Seminar für Statistik
Ludwig Maximilians Universität München

Dynamic Neural Regression Models

Thomas Briegel and Volker Tresp
Siemens AG, Corporate Technology
Department Neural Computation
Otto-Hahn-Ring 6
D-81730 München, Germany
thomas.briegel@mchp.siemens.de
volker.tresp@mchp.siemens.de

Discussion Paper

Abstract

We consider sequential or online learning in dynamic neural regression models. By using a state space representation for the neural network's parameter evolution in time we obtain approximations to the unknown posterior by either deriving posterior modes via the Fisher scoring algorithm or by deriving approximate posterior means with the importance sampling method. Furthermore, we replace the commonly used Gaussian noise assumption in the neural regression model by a more flexible noise model based on the Student t-density. Since the t-density can be interpreted as being an infinite mixture of Gaussians, hyperparameters such as the degrees of freedom of the t-density can be learned from the data based on an online EM-type algorithm. We show experimentally that our novel methods outperform state-of-the art neural network online learning algorithms like the extended Kalman filter method for both, situations with standard Gaussian noise terms and situations with measurement outliers.

1 Introduction

Linear regression is a widely used technique for data analysis which dominates empirical data modeling in various fields. The reason for its popularity is that it is both conceptually and computationally simple to identify a linear model. The most prominent deficiency of the linear model is its strong assumption about the true relationship in the data set: for data sets which do not conform well to a linear model, predictions based on a linear regression model may be completely inaccurate. Many real-world systems and phenomena exhibit complex *nonlinear* characteristics and cannot be treated satisfactorily using conventional linear models. This work focuses on *neural network regression models* which are able to extract general nonlinear relationships from empirical data. During the last decade, there has been a tremendously growing interest in using neural networks as an alternative approach for empirical data modeling. In recent years neural computing has proven a practical technology and has gained widespread acceptance with successful applications in many fields. The objective of this work is to develop *sequential* or *online learning* methods for situations where the characteristics of the data-generation process undergo changes over time. Sequential or online learning is of great interest in many applications where data sequences either exhibit non-stationary behavior or are difficult and expensive to obtain *before* the training process. These include time-series forecasting, tracking and surveillance, signal and image processing, communications, and fault detection. Sequential learning refers to situations where the learning process occurs uninterrupted over the whole lifetime of the operation, which allows us to track changing dynamics of the underlying process. We adopt a *Bayesian framework* to sequential estimation of neural network model parameters. A suitable way to online estimation of the probability distribution of the neural network parameter vector over time is to use a state space formulation. In this approach the network parameters are treated as hidden states which have to be estimated from observed input-output data. Two sequential approximate Bayesian learning methods are derived which turn out to be particularly promising. The first is based on a Gaussian approximation to the posterior parameter distribution where centers and widths of the approximating Gaussian are derived from posterior modes and curvatures of the parameter distribution and the second approach derives parameter updates from a Gausssian approximation based on approximate posterior means and covariances. A commonly used assumption in linear or neural regression models is that targets are disturbed by independent additive Gaussian noise. The most common reason why researchers depart from the Gaussian noise assumption is the presence of outliers. Dealing with outliers is of critical importance for online learning tasks. Here one is interested in avoiding inappropriate parameter changes due to measurement outliers to maintain stable online learning capability. State-of-the art online learning algorithms are known to be nonrobust against such outliers since they are based on a Gaussian error assumption. We replace the Gaussian noise model by a more ¤exible noise model based on the Student-t-density. The degrees of freedom of the t-density can be chosen such that as special cases either the Gaussian density or the Cauchy density are realized. The latter is commonly used for deriving robust learning methods. The application of the posterior mode approximation to t-distributed output errors leads to a robust sequential learning algorithm where the hyperparameters of the t-density can be determined from the data with an online EM-type algorithm.

The paper is organized as follows. In the next Section 2 we provide a brief introduction to the neural network methodology in the context of regression models. For simplicity, we assume that a set of batch data is given such that the neural network parameter vectors can be estimated with the maximum likelihood method. In Section 3 we develop sequential learning algorithms for Gaussian measurement noise terms (3.1) and t-distributed error terms (3.2). In the last two sections (Section 4 and Section 5) we present experiments and conlusions, respectively.

2 Neural Regression Models

Let us assume we have given a *training data* set $\mathcal{D} = \{\boldsymbol{u}_t, y_t\}_{t=1}^T$ where $\boldsymbol{u}_t = (u_{1,t}, \ldots, u_{k,t})^\top \in \mathbb{R}^k$ is the t-th k-dimensional (independent) input vector and $y_t \in \mathbb{R}$ is the t-th single valued dependent output variable.[1] $\top$ denotes the transpose operator. A *linear regression model* hypothesizes that the dependent variable y can be modeled as the outcome of an underlying deterministic *linear* relationship between the independent variable $\boldsymbol{u}$ and the output y, corrupted by additive noise terms ϵ

$$y = \sum_{i=1}^{k} w_i u_i + w_0 + \epsilon. \tag{1}$$

Hereby, ϵ denotes zero-mean uncorrelated Gaussian noise with known variance σ_ϵ^2, i.e. $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$.[2] $\boldsymbol{w} = (w_0, w_1, \ldots, w_k)^\top \in \mathbb{R}^{k+1}$ denotes the vector of parameter values in the model. The application of the *maximum likelihood method* leads to the optimization problem

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \sum_{t=1}^{T} \Big(y_t - \sum_{i=1}^{k} w_i u_{i,t} - w_0 \Big)^2 \tag{2}$$

which can be conveniently solved with a pseudo-inverse (Rao and Mitras 1971). The linear model is one of the most popular models used for learning relationships between continuous variables. The reason for its popularity is that it is both conceptually and computationally simple to learn a linear model and the resulting model is easily given an intuitive representation. The most prominent de£ciency of the linear model is its strong assumption about the true relationship in the data set. For data sets which do not conform well to a linear model, predictions may be completely

[1] For notational convenience, we restrict ourselves to one-dimensional outputs.

[2] In the following, $\mathcal{N}(z|\mu, \sigma^2)$ is the abbreviation for a Gaussian density with mean μ and variance σ^2, evaluated at $z \in \mathbb{R}$. We use $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$ without argument z to distinguish a Gaussian density from its distribution.

inaccurate. Many real-world systems and phenomena exhibit complex *nonlinear* characteristics and cannot be treated satisfactorily using conventional linear models. Therefore, it would be of great interest to have a broader class of models capable of handling more complex relationships than linear models provide. There are in fact many different ways in which to extract general nonlinear dependencies from data. Here, the focus is on *neural networks* for representing nonlinear mappings between multi-dimensional real-valued spaces. It is a mere half-century since the publication of arguably the £rst paper on neural network modeling by McCulloch and Pitts (1943). The early motivation originated in deriving simple mathematical models of nervous activity. This motivation was extended during the 1950's and 60's from cognitive scientists to constructing simpli£ed models of the human brain. The human brain very quickly achieves complex tasks because of a vast number of neurons, complex interneuron connections, and the massively parallel way in which many simple operations are carried out simultaneously. Research in cognitive science aims at discovering and emulating the precise structure and mode of operation of the brain. Our motivation for using neural networks, however, is to exploit, in nonneurological contexts, the potential of simple computational units interlinked in an appropriate way, to learn models for complex nonlinear dependencies that are interesting from the viewpoint of arti£cial intelligence or useful in engineering applications. For a general introduction to neural computation we suggest the book by Bishop (1995).

The neural network most commonly used in engineering applications is the *multi-layer perceptron network* (MLP) (Rumelhart et al. 1986; Bishop 1995). This network takes in a multivariate input vector $\boldsymbol{u} = (u_1, \ldots, u_k)^\top \in \mathbb{R}^k$ and computes one or more output values, $y_1, \ldots, y_q$, perhaps using some number of layers of *hidden units*. In a typical network with one hidden layer and a single *output unit*, such as illustrated in Figure 1 (left), the output value y might be computed as[3]

$$\begin{aligned} & g : \mathbb{R}^k \to \mathbb{R}, \\ y = g(\boldsymbol{u}; \boldsymbol{w}) &= \sum_{i=1}^{h} w_i b_i(\boldsymbol{u}) + w_0 \\ &= \sum_{i=1}^{h} w_i \tanh\Big(\sum_{j=1}^{k} w_{ij} u_j + w_{i0}\Big) + w_0. \end{aligned} \tag{3}$$

Here, w_{ij} are the *weights* on the connection from input unit j to hidden unit i. Similarly, w_i is the weight on the connection from hidden unit i to the output unit. The w_{i0} and w_0 are the *biases* of the hidden units and the output unit, respectively. The output value is just a weighted sum of h hidden unit values or *basis functions* $b_i(\boldsymbol{u})$, plus a bias. Each hidden unit i computes a similar weighted sum of input values, and then passes it through a nonlinear *activation function*. Here, the activation function is the hyperbolic tangent, an anti-symmetric function of sigmoidal shape, whose value is close to -1 for large negative arguments, zero for a zero argument, and close to $+1$ for large positive arguments. Figure 1 (right) displays the response $b_i(\boldsymbol{u})$ of a hidden unit with a hyperbolic tangent activation function to a bivariate input signal $\boldsymbol{u} = (u_1, u_2)^\top$. The dashed line shows the hyperplane $w_{i1}u_1 + w_{i2}u_2 = w_{i0}$ for $w_{i0} = 0$.

A straightforward extension of the linear regression model (1) is the *neural regression model*

$$y = g(\boldsymbol{u}; \boldsymbol{w}) + \epsilon \tag{4}$$

[3] For notational convenience, we group the adjustable parameters $\{w_0, w_1, \ldots, w_h, w_{10}, \ldots, w_{hk}\}$ of the MLP into the parameter vector $\boldsymbol{w} = \{w_i\}_{i=1,\ldots,h(k+2)+1}$ using a single index.

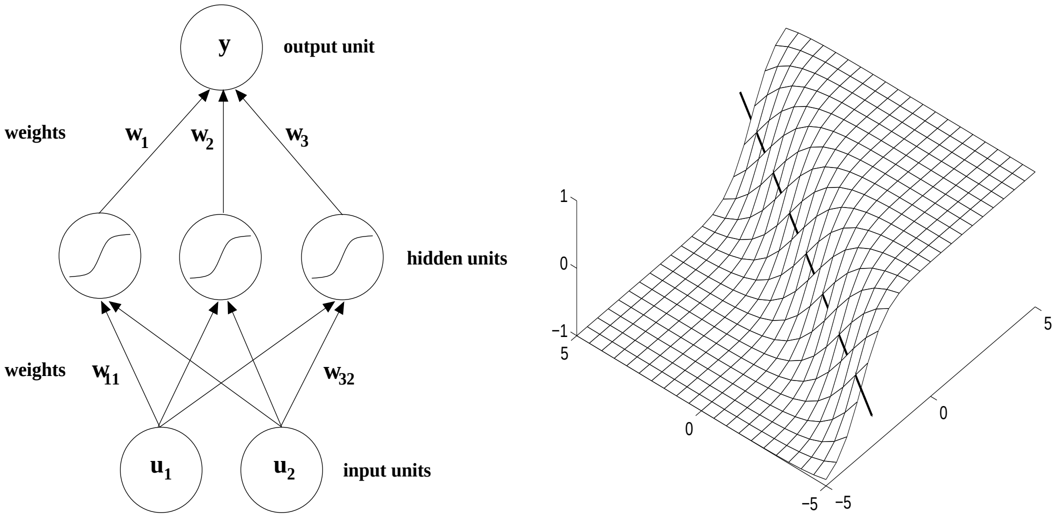

Figure 1. Left: A multi-layer perceptron with one layer of three hidden units. The input units at the bottom take in a bivariate input vector $\boldsymbol{u} = (u_1, u_2)^\top$. The values of the hidden units are then computed, followed by the value of the output unit y. The value of a unit is a function of the weighted sum of values received from other units connected to it via arrows. Each arrow is assigned a single weight which is determined by the learning procedure. **Right:** Response of a hidden unit basis function $b_i(\boldsymbol{u})$ to a two-dimensional input vector $\boldsymbol{u} = (u_1, u_2)^\top$. The dashed line indicates the hyperplane $w_{i1}u_1 + w_{i2}u_2 = 0$ with $w_{i1} = 1$ and $w_{i2} = -1$.

where the nonlinear regression function $g : \mathbb{R}^k \rightarrow \mathbb{R}$ is implemented using the multi-layer perceptron network (3) with weight vector $\boldsymbol{w} \in \mathbb{R}^{h(k+2)+1}$. The weight vector $\boldsymbol{w}$ in the neural network can be determined based on the training data set $\mathcal{D}$ by applying the maximum likelihood principle yielding to the optimization problem

$$\boldsymbol{w}_{\mathrm{ML}} = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \sum_{t=1}^{T} \big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w})\big)^2. \tag{5}$$

$E(\boldsymbol{w})$ is the commonly applied cost function to compute regression estimates in a neural network context. Unlike in the linear regression case, the optimal parameter vector $\boldsymbol{w}_{\mathrm{ML}}$ cannot be determined with the methods of linear algebra anymore. Finding the appropriate weight vector is now a nonlinear optimization problem, which is commonly solved using some gradient-based optimization method. Such methods take advantage of the gradients of the error function E which can be calculated by *backpropagation* (Werbos 1974; Rumelhart et al. 1986). Backpropagation[4] is an ef£cient method for calculating the gradients $\partial E/\partial w_i, \partial E/\partial w_{ij}$ of the error function with respect to all adjustable network weights and biases in one forward and backward pass through the network. For the multi-layer perceptron architecture given in (3) and the sum of squares error function (5) the backpropagation algorithm is derived in the Appendix. The gradient information calculated by backpropagation is used to compute the adjustments to be made to the network parameters with an iterative optimization method. The simplest such technique, and the one originally considered by Rumelhart et al. (1986), involves *gradient descent* and is given by the iterative

[4] Backpropagation can be traced back to Werbos, but only found widespread use after it was independently discovered by Rumelhart and co-workers in 1986.

update rule

$$\begin{aligned} \boldsymbol{w}^{\text{new}} &= \boldsymbol{w}^{\text{old}} - \eta \frac{\partial E}{\partial \boldsymbol{w}} = && (6) \\ & \boldsymbol{w}^{\text{old}} + \eta \sum_{t=1}^{T} \frac{\partial g}{\partial \boldsymbol{w}} \big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w})\big) && (7) \end{aligned}$$

where the *stepsize parameter* η is typically a small positive real number. (6) is applied iteratively until some convergence criterion is satisfied, e.g. such as that the norm of the error gradient $\partial E / \partial \boldsymbol{w}$ is within some specified tolerance of zero for $\boldsymbol{w} = \boldsymbol{w}^{\text{new}}$. The problem of minimizing continuous, differentiable functions of many variables, however, is one which has been extensively studied, and many of the conventional approaches to this problem are directly applicable to the training of neural networks. Therefore, details about sophisticated higher order methods, for instance, the family of Quasi-Newton methods, are not provided and the reader is referred to some of the textbooks which cover standard nonlinear optimization techniques, for instance, the textbook by Gill et al. (1981).

