The Kernel Trick

Yingzhen Li

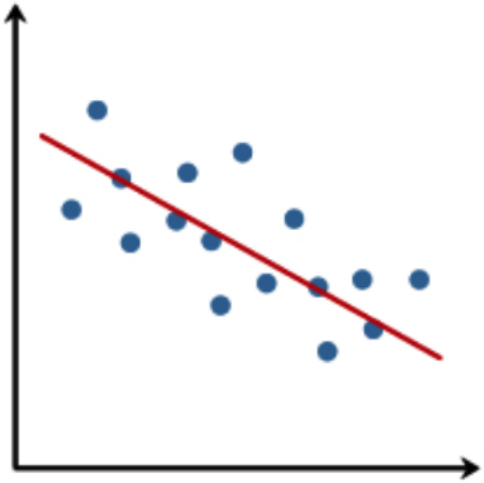
Department of Computing
Imperial College London

@liyzhen2
yingzhen.li@imperial.ac.uk

November 26, 2021

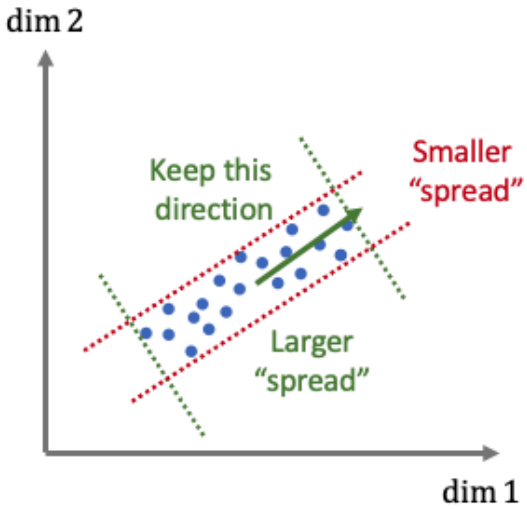
Linear models in ML

With linear features



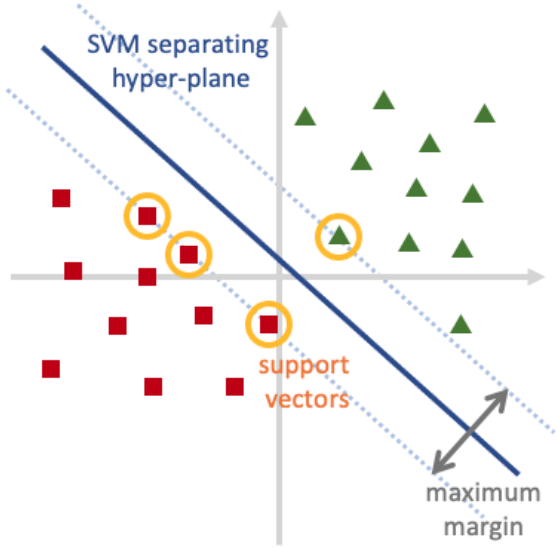
Linear regression

$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x}$



PCA

$\max \mathbb{V}[\mathbf{b}^\top \boldsymbol{x}]$

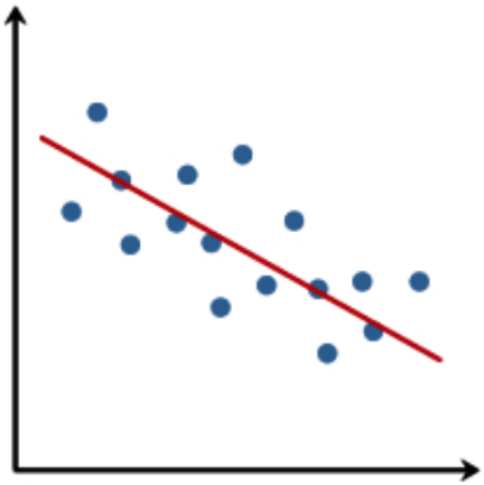


SVM

$f(\boldsymbol{x}) = \text{sgn}(\mathbf{w}^\top \boldsymbol{x} + b)$

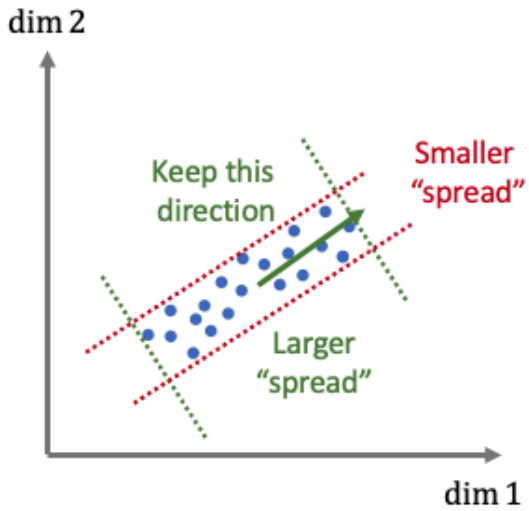
Linear models in ML

With **non-linear** features



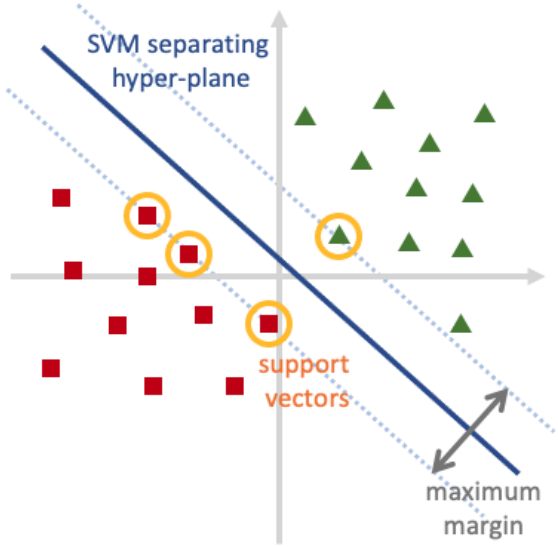
Linear regression

$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x})$



PCA

$\max \mathbb{V}[\mathbf{b}^\top \boldsymbol{\phi}(\boldsymbol{x})]$



SVM

$f(\boldsymbol{x}) = \text{sgn}(\mathbf{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b)$

How to choose feature mapping $\boldsymbol{\phi}(\boldsymbol{x})$?

Choice of feature map

The choice of feature map $\phi(\cdot)$: crucial for performance:

- Want a flexible feature map
  - example: sine waves with different frequency and phase
  - often achieved by using large $p > \dim(\boldsymbol{x})$

Choice of feature map

The choice of feature map $\phi(\cdot)$: crucial for performance:

- Want a flexible feature map
