Kernels

Course of Machine Learning

Master Degree in Computer Science

University of Rome “Tor Vergata”

Giorgio Gambosi

a.a. 2018-2019

Idea

- Thus far, we have been assuming that each object that we deal with can be represented as a fixed-size feature vector $\mathbf{x} \in \mathbb{R}^d$
- For certain kinds of objects (text document, protein sequence, parse tree, etc.) it is not clear how to best represent them in this way
    1. first approach: define a generative model of data (with latent variables) and define an object as the inferred values of latent variables
    2. second approach: do not rely on vector representation, but just assume a similarity measure between objects is defined

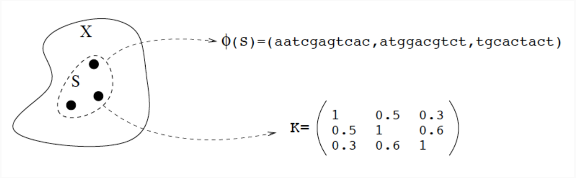
Representation by pairwise comparison

Idea

- Define a comparison function $\kappa : \chi \times \chi \mapsto \mathbb{R}$
- Represent a set of data items $\mathbf{x}_1, \dots \mathbf{x}_n$ by the $n \times n$ Gram matrix $\mathbf{G}$ such that

$$\mathbf{G}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

- $\mathbf{G}$ is always an $n \times n$ matrix, whatever the nature of data: the same algorithm will work for any type of data (vectors, strings, ...)



$\phi(S)$=(aatcgagtcac,atggacgtct,tgcactact)

$$K=\begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$$

Kernel definition

Given a set χ, a function $\kappa : \chi^2 \mapsto \mathbb{R}$ is a kernel on χ if there exists a Hilbert space $\mathcal{H}$ (essentially, a vector space with dot product $\cdot$) and a map $\boldsymbol{\phi} : \chi \mapsto \mathcal{H}$ such that for all $\mathbf{x}_1, \mathbf{x}_2 \in \chi$ we have

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1) \cdot \boldsymbol{\phi}(\mathbf{x}_2)$$

We shall consider the particular but common case when $\mathcal{H} = \mathbb{R}^d$ for some $d > 0$, $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x}))$ and $\boldsymbol{\phi}(\mathbf{x}_1) \cdot \boldsymbol{\phi}(\mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$

$\boldsymbol{\phi}$ is called a feature map a $\mathcal{H}$ a feature space of κ

Kernel definition

Positive definitess of κ is a relevant property in this framework.

Positive semidefinitess

Given a set χ, a function $\kappa : \chi^2 \mapsto \mathbb{R}$ is positive semidefinite if for all $n \in \mathbb{N}$, $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \chi^n$ the corresponding Gram matrix is positive semidefinite, that is $\mathbf{z}^T \mathbf{G} \mathbf{z} \geq 0$ for all vectors $\mathbf{z} \in \mathbb{R}^n$

Why is positive semidefinitess relevant?

Let $\kappa : \chi \times \chi \mapsto \mathbb{R}$. Then κ is a kernel iff for all sets $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ the corresponding Gram matrix $\mathbf{G}$ is symmetric and positive semidefinite

Only if: $\mathbf{G}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ then clearly $\mathbf{G}_{ij} = \mathbf{G}_{ji}$. Moreover for any $\mathbf{z} \in \mathbb{R}^d$

$$\begin{aligned}
\mathbf{z}^T\mathbf{G}\mathbf{z} &= \sum_{i=1}^{d}\sum_{j=1}^{d} z_i \mathbf{G}_{ij} z_j = \sum_{i=1}^{d}\sum_{j=1}^{d} z_i \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j) z_j \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d} z_i \left(\sum_{k=1}^{d} \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) \right) z_j = \sum_{k=1}^{n}\sum_{i=1}^{d}\sum_{j=1}^{d} z_i \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) z_j \\
&= \sum_{k=1}^{d} \left(\sum_{i=1}^{d} z_i \phi_k(\mathbf{x}_i) \right)^2 \geq 0
\end{aligned}$$

Why are positive definite kernels relevant?

If: Given $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ if $\mathbf{G}$ is positive definite it is possible to compute an eigenvector decomposition

$$\mathbf{G} = \mathbf{U}^T \mathbf{\Lambda} \mathbf{U}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues $\lambda_i > 0$ and the columns $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of $\mathbf{U}$ are the corresponding eigenvectors. Then,

$$\mathbf{G}_{ij} = (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{u}_i)^T (\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{u}_j)$$

Then if we define $\phi(\mathbf{x}_i) = \mathbf{\Lambda}^{\frac{1}{2}} \mathbf{u}_i$ we get

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \mathbf{G}_{ij}$$

This results is valid only wrt the domain $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$. For the general case, consider $n \to \infty$ (as for example in gaussian processes)

Why are positive definite kernels relevant?

