Class 3

Classical statistical techniques

Federico Girosi

Plan of class 3

- Unsupervised Learning;
  - Clustering;
  - Principal Component Analysis;
- Supervised Learning;
  - Local approximation;
  - Nearest Neighbor Techniques;
  - Kernel regression;
  - Parzen windows;
  - Dual kernels: global versus local approximation;

Cluster Analysis

Why Clustering?

- Explorary Data Analysis;

- Pre-processing;

- Vector Quantization;

Clustering

"Clustering algorithms are methods to divide a set of N observations into m groups so that members of the same group are more alike than members of different groups".

(B. Ripley: *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996)

Clustering

Clustering can also mean the following: *given a set of data points* $\{\mathbf{x}_i\}_{i=1}^N$ *in* R^d *means find a set of* m *representative points* $\{\mathbf{m}_k\}_{k=1}^m$.

Let S_k be the Voronoi polytopes of center $\mathbf{m}_k$:

$$S_k = \{\mathbf{x} : \|\mathbf{x} - \mathbf{m}_k\| < \|\mathbf{x} - \mathbf{m}_j\|, j \neq k\}$$

A set of representative points $\mathcal{M} = \{\mathbf{m}_k\}_{k=1}^m$ is optimal if solves:

$$\min_{\mathcal{M}} \sum_{k=1}^{m} \sum_{\mathbf{x}_i \in S_k} \|\mathbf{x}_i - \mathbf{m}_k\|$$

K-means algorithm

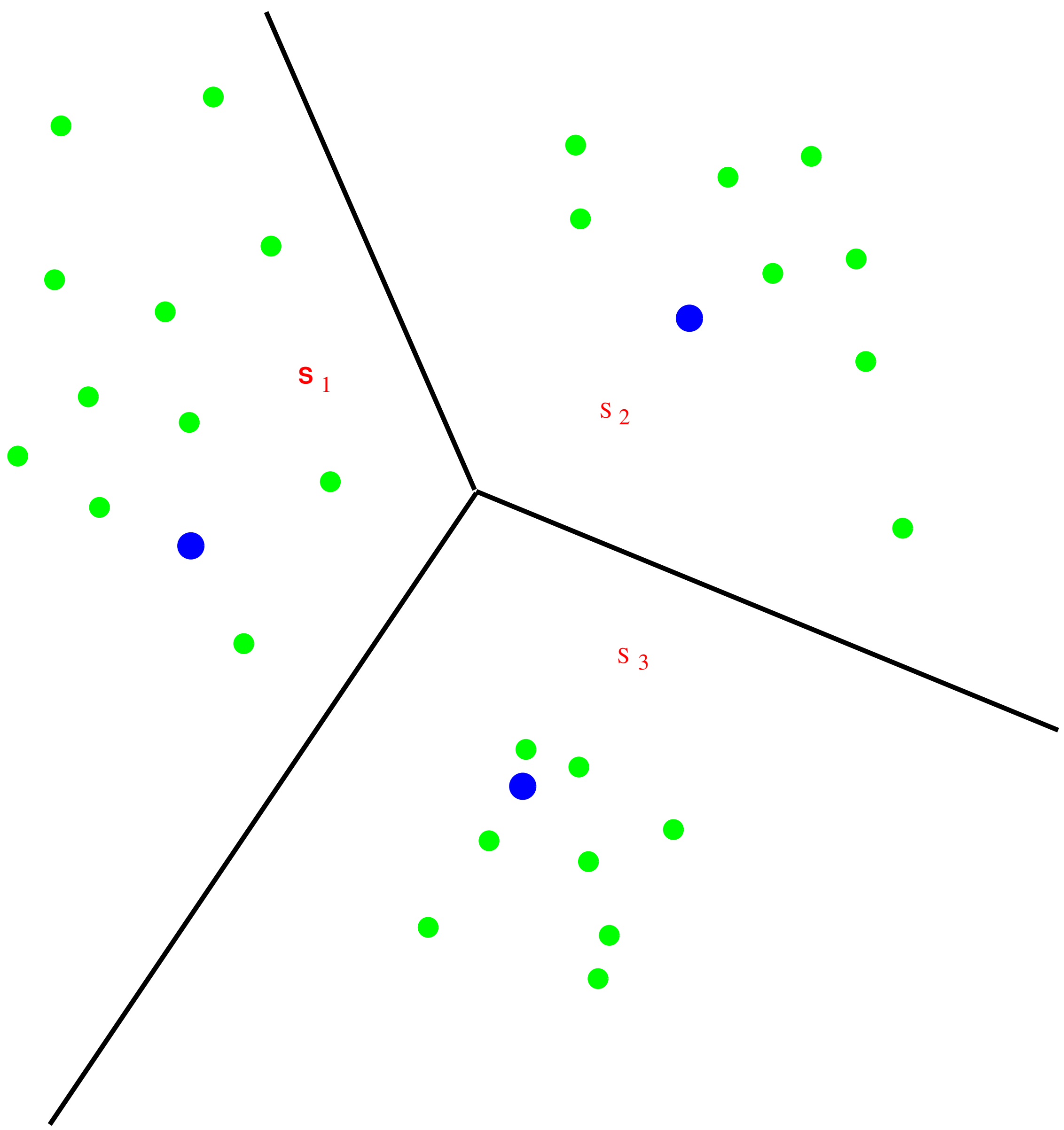
The *k-means algorithm* (MacQueen, 1967) is an iterative technique for finding optimal representative points (cluster centers):

$$\mathbf{m}_k^{(t+1)} = \frac{1}{\# S_k^{(t)}} \sum_{\mathbf{x}_i \in S_k^{(t)}} \mathbf{x}_i$$

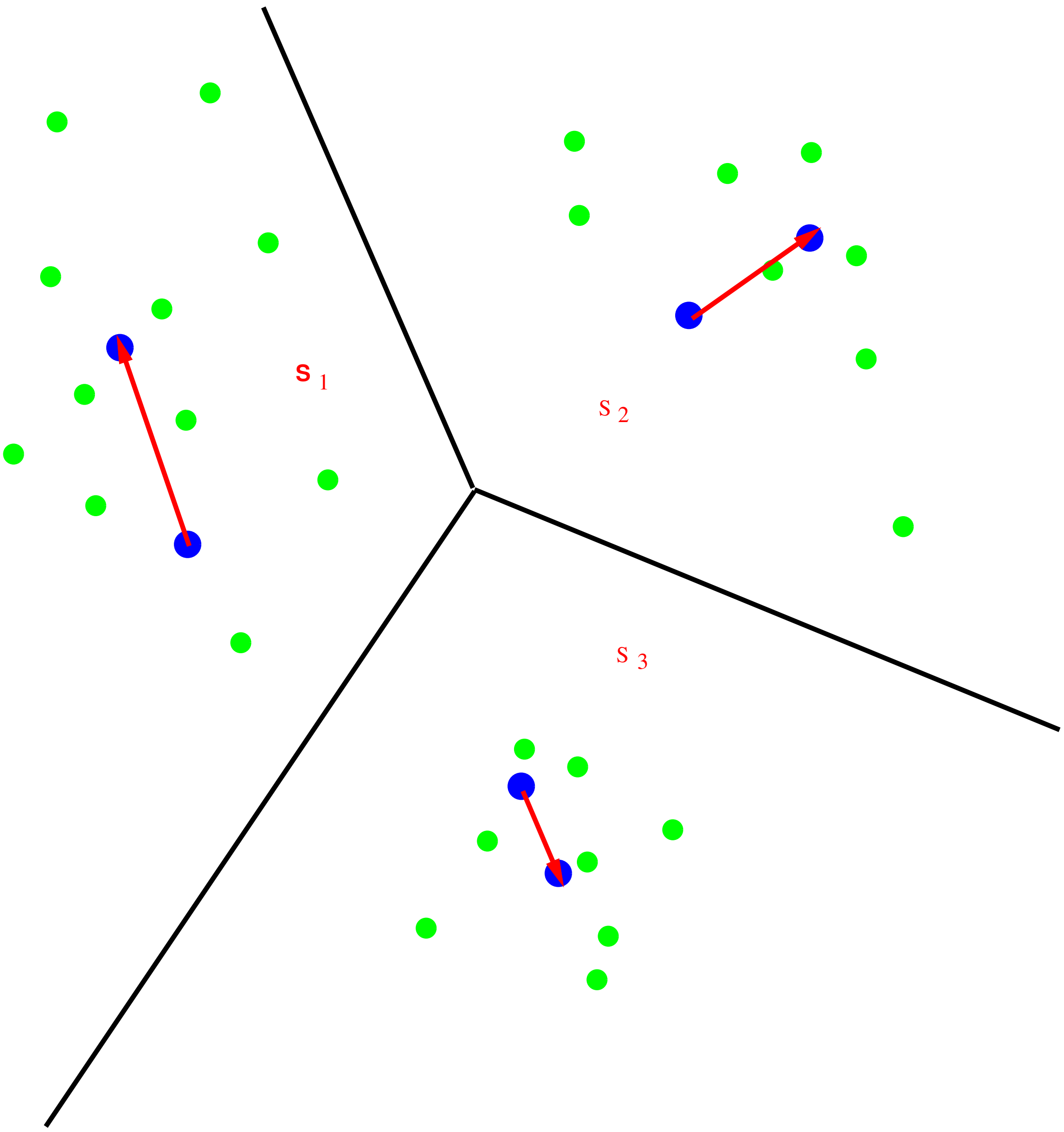
This algorithm is guaranteed to find a local minimum.

Many variations of this algorithm have been proposed.

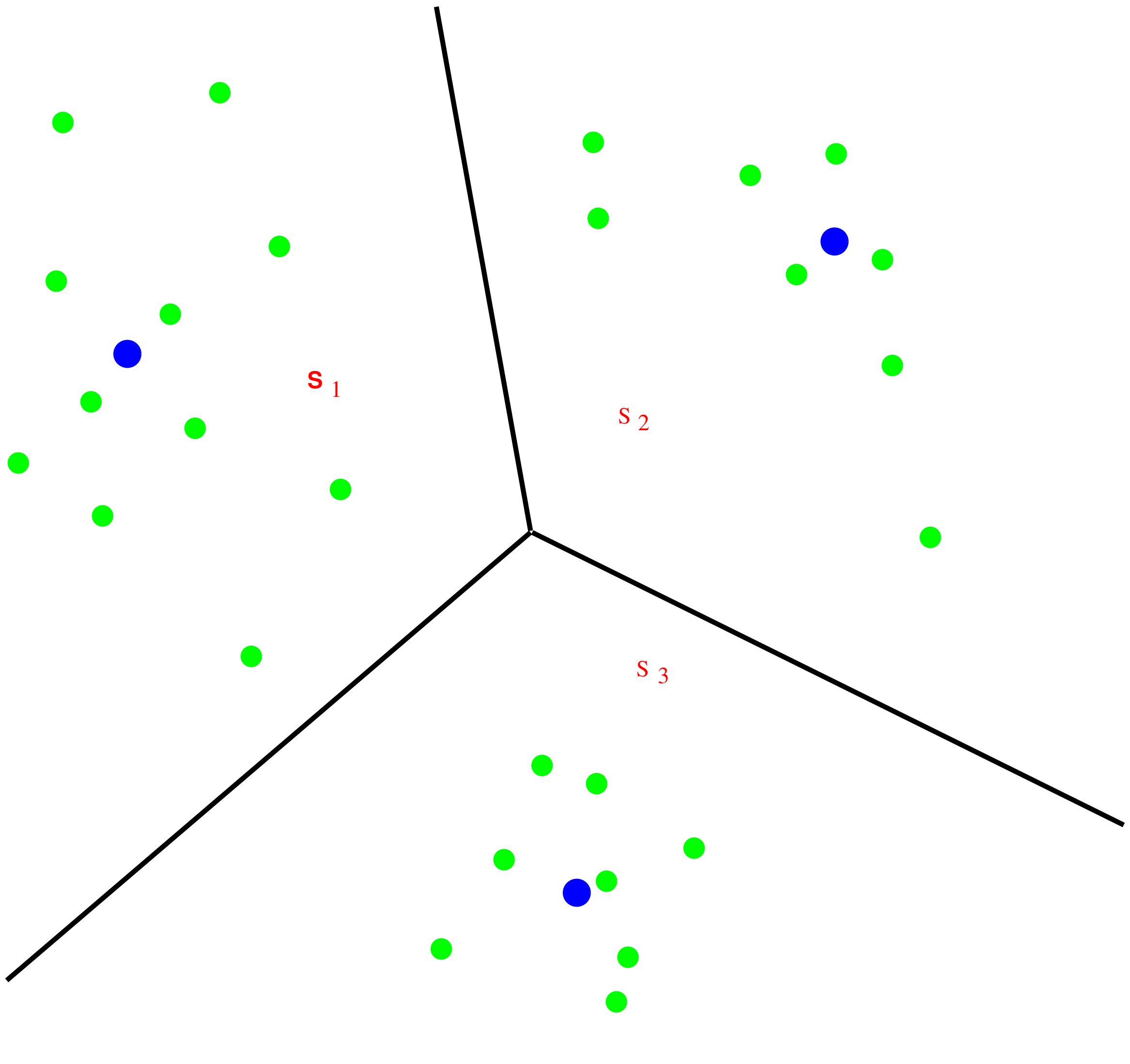
K-means algorithm



K-means algorithm



K-means algorithm



References on clustering

1. M.R. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, 1973.

2. J.A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

3. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

4. L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York, 1990.

Principal Component Analysis

Let $\{\mathbf{x}_\alpha\}_{\alpha=1}^{N}$ be a set of points in R^d, with d possibly very large.

