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.

Magnitude Sensitive Competitive Learning

Enrique Pelayo and David Buldain and Carlos Orrite

Aragon Institute for Engineering Research, University of Zaragoza - SPAIN

Abstract. This paper presents a new algorithm, *Magnitude Sensitive Competitive Learning (MSCL)*, which has the ability of distributing the unit weights following any magnitude calculated from the unit parameters or the input data inside the Voronoi region of the unit. This controlled behavior permits to surpass other standard Competitive Learning algorithms that only tend to concentrate neurons accordingly to the input data density. Some application examples applying different magnitude functions show the MSCL possibilities.

1 Introduction

Competitive Learning methods are a type of neural network models commonly used to find a solution to the approach of Vector Quantization task in an unsupervised way. Well known approaches are K-means, Frequency Sensitive Competitive Learning (FSCL) [1], Self-Organizing Maps (SOM), Neural Gas (NG) [5] and Generative Topographic Mapping (GTM) [2]. These methods produce a modeling representation of the data distribution in a group of neurons or units (codebook), in such a way that their weights, also called centers or prototypes (codewords), tend to concentrate in the densest areas of the data distribution.

This type of codification is optimal from the point of view of maximizing the Shannon's Information-Theory entropy for the use of codewords in a transmission task. Other applications may require an inverse relation between the codewords density and the data density, what is usually called as 'perceptual magnet' effect. It has been demonstrated that in biological systems, unusual stimuli are differentiated with high precision whereas frequent stimuli are distinguished only in a rough manner ([4]). Magnification control comprises several methods to modify this relation between both data and codewords density. Villmann et all ([7]) present a complete study of magnification in the SOM and the NG algorithms, including some possibilities of controlling it.

However it is not always desirable a codebook representation in function of data density. In this article we present a new algorithm, *Magnitude Sensitive Competitive Learning (MSCL)*. It has the property of distributing vector prototypes in certain data-distribution zones according to any arbitrary magnitude calculated locally for each unit, so opening the possibility of not only distributing the codewords in function of data density, but also in function of any other data-dependent magnitude, as it will be shown in the examples.

2 The MSCL algorithm

MSCL is a competitive algorithm with N units that are trained by a dataset of P data vectors $x(t)$ in a d-dimensional space. The total number of training iterations will be T, equal to the product of the number of samples (P) by the number of cycles (C). The unit weights are vectors $w_i(t)$, also in the d-dimensional space ($i = 1...N$). At training time t, a randomly selected input data $x(t)$ is presented to the network and the winner unit j (also called *Best Matching Unit* or BMU) is selected as the one that minimizes the product of a *Magnitude Function* $MF_i(t)$, evaluated for each unit i, and the distance of the unit weights to the input data vector, following:

$$j = argmin(MF_i(t)\|x(t) - w_i(t)\|); \quad i = \{1, \ldots, N\}, t = \{1, \ldots, T\} \tag{1}$$

where the Magnitude Function $MF_i(t)$ is evaluated as:

$$MF_i(t) = \beta(t) freq_i(t) + [1 - \beta(t)]M_i(t); \quad i = \{1, \ldots, N\} \tag{2}$$

where the parameter $\beta(t) = \beta_{ini}(\beta_{final}/\beta)^{t/T}$ is the learning factor, implemented to decay with time ($t = \{1, \ldots, T\}$), where β_{ini} and β_{final} are constants. It could also be defined to have a value 1 at the beginning of the training and 0 after a pre-defined time step. The function $freq_i(t)$ is a normalized count of the number of times that the unit has won competitions, updated iteratively for each training sample ($freq_i(t) = counter_i(t)/t$).

The frequency term is included to avoid the problem of dead units, so forcing the initial behavior to be like that of FSCL method, but as training progresses, its importance for deciding the BMU decays in favor of the magnitude term.

