02457 Non-linear signal processing

2016 - Lecture 10

Technical University of Denmark
DTU Compute, Kongens Lyngby, Denmark

Outline lecture 10

Nonparametric modeling
- Parzen windows
- Nearest neighbor methods
- Local linear models

Kernel methods
- Learning in high-dimensional spaces d > N
- Dual representation
- Properties of kernel matrices

Gaussian process prior
- Smoothness -> correlation between neighbors
- Gaussian prior on functions

Support Vector Machines
- Max margin classification
- Dual representation
- Sparse approximation

Nonparametric models

- We generally characterize models as
  - Parametric: e.g., normal distribution as a density model
  - Semi-parametric: variable parametrization e.g. a neural network or Gaussian mixture with variable number of hidden variables or components

$$p(x|D) = \int p(x|\theta)p(\theta|D)d\theta.$$

  - Nonparametric: Predictive distribution depends on all data!

Kernel density estimators (extreme Gaussian mixture....)

A training set of N data points $D = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\}$ is extrapolated to test points $\mathbf{x}$

$$p(\mathbf{x}|D, h) \;=\; \frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}|\mathbf{x}_n, h)$$

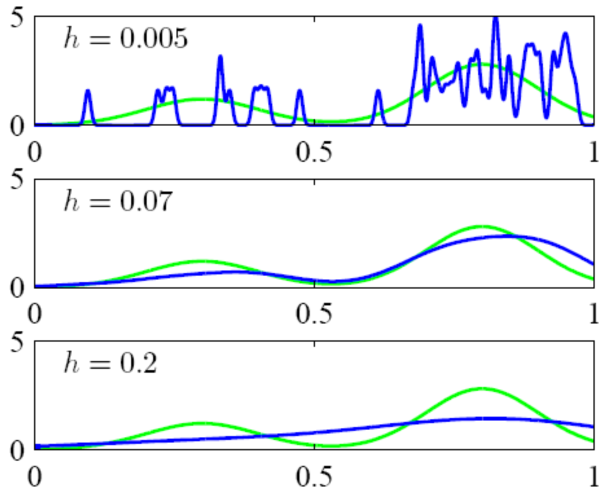
p(**x**) is normalized

with a 'kernel' $k(\mathbf{x}|\mathbf{x}_n, h)$ given eg. by

$$k(\mathbf{x}|\mathbf{x}_n, h) \;=\; \left(\frac{1}{2\pi h^2}\right)^{d/2} \exp\left(-\frac{1}{2h^2}(\mathbf{x}-\mathbf{x}_n)^2\right)$$

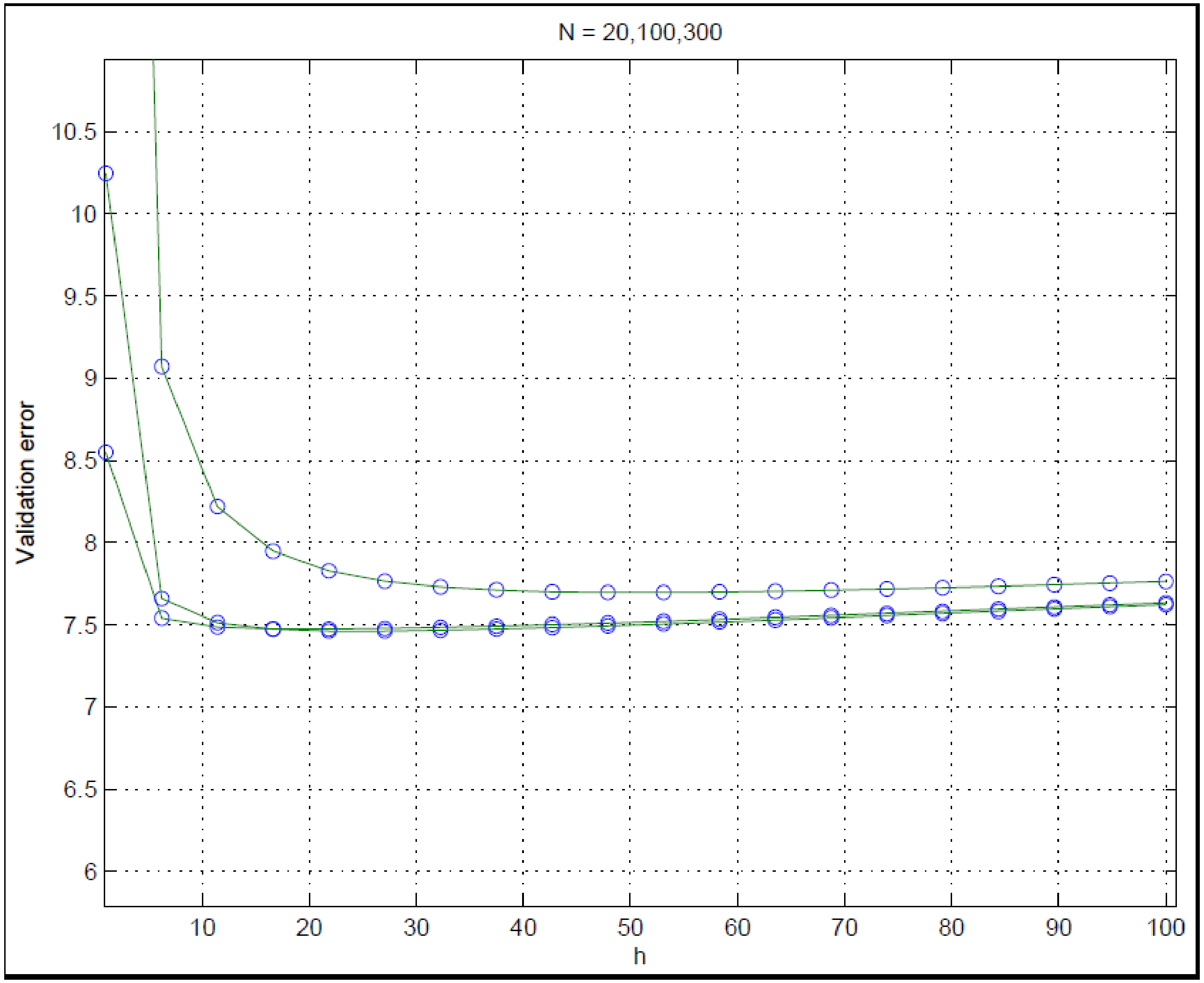
Kernel density estimators

The scale parameter h is a smoothing control parameter,
if h is small we roughly get a set of local 'delta functions',
if h is large we get a nearly uniform distribution.

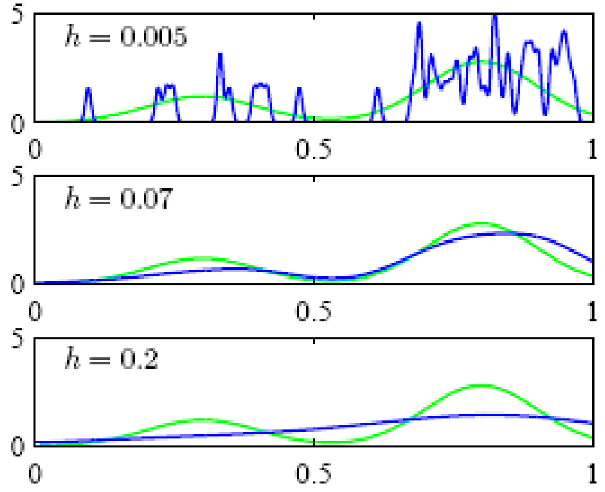


h can be chosen by cross-validation. Both direct search and gradient search is available

Estimation of smoothing width (h)



Relation between kernel methods and nearest neighbors



Let the kernel be unity in a box of volume one

$$k(\mathbf{u}) = \begin{cases} 1 & \text{if } \forall_i \ |u_i| < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Determine a width h so that the total number of points in the box is

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

Thus the density estimator is given by

$$p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

where the volume in D-dimensional space is h^D

Nearest neighbor methods

Remind that the probability of a certain region is

$$P(\mathcal{R}, \mathbf{x}) = \int_{\mathcal{R}(\mathbf{x})} p(\mathbf{x}')d\mathbf{x}'$$

