A Regularization Approach to Continuous Learning with an Application to Financial Derivatives Pricing

Dirk Ormoneit*
Department of Statistics
Stanford University
Stanford, CA 94305-4065
ormoneit@stat.stanford.edu

Abstract

We consider the training of neural networks in cases where the nonlinear relationship of interest gradually changes over time. One possibility to deal with this problem is by regularization where a variation penalty is added to the usual mean squared error criterion. To learn the regularized network weights we suggest the Iterative Extended Kalman Filter (IEKF) as a learning rule, which may be derived from a Bayesian perspective on the regularization problem. A primary application of our algorithm is in financial derivatives pricing, where neural networks may be used to model the dependency of the derivatives' price on one or several underlying assets. After giving a brief introduction to the problem of derivatives pricing we present experiments with German stock index options data showing that a regularized neural network trained with the IEKF outperforms several benchmark models and alternative learning procedures. In particular, the performance may be greatly improved using a newly designed neural network architecture that accounts for no-arbitrage pricing restrictions.

Keywords: Continuous Learning, Regularization, Maximum A Posteriori Estimation, Iterative Extended Kalman Filter, Derivatives Pricing

*This work is supported by a grant of the Deutsche Forschungsgemeinschaft (DFG) as part of its post-doctoral program. I am grateful to Volker Tresp and Trevor Hastie for helpful discussions. The DAX option and futures data were provided by the Deutsche Terminbörse AG.

1 Introduction

The identification of nonlinear relationships that change over time is one of the ongoing challenges of neural computation. The relevance of this problem is illustrated best by the fact that at the 1998 Neural Information Processing Systems conference even two workshops have been concerned with its understanding. As the result of one of these workshops, the term "continuous learning" has been coined to summarize many of the concepts that have been touched upon in this context, such as online, life-long, incremental, or sequential learning. In particular, continuous learning refers to situations in which learning occurs "uninterrupted through the whole lifetime of the operation" in order to track changes in the environment. In spite of the continuous efforts to cope with continuous learning, no particular algorithm has evolved as a primary candidate for the training of neural networks in nonstationary environments, such that online backpropagation may still be considered as the de facto state of the art. Part of the reason is that even though a straightforward probabilistic model of continuous learning is available in the form of state-space models, training algorithms based on this interpretation generally suffer from conceptual and computational shortcomings. One of the key difficulties is that rigid distributional assumptions about the nature of the changes in the nonlinear dependency of interest have to be made in order to arrive at a well-defined inference and learning problem. In addition, the nonlinear filtering problem involved in such approaches is well-known to be computationally very demanding and by itself subject to intensive research.

Within this paper, we suggest to circumvent both of the aforementioned problems by using a regularization approach to continuous learning. Though also the penalty term employed in regularization incorporates prior information on the nature of the time-variation, the underlying assumptions are less rigid. Thus, in many respects an approach based on regularization is more "robust" than the theoretically appealing Bayesian alternative. In particular, we suggest a penalized error criterion based on a measure of the variation of the approximating function over time. A suitable penalty term may be defined by considering pairwise divergence measures of the approximating functions at subsequent time steps. For the numerical minimization of the penalized error criterion, we suggest the "Iterative Extended Kalman Filtering" (IEKF) algorithm as a neural network learning rule. The IEKF has been used in the statistics literature to approach the closely related problem of generalized linear regression with changing coefficients by Fahrmeir (1992). While it has originally been suggested as a method for maximum a posteriori (MAP) estimation in a Bayesian framework, we use the IEKF merely as a numerical instrument for the minimization of the penalized error criterion in this work, thereby avoiding the previously mentioned problems that are inherent to a Bayesian interpretation.

To demonstrate the practical relevance of the regularization approach, we provide experimental results in a real-world financial forecasting application. In particular, we use regularized neural networks to price and hedge derivative assets, a relatively new application of neural networks pioneered by Hutchinson et al. (1994). The availability of closed option pricing formulas is of great interest to both researchers and practitioners and has attracted considerable attention in the finance literature. As a continuous learning problem, option pricing imposes interesting challenges. The observed market prices may well depend on unobservable quantities, such as stochastic volatility levels or changing factors of risk-aversion among the investors, which seriously limits the applicability of many "classical" option pricing approaches and makes the unknown pricing relation appear nonstationary to a modeler who is restricted to observable variables as input information. In experiments with high-frequency data from the Deutsche Terminbörse we find that training a specially designed neural network with either online backpropagation, the Extended Kalman Filter (EKF), or the IEKF leads to more accurate price predictions and to a superior hedging performance than a Black-Scholes benchmark model Black and Scholes (1973). In particular, the IEKF outperforms online backpropagation and the EKF as a learning rule.

