

Nonparametric Statistics

0 Introduction

General nonparametric regression model:

$$Y_j = m(X_j) + \varepsilon_j, \quad j = 1, \dots, N, \quad X_j, Y_j \in \mathbb{R}$$

$\varepsilon_1, \dots, \varepsilon_N$ i.i.d., $\mathcal{E} \varepsilon_j = 0$, $\sigma_\varepsilon^2 = \text{var} \varepsilon_j < \infty$ (abbreviated as: i.i.d. $(0, \sigma_\varepsilon^2)$)

Problem: Estimate m from sample $(X_1, Y_1), \dots, (X_N, Y_N)$.

If X_1, \dots, X_N random (stochastic design), then $m(x) = \mathcal{E}\{Y_j | X_j = x\}$, and, additionally, X_1, \dots, X_N i.i.d. and independent of $\varepsilon_1, \dots, \varepsilon_N$ is assumed.

If $m(x) = m_0$ const, $\frac{1}{N} \sum_{j=1}^N Y_j \xrightarrow{p} m_0$ by L.L.N.

If m smooth, then m locally approximately constant \rightarrow estimate $m(x)$ by local average: $\frac{1}{N_x} \sum_{j=1}^N \mathbf{1}_{[x-h, x+h]}(X_j) \cdot Y_j$ where $N_x =$ no. of nonvanishing summands.

More general: downweighting of data points where X_j is farther away from x . General form of weighted local averages or smoothers:

$$\frac{1}{N} \sum_{j=1}^N W_{Nj}(x) Y_j \quad \text{as estimate of } m(x)$$

where weights $W_{Nj}(x)$ large if $|x - X_j|$ small, and they may depend on all X_1, \dots, X_N simultaneously.

1 A Review of Common Smoothing Techniques

1.1 Kernel Estimates

Definition: A kernel K is a bounded, continuous function on \mathbb{R} satisfying $\int K(u) du = 1$.

In estimating functions (not their derivatives), a kernel usually has to satisfy additionally at least **(K)**: $K(u) \geq 0$ for all u , $\int uK(u) du = 0$, $\int u^2 K(u) du < \infty$.

Notation: $K_h(u) = \frac{1}{h} K(\frac{u}{h})$ rescaled kernel satisfying still $\int K_h(u) du = 1$

If $\text{supp}(K) = [-1, +1]$, then $\text{supp}(K_h) = [-h, h]$. h is called bandwidth or smoothing parameter.

Model 1 (deterministic equidistant design):

$$Y_j = m(x_j) + \varepsilon_j, \quad j = 1, \dots, N$$

$\varepsilon_1, \dots, \varepsilon_N$ i.i.d. $(0, \sigma_\varepsilon^2)$. $x_j = \frac{j}{N}$, $j = 1, \dots, N$.

Definition: The Priestley-Chao (PC) kernel estimate of $m : [0, 1] \rightarrow \mathbb{R}$ with bandwidth $h > 0$ is given as

$$\hat{m}(x, h) = \frac{1}{N} \sum_{j=1}^N K_h(x - x_j) Y_j = \frac{1}{Nh} \sum_{j=1}^N K\left(\frac{x - x_j}{h}\right) Y_j, \quad x \in [0, 1]$$

for some kernel K .

Remark: Corresponding weights $W_{Nj}(x) = K_h(x - x_j)$

Theorem 1.1 *Let kernel K satisfy (\mathbf{K}) . Let K be Lipschitz continuous and have a compact support (may be relaxed). Let m be twice continuously differentiable. Then,*

$$\text{mse}(\hat{m}(x, h)) = \mathcal{E}(\hat{m}(x, h) - m(x))^2 = \sigma_\varepsilon^2 \mathcal{O}\left(\frac{1}{Nh}\right) + (m''(x))^2 \mathcal{O}(h^4)$$

If $h \rightarrow 0$, $Nh \rightarrow \infty$ for $N \rightarrow \infty$, then $\hat{m}(x, h) \xrightarrow{p} m(x)$ (consistency).

Model 2 (stochastic design):

$$Y_j = m(X_j) + \varepsilon_j, \quad j = 1, \dots, N$$

X_1, \dots, X_N i.i.d. with density $p(x)$, $\varepsilon_1, \dots, \varepsilon_N$ i.i.d. $(0, \sigma_\varepsilon^2)$ independent of X_1, \dots, X_N .

Definition: a) The Rosenblatt-Parzen kernel density estimate of $p(x)$ with bandwidth $h > 0$ is given as

$$\hat{p}(x, h) = \frac{1}{N} \sum_{j=1}^N K_h(x - X_j)$$

b) The Nadaraya-Watson (NW) kernel estimate of $m : \mathbb{R} \rightarrow \mathbb{R}$ with bandwidth $h > 0$ is given as

$$\hat{m}(x, h) = \frac{1}{N} \sum_{j=1}^N K_h(x - X_j) Y_j / \hat{p}(x, h)$$

Remark: Corresponding weights $W_{Nj}(x) = K_h(x - X_j) / \hat{p}(x, h)$. Analogous adjustment also necessary for deterministic design if x_1, \dots, x_N not asymptotically uniformly distributed.

Typical kernels: Gauss-kernel $K(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$

Epanechnikov (or Bartlett-Priestley) kernel

$$K(u) = \frac{3}{4} (1 - u^2)^+$$

Remark: Kernel estimates are essentially convolutions which are known to smooth, e.g.

$$\begin{aligned} \mathcal{E} \hat{p}(x, h) &= \frac{1}{N} \sum \mathcal{E} K_h(x - X_j) = \mathcal{E} K_h(x - X_1) \\ &= \int K_h(x - u) p(u) du = (K_h * p)(x) \end{aligned}$$

Theorem 1.2 Let K be a kernel satisfying $K(u) \geq 0$ and $\lim_{|u| \rightarrow \infty} u K(u) = 0$, and let $\mathcal{E}Y_j^2 < \infty$. Then, for any x at which $p(x)$ and $m(x)$ are continuous and $p(x) > 0$, we have $\hat{p}(x, h) \xrightarrow{p} p(x)$ and $\hat{m}(x, h) \xrightarrow{p} m(x)$ if $h \rightarrow 0$, $Nh \rightarrow \infty$ for $N \rightarrow \infty$.

Corollary 1.3 $\mathcal{E} \hat{p}(x, h) = (K_h * p)(x)$, $\text{var} \hat{p}(x, h) = \frac{p(x)}{Nh} \int K^2(u) du + O(\frac{1}{Nh})$

Remark: Under model 2, we have
 $m(x) = \mathcal{E}\{Y_j | X_j = x\}$
 $m^2(x) + \sigma_\varepsilon^2 = \mathcal{E}\{Y_j^2 | X_j = x\}$
 $\text{var}\{Y_j | X_j = x\} = \sigma_\varepsilon^2$

Some remarks on conditional expectation:

Let X, Y be real random variables with $\mathcal{E}|X|, \mathcal{E}|Y| < \infty$ (corresponding remarks are valid for random vectors, too).

$\eta(x) = \mathcal{E}\{Y | X = x\}$, the conditional expectation of Y given that $X = x$ is known, is a measurable function of $x \in \mathbb{R}$. $\mathcal{E}(Y | X) = \eta(X)$, i.e. the function η applied to X , the conditional expectation of Y given X , is a random variable. $\eta(x)$ is characterized as a function such that

$$\mathcal{E} \eta(X) \cdot 1_B(X) = \mathcal{E} Y 1_B(X) \text{ for all Borel sets } B$$

where 1_B denotes the indicator function of B .

The conditional expectation satisfies the following relations

- CE1: If X, Y independent, then $\mathcal{E}\{Y | X = x\} \equiv \mathcal{E}Y$
- CE2: If $Y = g(X)$, then $\mathcal{E}\{g(X) | X = x\} = g(x)$
or, more general, $\mathcal{E}\{Zg(X) | X = x\} = g(x)\mathcal{E}\{Z | X = x\}$
- CE3: $\mathcal{E}\{\alpha_1 Y_1 + \alpha_2 Y_2 | X = x\} = \alpha_1 \mathcal{E}\{Y_1 | X = x\} + \alpha_2 \mathcal{E}\{Y_2 | X = x\}$
- CE4: $\mathcal{E}(\mathcal{E}\{Y | X\}) = \mathcal{E}Y$
- CE5: $\mathcal{E}(Y - \mathcal{E}\{Y | X\})^2 \leq \mathcal{E}(Y - \psi(X))^2$
for all measurable functions ψ , i.e. $\mathcal{E}\{Y | X = x\}$ is the best predictor for Y given that $X = x$ is known.

Let X, Y have a joint probability density $p(x, y)$ and marginal densities $p_X(x), p_Y(y)$. The conditional density of Y given $X = x$ is

$$p_{Y|X}(y|x) = \frac{p(x, y)}{p_X(x)} \text{ for } p_X(x) > 0.$$

Then,

$$\mathcal{E}\{Y | X = x\} = \int y p_{Y|X}(y|x) dy = \int y \frac{p(x, y)}{p_X(x)} dy$$

Def.: Let, for $x_0 \in \mathbb{R}$ and binwidth $b > 0$, $I_k = (x_0 + (k-1)b, x_0 + kb]$, $-\infty < k < \infty$, be a partition of \mathbb{R} into disjoint intervals of length b . Let X_1, \dots, X_N be i.i.d. with density $p(x)$, and let

$$N_k = \#\{j; X_j \in I_k\} = \sum_{j=1}^N 1_{I_k}(X_j).$$

Then,

$$\begin{aligned} H_N(x) &= \frac{1}{b} \frac{N_k}{N} \quad \text{for } x \in I_k \\ &= \frac{1}{b} \sum_{k=-\infty}^{\infty} \frac{N_k}{N} 1_{I_k}(x) \end{aligned}$$

is the (scaled) histogram of the sample X_1, \dots, X_N .

In contrast to more familiar definitions of the histogram we have introduced the factor $\frac{1}{b}$ such that $H_N(x)$ is a probability density, i.e. $\int H_N(x) dx = 1$. The law of large numbers implies for large N , small b :

$$\frac{N_k}{N} \approx \text{pr}(X_1 \in I_k) \approx p(x) \cdot b \quad \text{if } x \in I_k.$$

This idea can be transformed into a proof that $H_N(x)$ is also a consistent probability density estimate as $\hat{p}(x, h)$. Both can be looked at as examples of general estimates of the form

$$\hat{p}(x) = \frac{1}{N} \sum_{j=1}^N W_{Nj}(x)$$

where the weights $W_{Nj}(x)$ are large for X_j close to x . For the kernel estimate, we have $W_{Nj}(x) = K_h(x - X_j)$, and for the histogram

$$W_{Nj}(x) = \frac{1}{b} \sum_{k=-\infty}^{\infty} 1_{I_k}(x) \cdot 1_{I_k}(X_j).$$

1.2 Nearest-Neighbour Estimates

Def.: Let X_1, \dots, X_N be i.i.d. random variables with density $p(x)$. For fixed $x \in \mathbb{R}$, let $d_1(x) \leq d_2(x) \leq \dots \leq d_N(x)$ be the ordered distances $|x - X_j|$, $j = 1, \dots, N$. If $|x - X_k| = d_k(x)$, X_k is called the k -nearest neighbour of x . Let K be a kernel function.

$$\hat{p}_k(x) = \frac{1}{N d_k(x)} \sum_{j=1}^N K\left(\frac{x - X_j}{d_k(x)}\right)$$

is called a k -nearest neighbour (k -NN) density estimate.

