A Comparison of Nonlinear Kalman Filtering Applied to Feed–forward Neural Networks as Learning Algorithms

Wieslaw Pietruszkiewicz
SDART Ltd
One Central Park, Northampton Road
Manchester M40 5WW, United Kindgom
Email: wpietrus@sdart.co.uk

***Abstract*—In this article we present an application of Kalman filtering in Artificial Intelligence, where nonlinear Kalman filters were used as a learning algorithms for feed–forward neural networks. In the first part of this article we have examined two modern versions of nonlinear filtering algorithms i.e. Unscented Kalman Filter and Square Root Central Difference Kalman Filter. Later, we present performed experiments, where we have compared UKF and SRCDKF with an reference algorithm i.e. Error Backpropagation being the most popular neural network learning algorithm. To prove filters high learning abilities in case of noisy problems, we have used a noisy financial dataset during the experiments. This dataset was selected due to un-easily separable classes subspaces. The results of experiments, presented in the last part of this paper, show greater accuracy for nonlinear Kalman filters that overperformed popular Error Backpropagation learning algorithm.**

***Index Terms*—Neural Networks, Unscented Kalman Filter, Square Root Central Difference Kalman Filter, Bankruptcy prediction**

I. INTRODUCTION

Neural networks are one of the most popular Artificial Intelligence methods. They are used in regression or classification tasks. Their flexibility and ability to work as "black-box" causes that many researchers and practitioners chose them among the other algorithms. The crucial part of constructing neural networks, apart of selecting its appropriate architecture, is to use an accurate and robust learning algorithm. The most popular learning algorithm for feed–forward neural networks is Gradient Descent Error Backpropagation, which is easy to apply and is often efficient enough. However it gives unsatisfactory results when the learning material was affected by the noise. Herein we present the results of experiments, that were performed using noisy data. We compared 2 nonlinear Kalman learning algorithms i.e. *Unscented Kalman Filter* and *Square Root Central Difference Kalman Filter* with *Gradient Descent Error Backpropragation*. The two presented filters and other Kalman filters are very popular algorithms, having various applications in control systems or navigation. However, applications of Kalman filtering in Artificial Intelligence are found too rarely, while modern derivative–free nonlinear filters are almost "black-boxes" comparing to the older generation of filters.

There are some examples of the successful Kalman filtering usage for neural networks training. In [1] researchers presented various EKF approaches to neural network learning. Another research [2] described how EKF was applied in currency prediction done by neural network. EKF was also used to train neural network that was applied in chemistry [3]. In [4] comparison of EKF training with Levenberg-Marquardt algorithm was shown. The other papers describe different types of filters e.g [5] compared EKF as well as UKF or [6] presented a comparison of EKF, UKF and Divided Difference Filter (DDF). UKF was also used to learn neural network applied to mechanics [7], while [8] described an usage of UKF in neural framework for unmanned aerial vehicles.

II. THE UNSCENTED TRANSFORMATION AND UNSCENTED KALMAN FILTER

Real life applications of data processing often have to deal with noisy datasets. When an observed signal is a superposition of pure signal and noise, its usefulness is decreasing as well as accuracy of data processing methods. If we assume that observed object or process may be represented as a dynamic system, them its general form will be:

$$x_{k+1} = G(x_k, u_k, \nu_k) \tag{1}$$

$$y_k = H(x_k, n_k) \tag{2}$$

where: G – process equation, H – observation equation, x_k - state variables, u_k - system inputs, y_k - system outputs, ν_k - process noise and n_k - observation noise.

Kalman filters (sometimes loosely related) belong to one family, being a popular group of filtering algorithms. They have many applications in different technical problems e.g estimation of observed variables disturbed by noise or estimation of parameters of an unknown system – with is a typical machine learning task.

Retransformed system for parameter estimation transforms into:

$$w_{k+1} = w_k + r_k \tag{3}$$

$$d_k = g(x_k, w_k + e_k) \tag{4}$$

where: w_k – is stationary process relating to parameters (weights), r_k – is an artificial noise driving w vector and d_k – is an observation of w vector.

The estimation of parameter is impossible to be achieved by an original Kalman filter (KF - see [9]), due to its purely linear construction. This limitation does not apply to a group of nonlinear Kalman filters e.g. *Extened Kalman Filter* (EKF), *Unscented Kalman Filter* (UKF) or *Central Difference Kalman Filter* (CDKF). While EKF is the oldest and the most popular nonlinear algorithm, in a matter of fact it is very simple extension of KF to nonlinear problems, done by simple linearization of estimated systems in form of first order Taylor series. Therefore, EKF is not accurate enough in case where strong nonlinearity exists.