If we have a large data set N then the number of points K in the region will be given approximately as $K \approx NP$. If the region is nor too big we can approximate $P \approx p(\mathbf{x})V$, with V being

$$V(\mathbf{x}) = \int_{\mathcal{R}(\mathbf{x})} d\mathbf{x}'$$

$$p(\mathbf{x}) \approx \frac{K}{NV}$$

Supervised learning: Signal detection

Let us use the kernel density estimator for the individual classes

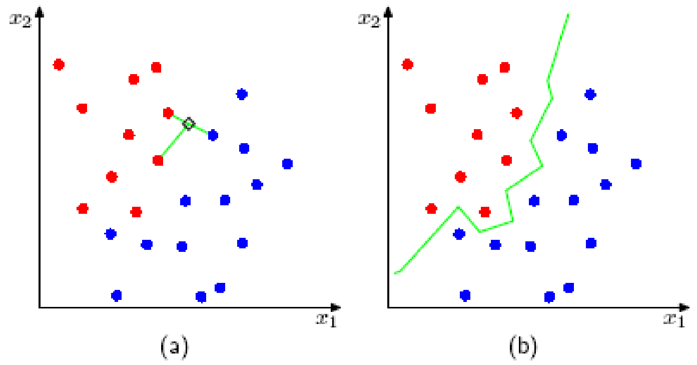
$$p(\mathbf{x}|C_k) \approx \frac{K_k}{N_k V}$$

and for the total density

$$p(\mathbf{x}) \approx \frac{K}{NV}$$

use frequencies to estimate the prior probabilities

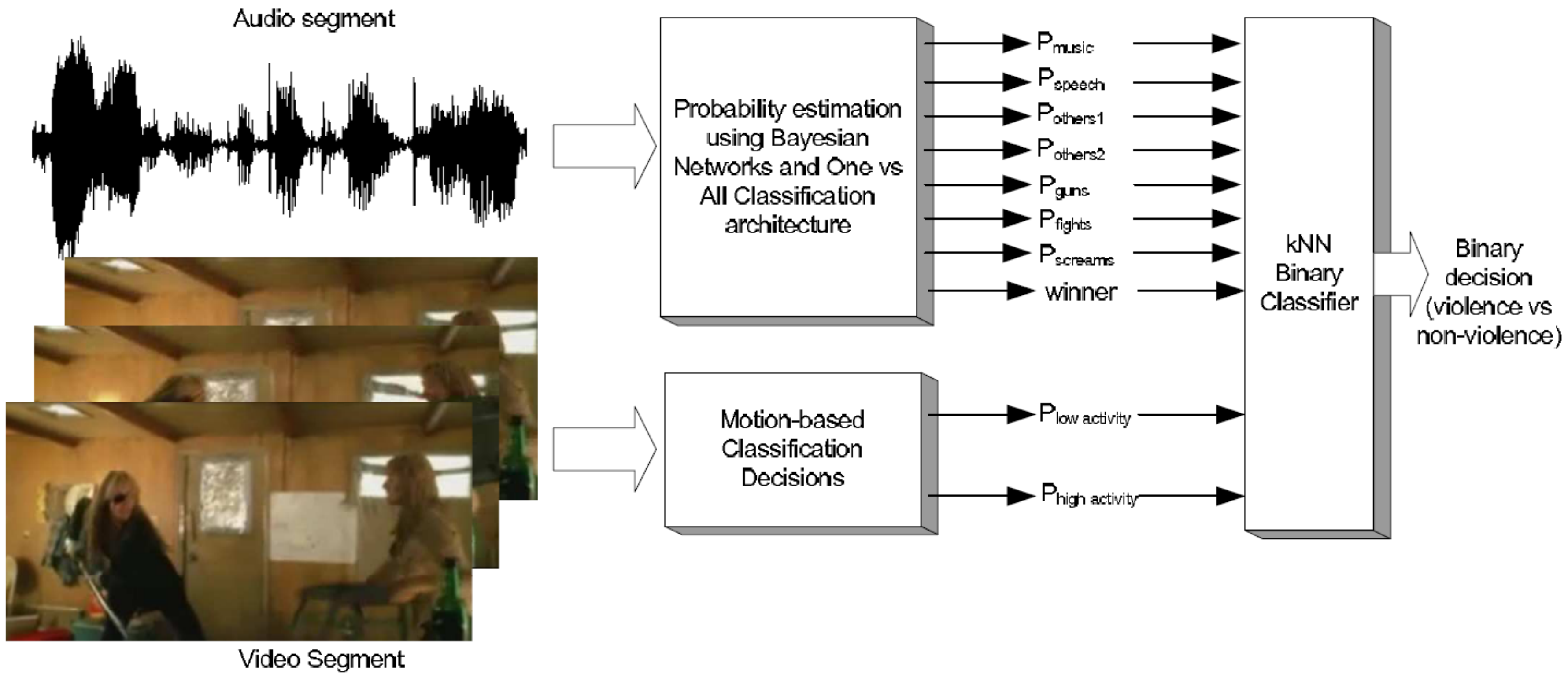
$$p(C_k) \approx \frac{N_k}{N}$$



Then we can get the posteriors from Bayes' theorem

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} \approx \frac{K_k}{K}$$

Example: Audio-visual integration



Theodoros Giannakopoulos, Alexandros Makris, Dimitrios I. Kosmopoulos, Stavros J. Perantonis, Sergios Theodoridis. Audio-Visual Fusion for Detecting Violent Scenes in Videos. In Proceedings of SETN'2010. pp.91-100

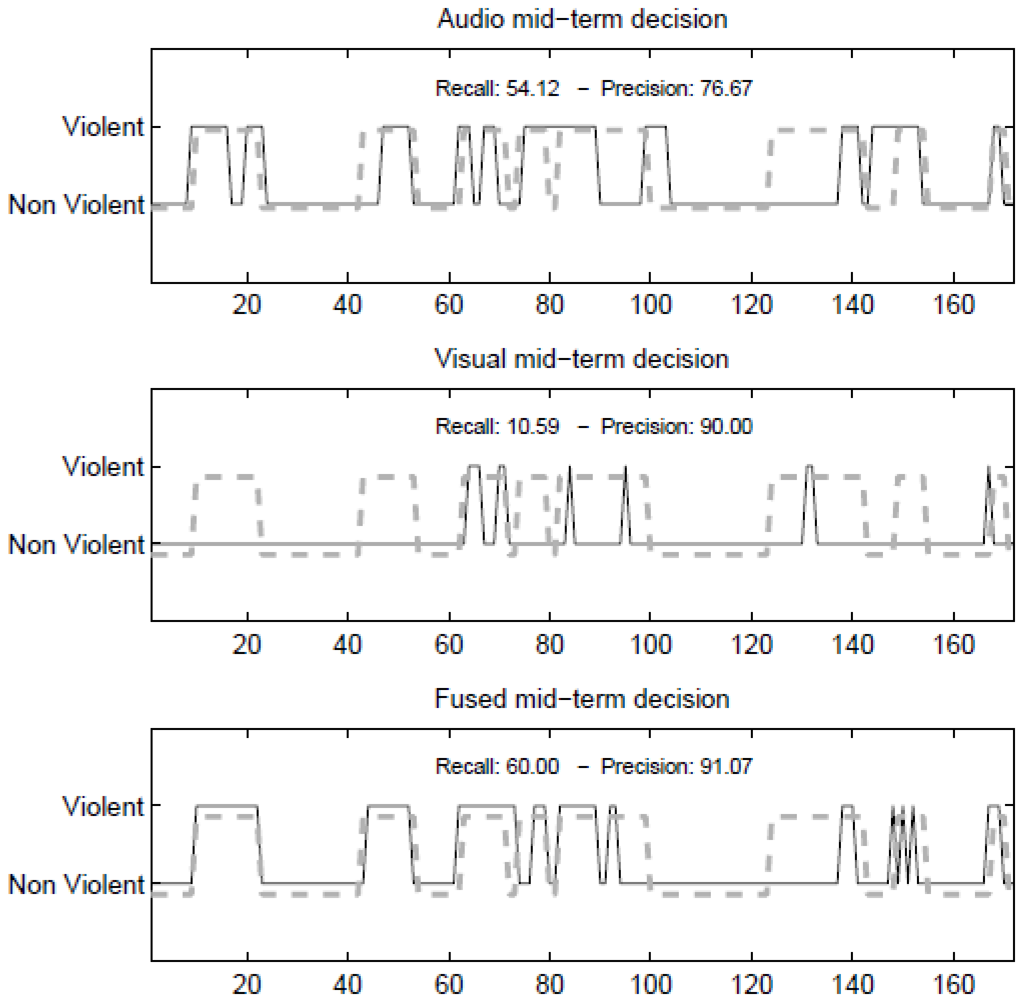


Fig. 4. Violence detection example for a movie audio stream.

