Accuracy versus interpretability in flexible modeling: implementing a tradeoff using Gaussian process models

Tony A. Plate (tap@mcs.vuw.ac.nz)
School of Mathematical and Computing Sciences
Victoria University of Wellington, Wellington, New Zealand

ABSTRACT

One of the widely acknowledged drawbacks of flexible statistical models is that they are often extremely difficult to interpret. However, if flexible models are constrained to be additive they are much easier to interpret, as each input can be considered independently. The problem with additive models is that they cannot provide an accurate model if the phenomenon being modeled is not additive. This paper proposes that a tradeoff between accuracy and additivity can be implemented easily in a particular type of flexible model: a Gaussian process model. One can build a series of Gaussian process models which begin with the completely flexible and are constrained to be more and more additive, and thus interpretable. Observations of how the test error and importance of interactions change as the model becomes more additive give insight into the importance and nature of interactions. Models in the series can also be interpreted graphically with a technique for visualizing the effects of inputs in non-additive models that was adapted from plots for generalized additive models. This visualization technique shows the overall effects of different inputs and also shows which inputs are involved in interactions and how strong those interactions are.

1. Introduction

The ability to provide good predictions on new data is a vital property of a statistical model. However, it is often desirable and sometimes essential to interpret the model also. This is especially true in medical domains, where black-box methods are viewed with great suspicion [23; 16]. Flexible statistical models, such as neural networks, bagged decision trees, or Gaussian processes tend to be rather difficult to interpret. When interpretation is important, simpler but more easily interpretable methods such as logistic regression are often used instead. There is a risk with such simpler methods that they may fail to discover an important relationship in the data because they lack the flexibility to model it.

The standard practice in the neural network modeling community, and, to a lesser extent, in the flexible statistical modeling community has been to minimize model complexity only in the service of minimizing prediction error. However, given the importance of interpretability in some domains, it could be useful to have a modeling technique in which we can trade slight decreases in predictive power for increases in interpretability. The primary cause of difficulty of interpretation of flexible models is interactions rather than non-linear effects in individual variables, so the interpretability of a model can be regarded as inversely proportional to the importance of interactions in the model. This paper combines ideas from generalized additive models [8] and from Gaussian process modeling [22] in an attempt to implement and graphically present the results of various different positions in the tradeoff between minimizing prediction error and maximizing interpretability (i.e., minimizing the importance of interactions).

Generalized additive models (GAMs) are among the most interpretable, flexible, general-purpose, statistical models.[1] GAMs are not fully flexible — in a GAM, variables can have non-linear effects but do not interact. The lack of interaction is, in fact, what makes them so easily interpretable: the independent effect of each variable can be shown in a simple plot. These plots provide a very informative visualization of the model, and although they were originally designed for additive models, Plate et al. [12] showed how these plots could be adapted to visualize non-additive models.

Gaussian process models are an older modeling technique that have recently captured wider interest. They are fully flexible models and can be used for both regression and classification. Rasmussen [13], in some careful studies, showed that Gaussian process models often out-perform other models such as neural networks and decision trees. The idea presented here is that of installing a "knob" on Gaussian processes, which can be used to lower, in steps, the influence of interactions in the model, which makes the model more additive and hence more interpretable.

The modeling technique advocated in this paper is to build a series of Gaussian process models progressively constrained to be more and more additive. This additivity knob is intended to serve three purposes: (1) to allow one to see whether interactions are merely gratuitous or are actually necessary to achieve good predictions; (2) to verify observations about the individual effects of variables in an unconstrained model; and (3) to investigate the form of the model and how it changes as the model is forced to be more interpretable. Additive effects plots and diagnostics offer insight into these three issues. The plot of generalization error versus degree of additivity shows how important interactions are in the model. The plots of the additive effects of individual variables in the series of models show the

[1] Classification and regression trees [6] are probably the only other easily-interpretable class of general-purpose, flexible, statistical models (as long as the trees are small and not used in committees as in Breiman's [5] bagged trees).

overall effects of each variable and which variables are involved in interactions.

2. Additive models and additive plots

Part of the attraction of generalized additive models (GAMs) [8] is that, although they are non-linear, they are easily interpreted from simple plots.