The remainder of this work is organized as follows. In Section 2, we give the formulation of continuous learning as a regularization problem, and in Section 3 we derive a numerical procedure for its solution. Section 4 gives a brief introduction to derivatives pricing and describes our experimental results. Section 5 presents conclusions.

2 A Regularization Approach

Regularization is a well-known method for the treatment of mathematically ill-posed problems (for an overview, see Szeliski (1996)). It has successfully been applied to numerous machine learning problems including the avoidance of overfitting in neural network training. An important application of regularization is in nonparametric regression, where a generic function f is used to fit a series of observations $D \equiv \{x_t \equiv (y_t, z_t), t = 1, \ldots, T\}$. In order to find a regularized solution to this problem, one typically defines a *stabilizing functional* or a *penalty term* of the form $\mathcal{P}[f] \equiv \int f''(x)^2 dx$ that rewards smoothness of the approximating function. It is well-known that by the virtue of this definition we obtain a cubic smoothing spline as the minimizer of the *penalized error criterion* (Ahlberg et al. (1967); Reinsch (1967))

$$\mathcal{PE}[f] = \frac{1}{2}\sum_{t=1}^{T}(y_t - f(z_t))^2 + \lambda\mathcal{P}[f]. \tag{1}$$

Here λ is a smoothing parameter whose optimal value must be determined, for example, using cross-validation.

In the case of continuous learning, the index t of the samples x_t in the definition of the training data set D has a special interpretation as indexing the order at which individual samples arrive over time. In this context, the approximating function may be adapted after the arrival of each new observation, thereby generating a sequence of approximations $\mathcal{F} \equiv \{f_t, t = 1, \ldots, T\}$, one for each time step. Consequently, a regularization approach to continuous learning must be defined in terms of the entire sequence $\mathcal{F}$ rather than in terms of an individual function as in (1). A suitable penalized error criterion may be written in the form

$$\mathcal{PE}[\mathcal{F}] = \frac{1}{2}\sum_{t=1}^{T}(y_t - f_t(x_t))^2 + \lambda\mathcal{P}[\mathcal{F}], \tag{2}$$

where $\mathcal{P}[\mathcal{F}]$ is a penalty operator defined on $\mathcal{F}$ and where λ regularizes the intensity of the penalty.

Within the framework of this paper, we use neural networks as an approximating architecture, i.e. $f_t(\cdot) \equiv f(\cdot;\theta_t)$ for some neural network architecture $f(\cdot;\theta_t)$ and a (time-varying) d-vector θ_t of neural network weights. In contrast to the time-constant regression problem (1), the penalty term $\mathcal{P}[\mathcal{F}]$ serves two purposes, namely to restrict the complexity of each individual function approximator f_t and to restrict the temporal variation of f_t across different functions in $\mathcal{F}$. As the complexity reducing property of $\mathcal{P}[\mathcal{F}]$ is of less interest to the presented work below we exclusively focus on the aspect of time-variation. In particular, we define $\mathcal{P}[\mathcal{F}]$ so as to penalize the pairwise distance between subsequent function estimates f_{t-1} and f_t. More precisely, the distance between two neural networks of equal size is assessed using the Eucledian distance between their weight vectors θ_{t-1} and θ_t in this work, such that the resulting penalty operator may be written as

$$\mathcal{P}[\mathcal{F}] \equiv \frac{1}{2}\sum_{t=2}^{T} ||\theta_t - \theta_{t-1}||^2. \tag{3}$$

By substituting (3) and the expression for the neural network architecture $f(z_t;\theta_t)$ in (2) we obtain the penalized error criterion

$$\mathcal{PE}[\mathcal{F}] = \frac{1}{2}\sum_{t=1}^{T}(y_t - f(z_t;\theta_t))^2 + \frac{\lambda}{2}\sum_{t=2}^{T} ||\theta_t - \theta_{t-1}||^2. \tag{4}$$

Note that, as $\mathcal{PE}[\mathcal{F}]$ involves an entire sequence of neural network weights, considerable care has to be taken to derive a numerically efficient method for its minimization. A suitable algorithm based on a state-space interpretation of the penalized error criterion (4) is suggested in the next section.

