing kernels

Creating more kernel functions using the following rules:

Composing kernels

Creating more kernel functions using the following rules:

If $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernels, the followings are kernels too

Composing kernels

Creating more kernel functions using the following rules:

If $k_1(\cdot, \cdot)$ and $k_2(\cdot, \cdot)$ are kernels, the followings are kernels too

- **conical combination**: $\alpha k_1(\cdot, \cdot) + \beta k_2(\cdot, \cdot)$ if $\alpha, \beta \geq 0$

Composing kernels

Creating more kernel functions using the following rules:

If $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernels, the followings are kernels too

- **conical combination**: $\alpha k_1(\cdot,\cdot) + \beta k_2(\cdot,\cdot)$ if $\alpha, \beta \geq 0$
- **product**: $k_1(\cdot,\cdot)k_2(\cdot,\cdot)$

Composing kernels

Creating more kernel functions using the following rules:

If $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernels, the followings are kernels too

- **conical combination**: $\alpha k_1(\cdot,\cdot) + \beta k_2(\cdot,\cdot)$ if $\alpha, \beta \geq 0$
- **product**: $k_1(\cdot,\cdot)k_2(\cdot,\cdot)$
- **exponential**: $e^{k(\cdot,\cdot)}$

Composing kernels

Creating more kernel functions using the following rules:

If $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernels, the followings are kernels too

- **conical combination**: $\alpha k_1(\cdot,\cdot) + \beta k_2(\cdot,\cdot)$ if $\alpha, \beta \geq 0$
- **product**: $k_1(\cdot,\cdot)k_2(\cdot,\cdot)$
- **exponential**: $e^{k(\cdot,\cdot)}$
- $\cdots$

Verify using the definition of kernel!

Case study: regularized linear regression

Kernel methods work for *many problems* and we take **regularized linear regression** as an example.

Case study: regularized linear regression

Kernel methods work for *many problems* and we take **regularized linear regression** as an example.

Recall the regularized least square solution, where $\phi : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$:

$$\begin{aligned} \boldsymbol{w}^* &= \operatorname*{argmin}_{\boldsymbol{w}} F(\boldsymbol{w}) \\ &= \operatorname*{argmin}_{\boldsymbol{w}} \left(\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2\right) \\ &= \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \end{aligned}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_{\mathsf{N}})^{\mathrm{T}} \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Case study: regularized linear regression

Kernel methods work for *many problems* and we take **regularized linear regression** as an example.

Recall the regularized least square solution, where $\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$:

$$\begin{aligned} \boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmin}}\, F(\boldsymbol{w}) \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \left(\|\boldsymbol{\Phi w} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2 \right) \\ &= \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y} \end{aligned}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_{\mathsf{N}})^{\mathrm{T}} \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Issue: *operate in space $\mathbb{R}^{\mathsf{M}}$ and M could be huge or even infinity!*

Case study: regularized linear regression

Kernel methods work for *many problems* and we take **regularized linear regression** as an example.

Recall the regularized least square solution, where $\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \rightarrow \mathbb{R}^{\mathsf{M}}$:

$$\begin{aligned} \boldsymbol{w}^* &= \operatorname*{argmin}_{\boldsymbol{w}} F(\boldsymbol{w}) \\ &= \operatorname*{argmin}_{\boldsymbol{w}} \left(\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{w}\|_2^2 \right) \\ &= \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y} \end{aligned}$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_{\mathsf{N}})^{\mathrm{T}} \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{\mathsf{N}} \end{pmatrix}$$

Issue: *operate in space $\mathbb{R}^{\mathsf{M}}$ and M could be huge or even infinity!*
Our aim: pose these computation as inner-products between $\boldsymbol{\phi}(\cdot)$.

A closer look at the least square solution, where

By setting the gradient of $F(\boldsymbol{w}) = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ to be $\mathbf{0}$:

$$\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}^* - \boldsymbol{y}) + \lambda\boldsymbol{w}^* = \mathbf{0}$$

A closer look at the least square solution, where

By setting the gradient of $F(\boldsymbol{w}) = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ to be $\mathbf{0}$:

$$\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}^* - \boldsymbol{y}) + \lambda\boldsymbol{w}^* = \mathbf{0}$$

we know

$$\boldsymbol{w}^* = \frac{1}{\lambda}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}^*) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \sum_{n=1}^{N}\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)$$

A closer look at the least square solution, where

By setting the gradient of $F(\boldsymbol{w}) = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ to be $\mathbf{0}$:

$$\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}^* - \boldsymbol{y}) + \lambda\boldsymbol{w}^* = \mathbf{0}$$

we know

$$\boldsymbol{w}^* = \frac{1}{\lambda}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}^*) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \sum_{n=1}^{N}\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)$$

Thus the least square solution is **a linear combination of features**!

