Soft Multiple Winners for Sparse Feature Extraction

Harri Lappalainen
Helsinki University of Technology
Laboratory of Computer and Information Science
Rakentajanaukio 2 C, FIN-02150 Espoo, FINLAND
email: Harri.Lappalainen@hut.fi

ABSTRACT

A simple and computationally inexpensive neural network method for generating sparse representations is presented. The network has a single layer of linear neurons and, on top of it, a mechanism, which assigns a winning strength for each neuron. Both input and output are real valued in contrast to many earlier methods, where either input or output must have been binary valued. Also, the sum of winning strengths does not have to be normalized as in some other approaches. The ability of the algorithm to find meaningful features is demonstrated in a simulation with images of handwritten numerals.

1. Introduction

The objective of a feature extraction system is to refine data in such a way that further processing becomes easier. In classification, such a system has two basic goals. One is to remove redundancies from the data. This helps cutting down computational costs later on. Another, contradictory goal is to discriminate data, which helps in distinguishing between similar inputs. A feature extraction system should make a reasonable compromise between these goals, but so far there have not been satisfactory general algorithms to do this. Typically the feature extraction level has been tailored individually for each application.

Principal component analysis (PCA) methods [6] are good in reducing redundancies, but they sometimes perform quite poorly in discrimination. On the other hand, vector quantization (VQ) [4] and its soft variants [3, 5] are good in discriminating, but make poor use of available resources, which may lead to extremely large networks. In this paper a system, which combines the behavior of both linear PCA and soft VQ, is introduced.

The model is based on a layer of linear neurons. On top of this layer there is a mechanism, which assigns a winning strength for each neuron. This mechanism tries to imitate a function found in mammalian cortex. Inhibitory lateral connections are believed to implement a function, which enhances the activity of neurons fitting well to the input and represses the activities of other neurons. This can be seen as a competition for activity between neurons. Neurons sensible to similar inputs compete stronger than neurons sensible to more different inputs, while neurons that are sensible to totally dissimilar inputs have no mutual competition at all. This enables the neurons to locally discriminate between inputs, while distant neurons work independently enabling the network to process more information in parallel.

Instead of faithfully describing the lateral connections that implement the competition mechanism [2, 8], a simple and computationally inexpensive method for determining the winning strengths of the neurons is developed.

2. The general model

The network model has a layer of linear neurons, which compute a weighted sum of their inputs x_j.

$$p_i = \sum_j w_{ij} x_j = \mathbf{w}_i^T \mathbf{x} \tag{1}$$

The lengths of the weight vectors $\mathbf{w}_i$ are normalized to unity, and thus the output p_i of each neuron is the projection of the input on the direction of the weight vector. The final output y_i is obtained by multiplying the projections by the corresponding winning strengths μ_i.

$$y_i = \mu_i p_i = \mu_i \mathbf{w}_i^T \mathbf{x} \tag{2}$$

The determination of the winning strengths μ_i is the central part of the algorithm. They should be chosen so that the following properties hold.

1. $0 \leq \mu_i \leq 1$
2. If $p_i \leq 0$, then $\mu_i = 0$, that is, all $y_i \geq 0$.
3. All y_i are smooth unimodal functions of $\mathbf{x}$.
4. The eq. (3) should give a good reconstruction of $\mathbf{x}$.
5. A neuron with $\mu_i = 0$ can be removed without affecting other μ.

Property 2 enables the system to distinguish between positive and negative input. Property 4 enables the derivation of learning rule and also guarantees that as much information as possible is preserved. Property 5 makes it possible to restrict the iteration of eq. (6) to a small subset of all neurons.

The learning rule is obtained from gradient descent of the expected reconstruction error. The reconstruction of the input is defined to be

$$\hat{\mathbf{x}} = \sum_i y_i \mathbf{w}_i = \sum_i \mu_i \mathbf{w}_i \mathbf{w}_i^T \mathbf{x}, \tag{3}$$

and the reconstruction error

$$\mathcal{E} = \sum_t \|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|^2, \tag{4}$$

where the index t denotes different inputs.

3. Assigning the winning strengths

It is quite easy to solve the μ_i explicitly in the cases of one or two neurons. For a single neuron

$$\mu_i = \begin{cases} 0 & \text{if } p_i \leq 0 \\ 1 & \text{if } p_i > 0 \end{cases} .$$

The case of two neurons reduces to that of a single neuron, if either p_1 or p_2 is non-positive. We can thus assume $p_i > 0$. Let us now define $c_{ij} = \mathbf{w}_i^T \mathbf{w}_j$. If $c_{12} < 0$, we would have $\mu_1 = \mu_2 = 1$. Therefore, we can also assume $c_{ij} > 0$. Let us further define

$$r_{ij} = \max\left[0, \min\left(1, \frac{1 - c_{ij}\frac{p_j}{p_i}}{1 - c_{ij}^2}\right)\right]. \tag{5}$$