3 The Iterative Extended Kalman Filter

For the minimization of the penalized error criterion (4), it is helpful to consider a Bayesian interpretation of the penalty term $\mathcal{P}[\mathcal{F}]$. As always in regularization, $\mathcal{P}[\mathcal{F}]$ may be regarded as minus the logarithm of a *prior probability density* on the space of parameter sequences $\Theta \equiv \{\theta_t, t = 1, \ldots, T\}$. In particular, (3) corresponds to a prior density of the form

$$p(\Theta) \equiv \prod_{t=2}^{T} N(\theta_t | \theta_{t-1}, \lambda^{-1} I_d) \cdot \pi(\theta_1), \tag{5}$$

where $N(\cdot|\mu, \Sigma)$ denotes the multivariate normal density function with mean μ and covariance matrix Σ, and I_d is the $d \times d$ unit matrix. For purely technical reasons, we need to include the non-informative prior $\pi(\theta_1)$ on the first element of the sequence. (3) and (5) are related via the equation $\lambda \mathcal{P}[\mathcal{F}] = c_1 - \log p(\Theta)$ for some constant c_1. Similarly, the first term in (4) may be used to construct the likelihood term

$$p(D|\Theta) \equiv \prod_{t=1}^{T} N(y_t | f(z_t; \theta_t), 1). \tag{6}$$

$p(D|\Theta)$ satisfies $1/2 \sum_{t=1}^{T} (y_t - f(z_t; \theta_t))^2 = c_2 - \log p(D|\Theta)$ for some scalar c_2. Together, (5) and (6) define a Bayesian state-space model with the property that the maximum a posteriori (MAP) estimate of Θ coincides with the regularized estimate of the neural network weight sequence, i.e.

$$\mathcal{PE}[\mathcal{F}] \propto -\log p(\Theta|D). \tag{7}$$

This identity may be used to bring nonlinear filtering and smoothing techniques to bear upon the minimization problem (4). In particular, the Iterative Extended Kalman Filter (IEKF) is a well-known technique to compute MAP estimates in nonlinear state-space models and may thus alternatively be used to minimize $\mathcal{PE}[\mathcal{F}]$ (for details, see Sage and Melsa (1971) and Fahrmeir (1992)). Applied to (7), the IEKF amounts to the iterative application of the usual Extended Kalman Filter (EKF) prediction and correction steps in the *forward pass* ($t = 1, \ldots, T$) of the algorithm:

$$\begin{aligned} \hat{\theta}_{t|t-1} &= \hat{\theta}_{t-1|t-1} \qquad\qquad \Omega_{t|t-1} = \Omega_{t-1|t-1} + \lambda^{-1} I_d && (8) \\ \Omega_{t|t} &= -\left[\nabla^2 \log N(y_t | f(z_t; \hat{\theta}_{t|t-1})) - \Omega_{t|t-1}^{-1}\right]^{-1} \\ \hat{\theta}_{t|t} &= \hat{\theta}_{t|t-1} + \Omega_{t|t} \nabla \log N(y_t | f(z_t; \hat{\theta}_{t|t-1})). && (9) \end{aligned}$$

Within the Bayesian framework, $\hat{\theta}_{t|t-1}$ and $\Omega_{t|t-1}$ approximate the mean and the covariance matrix of the posterior distribution of θ_t given the observations $\{(y_s, z_s) | 1 \leq$

$s < t\}$. Similarly, $\hat{\theta}_{t|t}$ and $\Omega_{t|t}$ describe the posterior distribution given all previous observations, including (y_t, z_t). For a linear neural network, (8) and (9) simplify to the usual Kalman filtering update steps, and $\hat{\theta}_{t|t}$ is a least-squares optimal estimate of θ_t. For a nonlinear network, $\hat{\theta}_{t|t}$ is the best estimate of θ_t with respect to a second-order Taylor series approximation of $\log N(y_t|f(z_t; \hat{\theta}_{t|t-1}))$ in terms of $\nabla \log N(y_t|f(z_t; \hat{\theta}_{t|t-1}))$ and $\nabla^2 \log N(y_t|f(z_t; \hat{\theta}_{t|t-1}))$.

In a time-series context, interest frequently focuses on the posterior distribution of θ_t given *all* observations in D; the computation of this posterior is called "smoothing". A simple approach is to augment the forward filtering pass using (8) and (9) with a *backward smoothing pass* $(t = T, \ldots, 1)$:

$$\begin{aligned} \hat{\theta}_{t|T} &= \hat{\theta}_{t|t} + B_t(\hat{\theta}_{t+1|T} - \hat{\theta}_{t+1|t}) \\ \Omega_{t|T} &= \Omega_{t|t} + B_t(\Omega_{t+1|T} - \Omega_{t+1|t}). \end{aligned} \tag{10}$$

Here $\hat{\theta}_{t|T}$ and $\Omega_{t|T}$ are the mean and the covariance of the "smoothing distribution", $B_t = \Omega_{t|t}\Omega_{t+1|t}^{-1}$, and ∇ and ∇^2 denote the gradient and the Hessian operator with respect to $\hat{\theta}_{t|t-1}$, respectively.[1] Note that, due to (7), $\hat{\theta}_{t|T}$ is is the exact solution to the minimization problem (4) in the case of a linear neural network. For nonlinear networks, the idea is obtain better and better approximations of the solution by the repeated application of filtering and smoothing passes. In detail, it can be shown that each pair of a filtering and a smoothing pass corresponds to a single Newton step for maximum a posteriori estimation. Thus, iterative filtering and smoothing amounts to the standard Newton procedure for the solution of (4) by the equivalence (7).

