INTERSPEECH 2009 BRIGHTON

Self-learning Vector Quantization for Pattern Discovery from Speech

Okko Johannes Räsänen[1], *Unto Kalervo Laine*[1], *and Toomas Altosaar*[1]

[1]Department of Signal Processing and Acoustics, Helsinki University of Technology, Finland
Okko.Rasanen@tkk.fi, Unto.Laine@tkk.fi, Toomas.Altosaar@tkk.fi

Abstract

A novel and computationally straightforward clustering algorithm was developed for vector quantization (VQ) of speech signals for a task of unsupervised pattern discovery (PD) from speech. The algorithm works in purely incremental mode, is computationally extremely feasible, and achieves comparable classification quality with the well-known k-means algorithm in the PD task. In addition to presenting the algorithm, general findings regarding the relationship between the amounts of training material, convergence of the clustering algorithm, and the ultimate quality of VQ codebooks are discussed.

Index Terms: speech recognition, pattern discovery, time series analysis, vector quantization, data clustering

1. Introduction

A conventional HMM based speech recognizer transforms input speech to a stream of continuous value feature vectors, and further, these vectors into a discrete sequence of the most likely phones by comparing them to internal acoustic models derived during the design phase. For *unsupervised language acquisition* task studied, e.g., in the ACORNS project [1], pre-defined models for atomic language units *are not readily available*, but *have to be discovered by the system itself* from the provided continuous speech material.

The first important step in our approach to this bottom-up pattern discovery task (PD) is the transformation of continuous speech signals to discrete time and discrete category units by vector quantization (VQ). For this purpose we have studied cognitively plausible ways to perform vector quantization and developed a novel straightforward method called self-learning vector quantization (SLVQ). It enhances a *basic sequential algorithm scheme* (BSAS, [2]) by incorporating adaptive cluster radii and merging of nearby clusters. The purely incremental and computationally straightforward algorithm converges quickly to a limited number of clusters with variable input data, where non-adaptive methods suffer from ever-increasing or slow convergence of cluster numbers. Despite the quick convergence, SLVQ also retains adaptivity towards totally new types of input data and creates new classes to quantify input if necessary. We show that codebooks created with SLVQ achieve comparable quality with codebooks created with the well-established k-means algorithm using only a fraction of the computational time needed in the latter. In addition, several important implications regarding vector quantization in bottom-up pattern discovery will be brought out.

The algorithm is first introduced in detail. This is followed by experiments used to determine the behavior of the algorithm, including a comparison with the k-means approach. Finally, the relationship between convergence of a cluster space and quality of the codebook is examined in a PD task.

2. SLVQ algorithm

2.1 The basic algorithm

Since one of the aims in ACORNS is to perform computational modeling of infant language acquisition, we wanted to perform cognitively plausible processing where new acoustic input changes the existing processing structures incrementally (*plasticity*). This excludes methods with batch processing of massive datasets and storage of all past acoustic inputs in detail. Also for bottom-up pattern discovery purposes, it was desirable to have an algorithm that can adapt to the properties of the input data so that hard decisions made by the user can be avoided. For example, instead of deciding on the exact number of clusters in advance, we wanted to allow some room for the data to speak for itself. We also wanted to have an open architecture where processes can be easily analyzed and understood, and therefore neural networks were not utilized.

Due to the strong emphasis on incrementality, the SLVQ algorithm is not purely divisive or agglomerative in the traditional sense (see, e.g., [3]), and resembles Kohonen maps [4], BSAS [2], and the VQ-INC-EXT algorithm [5] while being at the same time computationally extremely straightforward. It differs from, e.g., Learning Vector Quantization (LVQ) and its modifications (see, e.g., [6]) due to lack of supervision in training.

Incrementality sets some serious limitations to the way that data can be processed: knowledge about future inputs is not available to the algorithm so all classification decisions have to be performed on-line. Therefore the main functional principle of SLVQ is to take one feature vector at a time as input and compare it to the existing cluster structures. If no suitable match is found, a new cluster is created for the input. If a match is found, the input is merged to the existing centroid that becomes updated and then the original input vector is discarded (see the next subsection for a description of adaptive updating of the cluster radii).

