Performance Evaluation of Feedforward Networks Using Computational Methods

Stefan M. Rüger   Arnfried Ossen
Informatik, Sekr. FR 5-9
Technische Universität Berlin
10 587 Berlin, Germany
Submitted to NEURAP 1995

Abstract. We will demonstrate that the performance evaluation of feedforward neural networks using computational methods may result in overly pessimistic estimates of the prediction error. In fact, they capture unwanted variability in the distribution of weights, introduced by local maxima in the likelihood function in connection with deficiencies of gradient based learning procedures. An analysis of the influence of local maxima is hampered due to a nontrivial algebraic structure of the weight space; we will show that typical feedforward networks exhibit a large number of symmetries due to a nontrivial symmetry group acting on the weight space. We will present an algorithm which divides out these symmetries. In the resulting much smaller effective weight space, clustering algorithms may be used to improve the assessment of prediction errors. We will demonstrate that this method can be successfully applied.

1 Introduction

Feedforward networks can be interpreted as a form of nonlinear regression. They offer great flexibility at the price of a complicated structure. In general, there are several approximate methods to estimate the standard error of feedforward networks.

The development of ever more complicated statistical learning procedures and powerful computers has led to computational approaches to statistical inference like jackknife, cross-validation and bootstrap [2]. Recent studies suggest that computational methods perform best, "partly because they capture variability due to the choice of starting weights." [6]

However, we will argue that their naïve use can be misleading.

2 Weights and Symmetries

Most learning algorithms for feedforward networks try to minimize a certain cost function, which depends on a weight vector. In a statistical context (see section 3.1), equivalently, the likelihood of sample data of an input-output relation is maximized with respect to the weight vector. Many algorithms inevitably find local maxima overlooking a global solution. Although this might result in acceptable network functions, mixing the results of network functions arising from different local maxima (e. g. to estimate the prognosis error) might lead to wrong insights. It is highly desirable to separate the influences of different maxima.

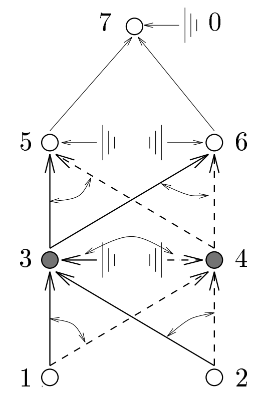
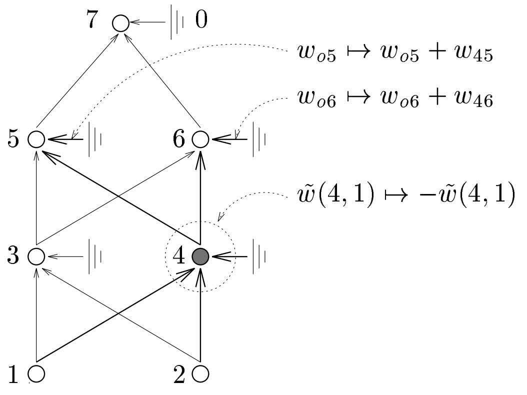


Figure 1: Symmetry of the Network Function out_w Induced by a Certain Weight Permutation.

Typical neural feedforward networks have intrinsic redundancies. Consider the network of Figure 1: due to the commutativity of the summation at nodes 5 and 6, node 3 might be exchanged with node 4 (together with their respective weights) without changing the network function out_w given by the weight vector w. If a hidden layer has n nodes, then $n!$ different weight vectors produce the same network function. Another type of symmetry can be induced by a symmetry of the activation function f of a hidden node. If, as is the case for the logistic function, $f(x) = 1 - f(-x)$, then a sign change of all weights involving the hidden node can be corrected by incrementing all bias weights of the next layer by a corresponding weight; see Figure 2 for details.