Example: Medical decision making

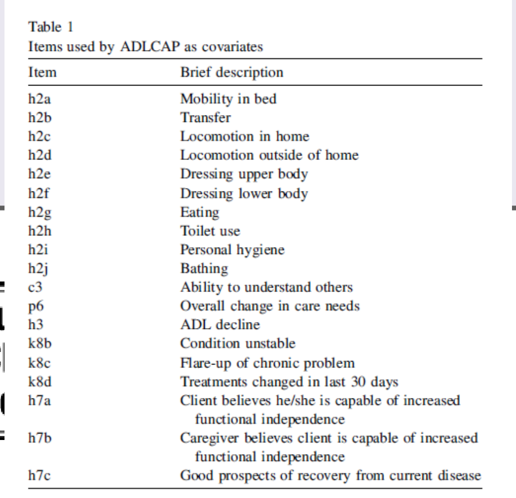
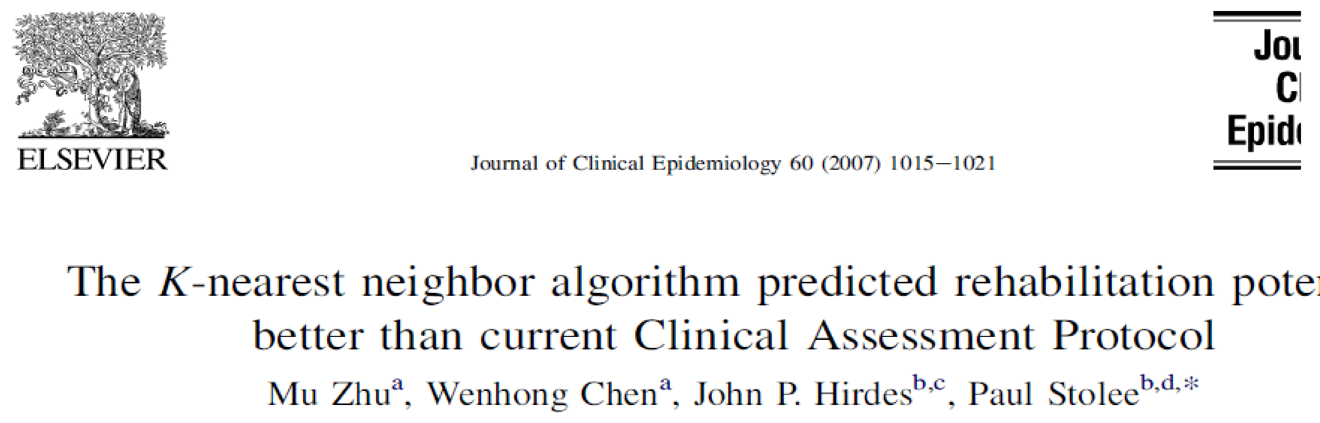


Table 1
Items used by ADLCAP as covariates

Item	Brief description
h2a	Mobility in bed
h2b	Transfer
h2c	Locomotion in home
h2d	Locomotion outside of home
h2e	Dressing upper body
h2f	Dressing lower body
h2g	Eating
h2h	Toilet use
h2i	Personal hygiene
h2j	Bathing
c3	Ability to understand others
p6	Overall change in care needs
h3	ADL decline
k8b	Condition unstable
k8c	Flare-up of chronic problem
k8d	Treatments changed in last 30 days
h7a	Client believes he/she is capable of increased functional independence
h7b	Caregiver believes client is capable of increased functional independence
h7c	Good prospects of recovery from current disease



ELSEVIER

Journal of Clinical Epidemiology 60 (2007) 1015–1021

Jou
Cl
Epid

The *K*-nearest neighbor algorithm predicted rehabilitation potential better than current Clinical Assessment Protocol

Mu Zhu[a], Wenhong Chen[a], John P. Hirdes[b,c], Paul Stolee[b,d,*]

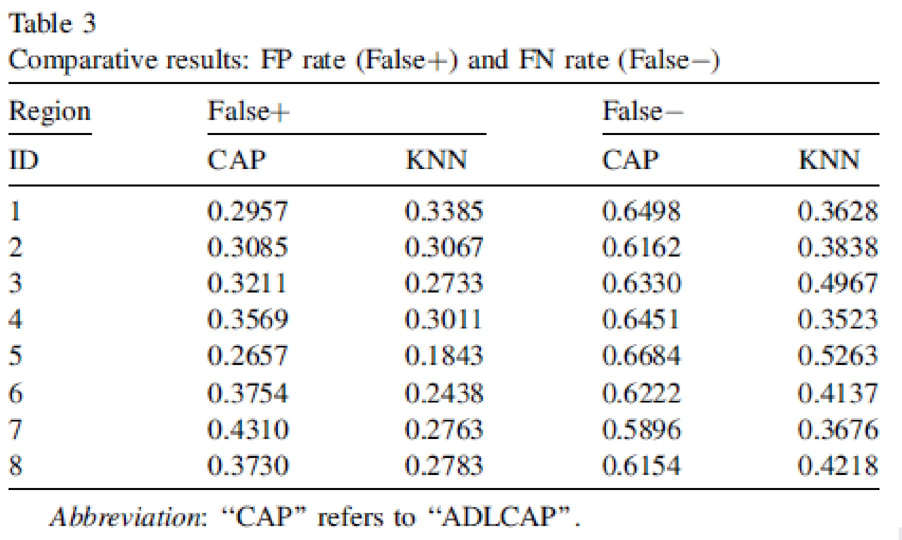


Table 3
Comparative results: FP rate (False+) and FN rate (False−)

Region	False+		False−	
ID	CAP	KNN	CAP	KNN
1	0.2957	0.3385	0.6498	0.3628
2	0.3085	0.3067	0.6162	0.3838
3	0.3211	0.2733	0.6330	0.4967
4	0.3569	0.3011	0.6451	0.3523
5	0.2657	0.1843	0.6684	0.5263
6	0.3754	0.2438	0.6222	0.4137
7	0.4310	0.2763	0.5896	0.3676
8	0.3730	0.2783	0.6154	0.4218

Abbreviation: "CAP" refers to "ADLCAP".

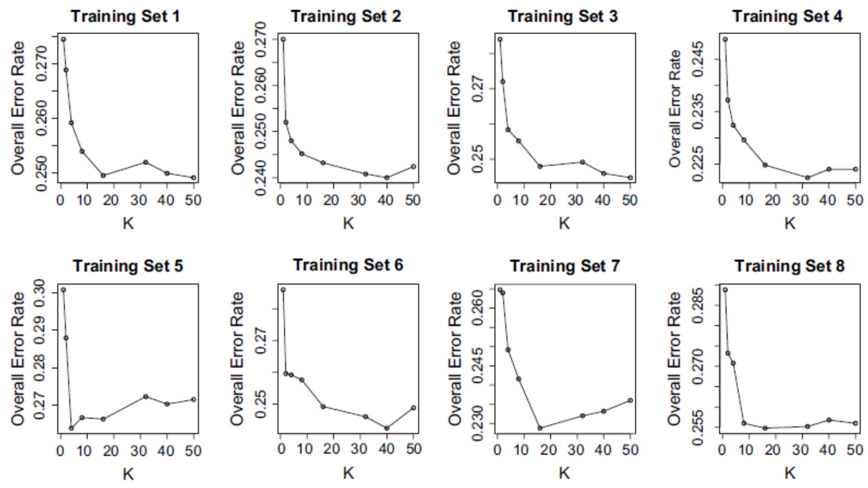
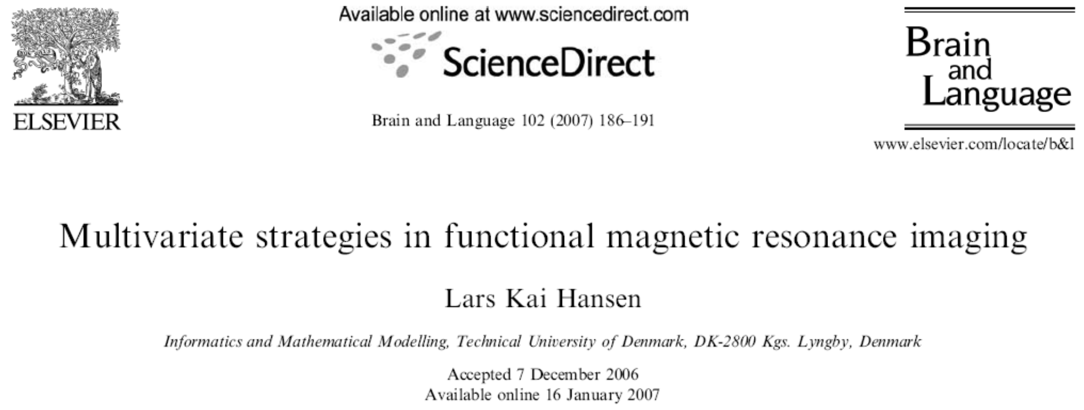


Fig. 1. Cross-validated overall error rate vs. *K*.

Example: Classification of brain images



Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Brain and Language 102 (2007) 186–191

Brain and Language

www.elsevier.com/locate/b&l

Multivariate strategies in functional magnetic resonance imaging

Lars Kai Hansen

Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

Accepted 7 December 2006
Available online 16 January 2007

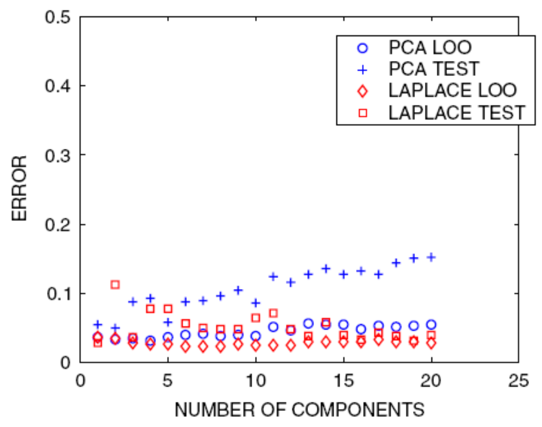


Fig. 3. Error rates for *k*-nearest neighbor classifiers trained on fMRI data from a single subject and generalizing to data not part of the training set. We train the classifier on the two different representations obtained by PCA and by Laplacian eigenmaps, as shown in Figs. 1 and 2. We estimate the optimal number of neighbors in the voting classifier in feature space of dimensionality $d = 1{:}20$. The leave-one-out optimal dimensionality is $d = 4$ for PCA and $d = 6$ for the Laplacian eigenmap. The resulting classifiers obtain unbiased test set classification error rates 5% and 9%, in favor of the non-linear features.

