An Adaptive Learning Vector Quantisation Algorithm for Cluster Analysis and Radar Pulse Train Deinterleaving

Stephen D. Elton
Electronic Warfare Division
Defence Science and Technology Organisation
P.O. Box 1500, Salisbury, SA 5108, Australia

ABSTRACT

We examine the use of an unsupervised learning vector quantisation (LVQ) algorithm for cluster analysis and radar pulse train deinterleaving. A modification to the conventional LVQ technique is proposed and results in a neural network with improved performance and which adapts itself to the signal environment. The new method provides an enhanced degree of robustness to cluster initialisation and to the order in which the data are presented to the clustering network. It also has the potential to offer computational savings and is general enough to be applicable to other areas of signal recognition and signal classification.

1. Introduction

Pattern recognition techniques, including those based on neural networks, are well suited to the types of processing carried out by a radar intercept receiver in monitoring the activity of the radar signal environment. Radar intercept receivers are frequently required to deinterleave a time interleaved signal, for example, that results from the simultaneous reception of the pulsed emissions from multiple radar sources. The application of cluster analysis techniques can be a particularly effective method of assisting with the required signal separation. Cluster analysis might also form the first component of a multi-stage deinterleaving process and could be followed by a pulse time of arrival deinterleaving method; the primary objective being to provide radar source identification in a timely manner. In addition to being computationally efficient, the signal separation algorithms must also be robust to signal environment uncertainty and to missing and false data.

Radar pulse train deinterleaving is performed using one or more radar signal descriptors that are made available to the deinterleaving processor, either on a pulse-by-pulse basis, or in time-ordered batch data format. The radar signal parameter or feature set typically includes the following: pulse width, pulse amplitude, pulse time of arrival, angle of arrival, radio frequency, and other intra-pulse features. In this article we consider the use of clustering techniques based on a signal parameter feature vector or input vector comprising the radio frequency (RF) and angle of arrival (AOA) parameters. The cluster based deinterleaving technique readily admits the incorporation of additional signal descriptors however.

The neural network based cluster method that we investigate here for signal separation and signal classification relies on an unsupervised version of the learning vector quantisation (LVQ) algorithm [2], and is an example of a competitive learning technique. The algorithm is also quite closely related to the classical K-means clustering method (e.g., see [3]). The LVQ algorithm in its standard form has been applied to the pulse train deinterleaving problem previously in [1].

In this article the conventional LVQ algorithm is applied to a simulated data set, but with limited success. To overcome several deficiencies of the standard technique when used in connection with the signal separation and recognition problem, we propose a modified, data driven version that provides improved performance and which adapts itself to the signal environment under consideration.

2. Cluster Analysis

Clustering algorithms are designed to partition unlabelled feature vectors into clusters or groups, in such a way that the input vectors belonging to a particular cluster exhibit similar features, while differing from those features associated with the remaining clusters.

2.1. Conventional LVQ Algorithm

Let $x(n)$ denote a stochastic vector of observations of dimension $P$, that represents the signal feature set selected for cluster analysis, with

$$x(n) = [x_1(n) x_2(n) \ldots x_P(n)]^T,$$  (1)
where \( n = 0, 1, \ldots, N - 1 \) denotes the discrete time indexing of the measurements. We also define a set of \( L \) codebook vectors, \( \{ m_i(n) \}_{i=1}^{L} \), each of dimension \( P \), such that

\[
m_i(n) = [m_{i1}(n) \ m_{i2}(n) \ldots m_{iP}(n)]^T,
\]

with \( n = 0, 1, \ldots, N - 1 \). For the deinterleaving problem of interest, these codebook vectors act as a series of signal reference feature sets for the intercepted radar emissions. Comparison of the codebook vectors with the feature vectors that are fed into a clustering network, allows the network to discern patterns and to recognise clusters in a multi-dimensional feature space. In the "winner takes all" approach to codebook comparison, the codebook vector, \( m_c(n) \), closest to \( z(n) \), in terms of a weighted minimum Euclidean distance measure

\[
d[z(n), m_c(n)] = ||z(n) - m_c(n)||_w, \]

\[
= \min \{ ||z(n) - m_i(n)||_w \},
\]

with \( l = 1, 2, \ldots, L \), is updated according to

\[
m_c(n + 1) = m_c(n) + \alpha(n) [z(n) - m_c(n)],
\]

where \( c \in \{1, 2, \ldots, L\} \), and \( \alpha(n) \) describes a scalar valued gain term. The remaining codebook vectors are left unchanged, i.e.,

\[
m_i(n + 1) = m_i(n) \quad \text{for} \quad l \neq c.
\]