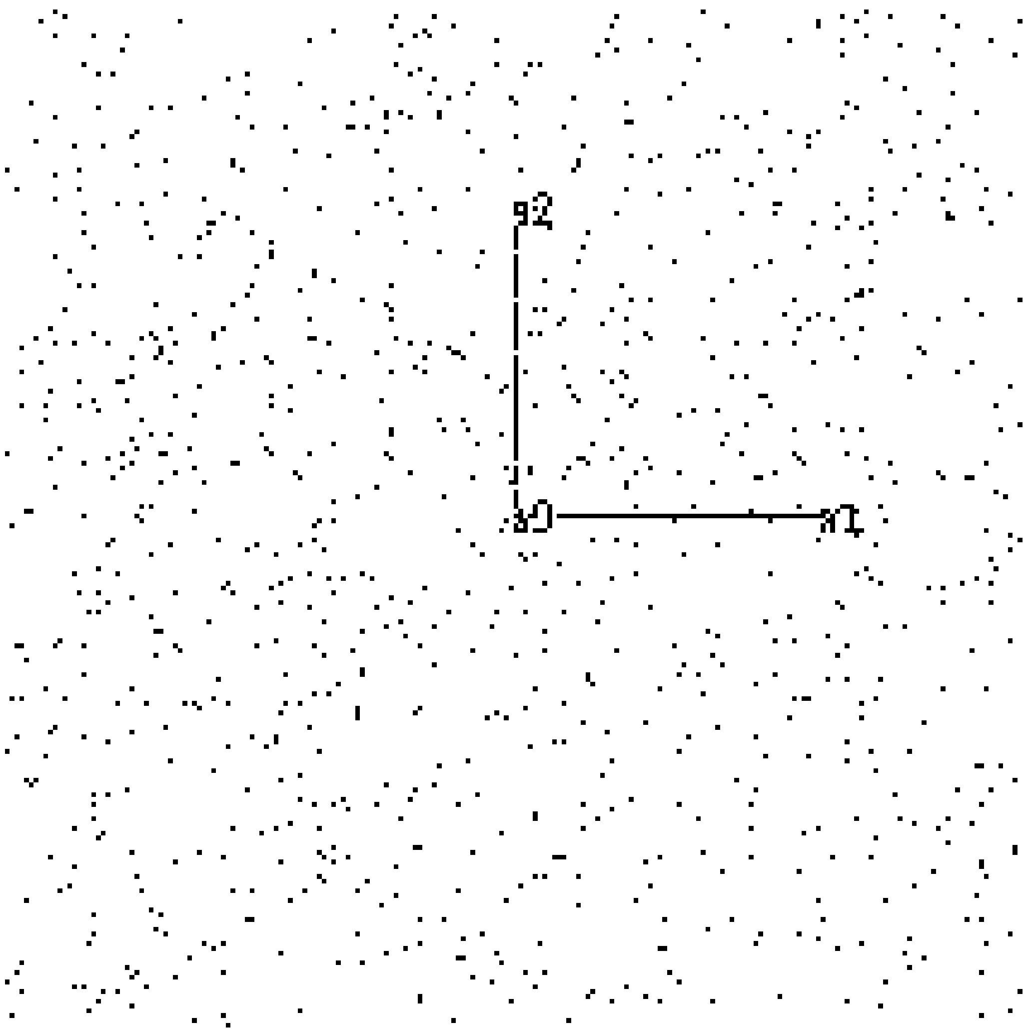
Every point is specified by d numbers.

Is there a more compact representation?

It depends on the distribution of the data points in R^d ...

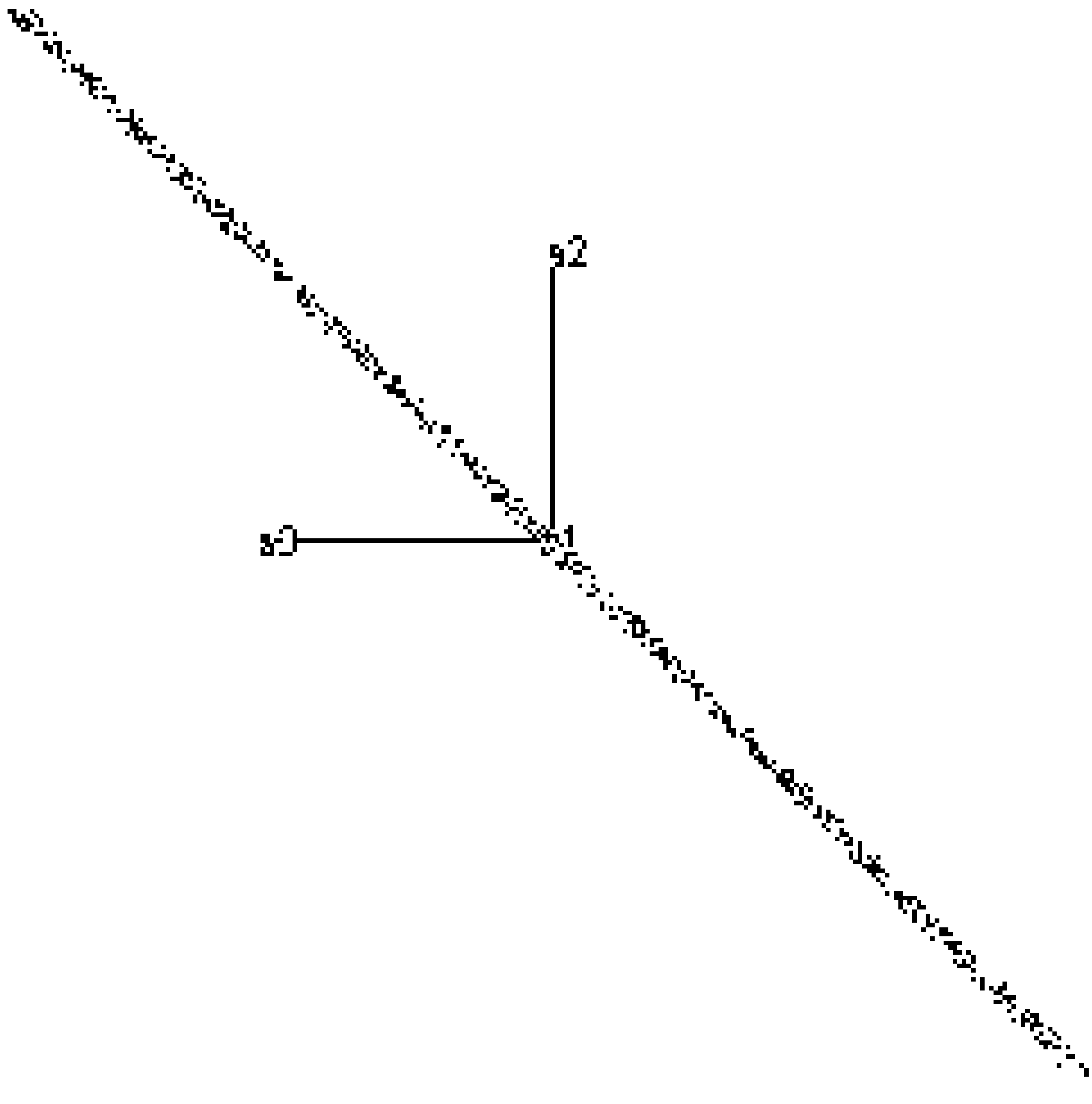
There are random points ...

0.058	0.567	0.147
0.337	0.918	0.5
0.005	0.273	0.678
0.112	0.688	0.821
0.948	0.91	0.961
0.867	0.475	0.117
0.013	0.554	0.912
0.655	0.244	0.708
0.405	0.091	0.628
0.285	0.385	0.032
0.034	0.972	0.003
0.529	0.816	0.542
0.973	0.14	0.692
0.263	0.266	0.283
0.403	0.504	0.425
0.498	0.069	0.019
0.172	0.06	0.01
0.352	0.46	0.835



and random points ...

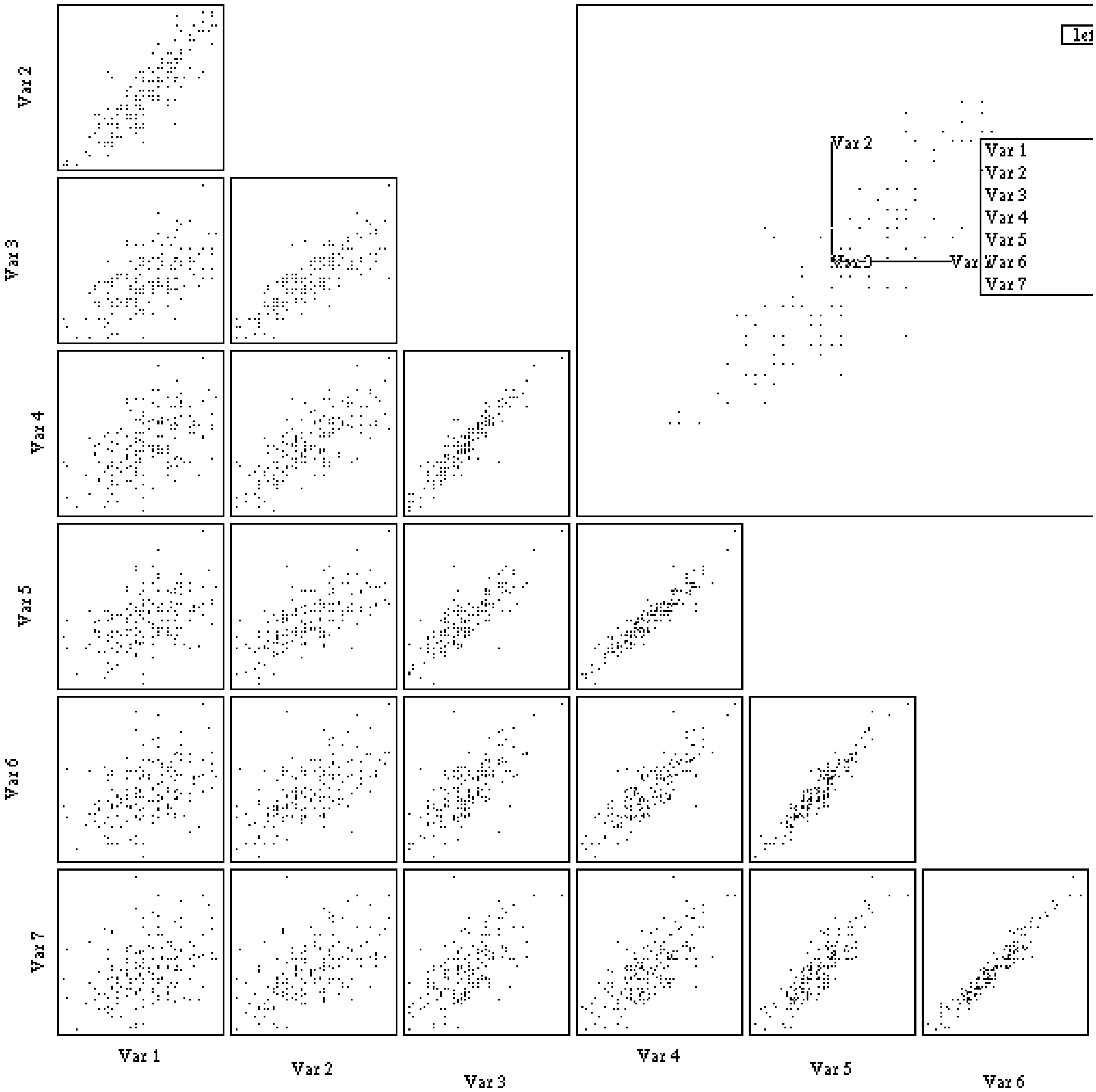
0.033	0.42	0.84
0.727	0.577	1.154
0.372	0.957	1.914
0.177	0.45	0.9
0.222	0.562	1.124
0.693	0.644	1.288
0.985	0.805	1.61
0.167	0.766	1.532
0.701	0.078	0.156
0.768	0.791	1.582
0.51	0.874	1.748
0.432	0.375	0.75
0.942	0.251	0.502
0.805	0.795	1.59
0.373	0.181	0.362
0.189	0.541	1.082
0.77	0.724	1.448



s2
s3
s1

Nr.	Feature
1	pupil to nose vertical distance
2	pupil to mouth vertical distance
3	pupil to chin vertical distance
4	nose width
5	mouth width
6	face width halfway between nose tip and eyes
7	face width at nose position
8-13	chin radii
14	mouth height
15	upper lip thickness
16	lower lip thickness
17	pupil to eyebrow separation
18	eyebrow thickness