The following pseudocode illustrates the main steps taken in the clustering process (see also fig. 1):

1. Assign first input vector $\boldsymbol{v}_i$ as the first cluster centroid with radius r_0.
2. Take the next input vector $\boldsymbol{v}_i$ and compute its distance $d_{i,j}$ to all existing clusters.
3. **if** $\forall d_{i,j} > r_j$, where r_j is radius of cluster $\mathbf{X}_j$
   create a new cluster with centroid $\boldsymbol{v}_i$ and radius r_0.
   else if $\exists\ d_{i,j} \leq r_j$,
   merge $\boldsymbol{v}_i$ to $\mathbf{X}_c$, where c = $arg_j min\ d_{i,j}$, by having $\boldsymbol{x}_c = (\boldsymbol{x}_c+\boldsymbol{v}_i)/(n_c+1)$, where $\boldsymbol{x}_c$ is the cluster centroid of $\mathbf{X}_c$ and n_c is the number of vectors already merged to the cluster c.
4. **update** all cluster radii
5. **merge** all cluster pairs $\mathbf{X}_i$ and $\mathbf{X}_j$ that satisfy $d_{i,j} < r_{\min}$ by having $\boldsymbol{x}_{ij} = (\boldsymbol{x}_i+\boldsymbol{x}_j)/(n_i+n_j)$. Go to 4 until all clusters satisfy $\forall d_{i,j} > r_{min}$.
6. **Go to** step 2.

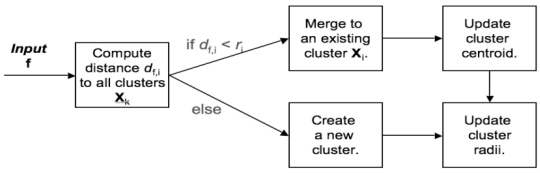

Figure 1: *Block diagram of the SLVQ algorithm*

Distance $d_{i,j}$ between cluster centroids i and j is computed using a chosen metric, so that d increases as the similarity decreases. Every time a new input vector $\boldsymbol{v}$ arrives, its distance $d_{i,j}$ to all existing cluster centroids is computed and the closest cluster i that has a larger radius than the mutual distance is chosen as the target cluster ($r_i > d_i$) and the vector is merged to the cluster centroid using a weighted average:

$$\bar{x}_i = \frac{n_i \bar{x}_i + \bar{v}}{n_i + 1} \tag{1}$$

where n_i is the number of vectors that have already been merged into the cluster. If no sufficiently close cluster is detected, a new cluster centroid $\mathbf{X}_f$ is created at the location defined by $\boldsymbol{v}$ with a radius of r_0.

Since the weighted merging causes the cluster centroids to move around in the cluster space, a case exists where two centroids drift very close to each other. If the mutual distance between two clusters becomes smaller than value r_{min} at any time, the clusters are merged together using equation (2), resulting in a single common centroid.

$$\bar{x}_{ij} = \frac{n_i \bar{x}_i + n_j \bar{x}_j}{n_i + n_j} \tag{2}$$

2.2 Adaptive cluster radii

While working with sequential clustering algorithms and varying input data like continuous speech, it becomes evident that the merging resolution needs to be relatively low with clusters of fixed radius or else the number of clusters will increase continuously as new data is introduced. In contrast, we hypothesized that a high resolution is still required at those sections of the feature space that contain large amounts of input data and where small nuances have to be differentiated, while setting a high resolution for the entire space leads to exploding (non-converging) number of very small clusters that are useless for the purpose of pattern discovery from quantized sequences.

To test this hypothesis, a method was devised that allows cluster radii to be defined adaptively: the radius r_i of a cluster i depends on the number of vectors it has received in comparison to other clusters. If a cluster $\mathbf{X}_i$ has more input vectors (n_i) than the mean of the number of vectors in all clusters of space $\mathbf{X}$ ($E\{n\}$), the radius of the $\mathbf{X}_i$ cluster is shrunk. On the other hand, if the cluster has less input vectors than on average, the radius of the cluster is increased.

$$r_{i,t+1} = \begin{cases} r_{i,t} - \Delta r & n_i > E\{n\} \\ r_{i,t} + \Delta r & n_i < E\{n\} \end{cases} \tag{3}$$

The rate that clusters adjust their radii is called the *rate of adaptation* γ and is defined as following:

$$\gamma = \frac{\Delta r}{N_v} \tag{4}$$

Variable Δr is the change in threshold (for a single cluster) and the N_v is the number of new input vectors during the change (~time), i.e., the rate of adaptation defines the amount of change in the cluster threshold for each new input vector.

In practical algorithm implementations, a counter can be used to count the number of input vectors since the last update, and if N_v is exceeded, the cluster radii are adjusted by value Δr and the counter is reset. Therefore, the value N_v is a tradeoff between computational complexity and the accuracy of the process: if N_v is high, the update is performed less frequently but with larger steps. If it is set too high, instability in the clustering process may occur due to oscillations of cluster sizes. For a majority of the experiments, radius update was performed following the presentation of each new utterance.