A closer look at the least square solution, where

By setting the gradient of $F(\boldsymbol{w}) = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ to be $\mathbf{0}$:

$$\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}^* - \boldsymbol{y}) + \lambda\boldsymbol{w}^* = \mathbf{0}$$

we know

$$\boldsymbol{w}^* = \frac{1}{\lambda}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}^*) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \sum_{n=1}^{N}\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)$$

Thus the least square solution is **a linear combination of features**!

Note this is true for perceptron and many other problems.

A closer look at the least square solution, where

By setting the gradient of $F(\boldsymbol{w}) = \|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$ to be $\boldsymbol{0}$:

$$\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}^* - \boldsymbol{y}) + \lambda\boldsymbol{w}^* = \boldsymbol{0}$$

we know

$$\boldsymbol{w}^* = \frac{1}{\lambda}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}^*) = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

Thus the least square solution is **a linear combination of features**!

Note this is true for perceptron and many other problems.

Of course, the above calculation does not show what $\boldsymbol{\alpha}$ is.

Why is this helpful?

Assuming we know $\boldsymbol{\alpha}$, the prediction of $\boldsymbol{w}^*$ on a new example $\boldsymbol{x}$ is

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Why is this helpful?

Assuming we know $\boldsymbol{\alpha}$, the prediction of $\boldsymbol{w}^*$ on a new example $\boldsymbol{x}$ is

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Therefore we do not really need to know $\boldsymbol{w}^*$. *Only inner products in the new feature space matter!*

Why is this helpful?

Assuming we know $\boldsymbol{\alpha}$, the prediction of $\boldsymbol{w}^*$ on a new example $\boldsymbol{x}$ is

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Therefore we do not really need to know $\boldsymbol{w}^*$. *Only inner products in the new feature space matter!*

As we've seen, we can now use Kernels to compute inner products *without knowing* ϕ.

Why is this helpful?

Assuming we know $\boldsymbol{\alpha}$, the prediction of $\boldsymbol{w}^*$ on a new example $\boldsymbol{x}$ is

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Therefore we do not really need to know $\boldsymbol{w}^*$. *Only inner products in the new feature space matter!*

As we've seen, we can now use Kernels to compute inner products *without knowing ϕ*.

Also, *this is a non-parametric method!*

Why is this helpful?

Assuming we know $\boldsymbol{\alpha}$, the prediction of $\boldsymbol{w}^*$ on a new example $\boldsymbol{x}$ is

$$\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x})$$

Therefore we do not really need to know $\boldsymbol{w}^*$. *Only inner products in the new feature space matter!*

As we've seen, we can now use Kernels to compute inner products *without knowing* ϕ.

Also, *this is a non-parametric method!*

But we need to figure out what $\boldsymbol{\alpha}$ is first!

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$G(\boldsymbol{\alpha}) = F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha})$$

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$\begin{aligned} G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\ &= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \end{aligned}$$

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$\begin{aligned} G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\ &= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \\ &= \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} \qquad (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}) \end{aligned}$$

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$
\begin{aligned}
G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \\
&= \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} \qquad\qquad (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}) \\
&= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}
\end{aligned}
$$

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$\begin{aligned}
G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \\
&= \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}) \\
&= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} \\
&= \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda\boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} && (\boldsymbol{K}^{\mathrm{T}} = \boldsymbol{K})
\end{aligned}$$

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$\begin{aligned}
G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\
&= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \\
&= \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}) \\
&= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} \\
&= \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda\boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} && (\boldsymbol{K}^{\mathrm{T}} = \boldsymbol{K})
\end{aligned}$$

This is sometime called the *dual formulation* of linear regression.

How to find $\boldsymbol{\alpha}$?

Plugging in $\boldsymbol{w} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}$ into $F(\boldsymbol{w})$ gives

$$\begin{aligned} G(\boldsymbol{\alpha}) &= F(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}) \\ &= \|\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha}\|_2^2 \\ &= \|\boldsymbol{K}\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} && (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}) \\ &= \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} \\ &= \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda\boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.} && (\boldsymbol{K}^{\mathrm{T}} = \boldsymbol{K}) \end{aligned}$$

This is sometime called the *dual formulation* of linear regression.

