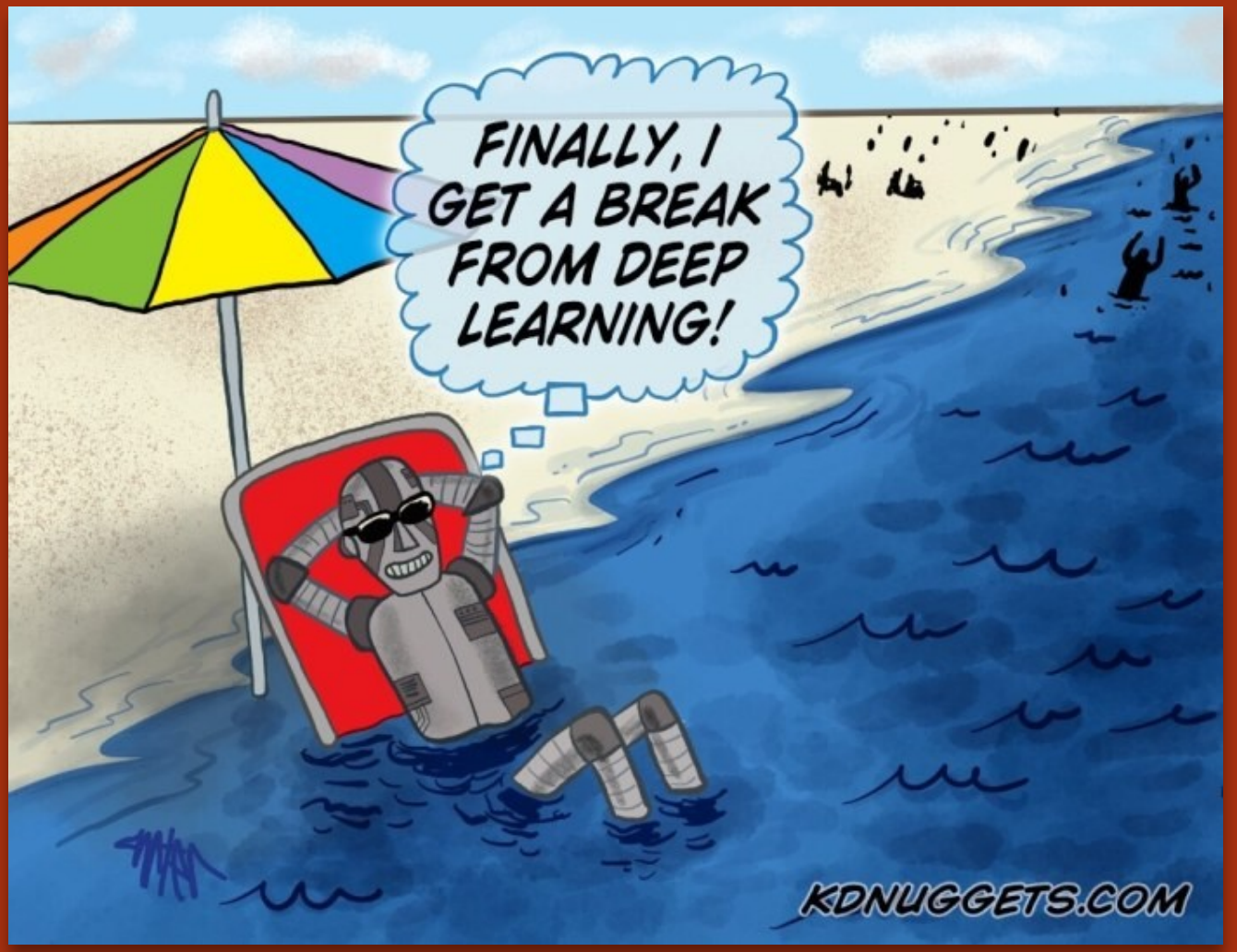
# Kernel Methods

Marius Popescu
University of Bucharest
popescunmarius@gmail.com

# What? Shallow Methods in Deep Learning Age?



# There are Tasks for which Shallow Methods are SOTA

Kernel methods can be preferable to deep learning in scenarios with limited data, when interpretability is crucial, or when computational resources are constrained

# Kernel Methods - A First Attempt of Definition

Kernel methods are an approach to machine learning problems (all kind, supervised, unsupervised, etc.)

