Unsupervised Learning of Image Recognition with Neural Society for Clustering

Marcin Wojnarski

Warsaw University, Faculty of Mathematics, Informatics and Mechanics
ul. Banacha 2, 02-097 Warszawa, Poland
mwojnars@ns.onet.pl

Abstract. New algorithm for partitional data clustering is presented, *Neural Society for Clustering* (NSC). Its creation was inspired by hierarchical image understanding, which requires unsupervised training to build the hierarchy of visual features. Existing clustering algorithms are not well-suited for this task, since they usually split natural groups of patterns into several parts (like k-means) or give crisp clustering.

Neurons comprising NSC may be viewed as a society of autonomous individuals, proceeding along the same simple algorithm, based on four principles: of locality, greediness, balance and competition. The same principles govern large groups of entities in economy, sociology, biology and physics. Advantages of NSC are demonstrated in experiment with visual data. The paper presents also a new method for objective and quantitative comparison of clustering algorithms, based on the notions of entropy and mutual information.

1 Introduction

To understand and reliably recognize complex images we need a multi-layer hierarchical system of visual features – small and simple in the bottom, more and more complex in higher layers. This is how visual perception in the brain is organized. Several methods for creation of such hierarchy were already proposed: LeNet convolutional neural network developed by LeCun et al. [1]; HMAX model of object recognition in cortex by Riesenhuber and Poggio [2]; recent extension of HMAX by Serre et al. [3]; Neural Abstraction Pyramid by Behnke [4].

However, their performance is still very far from performance of the brain. One of the reasons for this is underestimation of unsupervised training and neglecting the possibility of information extraction from patterns themselves, while the fact that unsupervised training does not require class labels is a big advantage, especially when creation of a hierarchical system is considered.

Generally, there are three ways to build a hierarchy of visual features: (1) manually, by giving a mathematical description of every feature; (2) with supervised training, by providing examples of input patterns together with their class labels; (3) with unsupervised training, when class labels are not available.

The main weakness of the first approach is that we do not know what features should be represented in each layer. We know only roughly what features are

S. Greco et al. (Eds.): RSCTC 2006, LNAI 4259, pp. 862–871, 2006.

recognized at first stages of visual perception. Moreover, the features represented by biological neurons are fuzzy so it is difficult to define them mathematically.

Supervised training for large, multi-layer systems is intractable, as target classification is known only in the last layer and it is difficult to back-propagate this information through many layers, composed of thousands of units. We might try to use supervised training for each layer separately, starting with the first one and going up the hierarchy when the previous layer is trained. However, with this method we must know what features are needed in each layer and we have to (manually) label all occurrences of all the features in training data.

With unsupervised training, we can build the hierarchy in the most natural order: from the bottom to the top, layer by layer, without need to label manually huge amount of data. The reason why unsupervised algorithm could create useful features is that visual stimuli are *not* random combinations of pixels. The stimuli are composed of structures characteristic for the domain of the problem being solved. These structures occur much more frequently in training images than they would in purely random data, where pixels are picked independently of each other. Examples of such frequent structures in the task of face recognition would be the shapes of mouth, nose or chin. Thus, an unsupervised algorithm which discovers unusually frequent patterns could produce features that are useful for image understanding.

To give even stronger evidence that unsupervised training is essential for hierarchical image recognition, we might wonder how the brain learns to recognize images. Can it be supervised learning? If so, who or what is the supervisor? There are two possibilities: parents (environment) or genes. The first possibility is unlikely, since supervision from environment requires well-developed perception to communicate information between learner and supervisor – and perception is just what has to be learned. Genes certainly hold large amount of information about organization of visual perception, since they must describe algorithms which drive development of perception, but they surely do not describe precisely every single connection between neurons. Firstly because this is huge amount of information, too big to be stored in genome. Secondly, this would be extremely inflexible, making adaptation to environment almost impossible.

The above argument shows that unsupervised training must form the basis of visual perception development in the brain, so perhaps it could be applied to computer vision, as well. Moreover, we know that biological perception can develop properly only in the presence of stimuli, which is yet another argument for the use of stimuli-driven training in computer vision.

