Introduction to Kernel Methods

Liva Ralaivola

LIF, UMR 6166 CNRS
Université de Provence
liva.ralaivola@lif.univ-mrs.fr

28 mars 2007

Outline

Outline

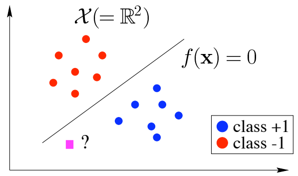
1. Classification
2. Quick reminders
3. A first Kernel Classifier
4. About kernels
5. Where are we heading to ? SVMs
6. What should be taken home ?
7. References

Classifying things (1/2)

- Classification is a real-world task
  - does some patient have a serious disease ?
  - what kind of secondary structure does a sequence of amino acids correspond to ?
  - is some molecule toxic/not toxic for some organism ?
- Automating this task
  - dealing with large number of items to classify
  - speed
  - cost

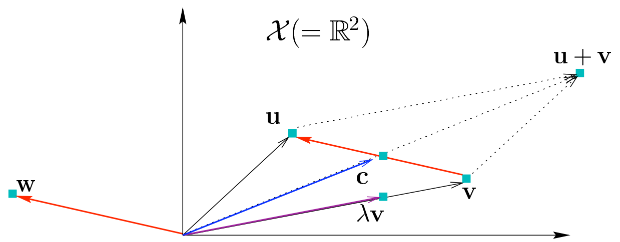
Classifying things (2/2)

- Important notions in *learning to classify*
  - limited number of *training* data (patients, sequences, molecules, etc.)
  - learning algorithm (how to build the classifier ?)
  - generalization : the classifier should correctly classify *test* data
- Quick formalization
  - $\mathcal{X}$ (e.g. $\mathbb{R}^d$, $d > 0$) is the space of data, called *input space*
  - $\mathcal{Y}$ (e.g. toxic/not toxic, or $\{-1, +1\}$) is the target space
  - $f : \mathcal{X} \rightarrow \mathcal{Y}$ is the classifier



Outline

Vectors and inner product (1/3)

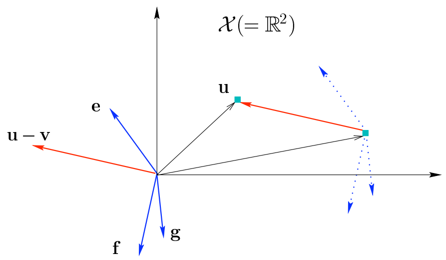


- $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{c}$ are vectors
- $\mathbf{w} = \mathbf{u} - \mathbf{v}$ (red arrows)
- $\mathbf{c} = \frac{1}{2}(\mathbf{u} + \mathbf{v})$
- Here : $0 < \lambda < 1$

Vectors and inner product (2/3)

- Inner product $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$:
  - symmetric : $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$
  - bilinear : $\langle \lambda \mathbf{u}_1 + \gamma \mathbf{u}_2, \mathbf{v} \rangle = \lambda \langle \mathbf{u}_1, \mathbf{v} \rangle + \gamma \langle \mathbf{u}_2, \mathbf{v} \rangle$
  - positive : $\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$
  - definite : $\langle \mathbf{u}, \mathbf{u} \rangle = 0 \Rightarrow \mathbf{u} = 0$
- An inner product
  - provides $\mathcal{X}$ with a structure
  - can be viewed as a 'similarity'
  - defines a norm $\| \cdot \|$ on $\mathcal{X}$: $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$
- Example in $\mathbb{R}^2$
  - $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} : \langle \mathbf{u}, \mathbf{v} \rangle = u_1 v_1 + u_2 v_2$

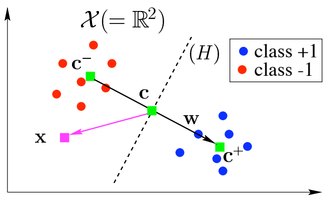
Vectors and inner product (3/3)



- $\langle \mathbf{u}-\mathbf{v}, \mathbf{e}\rangle > 0$: $\mathbf{u}-\mathbf{v}$ and $\mathbf{e}$ point to the 'same direction'
- $\langle \mathbf{u}-\mathbf{v}, \mathbf{f}\rangle = 0$: $\mathbf{u}-\mathbf{v}$ and $\mathbf{f}$ are orthogonal
- $\langle \mathbf{u}-\mathbf{v}, \mathbf{g}\rangle < 0$: $\mathbf{u}-\mathbf{v}$ and $\mathbf{g}$ point to 'opposite directions'