$w_{o5} \mapsto w_{o5} + w_{45}$
$w_{o6} \mapsto w_{o6} + w_{46}$
$\tilde{w}(4,1) \mapsto -\tilde{w}(4,1)$

Figure 2: Symmetry of the Network Function out_w Induced by a Symmetry of the Activation Function.

A classification of all local maxima of the likelihood is beyond all hope if the symmetries are ignored. Assume a standard feedforward network with k hidden layers $L_1, \ldots, L_k$, where each layer is fully connected to the next layer. Let the activation function of each hidden node exhibit the symmetry $f(x) = 1 - f(-x)$. In order to classify all local maxima found so far, we propose the following algorithm, which will be justified in the appendix:

Let $\hat{w}$ denote a weight vector resulting from some learning algorithm (or parameter estimation). Beginning with layer L_1 apply the symmetry operation of Figure 2, whenever a bias weight of a hidden node is negative until all hidden nodes have nonnegative bias weights. Then for every layer apply symmetry operations of Figure 1 such that the bias weights of each layer are in a definite order, say ascending from left to right, thus arriving at a "sorted" weight vector $s(\hat{w})$. The vector $s(\hat{w})$ produces the same network function as $\hat{w}$. Different results $\hat{w}^1, \hat{w}^2, \ldots$ for weight parameter estimates may result from different starting points for the learning algorithm or different data samples. Create effective weight vectors $s(\hat{w}^i)$ by the above procedure. Then use a clustering algorithm on $s(\hat{w}^1)$, $s(\hat{w}^2)$, ... to classify all weight vectors.

Within this paper, maxima $\hat{w}^1$ and $\hat{w}^2$ of the likelihood for which $s(\hat{w}^1) = s(\hat{w}^2)$ are termed *symmetric to each other.* Weight vectors, which cannot be identified by such a operation are termed *asymmetric.*

3 Standard Error Estimates for Neural Networks

A commonly used indicator of statistical accuracy of a learning procedure is its standard error, that is, the square root of the variance of the predicted values.

We will compare different estimates of standard error of predicted values in a simple class of feedforward neural networks. To illustrate the respective performance, we will use a network with one input unit, one linear output unit, and three sigmoidal hidden units.

$$\mathrm{out}_w(x) = w_{o5} + \sum_{h=2}^{4} w_{h5} f(w_{oh} + w_{1h}x)$$

where $f(x) = 1/(1 + \exp(-x))$.

The data are generated using a "true" model (its graph is shown in Figure 3) with a certain fixed weight vector $w^\star$. In the regression context, we assume "measurement" errors, so the actual data are modeled using:

$$y_i = \mathrm{out}_{w^\star}(x_i) + \varepsilon_i$$

where the errors ε_i are independent and identically $N(0, \sigma^2 = 0.1^2)$ normal distributed.

The training multiset D is $\{(x_1, y_1), \ldots, (x_{33}, y_{33})\}$ (see Figure 3).

3.1 Likelihood Approach

The classic approach of standard error estimation follows likelihood theory [6]. The log-likelihood function for $n = |D|$ data pairs is

$$l_D(w) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mathrm{out}_w(x_i))^2 - \frac{n}{2} \log(2\pi\sigma^2).$$

The result of learning or estimation, e. g., using gradient ascent, is given by

$$\hat{w} = \operatorname*{argmax}_w (l_D(w)).$$

Then, the observed information matrix $\hat{I}$ is the negative Hessian matrix of the log-likelihood with respect to the weights evaluated at $\hat{w}$. Using the delta method [2], we obtain an approximation of the standard error estimation $\widehat{\mathrm{se}}(\mathrm{out}_{\hat{w}}(x))$ by

$$\left(\nabla_w \mathrm{out}(x)|_{\hat{w}}^T \cdot \hat{I}^{-1} \cdot \nabla_w \mathrm{out}(x)|_{\hat{w}} \right)^{1/2}.$$

There are efficient algorithms, which exploit the feedforward structure of the network, to compute the Hessian and the inverse Hessian of the log-likelihood[1, 3].