2 Clustering Algorithms

To build the hierarchy in unsupervised manner, we need a *clustering* algorithm to train a single layer. More precisely, this should be a *partitional* algorithm [5], and it must define a partition of the whole input space, not only of the training set. It would be also desirable to obtain fuzzy partition, instead of crisp. Only few existing methods satisfy these requirements. The most popular approaches

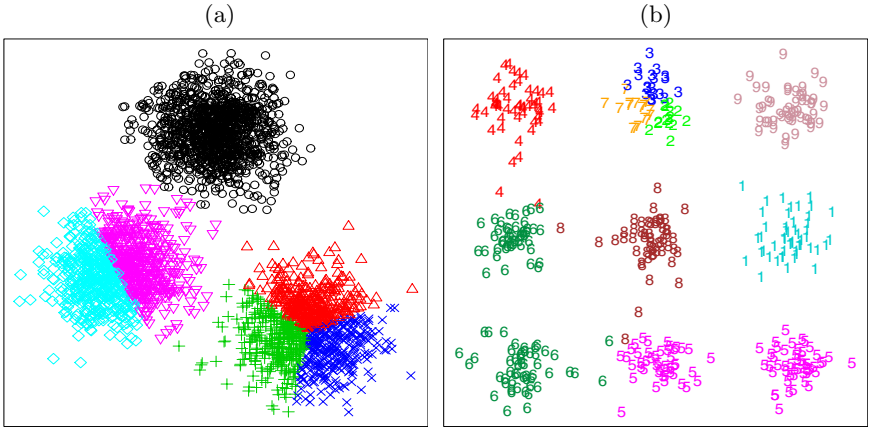

Fig. 1. Partitions obtained by k-means when k is larger (a) or the same (b) as the number of natural groups in data. In both cases some groups are split into 2 or 3 parts.

are k-means and gaussian mixture model (GMM) [5,6] trained with Expectation Maximization algorithm. However, both of them have serious disadvantages, which are overcome by the new algorithm introduced in this paper.

2.1 K-Means

The k-means algorithm finds cluster centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$ by minimizing:

$$\mathcal{E}(\mathbf{c}_1, \ldots, \mathbf{c}_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\mathbf{x}_i - \mathbf{c}_j\|^2 \ , \tag{1}$$

where n is the number of training patterns [5,6].

One of the weaknesses of k-means is that it partitions natural groups in the data into several separate clusters, even if k is exactly equal to the number of groups. This fact is illustrated in Figure 1. Two data sets composed of gaussian groups of points in the plane are clustered by k-means. For the first one, $k = 6$, which is more than the number of groups (3) – like in most of applications, where the exact number of groups is unknown and larger k should be chosen. For the second data set, k is equal to the number of groups (9). In both cases some groups are split into 2 or 3 parts, like a pie. Moreover, several groups from the second data set fall into the same cluster.

The above-mentioned characteristic of k-means is a weakness if the algorithm is used to build a hierarchy of visual features, as information propagated to the next layer incorrectly discriminates between patterns representing the same distorted prototype, thus biasing feature learning in the next layer.

Another disadvantage of k-means for hierarchical image understanding is that it assigns every pattern to exactly one cluster, in a crisp way, while some patterns may lie on the border between two or more features (then several clusters should

activate), and some others may lie far away from all cluster centers (then all clusters should be inactivated).

2.2 Gaussian Mixture Models

In gaussian mixture modelling we want to estimate the probability density function of data using a combination of parameterized normal densities [6]:

$$f(\mathbf{x};\Theta) = \sum_{i=1}^{k} w_i \phi(\mathbf{x};\theta_i) \ , \tag{2}$$

such that $\sum w_i = 1$, $\Theta = (\theta_1, \ldots, \theta_k, w_1, \ldots, w_k)$. The mixture is interpreted as a fuzzy partition of the input space, with cluster membership functions defined by posterior probabilities that pattern x belongs to component i. Parameters Θ of the mixture are found by maximization of the *log-likelihood* L of the data $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, as a function of Θ, which is usually done with Expectation Maximization (EM) algorithm [6]:

$$L(\Theta;\mathbf{X}) = \log f(\mathbf{X};\Theta) = \sum_{i} \log f(\mathbf{x}_i;\Theta) \ . \tag{3}$$