$\hat{p}_k(x)$ is a kernel density estimate with bandwidth $d_k(x)$ adapting itself to the number of data lying close to x . It is a general smoothing estimate with weights

$$W_{Nj}(x) = \frac{1}{d_k(x)} K\left(\frac{x - X_j}{d_k(x)}\right).$$

For the rectangular kernel $K(u) = \frac{1}{2} 1_{(-1, +1)}(u)$, where - for this particular case - we relinquish the continuity of K , we get the special case:

$$\hat{p}_k(x) = \frac{k - 1}{2N d_k(x)}.$$

Intuition: $2d_k(x)$ is the length of and $k - 1$ the number of observations in the interval $(x - d_k(x), x + d_k(x))$. As for the histogram:

$$\frac{k - 1}{N} \approx \text{pr}(X_1 \in (x - d_k(x), x + d_k(x))) \approx p(x) \cdot 2d_k(x).$$

If $k \uparrow$, then $d_k(x) \uparrow$ increases for all x , and $\hat{p}_k(x)$ becomes smoother. Therefore, k controls the global smoothness of $\hat{p}_k(x)$. The local smoothness is governed to be $d_k(x)$ and adapts itself to the density of the design points X_1, \dots, X_N . For this advantage a price has to be paid: due to the nondifferentiability of $d_k(x)$ at points of the form $(X_i + X_j)/2$ - midpoints between observations - in the center of the sample, $\hat{p}_k(x)$ is not differentiable, too, even if $p(x)$ is a very smooth function.

Def.: Consider model 2 above, let K be a kernel. Then

$$\hat{m}_k(x) = \frac{1}{N d_k(x)} \sum_{j=1}^N K\left(\frac{x - X_j}{d_k(x)}\right) Y_j / \hat{p}_k(x)$$

is called a k -nearest neighbour ($k - NN$)-estimate for the regression function $m(x)$.

$\hat{m}_k(x)$ is a NW-kernel estimate with adaptive bandwidth $d_k(x)$. For the rectangular kernel $K(u) = \frac{1}{2}1_{(-1,+1)}(u)$, we get a particular simple form, reminiscent of the PC-kernel estimate, as, here, $\hat{p}_k(x) = (k - 1)/(2N d_k(x))$:

$$\hat{m}_k(x) = \frac{1}{k - 1} \sum_{j=1}^N 1_{(-1,+1)}\left(\frac{x - X_j}{d_k(x)}\right) Y_j.$$

In this case, $\hat{m}_k(x)$ is just the average of the Y_j corresponding to the $k - 1$ nearest neighbours X_j of x .

Theorem 1.4 *Under the assumptions of Theorem 1.2, we have for all x at which $p(x)$ and $m(x)$ are continuous and $p(x) > 0$:*

$$\hat{p}_k(x) \xrightarrow{p} p(x) \text{ and } \hat{m}_k(x) \xrightarrow{p} m(x) \text{ for } N \rightarrow \infty, k \rightarrow \infty \text{ such that } \frac{k}{N} \rightarrow 0.$$

1.3 Spline Smoothers

Def.: Let $x_1 < x_2 < \dots < x_N$ be ordered design points. A cubic spline function corresponding to this design is a twice continuously differentiable (C^2)-function which in all intervals (x_j, x_{j+1}) , $1 \leq j < N$, coincides with an appropriate polynomial of degree 3.

For a function $g \in C^2$, $\int (g''(x))^2 dx$ is a measure of its total curvature. Given points $(x_1, y_1), \dots, (x_N, y_N)$, $x_1 < \dots < x_N$, then among all interpolating functions $g \in C^2$, the total curvature is minimized by a cubic spline function corresponding to the design x_1, \dots, x_N . It can be made unique by additional boundary conditions on the derivatives at x_1 and x_N .

In the regression model 2, due to the errors ε_j we do not want to interpolate. Recalling polynomial regression compared to polynomial interpolation, a least-squares (LS-)approach comes into mind:

$$\sum_{j=1}^N (Y_j - g(X_j))^2 = \min_g!$$

In a nonparametric context, where g is not specified by a finite number of parameters but only by regularity assumptions like $g \in C^2$, an unrestricted LS-estimate interpolates automatically. To get a reasonable procedure, one has to add a roughness penalty to the LS-distance, e.g.

$$S_\lambda(g) = \sum_{j=1}^N (Y_j - g(X_j))^2 + \lambda \int (g''(x))^2 dx = \min_{g \in C^2}!$$

Here, as a particular roughness penalty, the total curvature is chosen. λ is a tuning parameter of the procedure which serves to balance the two antagonistic goals "close fit to observations $(X_1, Y_1), \dots, (X_N, Y_N)$ " and "smoothness of regression function estimate". A function estimate minimizing $S_\lambda(g)$ or a similar expression with another type of roughness penalty is called a penalized least-squares (PLS-)estimate.

Proposition 1.5 $S_\lambda(g)$ is a convex Gateaux-differentiable functional on C^2 . Its Gateaux-derivative at f in direction g is

$$\begin{aligned} S'_\lambda(f; g) &\equiv \lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} \{S_\lambda(f + \varepsilon g) - S_\lambda(f)\} \\ &= -2 \sum_{j=1}^N (Y_j - f(X_j)) g(X_j) + 2\lambda \int f''(x) g''(x) dx \end{aligned}$$

$S'_\lambda(f; \cdot)$ is a continuous linear functional on C^2 .

Lemma 1.6 (from convex analysis): Let T be a convex Gateaux-differentiable functional on C^2 . Then, T assumes its minimum at $f \in C^2$ iff $T'(f; g) = 0$ for all $g \in C^2$.

Proposition 1.7 A solution $\hat{m}_\lambda(x)$ of $S_\lambda(g) = \min_{g \in C^2}!$ is a (cubic) spline smoother, i.e. it has the following properties:

a) $\hat{m}_\lambda(x)$ is a cubic spline function with respect to the design $X_{(1)} < \dots < X_{(N)}$

b) in the boundary points $X_{(1)}, X_{(N)} : \hat{m}'_\lambda(X_{(1)}) = 0 = \hat{m}'_\lambda(X_{(N)})$

c) $\hat{m}''_\lambda(x) = 0$ for all $x \notin [X_{(1)}, X_{(N)}]$

Alternatively, spline smoothers are defined as solutions of the constrained optimization problem:

$$\int (g''(x))^2 dx = \min_{g \in C^2} \text{ under the constraint } \sum_{j=1}^N (Y_j - g(X_j))^2 \leq \Delta$$

Let $\tilde{m}_\Delta(x)$ denote a solution. Then, using a Lagrange multiplier approach:

Proposition 1.8 *Let $G(\Delta) = \int (\tilde{m}''_\Delta(x))^2 dx$, $\lambda = -1/G'(\Delta)$. Then,*

$$\hat{m}_\lambda(x) = \tilde{m}_\Delta(x).$$

Both types of spline smoothers coincide. Only the exact definition of the tuning parameter λ and Δ resp. differs.

Proposition 1.9 *Let $\hat{m}_{\lambda,k}(x)$ be the spline smoother for the data set $(X_j, Y_{j,k})$, $j = 1, \dots, N$, $k = 1, \dots, K$, and $\alpha_1, \dots, \alpha_K \in \mathbb{R}$. Then, $\sum_{k=1}^K \alpha_k \hat{m}_{\alpha,k}(x)$ is the spline smoother for $(X_j, \sum_{k=1}^K \alpha_k Y_{j,k})$, $j = 1, \dots, N$.*

Corollary 1.10 *The spline smoother for (X_j, Y_j) , $j = 1, \dots, N$, has the form*

$$\hat{m}_\lambda(x) = \frac{1}{N} \sum_{j=1}^N W_{Nj}^{(\lambda)}(x) Y_j$$

where the weight $\frac{1}{N} W_{Nk}^{(\lambda)}(x)$ is the spline smoother for the data set (X_j, δ_{jk}) , $j = 1, \dots, N$, (as $Y_j = \sum_{k=1}^N Y_k \delta_{jk}$).

Theorem 1.11 (Silverman): *Let $[a, b] \supseteq \{X_1, \dots, X_N\}$, x "not too close" to the boundary, $\lambda \rightarrow 0$ for $N \rightarrow \infty$ "not too fast". Then, with $W_{Nk}^{(\lambda)}$ of Corollary 1.10, we have for $N \rightarrow \infty$:*

$$W_{Nk}^{(\lambda)}(x) \sim \frac{1}{p(X_k)} \frac{1}{h(X_k)} K^{sp} \left(\frac{x - X_k}{h(X_k)} \right)$$

with (spline) kernel $K^{sp}(u) = \frac{1}{2} e^{-|u|/\sqrt{2}} \sin \left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4} \right)$

and local bandwidth $h(x) = \sqrt[4]{\frac{\lambda}{Np(x)}}$.

$K^{sp}(u)$ is a symmetric kernel with $\int u^2 K^{sp}(u) du = 0$.

Asymptotically, spline smoothers are NW-estimates with bandwidth $h(x)$ depending on the global smoothing parameter λ and the local density of observations around x (via $p(x)$) - similar as for k -NN-estimates.

1.4 Local Polynomial Estimators

The NW-kernel estimate of the regression function $m(x)$ in Model 2 can be interpreted as the solution of the following weighted least-squares (WLS) problem:

$$\sum_{j=1}^N (Y_j - b_0)^2 K_h(x - X_j) = \min!$$

The solution \hat{b}_0 coincides with $\hat{m}(x, h)$ (set the derivative w.r.t. b_0 to 0). Therefore, $m(x)$ is approximated locally by a constant b_0 and that constant is estimated by WLS to guarantee that only observations close to x have a significant influence on the estimate \hat{b}_0 . In local polynomial regression, $m(x)$ is approximated locally by a polynomial of (low) degree p , and, then, the coefficients are estimated by WLS:

$$\sum_{j=1}^N (Y_j - b_0 - \dots - b_p(x - X_j)^p)^2 K_h(x - X_j) = \min! \quad (\text{LP}_p)$$

provides estimates $\hat{b}_0, \dots, \hat{b}_p$, and the local polynomial estimator of degree p for $m(x)$ is defined as

$$\hat{m}_p(x, h) = \hat{b}_0.$$

\hat{b}_1 estimates $m'(x)$, $2!\hat{b}_2$ estimates $m''(x)$, and so on.

Proposition 1.12 *Let x be fixed. Consider the $N \times (p + 1)$ design matrix \mathbf{X} with entries $X_{jk} = (x - X_j)^k$, $j = 1, \dots, N$, $k = 0, \dots, p$, and the diagonal $N \times N$ -matrix \mathbf{W} of weights with $W_{jj} = K_h(x - X_j)$, $j = 1, \dots, N$. If $\mathbf{X}^T \mathbf{W} \mathbf{X}$ is invertible, then the solution of (LP_p) $\hat{\mathbf{b}} = (\hat{b}_0, \dots, \hat{b}_p)$ is given by*

$$\hat{\mathbf{b}} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y}$$

with $\mathbf{Y} = (Y_1, \dots, Y_N)^T$.