Using positive definite kernels allows to apply the kernel trick wherever useful.

Kernel trick
Any algorithm which processes finite-dimensional vectors in such a way to consider only pairwise dot products can be applied to higher (possibly infinite) dimensional vectors by replacing each dot product by a suitable application of a positive definite kernel.

- Many practical applications
- Vectors in the new space are manipulated only implicitly, through pairwise dot products, computed by evaluating the kernel function on the original pair of vectors

Example: Support vector machines. Also, many linear models for regression and classification can be reformulated in terms of a dual representation

Dual representations: example

Regularized sum of squares in regression with predefined basis function $\boldsymbol{\phi}(\mathbf{x})$

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i) - t_i\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \\
&= \frac{1}{2}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t})^T(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}
\end{aligned}
$$

where by definition of $\boldsymbol{\Phi} \in \mathbb{R}^{n\times d}$ it is $\boldsymbol{\Phi}_{ij} = \phi_j(\mathbf{x}_i)$

Setting $\dfrac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$, the resulting solution is

$$\hat{\mathbf{w}} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\mathbf{I}_d)^{-1}\boldsymbol{\Phi}^T\mathbf{t} = \boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{t}$$

since it is possible to prove that for any matrix $\mathbf{A} \in \mathbb{R}^{r\times c}$ it is

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}_r)^{-1}\mathbf{A}^T = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}_c)^{-1}$$

Dual representations: example

If we define the dual variables $\mathbf{a} = (\mathbf{\Phi}\mathbf{\Phi}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{t}$, we get $\mathbf{w} = \mathbf{\Phi}^T\mathbf{a}$.

By substituting $\mathbf{\Phi}^T\mathbf{a}$ to $\mathbf{w}$ we express the cost function in terms of $\mathbf{a}$, instead of $\mathbf{w}$, introducing a dual formulation of J.

$$\begin{aligned} J(\mathbf{a}) &= \frac{1}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a} + \frac{1}{2}\mathbf{t}^T\mathbf{t} - \mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a} \\ &= \frac{1}{2}\mathbf{a}^T\mathbf{G}\mathbf{G}\mathbf{a} + \frac{1}{2}\mathbf{t}^T\mathbf{t} - \mathbf{a}^T\mathbf{G}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^T\mathbf{G}\mathbf{a} \end{aligned}$$

where $\mathbf{G} = \mathbf{\Phi}\mathbf{\Phi}^T$ is the Gram matrix, such that by definition $\mathbf{G}_{ij} = \sum_{k=1}^{d}\phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)$

Dual representations: example

Setting the gradient of $\frac{\partial J(\mathbf{a})}{\partial \mathbf{a}} = \mathbf{0}$ it results

$$\hat{\mathbf{a}} = (\mathbf{G} + \mathbf{I}\lambda_n)^{-1}\mathbf{t}$$

We can use this to make predictions in a different way

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^T\phi(\mathbf{x}) = \mathbf{a}^T\mathbf{\Phi}\phi(\mathbf{x}) = \mathbf{t}^T(\mathbf{G} + \mathbf{I}\lambda_n)^{-1}\mathbf{\Phi}\phi(\mathbf{x}) \\ &= \mathbf{k}(\mathbf{x})(\mathbf{G} + \mathbf{I}\lambda_n)^{-1}\mathbf{t} \end{aligned}$$

where

$$\begin{aligned} \mathbf{k}(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{\Phi} &= (\phi(\mathbf{x}_1)^T\phi(\mathbf{x}), \ldots, \phi(\mathbf{x}_n^T\phi(\mathbf{x}))^T \\ &= (\kappa(\mathbf{x}_1, \mathbf{x}), \ldots, \kappa(\mathbf{x}_n, \mathbf{x}))^T \\ &= (\kappa_1(\mathbf{x}), \ldots, \kappa_n(\mathbf{x}))^T \end{aligned}$$

