Quantifying Prediction Uncertainty in Regression using Random Fuzzy Sets: the ENNreg model

Thierry Denœux

***Abstract*—We introduce a neural network model for regression in which prediction uncertainty is quantified by Gaussian random fuzzy numbers (GRFNs), a newly introduced family of random fuzzy subsets of the real line that generalizes both Gaussian random variables and Gaussian possibility distributions. The output GRFN is constructed by combining GRFNs induced by prototypes using a combination operator that generalizes Dempster's rule of Evidence Theory. The three output units indicate the most plausible value of the response variable, variability around this value, and epistemic uncertainty. The network is trained by minimizing a loss function that generalizes the negative log-likelihood. Comparative experiments show that this method is competitive, both in terms of prediction accuracy and calibration error, with state-of-the-art techniques such as random forests or deep learning with Monte Carlo dropout. In addition, the model outputs a predictive belief function that can be shown to be calibrated, in the sense that it allows us to compute conservative prediction intervals with specified belief degree.**

***Index Terms*—Evidence theory, Dempster-Shafer theory, belief functions, machine learning, neural networks.**

I. Introduction

The quantification of prediction uncertainty in Machine Learning has recently gained a lot of attention (see, e.g., [1]–[4]). Whereas most approaches are based on Bayesian inference, other theories of uncertainty, such as Dempster-Shafer (DS) theory [5], [6], have also proved to be very promising [7], [8]. DS theory of belief functions, also known as theory of belief functions of evidence theory, is a mathematical formalism for reasoning with uncertainty [5], [6], which makes it possible to overcome some limitations of Bayesian inference. This formalism relies on two main components: (1) the representation of elementary pieces of evidence using belief functions, and (2) their combination by a conjunctive operator, referred to as Dempster's rule of combination. The greater flexibility of DS theory makes it suitable to reason and make decisions based on weak information, which would not be adequately represented by probability measures. In particular, it is possible, using belief functions, to distinguish between conflicting evidence equally supporting different hypotheses on the one hand, and sheer ignorance resulting from total lack of evidence on the other hand [9].

In machine learning, DS theory provides a powerful framework for expressing uncertainty about the output of a learning algorithm. Most applications so far have concerned clustering [10], [11] and classification [12]–[14]. In particular, in supervised classification, several algorithms have been proposed to learn *evidential classifiers*, defined as classifiers that represent class prediction uncertainty by belief functions. One of the first such classifier is the evidential K-nearest neighbor (EKNN) rule [12], which constructs a predictive belief function by combining the evidence of the K nearest neighbors of the feature vector under consideration using Dempster's rule. A similar idea is implemented in the evidential neural network (ENN) model introduced in [15], in which the learning vectors are summarized by prototypes, whose location in feature space is optimized together with other parameters. Recently, this idea has been applied to deep networks [7] [8] by adding a "DS layer" to a deep architecture.

The reason why applications of DS theory to machine learning have been confined to clustering and classification is that these learning tasks only require the definition of belief functions over a *finite* set of clusters or classes, a simple mathematical framework for which many tools and concepts of DS theory have been developed [6]. A belief function on a finite domain (or *frame of discernment*) can be conveniently represented by a mass function assigning probability masses to some subsets called *focal sets*. Such mass functions are easily combined by Dempster's rule. In contrast, applying DS theory to regression is more challenging, as the domain of the response variable is then the real line or a real interval. Whereas some mathematical models for generating belief functions on the real line, such as random intervals have been well studied [16], [17], combining such belief functions by Dempster's rule is often computationally difficult and requires Monte Carlo simulation [18].

A way to circumvent this difficulty is to discretize the response variable. This is the strategy followed in [19], in which a regression neural network directly extending the ENN model is proposed. However, discretizing a variable inevitably results in a loss of accuracy. Another approach, introduced in [20], is to directly extend the EKNN rule by defining, for each neighbor, a simple mass functions with two focal sets: a singleton composed of a real number and the frame of discernment. This simple method, called EVREG, was shown in [20] to have good performances as compared to nearest neighbor regression and other nonparametric methods, and to allow efficient handling of uncertain response data. However, as a nonparametric method, the performance of EVREG decreases with the input dimension, and it cannot compete with state-of-the-art regression methods.

In this paper, we propose a new regression neural network model[1] that quantifies prediction uncertainty using belief func-

T. Denœux is with Université de technologie de Compiègne, CNRS, Heudiasyc, France, and with Institut universitaire de France, Paris, France.

[1]This paper is an extended version of the short paper [21] presented at the 7th Int. Conference on Belief Functions, Paris, 26-28 October 2022.

tions, based on the theory of *epistemic random fuzzy sets* recently introduced in [22] [23]. Specifically, this work makes the following contributions:

1) We formulate a loss function for regression tasks in which the prediction is given in the form of a *predictive random fuzzy set*, and we introduce a corresponding notion of calibration;
2) We propose a new distance-based neural network model, called ENNreg, which computes predictions in the form of *Gaussian random fuzzy numbers (GRFNs)* [23] defined by three output values: mean, variance characterizing aleatory uncertainty, and precision reflecting epistemic uncertainty;
3) Through numerical experiments, we show the ENNreg model to be competitive with state-of-the-art machine learning models for regression tasks in terms of prediction accuracy and probabilistic calibration;
4) We show that the precision output can be adjusted to provide calibrated, conservative predictive belief functions.

The rest of this paper is organized as follows. The necessary background on random fuzzy sets and GRFNs is first recalled in Section II. The desired properties of predictive random fuzzy sets in regression tasks are then studied in Section III, and the proposed ENNreg model is introduced in Section IV. Experimental results are reported in Section V, and Section VI concludes the paper.

II. Random Fuzzy Sets

The theory of epistemic random fuzzy sets (RFSs) was introduced in [22] and [23] as a general framework encompassing both DS theory and possibility theory. We first recall some important definitions related to random fuzzy sets in Section II-A. Gaussian random fuzzy numbers, a parametric family of RFSs on the real line are then described in Section II-B.

A. General definitions

The notion of random fuzzy set generalizes that of random set [24] by mapping elements of a probability space to *fuzzy subsets* of the domain Θ of the variable of interest. This mathematical framework has been used to model a random experiment whose outcomes are fuzzy sets [25]–[27]. Here, as in [22] and [23], we use it as a model of unreliable and fuzzy evidence, directly generalizing the DS framework.

Formally, let $(\Omega, \Sigma_\Omega, P)$ be a probability space, (Θ, Σ_Θ) a measurable space, and $\widetilde{X}$ a mapping from Ω to the set $[0,1]^\Theta$ of *fuzzy subsets* of Θ. Under measurability conditions [28], the tuple $(\Omega, \Sigma_\Omega, P, \Theta, \Sigma_\Theta, \widetilde{X})$ is said to be a *random fuzzy set*.

Under the *epistemic* interpretation, we see Ω as a *set of interpretations* of a piece of evidence about a variable $\boldsymbol{\theta}$ taking values in Θ. If interpretation $\omega \in \Omega$ holds, we know that "$\boldsymbol{\theta}$ is $\widetilde{X}(\omega)$", i.e., $\boldsymbol{\theta}$ is constrained by the possibility distribution defined by $\widetilde{X}(\omega)$. The resulting random fuzzy set thus encodes a state of knowledge about variable $\boldsymbol{\theta}$. If all images $\widetilde{X}(\omega)$ are crisp, then $\widetilde{X}$ defines an ordinary random set. If $\widetilde{X}$ is a constant mapping, it is equivalent to specifying a unique fuzzy subset of Θ, which defines a possibility distribution [29].

Belief and plausibility functions: In the following, we assume random fuzzy set $\widetilde{X}$ to be *normalized*, i.e., to verifiy the following conditions:

1) For any $\omega \in \Omega$, $\widetilde{X}(\omega)$ is either the empty set, or a normal fuzzy set, i.e., the height of $\widetilde{X}(\omega)$, defined as $\mathsf{hgt}(\widetilde{X}(\omega)) = \sup_{\theta\in\Theta} \widetilde{X}(\omega)(\theta)$, is either 0 or 1;
2) The set of elements $\omega \in \Omega$ whose image is empty has zero probability, i.e., $P(\{\omega \in \Omega : \widetilde{X}(\omega) = \emptyset\}) = 0$.