Corollary 1.13 *Let $p = 1$. The local linear estimate for $m(x)$ is given by*

$$\hat{m}_1(x, h) = \frac{\frac{1}{N} \sum_{j=1}^N [\hat{q}_2(x, h) - (x - X_j) \hat{q}_1(x, h)] K_h(x - X_j) Y_j}{\hat{q}_2(x, h) \hat{q}_0(x, h) - \hat{q}_1^2(x, h)}$$

with $\hat{q}_k(x, h) = \frac{1}{N} \sum_{j=1}^N (x - X_j)^k K_h(x - X_j)$.

(Remark: $\hat{q}_0(x, h) = \hat{p}(x, h)$, the Rosenblatt-Parzen estimate of the density $p(x)$ of the X_j).

Remarks:

1. For small bandwidth h it may happen with positive probability that $X^T W X$ is not invertible, in particular for kernels K with compact support.
2. For nonequidistant (in particular, for random) design, local polynomial estimates have a bias of smaller order than the NW estimate. However, in practice, they seem to have a larger variability and roughness.
3. The main advantage is a considerably smaller bias at the boundary without using particular boundary kernels. However, that advantage is also paid for by an increase in variance.

2 On the choice of smoothing parameters

2.1 Measures of estimation quality and rates of convergence

We consider Model 2 of Section 1.1:

$$Y_j = m(X_j) + \varepsilon_j$$

and, e.g., a NW-kernel estimate $\hat{m}(x, h)$ of $m(x)$.

Definition (local and global error measures):

(a) mean-squared error $mse(\hat{m}(x, h)) = \mathcal{E}(\hat{m}(x, h) - m(x))^2$ for fixed x

(b) mean-integrated-squared error on $[a, b]$

$$mise(\hat{m}(\cdot, h)) = \mathcal{E} \int_a^b (\hat{m}(x, h) - m(x))^2 w(x) dx$$

with weight function $w(x)$.

In parametric, e.g. linear, regression, the typical rate of convergence of \hat{m}_{par} to m is \sqrt{N} , i.e.

$$mse(\hat{m}_{par}) = O\left(\frac{1}{N}\right).$$

In nonparametric regression, we typically have

$$mse(\hat{m}_{npar}) = O\left(\frac{1}{N^r}\right),$$

where $r < 1$ depends on smoothness of m , dimension of X_j , type of estimate \hat{m}_{npar} and on the order of derivative of m which is to be estimated.

Definition: A sequence $b_N \geq 0$, $b_N \rightarrow 0$, is called an optimal (global) rate of convergence if for constants c, C, N_0 .

1. $mise(\hat{m}(\cdot, h)) \leq C b_N$ for some kernel estimate \hat{m} , all $N \geq N_0$
2. $mise(\hat{m}(\cdot, h)) \geq c b_N$ for all kernel estimates \hat{m} , all $N \geq N_0$

A kernel estimate satisfying (a) for an optimal rate of convergence b_N is called asymptotically optimal.

Theorem 2.1 *Let the density $p(x)$ of X_1, X_2, \dots be twice continuously differentiable and $p''(x)$ be Hölder-continuous for some $\beta > 0$, i.e. for some L_β*

$$|p''(u) - p''(v)| \leq L_\beta |u - v|^\beta \quad \text{for all } u, v,$$

let the kernel $K \geq 0$ be bounded, continuous and

$$\int K(u) du = 1, \quad \int u K(u) du = 0, \quad \int |u|^{2+\beta} K(u) du < \infty.$$

Then, with $V_K = \int u^2 K(u) du$ and $M_{2,K} = \int K^2(u) du$:

$mse(\hat{p}(x, h)) \sim \frac{1}{Nh} p(x) M_{2,K} + (\frac{h^2}{2} p''(x) V_K)^2$ uniformly in x

$mise(\hat{p}(\cdot, h)) = \int mse(\hat{p}(x, h)) dx \sim \frac{1}{Nh} M_{2,K} + \frac{h^4}{4} \int (p''(x))^2 dx V_K^2$

Corollary 2.2 Under the conditions of Theorem 2.1, the optimal rate of convergence is $b_N = N^{-4/5}$. It is achieved for $h = h_N \sim \text{const } N^{-1/5}$. More precisely:

i) locally $h_x^5 \sim \frac{M_{2,K}}{V_K^2} \frac{p(x)}{(p''(x))^2} \cdot \frac{1}{N}$

ii) globally $h^5 \sim \frac{M_{2,K}}{V_K^2} \frac{1}{\int (p''(x))^2 dx} \cdot \frac{1}{N}$

Similar for regression problem in dimension $d \geq 1$:

$$Y_j = m(X_j) + \varepsilon_j, \quad X_1, \dots, X_N \text{ i.i.d. } \in \mathbb{R}^d$$

where now $K_h(u) = \frac{1}{h^d} K(\frac{u}{h})$ and, otherwise, the NW-estimates remain unchanged.

Theorem 2.3 Let m be p -times continuously differentiable, $m^{(p)}$ be Hölder-continuous for some $\beta > 0$, and let K satisfy regularity assumptions similar to Theorem 2.1. A kernel estimate for $m^{(k)}$ has optimal rate of convergence N^{-r} with

$$r = \frac{2(p - k + \beta)}{2(p + \beta) + d}$$

Consequences: $d \uparrow \implies r \downarrow$ (curse of dimensionality)
 $k \uparrow \implies r \downarrow$ (derivatives are harder to estimate)
 $p + \beta \uparrow \implies r \uparrow$ (smooth functions are easier to estimate)

Example: $d = 1$, $X_j = \frac{j}{N}$, $m : [0, 1] \rightarrow \mathbb{R}$, $m \in C^4$, $\beta > 0$, but small.

PC-estimate: $\hat{m}(x, h) = \frac{1}{Nh} \sum_1^N K(\frac{x-X_j}{h}) Y_j$

Estimate for m'' : $\hat{m}''(x, h) = \frac{\partial^2}{\partial x^2} \hat{m}(x, h) = \frac{1}{Nh^3} \sum_1^N K''(\frac{x-X_j}{h}) Y_j$

If $\beta \approx 0$, then optimal rate of convergence (for $h \approx \text{const} \cdot N^{-1/9}$) for \hat{m}'' is $\approx N^{-4/9}$.

2.2 Bandwidth selection by cross-validation

As an alternative to mise we consider

$$\underline{\text{average squared error}} \text{ ase}(\hat{m}(\bullet, h)) = \frac{1}{N} \sum_{j=1}^N (\hat{m}(X_j, h) - m(X_j))^2 w(X_j)$$

which is easier to calculate in simulations.

Proposition 2.4 If $p(x) > 0$ on $\{x; w(x) > 0\}$, then

$$\frac{\text{ase}(\hat{m}(\bullet, h))}{\text{mise}(\hat{m}(\bullet, h))} \xrightarrow{\text{a.s.}} 1$$

uniformly for $h \in [\frac{1}{N^{1-\delta}}, \frac{1}{N^\delta}]$ for arbitrary $\delta > 0$.

Abbreviation: $d_A(h) = \text{ase}(\hat{m}(\bullet, h))$

$$d_A(h) = \frac{1}{N} \sum_1^N \hat{m}^2(X_j) w(X_j) + \frac{1}{N} \sum_1^N m^2(X_j, h) w(X_j) - 2C(h)$$

with $C(h) = \frac{1}{N} \sum_1^N m(X_j) \hat{m}(X_j, h) w(X_j)$.

The first term of $d_A(h)$ does not depend on h , the second one can be calculated from the data. Therefore, only $C(h)$ has to be estimated somehow, and

$$\begin{aligned}\hat{h} &= \operatorname{argmin}_h \left(\frac{1}{N} \sum_1^N \hat{m}^2(X_j, h) w(X_j) - 2\hat{C}(h) \right) \\ &\approx \operatorname{argmin}_h d_A(h) \approx \operatorname{argmin}_h \operatorname{mise}(\hat{m}(\cdot, h))\end{aligned}$$

provides a bandwidth selection procedure asymptotically minimizing *mise*.

As $Y_j = m(X_j) + \varepsilon_j$, $\mathcal{E}\varepsilon_j = 0$, an intuitive way to estimate $C(h)$ is to replace the unknown $m(X_j)$ by Y_j :

$$\hat{C}_1(h) = \frac{1}{N} \sum_{j=1}^N Y_j \hat{m}(X_j, h) w(X_j).$$

The corresponding selection of bandwidth is

$$\begin{aligned}\hat{h}_1 &= \operatorname{argmin}_h \left(\frac{1}{N} \sum_{j=1}^N \hat{m}^2(X_j, h) w(X_j) - 2\hat{C}_1(h) \right) \\ &= \operatorname{argmin}_h \pi(h)\end{aligned}$$

with the average squared prediction error (from predicting Y_j by $\hat{m}(X_j, h)$)

$$\pi(h) = \frac{1}{N} \sum_{j=1}^N (Y_j - \hat{m}(X_j, h))^2 w(X_j),$$

as the two functions to be minimized differ only by $\frac{1}{N} \sum Y_j^2 w(X_j)$ which does not depend on h . However, \hat{h}_1 is a strongly biased estimate of $\operatorname{argmin} d_A(h)$. Intuitively, Y_j influences $\hat{m}(X_j, h)$ and, therefore, $|Y_j - \hat{m}(X_j, h)|$ systematically tends to be smaller than $|Y_j - m(X_j)|$. Therefore, consider as an alternative a crossvalidation estimate of $\pi(h)$

$$CV(h) = \frac{1}{N} \sum_{j=1}^N (Y_j - \hat{m}_{-j}(X_j, h))^2 w(X_j)$$

with $\hat{m}_{-j}(x, h) = \sum_{i \neq j} K_h(x - X_i) Y_i / \sum_{i \neq j} K_h(x - X_i)$ being the NW-estimate based on the sample with (X_j, Y_j) removed. The crossvalidatory selection of bandwidth is

$$\hat{h}_{CV} = \operatorname{argmin}_h CV(h).$$

Plugging in $Y_j = m(X_j) + \varepsilon_j$ into the definition of π and CV implies

$$\begin{aligned}\pi(h) &= d_A(h) + \frac{1}{N} \sum \varepsilon_j^2 w(X_j) - 2e_1(h) \\ CV(h) &= d_A(h) + \frac{1}{N} \sum \varepsilon_j^2 w(X_j) - 2e_2(h)\end{aligned}$$

The second term does not depend on h and is, therefore, negligible in minimizing. For the last term, we have

$$\mathcal{E}\{e_1(h) | X_1, \dots, X_N\} \sim \frac{\operatorname{const}}{Nh},$$

i.e. the error term is of the same size as $d_A(h)$ itself, and

$$\mathcal{E}\{e_2(h)|X_1, \dots, X_N\} = 0,$$

i.e. the error term is in the mean negligible for the crossvalidatory approach.

