Kernel design for non-vector -data

Mustererkennung mit
Support-Vektor-Maschinen (SVM)
WS 02/03

Pattern recognition from learning examples

Try to estimate a function $f : R^N \to \{\pm 1\}$ using training data—that is, N-dimensional patterns **x**$_i$ and class labels y$_i$,

$$(x_1, y_1), \ldots, (x_l, y_l) \in R^N \times \{\pm 1\}$$

such that f will correctly classify new examples(**x**,y), which were generated from the same underlying probability distribution **P**(**x**,y) as the training data.

Hyperplane classifiers

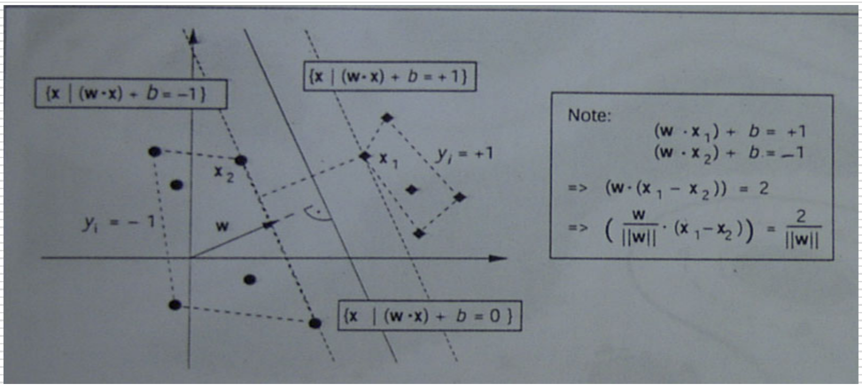
SV classifiers are based on the class of hyperplanes $(w \cdot x) + b = 0 \quad w \in R^N, b \in R$

Corresponding to decision functions

$$f(x) = sign\ ((w \cdot x) + b)$$

Linear separable data.

Given l training data that is linear separable, looks for the separating hyperplane with largest margin.



Suppose training patterns labeled +1 is above a hyperplane, patterns labeled -1 is below it.

$x_i \cdot w + b \geq +1 (y_i = +1)$ margin is $2 / \|w\|$

$x_i \cdot w + b \leq -1 (y_i = -1)$

i.e. minimizing $\|w\|^2$,subject to a set of l constraints $y_i(x_i \cdot w + b) - 1 \geq 0$

solution

Equivalent with maximizing (quadratic programming problem)

$$L_D = \sum_i a_i - \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i \cdot x_j$$

under constraints $0 \le a_i \le C$ $\sum_i a_i y_i = 0$

solution is of the form $w = \sum_i a_i y_i x_i$

decision function $f(x) = sign(\sum_i a_i y_i (x \cdot x_i) + b)$

crucial property: both quadratic programming problem and the decision function depend only on dot products between patterns

kernels

A kernel is a function k such that for all $a, b \in \mathrm{A}$

$$k(a,b) = \phi(a) \cdot \phi(b).$$

where Φ is a mapping from A to a feature space X.

Compute values of kernel function between original data to replace the dot products between the maps of the original data in other dot product space.

Decision function become:

$$f(x) = sign(\sum_i a_i y_i K(x, x_i) + b)$$

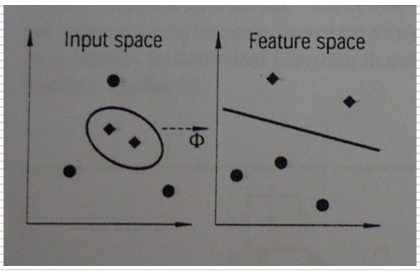
Advantages of kernel

The classification is take place in other space, but the result can be concluded in the original space without knowing the mapping.

Avoid the expensive computation in computing the mapping function and the dot products in high-dimensional space.

nonlinear classifiers.

Nonlinear classifiers



Map the training data nonlinearly into a higher-dimensional feature space via Φ, and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. By the use of kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space.

$$f(x) = sign\left(\sum_i a_i y_i K(x, x_i) + b\right)$$

Nonlinear decision function, which is determined by the kernel.

Example kernels

Polynomial kernel:

$$k(x, y) = (x \cdot y)^d$$

e.g. d=2 and $x, y \in R^2$ then Φ(x)= $(x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$

More generally, it can be prove that for every kernel that gives rise to a positive semi-definit matrix $(k(x_i, x_j))_{ij}$, it can be constructed a map Φ, such that $k(x, y) = (\Phi(x) \cdot \Phi(y))$.

