CS:E4830 Kernel Methods in Machine Learning

Lecture 1 : Basics and Introduction to Kernel Methods

Rohit Babbar

3rd March, 2022

Course Format and Logistics

- Lecture online via Zoom
  - Downloadable video lectures will also be made available
  - 10 lectures in total
- 3 assignments and one final exam
  - Exercise sessions to help with the assignments will be physical
  - Location TU1 SAAB auditorium
- Zulip service
  - To interact amongst yourself and TAs
  - Pls keep it professional, stick to Aalto guidelines
- Two ways to pass the course
  - Assignments (50%) + Exam (50%)
  - Exam only
  - More details on mycourses page : `https://mycourses.aalto.fi/course/view.php?id=32426`

Personnel

- Sessions with TAs
  - TAs - Petrus Mikkola, Adrian Mueller, Mohammadreza Qaraei
  - Python/Numpy tutorial on 10th March
- Doubts related to
  - Lecture material - Send me an email rohit.babbar@aalto.fi
  - Assignments - Use Zulip, TAs will get back to you

Nature of the course

- Masters/PhD level course
- **Theoretical** (Mathematical) and **Algorithmic**
- Prerequisites
  - Machine Learning Basic Principles (or an equivalent course such as CS-E4710 - Machine Learning: Supervised Methods)
  - Linear Algebra and Basics of Probability/Statistics
  - Programming in Python

Course content - 1

- Kernel Methods - Introduction
  - Feature spaces and maps
  - Positive Definiteness
  - Reproducing kernels and RKHS
  - Representer Theorem

Course content - 1

- Kernel Methods - Introduction
  - Feature spaces and maps
  - Positive Definiteness
  - Reproducing kernels and RKHS
  - Representer Theorem
- Learning theory overview
  - Generalization and overfitting
  - Empirical Risk Minimization and consistency
  - Uniform Convergence and Rademacher complexity

Course content - 2

- Supervised learning algorithms with kernels
  - Convex Optimization and Duality
  - Support Vector Machines
  - Kernel Logistic Regression

Course content - 2

- Supervised learning algorithms with kernels
  - Convex Optimization and Duality
  - Support Vector Machines
  - Kernel Logistic Regression
- Unsupervised learning algorithms
  - Algorithms such as PCA and k-means clustering
  - Kernel Variants

- Supervised learning algorithms with kernels
  - Convex Optimization and Duality
  - Support Vector Machines
  - Kernel Logistic Regression
- Unsupervised learning algorithms
  - Algorithms such as PCA and k-means clustering
  - Kernel Variants
- Advanced topics
  - Speeding up kernel Methods
  - Bochner's Theorem and Nystrom Approximation

Outline for Today

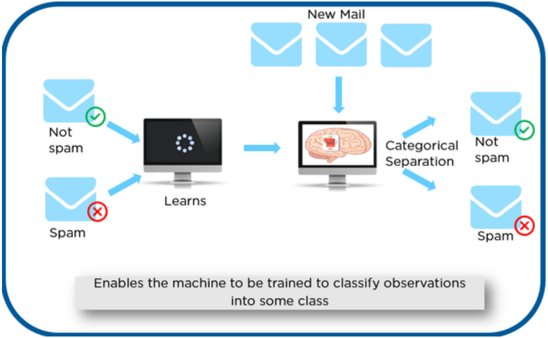
1. Course Format
2. Linear and Non-linear Classification
3. Vector Spaces
4. Kernels
5. Constructing new Kernels

Introduction

Classification - Supervised Learning

- Given a training set $\{(x_i, y_i)\}_{i=1}^{n}$ such that the inputs $x_i \in \mathcal{X}$ (the input space) and $y_i \in \{+1, -1\}$ for binary classification.
- Learn a classifier which predicts the class $\hat{y}$ for the novel (test) instance x
- Example - For designing a spam vs non-spam classifier
    - x_i refers to an email in the training set.
    - $y_i \in \{+1, -1\}$ could be used to indicate the label non-spam and spam respectively.

- Example - Spam vs non-spam classification by your email software



- What makes the above paradigm of **learning from data** so powerful compared to building hand-crafted rules for the same task?

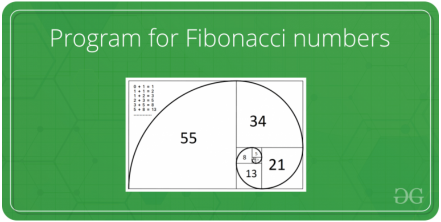
Learning from Data - Pros

- The data-driven learning paradigm can adapt itself to various tasks, i.e. feed different data from a different task to the **same learning algorithm** (ex - SVM, Neural network)
  - Get a different classifier for a different task altogether
  - For example - Feed in dogs vs cats image data to the **same learning algorithm**
- This is not possible in hand-crafted rule based system
  - For instance, rules for spam detection are very different from those of image classification
- Elimination (or reduced dependency on domain expert) for hand-crafting rules
  - Manual involvement in label supervision (recall $y_i \in \{+1, -1\}$)
  - Label supervision is typically cheaper or free (as in Facebook/Instgram tags)

Learning From Data - Cons

Corner cases

- In the **classical programming** paradigm, we tell the system how to handle each (base or corner) case explicitly
- Under the **machine learning** paradigm - corner case might occur infrequently, and hence the system may be unreliable for such cases
- Arguably, a good practical and engineering system is an appropriate combination of both the approaches



