LEARNING VECTOR QUANTIZATION: CLUSTER SIZE AND CLUSTER NUMBER

Christian Borgelt

School of Computer Science
University of Magdeburg, Universitätsplatz 2
D-39106 Magdeburg, Germany
e-mail: borgelt@iws.cs.uni-magdeburg.de

Daniela Girimonte, Giuseppe Acciani

Department of Electrotechnics and Electronics
Polytechnic of Bari, Via Re David 200
I-70125 Bari, Italy
e-mail: {girimonte,acciani}@deemail.poliba.it

ABSTRACT

We study learning vector quantization methods to adapt the size of (hyper-)spherical clusters to better fit a given data set, especially in the context of non-normalized activations. The basic idea of our approach is to compute a desired radius from the data points that are assigned to a cluster and then to adapt the current radius of the cluster in the direction of this desired radius. Since cluster size adaptation has a considerable impact on the number of clusters needed to cover a data set, we also examine how to select the number of clusters based on validity measures and, in the context of non-normalized activations, on the coverage of the data.

1. INTRODUCTION

Learning vector quantization (LVQ) [7] is a well-known prototype based clustering method, which describes a cluster by a center and possibly some size and shape parameters. It tries to adapt these parameters in order to fit the clusters to a given data set. Closely related approaches are k-means clustering [5, 4] and fuzzy clustering [2, 6].

In this paper we focus on LVQ variants that do not rely on a normalization of the activations. Such methods are especially desirable, because the assignments of data points to clusters that result from them are generally more intuitive. However, these methods suffer from the drawback that normalization is an important means to achieve a mutual dependence of the individual prototypes, without which several clusters may end up in the same position. To solve this problem, we employ a restricted hard clustering/winner takes all scheme, which leads to a (limited) dependence of the clusters without removing the possibility to have data points that belong to some degree to several clusters.

2. LEARNING VECTOR QUANTIZATION

Learning vector quantization (LVQ) [7] is a well-known method to form a quantized approximation of the distribution of an input data set $\mathbf{X} \subset \mathbb{R}^p$ using a finite number k of reference vectors $w_i \in \mathbb{R}^p$, $i = 1, 2, ..., k$. These vectors are stored in the connection weights of a neural network with two layers, which is trained with competitive learning: for each input $\vec{x}_j$ the closest reference vector $\vec{w}_c$ is determined (i.e. $c = \arg\min_i\{|\vec{x}_j - \vec{w}_i(t)|\}$). The corresponding neuron "wins" the competition and is updated according to

$$\vec{w}_c(t+1) = \vec{w}_c(t) + \alpha(t)(\vec{x}_j - \vec{w}_c(t)),$$

where α, $0 < \alpha < 1$, is a learning rate, which, starting from an initial value α_0, reduces monotonically to zero (for example, according to $\alpha(t) = \alpha_0 \cdot \eta^t$, $0 < \eta < 1$).

In the standard version of LVQ only the "winner neuron" is updated. However, the similarity of LVQ to k-means clustering [5, 4] and thus also to fuzzy clustering [2, 6] suggests a softened version, in which the weights are updated according to the activation of a neuron, which may be computed from a Cauchy ($f_{\text{Cauchy}}(x) = \frac{1}{d^2/\sigma^2+1}$) or Gaussian activation function ($f_{\text{Gauss}}(x) = \exp(-\frac{1}{2} \cdot d^2/\sigma^2)$). Both of these functions depend on the ratio between the Euclidean distance d and a (user-specified) reference radius σ. In this case several neurons may be updated for each data point.

3. NORMALIZATION AND DEPENDENCE

In softened LVQ (as well as in standard fuzzy clustering) the neuron activations are usually normalized to sum 1 over the neurons/clusters, so that each data point has the same weight. This normalization makes the clusters dependent on each other: whatever is gained in data point coverage by one cluster must be lost by another. The same holds, of course, also for hard LVQ with its "winner takes all" approach. Mutual cluster dependence is very important for successful clustering, because it drives the clusters apart and thus makes sure that all data points are covered. However, it also has its drawbacks. For example, situations can occur in which the (normalized) activation increases even though one moves away from a reference vector [6].

To obtain a more intuitive cluster description, different suggestions have been made. One of them is possibilistic

fuzzy clustering [8], in which there is no normalization of membership degrees. Hence it can yield very intuitive cluster descriptions. Unfortunately, possibilistic fuzzy clustering suffers from the fact that its objective function is truly minimized only if all cluster centers are identical. In practice reasonable results are achieved only because the algorithm gets stuck in local optima. But even then, clusters tend to merge if they are not very well separated.