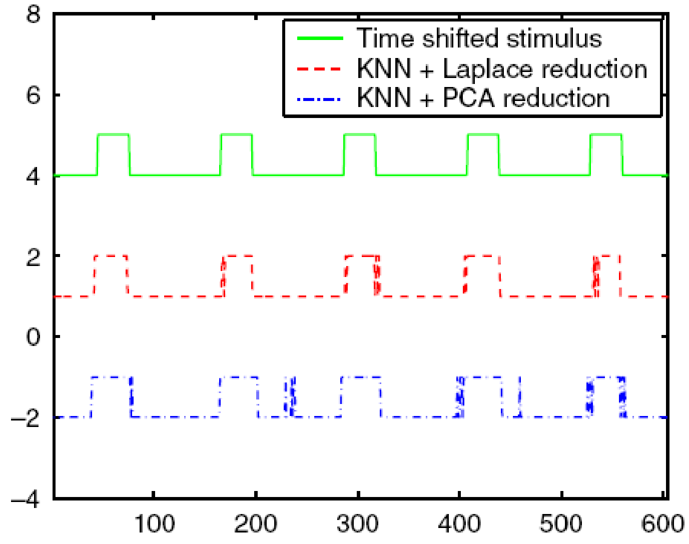


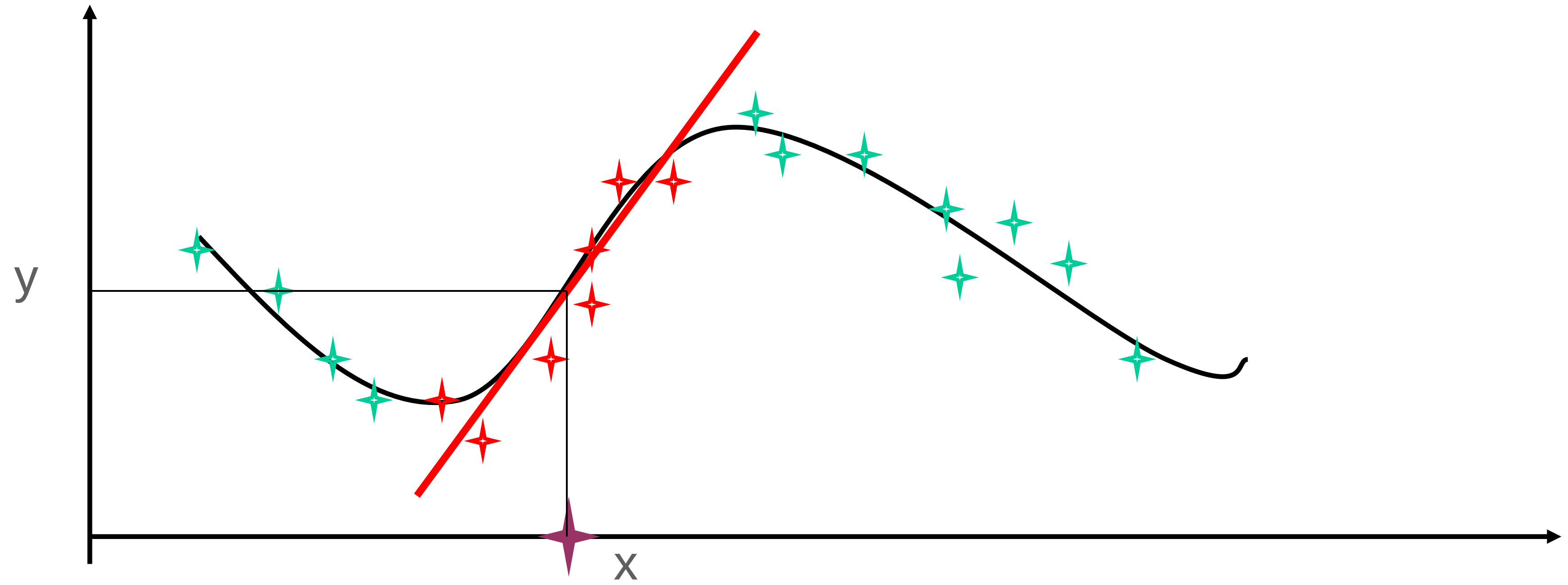
Fig. 4. Test set reference activation time course and activation time courses produced by *k*-nearest neighbor classifiers based on linear and non-linear feature spaces. The non-linear feature based classifier's errors basically occurs at the onset and at end of stimulation. The KNN model based on the linear feature representation make additional generalization errors in the baseline, where it suggest a few short burst of activation.

JAVA Demo of KNN

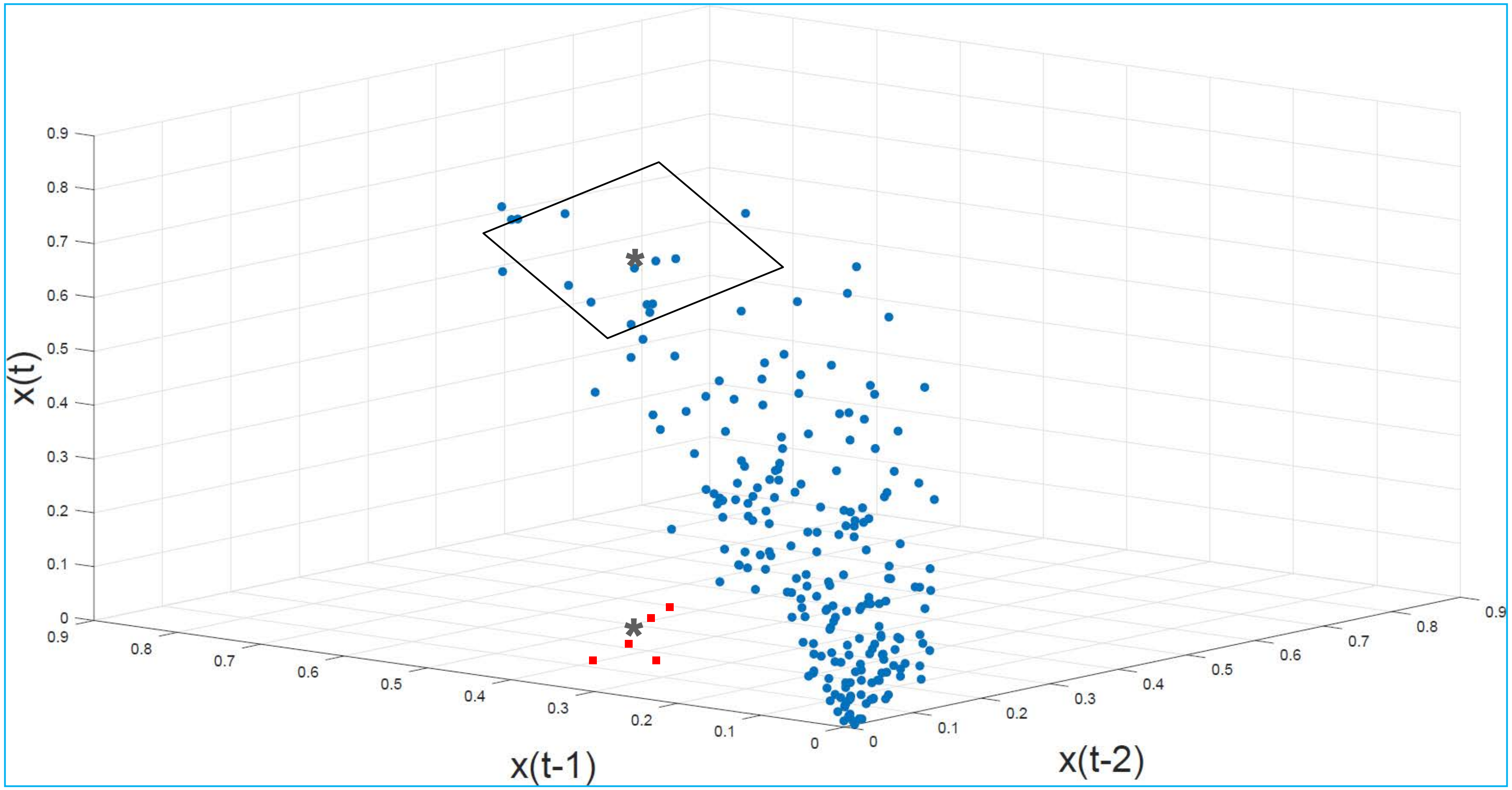
http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