Conditionally on $\omega \in \Omega$ being the true interpretation, we can define possibility and necessity measures [29] on Θ, respectively, as

$$\forall B \subseteq \Theta, \quad \Pi_{\widetilde{X}(\omega)}(B) = \sup_{\theta\in B} \widetilde{X}(\omega)(\theta), \tag{1a}$$

$$\text{and } N_{\widetilde{X}(\omega)}(B) = \begin{cases} 1 - \Pi_{\widetilde{X}(\omega)}(B^c) & \text{if } \widetilde{X}(\omega) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{1b}$$

Let $Bel_{\widetilde{X}}$ and $Pl_{\widetilde{X}}$ denote the mappings from Σ_Θ to $[0,1]$ that associate to each subset $B \in \Sigma_\Theta$, respectively, its expected necessity and its expected possibility, i.e.,

$$Bel_{\widetilde{X}}(B) = \int_\Omega N_{\widetilde{X}(\omega)}(B) dP(\omega), \tag{2a}$$

$$Pl_{\widetilde{X}}(B) = \int_\Omega \Pi_{\widetilde{X}(\omega)}(B) dP(\omega) = 1 - Bel_{\widetilde{X}}(B^c). \tag{2b}$$

It can be shown that $Bel_{\widetilde{X}}$ and $Pl_{\widetilde{X}}$ are, respectively, belief and plausibility functions [28].

Combination: Consider two epistemic random fuzzy sets $(\Omega_i, \Sigma_i, P_i, \Theta, \Sigma_\Theta, \widetilde{X}_i)$, $i = 1, 2$, encoding independent pieces of evidence. If interpretations $\omega_1 \in \Omega_1$ and $\omega_2 \in \Omega_2$ both hold, we know that "$\boldsymbol{\theta}$ is $\widetilde{X}_1(\omega_1)$" and "$\boldsymbol{\theta}$ is $\widetilde{X}_2(\omega_2)$". It is then natural to combine the fuzzy sets $\widetilde{X}_1(\omega_1)$ and $\widetilde{X}_2(\omega_2)$ by an intersection operator. As discussed in [22], the normalized product intersection operator $\odot$, defined for any two fuzzy subsets $\widetilde{F}$ and $\widetilde{G}$ of Θ as

$$(\widetilde{F} \odot \widetilde{G})(\theta) = \begin{cases} \dfrac{\widetilde{F}(\theta)\widetilde{G}(\theta)}{\mathsf{hgt}(\widetilde{F} \cdot \widetilde{G})} & \text{if } \mathsf{hgt}(\widetilde{F} \cdot \widetilde{G}) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

is suitable for combining fuzzy information from independent sources and it is associative [30]. We thus consider the mapping $\widetilde{X}_\odot$ from $\Omega_1 \times \Omega_2$ to $[0,1]^\Theta$ defined by $\widetilde{X}_\odot(\omega_1, \omega_2) = \widetilde{X}_1(\omega_1) \odot \widetilde{X}_2(\omega_2)$. To insure the normality condition, we need to condition the product measure $P_1 \times P_2$ on the set Θ^*_{12} of (partially) consistent interpretations, defined as

$$\Theta^*_{12} = \{(\omega_1, \omega_2) \in \Omega_1 \times \Omega_2 : \widetilde{X}_\odot(\omega_1, \omega_2) \neq \emptyset\}.$$

We then obtain the combined random fuzzy set

$$(\Omega_1 \times \Omega_2, \Sigma_1 \otimes \Sigma_2, P_{12}, \Theta, \Sigma_\Theta, \widetilde{X}_\odot),$$

where P_{12} is the conditioned product measure. This operation will be referred to as the *generalized product-intersection rule with hard normalization*[2]. The corresponding operator,

[2]In [23], we proposed a different operation based on a "soft normalization" process consisting in conditioning the product measure by a fuzzy subset of consistent interpretations. Here, hard normalization is preferred as it results in much simpler calculations.

denoted by $\boxplus$, is commutative and associative, and it is a proper generalization of Dempster's rule (it boils down to Dempster's rule when the combined random fuzzy sets are crisp).

B. Gaussian Random Fuzzy Numbers

To define a practical model of random fuzzy subset of the real line, we start with the definition of a *Gaussian Fuzzy Number* (GFN). A GFN is a fuzzy subset of $\mathbb{R}$ with membership function

$$\varphi(x; m, h) = \exp\left(-\frac{h}{2}(x-m)^2\right),$$

where $m \in \mathbb{R}$ is the mode and $h \in [0, +\infty]$ is the precision. Such a fuzzy number will be denoted by $\mathsf{GFN}(m, h)$. An interesting property of this family of fuzzy numbers is that it is closed with respect to the normalized product intersection: given two fuzzy numbers $\mathsf{GFN}(m_1, h_1)$ and $\mathsf{GFN}(m_2, h_2)$, their combination by the $\odot$ operator (3) yields the fuzzy number $\mathsf{GFN}(m_{12}, h_{12})$ with

$$m_{12} = \frac{h_1 m_1 + h_2 m_2}{h_1 + h_2} \quad \text{and} \quad h_{12} = h_1 + h_2. \tag{4}$$

A *Gaussian random fuzzy number* (GRFN) can now be defined as a GFN whose mode is a Gaussian random variable. More formally, let $(\Omega, \Sigma_\Omega, P)$ be a probability space and let $M : \Omega \to \mathbb{R}$ be a Gaussian random variable (GRV) with mean μ and variance σ^2. The random fuzzy set $\widetilde{X} : \Omega \to [0,1]^{\mathbb{R}}$ defined as

$$\widetilde{X}(\omega) = \mathsf{GFN}(M(\omega), h)$$

is called a GRFN with mean μ, variance σ^2 and precision h, which we write $\widetilde{X} \sim \widetilde{N}(\mu, \sigma^2, h)$. A GRFN is, thus, defined by a location parameter μ, and two parameters h and σ^2 corresponding, respectively, to possibilistic and probabilistic uncertainty.

A GRFN can be seen either as a generalized GRV with fuzzy mean, or as a generalized GFN with random mode. In particular, a GRFN $\widetilde{X}$ with infinite precision $h = +\infty$ is equivalent to a GRV with mean μ and variance σ^2, which we can write: $\widetilde{N}(\mu, \sigma^2, +\infty) = N(\mu, \sigma^2)$. If $\sigma^2 = 0$, M is a constant random variable taking value μ, and $\widetilde{X}$ is a possibilistic variable with possibility distribution $\mathsf{GFN}(\mu, h)$. Another case of interest is that where $h = 0$, in which case $\widetilde{X}(\omega)(x) = 1$ for all $\omega \in \Omega$ and $x \in \mathbb{R}$; the belief function induced by $\widetilde{X}$ is then vacuous.

As an illustration, Figure 1 shows ten realizations of two GRFNs $\widetilde{X}_1 \sim \widetilde{N}(0, (0.2)^2, 0.2)$ and $\widetilde{X}_2 \sim \widetilde{N}(5, 1, 5)$. GRFN $\widetilde{X}_1$ (left) has high imprecision and low variability, while $\widetilde{X}_2$ (right) is more precise but has higher variability. Formulas to compute the degrees belief $Bel_{\widetilde{X}}([x, y])$ and plausibility $Pl_{\widetilde{X}}([x, y])$ of any real interval $[x, y]$ are given in [23].

The usefulness of GRFNs as a model of uncertain information about a real quantity arises from the fact that GRFNs can easily be combined by the generalized product-intersection rule $\boxplus$ introduced in Section II-A. Given two GRFNs $\widetilde{X}_1 \sim \widetilde{N}(\mu_1, \sigma_1^2, h_1)$ and $\widetilde{X}_2 \sim \widetilde{N}(\mu_2, \sigma_2^2, h_2)$, we have, as a direction consequence of (4), $\widetilde{X}_1 \boxplus \widetilde{X}_2 \sim \widetilde{N}(\mu_{12}, \sigma_{12}^2, h_{12})$, with

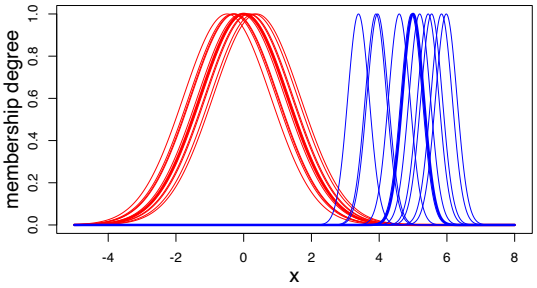


Fig. 1: Ten realizations of GRFNs $\widetilde{X}_1 \sim \widetilde{N}(0, (0.2)^2, 0.2)$ (left, in red) and $\widetilde{X}_2 \sim \widetilde{N}(5, 1, 5)$ (right, in blue). For each GRFN $\widetilde{X}_k$, the possibility distribution $\mathsf{GFN}(\mu_k, h_k)$ is also plotted as a thick line.

$$\mu_{12} = \frac{h_1\mu_1 + h_2\mu_2}{h_1 + h_2}, \quad \sigma_{12}^2 = \frac{h_1^2\sigma_1^2 + h_2^2\sigma_2^2}{(h_1 + h_2)^2}, \tag{5a}$$

$$\text{and} \quad h_{12} = h_1 + h_2. \tag{5b}$$

III. Predictive Random Fuzzy Sets

In this section and in the rest of this paper, we consider a regression problem, in which the task is to predict a continuous random response variable Y from a random vector $\boldsymbol{X} = (X_1, \ldots, X_p)$ of p input variables. In Section IV we will propose a neural network model that, given an observed input vector $\boldsymbol{X} = \boldsymbol{x}$, computes a GRFN $\widetilde{Y}(\boldsymbol{x})$ with associated belief function $Bel_{\widetilde{Y}(\boldsymbol{x})}$ that "approximates" Y in some way. The network will be trained using a training set $\mathcal{T} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ assumed to be an independent and identically distributed (iid) sample from $(\boldsymbol{X}, Y)$. In this section, we define desirable properties for a predictive RFS $\widetilde{Y}(\boldsymbol{x})$. A loss function measuring predictive accuracy is first defined in Section III-A. A notion of calibration is then introduced in Section III-B.

