Soft Comput (2018) 22:2695–2703
https://doi.org/10.1007/s00500-017-2525-7

METHODOLOGIES AND APPLICATION

Detecting and quantifying ambiguity: a neural network approach

Rui Ligeiro[1] · **R. Vilela Mendes**[2]

Published online: 1 March 2017

Abstract In general, it is not possible to have access to all variables that determine the behavior of a system. Once a number of measurable variables is identified, there might still exist hidden variables which influence the behavior of the system. The result is model ambiguity in the sense that, for the same (or very similar) input values, distinct outputs are obtained. In addition, the degree of ambiguity may vary across the range of input values. Therefore, to evaluate the accuracy of a model it is important to devise a method to obtain the degree of reliability for each output result. In this paper, we present such a scheme composed of two coupled neural networks, the first one computing the average predicted value and the other the reliability of the output, which is learned from the error values of the first one. As an illustration, the scheme is applied to a model for tracking slopes in a straw chamber and to a credit scoring model.

Keywords Uncertainty · Ambiguity · Neural networks

1 Introduction

When dealing with real-world problems, some degree of uncertainty can rarely be avoided. Modeling physical or social systems, either for further understanding or as a guide for decision processes, dealing with uncertainty is a critical issue. Uncertainty has been formalized in different ways leading to several uncertainty theories (Klir and Smith 2001; Klir 2006; Wong 1993; Vigo 2013; IEEE 1993; Zhang et al. 2014). Ambiguity, that is, uncertainty with unknown probabilities, is also a subject of concern on decision problems (Inukai and Takahashi 2009; Christensen 2013), information retrieval under queries (Roul and Sahay 2012; Clarke et al. 2009), parameter identification (Reppa et al. 2014) and dynamical systems reconstruction and control (Yan and Wang 2014; Hao and Jagannathan 2013; Alfaro-Ponce et al. 2014). Here, we are concerned with uncertainty in the construction of models from observed data. In this context, uncertainty may arise either from inaccuracies in the measurement of the observed variables or from the fact that the variables that are measured do not provide a complete specification of the behavior of the system. Our main concern in this paper is the latter situation, that is, the case of ambiguous data.

There are several models of distributed learning systems that can be used to reconstruct functional relationships between variables. It has been shown that they all are basically equivalent (Doyne 1990), and among them, neural networks, with or without adjustable node parameters (Dente and Vilela 1996), are capable to learn deterministic relations or extract the characteristic parameters of stochastic processes (Dente and Vilela 1997). Their computational power is also, at least, as wide as a large class of symbolic languages (Martins and Mendes 2001). Here, to implement our ambiguity detection algorithm, feedforward networks are used, but of course, other classifier modules might be used as well. The claimed novelty of our approach is not the neural networks architecture, but their use in a global algorithm, which in particular identifies and quantifies the degree of ambiguity in each region of the input variable space.

In the context of construction of models of physical phenomena by neural networks, the distinct problem of learning

Communicated by V. Loia.

✉ R. Vilela Mendes
rvilela.mendes@gmail.com; rvmendes@fc.ul.pt

[1] INOV INESC – Instituto de Novas Tecnologias, Rua Alves Redol 9, 1000-029 Lisbon, Portugal

[2] CMAF - Faculdade de Ciências, Univ. Lisboa, Lisbon, Portugal

from data with error bars has been addressed before by several authors (see for example Gernoth and Clark 1995; Gabrys and Bargiela 1999; Cawley et al. 2007; Huang et al. 2012; Alippi et al. 1995). Here, however, we will be concerned not with inaccuracies in the input data but with the fact that the observed variables might not completely specify the output, that is, some essential variables may not be accessible to be used as inputs. Of special interest is the characterization of the ambiguity across the subregions of the input space, to know in which regions the output is or is not reliable. Being mostly concerned with ambiguous data, rather than with smoothing out noisy data, that is, with "incomplete" rather than "inaccurate" measurements, the system that is developed might as well be used to estimate the degree of noisiness in the data.

The ambiguity situation is a rather complex one because, in general, the uncertainty is not uniform throughout the input variable space. There might be regions where the input variables provide an unambiguous answer and others where they are not sufficient to provide a precise answer. For example, in credit scoring, which we will use here as an example, the "no income, no job, no asset" situation is a clear sign of no credit reliability, but most other situations are not so clear-cut. Therefore, it is desirable to develop a method that, for each region of the input variable space, provides the most probable outcome and, at the same time, tells us how reliable the result is.

The system consists of two coupled networks, one to learn the most probable output value for each input and the other to provide the expected error (or variance) of the result for that particular input. The first network converges to an average of the target values in each region of the input space and the second to the expected uncertainty (or ambiguity) in that particular region of input space. One finds in practice, and in our examples, that the ambiguity varies greatly from region to region of input space. Therefore, it would not make sense to just compute the sample variance of the whole data. The second network does indeed compute a sample variance, but does so for each region of the input variable space and, because of the interpolating features of the network, does it in a more accurate way than if, for example, we were to divide arbitrarily the input space into subregions to do a numerical computation. In short, the main idea of our system is not to smooth out fluctuations in the data to obtain an approximate output. Instead, it is to characterize the ambiguity of the answer and in particular to quantify this ambiguity for each region of input space.