Supervised learning: Function approximation

Given a test input x, locate the K nearest neighbors, use simple linear regression model to predict the output



Sun spot (d=2, K=5)



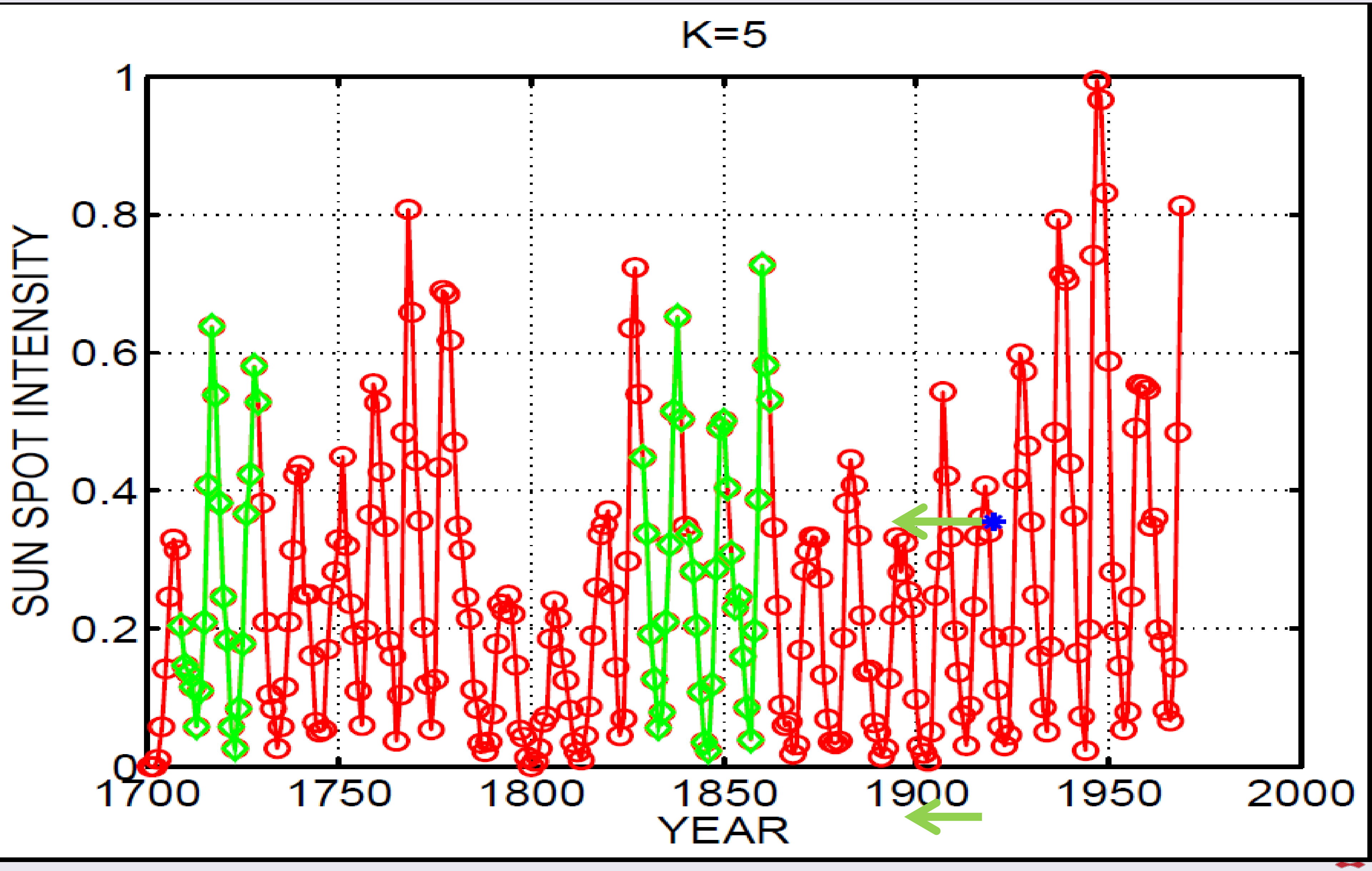
Local linear regression

Use regularized, linear regression method on neighbors (from Ex 4 .... N=K!)

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\{y(\mathbf{x}_n;\mathbf{w}) - t_n\}^2 + \frac{1}{2}\alpha\mathbf{w}^2 \\ &= \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^\top\mathbf{x}_n - t_n\}^2 + \frac{1}{2}\alpha\mathbf{w}^2. \\ \mathbf{X}^\top &= (\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \mathbf{x}_N) \end{aligned}$$

$$\boxed{\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \alpha\mathbf{1})^{-1}\mathbf{X}^\top\mathbf{t}.}$$

Sun spots $\boxed{d = 10}$



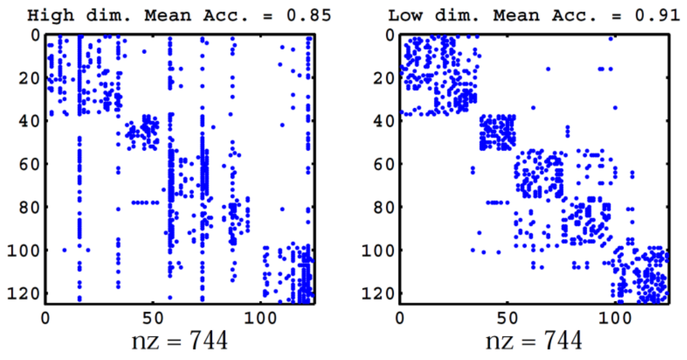
Open issues in nearest neighbor methods

Number of neighbors to use
- Cross-validation, leave-one-out is easy!

How to find the neighbors efficiently
- k-d trees

Pruning the training set
- Support vectors, relevance vectors

Weights on neighbors
- Distance based



"Hubs problem" asymmetry of neighborhoods
- Radovanović, Miloš, Alexandros Nanopoulos, Mirjana Ivanović. "Hubs in space: Popular nearest neighbors in high-dimensional data." *The Journal of Machine Learning Research* (2010): 2487-2531.

Everything is based on a metric
- Learning the metric from the data set
- Are all dimensions equally important? Feature selection

Soft neighborhoods

Kernel methods: Linear model motivaton

Assume we have a high-dimensional data set with input variables **x** in d-dimensional space and d > N

$$D = \{(t_1, \mathbf{x}_1), (t_2, \mathbf{x}_2), \ldots, (t_N, \mathbf{x}_N)\}$$

Linear model fitted from least squares $E(\mathbf{w}) = \sum_{n=1}^{N} (t_n - \sum_{j=1}^{d+1} w_j x_{j,n})^2$

$$\mathbf{w} = \mathbf{w}_\perp + \mathbf{w}_\parallel \qquad \boxed{\mathbf{w}_\parallel = \sum_{n=1}^{N} a_n \mathbf{x}_n,}$$

$$\begin{aligned} E(\mathbf{w}) &= \sum_{n=1}^{N} (t_n - \mathbf{w}^\top \mathbf{x}_n)^2 = \sum_{n=1}^{N} (t_n - \mathbf{w}_\parallel^\top \mathbf{x}_n)^2 = \sum_{n=1}^{N} (t_n - \sum_{m=1}^{N} a_m \mathbf{x}_m^\top \mathbf{x}_n)^2 \\ &= \sum_{n=1}^{N} (t_n - \sum_{m=1}^{N} a_m K_{m,n})^2 = \sum_{n=1}^{N} (t_n - (\mathbf{a}^\top \mathbf{K})_n)^2. \end{aligned}$$

$$(\mathbf{K})_{m,n} = K_{m,n} = \mathbf{x}_m^\top \mathbf{x}_n$$

Kernel methods

$$(\mathbf{K})_{m,n} = K_{m,n} = \mathbf{x}_m^\top \mathbf{x}_n$$

Hence we can ignore the component of the weight vector which is orthogonal to the subspace spanned by the data

Costfunction (likelihood function) is ”blind” to this subspace

We can reduce the fitting problem to estimation of an N-dimensional vector (**a**) forget about data vectors **x** and keep the kernel matrix **K**

$$\mathbf{w}_{||} = \sum_{n=1}^{N} a_n \mathbf{x}_n, \qquad (\mathbf{K})_{m,n} = K_{m,n} = \mathbf{x}_m^\top \mathbf{x}_n$$

Kernel methods: Implict features

Assume our model is based on a feature representation

$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

$$E(\mathbf{w}) = \sum_{n=1}^{N} (t_n - \mathbf{w}^\top \phi(\mathbf{x}_n))^2 = \sum_{n=1}^{N} (t_n - (\mathbf{a}^\top \mathbf{K})_n)^2$$

$$(\mathbf{K})_{m,n} = K_{m,n} = \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n)$$