A. Loss function

When predicting a random variable Y by a probability measure $\widehat{P}$ with density function $\widehat{f}$, we typically measure the prediction error (or loss) by the negative log-likelihood

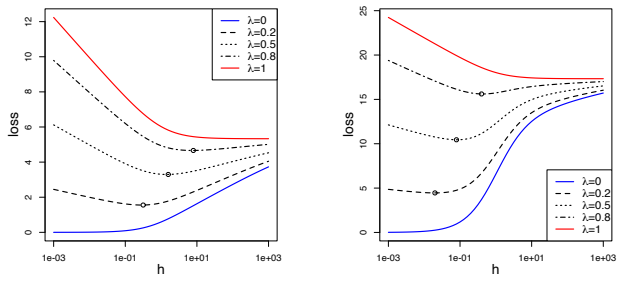
$$\mathcal{L}(y, \widehat{f}) = -\ln \widehat{f}(y), \tag{6}$$

which can be interpreted as follows. The continuous variable Y is always observed with finite precision, so we actually observe an interval $[y]_\epsilon = [y - \epsilon, y + \epsilon]$ centered at y. Denoting by $\widehat{F}$ the distribution function of $\widehat{P}$, the probability of that interval is

$$\widehat{P}([y]_\epsilon) = \widehat{F}(y + \epsilon) - \widehat{F}(y - \epsilon) \approx 2\widehat{f}(y)\epsilon,$$

and its logarithm is equal to $\ln \widehat{f}(y)$ plus a constant that does not depend on $\widehat{f}$. Minimizing (6) thus amounts to maximizing the probability of the observations.

In the case where the prediction takes the form of a RFS $\widetilde{Y}$, we no longer have a single probability of the observation $[y]_\epsilon$,



(a) $\widetilde{Y} \sim \widetilde{N}(1, 1, h)$ (b) $\widetilde{Y} \sim \widetilde{N}(5, 1, h)$

Fig. 2: Loss $\mathcal{L}_{\lambda,\epsilon}(0, \widetilde{Y})$ for a GRFN $\widetilde{Y} \sim \widetilde{N}(\mu, 1, h)$ as a function of h, for $\mu = 1$ (a) and $\mu = 5$ (b) and $\lambda \in \{0, 0.2, 0.5, 0.8, 1\}$.

but two numbers: a degree of belief $Bel_{\widetilde{Y}}([y]_\epsilon)$ and a degree of plausibility $Pl_{\widetilde{Y}}([y]_\epsilon)$. We could, thus, replace (6) by either $\overline{\mathcal{L}}_\epsilon(y, \widetilde{Y}) = -\ln Bel_{\widetilde{Y}}([y]_\epsilon)$, or $\underline{\mathcal{L}}_\epsilon(y, \widetilde{Y}) = -\ln Pl_{\widetilde{Y}}([y]_\epsilon)$, where the precision ϵ now appears explicitly in the expression of the loss function. However, none of these two loss functions adequately measures the quality of the imprecise predictions as expressed by a RFS $\widetilde{Y}$. To see this, let us assume that $\widetilde{Y} \sim \widetilde{N}(\mu, \sigma^2, h)$ is a GRFN with precision h. The following inequality holds: $Bel_{\widetilde{Y}}([y]_\epsilon) \leq \Phi\left(\frac{y+\epsilon-\mu}{\sigma}\right) - \Phi\left(\frac{y-\epsilon-\mu}{\sigma}\right)$, with an equality when $h = +\infty$. Consequently, whatever the values of μ and σ^2, a GRV with mean μ and variance σ^2 always achieves a smaller loss than that of $\widetilde{Y}$, which implies that a higher imprecision of the RFS is never rewarded by a decrease of the loss function. Now, when $h = 0$, we have $Pl_{\widetilde{Y}}([y]_\epsilon) = 1$ for any y, μ and σ^2: the loss $\underline{\mathcal{L}}_\epsilon(y, \widetilde{Y})$ is, thus, maximized by the vacuous, uninformative RFS.

From the above analysis, we propose to consider as the loss function a weighted sum of $\overline{\mathcal{L}}_\epsilon(y, \widetilde{Y})$ and $\underline{\mathcal{L}}_\epsilon(y, \widetilde{Y})$:

$$\mathcal{L}_{\lambda,\epsilon}(y, \widetilde{Y}) = \lambda \overline{\mathcal{L}}_\epsilon(y, \widetilde{Y}) + (1 - \lambda)\underline{\mathcal{L}}_\epsilon(y, \widetilde{Y}), \tag{7}$$

where $\lambda \in [0, 1]$ is a hyperparameter. Choosing a smaller value of λ amounts to favoring more cautious predictions. Figure 2 shows the variation of the loss $\mathcal{L}_{\lambda,\epsilon}(0, \widetilde{Y})$ with h for $\lambda \in \{0, 0.2, 0.5, 0.8, 1\}$, $\epsilon = 0.01$, and $\widetilde{Y} \sim \widetilde{N}(\mu, 1, h)$ with $\mu = 1$ (Figure 2a) and $\mu = 5$ (Figure 2b). We can see that the loss is minimized for $h = 0$ when $\lambda = 0$, for $h \to +\infty$ when $\lambda = 1$, and for some intermediate value of h when $0 < \lambda < 1$. For any given value of λ, the optimum precision h is smaller when $\mu = 5$: a larger error is compensated by a smaller precision.

When μ, σ^2 and h are free to vary, loss function (7) is minimal for $\mu \in [y]_\epsilon$, $h = +\infty$ and $\sigma^2 \to 0$. As in support vector regression [31], ϵ can be seen as a value under which the error is not penalized. We can also remark that, when ϵ is small, we have, for a probabilistic GRFN $\widetilde{Y} \sim \widetilde{N}(\mu, \sigma^2, +\infty)$, $\mathcal{L}_{\lambda,\epsilon}(y, \widetilde{Y}) \approx -\ln \phi(\frac{y-\mu}{\sigma}) - \ln \epsilon$, where ϕ denotes the standard normal probability density function; loss function (7) then approximates the negative log-likelihood, up to an additive constant.

In section IV, we will describe a neural network model for computing a GRFN $\widetilde{Y}(\boldsymbol{x}; \Psi)$ from an input vector $\boldsymbol{x}$ and a parameter vector Ψ. The network will be trained by minimizing the average of loss function (7) over the training set.

B. Evidential Calibration

In the case of probabilistic predictions of the response variable Y, we usually expect the predictive probability distribution to be close to the true conditional distribution of Y given $\boldsymbol{X} = \boldsymbol{x}$. More precisely, let $\widehat{F}_{Y|\boldsymbol{x}}$ be a predictive cumulative distribution function (cdf) for Y given $\boldsymbol{X} = \boldsymbol{x}$. For any $\alpha \in (0, 1)$, we can construct a prediction interval at level α as

$$\mathcal{C}_\alpha(\boldsymbol{x}) = \left[\widehat{F}^{-1}_{Y|\boldsymbol{x}}\left(\frac{1-\alpha}{2}\right), \widehat{F}^{-1}_{Y|\boldsymbol{x}}\left(\frac{1+\alpha}{2}\right)\right].$$

These intervals are said to be *calibrated* if

$$\forall \alpha \in (0, 1), \quad P_{\boldsymbol{X},Y}\left(Y \in \mathcal{C}_\alpha(\boldsymbol{X})\right) \approx \alpha. \tag{8}$$

This property can be checked by estimating the coverage rates $P_{\boldsymbol{X},Y}\left(Y \in \mathcal{C}_\alpha(\boldsymbol{X})\right)$ for different values of α using a validation set and plotting the estimates vs. the nominal values α. The resulting graph is known as a *calibration plot*.