The ambiguity problem in model reconstruction has been addressed in the past by other authors. For example, in the context of fuzzy models, ambiguity is dealt with by increasing the number of fuzzy sets or changing the membership function from bell-shaped to trapezoid-shaped surfaces (Cox 2005). The lattice basis reduction used by some authors (Svendsen 2003) to assign an approximate output to ambiguous inputs is similar to our average value output of the first network, but lacks the quantitative estimate of ambiguity that is provided by the second network. In other approaches, a collection of classifiers and learning algorithms is used to arbitrate between the results by using the most reliable classifier for each subdomain (Ortega et al. 2001; dos Santos et al. 2007). This, of course, is useful if there is a domain-specific adequacy of the classifiers but does not help if the ambiguity is intrinsic to the data. In other cases, the most ambiguous data subsets are gathered into clusters (Lin et al. 2006) and the classifiers retrained (Albalate et al. 2010) or the input data resampled (Bailey-Kellogg and Ramakrishnan 2001) in these domains. Again, this helps only if some new input variables are used in the ambiguity regions, which is not always possible. For example, in a credit scoring modeling problem if all known socio-economic parameters are fed into the system, what else can we use? And in this particular case, at least, a great deal of ambiguity is known to exist. In our system, we simply aim at detecting the ambiguity and quantifying it in each region of the input space.

For definiteness, the system is formalized as the problem of learning random functions in the next section. Then, we study two application examples, the first being the measurement of track angles by straw chambers in high-energy physics and the other a credit scoring model.

2 Learning the average and variance of random functions

The general setting which is analyzed is the following.

The signal to be learned is a random function $\theta(\vec{X})$ with distribution $F_{\vec{X}}(\theta)$. For simplicity we consider θ to be a scalar and the set $\{\vec{X}\}$ to be vector-valued, $\vec{X} \in \mathbb{R}^i$. Notice that we allow for different distribution functions at different points.

In the straw chamber example, to be discussed later, $\vec{X}$ would be the set of delay times and θ the track angle. For the credit score example, $\vec{X}$ would be the set of client parameters and θ the credit reliability.

In our learning system, the $\vec{X}$ values are inputs to a multilayer (feedforward) network, $\{W\}$ denoting the full set of connection strengths, the output being $Y(\vec{X}) = f_W(\vec{X})$. The aim is to chose a set of connection strengths $\{W\}$ that annihilates the expectation value

$$\mathbb{E}\left\{\sum_{\{\vec{X}\}}\left(f_W\left(\vec{X}\right)-\theta\left(\vec{X}\right)\right)^2\right\}=0 \tag{1}$$

However, what, for example, the backpropagation algorithm does is to minimize $\mathbb{E}(f_W(\vec{X}) - \theta(\vec{X}))^2$ for each realization of the random variable $\vec{X}$. Hence, let us fix $\vec{X}$ and consider $f_W(\vec{X})$ evolving in learning time. That is, we are considering, in the learning process, the subprocess corresponding to the sampling of a particular fixed region of the variables. Then, the time evolution of the network output is framed as

$$f_W\left(\vec{X}, \tau + 1\right) = f_W\left(\vec{X}, \tau\right) - 2\eta \frac{\partial f_W}{\partial W} \cdot \left(f_W\left(\vec{X}\right) - \theta\left(\vec{X}\right)\right) \frac{\partial f_W}{\partial W} \tag{2}$$

where $\Delta W = -\eta \frac{\partial e}{\partial W}$, η being the learning rate and $e = (f_W(\vec{X}) - \theta(\vec{X}))^2$ the error function. Let the input random variable $\theta(\vec{X})$ at learning step τ be modeled as

$$\theta\left(\vec{X}\right)_\tau = \overline{\theta}\left(\vec{X}\right) + B\left(\vec{X}, \tau\right) \tag{3}$$

$\overline{\theta}(\vec{X})$ being the expectation (average) value of the variable and $B(X, \tau)$ a zero-mean Wiener process. Then, taking expectation values in Eq. (2), and because the $\frac{\partial f_W}{\partial W}$ quantities are deterministic functions, one obtains

$$\mathbb{E}\left[f_W\left(\vec{X}, \tau + 1\right)\right] = \mathbb{E}\left[f_W\left(\vec{X}, \tau\right)\right] - 2\eta \frac{\partial f_W}{\partial W} \cdot \left(\mathbb{E}\left[f_W\left(\vec{X}, \tau\right)\right] - \overline{\theta}\left(\vec{x}\right)\right) \frac{\partial f_W}{\partial W} \tag{4}$$

Notice that we are now dealing with two time scales: the time scale of the $\theta(\vec{X})$ random variable and the time scale of the leaning process, controlled by the learning rate η. If the learning rate η is sufficiently small for the learning time scale to be much smaller than the sampling rate of the $\theta(\vec{X})$ random variable, the last equality may be approximated by

$$f_W\left(\vec{X}, \tau + 1\right) = f_W\left(\vec{X}, \tau\right) - 2\eta \frac{\partial f}{\partial W} \cdot \left(f_W\left(\vec{X}, \tau\right) - \overline{\theta}\left(\vec{X}\right)\right) \frac{\partial f_W}{\partial W} \tag{5}$$

A fixed point is obtained at

$$f_W\left(\vec{X}\right) = \overline{\theta}\left(\vec{X}\right), \tag{6}$$

the average value of the random variable θ at the argument $\vec{X}$. In practice, the convergence to the average of the objective variable is better achieved by making η converge slowly to zero during the learning process.