In order to set an allowed value range for possible cluster radii, a minimum radius r_{min} and maximum radius r_{max} are defined. Minimum radius r_{min} defines the highest possible resolution of the clustering, whereas r_{max} controls how rarely occurring inputs are treated and therefore has a large impact on the convergence of the algorithm. In principle there is also a free parameter r_0 that defines the default radius for new clusters before any adaptation takes place, but the mean of r_{min} and r_{max} was found to be a suitable choice for r_0 in the experiments. Setting r_0 too low in combination with a low γ will result in an explosion in the number of clusters since the cluster sizes and therefore cluster radii will not develop properly.

3. Experiments

3.1 Word recognition framework

The aim of the experiments was to determine how easily recurring phonetic structures are extracted from quantized time-series provided by SLVQ. Performance was tested in bottom-up pattern discovery experiments using the concept matrix (CM) framework described in [7] and [8]. Although the details of the PD algorithm are too broad to review here, the idea is to collect statistical models of co-occurrences of acoustic events (VQ label pairs) at different time distances in a simultaneous presence of a multimodal information source (e.g., a set of labels indicating what is present in the visual field). This creates associations between the acoustic input and events and items present in the surroundings of the learner, i.e., the system learns "*words*" that refer to some external entities (see also [9] for a similar learning framework). The CM system achieves comparable performance with discrete state HMMs in continuous digit recognition if visual information in training is replaced with tags indicating the presence of digit numbers in the utterance [7].

During training, the SLVQ codebook was first created using a subset of the training material. Then all material was quantized and used with the multimodal visual tags as input to the CM system. In the testing phase, only VQ sequences were shown to the CM and the system had to indicate which visual items are associated with the acoustic input.

3.2 Material and features

Speech material used in the experiments was taken from a corpus recorded as part of ACORNS project. The material consists of 2397 Finnish utterances, each containing 1-4 target words from a dictionary of 50 words the system is supposed to learn (+ numerous inflections inherent to Finnish!). Additionally, the target words are surrounded by a number of non-target words. The sentences were spoken by 4 speakers (two male), yielding a total of 9588 utterances.

As for input to the SLVQ, standard MFCC features were extracted using a Hamming window of length 20 ms with 10 ms shifts. The mean of each vector was removed and they were normalized to unit vectors. The cross-correlation of MFCC vectors was used as a distance metric in the experiments (note that cross-correlation increases with similarity in contrast to the distance notation used throughout the algorithm description).

3.3 Experiments

3.3.1 Baseline word recognition

By adjusting the $r_{\max}$ parameter, SLVQ codebooks of 5 different sizes were created (N = 600 utterances were used, learning rate γ = 0.005). As a frame of reference, standard *k-means* clustering was performed on the same data in order to produce codebooks of the same size. The CM algorithm was trained with 9000 utterances and tested with 400 previously unseen utterances (for a total of 1200 keyword occurrences). Figure 2 displays the recognition results as a function of codebook size. For larger codebooks, the difference between SLVQ and k-means is small, although k-means performs slightly better for the largest codebook. In case of smaller codebooks, SLVQ seems to code the speech more efficiently, leading to somewhat better recognition results.

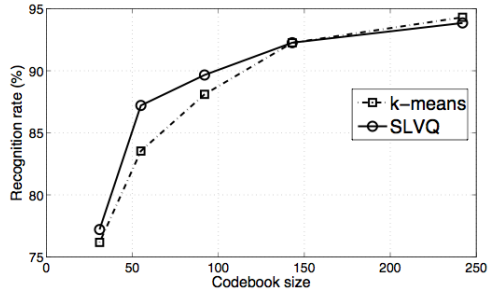

Figure 2: *Recognition rates (% words correct) as a function of codebook size for SLVQ and k-means codebook clustering.*

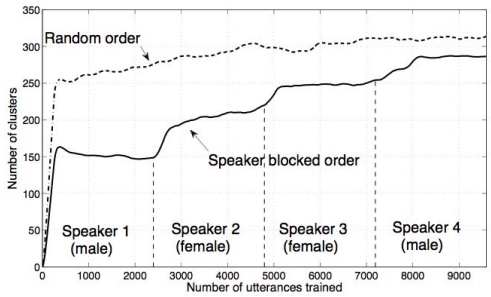

Figure 3: *Speaker blocked training. The number of clusters increases as new speakers are introduced. Randomly ordered clustering shown as a reference.*

3.3.2 Expansion due to novel input

Since the algorithm creates new clusters for input vectors that do not correspond to any existing centroids, the algorithm expands its codebook automatically if a totally new type of data is introduced. Figure 3 shows an example where 9588 utterances are trained in four speaker specific blocks. As can be seen, the number of clusters increases when a new speaker is introduced to the system. This is a useful property in unsupervised learning, where forcing of entirely new types of input into existing structures may not be desired. However, this property has not been utilized so far nor studied in depth.