In the case considered here, where the response is predicted for $\boldsymbol{X} = \boldsymbol{x}$ by a GRFN $\widetilde{Y}(\boldsymbol{x})$, we also need a notion of calibration that can easily be checked graphically. Different notions of calibration for belief functions are reviewed in [32]. Recently, Cella and Martin [33] proposed two definitions for "valid" preditions in the belief function framework. However, the weaker definition is too weak to be practically useful, while the stronger one appears to be verified only for consonant belief functions. Here, we will adopt a simple definition based on an immediate generalization of (8).

For any $\alpha \in (0, 1]$, we define an α-level *belief prediction interval (BPI)* as an interval $\mathcal{B}_\alpha(\boldsymbol{x}) = \mu(\boldsymbol{x}) \pm a(\boldsymbol{x})$ centered at $\mu(\boldsymbol{x})$, such that $Bel_{\widetilde{Y}(\boldsymbol{x})}(\mathcal{B}_\alpha(\boldsymbol{x})) = \alpha$. A BPI at level α is, thus, an interval that is believed to a degree α to contain the true value of the response variable Y. The predictions will be said to be *calibrated* if, for all $\alpha \in (0, 1]$, α-level BPIs have a coverage probability at least equal to α, i.e,

$$\forall \alpha \in (0, 1], \quad P_{\boldsymbol{X},Y}\left(Y \in \mathcal{B}_\alpha(\boldsymbol{X})\right) \geq \alpha. \tag{9}$$

As in the probabilistic case, the calibration of evidential predictions can be checked graphically using a calibration plot (see Figure 5 in Section V-A). The predictions are calibrated if the curve lies above the first diagonal, and the predictions are all the more *precise* that the curve is close to the first diagonal.

IV. Neural network model

In this section, we propose a neural network model[3], called ENNreg, which for an observed input vector $\boldsymbol{X} = \boldsymbol{x}$ computes a GRFN $\widetilde{Y}(\boldsymbol{x})$, with associated belief function $Bel_{\widetilde{Y}(\boldsymbol{x})}$ quantifying the uncertainty on Y given $\boldsymbol{x}$. As the ENN model introduced in [15] for classification, ENNreg is based on prototypes. The distances to the prototypes are

[3]This model is implemented in the R package `evreg` [34] available from CRAN at https://CRAN.R-project.org/package=evreg.

treated as independent pieces of evidence about the response and are combined by the generalized product-intersection rule $\boxplus$ recalled in Section II-A. The network architecture and propagation equations will first be described in Section IV-A. Learning will then be discussed in Section IV-B. Finally, a brief comparison with recent models for uncertainty quantification in deep networks will performed in Section IV-C.

A. Neural network architecture

Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K$ denote K vectors in the p-dimensional feature space, called prototypes. The similarity between input vector $\boldsymbol{x}$ and prototype $\boldsymbol{w}_k$ is measured by

$$s_k(\boldsymbol{x}) = \exp(-\gamma_k^2 \|\boldsymbol{x} - \boldsymbol{w}_k\|^2), \tag{10}$$

where γ_k is a positive scale parameter. The evidence of prototype $\boldsymbol{w}_k$ is represented by a GRFN

$$\widetilde{Y}_k(\boldsymbol{x}) \sim \widetilde{N}(\mu_k(\boldsymbol{x}), \sigma_k^2, s_k(\boldsymbol{x}) h_k)$$

where σ_k^2 and h_k are variance and precision parameters for prototype k; the mean $\mu_k(\boldsymbol{x})$ is defined as

$$\mu_k(\boldsymbol{x}) = \boldsymbol{\beta}_k^T \boldsymbol{x} + \beta_{k0}, \tag{11}$$

where $\boldsymbol{\beta}_k$ is a p-dimensional vector of coefficients, and β_{k0} is a scalar parameter. The quantity $\mu_k(\boldsymbol{x})$ can be seen as an estimate of the conditional expectation of the response Y given that $\boldsymbol{x}$ is close to $\boldsymbol{w}_k$, while σ_k^2 is an estimate of the conditional variance. When the distance $\|\boldsymbol{x} - \boldsymbol{w}_k\|$ tends to infinity, the precision $s_k(\boldsymbol{x}) h_k$ tends to 0 and $\widetilde{Y}_k(\boldsymbol{x})$ becomes vacuous. The vector ψ_k of parameters associated to prototype k is, thus, $\psi_k = (\boldsymbol{w}_k, \gamma_k, \boldsymbol{\beta}_k, \beta_{k0}, \sigma_k^2, h_k)$.

The output $\widetilde{Y}(\boldsymbol{x})$ for input $\boldsymbol{x}$ is computed by combining the GRFNs $\widetilde{Y}_k(\boldsymbol{x})$, $k = 1, \ldots, K$ induced by the K prototypes using the $\boxplus$ operator. From (5), it is a GRFN $\widetilde{Y}(\boldsymbol{x}) \sim \widetilde{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}), h(\boldsymbol{x}))$, with

$$\mu(\boldsymbol{x}) = \frac{\sum_{k=1}^K s_k(\boldsymbol{x}) h_k \mu_k(\boldsymbol{x})}{\sum_{k=1}^K s_k(\boldsymbol{x}) h_k}, \sigma^2(\boldsymbol{x}) = \frac{\sum_{k=1}^K s_k^2(\boldsymbol{x}) h_k^2 \sigma_k^2}{\left(\sum_{k=1}^K s_k(\boldsymbol{x}) h_k\right)^2},$$

and $h(\boldsymbol{x}) = \sum_{k=1}^K s_k(\boldsymbol{x}) h_k$. We can observe that $\sigma^2(\boldsymbol{x})$ and $h(\boldsymbol{x})$ represent different kinds of uncertainty. The variance output $\sigma^2(\boldsymbol{x})$ estimates the conditional variance of Y given the input $\boldsymbol{x}$: it, thus, corresponds to *aleatory uncertainty* (the larger $\sigma^2(\boldsymbol{x})$, the more uncertainty). In contrast, the precision output $h(\boldsymbol{x})$ gets smaller and tends to zero when the distance to the prototypes increases: it corresponds to *epistemic uncertainty* (the smaller $h(\boldsymbol{x})$, the more uncertainty).

The calculation of $\widetilde{Y}(\boldsymbol{x})$ can be performed by a neural network with an input layer, two hidden layers of $2K$ units and an output layer of three units, as shown in Figure 3. The first hidden layer is composed of K pairs of units totally connected to the input units: a radial basis function (RBF) unit with weight vector $\boldsymbol{w}_k$ that computes $s_k(\boldsymbol{x})$ and a linear unit with weight vector $\boldsymbol{\beta}_k$ and bias β_{k0} that computes $\mu_k(\boldsymbol{x})$. The second layer is also composed of K pairs of units: in each pair k, a "squaring" unit connected to RBF unit k of the previous layer computes $s_k^2(\boldsymbol{x})$, and a "product" unit connected to RBF unit k and linear unit k of the previous layer computes the product $s_k(\boldsymbol{x})\mu_k(\boldsymbol{x})$. Finally, the output layer is composed of (1) a linear unit with shortcut connections from the RBF units in the first hidden layer and weights (h_k) that computes $h(\boldsymbol{x})$; (2) a unit connected to the squaring units of the previous layer with weights $(h_k^2 s_k^2(\boldsymbol{x}))$ that outputs $\sigma^2(\boldsymbol{x})$, and (3) a unit connected to the product units of the previous layer with weights (h_k) that computes $\mu(\boldsymbol{x})$.

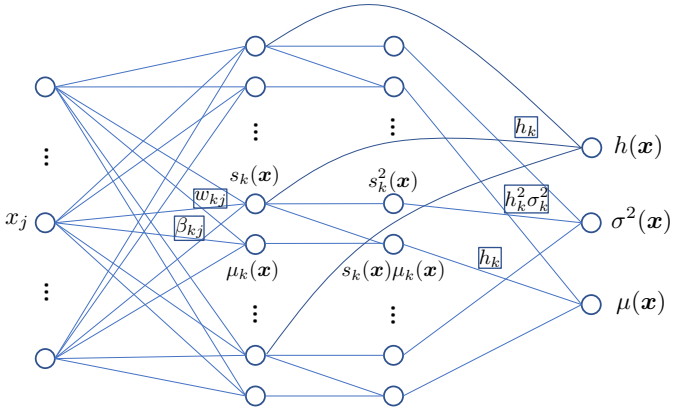


Fig. 3: Neural network architecture.