The LVQ algorithm as presented above, may be implemented using a feedforward neural network and represents the technique in its simplest form, other more elaborate versions also exist (e.g., see [2]). The input layer of the network is made up of \( N \) nodes, each of which is connected directly to \( L \) neurons which comprise the output layer. The neural network weights associated with the processing elements of the output layer, correspond to the set of reference codebook vectors \( \{ m_i(n) \}_{i=1}^{L} \). A parallel implementation of the LVQ technique results in a computationally efficient clustering algorithm.

The gain term, \( \alpha(n) \), in Equation (5) sets the rate of learning, where \( 0 < \alpha(n) < 1 \), and is usually taken to be a monotonically decreasing sequence with respect to \( n \). From [1,2], we will use a gain function of the following form

\[
\alpha(n) = \alpha(0) \left(1 - \frac{n}{N_{\text{max}}} \right),
\]

where \( n = 1, 2, \ldots, N - 1 \), \( \alpha(0) \) defines the initial rate of learning, and \( N_{\text{max}} \) is an integer such that \( N_{\text{max}} \geq N \). As a "rule of thumb," it has been suggested that the number of iterations used by the LVQ algorithm should be at least \( 500L \) [2], where \( L \) is the number of neural network processing elements. It was also suggested in [2] that data sets of insufficient length be recycled multiple times to meet the required number of iterations, in which case \( n \) is no longer a discrete time index, but rather, an iteration index number.

In this work the Mahalanobis distance is used to compute distances for feature vectors containing signal parameters recorded in different units. Each feature vector is assumed to be corrupted by additive zero-mean Gaussian white noise with covariance matrix denoted by \( \Sigma_x \). The Mahalanobis distance between \( z(n) \) and \( m_i(n) \) is then

\[
||z(n) - m_i(n)||_w = \left( \sum_{p=1}^{P} \sum_{q=1}^{P} w_{pq} \right)^{\frac{1}{2}},
\]

with scalar weight \( w_{pq} = [\Sigma_x^{-1}]_{pq} \), where \([\cdot]_{pq}\) denotes the \([p, q]\) element of a matrix.

2.2. LVQ Based Cluster Analysis

To summarise, the LVQ algorithm operates by cycling through each of the \( N \) feature vectors, with \( z(n) \in \mathbb{R}^P \), comparing the weighted Euclidean distance between each codebook vector and the current feature vector, and moving the closest codebook vector, \( m_c(n) \), towards that feature vector. Under appropriate conditions, the codebook vectors tend to gravitate towards the true cluster centroids as new data are presented to the network. The LVQ algorithm itself does not necessarily perform clustering however. Rather, it generates estimates of the cluster centroid positions which we interpret as an output from the cluster network’s processing elements. In our particular application, once all of the feature vectors have been processed using a single pass through the data and the final estimates of the cluster centroids, \( \{ m_i(N-1) \}_{i=1}^{L} \), arrived at, each feature vector is again considered in turn (to allow, in part, for changes in the closest codebook vector), and classified as a member of a specific cluster based on the following criteria:

1. nearest neighbour selection using the minimum distance between the feature vector \( z(n) \) and the codebooks \( \{ m_i(N-1) \}_{i=1}^{L} \);
2. inclusion of a feature vector within a cluster boundary.

We note that missing data are not likely to represent a significant problem to the cluster algorithm. However, the latter of the above two criteria is used in the signal separation problem to reject outliers corresponding to noise or false pulses.

