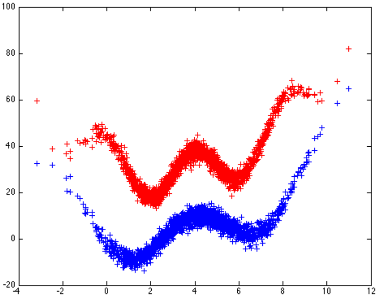
Kernels in machine learning

October 2021

Thomas Corpetti

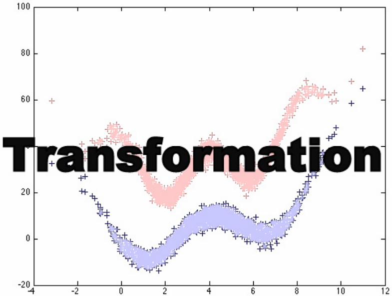
Main principles

- We have separable data, but not linearly separable data



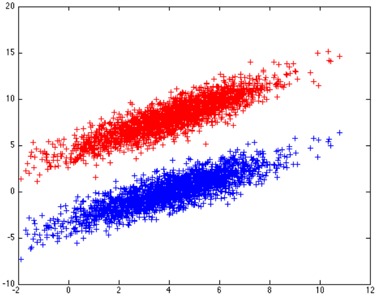
Main principles

- We have separable data, but not linearly separable data
- Make a projection in another space



Main principles

- We have separable data, but not linearly separable data
- Make a projection in another space
- In this new space, data are linearly separable



Outline

1. Introduction

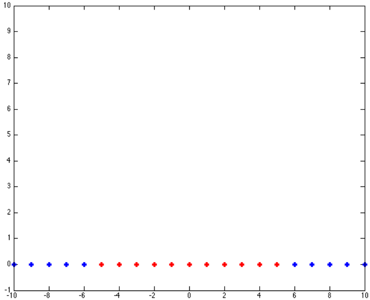
2. What is a kernel ?
   - Introduction
   - Positive definite kernels
   - SVM

3. Evaluation
   - Time series

Separability of data

- Ex : find a linear separation of data X in dimension 1

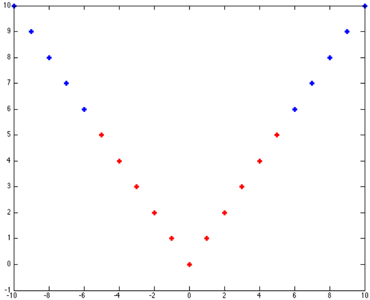
  $\Longrightarrow$ no solution



Separability of data

- We artificially add a dimension : $X' = (X, \sqrt{X^2})$

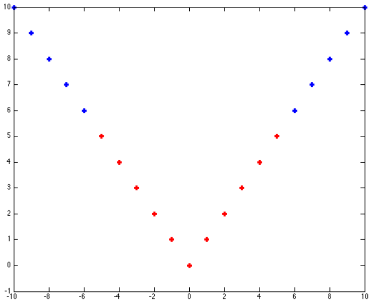
$\Longrightarrow$ linear solution



Separability of data

- We artificially add a dimension : $X' = (X, \sqrt{X^2})$

$\Longrightarrow$ linear solution



In general, the augmentation on the dimension enables to increase the separability.

Data in large dimension

Make a transformation in a higher dimensional spacen

- Principle : the initial representation space is transformed :

$$X = [X_1, ..., X_P] \Longrightarrow \phi(X) = [\phi(X_1), ..., \phi(X_P)]$$

with $\phi : \mathbb{R}^N \rightarrow \mathcal{F}^M$ (in general $M >> N$).

- Difficulty : large computational cost.

Main idea

- Instead of computing the transformation with ϕ on every elements of X, we use a une similarity function :

$$K : \mathbb{R}^N \times R^N \rightarrow \mathbb{R}$$

$\Longrightarrow$ K can be seen as a variance-covariance function in the new space (also named *feature space*)

- We try to manipulate only dot products or variance-covariance matrices

K is a kernel function

Data in large dimension

Make a transformation in a higher dimensional spacen

- Principle : the initial representation space is transformed :

$$X = [X_1, ..., X_P] \Longrightarrow \phi(X) = [\phi(X_1), ..., \phi(X_P)]$$

with $\phi : \mathbb{R}^N \to \mathcal{F}^M$ (in general $M >> N$).

- Difficulty : large computational cost.

Main idea

- Instead of computing the transformation with ϕ on every elements of X, we use a une similarity function :

$$K : \mathbb{R}^N \times R^N \to \mathbb{R}$$

$\Longrightarrow$ K can be seen as a variance-covariance function in the new space (also named *feature space*)

- We try to manipulate only dot products or variance-covariance matrices

K is a kernel function

Data in large dimension

Make a transformation in a higher dimensional spacen

- Principle : the initial representation space is transformed :

$$X = [X_1, ..., X_P] \Longrightarrow \phi(X) = [\phi(X_1), ..., \phi(X_P)]$$

with $\phi : \mathbb{R}^N \rightarrow \mathcal{F}^M$ (in general $M >> N$).

- Difficulty : large computational cost.

