Palestine Journal of Mathematics

Vol. 3(2) (2014) , 175–184

PREDICTION OF CO_2 FUTURE PRICES FOR ENERGY RISK MANAGEMENT VIA NEURAL NETWORK ADAPTED STOCHASTIC PROCESSES

Martin Rainer and Stefan Giebel

Communicated by Ayman Badawi

MSC 2010 Classifications: Primary 60G25, 62M20, 62M45, 68T05; Secondary 91B25, 91B70, 91B76.

Keywords and phrases: learning algorithms, neural networks, stochastic models, prediction, mathematical finance, market risk, energy markets, environmental economics, CO_2 allowances.

Abstract Energy risk management requires an efficient prediction of CO_2 future prices. We apply a neural network calibrated stochastic model to predict day ahead future prices suitable for risk management with a given time horizon (e.g. 10 business days). The model is training neural perceptron layers to learn the weights of historical time series points within a past horizon equal to the the memory depth (e.g. previous 10 business days). With this weights, the parameters of the stochastic model are optimized, and the day ahead carbon future price is computed.
Iterating these steps, we obtain price forecasts, which demonstrate the learning efficiency of our method, which essentially corresponds to a stochastic process with time-dependent parameters, the dynamics of the parameters being themselves learned continuously by the neural network. The back propagation in training the previous weights is limited by the memory depth. The latter is the analogue of the maximal time lag of an autoregressive processes.

1 Introduction

The issue of correctly pricing forward contracts is important for all market participants in the energy trading sector. Presently, embedding CO_2 trading into the risk assessment and mitigation strategies has become standard. Hence CO_2 emissions nowadays have emerged as a characteristic new risk factor of the energy production and trading sector. In order to integrate CO_2 emissions into the risk-return analysis, an efficient and realistic prediction of their future prices is crucial. Day ahead prices of CO_2 emission allowances can be used for risk management with a given time horizon (e.g. 10 days).
An efficient stochastic model for energy prices and their derivatives was considered in [1], and for wind energy a certain class of such stochastic models has been compared in [2] to an alternative model with less structure but with a learning calibration (via a 2-layer MLP neural network). In any case, the calibration of the stochastic process is crucial. In the following we will focus on this problem.
Usually, stochastic model parameters for price processes of traded energy or related energy and climate resources are calibrated structurally to a given history of the processes by via maximum likelihood estimation (MLE) and nonlinear regression methods. These methods are however neglecting the fact that the market making agents do not weight all historical prices equally but rather put different weights on prices at different points of the historical time series. The weights are different due to both, different importance given to different structures (e.g. Elliott waves) of the time series, and memory decaying with the size of the time lag.
For standard nonlinear regression methods based on the Levenberg-Marquardt method [3], [4] there exists a well-elaborated theory, including proofs of convergence in sufficiently well-behaved local regions of the parameter space. Nevertheless, in larger regions of the parameter space they usually fail to detect the correct minimum among several local ones. Therefore, simulated annealing [5] or adaptive lattice algorithms [6] have been proposed as alternatives for global calibration of nonlinear functions. Apart from that, a common weakness of all the conventional calibration methods is that, they are applied once for all, rather than dynamically.
As we will see below, learning neural perceptron layers may provide a method to calibrate stochastic asset models with more realistic and dynamical weights on the historical input data.
Neural networks as mathematical methods have been developed in particular in the context of pattern recognition (see e.g. [7]). their applications in this context have a wide range including such different topics like criminal profiling [8], electronic noses for odour detection and classification [9], tumor shape analysis [10], and many more.