An important aspect that makes neural networks interesting for practical applications is the fact that they are *universal approximators*. Even the relatively simple multi-layer perceptron network as defined in (3) is capable of approximating arbitrarily well any nonlinear functional continuous relationship from one finite-dimensional space to another, provided the number of hidden neurons h is sufficiently large (Cybenko 1989; Hornik 1991). The ability of neural networks to approximate arbitrary nonlinear functional mappings, however, is not extraordinary. Polynomials and Fourier series, for instance, also obey this universal approximation ability. The importance of neural networks lies in the way in which they deal with the problem of scaling with dimensionality. Jones (1992) and Barron (1993, 1994) have studied the way in which the sum of squares error E decreases for an interesting class of functions as the number of weights in the model increases. For the multi-layer perceptron (3) they showed that this error falls as $O(1/h)$ where h is the number of hidden units in the network. This is a remarkable result since the error is independent of the input-dimension. By contrast, the error only decreases as $O(1/h^{2/k})$, where k is the dimensionality of the input space, for polynomials or any other series expansion where the coefficients of linear combinations of fixed basis functions are adapted. The approximation accuracy is also determined by the complexity of the underlying function which gives rise to the number h of hidden units in the network. Barron (1994) has also shown that the number of hidden units only needs to grow as the complexity of the problem itself grows, and not simply as the dimensionality grows. Practical experience shows that the number of free parameters in a multi-layer perceptron, for a given number of hidden units, typically only grows linearly or quadratically, with the dimensionality of the input space, as compared to k^h growth for a general h-order polynomial (Tresp 1995). Hence, multi-layer perceptrons overcome Bellman's "curse of dimensionality".

Neural networks offer a dramatic advantage for nonlinear function approximation in multi-dimensional spaces. The key point that makes this difference is that — in contrast to standard universal approximation techniques — the shape of each single basis function in the neural network can be determined in an optimal way by adapting the connections from the input to the hidden units. When using conventional function approximators like polynomials, the basis functions are predetermined and fixed. The price we have to pay for this efficient scaling with dimensionality is that we face a nonlinear optimization problem that is computationally intensive.

3 Dynamic Neural Regression Models

The traditional way to *sequential* or *online* estimation of neural network models is to apply *online backpropagation* (Rumelhart et al. 1986). In online backpropagation one computes a single gradient step in the error each time a new data pattern arrives to update the neural network weight vector sequentially. Although known for more than a decade now, online backpropagation may still be considered as the de facto state-of-the art neural network training method for online environments. In the last section neural network weight vectors were treated as unknown fixed constants. In the maximum likelihood framework described there, the goal was to find the single most likely value for the parameters given the observed data. For online learning purposes, however, it is convenient to employ a *sequential Bayesian approach* where the neural network weights $\boldsymbol{w}$ are treated as random variables. To account for changing dynamics in the underlying data-generation process we assume the weight vector $\boldsymbol{w}$ to be time-variant, i.e. $\boldsymbol{w} = \boldsymbol{w}_t$. We shall call a neural regression model with time-variant weight vectors a *dynamic neural regression model* following the terminology "dynamic generalized linear model" used in the statistics literature (Fahrmeir and Tutz 1994).

We account for time-variant neural network weights by assuming that the parameter vector at time t depends on the previous value $\boldsymbol{w}_{t-1}$ and a stochastic component $\boldsymbol{\epsilon}_t$. It is convenient to employ a state space representation to model the neural network's weight evolution in time. The neural network weights are interpreted as time-varying hidden states, following a first order random walk

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{\epsilon}_t. \tag{8}$$

The process $\boldsymbol{\epsilon}_t$ may represent our uncertainty in how the neural network parameters evolve with time. Assuming Gaussian distributed increments $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ leads to the state transition distribution

$$\boldsymbol{w}_t | \boldsymbol{w}_{t-1} \sim \mathcal{N}(\boldsymbol{w}_{t-1}, \mathbf{Q}_t). \tag{9}$$

The measurement equation describes the nonlinear relation between the input vector $\boldsymbol{u}_t \in \mathbb{R}^k$ and the corresponding measurement output $y_t \in \mathbb{R}$

$$y_t = g(\boldsymbol{u}_t; \boldsymbol{w}_t) + v_t. \tag{10}$$

The initial weight vector is normally distributed, i.e. $\boldsymbol{w}_0 \sim \mathcal{N}(\boldsymbol{a}_0, \mathbf{Q}_0)$. $\boldsymbol{w}_0$ may be initialized with the maximum likelihood regression estimate obtained from offline adaptation of the network if prior batch data is available. (10) implies the observation density

$$p(y_t | \boldsymbol{u}_t, \boldsymbol{w}_t) = p_{v_t}\big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w}_t)\big). \tag{11}$$

Figure 2 shows a graphical representation of the above state space model in engineering block form. With the state space formulation above the weight vectors $\boldsymbol{w}_t$ are now considered as random variables in a Bayesian context. Each time a new measurement y_t and input $\boldsymbol{u}_t$ arrives, our belief about the distribution of the weight vector $\boldsymbol{w}_t$ can be updated using the *state prediction density*

$$p(\boldsymbol{w}_t | \boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1}) = \int p(\boldsymbol{w}_t | \boldsymbol{w}_{t-1})\, p(\boldsymbol{w}_{t-1} | \boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1})\, d\boldsymbol{w}_{t-1} \tag{12}$$

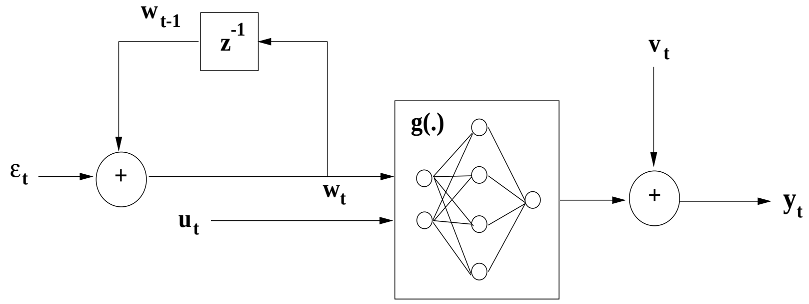

Figure 2. Dynamic neural regression model in engineering block form. The z^{-1} block is a unit time delay. The nonlinearity $\boldsymbol{g}(.)$ is represented by a multi-layer perceptron neural network model. The covariance matrix of the Gaussian disturbance term $\boldsymbol{\epsilon}_t$ is $\mathbf{Q}_t$ and the variance of the Gaussian measurement noise v_t is $\sigma^2_{v_t}$.

and the *filtering density*

$$p(\boldsymbol{w}_t|\boldsymbol{Y}_t, \boldsymbol{U}_t) = \frac{p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, p(\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1})}{\int p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, p(\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1})\, d\boldsymbol{w}_t} \tag{13}$$

where $\boldsymbol{Y}_t = (y_1, \ldots, y_t)^\top$ and $\boldsymbol{U}_t = (\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_t^\top)^\top$ denote the sequence of inputs and measurements up to time t. The filtering density $p(\boldsymbol{w}_t|\boldsymbol{Y}_t, \boldsymbol{U}_t)$ constitutes the complete solution to the sequential learning problem. Each time a new data tupel arrives the prior belief about the network weights — compressed in the state prediction density $p(\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1})$ — is updated by employing the information contained in the new data pattern $(\boldsymbol{u}_t, y_t)$. By computing the densities recursively we do not have to keep track of the complete history of weight distributions. While for the linear regression model (1) with Gaussian observation density

$$p_{v_t}\Big(y_t - \sum_{i=1}^{k} w_i u_{i,t} - w_0\Big) = \mathcal{N}\Big(y_t\Big|\sum_{i=1}^{k} w_i u_{i,t} + w_0, \sigma^2_{v_t}\Big) \tag{14}$$

the linear Kalman filter algorithm may be used to evaluate (12) and (13) analytically, for the general nonlinear case $g(\boldsymbol{u}_t; \boldsymbol{w}_t)$ involving either Gaussian or non-Gaussian observation densities $p_{v_t}(.)$, the above integrals are analytically intractable. This is why we need to resort to alternative *approximate* approaches to obtain feasible methods. Some researchers have derived solutions for the distributions of interest above by numerical integration methods. Kitagawa (1987) used this method to replace the state prediction and filtering integrals by finite sums over a large set of equally spaced grid points. Pole and West (1990) have attempted to reduce the problem of choosing the grid's location by implementing a dynamic grid allocation method. However, these methods are very difficult to implement in high-dimensional neural network weight spaces. Computing at every point in a dense multi-dimensional grid becomes prohibitively expensive in such environments (Gilks et al. 1996). In view of the above shortcomings and limitations, we propose two *approximate* Bayesian methods which are based on Gaussian approximations to the posterior. Gaussian approximations have the great advantage that we only have to propagate estimated means and covariances as sufficient statistics for the Gaussian state prediction and filtering densities, rather than propagating a set of samples for representing these densities or using numerical integration. In the first method (Section 3.1.1) we estimate the sufficient statistics of the Gaussian approximation by applying the Fisher scoring algorithm developed by Fahrmeir and Kaufmann

(1991). The second method (Section 3.1.2) propagates Gaussian approximations where approximate means and covariances of the Gaussian densities are derived from Monte Carlo integration implemented with importance sampling. For the first approach, we distinguish between the standard Gaussian noise assumption in the following Section 3.1 and a heavy-tailed noise model in 3.2, which is suitable for handling additive measurement outliers for robust online learning tasks.

3.1 Gaussian Measurement Noise

We start with the standard assumption of Gaussian distributed measurement noise terms $v_t \sim \mathcal{N}(0, \sigma^2_{v_t})$ resulting in the observation distribution

$$y_t|\boldsymbol{u}_t, \boldsymbol{w}_t \sim \mathcal{N}\big(g(\boldsymbol{u}_t; \boldsymbol{w}_t), \sigma^2_{v_t}\big). \tag{15}$$

The most popular advanced approach to sequentially estimating the weight sequence, i.e. to approximate the integrals above, is a Gaussian approximation to the state prediction and filtering density using the *extended Kalman filter algorithm.* This method and the aforementioned online backpropagation algorithm still represent the state-of-the art for sequentially adapting neural networks (Sum et al. 1999). One of the earliest implementations of extended Kalman filter trained neural networks is due to Singhal and Wu (1989). In their method, they used the state space formulation (8) and (10) with the noise distribution (15) to update the weight sequence in accordance with the following extended Kalman filter equations.

Algorithm extended Kalman filter based online learning

Initialization: $\boldsymbol{w}_{0|0} = \boldsymbol{a}_0, \quad \boldsymbol{\Sigma}_{0|0} = \mathbf{Q}_0$

$t = 1, 2, \ldots$

Predictor step:

$$\boldsymbol{w}_{t|t-1} = \boldsymbol{w}_{t-1|t-1} \tag{16}$$

$$\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{\Sigma}_{t-1|t-1} + \mathbf{Q}_t \tag{17}$$

Corrector step:

$$\boldsymbol{w}_{t|t} = \boldsymbol{w}_{t|t-1} + \mathbf{K}_t\big(y_t - \boldsymbol{g}(\boldsymbol{u}_t; \boldsymbol{w}_{t|t-1})\big) \tag{18}$$

$$\boldsymbol{\Sigma}_{t|t} = \boldsymbol{\Sigma}_{t|t-1} - \mathbf{K}_t\mathbf{G}_t\boldsymbol{\Sigma}_{t|t-1} \tag{19}$$

$$\mathbf{K}_t = \boldsymbol{\Sigma}_{t|t-1}\mathbf{G}_t^\top\big(\mathbf{G}_t\boldsymbol{\Sigma}_{t|t-1}\mathbf{G}_t^\top + \sigma^2_{v_t}\big)^{-1} \tag{20}$$

The entries of the (single row) Jacobian matrices $\mathbf{G}_t = \partial g(\boldsymbol{u}_t; \boldsymbol{w}_t)/\partial \boldsymbol{w}_t\big|_{\boldsymbol{w}_t=\boldsymbol{w}_{t|t-1}}$, i.e. the derivatives of the network outputs with respect to the weights, are calculated by backpropagating the output observations through the network (Appendix). The extended Kalman filter is an improvement over conventional neural network online learning techniques such as online backpropagation, in that it makes use of second order statistics (covariances). Online backpropagation is, in fact, a degenerate of the extended Kalman filter algorithm above. This can be seen easily when writing the corrector step in a slightly different form by applying the matrix inversion lemma (Anderson and Moore 1979) leading to

$$\begin{aligned}\boldsymbol{\Sigma}_{t|t} &= \left(\boldsymbol{\Sigma}^{-1}_{t|t-1} + \mathsf{G}_t \sigma^{-2}_{v_t} \mathsf{G}_t^\top\right)^{-1} && (21)\\ \boldsymbol{w}_{t|t} &= \boldsymbol{w}_{t|t-1} + \boldsymbol{\Sigma}_{t|t} \mathsf{G}_t^\top \sigma^{-2}_{v_t} \big(y_t - \boldsymbol{g}(\boldsymbol{u}_t; \boldsymbol{w}_{t|t-1})\big). && (22)\end{aligned}$$

In online backpropagation, the filtering error covariance matrix $\boldsymbol{\Sigma}_{t|t}$ is constrained to a small positive constant value ($\sigma^{-2}_{v_t} \boldsymbol{\Sigma}_{t|t} = \eta_t$) resulting in the well-known online backpropagation update equation

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} + \eta_t\, \mathsf{G}_t \big(y_t - \boldsymbol{g}(\boldsymbol{u}_t; \boldsymbol{w}^{\text{old}})\big). \qquad (23)$$

The Gaussian approximation to the Bayesian solution, implemented with the extended Kalman filter, results in a simple and elegant framework that is amenable to the design of sequential learning algorithms. Nevertheless, following arguments given in Briegel and Tresp (1999a), it is not well-defined where the approximating Gaussian posterior densities are centered with respect to the true (unknown) posterior density function. As pointed out in Sage and Melsa (1971), the Gaussian approximation derived from extended Kalman filtering are not centered about the mode of the posterior weight densities, they only provide an approximation to the posterior mode. Therefore, for highly nonlinear mappings $g(.; \boldsymbol{w}_t)$, the Gaussian approximation derived from extended Kalman filtering may be not guaranteed to cover much of the probability mass of the posterior filtering density. In the following Section 3.1.1, we extend the work of Fahrmeir & Kaufmann to our online environment by proposing a Laplace approximation to the posterior weight distribution. In particular, the posterior weight distribution is approximated by a Gaussian, centered about the mode, thereby determining the covariance of the approximating Gaussian by the local curvature of the posterior. This is achieved by adopting the Fisher scoring framework from Fahrmeir & Kaufmann to the state space model (8) and (10).