An example of more accurate nonlinear Kalman filter can be Unscented Kalman filter that incorporates Unscented Transformation (UT in abbrev. – see [10] or [11]). The main idea of UT is to calculate statistics of random variables by propagating a set of specially chosen points (sigma points) via the real model of the system and calculating their statistic parameters, comparing to approximation achieved from inaccurate system's simplification (EKF). The detailed steps of UKF in form for parameter estimation are (more details and other UKF forms may be found in [12], [13] or [14]):

Step I – Initialization

$$\hat{x}_0 = E[w] \tag{5}$$

$$P_{w_0} = E\left[(x - \hat{w}_0)\,(w - \hat{w}_0)^T\right] \tag{6}$$

For $k \in \{1, \ldots, \infty\}$

Step II – Time update:

$$\hat{w}_k^- = \hat{w}_{k-1} \tag{7}$$

$$P_{w_k^-} = P_{w_{k-1}} + R_{r_{k-1}} \tag{8}$$

Step III – Calculation of sigma points for measurement–update:

$$\chi_{k|k-1} = \left[\hat{w}_k \quad \hat{w}_k + \tau\sqrt{P_{w_k^-}} \quad \hat{w}_{\overline{k}} - \tau\sqrt{P_{w_k^-}}\right] \tag{9}$$

Step IV - Measurement update equations:

$$\Upsilon_{k|k-1} = \mathrm{g}\left(x_k, \chi_{k|k-1}\right) \tag{10}$$

$$\hat{d}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \Upsilon_{i,k|k-1} \quad or \quad \hat{d}_k^- = g(x_k, \hat{w}_k^-) \tag{11}$$

$$P_{\tilde{d}_k} = \sum_{i=0}^{2L} w_i^{(c)} \left(\Upsilon_{i,k|k-1} - \hat{d}_k^-\right)\left(\Upsilon_{i,k|k-1} - \hat{d}_k^-\right)^T + R_{n_k} \tag{12}$$

$$P_{w_k,d_k} = \sum_{i=0}^{2L} w_i^{(c)} \left(\chi_{i,k|k-1} - \hat{w}_k^-\right)\left(\Upsilon_{i,k|k-1} - \hat{d}_k^-\right)^T \tag{13}$$

$$K_k = P_{w_k,d_k} P_{\tilde{d}_k}^{-1} \tag{14}$$

$$\hat{w}_k = \hat{w}_k^- + K_k\left(d_k - \hat{d}_k^-\right) \tag{15}$$

$$P_{w_k} = P_{w_k}^- - K_k P_{\tilde{d}_k} K_k^T \tag{16}$$

Where:

$$x^q = \left[x^T \ \nu^T \ n^T\right]^T \ , \tag{17}$$

$$\chi^a = \left[(\chi^x)^T \, (\chi^\nu)^T \, (\chi^n)^T\right]^T \ , \tag{18}$$

$$\tau = \sqrt{l + \lambda}\,. \tag{19}$$

The used symbols denote: λ – scaling parameter, L – dimensionality of extended state vector, R^ν – process noise covariance, R^n – measurement noise covariance.

III. SQUARE ROOT CENTRAL DIFFERENCE KALMAN FILTER

The second nonlinear filter we have used is *Square Root Central Difference Kalman Filter* (SRCDKF) being a derivative-free, like all *Sigma Point Kalman Filters* (SPKF) and less computational variant of *Central Difference Kalman Filter*. The term "Square Root" denotes a more efficient method of matrix inversion comparing to CDKF. SRCDKF steps and equations, in version for parameter estimation, are [12]:

Step I - Initialization:

$$\hat{w}_0 = E\left[w\right] \tag{20}$$

$$S_{w_0} = \mathrm{chol}\left\{E\left[(w_0 - \hat{w}_0)\,(w_0 - \hat{w}_0)^T\right]\right\} \tag{21}$$

For $k \in \{1, \ldots, \infty\}$

Step II - Time–update:

$$S_{\overline{w}_k} = \sqrt{\lambda_{RLS}} S_{w_{k-1}} \text{ or } S_{\overline{w}_k} = S_{w_{k-1}} + D_{r_{k-1}} \tag{22}$$

where:

$$D_{r_{k-1}} = -\operatorname{diag}\left\{S_{w_{k-1}}\right\} + \sqrt{\operatorname{diag}\left\{S_{w_{k-1}}\right\}^2 + \operatorname{diag}\left\{R_{r_{k-1}}\right\}}$$