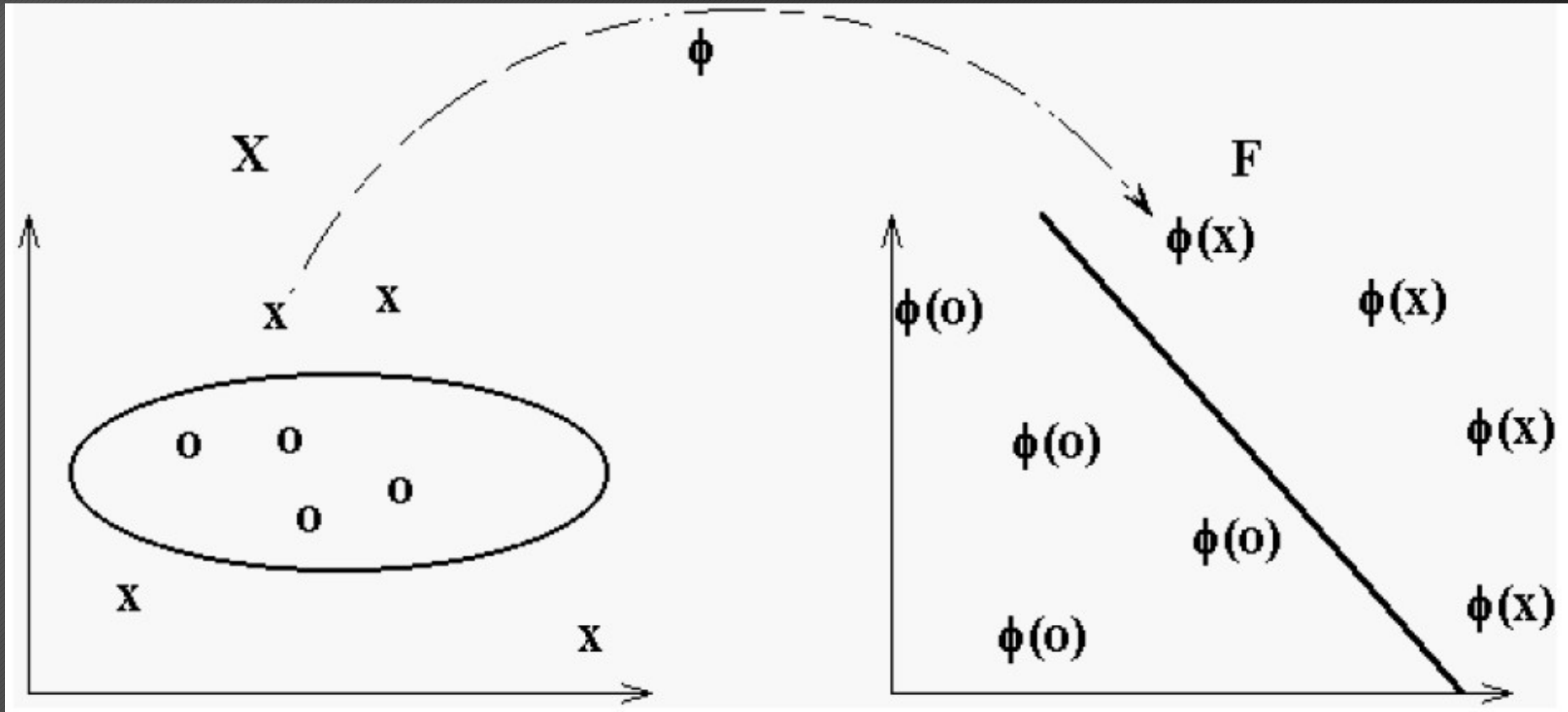
The kernel methods combine the theoretically well-founded approach specific to linear learning methods (like perceptron and ridge regression), with the flexibility of nonlinear methods (like neural networks) and with the rigour of statistical approach (like regularisation from multidimensional statistics).

# Kernel Methods Strategy

Kernel-based learning algorithms work by embedding the data into a feature space (a Hilbert space), and searching for linear relations in that space. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly.

Despite of looking for linear relations in the feature space, in the original space can be discovered nonlinear relations because the embedding process (kernel function) can be (and usually is) a nonlinear one. In this way the nonlinear relation in the original data are transformed in linear relations in the new space in which the data are embedded

