Ecological Informatics 38 (2017) 26–30

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

Combining expert knowledge with machine learning on the basis of fuzzy training

Ralf Wieland*, Wilfried Mirschel

Leibniz Centre for Agricultural Landscape Research, Institute of Landscape Systems Analysis, Eberswalder Str. 84, Muencheberg 15374, Germany

ARTICLE INFO

Article history:
Received 8 September 2016
Received in revised form 11 January 2017
Accepted 12 January 2017
Available online 16 January 2017

Keywords:
Fuzzy modeling
Expert knowledge
Machine learning
Nonlinear regularization
Optimization
Yield modeling

ABSTRACT

The paper introduces a fuzzy training approach based on nonlinear regularization in an effort to avoid over training. The main idea is to restrict training so that the basic expert knowledge used to build the model is still visible. This is implemented by a new nonlinear regularization approach which can be applied to any kind of training data set. The approach is demonstrated using a large crop yield data set (>4500 field records) for sugar beet collected in agricultural farms over a 14-year period (1976–1989) in East Germany. The software is implemented in SAMT2, free and open source software, using the Python programming language.

1. Introduction

In the 1980s and 1990s, scientific focus was directed towards expert systems. The idea was to transfer expert knowledge to a computer and use it in form of an expert system (Buchanan and Shortliffe, 1984). Expert knowledge was used as a consistent logical base to build expert systems. However, the logical base was unable to handle uncertain knowledge. An important step came in the basic paper of fuzzy theory (Zadeh, 1965), which introduced a formal system of uncertain knowledge as shown in Liu and Chen (1995).

A short time later, after the expert systems boom was established, the fuzzy modeling technique was also introduced in environmental science as shown in Bock and Salski (1998). Today, the focus has shifted from the use of fuzzy models as standalone models to their use as part of a complex modeling approach (Zarandi and Ahmadpour, 2009; Lee and Pan, 2004). A second modern direction of fuzzy modeling was the combination of fuzzy models with training algorithms as shown in Alves et al. (2011) and Mouton et al. (2011). The training of fuzzy models can also include the generation of fuzzy rules from a data set (Wieland et al., 2011). One adaptation of membership function using a genetic algorithm is described by Liu et al. (2013). Here the membership functions were adapted to the outputs only (Wieland and Mirschel, 2008).

As result of the developments in the 1990s, the idea was to use expert knowledge in environmental modeling. Fuzzy models can be understood easily (they implement a consistent logical base), can be explained to stakeholders etc. On this basis, fuzzy models are much more trustworthy for stakeholders than black-box models (Sami et al., 2014). On the other hand, training of fuzzy models can also make them more accurate. The questions arise: how much training is tolerable that the fuzzy model still represents expert knowledge? What about “over training” (when will fuzzy models lose their generality and become logically inconsistent)?

Over training became obvious with an artificial neural network (Kavzoglu, 2009), but over training is still possible within most other machine learning approaches, such as support vector machines (Pouteau et al., 2012) or random forest modeling (Nam et al., 2015). Lima et al. (2015) provide a brief overview.

This paper introduces a new training approach for fuzzy models, based on regularization which helps avoid over training. The main idea is to restrict training so that the expert knowledge is still visible. This new approach is implemented using the free and open source software SAMT2 (Wieland et al., 2015).

2. Method

2.1. A description of fuzzy modeling

A fuzzy model consists of a set of inputs ($x_i \in X$), a set of membership functions for each input ($\mu_{ij}(x_i) \in M$), a set of outputs ($o_l \in O$)

* Corresponding author
E-mail address: rwieland@zalf.de (R. Wieland).

http://dx.doi.org/10.1016/j.ecoinf.2017.01.001
1574-9541/© 2017 Elsevier B.V. All rights reserved.

and a set of rules ($M \times O$). In the following, the membership functions for the inputs have trapezoidal or triangular shapes. The outputs are fixed values, so called " singletons".

The fuzzy algorithm used here was first published in Wieland and Mirschel (2008) and will be briefly explained. The algorithm operates with a Takagi-Sugeno-Kang (TSK)-type Fuzzy Rule-Based System (FRBS) of zero order. In the example below it's shown a FRBS with n rules, three inputs and one output:

$$\begin{aligned}
&\text{IF}\mu_{11}(x_1) \wedge \mu_{21}(x_2) \wedge \mu_{31}(x_3) \Rightarrow o_1\\
&\text{IF } \mu_{12}(x_1) \wedge \mu_{22}(x_2) \wedge \mu_{32}(x_3) \Rightarrow o_2\\
&\qquad\qquad\qquad\qquad \cdots\ \cdots\\
&\text{IF } \mu_{1n}(x_1) \wedge \mu_{2n}(x_2) \wedge \mu_{3n}(x_3) \Rightarrow o_n
\end{aligned}$$

where $\mu_{ij}(x_i)$ is a membership function over the input x_i (i counts the inputs, j counts the rules), and the output $o_l \in O$. It should be remarked that different rules can have the same output (o_l). The antecedent combines different inputs using "and". To express "or" combinations, a set of rules with different inputs but the same output can be used. The input carries information about the uncertainty, whereas outputs are fixed. This has the advantage of high operation speed due to simplified defuzzification. More importantly, crisp outputs can model linear and nonlinear functional behavior well.