3.2 Bootstrap Approach

The bootstrap method [2] is based on re-estimations of the parameter vector on B bootstrap samples of the training set. The bth bootstrap sample is a random multiset $D^{*b} = \{(x_1^{*b}, y_1^{*b}), \ldots, (x_n^{*b}, y_n^{*b})\}$ drawn from the training data *with replacement*, i. e., some of the original data pairs will not appear, and some will appear multiply. A regression line is obtained by averaging

$$\overline{\mathrm{out}}^B(x) := \frac{1}{B} \sum_{b=1}^{B} \mathrm{out}_{\hat{w}^{*b}}(x)$$

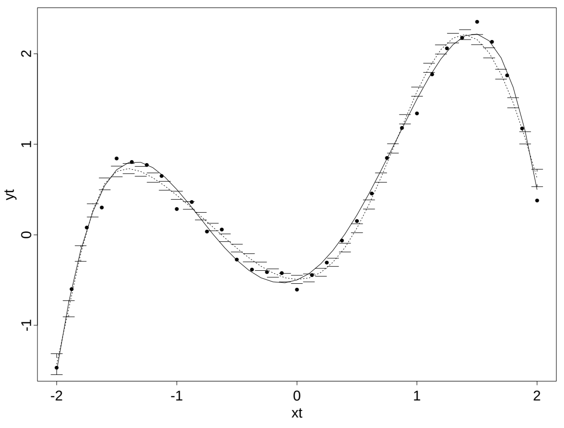


Figure 3: True Model (Solid Line), Data, Maximum-Likelihood Regression Line (Dotted) and Standard Errors

where $\hat{w}^{*b}$ is the estimated weight vector according to $\hat{w}^{*b} = \underset{w}{\operatorname{argmax}}(l_{D^{*b}}(w))$.

Then the standard error is approximately

$$\Big(\frac{1}{B-1}\sum_{b=1}^{B}\big(\text{out}_{\hat{w}^{*b}}(x) - \overline{\text{out}}^{B}(x)\big)^2\Big)^{1/2}.$$

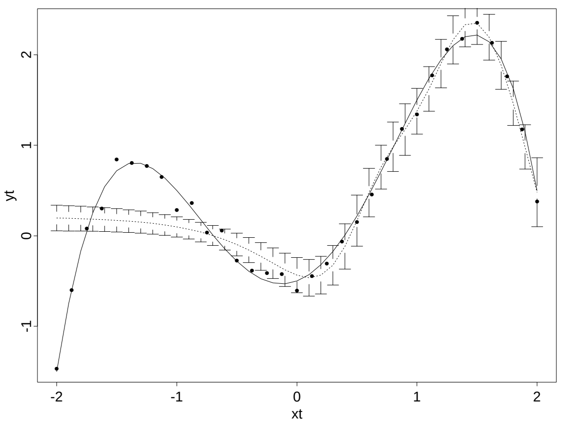


Figure 4: True Model (Solid Line), Data, Maximum-Likelihood Regression Line at Weaker Local Maximum (Dotted) and Standard Errors

4 Simulations

In this section, we will demonstrate that bootstrap and likelihood approaches to standard error estimation yield quite different results. Figures 3 and 4 show two estimates based on the likelihood approach. The respective regression lines (dotted) are also quite different and correspond to asymmetric local maxima in the likelihood. Note that given a gradient based learning procedure the local maximum will be reached a significant number of times even if the number of available data points will become large. Asymptotic normality does not seem to help to close the gap between the likelihood and bootstrap approaches.

