ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Similarity networks for heterogeneous data

Lluís A. Belanche[1] and Jerónimo Hernández[2]

1- Faculty of Computer Science - Dept. of Software
Technical University of Catalonia, Barcelona, Spain

2- Intelligent Systems Group - Dept. of Computer Science and Artificial Intelligence
University of the Basque Country, Donostia, Spain

Abstract. A two-layer neural network is developed in which the neuron model computes a user-defined *similarity function* between inputs and weights. The neuron model is formed by the composition of an adapted logistic function with the mean of the partial input-weight similarities. The model is capable of dealing directly with variables of potentially different nature (continuous, ordinal, categorical); there is also provision for missing values. The network is trained using a fast two-stage procedure and involves the setting of only one parameter. In our experiments, the network achieves slightly superior performance on a set of challenging problems with respect to both RBF nets and RBF-kernel SVMs.

1 Introduction

A shortcoming of the existent neural network models is the difficulty of adding prior knowledge to the model in a principled way. In this sense, the integration of heterogeneous data sources in information processing systems has been advocated elsewhere [1]. Under the conceptual cover of *similarity measures*, we develop a class of neurons that accept heterogeneous inputs and weights and compute a user-defined similarity function between these inputs and weights. The neuron transfer function is the composition of a parameterized sigmoid-like function adapted to the $[0, 1]$ interval taking the averaged vector of partial input-weight similarities as argument. The basic idea is that a combination of similarity functions, comparing variables independently, is more capable of catching better the singularity of an heterogeneous dataset than other methods which require *a priori* data transformations. The resulting neuron model then accepts mixtures of continuous and discrete quantities, with explicit provision for missing information. Other data types are possible by extension of the model. An appealing advantage is found in the enhanced interpretability of the learned weights, so often neglected in the neural network community.

2 Similarity measures

A *similarity measure* s is a symmetric function expressing how "like" two observations are. We start by developing specific similarity measures for different types of variables, defined in the common codomain $I_s = [0, 1]$. We use s_{ijk} to mean $s_k(x_{ik}, x_{jk})$, the similarity of observations $\mathbf{x}_i, \mathbf{x}_j$ according to variable k.
Nominal variables. It is assumed that no partial order exists and the only possible comparison is equality. The basic measure for these variables is the

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

overlap. Let x_{ik}, x_{jk} be the modalities taken by two examples $\mathbf{x}_i, \mathbf{x}_j$, then $s_{ijk} = 1$ if $x_{ik} = x_{jk}$ and 0 otherwise. We introduce in this paper a frequency-based approach that goes beyond this simple equal/not-equal scheme:

$$s_{ijk} = \begin{cases} 0 & \text{if } x_{ik} \neq x_{jk} \\ 1 - P_{ik} & \text{if } x_{ik} = x_{jk} \end{cases} \tag{1}$$

where P_{lk} is the *fraction* of examples in a sample that take the value x_{lk} for variable k (ideally, one could use the *probability* of this event, if this knowledge is available). Therefore, if the values are different, there is not similarity. If they happen to be equal, then the similarity is inversely proportional to their probability. For instance, if two patients are being compared on their current illness, both having a rare illness makes them more similar than both having a very common one. Other ways of inverting the probability are possible. In the absence of further knowledge, the linear one is the simplest choice.

Ordinal variables. These variables form a linearly ordered space $(\mathcal{O}, \preceq)$ and can be seen as a bridge between categorical and continuous variables. Let $x_{ik}, x_{jk} \in \mathcal{O}$, such that $x_{ik} \preceq x_{jk}$, and P_{lk} be defined as above. Define [2],

$$s_{ijk} = \frac{2\log(P_{ik} + \ldots + P_{jk})}{\log P_{ik} + \log P_{jk}} \tag{2}$$

The summation runs through all values x_{lk} such that $x_{ik} \preceq x_{lk}$ and $x_{lk} \preceq x_{jk}$.

