Neural Networks and Belief Logic

Yuan Yan Chen and Joseph J. Chen
PNN Technologies
yan_chen@pnntech.com, josephchen@pnntech.com

Abstract

Many researchers have observed that neurons process information in an imprecise manner - if a logical inference emerges from neural computation, it is inexact at best. Thus, there must be a profound relationship between belief logic and neural networks. In Chen (2002), a plausible neural network model that can compute probabilistic and possibilistic logic was proposed. In this article we further extend this model to continuous variables for function and relation estimation. We discuss why and how belief logic is derived from neural computation.

1. Introduction

Neural processing is often aimed at detecting differences in action potential rather than absolute values (e.g., Hopfield, 1995). For example, neural processing detects contrast rather than pure luminance, edges rather than areas, and so on. In evidential reasoning the difference in action potential means the weight of evidence favors the hypothesis, which in turn can be transformed into the belief (necessity) of the possibility measure, and default into Boolean logic. The competitive nature of neuron activities induces the belief judgment.

2. Plausible Neural Network

A plausible neural network (PNN) model that can compute probabilistic and possibilistic inference for binary variables was introduced in Chen (2002). In this article we summarized the extension of PNN model to continuous variables, a more detail description of this model can be found in Chen and Chen (2003).

The weight of connection of PNN between any two neurons is given by the *mutual information content*

$$\omega_{12} = \ln (P(X, Y) / P(X)\, P(Y)), \qquad (1)$$

where X and Y are continuous variables in [0,1], which represent the state of the neurons. The maximum likelihood estimate of ω_{12} is given by

$$\hat{\omega}_{12} = \log (n\Sigma_i\, x_i\, y_i / \Sigma_i\, x_i\, \Sigma_i\, y_i), \qquad (2)$$

where x_i and y_i are the history of the neuron states.

The activation function of PNN is based on the winner-take-all (WTA) function (e.g. Maass (2000)). By normalization, the firing pattern of the neuron ensemble can be interpreted as the α-cut operation of the fuzzy set. The activation of the neuron ensemble is given as follows:

$$y_j = s (\Sigma_i\, \omega_{ij}\, x_i),\ \forall\ j\ ,\ e^{\beta\Sigma i\, \omega ij\, xi} / \sup_j e^{\beta\Sigma i\, \omega ij\, xi} > \alpha$$
$$y_j = 0, \text{ otherwise}, \qquad (3)$$

where $s (t_j) = e^{\beta tj} / \sum_j e^{\beta tj}$, which is usually referred to as the softmax function.

Unsupervised learning induces factorial encoding (e.g., Barlow, 1989). Thus, in higher levels of neural processing, if y_1 and y_2 are two competitive hypothesis, which receive the input from $x_1, x_2, \ldots x_n$, their action potentials are

$$\Sigma_i\, \omega_{ij}\, x_i = \Sigma_i \ln(p(x_i \mid y_j)) - \Sigma_i \ln (p(x_i)) \qquad (4)$$

By taking the difference of their action potentials, the second term of (4) is canceled, with the assumption of independence we have

$$\ln ((p(x_1, x_2, \ldots x_n \mid y_1)/ (p(x_1, x_2, \ldots x_n \mid y_2)) \quad (5)$$

The *log likelihood ratio* is often referred to as the weight of evidence (e.g., Good, 1950, Chen, 1995). Note that (5) does not have the biased term $\ln (p(y_1)/p(y_2))$, as occurs in the Bayesian inference. If the weight of evidence for y_1 is much larger than y_2, by (3) the weight of evidence for y_2 has been eliminated by the threshold cut and we have $p(y_1 \mid x_1, x_2, \ldots x_n) =$

Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)
0-7695-2291-2/04 $ 20.00 IEEE

Bel ($y_1 \mid x_1, x_2, \ldots x_n$) = 1, where Bel is a belief measure (e.g., Chen (1995)); thus the network computes as Boolean logic.

In PNN coding, a competitive neuron ensemble corresponds to a variable in statistical inference. If the variable is continuous, data is encoded as complementary fuzzy sets; this process is known as fuzzification.

The learning algorithm of PNN with hidden neurons is intrinsically an E-M algorithm, and given as follows

1. Fire the hidden neurons randomly.
2. M- step: estimate the weight connections of input neurons with hidden neurons.
3. E - step: compute the action potentials of hidden neurons and normalize into [0,1]. If the activation level of a neuron is larger than threshold, α, then it fires.
4. Update the synaptic weight if the firing of the hidden neuron changes.
5. Repeat the procedure until the network stabilizes.

When the predicted variables are continuous in supervised learning, PNN inference can perform both function and relation estimation. Function estimation of the PNN algorithm is demonstrated by analyzing the data of a Henon map. The data is trained with the time series and a first order time lag. For predictions we input the lagged time series in the trained network, and compute the fuzzy centroid of the output values. This process is referred to as defuzzification. Figure 1 shows the experimental result of the PNN algorithm. The solid line are the predictions, the dotted lines are the actual time series.
It predicts correlated multiple time series simultaneously.

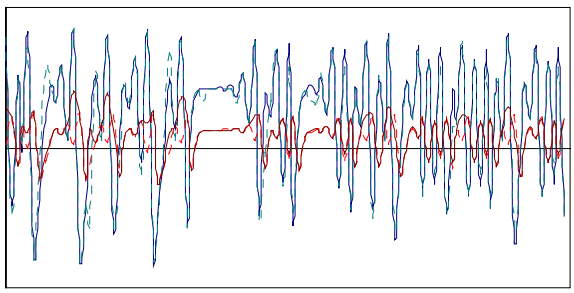

Figure 1 Experimental results of PNN algorithm for estimating Henon map time series

3. Conclusion

In the physics of computational systems, if there is no energy potential difference between the communication channels, there is no diffusion process (e.g., Mead, 1989); and no signal can be detected. Thus, the contrast computation is used extensively in neural network systems. We propose that the contrast of evidence potentials of rival hypotheses compute belief judgment. Although the probability and possibility measures are transferable in the cognitive process, the contrast computation provides the model/hypothesis selection before the inference can be normalized into the uncertainty measure.

Reference

[1] Barlow, H. B. (1989). Unsupervised learning, *Neural Computation*, **1**, 295-311.

[2] Chen, Y. Y. (1995). Statistical inference based on the possibility and belief measures. *Trans. Amer. Math. Soc*. **347**, 1855-1863.

[3] Chen, Y. Y. (2002). Plausible neural networks. *Advance in Neural Networks World.* Ed. Grmela, A. and Mastorakis, N. E. WSEAS Press, 180-185.

[4] Chen, Y. Y. and Chen, J. J. (2003). "Plausible Neural Network with Supervised and Unsupervised Cluster Analysis", US patent 0140020-A1.

[5] Good I. J. (1950), *Probability and the weighting of evidence.* Griffin, London.

[6] Hopfield, J. J. (1995). Pattern recognition computing using action potential timing for stimulus representation. *Nature*, **376**, 33-36.

[7] Maass, W. (2000). On the computational power with winner-take-all, *Neural Computation*, **12(11)**, 2519-2536.

[8] Mead, C. (1989). *Analog VLSI and Neural Systems.* Addison-Wesley.