A GAM can be expressed as

$$g(\mathbf{x}) = \mathbf{h}(\theta_\mathbf{1}(\mathbf{x_1}) + \theta_\mathbf{2}(\mathbf{x_2}) + \cdots + \theta_\mathbf{k}(\mathbf{x_k}))$$

where $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k})$ is the vector of input values and $g(\mathbf{x})$ is the predicted output value. The functions θ_1, θ_2 etc can be arbitrary non-linear functions. The function h relates the sum of the θ_i's to the output variable and depends on the distribution of the target variable and the distribution of errors in the target values. For example, for continuous target variables with Gaussian errors of constant variance, the identity function is appropriate for h. For binary valued target variables, the logistic function is appropriate for h: $h(a) = 1/(1+e^{-a})$, which gives the probability of the output being 1. The possibility of different functions for h is what makes GAMs generalized: appropriate functions exist for a wide range of target error distributions. The way the functions of the inputs combine is what makes GAMs additive. Although the effect of a particular input can be non-linear there are no interactions between inputs: the contribution of a particular input does not depend on the values of other inputs.[2]

A GAM can be easily interpreted via plots of the effects of inputs on the additive scale of the model, i.e., plots of θ_i. The θ_i are usually functions of one variable, and hence can be plotted easily.

3. Gaussian process models and additive Gaussian process models

Only an informal explanation of Gaussian process models will be given here; for a detailed description of how Gaussian process models can be applied to regression and classification tasks, the reader should consult Williams and Rasmussen [22], Rasmussen [13], or Neal [11].

Gaussian process models, as used in these papers, can be thought of as a type of smooth nearest neighbor model with an adaptive distance metric.

To begin with, ignore the inputs. Suppose we have N training examples, and we want to make a prediction on 1 new example. The outputs are assumed to be from a zero-mean $N+1$ dimensional Gaussian distribution,[3] with by an $N+1$ by $N+1$ covariance matrix. The vector of outputs can be thought of as a sample from this Gaussian distribution. The predicted distribution for the output for the new example is calculated by conditioning on the other (training) outputs. The mean (or median, or mode) of this distribution can be used as a point value prediction for this output. For example, if there is one training case, and the covariance between the training output and the new output is high, then the new output will have a narrow distribution with a mean close to the training case value. If the covariance is low, the mean will be close to zero and the distribution will be wide.

To make this prediction, we have used a covariance matrix which has $(N+1)(N+2)/2$ possible degrees of freedom. This may seem like an enormous number of parameters when we only have N training cases. However, this is where the inputs enter the picture. The elements of the covariance matrix are not free but are derived in a highly parameterized manner from the inputs. Element k, j of the covariance matrix is a function of the distance between the input vectors for the kth and jth examples (here, training and testing examples are treated equally). The distance metric is parameterized to allow the importance of different dimensions of the input space to vary. For pairs of examples which are close in input space according to the distance metric, the corresponding elements of the covariance matrix will be large, which means that their outputs are expected to be close.

Williams and Rasmussen [22], Rasmussen [13], and Neal [11] use small variations on the following covariance function,[4] where the input vectors are d dimensional and $\mathbf{x^{(k)}}$ is the input for example k:

$$\begin{aligned} C_{kj} &= c + \sum_{u=1}^{d} \sigma_u^2 x_u^{(k)} x_u^{(j)} \\ &\quad + \eta^2 \exp\left(- \sum_{u=1}^{d} \rho_u (x_u^{(k)} - x_u^{(j)})^2 \right) + J\delta_{kj}. \end{aligned}$$

The above explanation of the covariance matrix in terms of distances between inputs assumes that the exponential term, which involves the distance between $x^{(k)}$ and $x^{(j)}$, dominates this formula.

The parameters of the model are c, the σ_u, η, the ρ_u and J. Predictions can be made based on maximum likelihood values for these parameters, or in a Bayesian technique using Markov Chain Monte Carlo (MCMC) sampling of the parameter space. The publicly available software described in Neal [11] uses the latter, and this software was used for the experiments described in this paper.

A Gaussian process model of the above form can easily be made to be additive by changing the covariance function to be an additive function of distances on different input dimensions [13; 11]. In fact, a general exponential part can also be included, which gives the model both additive and general parts. The form of the covariance function used in this paper is as follows:

$$\begin{aligned} C_{kj} &= c + \eta_0^2 \exp\left(- \sum_{u=1}^{d} \rho_u (x_u^{(k)} - x_u^{(j)})^2 \right) \\ &\quad + \sum_{u=1}^{d} \eta_u^2 \exp\left(-\phi_u (x_u^{(k)} - x_u^{(j)})^2 \right) + J\delta_{kj}. \end{aligned}$$

The first exponential term in this formula, which is the same as above, is the general term which can take account of distance in any direction. The other

[2] It is possible to use functions of more than one variable in a component function in a GAM. This allows for interactions between particular inputs.

