

Plausible Neural Networks

YUAN YAN CHEN
Center for Army Analysis
6001 Goethals Road
Fort Belvoir, VA 22060-5230
USA

Abstract – Most literature that attempts to explain the neural network computation and the uncertainty of inference based on the probability measure; e.g. see [2], [10], [11], [12] and [13]. This paper discusses a new neural network model, that uses a statistical inference model proposed in [5] and [6]; under such a model the interpretation of neural networking is both possibilistic and probabilistic in nature, and it is referred to as PLANN.

Key-Words: possibility measure, mutual information, likelihood function, neural networks, E-M algorithm, unsupervised learning, supervised learning.

1 Introduction

In human reasoning, there are two modes of thinking. One is expectation and the other is likelihood. Expectation is for planing or predicting the true state of the future, likelihood is for judging the truth of a current state. These two modes of thinking interact with each other; for example we need to recognize our environment in order to make a prediction. A statistical inference model that has the interaction of these two modes of thinking, which is a hybrid of probability and possibility measures, was discussed in [5]. The inference is based on *the principle of inverse inference*, which can be stated as follows

Given the evidence, the more probable a hypothesis can produce such evidence the more likely it to be true.

The statistical inference based on possibility measure for parameter estimation and hypothesis testing has several interesting properties. Mathematically it is parallel to Bayesian inference, but in application it is close to frequentist inference. In the parametric case it is equivalent to the likelihood ratio test, and in the nonparametric case it is equivalent to empirical likelihood inference [14].

In machine learning the consideration of model is often necessary, since the inverse problem is ill posed without the model selection. The relationship of prior, data, and model under Bayesian inference is

$$\begin{aligned} P(model | data, prior) &= P(data | model, prior) \\ P(model | prior) / \sum_{model} P(data | model, prior) \\ P(model | prior). \end{aligned}$$

If we use the possibility measure, there is a universal uninformative prior, and the computation can be simplified as

$$Pos(model | data) = P(data | model) / \sup_{model} P(data | model).$$

The goal of machine learning is judging which model is more likely to be true. This can be achieved by fuzzy ranking of the possibility measure. The additive property of probability measure is unnecessary.

The relationships between statistical inferences and neural networks in machine learning and pattern recognition have attracted a lot of research attention. Previous connections were discussed in terms of the Bayesian inference, e.g. [11], [12] and [13]. Bayesian neural networks require the assignment of prior belief on the weight distributions. Unfortunately, this makes the

computation of large-scale networks almost impossible.

2 PLANN Model and Learning Algorithm

For each variable X there are two distinct meanings. One is $P(X)$, which considers the population distribution of X , and the other is $\Pr(X)$, which is a random sample based on the population. If the population $P(X)$ is unknown, it can be considered as a fuzzy variable or a fuzzy function, which is referred to as stationary variable or stationary process in [5]. Based on sample statistics we can have a likelihood estimate of $P(X)$. The advantage of using the possibility measure on a population is that it has a universal vacuous prior, thus the prior does not need to be considered as it is in the Bayesian inference.

The statistical inference model of discussed in [5] is given as follows

$$l(\theta | x) = p(x | \theta) / \sup_{\theta} p(x | \theta) \quad (1)$$

where $l(\theta | x)$ is a likelihood function or a possibility function. The likelihood function can be estimated from the data, and the estimates can be used for prediction. If we alternate the two procedures, we have the E-M algorithm, which is used extensively for machine learning. Thus, this inference model provides another theoretical justification for the E-M algorithm.

Let be X be a neuron, which is a binary variable. At any given time t , $X_t = 1$ is when the neuron fires, and $X_t = 0$ is when the neuron is at rest. For simplicity we drop the subscript t . The weight connection between neuron X and neuron Y is given as follows

$$\omega_{12} = \log(P(X=1, Y=1)/P(X=1)P(Y=1)) \quad (2)$$

which contain the firing history or *mutual information content* of two neurons. Equation (1) has a Hebbian-type interpretation, the synapse weight increase in strength by coincidence of presynaptic and postsynaptic signals.