Linear vs Non-linear Classification

Linear classification

The prediction function is a linear combination of input features

Linear classification

- Consider the classification function f_1 below, which is linear in both the input features and weights

$$f_1(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)}$$

- In this case, the decision function $f_1(x)$ is trying to capture only **linear combination** of the input components $x^{(1)}, x^{(2)}$
- Linear feature map $\phi_1 : \mathbb{R}^2 \mapsto \mathbb{R}^2$, and is given by, $\phi_1(x) = (x^{(1)}, x^{(2)})^T$
- f_1 is parameterized as $f_1(.) = (w^{(1)}, w^{(2)})^T$

Linear classification - Pictorial depiction

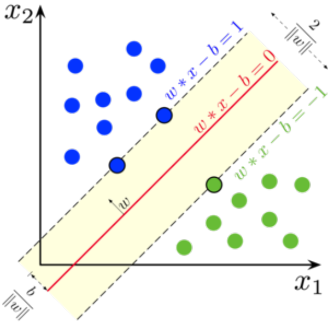


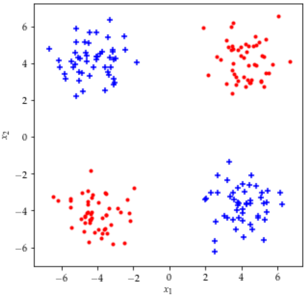
Figure: Caution : x_1 and x_2 in the picture represents the components $x^{(1)}$ and $x^{(2)}$ in the text on the previous slide

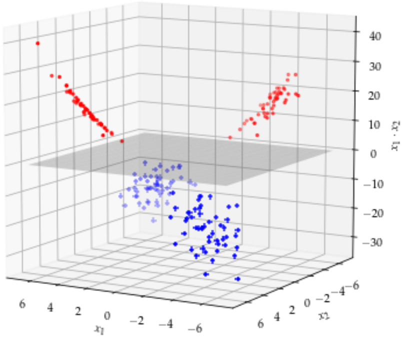
In linear classification :

- classifier is a **straight line** in 2D (as shown above) **in the input space**
- generally called a **plane** or hyperplane in high dimensions

Non-linear Classification Example

- Dataset in 2-D (left), which is not linearly separable
- It can be separated by a plane in 3-D (third feature is the product $x_1 x_2$)





Non-linear classification

Prediction function can involve non-linear combination of features

- For the classification function f_2 below, which is linear in weights and **non-linear in input features**

$$f_2(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(1)}x^{(2)} + w^{(4)}x^{(2)}x^{(1)}$$

- Here, the decision function $f_2(x)$ is trying to capture **non-linear combination** of the input components as well such as $x^{(1)}x^{(2)}, x^{(2)}x^{(1)}$

Non-linear classification

Prediction function can involve non-linear combination of features

- For the classification function f_2 below, which is linear in weights and **non-linear in input features**

$$f_2(x) = w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + w^{(3)}x^{(1)}x^{(2)} + w^{(4)}x^{(2)}x^{(1)}$$

- Here, the decision function $f_2(x)$ is trying to capture **non-linear combination** of the input components as well such as $x^{(1)}x^{(2)}, x^{(2)}x^{(1)}$
- Non-linear feature map $\phi_2 : \mathbb{R}^2 \mapsto \mathbb{R}^4$, and is given by $\phi_2(x) = (x^{(1)}, x^{(2)}, x^{(1)}x^{(2)}, x^{(2)}x^{(1)})^T$
    - $\phi_2(x) \in \mathcal{H}$, which is referred to as the feature space
- Note that the decision function $f_2(x)$ **is still linear in the weight vector co-efficients** $w^{(j)}$'s and is parameterized by $(w^{(1)}, w^{(2)}, w^{(3)}, w^{(4)})^T$

Kernel Methods - Motivation

- Data may not exhibit linear separability in the input space $\mathcal{X}$, and hence we need to apply feature transformation $\phi : \mathcal{X} \mapsto \mathcal{H}$
- Dimensionality of feature space $\mathcal{H} >>$ dimensionality of the input space $\mathcal{X}$
- As a result, we get data separability (statistical advantage) but at the cost of increased calculations (computational disadvantage)

Kernel Methods - Motivation

- Data may not exhibit linear separability in the input space $\mathcal{X}$, and hence we need to apply feature transformation $\phi : \mathcal{X} \mapsto \mathcal{H}$
- Dimensionality of feature space $\mathcal{H} >>$ dimensionality of the input space $\mathcal{X}$
- As a result, we get data separability (statistical advantage) but at the cost of increased calculations (computational disadvantage)
- Next - **Kernels to the rescue**!