In mathematical finance, neural networks of multilayer perceptron (MLP) type have only in recent years being investigated as a serious alternative to statistical estimation such as MLE in the context of stochastic processes (see e.g. [11]).
Below we use a neural network calibrated stochastic model to predict day ahead prices for CO_2 emission permits. This new synthesis of neural networks and stochastic processes was tested for prediction of predict financial risk factors in [12] and [13], and for energy and climate related risk factors in [14]. We demonstrate the neural network calibration method as a serious alternative to the standard MLE methods.
We proceed as follows. First, we review the stochastic processes and the calibration methods usually applied in models for energy-related risk factors. Then we review the MLP neural network methodology.
Following the approach of [2] and [14] we combine stochastic process and neural network, calibrating the process parameters via MLP-estimation.
As an example, we apply our model to CO_2 emission allowances quoted at the Inter-Continental Exchange (ICE) under the ISIN number XC000A0C4KJ2.

2 Stochastic processes for energy risk factors

The price of CO_2 emission allowances constitutes a typical energy related risk factor. In this section we review the general setting for the stochastic processes underlying to energy prices, related risk factors, and further derivatives.
We consider a stochastic asset modeled over an semimartingale independent increment RCLL (cadlag) process I in $\mathbb{R}$, having the Levy-Kintchine representation

$$I(t) \quad = \quad \gamma(t) + M(t) + \int_0^t \int_{|z|<1} z\tilde{N}(ds,dz) + \int_0^t \int_{|z|\geq 1} zN(ds,dz), \tag{2.1}$$

where γ is of finite variation on finite intervals, M is a local continuous martingale with finite quadratic variation C, $N(\cdot)$ is a random jump measure and $\tilde{N}(\cdot) := N(\cdot) - \mathbb{E}[N(\cdot)]$ is the compensated random jump measure.
In the special case when I is stationary, the compensator factorizes,

$$\mathbb{E}[N(dt,dz)] \quad = \quad dt\ell(dz) \quad . \tag{2.2}$$

Here I is a Levy process with Levy measure $\ell(\cdot)$ and characteristic triplet $(\gamma, C, \ell(\cdot))$.
Given I, a CAR(n) process $\mathbf{X}$ in $\mathbb{R}^n$ is defined as

$$\begin{aligned} dX_q &= X_{q+1}\, dt, \quad q = 1,\ldots,n-1 \\ dX_n &= -\left(\sum_{q=1}^{n} \alpha_q(t) X_q\right) dt + \sigma(t) dI(t) \quad , \end{aligned} \tag{2.3}$$

which can be written in compact matrix form as

$$d\mathbf{X} \quad = \quad \mathbf{A}(t)\mathbf{X}dt + \sigma(t)\mathbf{e}_n dI(t) \quad , \tag{2.4}$$

with $n \times n$ matrix

$$A(t) \quad := \quad \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & . & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \\ -\alpha_1(t) & -\alpha_2(t) & \cdots & \cdots & -\alpha_n(t) \end{pmatrix} \quad . \tag{2.5}$$

The CAR(n) case above is a special case of a more general CARMA(n,m) process, which can be defined generally similar as in [15]. There it was introduced for the case of I being a second-order Levy processes with $E[L(1)^2] < \infty$.
Finally, the asset process S in $\mathbb{R}$ is modeled on the basis of the first component X_1 of the CAR process. It is modeled through a real-valued function $G(x_1, x_2)$ with $x_1 = g(t)$ given as as a deterministic function of time, and $x_2 = f(X(t))$ as a function of the given CAR(n) process X.
The function $g(t)$ is supposed to capture all known deterministic features such as trend, seasonalities, etc. Its parameters can be fitted to the historical data via a least square regression.

The function $f(X)$ is a deterministic transformation of the stochastic process X. In practice, martingale asset processes S are often related to a fundamental stochastic process X via

$$dS = S^{\beta} dX \quad , \beta \in [0,1] \tag{2.6}$$

Here the inverse of the solution $S = f(X)$ is determined (modulo integration constant) as