Linking the neuron's synapse weight to information theory has several advantages. The explanation of knowledge and synapse weight is transparent. Information and energy are exchangeable. And neuron learning becomes statistical inference. From a statistical inference point of view, neuron activity for a pair of connected neurons is Bernoulli's trial for two dependent random variables. Bernoulli trials of a single random variable are discussed in [5].

Let (X, Y) be bivariate Bernoulli random variable with parameters $\theta_1, \theta_2, \theta_{12}$, where $\theta_1 = P(X=1)$, $\theta_2 = P(Y=1)$ and $\theta_{12} = P(X=1, Y=1)$. The joint likelihood function of the parameters is

$$l(\theta_1, \theta_2, \theta_{12} | x, y) = \theta_{12}^{xy} (\theta_1 - \theta_{12})^{x(1-y)} (\theta_2 - \theta_{12})^{(1-x)y} (1 - \theta_1 - \theta_2 + \theta_{12})^{(1-x)(1-y)} / \sup_{\theta_1, \theta_2, \theta_{12}} \theta_{12}^{xy} (\theta_1 - \theta_{12})^{x(1-y)} (\theta_2 - \theta_{12})^{(1-x)y} (1 - \theta_1 - \theta_2 + \theta_{12})^{(1-x)(1-y)} \quad (3)$$

Let $g(\theta_1, \theta_2, \theta_{12}) = \log(\theta_{12} / \theta_1 \theta_2)$. The likelihood function of ω_{12} given data x, y is

$$l(\omega_{12} | x, y) = \sup_{\theta_1, \theta_2, \theta_{12} : \omega_{12} = g(\theta_1, \theta_2, \theta_{12})} l(\theta_1, \theta_2, \theta_{12} | x, y) \quad (4)$$

This is based on the extension principle of the fuzzy set theory (e.g. [8]). When a synapses with a memory of $\mathbf{x}, \mathbf{y}$ receives a new information x_t, y_t , the weight is updated by the likelihood rule [5],

$$l(\omega_{12} | \mathbf{x}, \mathbf{y}, x_t, y_t) = l(\omega_{12} | \mathbf{x}, \mathbf{y}) l(\omega_{12} | x_t, y_t) / \sup_{\omega_{12}} l(\omega_{12} | \mathbf{x}, \mathbf{y}) l(\omega_{12} | x_t, y_t). \quad (5)$$

The objective of learning is to find the optimum parameter that maximized the log likelihood function, which is the same as the natural gradient descent learning, e.g. [15], however the computation is different.

A confidence measure for ω_{12} is represented by the α -cut set or $1-\alpha$ likelihood interval [6]. This is needed only if the size of the training sample is small. If the sample size is large enough the maximum

likelihood estimate of ω_{12} will be sufficient, which can be computed from the maximum likelihood estimate of θ_1 , θ_2 and θ_{12} . Since $\hat{\theta}_1 = \sum_i x_i/n$, $\hat{\theta}_2 = \sum_i y_i/n$, $\hat{\theta}_{12} = \sum_i x_i y_i/n$, we have

$$\hat{\omega}_{12} = \log (n \sum_i x_i y_i / \sum_i x_i \sum_i y_i), \quad (6)$$

In [5] the parameter is considered a stationary variable. If the input patterns change with time, the weight of the network is nonstationary, a dynamical system will emerge.

In the Bernoulli trial, the knowledge of the experiment is stored in the parameter, the more data information the crispier the likelihood function. Similarly the knowledge/memory of two neurons are stored in the synapse weight. If the synapse weight between two neurons increases then the entropy decreases. This is the principle of energy and information exchange.

The learning rule based on mutual information is consistent with the Hopfield learning rule, which is discussed in [11]. However, there is no weight connection between two neurons if they fire independently; and the less frequently a neuron fires the higher the learning rate.