A formal discussion of the relationship between EKF and the Newton algorithm is beyond the scope of this paper and can be found in Sage and Melsa (1971); Fahrmeir (1992). To gain an intuitive understanding, note that the Newton update equation is based on a second-order Taylor series expansion of the function that is to be minimized, in our case (4), around some initial estimate, Θ^0. However, as (4)

[1] Alternative formulations of (9) which may be obtained by applying the "matrix inversion lemma" may be found in the literature. In our experiments, we found that the formulation above was numerically the most advantageous.

is additive in the terms

$$\frac{1}{2}(y_t - f(z_t;\theta_t^0))^2 \quad \propto \quad -\log N(y_t|f(z_t,\theta_t^0)),$$

its Taylor expansion amounts to the Taylor expansion of $\log N(y_t|f(z_t,\theta_t^0))$, involving the terms $\nabla N(y_t|f(z_t,\theta_t^0))$ and $\nabla^2 N(y_t|f(z_t,\theta_t^0))$. We mentioned above that the Extended Kalman Filter and Smoother yields the least-squares optimal estimate of the hidden states θ_t given D under an identical second-order Taylor series expansion of $\log N(y_t|f(z_t;\hat{\theta}_{t|t-1}))$. In particular, the posterior distribution of Θ given D is jointly normal under this approximation, such that its mean vector $(\hat{\theta}_{1|T},\ldots,\hat{\theta}_{T|T})$ coincides with the posterior mode and, consequently, with the minimizer of the local quadratic approximation to $\mathcal{PE}[\mathcal{F}]$ as claimed.

A schematic implementation of the IEKF as a learning algorithm is shown in Figure 1. Suppose we previously used the IEKF to compute the MAP estimate

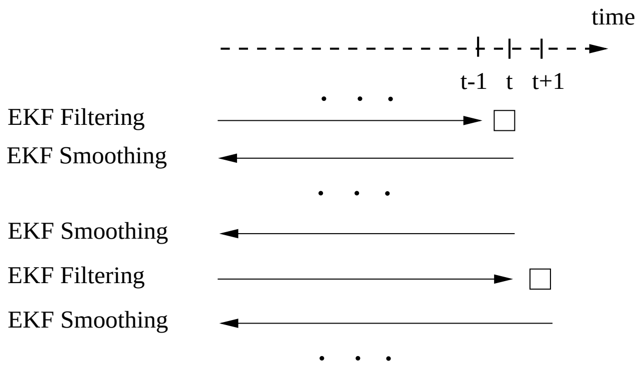

Figure 1: *Schematic representation of the IEFK.*

of $\{\theta_s, s = 1,\ldots,t-1\}$ given the first $t-1$ observations, and that we finished the previous processing step with a forward pass. Based on the current estimates $\hat{\theta}_{t-1|t-1}$ and $\Omega_{t-1|t-1}$ we may then carry out an additional filtering step to obtain $\hat{\theta}_{t|t}$ and $\Omega_{t|t}$ (indicated by the box in Figure 1). Next we carry out a backward smoothing pass using (10) to obtain $\hat{\theta}_{s|t}$ and $\Omega_{s|t}$. Then we repeat filtering and smoothing passes until the Newton algorithm converges to a local optimum. In particular, we finish the iteration with a forward pass so that the procedure above may be repeated at time t.

It is clear that, as the IEKF 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 operation it may thus be necessary to truncate the current data history at some fixed horizon. Other numerical aspects of the application of the EKF update equations (8) through (10) as a neural network learning rule are discussed in Appendix A. Note that the filtering rules (8) and (9) in the context of continuous learning are considerably simpler than the general prediction and correction steps of the EKF, which is due to the special structure implied by (5) and (6). Pseudo-code to simplify the practical implementation of the IEKF is provided in Appendix C.

4 Financial Derivatives Pricing and Hedging

In this section, we apply the IEKF as a learning algorithm for a neural network that predicts prices and hedge ratios of financial derivative assets. In particular, we analyze a set of call options on the German stock index DAX, which give their owner the right (but not the obligation) to buy the DAX index for some contractual *strike price* K at a predetermined *expiration date* T. Technically, it is appropriate to think of a call option as the right to receive a payment of $\max\{S_T - K, 0\}$ at time T where $\{S_t, t = 1, \ldots, T\}$ denotes the price process characterizing the underlying asset (the DAX).[2] At any date $t < T$ prior to expiration, the option payoff is a random quantity and it is not clear how much an investor should be willing to pay for the option. Owing to the central importance of derivative assets for risk transfer, option pricing is one of the most active areas of financial research. Black and Scholes (1973) derived an explicit formula for the option price as a function of the price of the underlying asset S_t, a volatility parameter ρ characterizing the variance of changes in S_t, the interest rate r, and the time-to-expiration $T-t$. However, their derivation critically depends on several assumptions that are commonly violated in practice. As an alternative approach, neural network models recently have been

