
*Statistical Models and Artificial Neural Networks:
Supervised Classification and Prediction Via Soft
Trees*

Antonio Ciampi and Yves Lechevallier

*McGill University, Montreal, QC, Canada
INRIA-Rocquencourt, Le Chesnay, France*

Abstract: It is well known that any statistical model for supervised or unsupervised classification can be realized as a neural network. This discussion is devoted to supervised classification and therefore the essential framework is the family of feed-forward nets.

Ciampi and Lechevallier have studied 2- and 3-hidden layer feed-forward neural nets that are equivalent to trees, characterized by neurons with “hard” thresholds. Softening the thresholds has led to more general models. Also, neural nets which realize additive models have been studied, as well as networks of networks which represent a “mixed” classifier (predictor) consisting of a tree component and an additive component. Various “dependent” variables have been studied, including the case of censored survival times.

A new development has recently been proposed: the soft tree. A soft tree can be represented as a particular type of hierarchy of experts. This representation can be shown to be equivalent to that of Ciampi and Lechevallier. However, it leads to an appealing interpretation, to other possible generalizations and to a new approach to training. Soft trees for classification and prediction of a continuous variable will be presented. Comparisons between conventional trees (trees with hard-thresholds) and soft trees will be discussed and it will be shown that the soft trees achieve better predictions than the hard tree.

Keywords and phrases: Prediction trees, probabilistic nodes, hierarchy of experts

16.1 Introduction

The task of learning from data has been the object of two independent but converging traditions: machine learning, which emphasizes algorithmic approaches, and statistical modeling, which emphasizes choice of a model for the probability distribution of the observed data. From the point of view of statistical learning theory, the well-known distinction between supervised and unsupervised learning corresponds to the problems of conditional and unconditional density estimation, respectively.

Traditional parametric modelling uses data to search for an optimal value of a parameter that varies within a space of specified dimension. Complex data, such as those encountered in contemporary data analysis can seldom be fully studied by this traditional approach. Statistical learning theory [Hastie, Tibshirani and Friedman (2001)] aims to extract information from complex data by a new, flexible approach known as adaptive modeling. Typically, dictionaries—i.e., “super-families” of models of unspecified dimension—replace parametric models. And model searches are conducted within dictionaries relying on various heuristics, often inspired by simple cognitive strategies. Although the goal of statistical learning remains to be the modeling of a probability distribution, the approach becomes increasingly algorithmic. Also, the goal of finding a global optimum is often reduced to the search for a sub-optimal but meaningful solution such as a reasonably stable local optimum.

In this chapter, we are concerned with supervised learning and, in particular, with the prediction problem. The aim is to construct from data the regression function, $E[y|x]$, which is the expectation of the variable to predict y given the vector of predictors x . Two common examples of regression functions are *ordinary linear regression*, in which y is a continuous variable and $E[y|x]$ is assumed linear in the x ; and *generalized linear regression*, in which linearity is assumed for a monotone transformation of $E[y|x]$, as in logistic and Poisson regressions. In the statistical modeling tradition, variable selection algorithms for multivariable regression may be considered as the ancestors of adaptive modeling in supervised learning.

Much of contemporary research in statistical supervised learning focuses on approaches shared with machine learning, such as trees and, to a lesser extent, feed-forward Artificial Neural Networks (ANN). Neural networks appear particularly attractive because they are universal approximators [Hornik, Stinchcombe and White (1989)]. However, this very desirable property is of little use in practice. Indeed, before training an ANN, one needs to choose its architecture. This is not done by use of mathematical results, but rather by trial and error and reliance on past experience. This lack of mathematical definition may well be one of the main reasons why ANNs have been to date less

attractive to statisticians. Indeed, statisticians have directed their attention to the sampling properties of ANNs and to some training algorithms for simple architectures [Venable and Ripley (2002)], but not so much to the design of innovative architectures.

On the other hand, owing to the universal approximation property, any statistical model can be realized as an ANN. Therefore, one can use a statistical model that produces reasonably good predictions as the basis on which to build the architecture of an ANN. Once trained, this ANN should produce predictions at least as good as the original statistical model. In the last few years, we have pursued this theme in the context of supervised learning [Ciampi and Lechevallier (1995a,b, 1997, 2000, 2001), Ciampi and Zhang (2002), and Ciampi, Couturier and Li (2002)]. In this chapter, we present some of the results obtained in the endeavour. In Section 16.2, we state more formally the correspondence between statistical models and ANNs. We also demonstrate the realization of linear, generalized linear, and tree-shaped prediction models as specific ANN architecture. Furthermore, we show how these specific architectures can be made more flexible and to produce indeed ANNs which should be at least as good as the original ones. In Section 16.3, we introduce a simple approach to the design of an ANN which has statistical models as building blocks, and explore its advantages and shortcomings. A subtler approach to combining models, developed in Section 16.4, is based on the notion of hierarchy of experts. In Section 16.5, we propose a new architecture, the soft tree, which can be seen as a special case of hierarchy of experts, but one that can be constructed directly from data. This constructive approach is extended in Section 16.5 to a more general hierarchy of experts. Section 16.6 contains a few concluding remarks.

16.2 The Prediction Problem: Statistical Models and ANN's

For the purpose of this work, the prediction problem can be formulated as follows. Suppose we have obtained a data matrix $D = [Y|X]$, whose columns represent measurements on n randomly sampled units of the variable y and of the vector of predictors x . Assume a statistical model for the conditional distribution of y given x as

$$y|x \sim f(y; \theta(x), \phi),$$

where θ is a parameter which can be influenced by x , while ϕ is a possibly infinite dimensional parameter which does not depend on x and may be considered as a

nuisance parameter. Often, we have $\theta(x) = \mu(x)$, the expected value of y given x . The task is to estimate from D a functional form for $\theta(x)$, $\hat{\theta}(x)$.