Main idea

- Instead of computing the transformation with ϕ on every elements of X, we use a une similarity function :

$$K : \mathbb{R}^N \times R^N \rightarrow \mathbb{R}$$

$\Longrightarrow$ K can be seen as a variance-covariance function in the new space (also named *feature space*)

- We try to manipulate only dot products or variance-covariance matrices

K is a kernel function

Data in large dimension

Make a transformation in a higher dimensional spacen

- Principle : the initial representation space is transformed :

$$X = [X_1, ..., X_P] \Longrightarrow \phi(X) = [\phi(X_1), ..., \phi(X_P)]$$

with $\phi : \mathbb{R}^N \to \mathcal{F}^M$ (in general $M >> N$).

- Difficulty : large computational cost.

Main idea

- Instead of computing the transformation with ϕ on every elements of X, we use a une similarity function :

$$K : \mathbb{R}^N \times R^N \to \mathbb{R}$$

$\Longrightarrow$ K can be seen as a variance-covariance function in the new space (also named *feature space*)

- We try to manipulate only dot products or variance-covariance matrices

K is a kernel function

Data in large dimension

Make a transformation in a higher dimensional spacen

- Principle : the initial representation space is transformed :

$$X = [X_1, ..., X_P] \Longrightarrow \phi(X) = [\phi(X_1), ..., \phi(X_P)]$$

with $\phi : \mathbb{R}^N \rightarrow \mathcal{F}^M$ (in general $M >> N$).

- Difficulty : large computational cost.

Main idea

- Instead of computing the transformation with ϕ on every elements of X, we use a une similarity function :

$$K : \mathbb{R}^N \times R^N \rightarrow \mathbb{R}$$

$\Longrightarrow$ K can be seen as a variance-covariance function in the new space (also named *feature space*)

- We try to manipulate only dot products or variance-covariance matrices

K is a kernel function

Remarks on kernels

General remarks on kernels

- A similarity matrix is a square real matrix, whatever the dimension of data X.
- If a technique involves only similarity matrices, the similarity can be replace by any other function (modularity).
- Whatever the dimension of the transformation ϕ, the kernel always gives a real function, with a similarity matrix of size $N \times N$.
- But kernels have to be chosen properly.

Remarks on kernels

General remarks on kernels

- A similarity matrix is a square real matrix, whatever the dimension of data X.
- If a technique involves only similarity matrices, the similarity can be replace by any other function (modularity).
- Whatever the dimension of the transformation ϕ, the kernel always gives a real function, with a similarity matrix of size $N \times N$.
- But kernels have to be chosen properly.

Remarks on kernels

General remarks on kernels

- A similarity matrix is a square real matrix, whatever the dimension of data X.
- If a technique involves only similarity matrices, the similarity can be replace by any other function (modularity).
- Whatever the dimension of the transformation ϕ, the kernel always gives a real function, with a similarity matrix of size $N \times N$.
- But kernels have to be chosen properly.

Remarks on kernels

General remarks on kernels

- A similarity matrix is a square real matrix, whatever the dimension of data X.
- If a technique involves only similarity matrices, the similarity can be replace by any other function (modularity).
- Whatever the dimension of the transformation ϕ, the kernel always gives a real function, with a similarity matrix of size $N \times N$.
- But kernels have to be chosen properly.

Remarks on kernels

Definition

A kernal K defined on $\mathbb{R}^N$ is a positive-definite kernel if :

- $K : \mathbb{R}^N \times \mathbb{R}^N \longrightarrow \mathbb{R}$
- $\forall (X, Y) \in (\mathbb{R}^N)^2, K(X, Y) = K(Y, X)$
- $\forall M \in \mathbb{N}, \forall (X_1, ..., X_M) \in \mathbb{R}^M, \forall (\alpha_1, ..., \alpha_M) \in \mathbb{R}^M,$

$$\sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j K(X_i, X_j) \geqslant 0$$

This is equivalent to say that the similarity matrix is positive semi-definite.

Remarks on kernels

Definition

A kernal K defined on $\mathbb{R}^N$ is a positive-definite kernel if :

- $K : \mathbb{R}^N \times \mathbb{R}^N \longrightarrow \mathbb{R}$
- $\forall (X,Y) \in (\mathbb{R}^N)^2, K(X,Y) = K(Y,X)$
- $\forall M \in \mathbb{N}, \forall (X_1, ..., X_M) \in \mathbb{R}^M, \forall (\alpha_1, ..., \alpha_M) \in \mathbb{R}^M,$
$$\sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j K(X_i, X_j) \geqslant 0$$

This is equivalent to say that the similarity matrix is positive semi-definite.

Positive definite kernels

Exemples

- Linear kernel :

$$\forall (X,Y) \in (\mathbb{R}^N)^2, K(X,Y) =< X,Y >= X^T Y$$

  - We have symmetry : $< X,Y >=< Y,X >$
  - We have positivity :

$$\sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_i \alpha_j < X_i, X_j >= \| \sum_{i=1}^{M} \alpha_i X_i \|^2 \geqslant 0$$

- The projection function is the identity

Positive definite kernels

Exemples

- Linear kernel :

$$\forall (X, Y) \in (\mathbb{R}^N)^2, K(X, Y) =< X, Y >= X^T Y$$

  - We have symmetry : $< X, Y >=< Y, X >$
  - We have positivity :

$$\sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j < X_i, X_j >= \| \sum_{i=1}^{M} \alpha_i X_i \|^2 \geqslant 0$$