  - example: sine waves with different frequency and phase
  - often achieved by using large $p > \dim(\boldsymbol{x})$
- Can we make $p \to +\infty$?
- What if the output domain of $\phi(\cdot)$ is not even $\mathbb{R}^p$?

Choice of feature map

The choice of feature map $\phi(\cdot)$: crucial for performance:

- Want a flexible feature map
  - example: sine waves with different frequency and phase
  - often achieved by using large $p > \dim(\boldsymbol{x})$
- Can we make $p \to +\infty$?
- What if the output domain of $\phi(\cdot)$ is not even $\mathbb{R}^p$?

Yes! Using the **kernel trick**:
compute $\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$ directly without explicitly evaluating $\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)$

Solution of non-linear ridge regression

Fitting non-linear ridge regression models:

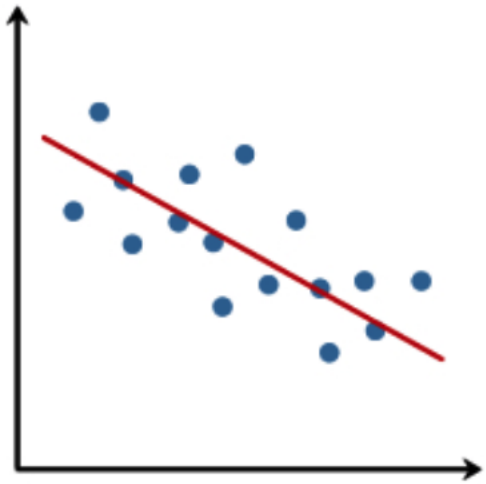
- Dataset: $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$,
  $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$,
  $\mathbf{y} = [y_1, ..., y_N]^\top \in \mathbb{R}^{N \times 1}$
- The non-linear regression model:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta}, \quad \boldsymbol{\phi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^p$$

$$y = f(\boldsymbol{x}, \boldsymbol{\theta}) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

