Neural networks and logistic regression
Part II

Werner Vach[1], Reinhard Roßner[2] and Martin Schumacher[1,2]

[1]Center for Data Analysis and Model Building
University of Freiburg
Albertstr. 25-28, D-79104 Freiburg im Breisgau
Germany

[2]Institute of Medical Biometry and Informatics
University of Freiburg
Stefan-Meier-Str. 26, D-79104 Freiburg im Breisgau
Germany

7 Introduction to Part II

This is the second part of a paper investigating the relationship between feed forward neural nets and logistic regression. In the first part (Schumacher, Roßner & Vach, this volume) we focussed on computational and practical aspects. We also demonstrated that feed-forward neural networks with a hidden layer can be used to increase the class of regression functions for $Pr(y = 1|x)$. This increased flexibility is often seen as a substantial advantage of neural networks; in the first part we have already seen that this may also be a source of dramatic overfitting. Hence in this second part we want to investigate in more detail, what we can expect when we use neural networks in settings which seem to require an increased flexibility in modeling $Pr(y = 1|x)$.

In the first part of the paper we have only considered linear logistic models, where the logits are linear in x, i.e.

$$\mathrm{logit}(x) = \log \frac{Pr(y = 1|x)}{Pr(y = 0|x)} = \beta_0 + \sum_{i=1}^{I} \beta_i x_i \ .$$

In the second part we will consider models where the logits also involve quadratic terms and multiplicative interactions. In applied statistics such models are usually considered as the first step to overcome the stringent assumption of additive and purely linear effects of the covariates. They are a traditional approach to increase the class of regression functions $Pr(y = 1|x)$ and hence they are natural competitors to neural networks with a hidden layer.

In section 8 we investigate how and how well regression functions of logistic models can be approximated by a feed-forward neural net with one hidden layer. It is well known that such a network can approximate any reasonable function with arbitrary precision, if the number of hidden units tends to infinity (Cybenko 1989, Funahashi 1989, Hornik et al. 1989). The question in practice is, whether we can approximate regression functions relevant in biomedical applications with a few hidden units. Hence our investigation can be seen as an empirical supplement to the theoretical results of Barron (1993). It will also illustrate some difficulties in interpreting the weights.

In section 4.2 and 5.1 we have already discussed the use of weight decay. Section 9 further illustrates the necessity of weight decay in order to avoid overfitting in problems with sample sizes relevant for biomedical examples.

The behaviour of neural nets in such settings is compared with model selection procedures based on logistic models in Section 10 by means of a simulation study.

8 Approximation of logistic regression functions by feed-forward neural nets

A feed-forward neural net with one hidden layer and J hidden units represents the regression functions

$$f(x) = \Lambda(W_0 + \sum_{j=1}^{J} W_j \Lambda(w_{0j} + \sum_{i=1}^{I} w_{ij} x_i)) \tag{8.1}$$

with arbitrary values for $W_0, \ldots, W_j, w_{01}, \ldots, w_{IJ}$. In this section we investigate how regression functions of this type can approximate specific regression functions of logistic models with quadratic terms and multiplicative interactions, that is regression functions from the class

$$g(x) = \Lambda(\beta_0 + \sum_{i=1}^{I} \beta_i x_i + \sum_{i=1}^{I} \gamma_i x_i^2 + \sum_{i<i'} \delta_{ii'} x_i x_{i'}) \tag{8.2}$$

with arbitrary values for $\beta_0, \ldots, \beta_I, \gamma_1, \ldots, \gamma_I, \delta_{12}, \ldots, \delta_{I-1,I}$. To investigate, how well a given regression function $g(x)$ is approximated by a function $f(x)$, we consider the distance measures

$$D_2(f,g) = \int_{x\in[0,1]^I} (f(x) - g(x))^2 \, dx$$

and

$$D_{KL}(f,g) = -\int_{x\in[0,1]^I} f(x) \ln g(x) + (1 - f(x)) \ln(1 - g(x)) \, dx$$

These distance measures are justified by the fact, that if the number N of individuals tends to infinity and $x^{(n)}$ is uniformly distributed on $[0,1]^I$, then the solutions of the minimization problems (3.2) or (3.5) tend to the regression function f_0 minimizing $D_2(\cdot|g)$ or $D_{KL}(\cdot|g)$, respectively, if g is the true regression function. In the sequel we present some results, where for a given function g the optimal function f_0 within the class (8.1) for a

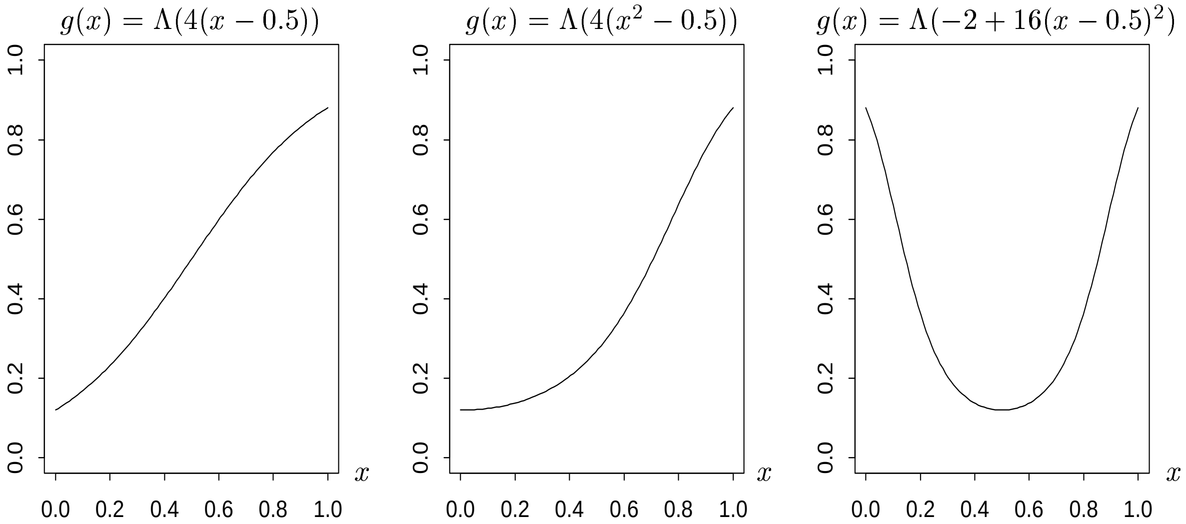


Figure 8.1: Graphical representations of the true regression functions $g(x)$ of Table 8.1.

fixed number J of hidden units is given. The necessary computations have been done by the `FindMinimum` command of MATHEMATICA (Wolfram, 1991). Start values were determined by fitting a neural net to a large sample generated according to $Pr(y = 1|x) = g(x)$ with uniformly distributed x. For a representation of the optimal function f_0 we always chose a parametrization with $W_j \geq 0 \, \forall j = 1, \ldots, J$. This is always possible due to the fact that

$$W_j \Lambda(w_{0j} + \sum_{i=1}^{I} w_{ij} x_j) = W_j - W_j \Lambda(-w_{0j} - \sum_{i=1}^{I} w_{ij} x_j)$$

As the optimal functions with respect to D_2 and D_{KL} are nearly the same, we present in the following only the results with respect to D_{KL}.

We start our investigation with the case $I = 1$. For three regression functions (shown in Figure 8.1) the best approximations are presented in Table 8.1; in order to judge the achieved accuracy the root of the integrated squared distance and the maximal absolute deviations are given, too. For the first function the logit is linear in x. Such a function can be always approximated exactly using a single hidden unit, if we allow that the weights W_0 and W_1 tends to infinity. This approximation is achieved in the following manner: The weights $w_{01} := -0.5c$ and $w_{11} := c$ describe for small values of c

$g(x)$	J	$f(x)$	$\sqrt{D_2(f,g)}$	$\max_x \lvert f(x)-g(x)\rvert$
$\Lambda(4(x-0.5))$	1	$\lim_{c\to 0}\Lambda(-\frac{8}{c}$ $+\frac{16}{c}\Lambda(-0.5c+cx))$	0	0
$\Lambda(4(x^2-0.5))$	1	$\Lambda(-2.79$ $+43.8\Lambda(-4.21+2.14x))$	0.006	0.015
$\Lambda(-2+16(x-0.5)^2)$	2	$\Lambda(-11.8$ $+18.8\Lambda(-3.50+3.72x)$ $+56.2\Lambda(-1.18-1.63x))$	0.007	0.023

Table 8.1: Best approximation $f(x)$ within the class (8.1) to $g(x)$ with respect to $D_{KL}(f,g)$.

a mapping $x \to w_{01} + w_{11}x$, which maps the interval $[0,1]$ to a small interval symmetrically around 0. Within this interval the logistic function $\Lambda(\cdot)$ is nearly linear, hence the output function $\Lambda(w_{01} + w_{11}x)$ of the hidden unit is nearly linear and maps the interval $[0,1]$ to a small interval symmetrically around 0.5. Finally the weights $W_0 := -\frac{8}{c}$ and $W_1 := \frac{16}{c}$ are chosen such that the small interval is mapped to the interval $[-2,2]$, and hence $W_0 + W_1\Lambda(w_{01} + w_{11}x)$ is nearly identical to $4(x-0.5)$.