Dual representations: another example

- As well known, a perceptron is a linear classifier with prediction $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- Its update rule is: If $\mathbf{x}_i$ is misclassified, that is $\mathbf{w}^T\mathbf{x}_i t_i < 0$, then $\mathbf{w} := \mathbf{w} + t_i\mathbf{x}_i$
- If we assume a zero initial value for all w_k, then $\mathbf{w}$ is the sum of all items that have been considered as misclassified by the algorithm, where each item is weighted by the number of times it has been considered.
- We may then define a dual formulation by setting $\mathbf{w} = \sum_{k=1}^n a_k\mathbf{x}_k$, which results in prediction $y(\mathbf{x}) = \sum_{k=1}^n a_k\mathbf{x}_k^T\mathbf{x}$
- and update rule: if $\mathbf{x}_i$ is misclassified, that is $\sum_{k=1}^n a_k\mathbf{x}_k^T\mathbf{x}_i < 0$, then $a_i := a_i + 1$
- a kernelized perceptron can be defined with $y(\mathbf{x}) = \sum_{k=1}^n a_k\boldsymbol{\phi}(\mathbf{x}_k)^T\boldsymbol{\phi}(\mathbf{x})$ or with $y(\mathbf{x}) = \sum_{k=1}^n a_k\kappa(\mathbf{x}_k, \mathbf{x})$, by just using a positive definite kernel κ

Kernelization: one more example

- The k-nn classifier selects the label of the nearest neighbor: assume the Euclidean distance is considered

$$||\mathbf{x}_i - \mathbf{x}_j||^2 = \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$$

- We can now replace the dot products by a valid positive definite kernel and we obtain:

$$d(\mathbf{x}_i, \mathbf{x}_j)^2 = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_j, \mathbf{x}_j) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j)$$

- This is a kernelized nearest-neighbor classifier
- We do not explicitly compute vectors

Why referring to the dual representation?

- While in the original formulation of linear regression $\mathbf{w}$ can be derived by inverting the $m \times m$ matrix $\mathbf{\Phi}^T\mathbf{\Phi}$, in the dual formulation computing $\mathbf{a}$ requires inverting the $n \times n$ matrix $\mathbf{G} + \mathbf{I}\lambda$.
- Since usually $n \gg m$, this seems to lead to a loss of efficiency.

Dealing with kernels

Since not all functions $f : \chi \mapsto \mathbb{R}^d$ are positive definite kernel, some method to define them must be applied.

- the straighforward way is just to define a basis function $\boldsymbol{\phi}$ and define $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$. κ is a positive definite kernel since

1. $\phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) = \phi(\mathbf{x}_2)^T \phi(\mathbf{x}_1)$
2. $\sum_{i=1}^n \sum_{j=1}^n c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = || \sum_{i=1}^n c_i \phi(\mathbf{x}_i) ||^2 \geq 0$

Dealing with kernels

- a second method defines a possible kernel function κ directly: in order to ensure that such function is a valid kernel, apply Mercer's theorem and prove that κ is a positive definite kernel by showing it is simmetric and the corresponding Gram matrix $\mathbf{G}$ is positive definite for all possible sets of items. In this case we do not define $\boldsymbol{\phi}$

A simple positive definite kernel

Let $\chi = \mathbb{R}$: the function $\kappa : \mathbb{R}^2 \mapsto \mathbb{R}$ defined as

$$\kappa(x_1, x_2) = x_1 x_2$$

is a positive definite kernel. In fact,

- $x_1 x_2 = x_2 x_1$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \kappa(x_i, x_j) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j x_i x_j = \left(\sum_{i=1}^{n} c_i x_i)^2\right) \geq 0$

Another simple positive definite kernel

Let $\chi = \mathbb{R}^d$: the function $\kappa : \chi^2 \mapsto \mathbb{R}$ defined as

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{x}_2$$

is a positive definite kernel. In fact,

- $\mathbf{x}_1^T \mathbf{x}_2 = \mathbf{x}_2^T \mathbf{x}_1$
- $\sum_{i=1}^n \sum_{j=1}^n c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \mathbf{x}_i^T \mathbf{x}_j = || \sum_{i=1}^n c_i \mathbf{x}_i ||^2 \geq 0$

Dealing with kernels

- a third method defines again a possible kernel function κ directly: in order to ensure that such function is a valid kernel, a basis function $\boldsymbol{\phi}$ must be found such that $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$ for all $\mathbf{x}_1, \mathbf{x}_2$

Example

A polynomial kernel in 2d: $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$

$$\begin{aligned}
\kappa(\mathbf{x}_1, \mathbf{x}_2) &= (x_{11}^2, \sqrt{2}x_{11}x_{12}, x_{12}^2)^T(x_{21}^2, \sqrt{2}x_{21}x_{22}, x_{22}^2) \\
&= x_{11}^2 x_{21}^2 + 2x_{11}x_{12}x_{21}x_{22} + x_{12}^2 x_{22}^2 \\
&= ||\mathbf{x}_1^T \mathbf{x}_2||^2
\end{aligned}$$