Example kernels (cont)

Radial basis function kernels:

$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$

Sigmoid kernels:

$$k(x, y) = \tanh(\kappa(x \cdot y) + \Theta)$$

Non-vector-data classification

Not all data comes naturally as vectors.

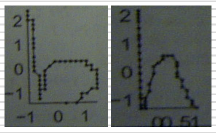
Data could be any “structured objects” such as sequences of different lengths, trees, piece of text, chemical materials, etc.

To apply linear method to such data:

- Kernels for non-vector-data
  - dynamic time warping kernel
- Kernels for feature mappings
  - casebased features
  - text data
  - chemical materials

On-line HWR

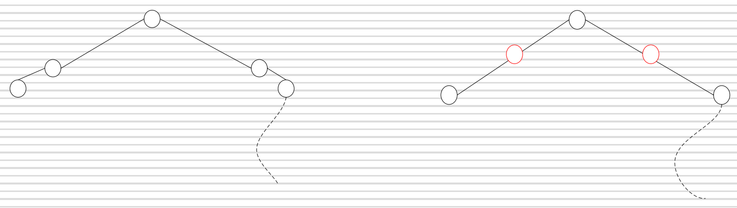
In On-line HWR patterns are defined as variable-size sequence of feature vectors that may have been distorted in particular ways, each vector computed from sampled coordinates of the pen tip curve.



```
####### character no. 9:
 U  1  # label & no. components
    25         (3 199) (0 177) (0 157) (0 146) (0 123) (6 95)
      (9 82) (16 66) (29 39) (43 18) (49 12) (66 3)
      (84 0) (92 1) (110 5) (124 16) (138 37) (150 86)
      (153 127) (152 166) (148 182) (146 191) (145 197) (140 197)
      (135 191)
####### character no. 13:
 V  1  # label & no. components
    18         (0 199) (10 167) (12 151) (23 108) (35 59) (48 19)
      (53 6) (66 0) (70 7) (79 37) (84 57) (95 103)
      (110 141) (117 159) (133 185) (142 193) (152 199) (164 195)
```

Since SVM techniques are based on feature spaces with fixed dimension, SVM method can’t apply to those data directly.

On-line HWR data



compute intermediate sample points such that the length of the vector sequences is equal.

Dynamic time warping

Given 2 vector sequences

$$\Im = (t_1,\ldots, t_{N_\Im}) \qquad \Re = (r_1,\ldots, r_{N_\Re}) \qquad t_i, r_i \in R^F$$

Define *alignment path* $\phi = (\phi(1),\ldots,\phi(N))$ each

$$\phi(n) = (\phi_\Im(n), \phi_\Re(n)) \in \{1,\ldots,N_\Im\} \times \{1,\ldots,N_\Re\}$$

Define local distance e.g. $d(t_i, r_j) = \left\| t_i - r_j \right\|^2$

The distance of 2 vector sequences wrt a particular Φ

$$D_\phi(\Im, \Re) = \frac{1}{N} \sum_{n=1}^{N} d(t_{\phi_\Im(n)}, r_{\phi_\Re(n)})$$

Dynamic time warping (cont)

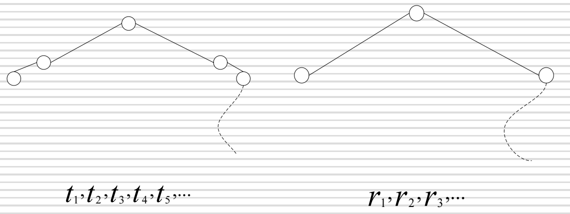
The distance of the 2 vector sequences:
the smallest distance with respect to any alignment path.

$$D(\Im,\Re) = D_{\phi^*}(\Im,\Re) = \min_{\phi}\{D_{\phi}(\Im,\Re)\}$$

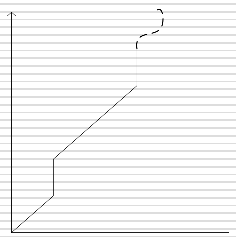
That alignment path is called *Viterbi path* ϕ^*.

It is convenient to model Φ as a sequence of local transitions.
Use *Sakoe-Chiba transitions*.
Only allow forward steps of size 1 in T, or R or both of them.
i.e. Φ(n+1)-Φ(n) equals (1,0),(0,1) or (1,1).