Slide 24

Dual representation of the optimal hyperplane

- Consequently, the functional margin $y\mathbf{w}^T\mathbf{x}$ also can be expressed using the support vectors:

$$y\mathbf{w}^T\mathbf{x} = y\sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i^T\mathbf{x}$$

- The norm of the weight vector can be expressed as

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{m}\alpha_j y_j \mathbf{x}_j = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$$

- Note that the training data appears in pairwise inner products: $\mathbf{x}_i^T\mathbf{x}_j$

24

Kernel Methods - Motivation

- **Kernels to the rescue**!
- Most learning algorithms such as Support Vector Machines, and Logistic regression (classification part used in deep networks) can be written in the form of Inner/dot product between vectors in the feature space $\phi(x_i)$s, i.e. $\langle \phi(x_i), \phi(x_j) \rangle$.
- The prediction function has the following form :

$$f(.) = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \text{Some function of} \langle (\phi(x_i), \phi(x_j)) \rangle \right)$$

- Kernels are functions which give us the dot product $\langle \phi(x_i), \phi(x_j) \rangle$ directly without explicitly computing the feature expansion $\phi(.)$

Dot Product - recall

- Dot product between feature vector of objects is a **measure of similarity between objects**
- In $\mathbb{R}^d$, dot product between two vectors x, y is given by $\langle x, y \rangle = ||x|| \times ||y|| \times cos(\theta)$, where θ is an angle between the vectors x and y.

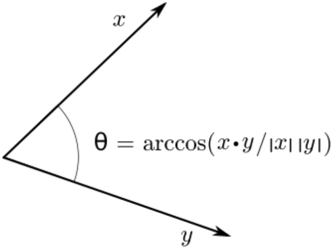


Figure: Figure from Wikipedia

Kernel Methods - Motivation

- Computing the dot product **explicitly** $\langle \phi(x_i), \phi(x_j) \rangle$ can be very computationally expensive
  - Interesting feature spaces such as those induced by the Gaussian Kernel) are infinite dimensional
  - In such a case, computing the inner product $\langle \phi(x_i), \phi(x_j) \rangle$ **explicitly** might not even be possible
- Kernel methods are a computational trick to compute the dot product **implicitly**
  - We do not have write down the explicit form of the feature maps $\phi(x_i)$, but rather compute the dot product directly using the kernel function $k(.,.)$
  - $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
  - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
  - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \ldots, x^{(d)}x^{(d)}\}$
- Computational complexity of
  - $\langle \phi(x), \phi(z) \rangle$?

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
  - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
  - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \ldots, x^{(d)}x^{(d)}\}$
- Computational complexity of
  - $\langle \phi(x), \phi(z) \rangle$? $O(d^2)$

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \dots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
    - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
    - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \dots x^{(1)}x^{(d)}, \dots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \dots, x^{(d)}x^{(d)}\}$
- Computational complexity of
    - $\langle \phi(x), \phi(z) \rangle$? $O(d^2)$
    - $k_{poly}(x, z)$?

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
  - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
  - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \ldots, x^{(d)}x^{(d)}\}$
- Computational complexity of
  - $\langle \phi(x), \phi(z) \rangle$? $O(d^2)$
  - $k_{poly}(x, z)$? $O(d)$

Implicit vs Explicit Computation

- Example - Polynomial kernel of degree 2. Assuming training data points $x, z \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$$k_{poly}(x, z) := (\langle x, z \rangle)^2 = \langle \phi(x), \phi(z) \rangle$$

- What is the dimensionality of the input space and feature space, $\phi(.)$ (keeping order into account. i.e., $x^{(1)}x^{(2)}$ is different from $x^{(2)}x^{(1)}$)?
  - $\phi : \mathbb{R}^d \mapsto \mathbb{R}^{d^2}$
  - $\phi(x) = \{x^{(1)}x^{(1)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(d)}x^{(1)}, x^{(d)}x^{(2)}, \ldots, x^{(d)}x^{(d)}\}$
- Computational complexity of
  - $\langle \phi(x), \phi(z) \rangle$? $O(d^2)$
  - $k_{poly}(x, z)$? $O(d)$
- What if we consider another kernel with a higher degree such as $k_{poly}(x, z) = (\langle x, z \rangle)^{10}$

Gaussian kernel

Gaussian kernel - Closer points are more similar

- is given by

$$k_{gaussian}(x, z) := \exp\left(-\frac{||x - z||^2}{2\sigma^2}\right), \forall x, z \in \mathbb{R}^d$$

  where $\sigma > 0$ is the kernel bandwidth
- What is the range of values for the Gaussian kernel $k_{gaussian}(x, z)$?

Gaussian kernel

Gaussian kernel - Closer points are more similar

- is given by

$$k_{gaussian}(x, z) := \exp\left(-\frac{||x - z||^2}{2\sigma^2}\right), \forall x, z \in \mathbb{R}^d$$

  where $\sigma > 0$ is the kernel bandwidth
- What is the range of values for the Gaussian kernel $k_{gaussian}(x, z)$?
- For $\sigma = 1$
  - $k(x, z) = 1$ when $x = z$, and
  - $k_{gaussian}(x, z) \approx 0$ when $||x - z||^2 = 10$ (Since $\exp(-5) = 0.006$)

Towards defining a Kernel

How should a kernel be defined, which takes into account

- **Inner product** $\langle ., . \rangle$ for quantifying similarity
- The high (potentially infinite) dimensional **feature map** $\phi(.)$
- The high (potentially infinite) dimensional **feature space** $\mathcal{H}$

What is a Kernel - Definition

Definition

For a non-empty set $\mathcal{X}$, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined to be a kernel if there exists a Hilbert Space $\mathcal{H}$ and a function $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$, $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$.