- The projection function is the identity

Positive definite kernels

Exemples

- Polynomial in dimension 1 (if $X, Y \in \mathbb{R}$) :

$$\forall (X, Y) \in \mathbb{R}, K(X, Y) = (1 + XY)^2$$

Positive definite kernels

Exemples

- Polynomial in dimension 1 (if $X, Y \in \mathbb{R}$) :

$$\forall (X,Y) \in \mathbb{R}, K(X,Y) = (1 + XY)^2$$

- The projection function is

$$\phi(x) = (1, x\sqrt{2}, x^2)$$

Positive definite kernels

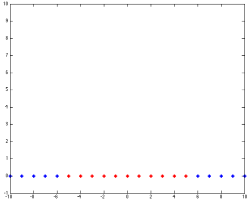
Exemples

- Polynomial in dimension 1 (if $X, Y \in \mathbb{R}$) :

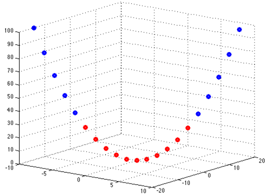
$$\forall (X, Y) \in \mathbb{R}, K(X, Y) = (1 + XY)^2$$

- The projection function is

$$\phi(x) = (1, x\sqrt{2}, x^2)$$



$\Longrightarrow_\phi$



Positive definite kernels

Exemples

- Polynomial in dimension d (si $X, Y \in \mathbb{R}$) :

$$\forall (X,Y) \in (\mathbb{R})^2, K(X,Y) = (1 + XY)^d$$

  The projection function is more tricky to write

Remarks

- If K is a p.s.d, then αK $(\forall \alpha > 0)$ is p.s.d too
- If K_1 et K_2 are two p.s.d, then $K_1 + K_2$ is p.s.d too
- If K_1 et K_2 are two p.s.d, then $K_1 K_2$ is p.s.d too

Positive definite kernels

Exemples

- Polynomial in dimension d (si $X, Y \in \mathbb{R}$) :

$$\forall (X, Y) \in (\mathbb{R})^2, K(X, Y) = (1 + XY)^d$$

The projection function is more tricky to write

Remarks

- If K is a p.s.d, then αK ($\forall \alpha > 0$) is p.s.d too
- If K_1 et K_2 are two p.s.d, then $K_1 + K_2$ is p.s.d too
- If K_1 et K_2 are two p.s.d, then $K_1 K_2$ is p.s.d too

Translation invariant kernels

Invariance by translation

- Kernels such as $K(X_i, X_j) = Q(X_i - X_j)$
- Classical example : Gaussian kernel :

$$K(X_i, X_j) = \exp\left(-\gamma \|X_i - X_j\|^2\right)$$

What is the feature space ϕ associated ?

- Many theoretical works on kernel methods. Most common : Gaussian since it projects data in an infinite dimensional space (best separability)
- Kernel trick : change dot products by kernels

Translation invariant kernels

Invariance by translation

- Kernels such as $K(X_i, X_j) = Q(X_i - X_j)$
- Classical example : Gaussian kernel :

$$K(X_i, X_j) = \exp\left(-\gamma\|X_i - X_j\|^2\right)$$

What is the feature space ϕ associated ?

- Many theoretical works on kernel methods. Most common : Gaussian since it projects data in an infinite dimensional space (best separability)
- Kernel trick : change dot products by kernels

Translation invariant kernels

Invariance by translation

- Kernels such as $K(X_i, X_j) = Q(X_i - X_j)$
- Classical example : Gaussian kernel :

$$K(X_i, X_j) = \exp\left(-\gamma \|X_i - X_j\|^2\right)$$

What is the feature space ϕ associated ?

- Many theoretical works on kernel methods. Most common : Gaussian since it projects data in an infinite dimensional space (best separability)
- Kernel trick : change dot products by kernels

Translation invariant kernels

Invariance by translation

- Kernels such as $K(X_i, X_j) = Q(X_i - X_j)$
- Classical example : Gaussian kernel :

$$K(X_i, X_j) = \exp\left(-\gamma \|X_i - X_j\|^2\right)$$

What is the feature space ϕ associated ?

- Many theoretical works on kernel methods. Most common : Gaussian since it projects data in an infinite dimensional space (best separability)
- Kernel trick : change dot products by kernels

Applications

Many Machine Learning applications use kernels

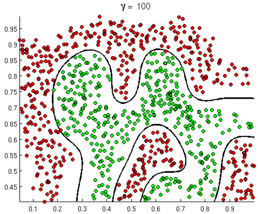
- Kernel ACP
- Perceptron
- Support Vector Machines
- Kernel Regression
- ...

Applications

Many Machine Learning applications use kernels

- Kernel ACP
- Perceptron
- Support Vector Machines
- Kernel Regression
- ...

Enables to easily deal with non linear data



source : http ://openclassroom.stanford.edu/

Links between distance and kernels

From a distance function, one can write it with kernels, and conversely

$$
\begin{aligned}
d(\phi(X),\phi(Y))^2 &= \|\phi(X)-\phi(Y)\|^2 \\
&= \phi(X).\phi(X) - 2\phi(X).\phi(Y) + \phi(Y).\phi(Y) \\
&= K(X,X) - 2K(X,Y) + K(Y,Y)
\end{aligned}
$$

Example : distance function for Gaussian kernel ?

$$K(X,Y) = \exp\left(-\gamma\|X-Y\|^2\right)$$

Links between distance and kernels

From a distance function, one can write it with kernels, and conversely

$$\begin{aligned} d(\phi(X),\phi(Y))^2 &= \|\phi(X)-\phi(Y)\|^2 \\ &= \phi(X).\phi(X) - 2\phi(X).\phi(Y) + \phi(Y).\phi(Y) \\ &= K(X,X) - 2K(X,Y) + K(Y,Y) \end{aligned}$$