3.1.1 Sequential Learning Based on Posterior Modes

We start by considering the problem from an offline learning perspective where batch data sets $\boldsymbol{U}_t$ and $\boldsymbol{Y}_t$ are given. An offline smoothed estimate for the weight sequence $\boldsymbol{W}_t = (\boldsymbol{w}_0^\top, \ldots, \boldsymbol{w}_t^\top)^\top$ is obtained by maximizing the log-posterior with respect to $\boldsymbol{W}_t$, i.e.

$$\boldsymbol{W}_t^{\text{FS}} = \arg\max_{\boldsymbol{W}_t} \log p(\boldsymbol{W}_t | \boldsymbol{Y}_t, \boldsymbol{U}_t) \qquad (24)$$

or, equivalently after applying Bayes' rule, by maximizing the following expression

$$\begin{aligned}\boldsymbol{W}_t^{\text{FS}} &= \arg\max_{\boldsymbol{W}_t} \Big\{\log p(\boldsymbol{W}_t) + \log p(\boldsymbol{Y}_t | \boldsymbol{W}_t, \boldsymbol{U}_t)\Big\} && (25)\\ &= \arg\max_{\boldsymbol{W}_t} \Big\{\log p(\boldsymbol{w}_0) + \sum_{s=1}^{t} \log p(\boldsymbol{w}_s | \boldsymbol{w}_{s-1}) + \sum_{s=1}^{t} \log p\big(y_s | g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)\Big\}. && (26)\end{aligned}$$

Inserting the model assumptions (9) and (15), multiplying by (-1) and dropping constant terms leads to the *penalized log-likelihood criterion*

$$\begin{aligned} pll(\boldsymbol{W}_t) \;\propto\; & \frac{1}{2}\sum_{s=1}^{t}\frac{1}{\sigma_{v_s}^2}\big(y_s - g(\boldsymbol{u}_s;\boldsymbol{w}_s)\big)^2 + \frac{1}{2}(\boldsymbol{w}_0 - \boldsymbol{a}_0)^\top \mathbf{Q}_0^{-1}(\boldsymbol{w}_0 - \boldsymbol{a}_0) \\ & + \frac{1}{2}\sum_{s=1}^{t}(\boldsymbol{w}_s - \boldsymbol{w}_{s-1})^\top \mathbf{Q}_s^{-1}(\boldsymbol{w}_s - \boldsymbol{w}_{s-1}) \end{aligned} \tag{27}$$

which is subject to minimization. The £rst term in (27) is the standard sum of squares error function which is usually employed in neural network learning tasks. The second and third term in (27) can be interpreted as *penalty terms*. From a regularization theoretical point of view[5] these penalty terms serve two purposes, namely to restrict the complexity of each individual network $g(.;\boldsymbol{w}_t)$ and to restrict the temporal variation of each network model, i.e. penalize roughness in the weight sequence. The justi£cation for the latter is that in the case where $g(.;\boldsymbol{w}_t)$ changes slowly relative to the arrival rate of new data pattern we can expect to track the underlying relationship within reasonable bounds. The cost function (27) is the straightforward extension to the standard weight decay cost function which results from Laplace approximation to the posterior in neural of¤ine regression tasks (Bishop 1995).

A suitable way to determine a stationary point of $pll(\boldsymbol{W}_t)$ — the posterior mode estimate of $\boldsymbol{W}_t$ — is to apply Fisher scoring. With the current estimate $\boldsymbol{W}_t^{\text{old}}$ we get a new estimate $\boldsymbol{W}_t^{\text{new}} = \boldsymbol{W}_t^{\text{old}} + \eta\boldsymbol{\delta}$ for the unknown weight sequence $\boldsymbol{W}_t$ where $\boldsymbol{\delta}$ is the solution of

$$\mathbf{S}(\boldsymbol{W}_t^{\text{old}})\boldsymbol{\delta} = \boldsymbol{s}(\boldsymbol{W}_t^{\text{old}}) \tag{28}$$

with the score function

$$\boldsymbol{s}(\boldsymbol{W}_t) = -\frac{\partial\, pll(\boldsymbol{W}_t)}{\partial \boldsymbol{W}_t} \tag{29}$$

and the expected information matrix

$$\mathbf{S}(\boldsymbol{W}_t) = \mathrm{E}\Big[\frac{\partial^2 pll(\boldsymbol{W}_t)}{\partial \boldsymbol{W}_t \partial \boldsymbol{W}_t^\top}\Big]. \tag{30}$$

The derivations in Fahrmeir and Kaufmann (1991) apply straightforward to this situation. Hence, solving (28), i.e. to compute the inverse of the expected information matrix, can be performed by $\mathbf{LDL}^\top$ decomposition in one forward and backward pass since the expected information matrix is a block-tridiagonal matrix. The forward-backward steps have to be iterated to obtain the posterior mode estimate $\boldsymbol{W}_t^{\text{FS}} = \big(\boldsymbol{w}_{0|t}^{\text{FS}\top},\ldots,\boldsymbol{w}_{t|t}^{\text{FS}\top}\big)^\top$ for $\boldsymbol{W}_t$. A single forward-backward step is given by the following algorithm.

[5] For an overview of regularization in neural networks, consult Bishop (1995).

Algorithm Fisher scoring based online learning

Initialization: $\boldsymbol{w}^{\mathrm{FS}}_{0|0} = \boldsymbol{a}_0, \quad \boldsymbol{\Sigma}_{0|0} = \mathbf{Q}_0$

Forward recursions: $s = 1, \ldots, t$

$$\begin{aligned}
\boldsymbol{\Sigma}_{s|s-1} &= \boldsymbol{\Sigma}_{s-1|s-1} + \mathbf{Q}_s && (31)\\
\mathbf{B}_s &= \boldsymbol{\Sigma}_{s-1|s-1}(\boldsymbol{\Sigma}_{s|s-1})^{-1} && (32)\\
\boldsymbol{\Sigma}_{s|s} &= \big((\boldsymbol{\Sigma}_{s|s-1})^{-1} + \mathbf{G}_s(\boldsymbol{w}^{\mathrm{FS}}_{s|t})\sigma_{v_s}^{-2}\mathbf{G}_s^\top(\boldsymbol{w}^{\mathrm{FS}}_{s|t})\big)^{-1} && (33)\\
\boldsymbol{\gamma}_s &= \boldsymbol{s}_t(\boldsymbol{w}^{\mathrm{FS}}_{s|t}) + \mathbf{B}_s{}^\top \boldsymbol{\gamma}_{s-1} && (34)
\end{aligned}$$

Corrector step: $\boldsymbol{\delta}_t = \boldsymbol{\Sigma}_{t|t}\boldsymbol{\gamma}_t$

Backward recursions: $s = 1, \ldots, t$

$$\begin{aligned}
(\mathbf{D}_{s-1})^{-1} &= \boldsymbol{\Sigma}_{s-1|s-1} - \mathbf{B}_s\boldsymbol{\Sigma}_{s|s-1}\mathbf{B}_s{}^\top && (35)\\
\boldsymbol{\Sigma}_{s-1|t} &= (\mathbf{D}_{s-1})^{-1} + \mathbf{B}_s\boldsymbol{\Sigma}_{s|t}\mathbf{B}_s{}^\top && (36)\\
\boldsymbol{\delta}_{s-1} &= (\mathbf{D}_{s-1})^{-1}\boldsymbol{\gamma}_{s-1} + \mathbf{B}_s\boldsymbol{\delta}_s && (37)
\end{aligned}$$

where $\mathbf{G}_s(\boldsymbol{z}) = \partial g(\boldsymbol{u}_s; \boldsymbol{z})/\partial \boldsymbol{z}$ and $\boldsymbol{s}_t(\boldsymbol{z}) = -\partial pll(\boldsymbol{W}_t)/\partial \boldsymbol{w}_{\boldsymbol{s}}\big|_{\boldsymbol{w}_s=\boldsymbol{z}}$. The score function $\boldsymbol{s} = -\partial pll(\boldsymbol{W}_t)/\partial \boldsymbol{W}_{\boldsymbol{t}} = (\boldsymbol{s}_0^\top, \ldots, \boldsymbol{s}_t^\top)^\top$ can be derived as

$$\begin{aligned}
\boldsymbol{s}_0 &= -\mathbf{Q}_0^{-1}(\boldsymbol{w}_0 - \boldsymbol{a}_0) + \mathbf{Q}_1^{-1}(\boldsymbol{w}_1 - \boldsymbol{w}_0)\\
&\cdots\\
\boldsymbol{s}_s &= \mathbf{G}_s\sigma_{v_s}^{-2}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big) - \mathbf{Q}_s^{-1}(\boldsymbol{w}_s - \boldsymbol{w}_{s-1})\\
&\quad + \mathbf{Q}_{s+1}^{-1}(\boldsymbol{w}_{s+1} - \boldsymbol{w}_s), \qquad s = 1, \ldots, t && (38)\\
&\cdots\\
\boldsymbol{s}_t &= \mathbf{G}_t\sigma_{v_s}^{-2}\big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w}_t)\big) - \mathbf{Q}_t^{-1}(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}).
\end{aligned}$$

The above algorithm provides estimates only after a block of t consecutive £lter steps. For *online posterior mode smoothing* it is of interest to smooth backwards after *each* new data pattern. If posterior mode estimation is applied sequentially for $t = 1, 2, \ldots$, then

$$\boldsymbol{W}^{\mathrm{FS}}_{t-1} = \big(\boldsymbol{w}^{\mathrm{FS}}_{0|t-1}{}^\top, \ldots, \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1}{}^\top\big)^\top, \tag{39}$$

the posterior mode smoother for time $t-1$, grouped together with the step-one predictor

$$\boldsymbol{w}_{t|t-1} = \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1} \tag{40}$$

into

$$\boldsymbol{W}^{\mathrm{old}}_t = \big(\boldsymbol{W}^{\mathrm{FS}}_{t-1}, \boldsymbol{w}^\top_{t|t-1}\big)^\top \tag{41}$$

is a reasonable starting vector for obtaining the posterior mode smoother $\boldsymbol{W}^{\mathrm{FS}}_t$ at time step t as soon as a new data tupel $(\boldsymbol{u}_t, y_t)$ becomes available. It is clear that as the algorithm has to traverse all previous observations repeatedly to find a solution, the processing time for a new pattern becomes longer and longer as new patterns arrive. To allow for online processing it may thus be necessary to truncate the current data history at some fixed horizon, e.g. account only for the latest τ_t data tupels, to reduce the computational load. This is a reasonable assumption in non-stationary environments for online purposes. A formal justification of such a truncation approach in the context of inference in dynamic Bayesian networks has recently been provided by Boyden and Koller (1999).

If we further make the assumption that in most cases a new data tupel $(\boldsymbol{u}_t, y_t)$ should *not* change weight estimates too drastically implies that a *single* Fisher scoring step often suffices to obtain the new posterior mode estimate at time t. The resulting *single Fisher scoring step algorithm* is derived in Fahrmeir and Kaufmann (1991) in the context of dynamic exponential family regression and is extended here to sequential Bayesian neural network learning. The algorithm with lookback parameter τ_t is given below.

Algorithm single Fisher scoring step online learning

Initialization: $\boldsymbol{w}^{\mathrm{FS}}_{0|0} = \boldsymbol{a}_0\,, \quad \boldsymbol{\Sigma}_{0|0} = \mathbf{Q}_0\,, \quad \gamma_0 = \boldsymbol{s}_0(\boldsymbol{a}_0)$

Predictor step: $\boldsymbol{w}_{t|t-1} = \boldsymbol{w}_{t-1|t-1}$

Forward recursions: $s = 1, \dots, t$

$$\boldsymbol{\Sigma}_{s|s-1} = \boldsymbol{\Sigma}_{s-1|s-1} + \mathbf{Q}_s \tag{42}$$

$$\mathbf{B}_s = \boldsymbol{\Sigma}_{s-1|s-1}\boldsymbol{\Sigma}^{-1}_{s|s-1} \tag{43}$$

$$\boldsymbol{\Sigma}_{s|s} = \left(\boldsymbol{\Sigma}^{-1}_{s|s-1} + \mathbf{G}^{\top}_s(\boldsymbol{w}^{\mathrm{FS}}_{s|t-1})\sigma^{-2}_{v_t}\mathbf{G}_s(\boldsymbol{w}^{\mathrm{FS}}_{s|t-1})\right)^{-1} \tag{44}$$

$$\gamma_s = \boldsymbol{s}_s(\boldsymbol{w}^{\mathrm{FS}}_{s|t-1}) + \mathbf{B}^{\top}_s\gamma_{s-1} \tag{45}$$

Corrector step: $\boldsymbol{w}^{\mathrm{FS}}_{t|t} = \boldsymbol{w}_{t|t-1} + \eta\boldsymbol{\Sigma}_{t|t}\gamma_t$

Backward recursions: $s = t, ..., t-\tau_t$

$$\mathbf{D}^{-1}_{s-1} = \boldsymbol{\Sigma}_{s-1|s-1} - \mathbf{B}_s\boldsymbol{\Sigma}_{s|s-1}\mathbf{B}^{\top}_s \tag{46}$$

$$\boldsymbol{w}^{\mathrm{FS}}_{s-1|t} = \boldsymbol{w}_{s-1|t-1} + \mathbf{B}_s(\boldsymbol{w}_{s|t} - \boldsymbol{w}_{s|t-1}) + \eta\mathbf{D}^{-1}_{s-1}\gamma_{s-1} \tag{47}$$

$$\boldsymbol{\Sigma}_{s-1|t} = \mathbf{D}^{-1}_{s-1} + \mathbf{B}_s\boldsymbol{\Sigma}_{s|t}\mathbf{B}^{\top}_s \tag{48}$$

where $\mathbf{G}_s(\boldsymbol{z}) = g(\boldsymbol{u}_s;\boldsymbol{z})/\partial\boldsymbol{z}$.

The two algorithms above provide a convenient means of propagating Gaussian approximations to the posterior weight distributions. *Step-one predictions* $\hat{y}_t$ of the outcome y_t for a new data point $\boldsymbol{u}_t$ — based on the information available at the current time step $t-1$ — can be derived in the following way. To obtain the optimal step-one predictor, we have to evaluate the expectation of

$$p(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1}) = \int p\big(y_t|g(\boldsymbol{u}_t;\boldsymbol{w}_t)\big)\, p(\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1})\, d\boldsymbol{w}_t. \tag{49}$$

In order to evaluate this density we shall make use of the Gaussian approximation obtained from the above (single) Fisher scoring step algorithm. If we further make the assumption that the width of the posterior distribution (determined by the covariance $\boldsymbol{\Sigma}_{t-1|t-1}$) is suf£ciently narrow then we may approximate the network function $g(\boldsymbol{u}_t; \boldsymbol{w}_t)$ by its £rst order Taylor series expansion about the posterior mode, i.e.

$$g(\boldsymbol{u}_t; \boldsymbol{w}_t) \doteq g(\boldsymbol{u}_t; \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1}) + \mathbf{G}_t(\boldsymbol{w}_t - \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1}) \tag{50}$$

where $\mathbf{G}_t = \partial g(\boldsymbol{u}_t; \boldsymbol{w}_t)/\partial \boldsymbol{w}_t\big|_{\boldsymbol{w}_t = \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1}}$. Hence, we can easily derive the expectation of (49) resulting in

$$\hat{y}_t = \mathrm{E}[y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1}, \boldsymbol{u}_t] = g(\boldsymbol{u}_t; \boldsymbol{w}^{\mathrm{FS}}_{t-1|t-1}). \tag{51}$$

This solution is intuitively appealing since it represents the obvious way of predicting a new outcome, namely basing the prediction on the most recent weight vector available.