In this paper we draw on the idea of possibilistic fuzzy clustering and do not normalize the activations to sum 1. We try to overcome the resulting drawbacks by introducing a mutual dependence of the clusters through a restricted winner takes all scheme. If a data point has a distance from a reference vector that is less than a reference radius, it is assigned exclusively to the corresponding neuron (or, more generally, to the neuron yielding the highest non-normalized activation) and only this neuron is updated. For a data point outside the "winner takes all regions" non-normalized activations are computed and several neurons may be updated.

4. CLUSTER SIZE

To obtain a more flexible clustering scheme, one may make the reference radius σ in the activation functions neuron-dependent, updating it in each iteration, so that clusters of different size can be found. The general idea of the update is to compute a desired reference radius from the data points assigned to a cluster center/reference vector and then to

- set the reference radius to this desired radius or to
- change the current reference radius in the direction of the desired radius using a learning rate as for the update of the reference vectors.

The simplest choice for a desired radius is the average distance of the data points to a reference vector (or, alternatively, the square root of the average squared distance), with the data points weighted with the neuron activation. If online training is used for LVQ, a similar behavior can be achieved by updating the current radius according to

$$\sigma_i(t+1) = \sigma_i(t) + \alpha(t)(d(\vec{x}, \vec{w}_i(t)) - \sigma_i(t)),$$

where α may be the same learning rate as the one that is used for the neuron weights. In [1] a slightly more complex scheme is used, which distinguishes whether a data point is inside the (hyper-)sphere defined by the current radius (then only this radius is decreased) or outside the radius (hyper-) spheres of all clusters (then all radii are increased).

Other approaches are based on the relative weight of assigned data points, thus trying to find clusters that do not differ too much in the number of data points they cover. An example is frequency sensitive competitive learning [9], in which the distance to a reference vector is modified according to the number of data points that are assigned to this reference vector, i.e.

$$d_{\mathrm{mod}}(\vec{x}_j, \beta_i) = \frac{n_i}{n} d(\vec{x}_j, \beta_i),$$

where n_i is the number of data points assigned to reference vector β_i in the previous epoch and n is the total number of data points. Obviously, this is equivalent to using a reference radius $r = \frac{n}{n_i}$ to modify the activation.

Drawing on this idea, we may also state explicitely that our goal is to assign (roughly) the same number of data points to each cluster. That is, we desire $\frac{n}{c}$ data points per cluster, where n is the total number of data points and c the number of clusters. If a given radius r leads to an assignment of n_i data points, the desired radius is computed as

$$r_{\mathrm{desired}} = r \cdot \frac{n}{c \cdot n_i}.$$

The rationale is to decrease the radius if the desired number of data points is less than the current number and to increase it if it is greater, thus balancing the number of data points.

It should be noted that if we do not normalize the neuron activations, size adaptation can be slightly problematic, because in this case the sum of the activations over all neurons and all data points will, in general, differ from the total number of data points. Depending on the method to determine the desired radius, this can lead to collapsing clusters in some cases (e.g., if the average distance is computed from distances that are weighted with the activation). To cope with this problem, we introduce a parameter by which we multiply the computed desired radius before we use it to adapt the current reference radius.

5. CLUSTER NUMBER

One of the main problems in clustering is how to determine the optimal number of clusters. Usually the user has to specify how many clusters are to be found. Automatic approaches rely, for instance, on so-called *validity measures*, with which a given clustering result can be assessed, so that the best cluster number can be determined. Well-known validity measures, developed for classical and fuzzy clustering, are [6]: The fuzzy hypervolume (FH), the partition density (PD), and the average partition density (APD).

If we do not normalize the activations, another simple scheme suggests itself: With each new cluster some more data points should be covered. Hence we can compute the (absolute) coverage of the data as the sum of the activations over all data points and all clusters and stop adding clusters once the relative coverage (i.e. the absolute coverage divided by the number of data points) exceeds some user-defined threshold. Alternatively, we may stop adding another cluster if adding it increases the coverage only by a small amount, so that it is likely that the new clusters mainly steels data points from other clusters.

6. EXPERIMENTAL RESULTS

To illustrate the properties of different size adaptation methods in the context of non-normalized activations, we conducted experiments on simple artificial data sets with two point clouds of different size and with different numbers of points, each at two different distances from each other (see Figures 1, 2, and 3). To each of these data sets we applied five variants of generalized LVQ. The cluster are depicted as grey dots for the centers, dark grey circles at the reference radius and light grey circles at two times this radius.