Features 8 to 14



Principal Component Analysis

The data might lie (at least approximately) in a k-dimensional linear subspace of R^d, with $k << d$:

$$\mathbf{x}_\alpha \approx \sum_{i=1}^{k} (\mathbf{x}_\alpha \cdot \mathbf{d}_i)\mathbf{d}_i$$

where $\mathcal{D} = \{\mathbf{d}_i\}_{i=1}^{k}$ is an orthonormal set of vectors:

$$\mathbf{d}_i \cdot \mathbf{d}_j = \delta_{ij}$$

Principal Component Analysis let us find *the best orthonormal set of vectors* $\{\mathbf{d}_i\}_{i=1}^{k}$, which are called the *first* k *principal components*.

The orthonormal basis $\mathcal{D} = \{\mathbf{d}_i\}_{i=1}^{k}$ is found by solving the following *least squares* problem:

$$\min_{\mathcal{D}} \sum_{\alpha=1}^{N} (\mathbf{x}_\alpha - \sum_{i=1}^{k} (\mathbf{x}_\alpha \cdot \mathbf{d}_i)\mathbf{d}_i)^2 =$$

$$\max_{\mathcal{D}} \sum_{\alpha=1}^{N} \sum_{i=1}^{k} (\mathbf{x}_\alpha \cdot \mathbf{d}_i)^2 =$$

$$= \frac{1}{N} \max_{\mathcal{D}} \sum_{i=1}^{k} \mathbf{d}_i \cdot C\mathbf{d}_i$$

where C is the *correlation matrix*:

$$C_{ij} = \frac{1}{N} \sum_{\alpha=1}^{N} x_\alpha^i x_\alpha^j$$

Let us consider the case $k = 1$, and $\mathbf{d}_1 = \mathbf{d}$. The problem is now to solve:

$$\frac{1}{N} \max_{\|\mathbf{d}\|=1} \mathbf{d} \cdot C\mathbf{d}$$

where C is a symmetrix matrix (correlation matrix). Setting

$$C = R^T \Lambda R \ , \qquad \mathbf{d}^* = R\mathbf{d}$$

with R a rotation matrix and Λ diagonal, we have now to solve:

$$\frac{1}{N} \max_{\|\mathbf{d}^*\|=1} \mathbf{d}^* \cdot \Lambda \mathbf{d}^*$$

Notice that, since $\|\mathbf{d}^*\| = 1$, then:

$$\mathbf{d}^* \cdot \Lambda \mathbf{d}^* \leq \lambda_{max}$$

If we choose the vector $\mathbf{d}^*_{max}$ such that $\mathbf{d}^*_{max} = (0, 0, \ldots, 1, 0, 0 \ldots, 0)$ and

$$\Lambda \mathbf{d}^*_{max} = \lambda_{max} \mathbf{d}^*_{max}$$

then

$$\mathbf{d}^*_{max} \cdot \Lambda \mathbf{d}^*_{max} = \lambda_{max}$$

and $\mathbf{d}^*_{max}$ maximizes the quadratic form $\mathbf{d}^* \cdot \Lambda \mathbf{d}^*$.

The first principal component, that is the vector $\mathbf{d}_{max}$ that maximizes the quadratic form $\mathbf{d} \cdot C\mathbf{d}$, is therefore

$$\mathbf{d}_{max} = R^T \mathbf{d}^*_{max}$$

and it can be easily seen that satisfies the eigenvector equation:

$$C\mathbf{d}_{max} = \lambda_{max}\mathbf{d}_{max}$$

The first principal component is therefore the eigenvector of the correlation matrix corresponding to the maximum eigenvalue.

Case $k > 1$

It can be shown that *the first k principal components are the eigenvectors of the correlation matrix C corresponding to the first k largest eigenvalues*.

Try this at home:

The approximation error of the first k principal components $\{\mathbf{d}_i\}_{i=1}^{k}$

$$E(k) = \sum_{\alpha=1}^{N} (\mathbf{x}_\alpha - \sum_{i=1}^{k} (\mathbf{x}_\alpha \cdot \mathbf{d}_i)\mathbf{d}_i)^2$$

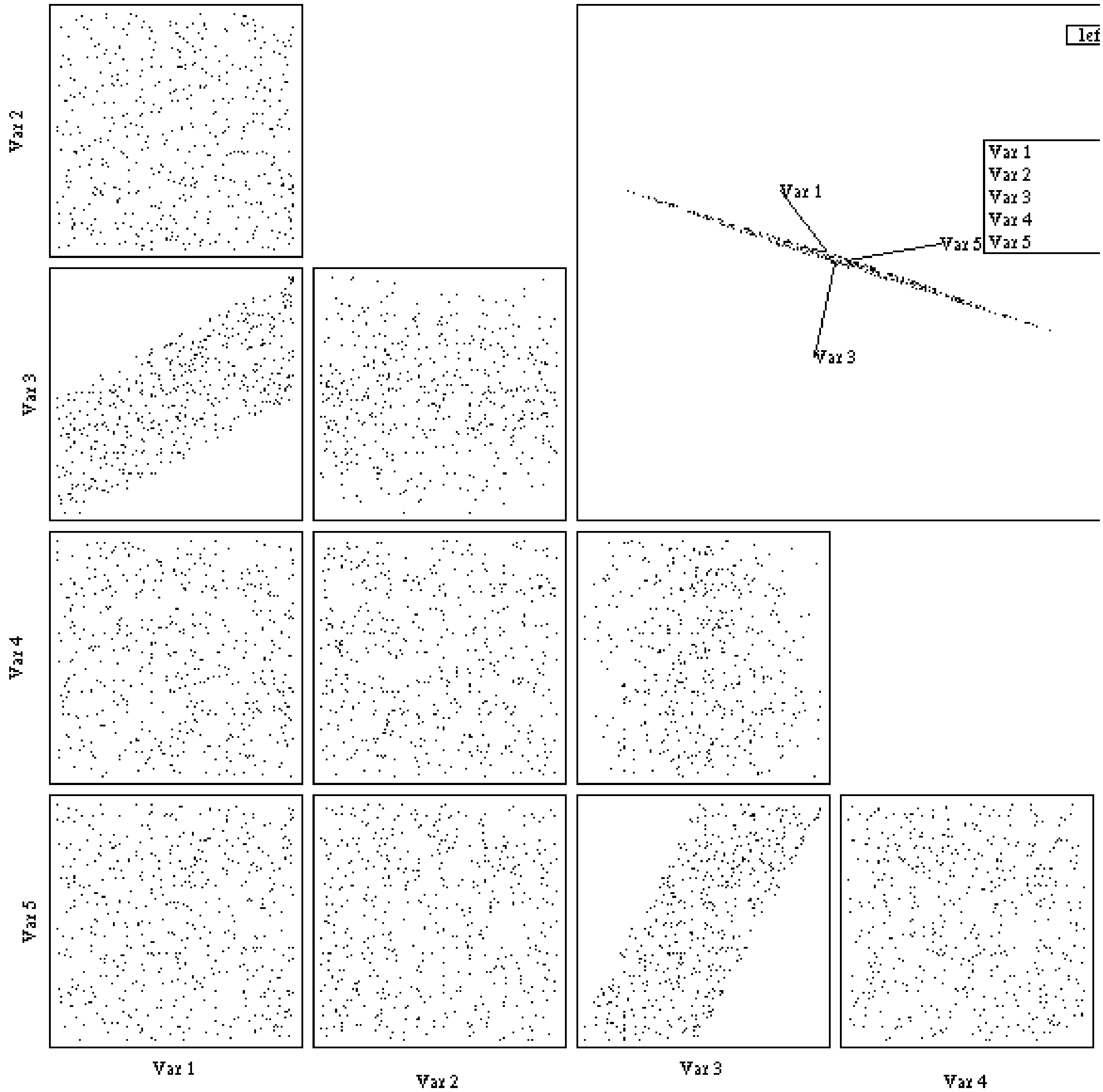
is given by

$$E(k) = \sum_{i=k+1}^{N} \lambda_i$$

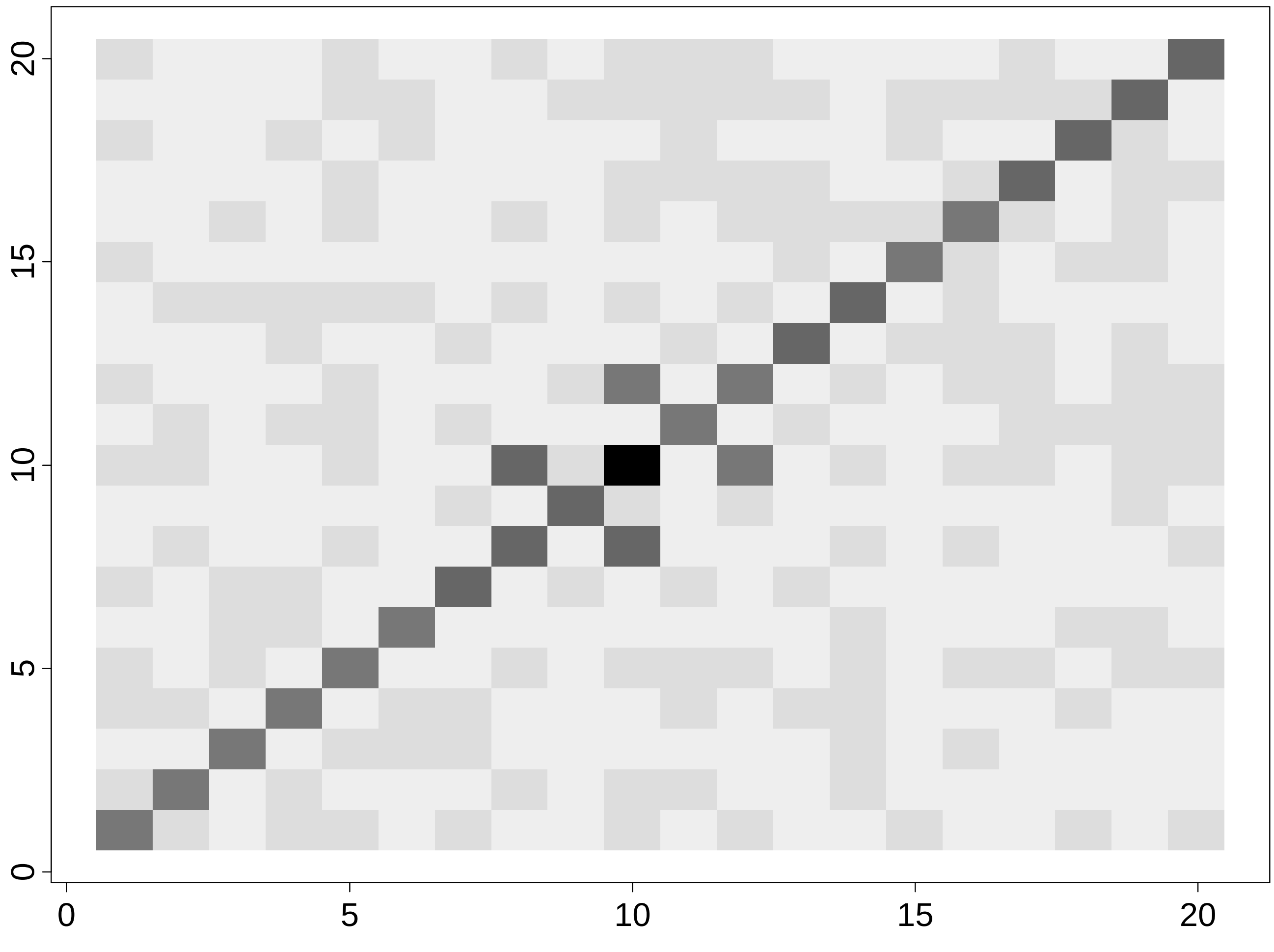
where λ_i are the eigenvalues of the correlation matrix C.