2.2. The fuzzy algorithm

The fuzzy algorithm consists of the following three steps:

- Step1 (fuzzification): for every input $\vec{x} = (x_1, x_2, x_3)$ all membership functions $\mu_{ij}(x_i)$ will be calculated
- Step2 (inference):
  - implication: assigns a value a_l^m using the minimum or product operator according to Gupta and Qi (1991) to each rule (l points to the outputs; m points to different rules with the same output l): $a_l^m = \mu_{1j}(x_1) \times \mu_{2j}(x_2) \times \mu_{3j}(x_3)$
  - aggregation: selects the best rule from the m rules with same output l using maximum operator: $\hat{a}_l = \max\{a_l^1, a_l^2, \cdots\}$
- Step3 (defuzzification): $y = \frac{\sum_l \hat{a}_l \times o_l}{\sum_l \hat{a}_l}$.

The defuzzification is implemented as a simple weighted sum with the general output y. This has the advantage that the defuzzification is very fast, which is important for GIS based applications and the defuzzification introduces no additional nonlinearity in the fuzzy system which is often the case using more elaborate defuzzification methods like "center of gravity" (Lutz and Wendt, 1998).

When singletons have been implemented by the modeler they are often not adapted during the parametrization. This was a reason to adapt the outputs by training, thus enhancing the accuracy of fuzzy models as shown in Wieland and Mirschel (2008). This implementation of training worked well for some data sets, but it had the disadvantage of failing to exclude over training. Here, over training means that some outputs began to approach each other ($\exists l, o_l \approx o_{l+1} \text{or} o_{l-1} \approx o_l$); this may minimize the root mean square error (RMSE), but the trained model could not be applied to any other data set. It should be remembered that the fuzzy models used here are based on separated singletons as outputs. In case of over training the relation to the original model will be destroyed.

2.3. Regularization

The adaptation of the outputs should be restricted to a certain degree. Fig. 1 illustrates the idea of the so-called "spring concept".

Output o_l can be moved towards output o_{l+1} up to the range where the springs will reduce the movement depending on the distance to the next o_{l+1}, i.e. the lower the distance to o_{l+1} the higher the movement reduction. This idea is called regularization and is described in Bishop (2006) in detail.

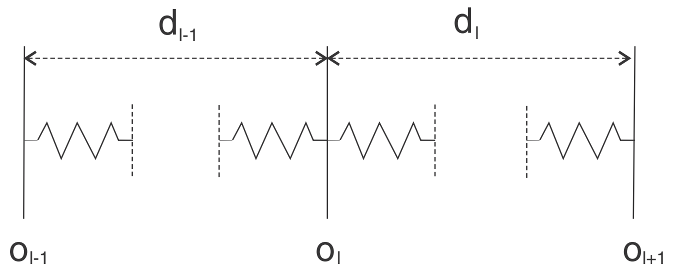

Fig. 1. Output regularization.

In general, outputs are different for different fuzzy models; therefore, a general solution for regularization has to be introduced. For example, the output of a habitat model (Bock and Salski, 1998) which estimates the habitat quality using fuzzy values $o_l \in [0, 1]$, is in the range of $o_1 = 0 \cdots o_n = 1$, and for a crop yield model for sugar beet it is in the range of $o_1 = 30.0 \cdots o_n = 800.0$ [dt/ha]. This implies different ranges for the RMSE$=\sqrt{\frac{\sum_i^n (y_i - ym_i)^2}{n}}$ (with y_i as predicted value and ym_i as measured value). The RMSE before training ($RMSE_0$) is used to normalize the $RMSE_t$ for each step $tR\bar{M}SE_t = \frac{RMSE_t}{RMSE_0}$. Trials show that the enhancement due to training is:

$$R\bar{M}SE_t = \frac{RMSE_t}{RMSE_0} \in [0.5, 1] \tag{1}$$

Enhancements "smaller" than 0.5 are rare (the modeler should check his model in this case). The range of the $R\bar{M}SE_t$ is used to implement the regularization (behavior of the springs). The basic idea for the regularization (reg) is to use a nonlinear behavior: the closer the outputs according the distances (d_l) between two outputs before training are, the stronger the springs' force. This heuristic ensures that small changes to the outputs can be made without significant influence on the springs, but large changes have to address the strength of the springs.

$$reg_t = w \times \frac{\sum_{l=0}^{n-1} 1.0/(d_l(t)/d_l(0))^2}{n-1} \tag{2}$$

The $d_l(t) = o_{l+1}(t) - o_l(t)$ are the distances between two outputs at iteration step t, n is the number of outputs, and w is a weighting factor to control the influence of regularization according to the $R\bar{M}SE_t$. Test simulation with fixed $w = 1$ gave a range of $reg_t \in [2, 20]$. Therefore, a $w = 0.01$ leads to a $reg_t \in [0.02, 0.2]$, which seems to be a reasonable compromise according to the range of the $R\bar{M}SE_t \in [0.5, 1]$. The objective function for training is shown in Function (3).

$$R\bar{M}SE_t + reg_t \Rightarrow Min! \tag{3}$$

During the optimization the $R\bar{M}SE_t$ will be reduced but the term reg_t precludes over training.

2.4. Implementation

The output training with regularization was implemented as a Python module "Pyfuzzy.py" as part of the SAMT2 open source software (Wieland et al., 2015). For optimization, nlopt[1] software was

[1] https://github.com/stevengj/nlopt