The magnitude$M_i(t)$ is a measure, normalized between 0 and 1, of any feature or property of the data inside the Voronoi region of the unit i, or a function of the unit parameters. Calculating $M_i(t)$ for every unit, each time a sample is presented to the network, could require high computational power if the function is complex, therefore, $M_i(t)$ is updated episodically each time that a number of data samples has been presented (epoch). The size of the epochs can be constant or variable along the training process (preferably increasing with time).

The idea behind the use of the magnitude term is that, in the case of a sample placed at equal distance from two competing units, the winner will be the unit with lower magnitude value. At the end of the training, units will be redistributed by moving from the data regions with low $M_i(t)$ values to regions where this magnitude is higher. An special case is when $M_i(t)$ is the normalized winning frequency of the unit, then we have a competitive behavior like that of the FSCL algorithm. Therefore MSCL is an extension of FSCL, being more versatile because the Magnitude Function permits to represent other data characteristics apart from the density, as we will see in the examples.

The winner's weights are adjusted iteratively for each training sample:

$$w_j(t + 1) = w_j(t) + \alpha(t) [x(t) - w_j(t)] \tag{3}$$

where the parameter $\alpha(t) = \alpha_{ini}(\alpha_{final}/\alpha)^{t/T}$ is the learning factor forced to decay with time ($t = \{1, \dots, T\}$), and α_{ini} and α_{final} are constants.

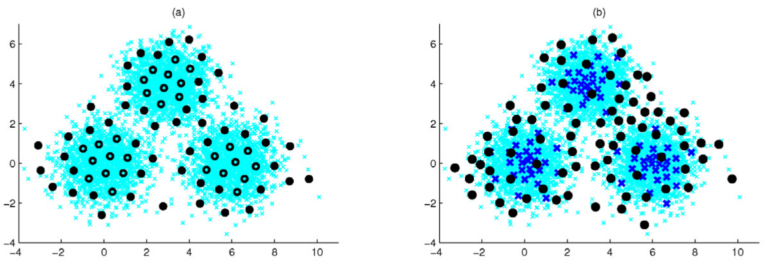

Fig. 1: First example using different M_i: (a) Q_{err} (b) Expansion/Contraction

3 Examples

For the sake of clarity about the Magnitude selection involved in the MSCL proposal, we evaluate the algorithm in three examples.

3.1 Modelling Gaussian distributions

Data set consists of $P = 5000$ samples in a plane from a mixture of three normal distributions shown in figure 1 as cyan crosses. To train the network modelling the data, we used 10 cycles, $\alpha_{ini} = \beta_{ini} = 0.9$, $\alpha_{final} = \beta_{final} = 0.01$ and $N = 80$ units. Figures 1a and 1b show the resulting centers corresponding to the MSCL algorithm for three different magnitudes. Figure 2a shows the training evolution of the Davies-Bouldin Index (DB Index)[3] for the three magnitude examples and FSCL. This index takes lower values for better clustering representations.

First example (see Figure 1a) corresponds to MSCL using a magnitude that forces units to distribute uniformly in the input data space. The value selected for $M_i(t)$ is the mean Q_{err} of all samples within the Voronoi region of unit i. $Q_{err}(x)$ is the distance between x and its corresponding BMU prototype. As we can appreciate in figure 2a this magnitude shows the better clustering representation by presenting the lowest values for the DB Index.

The next two examples need an estimation of the data density. In order to obtain this estimation, we make use of the prototype representation obtained in this first example. Lets suppose that the Voronoi area of each unit is similar, so we can approximate the density function as the number of hits of each unit. According to this discrete estimation of density, we can separate units in two groups using the Otsu method to obtain a suited threshold. Units with higher density are marked with black empty circles (Figure 1a), while black points represent the units with density below the threshold. This prototype representation will be used as a reference table to define a low density (w_{sparse}) and a high density regions (w_{dense}).

In the second example, the magnitude is oriented to concentrate units in data zones with high density, identified by w_{dense}. Magnitude $M_i(t)$ for each unit i is the distance $D_i(t)$ from its weights to the nearest prototype in the set w_{sparse}. This distance is normalized by the maximum. Using this magnitude function, the resulting unit representation (blue crosses in Figure 1b) is concentrated in cluster centers.