Let X_i be the neurons that fire to X_j , The activation is given by integration and fire model

$$X_j = s(\sum_i \omega_{ij} x_i), \quad (7)$$

where s is a signal function.

A plausible neural network (PLANN) is a network with the weight connection given by (2) and activation function given by (7). Having symmetric weight connections ensures a stable state of the network.

If $Pl(X_j=1 | \mathbf{x}, \boldsymbol{\omega}) = s(\sum_i \omega_{ij} x_i)$ is a plausibility function, which can be a probability or possibility function depending on the normalization, based on inference discussed in [5], we have

$$Pl(X_j=1 | \mathbf{x}) = \sup_{\boldsymbol{\omega}} Pl(X_j=1 | \mathbf{x}, \boldsymbol{\omega}) / (\boldsymbol{\omega} | \mathbf{x}) = s(\sum_i \hat{\omega}_{ij} x_i) \quad (8)$$

Thus, it justifies the maximum likelihood estimation for weight parameter, while under Bayesian network maximum aposterior belief of weight parameter is simply an approximation.

3 PLANN Architecture and Inference

PLANN is a recurrent network, technically it can have full interconnections as the design of Boltzmann machine [1]. However, a layer network is more efficient in energy conservation, which is favored by nature in organization.

A classification model FASE based on the possibility measure is discussed in [7]. FASE is a single layer neural network, with each attribute neurons connected to class neurons. The attribute neuron statistically independent of class neuron has no weight connection; it does not contribute any evidence. The class neurons receive more information from the attribute neurons are more likely to fire, this explains *the principle of inverse inference*. The FASE model handles the dependency of attribute information by the t-norm operation. The classification based on possibility measure has another advantage that it is not restricted to mutually exclusive categories. The class can be overlapping or in hierarchy order; for example a document can be labeled as computer program and C++ simultaneously in a training sample.

FASE model is mathematically attractive, but the selection of t-norm is still a question. An alternative approach is employing hidden neurons to perform competitive learning, to circumvent the difficulty of conditional dependence.

Unsupervised learning is designed as a network of input neurons connected to hidden competitive neurons, which can have multiple levels to provide hierarchy clustering. Although the classification is usually considered as supervised learning, we use the same design as unsupervised learning in the PLANN network. The input

pattern contains class labels in addition to input pattern; if some class labels are missing then it is semi-supervised learning. The hidden layer combines the information of class and attribute neurons. The posterior belief of the class variable can be estimated from the network. The information of attribute neurons feed forward to the competitive hidden neurons, and the winning neurons feed back to the class neurons. This is similar to the wake-sleep algorithm of [9], where each hidden neuron represents a cluster in the generative model.

Let Y_j be the competitive hidden neuron connected to the input neurons X_i . From equation (2) the active potential of the hidden neurons y_j received from input pattern $x_1, x_2, \dots, x_n$ is

$$\sum_i \omega_{ij} x_i = \sum_i \ln(p(x_i | y_j)) - \sum_i \ln(p(x_i)). \quad (9)$$

The second term of (9) can be removed by either probability or possibility normalization. However, the possibility normalization provides the advantage of being less sensitive to the number of hidden neurons. If the normalized action potential of a neuron is larger than threshold, i.e.

$$s(\sum_i \omega_{ij} x_i) > 1 - \alpha, \quad (10)$$

then it fires, where $s(t_j) = \exp(t_j) / \sum_j \exp(t_j)$.

The threshold can be considered as a confidence level that an input pattern matches with the stored pattern of latent variable or hypothesis. This is similar to the vigilance parameter in ART network [4]. After training the cluster will form, since it is the stable state of the competitive network. The stable state is also similar to the resonance of ART.

4 Experiments of PLANN

In the simulation of PLANN, we use the learning algorithm based on the maximum likelihood estimation given in (6) and additive activation function. The uncertainty measure of the weights is not considered.