Although the structure of this network is more complex than that of RBF networks [35] [36], the complexity of one input propagation is the same, i.e., $O(pK)$. We can also remark that this neural network model generalizes both RBF networks and the linear model:

- If $\boldsymbol{\beta}_k = 0$ for all k, then $\mu_k(\boldsymbol{x}) = \beta_{k0}$, and $\mu(\boldsymbol{x})$ is identical to the output of an RBF neural network [37] with K hidden units, hidden-to-output weights $h_k \beta_{k0}$ and normalized outputs;
- If $\gamma_k = 0$ for all k, the output $\mu(\boldsymbol{x})$ becomes linear in $\boldsymbol{x}$ and the variance component $\sigma^2(\boldsymbol{x})$ is constant. We then have a linear model with constant variance.

Finally, ENNreg can also be interpreted as a special kind of fuzzy system (see [38] for a general introduction and, e.g., [39] for a recent application to fuzzy control). From this point of view, each prototype k can be interpreted as a fuzzy rule

IF x_1 is $\widetilde{W}_{k1}$ AND x_2 is $\widetilde{W}_{k2}$ AND $\ldots$
AND x_p is $\widetilde{W}_{kp}$ THEN Y is $\widetilde{N}(\mu_k(\boldsymbol{x}), \sigma_k^2, h_k)$,

where $\widetilde{W}_{kj}$, $j = 1, \ldots, p$ are fuzzy sets of the real line with membership function $\widetilde{W}_{kj}(x_j) = \exp(-\gamma_k (x_i - w_{kj})^2)$. Using the product t-norm, the degree of firing of this rule is $\prod_{j=1}^p \widetilde{W}_{kj}(x_j) = s_k(\boldsymbol{x})$. The GRFN in the consequent part is then "discounted" by multiplying its precision by $s_k(\boldsymbol{x})$, and the outputs from each of the K rules are combined by the generalized product-intersection operator $\boxplus$. This analogy with fuzzy systems could suggest alternative models based on RFSs; this is left for further research.

B. Learning

Let $\Psi = (\psi_1, \ldots, \psi_K)$ denote the vector of all parameters in the model. Given a training set $\mathcal{T} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, the accuracy of the predictions can be measured by the loss

function (7) introduced in Section III-A. To train the ENNreg model described in Section IV-A, we will use the regularized average loss

$$C^{(R)}_{\lambda,\epsilon,\xi,\rho}(\Psi) = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\lambda,\epsilon}(y_i,\widetilde{Y}(\boldsymbol{x}_i;\Psi))}_{C_{\lambda,\epsilon}(\Psi)} + \underbrace{\frac{\xi}{K}\sum_{k=1}^{K}h_k}_{R_1(\Psi)} + \underbrace{\frac{\rho}{K}\sum_{k=1}^{K}\gamma_k^2}_{R_2(\Psi)}, \quad (12)$$

where $C_{\lambda,\epsilon}(\Psi)$ is the average loss, $R_1(\Psi)$ and $R_2(\Psi)$ are two regularizers, and ξ, ρ are regularization coefficients. The first regularizer $R_1(\Psi)$ has the effect of reducing the number of prototypes used for the prediction (setting $h_k = 0$ amounts to discarding prototype k), while $R_2(\Psi)$ shrinks the solution towards a linear model (as mentioned before, setting $\gamma_k = 0$ for all k yields a linear model). Loss function (12) can be minimized using any batch or online gradient-based optimization procedure. The constraints $h_k \geq 0$ are imposed by introducing intermediate variables η_k such that $h_k = \eta_k^2$. The gradient can be computed by error back-propagation, with complexity $O(Kp)$.

As in RBF networks, the prototypes in the ENNreg model can easily be initialized using a clustering procedure such as the K-means algorithm. As before, let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ be the training set. Let $\mathcal{I}_k$ denote the set of indices of input vectors in cluster k, and $n_k = |\mathcal{I}_k|$. The prototypes are initialized as the cluster centers, $\boldsymbol{w}_k = \frac{1}{n_k}\sum_{i\in\mathcal{I}_k}\boldsymbol{x}_i$, and γ_k is initialized proportionally to the inverse square root of the mean squared distances within cluster k:

$$\gamma_k = \frac{1}{\sqrt{2}}\left(\sum_{i\in\mathcal{I}_k}\frac{1}{n_k}\|\boldsymbol{x}_i - \boldsymbol{w}_k\|^2\right)^{-1/2}.$$

Parameters β_{k0} and σ_k^2 are set, respectively, to the mean and variance of the response variable in cluster k. Finally, we set $\boldsymbol{\beta}_k = 0$ and $h_k = 1$.

Overall, the model has five hyperparameters: the number K of prototypes, coefficients ϵ and λ in the definition of the loss (7), and regularization coefficients ξ and ρ in (12). In practice, K can be set to a large value, and the effective number of prototypes can be controlled by ξ. We found the network performance to be quite robust to the choice of ϵ and λ. In the experiments reported in Section V, these hyperparameters were fixed to $\epsilon = 0.01\widehat{\sigma}_Y$, where $\widehat{\sigma}_Y$ is the sample standard deviation of Y, and $\lambda = 0.9$. Only ξ and ρ thus have to be tuned using either cross-validation or the hold-out method.

C. Comparison with alternative approaches

In ENNreg, prediction uncertainty is quantified by the output GRFN $\widetilde{Y}(\boldsymbol{x})$ obtained by combining evidence from different prototypes, in line with DS theory [6]. Our approach is, thus, radically different from recently proposed methods for quantifying prediction uncertainty in deep networks, which are rooted in Bayesian inference. For instance, probabilistic back-propagation (PBP) [1] performs approximate inference in a Bayesian neural network in which each weight has a one-dimensional Gaussian prior distribution; the marginal distributions of the activations in each layer are approximated by one-dimensional Gaussians with the same means and variances. Monte Carlo dropout, another method introduced in [2], can be interpreted as approximate Bayesian inference of parameters in a Gaussian process; the first two moments of the output predictive distribution are estimated. In the deep ensemble method [3], a set of neural networks is trained by minimizing the negative log-likelihood; the predictive distribution is a mixture of Gaussians, which is approximated by a Gaussian distribution with the same mean and variance. Finally, "deep evidential regression"[4] introduced in [4] assumes a Gaussian prior on the conditional mean of the response variable and an Inverse-Gamma prior on its conditional variance; the hyperparameters of these "evidential priors" are learnt by minimizing a suitable loss function.

By considering only the mean $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$ outputs of our method, it is possible to compare it experimentally with the above Bayesian approaches. This will be done in Section V-B, where we will show that ENNreg has slightly better prediction accuracy than Bayesian methods, and is on par in terms of probabilistic calibration. Additionally, our model has a precision output $h(\boldsymbol{x})$ that can be adjusted to provide *conservative predictions* calibrated in the sense of the definition introduced in Section III-B.

V. Experimental results

We first give an illustrative example in Section V-A. Results from a comparative experiment are then reported in Section V-B.

A. Illustrative example

As an illustrative example, we consider independent and identically distributed (iid) data simulated from the following distribution: the one-dimensional input X has a uniform distribution in the interval $[-2, 2]$, and

$$Y = X + (\sin 3X)^3 + \frac{X+2}{4\sqrt{2}}U,$$

where $U \sim N(0,1)$ is a standard normal random variable. The conditional standard deviation of Y given $X = x$ thus increases linearly from 0 for $x = -2$ to $1/\sqrt{2}$ for $x = 2$.

We generated a learning set of size $n = 300$ and a validation set of the same size. We initialized a network with $K = 30$ prototypes. The values of hyperparameters ξ and ρ minimizing the mean squared error on the validation set were $\xi = 10^{-4}$ and $\rho = 0.01$. Figure 4 shows the outputs $\mu(x)$ of a network trained with these values, together with BPIs at levels $\alpha \in \{0.5, 0.9, 0.99\}$.

A calibration plot is shown in Figure 5, representing the proportion of α-level BPIs containing the observed value of the response in the validation set, for $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$. In Figure 5, we also show the calibration graph for the probabilistic prediction intervals defined as $\mu(x) \pm u_{(1+\alpha)/2}\sigma(x)$.

[4]This method is not related to the DS theory of evidence.

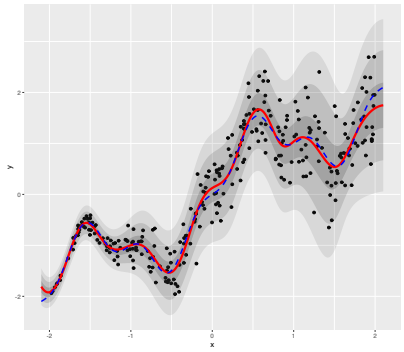


Fig. 4: Learning data, true regression function (blue broken line) and predictions obtained by the trained ENNreg model with $K = 30$ prototypes: expected values $\mu(x)$ (red solid line) and BPIs at levels $\alpha \in \{0.5, 0.9, 0.99\}$ (grey areas).