For the second function the logit is a quadratic, but still monotone function of x. An exact approximation with one hidden unit is not possible, but a rather accurate approximation with a maximal deviation of 0.015 is. Here the weights w_{01} and w_{11} are chosen such that the interval $[0,1]$ is mapped to the interval $[-4.21, -2.07]$. Within this interval the logistic function Λ is similar to a monotone quadratic function, hence the shape of the output function of the hidden unit approximates the shape of $x \mapsto 4(x^2-0.5)$. For the third function the logit is not a monotone function of x. As a function represented by a net with one hidden unit is always a monotone function of x, a sufficient approximation based on one hidden unit is not possible. However, with two hidden units a sufficient approximation is possible. It is based on the decomposition of the function $x \mapsto -2 + 16(x-0.5)^2$ into the sum of two monotone functions represented by the output functions of the hidden units.

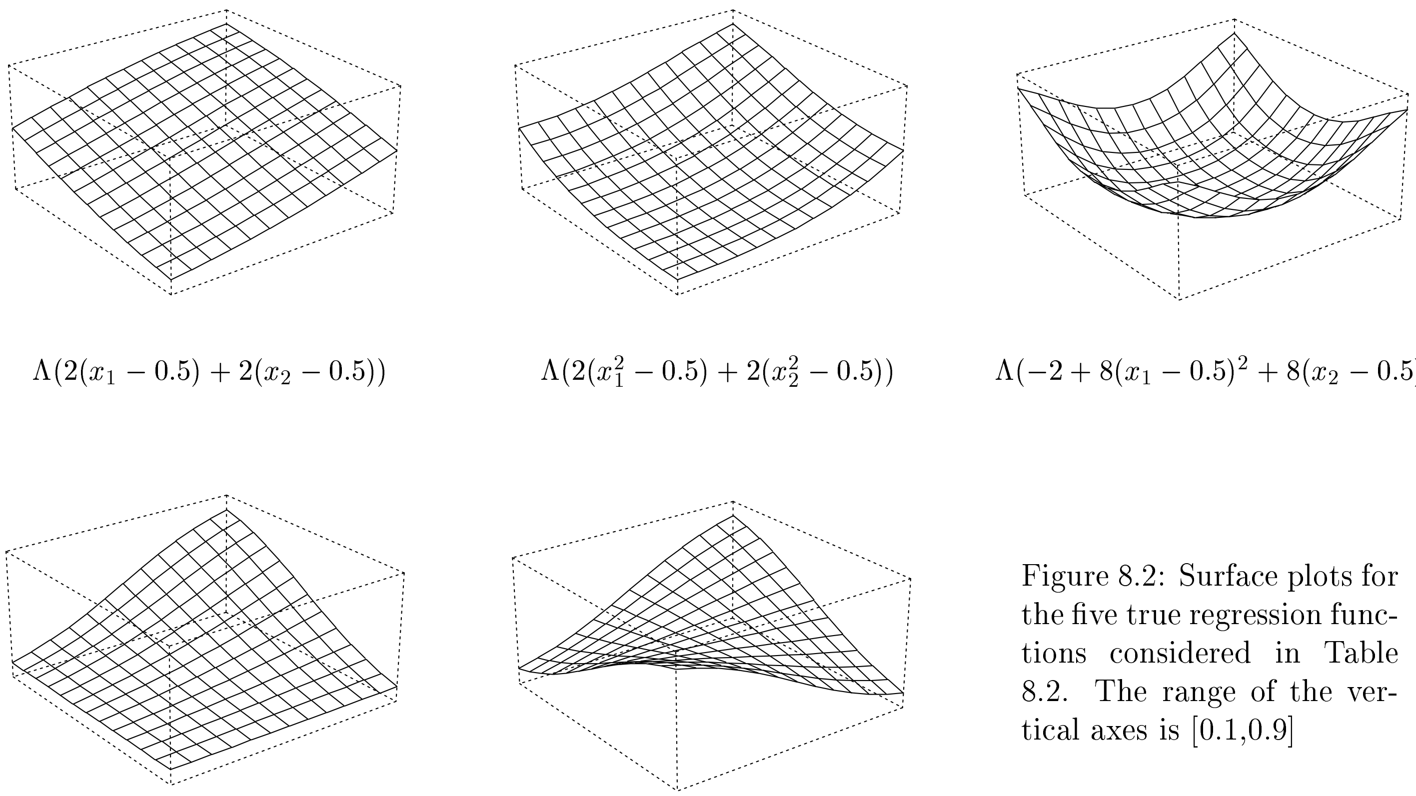
In the second step we consider 5 regression functions of two covariates, i.e. $I = 2$ (Table 8.2). Surface plots of the 5 functions are shown in Figure 8.2. If, as for the first function, the logit is a linear function of x_1 and

x_2, then an exact approximation with one single unit is possible again. For the second function the logit is the sum of two monotone functions of x_1 and x_2, respectively. As shown above each of these two functions can be approximated by the output function of a single hidden unit, hence it is not surprising that now we can find a sufficient approximation with two hidden units, where the output function of each unit depends mainly on one covariate. For the third function the logit is a sum of two non-monotonous functions of x_1 and x_2, respectively. From our considerations above we can expect a sufficient approximation using 4 hidden units. However, 3 hidden units turn out to be sufficient. The gradients of the output functions of the three hidden units are shown in Figure 8.3, illustrating how the shape of the logit is approximated by the sum of three functions arranged is a circular manner around the midpoint of $[0, 1]^2$.

So far we have considered regression functions which belong to the class of generalized additive models, i.e. after a transformation of the covariates their effects are still additive (cf. Hastie & Tibshirani 1990). The next step is to allow that one covariate modifies the effect of another and vice versa, that is, we have an interaction. For the fourth regression function the logit is given by $4(x_1x_2 - 0.5)$, i.e. the effect of one covariate increases with the effect of the other covariate. As only the size of the effect, but not the sign varies, we speak of a quantitative interaction. Contrary for the fifth regression function the effect of the second covariate is positive if the first covariate is larger than 0.5, and otherwise it is negative; hence we speak of a qualitative interaction (cf. Byar 1985, Gail & Simon 1985). To approximate the regression function representing a quantitative interaction two hidden units are sufficient. The output of the two hidden units are shown in Figure 8.4. Surprisingly they show a very small variation in dependence of x_1, because the weights w_{11} and w_{12} are both very small. How can we approximate in this way the function shown in Figure 8.2, where the dependence on x_1 vanishes for $x_2 = 0$, but is distinct for $x_2 = 1$? Now, if $x_2 = 1$, the input of the hidden units is around 0, where the curvature of the activation function Λ is high, hence the variation of x_1 has a strong impact on the output of the hidden units; if $x_2 = 0$, the input of the hidden units is around 3.0 or -3.0, respectively, where the curvature of the activation function is small, and here the variation of x_1 has nearly no effect on the output. To approximate the regression function representing a qualitative interaction three hidden units are necessary. In all these examples with $I = 2$ there exists a second optimal f for which the role of x_1 and x_2 are exchanged.