Example

If $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ define $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2 = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$, where

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \ldots, x_d^2, x_1 x_2, \ldots, x_1 x_d, x_2 x_1, \ldots, x_d x_{d-1})^T$$

Example

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2$ is a valid kernel function, since

$$\begin{aligned}
\kappa(\mathbf{x}_1, \mathbf{x}_2) &= (x_{11}x_{21} + x_{12}x_{22})^2 \\
&= x_{11}^2 x_{21}^2 + x_{12}^2 x_{22}^2 + 2x_{11}x_{12}x_{21}x_{22} \\
&= (x_{11}^2, x_{12}^2, x_{11}x_{12}, x_{11}x_{12}) \cdot (x_{21}^2, x_{22}^2, x_{21}x_{22}, x_{21}x_{22}) \\
&= \boldsymbol{\phi}(\mathbf{x}_1) \cdot \boldsymbol{\phi}(\mathbf{x}_2)
\end{aligned}$$

The basis function thus results $\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_1x_2)^T$.

Example

- In general, if $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$ then $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2)^2 = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$, where

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_1x_d, x_2x_1, \ldots, x_dx_{d-1})^T$$

- the d-dimensional input space is mapped onto a space with dimension $m = d^2$
- observe that computing $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ requires time $O(d)$, while deriving it from $\boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2)$ requires $O(d^2)$ steps

Example

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^2$ is a kernel function, since

$$\begin{aligned}\kappa(\mathbf{x}_1, \mathbf{x}_2) &= (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_{1i}x_{1j}x_{2i}x_{2j} + \sum_{i=1}^{n}(\sqrt{2}cx_{1i})(\sqrt{2}cx_{2i}) + c^2 \\ &= \boldsymbol{\phi}(\mathbf{x}_1)^T\boldsymbol{\phi}(\mathbf{x}_2)\end{aligned}$$

for

$$\boldsymbol{\phi}(\mathbf{x}) = (x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_1x_d, x_2x_1, \ldots, x_dx_{d-1}, \sqrt{2c}x_1, \ldots, \sqrt{2c}x_d$$

This implies a mapping from a d-dimensional to a $(d+1)^2$-dimensional space.

Example

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^t$ is a kernel function corresponding to a mapping from a d-dimensional space to a space of dimension

$$m = \sum_{i=0}^{t} d^i = \frac{d^{t+1} - 1}{d - 1}$$

corresponding to all products $x_{i_1} x_{i_2} \ldots x_{i_l}$ with $0 \leq l \leq t$.

Observe that, even if the space has dimension $O(d^t)$, evaluating the kernel function requires just time $O(d)$.

Constructing kernels from kernels

More complex kernels can be derived from simpler ones by appying suitable transormation and composition rules. In fact, given kernel functions $\kappa_1(\mathbf{x}_1, \mathbf{x}_2)$, $\kappa_2(\mathbf{x}_1, \mathbf{x}_2)$, the function $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ is a kernel in all the following cases

- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{\kappa_1(\mathbf{x}_1, \mathbf{x}_2)}$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_1(\mathbf{x}_1, \mathbf{x}_2) + \kappa_2(\mathbf{x}_1, \mathbf{x}_2)$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_1(\mathbf{x}_1, \mathbf{x}_2)\kappa_2(\mathbf{x}_1, \mathbf{x}_2)$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = c\kappa_1(\mathbf{x}_1, \mathbf{x}_2)$, for any $c > 0$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T \mathbf{A} \mathbf{x}_2$, with $\mathbf{A}$ positive definite
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_1)\kappa_1(\mathbf{x}_1, \mathbf{x}_2)g(\mathbf{x}_2)$, for any $f, g : \mathbb{R}^n \mapsto \mathbb{R}$
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = p(\kappa_1(\mathbf{x}_1, \mathbf{x}_2))$, for any polynomial $p : \mathbb{R}^q \mapsto \mathbb{R}$ with non-negative coefficients
- $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa_3(\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2))$, for any vector $\boldsymbol{\phi}$ of m functions $\phi_i : \mathbb{R}^n \mapsto \mathbb{R}$ and for any kernel function $\kappa_3(\mathbf{x}_1, \mathbf{x}_2)$ in $\mathbb{R}^m$

Costructing kernel functions

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ is a kernel function. In fact,