Hint: when $k = N$ the error is zero because the correlation matrix C is symmetric, and its eigenvectors form a complete basis.

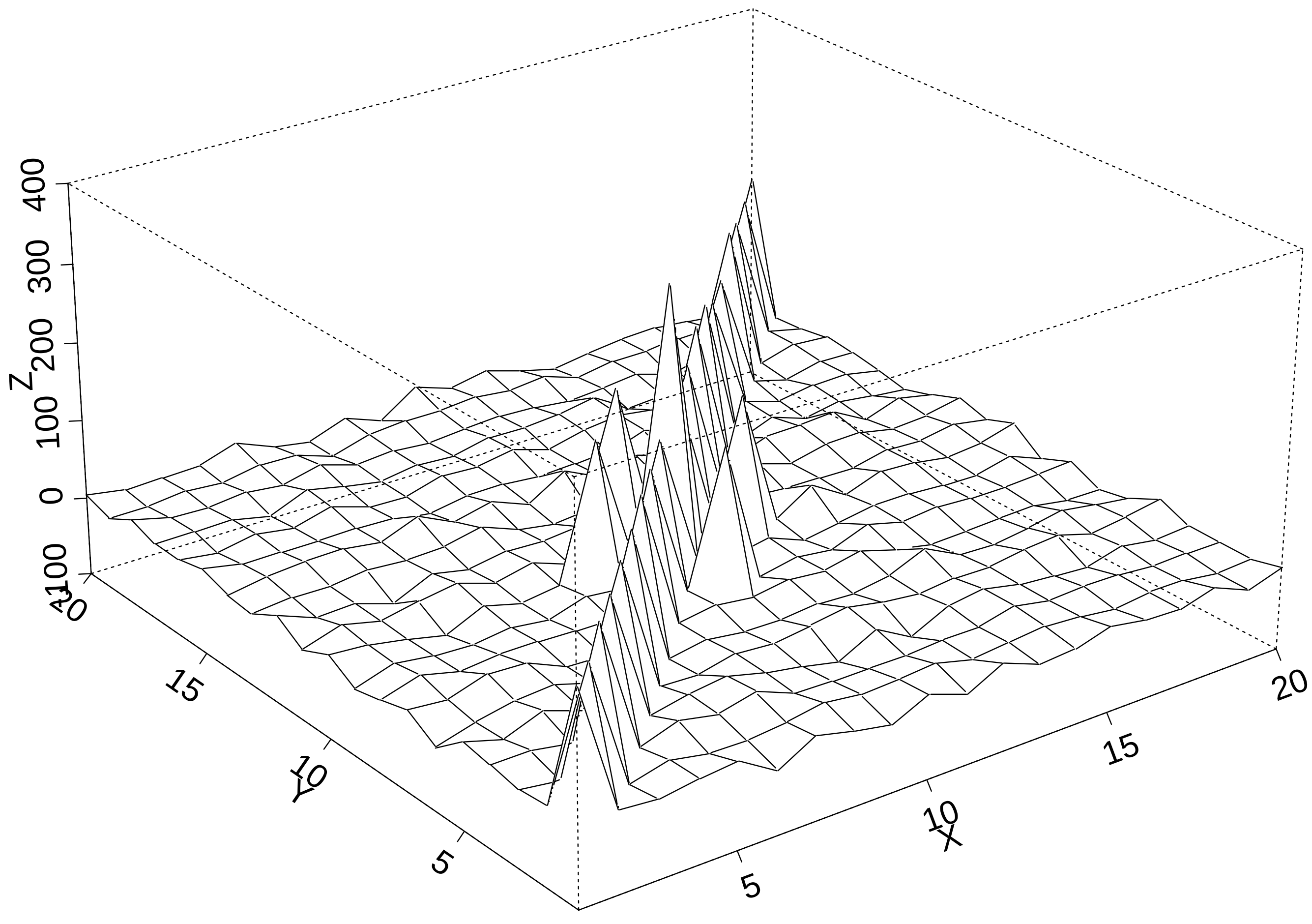
An example in 20 dimensions: $\mathbf{x} \in R^{20}$, $x^{10} = x^8 + x^{12}$, 500 points.



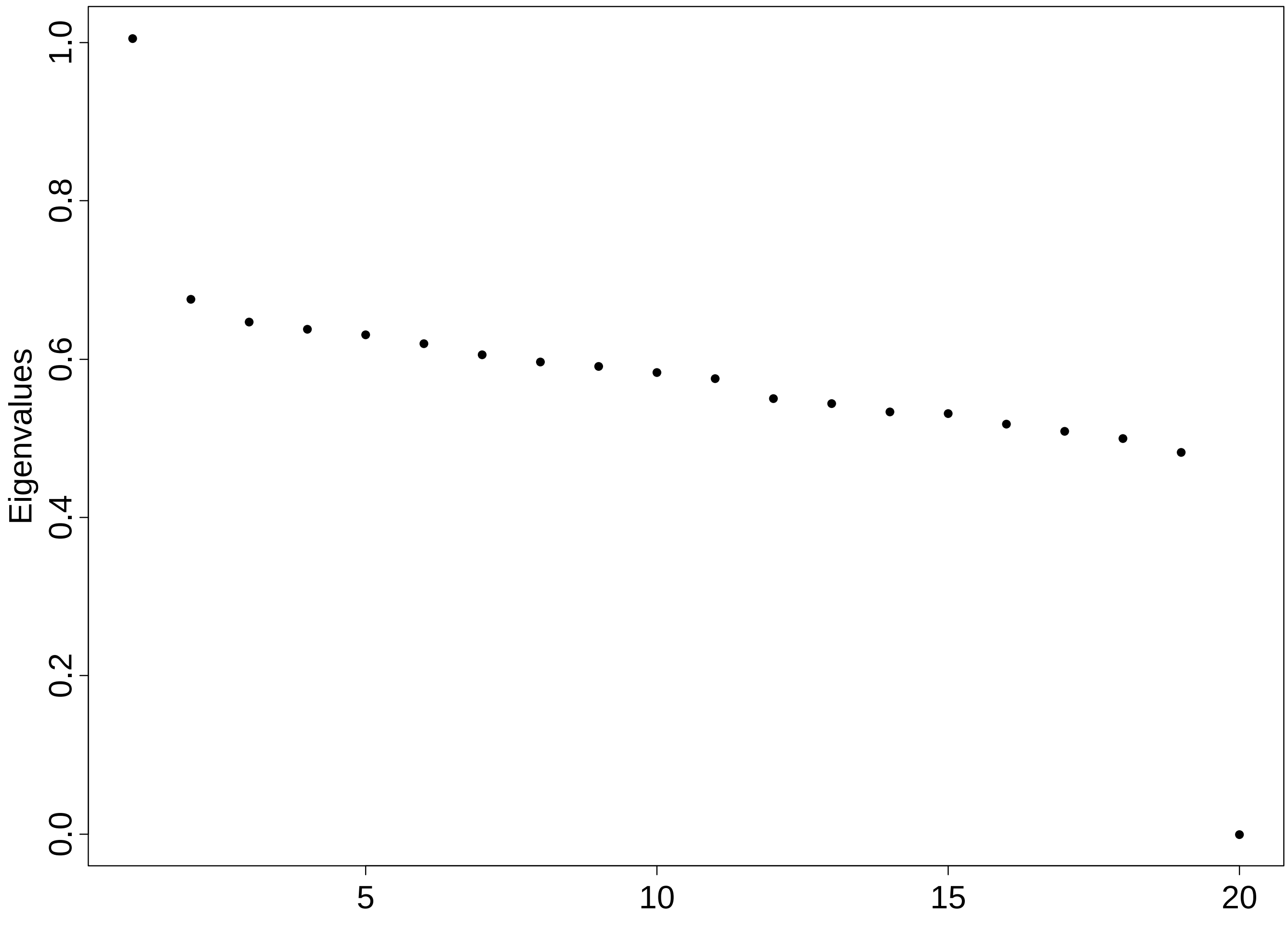
Correlation matrix



Correlation matrix



Eigenvalues of the correlation matrix



Another point of view on PCA

Let $\{\mathbf{x}_\alpha\}_{\alpha=1}^{N}$ be a set of points in R^d.

Problem: *find the unit vector* $\mathbf{d}$ *for which the projection of the data on* $\mathbf{d}$ *has the maximum variance.*

Assuming that the data points have zero mean, this means:

$$\max_{\|\mathbf{d}\|=1} \frac{1}{N} \sum_{\alpha=1}^{N} (\mathbf{x}_\alpha \cdot \mathbf{d})^2$$

This is formally the same as finding the *first principal component*.

The second principal component is the projection of maximum variance in a subspace orthogonal to the first principal component.

...

The k-th principal component is the projection of maximum variance in a subspace wich is orthogonal to the subspace spanned by the first $k-1$ principal components.

Empirical risk minimization

Current approach:

DATA

$$\Downarrow$$

FUNCTION

$$\Downarrow$$

FUNCTION VALUES AT POINTS

Memory Based Learning

DATA

$\Downarrow$

FUNCTION VALUES AT POINTS

No “model of the world” is explicitely built

Memory Based Learning

Main idea: at each point fit a very simple **local** model, based only on neighboring data points.

Examples of simple models:

- Constant;
- Linear;

Nearest Neighbor for classification

The simplest example of memory based learning for classification is the *Nearest Neighbor* technique:

A point is classified as belonging to the same class of the point in the data set that is nearest to him

Nearest Neighbor for classification

Let $\{(\mathbf{x}_i, \omega_i)\}_{i=1}^{N}$ the data set, where ω_i is either 0 or 1. The nearest neighbor rule classifies a point $\mathbf{x}$ as follows:

$$\omega(\mathbf{x}) = \omega_{k(\mathbf{x})}$$

where

$$k(\mathbf{x}) = \arg\min_k \|\mathbf{x} - \mathbf{x}_k\|$$

Nearest Neighbor for regression

The simplest example of memory based learning for regression is the *Nearest Neighbor* technique:

the value of a function at a point that does not belong to the data set is the value of its closest data point.

How well does NN work?

Let R be the probability of error of the NN rule and let R^* be the Bayes probability of error. Then, *when the number of data points is very large*:

$$R^* \leq R \leq 2R^*(1 - R^*)$$

Therefore *any other decision rule can cut the probability of error by at most one half*.

Bayes decision rule

Suppose the input data $\mathbf{x}$ can belong to one of M classes: $(\omega_1, \ldots, \omega_M)$.

$P(\mathbf{x}|\omega_i)$ = probability distribution of the input within class ω_i;

$P(\omega_i)$ = a priori probability of class ω_i;

$P(\omega_i|\mathbf{x})$ = posterior probability of class ω_i;

Suppose f is a classifier, whose output is one of the labels $(\omega_1, \ldots, \omega_M)$.

Let $d(\omega_i, \omega_j)$ be the *zero-one loss function*:

$$d(\omega_i, \omega_j) = 1 - \delta_{ij}$$

The average error of the classifier f is:

$$E[f] = \sum_{i=1}^{M} \int d\mathbf{x} \; d(\omega_i, f(\mathbf{x}))P(\mathbf{x})P(\omega_i|\mathbf{x})$$

Which is the optimal classifier?

