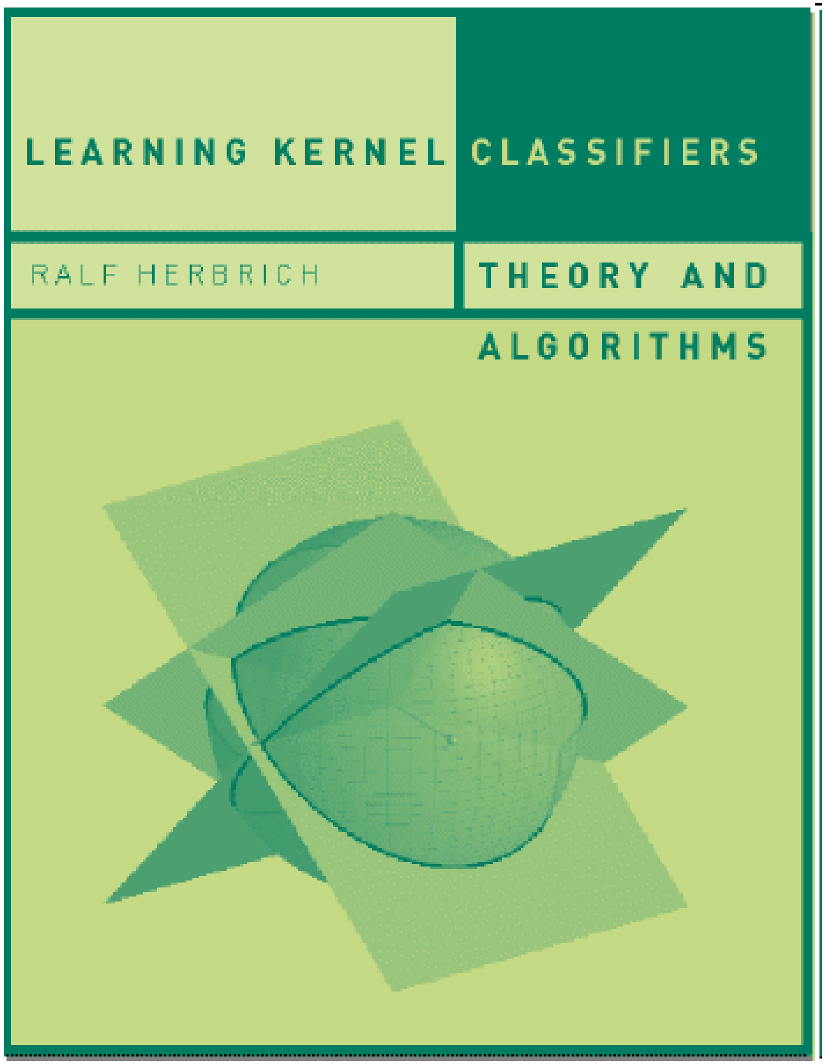
Kernel Methods

Barnabás Póczos

Outline

- Quick Introduction
- Feature space
- Perceptron in the feature space
- Kernels
- Mercer's theorem
  - Finite domain
  - Arbitrary domain
- Kernel families
  - Constructing new kernels from kernels
- Constructing feature maps from kernels
- Reproducing Kernel Hilbert Spaces (RKHS)
- The Representer Theorem

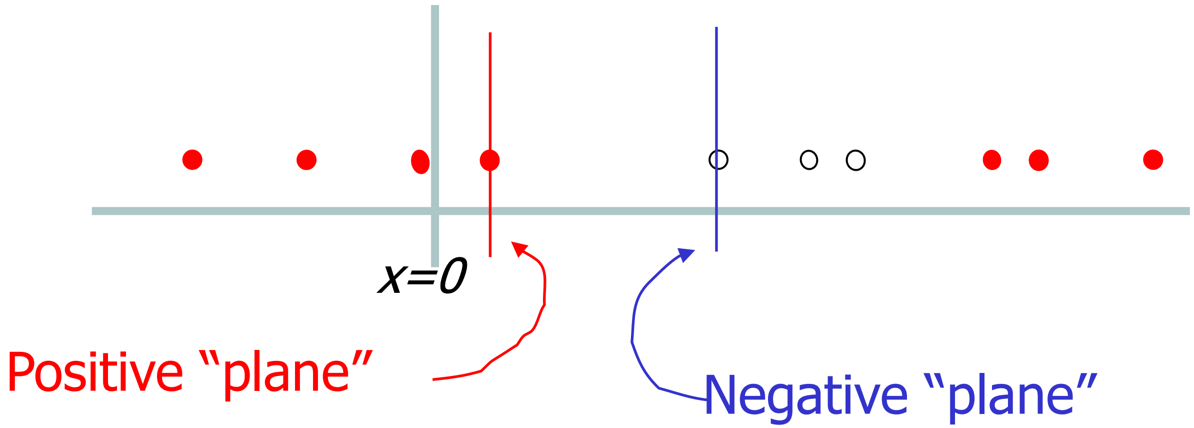
Ralf Herbrich: Learning Kernel Classifiers Chapter 2



Quick Overview

Hard 1-dimensional Dataset

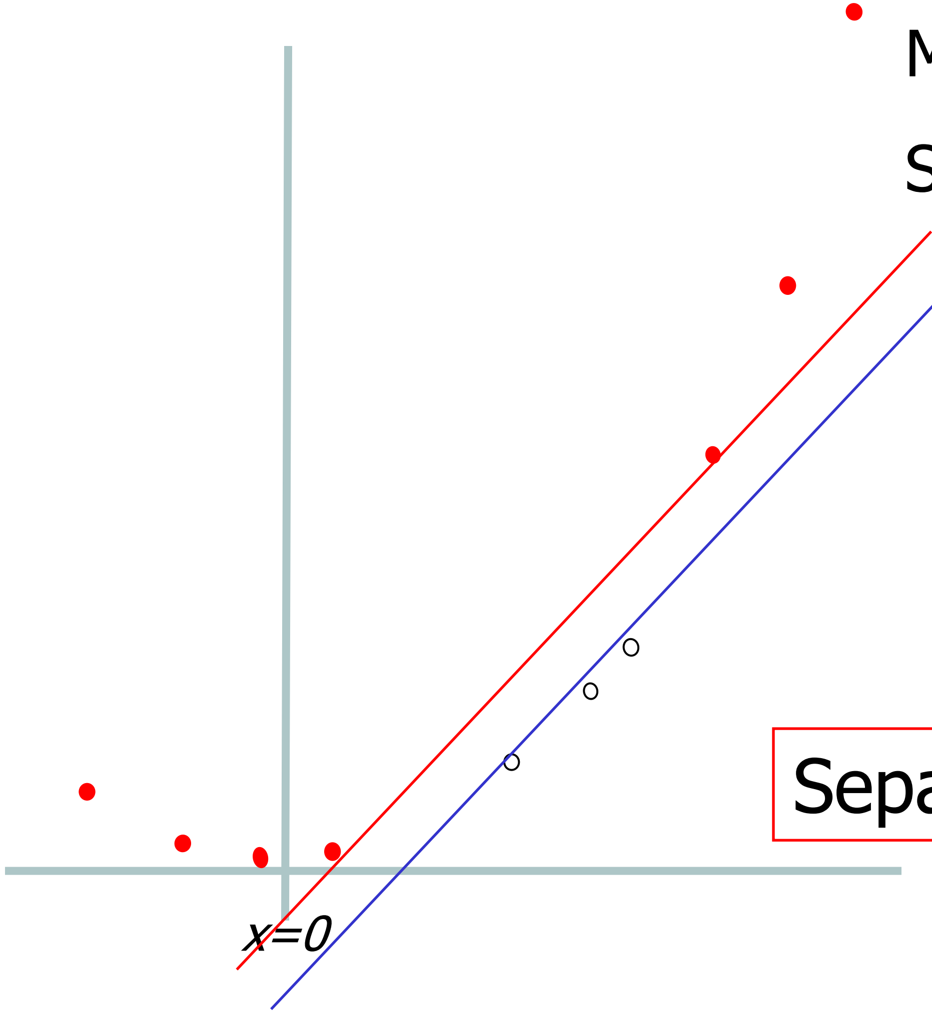
- If the data set is **not** linearly separable, then adding new features (mapping the data to a larger feature space) the data might become linearly separable



- m general! points in an m-1 dimensional space is always linearly separable by a hyperspace!
  $\Rightarrow$ it is good to map the data to high dimensional spaces

(For example 4 points in 3D)

Hard 1-dimensional Dataset



M*ake up a new feature!*

Sort of…

… computed from original feature(s)

$$\mathbf{z}_k = (x_k, x_k^2)$$

Separable! MAGIC!

Now drop this "augmented" data into our linear SVM.

taken from Andrew W. Moore

Feature mapping

- *m* general! points in an *m-1* dimensional space is always linearly separable by a hyperspace!
  $\Rightarrow$ it is good to map the data to high dimensional spaces

- Having *m* training data, is it always enough to map the data into a feature space with dimension *m-1*?

  - *Nope... We have to think about the test data as well! Even if we don't know how many test data we have...*

  - *We might want to map our data to a huge (∞) dimensional feature space*

  - *Overfitting? Generalization error?... We don't care now...*

Feature mapping, but how???

Let us have m training objects: $\vec{x}_i = [\vec{x}_{i,1}, \vec{x}_{i,2}] \in \mathbb{R}^2$, $i = 1, \ldots, m$

The possible test objects are denoted by $\vec{x} = [\vec{x}_1, \vec{x}_2] \in \mathbb{R}^2$

How to map x to a huge dimensional space?

... for example by a random map:

$$\text{Let } \phi(\vec{x}) \doteq \underbrace{[sin(\vec{x}_2), \exp(\vec{x}_2 + \vec{x}_1), \vec{x}_1, \vec{x}_2^{\tan(\vec{x}_1)}, \ldots]}_{\infty}$$

Observation

Several algorithms use the inner products only, but not the feature values!!!

E.g. Perceptron, SVM, Gaussian Processes...

The Perceptron

Algorithm 2 Perceptron learning algorithm (in dual variables).

Require: A feature mapping $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{K} \subseteq \ell_2^n$

Ensure: A linearly separable training sample $\boldsymbol{z} = ((x_1, y_1), \ldots, (x_m, y_m))$

$\boldsymbol{\alpha} = \mathbf{0}$
repeat
  for $j = 1, \ldots, m$ **do**
    if $y_j \sum_{i=1}^{m} \alpha_i \langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j) \rangle \leq 0$ **then**
      $\alpha_j \leftarrow \alpha_j + y_j$
    end if
  end for
until no mistakes have been made within the **for** loop
return the vector $\boldsymbol{\alpha}$ of expansion coefficients

SVM

Maximize $\underset{\alpha_k}{} \sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints:

$$0 \le \alpha_k \le C \quad \forall k \qquad \sum_{k=1}^{R} \alpha_k y_k = 0$$

Inner products

So we need the inner product between

$$\mathbf{x}_i = \phi(\vec{x}_i) \doteq [sin(\vec{x}_{i,2}), \exp(\vec{x}_{i,2} + \vec{x}_{i,1}), \vec{x}_{i,1}, \vec{x}_{i,2}^{\tan(\vec{x}_{i,1})}, \ldots]$$

and

$$\mathbf{x_j} = \phi(\vec{x}_j) \doteq [sin(\vec{x}_{j,2}), \exp(\vec{x}_{j,2} + \vec{x}_{j,1}), \vec{x}_{j,1}, \vec{x}_{j,2}^{\tan(\vec{x}_{j,1})}, \ldots]$$

$$\boxed{k(\vec{x}_i, \vec{x}_j) \doteq \langle \mathbf{x_i}, \mathbf{x_j} \rangle = ???}$$

Looks ugly, and needs lots of computation...

Can't we just say that let

$$k(\vec{x}_i, \vec{x}_j) \doteq \exp(-\|\vec{x}_i - \vec{x}_j\|^2) \ ???$$

There might exist a map $\phi(\vec{x})$ to this function k...