Similarly, if a second network [with output $g_{W'}(\vec{X})$] and the same input $\vec{X}$ is constructed according to the learning law

$$g_{W'}\left(\vec{X}, \tau' + 1\right) = g_{W'}\left(\vec{X}, \tau'\right) + \frac{\partial g_{W'}}{\partial W'} \cdot \Delta W' \tag{7}$$

with error function

$$e' = \left(g_{W'}\left(\vec{X}\right) - \left(f_W\left(\vec{X}\right) - \theta\left(\vec{X}\right)\right)^2\right)^2 \tag{8}$$

and $\Delta W' = -\eta' \frac{\partial e'}{\partial W'}$, then

$$g_{W'}\left(\vec{X}, \tau' + 1\right) = g_{W'}\left(\vec{X}, \tau'\right) - 2\eta' \frac{\partial g_{W'}}{\partial W'} \cdot \frac{\partial g_{W'}}{\partial W'} \left(g_{W'}\left(\vec{X}, \tau'\right) - \left(f_W\left(\vec{X}\right) - \theta\left(\vec{X}\right)\right)^2\right)$$

and, under the same assumptions as before concerning the smallness of the learning rates, $g_{W'}(\vec{X})$ has the fixed point

$$g_{W'}\left(\vec{X}\right) = \overline{\left(\theta\left(\vec{X}\right) - \overline{\theta}\left(\vec{X}\right)\right)^2} \tag{9}$$

In conclusion: The first network reproduces the average value of the random function θ for each input $\vec{X}$ and the second one, receiving as data the errors of the first, reproduces the variance of the function at $\vec{X}$. Instead of the variance, the second network might as well be programed to learn the expected value of the absolute error $\mathbb{E}|\theta(\vec{X}) - \overline{\theta}(\vec{X})|$. Actually, for numerical convenience, we will use this alternative in the examples of the next section. Figure 1 is a schematic representation of the learning process.

In practice, the training of the second network should start after the first one because, before the first one becomes to converge, its errors are not representative of the fluctuations of the random function. In general, it seems reasonable to have $\eta'(t) < \eta(t)$ with $\eta(t)$ decreasing in time.

Implicit in the derivation sketched above is the assumption that we are already in the basin of attraction of the global minimum of the cost functions. In practice, the existence of local minima is an issue to be taken into account in all modeling and optimization problems. To avoid convergence to local minima, one may use occasional random perturbations. Our approach however has been to run several times the algorithm starting from different initial conditions for the neural network parameters.

Because in both networks one wants convergence to average values of the target functions, a critical issue is also to avoid overfitting in the design of the networks. Because for

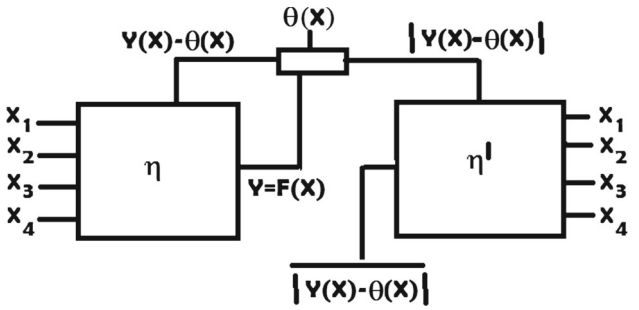

Fig. 1 A schematic representation of the learning process. The two-network system, given inputs X_i and target values θ_k, learns the average values $\overline{Y_k}(X_i)$ and average errors $\overline{|Y_k - \theta_k|}$for each set $\{X_i\}$ of input values

our examples we use networks with one hidden layer, the number of neurons in the hidden layer is the parameter that should be of our concern. In general for a learning machine, the ideal situation is to have a Vapnik–Chervonenkis (VC) dimension (Vapnik and Chervonenkis 1971; Vapnik 2000) equal to the number of independent functions that one wants to discriminate or, in a classification setting, the number of points that one wants to shatter. Methods have been developed to estimate the VC dimension of a learning machine (Vapnik et al. 1994; Bartlett and Maass 2003). In the spirit of the final prediction error criterion (Alippi 1999), we use here a simple approach to estimate the right number of neurons in the hidden layer. In "Appendix", we show the evolution of the mean square error, after training of the networks in the two examples, when the number of neurons in the hidden layer changes. One sees that for the scoring case 14 hidden layer neurons seem to be an appropriate number to obtain a good fit without overfitting and similarly for the straw chamber a number between 10 and 15 is adequate. Several other methods have been proposed in the literature to choose the number of hidden layer neurons. A popular method is the clipping method where during the learning process the synapses with the smallest strengths are suppressed. As we have found out, this method is not very effective when there is a high level of ambiguity in the data. Therefore, the control of the mean square error seems more appropriate in this case.

Finally, for the scoring example we have used 14 hidden layer neurons and for the straw chamber both 14 and 25. The test with 25 is included to reproduce the setting of Denby et al. (1990) who use 25 hidden layer neurons.

