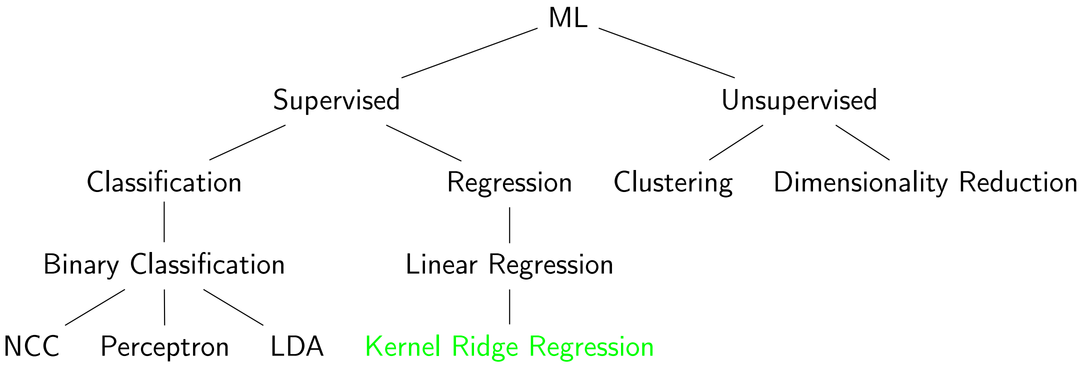
Cognitive Algorithms: Tutorial 4

Kernel Methods

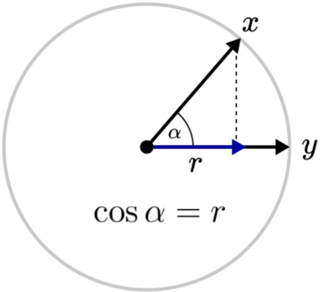
Joanina, Ken, Augustin

Outline

The Tree of CA



Dot Product as a Similarity Measure



https://www.tivadardanka.com/blog/how-the-dot-product-measures-similarity

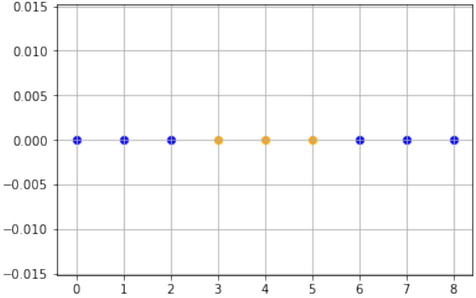
Kernel Trick

$\Rightarrow$ We can replace the linear kernels/simple dot products by other kernel functions, to make our models more powerful!

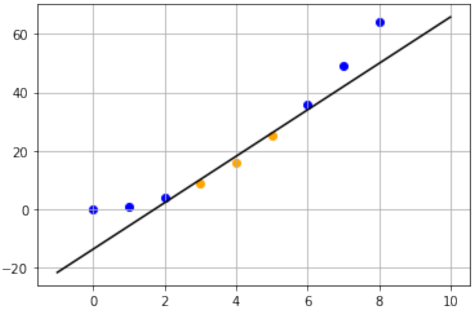
Kernel Trick

$\Rightarrow$ We can replace the linear kernels/simple dot products by other kernel functions, to make our models more $\underbrace{\text{powerful!}}_{\Rightarrow\text{Mappings}}$

Mappings



Not linearly separable 1-dimensional data



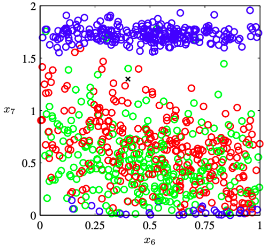
Data in feature space $\phi(x) = (x, x^2)^\mathsf{T}$

Curse of Dimensionality

- consider a dataset $X \subseteq \mathbb{R}^{12}$ (i.e. a dataset of high dimensionality)

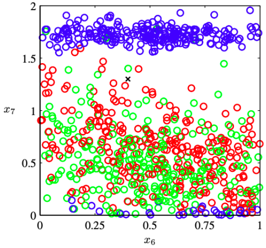
Curse of Dimensionality

- consider a dataset $X \subseteq \mathbb{R}^{12}$ (i.e. a dataset of high dimensionality)



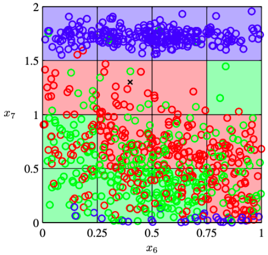
Curse of Dimensionality

- consider a dataset $X \subseteq \mathbb{R}^{12}$ (i.e. a dataset of high dimensionality)