Finite example

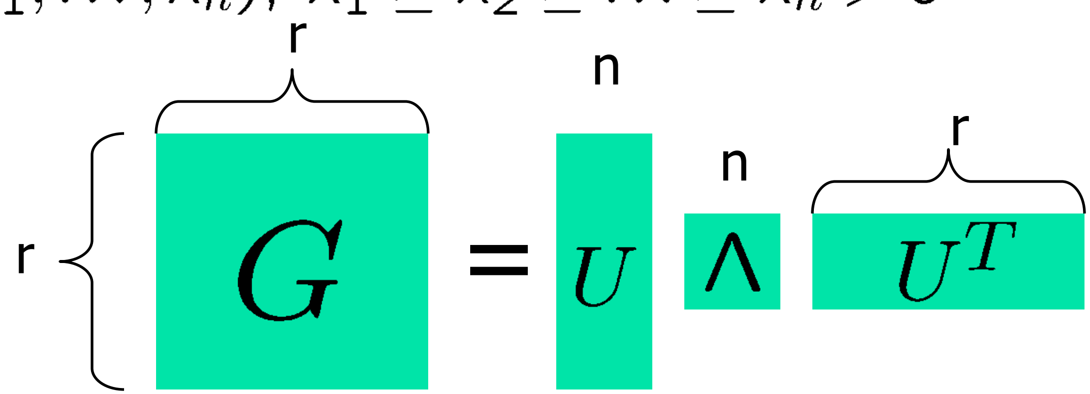
$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a FINITE set } \mathcal{X} = \{x_1, \ldots, x_r\}\end{array}\right\} \Rightarrow \text{construct } \mathcal{K} \text{ and } \phi$$

$\Rightarrow G \in \mathbb{R}^{r \times r}$, $G_{ij} = k(x_i, x_j)$ can be calculated

$$G \text{ is symmetric, PSD} \Rightarrow G = U \Lambda U^T \text{ by SVD.}$$

$$U^T U = I_n,\ n = rank(U),\ U = \begin{bmatrix} u_1^T \\ \vdots \\ u_r^T \end{bmatrix} \in \mathbb{R}^{r \times n}$$

$$\Lambda = diag(\lambda_1, \ldots, \lambda_n),\ \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$$



Finite example

Lemma:

Let $\mathcal{K} = span\{\phi(x_1), \ldots \phi(x_r)\}$

$$\Rightarrow \phi(x_i) \doteq \Lambda^{1/2} u_i \in \mathbb{R}^n, \; i = 1, \ldots, r$$

leads back to the Gram matrix G

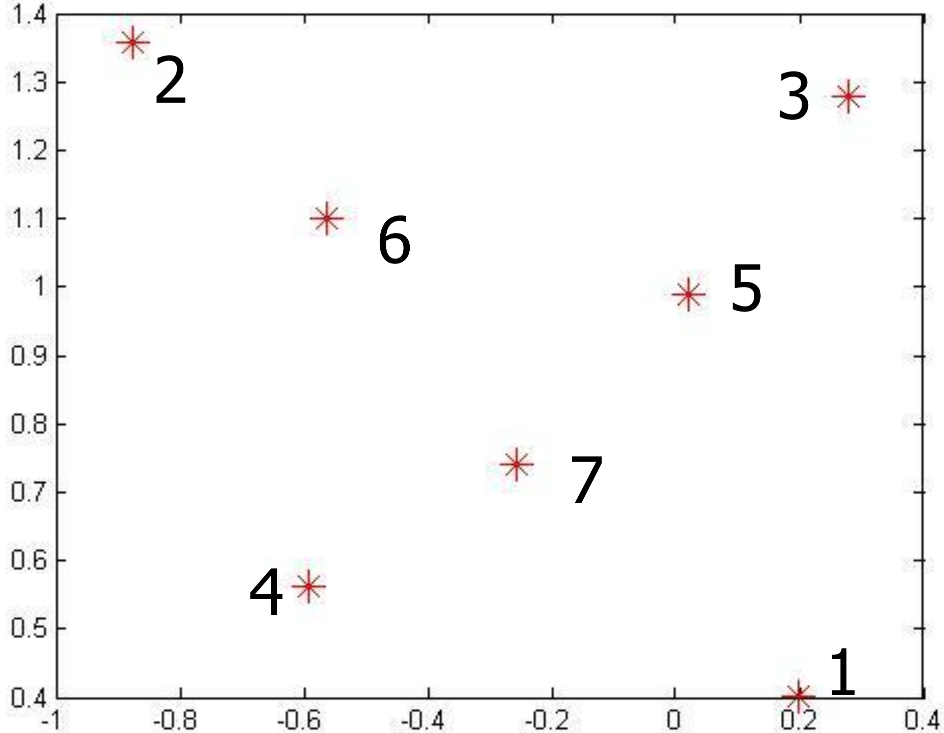
Proof:

$$\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{K}} = (\Lambda^{1/2} u_i)^T \Lambda^{1/2} u_j = u_i^T \Lambda u_j = G_{ij}$$

For **general** $\mathcal{X}$ sets

the necessary and sufficient conditions of $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to be a kernel are given by the Mercer's theorem. (See later)

Finite example



Choose 7 2D points

Choose a kernel k

$$G_{ij} = \exp(-|x_i - x_j|^2/10) \text{ can be calculated.}$$

G =

1.0000	0.8131	0.9254	0.9369	0.9630	0.8987	0.9683
0.8131	1.0000	0.8745	0.9312	0.9102	0.9837	0.9264
0.9254	0.8745	1.0000	0.8806	0.9851	0.9286	0.9440
0.9369	0.9312	0.8806	1.0000	0.9457	0.9714	0.9857
0.9630	0.9102	0.9851	0.9457	1.0000	0.9653	0.9862
0.8987	0.9837	0.9286	0.9714	0.9653	1.0000	0.9779
0.9683	0.9264	0.9440	0.9857	0.9862	0.9779	1.0000

[U,D]=svd(G), $UDU^T=G$, $UU^T=I$

U =

-0.3709	0.5499	0.3392	0.6302	0.0992	-0.1844	-0.0633
-0.3670	-0.6596	-0.1679	0.5164	0.1935	0.2972	0.0985
-0.3727	0.3007	-0.6704	-0.2199	0.4635	-0.1529	0.1862
-0.3792	-0.1411	0.5603	-0.4709	0.4938	0.1029	-0.2148
-0.3851	0.2036	-0.2248	-0.1177	-0.4363	0.5162	-0.5377
-0.3834	-0.3259	-0.0477	-0.0971	-0.3677	-0.7421	-0.2217
-0.3870	0.0673	0.2016	-0.2071	-0.4104	0.1628	0.7531

D =

6.6315	0	0	0	0	0	0
0	0.2331	0	0	0	0	0
0	0	0.1272	0	0	0	0
0	0	0	0.0066	0	0	0
0	0	0	0	0.0016	0	0
0	0	0	0	0	0.000	0
0	0	0	0	0	0	0.000

Mapped points=sqrt(D)*U^T

Mapped points =

-0.9551	-0.9451	-0.9597	-0.9765	-0.9917	-0.9872	-0.9966
0.2655	-0.3184	0.1452	-0.0681	0.0983	-0.1573	0.0325
0.1210	-0.0599	-0.2391	0.1998	-0.0802	-0.0170	0.0719
0.0511	0.0419	-0.0178	-0.0382	-0.0095	-0.0079	-0.0168
0.0040	0.0077	0.0185	0.0197	-0.0174	-0.0146	-0.0163
-0.0011	0.0018	-0.0009	0.0006	0.0032	-0.0045	0.0010
-0.0002	0.0004	0.0007	-0.0008	-0.0020	-0.0008	0.0028
$\phi(x_1)$	$\phi(x_2)$	$\phi(x_3)$	$\phi(x_4)$	$\phi(x_5)$	$\phi(x_6)$	$\phi(x_7)$

You can check now that

$$\langle\phi(x_i),\phi(x_j)\rangle \doteq \phi(x_i)^T\phi(x_j) = \exp(-|x_i - x_j|^2/10)\ \forall i,j$$

Roadmap I

We need feature maps

Explicit (feature maps)

$$\phi(\vec{x}) = [\vec{x}_1, \vec{x}_1\vec{x}_2^2, \vec{x}_1 - \vec{x}_2, \ldots]$$

Implicit (kernel functions)

$$k(\vec{x}, \vec{y}) = \exp(-\|\vec{x} - \vec{y}\|^2)$$

Several algorithms **need the inner products** of features only!

It is much **easier to use implicit** feature maps (kernels)

Given a function $k(\vec{x}, \vec{y}) = -\|\vec{x}\|^{42}\|\vec{y}\|^{42} + \pi$

Is it a kernel function???

Roadmap II

Given a function $k(x,\tilde{x}) = -\|\mathbf{x}\|^{42}\|\tilde{x}\|^{42} + \pi$

Is it a kernel function???

Finite $\mathcal{X}$

- SVD,
- eigenvectors, eigenvalues
- Positive semi def. matrices
- Finite dim feature space

Arbitrary $\mathcal{X}$

We have to think about the test data as well...

- Mercer's theorem,
- eigenfunctions, eigenvalues
- Positive semi def. integral operators
- Infinite dim feature space (l_2)

If the kernel is pos. semi def. $\Leftrightarrow$ feature map construction

Mercer's theorem

$$
(*) \quad \begin{cases} k(\cdot,\cdot) \in L_2(\mathcal{X} \times \mathcal{X}), \\ k \text{ is symmetric: } k(x,\tilde{x}) = k(\tilde{x},x) \\ (T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot,x) f(x) dx \text{ operator is pos. semi definit} \\ \psi_i,\ i = 1,2,\ldots \text{ are the eigenfunctions of } T_k \\ \text{with eigenvalues } \lambda_i \end{cases}
$$

$$
\Rightarrow \begin{cases} (\lambda_1, \lambda_2, \ldots) \in l_1, \quad \lambda_i \geq 0 \ \forall i \\ \psi_i \in L_\infty(\mathcal{X}), \quad \forall i = 1,2,\ldots \\ k(x,\tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\tilde{x}) \quad \forall x, \tilde{x} \end{cases}
$$

2 variables

1 variable

Mercer's theorem

We like the Mercer's theorem becuase of the **expansion**:

$$k(x,\tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\tilde{x}) \quad \forall x, \tilde{x}$$

It shows the **existence of the feature map** $\phi : \mathcal{X} \to \mathcal{K} \subset l_2$

Let $\mathcal{K} \doteq l_2$,
and let $\phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \dots)^T$

$$\begin{aligned}
&\Rightarrow \langle \phi(x), \phi(\tilde{x}) \rangle_{l_2} \\
&= (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \dots)^T (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \dots) \\
&= \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\tilde{x}) = k(x, \tilde{x})
\end{aligned}$$

…☺

$\psi(x) = (\psi_1(x), \psi_2(x), \dots)$ is known as **Mercer map**

Roadmap III

We want to know which functions are kernels

- How to make new kernels from old kernels?
- The polynomial kernel: $k(u,v) \doteq (\langle u,v\rangle_{\mathcal{X}})^p$

For a given kernel $k(\cdot,\cdot)$ we already know how to define feature space $\mathcal{K}$, and $\phi : \mathcal{X} \to \mathcal{K}$ feature map (Mercer map):

$$\mathcal{K} = l_2, \text{ and } \phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T$$

We will show another way using RKHS:

$\mathcal{K} = \mathcal{F}$, and $\phi(x) \doteq k(x,\cdot) \in \mathcal{F}$

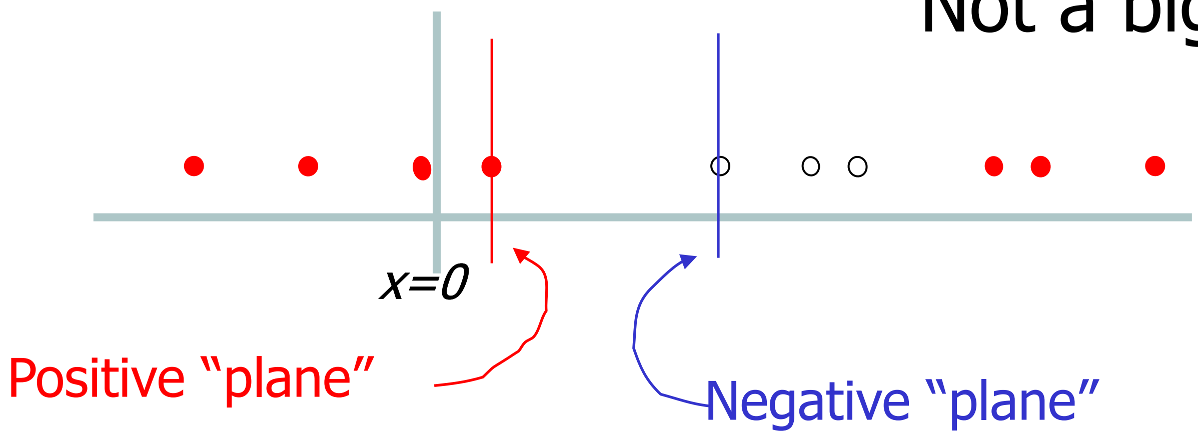
Inner product=???