$g(x_1, x_2)$	J	$f(x_1, x_2)$	$\sqrt{D_2(f,g)}$	$\max_{x_1,x_2} \lvert f(x_1,x_2) - g(x_1,x_2)\rvert$
$\Lambda(2(x_1 - 0.5) + 2(x_2 - 0.5))$	1	$\lim_{c\to 0} \Lambda(-\frac{4}{c} + \frac{8}{c}\Lambda(-0.5c + cx_1 + cx_2))$	0	0
$\Lambda(2(x_1^2 - 0.5) + 2(x_2^2 - 0.5))$	2	$\Lambda(-2.23 + 3.81\Lambda(-3.25 + 3.49x_1 - 0.13x_2) + 2.92\Lambda(-3.87 + 0.19x_1 + 4.63x_2))$	0.005	0.027
$\Lambda(-2 + 8(x_1 - 0.5)^2 + 8(x_2 - 0.5)^2)$	3	$\Lambda(-153 + 197\Lambda(-1.92 + 0.69x_1 + 0.43x_2) + 300\Lambda(-1.57 - 0.51x_1 + 0.41x_2) + 195\Lambda(-0.40 + 0.02x_1 - 0.73x_2))$	0.004	0.014
$\Lambda(4(x_1 x_2 - 0.5))$	2	$\Lambda(-32.3 + 29.8\Lambda(3.16 + 0.38x_1 - 3.13x_2) + 34.9\Lambda(-3.09 + 0.12x_1 + 2.83x_2))$	0.008	0.042
$\Lambda(8(x_1 - 0.5)(x_2 - 0.5))$	3	$\Lambda(-46.2 + 29.4\Lambda(2.39 - 2.16x_1 + 0.59x_2) + 34.6\Lambda(-2.77 + 1.98x_1 + 0.85x_2) + 25.6\Lambda(1.05 - 0.39x_1 - 1.19x_2))$	0.008	0.028

Table 8.2: Best approximation $f(x_1, x_2)$ within the class (8.1) to $g(x_1, x_2)$ with respect to $D_{KL}(f, g)$



$\Lambda(2(x_1 - 0.5) + 2(x_2 - 0.5))$ $\Lambda(2(x_1^2 - 0.5) + 2(x_2^2 - 0.5))$ $\Lambda(-2 + 8(x_1 - 0.5)^2 + 8(x_2 - 0.5)^2)$

$\Lambda(4(x_1 x_2 - 0.5))$ $\Lambda(8(x_1 - 0.5)(x_2 - 0.5))$

Figure 8.2: Surface plots for the five true regression functions considered in Table 8.2. The range of the vertical axes is [0.1,0.9]

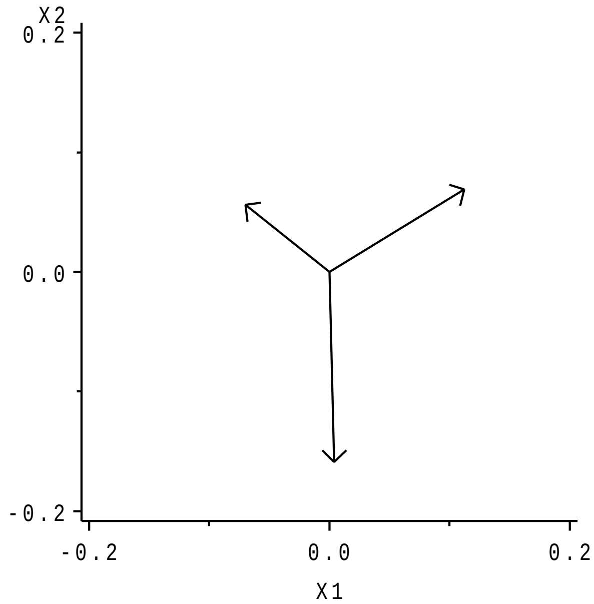


Figure 8.3: The gradients of the functions $\Lambda(-1.92 + 0.69x_1 + 0.43x_2)$, $\Lambda(-1.57 - 0.51x_1 + 0.41x_2)$ and $\Lambda(-0.40 + 0.02x_1 - 0.73x_2)$ at the point $(x_1, x_2) = (0.5, 0.5)$.

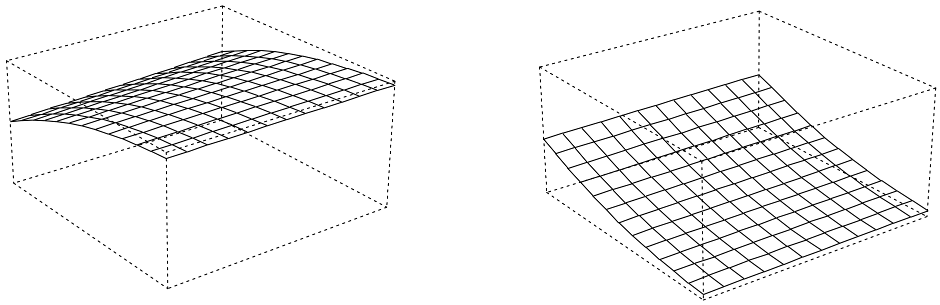


Figure 8.4: The output of the hidden units as a function of (x_1, x_2) for the best approximation to $g(x) = \Lambda(4(x_1x_2 - 0.5))$. The range of the vertical axes is $[0.1, 0.9]$.

9 The necessity of weight decay

The family (8.1) of regression functions contains functions of very different degrees of smoothness. In applied statistics it is well known, that if we try to fit functions from such a family to data from a small population, functions with a low degree of smoothness are more likely to yield a good fit than functions with a high degree of smoothness. With respect to neural nets this effect is demonstrated in Figure 9.1 for the case $I = 1$. We generate five data sets with $N = 50$ individuals, where $x^{(n)}$ is chosen from a uniform distribution on the interval $[0, 1]$, and $y^{(n)}$ is generated according to the law $Pr(y^{(n)} = 1|x^{(n)} = x) = \Lambda(4(x - 0.5))$, corresponding to the first regression function considered above. To each of these five data sets we fit a regression function using a feed-forward net with one hidden unit [1]. The five estimated regression functions are shown in the middle part of Figure 9.1, whereas the right part shows the five functions resulting from fitting a linear logistic model. We observe, that the functions fitted by the neural net do not reflect the smoothness of the original function. Moreover, neural nets seem to prefer functions which are (nearly) jump functions, i.e. functions of the form

$$f^*(x) = \begin{cases} \Lambda(c_0) & \text{if } x \leq x_0 \\ \Lambda(c_1) & \text{if } x > x_0 \end{cases} .$$

Indeed, with a single hidden unit we can approximate such functions, because

$$f^*(x) := \lim_{c \to \infty} \Lambda(c_0 + (c_1 - c_0)\Lambda(-cx_0 + cx)) ,$$

i.e. these functions arise if $|w_{11}|$ tends to infinity. The preference of such jump functions is well known from isotonic regression, where we look for a fit within the class of all monotone functions (Barlow, Bartholomew, Bremner & Brunk, 1972, p. 13). The same phenomena occur in the case $I = 2$. In Figure 6.2 we consider three data sets with $N = 400$ individuals. $x^{(n)}$ is chosen from a uniform distribution, and $y^{(n)}$ is generated according to the law

$$Pr(y^{(n)} = 1|x^{(n)} = x_1, x_2)) = \Lambda(-2 + 8(x_1 - 0.5)^2 + 8(x_2 - 0.5)^2) .$$

[1] We use the program `nnet` written by B.D.Ripley as a subroutine for S-Plus (cf. Section 4.3). To avoid local minima we choose always the best fit within 10 repetitions with randomly chosen start values. We use always the energy function $E^*(w)$ based on the Kullback-Leibler distance.

From the last section we know that we need at least 3 hidden units to be able to find a good approximation to the true regression function. However, the functions fitted by a neural net with three hidden units fail again to reflect the smoothness of the true regression function.

One approach to ensure that the fitted function f is smooth is to add a term to the fit criterion (3.5) penalizing large weights, because a low degree of smoothness corresponds to some large weights as shown above. The specific approach used for neural nets is weight decay, where the penalty is the sum of squares of the weights. Precisely we consider minimization of the criterion

$$E^*(w) + d \times (\sum_{j=1}^{J} W_j^2 + \sum_{i=1}^{I} \sum_{j=1}^{J} w_{ij}^2)$$

where d is a prespecified constant. This corresponds to well known statistical procedures like ridge regression or penalized likelihood, cf. Ripley (1993, section 4.7) and Ripley (1994, section 6.2). In Figure 9.3 we illustrate the application of the weight decay. We consider the five data sets already used in Figure 9.1 and additionally five data sets with $N = 50$ individuals, where $y^{(n)}$ is now generated according to the law $P(y^{(n)} = 1|x^{(n)} = x) = \Lambda(-2 + 16(x - 0.5)^2)$. We fit regression functions using a neural net with one hidden unit and the decay parameters $d = 0.0001, d = 0.001, d = 0.01$ and $d = 0.1$. Choosing d large enough, the regression functions are now smooth, but choosing d too large, the regression functions become too smooth. The problem of choosing an optimal value of d is of course an analogue to similar problems for other techniques in non-parametric regression like choosing a band width for kernel smoothers or choosing the penalty constant in fitting splines. In principle, techniques like cross validation can be used to choose d, but such techniques may computationally too expensive. Similar parallels between neural nets and non-parametric regression are discussed by Geman, Bienenstock and Doursat (1992).