3.1.2 Sequential Learning Based on Posterior Means

As stated above, in the linear Gaussian case the state prediction and £ltering densities (12)-(13) are Gaussian and the linear Kalman £lter algorithm propagates the £rst two moments as suf£cient statistics of these Gaussians using a sequential recursive scheme. For nonlinear systems, however, the posteriors are not of well-known type and they cannot be calculated analytically. The strategy from the last section was to propagate mode and local curvature of the posterior weight distribution as centers and widths of a Gaussian approximation. The idea in this section is to propagate the £rst two moments of the £ltering density (13), i.e. *mean* and *covariance*, in a sequential fashion, given the £rst two moments of an *approximate* state prediction density (12). The £ltering density (13) at time $t-1$ can be characterized by its mean $\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}$ and covariance $\boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1}$. However, it cannot be fully characterized by the £rst two moments as suf£cient statistics like in the Gaussian case. But if we approximate this density to be Gaussian with the £rst two moments of the £ltering density, i.e.

$$\boldsymbol{w}_{t-1}|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1} \sim \mathcal{N}(\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1}) \tag{52}$$

then we obtain an *approximate* state prediction density (12)

$$p(\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1}) \approx \int p(\boldsymbol{w}_t|\boldsymbol{w}_{t-1})\, \mathcal{N}(\boldsymbol{w}_{t-1}|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1})\, d\boldsymbol{w}_t \tag{53}$$

which can be evaluated analytically (due to the linear Gaussian transition equation (8)) resulting in the approximate state prediction distribution

$$\boldsymbol{w}_t|\boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1} \sim \mathcal{N}(\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t). \tag{54}$$

In order to obtain the £rst two moments of the £ltering density at time step t we have to solve

$$\begin{aligned} \boldsymbol{w}^{\mathrm{m}}_{t|t} &= \int \boldsymbol{w}_t\, p(\boldsymbol{w}_t|\boldsymbol{Y}_t, \boldsymbol{U}_t)\, d\boldsymbol{w}_t \\ &= \frac{\int \boldsymbol{w}_t p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t)\, d\boldsymbol{w}_t}{\int p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t)\, d\boldsymbol{w}_t} \end{aligned} \tag{55}$$

for deriving the posterior mean and

$$\begin{aligned}\boldsymbol{\Sigma}^{\mathrm{m}}_{t|t} &= \int \boldsymbol{w}_t\boldsymbol{w}_t^{\top} p(\boldsymbol{w}_t|\boldsymbol{Y}_t, \boldsymbol{U}_t)\, d\boldsymbol{w}_t - \boldsymbol{w}^{\mathrm{m}}_{t|t}\boldsymbol{w}^{\mathrm{m}\,\top}_{t|t} \\ &= \frac{\int \boldsymbol{w}_t\boldsymbol{w}_t^{\top} p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t)\, d\boldsymbol{w}_t}{\int p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t) d\boldsymbol{w}_t} - \boldsymbol{w}^{\mathrm{m}}_{t|t}\boldsymbol{w}^{\mathrm{m}\,\top}_{t|t} \qquad (56)\end{aligned}$$

for obtaining the posterior covariance at time step t. Both integrals can be calculated easily from

$$\boldsymbol{I}(\boldsymbol{\alpha}(\boldsymbol{w}_t)) = \int \boldsymbol{\alpha}(\boldsymbol{w}_t)\, p(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t)\, d\boldsymbol{w}_t \qquad (57)$$

using the identities (Hennevogl 1991; Schnatter 1992)

$$\boldsymbol{w}^{\mathrm{m}}_{t|t} = \frac{\boldsymbol{I}(\boldsymbol{w}_t)}{\boldsymbol{I}(1)}, \qquad \boldsymbol{\Sigma}^{\mathrm{m}}_{t|t} = \frac{\boldsymbol{I}(\boldsymbol{w}_t\boldsymbol{w}_t^{\top})}{\boldsymbol{I}(1)} - \boldsymbol{w}^{\mathrm{m}}_{t|t}\boldsymbol{w}^{\mathrm{m}\,\top}_{t|t}. \qquad (58)$$

In the context of dynamic generalized linear models Schnatter (1992) proposes to compute these moments with numerical Gauss-Hermite integration. As stated above, due to the exponentionally increasing effort in high dimensions, this approach is computationally intractable for neural network applications. Instead, we follow the approach by Hennevogl (1991) and make use of importance sampling (MacKay 1999) to derive a Monte Carlo estimate of $\boldsymbol{I}(.)$. Since we use importance sampling in a sequential fashion, each time step with *new* normalized weights, we can circumvent the problems generally associated with importance sampling (Hennevogl 1991; MacKay 1999). In order to keep the computational effort low, we choose a Gaussian importance density $q(.)$ with centers and widths derived from the extended Kalman filter corrector step (18)-(20), i.e.

$$q \sim \mathcal{N}(\boldsymbol{w}_{t|t}, \boldsymbol{\Sigma}_{t|t}). \qquad (59)$$

Hence, importance sampling yields to the solution

$$\boldsymbol{w}^{\mathrm{m}}_{t|t} = \sum_{s=1}^{S} \frac{c(y_t|\boldsymbol{u}_t, \boldsymbol{w}^{s}_{t|t})}{\sum_{\tilde{s}=1}^{S} c(y_t|\boldsymbol{u}_t, \boldsymbol{w}^{\tilde{s}}_{t|t})}\, \boldsymbol{w}^{s}_{t|t} = \sum_{s=1}^{S} \tilde{c}^{s} \boldsymbol{w}^{s}_{t|t} \qquad (60)$$

for the filtered mean and to

$$\boldsymbol{\Sigma}^{\mathrm{m}}_{t|t} = \sum_{s=1}^{S} \tilde{c}^{s} \boldsymbol{w}^{s}_{t|t}\boldsymbol{w}^{s\,\top}_{t|t} - \boldsymbol{w}^{\mathrm{m}}_{t|t}\boldsymbol{w}^{\mathrm{m}\,\top}_{t|t} \qquad (61)$$

for the covariance of the filtering density at time step t. $\boldsymbol{w}^{s}_{t|t}, s = 1, \ldots, S$ are i.i.d. samples from the Gaussian importance density $q(.)$. The weighting factors $c(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t)$ are defined as

$$c(y_t|\boldsymbol{u}_t, \boldsymbol{w}_t) = \frac{\mathcal{N}\big(y_t|g(\boldsymbol{u}_t; \boldsymbol{w}_t), \sigma^2_{v_t}\big)\, \mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}, \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t)}{\mathcal{N}(\boldsymbol{w}_t|\boldsymbol{w}_{t|t}, \boldsymbol{\Sigma}_{t|t})}. \qquad (62)$$

With importance sampling we can evaluate the first two moments of the filtering density *exactly* given a sufficient amount of samples. The computational effort is low since we do not have to propagate a set of (high-dimensional) samples for describing the filtering density, rather we keep track only of mean and covariance estimates. The overall *approximate predictor conditional mean filter* algorithm is given below (with the normalized weighting factors $\tilde{c}^s$ and $\boldsymbol{w}^s_{t|t}$ as defined above).

Algorithm approximate predictor conditional mean filter based online learning

Initialization: $\boldsymbol{w}^{\mathrm{m}}_{0|0} = \boldsymbol{a}_0, \quad \boldsymbol{\Sigma}^{\mathrm{m}}_{0|0} = \mathbf{Q}_0$

$t = 1, 2, \ldots$

Predictor step:

$$\begin{aligned} \boldsymbol{w}_{t|t-1} &= \boldsymbol{w}^{\mathrm{m}}_{t-1|t-1} && (63)\\ \boldsymbol{\Sigma}_{t|t-1} &= \boldsymbol{\Sigma}^{\mathrm{m}}_{t-1|t-1} + \mathbf{Q}_t && (64) \end{aligned}$$

Corrector step:

$$\begin{aligned} \boldsymbol{w}_{t|t} &= \boldsymbol{w}_{t|t-1} + \mathbf{K}_t\big(y_t - \boldsymbol{g}(\boldsymbol{u}_t; \boldsymbol{w}_{t|t-1})\big) && (65)\\ \boldsymbol{\Sigma}_{t|t} &= \boldsymbol{\Sigma}_{t|t-1} - \mathbf{K}_t \mathbf{G}_t \boldsymbol{\Sigma}_{t|t-1} && (66)\\ \mathbf{K}_t &= \boldsymbol{\Sigma}_{t|t-1} \mathbf{G}_t^{\top} \big(\mathbf{G}_t \boldsymbol{\Sigma}_{t|t-1} \mathbf{G}_t^{\top} + \sigma^2_{v_t}\big)^{-1} && (67)\\ \boldsymbol{w}^{\mathrm{m}}_{t|t} &= \sum_{s=1}^{S} \tilde{c}^s \boldsymbol{w}^s_{t|t} && (68)\\ \boldsymbol{\Sigma}^{\mathrm{m}}_{t|t} &= \sum_{s=1}^{S} \tilde{c}^s \boldsymbol{w}^s_{t|t} {\boldsymbol{w}^s_{t|t}}^{\top} - \boldsymbol{w}^{\mathrm{m}}_{t|t} {\boldsymbol{w}^{\mathrm{m}}_{t|t}}^{\top} && (69) \end{aligned}$$

Step-one predictions of the outcome y_t are derived the same way as for the Fisher scoring approximation, i.e. by evaluating the expectation of (49), resulting in

$$\hat{y}_t = \mathrm{E}[y_t | \boldsymbol{Y}_{t-1}, \boldsymbol{U}_{t-1}, \boldsymbol{u}_t] = g(\boldsymbol{u}_t; \boldsymbol{w}^{\mathrm{m}}_{t-1|t-1}). \qquad (70)$$

Independently from this work, de Freitas et al. (1998, 1999) have also addressed sequential learning in dynamic neural regression models. They take a full-flash Bayesian approach for solving both, the state prediction density (12) and the filtering density (13) by Monte Carlo integration with a sequential version of importance sampling. The computational load of their method is significantly higher, since first, they propagate samples rather than sufficient statistics, and secondly, they have to update *each* single sample by application of extended Kalman filtering. Since step-one predictions in their approach are also only based on posterior modes or posterior means — rather than evaluating the integral (49) using Monte Carlo integration — the usefulness of propagating and updating a huge amount of samples is not obvious and their method appears to be a computational overkill. In effect, as our experiments will show, the two approximate Bayesian approaches are quite competitive to their approach, while limiting the computational resources to a reasonable level.

3.2 Handling Additive Measurement Outliers

The commonly used assumption in neural network regression modeling is that output measurements y_t are disturbed by independent additive Gaussian noise terms. Although one can derive the Gaussian noise assumption based on a maximum entropy approach (Deco and Brauer 1995), the main reason for employing this noise model is practicability. Under the Gaussian noise assumption standard online- or offline backpropagation training can be performed by minimization of squared errors. Despite its common use it is far from clear that the Gaussian noise assumption is a good choice for many practical problems. The most common reason why researchers depart from the Gaussian noise assumption is the presence of additive measurement outliers. Outliers are errors which occur with low probability and which are not generated by the data-generation process that is subject to identification. The general problem is that a few (maybe even one) outliers of high leverage are sufficient to throw the standard Gaussian error estimators completely off-track (Rousseeuw and Leroy 1987). Dealing with additive measurement outliers is of critical importance for online learning tasks. Here one is interested in avoiding inappropriate weight changes due to measurement outliers to maintain stable online learning capability. Outliers might result in highly fluctuating weights and possibly even instability when estimating the neural network weight vector online using a Gaussian error assumption. Advanced neural online learning algorithms like the extended Kalman filter sequential learning method from Section 3.1, for instance, are known to be nonrobust against such outliers (Meinhold and Singpurwalla 1989) since they are based on a Gaussian error assumption. In this section we follow the approach of Meinhold and Singpurwalla (1989), Lange et al. (1989) and Fahrmeir and Künstler (1999), among others, and replace the Gaussian noise assumption with a “heavy-tailed” noise model, i.e. a distribution which has longer than normal tails, to account for additive measurement outliers in dynamic neural regression models.

According to Lange et al. (1989) and Fahrmeir and Künstler (1999), we replace the Gaussian noise model by a more flexible model based on the *Student-t-distribution*. The t-distribution contains two free parameters — the degrees of freedom ν and a width parameter σ_v^2. A nice feature of the t-distribution is that if the degrees of freedom approach infinity, we recover the Gaussian noise model. If $\nu < \infty$ we obtain distributions which are more heavy-tailed than the Gaussian distribution including the Cauchy noise model with $\nu = 1$. The latter is commonly used for robust regression. Some researchers report good modeling capabilities when using the t-distribution as measurement noise model, even if the data from the underlying relationship is not generated using this distribution (West 1981). Surely, there are many different ways to define heavy-tailed noise models but we focus on the t-density due to its desirable properties (Lange et al. 1989) and due to the fact that it naturally extends the Gaussian noise model towards a robust error measure.

In the following we modify the Fisher scoring approach from Section 3.1.1 and derive a *robust* online learning scheme for dynamic neural regression models. The related problem of robust maximum likelihood learning in the neural regression model (4) for offline environments with batch data sets is addressed in the article by Briegel and Tresp (1999c).

3.2.1 Robust Online Learning based on Posterior Modes

The Gaussian noise assumption $v_t \sim \mathcal{N}(0, \sigma_{v_t}^2)$ from (15) is replaced by the Student-t-distribution with probability density function

$$p_{v_t}(z) = \frac{\Gamma\left(\frac{\nu_{v_t}+1}{2}\right)}{\sigma_{v_t}\sqrt{\pi\nu_{v_t}}\,\Gamma\left(\frac{\nu_{v_t}}{2}\right)} \left(1 + \frac{z^2}{\sigma_{v_t}^2 \nu_{v_t}}\right)^{-\frac{\nu_{v_t}+1}{2}}, \quad \nu_{v_t}, \sigma_{v_t} > 0. \tag{71}$$

It is immediately apparent that for $\nu_{v_t} = 1$ we recover the heavy-tailed Cauchy distribution. What is not so obvious is that for $\nu_{v_t} \to \infty$ we obtain a Gaussian distribution. To compare different noise models it is useful to evaluate the "ψ-function" defined as (Huber 1964)

$$\psi(z) = -\partial \log p_{v_t}(z)/\partial z, \tag{72}$$

that is the negative score function of the noise density. In the case of i.i.d. samples the ψ-function reflects the influence of a single measurement on the resulting estimator. Assuming Gaussian measurement errors $p_{v_t} \sim \mathcal{N}(0, \sigma^2_{v_t})$ we derive

$$\psi(z) = z/\sigma^2_{v_t} \tag{73}$$

which means that for $|z| \to \infty$ a single outlier z can have an infinite leverage on the estimator. In contrast, for constructing robust estimators West (1981) states that large outliers should not have *any* influence on the estimator, i.e. $\psi(z) \to 0$ for $|z| \to \infty$. For the t-distribution (71) we obtain

$$\psi(z) = \frac{\nu_{v_t} + 1}{\nu_{v_t} + z^2/\sigma^2_{v_t}} \frac{z}{\sigma^2_{v_t}}. \tag{74}$$

When comparing the two ψ-functions above one will recognize that in the latter outliers are weighted down. The degrees of freedom determine how much "weight"

$$\alpha_t = \frac{\nu_{v_t} + 1}{\nu_{v_t} + z^2/\sigma^2_{v_t}} \tag{75}$$

outliers obtain in influencing the regression estimate. For large positive or negative arguments z the weights α_t approach zero. Hence, the condition $\psi(z) \to 0$ for $|z| \to \infty$ is satisfied here. For $\nu_{v_t} \to \infty$ the weights α_t approach one and we recover the Gassian ψ-function. Figure 3 (left) shows $\psi(z)$ for different ν_{v_t} for the Student-t-distribution.