Outline

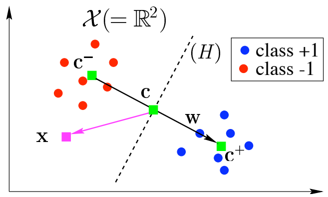
A simple linear classifier



- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- Idea (see [Schölkopf and Smola, 2002] for details) : classify points $\mathbf{x}$ according to which of the two class means $\mathbf{c}^+$ or $\mathbf{c}^-$ is closer :
  - for $\mathbf{x} \in \mathcal{X}$, it is sufficient to take the sign of the inner product between $\mathbf{w}$ and $\mathbf{x}-\mathbf{c}$
  - if $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}-\mathbf{c}\rangle$, we have the classifier $f(\mathbf{x}) = \text{sign}\,(h(\mathbf{x}))$
  - the (dotted) hyperplane (H), of normal vector $\mathbf{w}$, is the decision surface

A simple linear classifier

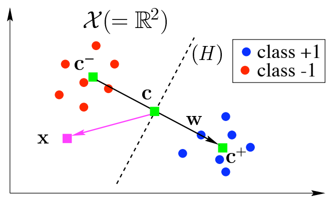


- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- On evaluating $h(\mathbf{x})$

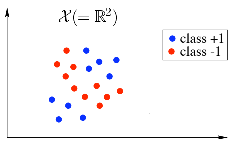
$$
\begin{aligned}
h(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x}-\mathbf{c}\rangle = \langle \mathbf{w}, \mathbf{x}\rangle - \langle \mathbf{w}, \mathbf{c}\rangle \\
&= \langle \mathbf{c}^+, \mathbf{x}\rangle - \langle \mathbf{c}^-, \mathbf{x}\rangle - \langle \mathbf{c}^+, \mathbf{c}\rangle + \langle \mathbf{c}^-, \mathbf{c}\rangle \\
&= \sum_{i=1,\ldots,m} \alpha_i \langle \mathbf{x}_i, \mathbf{x}\rangle + b, \quad \text{with } b \text{ a real constant}
\end{aligned}
$$

A simple linear classifier



- $\mathbf{c}^+ = \frac{1}{m^+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i$
- $\mathbf{c}^- = \frac{1}{m^-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i$
- $\mathbf{c} = \frac{1}{2}(c^+ + c^-)$
- $\mathbf{w} = c^+ - c^-$

- To summarize : $h(\mathbf{x}) = \sum_{i=1,\dots,m} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b$ [Schölkopf et al., 2000]
- Question : what if the dataset is not linearly separable, i.e. (H) fails to separate red and blue disks ?



The kernel trick (1/4)

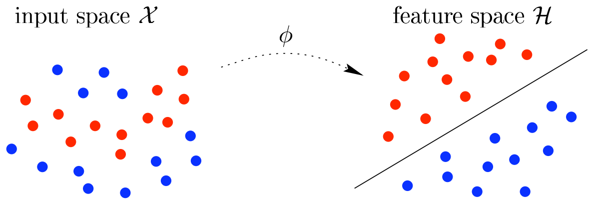
- Context : nonlinearly separable dataset $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$
- Idea to learn a nonlinear classifier
  - choose a (nonlinear) mapping ϕ

$$\phi : \begin{array}{ccc} \mathcal{X} & \rightarrow & \mathcal{H} \\ \mathbf{x} & \mapsto & \phi(\mathbf{x}) \end{array}$$

    where $\mathcal{H}$ is an inner product space (inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$), called *feature space*
  - find a linear classifier (i.e. a separating hyperplane) in $\mathcal{H}$ to classify $\{(\phi(\mathbf{x}_1), y_1), \dots, (\phi(\mathbf{x}_m), y_m)\}$

The kernel trick (2/4)

- Linearly classifying in *feature space*



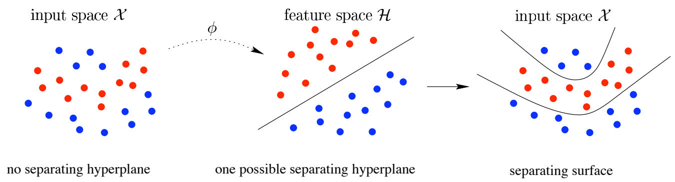
- Taking the previous linear algorithm and implementing it in $\mathcal{H}$:

$$h(\mathbf{x}) = \sum_{i=1,\dots,m} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}} + b$$

The kernel trick (3/4)