Ready for the details?
;)

Hard 1-dimensional Dataset

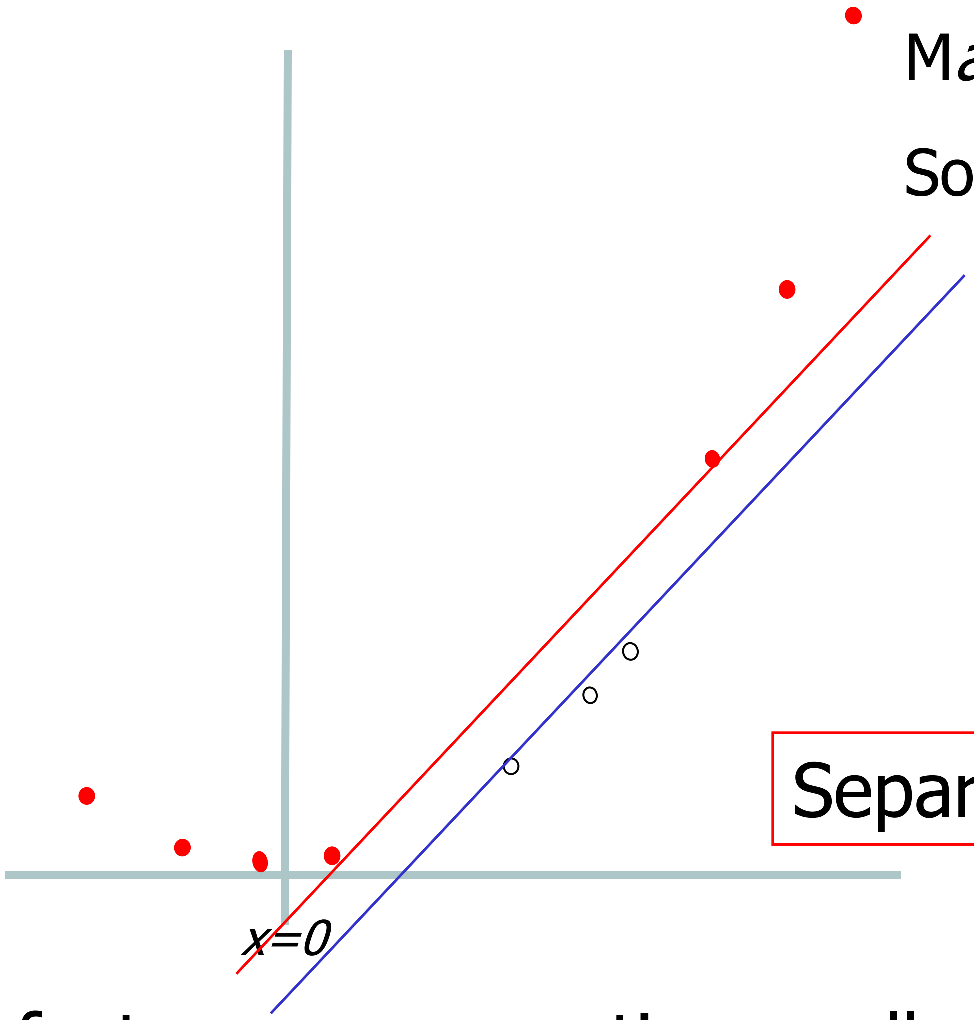
What would SVMs do with this data?

Not a big surprise



Doesn't look like slack variables will save us this time…

Hard 1-dimensional Dataset



M*ake up a new feature!*

Sort of…

… computed from original feature(s)

$$\mathbf{z}_k = (x_k, x_k^2)$$

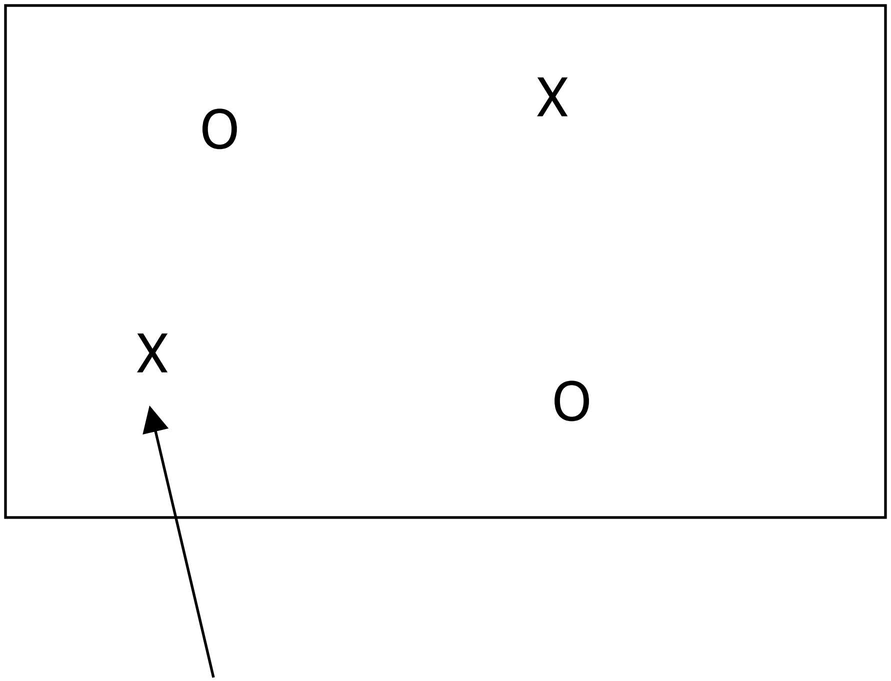
Separable! MAGIC!

New features are sometimes called *basis functions.*

Now drop this "augmented" data into our linear SVM.

taken from Andrew W. Moore

Hard 2-dimensional Dataset



Let us map this point to the 3rd dimension...

Kernels and Linear Classifiers

Let $\vec{x} = [\vec{x}_1, \vec{x}_2] \in \mathbb{R}^2$ be a vectorial represenation
of object $x \in \mathcal{X}$

Let $\phi : \mathcal{X} \to \mathcal{K} \subset \mathbb{R}^3$ feature map be given by

$$\phi(\vec{x}) \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T \in \mathcal{K} \subset \mathbb{R}^3$$

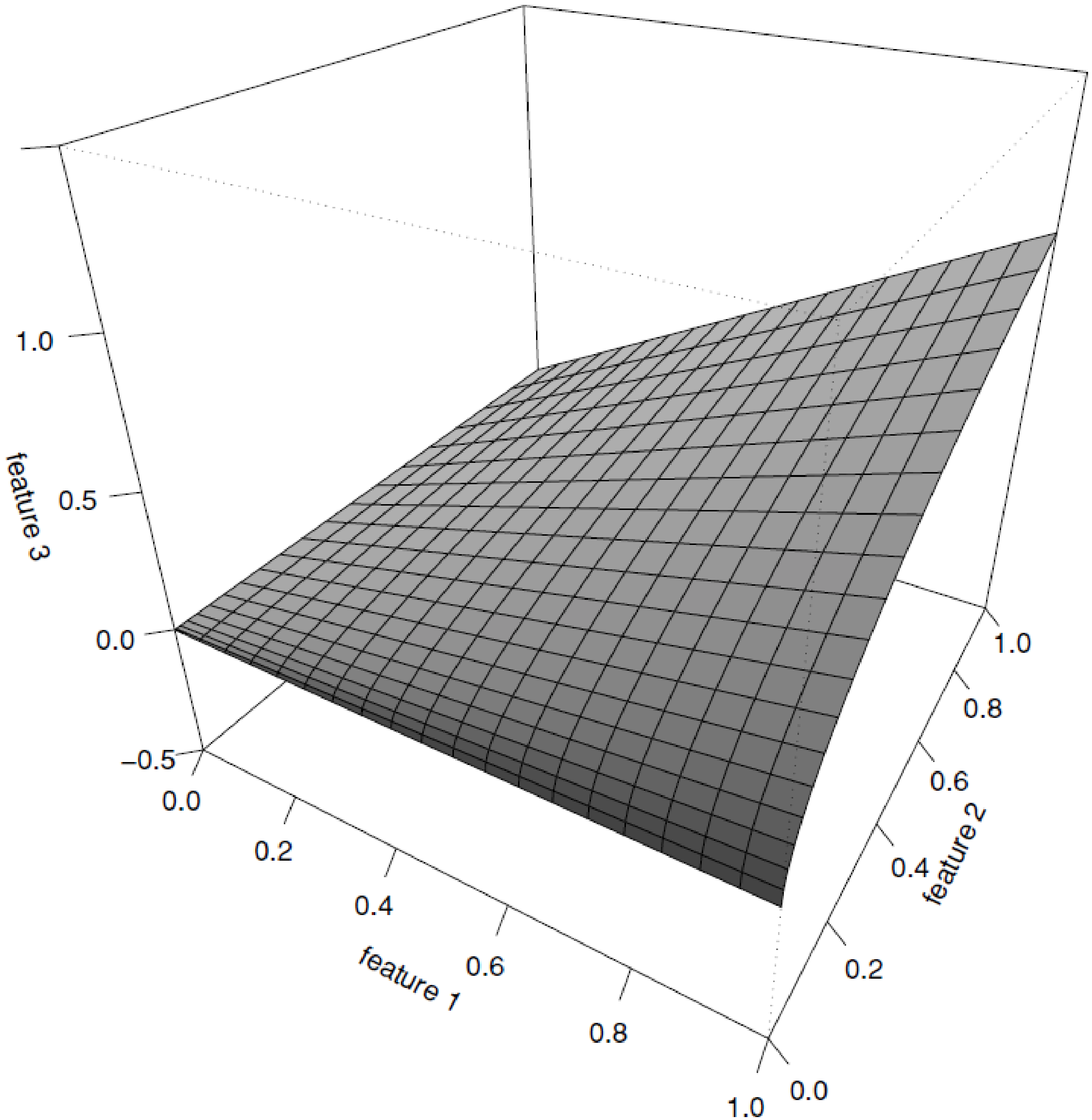
Def. Feature space: $\mathcal{K}$

We will use linear classifiers in this feature space.

In the original space $\mathbb{R}^2$ for a given $\mathbf{w} \in \mathbb{R}^3$ the decision surface is:

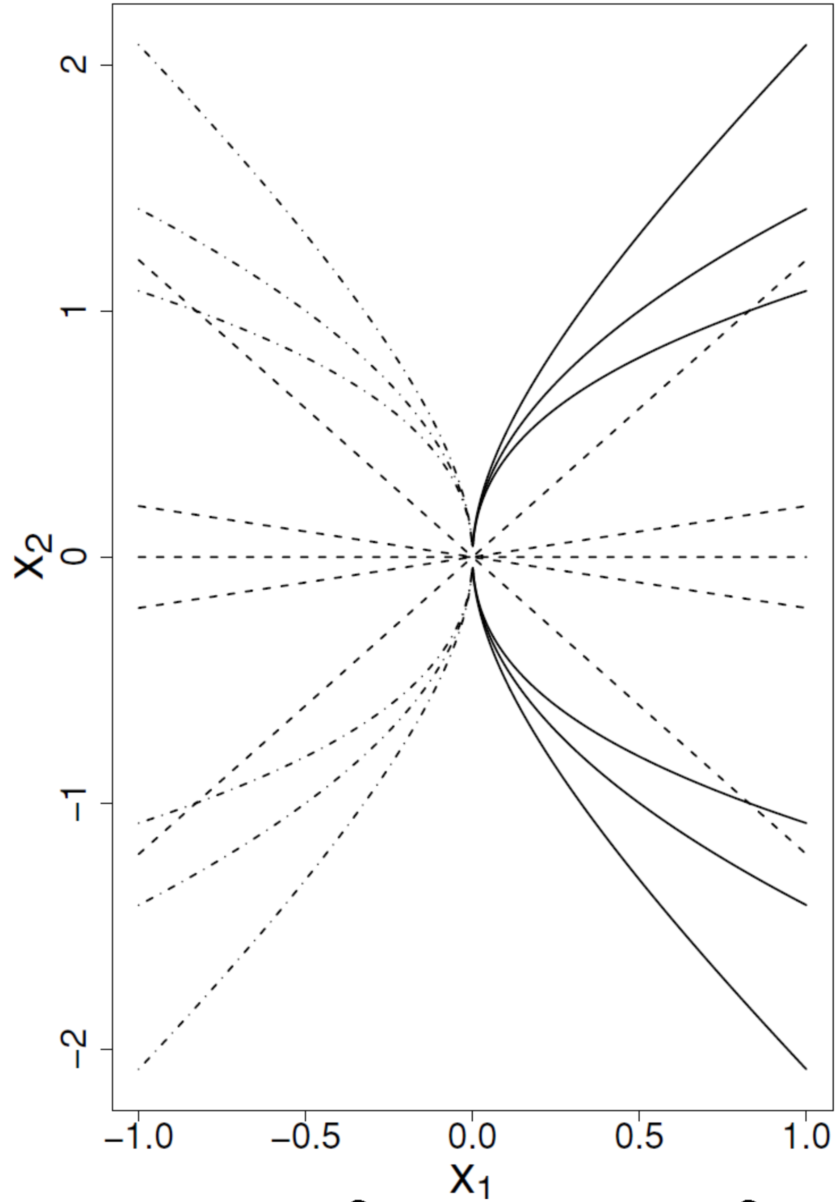
$$\boxed{\tilde{X}_0(\mathbf{w}) = \{\vec{x} \in \mathbb{R}^2 \mid w_1\vec{x}_1 + w_2\vec{x}_2^2 + w_3\vec{x}_1\vec{x}_2 = 0\}}$$

- This is nonlinear in $\vec{x} \in \mathbb{R}^2$
- This is linear in the feature space $\phi(\vec{x}) \in \mathcal{K} \subset \mathbb{R}^3$



$$\phi(\vec{x}) \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T \in \mathcal{K} \subset \mathbb{R}^3 \text{ feature map}$$

Picture is taken from R. Herbrich



The $\tilde{X}_0(\mathbf{w}) = \{\vec{x} \in \mathbb{R}^2 \mid w_1\vec{x}_1 + w_2\vec{x}_2^2 + w_3\vec{x}_1\vec{x}_2 = 0\}$ decision surface for different fixed $\mathbf{w}$ vectors.