In the third example we aim to concentrate units in data areas with low data-density. We proceed in a similar way to the previous case, but using $M_i(t) = (1 - D_i(t))$. Therefore, units tend to move to the w_{sparse} region, usually in the boundaries of the clusters. This can be appreciated in Figure 1b, where black points represent this third example unit representation, more concentrated around the boundaries of the Gaussian distributions. Novelty detection applications, precisely need to clearly distinguish these boundaries, in order to identify if a new data sample belongs to the data distribution or not.

As these two last examples do not pretend to achieve an optimum clustering of the dataset, their DB Index evolution present final values higher than the other magnitudes (FSCL and $Q_{err}(x)$ in Figure 2a).

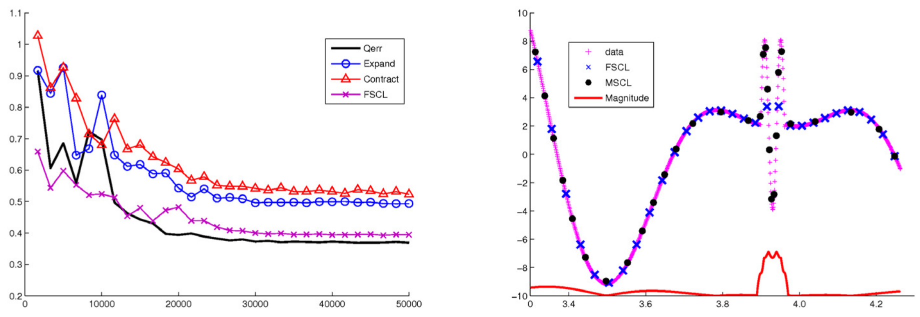

Fig. 2: (a) DB-Index for first example (*left*), (b) Interpolation example (*right*).

3.2 Interpolation application

Next example consists of interpolating data series. Data set was generated by uniformly sampling a function with a high frequency perturbation ($hf(x)$):

$$y(x) = 10\cos 12x + 12\sin 10x + hf(x); \quad x = \{3.3, \dots, 4.3\} \tag{4}$$

Two networks with 30 units and the same training parameters ($\alpha_{ini} = \beta_{ini} = 0.5, \alpha_{final} = \beta_{final} = 0.01$) were trained for 10 cycles with input data vectors (x, y): one with FSCL and the other with MSCL. The selected magnitude is focused to detect high frequency peaks. To do it, $M_i(t)$ is the average of the function $S(x)$ for all data samples assigned to unit i:

$$S(x) = smooth(|y(x) - y(x-1)|) \tag{5}$$

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.

The smooth function generate a five point moving average filter of the difference of two consecutive points in the series $y(x)$.

Figure 2b shows that MSCL (black circles) is more efficient than FSCL (blue crosses) to represent high frequency peaks. It is also represented (red line) the value of the magnitude function, that takes high values when $y(x)$ presents abrupt changes, and takes a lower value when $y(x)$ is near constant.

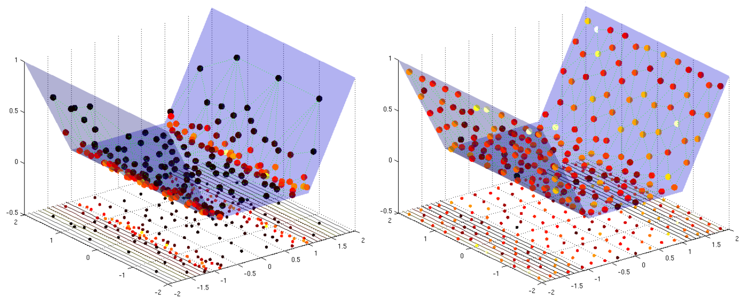

Fig. 3: Surface example: (a) Curvature (*left*), (b) Surface (*right*).