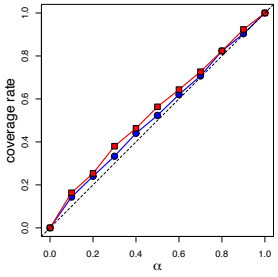


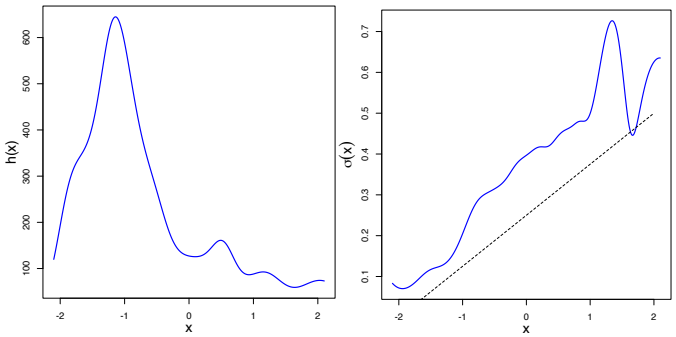
Fig. 5: Calibration plots for probabilistic prediction intervals (blue circles) and belief prediction intervals (red squares), showing the validation coverage rates vs. the belief or confidence levels $\alpha \in \{0.1, \ldots, 0.9\}$.

In this particular case, both prediction intervals are calibrated, the BPIs being more conservative than the probabilistic ones. We can remark that we can make the BPIs less conservative by multiplying the output precisions $h(x)$ by some constant $c > 1$. The probabilistic intervals are recovered when $c \to +\infty$. Conversely, we can make the BPIs more conservative by multiplying the output precision by a constant $c < 1$. This technique will be illustrated in Section V-B.

From Figure 5, we can see that the calibration curve of BPIs is close to that of probabilistic prediction intervals, which indicates that, for this dataset, the output precision $h(x)$ is quite high. This is confirmed by Figure 6a showing $h(x)$ vs. x. Precision first increases, and then decreases with x as the errors $|y_i - \mu(x_i)|$ increase, which can be explained by Figure 2: larger errors result in more imprecise predictions. Figure 6b shows the output standard deviation $\sigma(x)$ as a function of x: it slightly overestimates the true standard deviation, but the increasing trend is well captured.

B. Comparative experiments

a) Comparison with state-of-the-art regression methods: We compared the performance of ENNreg in terms of root mean squared (RMS) error to those of eight alternative regression methods on ten datasets. Eight of the datasets



(a) (b)

Fig. 6: Output precision $h(x)$ (a) and standard deviation $\sigma(x)$ (b) for the simulated data. The broken line indicates the true standard deviation.

TABLE I: Characteristics of the datasets.

	n	p	response
Boston	506	13	`medv`
Energy	768	8	`Y2`
Concrete	1030	8	`strength`
Yacht	308	6	`Y`
Wine	1599	11	`quality`
kin8nm	8192	8	`V9`
Crime	1994	100	`ViolentCrimesPerPop`
Residential	372	103	`V10`
Airfoil	1503	5	`Y`
Bike	731	9	`cnt`

(Energy efficiency, Concrete Compressive Strength, Yacht Hydrodynamics, Wine quality – red, Communities and Crime, Residential building, Airfoil Self-Noise, Bike sharing) are available from the UCI Machine Learning Repository[5]. The Boston dataset is included in the R package `MASS` [40], and the kin8nm dataset was downloaded from the OpenML web site[6]. The characteristics of these datasets are summarized in Table I. We also give in Table I the name of the response variable, as some datasets have several possible response variables. For the "Communities and Crime" dataset, variables with two missing values or more were discarded. For the "Bike sharing dataset", the nine predictors were `season`, `year`, `month`, `holiday`, `weekday`, `weathersit`, `atemp`, `hum` and `windspeed`.

The eight methods are:

- Four nonlinear regression algorithms using RBF kernel functions: RBF networks, Relevance Vector Machines (RVM), Support Vector Machines (SVM), and Gaussian Processes (GP);
- Two state-of-the-art methods for nonlinear regression: Random Forests (RF) and Multilayer Perceptrons (MLP) with one hidden layer of sigmoidal units;
- Two regularized linear regression methods: Ridge and Lasso.

For all methods, except RBF networks, we used the implementation in the R package `caret` [41]. For training ENNreg, we used batch learning using the algorithm described in [42]

[5]https://archive.ics.uci.edu/ml/.
[6]https://www.openml.org.

TABLE II: Average RMS and standard errors for ENNreg and eight alternative algorithms on ten datasets.

	ENNreg	RBF	RVM	SVM	GP	RF	MLP	ridge	lasso
Boston	**2.87 ± 0.14**	3.31 ± 0.19	3.42 ± 0.17	3.17 ± 0.15	3.70 ± 0.22	**3.11 ± 0.14**	**3.14 ± 0.14**	5.05 ± 0.23	5.02 ± 0.21
Energy	**1.06 ± 0.05**	2.06 ± 0.08	1.79 ± 0.05	1.39 ± 0.06	2.58 ± 0.07	1.75 ± 0.06	**0.95 ± 0.16**	3.56 ± 0.10	3.26 ± 0.09
Concr.	5.10 ± 0.12	6.30 ± 0.19	6.38 ± 0.16	5.62 ± 0.13	6.93 ± 0.13	**4.64 ± 0.12**	**4.82 ± 0.16**	10.71 ± 0.17	10.63 ± 0.18
Yacht	**0.44 ± 0.04**	2.00 ± 0.20	1.88 ± 0.20	1.93 ± 0.11	6.12 ± 0.31	0.96 ± 0.08	**0.50 ± 0.05**	8.48 ± 0.24	8.35 ± 0.23
Wine	0.63 ± 0.01	0.63 ± 0.01	0.80 ± 0.02	0.61 ± 0.01	0.61 ± 0.01	**0.56 ± 0.01**	0.77 ± 0.01	0.65 ± 0.01	0.65 ± 0.01
kin8nm	0.08 ± 0.00	0.11 ± 0.00	–	0.09 ± 0.00	0.08 ± 0.00	0.14 ± 0.00	**0.07 ± 0.00**	0.20 ± 0.00	0.20 ± 0.00
Crime	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**	**0.14 ± 0.00**
Resid.	**0.11 ± 0.01**	0.16 ± 0.01	0.17 ± 0.01	0.15 ± 0.01	0.22 ± 0.01	0.16 ± 0.01	0.14 ± 0.01	0.18 ± 0.01	0.17 ± 0.01
Airfoil	**1.46 ± 0.03**	1.70 ± 0.04	2.58 ± 0.04	2.37 ± 0.04	2.49 ± 0.04	**1.44 ± 0.04**	1.53 ± 0.04	3.84 ± 0.04	3.83 ± 0.04
Bike	**6.59 ± 0.19**	**6.49 ± 0.15**	**6.64 ± 0.14**	7.11 ± 0.16	7.55 ± 0.14	6.86 ± 0.17	9.68 ± 0.20	7.82 ± 0.16	7.76 ± 0.17

TABLE III: Average CPU time and standard errors for ENNreg and RBF networks on five datasets.

Dataset	ENNreg	RBF
Boston	2.63 ± 0.09	2.88 ± 0.08
Concrete	2.45 ± 0.08	8.16 ± 0.19
Wine	1.92 ± 0.03	4.97 ± 0.24
Resid.	11.38 ± 0.47	23.43 ± 0.55
Airfoil	20.50 ± 1.30	28.90 ± 0.39

for all datasets, except kin8nm for which we used mini-batch stochastic gradient descent. Each dataset was split randomly into training and test sets containing, respectively, 90% and 10% of the observations. The random splits were repeated 20 times. All input variables were scaled to have zero mean and unit standard deviation. For each method, hyperparameters were tuned by 5-fold cross-validation. For ENNreg, we set $\lambda = 0.9$ and $\epsilon = 0.01\widehat{\sigma}_Y$ (where $\widehat{\sigma}_Y$ is the estimated standard deviation of the response variable) for all the simulations. The number of prototypes was fixed to $K = 30$ for all datasets except kin8nm and Airfoil, for which it was set to $K = 100$. Hyperparameters ξ and ρ were tuned by cross-validation.