$\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \in \mathbb{R}^{\mathsf{N}\times\mathsf{N}}$ is called **Gram matrix** or **kernel matrix** where the (i, j) entry is

$$\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)$$

Examples of kernel matrix $\boldsymbol{K}$

3 data points in $\mathbb{R}$

$$x_1 = -1, x_2 = 0, x_3 = 1$$

$\boldsymbol{\phi}$ is polynomial basis with degree 4:

$$\boldsymbol{\phi}(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix}$$

Examples of kernel matrix $\boldsymbol{K}$

3 data points in $\mathbb{R}$

$$x_1 = -1, x_2 = 0, x_3 = 1$$

$\boldsymbol{\phi}$ is polynomial basis with degree 4:

$$\boldsymbol{\phi}(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix}$$

$$\boldsymbol{\phi}(x_1) = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \quad \boldsymbol{\phi}(x_2) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{\phi}(x_3) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Calculation of the Gram matrix $\boldsymbol{K}$

$$\boldsymbol{\phi}(x_1) = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \quad \boldsymbol{\phi}(x_2) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \boldsymbol{\phi}(x_3) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Gram/Kernel matrix

$$\begin{aligned} \boldsymbol{K} &= \begin{pmatrix} \boldsymbol{\phi}(x_1)^{\mathrm{T}}\boldsymbol{\phi}(x_1) & \boldsymbol{\phi}(x_1)^{\mathrm{T}}\boldsymbol{\phi}(x_2) & \boldsymbol{\phi}(x_1)^{\mathrm{T}}\boldsymbol{\phi}(x_3) \\ \boldsymbol{\phi}(x_2)^{\mathrm{T}}\boldsymbol{\phi}(x_1) & \boldsymbol{\phi}(x_2)^{\mathrm{T}}\boldsymbol{\phi}(x_2) & \boldsymbol{\phi}(x_2)^{\mathrm{T}}\boldsymbol{\phi}(x_3) \\ \boldsymbol{\phi}(x_3)^{\mathrm{T}}\boldsymbol{\phi}(x_1) & \boldsymbol{\phi}(x_3)^{\mathrm{T}}\boldsymbol{\phi}(x_2) & \boldsymbol{\phi}(x_3)^{\mathrm{T}}\boldsymbol{\phi}(x_3) \end{pmatrix} \\ &= \begin{pmatrix} 4 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 4 \end{pmatrix} \end{aligned}$$

Gram matrix vs covariance matrix

	dimensions	**entry** (i,j)	**property**
$\mathbf{\Phi\Phi}^{\mathrm{T}}$			
$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$			

Gram matrix vs covariance matrix

	dimensions	**entry** (i, j)	**property**
$\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$	$\mathsf{N} \times \mathsf{N}$		
$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$			

Gram matrix vs covariance matrix

	dimensions	**entry** (i, j)	**property**
$\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$	$\mathrm{N} \times \mathrm{N}$		
$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$	$\mathrm{M} \times \mathrm{M}$		

Gram matrix vs covariance matrix

	dimensions	**entry** (i, j)	**property**
$\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$	N × N	$\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)$	
$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$	M × M		

Gram matrix vs covariance matrix

	dimensions	entry (i, j)	property
$\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$	N × N	$\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)$	
$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$	M × M	$\sum_{n=1}^{N} \phi(\boldsymbol{x}_n)_i \phi(\boldsymbol{x}_n)_j$	

Gram matrix vs covariance matrix

	dimensions	entry (i, j)	property
$\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$	N $\times$ N	$\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)$	both are **symmetric** and **positive semidefinite**
$\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$	M $\times$ M	$\sum_{n=1}^{N} \phi(\boldsymbol{x}_n)_i \phi(\boldsymbol{x}_n)_j$	

How to find α?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}$$

How to find $\boldsymbol{\alpha}$?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt}.$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y}$$

How to find α?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda\boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda\boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}\left((\boldsymbol{K} + \lambda\boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{y}\right)$$

How to find $\boldsymbol{\alpha}$?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}\left((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{y}\right)$$

Thus $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ **is a minimizer**

How to find $\boldsymbol{\alpha}$?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}\left((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{y}\right)$$

Thus $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ **is a minimizer** and we obtain

$$\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

How to find $\boldsymbol{\alpha}$?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt}.$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}\left((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{y}\right)$$

Thus $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ **is a minimizer** and we obtain

$$\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

Exercise: *are there other minimizers?*

How to find $\boldsymbol{\alpha}$?

Minimize the dual formulation

$$G(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^{\mathrm{T}}(\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{\alpha} + \text{cnt.}$$

Setting the derivative to $\mathbf{0}$ we have

$$\mathbf{0} = (\boldsymbol{K}^2 + \lambda \boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{y} = \boldsymbol{K}\left((\boldsymbol{K} + \lambda \boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{y}\right)$$

Thus $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ **is a minimizer** and we obtain

$$\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\alpha} = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$$

Exercise: *are there other minimizers? and are there other $\boldsymbol{w}^*$'s?*

Comparing two solutions

Minimizing $F(w)$ **gives** $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Comparing two solutions

Minimizing $F(w)$ **gives** $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ **gives** $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Comparing two solutions

Minimizing $F(w)$ gives $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ gives $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Comparing two solutions

Minimizing $F(w)$ **gives** $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ **gives** $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

Comparing two solutions

Minimizing $F(w)$ **gives** $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ **gives** $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

They have to be the same because $F(\boldsymbol{w})$ *has a unique minimizer!*

Comparing two solutions

Minimizing $F(\boldsymbol{w})$ gives $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ gives $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

They have to be the same because $F(\boldsymbol{w})$ has a unique minimizer!