Clustering with gaussian mixtures seems to be much more sophisticated than k-means, thus perhaps it could give better results. E.g. if the number of components is bigger than the number of natural groups in the data, the mixture model may (sometimes) converge to a solution where several gaussians share the same parameters – this situation is easy to detect and fix. Moreover, GMM clustering is fuzzy, which is better for hierarchical image understanding.

However, despite the sophistication and complexity of GMM+EM algorithm, for multi-attribute data, e.g. images, it behaves exactly like k-means (!) – this is what "curse of dimensionality" means for GMM. It comes out that the boundaries between fuzzy clusters defined by GMM get so small in multidimensional space that in fact they disappear. The clustering becomes crisp and then the EM update rule becomes the same as in k-means.

3 Neural Society for Clustering

This section introduces a new clustering algorithm, *Neural Society for Clustering* (NSC), which may be seen as a type of a single-layer artificial neural network. However, the most important part of the system – its training algorithm – is devised in a completely different way than for standard neural networks. The algorithm is not a result of applying an optimization method to some error function, but instead it is designed to satisfy several simple principles formulated in natural language. These principles govern real societies in sociology, economy, biology or even physics – that is why the presented system is called "society" instead of "network".

Similar methodology, like the use of *locality* or *greediness* rules, can be found in *Local Transfer Function Classifier* (LTF-C) – a neural network for classification problems introduced by Wojnarski [7]. LTF-C-based systems are double winners of the EUNITE[1] world-wide machine-learning competitions, on "Modelling the Bank's Client behaviour using Intelligent Technologies" (2002) and "Prediction of product quality in glass manufacturing process" (2003).

3.1 General Assumptions

NSC is composed of some number of neurons, which generate activations in the range of $[0; 1]$. Every neuron corresponds to one cluster – the neuron activation defines cluster membership function – thus clusters are fuzzy and may overlap, or some regions of the input space may belong to no cluster (or have very low value of membership function).

Further in this paper, we will also use the notion of a neural *receptive field*, i.e. the subset of the input space on which the activation of a given neuron is high. Receptive field is a fuzzy set and in fact it is exactly the cluster represented by the neuron. Moreover, to find proper values of adaptive parameters of a neuron means to find a proper receptive field for that neuron, so a neuron is in fact the receptive field. Thus, in the following sections we will use the notions of cluster, neuron and receptive field interchangeably.

Training process is composed, as usually, of some number of cycles. Each cycle consists of: drawing randomly a training pattern, computing responses of all neurons and adjusting adaptive parameters.

3.2 The Principles

Creation of the training algorithm is a two-stage process. First, general principles of the training are formulated. Then, specific mathematical formulas are devised, which should satisfy the general rules. The principles of NSC are the following:

- *Locality*: the neuron must be activated to undergo training.
- *Greediness*: the neuron wants to be activated as often as possible, so it gets positive feedback after (moderate) activation.
- *Balance*: total activation of the network should be moderate, so neurons get negative feedback when total activation is too large and positive otherwise.
- ***Competition***: if several neurons activate simultaneously, only the winner gets positive feedback, others – negative.

The greediness principle does not affect fully activated neurons, because they do not need a feedback – full activation cannot be even larger. The competition principle is the most important one, as it drives the process of setting *decision borders* (the borders between clusters) in appropriate positions. Let us

[1] European Network of Excellence on Intelligent Technologies for Smart Adaptive Systems, http://www.eunite.org/.

consider a simple situation depicted in Figure 2. There are two groups of one-dimensional training patterns (dashes), distributed according to the presented density function. There are also two neurons, represented by their activation functions. However, the decision border (point where both activations are equal) lies far away from the minimum of the density. Will it get closer in next cycles?