The ANN modelling approach to the prediction problem is summarized in Figure 16.1. The “input” data vector x enters the ANN through the input layer and then flows through the inner layers towards the output layer. Each unit or (artificial neuron) transforms an input into an output through a specified function, known as the *activation function* of the neuron. Neurons are disposed in layers. Usually the neurons of a layer share the same activation function. The first layer is the input layer; its neurons receive as input one of the components of x and the output repeats the input (identity activation function). Each neuron in the inner layers is connected to some or all of those in the immediately preceding layer; the lines in the figure represent the connections. The input of each inner layer neuron is a linear combination of the outputs of the neurons directly connected to it. The coefficients of the linear combination are called *connection weights* and have to be determined from the data. They can be considered as constituting a high-dimensional vector parameter \mathbf{w} . It should be noted here that, from the point of view of the classical statistical modeller, trained to strive for parsimony of parameterization and interpretability of parameters, the role of \mathbf{w} as “parameter” is highly unusual: the ANN does not appear parsimonious, but rather over-parameterized. The activation function of the inner layers’ neurons is usually a sigmoid function, e.g., the logistic function. At the output of the outer layer neurons (in Figure 16.1, there is only one), one reads the output $out(x|\mathbf{w})$, which represents the prediction. The activation function of the neurons of the outer layer depends on the specific problem

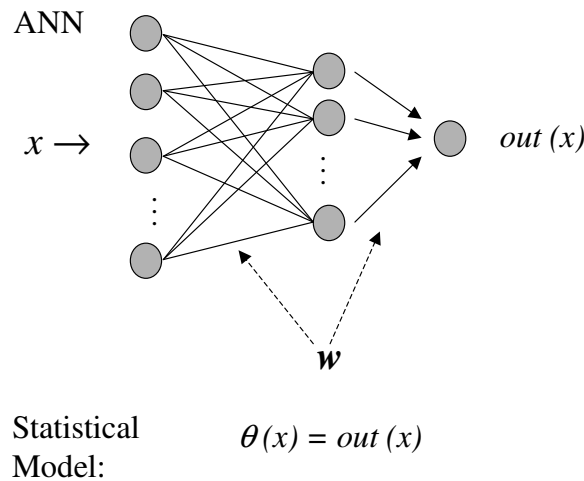


Figure 16.1: ANN and prediction models

In the training phase, connection weights are determined from data. In general, the optimal weights vector parameter $\hat{\mathbf{w}}$ is determined by maximizing a “cost function” $c(Y, out(X|\mathbf{w}))$. Back-propagation is the most popular maximization algorithm: it has an intuitive appeal in terms of learning theory, but it is by no means the only way to approach the maximization problem.

It is natural to identify $out(x)$ with $\theta(x)$, and, when training is completed, $out(x|\hat{\mathbf{w}})$ with $\hat{\theta}(x)$. While the choice of the cost function can be dictated by a number of pragmatic considerations, a statistically natural one is to identify the cost function with the deviance, i.e., the negative of twice the log-likelihood of the data. It is clear that with these choices, the neural network family is simply a highly flexible model family and ANN training is just one approach to likelihood maximization.

We shall now see how some familiar statistical models can be represented as ANNs.

16.2.1 Generalized linear models as ANNs

Consider the generalized linear model family:

$$f(y|x) = f(y|\theta(x)), \quad (16.1)$$

$$\eta(\mu(x)) = \eta(\theta(x)) = \beta x, \quad (16.2)$$

where η is the link function and f is a density of the exponential family [McCullagh and Nelder (1989)]. For ease of notation, we do not explicitly indicate the dispersion parameter, since it does not play a role in the estimation of the regression coefficients.

Several useful regression models are included in this general definition: one simply has to specify the link and the density f . For example, the normal linear regression model corresponds to the choice of the identity link and of a normal density. Two other examples, particularly useful in health statistics, are the logistic regression model, with the logit link function and the binomial distribution, and the Poisson regression model, with the logarithmic link and the Poisson distribution. The Poisson regression model can be easily adapted to treat censored survival data when the survival time is assumed to be exponentially distributed.

Figure 16.2 represents the generalized linear model as an ANN.

The architecture consists of one inner layer, with identity activation function for its neurons; the weights of the connections from the input to the inner layer are fixed and equal, e.g., all equal to 1; and the output neuron has activation function equal to the inverse of the link function. Clearly, the only weights to learn are those of the connections between the inner layer and the output neuron, so that \mathbf{w} is identified with β , the vector of the regression coefficients.

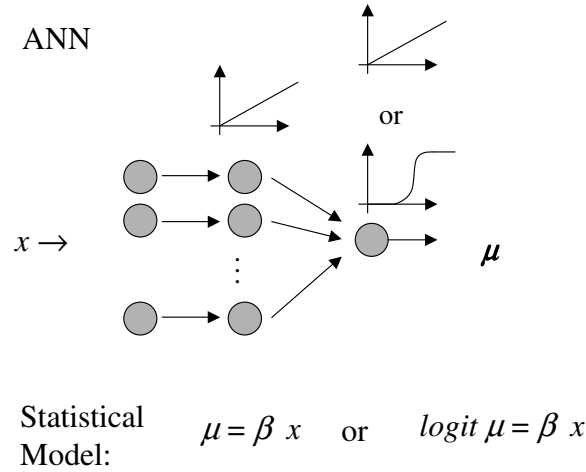


Figure 16.2: Linear predictor

In particular, for ordinary linear regression we have

$$\text{out}(x|\beta) = \theta(x) = \mu(x) = \beta x, \quad (16.3)$$

and for logistic regression we have

$$\text{out}(x|\beta) = \theta(x) = \mu(x) = p(x) = \frac{e^{\beta x}}{1 + e^{\beta x}}. \quad (16.4)$$

A likelihood-based cost function is then

$$c(Y, \text{out}(X, \beta)) = -2 \log L = -2 \sum_{i=1}^n \log f(y_i | \text{out}(x_i | \beta)). \quad (16.5)$$

Thus it can be seen that from an ANN perspective, the choice of the density function and of the link function correspond to the choice of the cost function and of the activation function of the output neuron, respectively.