[2] More precisely, our definition characterizes an European-style, cash-settled "vanilla" option. Discussions of option styles and pricing approaches may be found in Hull (1997).

brought to bear upon the option pricing problem by regressing historically observed option prices on suitable explanatory variables Hutchinson et al. (1994). Here the observed market prices may depend on unobservable quantities, such as stochastic volatility levels or changing factors of risk-aversion among the investors. The identification of option pricing formulas from historical prices is thus realistically treated within a continuous learning framework.

We analyze the set of call options issued in March 1997 and expiring in December 1997. For each trade during this period, our data consists of the exact time of the trade, the price at which the option was exchanged, the strike price, and the last value of the DAX future prior to the trade in the option.[3] Altogether, there are 32104 data records. After removing outliers and entries with missing data, we were left with a data set consisting of 25921 observations. As a benchmark we report Black-Scholes option prices assuming a constant interest rate of $r = \log(1.05)$ in Table 1. The volatility ρ entering into the Black-Scholes model was chosen in three different ways. First, we report results for multiple *fixed* values of ρ, corresponding to the naive strategy of "guessing" a volatility level and using it for prediction ("BS-F"). Second, we used the *implied volatility of the previous call option trade* ("BS-IV"). The latter approach is based on the observation that implied volatility typically varies smoothly over time. In the third approach, we use the normalized standard deviation of the returns of the DAX futures contract, where the standard deviation was computed using either 60, 200, or 500 past observations.

The prediction accuracy is assessed by means of the mean squared error

$$MSE \equiv \frac{1}{T}\sum_{t=1}^{T}[y_t - f(z_t;\hat{\theta}_{t-1})]^2,$$

in column three of Table 1. In addition, to indicate the performance improvement relative to a naive unconditional prediction based on the sample average

[3]The reason for using DAX futures contracts rather than the DAX index itself is that DAX futures quotes are available at very high frequency. For hedging, using DAX futures also seems far more realistic. For the purposes of this paper, it suffices to think of the futures price as a close approximation of the DAX index.

Algorithm	**Parameter**	$MSE \times 10^4$	R^2	$\overline{HE}$
BS-F	$\rho = 0.25$	0.51	0.991	210.8 (246.4)
BS-F	$\rho = 0.3$	0.34	<u>0.994</u>	201.8 (236.9)
BS-F	$\rho = 0.35$	0.67	0.988	194.1 (228.2)
BS-F	$\rho = 0.4$	1.55	0.973	<u>187.0</u> (220.3)
BS-IV	–	<u>0.31</u>	<u>0.994</u>	252.8 (288.7)
BS-H	60	8.67	0.854	274.5 (309.6)
BS-H	200	7.05	0.881	276.7 (312.5)
BS-H	500	7.60	0.867	281.5 (317.0)

Table 1: *Mean squared pricing error, R^2, and mean hedging error of the Black-Scholes benchmark models.*

$\bar{y} \equiv 1/T \sum_{t=1}^{T} y_t$, we also report the R^2-measure in column four:

$$R^2 \equiv 1 - \frac{T \cdot MSE}{\sum_{t=1}^{T}(y_t - \overline{y})^2}.$$

Note that when evaluating these expressions, the squared error at time t was based on the parameter estimate from the previous epoch, such that the reported results indicate out-of-sample performance. In the rightmost column, we report the mean *hedging error*, i.e. the amount of money that is gained or lost by the failure of the option pricing model to determine a risk-neutral portfolio. A small hedging error is very important for financial institutions wishing to insure their positions against undesired market movements. Roughly, to set off profits and losses from holding a single option, we construct a portfolio containing n_t stock index units and B_t Dollars of cash in a fictive "margin account" at time t.[4] Also, the portfolio contains the single option contract; thus, the value of the portfolio at time t is

$$W_t \equiv n_t S_t + C_t + B_t.$$

Our objective is to make W_t as insensitive as possible to changes in S_t. A reasonable approximation is to choose n_t so as to ensure $\frac{dW_t}{dS_t} = 0$. Since the margin account is

[4]The margin account is assumed to pay interest at the continuously compounded interest rate r. As mentioned above, we really use index futures instead of the index itself to implement the hedge.