Example : distance function for Gaussian kernel ?

$$K(X,Y) = \exp\left(-\gamma\|X-Y\|^2\right)$$

Outline

1. Introduction
2. What is a kernel ?
   - Introduction
   - Positive definite kernels
   - SVM
3. Evaluation
   - Time series

SVM

Main equation

- We aim at maximizing the margin, i.e. find the equation if the hyperplan with maximal margin :

$$\frac{1}{2}\min_{\tilde{\boldsymbol{w}},w_0} \|\tilde{\boldsymbol{w}}\|$$

Under the constraints

$$y_i\big(< \tilde{\boldsymbol{w}}, x_i > +w_0\big) \geqslant 1, \quad \forall i \in \{1,...,N\}$$

The solution depends only on points called support vectors

Important note : only dot products are involved !

SVM

Main equation

- We aim at maximizing the margin, i.e. find the equation if the hyperplan with maximal margin :

$$\frac{1}{2} \min_{\tilde{\boldsymbol{w}}, w_0} \|\tilde{\boldsymbol{w}}\|$$

Under the constraints

$$y_i \big(< \tilde{\boldsymbol{w}}, x_i > + w_0 \big) \geqslant 1, \quad \forall i \in \{1, ..., N\}$$

The solution depends only on points called support vectors

Important note : only dot products are involved !

SVM

Main equation

- We aim at maximizing the margin, i.e. find the equation if the hyperplan with maximal margin :

$$\frac{1}{2} \min_{\tilde{\boldsymbol{w}}, w_0} \|\tilde{\boldsymbol{w}}\|$$

Under the constraints

$$y_i \big(< \tilde{\boldsymbol{w}}, x_i > + w_0 \big) \geqslant 1, \quad \forall i \in \{1, ..., N\}$$

The solution depends only on points called support vectors

Important note : only dot products are involved !

SVM

Main equation

- We aim at maximizing the margin, i.e. find the equation if the hyperplan with maximal margin :

$$\frac{1}{2}\min_{\tilde{\boldsymbol{w}},w_0} \|\tilde{\boldsymbol{w}}\|$$

Under the constraints

$$y_i\big(< \tilde{\boldsymbol{w}}, x_i > + w_0\big) \geqslant 1, \quad \forall i \in \{1, ..., N\}$$

The solution depends only on points called support vectors

Important note : only dot products are involved !

SVM

Main equation

- We aim at maximizing the margin, i.e. find the equation if the hyperplan with maximal margin :

$$\frac{1}{2}\min_{\tilde{\boldsymbol{w}},w_0} \|\tilde{\boldsymbol{w}}\|$$

  Under the constraints

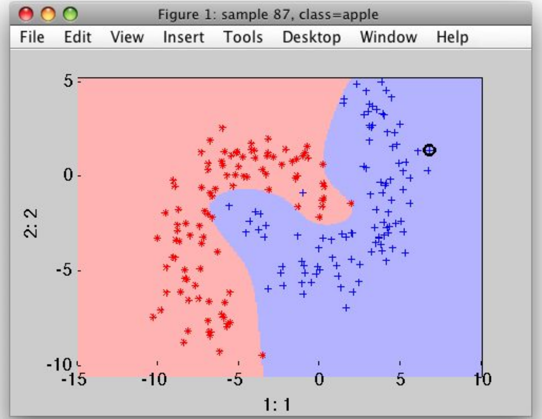
$$y_i\Big(K(\tilde{\boldsymbol{w}},x_i)+w_0\Big) \geqslant 1, \quad \forall i \in \{1,...,N\}$$

The solution depends only on points called support vectors

Important note : only dot products are involved !

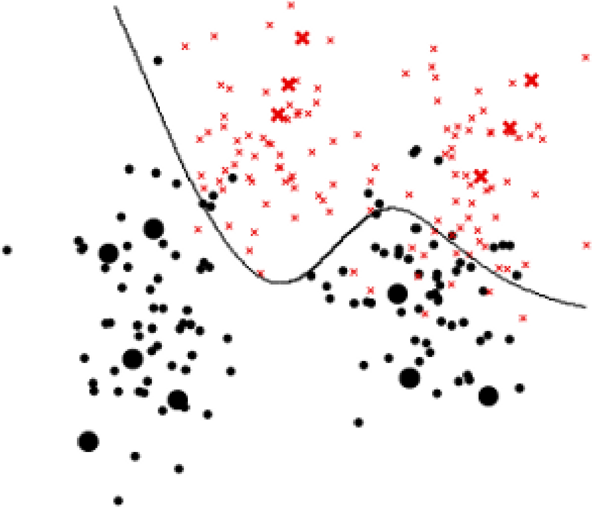
SVM – Illustrations

Illustration (http ://perclass.com/doc/guide/classifiers/svm.html)



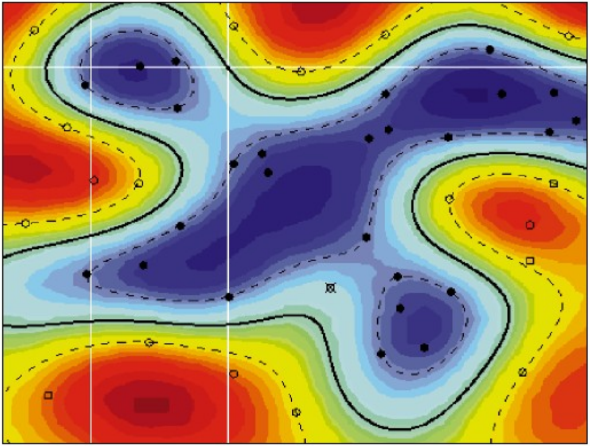
SVM – Illustrations

Illustration (http ://www.ipb.uni-bonn.de/ivm/)