The results[7] are reported in Table II. We can see that ENNreg stands among the best methods for seven out of the ten datasets. It is only outperformed by RF and MLP on the Concrete dataset, RF on the Wine dataset, and MLP on the kin8nm dataset. In most cases, it performs strictly better than other RBF-based methods (except for the Crime dataset, on which all methods have equivalent performances, and for the Bike dataset, on which it performs as well as RBF and RVM). Overall, it appears that ENNreg is a very competitive regression method in terms of prediction accuracy.

Comparing computing time across all methods is difficult because it highly depends on implementation. We have seen in Section IV that both forward and back propagation in ENNreg can be performed in linear time with respect to both the number of inputs and the number of prototypes; consequently, the algorithmic efficiency of ENNreg is similar to those of other neural network methods. We have done a comparison with the RBF network model because it was implemented in a similar way as ENNreg and with the same optimization algorithm. Average computing times for both methods on five datasets are shown in Table III. We can see that ENNreg is actually faster than RBF networks with the same number of prototypes, which can be explained by faster convergence due to greater flexibility.

b) Comparison with alternative uncertainty quantification methods: We also compared our results to published results obtained with the four neural network methods reviewed in Section IV-C, on six of the ten datasets for which published results with these methods are available. The RMS values are shown in Table IV. We can see that ENNreg performs strictly better than, or as well as these four methods on the six considered datasets in terms of RMS.

The PBP, Monte Carlo dropout, deep ensemble and evidential regression methods were actually introduced for uncertainty quantification in deep networks. Probabilistic calibration is usually measured by the negative loglikelihood (NLL), assuming Gaussian errors. For ENNreg, NNL can be computed using the mean $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$ output, ignoring the precision $h(\boldsymbol{x})$. ENNreg is then seen as a probabilistic method. The NLL values for ENNreg and the four neural network methods on the six datasets (as reported in [1]–[4]) are shown in Table IV. We can see that ENNreg performs similarly to the other methods and stands among the best methods for four out of the six datasets in terms of probabilistic calibration as measured by NLL.

c) Evidential calibration: As mentioned in Section V-A, the ENNreg model provides not only conditional mean and variance estimates as probabilistic models, but a precision output $h(\boldsymbol{x})$ that can be used to compute calibrated BPIs as defined by (9). Figure 7 shows calibration plots for four or the six datasets above. Each plot shows the calibration curves of (1) probabilistic prediction intervals (using only $\mu(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x})$), (2) raw BPIs and (3) adjusted BPIs obtained by multiplying the output $h(\boldsymbol{x})$ by a constant $c > 0$ in such a way that the calibration curve lies above the diagonal, but as close as possible to it. We can distinguish several cases. For the Boston and Energy datasets (Figures 7a and 7b), the BPIs are calibrated but too conservative. Applying a correction factor $c = 2$ increases the precision of the predictions. In the case of the Concrete dataset (Figure 7c), the BPIs before adjustment are not calibrated. In this case, we need to apply a correction factor $c < 1$ to make the predictions more imprecise. Here, we used $c = 0.1$. Finally, for the Wine dataset (Figure 7d), the calibration curves corresponding to probabilistic predictions are already above the diagonal, which

[7]Results with the RVM algorithm are not reported for the kin8nm dataset, because of the excessive computing time on our machine (a 2019 16" MacBook Pro with a 2.4 GHz 8-core Intel i9 processor).

TABLE IV: Average and standard errors of RMS and NLL for ENNreg, probabilistic backpropagation (PBP), Monte Carlo dropout, deep ensembles, and deep evidential regression (DER) for six datasets.

	RMS					NLL				
	ENNreg	PBP	MC-dropout	Deep ens.	DER	ENNreg	PBP	MC-dropout	Deep ens.	DER
Boston	**2.87 ± 0.14**	**3.01 ± 0.18**	**2.97 ± 0.19**	**3.28 ± 1.00**	**3.06 ± 0.16**	2.53 ± 0.07	2.57 ± 0.09	**2.46 ± 0.06**	**2.41 ± 0.25**	**2.35 ± 0.06**
Energy	**1.06 ± 0.05**	1.80 ± 0.05	1.66 ± 0.04	2.09 ± 0.29	2.06 ± 0.10	**1.14 ± 0.07**	2.04 ± 0.02	1.99 ± 0.02	**1.38 ± 0.22**	1.39 ± 0.06
Concr.	**5.10 ± 0.12**	5.67 ± 0.09	**5.23 ± 0.12**	6.03 ± 0.58	5.85 ± 0.15	3.38 ± 0.13	3.16 ± 0.02	**3.04 ± 0.02**	**3.06 ± 0.18**	**3.01 ± 0.02**
Yacht	**0.44 ± 0.04**	1.02 ± 0.05	1.11 ± 0.09	1.58 ± 0.48	1.57 ± 0.56	**0.13 ± 0.12**	1.63 ± 0.02	1.55 ± 0.03	1.18 ± 0.21	1.03 ±0.19
Wine	**0.63 ± 0.01**	**0.64 ± 0.01**	**0.62 ± 0.01**	**0.64 ± 0.04**	**0.61 ± 0.02**	**0.94 ± 0.01**	0.97 ± 0.01	**0.93 ± 0.01**	**0.94 ± 0.12**	**0.89 ± 0.05**
kin8nm	**0.08 ± 0.00**	0.10 ± 0.00	0.10 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	**-1.19 ± 0.00**	-0.90 ± 0.01	-0.95 ± 0.01	**-1.20 ± 0.02**	-1.24 ± 0.01

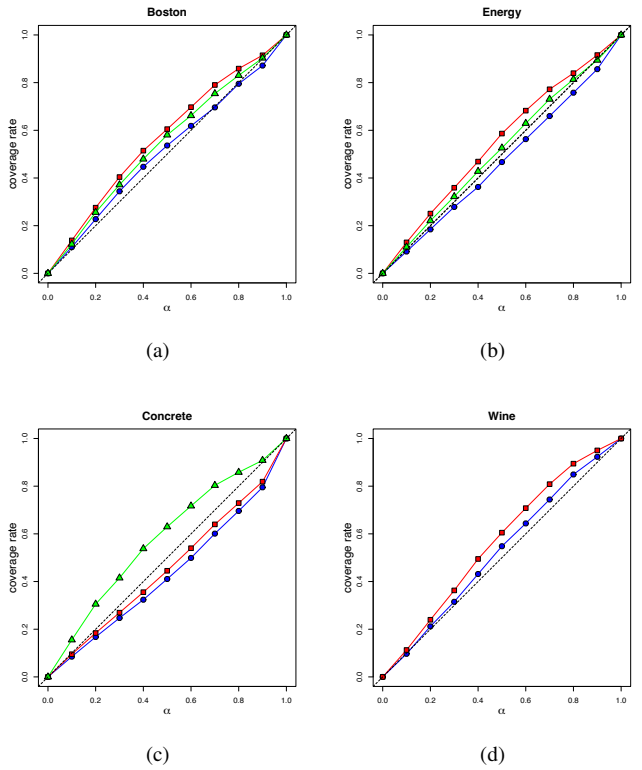


Fig. 7: Calibration plots for ENNreg and four datasets: probabilistic predictions (blue circles), raw evidential predictions (red squares) and adjusted evidential predictions (green triangles).

means that the probabilistic prediction intervals are already calibrated BPIs. In such a case, we can consider that the probabilistic predictions provide an adequate description of prediction uncertainty and neglect the precision output $h(\boldsymbol{x})$ or, equivalently, multiply it by $c = \infty$.

VI. Conclusions

We have introduced an evidential neural network model for regression, called ENNreg. In this model, distances to K prototypes are considered as pieces of evidence about the response variable and are described by GRFNs. The total evidence is then pooled by the generalized product-intersection rule, an extension of Dempster's rule in the epistemic random fuzzy set setting. The network architecture is composed of a first hidden layer of K RBF units and K linear units, a second hidden layer of $2K$ units, and an output layer of three units with cross-cut connections from the two hidden units. The network output is a GRFN described by three numbers: a point prediction $\mu(\boldsymbol{x})$, a conditional variance estimate $\sigma^2(\boldsymbol{x})$, and a precision parameter $h(\boldsymbol{x})$ whose value depends on the distances between the input vector and the prototypes, and which can be seen as describing the reliability of the probabilistic predictions. This additional degree of freedom makes it possible to quantify not only random uncertainty, but also epistemic uncertainty.

The network output can also be expressed as a predictive belief function on the real line induced by the output GRFN. We have defined a loss function for such outputs, extending the negative log-likelihood to evidential regression. We have also discussed the calibration of predictive belief functions for regression tasks and introduced a definition based on the coverage rates of belief prediction intervals (prediction intervals with specified belief degree), which are required to be conservative. Calibrated belief prediction intervals at degree α thus contain, on average, at least $100\alpha\%$ of future observations. This calibration property can be assessed visually by drawing calibration plots.