As long as we wish to limit ourselves to fitting generalized linear models, the correspondence we have outlined amounts to nothing more than a formal remark; in particular, back propagation is not especially useful in this situation, as there are simple and powerful algorithms to learn the parameters, which require no iterations for the linear model and generally few iterations for the generalized linear models. However, suppose we want to allow little deviations from linearity. Then we can use logistic thresholds in the inner layer, allow variable connection weights between the input and the inner layer, but choose an initialization of these weights so that the logistic activation functions behave virtually as identity functions at the first step of the training. Also, at initialization, the weights of the connections of the inner layer with the output layer can

be taken as the regression coefficients of the regression estimated by the appropriate statistical procedure. During training, these initial weights may change very little or not at all, in which case the linear model is consistent with the data, or they may change substantially, indicating that the linear assumption is not valid. Therefore, the data will decide whether a deviation from the linear model is warranted. We stress, however, that if such deviation is warranted, it will be of a particular nature: the final model will be linear in some logistic transformation of the original predictors. Therefore, it will still be “close” to a linear model, in the sense that it will be a simple case of additive model.

Another way to enlarge the generalized linear model ANN (GLM-ANN) is to add a second hidden layer, initialize the training process so that the new ANN is undistinguishable from the GLM-ANN, and then let the data determine whether the generalization is warranted. In principle, the result can be any ANN with one hidden layer. The “linear” initialization underlies the approach developed in Ciampi and Zhang (2002) for training an ANN. It has a number of advantages on the standard random initialization, at least when the variable to predict is a binary variable. In this case, training is achieved in a shorter time than with random initialization and the predictive accuracy is not inferior to that obtained by the more costly initialization; furthermore, one demonstrably improves the predictive accuracy of the ‘initial’ linear regression model. A similar approach was developed in Ciampi and Lechevallier (2000, 2001) for censored survival data, using the formal equivalence of censored exponential regression with a Poisson model referred to above.

16.2.2 Generalized additive models as ANNs

Generalized additive models are an extension of the generalized linear model that has proved very useful in a variety of applications [Hastie and Tibshirani (1990)]. The linear assumption in Eq. (16.2) is replaced by the additive assumption

$$\eta(\mu(x)) = \theta(x) = \sum g_i(x_i), \quad (16.6)$$

where the g 's are arbitrary continuous functions, to be determined from the data through flexible modelling.

An ANN, as shown in Figure 16.3, can also represent the most general additive model. Here, instead of having an inner layer of neurons, we have an inner layer of ANNs; in other words, the architecture is that of a network of networks.

Each component x_i of the predictor vector x serves as input to a standard ANN whose internal connection weights have to be learned from the data. The output of the i th inner layer ANN is denoted by $g_i(x_i)$; because of the universal approximation property of ANNs, the form of this function is totally flexible. The outputs of the inner layer ANNs flow towards the output of the network

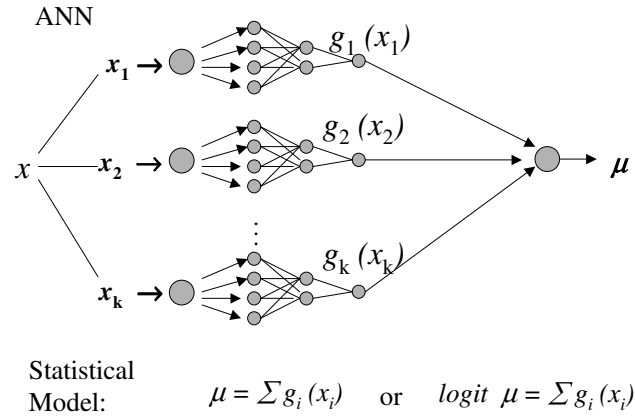


Figure 16.3: Additive predictor

of networks, with fixed, equal connection weights. As for the linear case, the form of the activation function of the output neuron and the choice of the cost function determine the type of additive model. The output functions for ordinary and for logistic additive regression are, respectively,

$$\text{out}(x|\beta) = \theta(x) = \mu(x) = \sum_{i=1}^n \beta x_i \quad (16.7)$$

and

$$\text{out}(x|\beta) = \theta(x) = \mu(x) = p(x) = \frac{e^{\sum g_i(x_i)}}{1 + e^{\sum g_i(x_i)}}. \quad (16.8)$$

The ANN for additive modelling can be enlarged, in a manner similar to that discussed at the end of the previous section. We can, for instance, add another inner layer between the layer of networks and the output layer, initialize it in a way that it is indistinguishable from the additive model ANN and then let the data determine departures from the initial model.

16.2.3 Classification and regression trees as ANNs

Tree-structured prediction is becoming increasingly popular in a vast domain of applications. A prediction tree is a graph as the one shown in Figure 16.4, to which the following statistical model is associated:

$$\mu(x) = \mu_1 I[x_1 > a_1] I[x_1 > a_2] + \mu_2 I[x_1 > a_1] I[x_1 \leq a_2] + \mu_3 I[x_1 \leq a_2]. \quad (16.9)$$

Here $I[x \in A]$ is the characteristic function of the set A , i.e., $I[x \in A](x)$ is 1 if x is in A and 0 otherwise. Also this model has an ANN representation,

which was proposed in Ciampi and Lechevallier (1995b, 1997) and Ciampi and Zhang (2002) and is shown in Figure 16.4.

This ANN has two inner layers, with activation functions $I[x \leq 0]$. The first inner layer has as many neurons as there are inner nodes in the tree: it creates the questions at the nodes, in the sense that each neuron has output 1 or 0 depending on whether or not the question defining the branching has answer *yes* or *no*. The second inner layer has as many neurons as there are leaves and the output of each neuron is 1 or 0 depending on whether the subject with predictor x is assigned or not to the corresponding leaf. The output layer simply realizes equation (16.9).