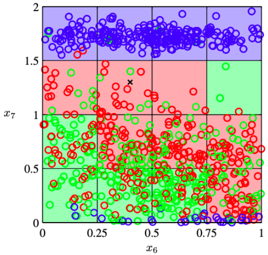


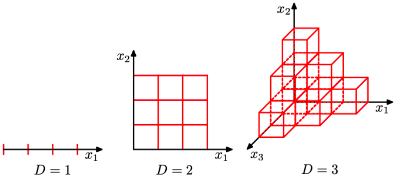
- Consider the following (bad) classification algorithm:
  1. Divide the space into a regular grid
  2. Assign new point x to the class that appears most often in the cell of x

Curse of Dimensionality



Curse of Dimensionality





Curse of Dimensionality

- Take the example of a quadratic expansion

Curse of Dimensionality

- Take the example of a quadratic expansion
- 1-dimensional:

$$\phi(x) = \{1, x, x^2\}$$

Curse of Dimensionality

- Take the example of a quadratic expansion
- 1-dimensional:

$$\phi(x) = \{1, x, x^2\}$$

- 2-dimensional:

$$\phi(x, y) = \{1, x, y, x^2, xy, y^2\}$$

Curse of Dimensionality

- Take the example of a quadratic expansion
- 1-dimensional:

$$\phi(x) = \{1, x, x^2\}$$

- 2-dimensional:

$$\phi(x, y) = \{1, x, y, x^2, xy, y^2\}$$

- 3-dimensional:

$$\phi(x, y, z) = \{1, x, y, z, xy, xz, yz, x^2, y^2, z^2\}$$

Curse of Dimensionality

- Take the example of a quadratic expansion
- 1-dimensional:

$$\phi(x) = \{1, x, x^2\}$$

- 2-dimensional:

$$\phi(x, y) = \{1, x, y, x^2, xy, y^2\}$$

- 3-dimensional:

$$\phi(x, y, z) = \{1, x, y, z, xy, xz, yz, x^2, y^2, z^2\}$$

$\rightarrow$ Infeasible to compute and store

Constructing Valid Kernels

1. Option

2. Option

Constructing Valid Kernels

1. Option

- Find it by starting with a feature space mapping:
  $k(x, x') = \phi(x)^T \phi(x')$

2. Option

Constructing Valid Kernels

1. Option

- Find it by starting with a feature space mapping: $k(x, x') = \phi(x)^T \phi(x')$

2. Option

- Show the kernel function corresponds to a scalar product in some feature space

Constructing Valid Kernels

1. Option

- Find it by starting with a feature space mapping: $k(x, x') = \phi(x)^T \phi(x')$

2. Option

- Show the kernel function corresponds to a scalar product in some feature space
- Option 1: show $K := [k(x_i, x_j)]_{ij}$ is symmetric PSD for all possible choices of $X \subseteq \mathcal{X}$

Constructing Valid Kernels

1. Option

- Find it by starting with a feature space mapping: $k(x, x') = \phi(x)^T \phi(x')$

2. Option

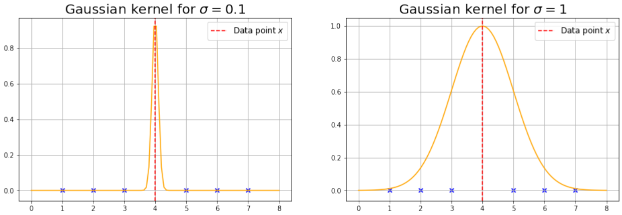
- Show the kernel function corresponds to a scalar product in some feature space
- Option 1: show $K := [k(x_i, x_j)]_{ij}$ is symmetric PSD for all possible choices of $X \subseteq \mathcal{X}$
- Option 2: construct directly from other valid kernels!

Gaussian Kernel

- $k(x, x') := exp\{-\frac{||x-x'||^2}{2\sigma^2}\}$

Gaussian Kernel

- $k(x, x') := exp\{-\frac{||x-x'||^2}{2\sigma^2}\}$



Polynomial Kernel

$$< x, y >^m = \phi(x) \cdot \phi(y)$$

- to see what the feature map $\phi(x)$ looks like in general, please refer to this article
- in the case of $m = 2$ and $d = 2$ it becomes:

$$\begin{aligned}< x, y >^2 &= (x_1 y_1 + x_2 y_2)^2 \\ &= x_1^2 y_1^2 + 2 x_1 x_2 y_1 y_2 + x_2^2 y_2^2 \\ &= [x_1^2, \sqrt{2} x_1 x_2, x_2^2]^\top \cdot [y_1^2, \sqrt{2} y_1 y_2, y_2^2]^\top \\ &= \phi(x) \cdot \phi(y)\end{aligned}$$

Task 2

1. For the RBF Kernel, prove that the subspace of ϕ would map to an infinite dimensional space.

Task 2

2. (a) You are a researcher at a natural language processing firm and they have given you the task of classifying greetings into either English or German. You are given a six-greeting data set, three in each language.