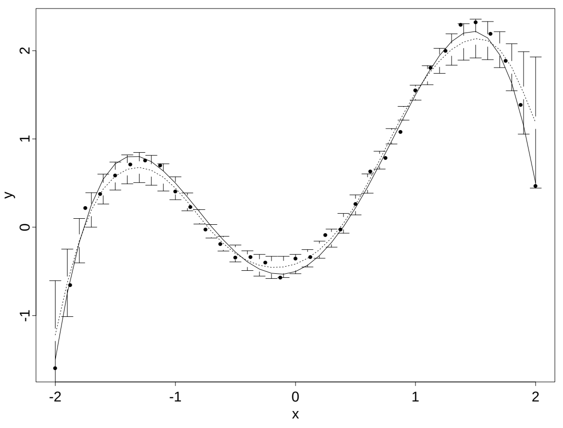


Figure 5: True Model (Solid Line), Bootstrap Regression Line (Dotted), Data and Standard Errors

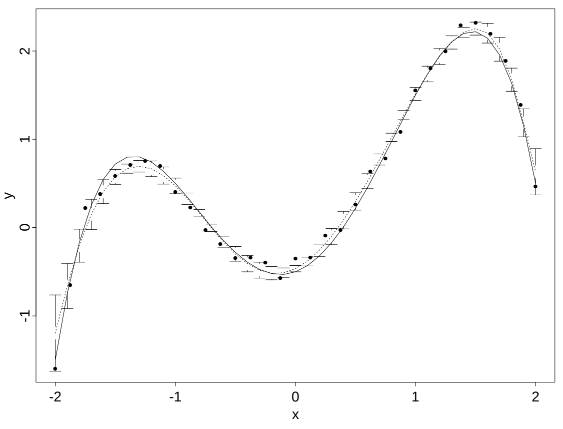


Figure 6: True Model (Solid Line), "Clustered" Bootstrap Regression Line (Dotted), Data and Standard Errors

Figure 5 shows the average bootstrap regression line and — in comparison to the likelihood-based estimates — rather large standard errors. It is obvious that bootstrap has to average over different local maxima resulting in large standard errors.

In Figure 6 the bootstrap distribution of weights is restricted to one maximum of the likelihood: The estimated weight vectors were transformed in order to represent their orbit under the symmetry group S as

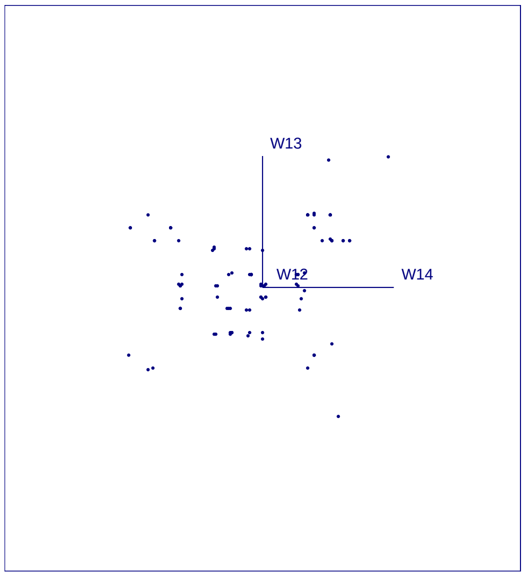


Figure 7: A Projection of a Sample Weight Vector Distribution after Maximizing the Log-Likelihood of the Data With Different Starting Values

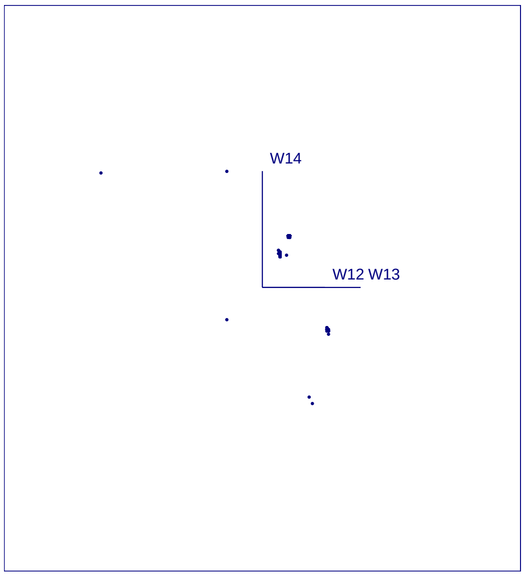


Figure 8: ... and Corresponding Representatives of their Orbits under S.