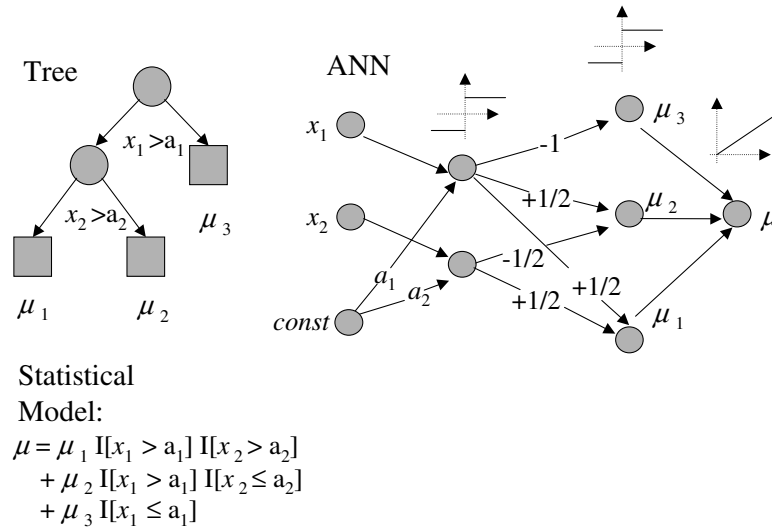


Figure 16.4: Prediction tree and its associated ANN

As in Section 16.2.1, the main interest of the correspondence between tree and ANN is that it can be used to train a neural net that can in fact improve on the tree. The problem with the description of Figure 16.4(b) is that we have ‘hard thresholds’ as activation functions for the inner layer neurons, which makes optimization hard and impossible with techniques based on differentiation of the activation function, like back-propagation. However, by appropriate choice of the initial conditions, one can replace hard thresholds with very steep soft thresholds without changing the numerical value of the cost function at initialization. As training proceeds, the weights may evolve so that the activation functions become quite far from the initial steep threshold, in which case the indication given by the data is that a tree model can be improved upon. Our experience shows that this behaviour is observed in practical applications and that substantial improvement on the tree may result [Ciampi and Lechevallier (1995b, 1997) and Ciampi and Zhang (2002)]. Unfortunately, when thresholds

become soft, the interpretation of the model is not clear. However, it is possible to interpret the output of the neurons of the first inner layer: if we still consider an inner layer neuron as corresponding to a node in the tree, then the output of such a neuron is between 0 and 1; it is close to 1 if the input is large and close to 0 if it is small. It is suggestive to interpret this output as a probabilistic answer to the question defining the node, which could be restated as “is x_1 large?” For instance, if x_1 is very large or very small for a subject, then that subject goes left or right; for “intermediate” values of x_1 , the subject goes to the left with the probability given by the output of the neuron and to the right with complementary probability. We are implicitly assuming that there is a latent binary variable, taking values 1 for “large” and 0 for “small,” for which x_1 is an indicator.

We will see later that this interpretation can be developed more fully but only after we propose an alternative representation of the tree as an ANN.

16.3 Combining Prediction Models: Hierarchy of Experts

We have seen that an additive model can be represented as a network with an inner layer consisting of networks. There is, however, the important restriction that each neuron of the input layer, corresponding to a one-dimensional component of x , is connected to only one of the inner networks. What if we drop this restriction and allow full connectivity between the outer and the inner layer? We proposed this in Ciampi and Lechevallier (2000), and suggested that each inner network might represent a statistical model, e.g., we might have two inner networks, one representing a linear model, the other a tree. The output of the whole network would then appropriately weight the two models and suggest whether a tree or a linear model is more consistent with the data. Again, our experience shows that, when the data warrant it, the ANN based on the combination of two statistical models does better than the individual models themselves and the associated ANNs [Ciampi and Lechevallier (2000)].

While the idea of network of networks has proved useful in the direct form proposed in Ciampi and Lechevallier (2000), we have recently pursued an interesting alternative to the development of a network of networks: we can arrange several networks in a hierarchical structure called *hierarchy of experts* [Jordan and Jacobs (1994)]. Figure 16.5 represents such a structure.

At the core of the graph, there is a tree structure. However, data flow from the leaves to the root of the tree. The leaves of the tree, represented by darkly filled squares, are neural networks, receiving input x . Each node, represented as usual by a circle, is a neuron with sigmoid activation function, receiving

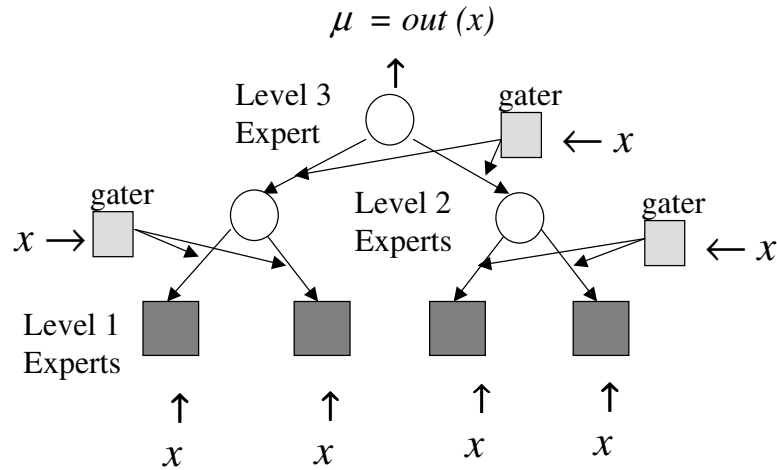


Figure 16.5: Alternative architecture: Hierarchy of experts

input from its children and sending its output towards its parent node. Next to each branch, there is a neural network, represented by a lightly filled box and known as gating network or, more briefly, *gater*. A gater also receives x as input, and outputs a number p between 0 and 1. The role of the gater is to weight the outputs of the children with its own output. Conventionally the left branch receives weight p and the right branch receives weight $1-p$. Then these weighted outputs serve as input to the parent node. The output of the root node is the output of the whole system.

The name of this architecture is explained by the following suggestive interpretation. A leaf of the tree represent a ‘first level’ expert, who looks at the data and makes a prediction. This prediction is examined by a ‘second level’ expert, who also examines the data but only to decide how to weight the predictions of the first level experts. Second level experts submit their own predictions, based on the weighted predictions of their subordinates, to third level experts; and so on, until a ‘super expert’, represented by the root of the tree, makes the final prediction.

The idea of mixing ANNs based on statistical models can easily be realized with the ‘hierarchy of experts’ architecture. For example, one can place classical statistical models at the leaves, and use as gaters networks with constant output: this will be equivalent to the network of networks architecture discussed at the beginning of this section. More generally, using linear or additive ANNs as gaters allows greater flexibility, since models are weighted by weights that may depend on the predictors.