[3] Although the mean of the output distribution is assumed to be zero, this does not imply that the mean output for a particular set of examples is zero.

[4] Neal [11] allows covariance functions to have more than one exponential part, and allows the power in the exponential parts to vary.

exponential terms (in the sum) can each only look at distance in one input direction. Thus, if $\eta_0 = 0$, the covariance function is an additive function of the x_u. Since the predictions of a Gaussian process model are additive with respect to the terms in the covariance, the predictions of the model will be additive (if $\eta_0 = 0$). In some cases where the output actually is an additive function on the inputs, the general exponential part is automatically found to have zero importance (i.e, $\eta_0 = 0$).

The technique suggested in this paper for producing a series of models which range from a general function of inputs to a purely additive function of inputs is as follows: (1) Train a model with all parameters free (predictions are made based on the MCMC samples from the posterior density); (2) Train a series of models with decreasing fixed values of η_0. When η_0 is very small, the model will be additive. The proposal of this paper is that insight into the input-output relationship in the data can be gained by visualizing the function computed, and observing the quality of predictions and the importance of interactions in the effects of individuals inputs, as the model is forced to be more and more additive.

4. Plotting non-additive functions with additive plots

Plate et al. [12] describe how additive plots [8] can be adapted for visualizing non-additive models. Only a brief description is given here.

Consider a general function of d variables, $f(x_1, x_2, \ldots, x_d)$, and the problem of understanding how f varies with the x_i. If $d = 2$ we can plot a surface, but if d is higher, conventional plotting techniques fail us. The problem is that the value of f with x_i set to some value can vary with the remaining x_j. The method advocated in Plate et al. [12] is create a scatterplot of *effects* for each input variable. The scatterplot for input i shows what effect changing the value for input i has on the output (on the appropriate additive scale, which for a neural network will usually be the total input of the output unit). Each scatterplot consists of a number of short line segments, one for each training or test example. The segments for the first input are located at:

$$(x_1^{(k)}, f(x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)}) - f(b_1, x_2^{(k)}, \ldots, x_d^{(k)}))$$

where k is the example index and b_i is a *reference* value for the ith input dimension. The slope of each line segment is set to $\frac{\partial f}{\partial x_i}|_{x^{(k)}}$. Each line segment shows the effect, in the context of example k, of changing input i from b_i to the value actually occurring in example k.

For models which are truly additive, this technique reproduces the standard GAM plots of the component functions. For non-additive models, the plots give information about the following: (a) the overall importance of each input; (b) the trend in the effect of each input; (c) non-linearities in the effect of an input; and (d) interactions between the one input and others.

These plots give a considerable degree of insight into the function computed by a flexible statistical model. Plate et al. [12] used this method to visualize the effects of various variables in a MLP model of the risk of developing lung cancer. The plots revealed trends in the effects of individual variables similar to those found in a logistic regression model, but also showed the presence of considerable interactions. As the MLP was only slightly superior to the logistic regression model in terms of predictive performance, the questions arises as to whether the bulk of the observed interactions in the MLP are really useful or merely gratuitous — a product of excess flexibility in the model and noise in the data. This question was one of the motivations for the work described in the current paper — the development of a model in which one can turn down the degree of interaction and observe the effect on the model and its predictive power. Unfortunately, the lung-cancer data was not suitable to test the modeling technique as there are too many examples for use with current Gaussian Process modeling software, and too little difference in the performance of logistic regression and more flexible models.

5. Modeling spectroscopy data

Thodberg [19] reported impressive results using a neural network to predict fat content in meat from near-infrared spectroscopy measurements. On this data, consisting of 172 training examples, and 43 test examples, a stepwise linear regression model has a mean squared test error of 7.72 (4.8% of the variance unexplained), and a stepwise quadratic regression model has a mean squared test error of 0.64 (0.40%). Thodberg reports a mean squared test error of 0.13 (0.08%) for a committee of neural networks trained with Automatic Relevance Determination. All of these models use the 10 most significant principal components of the inputs (in normalized form). It is interesting to try to understand what features of the data the better models extract, especially as they are so much better than the linear model.