- Solution: $\Phi = [\boldsymbol{\phi}(\boldsymbol{x}_1), ..., \boldsymbol{\phi}(\boldsymbol{x}_N)]^\top \in \mathbb{R}^{N \times p}$

$$\boldsymbol{\theta}^* = (\sigma^2 \lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$$



$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta}$$

The kernel trick

Solution of non-linear ridge regression:

$$\boldsymbol{\theta}^* = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

Constructing predictions for new inputs $\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_M$ with features $\hat{\boldsymbol{\Phi}} = [\boldsymbol{\phi}(\hat{\boldsymbol{x}}_1), ..., \boldsymbol{\phi}(\hat{\boldsymbol{x}}_M)]^\top$:

$$\hat{\mathbf{y}} = \hat{\boldsymbol{\Phi}}\boldsymbol{\theta}^* = \hat{\boldsymbol{\Phi}}(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

The kernel trick

Solution of non-linear ridge regression:

$$\boldsymbol{\theta}^* = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

Constructing predictions for new inputs $\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_M$ with features $\hat{\boldsymbol{\Phi}} = [\boldsymbol{\phi}(\hat{\boldsymbol{x}}_1), ..., \boldsymbol{\phi}(\hat{\boldsymbol{x}}_M)]^\top$:

$$\hat{\mathbf{y}} = \hat{\boldsymbol{\Phi}}\boldsymbol{\theta}^* = \hat{\boldsymbol{\Phi}}(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})\boldsymbol{\Phi}^\top = \sigma^2\lambda\boldsymbol{\Phi}^\top + \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)$$

The kernel trick

Solution of non-linear ridge regression:

$$\boldsymbol{\theta}^* = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

Constructing predictions for new inputs $\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_M$ with features $\hat{\boldsymbol{\Phi}} = [\boldsymbol{\phi}(\hat{\boldsymbol{x}}_1), ..., \boldsymbol{\phi}(\hat{\boldsymbol{x}}_M)]^\top$:

$$\hat{\mathbf{y}} = \hat{\boldsymbol{\Phi}}\boldsymbol{\theta}^* = \hat{\boldsymbol{\Phi}}(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})\boldsymbol{\Phi}^\top = \sigma^2\lambda\boldsymbol{\Phi}^\top + \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)$$

$$\Rightarrow \boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1} = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top$$

The kernel trick

Solution of non-linear ridge regression:

$$\boldsymbol{\theta}^* = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

Constructing predictions for new inputs $\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_M$ with features $\hat{\boldsymbol{\Phi}} = [\boldsymbol{\phi}(\hat{\boldsymbol{x}}_1), ..., \boldsymbol{\phi}(\hat{\boldsymbol{x}}_M)]^\top$:

$$\hat{\mathbf{y}} = \hat{\boldsymbol{\Phi}}\boldsymbol{\theta}^* = \hat{\boldsymbol{\Phi}}(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}$$

- Notice the following identities:

$$(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})\boldsymbol{\Phi}^\top = \sigma^2\lambda\boldsymbol{\Phi}^\top + \boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)$$

$$\Rightarrow \boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1} = (\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top$$

$$\Rightarrow \hat{\boldsymbol{\Phi}}\boldsymbol{\Phi}^\top(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}\boldsymbol{\Phi}^\top)^{-1}\mathbf{y} = \hat{\boldsymbol{\Phi}}(\sigma^2\lambda\mathbf{I} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top\mathbf{y} = \hat{\boldsymbol{\Phi}}\boldsymbol{\theta}^*$$

The kernel trick

Constructing predictions for new inputs $\hat{\mathbf{X}}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$$

$$\Leftrightarrow \text{ Kernel regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1}\mathbf{y}$$

The kernel trick

Constructing predictions for new inputs $\hat{\mathbf{X}}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$$

$$\Leftrightarrow \text{ Kernel regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\Phi^\top(\sigma^2\lambda\mathbf{I} + \Phi\Phi^\top)^{-1}\mathbf{y}$$

Advantages of the kernel regression view:

- Efficient in $N << p$ scheme:
  - $\Phi^\top\Phi \in \mathbb{R}^{p\times p} \Rightarrow$ matrix inversion needs $\mathcal{O}(p^3)$ time
  - $\Phi\Phi^\top \in \mathbb{R}^{N\times N} \Rightarrow$ matrix inversion needs $\mathcal{O}(N^3)$ time