Hierarchy of experts have great flexibility and can be used to mimic a wide variety of situations. One such situation is the following: leaf networks represent “experts” working with different sets of variables: for example, one expert might

make a tentative diagnosis based on standard blood tests, another on genetic tests, yet another on an X-ray examination, and so on. Then the role of a “super expert” is to combine these different points of view and, based on the totality of the information, to weight the partial diagnoses and issue a comprehensive one.

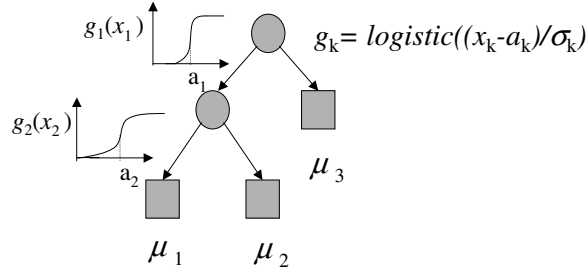
In spite of many impressive features, the hierarchy of experts remains an *architecture*, i.e., a structure which should be designed a priori before using the data for learning the weights of its components. This is a limitation that we are attempting to overcome by developing algorithms that can construct a hierarchy of experts directly from data. The key idea at the root of this development is the soft tree.

16.4 The Soft Tree

In Section 16.2.3, we have discussed an ANN architecture based on a tree; we also pointed out that during training, the resulting predictor may differ quite markedly from the original tree structure and may indeed lose the advantage of interpretability. The hierarchy of the experts offers an alternative approach to the design of a network based on a tree, which remains interpretable even if the data suggest departures from the original tree structure. Indeed, a hard tree can be easily represented as a hierarchy of experts with very specific features. The expert at the leaves should be imagined as making the same prediction for all subjects, regardless of the values of the predictor variables x ; in other words, the outputs of the leaf networks do not depend on the input. Further, each gater should be thought of as a hard threshold on a single variable, characteristic of the corresponding node. In terms of the expert interpretation, the super expert simply chooses one of the predictions of its subordinate experts, and does so on the basis of just one variable.

As in Section 16.2.3, one can replace hard thresholds with soft thresholds, initialize all the gates, so that at the beginning of the learning process they behave as hard thresholds, and then train the networks while allowing departures from the hard thresholds. The resulting structure is a tree with *soft nodes*, or, for brevity, a *soft tree*. It is represented in Figure 16.6.

The process for creating such a structure may be entirely constructive: the data are used a first time to determine the hard tree structure on which to base the architecture of hierarchy of experts, and a second time to soften the hard thresholds, if warranted to improve prediction. It could be easily implemented as an algorithm. However, we have developed a more efficient alternative approach, which determines recursively both the variable at each node and the gater associated to it. The details of the algorithms to predict a binary response



Statistical Model:
$$\mu = \mu_1 g_1[x_1 > a_1] g_2[x_2 > a_2] + \mu_2 g_1[x_1 > a_1] (1 - g_2[x_2 > a_2]) + \mu_3 (1 - g_1[x_1 > a_1])$$

Figure 16.6: Example of soft tree

are discussed in Ciampi and Couturier (2002). In what follows, we outline the main ideas.

16.4.1 General concepts in tree construction

The construction algorithm proposed in Ciampi and Couturier (2002) rests on some intuitive concepts which are simple generalizations of their counterparts for a tree with ordinary hard thresholds, see Hornik, Stinchcombe and White (1989); we will refer to the latter as *hard tree*. Consider first a simple binary split with soft threshold based on the variable x_1 , say: thus a subject goes to the left branch with probability $g(x_1)$ and to the right branch with complementary probability $1 - g(x_1)$. The trivial tree consisting of the root node only is a statistical model corresponding to the hypothesis that the distribution of $y|x$ does not depend on x . The tree with the two nodes of the split is also a statistical model, according to which the distribution of $y|x$ is a mixture of distributions of the form

$$f(y|x) = g(x_1)f_1(y) + (1 - g(x_1))f_2(y). \tag{16.10}$$

We define the *information content* of the split as the likelihood ratio statistic (LRS) for comparing these two models. This definition is easily generalized to a tree of general structure T . In fact, we can construct a LRS for comparing the root node model with the model associated to the leaves of T , i.e., a formula for the conditional density that generalizes (16.10) in a direct way

$$f(y|x) = \sum_{l=1}^L G_l(x) f_l(y) \tag{16.11}$$

where $f_l(y)$ denotes the probability density associated to the l th leaf, the sum is over the L leaves of T , and the coefficient of $f_l(y)$'s is obtained from basic properties of probability, as a product of "soft threshold" functions $g_\ell(x_\ell)$ and of complements of such functions associated to the nodes which are ancestor of the l th leaf. Thus the information content of a tree is

$$IC(T : Root|data) = LRS(T : Root|data). \quad (16.12)$$

Other concepts used in (soft) tree-growing are Information Gain (IG) and Information Loss (IL). If T^* is a rooted subtree of T (a subtree containing the root node), then the IG of T with respect to T^* and the IL of T^* with respect to T are

$$IG(T : T^* |data) = LRS(T : T^* |data) = IC(T|data) - IC(T^* |data)$$

and

$$IL(T^* : T|data) = LRS(T^* : T|data) = IC(T^* |data) - IC(T|data).$$

Given a tree T , we define the best 1-split augmentation of T as the tree obtained from T by adding to one split one leaf such that the augmented tree has the highest information gain with respect to T .

16.4.2 Constructing a soft tree from data

We can now write the general outline of our soft tree construction algorithm:

GENERAL ALGORITHM:

1. FIX admissibility conditions and selection rules (*AIC* or *BIC*)
2. STEP 0: INITIALIZE:
Estimate the parameter for the trivial tree $T_0=Root$
.....
3. STEP k :ENLARGE TREE BY 1 SPLIT
Find best 1-split augmentation of T_k
4. UPDATE: $T_k \leftarrow T_{k+1}$
5. If $IG(T^* : T)$ is small enough
THEN STOP
ELSE, GO TO 3.