riskless, we have $\frac{dB_t}{dS_t} = 0$. The option price C_t is assumed to depend on S_t through the option pricing formula (i.e. our neural network model). Thus, it is natural to add

$$n_t = -\frac{dC_t}{dS_t} \tag{11}$$

stock index units to the portfolio. Of course, the described hedge will be imperfect in practice: first, because $\frac{dW_t}{dS_t} = 0$ is only an approximation, and second, because the neural network produces a noisy estimate of $\frac{dC_t}{dS_t}$. The hedging error summarizes the potential payments that result from these defects of the hedging strategy:

$$HE \equiv |B_T + \max\{S_T - K, 0\}|.$$

For a perfect hedge, the (possibly negative) value of the margin account at termination would just offset the option payoff, that is $B_T = -\max\{S_T - K, 0\}$. For an imperfect hedge, HE simply summarizes the absolute gain or loss that was generated. By considering the absolute value, we may alternatively use HE to assess the ability of a hedging strategy to account for short positions; that is, to cover the payoff of an option that has previously been sold to another party. We simulated the hedging strategy (11) for all strike prices K at which options were traded in the specified period, and report the mean of the results in the rightmost column (standard deviations are given in parentheses). As can be seen, the implied volatility approach performs best with regard to the mean squared error, while the hedging error is minimized using a fixed volatility of $\sigma = 0.4$.

Next, we used a time-varying multi-layer perceptron with two sigmoidal hidden units to model the data. Here the target values y_t are the observed option prices C_t divided by their strike price K, and the network inputs z_t are the prices S_t of the DAX futures contract, also divided by K. The network weights were initialized such that the points of inflection of the sigmoidals were located at $z = 1.0$ and $z = 1.2$. Subsequently, the networks were trained using either online backpropagation ("online BP") according to the learning rule

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \eta \nabla \log N(y_t | f(z_t; \hat{\theta}_{t-1})),$$

or the EKF or the IEKF algorithm as described in Section 3. Our results are reported

Algorithm	**Parameter**	$MSE \times 10^4$	R^2	$\overline{HE}$
online BP	$\eta = 10^{-6}$	3.93	0.933	90.9 (119.2)
online BP	$\eta = 10^{-7}$	5.95	0.900	108.8 (140.3)
EKF	$\lambda = 10^{-7}$	6.65	0.888	124.4 (171.6)
EKF	$\lambda = 10^{-8}$	6.15	0.896	101.5 (151.6)
EKF	$\lambda = 10^{-9}$	6.64	0.888	98.5 (140.7)
IEKF	$\lambda = 10^{-5}$	4.24	0.928	76.3 (101.2)
IEKF	$\lambda = 10^{-6}$	<u>1.16</u>	<u>0.980</u>	72.4 (95.0)
IEKF	$\lambda = 10^{-7}$	1.36	0.977	<u>66.1</u> (89.0)

Table 2: *Mean squared pricing error, R^2, and mean hedging error of online backpropagation, the extended Kalman filter and the iterative extended Kalman filter. The best result in each category is underlined.*

in Table 2. Here the learning parameter in the second column indicates the learning rate η for online backpropagation and the regularization parameter λ for the EKF and IEKF algorithms. It can be seen that the best performance both with respect to the mean squared error and with respect to the mean hedging error is obtained using the IEKF as a learning rule. However, even for the optimal parameter choice $\lambda = 10^{-6}$ the performance of the neural network approach (1.16×10^{-4}) is still significantly worse with regard to the mean squared error than the best Black-Scholes model (0.31×10^{-4}). With respect to the hedging error, there is a considerable improvement from 187.0 to 66.1.

A possible reason for the inferior performance of the neural network model is the "kink" in the pricing function as the time-to-expiration approaches zero. The reason is that as the option is known to pay $\max\{S_T - K, 0\}$ at time T, the option price must approach this limit as t approaches T. Otherwise, riskless profits could be realized, for example by buying the option for a price $C_t < \max\{S_T - K, 0\}$ just prior to T and receiving the payoff $\max\{S_T - K, 0\}$ at time T, thereby generating a riskless profit of $\max\{S_T - K, 0\} - C_t$. (If $C_t > \max\{S_T - K, 0\}$, the reverse argument holds.) Consequently, we have $y_T = \max\{S_T - K, 0\}/K = \max\{z_T - 1, 0\}$ at expiration. A mixture of two sigmoidals is hardly suited to accommodate this boundary behavior. We thus define a new neural network architecture specifically for the purpose of call option pricing based on a fundamental result of financial theory which, in the absence of arbitrage, allows us to write the option price C_t as the (discounted) expectation of $\max\{S_T - K, 0\}$ with respect to a so-called *risk-neutral* probability density. In particular, by choosing a Gaussian mixture as a model of the risk-neutral density, the predicted option price may be written in the form

$$f(z;\theta) \equiv \sum_{k=1}^{n} \frac{\alpha_k^2}{\sum_l \alpha_l^2} \xi_k \text{ where} \tag{12}$$

$$\xi_k \equiv z\Phi\left(\frac{\log z + \tilde{\mu}_k}{\tilde{\sigma}_k} + \frac{1}{2}\tilde{\sigma}_k\right) - e^{-\tilde{\mu}_k}\Phi\left(\frac{\log z + \tilde{\mu}_k}{\tilde{\sigma}_k} - \frac{1}{2}\tilde{\sigma}_k\right). \tag{13}$$