- The kernel trick can be applied if there is a function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that : $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle_{\mathcal{H}}$
  If so, all occurrences of $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle_{\mathcal{H}}$ are replaced by $k(\mathbf{x}_i, \mathbf{x})$
- Keypoint : emphasis is sometimes more on k than on ϕ
- Kernels must verify Mercer's property to be valid kernels
  - ensures that there exist a space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle_{\mathcal{H}}$
  - however non valid kernels have been used with success
  - and, research is in progress on using non semi-definite kernels
- k might be viewed as a similarity measure

The kernel trick (4/4)



- *Kernel trick* recipe
  - consider a nonlinear classification problem on $\mathcal{X} \times \mathcal{Y}$
  - choose a linear classification algorithm (expr. in terms $\langle \cdot, \cdot \rangle$)
  - replace all occurrences of $\langle \cdot, \cdot \rangle$ by a kernel $k(\cdot, \cdot)$
- Obtained classifier : $f(\mathbf{x}) = \text{sign}\left(\sum_{i=1,\dots,m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \right)$

Outline

(Kernel) Gram matrices (1/2)

- Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a Mercer kernel ($\mathcal{X}$ may be a space of sequences)
  - for a set of patterns $\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$

$$K_\mathcal{S} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_\ell) \\ k(\mathbf{x}_1, \mathbf{x}_2) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_\ell) \\ & \cdots & \cdots & \\ k(\mathbf{x}_1, \mathbf{x}_\ell) & k(\mathbf{x}_2, \mathbf{x}_\ell) & \cdots & k(\mathbf{x}_\ell, \mathbf{x}_\ell) \end{bmatrix}$$

  is the Gram matrix of k with respect to $\mathcal{S}$
  - if corresponding targets $y_1, \dots, y_\ell$ are available

$\Rightarrow K_\mathcal{S}$ is sufficient for any Kernel Machine to be trained

(Kernel) Gram matrices (2/2)

- A property of the Gram matrix (Mercer's property)

Proposition (Semi-Positiveness of the Gram matrix)

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function.
k is a Mercer kernel $\Leftrightarrow$

$$\forall \mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}, \mathbf{x}_i \in \mathcal{X}, \mathbf{v} K_\mathcal{S} \mathbf{v} \geq 0, \forall \mathbf{v} \in \mathbb{R}^\ell$$

- This means that for any Mercer kernel k and any set of patterns $\mathcal{S}$, the Gram matrix $K_\mathcal{S}$ has only nonnegative eigenvalues
- This gives, in particular, k_1 and k_2 being Mercer kernels
  - $k_1^p,\ p \in \mathbb{N}$ is a Mercer kernel
  - $\lambda k_1 + \gamma k_2,\ \lambda, \gamma > 0$ is a Mercer kernel
  - $k_1 k_2$ is a Mercer kernel

Common kernels (1/3)

- Gaussian kernel
  - $k(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right), \quad \sigma^2 > 0$
  - the corresponding $\mathcal{H}$ is of infinite dimension
- Polynomial kernel
  - $k(\mathbf{u},\mathbf{v}) = (\langle \mathbf{u},\mathbf{v}\rangle + c)^d, \quad c \in \mathbb{R}, d \in \mathbb{N}$
  - a corresponding analytic ϕ may be constructed (see below)
- Tangent kernel (it is *not* a Mercer kernel)
  - $k(\mathbf{u},\mathbf{v}) = \tanh(a\langle \mathbf{u},\mathbf{v}\rangle + c), \quad a, c \in \mathbb{R}$

Common kernels (2/3)

- Let $k = \langle \mathbf{u},\mathbf{v}\rangle^2_{\mathbb{R}^2}$ (polynomial kernel with $c = 0$ and $d = 2$) defined on $\mathbb{R}^2 \times \mathbb{R}^2$
- Consider the mapping :

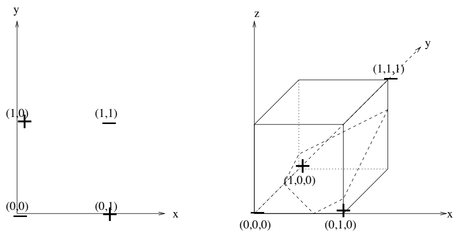
$$\begin{array}{rccc} \phi : & \mathbb{R}^2 & \to & \mathbb{R}^3 \\ & \mathbf{x} = [x_1, x_2]^\top & \mapsto & \phi(\mathbf{x}) = \left[x_1^2, \sqrt{2}x_1x_2, x_2^2\right]^\top \end{array}$$