$$d(f^{-1}(y)) = \frac{dy}{y^{-\beta}} \quad . \tag{2.7}$$

The two most important special cases are $\beta = 0$, i.e. $f(X) = X$. and $\beta = 1$, i.e. $f(x) = e^X$. The initial condition $X(0) = 0$ implies $f(0) = 0$ in the former case, and $f(0) = 1$ in the latter. The choice of G should be such that the stochastic part of the process adds to the deterministic part in a natural way. Hence for $\beta = 0$, $f(X) = X$, we choose $G(x_1, x_2) = x_1 + x_2$, yielding

$$S(t, X(t)) = g(t) + f(X(t)) \quad , \tag{2.8}$$

and for $\beta = 1$, $f(X) = e^X$, we choose $G(x_1, x_2) = x_1 \cdot x_2$, yielding

$$S(t, X(t)) = g(t) e^{X(t)} = e^{\int_0^t \mu(s) ds + X(t)} \quad , \tag{2.9}$$

with $\mu(t) := \frac{d}{dt} \ln g(t)$.
Summing up, S is given as

$$S(t, X(t)) = C\left(g(t), f(X_1)\right) \tag{2.10}$$

$$dX_q = X_{q+1}\, dt, \quad q = 1, \ldots, n-1 \tag{2.11}$$

$$dX_n = -\left(\sum_{q=1}^{n} \alpha_q(t) X_q \right) dt + \sigma(t) dI(t) \quad . \tag{2.12}$$

Discretization of the CAR(n) process X with $dt_i := t_{i+1} - t_i$ reads

$$X_{q+1}(t_i) = \frac{X_q(t_{i+1}) - X_q(t_i)}{dt_i}, \quad q = 1, \ldots, n-1, \tag{2.13}$$

$$X_n(t_{i+1}) - \left(\left(\frac{1}{dt_i} - \alpha_q(t_i) \right) X_n(t_i) - \sum_{q=1}^{n-1} \alpha_q(t_i) X_q(t_i) \right) dt_i = \sigma(t_i) \varepsilon_i \quad , \tag{2.14}$$

where ε_i is a random number distributed according to the pdf of $dI(t_i)$. According to (2.14) the coefficients functions $\alpha_q(t)$, $q = 1, \ldots, n$, can determined by regression. For time-independent constants α_q, the regression becomes linear. Recursion of (2.13), inserting into (2.14), and resolving for X_1 then shows that the discretized CAR(n) process is in fact an AR(n) process,

$$X_1(t_{n+1}) = \sum_{q=1}^{n} \gamma_q \, X_1(t_q) \tag{2.15}$$

with parameters γ_q depending linearly on the original reversion coefficients α_q.
In the special case where X is CAR(1), it is a mean reverting Ornstein-Uhlenbeck (OU) process

$$dX = -\alpha(t) X dt + \sigma(t) dI(t) \quad . \tag{2.16}$$

Such an OU process is known to have the unique strong solution

$$X(t) = X_0 e^{\int_0^t \alpha(s) ds} + \int_0^t \sigma(u) e^{\int_0^u \alpha ds} dI(t) \quad . \tag{2.17}$$

The OU process driven by Brownian motion was applied in [16] for yield modeling with dynamically controlled time-dependent volatility.
For $\beta = 1$, the asset process then follows as

$$\begin{aligned} S(t) &= e^{\int_0^t (\mu(s) ds + X(t)} \\ &= e^{\int_0^t (\mu(s) - \alpha(s) X(s)) ds + \int_0^t \sigma(t) dI(t)} \quad . \end{aligned} \tag{2.18}$$

The general Ito-formula for semimartingales, can be applied to $S = f(X)$, in order to yield an explicit form of the dynamics of $S(t)$.
Specializing to the case of a standard Brownian motion $dI = dW$, $\mu(t) = \mu_y$ and $\sigma(t) = \sigma_y$ constant, the yield $y := \ln S$ follows the SDE

$$dy = \mu_y dt + \sigma_y dW \tag{2.19}$$

$$\mu_y := \mu - \frac{\sigma^2}{2} \tag{2.20}$$

$$\sigma_y := \sigma \tag{2.21}$$

while

$$dS = \mu S dt + \sigma S dW. \tag{2.22}$$

The solution of the SDE (2.19) now reads

$$\begin{aligned} y(t) &= y(0) + \mu_y t + \sigma_y \int_0^t dW \\ &= y(0) + \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma \int_0^t dW \quad . \end{aligned} \tag{2.23}$$