If the variable is discrete with k categories, it can be encoded by $\mathbf{X} = (X_1,$

$X_2, \dots, X_k)$. Each neuron is an *indicator function* of a particular data value. A null vector represents a missing data value. If the variable is continuous, based on the inference discussed in [6], there is a universal nonparametric functional estimate; however the computation of likelihood function is too intensive. For simulation we let each neuron $X_1, X_2, \dots, X_k$ be sensitive to a range of values, that correspond to overlapping bins or kernel function (e.g. radial basis function) units. When a data x is observed, several neurons sensitive to the value will fire. Experimental results show that the synaptic weights, trained with the learning rule, form a *Mexican hat* function as in the feature map, if the inhibitory weights at a distance were disregarded.

The unsupervised learning algorithm is 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 active potentials of hidden neurons and normalize into $[0,1]$. If the action potential of a neuron is larger than threshold, $1 - \alpha$, then it fires.
4. Update the synaptic weight if the firing of the hidden neuron changes.
5. Repeat the procedure until the network stabilizes.

The M-step is executed locally, with the synaptic weight being updated only when the activity of the neuron changes; thus, it is faster than many other algorithms. This learning algorithm is also similar to that of the Boltzmann machine [1], where the visible neurons are clamped, and the hidden neurons run freely until stabilized. The number of hidden neurons in the network is flexible as long as it is sufficient. Some of them lose the competition and have no weight; some of them represent the same pattern.

The PLANN algorithm has been tested on the datasets of UCI machine learning repository [3] for supervised and unsupervised learning. It can extract multi-

dimensional pattern such as tic-tac-toe and led data. The weights of the trained network can be easily transformed into uncertainty measure. We demonstrate this feature by using the zoo dataset.

Table 1 shows the network output of the PLANN unsupervised learning algorithm for the zoo data (the class type in the data is not used). It is trained with fifteen hidden neurons. After training ten clusters are formed. Table 1(a) shows the output probabilities, which are transformed from the weights of the network. Simple inference

rules can be derived from the table. Table 1(b) gives the list of animals in the clusters. We see that the networks find the distinctive patterns, regardless of the training sample size. The normalized action potential after training can provide the possibility measure of the hidden neurons, which can be interpreted as the fuzzy membership of data input belonging to a cluster. They are summarized in table 1(c), for example mink belongs to cluster C_5 , but it also belongs to cluster C_3 with a fuzzy membership of .96.

Table 1. Experimental results of PLANN algorithm for zoo database

(a) Conditional probability of attributes given the cluster

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	no legs	2 legs	4 legs	5 or more	tail	domestic	catsize
C_1	1	0	0	1	0.4	0	0	1	1	1	0	0	0	1	0	0	0	0.2	0.6
C_2	0	0	1	0	0	1	0.69	1	1	0	0.08	1	1	0	0	0	1	0	0
C_3	1	0	0	1	0	0	0	1	1	1	0	0	0	0	1	0	0.9	0.23	0.77
C_4	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	1	0	1
C_5	0.67	0	0	1	0	1	1	0.83	1	1	0	0.67	0.5	0	0	0	0.83	0	1
C_6	0	1	1	0	0.8	0	0	0	1	1	0	0	0	1	0	0	1	0.15	0
C_7	0	0	1	0	0	0.86	1	0	0	0	0.14	0	0.29	0	0	0.57	0	0	0
C_8	0	0	1	0	0	0.75	0.75	1	1	1	0	0	0	0	1	0	0	0	0
C_9	0	0	0.67	0	0	0	1	1	1	0	0.67	0	1	0	0	0	1	0	0
C_{10}	0	0	0.91	0	0.55	0	0	0	0	1	0.27	0	0	0	0	0.82	0	0	0