Vector Spaces (a slight detour)

Vector Space

Definition (Abstract Vector Space)

A vector space is non-empty set V, that is equipped or associated with two operations, (i) *"addition"* - For each pair of elements $v, w \in V$, it associates an element in V, denoted $w + v$, and (ii) *"scalar multiplication"* - For each element $v \in V$ and $\alpha \in \mathbb{R}$, it associates an element in V denoted by αv, **and**

Vector Space

Definition (Abstract Vector Space)

A vector space is non-empty set V, that is equipped or associated with two operations, (i) *"addition"* - For each pair of elements $v, w \in V$, it associates an element in V, denoted $w + v$, and (ii) *"scalar multiplication"* - For each element $v \in V$ and $\alpha \in \mathbb{R}$, it associates an element in V denoted by αv, **and**

If the above two operations of *addition* and *scalar multiplication* satisfy a set of (following) requirements, then V is called a vector space.

- For all $v, w \in V, v + w = w + v$
- There exists an element, called $\mathbf{0}$, in V, such that $\forall v \in V$, $v + \mathbf{0} = v$
- For $\alpha, \beta \in \mathbb{R}$, and $\forall v \in V$, $(\alpha + \beta)v = \alpha v + \beta v$
- For each $v, w \in V$, and $\alpha \in \mathbb{R}$, we have $\alpha(v + w) = \alpha v + \alpha w$
- For each $v \in V$, there exists $-v \in V$, such that $v + (-v) = \mathbf{0}$
- A few other similar conditions ...(**no need to memorize**!)

Examples of Vector Space

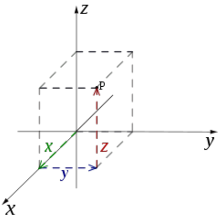


Figure: The familiar vector space $\mathbb{R}^3$ from linear algebra

1. $\mathbb{R}^d$ is a vector space
2. Space of functions can also be vector space. For example - The set W of polynomials of degree atmost 3

$$
\begin{aligned}
P(x) = & \quad a_3x^3 + a_2x^2 + a_1x + a_0, \text{ such that } x, a_i \in \mathbb{R} \\
Q(x) = & \quad b_3x^3 + b_2x^2 + b_1x + b_0, \text{ such that } x, b_i \in \mathbb{R}
\end{aligned}
$$

- $P(x) + Q(x) := (a_3 + b_3)x^3 + (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0) \in W$,
- For $\alpha P(x) := \alpha \times a_3x^3 + \alpha \times a_2x^2 + \alpha \times a_1x + \alpha \times a_0 \in W$, for $\alpha \in \mathbb{R}$

Normed Vector Spaces

Definition (Norm)

Let V be a vector space. A norm on V is a function (denoted as $||.||$)

$$||.|| : V \to \mathbb{R}^+$$

that satisfies the following requirements :

- $||v + w|| \leq ||v|| + ||w||, \forall v, w \in V$ (Triangle Inequality)
- $||\alpha v|| = |\alpha| \times ||v||, \forall v \in V$, and $\alpha \in \mathbb{R}$
- $||v|| \geq 0, \forall v \in V$, and $||v|| = 0$ if and only if $v = \mathbf{0}$ (Non-negativity)

A vector space equipped with a norm is called a normed vector space.

Examples of Normed Vector Spaces

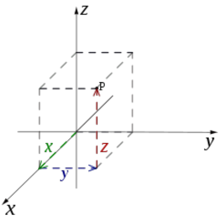


Figure: Figure from Wikipedia

Example1 - $\mathbb{R}^d$ (For $d = 3$, shown above) is a normed vector space with the ℓ_2 norm of an element $v \in \mathbb{R}^d$ given by $||v||_2 := \sqrt{\sum_{i=1}^{d} v_i^2}$

Examples of Normed Vector Spaces

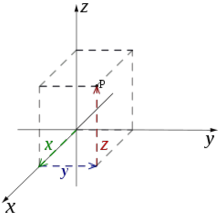


Figure: Figure from Wikipedia

Example1 - $\mathbb{R}^d$ (For $d = 3$, shown above) is a normed vector space with the ℓ_2 norm of an element $v \in \mathbb{R}^d$ given by $||v||_2 := \sqrt{\sum_{i=1}^{d} v_i^2}$

Example2 - $C[a, b] := \{f : [a, b] \to \mathbb{R} | f$ is a continuous function$\}$ (set of continuous functions over a bounded interval $[a, b]$) with the norm defined as

$$||f||_\infty := \max_{\{x \in [a,b]\}} |f(x)|$$

is a normed vector space

Inner Product Spaces

Definition (Inner Product)

Let V be a vector space. An inner product on V is a function

$$\langle ., . \rangle : V \times V \rightarrow \mathbb{R}$$

that satisfies the following requirements :

- $\langle \alpha v + \beta w, u \rangle = \alpha \langle v, u \rangle + \beta \langle w, u \rangle \forall u, v, w \in V$, and $\alpha, \beta \in \mathbb{R}$ (Linearity)
- $\langle v, w \rangle = \langle w, v \rangle \forall v, w \in V$ (Symmetry)
- $\langle v, v \rangle \geq 0, \forall v \in V$, and $\langle v, v \rangle = 0$ if and only if $v = 0$

A vector space equipped with an inner product is called an inner product space.