Cluster boundaries are constructed using the estimated means of the noise corrupted feature vectors, together with the covariance matrix, \( \Sigma_x \), of the noise which we assume known. If in fact \( \Sigma_x \) is
unknown, it should be estimated as part of the cluster analysis process. The observed within-cluster variations in feature vectors, which we model using Gaussian distributions, are assumed to arise from intercept receiver measurement errors. Let \( \{z(n)\}_{n=0}^{N-1} \) denote a sequence of vector observations with \( z(n) \) distributed according to a multivariate Gaussian distribution with covariance \( \Sigma_z \). A feature vector \( z(n) \) is then accepted as a member of the nearest cluster with codebook \( m_c(N-1) \) if

\[
||z(n) - m_c(N-1)||_w \leq d_c, \tag{9}
\]

where \( d_c \) defines the cluster acceptance threshold and is sometime referred to as the "number of sigmas" of the region. The cluster association region represented by Equation (9) corresponds to an ellipsoid of probability concentration.

Allocation of a feature vector to a particular cluster constitutes the signal separation or deinterleaving process and feature vectors that are clustered together could then be passed off to a postprocessor if desired. Final codebook vectors could also be used to assist with radar source classification and identification.

### 2.3. Cluster Network Initialisation

If training data sets are available in the form of labelled feature vectors that are known to be associated with a particular source class, codebook vectors could be assigned during a cluster network training phase using supervised learning. Unfortunately, supervised learning is not likely to be practical for the pulse train deinterleaving problem for several reasons. First, the AOA parameter, which is recognised as a particularly effective signal discriminator, cannot be known a priori. Second, the large number of radar emitter types and modes of emitter operation will result in a prohibitively large number of codebook vectors for comparison with the input vectors. In the absence of training data, random initialisation of the codebook vectors would appear to be standard practice; the only provision being that the codebooks are seeded using different numerical values.

Note that in the standard implementation of LVQ, the number, \( L \), of neural processing elements or codebook vectors must be set prior to the analysis of any data. Naturally, this number reflects the number of clusters that are believed to exist in the data set. In the pulse train deinterleaving application however, one is effectively trying to solve a blind signal separation problem in which one does not have prior knowledge of the number of sources that give rise to the signal environment. In Section 2.5 we propose a cluster allocation method that largely avoids this issue by dynamically creating new cluster processing elements as needed.

### 2.4. Pulse Train Deinterleaving Example

We now consider an example that illustrates the performance of the LVQ algorithm for the radar pulse train deinterleaving problem. We will use a simulated benchmark data set comprising uncorrelated RF and AOA measurements, and define the feature vector \( z(n) \) by

\[
z(n) = [rf(n) \ aoa(n)]^T. \tag{10}
\]

Each RF and AOA measurement is assumed to be independently Gaussian distributed with a variance of \( \sigma_{rf}^2 \) and \( \sigma_{aoa}^2 \) respectively, so that \( \Sigma_z \) is given by \( \Sigma_z = \text{diag}(\sigma_{rf}^2, \sigma_{aoa}^2) \). For the purpose of evaluating the performance of the LVQ algorithm, we will concentrate on the data that defines the RF-AOA "region of interest" shown in Figure 1, and which results in \( N = 755 \) feature vectors for analysis.

The LVQ algorithm was implemented using a network made up of \( L = 5 \) processing elements (to match the true number of clusters in the data set), and with \( a(0) = 0.9, N_{\text{max}} = N, \sigma_{rf} = 1 \) MHz and \( \sigma_{aoa} = 1 \) deg. The initial codebooks, \( \{m_c(0)\}_{c=1}^5 \), were assigned by sampling from uniformly distributed random vectors that span the RF-AOA region of interest. Results typical of LVQ based cluster analysis are shown in Figure 1, from which we conclude that the conventional LVQ technique has performed rather poorly since only one cluster was correctly identified in the data. Cluster boundaries are also shown and correspond to \( d_c = 4 \).