Here $\int_0^t dW$ is time-integrated white noise, normally distributed with

$$\int_0^t dW \sim N(0, \sqrt{t}). \tag{2.24}$$

Hence for simulation purpose, we may replace it by a dynamically scaled standard normal random variable ϵ,

$$\int_0^t dW \stackrel{d}{=} \sqrt{t}\epsilon \quad . \tag{2.25}$$

With $S_0 := e^{y(0)}$, the solution of the SDE (2.22) may be written as

$$\begin{aligned} S(t) &= e^{y(t)} \\ &\stackrel{d}{=} S_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma\sqrt{t}\epsilon} \quad , \end{aligned} \tag{2.26}$$

and its expected value is

$$\begin{aligned} E[S(t)] &= E[e^{y(t) + \frac{1}{2}\sigma^2 t}] \\ &= S(0)e^{\mu t} \quad . \end{aligned} \tag{2.27}$$

Setting $\epsilon := k$, the $k\sigma$ bound of (2.19) is obtained

$$y_{k\sigma}(t) = y(0) + \left(\mu - \frac{\sigma^2}{2}\right) t + k\sigma\sqrt{t} \quad , \tag{2.28}$$

which yields a corresponding bound for the log-normal process

$$\begin{aligned} S_{k\sigma}(t) &= S(0)e^{\left(\mu - \frac{\sigma^2}{2}\right)t + k\sigma\sqrt{t}} \\ &= S_{med}(t)e^{k\sigma\sqrt{t}} \quad . \end{aligned} \tag{2.29}$$

The case $k = 0$ yields the median curve

$$S_{med}(t) = S(0)e^{\left(\mu - \frac{\sigma^2}{2}\right)t} \quad , \tag{2.30}$$

which is related to the forward (expected) price

$$F(t) := E[S(t)]$$

by

$$F(t) = S_{med}(t)e^{\frac{1}{2}\sigma^2 t} \quad . \tag{2.31}$$

Therefore, corresponding bounds for the forward price are given as

$$\begin{aligned} F_{k\sigma}(t) &= S_{med}(t)e^{\frac{1}{2}\sigma^2 t + k\sigma\sqrt{t}} \\ &= F(t)e^{k\sigma\sqrt{t}} \quad . \end{aligned} \tag{2.32}$$

3 MLE parameter calibration

The discretization of Ito-processes at t_{i-1} requires to consider on the *forward* time interval $[t_{i-1}, t_i]$, the difference

$$dt_i \quad := \quad t_i - t_{i-1} \quad , \tag{3.1}$$

the finite difference of the log-normal random variable

$$dS_i \quad := \quad S_i - S_{i-1} \quad , \tag{3.2}$$

the return

$$\frac{dS_i}{S_{i-1}} \quad := \quad \frac{S_i}{S_{i-1}} - 1 \quad , \tag{3.3}$$

and the finite difference of the corresponding normal random variable

$$dy_i \quad := \quad \ln \frac{S_i}{S_{i-1}} \quad . \tag{3.4}$$

Correspondingly, the (dt_i-weighted) estimators are

$$\begin{aligned} \hat{\mu} \quad &:= \quad \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dt_i \left[\frac{1}{dt_i} \frac{dS_i}{S_{i-1}} \right] \\ &= \quad \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n \frac{dS_i}{S_{i-1}} \end{aligned} \tag{3.5}$$

$$\hat{\mu}_y \quad := \quad \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dy_i \tag{3.6}$$

$$\begin{aligned} \hat{\sigma}^2 \quad &:= \quad \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dt_i \left[\frac{1}{dt_i} \left(\frac{dS_i}{S_{i-1}} \right)^2 \right] \\ &= \quad \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n \left(\frac{dS_i}{S_{i-1}} \right)^2 \end{aligned} \tag{3.7}$$

4 Elements of neural networks

Neural networks have been developed originally in order to understand the cognitive processes. Nowadays there are a lot of applications of neural networks as a mathematical method in quite different disciplines.