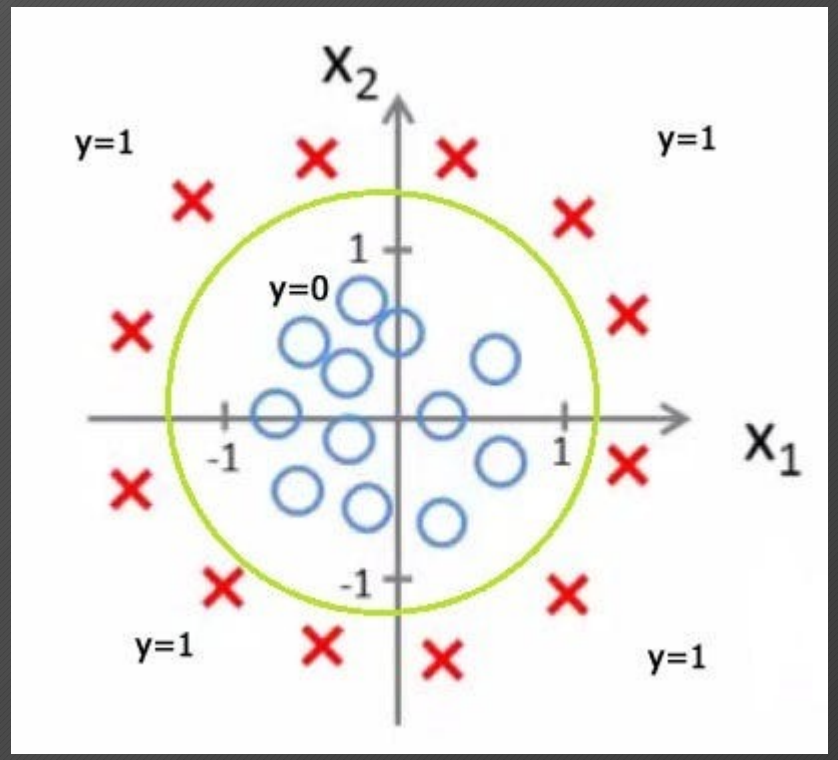
# Kernel Methods Strategy



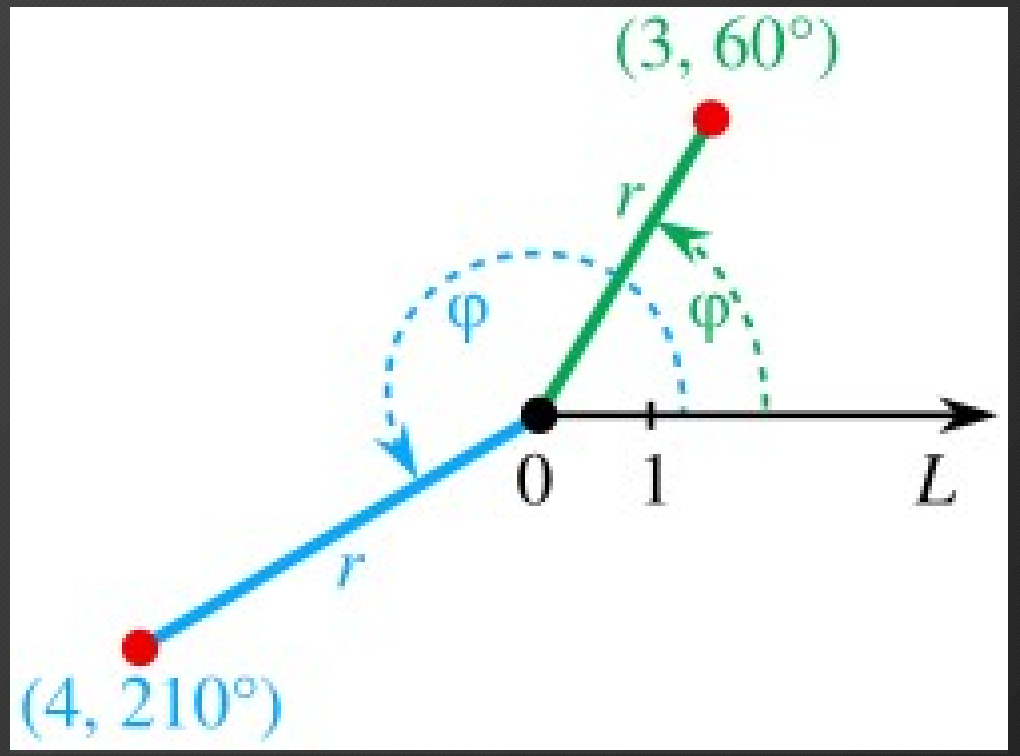

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

# An Example

**Are linearly separable?**




**What about the polar coordinate system?**



# Kernel Methods Strategy

**The main ingredients of kernel methods:**

(i) Data items are embedded into a vector space called the feature space.

(ii) Linear relations are sought among the images of the data items in the feature space.

(iii) The algorithms are implemented in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products.

(iv) The pairwise inner products can be computed efficiently directly from the original data items using a kernel function.

(ii) Linear relations are sought among the images of the data items in the feature space.

# Linear Regression

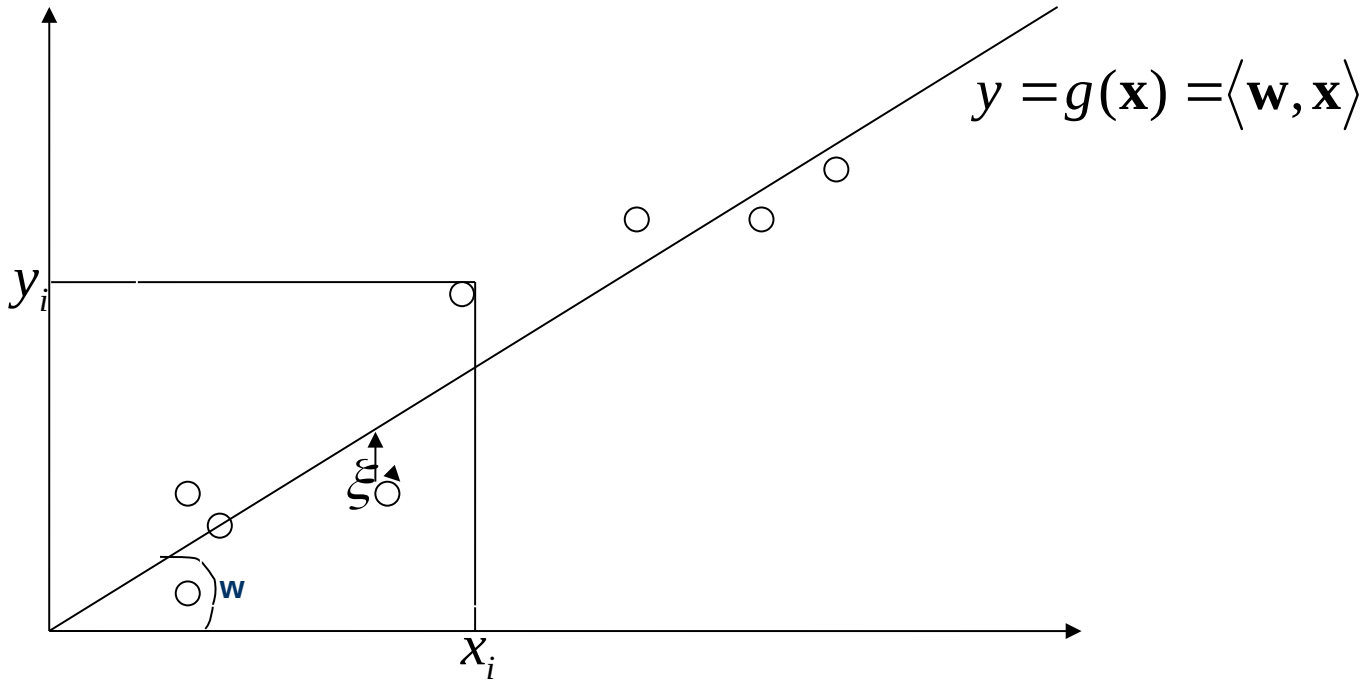
The problem of finding $g$ of the form:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