Continuous variables. Let $x_{ik}, x_{jk} \in A = [r^-, r^+] \subset \mathbb{R}, r^+ > r^-$. The standard metric in $\mathbb{R}$ is a metric in A. Therefore, for any two values $x_{ik}, x_{jk} \in A$:

$$s_{ijk} = \hat{s}\left(\frac{|x_{ik} - x_{jk}|}{r^+ - r^-}\right) \tag{3}$$

where $\hat{s} : [0,1] \to [0,1]$ is a decreasing continuous function. A very simple family is $\hat{s}(z) = (1 - z^\beta)^\alpha, 0 < \beta \leq 1, \alpha \geq 1$ (we take the choice $\alpha = \beta = 1$).

2.1 Normalized aggregation of similarities

When we aggregate (e.g. by averaging) the partial similarities we are assuming that all of them have the same importance. However, each partial similarity covers its codomain $[0,1]$ in a different way. Similarities that accumulate on the upper half of the interval have more influence in the overall value, because they do a more important contribution to the aggregation. We argue that this bias should be corrected so that all the partial similarities have a common baseline.

Let $s_{..k}$ be the mean similarity among all examples in the analyzed data sample, according to variable k only. We first rescale all the similarities as $\hat{s}_{ijk} = \frac{s_{ijk}}{s_{..k}}$. Then a normalization function $n : (0, +\infty) \to (0,1)$ is applied:

$$n(x) = \frac{x^a}{x^a + 1} \tag{4}$$

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

where a conveniently controls the shape of the function. When a similarity computation is needed, it is calculated as $n(\hat{s}_{ijk})$ instead of s_{ijk}. The similarity between two elements x_{ik}, x_{jk} is now computed as:

$$s_{ijk} = \begin{cases} n\left(\frac{s(x_{ik},x_{jk})}{s_{..k}}\right) & \text{if neither of } x_{ik}, x_{jk} \text{ are missing} \\ \frac{1}{2} & \text{otherwise} \end{cases} \quad (5)$$

This is so because, when $s_{..k}$ is used to replace the missing *similarity* value, we have $n(\frac{s(x_{ik},x_{jk})}{s_{..k}}) = n(\frac{s_{..k}}{s_{..k}}) = \frac{1}{2}$ (this holds regardless of the value of a).

The method looks very naive and indeed it is; on the other hand, it is very intuitive and computationally simple. It does not require the estimation of the missing information (a delicate and risky undertaking), only the estimation of the *overall* similarity between two observations, in a situation in which some of the partial similarities could not be computed. We argue that this task is easier and, after all, is what we really are interested in: the similarity value.

2.2 Similarity-based data clustering

In a clustering task the examples are grouped attending to some similarity measure. The LEADER algorithm is a simple and attractive unsupervised clustering method [3]. In essence, the algorithm processes the examples of the dataset taking one at a time and evaluating if it can belong to any cluster already created. If it cannot, a new cluster will be created using this new example as *leader*.

We have developed a new version of the algorithm that represents an improvement in two ways. First, the algorithm now works using general similarities instead of metric distance functions. Second, given $s_0 \in I_s$, the method is guaranteed to fulfill a number of interesting properties:

1. For any example, the similarity with its leader is *at least* s_0.
2. The similarity between any two leaders is *lower* than s_0.
3. If two examples are repeated in the dataset, they will have the *same* leader.
4. For any example, the similarity with its leader is *higher* than that with any other leader.

In summary, the algorithm needs the specification of one parameter ($s_0 \in I_s$) and the returned leaders are a subset of the data set (thus there is no problem in delivering "impossible centroids" as many algorithms do). The number of clusters cannot be estimated beforehand, but it is possible to establish a relationship with the s_0 parameter, a higher s_0 implying a larger number of clusters.

3 Similarity neural networks

Consider $s : X^n \times X^n \to I_s$ a similarity function in $X^n = X^{(1)} \times \ldots \times X^{(n)}$, the cartesian product of a number n of *source sets*. This function is formed by the

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

aggregation of n partial similarities $s_k : X^{(k)} \times X^{(k)} \to I_s$, each $X^{(k)}$ being the domain of the predictive variable k, and where $I_{s_k} = [0, 1]$, for coherence.

The s_k are computed with the similarity measures for the different variable types (although this alone does not necessarily determine the right measure). A *neuron model* can be devised that is both similarity-based and handles data heterogeneity and missing values. Let $\phi_i(\mathbf{x}), \mathbf{x} \in X^n$ denote the function computed by the i-th neuron, using the weight vector $\mu_i \in X^n$ and smoothing parameter p_i. Then define $\phi_i(\mathbf{x}) = f(s(\mathbf{x}, \mu_i), p_i)$, where $s(\mathbf{x}, \mu_i) = \frac{1}{n} \sum_{k=1}^{n} s_k(x_k, \mu_{ik})$.