Here Φ denotes the standard normal distribution function and the normalization of α_k in (12) serves to enforce suitable constraints on the mixture weightings. ξ_k equals the expected value of $\max\{S_T/K - 1, 0\}$ under a normal distribution with mean $\tilde{\mu}_k$ and variance $\tilde{\sigma}_k^2$, and thus coincides with the (normalized) Black-Scholes price using the interest rate $\tilde{\mu}_k$ and the volatility $\tilde{\sigma}_k$. From a neural network perspective, (13) constitutes a special hidden-unit activation function, and (12) defines the hidden-to-output connections of a special option pricing network. To obtain the experimental results reported in Table 3 we used an architecture containing two hidden units ($m = 2$) and a time-varying vector of neural network weights $\theta_t = (\alpha_{t,1}, \alpha_{t,2}, \tilde{\mu}_{t,1}, \tilde{\mu}_{t,2}, \tilde{\sigma}_{t,1}, \tilde{\sigma}_{t,2})'$. Again, we trained the network using either online backpropagation, the EKF, or the IEKF algorithm. The derivatives that are needed for training are summarized in Appendix B.

The results in Table 3 indicate a drastic performance improvement of the option pricing network by more than an order of magnitude both in comparison to the best Black-Scholes approach and in comparison to the best multilayer perceptron. As for the choice of the learning algorithm, the IEKF performs best with a mean squared error of 0.018×10^{-4}. In terms of the hedging error, online backpropagation outperforms the other learning algorithms with a performance of 29.3.

Algorithm	**Parameter**	$MSE \times 10^4$	R^2	$\overline{HE}$
online	$\eta = 10^{-6}$	0.031	0.9994	<u>29.3</u> (39.0)
online	$\eta = 10^{-7}$	0.119	0.9979	40.5 (49.2)
EKF	$\eta = 10^{-6}$	0.054	0.9990	37.8 (47.2)
EKF	$\eta = 10^{-7}$	0.038	0.9993	34.3 (43.4)
EKF	$\eta = 10^{-8}$	0.062	0.9989	38.1 (46.3)
IEKF	$\eta = 10^{-5}$	0.032	0.9994	73.1 (87.2)
IEKF	$\eta = 10^{-6}$	<u>0.018</u>	<u>0.9996</u>	60.3 (69.4)
IEKF	$\eta = 10^{-7}$	0.021	<u>0.9996</u>	72.8 (85.7)

Table 3: *Mean squared pricing error, R^2, and mean hedging error of the specialized option pricing network.*

5 Conclusion

We presented a continuous learning algorithm based on regularization. In order to identify the regularized neural network weights we suggested the Iterative Extended Kalman Filter which may be implemented efficiently as a learning rule. When applied to the task of pricing and hedging DAX options we found that a multi-layer perceptron trained with the suggested algorithm leads to a performance which is comparable to that of Black-Scholes benchmark models. In particular, the IEKF outperformed online backpropagation and extended Kalman filtering as a neural network learning rule in our experiments. For the option pricing problem, a considerable performance improvement could be obtained by using a new network architecture that is specifically designed for the purpose of option pricing.

A Numerical Aspects

One of the critical operations involved in applying the IEKF is the evaluation of the Hessian

$$\mathcal{H}_t(\theta) \equiv \nabla^2 \log N(y_t | f(z_t; \hat{\theta}_{t|t-1}))$$

in the filtering step (8). As we mentioned previously, we typically have to rely on numerical approaches compute this magnitude. A feasible strategy is to take numerical derivatives of the gradient of the log-likelihood $g_t(\theta) \equiv \nabla \log N(y_t|f(z_t;\hat{\theta}_{t|t-1}))$ which can be computed analytically for neural networks. More precisely, we build up our estimate H of $\mathcal{H}_t(\theta)$ in a row-by-row fashion, where row i of H is computed as

$$\frac{g_t(\theta+\varepsilon 1_i) - g_t(\theta - \varepsilon 1_i)}{2\varepsilon}.$$

1_i denotes the i-th unit vector and $\varepsilon = 10^{-6}$ in our implementation. In order to enforce symmetry we recommend to carry out the additional transformation

$$\tilde{H} \equiv 1/2HH'.$$

Note that in the central difference approach described above, the computation of H requires $2d$ evaluations of the gradient and converges for sufficiently small ε to the true derivative at a rate of $O(\varepsilon^2)$. For an alternative implementation based on forward differences an approximation could be obtained using only $d+1$ evaluations at the expense of the slower rate of convergence $O(\varepsilon)$.