The kernel trick

Constructing predictions for new inputs $\hat{\mathbf{X}}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$$

$$\Leftrightarrow \text{ Kernel regression:} \quad \hat{\mathbf{y}} = \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}}(\sigma^2\lambda\mathbf{I} + \mathbf{K}_{\mathbf{X}\mathbf{X}})^{-1}\mathbf{y}$$

Advantages of the kernel regression view:

- No need to explicitly define $\phi(\cdot)$:
  - The entries of $\mathbf{K}_{\mathbf{X}\mathbf{X}} := \Phi\Phi^\top$: $\mathbf{K}_{\mathbf{X}\mathbf{X}}[i,j] := \phi(\boldsymbol{x}_i)^\top\phi(\boldsymbol{x}_j) := \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$
  - The entries of $\mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}} := \hat{\Phi}\Phi^\top$: $\mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}}[k,l] := \phi(\hat{\boldsymbol{x}}_k)^\top\phi(\boldsymbol{x}_l) := \mathcal{K}(\hat{\boldsymbol{x}}_k, \boldsymbol{x}_l)$
  - Enables the usage of $p = \infty$-dim features (or even non-$\mathbb{R}^p$ features)

The kernel trick

Constructing predictions for new inputs $\hat{\mathbf{X}}$ with features $\hat{\Phi}$:

$$\text{Parametric regression:} \quad \hat{\mathbf{y}} = \hat{\Phi}\boldsymbol{\theta}^* = \hat{\Phi}(\sigma^2\lambda\mathbf{I} + \Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$$

$$\Leftrightarrow \text{ Kernel regression:} \quad \hat{\mathbf{y}} = \mathbf{K}_{\hat{\mathbf{X}}\mathbf{X}}(\sigma^2\lambda\mathbf{I} + \mathbf{K}_{\mathbf{X}\mathbf{X}})^{-1}\mathbf{y}$$

Advantages of the parametric regression view:

- Easy to design $\phi(\cdot)$ with prior knowledge
- Better interpretability for non-linear features $\phi(\cdot)$
- (Mark will argue both points for the kernel view as well :)

Solution of non-linear PCA

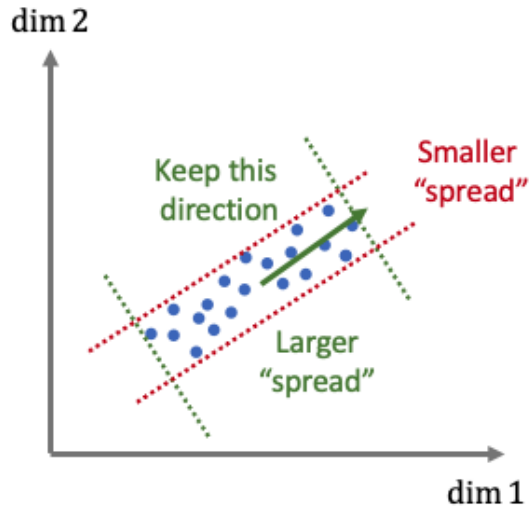
Fitting non-linear PCA:

- Dataset: $\mathcal{D} = \mathbf{X}$,
  $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$
- The non-linear PCA model:

$$\boldsymbol{z}_n := \mathbf{B}^\top \boldsymbol{\phi}(\boldsymbol{x}_n), \quad \boldsymbol{\phi}(\cdot) : \mathbb{R}^D \to \mathbb{R}^p$$

$$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_M] \in \mathbb{R}^{p \times M}$$

is an orthonormal basis



- Solution: the leading M eigenvector of

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\boldsymbol{x}_n) \boldsymbol{\phi}(\boldsymbol{x}_n)^\top$$

$$\max \mathbb{V}[\mathbf{b}^\top \boldsymbol{\phi}(\boldsymbol{x})]$$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

The objective of PCA for $\mathbf{b}_m$:

$$\mathbf{b}_m^\top \mathbf{S}\mathbf{b}_m = \mathbf{b}_m^\top \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} ||\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m||_2^2$$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