In order to perform STEP k , we need to repeatedly estimate the parameters of the soft tree T_k and of all its 1-split augmentations. The parameters of a soft tree describe, on the one hand, the constant leaf predictors; on the other hand,

there are also parameters describing the soft nodes. If we model the gater at a node by a logistic function

$$f_k(x) = \frac{e^{a_k + b_k x}}{1 + e^{a_k + b_k x}},$$

we have 2 parameters to estimate for each node: a_k and b_k . If L is the number of leaves and we have K parameters per leaf, we have a total of $(2 + K)L - 2$ parameters, or $3L - 2$ for $K = 1$, the case of a single continuous or binary response variable.

We estimate the parameters by an EM-type algorithm, described below. Indeed, the actual likelihood of a soft tree of a given structure is rather hard to write down and maximize directly. On the other hand, it is easy to ‘complete’ the data and to write a ‘complete data’ likelihood, based on the following idea. Suppose that there is an unobserved binary variable at each node k , such that $\zeta_k = 1$ means ‘go left’ and $\zeta_k = 0$ means ‘go right’. Then we can complete the observed data by adding hypothetical values of ζ_k so that the soft tree becomes a hard tree: when this is done, the task of writing and maximizing the likelihood is trivial or, at least, reducible to a well-known one.

**EM ALGORITHM FOR ESTIMATING THE PARAMETERS OF
A SOFT TREE**

1. INITIALIZE:

Assign initial values to all parameters $\theta^{(0)}$

2. E-STEP:

Calculate $E[\zeta_k \mid y; \theta^{(r)}]$ and substitute this to ζ_k in the complete likelihood

3. M-STEP:

Maximize complete likelihood to obtain $\theta^{(r+1)}$

4. UPDATE: $\theta^{(r)} \leftarrow \theta^{(r+1)}$

**5. IF update parameters are ‘close enough’ to old parameter,
THEN STOP**

ELSE GO TO 2

While the algorithms of tree construction and parameter estimation are quite general in the form presented here, several technical difficulties had to be overcome in practice. Different types of response variable present different problems. We now have a stable algorithm for the case of binary and continuous responses. An improved version of this approach, which can handle continuous, binary and multinomial responses, is now under development and the results seem promising.

16.4.3 An example of data analysis

We summarize here the results of the construction of a soft tree predictor from a well-known data set in the public domain: the Pima Indians Diabetes data [Venable and Ripley (2002)]. Data are available on 532 subjects of Pima Indian heritage, all females and of age 21 or older, living near Phoenix, Arizona. The goal is to predict a binary variable which indicates presence of Type II diabetes (WHO definition), from the following predictors: Number of pregnancies (npreg), Plasma glucose concentration (Glu), Diastolic blood pressure (bp), Triceps skin fold thickness (skin), Body mass index (BMI), Diabetes pedigree function (Ped), and Age. Figure 16.7 shows a soft tree constructed from these data according to the algorithm outlined in Section 16.4.2.

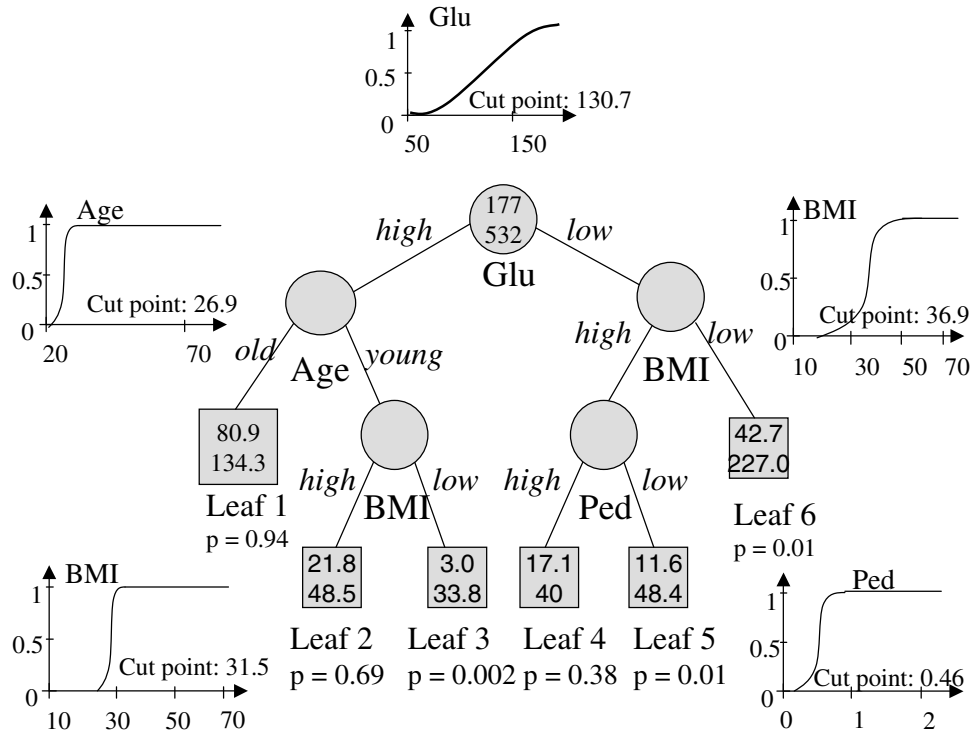


Figure 16.7: Soft tree for Pima Indians data

Notice that the soft nodes vary in degree of ‘softness’, as shown by the gater functions at the nodes: for instance the first node, determined by Plasma glucose concentration is very soft, while the lowest node based on BMI is nearly hard. For purpose of comparison, we also constructed a hard tree from the same data, shown in Figure 16.8.