Theorem 2.5 *Let $H_N = [\frac{1}{N^{1-\delta}}, \frac{1}{N^\delta}]$ for some $\delta > 0$. Under smoothness and other regularity assumptions on K, m, p and moment conditions on ε_j , \hat{h}_{CV} is asymptotically optimal, i.e.*

$$\frac{d_A(\hat{h}_{CV})}{\inf_{h \in H_N} d_A(h)} \xrightarrow{a.s.} 1 \quad \text{for } N \rightarrow \infty.$$

2.3 Asymptotic distribution of kernel estimates

Let X_1, \dots, X_N i.i.d. with density $p(x)$. The kernel density estimate

$$\hat{p}(x, h) = \frac{1}{N} \sum_{j=1}^N K_h(x - X_j) \equiv \frac{1}{N} \sum_{j=1}^N Z_{j,N}$$

is the sample mean of i.i.d. random variables $Z_{1,N}, \dots, Z_{N,N}$. Central limit theorems for such triangular arrays of random variables (like Lyapunov's or Lindeberg's central limit theorem) imply

$$\sqrt{Nh}(\hat{p}(x, h) - \mathcal{E}\hat{p}(x, h)) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma_x^2)$$

with $\sigma_x^2 = p(x) \int K^2(u)du$ (see Corollary 1.3). If the bias is of smaller order than the standard deviation, $\sqrt{Nh}(\hat{p}(x, h) - p(x))$ has the same limit distribution. If bias² and variance are balanced, e.g., for $p \in C^2$, if $h \sim N^{-1/5}$:

$$\sqrt{Nh}(\hat{p}(x, h) - p(x)) \xrightarrow{\mathcal{L}} \mathcal{N}\left(\frac{1}{2}p''(x) \int u^2 K(u)du, \sigma_x^2\right).$$

Similar results hold for the NW-kernel estimate under model 2:

Theorem 2.6 *Let K be a bounded, continuous kernel with $\int uK(u)du = 0$. If*

i) $m, p \in C^2$, ii) $h \sim N^{-1/5}$ and iii) $\mathcal{E}\{|Y_j|^{2+\delta}|X_j = x\} < \infty$ for all x for some $\delta > 0$, then

a) For all x with $p(x) > 0$

$$\sqrt{Nh}(\hat{m}(x, h) - m(x)) \xrightarrow{\mathcal{L}} \mathcal{N}(b_x, \sigma_x^2)$$

with the asymptotic variance

$$\sigma_x^2 = \frac{\sigma_\varepsilon^2}{p(x)} \int K^2(u)du$$

and the asymptotic bias

$$b_x = \left(m''(x) + 2m'(x) \frac{p'(x)}{p(x)} \right) \int u^2 k(u)du$$

b) For $x_1 < x_2 < \dots < x_k$ with $p(x_j) > 0$, $j = 1, \dots, k$, $\hat{m}(x_1, h), \dots, \hat{m}(x_k, h)$ are asymptotically independent.

2.4 Boundary kernels

In this section, K is assumed to have compact support, say $[-1, +1]$. Then, the support of K_h is $[-h, h]$, and the support of $K_h(x - \bullet)$ is $[x - h, x + h]$. If, e.g., all $X_j \geq 0$ and $x < h$, then the support of the smoothing kernel extends into a region $\{z; z < 0\}$ where no data are available. This creates an additional bias. For fixed $x > 0$, this bias vanishes for $N \rightarrow \infty$ as, then, $h \rightarrow 0$. For an asymptotic theory which correctly describes the boundary effects in finite samples, we have to consider estimation of $p(x)$ or $m(x)$ at points x converging to the boundary for $N \rightarrow \infty$.

For sake of illustration, consider Model 1 (deterministic equidistant design) and the Priestley-Chao estimate at the left boundary 0:

$$\hat{m}(x, h) = \frac{1}{N} \sum_{j=1}^N K_h(x - x_j) Y_j, \quad x_j = \frac{j}{N}$$

for $x = \rho h$, ρ fixed. As in the proof of Theorem 1.1:

$$\begin{aligned} \mathcal{E} \hat{m}(x, h) &= \frac{1}{N} \sum_1^N K_h(x - x_j) m(x_j) \approx \int_0^1 K_h(x - u) m(u) du \\ &= \int_{(x-h)^+}^{x+h} K_h(x - u) m(u) du = \int_{-1}^{\rho \wedge 1} K(v) m(x - vh) dv \end{aligned}$$

with $\rho \wedge 1 \equiv \min(\rho, 1)$, as the support of $K_h(x - \bullet)$ is $[x - h, x + h]$. Taylor expansion of $m(x - vh)$ up to order 2 (assuming $m \in C^2$) gives

$$\mathcal{E} \hat{m}(x, h) = \mu_0(\rho) m(x) - h \mu_1(\rho) m'(x) + \frac{1}{2} h^2 \mu_2(\rho) m''(x) + o(h^2)$$

with the abbreviations

$$\mu_i(\rho) = \int_{-1}^{\rho \wedge 1} v^i K(v) dv, \quad i = 0, 1, 2.$$

If $\rho \geq 1$, i.e. $x = \rho h \geq h$ is no boundary point, then

$$\mu_0(\rho) = 1, \quad \mu_1(\rho) = 0 \quad \text{and} \quad \mu_2(\rho) = \int v^2 K(v) dv,$$

and we get the usual asymptotic bias:

$$\mathcal{E} \hat{m}(x, h) - m(x) = \frac{1}{2} h^2 m''(x) \int v^2 k(v) dv + o(h^2).$$

For boundary points ($0 < \rho < 1$), the bias is of larger order

$$\mathcal{E} \hat{m}(x, h) - m(x) = (\mu_0(\rho) - 1) m(x) - h \mu_1(\rho) m'(x) + \frac{1}{2} h^2 m''(x) \mu_2(\rho) + o(h^2).$$

Based on the general jackknife techniques for bias reduction, one can achieve a bias of order h^2 by an estimate (for $x < h$ only)

$$\hat{m}_J(x, h) = \lambda \hat{m}(x, h) + \lambda_\alpha \hat{m}(x, \alpha h).$$

Using the abbreviations $m_i = \mu_i(\rho)$, $a_i = \mu_i(\rho/\alpha)$, for $\lambda = (m_0 - m_1 \frac{a_0}{\alpha a_1})^{-1} > 0$ and $\lambda_\alpha = -\lambda \frac{m_1}{\alpha a_1} < 0$, the two dominant parts of the bias vanish, and $\mathcal{E} \hat{m}_J(x, h) - m(x) = O(h^2)$. $\alpha > 1$ is a tuning parameter which is not given in advance. Rice (1984) recommends $\alpha = 2 - \rho$.

$\hat{m}_J(x, h)$ is itself a kernel estimate, but with a kernel depending on ρ and therefore on x :

$$\hat{m}_J(x, h) = \frac{1}{N} \sum_{j=1}^N K_h^{J,\rho}(x - x_j) Y_j$$

with $K_h^{J,\rho}(\cdot) = \frac{1}{h} K^{J,\rho}(\frac{\cdot}{h})$, $K^{J,\rho}(\cdot) = \lambda K(\cdot) + \lambda_\alpha K_\alpha(\cdot)$. Other types of boundary kernels have been considered in the literature, but the basic idea is always the same: for x close to the boundary, use special kernels (usually asymmetric) which depend on x .

3 Orthogonal Series Expansions and Wavelets

3.0 Fourier series - some basic facts

$$L_T^2 = \{f : \mathbb{R} \rightarrow \mathbb{C}; f(x+T) = f(x) \text{ for all } x, \|f\|_T^2 = \int_t^{t+T} |f(x)|^2 dx < \infty\}$$

Hilbert space of functions which are periodic with period T and square-integrable over one period; $\|f\|_T$ does not depend on t .

L_T^2 is a Hilbert-space with scalar product (independent of t)

$$\langle f, g \rangle_T = \int_t^{t+T} \overline{f(x)} g(x) dx$$

and orthogonal basis

$$e_n(x) = e^{i\omega_n x}, \quad n \in \mathbb{Z},$$

where $\omega_n = (2\pi n)/T$ are called Fourier-frequencies. We have

$$\langle e_n, e_m \rangle_T = T \cdot \delta_{mn}$$

L_T^2 has a countable basis $\mapsto L_T^2$ isomorphic to

$$l_2(\mathbb{Z}) = \{(\dots, c_{-1}, c_0, c_1, c_2, \dots); c_n \in \mathbb{C}, \sum_{n=-\infty}^{\infty} |c_n|^2 < \infty\}$$

i.e. the space of doubly-infinite square-summable sequences. The isomorphism between L_T^2 and $l_2(\mathbb{Z})$ is given by the bijection

$$(c_n) \longleftrightarrow f(x) = \frac{1}{T} \sum_{n=-\infty}^{\infty} c_n e_n(x).$$

It is also an isometry:

$$\|(c_n)\|_{l_2}^2 = \sum_{n=-\infty}^{\infty} |c_n|^2 = \int_t^{t+T} |f(x)|^2 dx = \|f\|_T^2$$

(Parseval's equation)

Furthermore, for $g(x) = \frac{1}{T} \sum_{n=-\infty}^{\infty} b_n e_n(x)$, we have

$$\langle (b_n), (c_n) \rangle_{l_2} = \sum_{n=-\infty}^{\infty} \bar{b}_n c_n = \int_t^{t+T} \overline{g(x)} f(x) dx = \langle g, f \rangle_T.$$

The isomorphism represents the expansion of periodic functions into an orthogonal system:

$$f(x) = \frac{1}{T} \sum_{n=-\infty}^{\infty} c_n e^{i\omega_n x}$$

i.e. the synthesis of the signal $f(x)$ using elementary harmonic waves $\cos \omega_n x + i \sin \omega_n x$. If f is real-valued, this can be expressed as

$$f(x) = \frac{1}{T} \sum_{n=-\infty}^{\infty} a_n \cos(\omega_n x + \varphi_n)$$

i.e. as a superposition of cosines with amplitudes a_n , frequencies $\omega_n = (2\pi n)/T$ and phases φ_n . The inverse mapping given by

$$c_n = \langle e_n, f \rangle_T = \int_t^{t+T} e^{-i\omega_n x} f(x) dx$$

represents the Fourier analysis of the signal into elementary waves with frequencies ω_n . The contribution of the wave with frequency w_n to the energy per period of the signal $f(x)$ is proportional to the square of its amplitude:

$$|c_n|^2 = \text{contribution of } n\text{-th harmonic wave } e^{i\omega_n x} \text{ to energy per period of } f(x)$$

$$\|f\|_T^2 = \sum_{n=-\infty}^{\infty} |c_n|^2 = \text{total energy per period of signal } f(x)$$

3.1 Orthogonal Series Density Estimates

Let X_1, \dots, X_N be i.i.d. with density $p(x)$ vanishing outside the interval (a, b) for some $-\infty \leq a < b \leq \infty$.

Let $\varphi_1(x), \varphi_2(x), \dots$ be an orthonormal basis of $L^2(a, b)$, i.e.

$$\int_a^b \overline{\varphi_j(x)} \varphi_k(x) dx = \langle \varphi_j, \varphi_k \rangle = \delta_{jk}.$$

If (a, b) is a finite interval, then the Fourier basis $\varphi_k(x) = \frac{1}{\sqrt{T}} e_k(x)$ of section 3.0 would be a possible choice. Popular choices are - also for infinite intervals - appropriate systems of orthonormal polynomials like the Hermite-, Legendre- or Laguerre-polynomials.