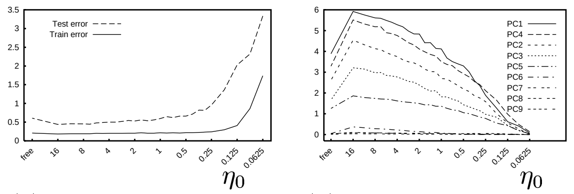
Gaussian process models with ten additive exponential parts (one general and one for each input) were applied to this data, using the nine most significant principal components as inputs. Firstly, an unconstrained model was trained.[5] The average value for η_0 in this unconstrained model was 0.84. Next, a series of models with η_0 set to different values were trained and tested. Although the test errors are not as low as Thodberg achieved, they are quite good (the lowest is around 0.44, or 0.27%) and one can make out interesting features discovered by the models. Figures 1 shows diagnostics[6] and Figure 2 shows the effects of the different principal components. The following conclusions can be drawn from these plots:

[5] The modeling was performed using Neal's [11] software, version 5.3. The relevant specifications were as follows: `gp-spec 9 1 10 - 0.01 / 0.05:0.5 0.2:2:1 / 0.4:1 0.2:2 omit:-1 / ...` (repeated for nine inputs.) `; model-spec logfile real 0.05:0.5 ; mc-spec logfile heatbath hybrid 20:4 0.5`. One hundred Monte-Carlo samples were taken, and the last fifty were used for predictions. η_0 was set to a specific value by changing the first prior specification after the slash in `gp-spec` from `0.05:5` to the desired value (i.e., a single value, no colon).

[6] The importance of interactions to the effect of x_i is measured by calculating the average squared error in a smooth fit to the gradient values in the plot for x_i, i.e., the points $\left(x_i^{(k)}, \frac{\partial f}{\partial x_i}|_{x^{(k)}}\right)$. The smooth fits were computed using the BRUTO curve fitter described in Hastie and Tibshirani [8].

- The interactions present in the unconstrained model are not gratuitous. In fact, a model forced to have a higher degree of interaction (i.e., a higher η_0) has slightly better predictions. Models with lower degrees of interaction perform worse.
- Components 1 to 5 have a large effect and are involved in important interactions.
- Component 1 has a strong positive trend in its effect, with a slight drop-off at high values. This is confirmed in the plot for the nearly-additive model ($\eta_0 = 00625$).
- Component 2 has no strong trend but is involved in significant interactions.
- Components 3 and 4 have strong negative trends, and components 5 and 6 mild positive trends.

One final point to note is that easy to produce these plots for the effects of the original variables rather than the principal component terms. However, lack of space prevents presenting or discussing these plots here.



(a) Average training and test errors

(b) Overall importance of interactions in the effect of each variable

Fig. 1: Model diagnostics for different values of η_0.

6. Discussion

Methods for interpreting complex models have received some attention within the field of neural networks. Three different approaches can be discerned. The first, and most prominent, is that of extracting rules that describe either how the network computes a function [20; 4; 1] or the function computed by the network [14]. This approach is most suited to applications in which inputs are discrete features or in which classification decisions are clear-cut, i.e., applications where smooth continuity of the computed function is relatively unimportant. The visualization techniques described in this paper (which can be easily applied to any kind of statistical model with continuous inputs) are intended for precisely the opposite kind of application: ones in which inputs are continuous and outputs vary smoothly. The second approach to interpreting neural networks is based on providing an interpretable view of the internal representational space [17; 18; 15]. Again, this approach is more suited to tasks with discrete inputs and clearcut outputs. The third approach is based on providing quantitative or graphical indications of the effect of input variables on the output [2; 3; 10]. This approach is the one most similar to that presented in this paper. However, the work of Baxt mainly concerned models with discrete inputs. One important difference between Baxt's methods and the plots of additive effects is that the latter make clear how the magnitude of the effect varies with the value of the input.

Methods for enforcing or formulating additivity in various families of flexible models have been investigated by a number of researchers. Moody and Rögnvaldsson [9] discuss various smoothing terms for feedforward neural networks which penalize higher order derivatives with respect to inputs. A smoothing term which penalized off-diagonal second derivatives of model outputs with respect to inputs would push the model towards additivity. Girosi, Jones, and Poggio [7] discuss formulations of additive models in terms of regularization networks. The smoothing-spline ANOVA (SS-ANOVA) of Wahba, Wang, Gu, Klein, and Klein [21] is also a Gaussian process model, though with a different covariance function to that used here. SS-ANOVA is also based on additive components, and is actually more general than the model used in this paper: it starts with functions of single variables, then adds in functions of two variables, and so on up to the full interaction term. The Gaussian process model used in this paper only has functions of single variables and the full interaction term. There are two practical advantages of this type of Gaussian process model over neural networks or a more general Gaussian process model: (i) there is a single knob (the η_0 value) that controls the degree of additivity, and (ii) it is simple to specify and fit a series of models with range of degrees of additivity.