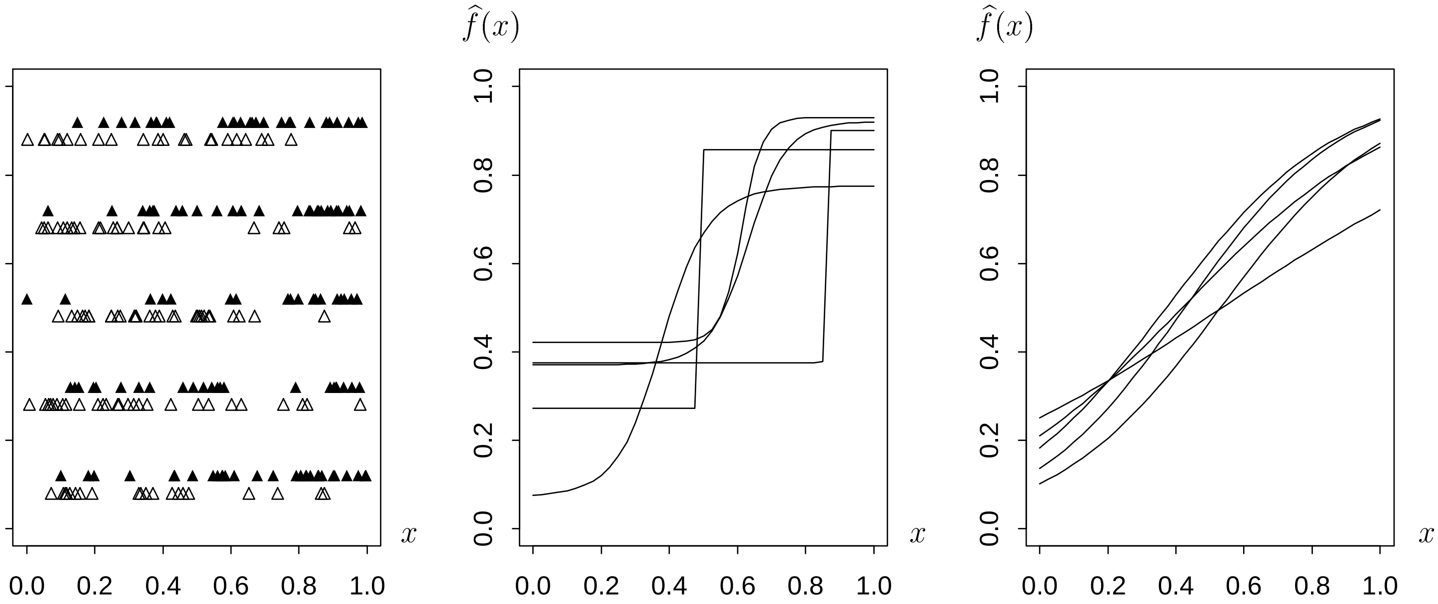


Figure 9.1: Left: Five artificial data sets. Black triangles refer to $y^{(n)} = 1$, white triangles refer to $y^{(n)} = 0$. Middle: Fitted regression functions using a feed-forward neural net with one hidden unit. Right: Fitted regression functions using logistic regression.

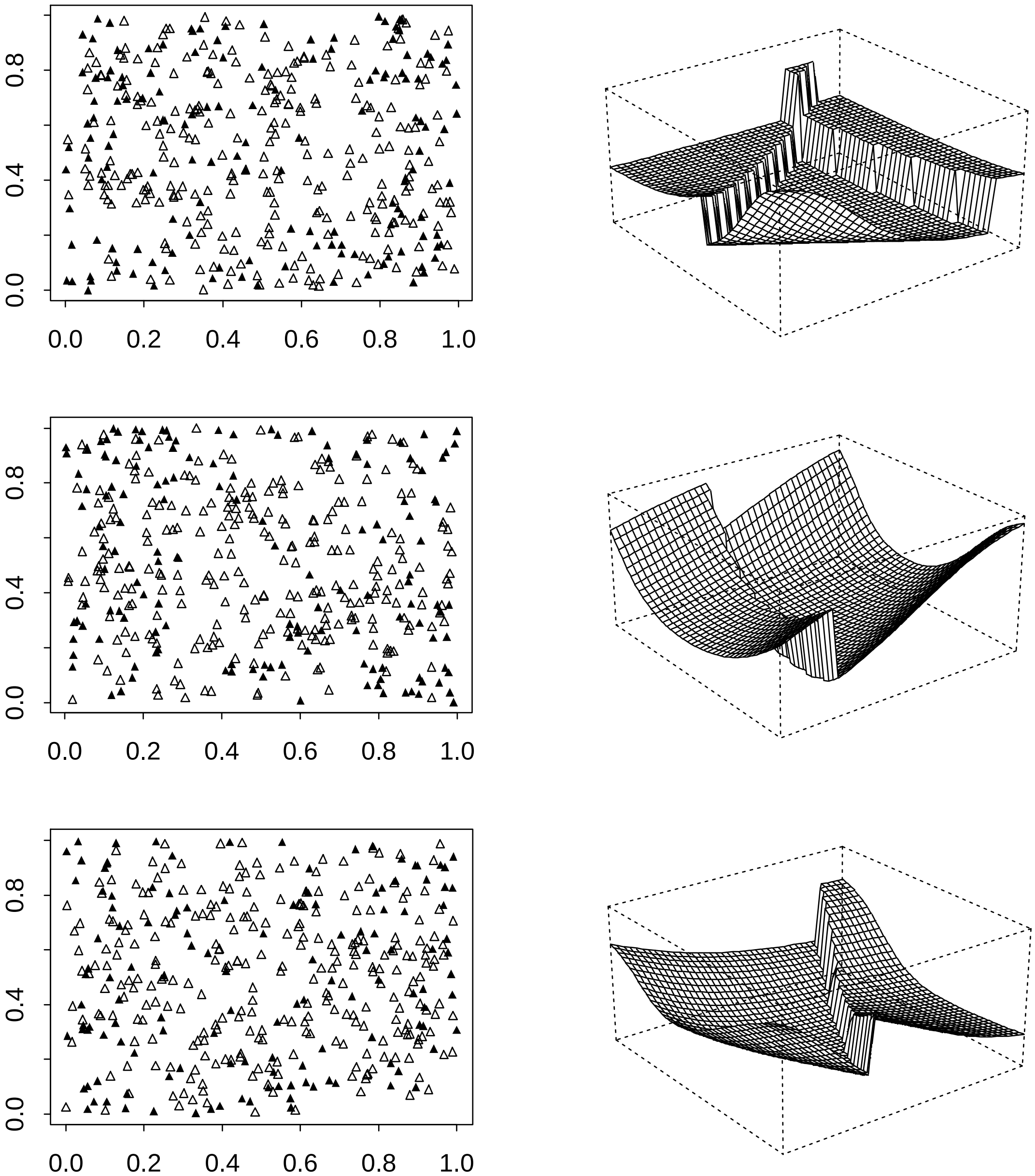


Figure 9.2: Left: Three artificial data sets with two covariates and $N = 400$. Black triangels refer to $y^{(n)} = 1$, white triangles to $y^{(n)} = 0$. Right: Fitted regression functions using a feed forward neural net with 3 hidden units.