The reason for the poor performance of the standard LVQ algorithm could be related to the fact that we chose to compute codebook vectors using just one pass through the data set; the resulting number of algorithm iterations falling far short of the 2500 suggested by the rule of thumb of [2]. Certainly, the LVQ algorithm was found to be sensitive to the initialisation of the codebook vectors, since significant improvement was achieved by manually seeding the codebooks with vectors close to the true cluster centroids. For other data sets, additional problems included the attraction of a single cluster codebook to the mid-point of two closely spaced data distributions. Rather than increasing the number of algorithm iterations by recycling the data however, we will opt for a less computationally intensive strategy that is better suited to the near real-time processing constraints imposed on a radar pulse train deinterleaver.

### 2.5. Adaptive LVQ Based Cluster Analysis

In light of the apparent deficiencies of the conventional LVQ based clustering algorithm for the radar pulse train deinterleaving application, we propose a simple modification that yields improved results and which, depending on the processor architecture
used for implementation, has the potential to offer computational savings. In the conventional implementation of LVQ, the potential number of clusters is established prior to the processing of any data. In contrast to this approach, we propose a data driven methodology, whereby a new codebook vector is dynamically allocated when a new cluster is required. Although an upper limit could be set for the total number of neural network processing elements created, the number of processing elements becomes a dynamical quantity, allowing the network to adapt itself to the signal environment. We term the modified method an adaptive learning vector quantisation (ALVQ) based clustering technique.

The ALVQ algorithm begins by assuming that a single cluster exists, i.e., we set \( L(0) = 1 \), and seeding that cluster with the codebook \( m_1(0) = x(0) \). We also let \( \{x(n) = x(n+1)\}_{n=-2}^{N-1} \). Additional codebooks are then created when a new input vector is deemed to lie sufficiently distant from all of the codebook vectors that currently exist. Variations on this simple approach were taken in the application of the K-means algorithm in [3], and a supervised learning version of LVQ in [4]. Once again, we assume that the dispersion of feature vectors about the cluster centroids results from measurement errors, and that the measurement error covariance matrix is known. As each feature vector, \( x(n) \), is processed by the ALVQ algorithm, its distance to each of the \( L(n) \) current codebooks is calculated in the usual way, and the minimum distance \( d[x(n), m_c(n)] \) recorded. If \( d_{thr} \) denotes a cluster network expansion threshold parameter and \( d[x(n), m_c(n)] \leq d_{thr} \), codebooks are updated as in the LVQ algorithm and \( L(n+1) = L(n) \). If however, the threshold parameter is exceeded, the number of neural network processing elements is incremented by one, i.e.,

\[
L(n+1) = L(n) + 1. \tag{12}
\]

In this work we let \( d_{thr} = 6 \). The new cluster is also seeded with a codebook vector given by

\[
m_{L(n+1)}(n+1) = x(n). \tag{13}
\]

False pulses are assumed to generate singleton clusters which would fail a cluster validation test. In the ALVQ based cluster algorithm, two passes are made through the data set \( \{x(n)\}_{n=-1} \), the first to compute estimates of cluster centroids and the second to classify feature vectors as members of a particular cluster. The second pass through the data could also be used to refine the cluster centroid estimates. If the data distributions are sufficiently well separated, however, a single pass should be possible that permits pulse-by-pulse deinterleaving. Centroid estimate updates and feature vector classification are then performed simultaneously.

In Figure 2 we show the results of the ALVQ based clustering algorithm (including four-sigma contours) for the RF and AOA pulse train data discussed previously and with \( \sigma(0) = 0.9, N_{max} = N, \sigma_{RF} = 1 \text{ MHz} \) and \( \sigma_{AOA} = 1 \text{ deg} \) as before. The ALVQ algorithm has clearly performed significantly better than the LVQ method for this data set, with a total of five clusters correctly identified.

3. Discussion

The pulse train signal separation problem described herein brings to light two key questions that often arise when applying a clustering algorithm with unsupervised learning. First, how should we choose

---

**Figure 1:** Results of the conventional LVQ algorithm for the pulse train deinterleaving example and five randomly initialised codebook vectors. Initial codebooks are denoted by '+'s, final codebooks by '*'s and classified feature vectors by 'o's. Unclassified feature vectors are represented by 'o's.