7. Conclusion

The visualization methods described here can be used to show the effects of inputs on a single scalar output of any statistical model, including complex models such as committees of neural networks or bagged decision trees. These methods can also be used to display effects of original data when the model is based on transformed data (by perturbing data in the original input coordinates). The gain that comes from combining this visualization method with the general+additive-term formulation of Gaussian process models is that one has a "knob" to control the degree of additivity (and thus interpretability) of the model and a way of visualizing the effects of turning the knob. The visualizations show three important aspects of the model: the overall importance of interactions, the general effects of each variable, and which variables are involved in interactions and how important those interactions are.

Acknowledgments

Thanks to Radford Neal and Chris Williams for very useful discussions and comments on a draft of this paper, and to Radford for the use of his Markov-Chain Monte-Carlo Gaussian process modeling software and advice on using it. Thanks also to Hans Henrik Thodberg and the Tecator company for making available the near-infrared spectroscopy data recorded on the Tecator Infratec Food and Feed Analyzer.

References

[1] Alexander, J. A. and M. C. Mozer (1995). Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems*, Volume 7, pp. 609–616. The MIT Press, Cambridge.

[2] Baxt, W. G. (1992). Analysis of the clinical variables driving decision in an artificial neural network trained to identify

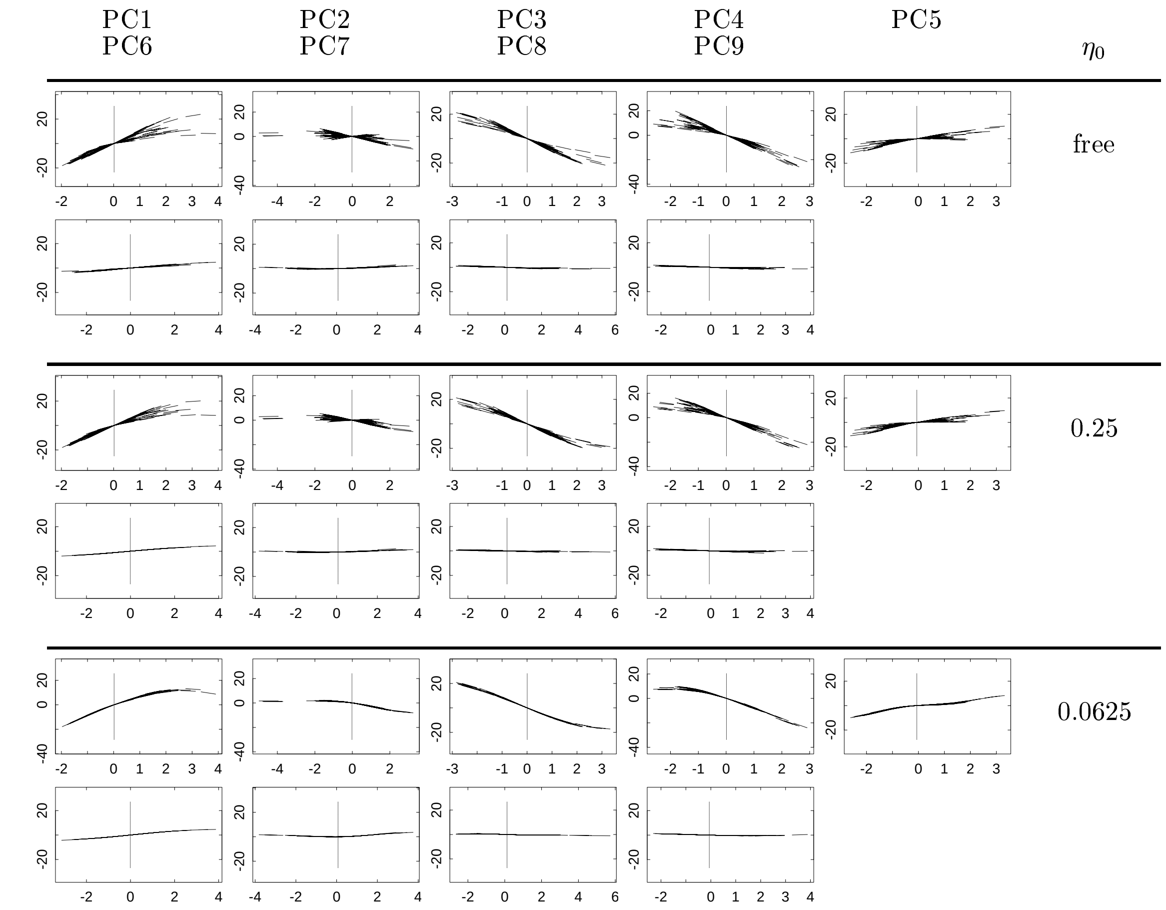


Fig. 2: Additive effects of the first nine principal components in three Gaussian process models for the spectroscopy data.

the presence of myocardial infarcation. *Annals of Emergency Medicine 21*(12), 1439–1444.

[3] Baxt, W. G. and H. White (1995). Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acture myocardial infarction. *Neural Computation 7*, 624–638.

[4] Blasig, R. (1994). GDS: Gradient descent generation of symbolic classification rules. In J. D. Cowan, G. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6 (NIPS*93)*, San Mateo, CA, pp. 1093–1100. Morgan Kaufmann.

[5] Breiman, L. (1996). Bagging predictors. *Machine Learning 26*(2), 123–140.

[6] Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees.* Belmont, CA.: Wadsworth.

[7] Girosi, F., M. Jones, and T. Poggio (1995). Regularization theory and neural networks architectures. *Neural Computation 7*(2), 219–269.

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