Step III - Calculate sigma–points for measurement–update:

$$\chi_{k|k-1} = \left[\hat{w}_k^- \quad \hat{w}_k^- + hS_{w_k}^- \quad \hat{w}_k^- - hS_{w_k}^-\right] \tag{23}$$

Step IV - Measurement–update:

$$\Upsilon_{k|k-1} = g\left(w_k, \chi_{k|k-1}\right) \tag{24}$$

$$\hat{d}_k^- = \sum_{i=0}^{2L} w_i^{(m)} \Upsilon_{i,k|k-1} \tag{25}$$

$$S_{\tilde{d}_k} = \mathrm{qr}\Bigg\{\Big[\sqrt{w_1^{(c_1)}}\left(\Upsilon_{1:L,k|k-1} - \Upsilon_{L+1:2L,k|k-1}\right) \tag{26}$$

$$\sqrt{w_1^{(c_2)}}\left(\Upsilon_{1:L,k|k-1} + \Upsilon_{L+1:2L,k|k-1} - 2\Upsilon_{0,k|k-1}\right)\sqrt{S_n}\Big]\Bigg\}$$

$$P_{w_k d_k} = \sqrt{w_1^{(c_1)}} S_{w_k^-}\left[\Upsilon_{1:L,k|k-1} - \Upsilon_{L+1:2L,k|k-1}\right]^T \tag{27}$$

$$K_k = \left(P_{w_k d_k} / S_{\tilde{w}_k}^T\right) / S_{\tilde{d}_k} \tag{28}$$

$$\hat{w}_k = \hat{w}_k^- + K_k\left(d_k - \hat{d}_k^-\right) \tag{29}$$

$$U = K_k S_{\tilde{d}_k} \tag{30}$$

$$S_{w_k} = \mathrm{cholupdate}\left\{S_{w_k^-}, U, -1\right\} \tag{31}$$

The used operators denote: $chol$ – is Cholesky method of matrix factorization, $diag$ – is operation of null–ing all elements of square matrix apart of its diagonal, qr – is QR matrix decomposition and $cholupdate$ – is a Cholesky factor updating.

From systems theory perspective, the process of machine learning is a parameters estimation for identified (learned) dynamic system. For feed–forward neural networks system's Equations 1 and 2 may be transformed into a form of [1]:

$$w_{k+1} = w_k + \nu_k \tag{32}$$

$$y_k = h_k\left(w_k, u_k\right) + n_k \tag{33}$$

where: k – step number, w – state variables i.e. neurons weights vector, y – output vector and u – input vector. It is assumed that both ν and n are white noise i.e. $E\left[\nu_i \nu_j^T\right] = \delta_{ij} R_j^\nu$ and $E\left[n_i n_j^T\right] = \delta_{ij} R_j^n$ covariance matrices respectively, while $E\left[\nu_i \omega_j^T\right] = 0$ for all i and j.

The Equation 32 represents optimized neural weights being driven by noise (estimated noise is equivalent to the weights correction in error backpropagation algorithm). The network observation is represented by Equation 33

UKF and SRCDKF were selected as a learning algorithms for neural networks instead of EKF, because numerous experiments prove better performance of SPKF (a general group where UKF and SRCDKF belong to) over standard EKF e.g. [15], [12] or [14]. It is also important that SPKFs are also simpler to apply (comparing to EKF) as they do not require to compute matrix of partial derivatives (the Jacobian matrix).

We compare both nonlinear filters i.e. UKF and SRCDKF with *Error Backpropagation*, which for a loop-free network with an error $E = \sum_k \left(t_k - o_k\right)^2$, where k is a number of output units, calculates the optimal weights w_{ij} according to gradient descent rule [16]:

$$\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}} = 2\sum\left(t_k - o_k\right)\frac{\partial o_k}{\partial w_{ij}}\ , \tag{34}$$

In the classic and the most popular version of *Error Backpropagation – Gradient Descent* algorithm, the weights corrections are multiplied by a constant value (*learning ratio*). It can lead to increase of error for noise data in case of large values of learning ratio, while low values of this parameter increase the number epochs. The important capability of Kalman filters is an auto–adjustment of gain ratio according to the estimated error. As the result, weights correction are finely tuned to samples i.e. noisy samples do not cause weights oscillations like for *EB* algorithm.