Picture is taken from R. Herbrich

Kernels and Linear Classifiers

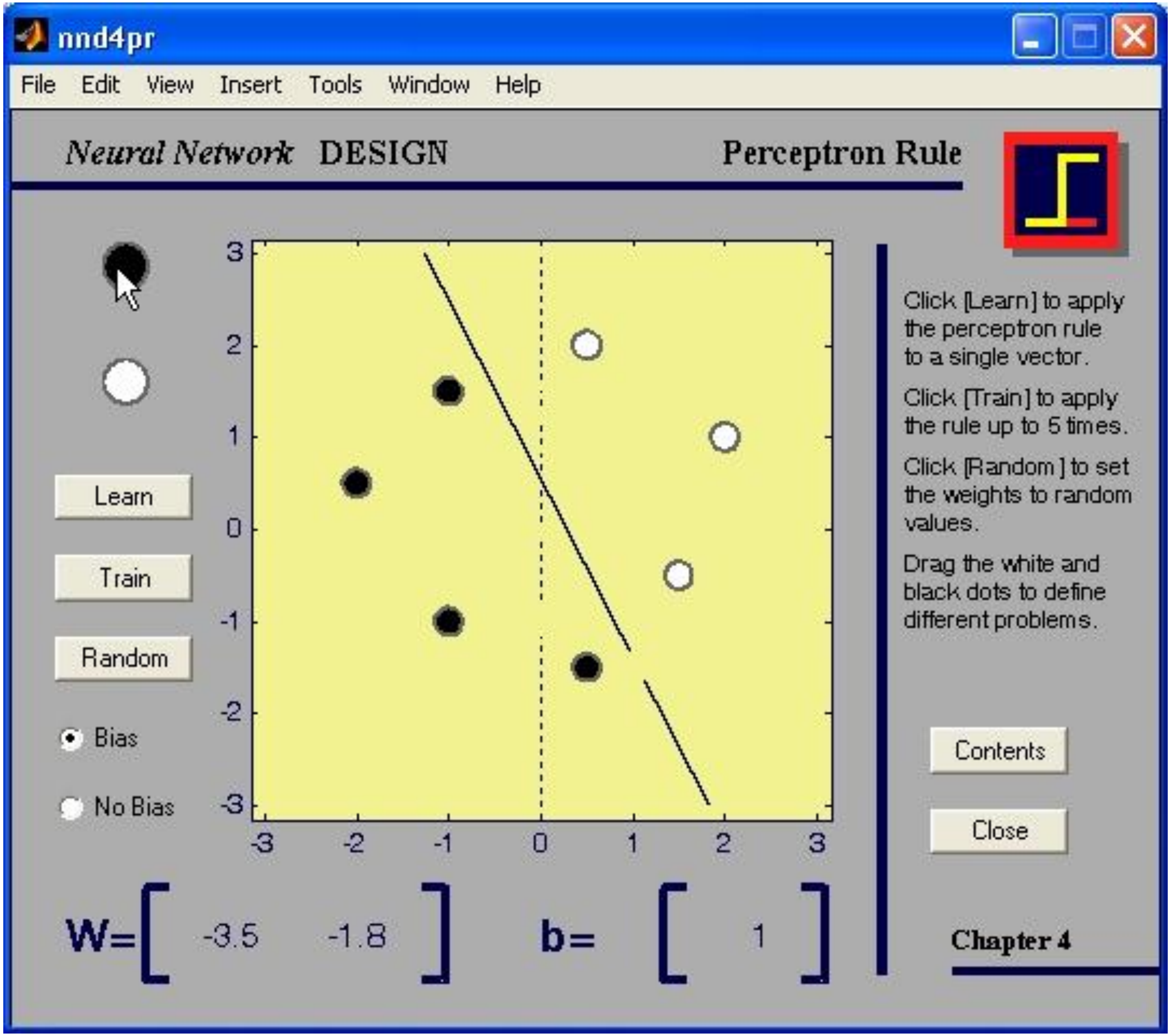
$$\phi(\vec{x}) \doteq [\phi_1(\vec{x}), \phi_2(\vec{x}), \phi_3(\vec{x})] \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T$$

Feature functions

- We seek for a small set of basis vectors $\{\phi_i\}$ which allows perfect discrimination between the classes in $\mathcal{X}$ (**Feature selection**)

- If we have too many features $\Rightarrow$ overfitting can happen.

Back to the Perceptron Example



The Perceptron

- **The primal algorithm in the feature space**

$D = \{(x_i, y_i), i = 1, \ldots, m\}$ training data set.

$\mathbf{x}_i = \phi(x_i) \in \mathcal{K} \subset \mathbb{R}^n$ feature map.

1., $\mathbf{w} = 0 \in \mathbb{R}^n$

2., $\forall\ (x_i, y_i),\ i = 1, \ldots, m$, evaluate $sign(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle)$

3., If x_i is misclassified $(sign(y_i \langle \mathbf{x}_i, \mathbf{w} \rangle) < 0)$
then $\mathbf{w} := \mathbf{w} + y_i \mathbf{x}_i$

4., If no mistakes occur $\Rightarrow$ STOP

The primal algorithm in the feature space

Algorithm 1 Perceptron learning algorithm (in primal variables).

Require: A feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^n$

Ensure: A linearly separable training sample $\boldsymbol{z} = ((x_1, y_1), \ldots, (x_m, y_m))$

$\mathbf{w}_0 = \mathbf{0}$; $t = 0$
repeat
  for $j = 1, \ldots, m$ **do**
    if $y_j \langle \boldsymbol{\phi}(x_j), \mathbf{w} \rangle \leq 0$ **then**
      $\mathbf{w}_{t+1} = \mathbf{w}_t + y_j \boldsymbol{\phi}(x_j)$
      $t \leftarrow t + 1$
    end if
  end for
until no mistakes have been made within the **for** loop
return the final weight vector $\mathbf{w}_t$

If x_j is misclassified

Picture is taken from R. Herbrich

The Perceptron

We start at $\mathbf{w}_0 = 0 \in \mathcal{K} \subset \mathbb{R}^n$

$m=$ num of training examples,
$n = dim(\mathcal{K})$,

$t=$ num of mistakes so far

$\Rightarrow \mathbf{w}_t = \sum_{i=1}^{m} \alpha_i \phi(x_i) = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \in \mathbb{R}^n$ at time step t

Thus instead of tuning n variables
$\mathbf{w} = (w_1, \ldots, w_n)$ (**Primal variables**)
in the large n-dimensional feautre space $\mathcal{K}$, it is
enough to learn $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ values (**Dual variables**).

The Perceptron

The Dual Algorithm in the feature space

$D = \{(x_i, y_i), i = 1, \ldots, m\}$ training data set.

$\mathbf{x}_i = \phi(x_i) \in \mathcal{K} \subset \mathbb{R}^n$ feaure map, $i = 1, \ldots, m$

$t=$ num of mistakes so far

$\Rightarrow \mathbf{w}_t = \sum_{i=1}^{m} \alpha_i \phi(x_i) = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \in \mathbb{R}^n$ at time step t

We update $\boldsymbol{\alpha}_t \in \mathbb{R}^m$ whenever a mistake occurs

1., $\boldsymbol{\alpha}_0 = 0 \in \mathbb{R}^m$

2., $\forall j = 1, \ldots, m$ evaluate

$$y_j \langle \mathbf{x}_j, \mathbf{w}_t \rangle = y_j \langle \mathbf{x}_j, \sum_{i=1}^{m} \alpha_i \mathbf{x}_i \rangle = y_j \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}_j, \mathbf{x}_i \rangle$$

3., If x_j is misclassified $(y_j \langle \mathbf{x}_j, \mathbf{w}_t \rangle < 0)$ then update $\boldsymbol{\alpha}_t \in \mathcal{K}$

4., If no mistakes occur $\Rightarrow$ STOP

The Dual Algorithm in the feature space

Algorithm 2 Perceptron learning algorithm (in dual variables).

Require: A feature mapping $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{K} \subseteq \ell_2^n$

Ensure: A linearly separable training sample $\boldsymbol{z} = ((x_1, y_1), \ldots, (x_m, y_m))$

$\boldsymbol{\alpha} = \mathbf{0}$

repeat

for $j = 1, \ldots, m$ **do**

if $y_j \sum_{i=1}^{m} \alpha_i \left\langle \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j) \right\rangle \leq 0$ **then**

If x_j is misclassified

$\alpha_j \leftarrow \alpha_j + y_j$

end if

end for

until no mistakes have been made within the **for** loop

return the vector $\boldsymbol{\alpha}$ of expansion coefficients

Picture is taken from R. Herbrich

The Dual Algorithm in the feature space

For the classification of a new object (x, y)
we have to evaluate

$$y \sum_{i=1}^{m} \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle$$

We don't have to know the actual values of $\mathbf{x} = \phi(x)$!

It is enough to know the inner products

$$\langle \mathbf{x}, \mathbf{x}_i \rangle \quad \forall i = 1, \ldots, m$$

between the object and the training points

Kernels

Definition: (kernel)

We are given $\phi : \mathcal{X} \to \mathcal{K} \subset l_2^n$ feautre mapping.

The **kernel** $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the corresponding inner product function:

$$\boxed{k(x_i, x_j) \doteq \langle \underbrace{\phi(x_i)}_{\mathbf{x}_i}, \underbrace{\phi(x_j)}_{\mathbf{x}_j} \rangle_{\mathcal{K}} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{K}}}$$

Kernels

Definition: **(Gram matrix, kernel matrix)**

Gram matrix $G \in \mathbb{R}^{m \times m}$ of kernel k at $\{x_1, \ldots, x_m\}$:

$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a training set } \{x_1, \ldots, x_m\}\end{array}\right\} \Rightarrow G_{ij} \doteq k(x_i, x_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

Definition: **(Feature space, kernel space)**

$$\mathcal{K} \doteq span\{\phi(x) \mid x \in \mathcal{X}\} \subset \mathbb{R}^n$$

Kernel technique

Definition:

Matrix $G \in \mathbb{R}^{m\times m}$ is positive semidefinite (PSD)
$\Leftrightarrow G$ is symmetric, and $0 \leq \boldsymbol{\beta}^T G \boldsymbol{\beta} \;\forall \boldsymbol{\beta} \in \mathbb{R}^{m\times m}$

$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a training set } \{x_1, \ldots, x_m\}\end{array}\right\} \Rightarrow G_{ij} \doteq k(x_i, x_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_{\mathcal{K}}$$

Lemma:

The Gram matrix is symmetric, PSD matrix.

Proof:

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{n\times m} \Rightarrow G = \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{m\times m}$$

$$0 \leq \langle \mathbf{X}\boldsymbol{\beta}, \mathbf{X}\boldsymbol{\beta} \rangle_{\mathcal{K}} = \boldsymbol{\beta}^T G \boldsymbol{\beta}$$

Kernel technique

We already know that several algorithms use the **kernel values** only (...and NOT the **feature values**)!