If $p \in L^2(a, b)$, i.e. $\int_a^b |p(x)|^2 dx < \infty$, then - analogously to 3.0 - we have the expansion

$$p(x) = \sum_{k=1}^{\infty} \beta_k \varphi_k(x)$$

with

$$\beta_k = \langle \varphi_k, p \rangle = \int_a^b \overline{\varphi_k(x)} p(x) dx,$$

where the series converges in $L^2(a, b)$, i.e.

$$\int_a^b |p(x) - \sum_{k=1}^M \beta_k \varphi_k(x)|^2 dx \longrightarrow 0 \text{ for } M \rightarrow \infty.$$

As $\beta_k = \mathcal{E} \overline{\varphi_k(X_1)}$, a consistent estimate of β_k is

$$\hat{\beta}_k = \frac{1}{N} \sum_{j=1}^N \overline{\varphi_k(X_j)}$$

for all k . However, the series $\sum_{k=1}^{\infty} \hat{\beta}_k \varphi_k(x)$, which would be an intuitive estimate for $p(x)$, does not converge to a function in $L^2(a, b)$ as the $\hat{\beta}_k$ are the coefficients corresponding to a discrete measure, not to a measure with a $L^2(a, b)$ -density:

$$\hat{\beta}_k = \int_a^b \overline{\varphi_k(x)} d\xi_N(x), \quad 1 \leq k < \infty,$$

where $\xi_N = \frac{1}{N} \sum_{j=1}^N \delta_{x_j}$ is the empirical measure of the sample X_1, \dots, X_N , i.e. the measure corresponding to the usual empirical distribution function $\hat{F}_N(x)$. As a remedy, truncated series are considered:

$$\hat{p}_M(x) = \sum_{k=1}^M \hat{\beta}_k \varphi_k(x).$$

If the truncation point $M \rightarrow \infty$ with an appropriate rate, \hat{p}_M is consistent. Typical results are of the form:

$$\hat{p}_M(x) \xrightarrow{p} p(x) \text{ for } N \rightarrow \infty, M \rightarrow \infty \text{ such that } M^\gamma/N \rightarrow 0,$$

where γ depends on the particular basis $\{\varphi_k(x)\}$ and on moment assumptions on the X_j .

A slightly more general approach allows for downweighting $\hat{\beta}_k$ for large k :

$$\hat{p}_M(x) = \sum_{k=1}^M w_k \hat{\beta}_k \varphi_k(x)$$

with weights $w_k \in [0, 1]$. Estimates of this type are asymptotically equivalent to kernel estimates where M is the smoothing parameter roughly equivalent to $1/\text{bandwidth}$ in kernel estimation as can be seen from asymptotic expansions for bias and variance of \hat{p}_M .

Example: $(a, b) = (-\pi, \pi)$, $\varphi_k(x) = \frac{1}{\sqrt{2\pi}} e^{ikx}$, $-\infty < k < \infty$

$$\hat{p}_M(x) = \frac{1}{\sqrt{2\pi}} \sum_{k=-M}^M w_k \hat{\beta}_k e^{ikx}$$

with

$$\hat{\beta}_k = \frac{1}{\sqrt{2\pi}N} \sum_{j=1}^N e^{-ikX_j}, \quad -M \leq k \leq M.$$

As \hat{p}_M is the Fourier transform of the product sequence $w_k \cdot \hat{\beta}_k$, it coincides with the convolution of the Fourier transform of $\{w_k\}$:

$$W_M(u) = \frac{1}{\sqrt{2\pi}} \sum_{k=-M}^M w_k e^{ikx}$$

and the Fourier-Stieltjes-transform of the $\hat{\beta}_k$ which is just the empirical measure ξ_N :

$$\begin{aligned} \hat{p}_M(x) &= W_M * \xi_N(x) \equiv \int_{-\pi}^{\pi} W_M(x-u) d\xi_N(u) \\ &= \frac{1}{N} \sum_{j=1}^N W_M(x - X_j) \end{aligned}$$

$\hat{p}_M(x)$ is a kernel-type density estimate with kernel W_M .

Special cases: i) $w_k = 1$, $-M \leq k \leq M \rightsquigarrow$ Dirichlet kernel

$$W_M(u) = \begin{cases} \frac{1}{\sqrt{2\pi}} \sin \left\{ \frac{(2M+1)u}{2} \right\} / \sin \left\{ \frac{u}{2} \right\} & \text{if } u \neq 0 \\ \frac{1}{\sqrt{2\pi}}(2M+1) & \text{if } u = 0 \end{cases}$$

ii) $w_k = 1 - \frac{|k|}{M}$, $-M \leq k \leq M \rightsquigarrow$ Fejer kernel

$$W_M(u) = \begin{cases} \frac{1}{\sqrt{2\pi}} \frac{1}{M+1} (\sin \left\{ \frac{(M+1)u}{2} \right\} / \sin \left\{ \frac{u}{2} \right\})^2 & \text{if } u \neq 0 \\ \frac{M+1}{\sqrt{2\pi}} & \text{if } u = 0 \end{cases}$$

3.2 Orthogonal Series Regression Estimates

We consider the regression model with fixed equidistant design

$$Y_j = m(x_j) + \varepsilon_j, \quad x_j = \frac{j}{N}, \quad j = 1, \dots, N$$

with ε_j i.i.d. $(0, \sigma_\varepsilon^2)$. We want to estimate $m(x)$, $0 \leq x \leq 1$. By periodic continuation with period $T = 1$, we get from section 3.0

$$m(x) = \sum_{k=-\infty}^{\infty} c_k e^{2\pi i k x}$$

with

$$c_k = \int_0^1 e^{-2\pi i k x} m(x) dx \approx \frac{1}{N} \sum_{j=1}^N e^{-2\pi i k x_j} m(x_j),$$

replacing the integral by a Riemann sum. We replace $m(x_j)$ by Y_j and get as estimates of the c_k :

$$\hat{c}_k = \frac{1}{N} \sum_{j=1}^N e^{-2\pi i k x_j} Y_j,$$

i.e. the so-called discrete Fourier transform of the data Y_1, \dots, Y_N . As in section 3.1, we have to truncate the inverse Fourier transform to get a consistent estimate $\hat{m}_K(x)$ of m for $K \rightarrow \infty$ slower than N :

$$\hat{m}_K(x) = \sum_{k=-K}^K \hat{c}_k e^{2\pi i k x}.$$

An alternative to truncating the infinite sum, i.e. to neglecting the \hat{c}_k , $|k| > K$ (which will be small with high probability for large N and appropriate K), is the so-called thresholding, i.e. neglecting the small \hat{c}_k independent of their index. A threshold is chosen, then we consider either hard-thresholding

$$\hat{c}_k^* = \begin{cases} \hat{c}_k & \text{if } |\hat{c}_k| > \sigma \\ 0 & \text{else} \end{cases}$$

or soft-thresholding

$$\hat{c}_k^* = \begin{cases} \hat{c}_k \cdot \frac{(|\hat{c}_k| - \sigma)^+}{|\hat{c}_k|} & |\hat{c}_k| > \sigma \\ 0 & \text{else} \end{cases}$$

and we define

$$\hat{m}_\sigma(x) = \sum_k \hat{c}_k^* e^{2\pi i k x}.$$

If $\sigma \rightarrow 0$ for $N \rightarrow \infty$ with an appropriate rate, this estimate will also be consistent for m .

3.3 The Windowed Fourier transform

Local effects in a signal which are not present periodically cannot be adequately described by the Fourier transform. To get information about the localization in time of a particular feature additionally to information about its dominant local frequencies one looks at the signal through a window. The Fourier analysis is applied to $f_t(s) = f(s) 1_{[t-h, t+h]}(s)$ instead of $f(s)$. More generally, one considers a Kernel $K_h(u) = \frac{1}{h} K(\frac{u}{h})$ instead of the indicator function.

Definition: Let K be a kernel as in chapter 2, and let

$$f_t(s) = K_h(t - s)f(s)$$

be the windowed signal with location center t . Then, the Fourier transform of f_t

$$\begin{aligned} \tilde{f}(\omega, t) &\equiv \hat{f}_t(\omega) = \int e^{-i\omega s} f_t(s) ds \\ &= \int e^{-i\omega s} K_h(t - s) f(s) ds \end{aligned}$$

is the windowed Fourier - or Gabor-transform of f .

Let $g_{\omega, t}(s) = e^{i\omega s} K_h(t - s)$, and

$$\begin{aligned} \hat{g}_{\omega, t}(\delta) &= \int e^{-i\delta s} g_{\omega, t}(s) ds \\ &= \int e^{-i(\delta - \omega)s} K_h(t - s) ds = e^{i(\omega - \delta)t} \hat{K}_h(\delta - \omega) \end{aligned}$$

be its Fourier transform, where \hat{K}_h denotes the Fourier transform of K_h . Then, using Parseval's equation (for continuous time Fourier transform)

$$\begin{aligned} \tilde{f}(\omega, t) &\equiv \langle g_{\omega, t}, f \rangle = \frac{1}{2\pi} \langle \hat{g}_{\omega, t}, \hat{f} \rangle \\ &= e^{-i\omega t} \frac{1}{2\pi} \int e^{i\delta t} \overline{\hat{K}_h(\delta - \omega)} \hat{f}(\delta) d\delta \end{aligned}$$

i.e. $\tilde{f}(\omega, t)$ can be calculated by means of a convolution w.r.t. frequency as well as a convolution w.r.t. time (in its definition above).

4 Neural Networks

A **neural network** is a nonlinear system transforming real input variables x_1, \dots, x_p into one or several output variables y_1, \dots, y_q , using several intermediate steps. Graphically, a network is represented as a directed graph as in Figure 1, where the nodes of the graph are partitioned into several layers. In the **input layer**, each node represents one input variable, and, analogously, in the **output layer**, each node corresponds to one output variable. Inbetween, there are one or several **hidden layers**, the nodes of which are neither sources nor sinks of the graph. The network of Figure 1 has only one hidden layer. Additionally, it is a **feedforward network**, as all edges which originate from a node end up in a node of one of the subsequent layers, never in a node of the same or one of the previous layers.