IV. RESULTS OF EXPERIMENTS

To prove high capabilities of SPKFs i.e. UKF and SRCDKF, we have performed experiments over a financial dataset described at [17] (dataset is available at [18]) , that was used test classification accuracy for neural networks, which where trained by different learning algorithms – UKF, SRCDKF and EB. The used dataset contained samples from 128 bankrupt and 112 non-bankrupt companies, being described by 30 real-valued attributes and 1 class variable (*failed*/*non-failed*). All 30 input attributes were financial ratios and as it can be noticed on Figure 1 samples from 2 groups were not clearly separated (the colour of dots denotes their class assignments). Therefore, class attribute was noisy. However, the noise wasn't artificially added to the dataset, but was a result of unclear bankruptcy procedures used by courts (giving different values of class attribute for similar companies). We expected that *Error Backpropagation* algorithm will be constantly oscillating due to opposite weight corrections for misrecognized samples.

As many from the input attributes were redundant, we selected 5 attributes. These the most important attributes were presented in Table I. A subset of them, containing from 2 to 5 attributes to was used to build neural networks in different sizes. The number of neurons in hidden layer was set to 5, 10 or 15 neurons and activation function was *tansign*. For each network variant (no. of inputs and no. of hidden neurons) we tested classification accuracy using UKF (Table II), SRCDKF (Table III) and Gradient Descent Error Backpropagation (Table IV).

The Figure 2 presets how classification error was decreasing when the numbers of epochs was increasing. For this part of experiments we have tested the progress of learning using different values of learning ratio i.e. 0.1, 0.05 and 0.02.

The Error Backpropagation *epochs* are equivalent for Kalman learners *iterations*. For Kalman learners – UKF and SRCDKF – it was necessary to examine influence of iterations number on classification error. The results of these experiments are presented on Figure 3 (for UKF filter) and Figure 4 (for SRCDKF filter). As it can be noticed on Figure 3 increasing number of iterations caused reduction of the classification error.

The similar observation was possible for SRCDKF trained networks (Figure 4) and for both variants of networks it was

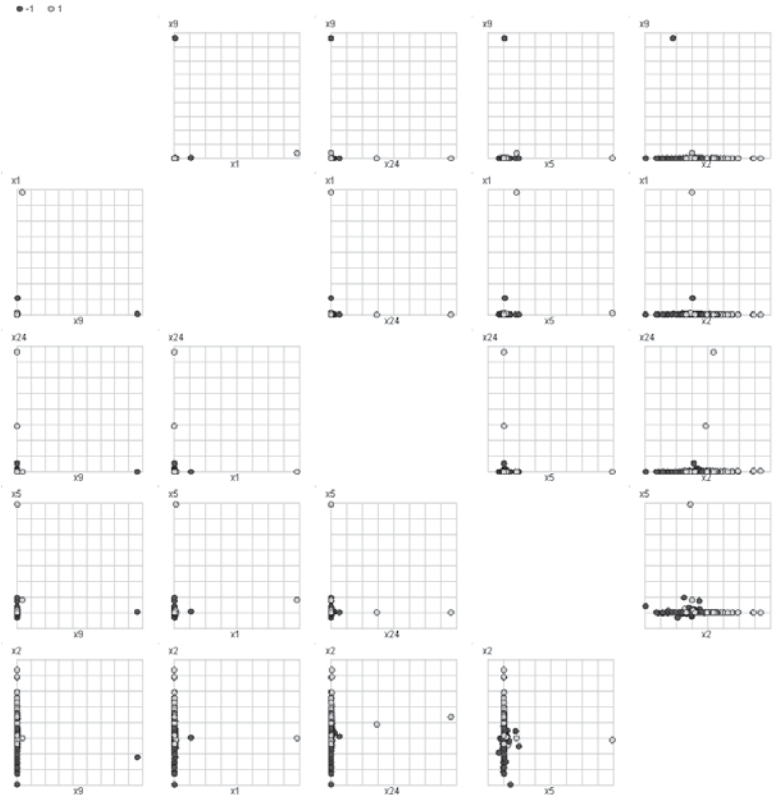

Fig. 1. Samples distribution for selected 5 attributes

TABLE I
THE MOST IMPORTANT ATTRIBUTES (DECREASING ORDER OF SIGNIFICANCE)

Attribute	Ratio		
		Selected	***attributes***
X_9	net profit	$\div$	total assets
X_1	cash	$\div$	current liabilities
X_{24}	net profit	$\div$	sales
X_5	working capital	$\div$	total assets
X_2	cash	$\div$	total assets
		Rejected	***attributes***
X_{11}	net profit	$\div$	receivables
X_{15}	net profit	$\div$	(equity + long term liabilities)
X_6	working capital	$\div$	sales
...	...	$\div$	...