Plugging-in the above reparam. of $\mathbf{b}_m$:

$$\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m = \sum_{n'=1}^{N} \alpha_{mn'}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_{n'}) = \sum_{n'=1}^{N} \alpha_{mn'}\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$$

$\Rightarrow$ we can write $\mathbf{b}_m^\top \mathbf{S}\mathbf{b}_m$ as a function of $\{\alpha_{mn}\}$ and $\{\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

Constraints for $\mathbf{b}_m$ to satisfy:

$$||\mathbf{b}_m||_2^2 = 1 \quad \Rightarrow \quad \sum_{n,n'} \alpha_{mn}\alpha_{mn'} \underbrace{\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_{n'})}_{=\mathcal{K}(\boldsymbol{x}_n,\boldsymbol{x}_{n'})} = 1$$

$$\mathbf{b}_i^\top \mathbf{b}_j = 0, i \neq j \quad \Rightarrow \quad \sum_{n,n'} \alpha_{in}\alpha_{jn'} \underbrace{\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_{n'})}_{=\mathcal{K}(\boldsymbol{x}_n,\boldsymbol{x}_{n'})} = 0$$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

Rewrite the objective & constraints using $\{\alpha_{mn}\}$ and $\{\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}$
$\Rightarrow$ solve the constrained opt. for $\{\alpha_{mn}\}$ which implicitly define $\mathbf{b}_m^*$

The kernel trick

Solving the following eigenvector computation problem:

$$\lambda_m \mathbf{b}_m = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$$

Define $\alpha_{mn} = \frac{1}{N\lambda_m}\phi(\boldsymbol{x}_n)^\top \mathbf{b}_m$:

$$\mathbf{b}_m = \sum_{n=1}^{N} \alpha_{mn}\phi(\boldsymbol{x}_n)$$

Rewrite the objective & constraints using $\{\alpha_{mn}\}$ and $\{\mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}$
$\Rightarrow$ solve the constrained opt. for $\{\alpha_{mn}\}$ which implicitly define $\mathbf{b}_m^*$
Prediction:

$$\hat{z}_m := \mathbf{b}_m^\top \phi(\hat{\boldsymbol{x}}) = \sum_{n=1}^{N} \alpha_{mn} \underbrace{\phi(\boldsymbol{x}_n)^\top \phi(\hat{\boldsymbol{x}})}_{=\mathcal{K}(\boldsymbol{x}_n, \hat{\boldsymbol{x}})}$$

Constructing kernels

What is a kernel function?

- In previous non-linear regression example:
  $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \lambda^{-1} \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j)$
- Can also directly define $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ without explicitly define $\boldsymbol{\phi}(\boldsymbol{x})$:
  - Non-negative: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}) \geqslant 0$ for any $\boldsymbol{x}$
  - Symmetric: $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{K}(\boldsymbol{x}', \boldsymbol{x})$
  - Positive semi-definite (PSD):
    for any $N > 0$, $\mathbf{X} \in \mathbb{R}^{N \times D}$, $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ is a PSD matrix.

Constructing kernels

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$$
$$\sigma_f = 1, \ell = 0.25$$

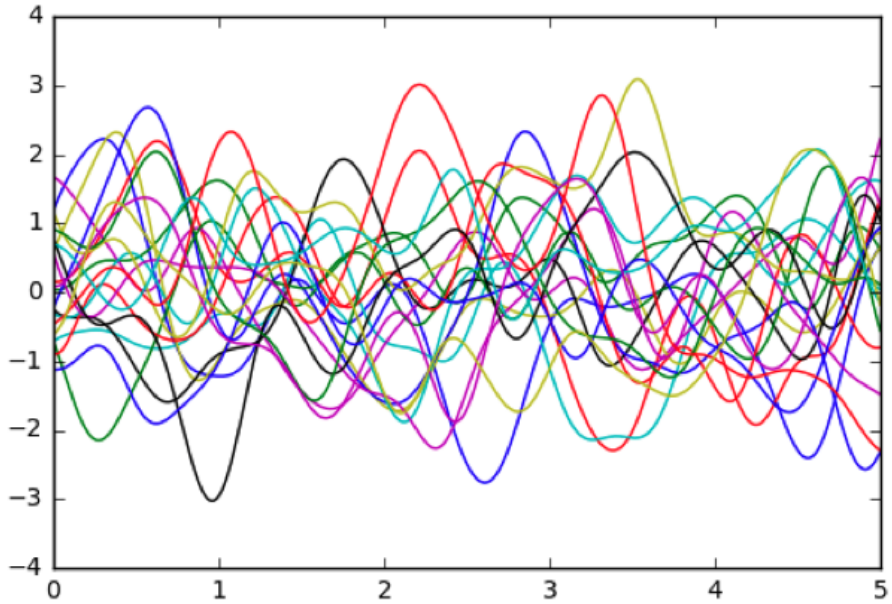


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$$
$$\sigma_f = 1, \ell = 4$$