Key idea:

Choose a nice kernel function k
rather than an ugly feautre mapping
$\phi : \mathcal{X} \to \mathbb{R}^n$

Kernel technique

We have seen so far how to build a kernel $k(\cdot,\cdot)$ from a given feature map $\phi : \mathcal{X} \to \mathbb{R}^n$

Now we want to do the opposite:

A function $k(\cdot,\cdot)$ is kernel $\Leftrightarrow$ there exists a feature space $\mathcal{K}$ and feature map $\phi : \mathcal{X} \to \mathcal{K}$, such that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{K}}$

Let us try to find ϕ and $\mathcal{K}$!

Finite example

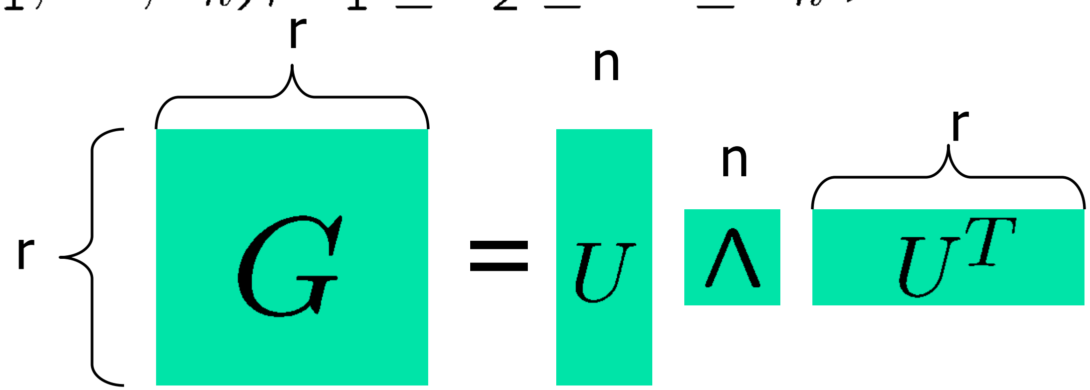
$$\left.\begin{array}{l}\text{Given a kernel } k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ \text{and a FINITE set } \mathcal{X} = \{x_1, \ldots, x_r\}\end{array}\right\} \Rightarrow \text{construct } \mathcal{K} \text{ and } \phi$$

$\Rightarrow G \in \mathbb{R}^{r \times r}$, $G_{ij} = k(x_i, x_j)$ can be calculated

G is symmetric, PSD $\Rightarrow G = U \Lambda U^T$ by SVD.

$$U^T U = I_n,\ n = rank(U),\ U = \begin{bmatrix} u_1^T \\ \vdots \\ u_r^T \end{bmatrix} \in \mathbb{R}^{r \times n}$$

$$\Lambda = diag(\lambda_1, \ldots, \lambda_n),\ \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$$



Finite example

Lemma:

Let $\mathcal{K} = span\{\phi(x_1), \ldots \phi(x_r)\}$

$$\Rightarrow \phi(x_i) \doteq \Lambda^{1/2} u_i \in \mathbb{R}^n,\ i = 1, \ldots, r$$

leads back to the Gram matrix G

Proof:

$$\langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{K}} = (\Lambda^{1/2} u_i)^T \Lambda^{1/2} u_j = u_i^T \Lambda u_j = G_{ij}$$

For **general** $\mathcal{X}$ sets

the necessary and sufficient conditions of $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to be a kernel are given by the Mercer's theorem. (See later)

Kernel technique, Finite example

We have seen:

If $\mathcal{X} = \{x_1, \ldots, x_r\}$ and
Gram matrix G is a symmetric, PSD matrix

$\Rightarrow$ we can construct feature space $\mathcal{K}$,
and feature map $\phi : \mathcal{X} \to \mathcal{K}$, compatible with G

Lemma:

These conditions are necessary

Kernel technique, Finite example

Proof: ... wrong in the Herbrich's book...

If $\exists \lambda_n < 0 \Rightarrow \exists v \in \mathbb{R}^r$ eigenvector s.t. $Gv = \lambda_n v$

$\Rightarrow v^T G v = v^T \lambda_n v = \lambda_n \|v\|^2 < 0$

G is a Gram matrix $\Rightarrow \exists \phi : \mathcal{X} \to \mathcal{K}$, s.t. $G_{ij} = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{K}}$

Consider the $w \doteq [\phi(x_1), \ldots \phi(x_r)]v \in \mathcal{K}$ vector.

$$\Rightarrow \|w\|_{\mathcal{K}}^2 = \langle w, w \rangle_{\mathcal{K}}$$
$$= \langle [\phi(x_1), \ldots \phi(x_r)]v, [\phi(x_1), \ldots \phi(x_r)]v \rangle_{\mathcal{K}} = v^T G v < 0$$

$\lightning$

Kernel technique, Finite example

Summary:

Given a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$,
and a FINITE set $\mathcal{X} = \{x_1, \ldots, x_r\}$

$k(\cdot, \cdot)$ is kernel $\Leftrightarrow G = \{k(x_i, x_j)\}_{ij}$ gram matrix is symmetric, PSD.

How to generalize this to general sets???

Integral operators, eigenfunctions

Instead of studying the $Gv = \lambda v \; G \in \mathbb{R}^{r \times r}$ problem, we examine its generalization:

num of objects r is countably infinite or continuum, and $\mathcal{X} = \{x | x \in \mathcal{X}\}$ is arbitrary.

Definition: Integral operator with kernel k(.,.)

$$(T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot, x) f(x) dx$$

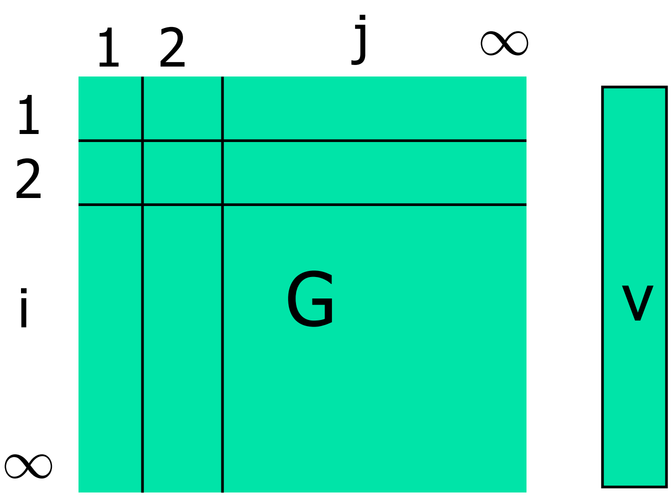
Remark:

$(T_G v)(i) \doteq (Gv)(i) \; i = 1, \ldots, r$ is a special case of this, when the integral is replaced by a finite sum.

From Vector domain to Functions

- Observe that each vector v = (v[1], v[2], …, v[n])
  is a mapping from the integers {1,2,…, n} to $\Re$

- We can generalize this easily to **INFINITE** domain
  w = (w[1], w[2], …, w[n], …)
  where w is mapping from {1,2,…} to $\Re$



$$(T_G v)(i) \doteq (Gv)(i) = \underbrace{\sum_{j=1}^{\infty}}_{\int_{\mathcal{X}}} \underbrace{G_{ij}}_{k(i,j)} \underbrace{v_j}_{f(j)}$$

From Vector domain to Functions

From integers we can further extend to

- $\Re$ or
- $\Re^m$
- Strings
- Graphs
- Sets
- Whatever
- …

L_p and l_p spaces

Definition A.33 (Normed space) *Suppose $\mathcal{X}$ is a vector space. A* normed space *$\mathcal{X}$ is defined by the tuple $(\mathcal{X}, \|\cdot\|)$ where $\|\cdot\| : \mathcal{X} \to \mathbb{R}^+$ is called a* norm, *i.e., for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and $c \in \mathbb{R}$,*

$$
\begin{aligned}
\|\mathbf{x}\| &\geq 0 \textit{ and } \|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}\,, \\
\|c\mathbf{x}\| &= |c| \cdot \|\mathbf{x}\|\,, \\
\|\mathbf{x} + \mathbf{y}\| &\leq \|\mathbf{x}\| + \|\mathbf{y}\|\,. \qquad \text{(A.18)}
\end{aligned}
$$

This clearly induces a metric ρ on $\mathcal{X}$ by $\rho(\mathbf{x}, \mathbf{y}) \doteq \|\mathbf{x} - \mathbf{y}\|$. Note that equation (A.18) is known as the triangle inequality.

Definition A.34 (ℓ_p^n and L_p) *Given a subset $X \subseteq \mathcal{X}$, the space $L_p(X)$ is the space of all functions $f : X \to \mathbb{R}$ such that*

$$
\begin{aligned}
\int_X |f(\mathbf{x})|^p \, d\mathbf{x} < \infty & \qquad \textit{if} \quad p < \infty\,, \\
\sup_{\mathbf{x} \in X} |f(\mathbf{x})| < \infty & \qquad \textit{if} \quad p = \infty\,.
\end{aligned}
$$

Picture is taken from R. Herbrich

L_p and l_p spaces

Endowing this space with the norm

$$\|f\|_p \stackrel{\text{def}}{=} \begin{cases} \left(\int_X |f(\mathbf{x})|^p \, d\mathbf{x}\right)^{\frac{1}{p}} & \text{if } p < \infty \\ \sup_{\mathbf{x}\in X} |f(\mathbf{x})| & \text{if } p = \infty \end{cases}$$

makes $L_p(X)$ a normed space (by Minkowski's inequality). The space ℓ_p^n of sequences of length n is defined by

$$\ell_p^n \stackrel{\text{def}}{=} \left\{ (x_1, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} \sum_{i=1}^n |x_i|^p < \infty & \text{if } 0 < p < \infty \\ \max_{i=1,\ldots,n} |x_i| & \text{if } p = \infty \end{array} \right\}.$$

Definition A.35 (ℓ_p–norms) *Given $\mathbf{x} \in \ell_p^n$ we define the ℓ_p*–norm $\|\mathbf{x}\|_p$ *by*

$$\|\mathbf{x}\|_p \stackrel{\text{def}}{=} \begin{cases} \sum_{i=1}^n \mathbf{I}_{x_i \neq 0} & \text{if } p = 0 \\ \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} & \text{if } 0 < p < \infty \\ \max_{i=1,\ldots,n} |x_i| & \text{if } p = \infty \end{cases}.$$

Picture is taken from R. Herbrich

L_2 and l_2 special cases

Example A.39 (ℓ_2^n **and** L_2) *Defining an inner product* $\langle \cdot, \cdot \rangle$ *in* ℓ_2^n *and* $L_2(X)$ *by (A.23) and*

$$\langle f, g \rangle = \int_X f(\mathbf{x})\, g(\mathbf{x})\, d\mathbf{x} \tag{A.24}$$

makes these two spaces inner product spaces

Picture is taken from R. Herbrich

Kernels

We don't need the $\mathcal{K} \subset l_2^n$ assumption. It is enough if $\mathcal{K}$ is a complete inner product (Hilbert) space.

Definition: inner product, Hilbert spaces

$\langle \cdot, \cdot \rangle : \mathcal{K} \times \mathcal{K} \to \mathbb{R}$ is an inner product in vector space $\mathcal{K}$, iff for all vectors $x, y, z \in \mathcal{K}$ and all scalars $a \in \mathbb{R}$:

* Symmetry: $\langle x, y \rangle = \langle y, x \rangle$.

* Linearity in the first argument:
$\langle ax, y \rangle = a\langle x, y \rangle$, $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$.

* Positive-definite: $\langle x, x \rangle \geq 0$ with equality only for $x = 0$.

This is more general than the inner product in $\mathbb{R}^n = l_2^n$

Examples:

- space of square integrable functions $L_2(\mathcal{X})$,
- space of square summable infinite series l_2

Integral operators, eigenfunctions

Definition: Eigenvalue, Eigenfunction

- λ is the eigenvalue,
- $\Psi \in L_2(\mathcal{X})$ is the eigenfunction

  of integral opreator $(T_k f)(\cdot) = \int\limits_{\mathcal{X}} k(\cdot, x) f(x) dx$

$$\Leftrightarrow \begin{cases} \int\limits_{\mathcal{X}} k(x, \bar{x}) \psi(\bar{x}) d\bar{x} = \lambda \psi(x) \quad \forall x \in \mathcal{X} \\ \\ \|\psi\|^2_{L_2} \doteq \int\limits_{\mathcal{X}} \psi^2(x) dx = 1 \end{cases}$$

The previous $Gv = \lambda v$ is a special case of this, when $\mathcal{X} = \{x_1, \ldots, x_r\}$ is a finite set.