3 Examples

3.1 Measuring track angles by straw chambers

One of the first applications of neural networks to the processing of high-energy physics data (Denby 1999) was the work by Denby et al. (1990) on the slopes of particle tracks

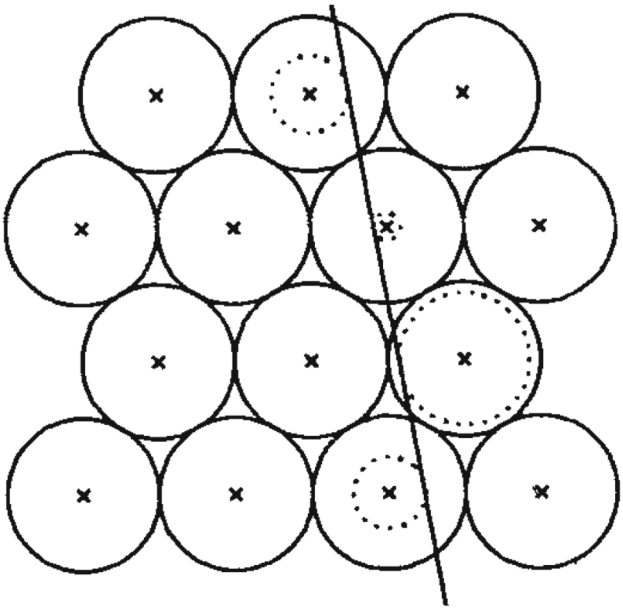

Fig. 2 A particle track through a straw chamber. The input values to the neural networks are the delay times, proportional to the distances of the particle to the wires

in straw tube drift chambers. In a straw chamber (Fig. 2), each wire receives a signal delayed by a time proportional to the distance of closest approach of the particle to the wire.

The neural network receives these times as inputs $\{\vec{X}\}$, with as many inputs as the number of wires and, for the training, the track angle $\theta(\vec{X})$ is the target function. The half cell shift of alternate layers in the straw chamber solves some of the left–right ambiguities, but this ambiguity still remains for many directions (Fig. 3).

The authors of Denby et al. (1990) required the training and test events to pass through at least four straws to avoid edge effects. Nevertheless, they consistently find large non-Gaussian error tails when testing the trained network. The authors have not separated the contribution to the tails coming from the ambiguities from those arising from eventual inadequacies on training or network architecture. We have repeated the simulations, and our results essentially reproduce those of Denby et al. (1990), showing that the non-Gaussian tails do indeed originate from the left–right ambiguities. If edge effects are allowed for, including in the training set events that pass through less than four straws, the degree of ambiguity and the tails increase even further. The important role of persistent fat tails on the error response of a learned system as a symptom of data ambiguity will be discussed later.

This example is therefore a typical example of the situation described in the introduction, where some regions of the input data correspond to a unique event, but others have an ambiguous identification. Also, it is a pure example of ambiguous data in the sense that the signals fed to the networks have no noise component. As the example shows, it

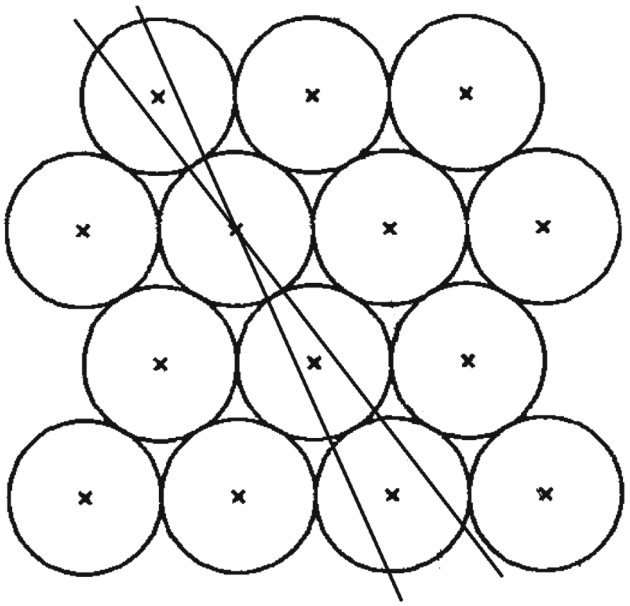

Fig. 3 An example with two different beam track angles generating the same input signal

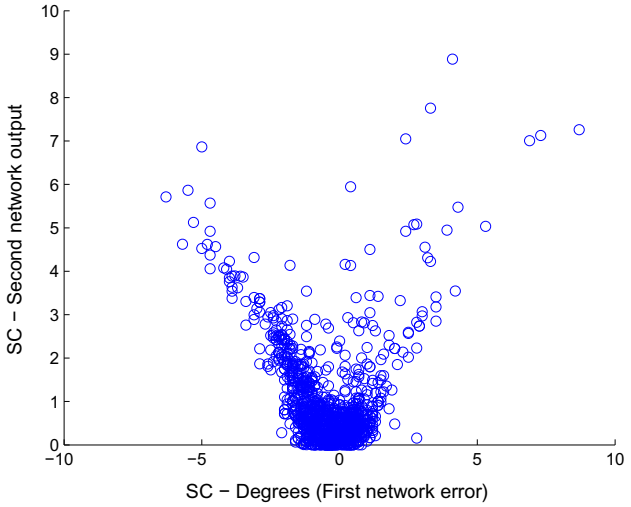

Fig. 4 Comparison of the actual error of the first network and the estimated uncertainty predicted by the second network (straw chamber data)

is not easy to separate the ambiguous regions from the non-ambiguous ones because they are mixed all over parameter space. It is therefore important to have a system that not only provides an answer but also states how reliable that answer is.