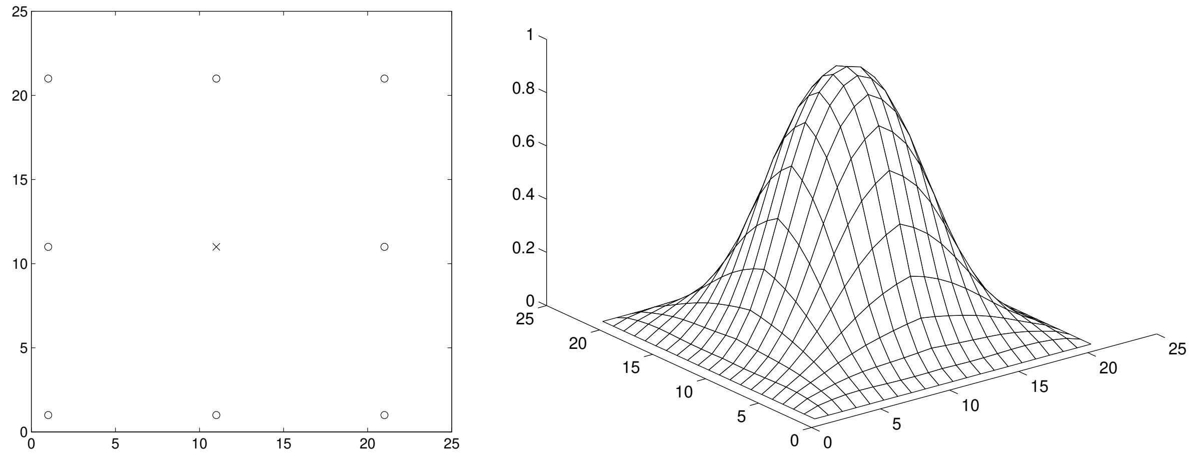
It can be readily shown that $\mu_1 = r_{12}$, and $\mu_2 = r_{21}$ will give a solution that minimizes the reconstruction error given by eq. (3), when $c_{12}, p_1, p_2 > 0$. Notice how the μ_i depend only on the direction of input vector, not on the magnitude.

The r_{ij} can be interpreted as the truth value for the statement 'Input is closer to neuron i than to neuron j'. This suggests a formulation for μ_i in the case of more than two neurons: it should be some kind of soft AND-function. The neuron i is winner if input is closer to neuron i than to any other neuron. The truth value of this statement would be $\prod_j r_{ij}$. However, a simple product doesn't agree with the requirement in property 5: that a neuron with $\mu_i = 0$ can be removed without affecting other μ. Together with the smoothness requirement in property 3, this suggests making the impact of r_{ij} in the product to depend on μ_j. We shall now write

$$m_i = \prod_j r_{ij}^{m_j}. \tag{6}$$

The solution of this equation can be computed by a simple fixed-point iteration. It can be seen from eq. 6 that if $m_j = 0$, then $r_{ij} > 0$ doesn't affect m_i. Let us define the set of winners $W = \{i | \forall j : r_{ij} > 0\}$. It will hold that $i \notin W$ implies $m_i = 0$. Thus the computation of m_i can be restricted to $i, j \in W$.

Letting $\mu_i = m_i$ would already yield a reasonable solution, but agreement with the reconstruction eq. (3) can be further improved by making a minor correction to m_i. Let us suppose there is a set of n neurons, which have nearly parallel weight vectors, and which are equally close to input, that is, for which $c_{ij} \approx 1$ and $p_i = p_j$ for all i and j. Then, by symmetry, $\mu_i = \mu_j$ for all i and j. From eq. (5) it can be seen that all $r_{ij} = 1/2$ for $i \neq j$. The eq. (3) gives the best reconstruction, when $\mu_i = 1/n$. From eq. (6) it follows



(a) (b)

Fig. 1: When the neurons are organized in two dimensional array like that in figure (a), the μ of the neuron 'x' will look like that in figure (b).

$m_i = (1/2)^{(n-1)m_i}$. Solving for $\mu_i = 1/n$ yields the correction

$$\mu_i = \frac{m_i}{m_i - \log_2 m_i}. \tag{7}$$

The figure 1 shows how the winning strength of a neuron is shaped by its neighbors. The Voronoi set in the context of VQ means the set of inputs, which are mapped to a particular neuron. Here the winning strengths μ_i are real valued and thus the Voronoi sets are fuzzy. If the weight vectors are nearly parallel and form a two dimensional array like that shown in figure 1 (a), the μ of the neuron denoted by 'x' would look like that in figure 1 (b). The picture shows the bell shape of the Voronoi set and how it extends to adjacent neurons.