This S-neuron adds a non-linear *activation* function to the linearly aggregated similarities. Such function could be any sigmoid-like automorphism (a monotonic bijection) in $[0, 1]$. In particular, we consider the simple family of functions:

$$f(x,p) = \begin{cases} \frac{-p}{(x-0.5)-a(p)} - a(p) & \text{if } x \leq 0.5 \\ \frac{-p}{(x-0.5)+a(p)} + a(p) + 1 & \text{if } x \geq 0.5 \end{cases} \tag{6}$$

where $a(p) = -0.25 + 0.5\sqrt{0.5^2 + 4p}$ and $p > 0$ is a parameter controlling the shape of the function. For all $p > 0$, $\lim_{p\to\infty} f(x, p) = x$ and $f(x, 0) = H(x - 0.5)$, being H the Heaviside function. In this work we set $p_i = 0.1$ for all neurons i.

A Similarity neural network (SNN) is designed as a feed-forward architecture, with a hidden layer composed of S-neurons and a standard output layer. The k-th output neuron of the SNN computes the function:

$$\Phi_k(\mathbf{x}) = \sum_{i=1}^{h} w_{ki}\phi_i(\mathbf{x}) + w_{k0},\ k = 1, \ldots, m$$

where $h > 0$ is the number of hidden S-neurons, m is the number of outputs and $W = (w_{ki})$ is the weight matrix. The SNN can be naturally seen as a generalization of the RBF [4]. This is so because the response of hidden neurons is localized: centered at a given object (the neuron weight μ_i, where response is maximum), falling down as the input is less and less similar to this center.

The main difference is in the *interpretability* of the model. First, the output is a linear combination of the set of similarities of the input with a selected subset of prototypical elements. Second, the similarities are user-defined, and both the input and weight vectors are expressed in the original variables.

Consider now a training data sample $\{(\mathbf{x}_l, y_l)\}_{l=1}^{N}$. Since the SNN is a two-layer feed-forward neural network with local computation units in the first layer, training can be solved efficiently in a two-stage procedure, as detailed next:

First layer weights. The first layer centers are a subset of the examples in the sample dataset. These centers are chosen to be the cluster *leaders* returned by the LEADER 2 clustering algorithm acting on the set $\{\mathbf{x}_l\}_{l=1}^{N}$. The algorithm uses the user-defined similarity as explained in previous sections.

Second layer weights. Regularization is a technique that incorporates a complexity penalty to control overfitting when learning the weights:

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

$$SSE_\lambda(W) = \sum_{i=1}^{N}\sum_{k=1}^{m}\left(y_{ik} - \Phi_k(\mathbf{x}_i)\right)^2 + \lambda\sum_{j=0}^{h}\sum_{k=1}^{m} w_{kj}^2$$

where the first term is the sum of squared errors and the second is the regularization term (in this case, this is known as *ridge regression*). The minimization of SSE_λ forces to compensate smaller errors against smaller weights. We define the $H = (h_{ij})$ matrix as $h_{ij} = \phi_j(\mathbf{x}_i), i = 1, \ldots, N, j = 0, \ldots, h$. Let $A = H'H + \lambda I$, $P = I - HA^{-1}H'$, and y the vector of outputs, where I is an identity matrix of appropriate dimensions. For single-output networks ($m = 1$), the optimal weight vector is $\mathbf{w}^* = A^{-1}H'y$, the minimizer of $SSE_\lambda(\mathbf{w})$ for a fixed $\lambda > 0$. The *Generalized Cross Validation* error is $GCV = \frac{Ny'P^2y}{(Tr(P))^2}$.
When the derivative of GCV is set to zero, the resulting equation can be manipulated so that one λ appears isolated in one side of the equation. The value of λ can be re-estimated iteratively until convergence [4], using

$$\lambda = \frac{y'P^2yTr\left(A^{-1} - \lambda A^{-2}\right)}{(\mathbf{w}^*)'A^{-1}\mathbf{w}^*Tr\left(P\right)} \tag{7}$$

An initial guess for λ is used to evaluate expression (7), which produces a new guess. The obtained sequence converges to a local minimum of GCV. We use in this work the best result of the initial set $\lambda \in \{10^{-6}, 10^{-3}, 1\}$.