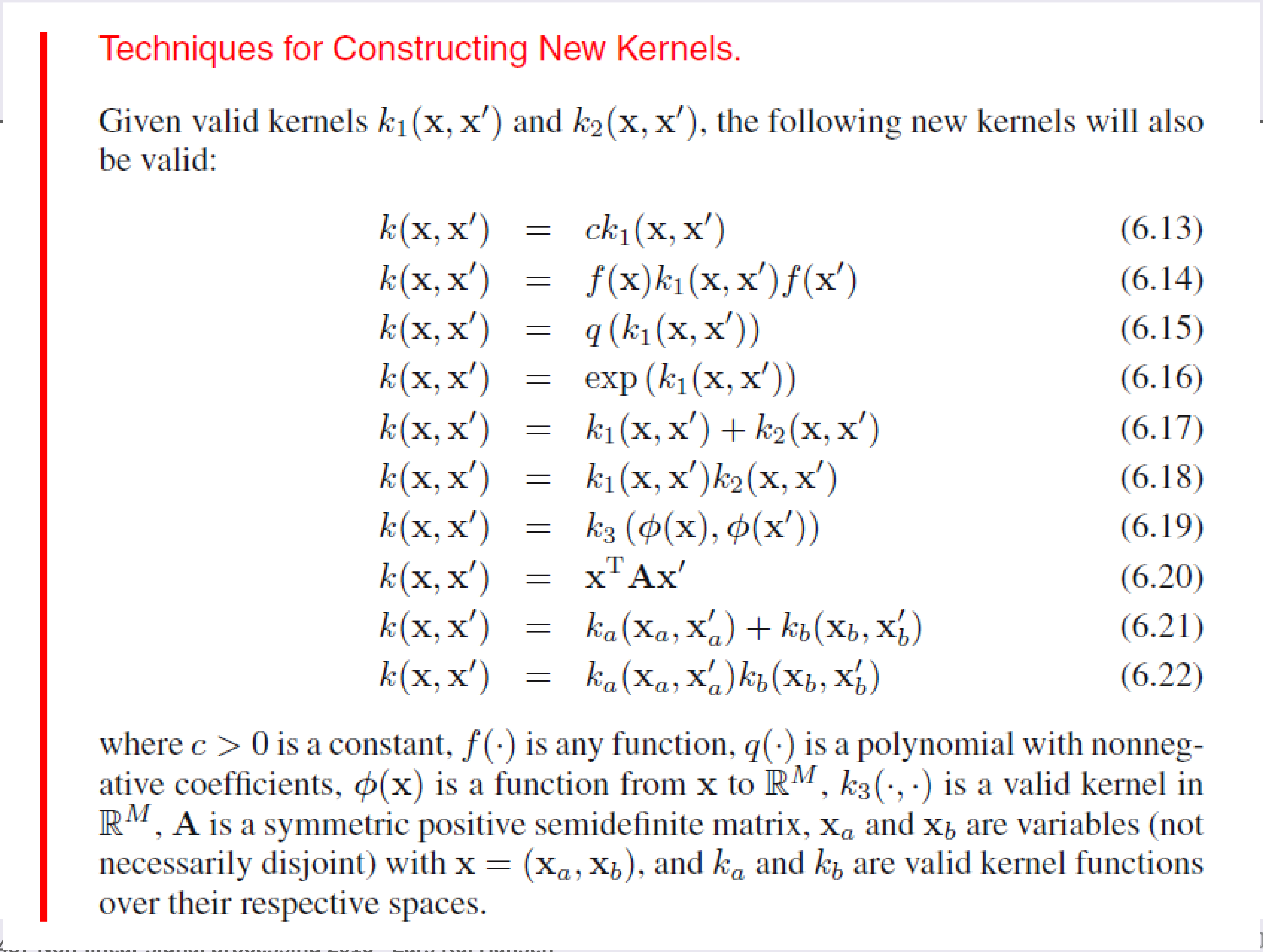
Kernel methods: Implict features

Now if we postulate a kernel function K(**x**,**x**') then we can on the other hand implicit define a feature representation

$$(\mathbf{K})_{m,n} = K_{m,n} = \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n)$$

When can a symmetric matrix generated from a function k($\mathbf{x}_m$,$\mathbf{x}_n$) be reconstructed as the inner products?

If any subset of points give rise to a positive definite matrix.



Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= c k_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}
$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Gaussian kernel, ”radial basis function kernel”

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

Is it an ok kernel?

$$\|\mathbf{x} - \mathbf{x}'\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2\right) \exp\left(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/\sigma^2\right) \exp\left(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2\right)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') \tag{6.14}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.16}$$

Kernel methods for supervised learning

The general idea of kernel representations for supervised learning is that similarity in input should lead to similarity in output

$$\boxed{(\mathbf{x}_m \approx \mathbf{x}_n) \Rightarrow (\mathbf{t}_m \approx \mathbf{t}_n)}$$

Gaussian processes for function approximation

$$\boxed{(\mathbf{x}_m \approx \mathbf{x}_n) \Rightarrow (\mathbf{t}_m \approx \mathbf{t}_n)}$$

In the Gaussian process model the similarity is implemented in a probabilistic setting

For additive noise model t(**x**) = y(**x**) + e, we can represent the similarity as an assumed Gaussian distribution of the function values for a general set of inputs

$$\mathrm{cov}(y_1, ..., y_N) = \mathbf{K}, \quad p(\mathbf{y}|\mathbf{K}) = \frac{1}{|2\pi\mathbf{K}|^{\frac{1}{2}}} \exp(-\tfrac{1}{2}\mathbf{y}^\top\mathbf{K}^{-1}\mathbf{y})$$

Kernel methods for supervised learning

The Gaussian distribution of the target then follows

$$\mathrm{cov}(t_1, ..., t_N) \equiv \mathbf{C} = \mathbf{K} + \beta^{-1}\mathbf{1}$$

Predictive distribution

$$p(\mathbf{t}_{\mathrm{test}}|\mathbf{t}_{\mathrm{train}}, \mathbf{C}_{\mathrm{train,test}}) = \frac{p(\mathbf{t}_{\mathrm{test}}, \mathbf{t}_{\mathrm{train}}|\mathbf{C}_{\mathrm{train,test}})}{p(\mathbf{t}_{\mathrm{train}}|\mathbf{C}_{\mathrm{train}})}$$

Conditioning in the normal distribution

The Matrix Cookbook

[http://matrixcookbook.com]

Kaare Brandt Petersen

Michael Syskind Pedersen

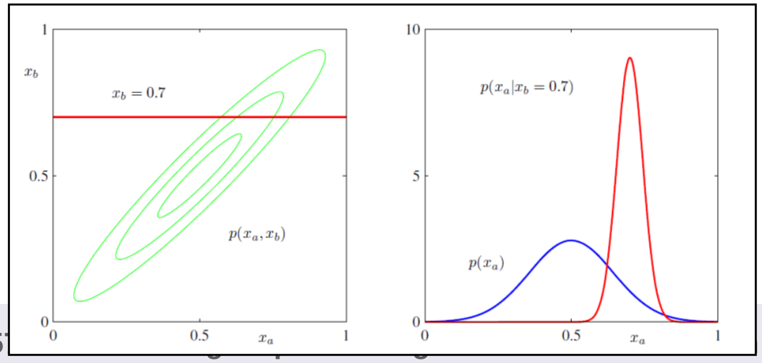
Version: November 15, 2012

Assume $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}(\mu, \mathbf{\Sigma})$ where

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_a & \mathbf{\Sigma}_c \\ \mathbf{\Sigma}_c^T & \mathbf{\Sigma}_b \end{bmatrix}$$

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}_{\mathbf{x}_a}(\hat{\mu}_a, \hat{\mathbf{\Sigma}}_a) \qquad \left\{ \begin{array}{lcl} \hat{\boldsymbol{\mu}}_a & = & \boldsymbol{\mu}_a + \mathbf{\Sigma}_c\mathbf{\Sigma}_b^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \\ \hat{\mathbf{\Sigma}}_a & = & \mathbf{\Sigma}_a - \mathbf{\Sigma}_c\mathbf{\Sigma}_b^{-1}\mathbf{\Sigma}_c^T \end{array} \right.$$

$$p(\mathbf{x}_b|\mathbf{x}_a) = \mathcal{N}_{\mathbf{x}_b}(\hat{\mu}_b, \hat{\mathbf{\Sigma}}_b) \qquad \left\{ \begin{array}{lcl} \hat{\boldsymbol{\mu}}_b & = & \boldsymbol{\mu}_b + \mathbf{\Sigma}_c^T\mathbf{\Sigma}_a^{-1}(\mathbf{x}_a - \boldsymbol{\mu}_a) \\ \hat{\mathbf{\Sigma}}_b & = & \mathbf{\Sigma}_b - \mathbf{\Sigma}_c^T\mathbf{\Sigma}_a^{-1}\mathbf{\Sigma}_c \end{array} \right.$$