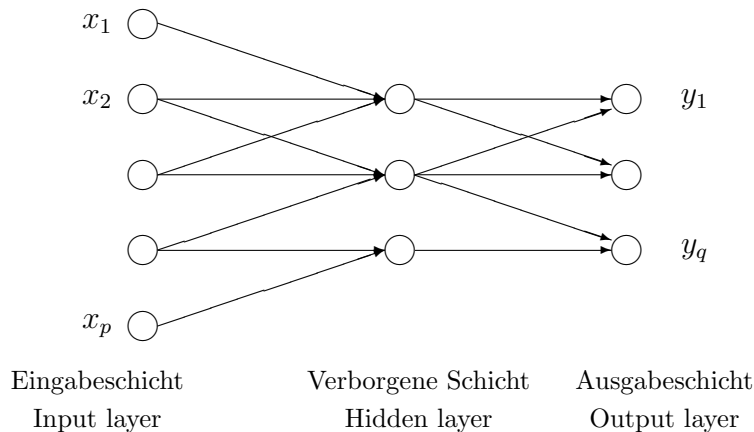


Figure 1

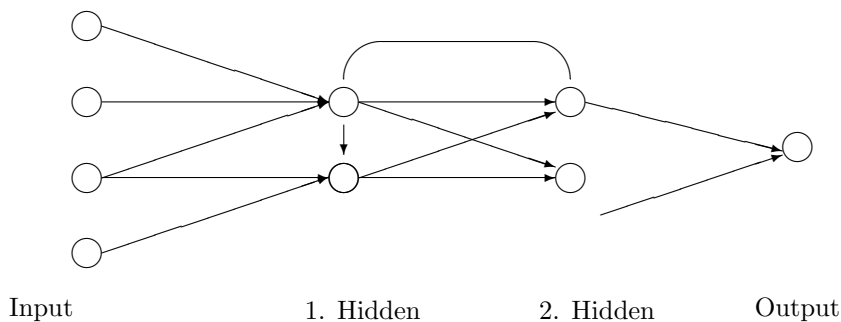


Figure 2 shows a **feedback network** with feedback edges joining nodes of the second hidden layer with the nodes of the first hidden layer. In the following, we restrict ourselves to feedforward networks.

4.1 From Perceptron to Nonlinear neuron

The perceptron is a simple mathematical model for the function of a neural cell which receives signals from other neural cells (the inputs) and, depending on their intensity, remains inactive or "fires", i.e. sends a signal to subsequent cells. In spite of its drawbacks, the perceptron was quite influential for the development of neural networks, and it is a good starting point for discussing the building blocks of networks. The perceptron works in two

steps:

- the input variables x_1, \dots, x_p are multiplied with **weights** w_1, \dots, w_p and added,
- a **thresholding operator** is applied to the result.

Let $\mathbf{x} = (x_1, \dots, x_p)^T$, $\mathbf{w} = (w_1, \dots, w_p)^T$ denote the vectors of inputs and weights, and, for some real b , let $f(u) = 1_{(b, \infty)}(u)$ be the corresponding thresholding function. Then, the output variable $y = f(\mathbf{w}^T \mathbf{x})$ of the perceptron is 1 (the cell "fires"), if the weighted sum of the input signals is above the threshold b , and 0 otherwise (the cell remains inactive).

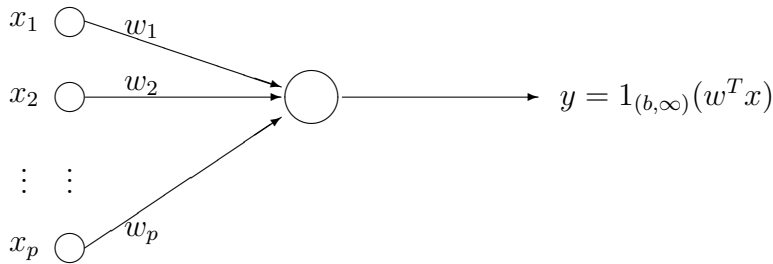


Figure 3: The perceptron

The 0-1-function of the input variables, given by the perceptron, depends on the weights w_1, \dots, w_p and the threshold b . An equivalent representation introduces the constant $x_0 \equiv 1$ as an additional input variable, multiplied by the weight $w_0 = -b$, and selects 0 as the value of the threshold, as, then,

$$1_{(b, \infty)} \left(\sum_{i=1}^p w_i x_i \right) = 1_{(0, \infty)} \left(\sum_{i=0}^p w_i x_i \right).$$

This representation is often more convenient, as one has not to distinguish between two different free parameters (weights and threshold).

A perceptron may be trained, i.e. its parameters are adapted, to solve classification problems of the following type:

Given objects belonging to one of two classes C_0 or C_1 . Use observations x_1, \dots, x_p , made on an object, to decide, if it belongs to C_0 or C_1 .

The perceptron with weights w_0, \dots, w_p classifies an object to belong to C_0 or C_1 , resp., if the output variable $y = y(x_1, \dots, x_p)$ is 0 or 1. To solve the classification problem as well as possible, i.e. without making too many errors, the weights w_0, \dots, w_p have to be "learned". For this purpose, we rely on a **training set** $(\mathbf{x}^{(1)}, z^{(1)}), \dots, (\mathbf{x}^{(T)}, z^{(T)})$ of T input vectors $\mathbf{x}^{(t)} = (x_1^{(t)}, \dots, x_p^{(t)})^T$ with correct classifications $z^{(1)}, \dots, z^{(T)} \in \{0, 1\}$ known. Using so-called **learning rules**, appropriate weights $\hat{w}_0, \dots, \hat{w}_p$ are determined from the training set.

In statistical language, we have to estimate the parameters of the perceptron from the data $(\mathbf{x}^{(t)}, z^{(t)}), t = 1, \dots, T$. A learning rule is an estimation procedure providing estimates

$\hat{w}_0, \dots, \hat{w}_p$.

A typical learning rule is, e.g., the delta- or **Widrow-Hoff-rule**:

The input vectors $\mathbf{x}^{(t)}, t = 1, \dots, T$, are successively used as inputs of the perceptron, and the outputs $y^{(t)}, t = 1, \dots, T$, are compared to their targets, i.e. to the correct classifications $z^{(t)}, t = 1, \dots, T$. If one of these steps results in $y^{(t)} = z^{(t)}$, then the weights are not changed. If, however, $y^{(t)} \neq z^{(t)}$, then the weight vector $\mathbf{w} = (w_0, \dots, w_p)^T$ is adapted to the data in the following manner:

$$\mathbf{w}_{\text{new}} = \mathbf{w} + \eta(z^{(t)} - y^{(t)}) \mathbf{x}^{(t)}$$

η is a small relaxation factor, which sometimes has to converge to 0, to achieve convergence of the algorithm. The initial value of \mathbf{w} is selected arbitrarily or randomly, e.g. uniformly distributed on $[0, 1]^{p+1}$.

The learning algorithm does not stop if all input vectors have been presented to the network, but after using $\mathbf{x}^{(T)}$ as an input, the algorithm starts over again at the start of the sample, using input $\mathbf{x}^{(1)}$. The iteration proceeds step by step through the whole training set several times, until all objects are classified correctly by the network or some measure for the number and importance of errors becomes satisfactorily small.

Remark: The weights w_0, \dots, w_p are identifiable only up to a positive scaling factor, i.e. for any $\alpha > 0$ the same classification is achieved with weights $\alpha w_0, \dots, \alpha w_p$. Using the Widrow-Hoff or other learning rules, it may happen that $\|w\|$ increases over all bounds, leading to numerical problems. To avoid this undesirable effect, so-called **weight decay** techniques are employed, which guarantee a stable norm $\|w\|$ of weights.

Example: Let $p = 2$ and $x_1, x_2 \in \{0, 1\}$. The classification to be learned is the logical XOR:

$$\begin{aligned} z &= 1, \text{ if } x_1 = 1 \text{ or } x_2 = 1, \\ z &= 0, \text{ if } x_1 = 0 \text{ and } x_2 = 0. \end{aligned}$$

The training set consists of input vectors (including as the first coordinate $x_0 = 1$)

$$\mathbf{x}^{(1)} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \mathbf{x}^{(2)} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \mathbf{x}^{(3)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{x}^{(4)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

with correct classifications $z^{(1)} = z^{(2)} = z^{(3)} = 1, z^{(4)} = 0$. The perceptron with weights w_0, w_1, w_2 classifies an object as 1 iff

$$w_0 x_0 + w_1 x_1 + w_2 x_2 > 0,$$

and as 0 else. As initial vector we use $\mathbf{w} = (0, 0, 0)^T$, and we set $\eta = 1$. The steps of the Widrow-Hoff-algorithm are now:

- 1) $\mathbf{w}^T \mathbf{x}^{(1)}$ implies $y^{(1)} = 0 \neq z^{(1)}$. The weights are modified:
 $\mathbf{w}_{\text{new}} = (0, 0, 0)^T + (1 - 0)(1, 1, 0)^T = (1, 1, 0)^T$

- 2), 3) The perceptron with these weights classifies the next two inputs $\mathbf{x}^{(2)}, \mathbf{x}^{(3)}$ correctly.

- 4) For $\mathbf{x}^{(4)}$ we get $\mathbf{w}^T \mathbf{x}^{(4)} = 1$, such that the weights are modified again:
 $\mathbf{w}_{\text{new}} = (1, 1, 0)^T + (0 - 1)(1, 0, 0)^T = (0, 1, 0)^T$
- 5) $\mathbf{x}^{(1)}$ is considered as an input again, and the perceptron with the current weights classifies correctly.
- 6) As $\mathbf{w}^T \mathbf{x}^{(2)} = 0$:
 $\mathbf{w}_{\text{new}} = (0, 1, 0)^T + (1 - 0)(1, 0, 1)^T = (1, 1, 1)^T$
- 7) As $\mathbf{w}^T \mathbf{x}^{(3)} = 3 > 0$, $\mathbf{x}^{(3)}$ is classified correctly.
- 8) $\mathbf{x}^{(4)}$ is misclassified, such that
 $\mathbf{w}_{\text{new}} = (1, 1, 1)^T + (0 - 1)(1, 0, 0)^T = (0, 1, 1)^T$

Now, the algorithm essentially stops, as the weight vector $(0, 1, 1)^T$ achieves correct classification of all input vectors in the training set. The perceptron has learned the XOR-function on the set $\{0, 1\}^2$.

The perceptron is not able to learn arbitrary classifications, i.e. 0-1-functions. The classical counterexample is the logical XOR (= exclusive or):

$$\begin{aligned} z &= 1, \text{ if either } x_1 = 1 \text{ or } x_2 = 1, \\ z &= 0, \text{ if } x_1 = x_2 = 0 \text{ or } x_1 = x_2 = 1. \end{aligned}$$

A perceptron with weights w_0, w_1, w_2 determines a hyperplane $w_0 + w_1 x_1 + w_2 x_2 = 0$ in the space \mathbb{R}^2 of inputs $(x_1, x_2)^T$, separating the set of objects classified as 0 by the perceptron from the objects classified as 1. It is obvious that, for the XOR-function, no hyperplane separates the inputs $\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ with function value 1 from the inputs $\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ with function value 0.

Definition: For $p \geq 1$ two subsets $\mathcal{X}_0, \mathcal{X}_1 \subseteq \mathbb{R}^p$ are called **linearly separable**, if there are $\mathbf{w} \in \mathbb{R}^p, w_0 \in \mathbb{R}$ such that

$$\begin{aligned} w_0 + \mathbf{w}^T \mathbf{x} &> 0 \quad \text{for } x \in \mathcal{X}_1, \\ w_0 + \mathbf{w}^T \mathbf{x} &\leq 0 \quad \text{for } x \in \mathcal{X}_0. \end{aligned}$$

The perceptron with p input variables x_1, \dots, x_p (plus the constant $x_0 \equiv 1$) is able to learn just those classifications which correspond to linearly separable subsets.