1. $\mathbf{x}_1 \cdot \mathbf{x}_2 = \mathbf{x}_1^T \mathbf{x}_2$ is a kernel function corresponding to the base functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$, with $\phi_i(\mathbf{x}) = \mathbf{x}$
2. c is a kernel function corresponding to the base functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$, with $\phi_i(\mathbf{x}) = \dfrac{\sqrt{c}}{n}$
3. $\mathbf{x}_1 \cdot \mathbf{x}_2 + c$ is a kernel function since it is the sum of two kernel functions
4. $(\mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$ is a kernel function since it is a polynomial with non negative coefficients (in particular $p(z) = z^d$) of a kernel function

Costructing kernel functions

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2}}$$

is a kernel function. In fact,

1. since $||\mathbf{x}_1 - \mathbf{x}_2||^2 = \mathbf{x}_1^T\mathbf{x}_1 + \mathbf{x}_2^T\mathbf{x}_2 - 2\mathbf{x}_1^T\mathbf{x}_2$, it results
$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = e^{-\frac{\mathbf{x}_1^T\mathbf{x}_1}{2\sigma^2}} e^{-\frac{\mathbf{x}_2^T\mathbf{x}_2}{2\sigma^2}} e^{\frac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}}$$
2. $\mathbf{x}_1^T\mathbf{x}_2$ is a kernel function (see above)
3. then, $\dfrac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}$ is a kernel function, being the product of a kernel function with a constant $c = \dfrac{1}{\sigma^2}$
4. $e^{\frac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}}$ is the exponential of a kernel function, and as a consequence a kernel function itself
5. $e^{-\frac{\mathbf{x}_1^T\mathbf{x}_1}{\sigma^2}} e^{-\frac{\mathbf{x}_1^T\mathbf{x}_1}{2\sigma^2}} e^{\frac{\mathbf{x}_1^T\mathbf{x}_2}{\sigma^2}}$ is a kernel function, being the product of a kernel function with two functions $f(\mathbf{x}_1) = e^{-\frac{\mathbf{x}_1^T\mathbf{x}_1}{2\sigma^2}}$ and

Relevant kernel functions

- Polynomial kernel

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^d$$

- Sigmoidal kernel

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \tanh\left(c_1 \mathbf{x}_1 \cdot \mathbf{x}_2 + c_2\right)$$

- Gaussian kernel

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{||\mathbf{x}_1 - \mathbf{x}_2||^2}{2\sigma^2}\right)$$

  where $\sigma \in \mathbb{R}$

Observe that a gaussian kernel can be derived also starting from a non linear kernel function $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ instead of $\mathbf{x}_1^T \mathbf{x}_2$.

Kernels of structured objects

Kernels are particularly useful when applied to structured objects.

Consider the case of strings (for example sequences of DNA bases or amino acids).

Given two strings $\mathbf{x}_1, \mathbf{x}_2$ on a same alphabet $\mathcal{A}$, we can define their similarity to be equal to the number of substrings they have in common.

More formally, let $\phi_s(\mathbf{x})$ be the number of times substring s occurs in $\mathbf{x}$ and let $\boldsymbol{\phi}(\mathbf{x})$ the corresponding vector of such functions for all substrings s: a kernel can be defined as

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1)^T \boldsymbol{\phi}(\mathbf{x}_2) = \sum_{s \in \mathcal{A}^*} w_s \phi_s(\mathbf{x}_1) \phi_s(\mathbf{x}_2)$$

where $w_s \geq 0$ are predefined weights.

Kernels of structured objects

If $w_s = 1$ for all considered substrings and we define $\boldsymbol{\phi}'(\mathbf{x}) = \dfrac{\boldsymbol{\phi}(\mathbf{x})}{||\boldsymbol{\phi}(\mathbf{x})||^2}$ as a normalized version of $\boldsymbol{\phi}$, we get

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \frac{\boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_2)}{||\boldsymbol{\phi}(\mathbf{x}_1)||^2 ||\boldsymbol{\phi}(\mathbf{x}_2)||^2}$$

that is, the well known cosine similarity measure.

Borrowing from information retrieval methods, a better similarity measure can be obtained by defining $\phi_s(\mathbf{x})$ through more sophisticated measures, such as tf-idf, instead of occurrences counting

Kernels of structured objects

Special cases:

- $w_s = 0$ if $|s| > 1$ is a bag-of-chars kernel, with $\phi_c(\mathbf{x})$ being the number of occurrences of character c in $\mathbf{x}$
- If only s delimited by white spaces are considered, we get a bag-of-words kernel.
- If only strings of fixed length $|s| = k$ are considered, we have a k-spectrum kernel
- …

The approach can be extended to the case of trees, in order to deal with, for example, parse or evolutionary trees

More complex kernel construction techniques have been defined.