that best interpolates a given training set

$$S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

# Linear Regression



# Linear Regression

The error of the linear function on a particular training example is :

$$\xi = (y - g(\mathbf{x}))$$

The collective loss (on all training data) is :

$$\mathrm{L}(g, S) = \mathrm{L}(\mathbf{w}, S) = \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2 =$$

$$= \sum_{i=1}^{\ell} \xi^2 = \sum_{i=1}^{\ell} \mathrm{L}(g, (\mathbf{x_i}, y_i))$$

# Linear Regression

Loss written vectorially:

$$\boldsymbol{\xi} = \mathbf{y} - \mathbf{X}\mathbf{w}$$

$$L(\mathbf{w}, S) = \|\boldsymbol{\xi}\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w})$$

# Linear Regression

The optimal $\mathbf{w}$ :

$$\frac{\partial \mathrm{L}(\mathbf{w}, S)}{\partial \mathbf{w}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

so (the normal equation) :

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{X}'\mathbf{y}$$

if the inverse exists :

$$\mathbf{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

# Ridge Regression

If the inverse dosen't exist, the problem is "ill-conditioned" and it needs regularisation. The optimisation criterion becomes :

$$\min_{\mathbf{w}} \mathsf{L}_{\lambda}(\mathbf{w}, S) = \min_{\mathbf{w}} (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2)$$

and solution will be :

$$\frac{\partial \mathsf{L}_{\lambda}(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2)}{\partial \mathbf{w}} = \mathbf{0}$$

# Ridge Regression

The solution :

$$\frac{\partial \mathrm{L}_{\lambda}(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial(\lambda \|\mathbf{w}\|^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w}))}{\partial \mathbf{w}} =$$

$$= 2\lambda \mathbf{w} - 2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_n)\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_n)^{-1}\mathbf{X}'\mathbf{y}$$

(iii) The algorithms are implemented in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products.

# Dual Ridge Regression

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = \lambda^{-1}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}) = \lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\text{But}: \mathbf{w} = \mathbf{X}'\boldsymbol{\alpha}$$

$$\text{so}: \boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

# Dual Ridge Regression

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\lambda\boldsymbol{\alpha} = (\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\mathbf{X}\mathbf{X}'\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha} = \mathbf{y}$$

$$(\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_\ell)\boldsymbol{\alpha} = \mathbf{y}$$

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda\mathbf{I}_\ell)^{-1}\mathbf{y}$$

$$\text{where: } \mathbf{G} = \mathbf{X}\mathbf{X}'$$

$$\text{is the Gram matrix } \mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

# Dual Ridge Regression

In the dual solution the information from the training examples is given only by the inner products between pairs of training points in the Gram matrix $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y}$$

The prediction function is given by

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i}=\mathbf{1}}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

Similarly, the information about a novel example **x** required by the predictive function is just the inner products between the training points and the new example **x**

(iv) The pairwise inner products can be computed efficiently directly from the original data items using a kernel function.