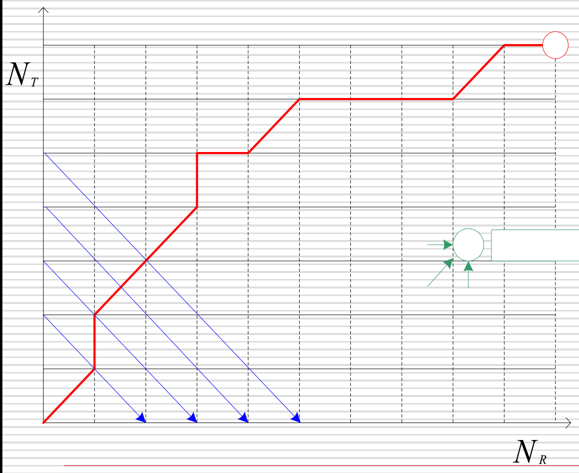
Dynamic time warping (an example)



$$\phi^* = ((1,1), (2,1), (3,2), (4,3), (5,3), \dots)$$



Compute the distance of 2 vector sequences and the *Viterbi path*



$(t_1,\dots,t_i)\,(r_1,\dots,r_j)$

$d(t_1,r_1)$

$cell(1,1)+d(t_1,r_2)$

$(\quad)\quad d(t_i,r_j)$

Apply SVM for On-line HWR data

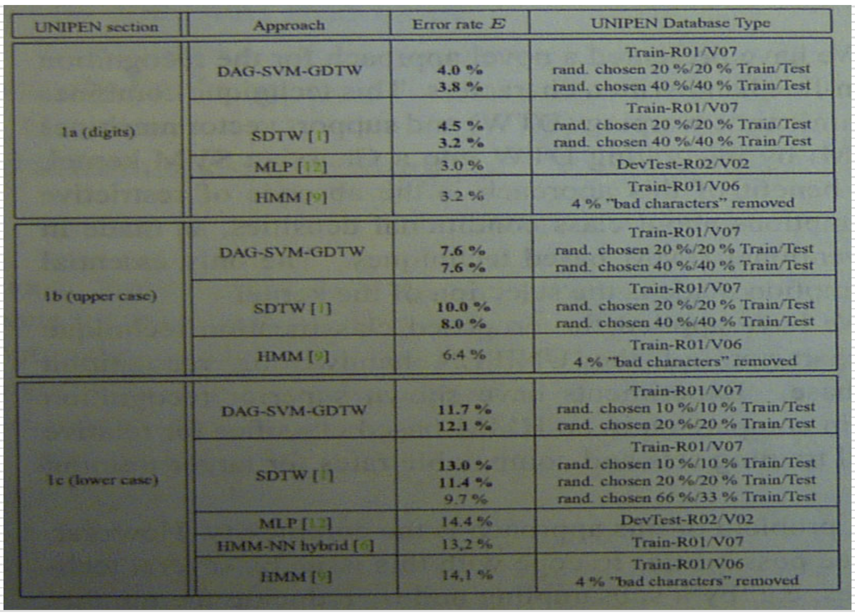
Gaussian Dynamic time warping kernel

radial basis function

$$k(x,y)=\exp\left(-\gamma\|x-y\|^2\right)$$

$$K(p_i,p_j)=\exp(-\gamma D(p_i,p_j))$$

Disadvantage: positive definiteness for GDTW kernel can’t be proved, so the solution of the optimization algorithm is not guaranteed to be the global optimum.



UNIPEN section	Approach	Error rate E	UNIPEN Database Type
1a (digits)	DAG-SVM-GDTW	**4.0 %** **3.8 %**	Train-R01/V07 rand. chosen 20 %/20 % Train/Test rand. chosen 40 %/40 % Train/Test
	SDTW [1]	**4.5 %** **3.2 %**	Train-R01/V07 rand. chosen 20 %/20 % Train/Test rand. chosen 40 %/40 % Train/Test
	MLP [12]	3.0 %	DevTest-R02/V02
	HMM [9]	3.2 %	Train-R01/V06 4 % "bad characters" removed
1b (upper case)	DAG-SVM-GDTW	**7.6 %** **7.6 %**	Train-R01/V07 rand. chosen 20 %/20 % Train/Test rand. chosen 40 %/40 % Train/Test
	SDTW [1]	**10.0 %** **8.0 %**	Train-R01/V07 rand. chosen 20 %/20 % Train/Test rand. chosen 40 %/40 % Train/Test
	HMM [9]	6.4 %	Train-R01/V06 4 % "bad characters" removed
1c (lower case)	DAG-SVM-GDTW	**11.7 %** **12.1 %**	Train-R01/V07 rand. chosen 10 %/10 % Train/Test rand. chosen 20 %/20 % Train/Test
	SDTW [1]	**13.0 %** **11.4 %** 9.7 %	Train-R01/V07 rand. chosen 10 %/10 % Train/Test rand. chosen 20 %/20 % Train/Test rand. chosen 66 %/33 % Train/Test
	MLP [12]	14.4 %	DevTest-R02/V02
	HMM-NN hybrid [6]	13,2 %	Train-R01/V07
	HMM [9]	14,1 %	Train-R01/V06 4 % "bad characters" removed

Case-based features

Often there are natural matching functions that may apply to structured objects.
Map a pair of objects to a real-valued score.