We have applied to this example the two-network scheme (Fig. 1) described before. Both networks have the same architecture and train using the same input data, the first one with the target track angles and the second with the absolute value of the errors of the first. To avoid big fluctuations in training convergence, the second network starts learning after the first has stabilized and finished training. Both networks have a feedforward network architecture with three neuron layers: input, hidden and output. They both train using a supervised backpropagation algorithm. The neuron activation function is the logistic sigmoid (tan-sigmoid). After some optimization, our sigmoid-based backpropagation became rather efficient. The use of radial basis functions (RBF) might, in some cases, provide faster learning rates if the RBFs are tuned to particular applications.

For the results presented here, we use 14 input neurons (representing the drift times in each straw), either 14 or 25 hidden neurons and an output neuron for the slope of each track. We use Monte Carlo generated data coded as follows: If the track does not meet the straw the input value is zero and if the track crosses the straw, the input value is the difference between the straw radius and the distance to the wire in the center of the straw. The output is the angle of the track slope. A training sample of 25,000 simulated tracks was generated. After training, the performance of the network was tested using a new set of 5000 independent tracks.

Figure 4 compares the actual error of the first network with the uncertainty predicted by the second. One sees that the largest errors do indeed correspond to large uncertainty prediction by the second network. Of course in a few cases large uncertainty is predicted when the actual error is small. It only means that particular result is unreliable in the sense that it was by chance that it fell in the middle of the error bar interval. The results obtained with either 14 or 25 hidden neurons are practically indistinguishable, meaning that the networks are indeed characterizing the ambiguity of the data.

Now that we are equipped with a system that predicts both an angle and its probable uncertainty, it makes sense to state that the result of a measurement is $\theta \pm \Delta_{\text{ann}}$, θ being the output of the first network and Δ_{ann} the output of the second. In this sense, we will count an output as an error only when the objective value is outside the error bars. The *effective error* will be the distance of the objective value to the boundary of the error bars. Figure 5 plots the effective error for a sample of 1000 tracks.

An important problem when attempting to model experimental data is the detection of ambiguities or equivalently to know whether the data set completely characterizes the phenomenon. As mentioned before, clustering methods have been proposed (Lin et al. 2006; Albalate et al. 2010; Bailey-Kellogg and Ramakrishnan 2001) to isolate the ambiguity regions and identify the origin of the ambiguities. This is not always possible, nor reliable, if ambiguity regions exist spread all over the input space and for which a subset of variable values are shared by non-ambiguous regions. This is the case in the straw chamber example. Therefore, the more conservative way of looking for fat tails in the error distribution and reconstructing an ambiguity predictor as proposed here is, in our opinion, more appropriate. As an illustration,

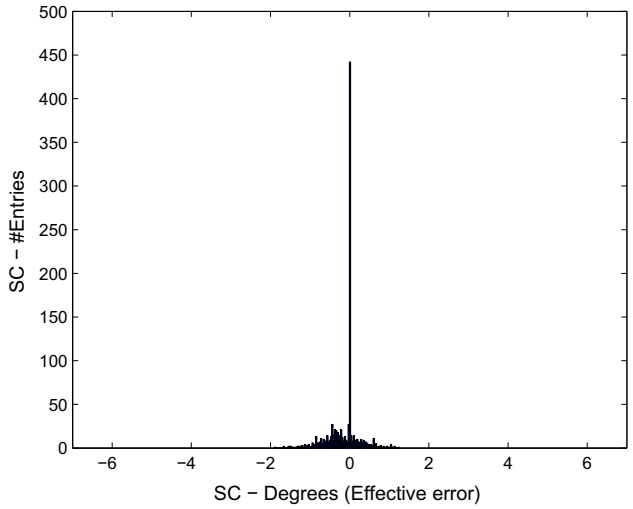

Fig. 5 Effective error (straw chamber data)

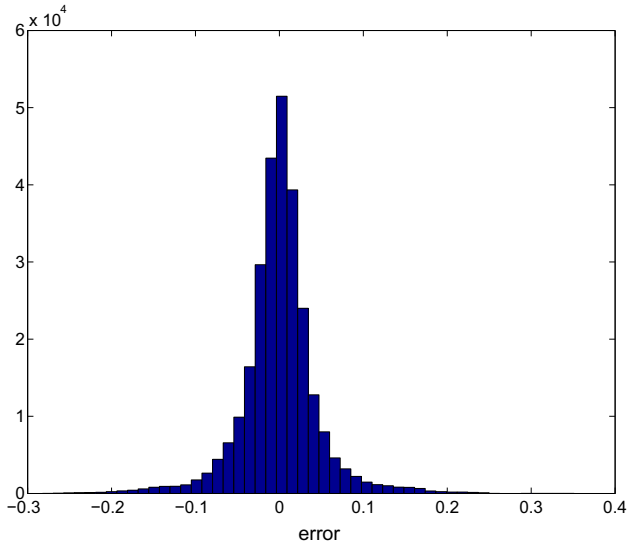

Fig. 6 Fat tails as a symptom of ambiguity

we have studied the error distribution of the first network for successively higher sizes of the training set to find out that indeed fat tails are persistent and their nature quite stable. Figure 6 shows the error distribution for a training set of 273,271 (4-hits) tracks, and as a clear symptom, the excess kurtosis of the plot is 5.3.