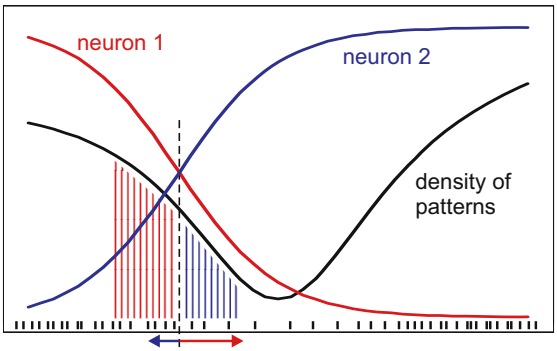

Fig. 2. Illustration of the competition principle. Two neurons (activation functions are shown) with overlapping reception fields compete, which leads to repositioning of decision border (*dashed line*). There is unequal distribution (*dashed regions*) of training patterns (*dashes* in the bottom) on both sides of the border, so their influence (*arrows*) is unbalanced and the border moves towards the minimum of density function.

The training process has stochastic nature (patterns are picked randomly), so we cannot say with certainty what will happen, but we can estimate an expected modification of neuron positions in the next cycle. Let us consider possible choices of the next training pattern x:

1. x lies far away from the decision border, where one of the neurons is fully activated and another one is quiet. In this case modification of adaptive parameters will be very small, due to the principles of locality (non-activated neuron does not undergo training) and greediness (fully activated neuron does not undergo training because its activation cannot be greater any more).
2. x lies to the left from the border, in its vicinity. Then, both neurons activate moderately, but the first one is the winner, so due to the competition principle it gets positive feedback and moves the reception field towards x – that is to the right. The second neuron is the loser, so it gets negative feedback and moves the reception field further from x – which also means to the right. Thus, the decision border gets moved to the right, as well, which is denoted by the right arrow in Figure 2.
3. x lies to the right from the border, in its vicinity. This situation is opposite to the previous one: now the second neuron is the winner, so it moves towards x, that is to the left. Similarly, the loser moves further from x, which also means to the left. In this way, the decision border moves to the left, as well.

The crucial point to observe is that the distribution of patterns on both sides of the border is unequal – the dashed area in Figure 2 is larger on the left side of the border than on the right. Hence, the right arrow is longer than the left one and the resultant force affecting the border is directed towards a minimum of density function. Thus, the competition rule drives repositioning of decision borders, forcing them to move towards decreasing density.

Note that similar principles as given above lie in the basis of many processes in sociology, economy, biology or physics. For example, the greediness and competition rules govern free-market economy – and this similarity with NSC is not a surprise, as one of major problems which an economic system must solve is how to cluster possible business activities and allocate them to firms. Moreover, the free-market economy achieves this goal by self-organization, as NSC.

3.3 The Training Algorithm

Till now, we have not specified the form of the neural activation function, because the principles are so general that they can be applied to very different types of activation functions. In this paper, we will assume sigmoidal form of activations:

$$f_i(\mathbf{x}) = \sigma\left(\mathbf{w}_i^{\mathrm{T}}\mathbf{x} - \alpha_i\right), \tag{4}$$

where $\mathbf{x}$ denotes a presented pattern (vector), $\mathbf{w}_i$ is a vector of weights, α_i is a threshold, i is the index of a neuron and σ denotes logistic function:

$$\sigma(t) = \frac{1}{1+\exp(-t)}\ . \tag{5}$$

After every cycle, weights and thresholds are adjusted according to the formulas:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta^{\mathrm{w}} F_i\, \mathbf{x}\ , \tag{6}$$

$$\alpha_i \leftarrow \alpha_i - \eta^{\alpha} F_i\ , \tag{7}$$

where η^{w} and η^{α} are predefined positive constants and F_i is the *total force* affecting the neuron with index i. The total force indicates whether the presented pattern would have positive or negative influence on the neuron. Namely, if the force is positive, the weights and the threshold are modified in such a way that the neuron activation would be stronger if the same pattern x is presented again. The total force is a combination of *balance force*, B_i, and *competition force*, C_i:

$$F_i = \gamma B_i + (1-\gamma) C_i\ , \tag{8}$$

where γ is a constant in $(0,1)$. Force B_i realizes the balance principle – it keeps total activation of neurons around one, while force C_i realizes the competition and greediness principles – it shifts receptive fields and decision boundaries:

$$B_i = y_i \left(1 - \sum_j y_j\right)\ , \tag{9}$$

$$C_i = y_i(1 - y_i)s_i \ . \tag{10}$$

In the above formulas, y_i denotes activation of neuron i in the last cycle, the sum runs over all neurons, and s_i is the winner indicator after the last presentation:

$$s_i = \begin{cases} +1, \text{ if neuron } i \text{ is the winner} \\ -1, \text{ if neuron } i \text{ is a loser} \end{cases} . \tag{11}$$

Factors y_i in the above formulas guarantee that locality principle is satisfied.

4 The Conformity Index

To perform a quantitative comparison of clustering algorithms, we devised the *conformity index*, κ, which measures similarity between two partitions of a data set. In our experiment, the first partition was the one obtained by NSC or k-means, and the second was true classification of the data (for NSC the clustering had to be made crisp by taking the most activated neuron for every pattern).

The difficulty in comparing partitions is that we do not know which clusters in the first partition correspond to which clusters in the second and it is usually impossible to draw exact correspondence. To solve this problem, an information-theoretic approach is used. Given two partitions P_1 and P_2 of a data set D, they are treated as random variables defined on D as a discrete stochastic space, with values in sets of cluster labels. Uniform distribution on D is assumed. Then, the conformity index is calculated as mutual information of P_1 and P_2 normalized by their joint entropy [8]:

$$\kappa(P_1, P_2) = \frac{\mathrm{I}(P_1; P_2)}{\mathrm{H}(P_1, P_2)} \ . \tag{12}$$

Intuitively, this index says what part of the whole information carried by P_1 *or* P_2 is contained in *both* of them. Such an index catches the intuition of similar partitions very well. It takes values between 0 (iff the partitions are stochastically independent) and 1 (iff the partitions are identical).

5 Experiment

Experiment with a set of artificially generated images was carried out. The data contained 20x20-pixel gray-scale images of four types: three groups of horizontal segments in different positions (top, middle, bottom) and a group of vertical segments. Each group contained 100 patterns of diverse length, orientation and exact position, as shown in Figure 3. Note the proximity of neighboring horizontal groups and the fact that vertical segments intersected with horizontal ones from all groups. The groups were also very wide in terms of Euclidean distance between extreme patterns. Pixel values were between 0 (black) and 1 (white).

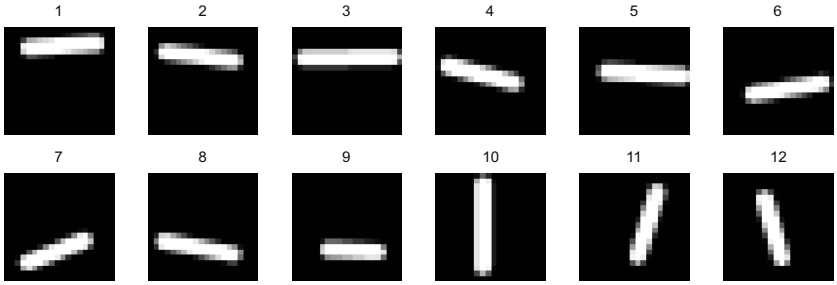

Fig. 3. Examples of training patterns. There are four groups of images: top horizontal segments (1-3), middle horizontal (4-6), bottom horizontal (7-9) and vertical (10-12).