Examples of Inner Product Spaces

Examples

- For $\mathbb{R}^d$ the inner product is given by $\langle v, w \rangle = \sum_{i=1}^{d} v_i w_i$, also called the dot-product
- For function space $C[a, b]$, the inner product between real valued functions over a closed interval $[a, b]$ can be written as

$$\langle f, g \rangle := \int_a^b f(x)g(x)dx$$

Inner Product Spaces as Normed vector spaces

Inner proudct as a norm

Let V be a real vector space with an inner product $\langle ., . \rangle$. Then

$$||v|| := \sqrt{\langle v, v \rangle}, v \in V$$

defines a norm on V.

Proof.

Need to prove that for $v \in V$, $\sqrt{\langle v, v \rangle}$ is a norm on V, i.e. it satisfies the definition required for a function to be norm on a vector space. $\square$

Hilbert Spaces

A Hilbert Space refers to an Inner product space with some additional technical condition

What is a Kernel - Definition

Definition

For a non-empty set $\mathcal{X}$, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is defined to be a kernel if there exists a Hilbert Space $\mathcal{H}$ and a function $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X},\ k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$.

- $\phi(.)$ is called the feature map, and $\mathcal{H}$ is called the feature space
- $\mathcal{X}$ is only required to be a non-empty set
- No structure (such as a vector space) is required over $\mathcal{X}$, it can just be raw documents or pictures

Example of Kernel functions

- Polynomial kernel. Assuming inputs $x, x' \in \mathbb{R}^d$, i.e. $x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

  $$k_{poly}(x, x') := (\langle x, x' \rangle + c)^m$$

  for $c \geq 0, m \in \mathbb{N}$.
  - For $m = 1, c = 0, k(x, x') = \langle x, x' \rangle$ is called linear kernel
- Gaussian kernel

  $$k_{gaussian}(x, x') := \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right), \forall x, x' \in \mathbb{R}^d$$

  where $\sigma > 0$ is the kernel bandwidth

Note : We have not verified above if the above definition if satisfied for functions $k_{poly}(.,.)$ and $k_{gaussian}(.,.)$

Non-uniqueness of Feature Map and Hilbert Space

For a given Kernel, the feature map $\phi(.)$ and the Hilbert space $\mathcal{H}$ is non-unique.

- For the linear kernel $k(x, x') := \langle x, x' \rangle$, two of the many possible choices of feature maps and Hilbert space are :
  - $\phi_1(x) = x$, and $\mathcal{H}_1 = \mathbb{R}$
  - $\phi_2(x) = \frac{1}{\sqrt{2}}(x, x)$, and $\mathcal{H}_2 = \mathbb{R}^2$

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

What do we need to do to prove it is a kernel

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

What do we need to do to prove it is a kernel

- Find a feature map $\phi(.)$
- Find a feature space $\mathcal{H}$

such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

What do we need to do to prove it is a kernel

- Find a feature map $\phi(.)$
- Find a feature space $\mathcal{H}$

such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$

Proof.

$$\alpha k(x, x')$$

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

What do we need to do to prove it is a kernel

- Find a feature map $\phi(.)$
- Find a feature space $\mathcal{H}$

such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$

Proof.

$$\alpha k(x, x') = \alpha \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

Positive Scalar Multiple of a kernel

Positive scalar multiple

For any $\alpha > 0$, if $k(.,.)$ is a kernel, then $\alpha k(.,.)$ is also a kernel.

What do we need to do to prove it is a kernel

- Find a feature map $\phi(.)$
- Find a feature space $\mathcal{H}$

such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$

Proof.

$$\alpha k(x, x') = \alpha \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} = \langle \sqrt{\alpha}\phi(x), \sqrt{\alpha}\phi(x') \rangle_{\mathcal{H}}$$

□

Conic Sum of Kernels

Conic Sum of Kernels

For kernels $(k_j)_{j=1}^K$, and $(\alpha_j)_{j=1}^K > 0, \sum_{j=1}^K \alpha_j k_j$ is also a kernel

Proof.

$\sum_{j=1}^K \alpha_j k_j(x, x')$

Conic Sum of Kernels

Conic Sum of Kernels

For kernels $(k_j)_{j=1}^K$, and $(\alpha_j)_{j=1}^K > 0, \sum_{j=1}^K \alpha_j k_j$ is also a kernel

Proof.

$\sum_{j=1}^K \alpha_j k_j(x, x') = \sum_{j=1}^K \alpha_j \langle \phi_j(x), \phi_j(x') \rangle_{\mathcal{H}_j}$

Conic Sum of Kernels

Conic Sum of Kernels

For kernels $(k_j)_{j=1}^K$, and $(\alpha_j)_{j=1}^K > 0, \sum_{j=1}^K \alpha_j k_j$ is also a kernel

Proof.

$\sum_{j=1}^K \alpha_j k_j(x, x') = \sum_{j=1}^K \alpha_j \langle \phi_j(x), \phi_j(x') \rangle_{\mathcal{H}_j} =$
$\sum_{j=1}^K \langle \sqrt{\alpha_j}\phi_j(x), \sqrt{\alpha_j}\phi_j(x') \rangle_{\mathcal{H}_j}$

Conic Sum of Kernels

Conic Sum of Kernels

For kernels $(k_j)_{j=1}^K$, and $(\alpha_j)_{j=1}^K > 0, \sum_{j=1}^K \alpha_j k_j$ is also a kernel

Proof.