Is one model clearly better than the other? Model comparison is summarized in Table 16.1, which contains four common cross-validated measures of

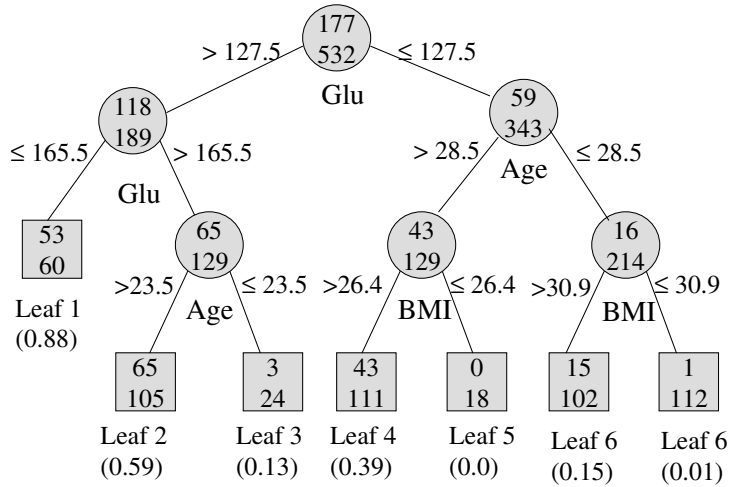


Figure 16.8: ‘Hard’ tree for Pima Indians data

predictive accuracy [Harrell (2002)] for the soft and the hard trees: Deviance, Area under the Receiving Operator Characteristic (ROC) curve, Brier’s Score, and misclassification error. Except for the Brier’s Score, which is the same for the two predictors, the measures reveal a clear advantage of the soft tree.

Table 16.1: Model comparison for Pima Indians data

	Soft tree	Hard tree
Deviance	448.23	453.67
Area under the ROC curve	86.54	85.81
Brier’s score	0.14	0.14
Misclassification error	19.55%	21.05%

16.4.4 An evaluation study

The analysis of the PIMA Indian data is part of an evaluation study presented in Ciampi and Couturier (2002). We chose 6 public domain data sets with a binary response: Breast Cancer, Pima Indian, Heart disease, Liver disease, Diabetes 2, and Prostate Cancer. By the following resampling approach, we compared soft and hard tree on these data.

The results are summarized in Tables 16.2 and 16.3, in which the comparison is based on the classification error. Clearly, as it could be expected, the soft tree performs statistically better than the hard tree.

1. Split at random the data into a learning set and a test set (10% of the data)
 2. Build two predictors, hard and soft tree, on the black *learning set*
 3. On the *test set*, compare the accuracy of the two predictors
- Repeat the random splitting 100 times

Table 16.2: Test set classification error

	Soft tree		Hard tree		<i>p</i> -values
	Mean	Std	Mean	Std	
Breast cancer	3.998	2.499	5.31	2.715	< 0.0001
Pima Indian	22.86	5.58	26.12	5.797	< 0.0001
Heart disease	22.37	7.429	33.73	7.803	< 0.0001
Liver disease	37.74	7.556	50.63	8.634	< 0.0001
Diabetes 2	15.68	5.34	14.62	5.47	0.0007
Prostate cancer	36.92	6.92	39.19	6.65	0.0013

Table 16.3: Proportion of time the hard tree test set classification error (E_H) is greater than the soft node test set classification error (E_S)

	$E_H > E_S$	$E_H \geq E_S$
Breast cancer	58%	84%
Pima Indian	69%	77%
Heart disease	82%	86%
Liver disease	86%	93%
Diabetes 2	3%	78%
Prostate cancer	54%	66%

The results of the comparisons based on other measures of predictive accuracy point in the same direction, and are not shown here.

16.5 Extending the Soft Tree

The soft tree model can be easily extended in order to increase *predictive accuracy* while sacrificing as little as possible of *interpretability* and *parsimony*. The idea is suggested, on the one hand, by the hierarchy of expert paradigm, and on the other by some generalized tree-growing algorithms for trees with hard nodes [Ciampi (1991), Chaudhuri *et al.* (1995), and Chipman, George and McCulloch (2002)]. One can replace the constant predictor at the leaves by a (generalized) linear predictor, i.e., a regression equation. The algorithm outlined in Section 16.4.2 has to be modified by allowing at each split some simple form of stepwise variable selection.

Although we are still in the process of developing stable algorithms, we will present here some new results. We have analyzed another public domain data set, the Car Mileage data [Venable and Ripley (2002)]. These data contain city-cycle fuel consumption in miles per gallon, to be predicted in terms of the following predictors: cylinders, displacement, horsepower, weight, acceleration, model-year. Hard, soft and extended soft trees were constructed from these data. They are shown in Figures 16.9, 16.10, and 16.11.

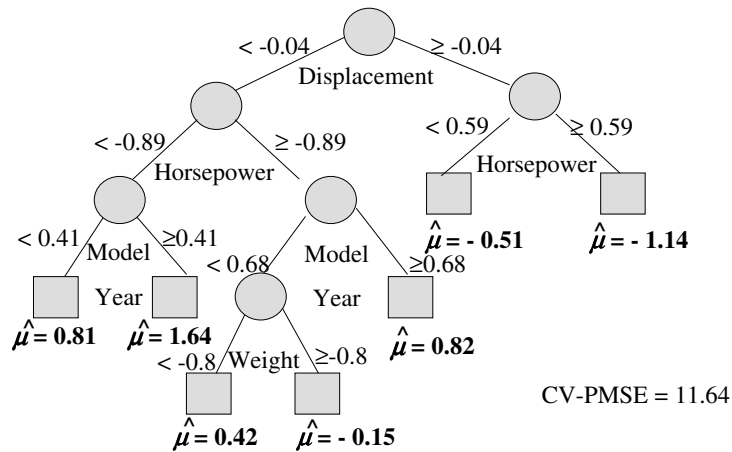


Figure 16.9: “Hard” tree for car mileage data

The cross-validated prediction mean square error was calculated for each of the predictors, obtaining 11.64 for the hard tree, 10.23 for the soft tree and 9.68 for the extended soft tree. Clearly, at least for these data, the extended