We have to solve:

$$\min_f E[f] = \min_f \int d\mathbf{x} P(\mathbf{x}) \left[\sum_{i=1}^{M} d(\omega_i, f(\mathbf{x})) P(\omega_i|\mathbf{x}) \right]$$

and therefore, for every given $\mathbf{x}$ we have to solve:

$$\min_f \sum_{i=1}^{M} d(\omega_i, f(\mathbf{x})) P(\omega_i|\mathbf{x})$$

For a given $\mathbf{x}$ we have $f(\mathbf{x}) = \omega_k$ for some $k = k(\mathbf{x})$. Therefore we have to solve

$$\min_k \sum_{i=1}^{M} (1 - \delta_{ik}) P(\omega_i | \mathbf{x}) =$$

$$= \max_k P(\omega_k | \mathbf{x})$$

Choose the class with highest a posteriori probability!

Bayes decision rule

If a *zero-one* loss criterion is adopted, then the best possible decision at point $\mathbf{x}$ is the one which maximizes the a posteriori probability $P(\omega_k|\mathbf{x})$.

The average error corresponding to this decision rule is the *Bayes classification error*.

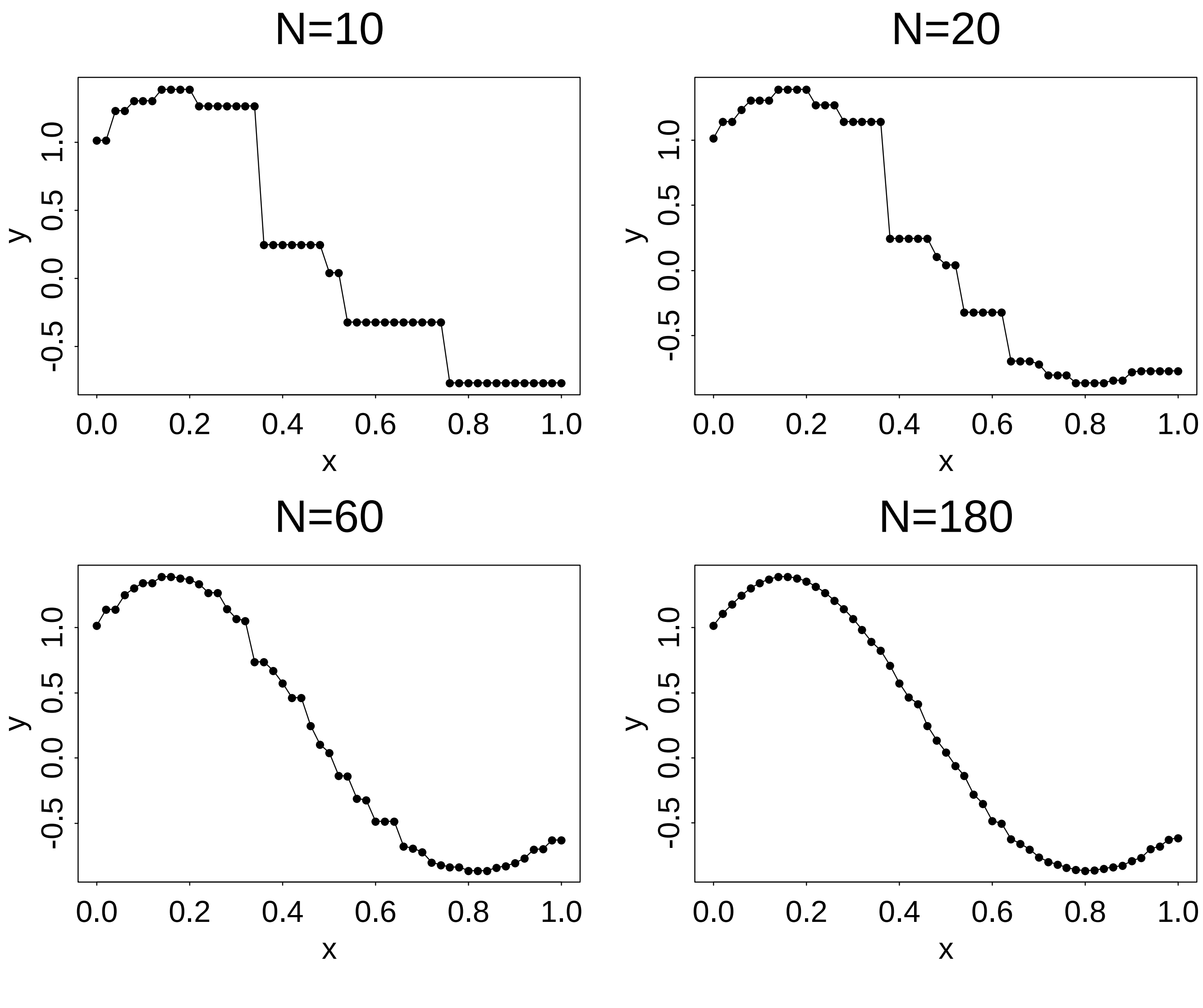
Rates of convergence of NN

How fast does the error go to zero?

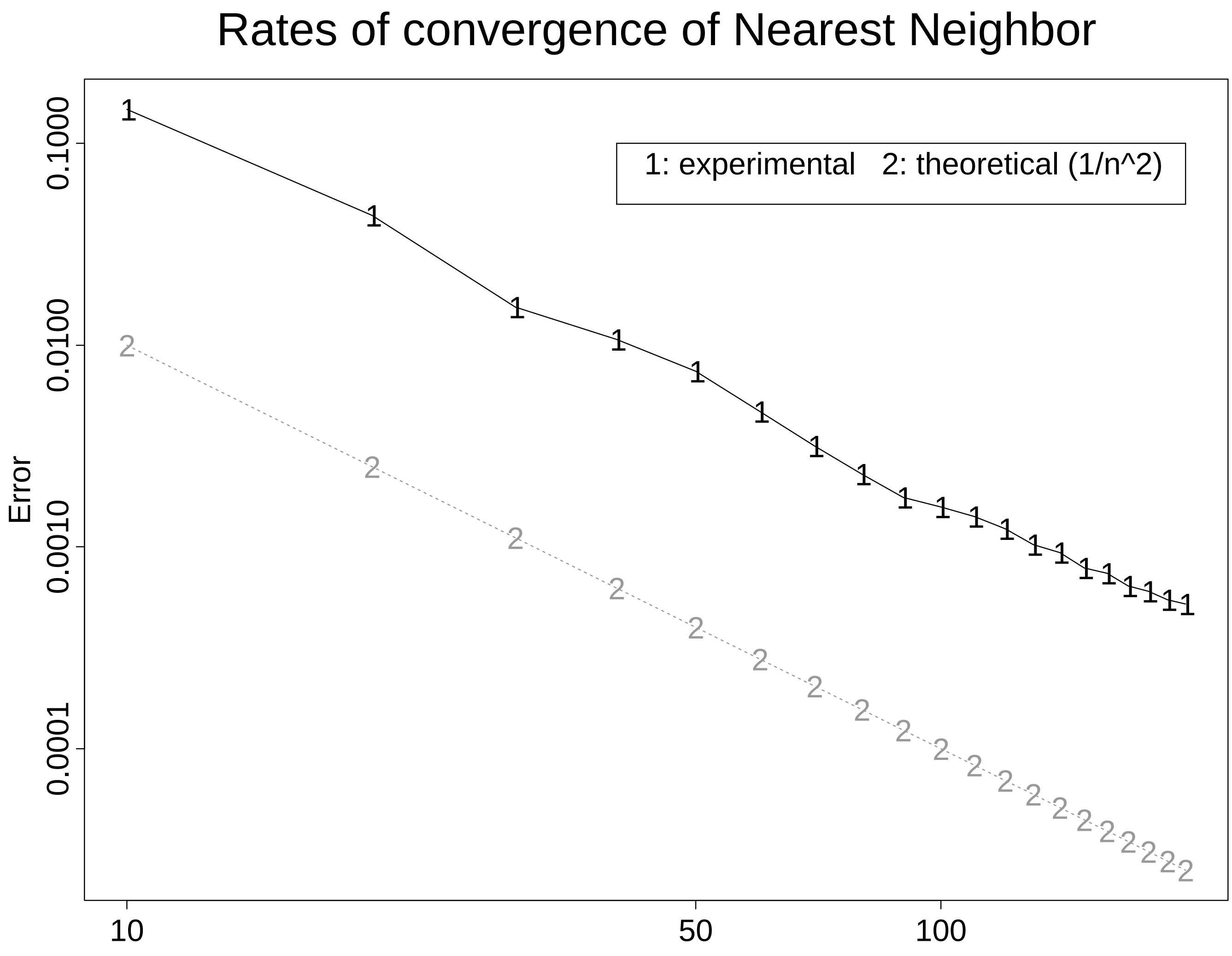
Fukunaga and Hummels (1987) prove that the error is of the order of $O(n^{-\frac{2}{d}})$ for classification problems;

Posner and Kulkarni (1994) prove a similar result for the more general case of regression.

A one dimensional example



Comparing the experimental rate of convergence and the theoretical rate $O(\frac{1}{n^2})$.



Running mean and k-nearest neighbor

At each point compute the mean of the values of the closest k points.

$$f(\mathbf{x}) = \theta(\mathbf{x})$$

the "local average" $\theta(\mathbf{x})$ is found as

$$\theta(\mathbf{x}) = \arg\min_{\theta} \sum_{i \in \mathcal{N}_{\mathbf{x}}} (y_i - \theta)^2$$

where $\mathcal{N}_{\mathbf{x}}$ is the set of k closest points of $\mathbf{x}$.

$$f(\mathbf{x}) = \frac{1}{\#\mathcal{N}_{\mathbf{x}}} \sum_{i \in \mathcal{N}_{\mathbf{x}}} y_i$$

Running line

At each point compute a linear approximation based on the values of the closest k points.

$$f(\mathbf{x}) = \mathbf{a}(\mathbf{x}) \cdot \mathbf{x} + b(\mathbf{x})$$

That is, for each given $\mathbf{x}$ we find:

$$\{\mathbf{a}(\mathbf{x}), b(\mathbf{x})\} = \arg\min_{\mathbf{a},b} \sum_{i \in \mathcal{N}_{\mathbf{x}}} (y_i - \mathbf{a} \cdot \mathbf{x}_i - b)^2$$

where $\mathcal{N}_{\mathbf{x}}$ is the set of k closest points of $\mathbf{x}$.

Running mean (also called “moving average”) and running line tend to produce non smooth results, because the sum over the k closest point does not depend smoothly on $\mathbf{x}$.

$$\Downarrow$$

$$\sum_{i \in \mathcal{N}_{\mathbf{x}}} \Rightarrow \sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i)$$

where $K(\mathbf{x})$ is a bell-shaped, local kernel (a Gaussian, or a box)

Kernel regression

The approximation is "locally constant":

$$f(\mathbf{x}) = \theta(\mathbf{x})$$

and the "local constant" $\theta(\mathbf{x})$ is found as

$$\theta(\mathbf{x}) = \arg\min_{\theta} \sum_{i=1}^{N} K_h(\mathbf{x} - \mathbf{x}_i)(y_i - \theta)^2$$

where $K_h(\mathbf{x} - \mathbf{x}_i)$ is a kernel of "bandwidth" h. The solution is

$$\boxed{f(\mathbf{x}) = \frac{\sum_{i=1}^{N} y_i K_h(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{N} K_h(\mathbf{x} - \mathbf{x}_i)}}$$

Frequently used one dimensional kernels

$$K_h(x) = e^{-\frac{x^2}{h^2}} \quad \text{(Gaussian)}$$

$$K_h(x) = \begin{cases} 1 & \text{if } |x| \le h \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Box)}$$

$$K_h(x) = \begin{cases} \frac{x}{h} + 1 & \text{if } -h \le x \le 0 \\ -\frac{x}{h} + 1 & \text{if } 0 \le x \le h \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Hat)}$$

$$K_h(x) = \begin{cases} \frac{1}{h^2}(h^2 - x^2) & \text{if } |x| \le h \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Epanechnikov)}$$

Kernel regression

$$\boxed{f(\mathbf{x}) = \frac{\sum_{i=1}^{N} y_i K_h(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{N} K_h(\mathbf{x}-\mathbf{x}_i)}}$$