$\sum_{j=1}^K \alpha_j k_j(x, x') = \sum_{j=1}^K \alpha_j \langle \phi_j(x), \phi_j(x') \rangle_{\mathcal{H}_j} =$
$\sum_{j=1}^K \langle \sqrt{\alpha_j} \phi_j(x), \sqrt{\alpha_j} \phi_j(x') \rangle_{\mathcal{H}_j} = \langle \hat{\phi}(x), \hat{\phi}(x') \rangle_{\hat{\mathcal{H}}}$ $\square$

where $\hat{\phi}(x) = (\sqrt{\alpha_1}\phi_1(x), \ldots, \sqrt{\alpha_K}\phi_K(x))$, and $\hat{\mathcal{H}} = \mathcal{H}_1 \oplus \ldots \oplus \mathcal{H}_K$. Here $\oplus$ denotes axes concatenation of vector spaces. For example - $\mathbb{R}^2 \oplus \mathbb{R}^1 = \mathbb{R}^3$. i.e. adding a third dimension to a plane leads to a 3D space.

Difference of Kernels

Difference of Kernels is not necessarily a kernel

- Consider two kernels k_1 and k_2. Let there be an $x \in \mathcal{X}$ such that $k_1(x, x) - k_2(x, x) < 0$. Otherwise consider $k_2(x, x) - k_1(x, x)$, the same argument holds.

Difference of Kernels

Difference of Kernels is not necessarily a kernel

- Consider two kernels k_1 and k_2. Let there be an $x \in \mathcal{X}$ such that $k_1(x,x) - k_2(x,x) < 0$. Otherwise consider $k_2(x,x) - k_1(x,x)$, the same argument holds.
- If $k_1 - k_2$ is a kernel, then $\exists \phi$ and $\mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k_1(x,x') - k_2(x,x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

Difference of Kernels

Difference of Kernels is not necessarily a kernel

- Consider two kernels k_1 and k_2. Let there be an $x \in \mathcal{X}$ such that $k_1(x,x) - k_2(x,x) < 0$. Otherwise consider $k_2(x,x) - k_1(x,x)$, the same argument holds.
- If $k_1 - k_2$ is a kernel, then $\exists \phi$ and $\mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k_1(x,x') - k_2(x,x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

- How about $x = x'$?

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Proof.

$\hat{k}(x, x') =$

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Proof.

$\hat{k}(x, x') = f(x)k(x, x')f(x') =$

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Proof.

$\hat{k}(x, x') = f(x)k(x, x')f(x') = f(x)\langle \phi(x), \phi(x') \rangle_{\mathcal{H}} f(x') =$

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Proof.

$\hat{k}(x, x') = f(x)k(x, x')f(x') = f(x)\langle \phi(x), \phi(x') \rangle_{\mathcal{H}} f(x') =$
$\langle f(x)\phi(x), \phi(x')f(x') \rangle_{\mathcal{H}}$

Mappings

For an arbitrary function $f : \mathcal{X} \mapsto \mathbb{R}$, and a kernel $k(.,.)$

$$\hat{k}(x, x') = f(x)k(x, x')f(x')$$

is also a kernel

Proof.

$\hat{k}(x, x') = f(x)k(x, x')f(x') = f(x)\langle\phi(x), \phi(x')\rangle_{\mathcal{H}} f(x') = \langle f(x)\phi(x), \phi(x')f(x')\rangle_{\mathcal{H}}$, a different feature map $\hat{\phi}(x) := f(x)\phi(x)$ but the same Hilbert Space $\mathcal{H}$. □

The kernel matrix

- A **kernel matrix** (also called the **Gram matrix**), is an $n \times n$ matrix of pairwise similarity values :

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \dots & k(x_n, x_n) \end{bmatrix}$$

- Each entry is an inner product between two data points $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, where $\phi(.)$ is a feature map in vector form
- Since an inner product is symmetric, therefore K is a symmetric matrix
- In addition, K is positive definite matrix (will be proved in the next lecture)

Product of Kernels

Product of Kernels

For kernels k_1 and k_2 on the input space $\mathcal{X}$, the product kernel $k_1 \times k_2$ is a kernel.

Proof.

Note : Even though the statement above is true in general, the proof below is for the case of kernels with finite dimensional feature maps.

Let there be any collection of n points in $\mathcal{X}$, and K_1 and K_2 be the corresponding kernel matrices with feature maps ϕ_1 and ϕ_2. For simplicity, let $\phi_1(x) = x$ and $\phi_2(x) = y$

$$[K_1]_{ij} \times [K_2]_{ij} \quad = \quad \langle \phi_1(x_i), \phi_1(x_j) \rangle \times \langle \phi_2(x_i), \phi_2(x_j) \rangle = (x_i^T x_j)(y_i^T y_j)$$

Product of Kernels

Product of Kernels

For kernels k_1 and k_2 on the input space $\mathcal{X}$, the product kernel $k_1 \times k_2$ is a kernel.

Proof.

Note : Even though the statement above is true in general, the proof below is for the case of kernels with finite dimensional feature maps.

Let there be any collection of n points in $\mathcal{X}$, and K_1 and K_2 be the corresponding kernel matrices with feature maps ϕ_1 and ϕ_2. For simplicity, let $\phi_1(x) = x$ and $\phi_2(x) = y$

$$\begin{aligned} [K_1]_{ij} \times [K_2]_{ij} &= \langle \phi_1(x_i), \phi_1(x_j) \rangle \times \langle \phi_2(x_i), \phi_2(x_j) \rangle = (x_i^T x_j)(y_i^T y_j) \\ &= trace((x_i^T x_j)(y_i^T y_j)) \qquad \{trace \text{ is sum of diagonal elems of a matrix}\} \end{aligned}$$

Product of Kernels

Product of Kernels

For kernels k_1 and k_2 on the input space $\mathcal{X}$, the product kernel $k_1 \times k_2$ is a kernel.