Positive (semi) definite operators

Definition: Positive Definite Operator

$k(\cdot,\cdot)$ is symmetric kernel,

$$\Rightarrow (T_k f)(\cdot) \doteq \int_{\mathcal{X}} k(\cdot, x) f(x) dx$$

$T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ operator is positive semi definit

$$\Leftrightarrow \int_{\mathcal{X}} \int_{\mathcal{X}} k(\tilde{x}, x) f(x) f(\tilde{x}) dx d\tilde{x} \geq 0 \quad \forall f \in L_2(\mathcal{X})$$

The previous $v^T G v \geq 0$ is a special case of this, when $\mathcal{X} = \{x_1, \ldots, x_r\}$ is a finite set.

Mercer's theorem

$$(*)\quad \begin{cases} k(\cdot,\cdot) \in L_2(\mathcal{X}\times\mathcal{X}), \\[1em] k \text{ is symmetric: } k(x,\tilde{x}) = k(\tilde{x},x) \\[1em] (T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot,x) f(x)dx \text{ operator is pos. semi definit} \\[1em] \psi_i,\ i = 1,2,\ldots \text{ are the eigenfunctions of } T_k \\ \text{with eigenvalues } \lambda_i \end{cases}$$

$$\Rightarrow \begin{cases} (\lambda_1,\lambda_2,\ldots) \in l_1, \quad \lambda_i \geq 0\ \forall i \\[1em] \psi_i \in L_\infty(\mathcal{X}), \quad \forall i = 1,2,\ldots \\[1em] k(x,\tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(\tilde{x}) \quad \forall x,\tilde{x} \end{cases}$$

2 variables

1 variable

Mercer's theorem

We like the Mercer's theorem becuase of the **expansion**:

$$k(x,\tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(\tilde{x}) \quad \forall x, \tilde{x}$$

It shows the **existence of the feature map** $\phi : \mathcal{X} \to \mathcal{K} \subset l_2$

Let $\mathcal{K} \doteq l_2$,
and let $\phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T$

$$\begin{aligned}
&\Rightarrow \langle \phi(x), \phi(\tilde{x}) \rangle_{l_2} \\
&= (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots) \\
&= \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(\tilde{x}) = k(x,\tilde{x})
\end{aligned}$$

…☺

$\psi(x) = (\psi_1(x), \psi_2(x), \ldots)$ is known as **Mercer map**

A nicer characterization

The (*) condition in the Mercer's theorem is a bit ugly, but we have a nicer form that characterizes when a function $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel

(i.e. scalar product in some inner product space)

Theorem: nicer kernel characterization

$k(\cdot,\cdot)$ is a (Mercer) kernel

$\Leftrightarrow$ $(T_k f)(\cdot)$ is a pos. semi definite operator

$\Leftrightarrow$ $G = (k(x_i, x_j))_{ij}^r \in \mathbb{R}^{r\times r}$ Gram matrix is pos. semi definite $\forall r$, $\forall (x_1, \ldots, x_r) \in \mathcal{X}^r$

Kernel Families

- Kernels have the intuitive meaning of similarity measure between objects.

- So far we have seen two ways for making a linear classifier nonlinear in the input space:

1. (explicit) Choosing a mapping ϕ
   $\Rightarrow$ Mercer kernel k

2. (implicit) Choosing a Mercer kernel k
   $\Rightarrow$ Mercer map ϕ

Designing new kernels from kernels

$k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are kernels $\Rightarrow$

1. $k(x, \tilde{x}) = k_1(x, \tilde{x}) + k_2(x, \tilde{x})$,
2. $k(x, \tilde{x}) = c \cdot k_1(x, \tilde{x}), \textit{for all } c \in \mathbb{R}^+$,
3. $k(x, \tilde{x}) = k_1(x, \tilde{x}) + c, \textit{for all } c \in \mathbb{R}^+$,
4. $k(x, \tilde{x}) = k_1(x, \tilde{x}) \cdot k_2(x, \tilde{x})$,
5. $k(x, \tilde{x}) = f(x) \cdot f(\tilde{x}), \textit{for any function } f : \mathcal{X} \to \mathbb{R}$

are also kernels.

Picture is taken from R. Herbrich

Designing new kernels from kernels

1. $k(x, \tilde{x}) = (k_1(x, \tilde{x}) + \theta_1)^{\theta_2}$, *for all* $\theta_1 \in \mathbb{R}^+$ *and* $\theta_2 \in \mathbb{N}$

2. $k(x, \tilde{x}) = \exp\left(\frac{k_1(x,\tilde{x})}{\sigma^2}\right)$, *for all* $\sigma \in \mathbb{R}^+$,

3. $k(x, \tilde{x}) = \exp\left(-\frac{k_1(x,x) - 2k_1(x,\tilde{x}) + k_1(\tilde{x},\tilde{x})}{2\sigma^2}\right)$, *for all* $\sigma \in \mathbb{R}^+$

4. $k(x, \tilde{x}) = \frac{k_1(x,\tilde{x})}{\sqrt{k_1(x,x) \cdot k_1(\tilde{x},\tilde{x})}}$

Picture is taken from R. Herbrich

Designing new kernels from kernels

The meaning of

$$k(x, \tilde{x}) = \frac{k_1(x, \tilde{x})}{\sqrt{k_1(x, x) k_1(\tilde{x}, \tilde{x})}}$$

is that we can normalize the data in the feature space without performing the explicit mapping.

Use the normailzed kernel k_{norm}:

$$k_{norm}(x, \tilde{x}) \doteq \frac{k(x, \tilde{x})}{\sqrt{k(x, x) k(\tilde{x}, \tilde{x})}} = \frac{\langle x, \tilde{x} \rangle}{\sqrt{\|x\|^2 \|\tilde{x}\|^2}} = \langle \frac{x}{\|x\|^2}, \frac{\tilde{x}}{\|\tilde{x}\|^2} \rangle$$

Kernels on inner product spaces

Note:

If $\mathcal{X}$ is a vector space with $\langle \cdot, \cdot \rangle_{\mathcal{X}}$ inner product
$\Rightarrow k(\cdot, \cdot) = \langle \cdot, \cdot \rangle_{\mathcal{X}}$ is a kernel function.

$$dim(\mathcal{X}) = N$$

Name	Kernel function	dim $(\mathcal{K})$
pth degree polynomial	$k(\vec{u}, \vec{v}) = (\langle \vec{u}, \vec{v} \rangle_{\mathcal{X}})^p$ $p \in \mathbb{N}^+$	$\binom{N+p-1}{p}$
complete polynomial	$k(\vec{u}, \vec{v}) = (\langle \vec{u}, \vec{v} \rangle_{\mathcal{X}} + c)^p$ $c \in \mathbb{R}^+, \ \ p \in \mathbb{N}^+$	$\binom{N+p}{p}$
RBF kernel	$k(\vec{u}, \vec{v}) = \exp\left(-\frac{\|\vec{u}-\vec{v}\|_{\mathcal{X}}^2}{2\sigma^2}\right)$ $\sigma \in \mathbb{R}^+$	∞
Mahalanobis kernel	$k(\vec{u}, \vec{v}) = \exp\left(-(\vec{u} - \vec{v})' \boldsymbol{\Sigma} (\vec{u} - \vec{v})\right)$ $\boldsymbol{\Sigma} = \mathrm{diag}\left(\sigma_1^{-2}, \ldots, \sigma_N^{-2}\right)$, $\sigma_1, \ldots, \sigma_N \in \mathbb{R}^+$	∞

Picture is taken from R. Herbrich

Common Kernels

- Polynomials of degree d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v}+1)^d$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v}+\nu)$$

- Gaussian kernels

$$K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{||\mathbf{u}-\mathbf{v}||^2}{2\sigma^2}\right)$$

Equivalent to $\phi(x)$ of infinite dimensionality!

The RBF kernel

Note:

The RBF kernel can be used as a density estimator over $\mathcal{X} \subset l_2^N = \mathbb{R}^N$

Proof:

Let $(x_1, \ldots, x_m) \in \mathbb{R}^{N \times m}$ m training examples.

Let $\sum_{i=1}^{m} \alpha_i = 1,\ \alpha_i \geq 0$

$$\Rightarrow f(x) \doteq \sum_{i=1}^{m} \alpha_i k(x, x_i) = \sum_{i=1}^{m} \alpha_i \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$$

(This puts a Gaussian on each x_i, Mixture of Gaussians)

The RBF kernel

Note:

The RBF kernel maps the input space $\mathcal{X}$ onto the surface of an infinite dimensional hypersphere.

Proof:

$$\|\phi(x)\| = \sqrt{k(x,x)} = \sqrt{\exp(0)} = 1$$

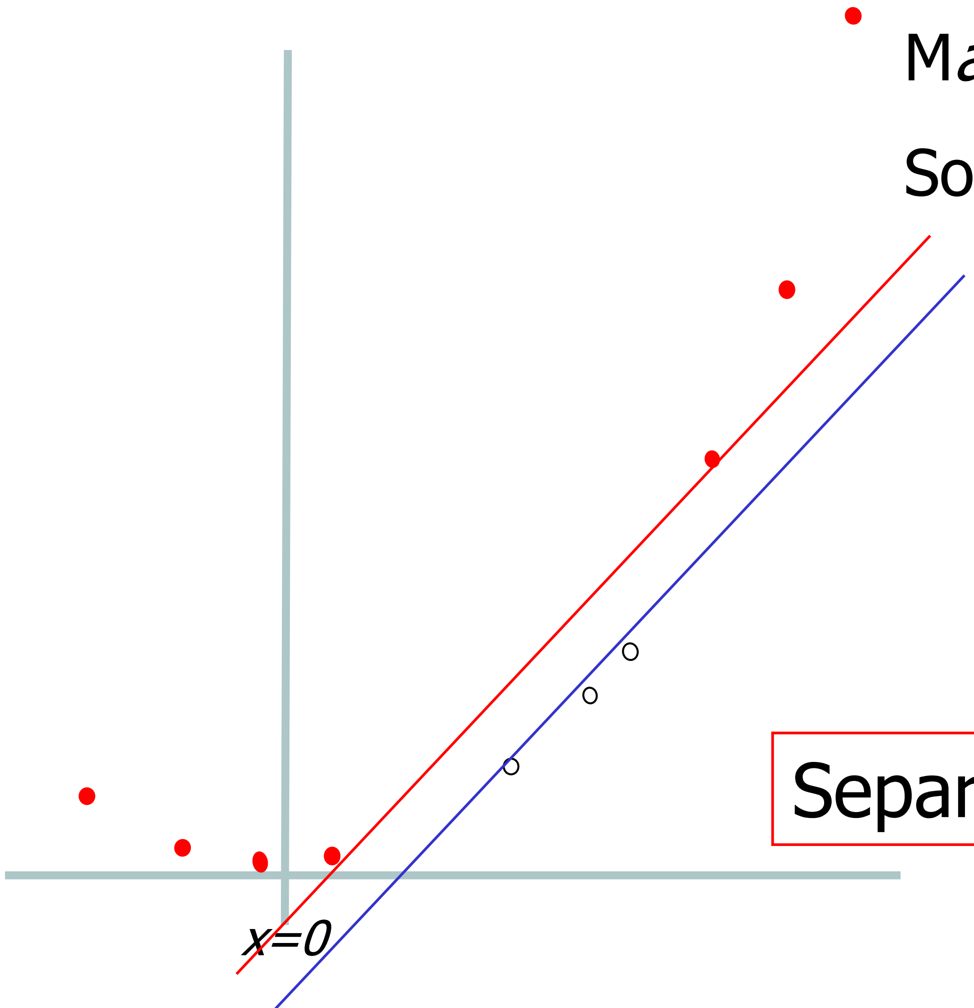
Note:

The RBF kernel is shift invariant:

$$k(u+a, v+a) = k(u,v), \ \forall a$$

The Polynomial kernel

Reminder: Hard 1-dimensional Dataset



M*ake up a new feature!*

Sort of…

… computed from original feature(s)

$$\boxed{\mathbf{z}_k = (x_k, x_k^2)}$$

Separable! MAGIC!

New features are sometimes called *basis functions.*

Now drop this "augmented" data into our linear SVM.

… New Features from Old …

- Here: mapped $\Re \to \Re^2$ by $\Phi: x \to [x, x^2]$
  - Found "extra dimensions" $\Rightarrow$ linearly separable!