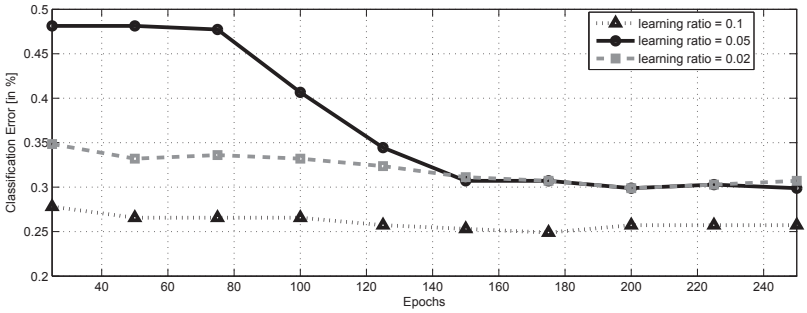

Fig. 2. Classification error vs number of epochs for different values of learning ratio (Gradient Descent Error Backpropagation)

TABLE II
THE ERRORS FOR CLASSIFICATION DONE BY UKF TAUGHT NETWORKS

	Unscented Kalman Filter Number of neurons		
Attributes	5	10	15
2	29.05%	30.29%	30.71%
3	23.65%	25.31%	29.05%
4	23.65%	24.48%	25.31%
5	25.31%	24.48%	28.63%

TABLE III
THE ERRORS FOR CLASSIFICATION DONE BY SRCDKF TAUGHT NETWORKS

	Sigma Points Kalman Filter Number of neurons		
Attributes	5	10	15
2	29.88%	28.22%	32.78%
3	27.80%	26.56%	30.29%
4	26.14%	25.73%	30.29%
5	24.07%	29.88%	26.14%

TABLE IV
THE ERRORS FOR CLASSIFICATION DONE BY EB TAUGHT NETWORKS

	Backpropagation Number of neurons		
Attributes	5	10	15
2	28.22%	29.88%	28.63%
3	27.39%	30.29%	30.29%
4	29.46%	28.63%	29.05%
5	29.04%	29.04%	29.86%

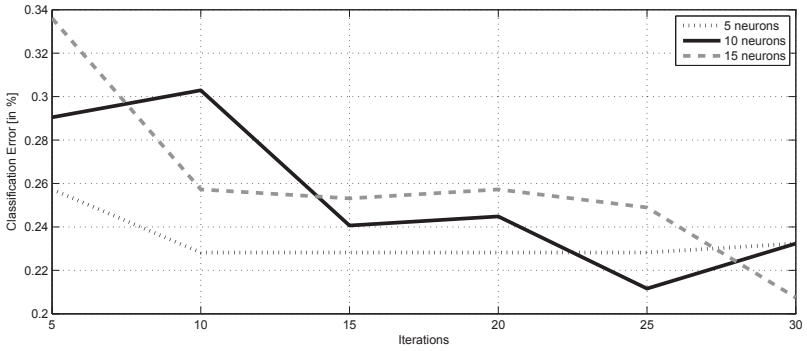

Fig. 3. The classification error vs number of iteration for 5, 10 and 15 neurons in the hidden layer (UKF algorithm)

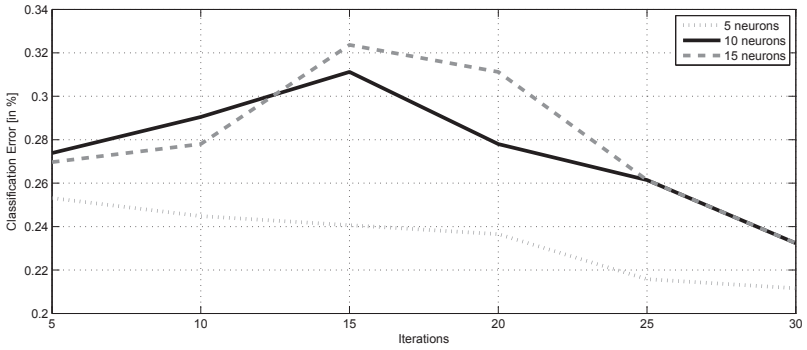

Fig. 4. The classification error vs number of iteration for 5, 10 and 15 neurons in the hidden layer (SPKF algorithm)

observed that large number of neurons in the hidden layer did not reflect their accuracy. This proves that network architecture must be tailored to the problem and the usage of oversized networks only increases numeric calculations.