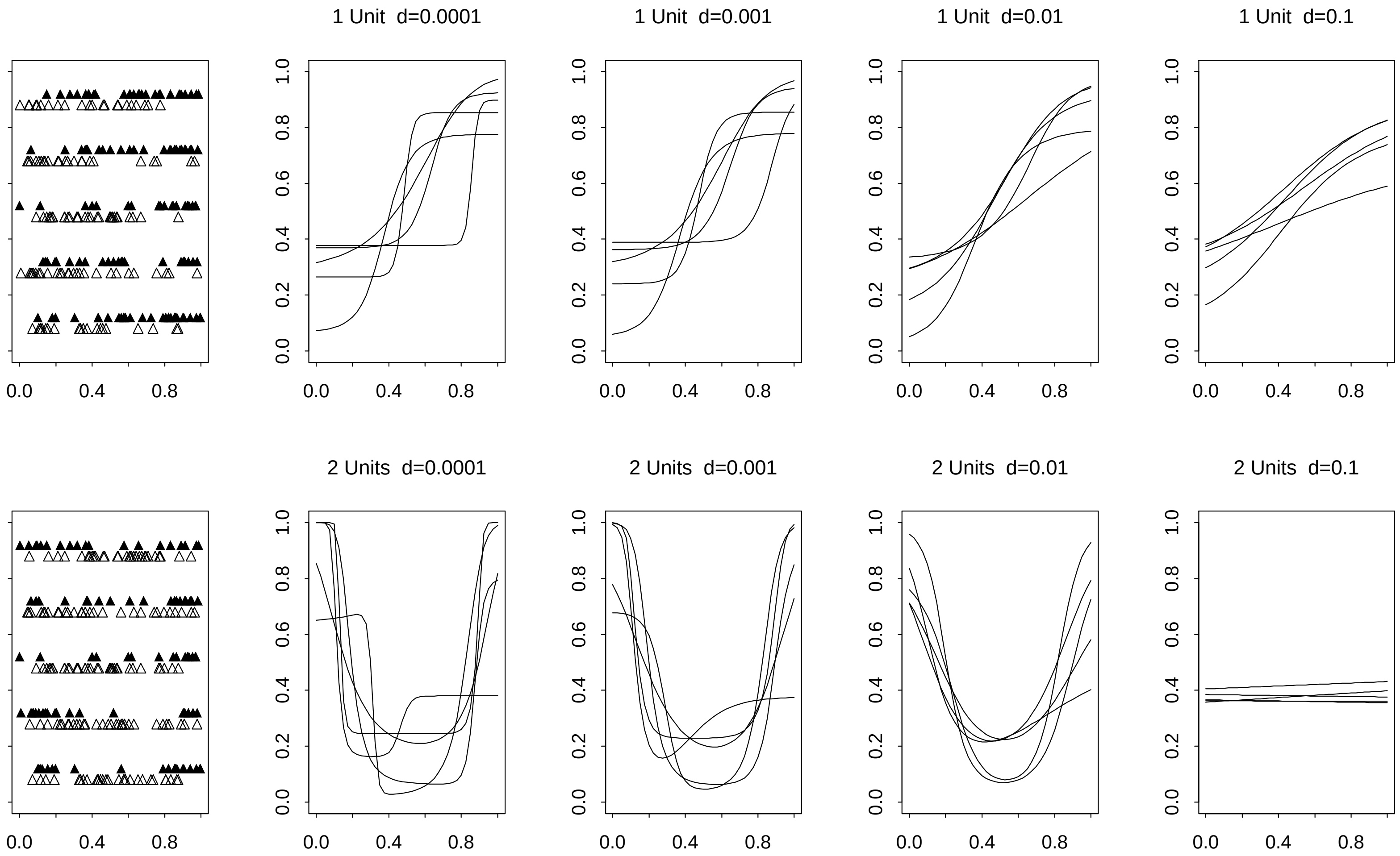


Figure 9.3: Upper row: Five data sets and fitted regression functions using a feed-forward neural net with 1 hidden unit and different deacy parameters. Lower row: Five data sets and fitted regression functions using a feed-forward neural net with 2 hidden units and different deacy parameters.

10 A comparison of neural nets and model selection based on logistic regression

Advocates of neural nets usually emphasize, that neural nets do not rely on the assumption of a linear model. Also most statisticians do not feel well by handling anything with a purely linear model. However, they also hesitate to consider regression models like (8.2) including additional quadratic terms for each covariate and interactions for each pair of covariates. They are afraid of extreme problems with respect to overfitting. As a consequence in many applications model selection strategies are used, at least implicitly. Such strategies aim to find a function within a class like (8.2) where only few parameters are different from 0. There exist a lot of proposals to formulate such strategies explicitly, but they are subject to a lot of criticism and discussions (Weisberg 1985, Sen & Svivastara 1990, Sauerbrei 1993, Cheeseman & Oldford 1994). From this point of view neural nets may be regarded as an interesting alternative. Hence in this section we want to compare the ability of neural nets and model selection procedures based on logistic regression models with respect to the estimation of regression functions corresponding to non-linear logistic regression models. We consider the case of $I = 5$ covariates and $N = 400$ individuals. With respect to neural nets we consider feed forward nets with $J = 1, \ldots, 9$ hidden units and the weight decay parameters $d = 0, 0.01, 0.033, 0.1$ and 0.33. The energy function is based on the Kullback-Leibler distance. With respect to logistic regression we consider ML estimation within four different classes of regression functions. The linear logistic model (L), the logistic model with all quadratic terms and interactions given by (5.2) (called the full model (F) in the sequel), the union of all models with one quadratic term or one interaction (S_1) and the union of all models with two quadratic terms and/or interactions (S_2). Regarding each of these methods as an estimator $\widehat{f}$ for the true regression function g, we measure the quality of the estimator by the expected squared distance $E\ D_2(\widehat{f}, g)$ between $\widehat{f}$ and g, where we assume that $x^{(n)}$ is uniformly distributed over $[0,1]^5$. We approximate $ED_2(\widehat{f}, g)$ by the average of 25 Monte Carlo simulations. In each repetition we generate new values $x^{(1)}, \ldots, x^{(N)}$ and $y^{(1)}, \ldots, y^{(N)}$ according to the law $P(y^{(n)} = 1 | x^{(n)} = x) = g(x)$ and approximate $D_2(\widehat{f}, g)$ by the average of 20000 values $(\widehat{f}(z) - g(z))^2$ with z uniformly from $[0,1]^5$. Instead of using the expected quadratic distance between f and g neural networks are often evaluated based on the error rate. Here the estimated regression function $\widehat{f}$ is reduced to a classification rule to predict $y = 1$ if $\widehat{f}(x) \geq 0.5$ and $y = 0$ if $\widehat{f}(x) \leq 0.5$. The error rate is then the probability to attain a misclassification, if y is generated according to $Pr(y = 1|x) = g(x)$ and x is generated by a prespecified law, here the uniform distribution on $[0, 1]^I$. However, in many biomedical applications the aim is often a more subtile classification or the cutpoint (for example for risk patients needing a special therapy) is far away from 0.5. Hence it is essential to estimate the complete regression function correctly, and the expected squared distance is more appropriate than the error rate.

We start our investigation with five regression models corresponding to those considered in Table 8.2, i.e. only the first two of the five covariates have an effect. If the true regression function is $g(x) = \Lambda(2(x_1-0.5)+2(x_2-0.5))$, then neural nets with a large weight decay compare similar to the linear logistic model (L) independent of the number of hidden units (Figure 10.1a). Model selection (S_1, S_2) or the full model (F) tend to distinctly worse results. The same picture (Figure 10.1.b) occurs for the case with $g(x) = \Lambda(2(x_1^2 - 0.5) + 2(x_2^2 - 0.5))$, i.e. a discrepancy from a linear model by a quadratic but still monotone influence of the covariates is – in this situation of $N = 400$ individuals – not large enough to imply a superiority of neural nets or model selection over the simple linear model. The results are quite different (Figure 10.1c) if we consider $g(x) = \Lambda(-2 + 8(x_1 - 0.5)^2 + 8(x_2 - 0.5)^2)$, where the influence of the first two covariates is not monotone. Here model selection (S_1, S_2), the full model (F) and most neural nets are superior to the linear logistic model. Neural nets with a small weight decay ($d = 0.01, d = 0.033$) are sensitive to the choice of the correct number of hidden units, which is 3 in correspondence with our results from Table 8.2. For a weight decay of $d = 0.1$ it seems to be advisable to use four or more hidden units, whereas the decay $d = 0.33$ is too large. However, the best results with neural nets are not superior to fitting the full model (F) and far away from the results obtained by model selection with enough additional terms (S_2).

Considering the regression function with a quantitative interaction (Figure 10.2a) the deviation from a linear model is not large enough such that the model selection procedure (S_1) becomes much better than the linear model (L). Similarly, neural nets are not better than the linear model, and behave well only for large values of the decay parameters. Contrary, for the regression function with a qualitative interaction (Figure 10.2b) the full model and model selection behave much better than the linear model, and the neural nets can do the same. However, they do not reach the precision shown by the model selection procedures. Neural nets behave best for $d = 0.1$, for smaller decay values the optimal number of nodes is 2 or 3, which slightly corresponds to the result of section 8, that 3 units are necessary for a sufficient approximation of the true regression function.

So far we have considered situations, where only two of the five covariates have an influence. In Figure 10.3 we consider two regression functions where four of the five covariates have an influence. The underlying models involve a qualitative interaction and a non-monotone influence of one covariate. If the single effects are small (Figure 10.3a), neural nets with a large weight decay ($d = 0.1, 0.33$) behave as well as the linear model (L) and the full model (F) and only model selection (S_1, S_2) leads to a small improvement. If the effects are larger (Figure 10.3b), then neural nets with an appropriate weight decay ($d = 0.1$) can be much better than the linear model (L), but fail to improve the full model (F) and are distinctly worse than model selection with two additional terms (S_2).