- We have, for $\mathbf{u},\mathbf{v} \in \mathbb{R}^2$:

$$\begin{aligned} \langle \phi(\mathbf{u}), \phi(\mathbf{v})\rangle_{\mathbb{R}^3} &= \langle [u_1^2, \sqrt{2}u_1u_2, u_2^2]^\top, [v_1^2, \sqrt{2}v_1v_2, v_2^2]^\top\rangle \\ &= (u_1v_1 + u_2v_2)^2 \\ &= \langle \mathbf{u},\mathbf{v}\rangle^2_{\mathbb{R}^2} \\ &= k(\mathbf{u},\mathbf{v}) \end{aligned}$$

Common kernels (3/3)

Another polynomial Mapping



(x,y) ----> (x,y,xy)

C : −x −y + 2xy +1/3 =0 <---- P : −x−y+2z+1/3=0

The decision surface in the input space is an ellipse

Kernel Perceptron

Primal algorithm

Require: $\mathcal{S} = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_\ell, y_\ell)\}$
Ensure: a classifier $f(\cdot) = \langle \mathbf{w}, \cdot\rangle, \mathbf{w} \in \mathcal{X}$

$t \leftarrow 0$
$\mathbf{w}_0 \leftarrow \mathbf{0}$
while there is i s.t. $y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \le 0$ **do**
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\mathbf{x}_i$
  $t \leftarrow t + 1$
end while
return $\mathbf{w}_t$

Kernel Perceptron

Dual algorithm

Require: $\mathcal{S} = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_\ell, y_\ell)\}$
Ensure: a classifier $f(\cdot) = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \cdot \rangle$, $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i \mathbf{x}_i$

$t \leftarrow 0$
$\boldsymbol{\alpha}^T = [0 \cdots 0]$ (size ℓ),
while there is i s.t. $y_i \sum_{j=1}^{\ell} \alpha_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \leq 0$ **do**
  $\alpha_{t+1}^i \leftarrow \alpha_t^i + y_i$
  $t \leftarrow t + 1$
end while
return $\boldsymbol{\alpha}_t$

Kernel Perceptron

Dual algorithm - Kernel version

Require: $\mathcal{S} = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_\ell, y_\ell)\}$, k a kernel (mapping ϕ)
Ensure: a classifier
$f(\cdot) = \sum_{i=1}^{\ell} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\cdot) \rangle = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \cdot)$, $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i \phi(\mathbf{x}_i)$

$t \leftarrow 0$
$\boldsymbol{\alpha}^T = [0 \cdots 0]$ (size ℓ),
while there is i s.t. $y_i \sum_{j=1}^{\ell} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \leq 0$ **do**
  $\alpha_{t+1}^i \leftarrow \alpha_t^i + y_i$
  $t \leftarrow t + 1$
end while
return $\boldsymbol{\alpha}_t$

Kernel Perceptron

Dual algorithm - Kernel version

Require: $\mathcal{S} = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_\ell, y_\ell)\}$, k a kernel (mapping ϕ)
Ensure: a classifier
$f(\cdot) = \sum_{i=1}^{\ell} \alpha_i \langle \phi(\mathbf{x}_i), \phi(\cdot) \rangle = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \cdot)$, $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i \phi(\mathbf{x}_i)$

$t \leftarrow 0$
$\boldsymbol{\alpha}^T = [0 \cdots 0]$ (size ℓ),
while there is i s.t. $y_i \sum_{j=1}^{\ell} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \leq 0$ **do**
  $\alpha_{t+1}^i \leftarrow \alpha_t^i + y_i$
  $t \leftarrow t + 1$
end while
return $\boldsymbol{\alpha}_t$

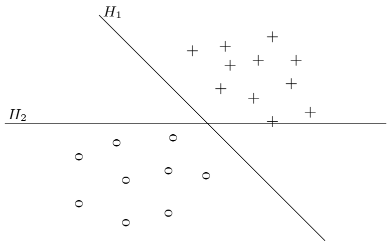
Question

Suppose that the training set $\mathcal{S}$ is separable with margin $\gamma > 0$ in the feature space and that we have $\|\phi(\mathbf{x})\| \leq R, \forall \mathbf{x} \in \mathcal{X}$. Give a maximal value for any $|\alpha_i|$ produced by the kernel percepton algorithm.