# Kernel Ridge Regression

"Kernel trick": just replace the inner product by another similarity function $k$ ($\langle\ \rangle \mapsto k$)

$$\mathbf{G} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_2, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_n \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \langle \mathbf{x}_n, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{pmatrix} \mapsto$$

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

# Kernel Ridge Regression

(dual) weights :

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y}$$

$$\downarrow$$

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

The prediction function :

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i}=1}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

$$\downarrow$$

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

# Kernels

# Kernels & Embedding

A kernel is a function $k : X \times X \rightarrow \Re$ for which there is a mapping $\phi : X \rightarrow F$ from $X$ to a Hilbert space $F$ such that :

$$\forall \mathbf{x}, \mathbf{z} \in X, \quad k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F$$

where $\langle .,. \rangle_F$ is the inner product in $F$

Theorem : $k$ is a kernel if and only if $k$ is symmetric and finitely positive semi - definite

# Kernels & Embedding

Consider a two-dimensional input space $X = \Re^2$

together with the feature map

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \Re^3$$

The hypothesis space of linear functions in $F$ would then be

$$\mathsf{G} = \left\{ g : \Re^2 \to \Re \mid g(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle = w_{11}x_1^2 + w_{22}x_2^2 + w_{12}\sqrt{2}x_1x_2 \right\}$$

# Kernels & Embedding

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \right\rangle$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = (x_1z_1 + x_2z_2)^2$$

$$\left\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \right\rangle_F = \left\langle \mathbf{x}, \mathbf{z} \right\rangle^2$$

$$k(\mathbf{x}, \mathbf{z}) = \left\langle \mathbf{x}, \mathbf{z} \right\rangle^2$$

the same kernel computes the inner product corresponding to the four dimensional feature map

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in F = \Re^4$$

# Kernels Examples

Polynomial kernel :

$$k(\mathbf{x},\mathbf{z}) = (\langle \mathbf{x},\mathbf{z} \rangle + c)^d \text{ with } c \in \Re^+ \text{ and } d \in N$$

the parameter $c$ allows some control of the relative weightings of the different degree monomials. increasing $c$ decreases the relative weighting of the higher order monomials

# Kernels Examples

Gaussian (or RBF) kernel:

$$k(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right) \text{ with } \sigma \in \Re^+$$

The feature space has infinite-dimension for every value of $\sigma$.

The parameter $\sigma$ controls the flexibility of the kernel in a similar way to the degree $d$ in the polynomial kernel. Small values of $\sigma$ correspond to large values of $d$

# Operations on Kernels

Summing Kernels = Concatenating the corresponding embedding features

$$< [a_1, b_1], [a_2, b_2] >=< a_1, a_2 >+< b_1, b_2 >$$

Some combining rules :

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$$

$$k(\mathbf{x}, \mathbf{z}) = \alpha k_1(\mathbf{x}, \mathbf{z}), \quad a \in \Re^+$$

$$k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$$

Normalisation (corresponding to $\phi(\mathbf{x}) \mapsto \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|}$)

$$k(\mathbf{x}, \mathbf{z}) \mapsto \frac{k(\mathbf{x}, \mathbf{z})}{\sqrt{k(\mathbf{x}, \mathbf{x}) k(\mathbf{z}, \mathbf{z})}}$$

# Combining Kernels Using Kernel Alignment

Kernel Alignment:

$$A(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}}$$

$$\langle \mathbf{M}, \mathbf{N} \rangle_F = \mathrm{tr}(\mathbf{M}'\mathbf{N}) \quad \text{Frobenius Norm}$$

# Combining Kernels Using Kernel Alignment

Ideal kernel for a classification problem : $\mathbf{YY}'$

$\alpha_1 \sim A(\mathbf{K}_1, \mathbf{YY}')$

$\alpha_2 \sim A(\mathbf{K}_2, \mathbf{YY}')$

# String Kernels

# *p*-Spectrum Kernel

Perhaps one of the most natural ways to measure the similarity of two strings is to count how many substrings of length $p$ the two strings have in common. This gives rise to the $p$-spectrum kernel. Formally, for two strings over an alphabet $\Sigma$, $s, t \in \Sigma^*$, the $p$-spectrum kernel is defined as:

$$k_p(s,t) = \sum_{v \in \Sigma^p} \mathrm{num}_v(s) \cdot \mathrm{num}_v(t),$$

where $\mathrm{num}_v(s)$ is the number of occurrences of string $v$ as a substring in $s$.[1] The feature map defined by this kernel associates a vector of dimension $|\Sigma|^p$ containing the histogram of frequencies of all its substrings of length $p$ ($p$-grams) with each string.

# *p*-Grams Presence Bits Kernel

A variant of this kernel can be obtained if the embedding feature map is modified to associate a vector of dimension $|\Sigma|^p$ containing the presence bits (instead of frequencies) of all its substrings of length $p$ with each string. Thus, the character $p$-grams presence bits kernel is obtained:

$$k_p^{0/1}(s,t) = \sum_{v \in \Sigma^p} \text{in}_v(s) \cdot \text{in}_v(t),$$

where $\text{in}_v(s)$ is 1 if string $v$ occurs as a substring in $s$, and 0 otherwise.

# Intersection Kernel

In computer vision, the (histogram) intersection kernel has successfully been used for object class recognition from images (Maji et al. 2008; Vedaldi and Zisserman 2010). In the work of Ionescu et al. (2014), the intersection kernel is used for the first time as a kernel for strings. The intersection string kernel is defined as follows:

$$k_p^{\cap}(s,t) = \sum_{v \in \Sigma^p} \min\{\text{num}_v(s), \text{num}_v(t)\},$$

where $\text{num}_v(s)$ is the number of occurrences of string $v$ as a substring in $s$.