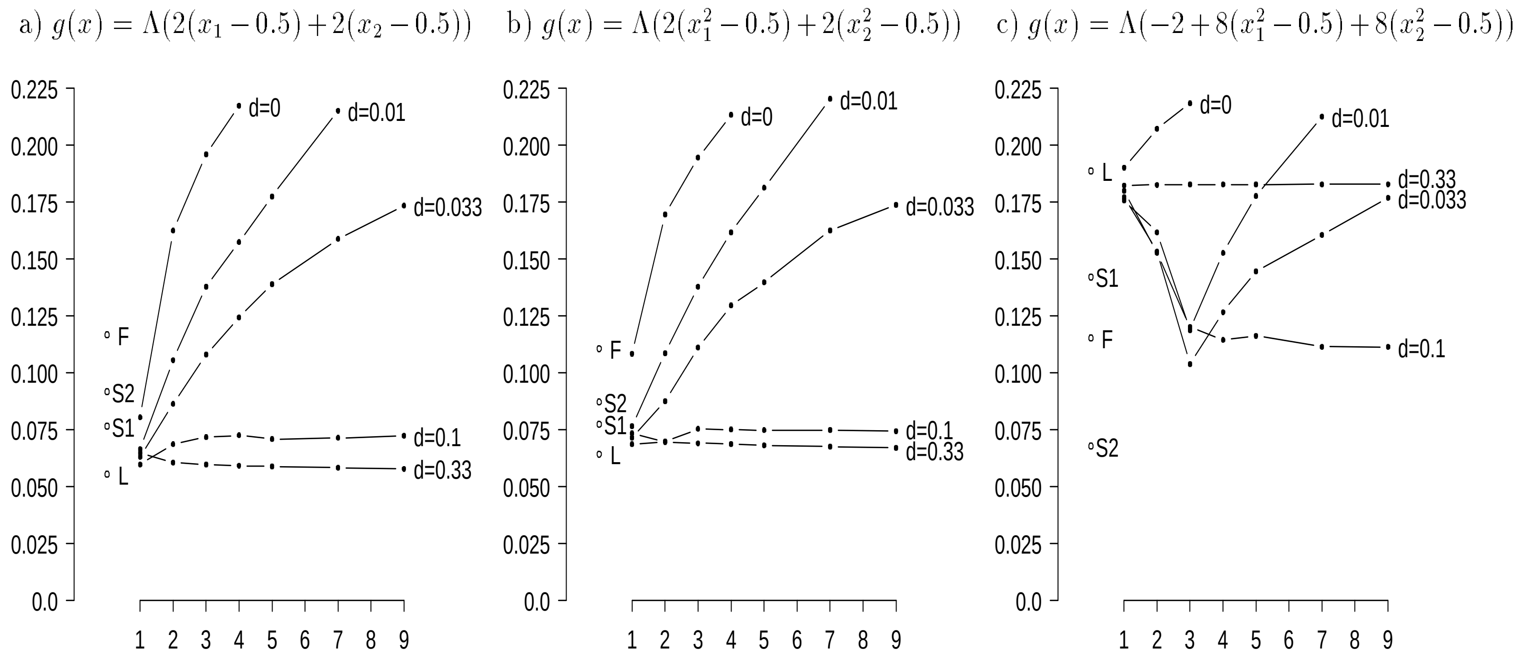


Figure 10.1: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: $\sqrt{ED_2(\hat{f}, g)}$.

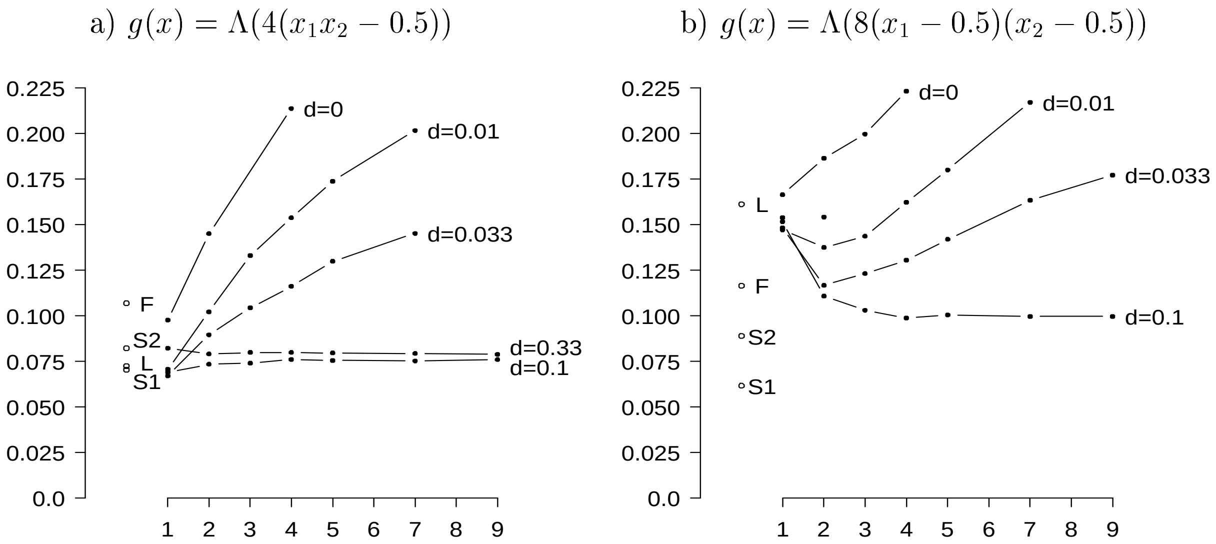


Figure 10.2: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: $\sqrt{ED_2(\widehat{f}, g)}$.

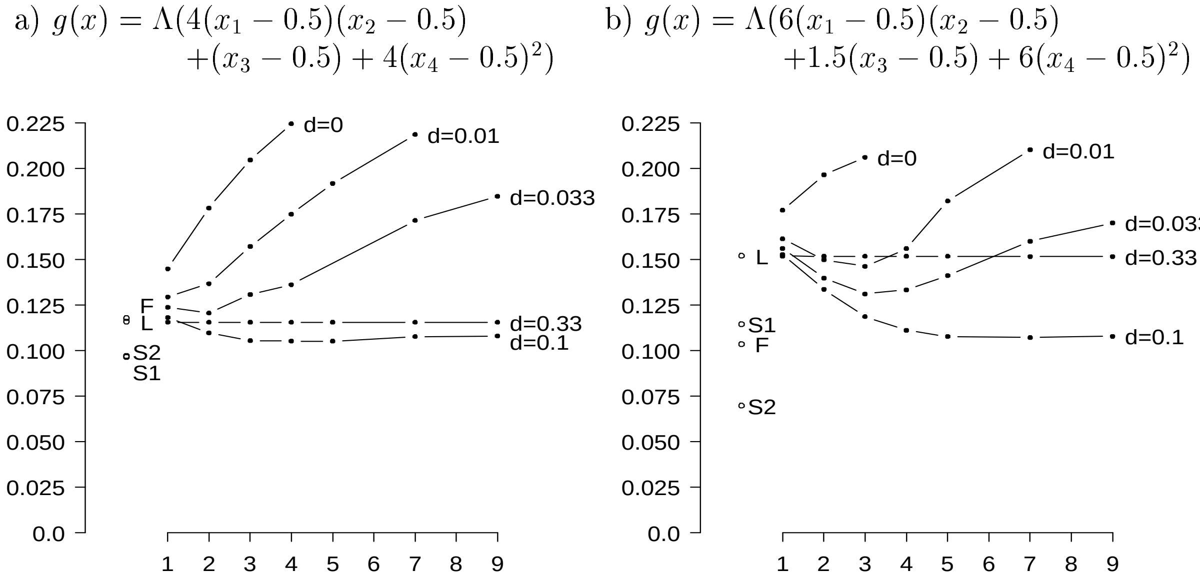


Figure 10.3: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: $\sqrt{ED_2(\widehat{f}, g)}$.

So far the comparison is a little bit unfair, because the true regression functions are always members of the class (8.2). Neural nets may show their superiority, if we consider true regression functions which are far away from being sufficiently approximated by a member of this class. One type of such functions is characterized by the existence of several local extrema. These functions may occur, if the influence of single covariates is a periodic or cubic function; hence two examples are given by

$$\begin{aligned} g(x) &= \Lambda(\sin(2\pi x_1) + \sin(2\pi x_2)) \text{ and} \\ g(x) &= \Lambda(32(x_1 - 0.5)^3 + 6(x_1 - 0.5) + 32(x_2 - 0.5)^3 + 6(x_2 - 0.5)) \end{aligned}$$

The surface plots of these functions (in dependence on x_1 and x_2) are shown in Figure 10.4. From Figure 10.5a) and b) we have to conclude, that the gain in using neural nets is rather small and nearly negligible for the first function. Here, $N = 400$ is not large enough to allow neural nets to demonstrate their enlarged flexibility. Increasing N to 800 (Figure 10.5c) and d)), neural nets can be really much better than logistic models especially for small decay parameters. For the second function even without weight decay we can obtain a substantial improvement.