It can be rewritten as

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) y_i}{\sum_{i=1}^{N} w_i(\mathbf{x})}$$

where

$$w_i(\mathbf{x}) = K_h(\mathbf{x} - \mathbf{x}_i)$$

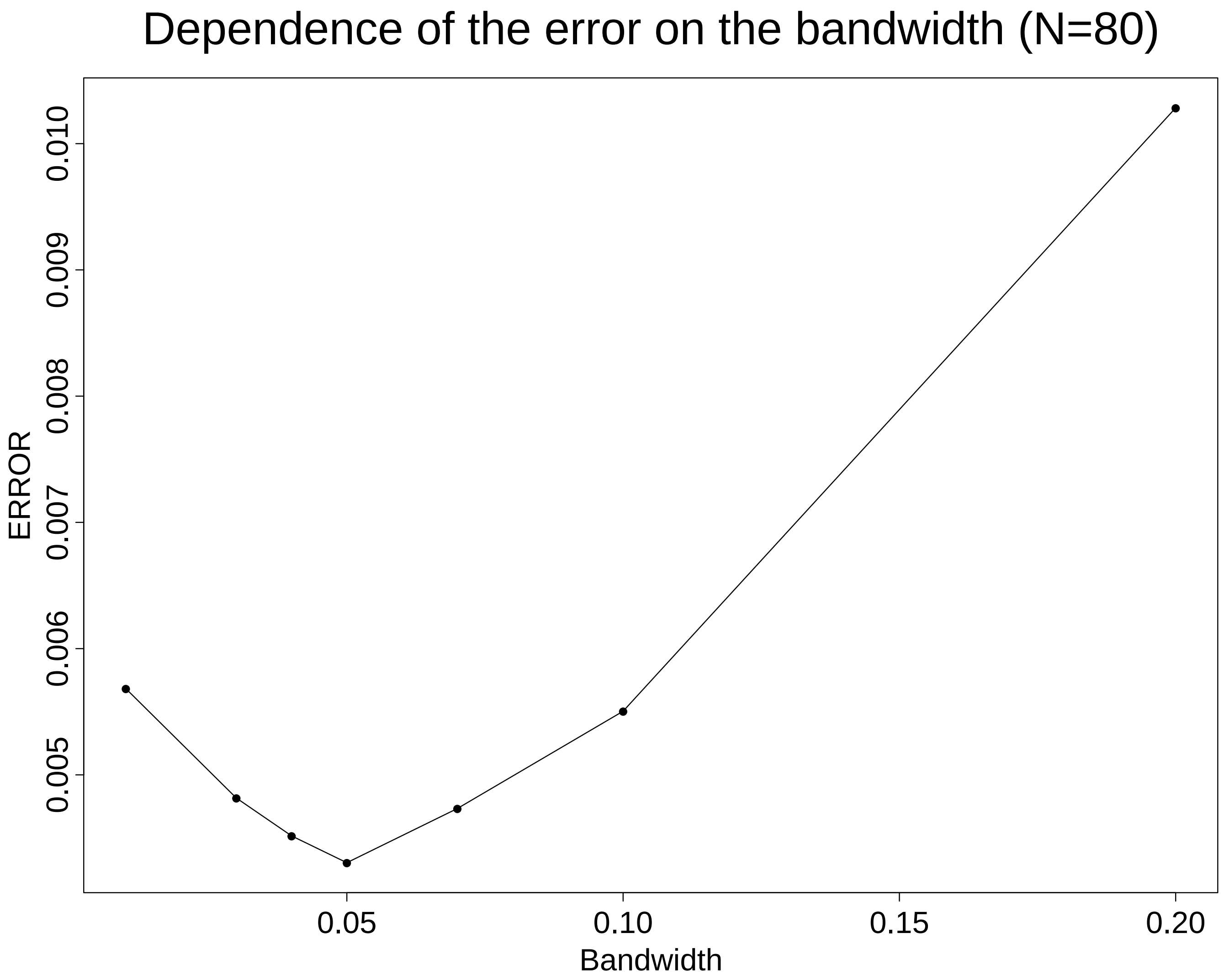
The value of f at point $\mathbf{x}$ is a weighted average of the values at the data points.

Kernel regression

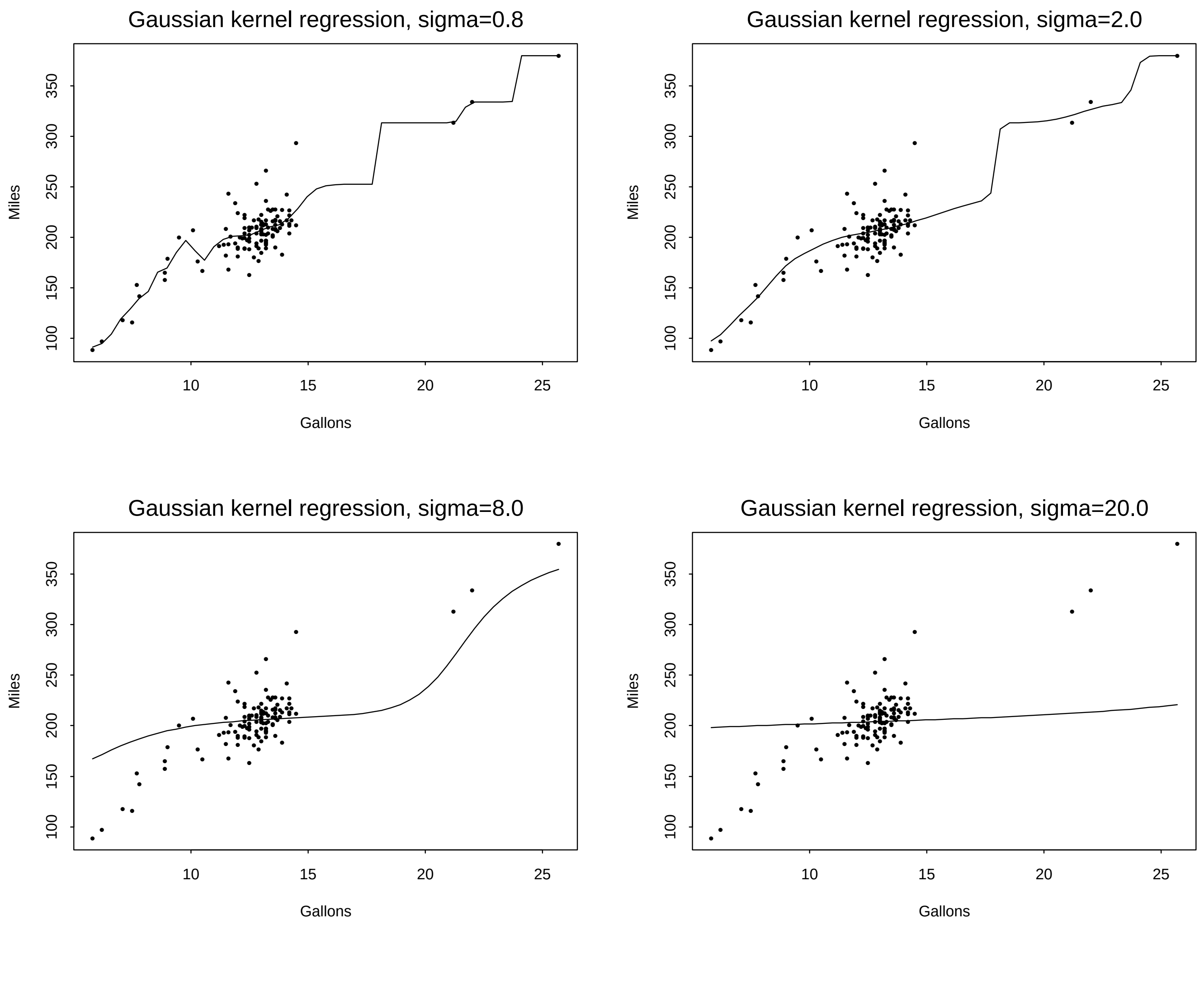
One of the difficulties of kernel regression is that it is sensitive to the choice of the “bandwidth” h, or “smoothing parameter”.

Intuitively

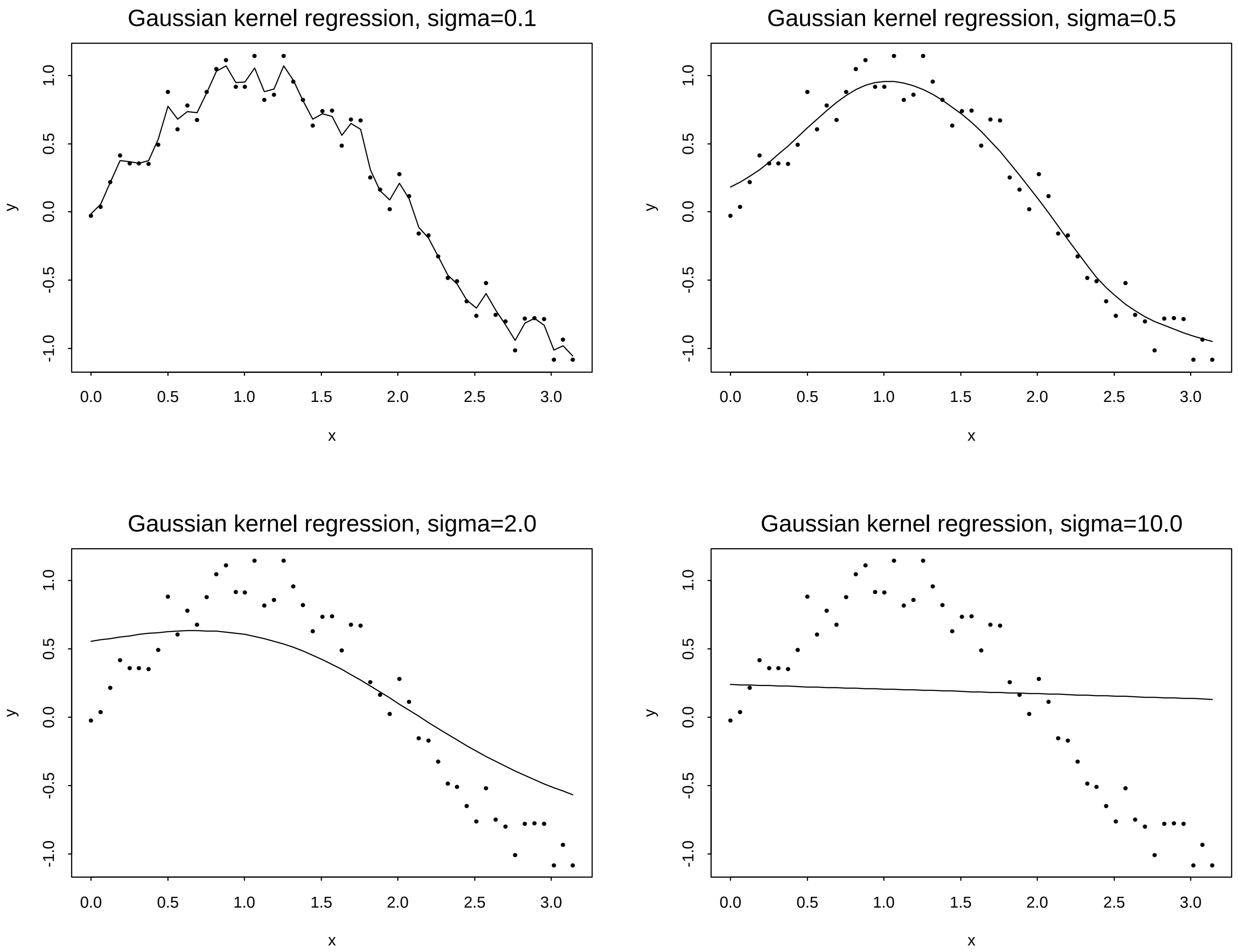
$$h \to 0 \quad \text{as} \quad N \to +\infty$$


Dependence of the error on the bandwidth (N=80)
ERROR
0.005
0.006
0.007
0.008
0.009
0.010
0.05
0.10
0.15
0.20
Bandwidth

Fuel consumption data (miles versus gallons of fuel). 1974 car owned by R.A. Becker