# Relation between the Three Kernels

For the $p$-spectrum kernel, the frequency of a $p$-gram has a very significant contribution to the kernel, since it considers the product of such frequencies. On the other hand, the frequency of a $p$-gram is completely disregarded in the $p$-grams presence bits kernel. The intersection kernel lies somewhere in the middle between the $p$-grams presence bits kernel and $p$-spectrum kernel, in the sense that the frequency of a $p$-gram has a moderate contribution to the intersection kernel. More precisely, the following inequality that describes the relation between the three kernels holds:

$$k_p^{0/1}(s,t) \leq k_p^{\cap}(s,t) \leq k_p(s,t).$$

# Blended Kernels

- Taking into account p-grams of different length and summing up the corresponding kernels, new kernels, termed *blended kernels*, can be obtained.

Summing Kernels = Concatenating the corresponding embedding features

$$< [a_1, b_1], [a_2, b_2] >=< a_1, a_2 >+< b_1, b_2 >$$

# Normalized Versions of the Kernels

To ensure a fair comparison of strings of different lengths, normalized versions of the $p$-spectrum kernel, the $p$-grams presence bits kernel, and the intersection kernel are being used:

$$\hat{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s) \cdot k_p(t,t)}},$$

$$\hat{k}_p^{0/1}(s,t) = \frac{k_p^{0/1}(s,t)}{\sqrt{k_p^{0/1}(s,s) \cdot k_p^{0/1}(t,t)}},$$

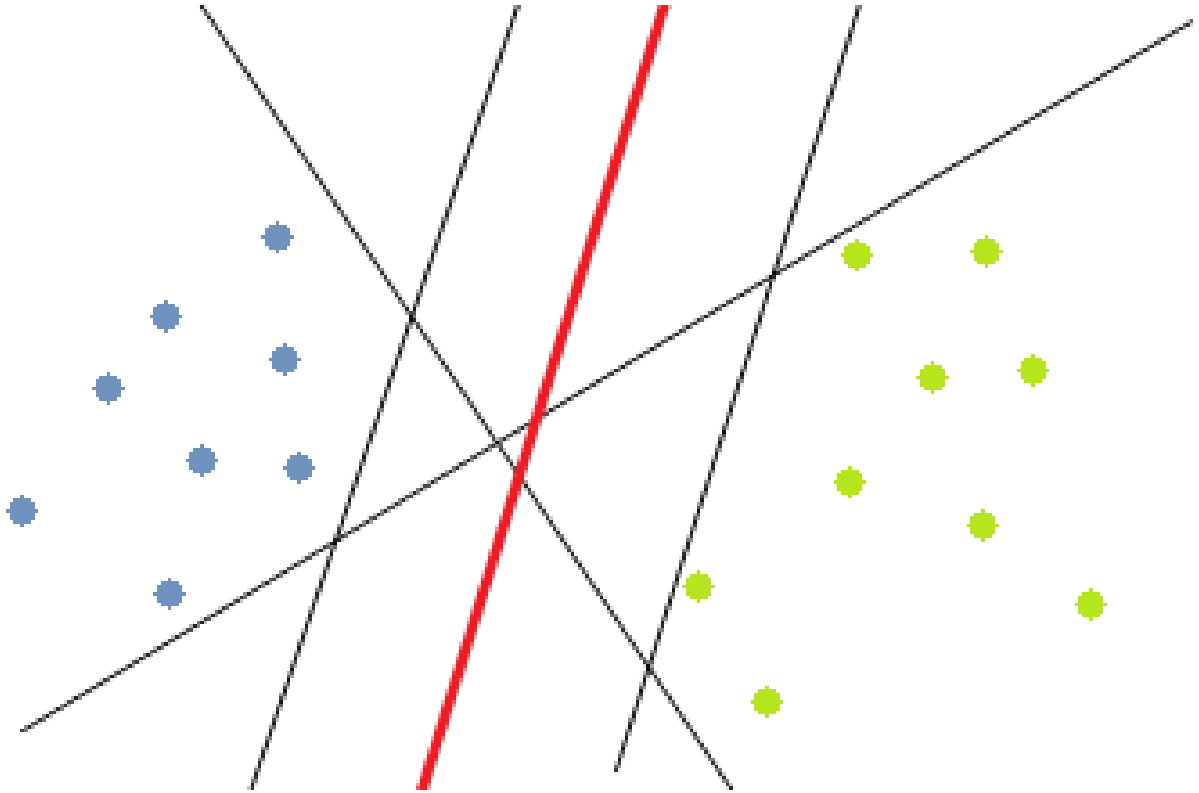
$$\hat{k}_p^{\cap}(s,t) = \frac{k_p^{\cap}(s,t)}{\sqrt{k_p^{\cap}(s,s) \cdot k_p^{\cap}(t,t)}}.$$

# Kernel Methods

- Support Vector Machines (SVM)
- Kernel Ridge Regression (KRR)
  Regularized Least-Squares Classification (RLSC)
- Kernel Fisher Discriminant (KFD)
- Kernel Partial Least Squares (KPLS)