Another approach to find true regression functions where neural nets are superior is to take a member of the class (8.1). Let us consider the subclass

$$g(x) = \Lambda(2h(c)(\Lambda(c(x_1 - 0.5)) - 0.5) + 2h(c)(\Lambda(c(x_2 - 0.5)) - 0.5))$$

$$\text{with } h(c) := (\Lambda(c\ 0.5) - 0.5)^{-1}$$

where the logit is composed as the sum of two monotone functions of x_1 and x_2, respectively. For small values of c these functions are nearly linear, hence the true regression function can be approximated by a linear model. For large values of c these functions become similar to a jump function with a jump at $x = 0.5$. For $c = 32$ the surface plot (in dependence on x_1 and x_2) is shown in Figure 10.6. These type of functions may have practical importance, if the true underlying (biological or technical) process involves mechanisms changing their behaviour at a fixed threshold value. However, the results shown in Figure 10.7 reveal that also here neural nets fail to improve the linear logistic model distinctly considering $N = 400$. For $N = 800$ an improvement can be obtained, but only if the number of the hidden units is correctly specified as $J = 2$.

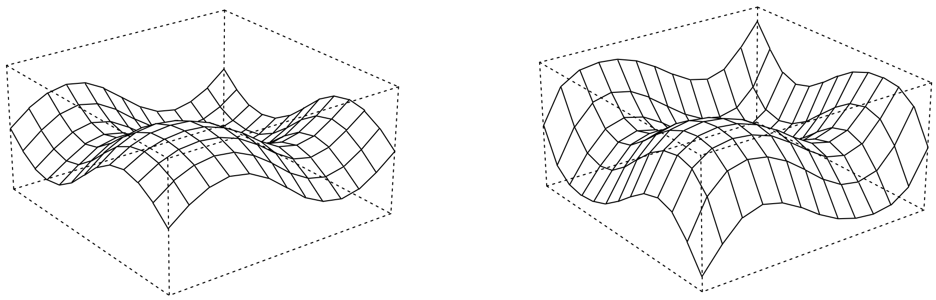


Figure 10.4: Surface plots of the functions $g(x) = \Lambda(\sin(2\pi x_1) + \sin(2\pi x_2))$ (left) and $g(x) = \Lambda(32(x_1 - 0.5)^3 + 6(x_1 - 0.5) + 32(x_2 - 0.5)^3 + 6(x_2 - 0.5))$ (right) on the unit square. The range of the vertical axes is $[0.1, 0.9]$.

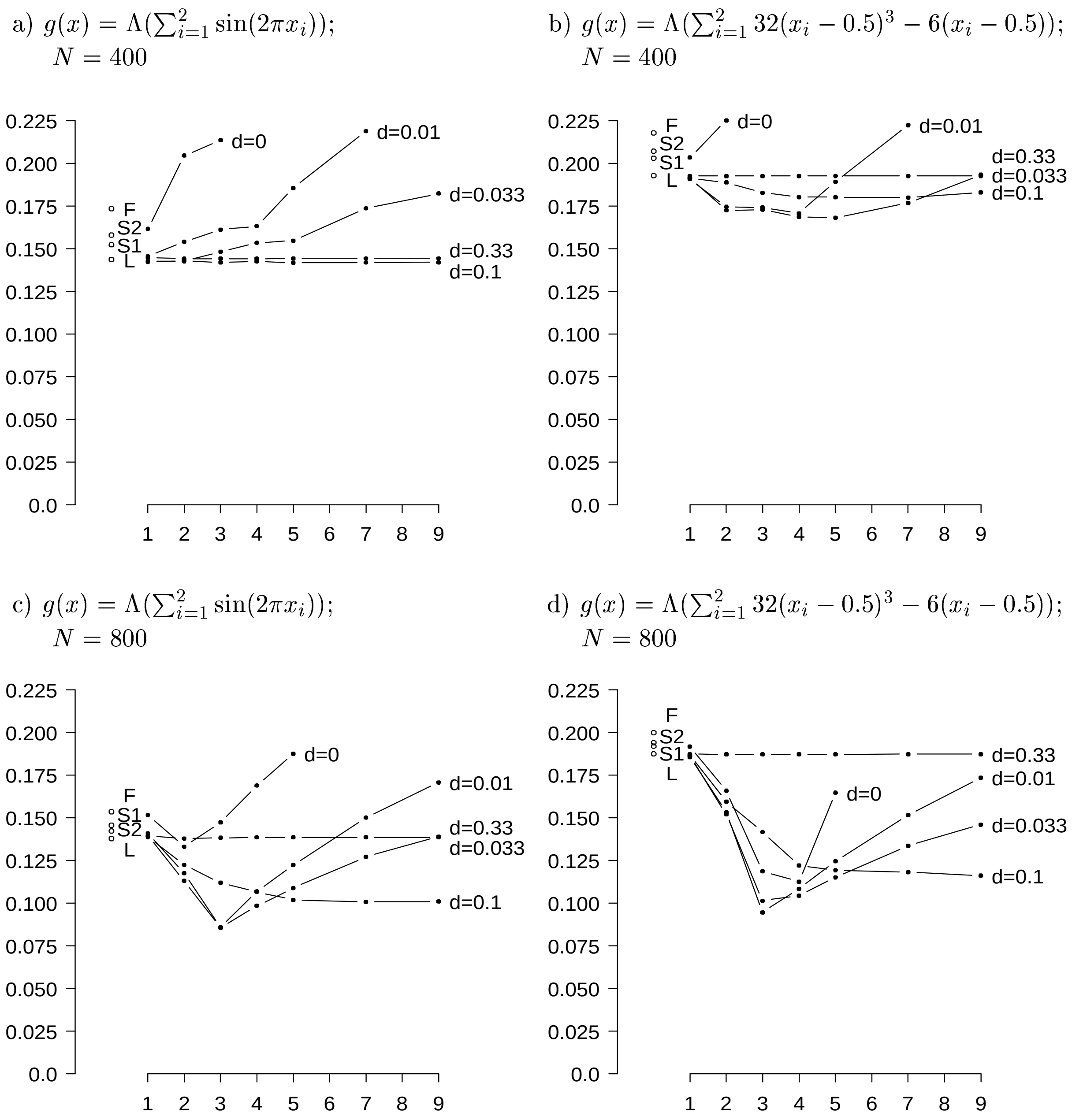


Figure 10.5: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: $\sqrt{ED_2(\widehat{f}, g)}$.

Finally we want to consider a rather extreme class of functions where we can expect superiority of neural nets. These functions are given by

$$g(x) = \begin{cases} 1 & \text{if } x \in \cap_{k=1}^{K} A_k \\ \Lambda(c_0) & \text{otherwise} \end{cases}$$

$$\text{with } A_k := \{x | \sum_{i=1}^{I} a_{ik} x_i \geq a_{0k}\} \text{ with } a_{ik} \in IR \; .$$

Such a regression function may occur, if there is a (small) subregion of $[0,1]^I$, where the occurrence of events is sure, and outside of this region events occur with a (small) constant probability. We have to expect, that neural nets can handle such functions well, because

$$g(x) = \lim_{c \to \infty} (\Lambda(c_0 + \sum_{k=1}^{K} (c\Lambda(c(-a_{0k} + \sum_{i=1}^{I} a_{ik} x_i)) \; ,$$

i.e. the functions $g(x)$ can be approximated by functions of the class (8.1). As a special example we consider the regression function

$$g(x) = \begin{cases} 1 & \text{if } \min(x_1, x_2, x_3) \geq 0.5 \\ \Lambda(-2) & \text{otherwise} \end{cases}$$

In such a situation the correct estimation of the complete function $g(x)$ is of minor interest, we would be satisfied if the two regions with different value of g are well discriminated. Hence we consider here as a measure the error rate of the classification based on $\widehat{f}(x) > 0.5$. Note that even if $\widehat{f}(x) = g(x)$, the error rate will be 0.104. From Figure 10.8 we can conclude, that indeed a neural net can improve the error rate compared with a linear logistic model, especially for $N = 800$ nearly all considered neural nets behave better. The above approximation suggests that 3 hidden units would suffice, but the results from the simulation study indicate that 4 or more hidden units are necessary to obtain optimal results.