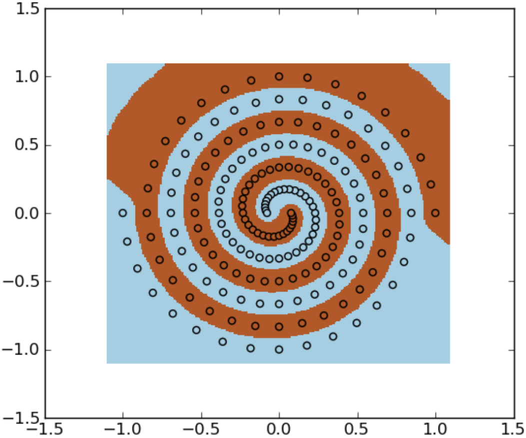
SVM – Illustrations

Illustration (http ://www.dtreg.com/svm.htm)



SVM – Illustrations

Illustration (http ://mlpy.sourceforge.net/)



Support Vector Regression

Principles

- Regression : find a function f that approaches $y \approx f(x)$
- Linear regression : find the vector $\tilde{w} = [\boldsymbol{w}, w_0]^T$ such that $f(x) = < \boldsymbol{w}, x > + w_0$

  $\Longrightarrow$ Least-square
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2} \|\tilde{w}\|^2 \\ \text{with constraints } \ |< \boldsymbol{w}, x > + w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Principles

- Regression : find a function f that approaches $y \approx f(x)$
- Linear regression : find the vector $\tilde{\boldsymbol{w}} = [\boldsymbol{w}, w_0]^T$ such that $f(x) =< \boldsymbol{w}, x > + w_0$

  $\Longrightarrow$ Least-square
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2}\|\tilde{w}\|^2 \\ \text{with constraints } \ |< \boldsymbol{w}, x > + w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Principles

- Regression : find a function f that approaches $y \approx f(x)$
- Linear regression : find the vector $\tilde{\boldsymbol{w}} = [\boldsymbol{w}, w_0]^T$ such that $f(x) =< \boldsymbol{w}, x > +w_0$

  $\Longrightarrow$ Least-square
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2}\|\tilde{w}\|^2 \\ \text{with constraints } \; |< \boldsymbol{w}, x > +w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Principles

- Regression : find a function f that approaches $y \approx f(x)$
- Linear regression : find the vector $\tilde{\boldsymbol{w}} = [\boldsymbol{w}, w_0]^T$ such that $f(x) =< \boldsymbol{w}, x > +w_0$

  $\Longrightarrow$ Least-square
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2}\|\tilde{w}\|^2 \\ \text{with constraints } \; |< \boldsymbol{w}, x > +w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Principles

- Regression : find a function f that approaches $y \approx f(x)$
- Linear regression : find the vector $\tilde{\boldsymbol{w}} = [\boldsymbol{w}, w_0]^T$ such that $f(x) = <\boldsymbol{w}, x> + w_0$

  $\Longrightarrow$ Least-square
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2} \|\tilde{w}\|^2 \\ \text{with constraints } \; |<\boldsymbol{w}, x> + w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Kernel-trick

- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \dfrac{1}{2}\|\tilde{w}\|^2 \\ \text{with constraints } \; |<\boldsymbol{w}, x> + w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

Support Vector Regression

Kernel-trick

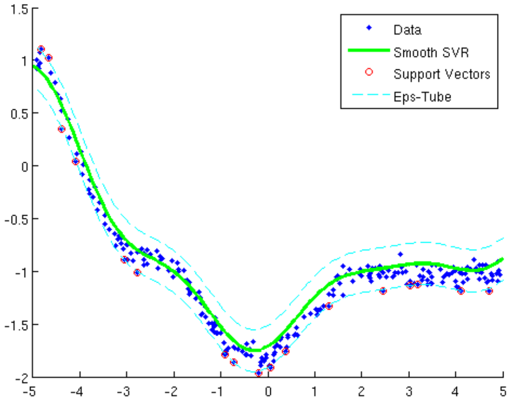
- Linear Support Vector Regression : similarly than SVM , we look for a hyperplane closed to all data :

$$\begin{cases} \min \frac{1}{2}\|\tilde{w}\|^2 \\ \text{with constraints } \; |<\boldsymbol{w}, x> + w_0| < \epsilon \end{cases}$$

The solution depends only on support vectors

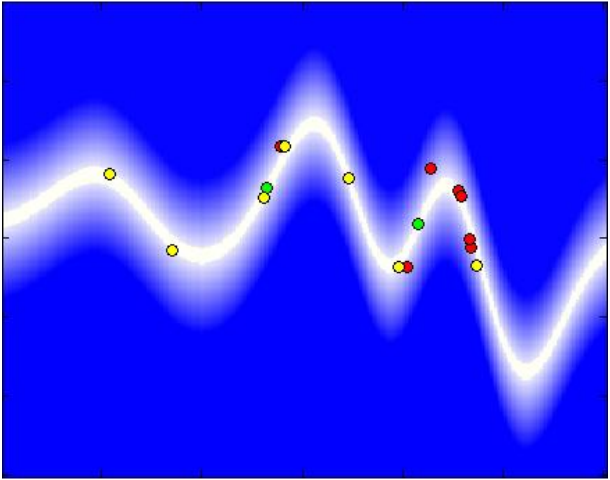
SVR – Illustrations

Illustration (http ://images.1233.tw/support-vector-machine-kernel/)