Outline

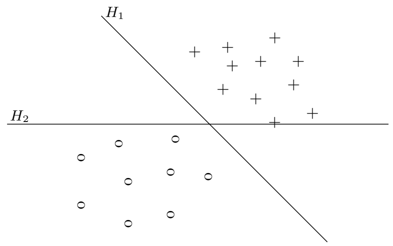
Optimal Separating Hyperplane

Question : which of H_1 or H_2 is the best hyperplane ?



Optimal Separating Hyperplane

Question : which of H_1 or H_2 is the best hyperplane ?



Intuitively, it is H_1 because it shows a larger (better) **margin** with respect to the training set. The hyperplane with the largest margin is the optimal hyperplane.

Optimal Separating Hyperplane

Theorem (Example of a margin theorem)

Théorème : (Vapnik, Bartlett, Shawe-Taylor 99)

Soient $P(\cdot,\cdot)$, distribution de probabilité sur $\mathbb{R}^n \times \{-1, 1\}$, $\delta > 0$, S un ensemble de l exemples i.i.d. selon P, $R > 0$ tq $\forall (x, y) \in S$, $||x|| < R$.

Soit f un classifieur linéaire et soit $\rho > 0$.

$R_\rho(f)$: proportion des exemples de S mal classés ou situés à une distance inférieure à ρ de l'hyperplan d'équation $f(x) = 0$.

Alors, avec une probabilité supérieure à $1 - \delta$, on a

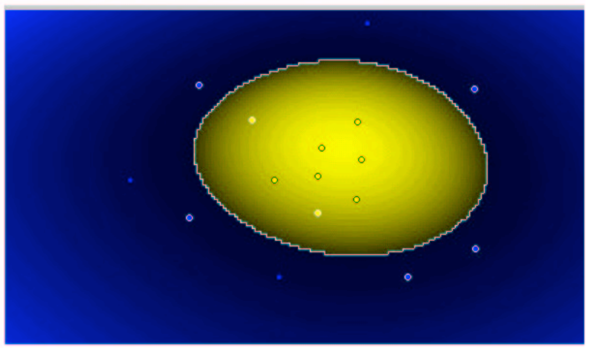
$$R(f) \le R_\rho(f) + \sqrt{\frac{c}{l}\left(\frac{R^2}{\rho^2}\ln^2\frac{l}{\rho} + \ln\frac{1}{\delta}\right)}$$

où c est une constante.

Kernel Optimal Hyperplanes : examples

RBF kernel

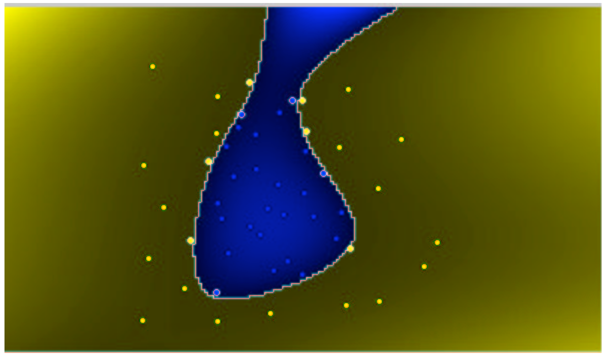
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma||\mathbf{x} - \mathbf{x}'||^2\right)$$



Kernel Optimal Hyperplanes : examples

Polynomial kernel

$$k(\mathbf{x},\mathbf{x}') = (\langle \mathbf{x},\mathbf{x}' \rangle + 1)^p$$



Outline

Kernels Methods

- are wonderful !
- field of active, thrilling research
- classification of structured objects might be envisioned
  - rough scale of the difficulty to classify objects

methods

many ← few

$\mathcal{X} =$ $\mathbb{N}^d, \mathbb{R}^d$ $\cdots$ sequences $\cdots$ graphs

"easy" → hard

difficulty

  - sequences : DNA strings, amino-acid strings, texts
  - graphs : structure of a molecule, disulfide bonds, GO
- "Breakthrough" work on Kernel Methods : Support Vector Machines [Boser et al., 1992, Cortes and Vapnik, 1995]

Outline

Boser, B., Guyon, I., and Vapnik, V. (1992).
A Training Algorithm for Optimal Margin Classifiers.
In *Proc. of the 5th Workshop on Comp. Learning Theory*, volume 5.

Cortes, C. and Vapnik, V. (1995).
Support Vector Networks.
Machine Learning, 20 :1–25.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2000).
A generalized representer theorem.
Technical Report NC-TR-00-081, NeuroCOLT.

Schölkopf, B. and Smola, A. J. (2002).
Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond.
MIT University Press.