Using the t-distribution (71) we can modify the cost function (27) leading to the penalized log-likelihood criterion

$$\begin{aligned} pll(\boldsymbol{W}_t) \quad \propto \quad & -\sum_{s=1}^{t} \log p_{v_s}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big) + \frac{1}{2}(\boldsymbol{w}_0 - \boldsymbol{a}_0)^\top \mathbf{Q}_0^{-1}(\boldsymbol{w}_0 - \boldsymbol{a}_0) \\ & + \frac{1}{2}\sum_{s=1}^{t}(\boldsymbol{w}_s - \boldsymbol{w}_{s-1})^\top \mathbf{Q}_s^{-1}(\boldsymbol{w}_s - \boldsymbol{w}_{s-1}) \end{aligned} \tag{76}$$

which is subject to optimization with respect to $\boldsymbol{W}_t$. For applying the Fisher scoring algorithm from 3.1.1 to minimize $pll(.)$ we need the negative score function and the expected information for the t-density. The negative score function is given by the above ψ-function and for the expected information we obtain (Lange et al. 1989)

$$\partial\psi/\partial z = \frac{\nu_{v_t} + 1}{\nu_{v_t} + 3}\sigma^{-2}_{v_t}. \tag{77}$$

We can apply the same algorithms as in the Gaussian case in 3.1.1 to obtain *robust* online posterior mode weight estimates by substituting the measurement variance $\sigma^{-2}_{v_s}$ in (33) and (44) by $(\nu_{v_s} + 1)/(\nu_{v_s} + 3)\sigma^{-2}_{v_s}$ and by substituting the Gaussian ψ-function $\sigma^{-2}_{v_s}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)$ with the t-density ψ-function

$$\frac{\nu_{v_t} + 1}{\nu_{v_t} + \big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2/\sigma^2_{v_t}}\sigma^{-2}_{v_s}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big) \tag{78}$$

to evaluate the score function (38).

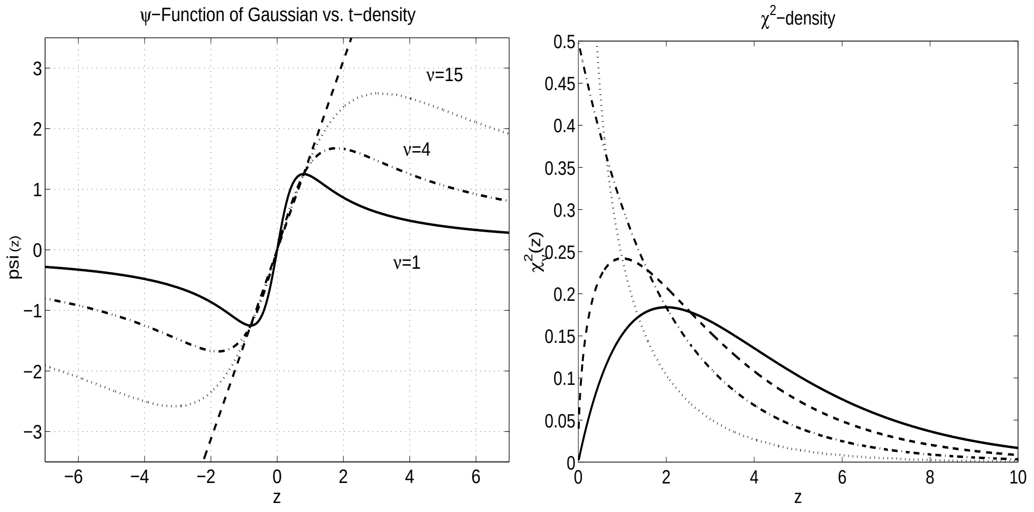

Figure 3. Left: ψ-functions for the Gaussian density (dashed) and the t-density with $\nu_{v_t} = 1, 4, 15$ degrees of freedom. The Gaussian ψ-function is unbounded whereas the ψ-function for the t-density is bounded and approaches zero for large arguments z. **Right:** Chi-square density with $\nu = 1, 2, 3, 4$ degrees of freedom. The smaller the degrees of freedom ν, the more probability mass lies in the interval $(0, 1]$ (dotted: $\nu = 1$ and dash-dotted: $\nu = 2$) forcing robust error distributions. For higher degrees of freedom (dashed: $\nu = 3$ and solid: $\nu = 4$) there is a shift of the probability mass towards larger arguments $z > 1$ forcing non-robust error distributions with the extreme case of the Gaussian error distribution for $\nu = \infty$. When picking two particular arguments $z = 1$ and $z < 1$ of the Chi-square distribution and by assigning probabilities π and $1 - \pi$ to these arguments we obtain Ormoneit and Neuneier's binary Gaussian mixture approach as a special case.

3.2.2 Hyperparameter Estimation

So far, we have assumed that the scale factor $\sigma^2_{v_t}$ and the degrees of freedom ν_{v_t} of the t-density are known. Given a suf£cient amount of batch data these hyperparameters can be estimated in an of¤ine manner with the maximum likelihood method when assuming constant values, i.e. $\sigma^2_{v_t} = \sigma^2_v, \nu_{v_t} = \nu_v$. For performing this task Briegel and Tresp (1999c) developed an EM algorithm by extending arguments given in Lange et al. (1989) to neural network of¤ine regression models. If batch data is not available in advance, another option is to set these hyperparameters to some reasonable values. Lange et al. (1989) use the value $\nu_v = 4$ for the degrees of freedom as a "sensible guesstimate" which works well in many applications. In order to account for changing dynamics in the underlying data generation process we apply results from Lange et al. (1989) and Fahrmeir and Künstler (1999) to our dynamic neural regression model and develop an *online EM-type algorithm* for approximate maximum likelihood estimation of the scale factors and the degrees of freedom of the t-density. The idea is to treat scale factors and degrees of freedom as constant values within a sliding time window of length $\tilde{\tau}_t$, e.g. $\sigma^2_{v_t} = \sigma^2_v, \nu_{v_t} = \nu_v, t \in \{t - \tilde{\tau}_t, t\}$ to account for slowly varying process noise dynamics. Note the difference that is made here between the sequential Bayesian approach for the neural network weight vectors which are treated as random variables and the hyperparameters which are assumed to be constant over a certain time window. For deriving the online EM-type algorithm it is important to note that the t-density can be thought of as an in£nite mixture of Gaussians with probability density function (Andrews and

Mallows 1974)

$$p_\nu(z) = \int \mathcal{N}(z|0, \sigma^2/\tilde{u})\, p(\tilde{u})\, d\tilde{u}. \tag{79}$$

Here, $p(\tilde{u})$ denotes the probability density function of the mixing variable $\tilde{u}$. The mixing variable is Chi-square distributed, i.e. $\tilde{u} \sim \chi^2_\nu/\nu$ where χ^2_ν denotes the Chi-square distribution with ν degrees of freedom. The probability density function of the Chi-square distribution is de£ned as

$$p_{\chi^2}(z) = \frac{\nu}{2^{\nu/2}\Gamma(\frac{\nu}{2})} \big(\nu z\big)^{\nu/2-1} \exp\big(-\frac{\nu z}{2}\big)\,, \quad z > 0. \tag{80}$$

This mixture approach can be seen as a continuous extension to the discrete Gaussian mixture model with a binary mixing variable $\tilde{u}$ as employed by Ormoneit and Neuneier (1995) for obtaining robust of¤ine neural regression estimates. In their method, the binary mixing variable $\tilde{u}$ takes the value 1 with probability π and the value $\lambda < 1$ with probability $1 - \pi$. The “narrow” Gaussian $\mathcal{N}(z|0, \sigma^2)$ ($\tilde{u} = 1$) represents the distribution of the non-outlier part of the data. The “wider” Gaussian $\mathcal{N}(z|0, \sigma^2/\lambda)$ ($\tilde{u} = \lambda$) accounts for the assumption that some data are located at larger distances from the prediction. The fact that outliers are basically exceptions is re¤ected by an appropriate choice of the mixture probabilities π and $1 - \pi$, which can be regarded as prior probabilities for each Gaussian (Ormoneit and Neuneier 1995). This binary mixture approach has two basic drawbacks compared to the in£nite mixture approach. First, there is an additional hyperparameter to estimate and second, evaluation of the ψ-function and the expected information matrix is computationally demanding. In particular, the expected information matrix can only be derived using numerical integration methods in contrast to the expected information matrix for the t-density, which is given by (77).

Using the continuous mixture approach, we obtain the observation distribution for a single data tupel $(\boldsymbol{u}_t, y_t)$ as

$$y_t|\boldsymbol{u}_t, \boldsymbol{w}_t, \tilde{u}_t \sim \mathcal{N}\big(g(\boldsymbol{u}_t; \boldsymbol{w}_t), \sigma_v^2/\tilde{u}_t\big) \tag{81}$$

where $\tilde{u}_t \sim \chi^2_{\nu_v}/\nu_v$. Outliers are modeled by “small” values of the mixing variable $\tilde{u}_t$ leading to wider Gaussians $\mathcal{N}(z|0, \sigma^2/\tilde{u}_t)$ whereas an “uncontaminated” data tupel is represented by a “large” value $\tilde{u}_t$ implying a narrow Gaussian. The widths of the Gaussians are determined by the degrees of freedom of the Chi-square distribution (see Figure 3 right).

With the above representation for the t-distribution we can treat the weight sequence $\boldsymbol{w}_t$ together with the mixture variables $\tilde{u}_t$ as missing for deriving the online EM-type algorithm. The E-step is given by

$$Q(\boldsymbol{\phi}, \boldsymbol{\phi}^{\text{old}}) = \mathrm{E}\big[\log p\big(\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \tilde{\boldsymbol{U}}_{t-\tilde{\tau}_t,t}, \boldsymbol{W}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}\big)\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\text{old}}\big] \tag{82}$$

with $\tilde{\boldsymbol{U}}_{t-\tilde{\tau}_t,t} = (\tilde{u}_{t-\tilde{\tau}_t}, \ldots, \tilde{u}_t)^\top$ as the sequence of mixing variables and $\boldsymbol{\phi}^{\text{old}} = (\sigma_v^{2,\text{old}}, \nu_v^{\text{old}})^\top$ as the vector of unknown hyperparameters to be estimated. The expectation can be decomposed into

$$\begin{aligned} Q(\boldsymbol{\phi}, \boldsymbol{\phi}^{\text{old}}) = \mathrm{E}\big[&\log p\big(\boldsymbol{Y}_{t-\tilde{\tau}_t,t}\big|\tilde{\boldsymbol{U}}_{t-\tilde{\tau}_t,t}, \boldsymbol{W}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}\big) \\ &+ \log p\big(\tilde{\boldsymbol{U}}_{t-\tilde{\tau}_t,t}\big) + \log p\big(\boldsymbol{W}_{t-\tilde{\tau}_t,t}\big)\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\text{old}}\big]. \end{aligned} \tag{83}$$

For deriving the maximum likelihood estimate for the scaling parameter σ_v^2 we have to solve the equation

$$\frac{\partial Q}{\partial \sigma_v} = \frac{\partial}{\partial \sigma_v} \sum_{s=t-\tilde{\tau}_t}^{t} \mathrm{E}\big[\log p(y_s|\tilde{u}_s, \boldsymbol{w}_s, \boldsymbol{u}_s)\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big] = 0. \qquad (84)$$

By inserting the model assumption (81) into (84) we obtain — after dropping constant terms

$$\sum_{s=t-\tilde{\tau}_t}^{t} \mathrm{E}\big[\log p(y_s|\tilde{u}_s, \boldsymbol{w}_s, \boldsymbol{u}_s)\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big] \propto$$
$$-\tilde{\tau}_t \log \sigma_v - \frac{1}{2\sigma_v^2} \sum_{s=t-\tilde{\tau}_t}^{t} \mathrm{E}\big[\tilde{u}_s\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big]. \qquad (85)$$

Applying the law of iterated expectations (Bar-Shalom and Li 1993) results in the expression

$$\mathrm{E}\big[\tilde{u}_s\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big] =$$
$$\mathrm{E}\Big[\mathrm{E}\big[\tilde{u}_s\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2\big|\boldsymbol{W}_{t-\tilde{\tau}_t,t}, \boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big]\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\Big] =$$
$$\mathrm{E}\big[\alpha_s\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big] \qquad (86)$$

where α_s

$$\alpha_s = \frac{\nu^{\mathrm{old}} + 1}{\nu^{\mathrm{old}} + \delta_s}, \quad \delta_s = \frac{\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2}{\sigma_v^{2,\mathrm{old}}} \qquad (87)$$

is the expected value of the unknown mixing component $\tilde{u}_s$ given the data point $(\boldsymbol{u}_s, y_s)$ *and* the weight vector $\boldsymbol{w}_s$ at time s (Lange et al. 1989). δ_s can be approximated by substituting the Fisher scoring solutions $\boldsymbol{w}_{s|t}^{\mathrm{FS}}$ for $\boldsymbol{w}_s$ leading to *approximate* weights

$$\alpha_{s|t} = \frac{\nu^{\mathrm{old}} + 1}{\nu^{\mathrm{old}} + \delta_{s|t}}, \quad \delta_{s|t} = \frac{\big(y_s - g(\boldsymbol{u}_t; \boldsymbol{w}_{s|t}^{\mathrm{FS}})\big)^2}{\sigma_v^2}. \qquad (88)$$

Substituting the weights α_s with the approximate weights $\alpha_{s|t}$ and applying a first order Taylor series expansion of $g(.; \boldsymbol{w}_s)$ about the posterior mode $\boldsymbol{w}_{s|t}^{\mathrm{FS}}$, i.e.

$$g(\boldsymbol{u}_s; \boldsymbol{w}_s) \doteq g(\boldsymbol{u}_s; \boldsymbol{w}_{s|t}^{\mathrm{FS}}) + \mathbf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})(\boldsymbol{w}_s - \boldsymbol{w}_{s|t}^{\mathrm{FS}}) \qquad (89)$$

leads to the approximate E-step equation for the scaling parameter σ_v^2

$$Q(\sigma_v^2, \boldsymbol{\phi}^{\mathrm{old}}) = -\tilde{\tau}_t \log \sigma_v - \frac{1}{2\sigma_v^2} \sum_{s=t-\tilde{\tau}_t}^{t} \mathrm{E}\big[\alpha_{s|t}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_s)\big)^2\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t}, \boldsymbol{U}_{t-\tilde{\tau}_t,t}, \boldsymbol{\phi}^{\mathrm{old}}\big] \qquad (90)$$
$$= -\tilde{\tau}_t \log \sigma_v - \frac{1}{2\sigma_v^2} \sum_{s=t-\tilde{\tau}_t}^{t} \alpha_{s|t}\big(y_s - g(\boldsymbol{u}_s; \boldsymbol{w}_{s|t}^{\mathrm{FS}})\big)^2 + \mathbf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})\boldsymbol{\Sigma}_{s|t}\mathbf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})^\top \qquad (91)$$

after substituting posterior means $\mathrm{E}[\boldsymbol{w}_s|\boldsymbol{Y}_{t-\tilde{\tau}_t,t},\boldsymbol{U}_{t-\tilde{\tau}_t,t}]$ with posterior modes $\boldsymbol{w}_{s|t}^{\mathrm{FS}}$ and posterior covariances $\mathrm{cov}[\boldsymbol{w}_s|\boldsymbol{Y}_{t-\tilde{\tau}_t,t},\boldsymbol{U}_{t-\tilde{\tau}_t,t}]$ with curvatures $\boldsymbol{\Sigma}_{s|t}$. Setting the £rst order derivatives of (91) with respect to σ_v to zero results in the M-step equation

$$\sigma_v^{2,\mathrm{new}} = \frac{1}{\tilde{\tau}_t}\sum_{s=t-\tilde{\tau}_t}^{t} \alpha_{s|t}\Big[\big(y_s - g(\boldsymbol{u}_s;\boldsymbol{w}_{s|t}^{\mathrm{FS}})\big)^2 + \mathsf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})\boldsymbol{\Sigma}_{s|t}\mathsf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})^\top\Big] \tag{92}$$

where $\mathsf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}})$ denotes the Jacobian of the neural network, i.e.

$$\mathsf{G}_s(\boldsymbol{w}_{s|t}^{\mathrm{FS}}) = \partial g(\boldsymbol{u}_s;\boldsymbol{w}_s)/\partial \boldsymbol{w}_s\big|_{\boldsymbol{w}_s=\boldsymbol{w}_{s|t}^{\mathrm{FS}}}. \tag{93}$$

For a t-density with in£nite degrees of freedom the weights approach one ($\alpha_{s|t}=1$) and the update rule degenerates to the update equation for dynamic neural regression models with Gaussian error sequences. For robust online learning $\alpha_{s|t}$ determines how much weight an individual data tupel $(\boldsymbol{u}_s, y_s)$ obtains when computing the maximum likelihood estimate. If a speci£c data tupel is detected as an outlier, $\alpha_{s|t}$ should approach zero which means that this data tupel is discarded to compute the estimate. Uncontaminated data tupels, on the other hand, may be assigned a larger weight. Note that the weights $\alpha_{s|t}$ can be interpretated as an approximation to the weights (75) of the ψ-function (74).