The term "neural networks" points to the model of a nerve cell, the neuron, and the cognitive processes carried and driven by the network of interacting neurons. A neuron perceives chemical and physical excitement from the environment by its dendrites. The neuron is processing this incoming data and sending the information to other neurons via axons and synapses.

The neuron:

McCulloch and Pitts implemented the biological processes of a nerve cell for the first time in a mathematical way [17]. Nerve cells have to access and process incoming data in order to evaluate target information. Therefore the corresponding neural networks are called supervised neural networks. An unsupervised neural network has no target and is similar to a cluster algorithm.

The data consists of n variables $x_1, \ldots, x_n$ on binary scale. For data processing, the ith variable x_i is weighted with w_i, normalized with $|w_i| \leq 1$.

The multiplication of x_i with w_i determines the relevance of x_i for a target y. The value w_i reflects the correlation between the input variable and the target, the sign indicating the direction of the influence of the input variable on the target. Weighting the input variables for a target variable is similar to discriminant analysis. Hence we are able tot understand the mathematical process, to determine the direction of influence and the relevance.

The critical quantity for the neuron is the weighted sum of input variables

$$q := \sum_{i=1}^n w_i \cdot x_i = w_1 \cdot x_1 + ... + w_n \cdot x_n \quad . \tag{4.1}$$

For a target y with binary scale, a threshold S is needed. Crossing the threshold yields 1 and falling below the threshold yields 0. Hence the activation function F can be written as

$$F(q) \quad = \quad \begin{cases} 1, & \text{if } x > S \\ 0, & \text{if } x \leq S \end{cases} \tag{4.2}$$

In comparison to discriminant analysis, for neural networks the threshold S has to be assigned, depending on properties of the target; it can not be derived from the data in a straightforward manner. Neural networks usually include no assumption about the data. Rather they are a numerical method.
With the input (4.1) of the activation function, we obtain $y = F(q)$ as

$$y \quad = \quad 1, \quad \text{if } \sum_{i=1}^{n} w_i \cdot x_i > S$$

$$y \quad = \quad 0, \quad \text{if } \sum_{i=1}^{n} w_i \cdot x_i \leq S$$

5 Multi-layer perceptrons

In general a given target may be reached only up to a certain error. Given a certain measure $E(\tilde{y}, y)$ for the distance between the given target state y and the state $\tilde{y}$ computed by the neural network, learning of the neural network corresponds to minimization of $E(\tilde{y}, y)$.
The following training algorithm is inspired by Rumelhart, Hinton and Williams [18]. The total error measure over all states of a given layer is defined as

$$E_{total}(\tilde{y}, y) := \frac{1}{2} \sum_{k=1}^{N} (\tilde{y}_k - y_k)^2 \quad . \tag{5.1}$$

It will be used below to reset the weights in each layer of the neural network.
For simplicity, we consider now a 2-layer perceptron network, which also will be sufficient below for our purpose of calibrating the stochastic process (2.19).
The processed state $\tilde{y}$ of the neural network is computed by the following steps.
First the critical parameter for the first layer is computed from n weighted input values as $\sum_{i=1}^{n} w_i \cdot x_i$. We consider a hidden output layer with m neurons. For $j = 1, \ldots, m$, let g_j be the activation function of the j-th neuron of the hidden layer, with an activation value of h_j, given as

$$h_j = g_j(\sum_{i=1}^{n} w_i \cdot x_i) \quad . \tag{5.2}$$

Usually for all neurons of a given layer a common activation function $g = g_1, \ldots, g_m$, e.g. a sigmoid function, is used.
Next, the output of the previous (hidden) layer becomes the input of the next layer, and the activation proceeds analogously to the previous layer.
Let f be the activation function of the pre-final (here the second) output layer. Then the pre-final critical value is