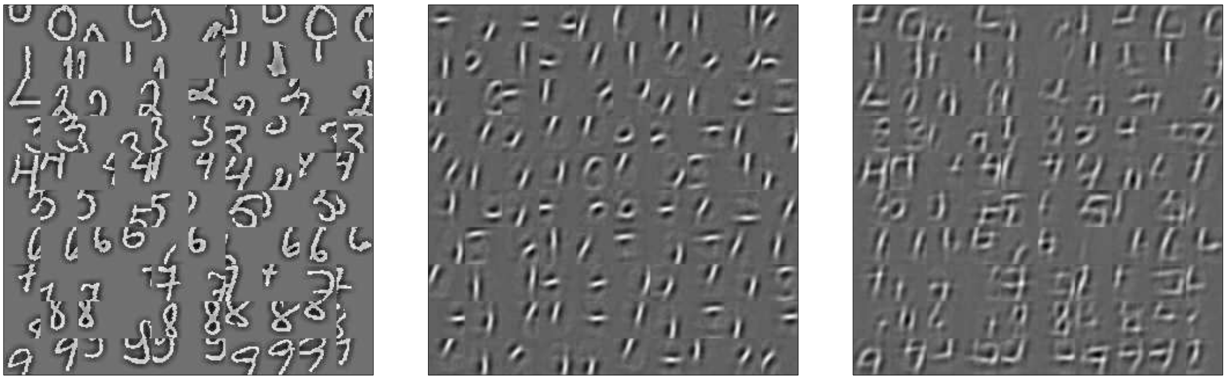
It should be noted that μ_i from eq. (7) does not strictly minimize the reconstruction error (3), but the solution is close enough to the minimum for the derivation of the learning rule. By assuming the μ_i optimal and constant, the learning rule can be obtained by gradient descent.

$$\mathbf{w}_i(t+1) = \frac{\mathbf{w}_i(t) + \alpha y_i(t)\mathbf{x}(t)}{\|\mathbf{w}_i(t) + \alpha y_i(t)\mathbf{x}(t)\|} = \frac{\mathbf{w}_i(t) + \alpha \mu_i(t)\mathbf{x}(t)\mathbf{x}(t)^T\mathbf{w}_i(t)}{\|\mathbf{w}_i(t) + \alpha \mu_i(t)\mathbf{x}(t)\mathbf{x}(t)^T\mathbf{w}_i(t)\|} \tag{8}$$

Calculating the r_{ij} in eq. (5) requires the dot products between weight vectors. It would be very time consuming to update the c_{ij} after each learning step. Therefore, it is advisable to use batch version of the learning rule.

4. Simulation results

The ability of the algorithm to find a meaningful representation for the data was verified in simulations with images of handwritten numerals. After randomly translating the images and locally removing the average, the 32x32 gray scale images were presented to the network. In figure 2 (a) there are 100 samples of the test set. Batch version of the learning rule of eq. 8 was used, and the training set consisted of 100 000 samples. The network of 100 neurons found elementary features from the data. The 100 weight vectors of the network are shown if figure 2 (b). One can clearly see that the emerged feature detectors agree with the intuitive view of the elementary features of handwritten numerals as being pieces of lines and arcs. In figure 2 (c) the reconstructions of the samples in figure 2 (a) are presented. The figure gives an idea of the amount of the information remaining in the outputs of the network. It has to be pointed out that the input data was deliberately made quite difficult by translations and removal of local averages. In spite of the difficult data the algorithm was able to find a meaningful representation for it.



(a) (b) (c)

Fig. 2: (a) 100 samples of test data. The 100 weight vectors were taught with similar samples. The results are presented in (b). (c) shows reconstructions of the images in (a).

5. Discussion

Several other authors have introduced algorithms that find a sparse representation for the data. One of the first algorithms was introduced by Földiák [2]. There outputs are binary valued. It also needs an extra parameter to control the sparsity of the outputs. Other algorithms [1, 7] only deal with binary input. Many algorithms that fall in the class of soft VQ [3, 5] normalize the sum of outputs, which means that the network cannot provide several streams of independent information. The cortex model of Sirosh [8] gives interesting results, but is not computationally efficient. It also yields population coding, which brings redundancy to the outputs. The redundancy is probably important in biological cortex, where one cannot trust in perfect operation of single neurons, but in computational models redundancy increases computational costs unnecessarily.

In the algorithm presented in this paper, both input and output are allowed to be real valued, and the outputs need not be normalized in any way. The outputs approximate population code with minimal redundancy. They depend linearly on the magnitude of the input. If this is not desirable, one can easily introduce a nonlinearity into p_i in eq. 2. Also, by integrating the p_i in time one could get invariant feature detectors [4].

References

[1] P. Dayan and R. S. Zemel, "Competition and multiple cause models," *Neural Computation*, vol. 7, pp. 565–579, 1995.

[2] P. Földiák, "Forming sparse representations by local anti-Hebbian learning," *Biol. Cybern.*, vol. 64, pp. 165–170, 1990.

[3] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 693–699, 1985.

[4] T. Kohonen, *Self-Organizing Maps.* Berlin: Springer-Verlag, 1995.

[5] T. Martinetz and K. Schulten, "A neural-gas network learns topologies," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula and J. Kangas, Eds. Amsterdam: North-Holland, 1991.

[6] E. Oja, "Principal component analysis," in *The Hand Book of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, Massachusetts: The MIT press, 1995.

[7] E. Saund, "A multiple cause mixture model for unsupervised learning," *Neural Computation*, vol. 7, pp. 51–71, 1995.

[8] J. Sirosh and R. Miikkulainen, "Cooperative self-organization of afferent and lateral connections in cortical maps," *Biol. Cybern.*, vol. 71, pp. 65–78, 1994.