Summarising, the results of classification were better for both UKF and SRCDKF than for Error Backpropagation. However, Kalman filters required adjustments of their parameters (like noise matrices or scaling parameters). In our opinion it is their disadvantage, while they are generally superior to Error Backpropagation.

V. CONCLUSIONS

The nonlinear Kalman filtering is a family of algorithms too rarely used in Artificial Intelligence. These algorithms, expanded by modern nonlinear versions of filters, may be efficiently used in different tasks. In this paper we have presented an application of nonlinear Kalman filters as learning algorithms for neural networs. The results of performed algorithms prove that Kalman learners offer high accuracy for networks taught in the presence of strong noise. Both tested algorithms i.e. UKF or SRCDKF achieved better results in presented experiments than Error Backpropagation and their accuracy was almost equal. It must be noted that Backpropagation required less adjustments of parameters and was easier to apply. However, in our opinion better accuracy of networks taught by Kalman filters compensated this. Therefore, as demand on better AI methods is increasing Kalman filters offer great potential.

In the future research we plan to examine if a set of suboptimal filtering parameters could be defined, that would make filtering applications easier. We also plan to conduct a research over the usage of nonlinear Kalman filtering for other types of neural networks.

REFERENCES

[1] G. V. Puskorius and L. A. Feldkamp, *Parameter-Based Kalman Filter Training - Theory and Implementation in: Kalman Filtering and Neural Networks, ed. S. Haykin*. John Wiley & Sons, 2001.

[2] P. J. Bolland and J. T. Connor, "A constrained neural network kalman filter for price estimation in high frequency financial data," *International Journal of Neural Systems*, vol. Vol. 8, No. 4, 1997.

[3] D. J. Lary and H. Y. Mussa, "Using and extended kalman filter learning algorthms for feed-forward neural networks to describe tracer correlations," *Atmos. Chem Phy. Discuss.*, vol. Vol. 4, 2004.

[4] I. Rivals and L. Personnaz, "A recursive algorithm based on the extended kalman filter for the training of feedforward neural models," *In Neurocomputing*, vol. Vol. 20(1-3), 1998.

[5] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," in *The IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000*, 2000.

[6] B. Todorovic, M. Stankovic, and C. Moraga, "On-line learning in recurrent neural networks using nonlinear kalman filters," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology 2003*, 2003.

[7] Z. Ronghui and J. Wan, "Neural network-aided adaptive unscented kalman filter for nonlinear state estimation," vol. Volume 13, Issue 7, 2006.

[8] A. Kallapur, M. Samal, V. Puttige, S. Anavatti, and M. Garratt, "A ukf-nn framework for system identification of small unmanned aerial vehicles," in *11th International IEEE Conference on Intelligent Transportation Systems, 2008. ITSC 2008*, 2008.

[9] R. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME - Journal of Basic Engineering*, March 2000.

[10] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," 1996, technical Report.

[11] J. Uhlmann, S. Julier, and M. Csorba, "Nondivergent simultaneous map building and localization using covariance intersection," in *Proceeding of American Control Conference*, 1995.

[12] R. R. Van Der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, 2004.

[13] R. van der Merwe and E. Wan, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," in *Proceedings of the Workshop on Advances in Machine Learning*, 2003.

[14] E. Wan and R. van der Merwe, *The Unscented Kalman Filter in: Kalman Filtering and Neural Networks, ed. S. Haykin*. John Wiley & Sons, 2001.

[15] A. Sitz, U. Schwarz, J. Kurths, and H. Voss, "Estimation of parameters and unobserved components for nonlinear systems from noisy time series," *Physical Review*, vol. E 66, 2002.

[16] M. Arbib, *The Handbook of Brain Theory and Neural Netowrks*. MIT Press, 2003.

[17] W. Pietruszkiewicz, "Dynamical systems and nonlinear kalman filtering applied in classification," in *Proceedings of 2008 7th IEEE International Conference on Cybernetic Intelligent Systems*. IEEE, 2008, pp. 263–268.

[18] "Bankruptcy dataset," available at http://www.pietruszkiewicz.com.