3.2 A credit scoring model

Defaulting on loans has recently increased, promoting the search for accurate techniques of credit evaluation by financial institutions. Credit scoring is a quantitative method, based on credit report information that helps lenders in the credit granting decision. The objective is to categorize credit applicants into two separate classes: the "good credit" class, that is, the one likely to repay loans on time and the "bad credit" class to which credit should be denied, due to a high probability of defaulting. For a more detailed understanding of credit scoring models, we refer to Lando (2004), Van Gestel and Baesens (2009) and Thomas et al. (2002).

Here, we have developed a credit scoring model based on the two-network scheme discussed before. Because complete information on the credit applicants is impossible to obtain and human behavior is dependent on so many factors, credit scoring is also a typical example of a situation where one is trying to predict an outcome based on incomplete information. Credit scoring models with neural networks had been proposed in the past (see, for example, West 2000; Pacelli and Azzollini 2011). The novelty of our system lies in that not only we provide a scoring result but we also obtain an estimate of how reliable the result is. A similar system has been successfully developed by the authors for a credit company where scoring ambiguities are of utmost importance for risk evaluation. For privacy restrictions, however, the data we use in our second example are from an open source.

We use here a publicly available credit data of anonymous clients, downloaded from UCI Irvine Machine (http://archive.ics.uci.edu/ml/). It is composed of 1000 cases, one per applicant, of which 700 cases correspond to creditworthy applicants and 300 cases correspond to applicants which were later found to be in the bad credit class. Each instance corresponds to 24 attributes (e.g., loan amount, credit history, employment status, personal information, etc.) with the corresponding credit status of each applicant coded as good (1) or bad (0). Inspecting the database, it is clear that some apparently good attributes correspond, in the end, to bad credit performance and conversely putting into evidence the incomplete information nature of the problem.

For our system, the attributes are numerically coded and we use a neural network architecture with 24 input neurons (representing the 24 numerical attributes), 14 hidden neurons and an output neuron indicating good or bad credit. To ensure that the network learns evenly, we randomly alternate between good and bad applicants instances. After training, the performance of the network was tested. Figure 7 shows a plot of the errors of the first network after training. Although, in general, the network provides good estimations, there are several customers classified as good when they are bad and vice-versa. In fact, there are some extremely incorrect network predictions, as can easily been perceived by the bins at the two ends of the histogram. These bins clearly reveal lack of information in the data set.

As in the previous example, Fig. 8 shows the comparison of the errors in the first network with the estimated uncertainty obtained by the second network and Fig. 9 shows the effective error distribution. Similarly to the previous straw chamber example, one obtains good uncertainty predictions by the second network. The second network wrongly clas-

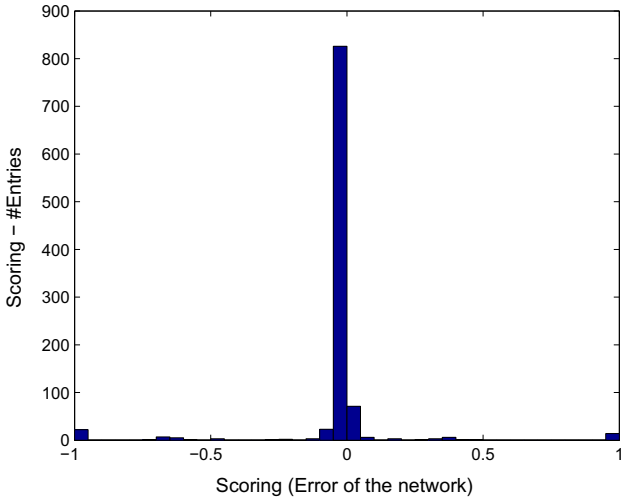

Fig. 7 Error distribution in the first network (credit scoring)

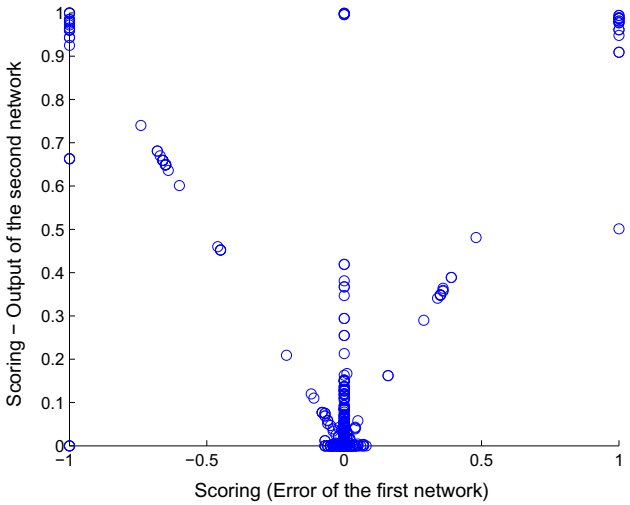

Fig. 8 Comparison of the actual error of the first network and the estimated uncertainty predicted by the second network (credit scoring)

sified very few cases: Only two occurrences with no actual errors were predicted having maximum uncertainty, and only three critical errors were unsuccessfully predicted without uncertainty.

Looking at the effective error distribution plot, it is easy to confirm the refinement in the degree of certainty in each estimation. Nevertheless, there still are a very few occurrences of estimations outside the error bar interval.