$$q = f(\sum_{j=1}^{m} u_j \cdot h_j) \quad . \tag{5.3}$$

Finally, the pre-final critical value q is interpreted by a final activation function F yielding

$$\tilde{y} = F(q) \tag{5.4}$$

as a final state value computed from the neural network with the given weights of the input variables from input and hidden layers.
Now the neural network performs a training step by modifying the weights of all input layers. The learning mechanism the weights is determined by the target distance measure

$$E \quad = \quad \frac{1}{2} \sum_{i=1}^{n} (y^i - \tilde{y}^i)^2 \quad .$$

The weights of both layers are changed according to the steepest descent, i.e.

$$\Delta w_i := \frac{\partial E}{\partial w_i} \tag{5.5}$$

$$\Delta u_j := \frac{\partial E}{\partial u_j} \tag{5.6}$$

With a learning rate α, which should be adapted to the data, the weights are changed as follows:

$$w_i^{new} = w_i^{old} - \alpha \cdot \Delta w_i \tag{5.7}$$

$$u_j^{new} = u_j^{old} - \alpha \cdot \Delta u_j \tag{5.8}$$

The necessary number of iterations depends on the requirements posed by the data, the user, and the discipline.

6 Neural networks related to stochastic models

In this section we apply neural networks to the estimation of the process (2.19).
First we demonstrate a simple combination using a 1-layer perceptron network. In this one layer neural network the variance σ^2 and the mean μ are weighted to predict the target value.

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{6.1}$$

With $\tilde{y} = \mu_0 u_1 + \sigma_0 u_2$ we obtain

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - (\mu_0 u_1 + \sigma_0 u_2))^2 \tag{6.2}$$

$$\Delta u_1 = -\mu_0 \sum_{i=1}^{n}(y_i - (\mu_0 u_1 + \sigma_0 u_2)) \tag{6.3}$$

$$\Delta u_2 = -\sigma_0 \sum_{i=1}^{n}(y_i - (\mu_0 u_1 + \sigma_0 u_2)) \tag{6.4}$$

A more complex combination uses a 2-layer neural network, where we compute variance and mean via weighted input variables. The first estimate of the mean is

$$\mu_0 = \frac{1}{n}\sum_{i=1}^{n} w_i x_i \quad ,$$

and for the variance it is

$$\sigma_0^2 = \frac{1}{n-1}\sum_{i=1}^{n}(w_i x_i - \mu_0)^2 \quad .$$

The neural network is then trained by adjusting the weights w_i of the first layer, and the weights u_j of the second layer, according to the respective sensitivities (5.5) and (5.6) of the error function (6.1).

7 Parameter calibration via perceptron layers

In this section we demonstrate the dynamical calibration of process parameters for (2.19).
Here, different points of the historical time series, receive different weights, which are learned dynamically when propagating through the historical training set.
In the first layer μ and σ are determined on the basis of weighted values $w_i y_i$ of the time series of y-values.

$$\hat{\mu}_y := \frac{1}{\sum_{i=1}^{n} dt_i}\sum_{i=1}^{n} w_i dy_i \tag{7.1}$$

$$\hat{\sigma}^2 := \frac{1}{\sum_{i=1}^{n} dt_i}\sum_{i=1}^{n} w_i dt_i \left[\frac{1}{dt_i}\left(\frac{dS_i}{S_{i-1}}\right)^2\right] \tag{7.2}$$

In the second layer a new value is determined with

$$\hat{y}_{i+1} \quad = \quad y_i + (u_{1,i}\hat{\mu}_y + u_{2,i}\hat{\sigma})dt_i \tag{7.3}$$

corresponding to an yet undetermined risk measure. The ratio

$$\lambda_i \quad := \quad \frac{u_{2,i}}{u_{1,i}} \tag{7.4}$$

has an econometric interpretation of a time-dependent market price of risk factor.
Including this value in the regression the process for determining μ and σ in the first layer is repeated. In contrast to normal neural networks the first weights are used according to the formula for μ and σ^2.
In every iteration the weights in the first and second layers are changed according to the steepest descent.
In our example the memory length is $n = 10$ business days to the past, which are used to predict next value. In the next step, the one deals with the weights w_{i+1} rather than w_i.