Diagrams a_1 and a_2 show the result of standard LVQ (winner takes all) with adaptive cluster size. For diagrams b_1 and b_2 we used activation normalization to sum 1 and the same desired radius for all clusters (average of the individual desired radii). For diagrams c_1 and c_2 we also used activation normalization to sum 1, but individual sizes for each cluster. Diagrams d_1, d_2, e_1 and e_2 show results of clustering without activation normalization, but with a restricted winner takes all scheme (if a data point is inside the dark grey circle for the winner neuron, only this neuron is updated). In diagrams d_1 and d_2 the desired radius is twice the average distance of the data points (weighted with the activation) from the cluster center, in diagrams e_1 and e_2 we applied the scheme that balances the number of data points per cluster with reference radius factor of 0.8.

For the first two data sets (Figure 2, same cluster size, same number of points) all algorithms work very well, even though in the second data set the two clusters are very close together. Note that the restricted winner takes all approach can compete well with the other approaches.

For the second pair of data sets (Figure 2, different cluster size, same number of points) results are less good. A strict winner takes all almost fails since one cluster almost collapses (diagram a_2), i.e., the size adaptation interferes harmfully with the cluster detection. The approach that tries to capture the same number of points in each cluster (diagrams e_1 and e_2) yields the best result—not surprisingly, because it is tailored for such situations.

The last pair of data sets (Figure 3, different size, difference number of points, same density) leads to the biggest problems. Here only the strict winner takes all approach (diagrams a_1 and a_2) leads to acceptable results, while all other approaches have severe problems getting the cluster positions right. However, the results based on a restricted winner takes all scheme and no normalization can still clearly compete with the other approaches.

For our experiments on selecting the number of clusters by the coverage of the data points we chose the well-known wine data sets from the UCI machine learning repository [3]. We used the attributes 7, 10, and 13, which are most relevant. The results for clustering with restricted winner takes all and no normalization are shown in table 1. Clearly the coverage of the data points gives a very good indication of the correct number of clusters (there are three classes), because the coverage reduces if four clusters are used. For other data sets we tried, the coverage could also compete with the known validity measures.

wine	1	2	3	4	5
FH	9.35	10.6	6.65	6.63	6.77
PD	12.5	11.1	18.1	18.1	18.2
APD	12.5	20.8	3119	2575	4266
coverage	130	131	160	157	163

Table 1. Selecting the number of clusters.

7. CONCLUSIONS

In this paper we considered LVQ methods to adapt the size of (hyper-)spherical clusters, especially in the context of non-normalized activations. Our experimental results show that, although normalization clearly helps the clustering process and stabilizes it, usable results can often be achieved without normalization if the desired radius is multiplied by a user-specified parameter to achieve stability (where necessary). It also turns out that without normalization a consideration of the coverage of the data is a feasible method to determine a good number of clusters.

8. REFERENCES

[1] G. Acciani, E. Chiarantoni, G. Fornarelli, and S. Vergura. A Feature Extraction Unsupervised Neural Network for an Environmental Data Set. *Neural Networks* 16(3–4):427–436. Elsevier Science, Amsterdam, Netherlands 1999

[2] J.C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York, USA 1981

[3] C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA, USA 1998. http://www.ics.uci.edu/˜mlearn/MLRepository.html

[4] B.S. Everitt. *Cluster Analysis*. Heinemann, London, UK 1981

[5] J.A. Hartigan and M.A. Wong. A k-means Clustering Algorithm. *Applied Statistics* 28:100–108. Blackwell, Oxford, UK 1979

[6] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis.* J. Wiley & Sons, Chichester, UK 1999

[7] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg, Germany 1995 (3rd ext. edition 2001)

[8] R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering. IEEE Transactions on Fuzzy Systems, 1:98–110. IEEE Press, Piscataway, NJ, USA 1993

[9] D. DeSieno. Adding a Conscience to Competitive Learning. *IEEE Int. Conf. on Neural Networks*, Vol. I, 117–124. IEEE Press, Piscataway, NJ, USA 1988