Kernel methods for supervised learning

”Matrix Cookbook” rules for conditioning

$$\begin{aligned}\boldsymbol{\mu}_{\text{test}|\text{train}} &= \mathbf{C}_{\text{test}|\text{train}}\mathbf{C}_{\text{train}}^{-1}\mathbf{t}_{\text{train}} \\ \mathbf{C}_{\text{test}|\text{train}} &= \mathbf{C}_{\text{test}} - \mathbf{C}_{\text{test}|\text{train}}\mathbf{C}_{\text{train}}^{-1}\mathbf{C}_{\text{train}|\text{test}}\end{aligned}$$

$$\hat{t}_m = (\boldsymbol{\mu}_{\text{test}|\text{train}})_m$$

$$y(\mathbf{x}) = \hat{t}(\mathbf{x}) = \mathbf{C}_{\mathbf{x}|\text{train}}\mathbf{C}_{\text{train}}^{-1}\mathbf{t}_{\text{train}} = \sum_{n=1}^{N} k(\mathbf{x},\mathbf{x}_n)\sum_{n'=1}^{N}(\mathbf{C}_{\text{train}}^{-1})_{n,n'}t_{n'}$$

$$\text{std}(t_m) = \sqrt{(\mathbf{C}_{\text{test}|\text{train}})_{m,m}} = \sqrt{(\mathbf{C}_{\text{test}})_{m,m} - (\mathbf{C}_{\text{test}|\text{train}}\mathbf{C}_{\text{train}}^{-1}\mathbf{C}_{\text{train}|\text{test}})_{m,m}}$$

Support vector machines

The general idea of kernel representations for supervised learning is that similarity in input should lead to similarity in output

For support vectors this is assumed to hold for the labels

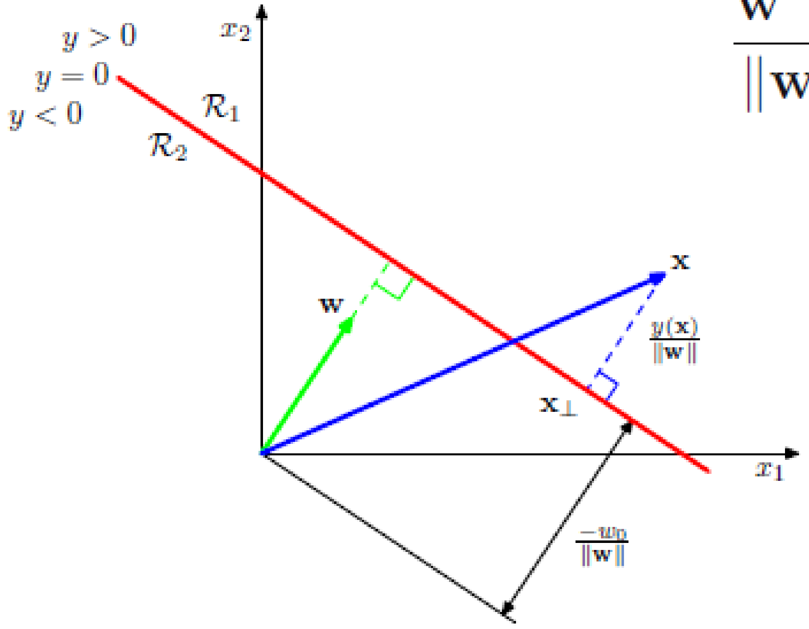
$$\boxed{(\mathbf{x}_m \approx \mathbf{x}_n) \Rightarrow (\mathbf{t}_m \approx \mathbf{t}_n)}$$

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

Geometry of linear discriminant

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}$$

Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.



$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

Maximum margin principle

$$\boxed{y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b}$$

Margin = $|y(\mathbf{x})| / \|\mathbf{w}\|$

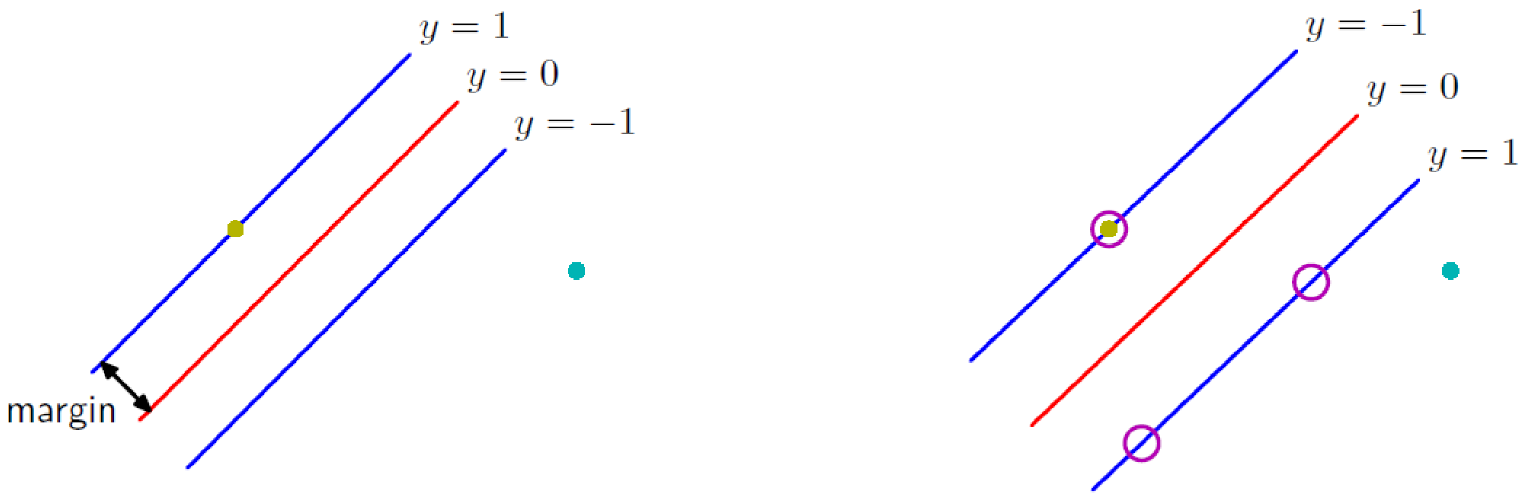
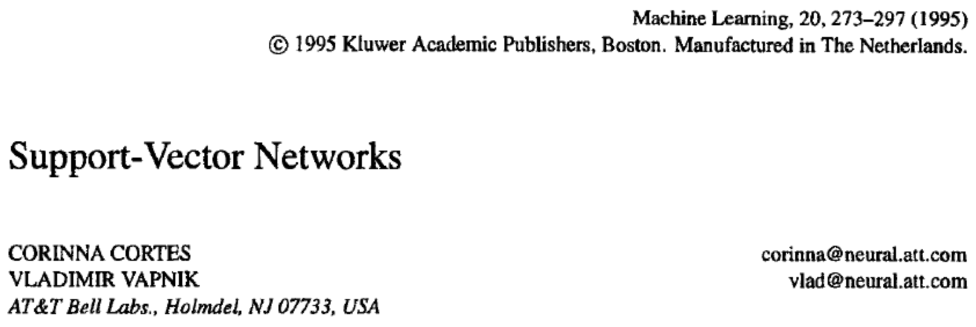


Figure 7.1 The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

Cortes & Vapnik



Machine Learning, 20, 273–297 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Support-Vector Networks

CORINNA CORTES corinna@neural.att.com
VLADIMIR VAPNIK vlad@neural.att.com
AT&T Bell Labs., Holmdel, NJ 07733, USA

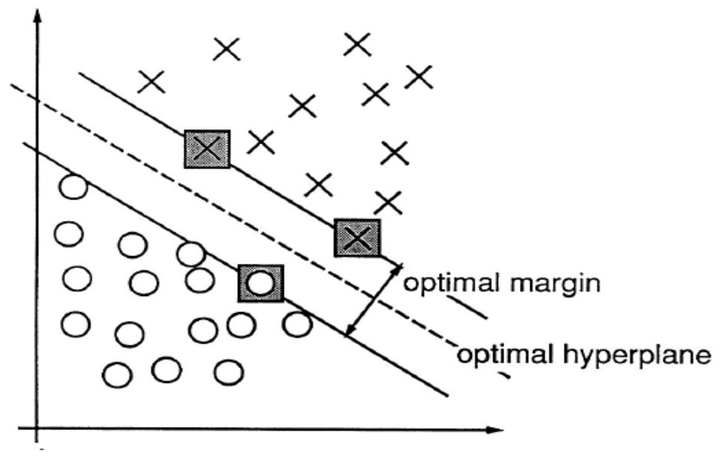
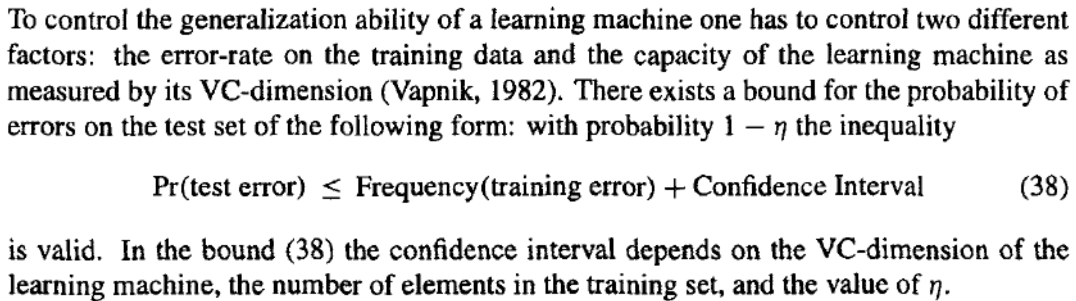


Figure 2. An example of a separable problem in a 2 dimensional space. The support vectors, marked with grey squares, define the margin of largest separation between the two classes.