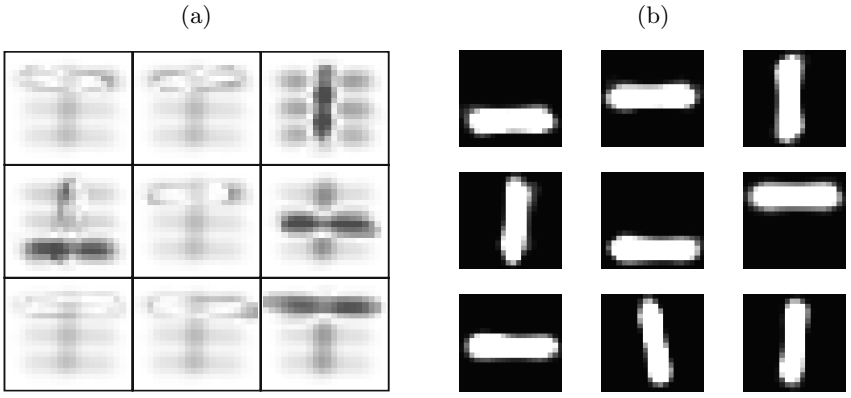

Fig. 4. (a) Weights of neurons of NSC (absolute values are depicted). Note the exact correspondence between meaningful neurons and genuine groups in the data. Unnecessary neurons atrophied and do not activate; (b) Cluster centers generated by k-means. Note that each group but the top horizontal was split into several clusters.

The data were clustered by NSC and k-means with 9 neurons or centers. Obtained neural weights and cluster centers are presented in Figure 4. Conformity index of the partitions was: 0.96 for NSC and 0.67 for k-means.

We may observe in Figure 4 that k-means splits natural groups into several clusters. Moreover, the cluster centers represent images that are asymmetric, distorted and far from prototypes of the groups. Such images are inappropriate as features in hierarchical image understanding. On the other hand, NSC can properly recognize the whole groups – unnecessary neurons are simply not used, their receptive fields are pushed away from the training patterns by competition and finally atrophy. As conformity index shows, the obtained partition corresponds almost perfectly to true classification.

The number of training cycles of NSC was 800. Values of parameters: $\eta^{\mathrm{w}} = \eta^{\alpha} = 0.1$, $\gamma = 0.3$. Neuron thresholds were initialized with 3 and weights with randomly picked training patterns scaled by 0.1.

6 Discussion

New clustering algorithm, *Neural Society for Clustering*, was presented in this paper. Development of this algorithm was motivated by the desire to build in unsupervised manner a hierarchical system for image understanding and recognition, although NSC is a general-purpose method. Existing clustering algorithms – k-means and Expectation Maximization for gaussian mixture model – are not well-suited for this task due to their tendency to split groups of similar patterns into several parts and because of crisp nature of the partition they produce when applied to multi-variable data. NSC gives fuzzy clustering and do not split natural groups of patterns – it can recognize if some neurons are unnecessary. NSC is based on four principles: of locality, greediness, balance and competition. The same principles govern real societies in economy, sociology and biology.

The paper presented results obtained by NSC and k-means in clustering of a data set of images. The results showed that NSC gives indeed a clustering which is very close to real partition into classes – contrary to k-means, which produces very fragmented partition. In order to quantitatively compare the algorithms, a measure of quality of partition, conformity index, was devised, based on the notions of entropy and mutual information. This index is intuitive and its boundary values are easy to interpret.

In the future, we plan to carry out experiments with real-world data sets, containing visual as well as non-visual data. We also plan to extend the presented algorithm to build a hierarchy of visual features.

Acknowledgements

The research has been supported by the grant 3T11C00226 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
2. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nature Neuroscience **2**(11) (1999) 1019–1025
3. Serre, T., Wolf, L., Poggio, T.: Object recognition with features inspired by visual cortex. In: IEEE CVPR. Volume 2. (2005) 994–1000
4. Behnke, S.: Hierarchical Neural Networks for Image Interpretation. Volume 2766 of Lecture Notes in Computer Science. Springer (2003)
5. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31**(3) (1999) 264–323
6. Ripley, B.D.: Pattern recognition and neural networks. Cambridge University Press, Cambridge (1996)
7. Wojnarski, M.: LTF-C: Architecture, training algorithm and applications of new neural classifier. Fundamenta Informaticae **54**(1) (2003) 89–105
8. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley (1991)
9. Jain, A.K., Law, M.H.C.: Data clustering: A user's dilemma. In: PReMI. (2005) 1–10