- In general,
  - Start with vector $x \in \Re^N$
  - Want to add in x_1^2 , x_2^2, …
  - Probably want other terms – eg $x_2 \cdot x_7$, …
  - Which ones to include?

    Why not ALL OF THEM???

Special Case

- $x=(x_1, x_2, x_3) \rightarrow$
  $(1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)$

- $\Re^3 \rightarrow \Re^{10}$, N=3, n=10;

In general, the dimension of the quadratic map:

$$N \rightarrow 1 + N + N + \binom{N}{2} = \frac{(N+2)(N+1)}{2} \approx \frac{N^2}{2}$$

So we map from the N dimensional space $\mathcal{X}$
to an $\approx N^2/2$ dimensional feature space $\mathcal{K}$.

taken from Andrew W. Moore

Quadratic Basis Functions

Let $$\Phi(x) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_N \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_N^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_N \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_1x_N \\ \vdots \\ \sqrt{2}x_{N-1}x_N \end{pmatrix}$$

Constant Term

Linear Terms

Pure Quadratic Terms

Quadratic Cross-Terms

What about those $\sqrt{2}$??

… stay tuned

Quadratic Dot Products

$$
<\Phi(\mathbf{a}),\Phi(\mathbf{b})>=
\begin{pmatrix}
1\\
\sqrt{2}a_1\\
\sqrt{2}a_2\\
\vdots\\
\sqrt{2}a_N\\
a_1^2\\
a_2^2\\
\vdots\\
a_N^2\\
\sqrt{2}a_1a_2\\
\sqrt{2}a_1a_3\\
\vdots\\
\sqrt{2}a_1a_N\\
\sqrt{2}a_2a_3\\
\vdots\\
\sqrt{2}a_1a_N\\
\vdots\\
\sqrt{2}a_{N-1}aN
\end{pmatrix}
\bullet
\begin{pmatrix}
1\\
\sqrt{2}b_1\\
\sqrt{2}b_2\\
\vdots\\
\sqrt{2}b_N\\
b_1^2\\
b_2^2\\
\vdots\\
b_N^2\\
\sqrt{2}b_1b_2\\
\sqrt{2}b_1b_3\\
\vdots\\
\sqrt{2}b_1b_N\\
\sqrt{2}b_2b_3\\
\vdots\\
\sqrt{2}b_1b_N\\
\vdots\\
\sqrt{2}b_{N-1}b_N
\end{pmatrix}
$$

$$
1 + \sum_{i=1}^{N} 2a_ib_i + \sum_{i=1}^{N} a_i^2b_i^2 + \sum_{i=1}^{N}\sum_{j=i+1}^{N} 2a_ia_jb_ib_j
$$

Quadratic Dot Products

$$<\Phi(\mathbf{a}),\Phi(\mathbf{b})>=$$

$$1+2\sum_{i=1}^{N} a_i b_i + \sum_{i=1}^{N}(a_i b_i)^2 + \sum_{i=1}^{N}\sum_{j=i+1}^{N} 2 a_i a_j b_i b_j$$

Now consider another fn of ***a*** and ***b***

$$(\mathbf{a}\cdot\mathbf{b}+1)^2$$

$$=(\mathbf{a}\cdot\mathbf{b})^2+2\mathbf{a}\cdot\mathbf{b}+1$$

$$=\left(\sum_{i=1}^{N} a_i b_i\right)^2 + 2\sum_{i=1}^{N} a_i b_i + 1$$

$$=\sum_{i=1}^{N}\sum_{j=1}^{N} a_i b_i a_j b_j + 2\sum_{i=1}^{N} a_i b_i + 1$$

$$=\sum_{i=1}^{N}(a_i b_i)^2 + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} a_i b_i a_j b_j + 2\sum_{i=1}^{N} a_i b_i + 1$$

They're the same!

And this is only O(N) to compute… not O(N^2)

Higher Order Polynomials

$$Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$$

$N \doteq dim(X)$, m = num of training examples

Poly-nomial	$\phi(\boldsymbol{x})$	Cost to build Q_{kl} matrix: *traditional*	Cost if N=100 dim inputs	$\phi(\boldsymbol{a})\cdot\phi(\boldsymbol{b})$	Cost to build Q_{kl} matrix: *sneaky*	Cost if 100 dim inputs
Quadratic	All $N^2/2$ terms up to degree 2	$N^2\, m^2/4$	2 500 m^2	$(\boldsymbol{a}\cdot\boldsymbol{b}+1)^2$	$N\, m^2/2$	50 m^2
Cubic	All $N^3/6$ terms up to degree 3	$N^3\, m^2/12$	83 000 m^2	$(\boldsymbol{a}\cdot\boldsymbol{b}+1)^3$	$N\, m^2/2$	50 m^2
Quartic	All $N^4/24$ terms up to degree 4	$N^4\, m^2/48$	1960000 m^2	$(\boldsymbol{a}\cdot\boldsymbol{b}+1)^4$	$N\, m^2/2$	50 m^2

taken from Andrew W. Moore

The Polynomial kernel, General case

$$\mathcal{X} \subset l_2^N = \mathbb{R}^N$$

$$\left.\begin{array}{l} u = (u_1, \dots u_N) \in \mathcal{X} \\ v = (v_1, \dots v_N) \in \mathcal{X} \end{array}\right\} \text{We are going to map these to a larger space}$$

$$k(u, v) \doteq (\langle u, v \rangle_{\mathcal{X}})^p$$

We want to show that this *k* is a kernel function

Let us try to find $\phi(u)$ and $\mathcal{K}$!

The Polynomial kernel, General case

$$\mathcal{X} \subset l_2^N = \mathbb{R}^N$$

$$u = (u_1, \dots u_N) \in \mathcal{X}$$

$$v = (v_1, \dots v_N) \in \mathcal{X}$$

We are going to map these to a larger space

$$k(u,v) \doteq \underbrace{(\langle u, v\rangle_{\mathcal{X}})^p}_{(\sum_{i=1}^{N} u_i v_i)^p} = (\sum_{i_1=1}^{N} u_{i_1} v_{i_1}) \cdots (\sum_{i_p=1}^{N} u_{i_p} v_{i_p})$$

P factors

because $(\sum a_i)(\sum b_j) = \sum \sum a_i b_j$

$$= \sum_{i_1=1}^{N} \cdots \sum_{i_p=1}^{N} \underbrace{(u_{i_1} \cdots u_{i_p})}_{\phi_{\vec{i}}(u)} \underbrace{(v_{i_1} \cdots v_{i_p})}_{\phi_{\vec{i}}(v)}$$

$$= \langle \phi(u), \phi(v) \rangle_{\mathcal{K}}$$ Let us try to find $\phi(u)$ and $\mathcal{K}$!

The Polynomial kernel, General case

We already know:

$$k(u,v) = \sum_{i_1=1}^{N} \cdots \sum_{i_p=1}^{N} \underbrace{(u_{i_1} \cdots u_{i_p})}_{\phi_{\vec{i}}(u)} \underbrace{(v_{i_1} \cdots v_{i_p})}_{\phi_{\vec{i}}(v)}$$

We want to get *k* in this form:

$$k(u,v) = \sum_{\vec{r}=(r_1,\ldots,r_N)} \alpha_{r_1,\ldots,r_N} u_1^{r_1} \cdots u_N^{r_N} v_1^{r_1} \cdots v_N^{r_N}$$

$$= \sum_{\vec{r}} \phi_{\vec{r}}(u) \phi_{\vec{r}}(v)$$

The Polynomial kernel

We already know:

$$k(u,v) = \sum_{i_1=1}^{N} \cdots \sum_{i_p=1}^{N} \underbrace{(u_{i_1} \cdots u_{i_p})}_{\phi_{\vec{i}}(u)} \underbrace{(v_{i_1} \cdots v_{i_p})}_{\phi_{\vec{i}}(v)}$$

$$u = (u_1, \ldots, u_N)$$

One factor in $k(u,v)$ can be written as $u_1^{r_1} \cdots u_N^{r_N}$

where $r_1 + r_2 + \ldots + r_N = p$, $r_i \in [0,p]$, $\vec{r} = (r_1, \ldots r_N)$

For example

Let $p = 3$, $N = 4$, now $u_1^2 u_4 = \underbrace{u_1 u_1 u_4}_{\vec{i}=(1,1,4)} = \underbrace{u_1 u_4 u_1}_{\vec{i}=(1,4,1)}$

$\vec{r} = (2,0,0,1)$

The Polynomial kernel

$$k(u,v) = \sum_{i_1=1}^{N} \cdots \sum_{i_p=1}^{N} \underbrace{(u_{i_1} \cdots u_{i_p})}_{\phi_{\vec{i}}(u)} \underbrace{(v_{i_1} \cdots v_{i_p})}_{\phi_{\vec{i}}(v)}$$

One factor in $k(u,v)$ can be rewritten as $u_1^{r_1} \cdots u_N^{r_N}$

The number of possible $\vec{r}$ vectors: $\binom{N+p-1}{p}$

because $r_1 + r_2 + \ldots + r_N = p$, $r_i \in [0,p]$,
$\vec{r} = (r_1, \ldots r_N)$

$$\boxed{\Rightarrow \text{ number of factors} = dim(\mathcal{K}) = \binom{N+p-1}{p}}$$

The Polynomial kernel

The $\vec{r} = (r_1, \ldots, r_N)$ term is calculated by

$$\alpha_{r_1,\ldots,r_N} \doteq \frac{p!}{r_1! \cdots r_N!} \text{ times}$$

$$r_1 + r_2 + \ldots + r_N = p,\ r_i \in [0, p],\ \vec{r} = (r_1, \ldots r_N)$$

$$\phi_{\vec{r}}(u) = \sqrt{\frac{p!}{r_1! \cdots r_N!}} u_1^{r_1} \cdots u_N^{r_N}$$

$$\phi_{\vec{r}}(v) = \sqrt{\frac{p!}{r_1! \cdots r_N!}} v_1^{r_1} \cdots v_N^{r_N}$$

$$\Rightarrow k(u, v) = \sum_{\vec{r}=(r_1,\ldots,r_N)} \alpha_{r_1,\ldots,r_N} u_1^{r_1} \cdots u_N^{r_N} v_1^{r_1} \cdots v_N^{r_N}$$

$$= \sum_{\vec{r}} \phi_{\vec{r}}(u) \phi_{\vec{r}}(v)$$

$\Rightarrow$ *k* is really a kernel!

Reproducing Kernel Hilbert Spaces

RKHS, Motivation

1.,

For a given kernel $k(\cdot,\cdot)$ we already know how to define feature space $\mathcal{K}$, and $\phi : \mathcal{X} \to \mathcal{K}$ feature map (Mercer map):

$$\mathcal{K} = l_2, \text{ and } \phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T$$

Now, we show another way using RKHS

2., What objective do we want to optimize?

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} |y_i - f(x_i)| + \lambda \|f\|_{\mathcal{F}}$$

$$\text{or } f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} |y_i - f(x_i)|^k + \lambda \|f\|_{\mathcal{F}}^j$$

$$\text{or } f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} |y_i - f(x_i)|^k + \lambda \exp\exp\exp(\|f\|_{\mathcal{F}}^j)$$

or ???

RKHS, Motivation

3., How can we minimize the objective over functions???

- Be PARAMETRIC!!!...