SVR – Illustrations

Illustration (http ://onlinesvr.altervista.org/Download.html



Evaluation criteria for classification

Precision - Recall - F1-score

- Precision for class i

$$precision(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem assigned to } i}$$

- Overall precision for K classes

$$\frac{\sum_{i=1}^{K} precision(i)}{K}$$

- Nice if $precision \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - F1-score

- Precision for class i

$$precision(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem assigned to } i}$$

- Overall precision for K classes

$$\frac{\sum_{i=1}^{K} precision(i)}{K}$$

- Nice if $precision \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - F1-score

- Precision for class i

$$precision(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem assigned to } i}$$

- Overall precision for K classes

$$\frac{\sum_{i=1}^{K} precision(i)}{K}$$

- Nice if $precision \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - F1-score

- Recall for class i

$$recall(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem in } \ i}$$

- Overall recall for K classes

$$\frac{\sum_{i=1}^{K} recall(i)}{K}$$

- Nice if $recall \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - F1-score

- Recall for class i

$$recall(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem in } \; i}$$

- Overall recall for K classes

$$\frac{\sum_{i=1}^{K} recall(i)}{K}$$

- Nice if $recall \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - F1-score

- Recall for class i

$$recall(i) = \frac{\text{nb elem correctly assigned to } i}{\text{nb elem in } \; i}$$

- Overall recall for K classes

$$\frac{\sum_{i=1}^{K} recall(i)}{K}$$

- Nice if $recall \approx 1$ (but not enough)

Evaluation criteria for classification

Precision - Recall - - F1-score

- Equal combination of recall and precision

$$F1 = 2\frac{precision\ .\ recall}{precision + recall}$$

- If precision is good but not recall (we miss data for some classe) : $F1$ decreases (since recall is small)
- If recall is good but not precision (to many data assigned to a class) : $F1$ decreases (since precision is small)
- If recall and precision are good : $F1 \approx 1$

Evaluation criteria for classification

Precision - Recall - - F1-score

- Equal combination of recall and precision

$$F1 = 2\frac{precision\ .\ recall}{precision + recall}$$

- If precision is good but not recall (we miss data for some classe) : $F1$ decreases (since recall is small)
- If recall is good but not precision (to many data assigned to a class) : $F1$ decreases (since precision is small)
- If recall and precision are good : $F1 \approx 1$

Evaluation criteria for classification

Precision - Recall - - F1-score

- Equal combination of recall and precision

$$F1 = 2\frac{precision\ .\ recall}{precision + recall}$$

- If precision is good but not recall (we miss data for some classe) : $F1$ decreases (since recall is small)
- If recall is good but not precision (to many data assigned to a class) : $F1$ decreases (since precision is small)
- If recall and precision are good : $F1 \approx 1$

Evaluation criteria for classification

Precision - Recall - - F1-score

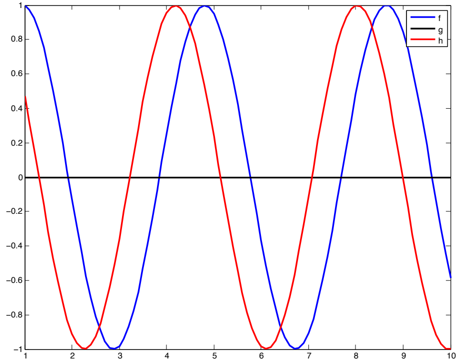
- Equal combination of recall and precision

$$F1 = 2\frac{precision\ .\ recall}{precision + recall}$$

- If precision is good but not recall (we miss data for some classe) : $F1$ decreases (since recall is small)
- If recall is good but not precision (to many data assigned to a class) : $F1$ decreases (since precision is small)
- If recall and precision are good : $F1 \approx 1$

About time series

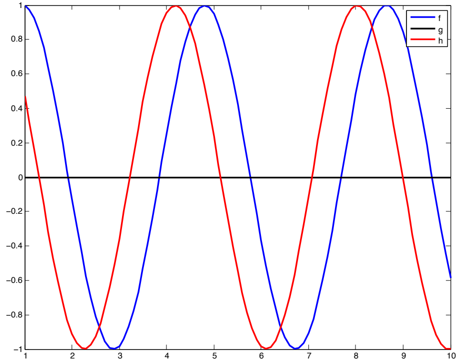
What possibilities for time series ?



- how to compare them ?

About time series

What possibilities for time series ?



- how to compare them ?