# Support Vector Machines (SVM)

# Optimal Separating Hyperplane



## Optimal Separating Hyperplane
## Maximal Margin Hyperplane
## Hard Margin SVM

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

$$\max_{\mathbf{w},b,\gamma} \gamma$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma$$

$$i = 1, \ldots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

$$i = 1, \ldots, \ell$$

# Hard Margin SVM

| | |
|---|---|
| Input | training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$ |
| Process<br>maximise<br>subject to | find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:<br>$W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$<br>$\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i$, $i = 1, \ldots, \ell$. |
| 4 | $\gamma^* = \sqrt{-W(\boldsymbol{\alpha}^*)}$ |
| 5 | choose $i$ such that $0 < \alpha_i^*$ |
| 6 | $b = y_i (\gamma^*)^2 - \sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \mathbf{x}_i)$ |
| 7 | $f(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b\right)$; |
| 8 | $\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$ |
| Output | weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin $\gamma^*$ and function $f$ implementing the decision rule represented by the hyperplane |

# SVM (Soft Margin)

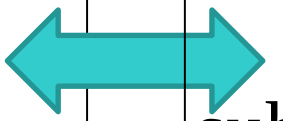
$$\min_{\mathbf{w},b,\gamma,\boldsymbol{\xi}} -\gamma + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma - \xi_i$$

$$\xi_i \geq 0 \quad i = 1,\ldots,\ell$$

$$\|\mathbf{w}\|^2 = 1$$



$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1,\ldots,\ell$$

# SVM (Soft Margin)

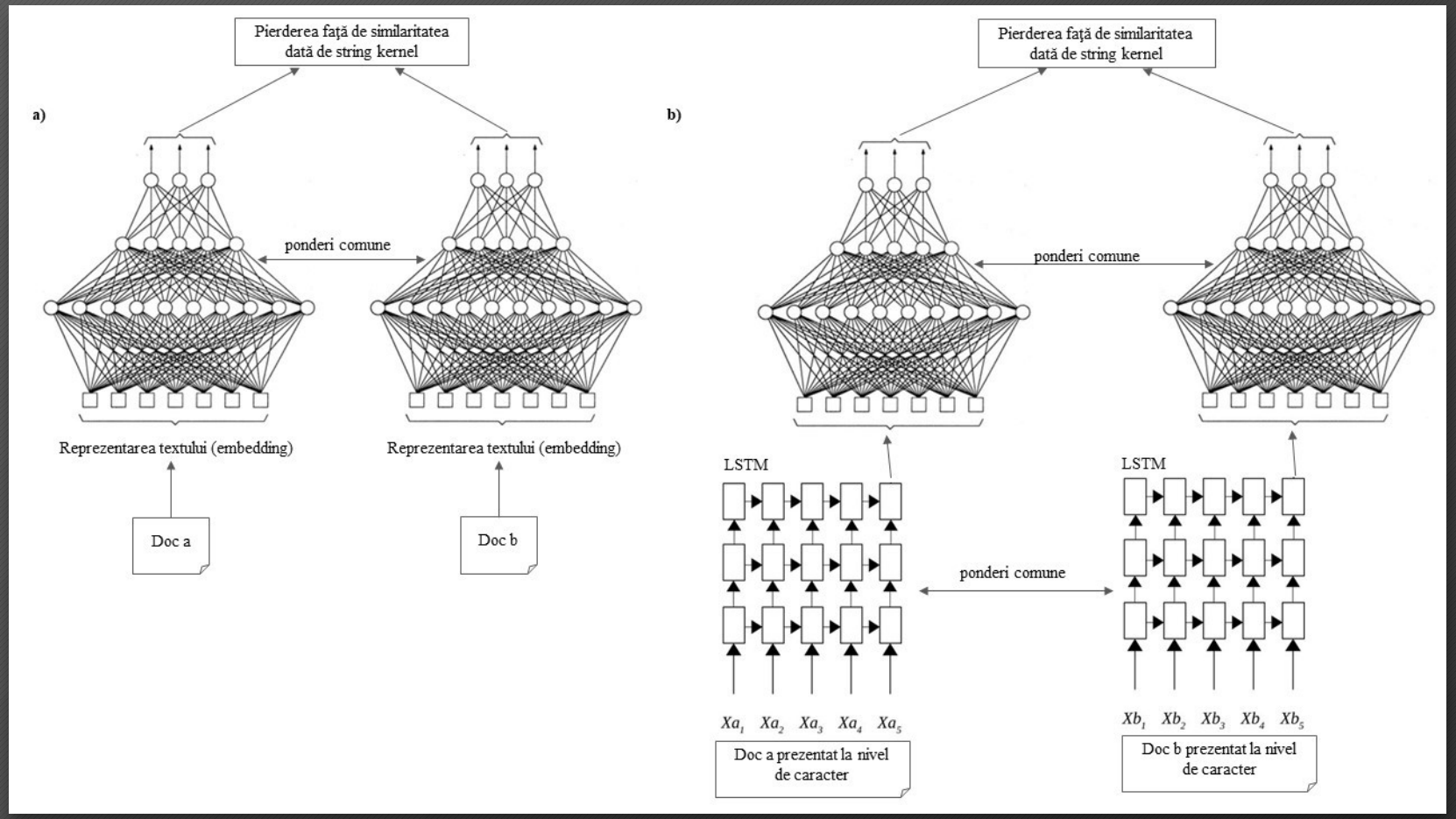
| | |
|---|---|
| Input | training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$, $C \in [1/\ell, \infty)$ |
| Process | find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem: |
| maximise | $W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$ |
| subject to | $\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i \leq C$, $i = 1, \ldots, \ell$. |
| 4 | $\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j)\right)^{1/2}$ |
| 5 | choose $i$, $j$ such that $-C < \alpha_i^* y_i < 0 < \alpha_j^* y_j < C$ |
| 6 | $b^* = -\lambda^*(\sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_i) + \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j))$ |
| 7 | $\gamma^* = 2\lambda^* \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j) + b^*$ |
| 8 | $f(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b^*\right)$; |
| 9 | $\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$ |
| Output | weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin $\gamma^*$ and function $f$ implementing the decision rule represented by the hyperplane |