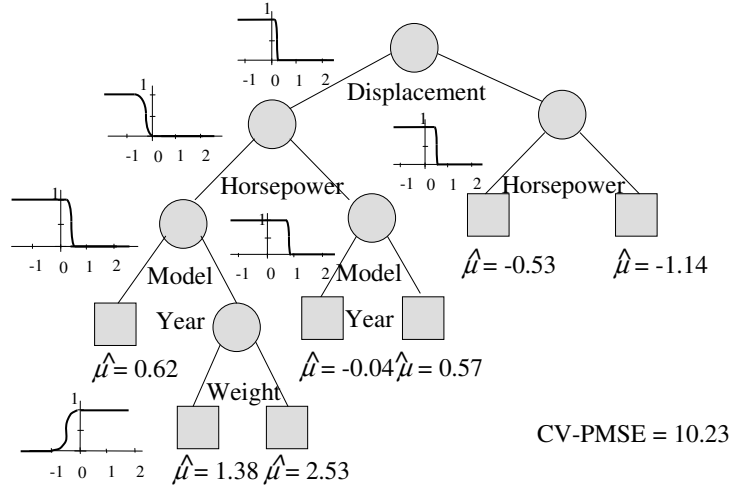


Figure 16.10: Soft tree for car mileage data

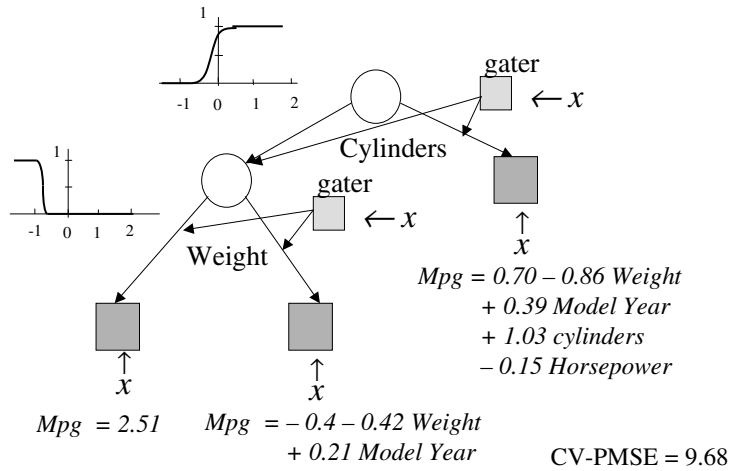


Figure 16.11: Extended soft tree

soft tree appears to be superior and the soft tree better than the hard tree. An empirical evaluation study based on several publicly available data sets is under way.

16.6 Conclusions

We have summarized some work and presented some new results based on the interplay of statistical modeling and Artificial Neural Networks. The simple remark that ANNs and statistical models are *equivalent* opens up new directions of methodological development. On the one hand, statistical models can be used to advantage for designing ANNs and initializing ANN training algorithms. On the other hand, ANN thinking may suggest new statistical modeling approaches. Indeed, new adaptive modeling approaches like the soft tree begin to emerge; they can be seen as true “hybrids” of statistical models and ANNs.

ANNs and statistical models share the same goal, *learning from data*. But this goal is pursued with different emphasis: ANNs privilege *predictive accuracy*, while statistical modeling privileges *interpretability* and *parsimony*. The approach we have been developing aims at striking a compromise between the two principal aims. The soft tree, which we have discussed in some detail, is a primary example: it aims to be *nearly as accurate* as an ANN and *nearly as interpretable/parsimonious* as a statistical model. After a first successful attempt, the basic soft tree model has been extended to increase *predictive accuracy* while sacrificing as little as possible of *interpretability* and *parsimony*. This has been done, as we have shown, by replacing the constant predictor at the leaves of the soft tree by a *regression equation*. We are currently working at refining the present algorithm. Careful evaluation is also in progress. Soft trees for more complex response variables are under development (survival time, longitudinal response, etc.).

The process we have outlined can be extended much further. One possible goal could be the construction of more general hierarchy of experts from data. Gating networks, which are for now of the simplest kind, can also be made a little more complex, while remaining interpretable. For instance, a simple gate could be one that responds to ‘*total number of symptoms or features*’ out of a specified list. This and other ideas will be explored in future research.

References

1. Chaudhuri, P., Lo W.-D., Loh, W.-Y., and Yang, C.-C. (1995). Generalized regression trees, *Statistica Sinica*, **5**, 641–666.
2. Chipman, H., George, E., and McCulloch, R. (2002). Bayesian treed models, *Machine Learning*, **48**, 299–320.
3. Ciampi, A. (1991), Generalized regression trees, *Computational Statistics & Data Analysis*, **12**, 57–78.
4. Ciampi, A., Couturier, A., and Li, S. (2002). Prediction trees with soft nodes for binary outcomes, *Statistics in Medicine*, **21**, 1145–1165.
5. Ciampi, A. and Lechevallier Y. (1995a). Réseaux de neurones et modèles statistiques, *La revue de Modulad*, **15**, 27–46.
6. Ciampi, A., and Lechevallier, Y. (1995b). Designing neural networks from statistical models: A new approach to data exploration, In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, pp. 45–50, AAAI Press, Menlo Park, California.
7. Ciampi, A., and Lechevallier Y. (1997). Statistical models as building blocks of neural networks, *Communication in Statistics*, **26**, 991–1009.
8. Ciampi, A., and Lechevallier, Y. (2000). Constructing artificial neural networks for censored survival data from statistical models, In *Data Analysis, Classification and Related Methods* (Eds., H. Kiers, J.-P. Rassin, P. Groenen, and M. Schader), pp. 223–228, Springer-Verlag, New York.
9. Ciampi, A., and Lechevallier, Y. (2001). Training an artificial neural network predictor from censored survival data, In *10th International Symposium on Applied Stochastic Models and Data Analysis* (Eds., G. Govaerts, J. Janssen, and N. Limnios), Vol. 1, pp. 332–337, Université de Technologie de Compiègne, France.
10. Ciampi, A., and Zhang F. (2002). A new approach to training back-propagation artificial neural networks: empirical evaluation on ten data sets from clinical studies, *Statistics in Medicine*, **21**, 1309–1330.
11. Hastie, T. J., and Tibshirani, R. J. (1990). *Generalized Additive Models*, Chapman and Hall, New York.

12. Hastie, T., Tibshirani, R., and Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer-Verlag, New York.
13. Harrell, F. E. Jr. (2002). *Regression Modeling Strategies*, Springer-Verlag, New York.
14. Hornik, K., Stinchcombe, M., and White H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, **2**, 359–366.
15. Jordan, M. I., and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm, *Neural Computation*, **6**, 181–214.
16. McCullagh, P., and Nelder, J. (1989). *Generalized Linear Models*, Second edition, Chapman and Hall, London, U.K.
17. Venable, W. N., and Ripley, B. D. (2002). *Modern Applied Statistics with S*, Fourth edition, Springer-Verlag, New York.