4 Experimental comparison

In this section we report on experimental work in which the SNN is compared to a standard RBF neural network (RBFNN) and to a SVM using the RBF kernel. These latter methods need a pre-processing of the data, carried out following the recommendations in [5] for non-continuous variables. The values of the different parameters (s_0 for the SNN, number of clusters and RBF width for the RBFNN, and cost and RBF width for the SVM) are optimized using a grid search.

Data set	C/R	Size	Variables	Missing?
Horse colic	C	368	21 (6N,7C,8O)	28%
Credit approval	C	690	15 (9N,6C)	5%
Voting records	C	435	16 (16N)	5.3%
Servo data	R	167	4 (2C,2N)	none

Table 1: Selected data sets: C/R stands for Classification/Regression. Legend for variable types: (N)ominal, (C)ontinuous, (O)rdinal.

Four challenging problems (see Table 1) have been selected as characteristic of modern modeling problems because of the diversity in data heterogeneity and the presence of missing values [6]. The resampling method consists in five repetitions of two-fold cross-validation (5×2 CV) [7]. A paired F-test can then be computed to assess potential statistical significance in the differences in performance. The hypothesis of two methods having the same error rate can be rejected at the 95% level when the F statistic exceeds $F > 4.74$ [8]. This test is difficult to satisfy.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

5 Discussion

The obtained 5×2 CV prediction errors are displayed in Table 2. The relevant quantity is the average mean square error (MSE), since this is the measure the methods have been trained to minimize. For classification problems, we also give the average percentage of errors. The F-test results are shown in Table 3. It can be seen that the SNN obtains better MSEs in all the problems. Sometimes this corresponds to a decrease in the percentage of errors (as in Horse Colic), sometimes not (as in Voting Records). Overall, it is the most competitive method of the three for these real-world problems described by a mixture of variables of radically diverse types, and in presence of missing information.

	Horse colic		Credit approval		Voting records		Servo
Method	error(%)	MSE	error(%)	MSE	error(%)	MSE	MSE
SNN	16.74	0.128	14.09	0.110	4.60	0.039	0.933
RBF	20.06	0.153	14.81	0.116	4.64	0.064	0.997
SVM	19.94	-	16.06	-	6.53	-	2.230

Table 2: Results in terms of average 5×2 CV and mean square errors (MSE).

	Horse colic		Credit approval		Voting records		Servo
Method	$F_{\%}$	F_{MSE}	$F_{\%}$	F_{MSE}	$F_{\%}$	F_{MSE}	F_{MSE}
RBF	3.053	**12.786**	1.337	2.884	0.805	1.036	0.786
SVM	2.584	-	3.276	-	2.298	-	**14.386**

Table 3: Results of the F statistic against the SNN, both for average 5×2 CV ($F_{\%}$) and mean square errors (F_{MSE}). Significant results (> 4.74) are boldfaced.

Acknowledgements. This study has been partially funded by the Spanish Government project TIN2009-13895-C02-01.

References

[1] Kaburlassos, V., Petridis, V. Learning and decision-making in the framework of fuzzy lattices. New learning paradigms in soft computing, Physica-Verlag (2002)

[2] D. Lin: An information-theoretic definition of similarity. In: International Conference on Machine Learning, Madison, Wisconsin, USA (1998)

[3] J. Hartigan. Clustering Algorithms. Wiley (1975)

[4] M. Orr. Introduction to Radial Basis Function Networks. Institute for Adaptive and Neural Computation. `http://www.anc.ed.ac.uk/rbf/papers/intro.ps.gz` (1996)

[5] L. Prechelt. Proben1-A set of Neural Network Benchmark Problems and Benchmarking Rules. Fac. für Informatik. U. Karlsruhe T.R. 21/94 (1994)

[6] Frank, A., Asuncion, A. UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. Univ. of California at Irvine, School of Information and Computer Science (2010)

[7] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10(7), pp. 1895–1923 (1998)

[8] E. Alpaydin. Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms. Neural Computation 11(8), pp. 1885–1892 (1999)