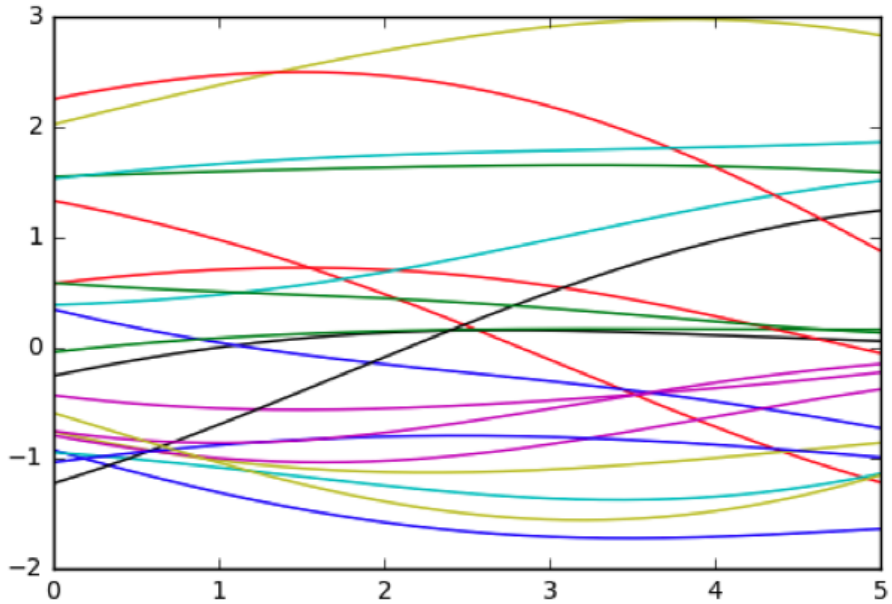


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$$
$$\sigma_f = 1, \ell = 1$$

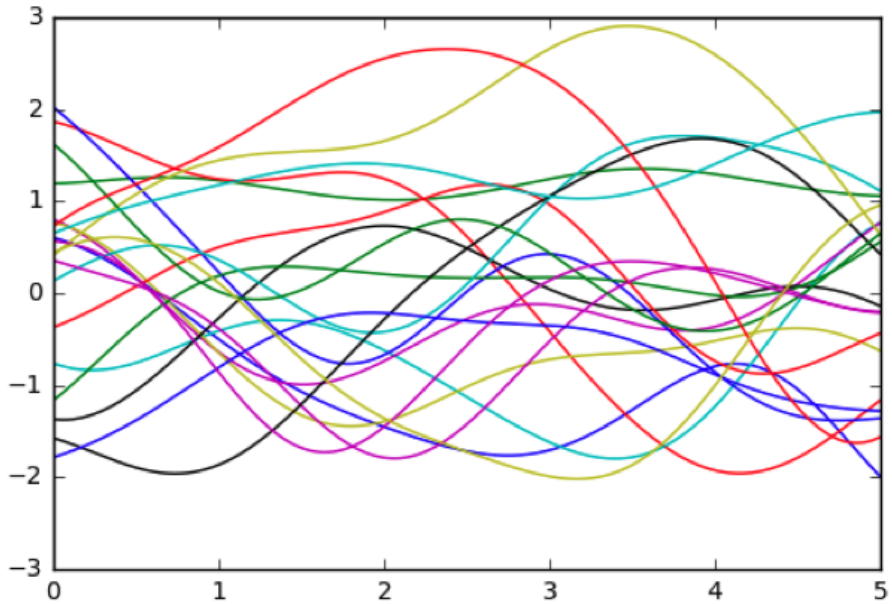


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2)$$
$$\sigma_f = 10, \ell = 1$$