**Figure 2:** Results of the ALVQ algorithm for the pulse train deinterleaving example. Each feature vector has been correctly classified as a member of a cluster.
the number of clusters to implement the algorithm with? Second, how should we choose to initialise each cluster? The ALVQ algorithm provides answers to these questions by appealing to the data themselves and its advantages over the conventional LVQ based method are as follows:

1. the total number of clusters does not have to be specified a priori, although an upper limit could be set;
2. reduced sensitivity to codebook initialisation;
3. reduced sensitivity to the order in which data are presented to the clustering network;
4. the potential for computational savings since new cluster processing elements are only created as needed.

The last advantage largely relates to a serial implementation of the technique, however, as opposed to the preferred implementation that uses a parallel processing architecture. It is also worth noting that the ALVQ algorithm is general enough to find application in areas of signal classification other than the deinterleaving problem described here.

In the Introduction we remarked that the LVQ based approach to cluster analysis is related to the classical K-means algorithm. The ALVQ algorithm proposed here bears quite a strong resemblance to the version of K-means discussed in [3]. This version of K-means uses a single pass through the data to compute estimates of cluster means and begins by taking the first \( K \) feature vectors as the initial codebook vector updates of LVQ. It then processes the remaining data using nearest neighbour updates for the cluster means in a similar fashion to the codebook vector updates of LVQ. If \( x(n + 1) \) denotes the current feature vector under consideration and \( \bar{x}_c(n) \) the closest cluster mean, \( \bar{x}_c(n) \) is updated according to the recursive version of the sample mean estimator, i.e.,

\[
\bar{x}_c(n+1) = \bar{x}_c(n) + g_c(n+1)[x(n+1)-\bar{x}_c(n)],
\]

where \( c \in \{1, 2, \ldots, K\} \), and \( g_c(n) \) is a monotonically decreasing scalar-valued gain term defined by

\[
g_c(n + 1) = \frac{1}{j_c(n + 1)},
\]

where \( j_c(n + 1) = j_c(n) + 1 \),

and where \( j_l(n) \) is the number of feature vectors associated with the \( l \)th cluster. Otherwise,

\[
j_l(n + 1) = j_l(n) \quad \text{for} \quad l \neq c,
\]

and

\[
\bar{x}_l(n + 1) = \bar{x}_l(n).
\]

The number of clusters may also be increased depending on how far the current feature vector is from the nearest cluster mean, or pruned if the separation between two cluster means is less than a specified threshold distance.

The K-means algorithm initialised with \( K = 1 \), is therefore quite similar to the proposed ALVQ method. The only significant difference being that the learning rate \( \alpha(n) \) is effectively "shared" across the \( I(l) \) processing elements in ALVQ, whereas in K-means, cluster means are updated using separate or independent gains \( \{g_l(n)\}^K_{l=1} \). Independent gains could also be incorporated into the ALVQ algorithm, however, and this could become a desirable property if the neural network is to respond with sufficient plasticity to new data for cluster processing elements that are created late in the network learning process. We note, for example, that the radar signal environment is in fact likely to exhibit a time dependency, with some radar emitters undergoing mode changes and producing signals that drop in and out of the signal environment. In addition to the dynamical allocation of a new cluster codebook vector as required, codebooks could also be aged out if the associated signals are no longer deemed to be present in the signal environment.

4. Conclusions

We have investigated the suitability of an unsupervised neural network based LVQ algorithm for cluster analysis in relation to the radar pulse train deinterleaving problem. The standard LVQ method was found to have several significant shortcomings for this particular application. In response to these deficiencies, we proposed an adaptive version of the LVQ algorithm which is data driven and which adapts itself to the signal environment under consideration. The modified algorithm was highly successful in meeting the signal separation objective and does not require that one specify the number of data clusters a priori. Furthermore, the ALVQ method was found to be robust to cluster initialisation and to the order in which data were presented to the clustering network. Finally, the new method has the potential to offer computational savings.

References