Comparative experiments have shown that ENNreg performs better in terms of RMS error than other RBF-based regression methods such as RBF networks, SVM, RVM and Gaussian processes, and that it is competitive with state-of-the art methods for regression such as random forests and multilayer perceptrons. Furthermore, a comparison with recent approaches to uncertainty quantification in deep networks shows that ENNreg is also competitive with these methods in terms of both prediction accuracy and probabilistic calibration, while offering the advantage of a richer quantification of uncertainty thanks to the random fuzzy set formalism.

This research can be extended in several directions. For structured inputs such as time series or images, additional feature-extraction layers could be inserted between the input and prototype layers, resulting in a deep architecture as described in [7] for image classification. For regression with multiple outputs, the ENNreg model could be extended to compute outputs in the form of Gaussian random fuzzy vectors as introduced in [23]. Finally, other families of random fuzzy numbers could be used to accommodate outputs with, e.g., skewed distribution or bounded support.

References

[1] J. M. Hernandez-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. of the 32nd*

Int. Conf. on Machine Learning, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 2015, pp. 1861–1869.

[2] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, USA: PMLR, 2016, pp. 1050–1059.

[3] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, I. G. et al., Ed., vol. 30. Curran Associates, Inc., 2017.

[4] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 927–14 937.

[5] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Annals of Mathematical Statistics*, vol. 38, pp. 325–339, 1967.

[6] G. Shafer, *A mathematical theory of evidence*. Princeton, N.J.: Princeton University Press, 1976.

[7] Z. Tong, P. Xu, and T. Denœux, "An evidential classifier based on Dempster-Shafer theory and deep learning," *Neurocomputing*, vol. 450, pp. 275–293, 2021.

[8] L. Huang, S. Ruan, P. Decazes, and T. Denœux, "Lymphoma segmentation from 3D PET-CT images using a deep evidential network," *International Journal of Approximate Reasoning*, vol. 149, pp. 39–60, 2022.

[9] T. Denœux, D. Dubois, and H. Prade, "Representations of uncertainty in artificial intelligence: Beyond probability and possibility," in *A Guided Tour of Artificial Intelligence Research*, P. Marquis, O. Papini, and H. Prade, Eds. Springer Verlag, 2020, vol. 1, ch. 4, pp. 119–150.

[10] T. Denoeux, S. Li, and S. Sriboonchitta, "Evaluating and comparing soft partitions: an approach based on Dempster-Shafer theory," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1231–1244, 2018.

[11] Z. Su and T. Denœux, "BPEC: Belief-peaks evidential clustering," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 1, pp. 111–123, 2019.

[12] T. Denœux, "A k-nearest neighbor classification rule based on Dempster-Shafer theory," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 25, no. 05, pp. 804–813, 1995.

[13] Z.-G. Liu, Y. Liu, J. Dezert, and F. Cuzzolin, "Evidence combination based on credal belief redistribution for pattern classification," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 4, pp. 618–631, 2020.

[14] Z. Liu, X. Zhang, J. Niu, and J. Dezert, "Combination of classifiers with different frames of discernment based on belief functions," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 7, pp. 1764–1774, 2021.

[15] T. Denœux, "A neural network classifier based on Dempster-Shafer theory," *IEEE Trans. on Systems, Man and Cybernetics A*, vol. 30, no. 2, pp. 131–150, 2000.

[16] A. P. Dempster, "Upper and lower probabilities generated by a random closed interval," *Annals of Mathematical Statistics*, vol. 39, no. 3, pp. 957–966, 1968.

[17] P. Smets, "Belief functions on real numbers," *International Journal of Approximate Reasoning*, vol. 40, no. 3, pp. 181–223, 2005.

[18] O. Kanjanatarakul, S. Sriboonchitta, and T. Denœux, "Prediction of future observations using belief functions: A likelihood-based approach," *International Journal of Approximate Reasoning*, vol. 72, pp. 71–94, 2016.

[19] T. Denœux, "Function approximation in the framework of evidence theory: A connectionist approach," in *Proceedings of the 1997 International Conference on Neural Networks (ICNN'97)*, vol. 1. Houston: IEEE, June 1997, pp. 199–203.

[20] S. Petit-Renaud and T. Denœux, "Nonparametric regression analysis of uncertain and imprecise data using belief functions," *International Journal of Approximate Reasoning*, vol. 35, no. 1, pp. 1–28, 2004.

[21] T. Denœux, "An evidential neural network model for regression based on random fuzzy numbers," in *Belief Functions: Theory and Applications*, S. Le Hégarat-Mascle, I. Bloch, and E. Aldea, Eds. Cham: Springer International Publishing, 2022, pp. 57–66.

[22] ——, "Belief functions induced by random fuzzy sets: A general framework for representing uncertain and fuzzy evidence," *Fuzzy Sets and Systems*, vol. 424, pp. 63–91, 2021.

[23] T. Denœux, "Reasoning with fuzzy and uncertain evidence using epistemic random fuzzy sets: General framework and practical models," *Fuzzy Sets and Systems*, vol. 453, pp. 1–36, 2023.

[24] H. T. Nguyen, "On random sets and belief functions," *Journal of Mathematical Analysis and Applications*, vol. 65, pp. 531–542, 1978.

[25] M. L. Puri and D. A. Ralescu, "Fuzzy random variables," *Journal of Mathematical Analysis and Applications*, vol. 114, no. 2, pp. 409–422, 1986.

[26] S. de la Rosa de Sáa, M. A. Lubiano, B. Sinova, M. A. Gil, and P. Filzmoser, "Location-free robust scale estimates for fuzzy data," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 6, pp. 1682–1694, 2021.

[27] J. C. Figueroa-Garca, C. A. Varn-Gaviria, and J. L. Barbosa-Fontecha, "Fuzzy random variable generation using α-cuts," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 3, pp. 539–548, 2021.

[28] I. Couso and L. Sánchez, "Upper and lower probabilities induced by a fuzzy random variable," *Fuzzy Sets and Systems*, vol. 165, no. 1, pp. 1–23, 2011.

[29] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[30] D. Dubois, H. Prade, and R. Yager, "Merging fuzzy information," in *Fuzzy sets in approximate reasoning and information systems*, J. C. Bezdek, D. Dubois, and H. Prade, Eds. Boston: Kluwer Academic Publishers, 1999, pp. 335–401.

[31] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9. MIT Press, 1996, pp. 155–161.

[32] T. Denœux and S. Li, "Frequency-calibrated belief functions: Review and new insights," *International Journal of Approximate Reasoning*, vol. 92, pp. 232–254, 2018.

[33] L. Cella and R. Martin, "Valid inferential models for prediction in supervised learning problems," *International Journal of Approximate Reasoning*, vol. 150, pp. 1–18, 2022.

[34] T. Denœux, *evreg: Evidential Regression*, 2023, R package version 1.0.1. [Online]. Available: https://CRAN.R-project.org/package=evreg

[35] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[36] M. Musavi, W. Ahmed, K. Chan, K. Faris, and D. Hummels, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, no. 4, pp. 595–603, 1992.

[37] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988.

[38] R. R. Yager and D. P. Filev, *Essentials of fuzzy modeling and control*. New-York: John Wiley and Sons, 1994.

[39] H. Zhang, X. Zhao, L. Zhang, B. Niu, G. Zong, and N. Xu, "Observer-based adaptive fuzzy hierarchical sliding mode control of uncertain under-actuated switched nonlinear systems with input quantization," *International Journal of Robust and Nonlinear Control*, vol. 32, no. 14, pp. 8163–8185, 2022.

[40] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002.

[41] M. Kuhn, *caret: Classification and Regression Training*, 2021, R package version 6.0-90, https://CRAN.R-project.org/package=caret. [Online]. Available: https://CRAN.R-project.org/package=caret

[42] F. M. Silva and L. B. Almeida, "Speeding up backpropagation," in *Advanced neural computers*, R. Eckmiller, Ed. New-York: Elsevier-North-Holland, 1990, pp. 151–158.

Thierry Denœux Thierry Denœux is a Full Professor (Exceptional Class) with the Department of Information Processing Engineering at the University of Compiègne, France, and a senior member of the French Academic Institute (*Institut Universitaire de France*). His research interests concern reasoning and decision-making under uncertainty and, more generally, the management of uncertainty in intelligent systems. His main contributions are in the theory of belief functions with applications to statistical inference, pattern recognition, machine learning and information fusion. He has published more than 300 papers in this area. He is the Editor-in-Chief of the *International Journal of Approximate Reasoning*, and an Associate Editor of several journals including *Fuzzy Sets and Systems* and *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*.