The update rule for the degrees of freedom ν_v is obtained from solving

$$\frac{\partial Q}{\partial \nu_v} = \frac{\partial}{\partial \nu_v}\sum_{s=t-\tilde{\tau}_t}^{t} \mathrm{E}\big[\log p(\tilde{u}_s)\big|\boldsymbol{Y}_{t-\tilde{\tau}_t,t},\boldsymbol{U}_{t-\tilde{\tau}_t,t},\boldsymbol{\phi}^{\mathrm{old}}\big] = 0 \tag{94}$$

where $p(\tilde{u}_s)$ is given by the Chi-square density p_{χ^2} as de£ned in (80). Similar arguments[6] lead to the M-step optimization problem for the degrees of freedom (Lange et al. 1989)

$$\frac{\tilde{\tau}_t}{2} + \frac{\tilde{\tau}_t}{2}\log(\frac{\nu_v^{\mathrm{new}}}{2}) - \frac{\tilde{\tau}_t}{2}\mathrm{DG}(\frac{\nu_v^{\mathrm{new}}}{2}) + \frac{1}{2}\sum_{s=t-\tilde{\tau}_t}^{t}\beta_{s|t} - \frac{1}{2}\sum_{s=t-\tilde{\tau}_t}^{t}\alpha_{s|t} = 0 \tag{95}$$

where $\beta_{s|t} = \mathrm{DG}(\frac{\nu_v^{\mathrm{old}}+1}{2}) - \log\big(\frac{1}{2}(\nu_v^{\mathrm{old}} + \delta_{s|t})\big)$ with the Digamma function $\mathrm{DG}(z) = \partial\Gamma(z)/\partial z$. Ef£cient numerical algorithms exist for computing the Digamma function (Abramowitz and Stegun 1989). A new update for the degrees of freedom is obtained by solving the one-dimensional nonlinear optimization problem (95) for ν_v^{new} (Gill et al. 1981).

de Freitas et al. (1998) propose alternative sequential update equations for the measurement noise variance $\sigma_{v_t}^2$ and the transition covariance matrix Q_t which are either based on adopting MacKay's evidence framework (MacKay 1992) to the online case or by maximizing the posterior density function with respect to the hyperparameters in a recursive fashion.

4 Simulations and Experiments

In this section we present experiments to test the new methods with the Gaussian and the t-distributed error measures using some arti£cial and real-world data sets. More experiments and simulations — also with the robust neural of¤ine EM algorithm — can be found in the articles by Briegel and Tresp (1999b, 1999c).

[6] That is, applying the law of iterated expectations, a £rst order Taylor series expansion of the network about the posterior mode and substitution of posterior means and covariances with posterior modes and curvatures.

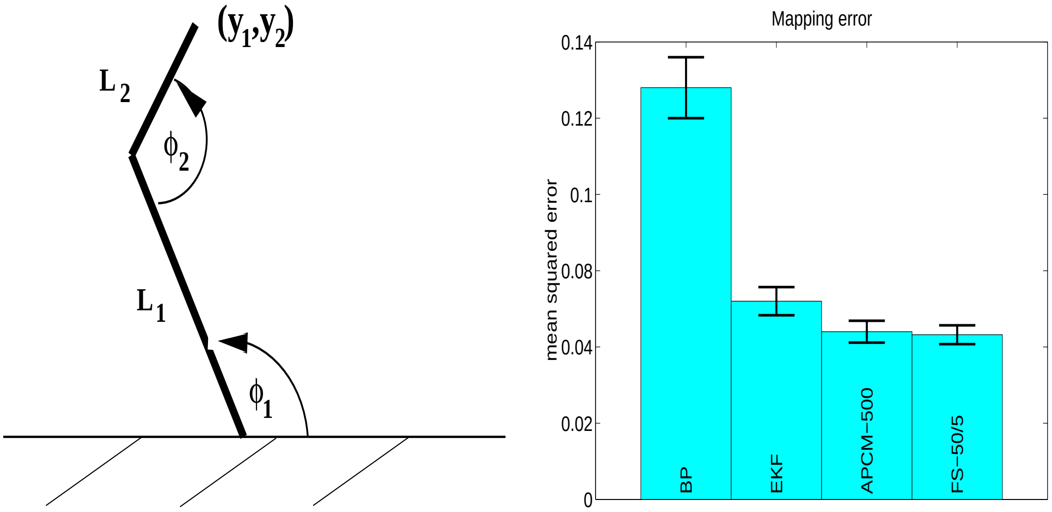

Figure 4. Left: Schematic illustration of the two link robot arm. **Right:** Mean squared errors on the underlying (unnoisy) transformations (96) and (97). The heights of the bars show average mean squared errors on a regular grid of the input space $[0.3, 1.2] \times [\pi/2, 3\pi/2]$.

4.1 Gaussian Measurement Errors

Experiment 1: Learning the Forward Kinematics of a Two-Link Robot Arm

As a first example of applying the two online learning procedures, we consider the forward kinematics of a simple two-link robot arm, as shown in Figure 4 (left). The robot arm data set is often used as a benchmark to compare neural network algorithms (MacKay 1992). For given values of the joint arm angles $\boldsymbol{\phi} = (\phi_1, \phi_2)^\top$, the end effector is moved to a position $\boldsymbol{y} = (y_1, y_2)^\top$ given by the cartesian coordinates

$$y_1 = L_1 \cos(\phi_1) - L_2 \cos(\phi_1 + \phi_2) \qquad (96)$$

$$y_2 = L_1 \sin(\phi_1) - L_2 \sin(\phi_1 + \phi_2) \qquad (97)$$

with $L_1 = 0.75$ and $L_2 = 0.3$ being the lengths of the two links of the robot arm. In the simulations, we restricted ϕ_1 to the range $[0.3, 1.2]$ and ϕ_2 to the range $[\pi/2, 3\pi/2]$. The mapping from (ϕ_1, ϕ_2) to (y_1, y_2) is known as the *forward kinematics*, and is single valued. A data set of length $T = 1000$ was generated by adding uncorrelated Gaussian noise $\mathbf{V}_t = 0.05\mathbf{I}_{2,2}$ to the simulated two-dimensional output values $\boldsymbol{y}_t = (y_{1,t}, y_{2,t})^\top$. The two-dimensional inputs $\boldsymbol{u}_t = (\phi_{1,t}, \phi_{2,t})^\top$ were drawn uniformly from the domain $[0.3, 1.2] \times [\pi/2, 3\pi/2]$.

The task was to learn the underlying (static) mapping (96)-(97) in an online manner using a MLP with twelve hidden units. Online backpropagation (BP) with a stepsize parameter $\eta = 0.1$ and extended Kalman filtering (EKF) again serve as benchmark methods. The noise covariances in the EKF approach were set to $\mathbf{Q}_t = 10^{-4}\mathbf{I}_{61,61}$ and to $\mathbf{V}_t = 0.05\mathbf{I}_{2,2}$ respectively. $\mathbf{Q}_0$ was set to $10\mathbf{I}_{61,61}$. The approximate predictor conditional mean filter (APCM) was assigned $S = 500$ samples $\boldsymbol{w}^s_{t|t}$ for approximating the conditional filtered means $\boldsymbol{w}^{\mathrm{m}}_{t|t}$ and covariances $\boldsymbol{\Sigma}^{\mathrm{m}}_{t|t}$. $\mathbf{Q}_t$ was set to $\mathbf{Q}_t = 10^{-3}\mathbf{I}_{61,61}$. Online posterior mode smoothing was performed using five forward-backward passes of the Fisher Scoring algorithm over a sliding time window of length $\tau_t = 50$

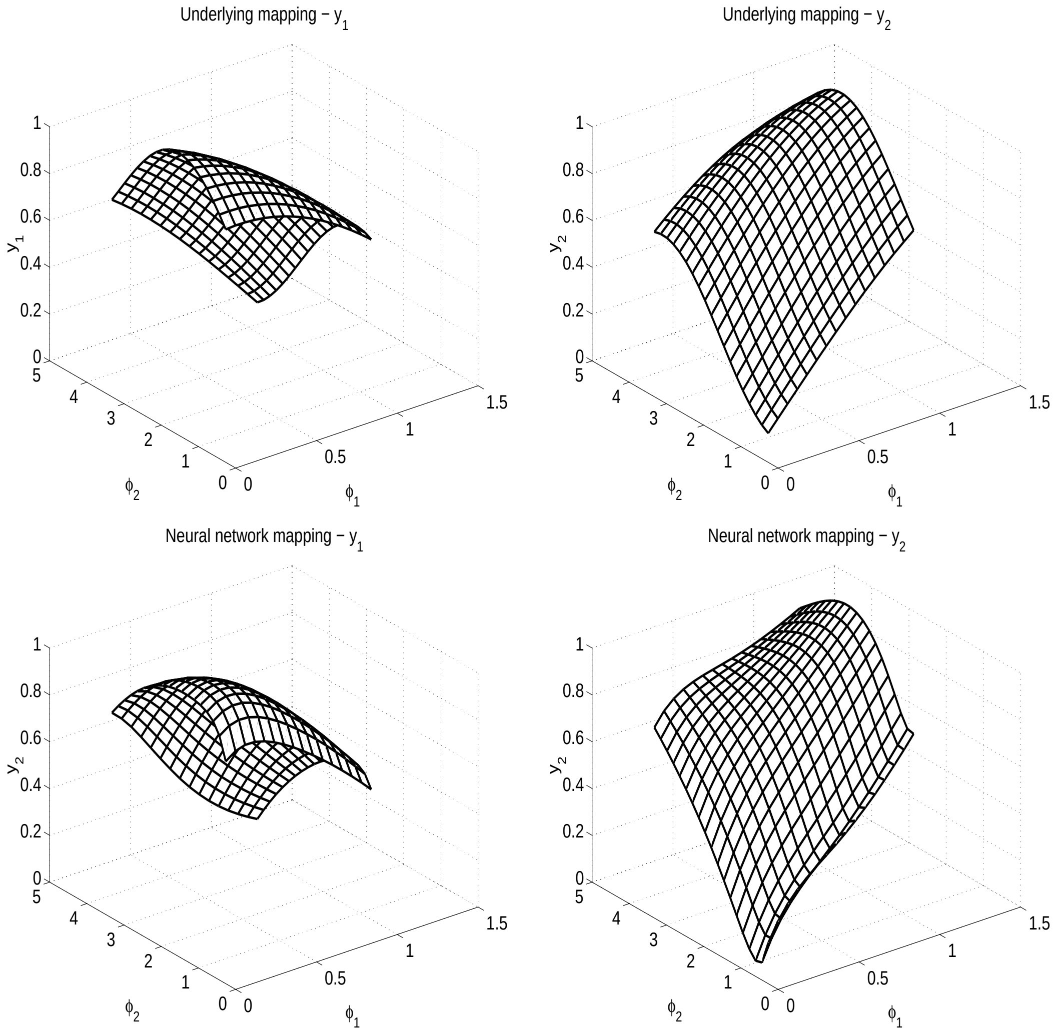

Figure 5. Original and neural network mappings after online adaptation with the Fisher scoring algorithm. The top plots show the original mappings (96) and (97) for the input space $[0.3, 1.2] \times [\pi/2, 3\pi/2]$. The bottom plots display the neural network mappings evaluated at weight vector $\boldsymbol{w}^{\mathrm{FS}}_{1000|1000}$ for the Fisher scoring online learning approach.

(FS-50/5). The weight transition noise covariance was set to $\mathbf{Q}_t = 10^{-4}\mathbf{I}_{61,61}$. The experiment was repeated 25 times, each time with new simulated data $\{\boldsymbol{u}_t, \boldsymbol{y}_t\}_{t=1}^{T}$ and new initial weights $\boldsymbol{w}_0$ drawn from a zero-mean Gaussian distribution with covariance $0.5\mathbf{I}_{61,61}$. Figure 4 (right) provides average mean squared errors between the neural networks and the underlying function on a regular grid of the input space and error bars corresponding to one standard deviation. The two new approaches outperform the two benchmark models significantly (based on the paired t-test with $\alpha = 0.01$) but are not significantly different (also based on the paired t-test with 1% significance level) indicating corresponding posterior means and modes. Figure 5 shows the mappings (96)-(97) and the neural network mappings on a regular grid on the input space after online adaptation for the Fisher scoring approach using the weight vector $\boldsymbol{w}^{\mathrm{FS}}_{1000|1000}$.