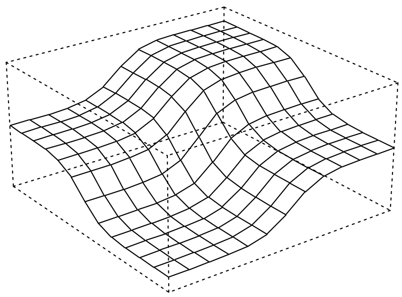


Figure 10.6: Surface plot of the function $g(x) = \Lambda(2 \cdot h(\Lambda(32(x_1 - 0.5)) - 0.5) + 2 \cdot h(\Lambda(32(x_2 - 0.5)) - 0.5))$ with $h := (\Lambda(16) - 0.5)^{-1}$ on the unit square. The range of the vertical axes is $[0.1, 0.9]$.

$$g(x) = \Lambda(\sum_{i=1}^{2} 2(\Lambda(16) - 0.5)^{-1}(\Lambda(32(x_i - 0.5)) - 0.5))$$

a) $N = 400$ b) $N = 800$

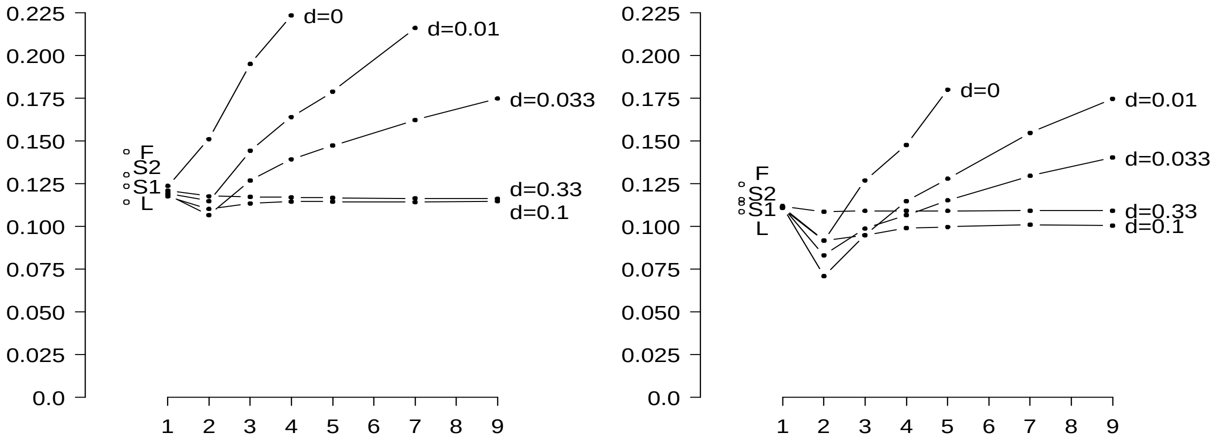


Figure 10.7: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: $\sqrt{ED_2(\widehat{f}, g)}$.

$$g(x) = \begin{cases} 1 & \text{if } \min(x_1, x_2, x_3) \geq 0.5 \\ \Lambda(-2) & \text{otherwise} \end{cases}$$

a) $N = 400$ b) $N = 800$

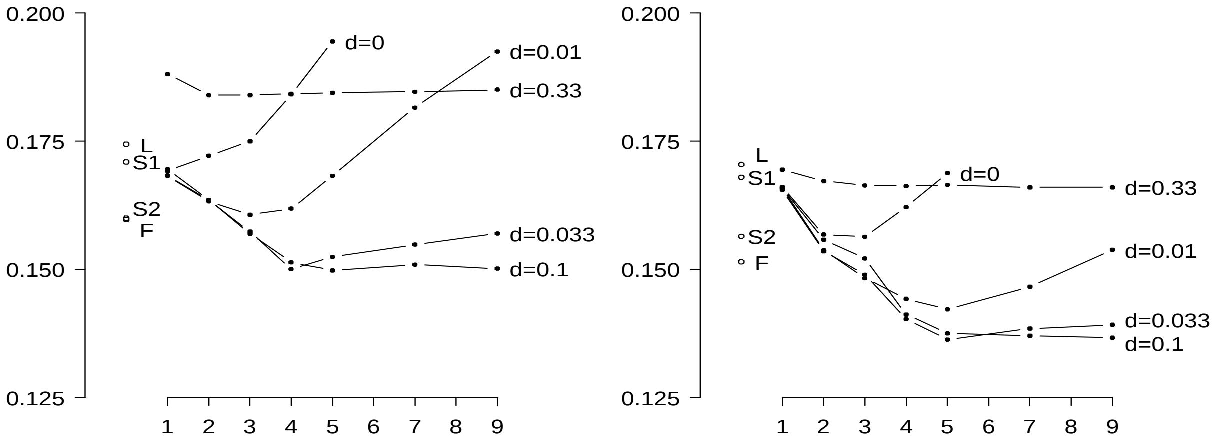


Figure 10.8: Results of simulation studies. Horizontal axis: Number of hidden units. Vertical axis: Error rate.

11 Conclusions

In the second part of this paper we investigated whether neural nets are worth to be considered as an alternative to logistic models in settings relevant for biomedical research. The first drawback of neural nets is that they give us no direct information on the value of a single covariate for the prediction. The examples in section 8 illustrate that there are no simple strategies to interpret the weights in this sense. It remains the question, whether even in those applications which focus on the estimation of the regression function, it is justified to neglect the possible gain in scientific knowledge available from the identification of influential factors.

If we restrict ourselves to estimation of the regression function many biomedical applications involve less than 400 observations and/or more than five covariates. Larger samples are the exception, but the investigations in Section 10 reveal that neural networks need large samples to take advantage of their flexibility. Even if we have large samples, neural nets will be superior to the considered model selection procedures only if the true regression function cannot be approximated by a parsimonious member of the class (8.2). Hence it remains the question about the need of more complex regression functions in a particular biomedical application. In our opinion for most applications such complex functions are not very plausible, because the covariates represent meaningful biological factors. It should be emphasized that the successful use of neural networks appears mainly in fields like pattern recognition, where covariates like the grey scale of a pixel are more or less meaningless with respect to their single values and we can expect results only from combining the values of many pixels.

As a final point we should mention that neural nets are far away from being a unique and reproducible method like logistic regression. Our investigations demonstrate the importance to choose the value of the decay parameter and the number of hidden units appropriately. As we have usually no a-priori information for this, we need automatic selection procedures to get an objective method. Here techniques similar to variable and model selection procedures can be used, but these are still under development (e.g. Murata et al. 1993) and we have to expect that they will suffer from the same difficulties as their statistical analogues. Moreover, there exists no consensus with respect to an algorithm to solve the minimization task, and due to the existence of local minima different algorithms can produce different results.

Summarizing, we believe that for most settings of biomedical research, where logistic regression has become a standard tool of statistical analysis, the use of neurals net does not necessarily imply any progress: They fail in translating their increased flexibility into an improved estimation of the regression function due to insufficient sample sizes, they do not give direct insight to the influence of single covariates, and they are lacking uniqueness and reproducibility.

References

Barlow, R.E., Bartholomew, D.J., Bremner, J.M., Brunk, H.D. (1972): *Statistical inference under order restrictions*, New York: Wiley.

Barron, A.R. (1993): *Universal approximation bounds for superpositions of a sigmoidal function.* IEEE Transactions on Information Theory IT-39, 930-945.

Byar, D.P. (1985): *Assessing apparent treatment-covariates interactions in randomized clinical trials.* Statistics in Medicine 4, 225-263.

Cheeseman, P., Oldford, R.W. (1994): *Selected models from data. Artificial Intelligence and Statistics IV*: Lecture Notes in Statistics 89.

Cybenko, G. (1989): *Approximation by superposition of a sigmoidal function.* Mathematics of Controls, Signals and Systems 2, 303-314.

Funahashi, K. (1989): *On the approximate realization of continuous mappings by neural networks.* Neural Networks 2, 183-192.

Gail, M., Simon, R. (1985): *Testing for qualitative interactions between treatment effects and patient subsets.* Biometrics 41, 362-372.

Geman, S., Bienenstock, E., Doursat, R. (1992): *Neural networks and the bias/variance dilemma.* Neural Computation 4, 1-58.

Hastie, T.J., Tibshirani, R.J. (1990): *Generalized additive models*, London: Chapman and Hall.

Hornik, K., Stinchcombe, M., White, H. (1989): *Multilayer feedforward networks are universal approximators.* Neural Networks 2, 359-366.

Murata, N., Yoshizawa, S., Amari, S.-I. (1993): *Network information criterion for determining the number of hidden units for artificial neural network models*: IEEE Transactions on Neural Networks, to appear.

Ripley, B.D. (1993): *Statistical aspects of neural networks.* In: Barnsdorff-Nielson, O.E., Cox, D.R., Jensen, J.L., Kendall, W.S. (eds.), *Chaos and networks - statistical and probabilistic aspects.* London: Chapman and

Hall.
Ripley, B.D. (1994): *Neural networks and related methods for classification.* Journal of the Royal Statistical Society B, 56, 409-456.
Sauerbrei, W. (1993): *Comparison of variable selection procedures in regression models - a simulation study and practical examples.* In: Michaelis, J., Hommel, G., Wellek, S. (eds.), *Europäische Perspektiven der Medizinischen Informatik, Biometrie and Epidemiologie*: München. MMV Medienverlag: 108-113.
Sen, A., Srivastava, M. (1990): *Regression analysis: Theory, methods and applications*, New York: Springer.
Weisberg, S. (1985): *Applied linear regression*, Second Edition, New York: Wiley.
Wolfram, S. (1991): *A system for doing mathematics by computer*, Second edition, Redwood City: Addison Wesley.