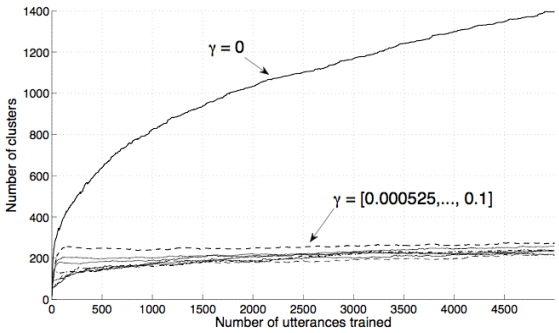

Figure 4: *Number of clusters as a function of trained utterances at different rates of adaptation γ (r_{min} = 0.6, r_{max} = 0.975, r_0 = 0.7875). A very large scale of gamma values (0.000925 - 0.01625) leads to a very similar end result after 5000 utterances. However, if the adaptation rate is set to zero, the number of clusters explodes.*

3.3.3 Effects of adaptation

If the radius adaptation is disabled (γ = 0), then all clusters will have the default radius that is the mean of $r_{\min}$ and $r_{\max}$. Figure 4 illustrates what happens in this case: although the mean resolution of the cluster space is approximately at the same level as with the spaces created using adaptation, the number of clusters becomes essentially higher and shows no signs of saturation towards the end of the training data. More importantly, the resolution without adaptation is much lower (0.7875 vs. 0.975) for the densest parts of the cluster space, resulting in a very sparse coding that can still miss distinctions between important classes of input data.

However, experiments with concept matrix speech recognition seem to indicate that adaptive resolution may not be necessary for good PD results. By setting the default threshold sufficiently low and disabling the adaptivity (γ = 0), an equal number of clusters can be obtained as with adaptive radii. When two equal size codebooks, one adaptive and one static, are compared, the recognition rate seems to be at equal levels or even better for passive clustering (e.g., 91.23 % for adaptive and 91.89 % for passive, N = 210 clusters, 5000 utterances for codebook training). What this seems to suggest, in contrast to the hypothesis motivating adaptivity in the first place, is that an extremely high resolution is not required for effective quantization of speech, so that MFCC vectors (normalized to unit vectors) originating from different phonetic units are sufficiently distributed in the space to be differentiated with lower cross-correlation values.

The next point of interest was that the adaptation might improve learning rate since the number of clusters stabilizes relatively quickly, whereas constant radii results in gradual increase in cluster numbers (fig. 4). In figure 5 this difference is further clarified: the adaptive method is relatively stable already after 100 utterances, whereas the constant radii results in a gradual increase in cluster numbers (the constant radius was set to a value that leads to the same number of clusters at 500 utterances as the adaptive process). If more data would be provided, the constant radii algorithm process would keep increasing the cluster numbers, whereas the number of clusters in an adaptive process would not be highly affected unless a significantly differing type of data would be introduced.

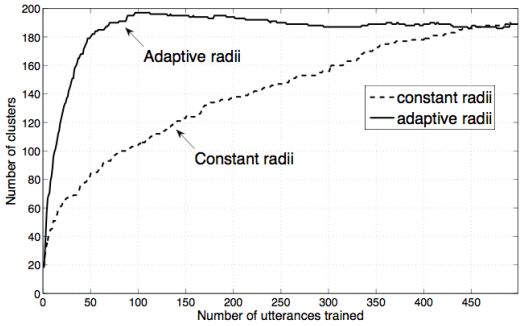

Figure 5: *Number of clusters as a function of number of trained utterances for constant and adaptive radii clustering ($r_{default}$ = 0.69 for static and r_{min} = 0.6 r_{max} = 0.975, γ = 0.008 for adaptive) when a codebook of size 188 is created.*

The first recognition test with these codebooks showed that when only 500 randomly chosen utterances were used for training of the VQ codebook, adaptive clustering performed clearly better than passive clustering, yielding a 92.72 % versus 90.23 % recognition rate with 188 clusters. However, when the amount of training material was further limited to 100 randomly chosen utterances (peak value of adaptive clustering in fig. 5), producing a total of 208 clusters, the recognition rate was now 91.06 % in the adaptive case. When the default radii r_0 was set sufficiently tight so that an equal number of clusters were created in the passive condition, the recognition result was 91.72 %, although the number of clusters was again far from stable. When an equal sized (N = 208 clusters) codebook was created from the same 100 utterances using the standard k-means algorithm, the recognition rate was 91.89 %.