(b) List of the animals in the clusters

C_1	fruitbat	girl	gorilla	vampire	wallaby														
C_2	bass	carp	catfish	chub	dogfish	haddock	herring	pike	piranha	seahorse	sole	stingray	tuna						
C_3	aardvark	antelope	bear	boar	buffalo	calf	cavy	cheetah	deer	elephant	giraffe	goat	hamster						
	hare	leopard	lion	lynx	mole	mongoose	opossum	oryx	polecat	pony	puma	pussycat	raccoon						
	reindeer	squirrel	vole	wolf															
C_4	tortoise																		
C_5	dolphin	mink	platypus	porpoise	seal	sealion													
C_6	chicken	crow	dove	duck	flamingo	gull	hawk	kiwi	lark	ostrich	parakeet	penguin	pheasant						
	rhea	skimmer	skua	sparrow	swan	vulture	wren												
C_7	clam	crab	crayfish	lobster	octopus	seawasp	starfish												
C_8	frog	newt	toad	tuatara															
C_9	pitviper	seasnake	slowworm																
C_{10}	flea	gnat	honeybee	housefly	ladybug	moth	scorpion	slug	termite	wasp	worm								

(c) Fuzzy membership of the animals belonging to the clusters*

C_3	mink (.96)		
C_4	platypus (.83)		
C_7	slug (.83)	worm (.83)	scorpion (.76)

*possibility or fuzzy membership ($>.75$).

5 Discussion

Our central hypothesis is that the parameter or synaptic weight of neural networks can be estimated by using the possibility measure. The advantage of such a model is that there is no prior assumption for the weight as in the Bayesian inference, and the E-M algorithm can be computed efficiently without gradient descent approximation. A hardware device can be designed to implement the weight update based on equation (5), and a very fast online learning neural network can be created.

From the neural computational perspective, categories are formed through the competition of neuron activities. Mathematically this is a relative measure; thus it is more straightforward to link the inference of classification to the possibility measure than to the probability measure. The connection of statistical dependency and weight configuration in a neural network is also very important. It gives a clearer interpretation of how the networks perform logical and plausible inference, and how the networks distinguish pattern and noise.

References:

- [1] Ackley, D. H., Hinton, G.E, and T. J. Sejnowski (1985). A learning algorithm for Boltzmann machine, *Cognitive Sci.* **9** (1985) 147-169.
- [2] Baum, E.B. and Wilczek F. (1988), Supervised learning of probability distributions by neural networks. In *Neural Information processing Systems*, ed. Anderson, D. Z. 52-61. New York: American Institute of Physics.
- [3] Blake, C. L. and Merz, C. J. (1998). UCI Repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA.[<http://www.ics.uci.edu/~mllearn/MLRepository.html>].
- [4] Carpenter, G. and Grossberg, S. (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *IEEE Computer*, **21** (3), 77-88.
- [5] Chen, Y.Y (1993). Bernoulli trials: from a fuzzy measure point of view. *J. Math. Anal. Appl.* **175**, 392-404.
- [6] Chen, Y. Y. (1995). Statistical inference based on the possibility and belief measures. *Trans. Amer. Math. Soc.* **347**, 1855-1863.
- [7] Chen, Y. Y. (2000). Fuzzy analysis of statistical inference. *IEEE Trans. Fuzzy Systems*. **8**, 796-799.
- [8] Dubois D. and Prade H. (1980) *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, London.
- [9] Hinton, G. E., Dayton, P., Frey, B. J. and Neal, R. M. (1995) The wake-sleep algorithm for unsupervised neural networks, *Science*, **268** 1158-1161.
- [10] Hopfield, J. J. (1987). Learning algorithm and probability distributions in feed-forward and feed-back networks. *Proceeding of the National Academy of Science*, USA 8429-8433.
- [11] Kononenko I. (1989) Bayesian neural networks. *Biological Cybernetics* **61**:361-370.
- [12] Lansner A. and Ekeberg O. (1989). A one-layer feedback, artificial neural network with a Bayesian Learning rule. *Int. J. Neural Systems* **1** 77-87.
- [13] MacKay D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation* **4**, 448-472.
- [14] Owen A. B. (2001). *Empirical Likelihood*. Chapman & Hall CRC Press.
- [15] Park H., Amari S., Fukumizu, K. (2000). Adaptive natural gradient learning algorithms for various stochastic models. *Neural Network* **13** 755-764.