3.3 Conclusions

1. The goal of this research was to develop a computational scheme with the ability to evaluate the degree of reliability of predictive models. Two application examples were studied, the first one being the measurement of track angles by straw chambers in high-energy physics and the other a credit scoring model. Both examples use data with incomplete information. A two-network system is used which, although not perfect, greatly improves the reliability check of the predicted results.
2. That the estimate of the reliability of the data modeling is sensitive to situations where uncertainty is not uniform throughout the parameter spaces, is an asset of the system. A weakness is of course the assumption that uncertainty is well modeled by the second momentum of the input data. Skewness, power-law distributions and rare events fall outside the scope of the system. In any case, to model such features with neural networks might not be appropriate and more complex systems involving, for example, estimates of characteristic functions (Dente and Vilela 1997) might have to be brought into play.

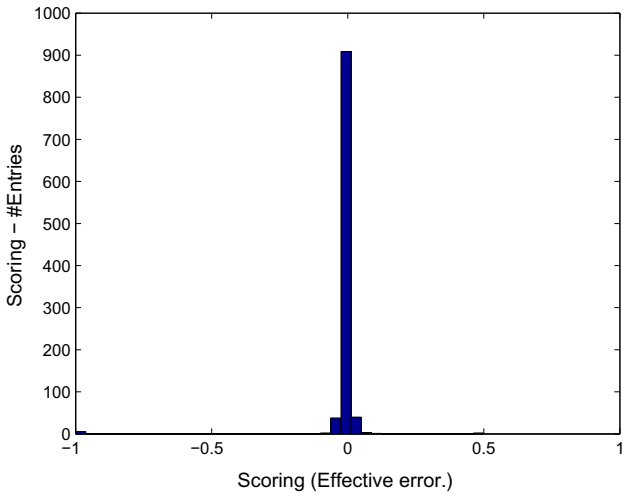

Fig. 9 Effective error (credit scoring)

Acknowledgements This study was funded by Fundação para a Ciência e Tecnologia, Portugal.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

Appendix: Evolution of the mean square error when the number of neurons in the hidden layer changes

In two figures (Figs. 10, 11), we display the mean square error after training, of the networks in the two examples, when the number of neurons in the hidden layer changes

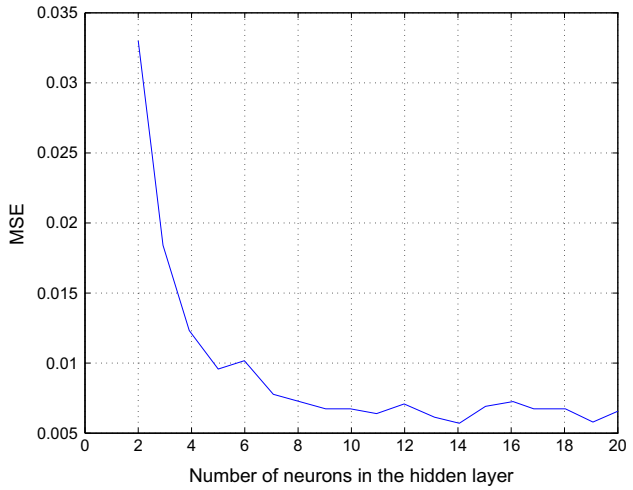

Fig. 10 Mean square error after training when the number of neurons in the hidden layer changes (straw chamber)

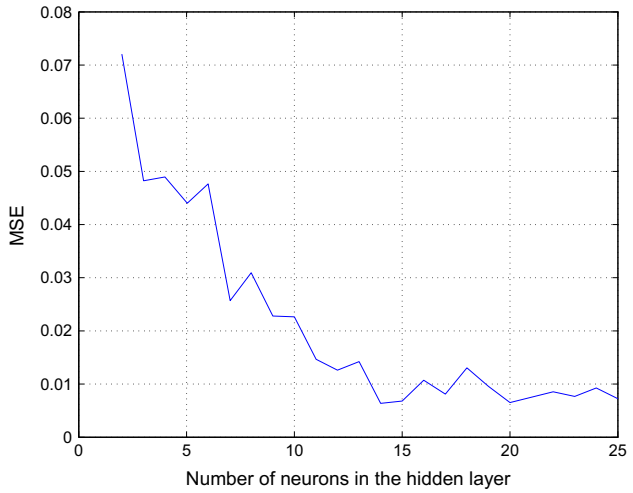

Fig. 11 Mean square error after training when the number of neurons in the hidden layer changes (credit scoring)

References

Albalate A, Suchindranath A, Soenmez MM, Suendermann D (2010) On ambiguity detection and postprocessing schemes using cluster ensembles, ICAART 2010. In: Proceedings of the international conference on agents and artificial intelligence. INSTICC Press, pp 623–630

Alfaro-Ponce M, Cruz AA, Chairez I (2014) Adaptive identifier for uncertain complex nonlinear systems based on continuous neural networks. IEEE Trans Neural Netw Learn Syst 25:483–494

Alippi C (1999) FPE-based criteria to dimension feedforward neural topologies. IEEE Trans Circuits Syst I 46:962–973

Alippi C, Piuri V, Sami M (1995) Sensitivity to errors in artificial neural networks: a behavioural approach. IEEE Trans Circuits Syst I 42:358–361

Bailey-Kellogg C, Ramakrishnan N (2001) Ambiguity-directed sampling for qualitative analysis of sparse data from spatially-distributed physical systems. In: Proceedings of the IJCAI

Bartlett PL, Maass W (2003) Vapnik–Chervonenkis dimension of neural nets. In: Arbib MA (ed) The handbook of brain theory and neural networks. MIT Press, Cambridge, pp 1188–1192