Sin function with additive uniform noise



Convergence of kernel regression

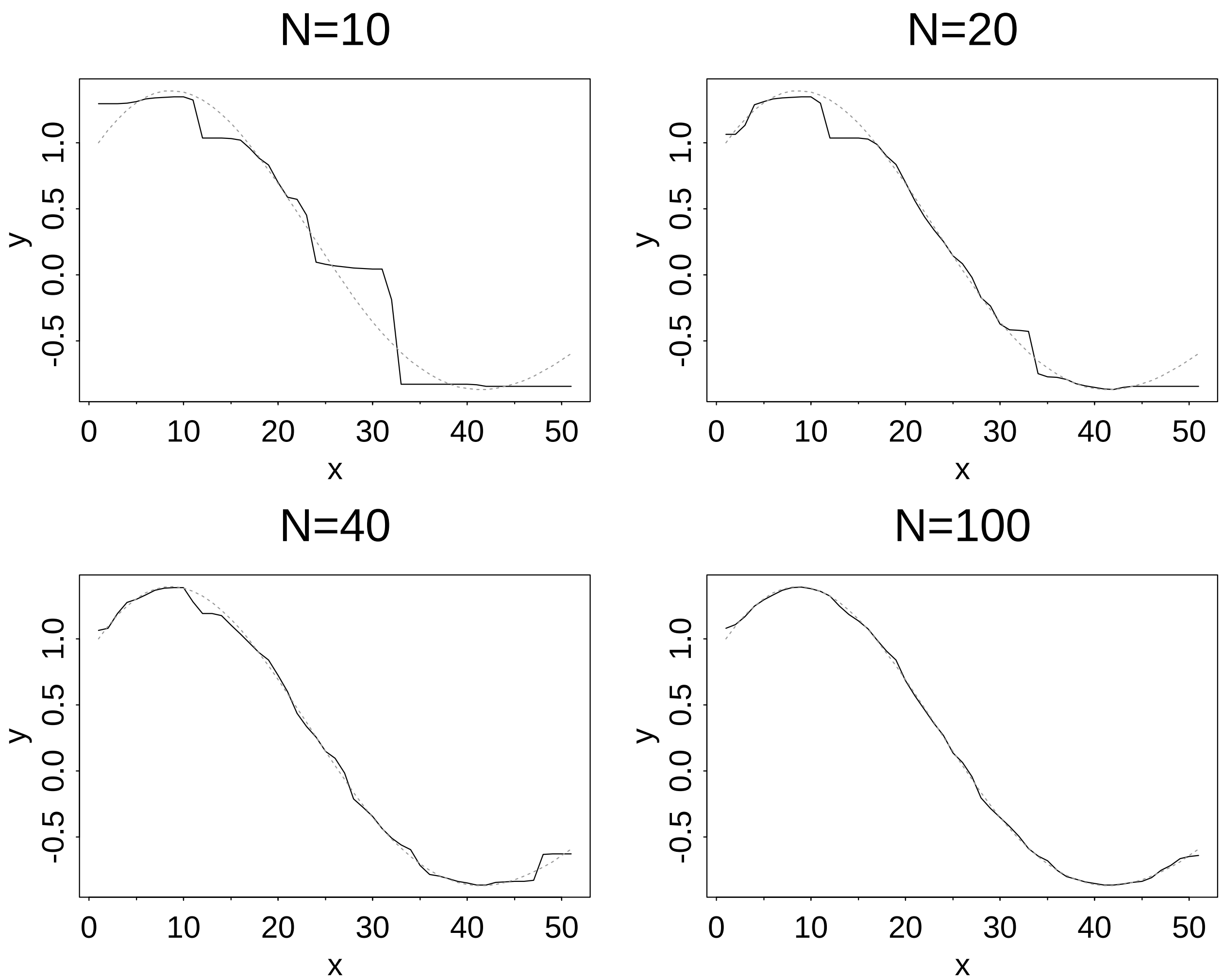
Let $\{x_i, y_i\}$ be a set of noisy data and let $h(x)$ be the regression function. Assume that

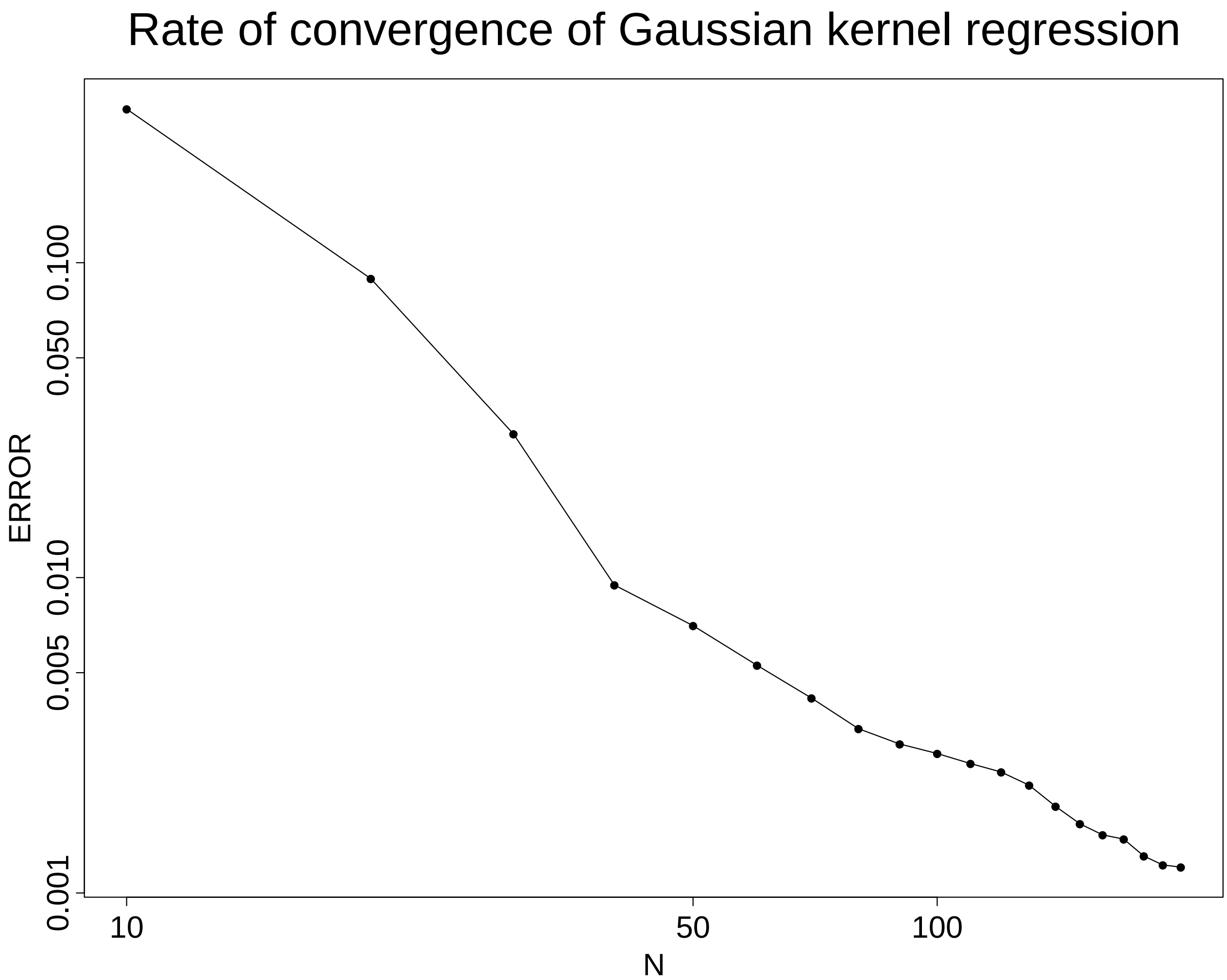
1. $\int |K(x)|dx < +\infty$,
2. $\lim_{|x|\to\infty} xK(x) = 0$,
3. $E[y^2] < +\infty$
4. $\boxed{n \to +\infty \ , \quad h_n \to 0 \ , nh_n \to +\infty}$

Then, for each x,

$$\frac{\sum_{i=1}^{N} y_i K_h(x - x_i)}{\sum_{i=1}^{N} K_h(x - x_i)} \to h(x)$$

in probability.


N=10
N=20
N=40
N=100
y
x
1.0
0.5
0.0
-0.5
0
10
20
30
40
50



Rate of convergence of Gaussian kernel regression
ERROR
0.100
0.050
0.010
0.005
0.001
10
50
100
N

Another interpretation of kernel regression

Kernel regression can also be derived in a different way, that involves the "Parzen windows" technique for density estimation.

Our goal is to minimize the expected risk

$$I[f] = \int d\mathbf{x} dy P(\mathbf{x}, y)(y - f(\mathbf{x}))^2 \ .$$

but this cannot be done, since the probability P is unknown, and usually the empirical risk I_{emp} is minimized instead:

$$I_{\text{emp}}[f] = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2$$

Alternative technique:

1. Approximate the unknown probability distribution $P(\mathbf{x}, y)$ with a Parzen window approximation $P^*(\mathbf{x}, y)$;

2. substitute P^* in the expected risk $I[f]$, obtaining an approximation of the expected risk $I^*[f]$:

$$I^*[f] = \int d\mathbf{x} dy P^*(\mathbf{x}, y)(y - f(\mathbf{x}))^2 \ .$$

3. minimize the approximated expected risk I^* over all the possible estimators:

$$\min_{f(\mathbf{x})} I^*[f]$$

Parzen windows

If regularization theory is applied to the problem of density estimation, the *Parzen windows* technique is recovered (Parzen, 1962).

A Parzen window estimator P^* for the probability distribution of a set of data $\{\mathbf{z}_i\}_{i=1}^N$ has the form:

$$P^*(\mathbf{z}) = \frac{1}{Nh} \sum_{i=1}^{N} \Phi \left(\frac{\mathbf{z} - \mathbf{z}_i}{h} \right)$$

where Φ is an appropriate kernel, for examples a Gaussian.

Density estimation as an ill-posed problem

A probability density $f(t)$ is the derivative of the distribution function $F(x) = P(t \leq x)$.

Since $f(x) = F'(x)$ we can also write

$$\int_{-\infty}^{+\infty} dt\ \theta(x-t) f(t) = F(x)$$

Although F is unknown, an *empirical estimate* of it is available:

$$F_l(x) = \frac{1}{l} \sum_{i=1}^{l} \theta(x - t_i)$$

Regularization approach: we want a function f for which

$$\int_{-\infty}^{+\infty} dt \; \theta(x-t) f(t) \approx F_l(x)$$

but also a function f which is *smooth*, that is for which

$$\phi[f] \quad \text{is small}$$

where $\phi[f]$ is a smoothness functional.

Regularization approach: minimize the following functional:

$$H[f] = \int dx \left[\int_{-\infty}^{+\infty} dt \ \theta(x-t) f(t) - F_l(x) \right]^2 + \lambda \phi[f]$$

where $\phi[f]$ is the usual smoothness functional:

$$\phi[f] = \int \ d\omega \ \frac{|\tilde{f}(\omega)|^2}{\tilde{G}(\omega)}$$

Homework

Prove that the function that minimizes the functional

$$H[f] = \int dx \left[\int_{-\infty}^{+\infty} dt\ \theta(x-t) f(t) - F_l(x) \right]^2 + \lambda \phi[f]$$

has the form:

$$f(t) = \frac{1}{l} \sum_{i=1}^{l} \Phi(t - t_i)$$

where the function Φ can be uniquely determined from the smoothness functional ϕ.

It is sufficient consider the case in which $\phi[f] = \int dx (f''(x))^2$.