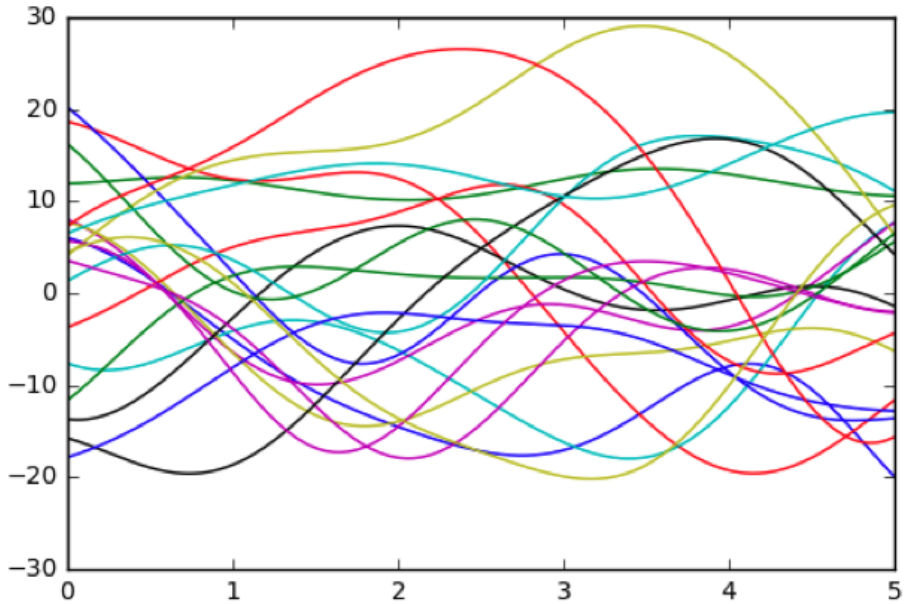
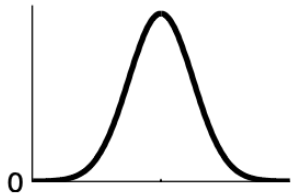
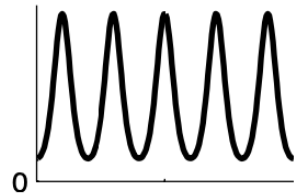
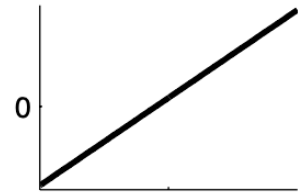
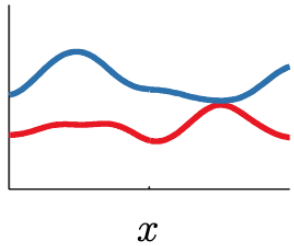
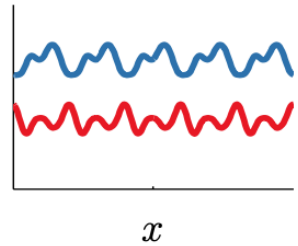
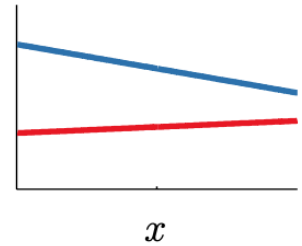


Figure from Richard Wilkinson's GPSS 2019 lecture

Constructing kernels

Kernel name:	Squared-exp (SE)	Periodic (Per)	Linear (Lin)
$k(x, x') =$	$\sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\sigma_f^2 \exp\left(-\frac{2}{\ell^2}\sin^2\left(\pi\frac{x-x'}{p}\right)\right)$	$\sigma_f^2(x-c)(x'-c)$
Plot of $k(x, x')$:			
Functions $f(x)$ sampled from GP prior:			
Type of structure:	local variation	repeating structure	linear functions

See David Duvenaud's PhD thesis, Chapter 2

Summary

The kernel trick

- $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$ for some feature mapping $\boldsymbol{\phi}(\cdot)$
  - As defining the "inner product" in some feature space (might not be Euclidean and might be infinite-dimensional)
- Useful when model prediction can be computed using inner products between features
- Examples:
  - Non-linear ridge regression $\Rightarrow$ Kernel ridge regression
  - PCA with non-linear features $\Rightarrow$ Kernel PCA
  - SVM (read MML book chapter 12)
  - Gaussian processes ("Probabilistic Inference" course in spring)