B Derivatives of the Option Pricing Network

In order to apply IEKF learning to the special option pricing network defined by (12) and (13), we need to determine the derivatives of f with respect to the network weights α_k, $\tilde{\mu}_k$, and $\tilde{\sigma}_k$:

$$\begin{aligned}
\frac{\partial}{\partial \alpha_k} f(z;\theta) &= \frac{2\alpha_k}{\sum_l \alpha_l^2}[\xi_k - f(z;\theta)] \\
\frac{\partial}{\partial \tilde{\mu}_k} f(z;\theta) &= \tilde{\alpha}_k \frac{1}{\tilde{\sigma}_k}\left[z\Phi'(d_{k,1}) + e^{-\tilde{\mu}_k}\left(\tilde{\sigma}_k \Phi(d_{k,2}) - \Phi'(d_{k,2})\right)\right] \\
\frac{\partial}{\partial \tilde{\sigma}_k} f(z;\theta) &= \tilde{\alpha}_k \frac{1}{\tilde{\sigma}_k}\left[e^{-\tilde{\mu}_k} d_{k,1}\Phi'(d_{k,2}) - z d_{k,2}\Phi'(d_{k,1})\right] \\
\frac{\partial}{\partial z} f(z;\theta) &= \sum_k \tilde{\alpha}_k \frac{1}{\tilde{\sigma}_k}\left[\tilde{\sigma}_k \Phi(d_{k,1}) + \Phi'(d_{k,1}) - z^{-1} e^{-\tilde{\mu}_k}\Phi'(d_{k,2})\right] \\
\frac{\partial^2}{\partial z^2} f(z;\theta) &= \sum_k \tilde{\alpha}_k \frac{1}{z^2 \tilde{\sigma}_k^2}\left[z\tilde{\sigma}_k \Phi'(d_{k,1}) + z\Phi''(d_{k,1}) + e^{-\tilde{\mu}_k}(\tilde{\sigma}_k \Phi'(d_{k,2}) - \Phi''(d_{k,2}))\right].
\end{aligned}$$

C Pseudo-Code for the IEKF Algorithm

```
for t = 1,...,T do
    if t==1        //=== initialize to some values
        θ̂_{t|t} = θ_0
        Ω_{t|t} = Ω_0
    else        //=== initial EKF step based on the previous value
        Ω_{t|t} = −[∇² log p(y_t|z_t, θ̂_{t|t−1}) − Ω^{−1}_{t|t−1}]^{−1}
        θ̂_{t|t} = θ̂_{t|t−1} + Ω_{t|t} ∇ log p(y_t|z_t, θ̂_{t|t−1})
    end

    do until convergence
        for s = t,...,1 do        //=== backward EKF smoothing pass
            B_s = Ω_{s|s} Ω^{−1}_{s+1|s}
            θ̂_{s|T} = θ̂_{s|s} + B_s(θ_{s+1|T} − θ̂_{s+1|s})
            Ω_{s|T} = Ω_{s|s} + B_s(Ω_{s+1|T} − Ω_{s+1|s})
        end

        for s = 1,...,t do        //=== forward EKF prediction and filtering pass
            Ω_{s|s} = −[∇² log p(y_s|z_s, θ̂_{s|s−1}) − Ω^{−1}_{s|s−1}]^{−1}
            θ̂_{s|s} = θ̂_{s|s−1} + Ω_{s|s} ∇ log p(y_s|z_s, θ̂_{s|s−1})
        end
    end
end
```

References

Ahlberg, J. H., Nilson, E. N., and Walsh, J. L. (1967). *The Theory of Splines and Their Application.* Academic.

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

Fahrmeir, L. (1992). Posterior mode estimation by extended Kalman filtering for multivariate dynamic generalized linear models. *Journal of the American Statistical Assiciation*, 87(418):501–509.

Hull, J. C. (1997). *Options, Futures, and Other Derivatives.* Prentice Hall, third edition.

Hutchinson, J. M., Lo, A., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, 49:771–818.

Reinsch, G. (1967). Smoothing by spline functions. *Numerische Mathematik*, 10:177–183.

Sage, A. P. and Melsa, J. L. (1971). *Estimation Theory with Applications to Communications and Control.* McGraw Hill.

Szeliski, R. (1996). Regularization in neural nets. In Smolensky, P., Mozer, M. C., and Rumelhart, D. E., editors, *Mathematical Perspectives on Neural Networks*, chapter 14, pages 497–532. Lawrence Erlbaum.