[9] Moody, J. E. and T. S. Rögnvaldsson (1996). Smoothing regularizers for projective basis function networks. Technical Report OGI CSE TR 96-006, Dept. of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology.

[10] Moseholm, L., E. Taudorf, and A. Frosig (1993). Pulmonary function changes in asthmatics associated with low-level SO_2 and NO_2, air pollution, weather, and medicine intake. *Allergy 48*, 334–344.

[11] Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. Technical Report TR9702, Dept. of Statistics, University of Toronto.

[12] Plate, T., J. Bert, J. Grace, and P. Band (1997). Visualizing the function computed by a feedforward neural network. In *Proceedings of The Fourth International Conference on Neural Information Processing (ICONIP'97)*. Springer Verlag.

[13] Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression.* Ph. D. thesis, University of Toronto. Available via http://www.cs.toronto.edu/~carl.

[14] Saito, K. and R. Nakano (1988). Medical diagnostic expert system based on pdp model. In *IEEE International Conference on Neural Networks*, San Diego CA, pp. 255–262.

[15] Sanger, D. (1989). Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. *Connection Science 1*, 115–138.

[16] Sharp, D. (1995). From "black box" to bedside, one day? (commentary). *The Lancet 346*, 1050.

[17] Shultz, T., Y. Oshima-Takane, and Y. Takane (1995). Analysis of unstandardized contributions in cross connected networks. In G. Tesauro, D. S. Touretzky, and T. K. Leen (Eds.), *Advances in Neural Information Processing Systems 7*, Cambridge, MA, pp. 601–608. MIT Press.

[18] Shultz, T. R. and J. L. Elman (1994). Analyzing cross connected networks. In J. D. Cowan, G. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6 (NIPS*93)*, San Mateo, CA, pp. 1117–1124. Morgan Kaufmann.

[19] Thodberg, H. H. (1996). A review of bayesian neural networks with an application to near infrared spectroscopy. *IEEE Transactions on Neural Networks 7*(1), 56–72.

[20] Towell, G. and J. Shavlik (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning 13*(1), 71–101.

[21] Wahba, G., Y. Wang, C. Gu, R. Klein, and B. Klein (1994). Structured machine learning for 'Soft' classification with smoothing spline anova and stacked tuning, testing and evaluation. In J. Cowan, G. Tesauro, and J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6*, pp. 415–422. Morgan Kauffman.

[22] Williams, C. K. I. and C. E. Rasmussen (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, Volume 8, pp. 514–520. The MIT Press, Cambridge.

[23] Wyatt, J. (1995). Nervous about artificial neural networks? (commentary). *The Lancet 346*, 1175–1177.