discussed in section 2 (visualized in Figures 7 and 8 for a sample weight distribution). A simple k-means cluster algorithm was applied to those representative weight vectors. A bootstrap average over weight vectors from a cluster with "good" likelihood was then applied in order to achieve the results of Figure 6. The regression line and the distribution of standard errors is much closer to the likelihood-based standard errors than the original bootstrap estimate.

5 Discussion

Local maxima in the likelihood function have a negative effect on the accuracy of estimated standard errors of predicted values. Not only could this lead to a misjudgement of the models, but also to the failure of procedures based on the precise assessment of statistical accuracy, e. g., dynamic pattern selection algorithms. We have shown that typical feedforward networks exhibit a large number of symmetries due to a nontrivial symmetry group in weight space. In order to improve the accuracy of estimation these symmetries have to be divided out, e. g., by using the algorithm given at the end of section 2. The assessment can be significantly improved if standard error estimation is confined to local regions of the weight space. We have given an example using k-means clustering.

Appendix: Symmetry Group of the Weight Space

The symmetries of the network function, which arise in section 2, are best described and analyzed in terms of their corresponding groups, see e. g. [5]. Let $\Sigma(M)$ denote the permutation group of a set M. Recall that every permutation can be written in terms of transpositions. The transposition $\tau(a, i)$ of a node $a \in L_i$ with its right neighbor node induces a certain permutation $\pi(a, i)$ of the weight vector components, which leaves the network function out_w invariant. This permutation $\pi(a, i)$ may be viewed as a linear operation on the weight space $\mathbb{R}^E$, and thus is an element of the group $\mathrm{GL}(E, \mathbb{R})$ of all linear functions $\mathbb{R}^E \to \mathbb{R}^E$.

Fix i; the mappings $\tau(a, i) \mapsto \pi(a, i)$ with varying a induce a monomorphism (i. e., injective homomorphism) from $\Sigma(L_i)$ to $\mathrm{GL}(E, \mathbb{R})$. Let Π_i denote the image of this monomorphism. Π_i can be identified with the group generated by $\pi(a_1, i), \pi(a_2, i), \ldots$ Further studies show that Π_i commutes with Π_j element by element if $i \neq j$ (check this in Figure 1 for the exchange of first node 5 with 6 and then node 3 with 4 and vice versa). It turns out that the group Π which is generated by $\Pi_1, \ldots, \Pi_k$ is isomorphic to the direct product of $\Pi_1, \ldots, \Pi_k$. Particularly, Π has $|L_1|! \cdot \ldots \cdot |L_k|!$ elements.

Let $t_a : \mathbb{R}^E \to \mathbb{R}^E$ denote the linear operation of Figure 2, which leaves out_w invariant by exploiting the symmetry $f(x) = 1 - f(-x)$ at hidden node a. Note that t_a is idempotent and thus the group induced by t_a is the cyclic group $T_a := \{1, t_a\}$ with two elements. Verify that t_a commutes with t_b for all hidden nodes a, b and that no t_a can be expressed by any combination $t_{b_1} \ldots t_{b_n}$ of other operations when all $b_i \neq a$. From this it can be deduced that the subgroup T which is generated by all the operations t_a is abelian and isomorphic to the direct product of all T_a. Particularly, T has $2^{|L_1 \cup \ldots \cup L_k|}$ elements.