Cawley GC, Janacek GJ, Haylock MR, Dorling SR (2007) Predictive uncertainty in environmental modelling. Neural Netw 20:537–549

Christensen S (2013) Optimal decision under ambiguity for diffusion processes. Math Meth Oper Res 77:207–226

Clarke CLA, Kolla M, Vechtomova O (2009) An effectiveness measure for ambiguous and underspecified queries. In: Advances in information retrieval theory, lecture notes in computer science, vol 5766, pp 188–199

Cox E (2005) Fuzzy modeling and genetic algorithms for data mining and exploration. Elsevier, Amsterdam

dos Santos E, Sabourin R, Maupin P (2007) Ambiguity-guided dynamic selection of ensemble of classifiers. In: 10th international conference on information fusion

Denby B (1999) Neural networks in high energy physics: a ten year perspective. Comput Phys Commun 119:219–231

Denby B, Lessner E, Lindsey CS (1990) Test of track segment and vertex finding with neural networks. In: Proceedings of the 1990 conference on computing in high energy physics, Sante Fe, NM. AIP conference proceedings, vol 209, p 211

Dente JA, Vilela Mendes R (1996) Unsupervised learning in general connectionist systems. Netw Comput Neural Syst 7:123–139

Dente JA, Vilela Mendes R (1997) Characteristic functions and process identification from neural networks. Neural Netw 10:1465–1471

Doyne J (1990) Farmer. A Rosetta stone for connectionism. Phys D 42:153–187

Gabrys B, Bargiela A (1999) Neural network based decision support in presence of uncertainties. ASCE J Water Resour Plan Manag 125:272–280

Gernoth KA, Clark JW (1995) A modified backpropagation algorithm for training neural networks on data with error bars. Comput Phys Commun 88:1–22

Hao Xu, Jagannathan S (2013) Stochastic optimal controller design for uncertain nonlinear networked control system via neuro dynamic programming. IEEE Trans Neural Netw Learn Syst 24:471–484

Huang G, Song S, Wu C, You K (2012) Robust support vector regression for uncertain input and output data. IEEE Trans Neural Netw Learn Syst 23:1690–1700

Inukai K, Takahashi T (2009) Decision under ambiguity: effects of sign and magnitude. Int J Neurosci 119:1170–1178

Klir GJ (2006) Uncertainty and information: foundations of generalized information theory. Wiley, Hoboken

Klir GJ, Smith RM (2001) On measuring uncertainty and uncertainty-based information: recent developments. Ann Math Artif Intell 32:5–33

Lando D (2004) Credit risk modeling. Princeton University Press, Princeton

Lin Y-M, Wang X, Ng WWY, Chang Q, Yeung DS, Wang X-L (2006) Sphere classification for ambiguous data. In: 2006 International conference on machine learning and cybernetics, IEEE conference publications, pp 2571–2574

Martins J, Vilela Mendes R (2001) Neural networks and logical reasoning systems. A translation table. Int J Neural Syst 11:179–186

Ortega J, Koppel M, Argamon S (2001) Arbitrating among competing classifiers using learned referees. Knowl Inf Syst 3:470–490

Pacelli V, Azzollini M (2011) An artificial neural network approach for credit risk management. J Intell Learn Syst Appl 3:103–112

Reppa V, Polycarpou MM, Panayiotou CG (2014) Adaptive approximation for multiple sensor fault detection and isolation of nonlinear uncertain systems. IEEE Trans Neural Netw Learn Syst 25:137–153

Roul RK, Sahay SK (2012) An effective information retrieval for ambiguous query. Asian J Comput Sci Inf Technol 2:26–30

Svendsen JGG (2003) A SearchFree approach to ambiguity resolution. In: Proceedings of the 16th international technical meeting of the satellite Division of the Institute of Navigation. Portland, pp 769–774

Thomas LC, Edelman DB, Crook JN (2002) Credit scoring and its applications. SIAM, Philadelphia

UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/

Van Gestel T, Baesens B (2009) Credit risk management: basic concepts: financial risk components. Rating analysis, models, economic and regulatory capital. Oxford University Press, Oxford

Vapnik V (2000) The nature of statistical learning theory. Springer, New York

Vapnik V, Chervonenkis A (1971) On the uniform convergence of relative frequencies of events to their probabilities. Theory Probab Appl 16:264–280

Vapnik V, Levin E, Le Cun Y (1994) Measuring the VC-dimension of a learning machine. Neural Comput 6:851–876

Vigo R (2013) Complexity over uncertainty in generalized representational information theory: a structure-sensitive general theory of information. Information 4:1–30

West D (2000) Neural network credit scoring models. Comput Oper Res 27:1131–1152

Wong SKM, Wang ZW (1993) Qualitative measures of ambiguity. In: Proceedings of the 9th conference on uncertainty in artificial intelligence, pp 443–450

IEEE (1993) Proceedings of second international symposium on uncertainty modeling and analysis. In: IEEE conference publications. doi:10.1109/ISUMA.1993.366801

Yan Z, Wang J (2014) Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. IEEE Trans Neural Netw Learn Syst 25:457–469

Zhang Q, Dong C, Cui Y, Yang Z (2014) Dynamic uncertain causality graph for knowledge representation and probabilistic reasoning: statistics base, matrix and application. IEEE Trans Neural Netw Learn Syst 25:645–663