8 Numerical results for CO_2 emission allowances

We test our method with CO_2 emission allowances XC000A0C4KJ2 quoted at ICE. We consider the half a year period from 08.01.2010 to 09.07.2010.
In every prediction step, $n = 10$ previous values are used to predict the next one. The model was implemented in Mathematica. In figure 1 the predicted values are shown in comparison to reality.

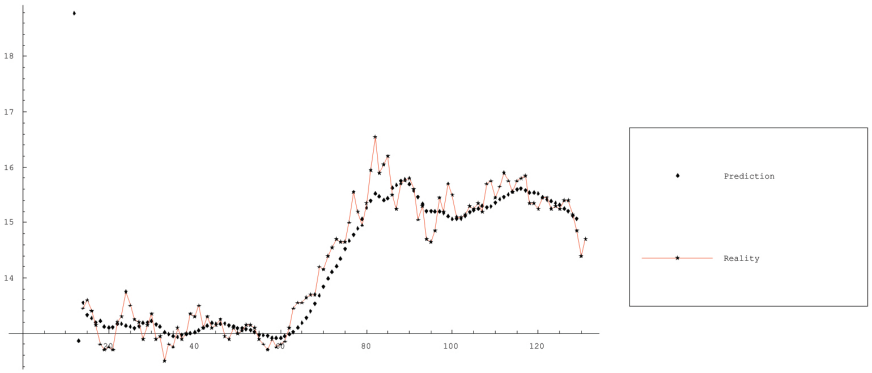

Figure 1. Price of CO_2 emission allowances XC000A0C4KJ2

Considering a target time horizon of 10 days, good agreement of the predicted values with the reality is achieved. The results show that the prediction is much less volatile than the reality, i.e. our prediction is tailored particularly for a risk management oriented towards time scales the memory length of 10 days or more. By a continuously neurally adapting calibration of the model-parameters to an up-to-date history (with memory of 10 days) the model learns to capture the optimal trend according to movements of the past 10 days (the memory horizon). Due to the memory length of 10 days, day-by-day short term fluctuations are suppressed in the prediction.

9 Conclusion and outlook

In our combined model of neural networks and stochastic processes, the neural transformation processes adapt in such a manner that, from a continuously updated history of fixed length the network is continuously learning the process parameters. As compared to traditional calibration, our neural methods is taking into account the historical process in a more detailed and dynamical manner. As a result, the parameters of the stochastic process are better calibrated than with traditional methods. As our example of CO_2 allowance prices demonstrates: Even a relatively simple stochastic process, in combination together with a smart neural network learning continuously the right parameters, the model yields satisfactory predictions. This demonstrates the efficiency of our combined stochastic-neural approach.

The research is to be continued to apply our neural network approach also to more demanding stochastic processes, as they may be required for estimations of other energy prices, such as gas or electricity prices. There, the different phases of normal and spiky modes of volatility have to be taken into account. This can be done on the one hand by regime switching models, on the other hand by working with more advanced semi-martingale processes including jumps, also with time-dependent frequency.
Finally, in risk management scenario simulations are a very common method. Note in this context, any such simulation can be built on the basis of the described neurally adapting stochastic process by simply adding the scenario-specific shifts on top.
Our example applied a 2-layer neural network in order to learn the optimal parameters of the CO2 future price process. It is a typical and very efficient application of a learning algorithm (in the sence of statistical learning theory) for adaptive predictions. Forthcoming research will explore futher such learning algorithms.