Let $S \subset \mathrm{GL}(E, \mathbb{R})$ denote the group generated by Π and T. By definition, T is a subgroup of S. T turns out to be normal, yielding the result $S = T\Pi = \Pi T$. It follows that each element s of the symmetry group S has a *unique* representation $s = \pi_k \dots \pi_1 t$ with $\pi_i \in \Pi_i$ and $t \in T$. S acts on $\mathbb{R}^E$ in a natural way: Distinct orbits $S(w) := \{x \in \mathbb{R}^E \mid x = s(w) \text{ for some } s \in S\}$ (with respect to the natural group action) partition $\mathbb{R}^E$.

There is an interesting *convex* set W of weight vectors which contains a representative of each orbit. Let $a^i_1, \dots, a^i_{|L_i|}$ denote the nodes of hidden layer L_i. Let $\tilde{w}(b,i)$ denote a subvector of w, containing all weights which involve the node $b \in L_i$. $\tilde{w}(4,1)$ of Figure 2 might be constructed as $(w_{o4}, w_{14}, w_{24}, w_{45}, w_{46})$. The lexicographic comparison $\tilde{w}(b,i) \preceq \tilde{w}(c,i)$ of two vectors, as usual, means that either $\tilde{w}(b,i) = \tilde{w}(c,i)$ or the first nonzero component of $\tilde{w}(c,i) - \tilde{w}(b,i)$ is positive. Then

$$W := \Big\{ w \in \mathbb{R}^E \mid 0 \preceq \tilde{w}(a^i_1, i) \preceq \dots \preceq \tilde{w}(a^i_{|L_i|}, i) \text{ for all } 1 \le i \le k \Big\}$$

is such a set. Note that W is a cone with apex 0 (i. e., $cw \in W$ for all $w \in W$ and $c > 0$). It suffices to look for a maximum of the likelihood in W instead of the much larger space $\mathbb{R}^E$, since $S(W) = \mathbb{R}^E$. Indeed applying T on W allows each first nonzero component of all subvectors $\tilde{w}(b,i)$ to change its sign and then applying Π creates all possible orders of the first nonzero components thus removing any restriction given to specify W. There have been attempts to study the algebraic structure of weight spaces [4]; our analysis is more general and the above search set W is *much* smaller than the one given in [4]. In a certain sense W represents a possible way to define a minimal search set with respect to S.

The space $\mathbb{R}^E$ of weight vectors is highly redundant. Ideally $\mathbb{R}^E$ should be replaced by the space $\mathbb{R}^E/S$ of distinct orbits, but then the problem arises of how to define learning algorithms or statistics in the manifold[1] $\mathbb{R}^E/S$, which has completely different geometric properties than $\mathbb{R}^E$. Instead of bothering with these questions, we propose to learn in the flat space $\mathbb{R}^E$ and map the learning result to W.

Those, who carefully read this subsection, might wish to replace "bias weight of node b" with "first nonzero component of the subvector $\tilde{w}(b,i)$" in the algorithm proposed in section 2. They would implement a minimal search set W. The algorithm proposed works with a slightly larger search set $W' \supset W$. The difference to the minimal one has Lebesgue measure zero.

[1]Strictly speaking, points w with a vanishing subvector $\tilde{w}(a,i)$, or with two coinciding subvectors of nodes in the same hidden layer must be removed in order to get a manifold; they represent singular points.

References

[1] Wray L. Buntine and Andreas S. Weigend. Calculating second derivatives on feed-forward networks. *IEEE Transactions on Neural Networks*, 1991.

[2] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability.* Chapman & Hall, 1993.

[3] Babak Hassibi and David G. Stork. Second order derivatives for network pruning. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems V.* Morgan Kaufmann, 1993.

[4] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In R. Eckmiller, editor, *Advanced Neural Computers*, Amsterdam, 1990. Elsevier.

[5] Ian D. Macdonald. *The theory of groups.* Oxford University Press, 1975.

[6] Robert Tibshirani. A comparison of some error estimates for neural networks. Technical report, University of Toronto, Dep. of Statistics, April 1994.