Proof.

Note : Even though the statement above is true in general, the proof below is for the case of kernels with finite dimensional feature maps.

Let there be any collection of n points in $\mathcal{X}$, and K_1 and K_2 be the corresponding kernel matrices with feature maps ϕ_1 and ϕ_2. For simplicity, let $\phi_1(x) = x$ and $\phi_2(x) = y$

$$\begin{aligned}
[K_1]_{ij} \times [K_2]_{ij} &= \langle \phi_1(x_i), \phi_1(x_j) \rangle \times \langle \phi_2(x_i), \phi_2(x_j) \rangle = (x_i^T x_j)(y_i^T y_j) \\
&= trace((x_i^T x_j)(y_i^T y_j)) \quad \{trace \text{ is sum of diagonal elems of a matrix}\} \\
&= trace((x_i^T x_j)(y_j^T y_i)) \quad \{symmetry \text{ of } K_2\}
\end{aligned}$$

Product of Kernels

Product of Kernels

For kernels k_1 and k_2 on the input space $\mathcal{X}$, the product kernel $k_1 \times k_2$ is a kernel.

Proof.

Note : Even though the statement above is true in general, the proof below is for the case of kernels with finite dimensional feature maps.

Let there be any collection of n points in $\mathcal{X}$, and K_1 and K_2 be the corresponding kernel matrices with feature maps ϕ_1 and ϕ_2. For simplicity, let $\phi_1(x) = x$ and $\phi_2(x) = y$

$$\begin{aligned}
[K_1]_{ij} \times [K_2]_{ij} &= \langle \phi_1(x_i), \phi_1(x_j) \rangle \times \langle \phi_2(x_i), \phi_2(x_j) \rangle = (x_i^T x_j)(y_i^T y_j) \\
&= trace((x_i^T x_j)(y_i^T y_j)) \quad \{\textit{trace is sum of diagonal elems of a matrix}\} \\
&= trace((x_i^T x_j)(y_j^T y_i)) \quad \{\textit{symmetry of } K_2\} \\
&= trace((y_i x_i^T)(x_j y_j^T)) \quad \{\textit{cyclic invariance of trace}\}
\end{aligned}$$

Product of Kernels

Product of Kernels

For kernels k_1 and k_2 on the input space $\mathcal{X}$, the product kernel $k_1 \times k_2$ is a kernel.

Proof.

Note : Even though the statement above is true in general, the proof below is for the case of kernels with finite dimensional feature maps.

Let there be any collection of n points in $\mathcal{X}$, and K_1 and K_2 be the corresponding kernel matrices with feature maps ϕ_1 and ϕ_2. For simplicity, let $\phi_1(x) = x$ and $\phi_2(x) = y$

$$\begin{aligned}
[K_1]_{ij} \times [K_2]_{ij} &= \langle \phi_1(x_i), \phi_1(x_j) \rangle \times \langle \phi_2(x_i), \phi_2(x_j) \rangle = (x_i^T x_j)(y_i^T y_j) \\
&= trace((x_i^T x_j)(y_i^T y_j)) \qquad \{trace \text{ is sum of diagonal elems of a matrix}\} \\
&= trace((x_i^T x_j)(y_j^T y_i)) \qquad \{symmetry \text{ of } K_2\} \\
&= trace((y_i x_i^T)(x_j y_j^T)) \qquad \{cyclic \text{ invariance of trace}\} \\
&= \langle z_i, z_j \rangle_{\mathbb{R}^{n^2}} \qquad \{trace(A^T B) = \langle vec(A), vec(B) \rangle\}
\end{aligned}$$

where $z_i = vec(x_i y_i^T)$ and $z_j = vec(x_j y_j^T)$. $[K_1]_{ij} \times [K_2]_{ij}$ can therefore be written as an inner-product in $\mathbb{R}^{n^2}$ and hence $k_1 \times k_2$ is a kernel. □

Polynomial Kernels

Polynomial kernel

Let $x, x' \in \mathbb{R}^d$ for $d \geq 1$, and $m \geq 1$ be an integer, and $c \geq 0$ is a positive real number, then

$$k(x, x') := (\langle x, x' \rangle + c)^m$$

is a kernel

Proof.

Homework exercise

Hint : Use Binomial Theorem $\square$

Polynomial Kernels

Polynomial kernel

Let $x, x' \in \mathbb{R}^d$ for $d \geq 1$, and $m \geq 1$ be an integer, and $c \geq 0$ is a positive real number, then

$$k(x, x') := (\langle x, x' \rangle + c)^m$$

is a kernel

Proof.