Principles : create a path between time series

- Given two series $\boldsymbol{t}_1 = [t_{1,1}, t_{1,2}, ..., t_{1,m}]^T$ and $\boldsymbol{t}_2 = [t_{2,1}, t_{2,2}, ..., t_{2,n}]^T$
- Penalization matrix P of size $m \times n$, each element $P(i,j)$ represents the cost to switch from $t_{1,i}$ to $t_{2,j}$:

$$P(i,j) = |t_{1,i} - t_{2,i}|. \tag{1}$$

Example : $\boldsymbol{t}_1 = (2,3,6,9,5,4,3)$(card $= 7$) $\boldsymbol{t}_2 = (1,2,5,9,4,2)$ (card $=$ 6)

$P(i,j) =$ (2)

	t_2						
t_1	**1**	**2**	**5**	**9**	**4**	**2**	*i=*
2	1	0	3	7	2	0	*1*
3	2	1	2	6	1	1	*2*
6	5	4	1	3	2	4	*3*
9	8	7	4	0	5	7	*4*
5	4	3	0	4	1	3	*5*
4	3	2	1	5	0	2	*6*
3	2	1	2	6	1	1	*7*
j =	*1*	*2*	*3*	*4*	*5*	*6*	

Elastic distances : DTW

Principles : create a path between time series

- Given two series $\boldsymbol{t}_1 = [t_{1,1}, t_{1,2}, ..., t_{1,m}]^T$ and $\boldsymbol{t}_2 = [t_{2,1}, t_{2,2}, ..., t_{2,n}]^T$
- Penalization matrix P of size $m \times n$, each element $P(i,j)$ represents the cost to switch from $t_{1,i}$ to $t_{2,j}$:

$$P(i,j) = |t_{1,i} - t_{2,i}|. \qquad (1)$$

Example : $\boldsymbol{t}_1 = (2,3,6,9,5,4,3)$(card = 7) $\boldsymbol{t}_2 = (1,2,5,9,4,2)$ (card = 6)

$$P(i,j) = \begin{array}{c|c|cccccc|c}
 & & & & t_2 & & & & \\
 & & 1 & 2 & 5 & 9 & 4 & 2 & i= \\
\hline
 & 2 & 1 & 0 & 3 & 7 & 2 & 0 & 1 \\
 & 3 & 2 & 1 & 2 & 6 & 1 & 1 & 2 \\
 & 6 & 5 & 4 & 1 & 3 & 2 & 4 & 3 \\
t_1 & 9 & 8 & 7 & 4 & 0 & 5 & 7 & 4 \\
 & 5 & 4 & 3 & 0 & 4 & 1 & 3 & 5 \\
 & 4 & 3 & 2 & 1 & 5 & 0 & 2 & 6 \\
 & 3 & 2 & 1 & 2 & 6 & 1 & 1 & 7 \\
\hline
 & j= & 1 & 2 & 3 & 4 & 5 & 6 &
\end{array} \qquad (2)$$

Elastic distances : DTW

Principles : create a path between time series

- Given two series $\boldsymbol{t}_1 = [t_{1,1}, t_{1,2}, ..., t_{1,m}]^T$ and $\boldsymbol{t}_2 = [t_{2,1}, t_{2,2}, ..., t_{2,n}]^T$
- Penalization matrix P of size $m \times n$, each element $P(i,j)$ represents the cost to switch from $t_{1,i}$ to $t_{2,j}$:

$$P(i,j) = |t_{1,i} - t_{2,i}|. \tag{1}$$

Example : $\boldsymbol{t}_1 = (2,3,6,9,5,4,3)$(card = 7) $\boldsymbol{t}_2 = (1,2,5,9,4,2)$ (card = 6)

$P(i,j) =$ (2)

	t_2: **1**	**2**	**5**	**9**	**4**	**2**	*i=*
t_1: **2**	**1**	**0**	3	7	2	0	*1*
3	2	**1**	2	6	1	1	*2*
6	5	4	**1**	3	2	4	*3*
9	8	7	4	**0**	5	7	*4*
5	4	3	0	4	**1**	3	*5*
4	3	2	1	5	**0**	2	*6*
3	2	1	2	6	1	**1**	*7*
j =	*1*	*2*	*3*	*4*	*5*	*6*	

Elastic distances : DTW

Principles : create a path between time series

- A *warping path* has to respect $W = w_1, ..., w_K, K \in [max(m,n), m+n-1]$:
  - $w_1 = (1,1)$ and $w_K = (m,n)$ (start and end points) ;
  - w_{i+1} is connected w_i for all $i \in [1, K-1]$ (continuity of the path) ;
  - $(w_{i+1} - w_i)(w_i - w_{i-1}) > 0$ for $i \in [2, K-1]$ (monotony : no backward path).

Dynamic Time Warping : extract of the minimal cost path to switch from series $\boldsymbol{t}_1$ to $\boldsymbol{t}_2$:

$$D_{dtw}(\boldsymbol{t}_1, \boldsymbol{t}_2) = \min \frac{\sum_{k=1}^{K} P(w_k)}{K}. \quad (3)$$

Elastic distances : DTW

Principles : create a path between time series

- A *warping path* has to respect $W = w_1, ..., w_K, K \in [max(m,n), m+n-1]$:
  - $w_1 = (1,1)$ and $w_K = (m,n)$ (start and end points) ;
  - w_{i+1} is connected w_i for all $i \in [1, K-1]$ (continuity of the path) ;
  - $(w_{i+1} - w_i)(w_i - w_{i-1}) > 0$ for $i \in [2, K-1]$ (monotony : no backward path).