If a perfect reproduction of a given classification by a perceptron is not possible, one may try at least to find a "good" classification, i.e. to determine the weights w_0, \dots, w_p such that some measure of misclassification is minimized. An example is the **least-squares (LS-) classification:**

Given the training set $(\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), \dots, (\mathbf{x}^{(T)}, \mathbf{z}^{(T)})$. For given w_0 determine the weights w_1, \dots, w_p such that

$$\begin{aligned} Q(\mathbf{w}) &= Q(w_1, \dots, w_p) = \sum_{i=1}^T (z^{(i)} - y^{(i)})^2 = \min! \\ &\text{with } y^{(i)} = 1_{(0, \infty)}(w_0 + \mathbf{w}^T \mathbf{x}^{(i)}), \quad \mathbf{w} = (w_1, \dots, w_p)^T \end{aligned}$$

w_0 can be chosen arbitrarily, as the weights w_0, \dots, w_p are indentifiable only up to a scale factor (see above). For the perceptron, which performs a binary classification, i.e. defines

a 0-1-function, $Q(\mathbf{w})$ is simply the number of misclassifications. The LS-approach may however be generalized to other statistical problems. The achievable minimum of $Q(\mathbf{w})$ is 0 (perfect classification of the elements of the training sets), iff the two sets

$$\mathcal{X}_0^{(T)} = \{\mathbf{x}^{(i)}, i \leq T; z^{(i)} = 0\}, \mathcal{X}_1^{(T)} = \{\mathbf{x}^{(i)}, i \leq T; z^{(i)} = 1\}$$

are linearly separable.

The Widrow-Hoff-algorithm solves the LS-classification problem, but there are lots of other learning rules or estimation procedures which do as well. The perceptron alone is not flexible enough for many applications. Therefore, more general types of **neurons** are considered as building blocks of neural networks:

Let $\mathbf{x} = (x_1, \dots, x_p)^T$, $\mathbf{w} = (w_1, \dots, w_p)^T$ be input- and weight vectors. For $\vartheta, \vartheta_0 \in \mathbb{R}$, let

$$f_\vartheta(t) = \frac{1}{1 + \exp(-\frac{t+\vartheta}{\vartheta_0})}$$

be the logistic function, due to the form of its graph often called "the" **sigmoid function**, though other functions with sigmoid graphs, e.g. the distribution function of the standard normal law, may be used as well. The output of the neuron is $y = f_\vartheta(\mathbf{w}^T \mathbf{x})$.

For $\vartheta_0 \rightarrow 0+$, $f_\vartheta(t)$ converges to a thresholding function:

$$f_\vartheta(t) \longrightarrow 1_{(0,\infty)}(t + \vartheta) \quad \text{for } \vartheta_0 \longrightarrow 0+,$$

such that the perceptron is a limiting case of a neuron with logistic activation function.

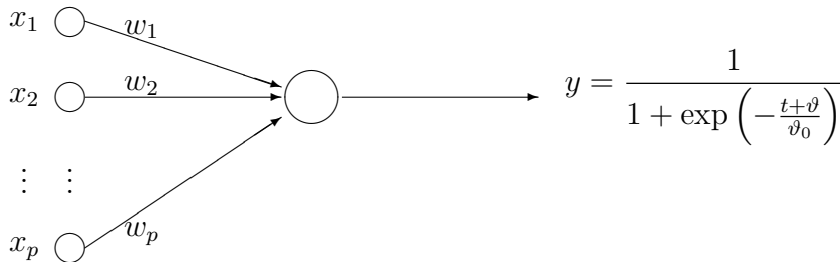


Figure 4: neuron mit sigmoider Übertragungsfunktion

ϑ_0 is often not written down explicitly, because as a scaling factor it may be incorporated into the other parameters w_1, \dots, w_p , ϑ of the neuron. If one sets additionally $w_0 = \vartheta$ and $x_0 \equiv 1$, the output may be written also as:

$$y = f(w_0 + \mathbf{w}^T \mathbf{x}) = f\left(\sum_{k=0}^p w_k x_k\right) \quad \text{with } f(t) = \frac{1}{1 + e^{-t}}.$$

Combining several neurons with sigmoid or - in the limit - thresholding activation functions to a feedforward neural network, we get a so-called **multilayer perceptron** (MLP). Figure

5 shows such a neural network with two input variables plus the constant $x_0 \equiv 1$, two sigmoid neurons in the hidden layer, which are connected by another sigmoid neuron to the output variable, where $f(t) = \{1 + e^{-t}\}^{-1}$ as above.

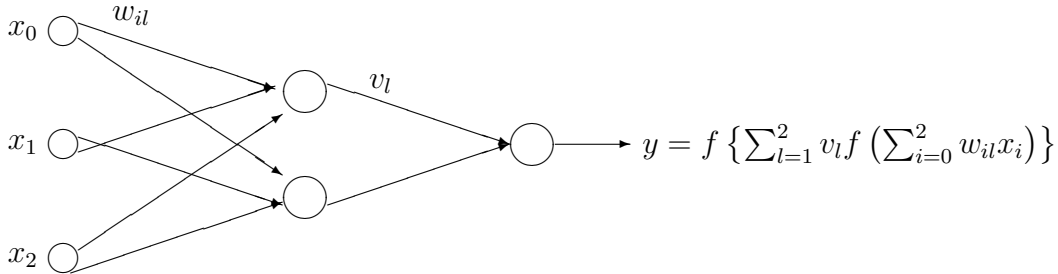


Figure 5

Correspondingly, neural networks with more than one hidden layer may be built, which generate more than one output variable. The connections between the nodes of the layer do not have to be included completely, i.e. the corresponding weights are set to 0 in advance. Instead of the logistic an other sigmoid functions, other activation functions may be considered for some or all neurons. Common choices are thresholding functions, the identity (called a "linear" perceptron) or so-called **radial basis functions**(RBF) where the latter are symmetric kernel-type functions like the density of the standard normal law. In this case, one speaks of a RBF-network instead of a MLP.

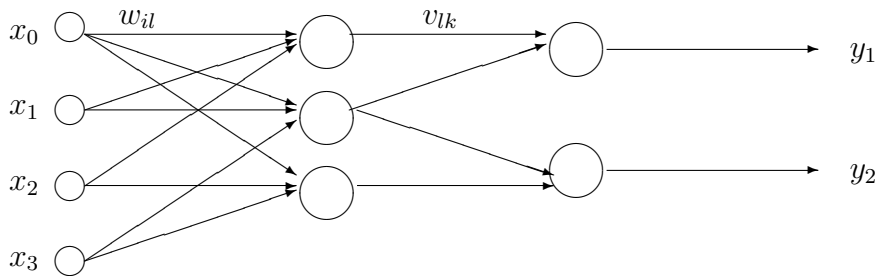


Figure 6

Figure 6 shows an incomplete neural network with two output variables. The weights w_{13} , w_{22} , w_{31} , v_{12} and v_{31} are set to 0, and the corresponding edges are not drawn in the network graph. The output variable y_1 is, e.g.,

$$y_1 = v_{11}f(w_{01} + w_{11}x_1 + w_{21}x_2) + v_{21}f(w_{02} + w_{12}x_1 + w_{32}x_3),$$

a linear combination of the outputs of the two upper neurons of the hidden layer.

Remark: Up to now, we discussed only the most common case that a neuron transforms a linear combination of numbers generated in the previous layer nonlinearly. Occasionally, the output of a neuron may also be of the form $f(\prod_{i=1}^p w_i x_i)$ or $f(\max_{i=1, \dots, p} x_i)$.

Neural networks of MLP-type may be used for solving classification problems as well as regression and forecasting problems. To find an appropriate network for a given task, the weights have to be learned from the training set, i.e. the network parameters have to be estimated from data given as training set $(\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), \dots, (\mathbf{x}^{(T)}, \mathbf{z}^{(T)})$. $\mathbf{x}^{(i)} \in \mathbb{R}^p$ are the observed input vectors, $\mathbf{z}^{(i)} \in \mathbb{R}^q$ are the target values for the outputs. These target vectors $\mathbf{z}^{(i)}$ are compared to the outputs $\mathbf{y}^{(i)} \in \mathbb{R}^q$ produced by the network. The weights are determined such that the deviations between the $\mathbf{z}^{(i)}$ and the $\mathbf{y}^{(i)}$ are small. An example is again the **least squares- (LS-) approach**:

Given the training set $(\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), \dots, (\mathbf{x}^{(T)}, \mathbf{z}^{(T)})$. The weights $w_{0h}, h = 1, \dots, H, x_0 \equiv 1$ are given in advance, where H is the number of neurons in the first hidden layer. Determine the weights of all other edges joining nodes of the network (between input layer and first hidden layer, between subsequent hidden layers and between the last hidden layer and the output nodes) such that

$$\sum_{k=1}^T \|\mathbf{z}^{(k)} - \mathbf{y}^{(k)}\|^2 = \min!$$

In the network of Figure 6 the minimization has to be done with respect to weights $w_{11}, w_{12}, w_{21}, w_{23}, w_{32}, w_{33}, v_{11}, v_{21}, v_{22}, v_{32}$. The weights w_{01}, w_{02}, w_{03} can be preselected arbitrarily due to the nonidentifiability of scale factors.

Remark: Instead of the LS-approach, other loss functions are used in practice too, e.g. weighted LS-distances or, in particular for classification, the Kullback-Leibler-distance:

$$\sum_{k=1}^T \sum_i \left\{ z_i^{(k)} \log \frac{z_i^{(k)}}{y_i^{(k)}} + (1 - z_i^{(k)}) \log \frac{1 - z_i^{(k)}}{1 - y_i^{(k)}} \right\} = \min!$$

As only the $y_i^{(k)}$ depend on the weights, this equivalent to minimizing the cross entropy between the z_i and the y_i , both assumed to be values in $(0, 1)$:

$$- \sum_{k=1}^T \sum_i \left\{ z_i^{(k)} \log y_i^{(k)} + (1 - z_i^{(k)}) \log(1 - y_i^{(k)}) \right\} = \min!$$

Backpropagation is a well-known learning rule which allows feedforward networks to learn their weights from the training set. Mathematically, it is nothing but a special numerical procedure for solving the nonlinear LS-problem. It requires not much storage, but may converge slowly or even not at all.

For sake of illustration, we consider a neural network with one output variable y (i.e. $q = 1$) and one hidden layer with only one neuron:

$$y = f(w_0 + \mathbf{w}^T \mathbf{x}).$$

f may be the logistic or any other activation function. The training set is $(x^{(1)}, z^{(1)}), \dots, (x^{(T)}, z^{(T)})$. The weight w_0 of the constant is fixed in advance. The function to be minimized is

$$Q(w) = \sum_{k=1}^T (z^{(k)} - y^{(k)})^2.$$

It depends only on the weights w_1, \dots, w_p of the input variables.

An elementary numerical procedure for minimizing Q is **gradient descent**. Starting from some weight vector $\mathbf{w}(N)$ the next approximation is calculated by proceeding into the direction where Q decreases as fast as possible:

$$\begin{aligned}\mathbf{w}(N+1) &= \mathbf{w}(N) - \eta \operatorname{grad} Q(\mathbf{w}(N)), \\ \operatorname{grad} Q(\mathbf{w}) &= -\sum_{k=1}^T 2(z^{(k)} - y^{(k)})f'(\mathbf{w}^T \mathbf{x}^{(k)})\mathbf{x}^{(k)}.\end{aligned}$$

To improve the rate of convergence, the tuning constant $\eta > 0$ may be decreased in the course of the iteration.