Create feature vector:

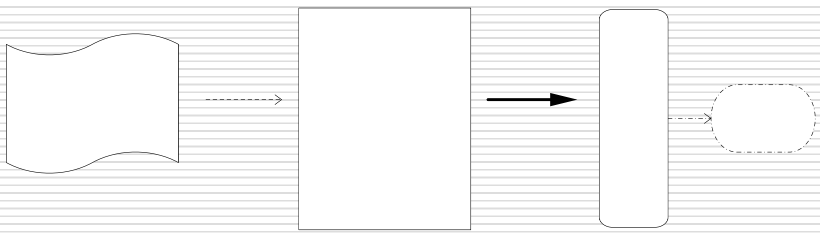
Given any function f:$\mathcal{A}\times\mathcal{A}\rightarrow$R, and an indexed set of cases a1,…,an a possible feature space mapping is:

$$\phi(b) = < f(a_1, b), \ldots, f(a_n, b) >$$

The feature vector have been computed explicitly.
No computational advantage in using a kernel.

Text data

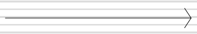
Text is considered as a sparse vector by using a dictionary, each dimension for each possible word.
Each entry is defined as the frequence of the words in the dictionary that have appeared in the text.



The result vectors is a sparse vector with only a few non-zero elements.

Sparse vector kernels

With an efficient sparse representation the dot-product of two sparse vectors can be computed in a time proportional to the total number of non-zero elements in the two vectors.



$$\begin{matrix}0\\1\\0\\0\\0\\0\\0\end{matrix} \cdot \begin{matrix}0\\1\\0\\1\\0\\1\\0\end{matrix} =1$$

Chemical materials

String representation:

```
167780
CCOC(=O)C(CC(C)C)NC(=O)C(Cc1ccc(cc1)N(CCCl)CCCl)NC(=O)C 26;
C-C-C-C-C=O,C-C-C-C-N-C-C-C-c:c:c:c-N-C-C-Cl,C-C-C-C-N-C-C-N-C-C,C-C-C-C-N-C-C-N-
    C=O,C-C-C-C-N-C=O,C-C-N-C-C-N-C-C=O,C-C-N-C-C-c:c:c:c:c:c,C-C-O-C-C-C-C-C,C-C-
    O-C-C-N-C-C-N-C-C,C-C-O-C-C-N-C-C-N-C=O,C-C-O-C-C-N-C=O,C-C-O-C=O,Cl-C-C-N-C-C-
    Cl,Cl-C-C-N-c:c:c:c-C-C-C-N-C-C-O-C-C,Cl-C-C-N-c:c:c:c-C-C-C=O,Cl-C-C-N-
    c:c:c:c-C-C-N-C-C,Cl-C-C-N-c:c:c:c-C-C-N-C=O,Cl-C-C-N-c:c:c:c:c:c,O=C-C-C-
    c:c:c:c:c:c,O=C-C-N-C-C-C-c:c:c:c-N-C-C-Cl,O=C-C-N-C-C-N-C=O,O=C-C-N-C=O,O=C-N-
    C-C-c:c:c:c:c:c,c:c:c:c:c:c-C-C-C-N-C-C-C-C,c:c:c:c:c:c-C-C-C-N-C-C-O-C-
    C,c:c:c:c:c:c-C-C-C-N-C-C=O,
```

For some chemical material classes try to find the characteristic set of sub-fragments for each class, combine them to form a dictionary.

New materials is mapped to a vector by using the dictionary.

Conclusion

Apply kernel method to structured objects:

Compute a kernel function to measure similarity between 2 objects.

Construct feature vectors explicitly.

literature

Claus Bahlmann, Bernard Haasdonk and Hans Burkhardt. On-line Handwriting Recognition with Support Vector Machines—A Kernel Approach

Marti Hearst. SVMs—a practical consequence of learning theory

Chris Watkins. Dynamic Alignment Kernels

Christopher J.C.Burges. A Tutorial on Support Vector Machines for Pattern Recognition