Dynamic Time Warping : extract of the minimal cost path to switch from series $\boldsymbol{t}_1$ to $\boldsymbol{t}_2$:

$$D_{dtw}(\boldsymbol{t}_1, \boldsymbol{t}_2) = \min \frac{\sum_{k=1}^{K} P(w_k)}{K}. \quad (3)$$

Elastic distances : DTW

Dynamic Time Warping : extraction of the minimal cost path to switch from series $\boldsymbol{t}_1$ to $\boldsymbol{t}_2$:

$$D_{dtw}(\boldsymbol{t}_1, \boldsymbol{t}_2) = \min \frac{\sum_{k=1}^{K} P(w_k)}{K}. \tag{4}$$

$P =$

		t_2 1	2	5	9	4	2	i=
t_1	**2**	**1**	**0**	3	7	2	0	*1*
	3	2	**1**	2	6	1	1	*2*
	6	5	4	**1**	3	2	4	*3*
	9	8	7	4	**0**	5	7	*4*
	5	4	3	0	4	**1**	3	*5*
	4	3	2	1	5	**0**	2	*6*
	3	2	1	2	6	1	**1**	*7*
	$j =$	*1*	*2*	*3*	*4*	*5*	*6*	

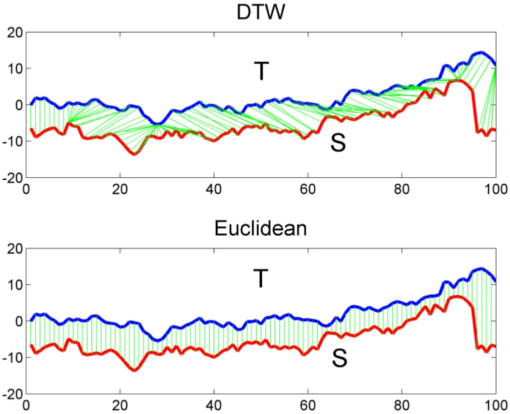
$D =$

		t_2 1	2	5	9	4	2	i=
t_1	**2**	**1**	**1**	4	11	13	13	*1*
	3	3	**2**	3	9	10	11	*2*
	6	8	6	**3**	6	8	12	*3*
	9	16	13	7	**3**	8	15	*4*
	5	20	16	7	7	**4**	7	*5*
	4	23	18	8	12	**4**	6	*6*
	3	25	19	10	14	5	**5**	*7*
	$j =$	*1*	*2*	*3*	*4*	*5*	*6*	

(5)

DTW – Illustrations

Illustration (Cassisi et al, 2012)



Average in the DTW distance (useful for k-means)

No analytical formulae

- Average of $[\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$ w.r.t distance D :

$$\boldsymbol{\mu} = \text{arg}\min_{\boldsymbol{x}^*} \sum_{i=1}^{N} D^2(\boldsymbol{x}_i, \boldsymbol{x}^*)$$

Average in the DTW distance (useful for k-means)

No analytical formulae

- For DTW :

$$\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{t}^*} \sum_{i=1}^{N} DTW^2(\boldsymbol{t}_i, \boldsymbol{t}^*)$$

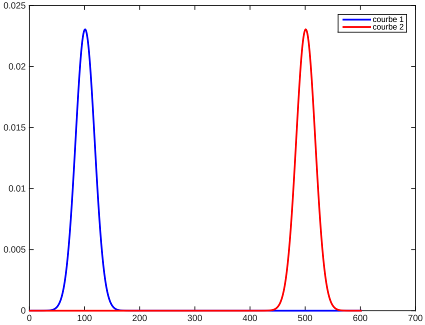
Average in the DTW distance (useful for k-means)

No analytical formulae

- For DTW :

$$\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{t}^*} \sum_{i=1}^{N} DTW^2(\boldsymbol{t}_i, \boldsymbol{t}^*)$$

Initial curves



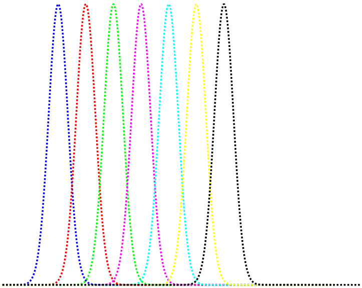
Average in the DTW distance (useful for k-means)

No analytical formulae

- For DTW :

$$\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{t}^*} \sum_{i=1}^{N} DTW^2(\boldsymbol{t}_i, \boldsymbol{t}^*)$$

Possible averages w.r.t. DTW



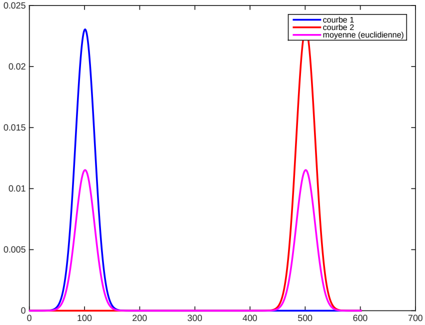
Average in the DTW distance (useful for k-means)

No analytical formulae

- For DTW :

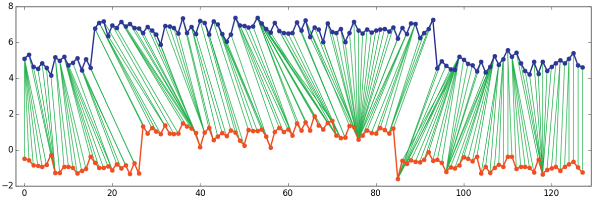
$$\boldsymbol{\mu}_t = \arg\min_{\boldsymbol{t}^*} \sum_{i=1}^{N} DTW^2(\boldsymbol{t}_i, \boldsymbol{t}^*)$$

Eucledian mean



Regularized DTW

Illustration DTW : pathological alignments



Regularized DTW

Illustration DTW : pathological alignments