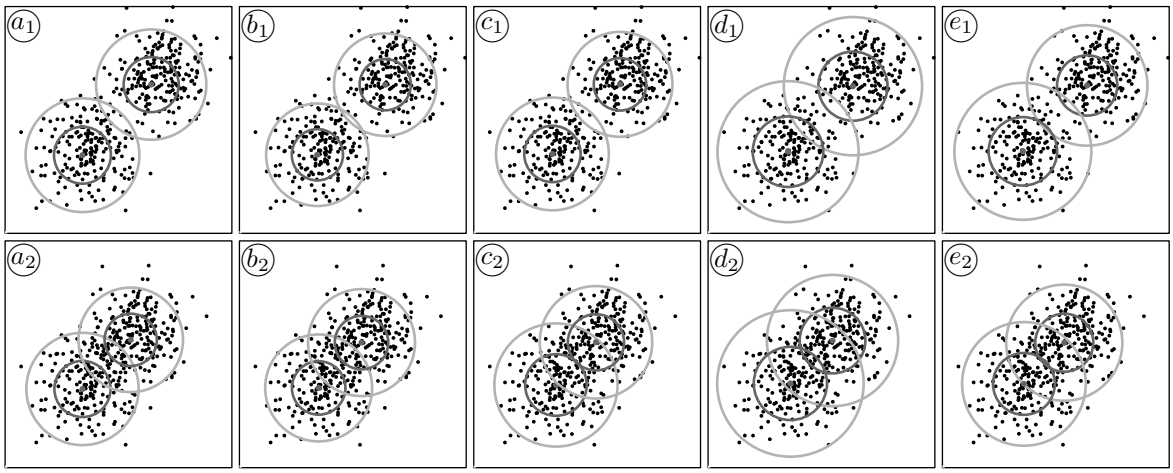

Fig. 1. Same cluster size, same number of points.

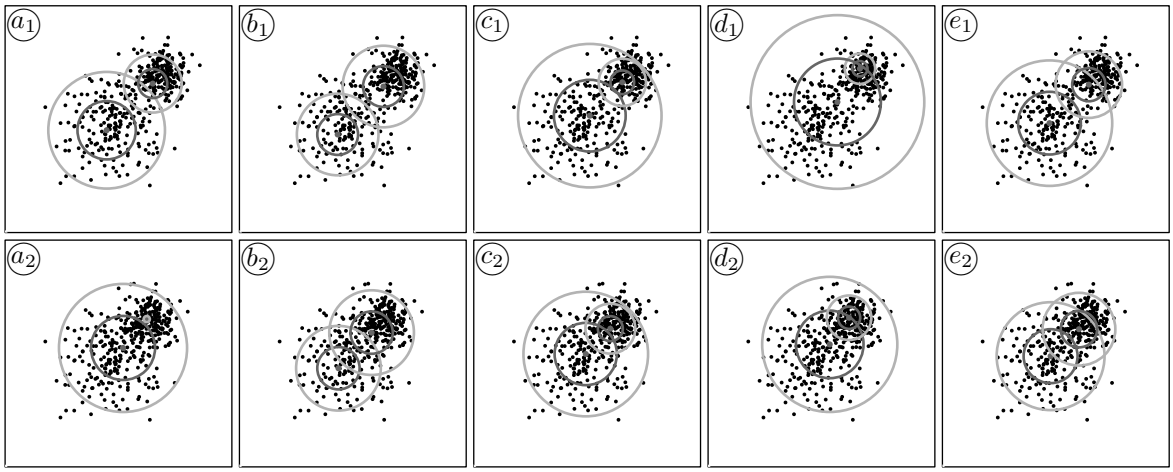

Fig. 2. Different cluster size, same number of points.

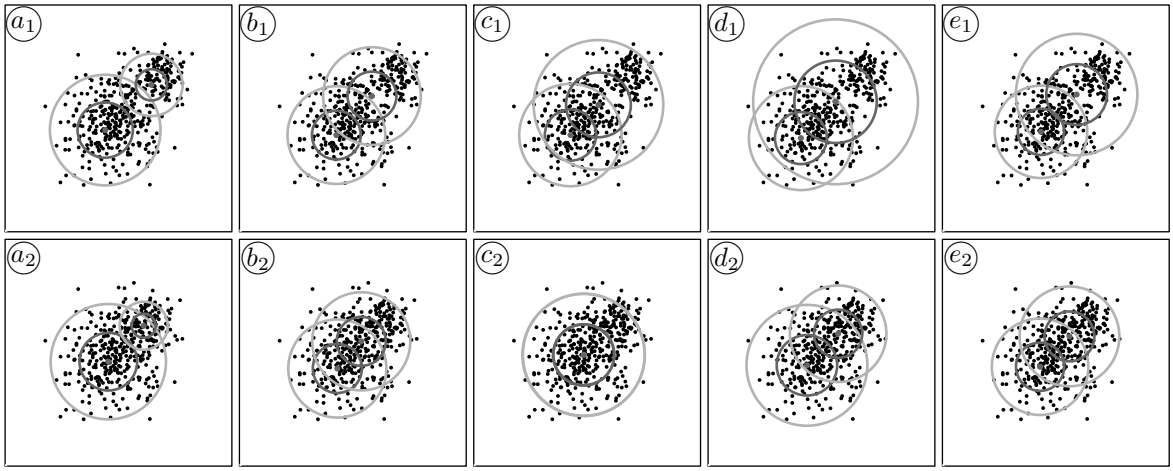

Fig. 3. Different cluster size, different number of points, same point density.