The gradient descent algorithm judges the quality of the weights $\mathbf{w}(N)$, i.e. of the current network, by looking simultaneously at all the data in the training set. The network is applied to all $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$, and the weight vector is modified only afterwards.

Backpropagation is also a kind of a gradient descent procedure, but it looks at single data points one after the other. The network is applied successively to the $\mathbf{x}^{(k)}$, and after each single step, the weights are modified, again in the direction of steepest descent, but of the function $Q_k(\mathbf{w}) = (z^{(k)} - y^{(k)})^2$ instead of Q :

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \eta \operatorname{grad} Q_k(\mathbf{w}(n)), \quad k = 1, \dots, T, \\ \operatorname{grad} Q_k(w) &= -2(z^{(k)} - y^{(k)})f'(\mathbf{w}^T \mathbf{x}^{(k)})\mathbf{x}^{(k)}.\end{aligned}$$

After one pass through the training set, the iteration starts over again with the first data point. T steps of backpropagation correspond roughly to one step of a batchmode algorithm like gradient descent.

The Widrow-Hoff learning rule is essentially a backpropagation algorithm, keeping in mind that the thresholding function $f(t) = 1_{(0, \infty)}(w_0 + t)$ is not differentiable. After presenting $\mathbf{x}^{(k)}$ to the network, the weights are modified into the direction of steepest descent of $Q_k(w)$ too, i.e. into the direction $\mathbf{x}^{(k)}$ for $z^{(k)} = 1, y^{(k)} = 0$ and into the direction $-\mathbf{x}^{(k)}$ for $z^{(k)} = 0, y^{(k)} = 1$. For correct classification, the weights remain unchanged.

Of course, one may use any numerical algorithm for solving nonlinear extremum problems, to determine the weights corresponding to a minimum of the function $Q(\mathbf{w})$. In particular, conjugate gradient methods seem to work well frequently in applications. All algorithms run the risk to end in a local minimum of $Q(\mathbf{w})$ however.

4.2 Neural network regression

We consider the general nonparametric regression model

$$Y_j = m(X_j) + \varepsilon_j \quad , \quad j = 1, \dots, N$$

$X_1, \dots, X_N \in \mathbb{R}^p$ i.i.d.

To estimate m , we fit a suitable neural network to the data or training set $(X_1, Y_1), \dots, (X_N, Y_N)$. For sake of simplicity, we consider only one hidden layer with H neurons having activation function ψ . The network function, depending on parameters $\vartheta = (v_h, 0 \leq h \leq H, w_{ih}, 0 \leq i \leq p, 0 \leq h \leq H)$, is given by

$$f_H(x; \vartheta) = v_0 + \sum_{h=1}^H v_h \psi(w_{0h} + \sum_{i=1}^p w_{ih} x_i).$$

ϑ is estimated by nonlinear least-squares:

$$\hat{\vartheta}_H = \arg \min_{\vartheta \in \Theta_H} \sum_{j=1}^N (Y_j - f_H(X_j; \vartheta))^2,$$

and $m(x)$ is estimated by

$$\hat{m}_H(x) = f_H(x; \hat{\vartheta}_H).$$

Θ_H is chosen such that the parameters or network weights are identifiable, i.e. $f_H(\cdot; \vartheta_1) \neq f_H(\cdot; \vartheta_2)$ if $\vartheta_1 \neq \vartheta_2$ for $\vartheta_1, \vartheta_2 \in \Theta_H$. If, e.g., $\psi(u) = -\psi(-u)$, identifiability is guaranteed by assuming $v_1 > v_2 > \dots > v_H > 0$.

For fixed H , $\hat{m}_H(x) \xrightarrow[p]{p} f_H(x; \vartheta_0)$ for $N \rightarrow \infty$, where $\vartheta_0 \in \Theta_H$ is the network parameter vector corresponding to the best fit to m :

$$\begin{aligned} \vartheta_0 &= \arg \min_{\vartheta \in \Theta_H} \int (m(x) - f_H(x; \vartheta))^2 p(x) dx_1 \dots dx_p \\ &= \arg \min_{\vartheta \in \Theta_H} D_0(\vartheta) \end{aligned}$$

with $D_0(\vartheta) = \mathcal{E}(Y_j - f_H(X_j; \vartheta))^2$, and $p(x)$ is the density of X_1, X_2, \dots . Consider also the sample version $\vartheta_N = \arg \min D_N(\vartheta)$ of ϑ_0 , where

$$D_N(\vartheta) = \frac{1}{N} \sum_{j=1}^N (m(X_j) - f_H(X_j; \vartheta))^2.$$

Let $\nabla^2 D_0(\vartheta) = (\frac{\partial^2}{\partial \vartheta_k \partial \vartheta_e} D_0(\vartheta))_{k,e}$, $\nabla D_0(\vartheta) = (\frac{\partial}{\partial \vartheta_k} D_0(\vartheta))_k$ denote Hesse Matrix and gradient of D_0 . Then:

Theorem 4.1 *Assume*

1. $\Psi \in C^2$ with bounded derivatives, m bounded, continuous

2. $D_0(\vartheta)$ has a unique global minimum ϑ_0 in $\text{int}(\Theta_H)$, and $A(\vartheta_0) \equiv \nabla^2 D_0(\vartheta_0)$ is positive definite

3. X_1, X_2, \dots i.i.d. with density $p(x)$; $\varepsilon_1, \varepsilon_2, \dots$ i.i.d. with

$$\mathcal{E}\{\varepsilon_j | X_j = x\} = 0, \quad \mathcal{E}\{\varepsilon_j^2 | X_j = x\} = \sigma_\varepsilon^2(x) < \infty$$

4. σ_ε^2 continuous, $0 < \delta \leq \sigma_\varepsilon^2(x) \leq \Delta < \infty$ for all x ,

$$\mathcal{E}\{|\varepsilon_j|^n | X_j = x\} \leq C_n < \infty \text{ for all } x, \quad n \geq 1.$$

Then, $\sqrt{N} \begin{pmatrix} \hat{\vartheta}_H - \vartheta_N \\ \vartheta_N - \vartheta_0 \end{pmatrix} \xrightarrow{p} \mathcal{N}\left(0, \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}\right)$,

with

$$\Sigma_i = A(\vartheta_0)^{-1} B_i(\vartheta_0) A(\vartheta_0)^{-1}, \quad i = 1, 2,$$

$$B_1(\vartheta) = 4 \int \sigma_\varepsilon^2(x) \nabla f_H(x; \vartheta) \nabla^T f_H(x; \vartheta) p(x) dx_1 \dots dx_p$$

$$B_2(\vartheta) = 4 \int (m(x) - f_H(x; \vartheta))^2 \nabla f_H(x; \vartheta) \nabla^T f_H(x; \vartheta) p(x) dx_1 \dots dx_p.$$

As a consequence $\sqrt{N}(\hat{\vartheta}_H - \vartheta_0)$ is asymptotically normal, too. Its two asymptotically independent parts $\hat{\vartheta}_H - \vartheta_N$ and $\vartheta_N - \vartheta_0$ represent the variance part and the bias part respectively. In particular, the bias part is asymptotically negligible ($\sqrt{N}(\hat{\vartheta}_H - \vartheta_0) \xrightarrow{p} 0$) if $m(x) = f_H(x; \vartheta_0)$, i.e. the network correctly specifies the true regression function m .

In general, the theorem only implies asymptotic normality of $\hat{m}_H(x) - f_H(x; \vartheta_0)$. To get a consistent estimate $\hat{m}_H(x) \xrightarrow{p} m(x)$, the network size, i.e. in particular H , has to increase with N with an appropriate rate.

4.3 Network specification

Determining the structure of a neural network as a nonparametric regression model uses analogous techniques as known from classical linear regression. The following questions have to be answered, given a data set $(X_1, Y_1), \dots, (X_N, Y_N)$:

- How many hidden layers?
- How many neurons in each hidden layer?
- Which nodes of the network (inputs, hidden neurons, outputs) have to be connected?

A parsimonious network is desirable, i.e. a network containing just enough, but not too many parameters to provide an adequate fit to the data.

We consider only networks with one hidden layer with one output unit, but the ideas can be used straightforwardly for more complicated networks, too. Let w_{ih} and v_h denote again the weights corresponding to connections between input nodes and hidden neurons resp. hidden neurons and output nodes, and let the network function $f_H(x; \vartheta)$ be defined as in 4.2.

- a) Repeated significance testing (like, e.g., stepwise regression)
Start with simple network, add one neuron no. H .

Test $H_0 : v_H = 0$ against $H_1 : v_H \neq 0$.

This is a nonstandard testing problem as parameters w_{0H}, \dots, w_{pH} are not identifiable under H_0 . Use, e.g., Lagrange multiplier tests.

If H_0 is rejected, test if some of the connections between inputs and the new neuron may be deleted:

$$H_0 : w_{iH} = 0 \quad \text{against} \quad H_1 : w_{iH} \neq 0.$$

Based on asymptotic results like Theorem 4.1, standard Wald tests do the job.

b) Crossvalidation and validation

Let $\hat{\vartheta}_H$ be defined as in 4.3, and the leave-one-out estimate $\hat{\vartheta}_{H,-j}$ analogously but using data (X_i, Y_i) , $i \neq j$, only. H may be selected as minimizer of

$$CV(H) = \frac{1}{N} \sum_{j=1}^N (Y_j - f_H(X_j; \hat{\vartheta}_{H,-j}))^2.$$

Usually, that is computationally too expensive. Instead, one uses only part of the data $(X_1, Y_1), \dots, (X_T, Y_T)$ as the training set from which $\hat{\vartheta}_H^{(T)}$ is estimated. The rest of the data $(X_{T+1}, Y_{T+1}), \dots, (X_N, Y_N)$ serves as validation set (or test set) to compare the performance of different networks, where

$$V(H) = \frac{1}{N-T} \sum_{j=T+1}^N (Y_j - f_H(X_j; \hat{\vartheta}_H^{(T)}))^2$$

should be small.

c) Network information criterion (NIC)

Based on the general AIC (Akaike's information criterion) procedure, Murata et. al. derived a special form for comparing different networks (assuming implicitly Gaussian residuals ε_j):

$$\hat{\sigma}_{\varepsilon, H}^2 = \frac{1}{N} \sum_{j=1}^N (Y_j - f_H(X_j; \hat{\vartheta}_H))^2$$

$$NIC(H) = N \hat{\sigma}_{\varepsilon, H}^2 + \text{tr} (BA^{-1}) = \min_H$$

where $\text{mse}_H(\vartheta) = \mathcal{E}(Y_j - f_H(X_j; \vartheta))^2$ and

$$A = \mathcal{E} \nabla^2 \text{mse}_H(\vartheta), \quad B = \text{covariance matrix of } \sqrt{N} \nabla \text{mse}_H(\vartheta).$$

In practice, A, B have to be estimated, where \mathcal{E} is replaced by a sample mean, e.g.

$$\hat{A}_{ke} = \frac{1}{N} \sum_{j=1}^N \frac{\partial^2}{\partial \vartheta_k \partial \vartheta_e} [(Y_j - f_H(X_j; \vartheta))^2].$$