Experiment 2: Pricing Options on the FTSE-100 Index

Mathematical modeling of financial derivatives has become increasingly important over the past decade (Hull 1997). The availability of sophisticated pricing models for options contracts is of great interest to both researchers and practitioners in financial institutions. Option pricing imposes interesting and challenging problems to the modeler due to the nonlinear, non-stationary, and stochastic behavior of options prices and therefore provides an ideal means for testing the algorithms derived in Section 3.1. A new area of research, which has been pioneered by Hutchinson et al. (1994) and Niranjan (1997), is concerned with applying neural networks to estimate option pricing formulas, i.e. compact descriptions of the „fair” price of an asset whose value is contingent on the price of a second, underlying asset. The former showed that good approximations to the widely used Black-Scholes formula may be obtained with neural networks, while the latter was concerned with the non-stationary aspects of the problem. Niranjan (1997) uses the extended Kalman filter to sequentially propagate neural network weights in an online manner. de Freitas et al. (1998) apply a fully Bayesian approach based on sequential importance sampling, to estimate the network parameters online. Here, the two online algorithms (FS and APCM) are compared to different benchmark models, including the extended Kalman filter algorithm and the fully Bayesian approach by de Freitas et al. The Black-Scholes partial differential equation (Black and Scholes 1973) which is the main industry standard for pricing options (Hull 1997) takes into account the most important factors influencing the options price.[7] It relates the current value of an option to the current value of the underlying stock, the volatility of the stock, the strike price and time to maturity of the option, and to the risk free interest rate. This basic equation is, however, only valid under several conditions, e.g. no risk-less arbitrage opportunities, an instantaneous risk-less portfolio, constant volatility and risk-free interest rate. In addition, the stock's price is assumed to be dictated by a geometric Brownian motion model. The uncertainty arising from the above model assumptions may be avoided by applying a neural network model with sequential parameter adaptation. We follow the approach by de Freitas et al. (1998) and map the stock's price and time to maturity to the call and put options prices using a MLP. Hereby, the stock's price and the options prices are normalized by the strike price. In order to make results comparable we used the same data sets as de Freitas et al. (1998), five call option contracts on the FTSE-100 index from Febuary 1994 to November 1994 to train our neural pricing models.

In the experiments MLPs with four hidden units, two inputs (the stock's price and the time to maturity) and a single output (the options price) were used. The noise covariances for the extended Kalman filter approach (EKF) σ_v^2 and $\mathbf{Q}_t$ were set to 10^{-4} and to $10^{-5}\mathbf{I}_{15,15}$ respectively. The initial weight vector $\boldsymbol{w}_0$ was drawn from a zero-mean Gaussian distribution with a covariance equal to $\mathbf{I}_{15,15}$. $\mathbf{Q}_0$ was set to $10\mathbf{I}_{15,15}$. The approximate predictor conditional mean filter approach (APCM-100) was assigned $S = 100$ weight vector samples $\boldsymbol{w}^s_{t|t}$ to obtain Monte Carlo approximations of the filtered posterior means $\boldsymbol{w}^{\mathrm{m}}_{t|t}$ and covariance matrices $\boldsymbol{\Sigma}^{\mathrm{m}}_{t|t}$. $\mathbf{Q}_t$ was set to $10^{-3}\mathbf{I}_{15,15}$ to allow the algorithm to explore a large region of the weight space. The single Fisher scoring step algorithm (FS-30) derives online posterior weight updates over a time window of length $\tau_t = 30$. $\mathbf{Q}_t$ was set to $10^{-5}\mathbf{I}_{15,15}$. Figure 6 (left) shows the one-step ahead predictions to the call option with strike price $X = 2925$ obtained from the APCM approach. The right plot shows the results obtained from the FS approach for the the put option with strike price $X = 3325$. Due to the regularization part in the cost function of the FS approach, which penalizes rough weight

[7] The textbook by Hull (1997) provides a very readable introduction to options and other derivatives.

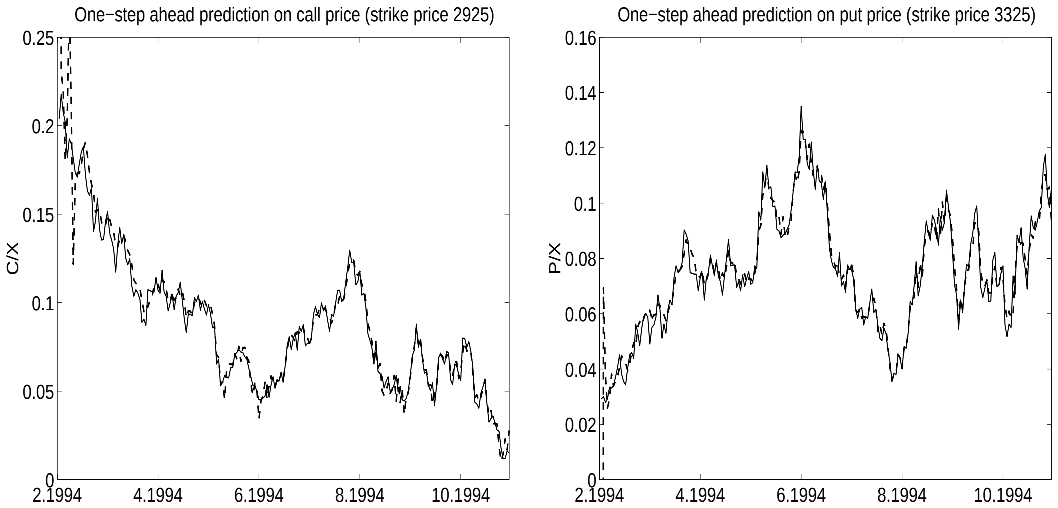

Figure 6. Left: The panel shows the call options data series with strike price $X = 2925$ (continuous). Dashed are shown the one-step ahead forecasts obtained from the APCM approach. **Right:** Shown is the put options data series with strike price $X = 3325$. The dashed line plots the step-one ahead predictions obtained from the FS approach. After having seen about one fourth of the options time series the weight sequences obtained from both algorithms start to converge and accurate step-one forecasts are obtained.

sequences, this approach yields to smoother step-one estimates than the APCM approach. With both algorithms very close fits to the measured data were obtained. Figure 7 shows the mean squared errors between the actual value of the options contract and the one-step ahead prediction for various methods. These errors are computed on data corresponding to the last 100 days of the data set allowing the algorithms to converge. Note that within the online setting considered here, all reported errors are computed on unseen data. For comparison purposes, mean squared errors reported in de Freitas et al. (1998) for different benchmark models are provided. The trivial method (TR) simply involves using the current value of the option as the next prediction, while the Black-Scholes method (BS) corresponds to a conventional Black-Scholes model. A fully Bayesian approach (SIS-100) uses sequential importance sampling with 100 samples of the weight vector per time step to approximate the state prediction and filtering densities. Details about the BS and the SIS-100 implementation can be found in de de Freitas et al. (1998). As one can see the errors obtained from FS and APCM are quite competitive to the errors obtained from the fully Bayesian approach. The reason might be, that the posterior densities are unimodal or sharply peaked about the mode evidencing also the corresponding errors of our two approaches. For high strike prices ($X = 3225$ and $X = 3325$) the results of the different neural network approaches are not significantly different but outperform the conventional Black-Scholes model clearly.

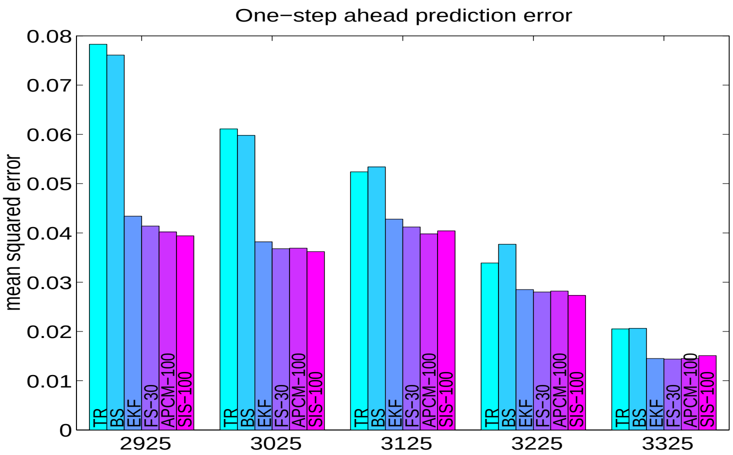

Figure 7. Mean squared errors on the five call options ($X = 2925$ to $X = 3325$) for six different methods. TR simply involves using the current value of the option as the next prediction. BS corresponds to a conventional Black-Scholes model and serves together with the EKF and a fully Bayesian approach by de Freitas et al. (SIS-100) as benchmarks to our approaches FS-30 and APCM-100. The heights of the error bars indicate that our approximate Bayesian approaches are quite competitive to the fully Bayesian approach by de Freitas et al.

4.2 Robust Sequential Learning

Experiment 1: One-Dimensional Toy Example

In this experiment the use of the t-distribution for a one-dimensional toy problem with additive outliers was examined. Data were generated from the following nonlinear FIR model

$$y_t = au_t^2 + b\sin(6u_t) - 1 + v_t\,, \quad t = 1, \ldots, 500 \tag{98}$$

with $a = -0.5$ and $b = 0.9$ and where u_t was uniformly drawn from the interval $[-1, 1]$. v_t denote uncorrelated zero-mean Gaussian noise terms with variance $\sigma_v^2 = 0.1$. With a probability of 15% outliers uniformly drawn from the interval $[-5, 5]$ were added to the measurements y_t. Figure 8 (left) shows the results obtained from two different training procedures using a MLP with four hidden units. The dash-dotted line plots the neural network model obtained from extended Kalman filtering based on Gaussian noise terms. The continuous line shows the nonlinearity in the model (98). The result with the Gaussian error model indicates that a few outliers of high leverage are sufficient to throw the extended Kalman filter based online learning method completely off-track. The dashed line displays the results obtained from the single-step Fisher scoring algorithm with a sliding time window of length $\tau_t = 30$ using a t-distributed error measure. The parameters of the t-density were initialized with $\nu = 10$ degrees of freedom and with a variance σ_v^2 of 1.0. Experiments with the EM-type algorithm suggest that — like in the offline learning framework for neural state space models — adaptation of hyperparameters should not start from the very beginning of the learning procedure since this can result in instability and high parameter fluctuations during the learning process. Here, adaptation of the hyperparameters was started after 200 measurements over a sliding time window of length $\tilde{\tau}_t = 150$. We obtained the estimates $\hat{\nu}_v = 5.162$ and $\hat{\sigma_v^2} = 0.1353$. Figure 8 (right) shows some of the weighting factors $\alpha_{s|t}$ which are used in the EM update equations for ν_v and σ_v^2. The numerical values of these weighting factors indicate the occurrence of outliers of large magnitude. Ideally, for a outlier of high leverage, $\alpha_{s|t}$

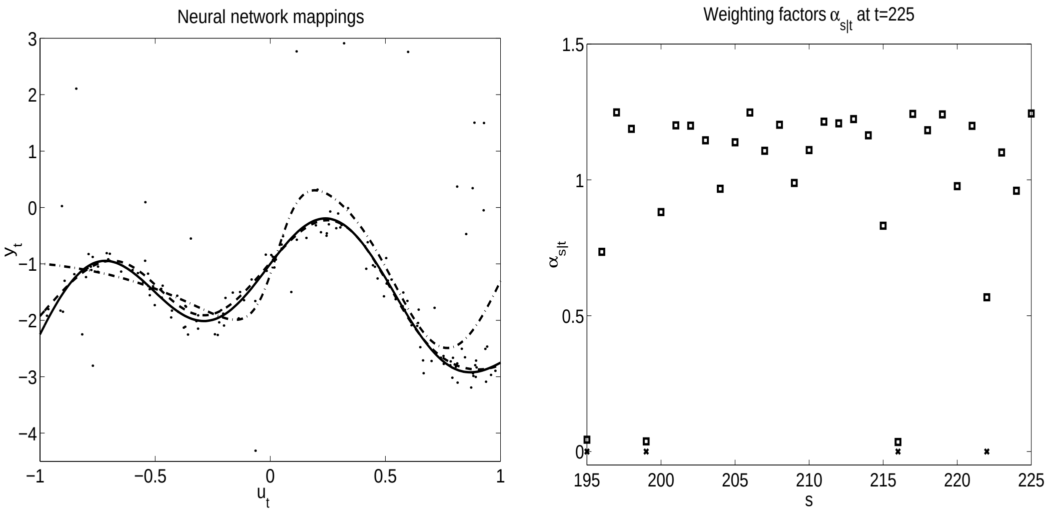

Figure 8. Left: 150 points of the data set including outliers (dots). Shown are the underlying mapping (continuous) and neural network based mappings: the EKF learned model with Gaussian measurement density (dash-dotted) and the model adapted with FS based on t-distributed measurement noise (dashed). **Right:** Weighting factors $\alpha_{s|t}$ at $t = 225$ (squares) which are used in the EM-type algorithm. The crosses mark additive outliers (of different magnitude) at time steps $t = 195, 198, 216$ and 222. Since extreme outliers should not be taken into account for estimating the hyperparameters, $\alpha_{s|t}$ should be close to zero for such values. The squares at time steps $t = 195, 198$ and 216 indicate that the EM-type algorithm was able to identify these measurements as extreme outliers. The measurement at time $t = 222$ on the other hand, was weighted down to a value of 0.57 but was not detected as an "extreme outlier" which turned out to be the "right decision" for that particular case.

should become zero since the outlier does not contribute at all to improving the estimate of the hyperparameter in this situation.

Experiment 2: Boston Housing Data

The Boston housing data originates with Harrison and Rubinfeld (1978) who were interested in the effect of air pollution on housing prices. Although Harrison & Rubinfeld's original focus was to obtain insight into factors affecting price, rather than to make predictions, the goal here is to derive accurate predictions on the housing prices based on certain attributes. The relationships between the attributes and the housing price are modeled by a MLP with six hidden units.

The data set concerns the median price in 1970 of owner-occupied houses in 506 census tracts within the Boston metropolitan area. 13 attributes for each census tract are available for use in predicting the median price as shown in Table 1. As Neal (1996) points out, the data is messy in several respects. Some of the attributes are not measured on a per-tract basis, but only for larger regions and the median prices for the highest-priced tracts appear to be censored. When taking into account these potential problems, it seems unreasonable to expect that the distribution of the target value (the median price) given the input data, will be of Gaussian shape. Instead, one would expect the measurement noise distribution to be heavy-tailed, with a few errors being much greater than the typical measurement noise. Hence, the data set seems to be ideally suited for applying the robust online learning methods of Section 3.2 based on the Student-t-distribution.[8]

[8] A related experiment in an offline adaptation framework was carried out by Briegel and Tresp (1999c).

Table 1. Description of the 13 input variables $\boldsymbol{u}_t = (u_{1,t}, \ldots, u_{13,t})^\top$ to the linear and neural network models.

Inputs	Description
crim	per capita crime rate by town
zn	proportion of residential land zoned for lots over 25,000 sq.ft.
indus	proportion of non-retail business acres per town
chas	Charles River dummy variable (1 if tract bounds river, 0 if not)
nox	nitric oxides concentration (parts per 10 million)
rm	average number of rooms per dwelling
age	proportion of owner-occupied units built prior to 1940
dis	weighted distances to five Boston employment centers
rad	index of accessibility to radial highways
tax	full-value property-tax rate per $10,000
ptratio	pupil-teacher ratio by town
b	$1000(\mathrm{Blk} - 0.63)^2$ where Blk is the proportion of blacks by town
lstat	percent lower status of the population

We divided the data set randomly into training and test sets of respectively 400 and 106 input-output patterns. The degrees of freedom of the t-distribution were initialized with $\nu_v = 5$ and the variance σ_v^2 was initialized with the value 0.8. A linear model trained with the Kalman filter (KF) and a neural network model trained with the extended Kalman filter (EKF) served as benchmarks for the neural network model based on Fisher scoring training with the t-density. Online posterior weight smoothing was performed on a sliding time window of length $\tau_t = 50$ time steps with five forward-backward steps over each time window. For all methods, weight transition matrices $\mathbf{Q}_t = 10^{-4}\mathbf{I}_{98,98}$ were chosen. Training was performed by repeated online runs over the training set until convergence of the parameters, typically taking about 30 to 40 cycles through the training data. Hyperparameter adaptation was started after 20 cycles through the data to avoid instabilities during the EM-type adaptation ($\tilde{\tau}_t = 250$). The experiment was carried out 10 times, each time started with different Gaussian distributed initial weight vectors and new training and test sets. Figure 9 (left) shows the explained variance $100 \times (1 - \mathrm{MSE_M}/\mathrm{MSE_{KF}})$ [in percent] where MSE is the mean squared prediction error using either the Kalman filter (KF), the extended Kalman filter (EKF) or the Fisher Scoring approach (FS-50/5). The heights of the bars show the explained variance of the two neural network models with respect to the linear model averaged over the 10 training runs. Clearly, the neural network approaches outperform the linear model by far. Hereby, the nonlinear model based on the t-distribution could explain significantly more variance (7.1% on the training set and 7.7% on the test set) than the neural network model based on the Gaussian noise density (paired t-test with $\alpha = 0.01$). Figure 9 (right) shows the normal probability plot after the extended Kalman filter based network adaptation for the training set. Clearly visible is the derivation from the Gaussian distribution for extreme target values. The plot evidences that the residuals follow a more heavy-tailed distribution than the Gaussian distribution.