Task 2

2. (a) You are a researcher at a natural language processing firm and they have given you the task of classifying greetings into either English or German. You are given a six-greeting data set, three in each language.
You construct a function $\phi : \mathbb{R} \to \{0, 1\}^{G+E}$, that takes a binary encoded greeting and maps it a one-hot vector with an index for every word in both German and English (G is the number of words in German, and E the number of words in English)

Task 2

2. (a) You are a researcher at a natural language processing firm and they have given you the task of classifying greetings into either English or German. You are given a six-greeting data set, three in each language.
You construct a function $\phi : \mathbb{R} \to \{0, 1\}^{G+E}$, that takes a binary encoded greeting and maps it a one-hot vector with an index for every word in both German and English (G is the number of words in German, and E the number of words in English)
Without actually solving the OLS equation, give a weight vector $\mathbf{w}_\phi$ which results in proper classification of each training example. Describe how this mapping into a higher dimension was the key to solving this problem.

Task 2

2. (b) It occurs to you that ϕ is terribly expensive to compute and takes up inordinate space. Thus, you have the epiphany to use the kernel trick. Derive a kernel function k and an expression for a classifier $f_k(x)$. Give some α that fits the data optimally - and explain how you arrived at it.

Kernelizing Ridge Regression

- Instead we compute:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

Kernelizing Ridge Regression

- Instead we compute:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= k(\mathbf{x}, \mathbf{X})\,\underbrace{(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}}_{\alpha}$$

Kernelizing Ridge Regression

- Instead we compute:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= k(\mathbf{x},\mathbf{X})\underbrace{(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}}_{\alpha}$$

Instead of $\mathbf{w}$ we compute α. What is α?

Kernelizing Ridge Regression

- Instead we compute:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= k(\mathbf{x},\mathbf{X})\,\underbrace{(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}}_{\alpha}$$

Instead of $\mathbf{w}$ we compute α. What is α?

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

Kernel Ridge Regression - Derivation

$$\mathbf{w} = (\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T} + \lambda\mathbf{I})^{-1}\phi(\mathbf{X})\mathbf{y}^\mathsf{T}$$

$$= (\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T} + \lambda\mathbf{I})^{-1}\phi(\mathbf{X})\underbrace{(\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X}) + \lambda\mathbf{I})(\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}}_{\mathbf{I}}\mathbf{y}^\mathsf{T}$$

$$= (\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T} + \lambda\mathbf{I})^{-1}(\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X}) + \lambda\phi(\mathbf{X}))(\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= (\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T} + \lambda\mathbf{I})^{-1}(\phi(\mathbf{X})\phi(\mathbf{X})^\mathsf{T} + \lambda\mathbf{I})\phi(\mathbf{X})(\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X}) + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= \phi(\mathbf{X})(\underbrace{\phi(\mathbf{X})^\mathsf{T}\phi(\mathbf{X})}_{\mathbf{K}} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= \underbrace{\phi(\mathbf{X})}_{\text{Problem}}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

Problem: We can't or don't want to compute $\phi(\mathbf{X}) \longrightarrow$ we can't compute $\mathbf{w}$

Kernel Ridge Regression - Derivation

- To solve the problem from the previous slide we compute:

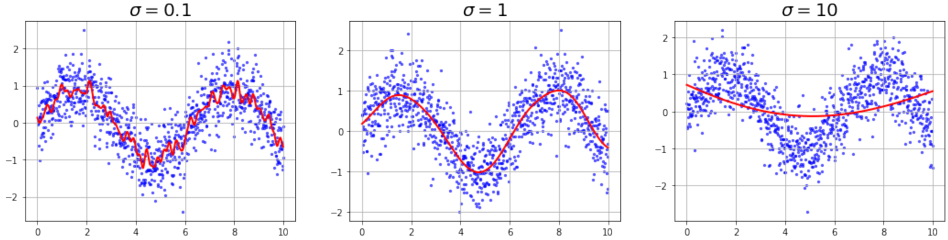
$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\phi(\mathbf{x}) = \phi(\mathbf{x})^\mathsf{T}\mathbf{w} = \underbrace{\phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{X})}_{k(\mathbf{x},\mathbf{X})}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}$$

$$= k(\mathbf{x},\mathbf{X})\,\underbrace{(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y}^\mathsf{T}}_{\alpha}$$

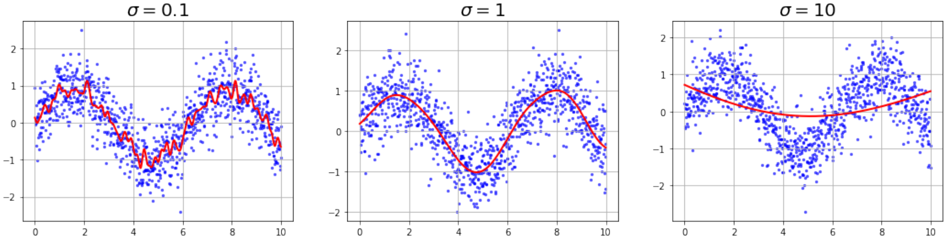
Instead of $\mathbf{w}$ we compute α. What is α?