To control the generalization ability of a learning machine one has to control two different factors: the error-rate on the training data and the capacity of the learning machine as measured by its VC-dimension (Vapnik, 1982). There exists a bound for the probability of errors on the test set of the following form: with probability $1 - \eta$ the inequality

$$\Pr(\text{test error}) \leq \text{Frequency}(\text{training error}) + \text{Confidence Interval} \qquad (38)$$

is valid. In the bound (38) the confidence interval depends on the VC-dimension of the learning machine, the number of elements in the training set, and the value of η.

Maximum margin principle

Margin = $|y(\mathbf{x})| / \|\mathbf{w}\|$

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$$

$$\underset{\mathbf{w},b}{\arg\max} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b \right) \right] \right\}$$

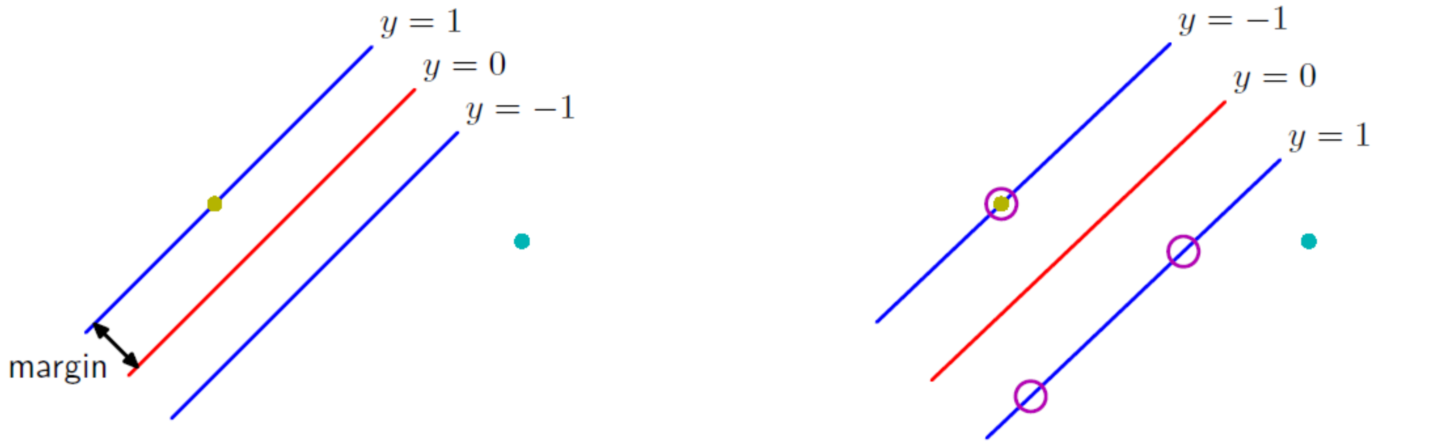


Figure 7.1 The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

Convex criteria

rescaling $\mathbf{w} \rightarrow \kappa\mathbf{w}$ and $b \rightarrow \kappa b$

$$t_n\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+b\right)=1$$ For points closest to y=0

$$t_n\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+b\right)\geqslant 1$$ For all points

$$\underset{\mathbf{w},b}{\arg\max}\left\{\frac{1}{\|\mathbf{w}\|}\min_n\left[t_n\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+b\right)\right]\right\}$$

$$\boxed{\underset{\mathbf{w},b}{\arg\min}\,\frac{1}{2}\|\mathbf{w}\|^2 \quad \text{with the above constraint}}$$

Convex optimization

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b) - 1 \right\}$$

$$\begin{aligned} \mathbf{w} &= \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \\ 0 &= \sum_{n=1}^{N} a_n t_n. \end{aligned}$$

Convex optimization = quadratic optimization w. constraint

Eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$ using these conditions then gives the *dual representation* of the maximum margin problem in which we maximize

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \qquad (7.10)$$

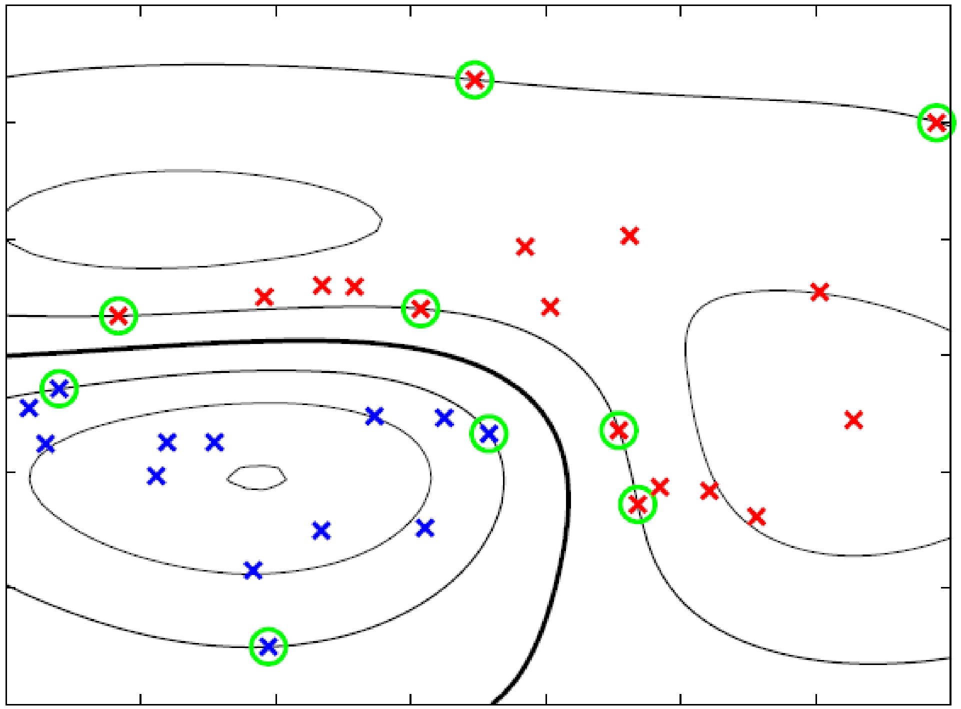
with respect to $\mathbf{a}$ subject to the constraints

$$a_n \geqslant 0, \qquad n = 1, \ldots, N, \qquad (7.11)$$

$$\sum_{n=1}^{N} a_n t_n = 0. \qquad (7.12)$$

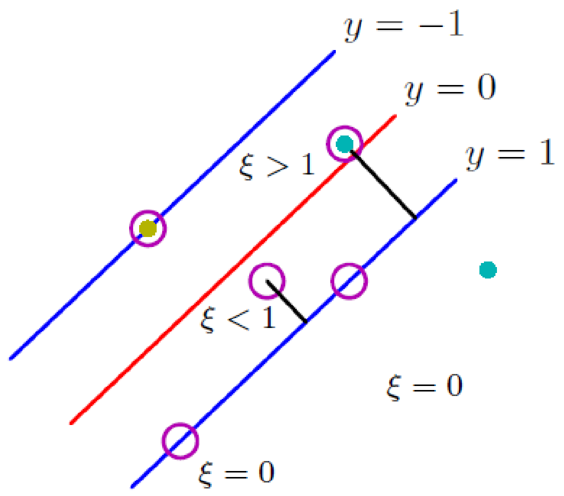
$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{x}')$$

Figure 7.2 Example of synthetic data from two classes in two dimensions showing contours of constant $y(\mathbf{x})$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.



Slack variables for noisy data (non-separable)

Figure 7.3 Illustration of the slack variables $\xi_n \geqslant 0$. Data points with circles around them are support vectors.



$$t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n,$$

Modified optimization problem

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \sum_{n,m=1}^{N} a_n a_m t_n t_m K(\mathbf{x}_n, \mathbf{x}_m),$$

$$0 \leq a_n \leq C,$$

$$\sum_{n=1}^{N} a_n t_n = 0,$$

$$b = \frac{1}{|M|} \sum_{n \in M} \left(t_n - \sum_{m \in S} a_m t_m K(\mathbf{x}_n, \mathbf{x}_m)\right).$$