Homework exercise

Hint : Use Binomial Theorem $\square$

Linear Kernel

For $m = 1$, and $c = 0$, $k(x, x') := \langle x, x' \rangle$ is called linear kernel

Exponential Kernel

Exponential Kernel

$$k_{exp}(x, x') = \exp(\langle x, x' \rangle), x, x' \in \mathbb{R}^d$$

is a kernel

Proof.

$$\exp(\langle x, x' \rangle) =$$

Exponential Kernel

Exponential Kernel

$$k_{exp}(x, x') = \exp(\langle x, x' \rangle), x, x' \in \mathbb{R}^d$$

is a kernel

Proof.

$$\exp(\langle x, x' \rangle) = \sum_{i=0}^{\infty} \frac{\langle x, x' \rangle^i}{i!} \text{ (By Taylor Series expansion of } \exp z\text{)}$$

Exponential Kernel

Exponential Kernel

$$k_{exp}(x, x') = \exp(\langle x, x' \rangle), x, x' \in \mathbb{R}^d$$

is a kernel

Proof.

$$\exp(\langle x, x' \rangle) = \sum_{i=0}^{\infty} \frac{\langle x, x' \rangle^i}{i!} \text{ (By Taylor Series expansion of } \exp z\text{)}$$

Since $\langle x, x' \rangle$ is a kernel, RHS is a kernel by sum and product rule $\square$

Cosine Kernel

For inputs $x, x' \in \mathbb{R}$, **Cosine Kernel** (given below)

$$k_{cosine}(x, x') = \cos(x - x')$$

is a kernel

Proof.

$$\cos(x - x')$$

Cosine Kernel

For inputs $x, x' \in \mathbb{R}$, **Cosine Kernel** (given below)

$$k_{cosine}(x, x') = \cos(x - x')$$

is a kernel

Proof.

$$\cos(x - x') = \cos(x)\cos(x') + \sin(x)\sin(x')$$

Cosine Kernel

For inputs $x, x' \in \mathbb{R}$, **Cosine Kernel** (given below)

$$k_{cosine}(x, x') = \cos(x - x')$$

is a kernel

Proof.

$$\cos(x - x') = \cos(x)\cos(x') + \sin(x)\sin(x')$$

Therefore, it is a kernel with feature map $\phi(x) = (\cos(x), \sin(x))^T$ $\square$

Cosine Kernel

For inputs $x, x' \in \mathbb{R}$, **Cosine Kernel** (given below)

$$k_{cosine}(x, x') = \cos(x - x')$$

is a kernel

Proof.

$$\cos(x - x') = \cos(x)\cos(x') + \sin(x)\sin(x')$$

Therefore, it is a kernel with feature map $\phi(x) = (\cos(x), \sin(x))^T$ $\square$

How about $\cos(x + x')$?

Gaussian Kernel

For inputs $x, x' \in \mathbb{R}^d$, **Gaussian Kernel** (given below)

$$k_{gaussian}(x, x') = \exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) \text{ for } x, x' \in \mathbb{R}^d$$

where σ is fixed, **is a kernel**

Proof.

$$\exp\left(-\frac{||x - x'||^2}{\sigma^2}\right)$$

Gaussian Kernel

For inputs $x, x' \in \mathbb{R}^d$, **Gaussian Kernel** (given below)

$$k_{gaussian}(x, x') = \exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) \text{ for } x, x' \in \mathbb{R}^d$$

where σ is fixed, **is a kernel**

Proof.

$$\exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) = \exp\left(-\frac{||x||^2 + ||x'||^2 - 2\langle x, x'\rangle}{\sigma^2}\right)$$

Gaussian Kernel

For inputs $x, x' \in \mathbb{R}^d$, **Gaussian Kernel** (given below)

$$k_{gaussian}(x, x') = \exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) \text{ for } x, x' \in \mathbb{R}^d$$

where σ is fixed, **is a kernel**

Proof.

$$\begin{aligned}
\exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) &= \exp\left(-\frac{||x||^2 + ||x'||^2 - 2\langle x, x'\rangle}{\sigma^2}\right) \\
&= \exp\left(-\frac{||x||^2}{\sigma^2}\right) \exp\left(\frac{2\langle x, x'\rangle}{\sigma^2}\right) \exp\left(-\frac{||x'||^2}{\sigma^2}\right)
\end{aligned}$$

Gaussian Kernel

For inputs $x, x' \in \mathbb{R}^d$, **Gaussian Kernel** (given below)

$$k_{gaussian}(x, x') = \exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) \text{ for } x, x' \in \mathbb{R}^d$$

where σ is fixed, **is a kernel**

Proof.

$$\begin{aligned}
\exp\left(-\frac{||x - x'||^2}{\sigma^2}\right) &= \exp\left(-\frac{||x||^2 + ||x'||^2 - 2\langle x, x'\rangle}{\sigma^2}\right) \\
&= \exp\left(-\frac{||x||^2}{\sigma^2}\right) \exp\left(\frac{2\langle x, x'\rangle}{\sigma^2}\right) \exp\left(-\frac{||x'||^2}{\sigma^2}\right) \\
&= f(x) \exp\left(\frac{2\langle x, x'\rangle}{\sigma^2}\right) f(x')
\end{aligned}$$

□

Summary

- Linear vs Non-linear classification
- Vector, Inner product, and Hilbert Spaces
- Kernels
  - Definition and Feature Mapping
  - Finite and infinite-dimensional feature spaces
- Kernel Properties
  - Conic combination of kernels
  - Product of kernels

The lecture follows material from below :

- http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/lecture4_introToRKHS.pdf

Books for further study

- Learning with kernels - Schoelkopf and Smola
- Kernel Methods for Pattern Analysis - Shawe-Taylor and Christianini