And they are:

$$
\begin{aligned}
&(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}
\end{aligned}
$$

Comparing two solutions

Minimizing $F(\boldsymbol{w})$ gives $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ gives $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

They have to be the same because $F(\boldsymbol{w})$ has a unique minimizer!

And they are:

$$
\begin{aligned}
&(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{\Phi}^{\mathrm{T}})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}
\end{aligned}
$$

Comparing two solutions

Minimizing $F(\boldsymbol{w})$ **gives** $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ **gives** $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

They have to be the same because $F(\boldsymbol{w})$ *has a unique minimizer!*

And they are:

$$
\begin{aligned}
&(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{\Phi}^{\mathrm{T}})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}
\end{aligned}
$$

Comparing two solutions

Minimizing $F(\boldsymbol{w})$ gives $\boldsymbol{w}^* = (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$

Minimizing $G(\boldsymbol{\alpha})$ gives $\boldsymbol{w}^* = \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$

Note $\boldsymbol{I}$ has different dimensions in these two formulas.

Natural question: *are they the same or different?*

They have to be the same because $F(\boldsymbol{w})$ has a unique minimizer!

And they are:

$$
\begin{aligned}
&(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{\Phi}^{\mathrm{T}})(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \\
&= (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})\boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \\
&= \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}
\end{aligned}
$$

Then what is the difference?

First, computing $(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}$ can be more efficient than computing $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \lambda \boldsymbol{I})^{-1}$ when $\mathsf{N} \leq \mathsf{M}$.

Then what is the difference?

First, computing $(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}$ can be more efficient than computing $(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}$ when $\mathsf{N} \leq \mathsf{M}$.

More importantly, computing $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ also *only requires computing inner products in the new feature space!*

Then what is the difference?

First, computing $(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}$ can be more efficient than computing $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \lambda \boldsymbol{I})^{-1}$ when N $\leq$ M.

More importantly, computing $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ also *only requires computing inner products in the new feature space!*

Now we can conclude that the exact form of $\phi(\cdot)$ is not essential; *all we need is computing inner products* $\boldsymbol{\phi}(\boldsymbol{x})^{T}\boldsymbol{\phi}(\boldsymbol{x}')$.

Then what is the difference?

First, computing $(\mathbf{\Phi\Phi}^{\mathrm{T}} + \lambda \boldsymbol{I})^{-1}$ can be more efficient than computing $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \lambda \boldsymbol{I})^{-1}$ when $\mathsf{N} \leq \mathsf{M}$.

More importantly, computing $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$ also *only requires computing inner products in the new feature space!*

Now we can conclude that the exact form of $\boldsymbol{\phi}(\cdot)$ is not essential; *all we need is computing inner products* $\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x}')$.

For some $\boldsymbol{\phi}$ it is indeed possible to compute $\boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}')$ without computing/knowing $\boldsymbol{\phi}$. This is the *kernel trick*.

Kernelizing other ML algorithms

Kernel trick is applicable to **many ML algorithms**:

- nearest neighbor classifier
- perceptron
- logistic regression
- SVM
- $\cdots$

Example: Kernelized NNC

For NNC with **L2 distance**, the key is to compute for any two points $\boldsymbol{x}$, $\boldsymbol{x}'$

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}'^{\mathrm{T}}\boldsymbol{x}' - 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'$$

Example: Kernelized NNC

For NNC with **L2 distance**, the key is to compute for any two points $\boldsymbol{x}$, $\boldsymbol{x}'$

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}'^{\mathrm{T}}\boldsymbol{x}' - 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'$$

With a kernel function k, we simply compute

$$d^{\mathrm{KERNEL}}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}')$$

Example: Kernelized NNC

For NNC with **L2 distance**, the key is to compute for any two points $\boldsymbol{x}$, $\boldsymbol{x}'$

$$d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}'^{\mathrm{T}}\boldsymbol{x}' - 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}'$$

With a kernel function k, we simply compute

$$d^{\mathrm{KERNEL}}(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}')$$

which by definition is the **L2 distance in a new feature space**

$$d^{\mathrm{KERNEL}}(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x}')\|_2^2$$