References

[1] Aydın, N.S., Rainer, M., 2013, Valuation of power swing options, Journal of Energy Markets (2013) **6**(3) 89-110.

[2] Giebel, S., Rainer, M., Aydın, N.S., 2013, Simulation and prediction of wind speeds: a neural network for Weibull, ENAMEC Preprint 2011 (Energy Finance Conference, Rotterdam); publ. in: J. Iran. Stat. Soc. (2013) **12**(3), 293-319.

[3] Levenberg, K., 1944, A Method for the Solution of Certain Non-Linear Problems in Least Squares *Quart. Appl. Math.,* **2**, 164-168.

[4] Marquardt, D., 1963, An Algorithm for Least-Squares Estimation of Nonlinear Parameters *SIAM J. Appl. Math.* **11**, 431-441.

[5] Deutsch, H.-P., 2004, Der Simulated Annealing Algorithmus. Sect. 33.2.3 in: *Derivate und Interne Modelle* (3rd edn) (Stuttgart: Schäfer-Poeschel).

[6] Rainer, M., 2009, Calibration of stochastic models for interest rate derivatives, *Optimization* **58**, 373-388.

[7] Bishop, C.M., 1995, *Neural networks for pattern recognition*, Clarendon Press, Oxford.

[8] Giebel, S., 2010, Zur Anwendung Neuronaler Netze in den Sozialwissenschaften (Application of neural networks in social science), Dr. Kovac, Hamburg

[9] Giebel, S., Rainer, M., Frechen, F.-B., 2011, Forecasting the Odor Concentration using Electronic Noses and time series measurements by a Combination of Neural Networks and Stochastic Processes, *Proc. IWA 4th Conference on odours and VOCs*, Vitoria, Brazil.

[10] Giebel, S., 2009, Application of Neural Networks for characteristic shapes in medicine, METU Ankara, available on http://www3.iam.metu.edu.tr

[11] McNelis, P.D., 2005, *Neural networks in finance: Gaining predictive edge in the market*, Elsevier, London.

[12] Giebel, S., Rainer, M., 2009, Forecasting Financial Asset Processes: Stochastic Dynamics Calibrated via Learning Neural Networks *Proceedings 4th. Statistical Days at the University of Luxembourg*, publ. in: *Bull. Soc. Sciences Médicales Luxembourg* **10**(1), 91-107 (2010).

[13] Giebel, S., Rainer, M., 2013, Neural network calibrated stochastic processes: forecasting financial assets, *Centr. Europ. J. Op. Res.* **21**(2), 277-293 .

[14] Giebel, S., Rainer, M., 2011, Stochastic processes adapted by neural networks with application to climate, energy, and finance, *Appl. Math. Comput.* **218**, 1003-1007.

[15] Brockwell, P. J., and Marquardt, T., 2005, Levy-driven and fractionally integrated ARMA processes with continuous time parameter, *Statistica Sinica* **15**, 477-494.

[16] M. Rainer, 2009, Calibration of Ornstein-Uhlenbeck yield processes and dynamical control of short rate models, *Workshop on Recent Developments in Applied Probability and Statistics* METU Ankara.

[17] McCulloch, W.S., Pitts, W., 1943, A logical calculus of the idea immanent in neural nets, *Bulletin of Mathematical Biophysics*, 115-137.

[18] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986, Learning internal representation by error propagation, 318-362 in: *Rumelhart, D.E., McClelland, J.L., PDP Research Group, Parallel. Distributes Processing*, MIT Press, Cambridge.

Author information

Martin Rainer, ENAMEC Institut Würzburg, Germany; Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey; FZ Risk Management, University of Würzburg, Germany.
E-mail: martin.rainer@enamec.de

Received: October 13, 2013. Accepted: December 29, 2013.

Stefan Giebel, University of Luxembourg, Luxembourg; ENAMEC Institut Würzburg, Germany.
E-mail: stefan.giebel@enamec.de

Received: October 6, 2013

Accepted: January 7, 2014