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

Kernelized RR with a Gaussian Kernel



Kernelized RR with a Gaussian Kernel



Q: How does this relate to the terms "Overfitting" and "Underfitting"?

Kernelized RR - The Algorithm

1. Compute α:

$$\alpha = (\mathbf{K} + \lambda I)^{-1}\mathbf{y}_{train}^T$$

$$\text{with } \mathbf{K} := k(\mathbf{X}_{train}, \mathbf{X}_{train})$$

Kernelized RR - The Algorithm

1. Compute α:

$$\alpha = (\mathbf{K} + \lambda I)^{-1} \mathbf{y}_{train}^T$$
$$\text{with } \mathbf{K} := k(\mathbf{X}_{train}, \mathbf{X}_{train})$$

2. Predictions:

$$\hat{y}_{new} = \alpha^T k(\mathbf{X}_{train}, x_{new})$$

Task 1

1. We want to fit a simple linear model $g_1(x) = w \cdot x$ to the data using (Kernel) Ridge Regression - (K)RR. Take $\lambda = 0.0001$; compute w, the corresponding function $g_1(x)$ and describe why this results in a poor fit.

Task 1

1. We want to fit a simple linear model $g_1(x) = w \cdot x$ to the data using (Kernel) Ridge Regression - (K)RR. Take $\lambda = 0.0001$; compute w, the corresponding function $g_1(x)$ and describe why this results in a poor fit.
2. Now we want to fit a better model, simply by adding a constant term, as well as a squared term: $g_2(x) = w_1 + w_2 \cdot x + w_3 \cdot x^2 = \mathbf{w}^\top \cdot \boldsymbol{\phi}(x)$. Compute $\mathbf{w}$ and the corresponding function $g_2(x)$. Is this approximation better? Why?

Task 1

3. (a) If we wanted to perform something akin to kernel *non*-ridge regression, we would set λ to 0; however, this is not allowed. Why not?

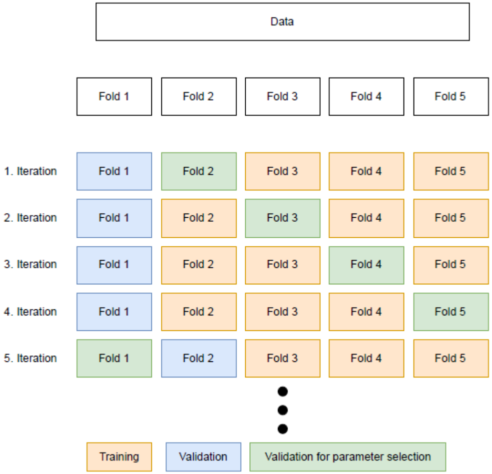
Task 1

3. (a) If we wanted to perform something akin to kernel *non*-ridge regression, we would set λ to 0; however, this is not allowed. Why not?
   (b) Compute k corresponding to ϕ in part 2 and compute the resulting classifier $g_3(x)$, again for $\lambda = 0.0001$. Why is g_3 the same as g_2?

Task 1

3. (a) If we wanted to perform something akin to kernel *non*-ridge regression, we would set λ to 0; however, this is not allowed. Why not?
   (b) Compute k corresponding to ϕ in part 2 and compute the resulting classifier $g_3(x)$, again for $\lambda = 0.0001$. Why is g_3 the same as g_2?
4. Consider the RBF kernel with kernel width $\sigma_1 = \sqrt{2}$ and $\sigma_2 = \frac{\sqrt{2}}{2}$. Compute the corresponding classifier g_{RBF1} and g_{RBF2} based on the same data, (with the same $\lambda = 0.0001$). What is the difference between them? Denote this in your plot in Task 1.5.

Nested Cross-Validation



Summary

$$\begin{tikzcd}[column sep=huge] k(\mathbf{x}, \mathbf{x}') \arrow[r, bend left, "\text{Mercer's theorem}"] & \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}') \arrow[l, bend left, "\text{Kernel trick}"] \end{tikzcd}$$

Kernels:

- Unknown (data in) feature space
- $D \gg n$
- Non-numeric data
- The entire dataset has to be stored

Feature maps:

- Feature map often infeasible to compute
- $n \gg D$