# And yet Deep Learning …

# The Greatest Problem of Kernel Methods

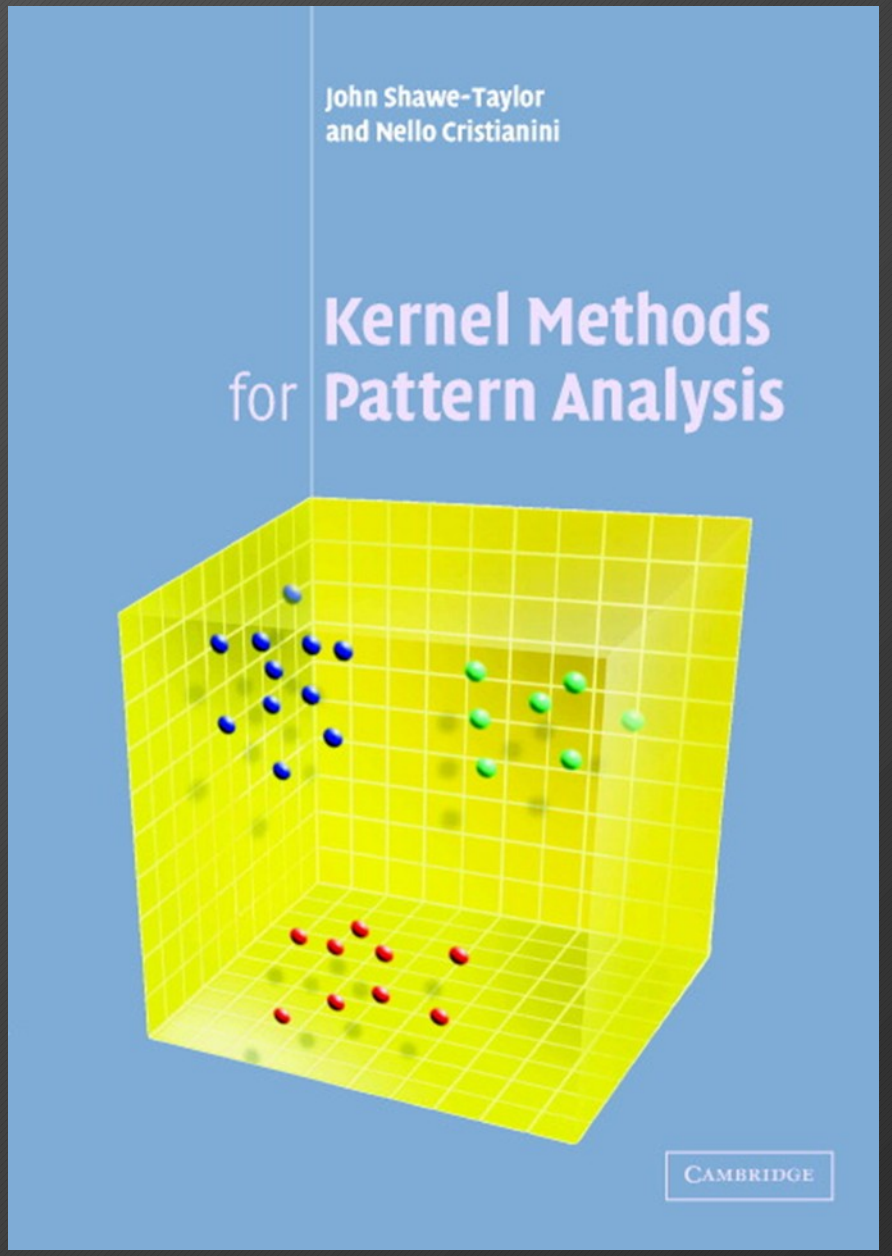
- At inference time you have to compute the kernel between the testing example and *all* training examples.
- Don’t scale for large training data sets

# Can we *distill* knowledge obtain from *feature engineering* into a *neural network* representation?




# One possible solution very suited for kernels induced knowledge

# Bibliography


John Shawe-Taylor
and Nello Cristianini
Kernel Methods
for Pattern Analysis
CAMBRIDGE

# Thank You