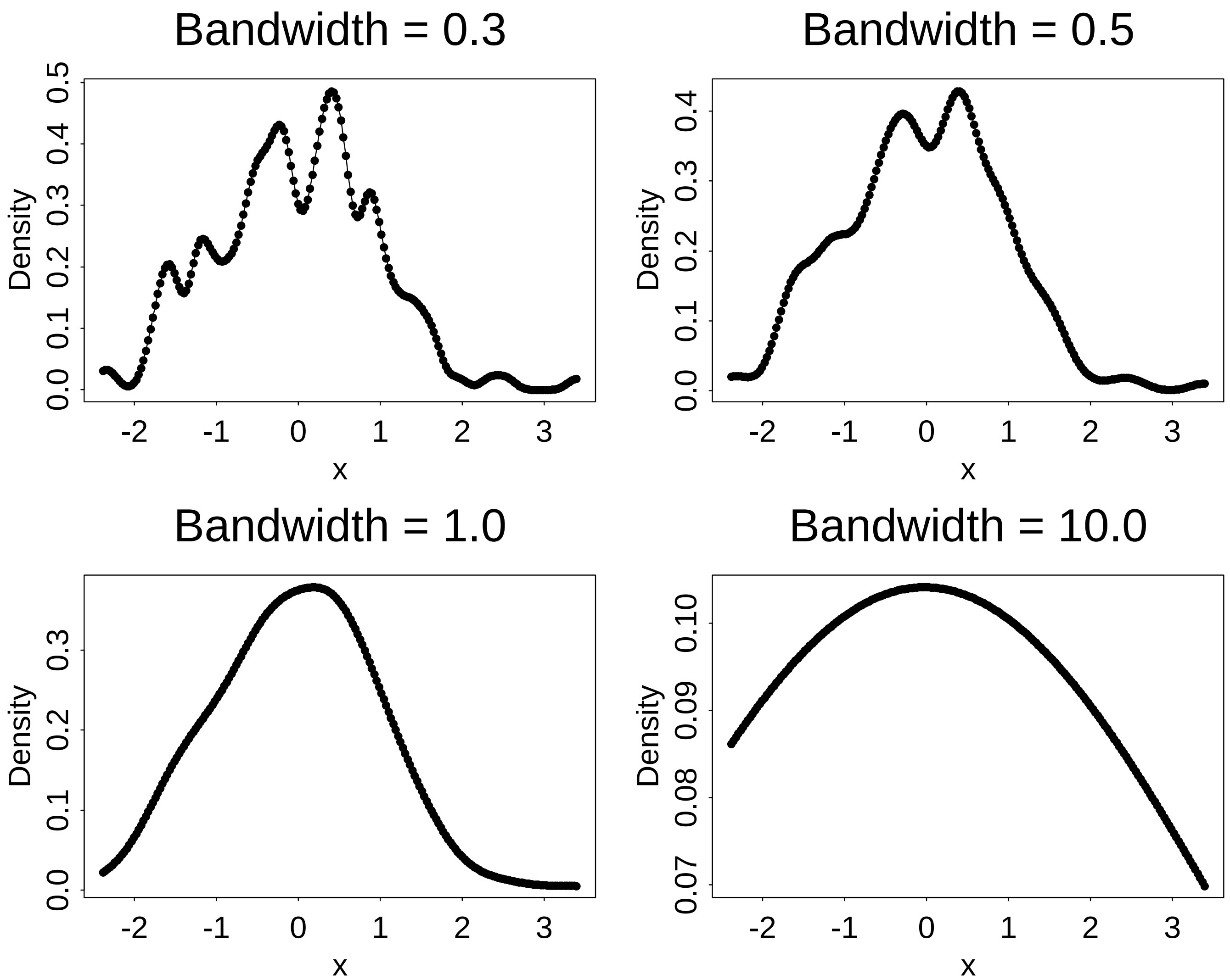
Parzen windows

1. Very simple to implement;

2. Slow convergence;

3. Choice of the “bandwidth” parameter may be problematic;

An example with $N = 300$ and a normal distribution



Our goal is to minimize the expected risk

$$I[f] = \int d\mathbf{x} dy P(\mathbf{x}, y)(y - f(\mathbf{x}))^2 \ .$$

but this cannot be done, since the probability P is unknown, and usually the empirical risk I_{emp} is minimized instead:

$$I_{\text{emp}}[f] = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2$$

Alternative technique:

1. Approximate the unknown probability distribution $P(\mathbf{x}, y)$ with a Parzen window approximation $P^*(\mathbf{x}, y)$;

2. substitute P^* in the expected risk $I[f]$, obtaining an approximation of the expected risk $I^*[f]$:

$$I^*[f] = \int d\mathbf{x} dy P^*(\mathbf{x}, y)(y - f(\mathbf{x}))^2 \ .$$

3. minimize the approximated expected risk I^* over all the possible estimators:

$$\min_{f(\mathbf{x})} I^*[f]$$

In our case the random variable $\mathbf{z}$ is $\mathbf{z} = (\mathbf{x}, y)$, and we choose a kernel of the following form

$$\Phi(\mathbf{z}) = K(\|\mathbf{x}\|)K(y)$$

where K is some radial kernel (Gaussian, for example). The Parzen window estimator to $P(\mathbf{x}, y)$ is therefore:

$$P^*(\mathbf{x}, y) = \frac{1}{N}\sum_{i=1}^{N} K\left(\|\mathbf{x} - \mathbf{x}_i\|\right) K(y - y_i)$$

An approximation to the expected risk is therefore obtained as:

$$I^*[f] = \frac{1}{N}\sum_{i=1}^{N} \int d\mathbf{x}dyK\left(\|\mathbf{x} - \mathbf{x}_i\|\right) K(y-y_i)(y-f(\mathbf{x}))^2$$

The solution to the problem

$$\min_{f(\mathbf{x})} \left[\sum_{i=1}^{N} \int \; d\mathbf{x} dy K \left(\|\mathbf{x} - \mathbf{x}_i\| \right) K(y - y_i)(y - f(\mathbf{x}))^2 \right]$$

is

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{N} y_i K(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{i=1}^{N} K(\|\mathbf{x} - \mathbf{x}_i)\|} \; .$$

that is a kernel estimator.

Memory based learning versus Regularization networks

- kernel regression: no complex global model of the world is assumed. Many simple local models instead.

- Regularization networks: fairly complex global model of the world

Are these two techniques related?

Least square Regularization networks

A model of the form

$$f(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\mathbf{x} - \mathbf{t}_\alpha)$$

is assumed and the parameters c_α and $\mathbf{t}_\alpha$ are found by

$$\min_{c_\alpha, \mathbf{t}_\alpha} E[\{c_\alpha\}, \{\mathbf{t}_\alpha\}]$$

where

$$E[\{c_\alpha\}, \{\mathbf{t}_\alpha\}] = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2$$

Least square Regularization networks

The coefficients c_α and the centers $\mathbf{t}_\alpha$ have to satisfy the conditions:

$$\frac{\partial E}{\partial c_\alpha} = 0 \ , \qquad \frac{\partial E}{\partial \mathbf{t}_\alpha} = 0 \quad \alpha = 1, \ldots, n$$

The equation for the coefficients gives:

$$c_\alpha = \sum_{i=1}^{N} H_{\alpha i} y_i$$

where

$$H = (G^T G)^{-1} G^T \ , \qquad G_{i\alpha} = G(\mathbf{x}_i - \mathbf{t}_\alpha)$$

Substituting the expression for the coefficients in the regularization network we obtain

$$f(\mathbf{x}) = \sum_{\alpha=1}^{n} \sum_{i=1}^{N} H_{\alpha i} y_i G(\mathbf{x} - \mathbf{t}_\alpha) = \sum_{i=1}^{N} y_i \sum_{\alpha=1}^{n} H^T_{i\alpha} G(\mathbf{x} - \mathbf{t}_\alpha)$$

$$\boxed{f(\mathbf{x}) = \textstyle\sum_{i=1}^{N} y_i b_i(\mathbf{x})}$$

where we have defined

$$b_i(\mathbf{x}) = \sum_{\alpha=1}^{n} H^T_{i\alpha} G(\mathbf{x} - \mathbf{t}_\alpha)$$

The representation above is called the "dual representation" and the basis function $b_i(\mathbf{x})$ are called "dual kernels".

Memory based learning versus Regularization networks

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{N} y_i b_i(\mathbf{x})}$$ *Regularization networks*

$$\Updownarrow$$

$$\boxed{f(\mathbf{x}) = \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) y_i}{\sum_{i=1}^{N} w_i(\mathbf{x})}}$$ *Kernel regression*

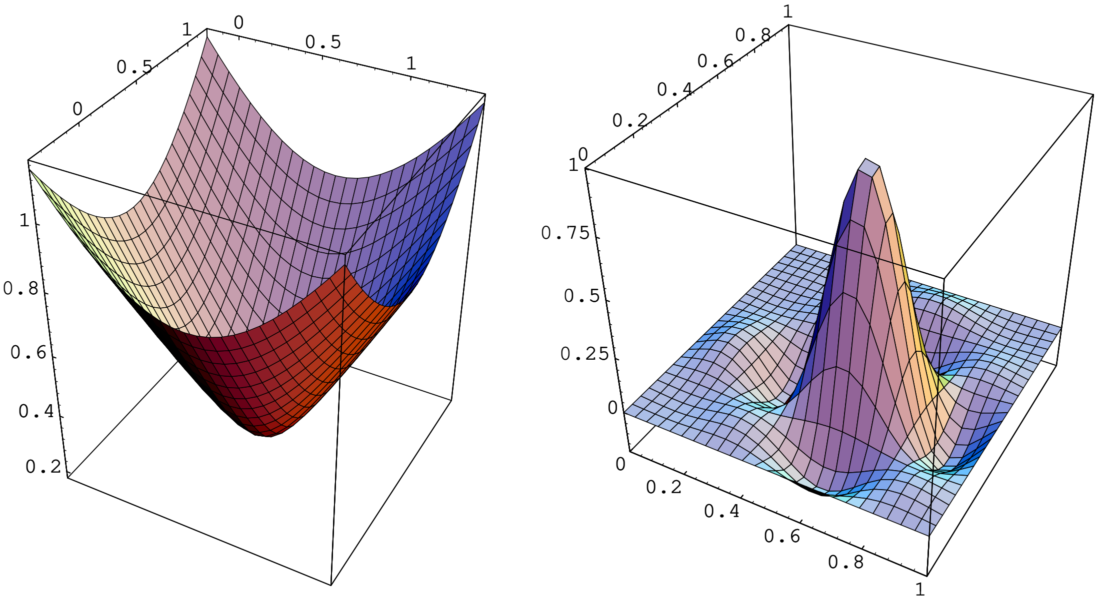
In both cases the value of f at point $\mathbf{x}$ is a weighted average of the values at the data points.

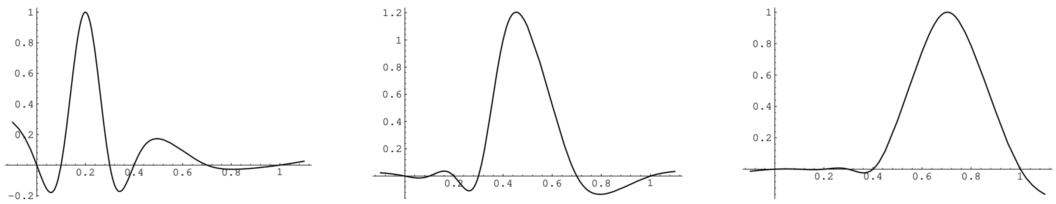
Memory based learning versus Regularization networks

In kernel regression the kernels are chosen to be bell-shaped.

If regularization networks and kernel regression are similar the equivalent kernels b_i should be bell-shaped too.

Equivalent kernels for the standard regularization solution with multiquadric basis functions





From (Girosi, Jones and Poggio, 1993)

The equivalent kernels for standard regularization theory are more esily understood in the case in which an infinite number of data is available, that is the function g to be approximated is known:

$$H[f] = \int_{R^d} d\mathbf{x} \ (f(\mathbf{x}) - g(\mathbf{x}))^2 + \lambda\phi[f]$$

In this case the solution is simply

$$f = g * H$$

where H is *always* a bell-shaped function such that

$$\tilde{H}(\mathbf{s}) = \frac{\tilde{G}(\mathbf{s})}{\tilde{G}(\mathbf{s}) + \lambda}$$

Memory based learning versus Regularization networks

It has been proved (Silverman, 1984) that the equivalent kernels for one dimensional cubic splines are bell-shaped and very well localized.

The bandwidth of the equivalent kernel is inversely proportional to the density of the data.

This result can be generalized to higher order splines (Silverman, 1984) and probably to all regularization networks.

References on kernel regression and nearest neighbor

1. T.M. Cover and P.E. Hart. Nearest neighbor pattern classifications. *IEEE transaction on information theory*, 13(1):21—27, 1967.

2. W. Härdle. *Applied nonparametric regression*, volume 19 of *Econometric Society Monographs*. Cambridge University Press, 1990.

3. T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.

4. L. Devroye and L. Györfi and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Number 31 in Applications of mathematics. Springer, New York, 1996.