3.3 Surface Modeling

Point clouds are most often created by 3D-Scanners, and can be directly rendered and inspected, but generally point clouds themselves are not directly usable in many 3D applications. Therefore they must be converted to polygon or triangle mesh models, NURBS surface models, or CAD models. By using MSCL the whole point cloud might be modeled with a reduced number of units. However, the MSCL algorithm does not include any intrinsic definition of neighborhood, unlike other algorithms like SOM. In this paper we use the neighborhood definition proposed by [6], where two centers i and j are considered to be connected if it exists any data sample for which both units are the first and second BMUs.

In order to do surface modeling we use some artificial data, measured from a three planes surface like those shown in figures 3 (points are not represented). To simulate the scanning process, the Z value is measured in 10000 points uniformly distributed in X-Y axes. Dispersion is simulated by adding a Gaussian noise of low magnitude in the Z measure.

Two magnitude functions were explored training networks with 200 units. The first magnitude function was intended to be a measure of the curvature of the surface. In order to estimate the curvature, let us suppose that along the training process units distribute over the surface. We can consider that the Voronoi data-subset of each unit spans a little piece of the surface. This plane-piece of surface can be considered almost flat and centered at the unit prototypes.

PCA analysis of these Voronoi data subsets, should generate first and second principal components contained in these plane-pieces, while the third principal components should be perpendicular to these planes. For each unit it is calculated the vector difference between the unit weights and those of the mesh neighbor-units. These difference vectors are projected on the third principal component mentioned previously and averaged to generate the magnitude $M_i(t)$. When the surface is flat, the units and their closest neighbors will present a magnitude near zero. However, if the area surrounding a unit is curved, the magnitude will take high values and more units will be recruited to this zone. As it can be seen in Fig.3a, units tend to concentrate in the two edges with highest curvature (darker color points means lower magnitude).

The magnitude in the second example is focused in distributing uniformly unit prototypes in the surface. We need a magnitude function measuring the area sorrounding the unit prototype in its Voronoi region. As a measure of the surface assigned to each unit, we use the area spanned by the first and second eigenvectors of the PCA over its Voronoi data-subset . This Magnitude function definition forces units to share the surface area uniformly (Figure 3b).

4 Conclusions

The proposed MSCL algorithm works like usual competitive learning, but adds a magnitude function as a factor of the measure used for the competition. The effect of this factor forces the units to follow any arbitrary magnitude function unlike other standard Competitive Learning algorithms that usually generate a discrete approximation to the data probability density-function. As a result, MSCL is more versatile for distributing units following any property or characteristic of the data expressed by a user defined magnitude. The examples showed the MSCL capabilities in different applications: three Gaussian distribution quantization, one example of data series interpolation and two examples of surface modeling. The future work will consist in exploring new magnitudes for new applications and a further comparison of MSCL with other algorithms.

References

[1] S. C. Ahalt, A. K. Krisnamurthy, P. Chen, and D. E. Melton, Competitive learning algorithms for vector quantization, *Neural Networks*, 3:277-290, 1990.

[2] C. M. Bishop, M.Svensen and C. K. I. Williams, GTM: The generative topographic mapping *Neural Computation*, 10:215-234, 1998.

[3] D.L. Davies, D.W. Bouldin, A cluster separation measure, *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 224-227, 1979.

[4] P. K. Kuhl, Human adults and human infants show a ‘perceptual magnet’ effect for the prototypes of speech categories, monkeys do not. *Perception and Psychophysics*, 50:93-107, 1991.

[5] Th. M. Martinetz, S. G. Berkovich, and K. J. Schulten, ‘Neural-gas’ network for vector quantization and its application to time-series prediction, *IEEE Trans. on Neural Networks*, 4:558-569, 1993.

[6] K. Tasdemir and E.Merenyi, Considering topology in clustering of the Self-Organizing Maps, In proc. of 5^{th} *Workshop on Self-Organizing Maps* (WSOM 2005), 439-446, 2005.

[7] T. Villmann, J. C. Claussen, Magnification Control in Self-Organizing Maps and Neural Gas, *Neural Computation*, 18:446-469 2006.