  (*nope, we do not like that...*)

- Use RKHS, and suddenly the problem will be finite dimensional optimization only (*yummy...*)

The Representer theorem will help us here

$$f^* = \arg\min_{f \in \mathcal{F}} R_{reg}[f, z] \doteq \arg\min_{f \in \mathcal{F}} \underbrace{g_{emp}[(x_i, y_i, f(x_i))_{i \in \{1 \ldots m\}}]}_{\text{1st term, empirical loss}} + \underbrace{g_{reg}(\|f\|)}_{\text{2nd term, regularization}}$$

Reproducing Kernel Hilbert Spaces

For a given kernel $k(\cdot,\cdot)$ we already know how to define feature space $\mathcal{K}$, and $\phi : \mathcal{X} \to \mathcal{K}$ feature map (Mercer map):

$$\mathcal{K} = l_2, \text{ and } \phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T$$

Now, we show another way using RKHS

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ given kernel $\Rightarrow \mathcal{F}_0 \doteq \{k(x,\cdot) | x \in \mathcal{X}\}$ function space

We will add inner product to $\mathcal{F}_0$ function space
$\Rightarrow$ Pre-Hilbert space

Completing (closing) a pre-Hilbert space $\Rightarrow$ Hilbert space

Reproducing Kernel Hilbert Spaces

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ given kernel $\Rightarrow \mathcal{F}_0 \doteq \{k(x,\cdot) | x \in \mathcal{X}\}$ function space

$$(x_1, \ldots, x_r) \text{ given } \Rightarrow f(\cdot) \doteq \sum_{i=1}^{r} \alpha_i k(x_i, \cdot) \quad \in \mathcal{F}_0$$

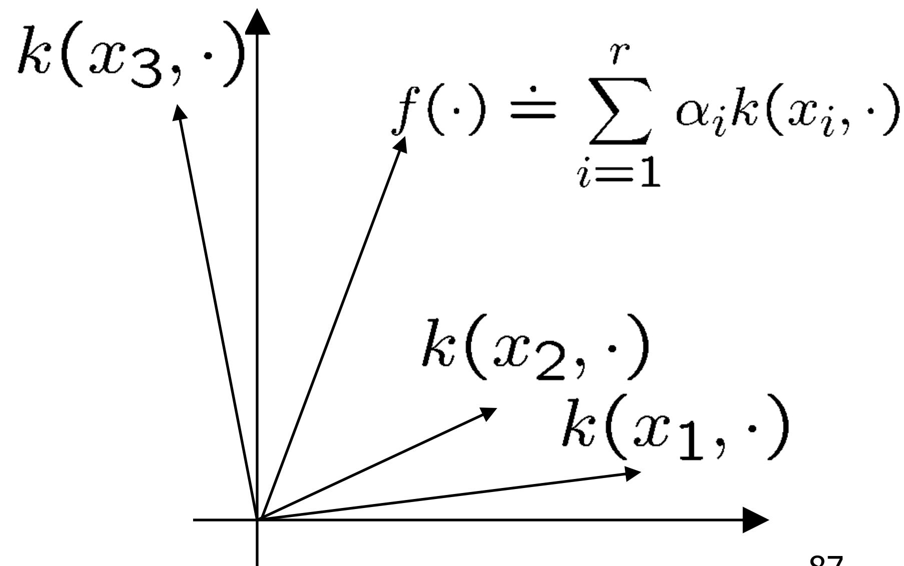
$$(\tilde{x}_1, \ldots, \tilde{x}_s) \text{ given } \Rightarrow g(\cdot) \doteq \sum_{j=1}^{s} \beta_j k(\tilde{x}_j, \cdot) \quad \in \mathcal{F}_0$$

The inner product:

$$\langle f, g \rangle_{\mathcal{F}_0} \doteq \sum_{i=1}^{r} \sum_{j=1}^{s} \alpha_i \beta_j k(x_i, \tilde{x}_j)$$

$$= \sum_{i=1}^{r} \alpha_i g(x_i)$$

$$= \sum_{j=1}^{s} \beta_j f(\tilde{x}_j) \quad (*)$$



Reproducing Kernel Hilbert Spaces

Note:

While for calculating $\langle f, g \rangle_{\mathcal{F}_0}$ we use their representations: $\alpha \in \mathbb{R}^r, \beta \in \mathbb{R}^s, \{x_i\}_{i=1}^r, \{\tilde{x}_j\}_{j=1}^s$
the $\langle f, g \rangle_{\mathcal{F}_0}$ is independent of the representation of f, g

Proof:

If we change $\alpha \in \mathbb{R}^r$ or $x_i \Rightarrow \langle f, g \rangle_{\mathcal{F}_0}$ doesn't change (because of (*)) The same for $\beta \in \mathbb{R}^s$

$$\boxed{\langle f, g \rangle_{\mathcal{F}_0} = \sum_{i_1}^{r} \alpha_i f(x_i) = \sum_{j=1}^{s} \beta_j f(\tilde{x}_j) \quad (*)}$$

Reproducing Kernel Hilbert Spaces

Lemma:

$\langle f, g \rangle$ is an inner product of $\mathcal{F}_0$

$\Rightarrow \mathcal{F}_0$ is pre-Hilbert space

$\mathcal{F} \doteq close(\mathcal{F}_0)$ is a Hilbert space

- **Pre-Hilbert** space:
  Like the Euclidean space with *rational* scalars only
- **Hilbert space**:
  Like the Euclidean space with *real* scalars

Proof:

1., $\langle f, g \rangle_{\mathcal{F}_0} = \langle g, f \rangle_{\mathcal{F}_0}$

2., $\langle cf + dg, h \rangle_{\mathcal{F}_0} = c\langle f, h \rangle_{\mathcal{F}_0} + d\langle g, h \rangle_{\mathcal{F}_0}$, $\forall c, d \in \mathbb{R}$, $\forall f, g, h \in \mathcal{F}_0$

3., $\langle f, f \rangle_{\mathcal{F}_0} \geq 0$

4., $\langle f, f \rangle_{\mathcal{F}_0} = 0 \Leftrightarrow f = 0$

Reproducing Kernel Hilbert Spaces

Lemma: (Reproducing property)

$$\langle f, k(x,\cdot)\rangle_{\mathcal{F}} = f(x)$$

Proof: definition of $\langle f, g\rangle_{\mathcal{F}}$

Lemma: The constructed features match to *k*

Huhh...

$$\langle \underbrace{k(x_i,\cdot)}_{\phi(x_i)}, \underbrace{k(x_j,\cdot)}_{\phi(x_j)} \rangle_{\mathcal{F}} = k(x_i, x_j)$$

Proof: reproducing property

Reproducing Kernel Hilbert Spaces

Proof of property 4.,:

$$0 \leq (f(x))^2 = \langle f, k(x,\cdot)\rangle_{\mathcal{F}}^2, \ \forall x$$

rep. property

$$\langle f, k(x,\cdot)\rangle_{\mathcal{F}}^2 \leq \langle f, f\rangle_{\mathcal{F}} \langle k(x,\cdot), k(x,\cdot)\rangle_{\mathcal{F}} \quad \forall x$$

CBS

For CBS we don't need 4.,
we need only that <0,0>=0!

Hence, if $\langle f, f\rangle_{\mathcal{F}} = 0 \Rightarrow (f(x))^2 = 0, \ \forall x \in \mathcal{X}$

$$\Rightarrow f(x) = 0, \ \forall x \in \mathcal{X}$$

$$\Rightarrow f = 0$$

Methods to Construct Feature Spaces

We now have two methods to construct feature maps from kernels

1., **Mercer map**:
$\mathcal{K} = l_2$, and $\phi(x) \doteq (\sqrt{\lambda_1}\psi_1(x), \sqrt{\lambda_2}\psi_2(x), \ldots)^T \in l_2$

2., **RKHS map**:
$\mathcal{K} = \mathcal{F}$, and $\phi(x) \doteq k(x, \cdot) \in \mathcal{F}$

For finite discrete $\mathcal{X}$, $|\mathcal{X}| = r$ we already know a 3^{rd} **method:**

3., $\mathcal{K} \subset \mathbb{R}^n$, $\phi(x_i) = \Lambda^{1/2} u_i \in \mathbb{R}^n$, $i = 1, \ldots r$

Well, these feature spaces are all isomorph with each other... ☺

The Representer Theorem

In the perceptron problem we could use the dual algorithm, because we had this representation:

$$f(x) \doteq \langle \phi(x), \mathbf{w} \rangle_{\mathcal{K}} = \sum_{i=1}^{m} \alpha_i k(x, x_i)$$

and thus we had to update $\alpha_1, \ldots, \alpha_m$ only, and not $\mathbf{w} \in \mathcal{K}$!

The **Representer theorem** provides us a big class of problems, where the solution can be represented by

$$\boxed{f(\cdot) = \sum_{i=1}^{m} \alpha_i k(x_i, \cdot), \quad \alpha \in \mathbb{R}^m}$$

The Representer Theorem

Theorem:

$$\left.\begin{array}{r} k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \text{Mercer kernel on } \mathcal{X} \\ \boldsymbol{z} = (x_1, y_1), \ldots, (x_m, y_m) \in (\mathcal{X} \times \mathcal{Y})^m \text{ training sample} \\ g_{emp} : (\mathcal{X} \times \mathcal{Y} \times \mathbb{R})^m \to \mathbb{R} \cup \{\infty\} \\ g_{reg} : \mathbb{R} \to [0, \infty) \text{ strictly increasing function} \\ \mathcal{F} : \ \text{RKHS induced by } k(\cdot,\cdot) \end{array}\right\} \Rightarrow$$

$$\Rightarrow f^* = \arg\min_{f \in \mathcal{F}} R_{reg}[f, \boldsymbol{z}]$$

$$\doteq \arg\min_{f \in \mathcal{F}} \ \underbrace{g_{emp}[(x_i, y_i, f(x_i))_{i \in \{1 \ldots m\}}]}_{\text{1st term, empirical loss}} + \underbrace{g_{reg}(\|f\|)}_{\text{2nd term, regularization}}$$

admits the following representation:

$$f^*(\cdot) = \sum_{i=1}^{m} \alpha_i k(x_i, \cdot), \quad \alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$$

The Representer Theorem

Message:

Optimizing in general function classes is difficult, but in RKHS it is only finite! (m) dimensional problem

Proof of Representer Theorem:

$$\phi(x) \doteq k(x, \cdot) = \phi(x)(\cdot)$$

$x_1, \ldots, x_m$ training samples are given

$$f \in \mathcal{F} \Rightarrow f(\cdot) = \sum_{i=1}^{m} \alpha_i \phi(x_i)(\cdot) + v(\cdot)$$

where $\mathcal{F} \ni v \perp span\{\phi(x_1), \ldots, \phi(x_m)\}$,

thus $\langle v, \phi(x_i) \rangle_{\mathcal{F}} = 0 \quad \forall i = 1, \ldots, m$

Proof of the Representer Theorem

Proof of Representer Theorem

$$f^* = \arg\min_{f\in\mathcal{F}} R_{reg}[f, \boldsymbol{z}] \doteq \arg\min_{f\in\mathcal{F}} \underbrace{g_{emp}[(x_i, y_i, f(x_i))_{i\in\{1\dots m\}}]}_{1^{st}\text{ term, empirical loss}} + \underbrace{g_{reg}(\|f\|)}_{2^{nd}\text{ term, regularization}}$$

$$\Rightarrow f(x_j) = \langle f, \underbrace{k(x_j,\cdot)}_{\phi(x_j)}\rangle_{\mathcal{F}} = \langle \sum_{i=1}^{m} \alpha_i \phi(x_i) + v, \phi(x_j)\rangle_{\mathcal{F}}$$

$$= \sum_{i=1}^{m} \alpha_i \langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{F}} = \sum_{i=1}^{m} \alpha_i k(x_i, x_j)$$

$\Rightarrow f(x_j)$ depends only on $\alpha_1, \dots, \alpha_m$, but independent from v!

$\Rightarrow 1^{st}$ term depends only on $\alpha_1, \dots, \alpha_m$, but not on v

Proof of the Representer Theorem

$$f^* = \arg\min_{f\in\mathcal{F}} R_{reg}[f,z] \doteq \arg\min_{f\in\mathcal{F}} \underbrace{g_{emp}[(x_i,y_i,f(x_i))_{i\in\{1\ldots m\}}]}_{\text{1st term, empirical loss}} + \underbrace{g_{reg}(\|f\|)}_{\text{2nd term, regularization}}$$

Let us examine the 2^{nd} term.

$$g_{reg}(\|f\|) = g_{reg}(\|\sum_{i=1}^m \alpha_i \phi(x_i) + v\|)$$

$$= g_{reg}\left(\sqrt{\|\sum_{i=1}^m \alpha_i \phi(x_i)\|_{\mathcal{F}}^2 + \|v\|_{\mathcal{F}}^2}\right.$$

since $\mathcal{F} \ni v \perp span\{\phi(x_1),\ldots,\phi(x_m)\}$

$$\geq g_{reg}(\|\sum_{i=1}^m \alpha_i \phi(x_i)\|_{\mathcal{F}})$$

with equality only if $v = 0$!

$\Rightarrow$ any minimizer f^* must have $v = 0$

$$\Rightarrow f^*(\cdot) = \sum_{i=1}^m \alpha_i k(x_i,\cdot)$$

Later will come

- **Supervised Learning**
  - SVM using kernels
  - Gaussian Processes
    - Regression
    - Classification
    - Heteroscedastic case

- **Unsupervised Learning**
  - Kernel Principal Component Analysis
  - Kernel Independent Component Analysis
    - Kernel Mutual Information
    - Kernel Generalized Variance
    - Kernel Canonical Correlation Analysis

If we still have time…

- Automatic Relevance Machines
- Bayes Point Machines

- Kernels on other objects
  - Kernels on graphs
  - Kernels on strings
- Fisher kernels
- ANOVA kernels
- Learning kernels

Thanks for the Attention! ☺