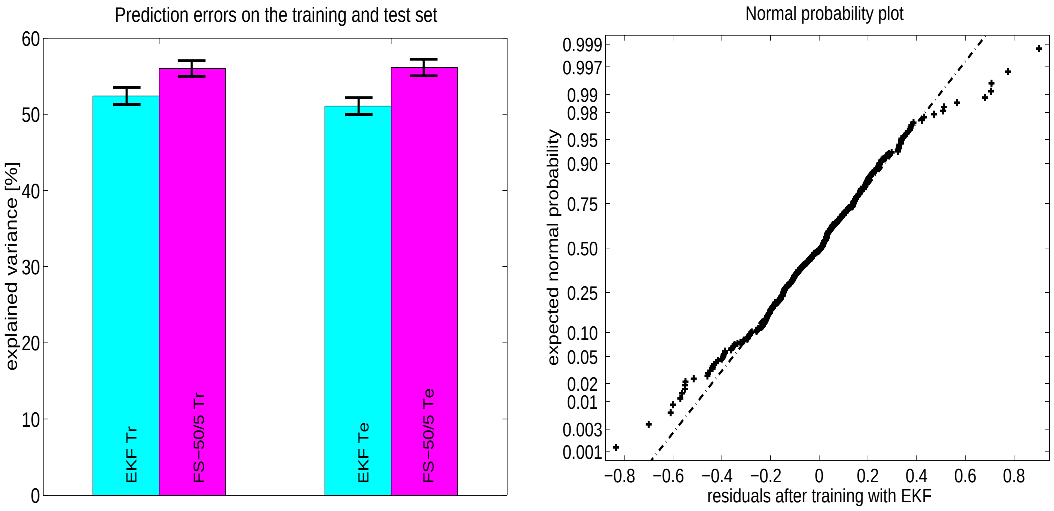

Figure 9. Left: Explained variance of the neural network models compared to the linear model for the training (Tr) and test sets (Te). The neural network model based on the t-distribution could explain significantly more of the variance than the neural network model based on the Gaussian density. The error bars show $\pm$ one standard deviation. **Right:** Normal probability plot of the training set after adaptation of the neural network model based on the Gaussian error density. The dashed line displays the expected normal probabilities. The plot clearly shows that the residuals follow a heavy-tailed distribution.

5 Conclusions

In this paper we were dealing with online learning methods for dynamic neural regression models. Online estimation of the neural network weights was based on a Gaussian approximation to the posterior weight distribution. Specifically, we introduced two particular methods which were based either on posterior modes and curvatures (by employing the Fisher scoring algorithm) or approximate posterior means and covariances (by employing importance sampling) as the centers and widths of the approximating posterior Gaussians. Both algorithms turned out to be particularly suitable for online learning tasks and outperformed state-of-the art methods significantly in the experiments and simulations. Furthermore, we suggested the use of a robust measurement density for handling additive outliers. The Gaussian density was replaced by the Student t-density which includes the Cauchy and the Gaussian densities as special cases. The former is widely used in robust statistics. The application of the Fisher scoring algorithm lead to a computationally attractive robust online learning algorithm where hyperparameters were estimated with an online EM-type algorithm. It was shown experimentially that the novel online learning methods outperform state-of-the online learning method like the extended Kalman filter algorithm for both, situations with Gaussian measurement noise terms as well as for situations with additive outliers.

Appendix: Gradient Computation for Of¤ine Neural Regression

In this section we derive the backpropagation algorithm, i.e. the gradients of the sum of squares error function

$$E(\boldsymbol{w}) = \sum_{t=1}^{T} \big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w})\big)^2 \tag{99}$$

with respect to the multi-layer perceptron weight vector $\boldsymbol{w}$. The functional form of the multi-layer perceptron with a single output unit is given by

$$\begin{aligned} & g : \mathbb{R}^k \to \mathbb{R}, \\ & y = g(\boldsymbol{u}; \boldsymbol{w}) \quad = \quad \sum_{i=1}^{h} w_i \tanh\big(\sum_{j=1}^{k} w_{ij} u_j + w_{i0}\big) + w_0. \end{aligned} \tag{100}$$

The gradients $\partial E/\partial w_i, \partial E/\partial w_{ij}$ are computed by iterative application of the chain-rule

$$\frac{\partial E}{\partial w_i} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial g} \frac{\partial g}{\partial w_i}\Big|_{\boldsymbol{u}=\boldsymbol{u}_t}, \quad \frac{\partial E}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial g} \frac{\partial g}{\partial w_{ij}}\Big|_{\boldsymbol{u}=\boldsymbol{u}_t}. \tag{101}$$

The gradients $\partial E_t/\partial g$ can be simply evaluated yielding to

$$\frac{\partial E_t}{\partial g} = -2\big(y_t - g(\boldsymbol{u}_t; \boldsymbol{w})\big). \tag{102}$$

The problem now reduces to computing $\partial g/\partial w_i, \partial g/\partial w_{ij}$. When de£ning the activities of the hidden units as $o_i = \tanh\left(\sum_{j=1}^{k} w_{ij} u_j + w_{i0}\right)$, then we can derive

$$\frac{\partial g}{\partial w_i} = o_i\,, \quad \frac{\partial g}{\partial w_0} = 1 \tag{103}$$

and with the de£nition $\hat{o}_i = \sum_{j=1}^{k} w_{ij} u_j + w_{i0}$ we can compute the gradients with respect to the the input to the hidden unit connections w_{ij}

$$\frac{\partial g}{\partial w_{ij}} = \frac{\partial g}{\partial o_i} \frac{\partial o_i}{\partial \hat{o}_i} \frac{\partial \hat{o}_i}{\partial w_{ij}} = w_i\,(1 - o_i)^2\, u_j\,, \quad \frac{\partial g}{\partial w_{i0}} = w_i\,(1 - o_i)^2. \tag{104}$$

Hence, one can easily verify that the computation of the gradients $\partial E/\partial w_i, \partial E/\partial w_{ij}$ can be performed in one forward pass where $\hat{o}_i, o_i$ are evaluated at $\boldsymbol{u} = \boldsymbol{u}_t, t = 1, \ldots, T$ and $g(\boldsymbol{u}_t; \boldsymbol{w})$ is computed followed by a backward pass where $\partial E_t/\partial g$ and respectively $\partial g/\partial w_i, \partial g/\partial w_{ij}$, also evaluated at $\boldsymbol{u} = \boldsymbol{u}_t, t = 1, \ldots, T$, are derived.

For the gradients of the multi-layer perceptron with respect to the network inputs $\partial g/\partial u_j$ we obtain in a similar manner

$$\frac{\partial g}{\partial u_j} = \sum_{i=1}^{h} \frac{\partial g}{\partial o_i} \frac{\partial o_i}{\partial \hat{o}_i} \frac{\partial \hat{o}_i}{\partial u_j} = \sum_{i=1}^{h} w_i\,(1 - o_i)^2\, w_{ij}. \tag{105}$$

References

Abramowitz, M. and I. Stegun (Eds.) (1989). *Handbook of Mathematical Functions*. New York, NY: Dover Publications.

Anderson, B. and J. Moore (1979). *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall.

Andrews, D. and C. Mallows (1974). Scale mixtures of normal distributions. *Journal of the Royal Statistical Society B 36*, 99–102.

Bar-Shalom, Y. and X.-R. Li (1993). *Estimation and Tracking: Principles, Techniques and Software*. Norwood, MA: Artech House.

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

Barron, A. (1994). Approximation and estimation bounds for arti£cial neural networks. *Machine Learning 14*, 115–133.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford, UK: Clarendon Press.

Black, F. and M. Scholes (1973). Pricing options and corporate liabilities. *Journal of Political Economy 81*, 637–654.

Boyden, X. and D. Koller (1999). Approximate learning of dynamic models. In M. Kearns, S. Solla, and D. Cohn (Eds.), *Advances in Neural Information Processing Systems 11, Proceedings of the 1998 Conference*, Cambridge, MA, USA, pp. 397–402. MIT Press.

Briegel, T. and V. Tresp (1999a). Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space mdoels. In M. Kearns, S. Solla, and D. Cohn (Eds.), *Advances in Neural Information Processing Systems 11, Proceedings of the 1998 Conference*, Cambridge, MA, pp. 403–409. MIT Press.

Briegel, T. and V. Tresp (1999b). Robust neural network online-learning in time-variant regression models. In E. Principe, K. Morgan, L. Giles, and A. Wilson (Eds.), *Neural Networks for Signal Processing, Proceedings of the 1999 Workshop*, Madison, WI.

Briegel, T. and V. Tresp (1999c). Robust neural network regression for of¤ine and online learning. In S. Solla, T. Leen, and K.-R. Müller (Eds.), *to appear in Advances in Neural Information Processing Systems 12, Proceedings of the 1999 Conference*, Cambridge, MA. MIT Press.

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of Control Signals and Systems 2*, 293–314.

de Freitas, J. F. G., M. Niranjan, and A. H. Gee (1998). Regularisation in sequential learning algorithms. In M. I. Jordan, M. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems 10, Proceedings of the 1997 Conference*, Cambridge, MA, pp. 458–464. MIT Press.

de Freitas, J. F. G., M. Niranjan, A. H. Gee, and A. Doucet (1998). Sequential Monte Carlo methods for optimisation of neural network models. Technical Report CUED/F-INFENG/TR 328, Dept. of Engineering, University of Cambridge.

de Freitas, J. F. G., M. Niranjan, A. H. Gee, and A. Doucet (1999). Global optimisation of neural network models via sequential sampling. In M. Kearns, S. Solla, and D. Cohn (Eds.), *Advances in Neural Information Processing Systems 11, Proceedings of the 1998 Conference*, Cambridge, MA, pp. 410–416. MIT Press.

Deco, G. and W. Brauer (1995). Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks 8*(4), 525–535.

Fahrmeir, L. and H. Kaufmann (1991). On Kalman filtering, posterior mode estimation and Fisher scoring in dynamic exponential family regression. *Metrika 38*, 37–60.

Fahrmeir, L. and R. Künstler (1999). Penalized likelihood smoothing in robust state space models. *Metrika 49*, 173–191.

Fahrmeir, L. and G. Tutz (1994). *Multivariate Statistical Modelling Based on Generalized Linear Models*. New York, NY: Springer.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (Eds.) (1996). *Markov Chain Monte Carlo in Practice*. Suffolk: Chapman and Hull.

Gill, P., W. Murray, and M. Wright (1981). *Practical Optimization* (10 ed.). London: Academic Press.

Harrison, D. and L. Rubinfeld (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics & Management 5*, 81–102.

Hennevogl, W. (1991). *Schätzung generalisierter Regressions- und Zeitreihenmodellen mit variierenden Parametern*. Ph. D. thesis, Wirtschaftswissenschaftliche Fakultät, Universität Regensburg.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks 4*, 251–257.

Huber, P. J. (1964). Robust estimation of location parameters. *Annals of Mathematical Statistics 35*, 73–101.

Hull, J. C. (1997). *Options, Futures and other Derivatives* (3 ed.). Prentice Hall.

Hutchinson, J. M., A. W. Lo, and T. Poggio (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance 49*(3), 851–889.

Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics 20*(1), 608–613.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time-series. *Journal of the American Statistical Association 82*, 1032–1063.

Lange, K., L. Little, and J. Taylor (1989). Robust statistical modeling using the t-distribution. *Journal of the American Statistical Association 84*, 881–896.

MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation 4*, 448–472.

MacKay, D. J. C. (1999). *Learning in Graphical Models*, Chapter Introduction to Monte Carlo methods, pp. 175–204. Cambridge, MA: MIT Press.

McCulloch, W. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133.

Meinhold, R. J. and N. D. Singpurwalla (1989). Robusti£cation of Kalman Filter models. *Journal of the American Statistical Asscociation 84*, 470–496.

Neal, R. (1996). *Bayesian Learning For Neural Networks*. Ph. D. thesis, Dept. of Computer Science, University of Toronto.

Niranjan, M. (1997). Sequential tracking in pricing £nancial options using model based and neural network appraoches. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9, Proceedings of the 1996 Conference*, pp. 960–966. MIT Press.

Ormoneit, D. and R. Neuneier (1995). Reliable neural network predictions in the presence of outliers and non-constant variances. In A. N. Refenes, Y. Abu-Mustafa, J. Moody, and A. Weigend (Eds.), *Neural Networks in Financial Engineering, Proceedings of the Third International Conference on Neural Networks in the Capital Markets*, pp. 578–587.

Pole, A. and M. West (1990). Ef£cient Bayesian learning in non-linear dynamic models. *Journal of Forecasting 9*, 119–136.

Rao, C. and S. Mitras (1971). *Generalized Inverse of Matrices and its Applications*. New York, NY: John Wiley & Sons.

Rousseeuw, P. and A. Leroy (1987). *Robust Regression and Outlier Detection*. John Wiley & Sons.

Rumelhart, D., G. Hinton, and R. Williams (1986). *Parallel Distributed Processing*, Volume 1, Chapter Learning internal representations by error backpropagation, pp. 318–362. Cambridge, MA: MIT Press.

Sage, A. and J. Melsa (1971). *Estimation Theory with Applications to Communications and Control*. New York, NY: McGraw-Hill.

Schnatter, S. (1992). Integration-based Kalman £ltering for a dynamic generalized linear trend model. *Computational Statistics & Data Analysis 13*, 447–459.

Singhal, S. and L. Wu (1989). Training multilayer perceptrons with the extended Kalman £lter algorithm. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 1, Proceedings of the 1988 Conference*, San Mateo, CA, pp. 133–140.

Sum, J., C. Leung, G. H. Young, and W. Kan (1999). On the Kalman £ltering method in neural-network training and pruning. *IEEE Transactions on Neural Networks 10*(1), 161–166.

Tresp, V. (1995). Die besonderen Eigenschaften Neuronaler Netze bei der Approximation von Funktionen. *Künstliche Intelligenz 5*, 12–17.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*. Ph. D. thesis, Harvard University, Boston, MA.

West, M. (1981). Robust sequential approximate Bayesian estimation. *Journal of the Royal Statistical Society B 43*, 157–166.