Taken together, these results seem to point to a number of important implications. Firstly, the stability in the number of clusters is not an indicator of the quality of the codebook for pattern discovery purposes like in the CM algorithm [7]. This should also apply for NMF [9] due to similarities in how co-occurrence statistics are used. Secondly, the effect of the amount of training material used to create the codebook seems to be very small with a limited number of speakers, since recognition rates are not highly affected whether 100 or 5000 utterances are used. Thirdly, the nature of the quantization method itself does not seem to play a big role in the ultimate PD task. SLVQ and k-means both result in very similar overall performance, although the former works in a purely incremental basis without any global error measure, whereas the latter minimizes the global quantization error in a batch process given the desired number of clusters.

4. Conclusions

A novel method called self-learning vector quantization (SLVQ) for quantization of multi-dimensional data was introduced. The method is especially designed for incremental learning problems where the size of the codebook and the amount of input material is difficult to determine beforehand but when some resolution limits are known. In the clustering process, new clusters are created if they are not sufficiently similar to existing ones. Adaptation helps to build an efficient coding of the input when the data is unequally distributed in the cluster space, making the pruning of small clusters unnecessary and leading to convergence of the cluster space. However, it is noteworthy that though the idea is to avoid hard decisions and therefore the algorithm does not require the number of clusters to be pre-specified, it is still necessary to set several parameters that indirectly affect the number of clusters. This has to be done by either a user or a system dealing with SLVQ output in order for the algorithm to achieve desired performance.

The algorithm is computationally efficient. Only one input frame and the existing cluster centroids have to be stored in memory at any one time. Since the classification decisions are made on-line, the number of computations per each input frame is low and does not increase as a function of input frames. In principle, the algorithm can cluster infinite amount of data. For example, batch mode clustering of 1000 utterances into a codebook of size 250 takes approximately 94 minutes for a standard k-means algorithm running in a MATLAB environment using fast C++ routines for vector distance computations, whereas an incremental SLVQ codebook of the same size takes only 65 seconds to create (tested on a 4 x 2.66 GHz Intel Xeon).

However, the obtained results from pattern discovery experiments are somewhat ambiguous and difficult to interpret. In principle, the SLVQ algorithm performs at the same level as the theoretically more sophisticated k-means algorithm. On the other hand, the quality of the clustering does not seem to be greatly affected by the manner in which clustering is performed when measured from the perspective of pattern discovery from VQ sequences. Adaptivity, the amount of training material, and even the blind incrementality versus batch mode optimization do not seem to cause significant differences. Mainly the overall size of the codebook has a significant impact on the quality of the learned patterns and therefore on the recognition accuracy. Since the SLVQ algorithm was designed for the pattern discovery task used in this paper, it may be that some other experimental framework would better contrast the suitability of the SLVQ algorithm as a general incremental classification mechanism to other existing approaches.

Acknowledgements

This research is funded as part of the EU FP6 FET project Acquisition of Communication and Recognition Skills (ACORNS), contract no. FP6-034362.

References

[1] Acquisition of Communication and Recognition Skills (ACORNS). EU FP7 FET Project. http://www.acorns-project.org

[2] Theodoridis, S., & Koutroumbas K., "Pattern Recognition", Academic Press, San Diego, California, 1998

[3] Liao, W., "Clustering of time series data – a survey. Pattern Recognition", Vol. 38, pp. 1857-1874, 2005

[4] Kohonen, T., "Self-Organizing Maps", 2nd extended ed., Springer, Berlin, Germany, 1995

[5] Lughofer, E., "Extensions of vector quantization for incremental clustering", Pattern Recognition, Vol. 41, pp. 995-1011, 2008

[6] Kohonen, T., "Improved versions of learning vector quantization", Proc. Neural Networks, vol. 1, pp. 545-550, 1990

[7] Räsänen, O. J., Laine, U. K., and Altosaar, T., "A Noise Robust Method for Pattern Discovery in Quantized Time-Series: the Concept Matrix approach", Proc. Interspeech, Brighton, England, 2009.

[8] Laine, U. K., Räsänen O. J., Altosaar T., Driesen J., Aimetti G. & Henter G.: "Methods for enhanced pattern discovery in speech processing", ACORNS project deliverable, http://lands.let.ru.nl/acorns/documents/, 2008

[9] Van hamme, H., "HAC-models: a Novel Approach to Continuous Speech Recognition", Proc. Interspeech, Brisbane, Australia, 2008