THESIS

ANALYSIS OF LVQ IN THE CONTEXT OF SPONTANEOUS EEG SIGNAL CLASSIFICATION

Submitted by

Daniel Kermit Ford

Department of Computer Science

In partial fulfillment of the requirements

for the degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer, 1996

COLORADO STATE UNIVERSITY

November 6, 1996

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY DANIEL KERMIT FORD ENTITLED ANALYSIS OF LVQ IN THE CONTEXT OF SPONTANEOUS EEG SIGNAL CLASSIFICATION BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Adviser

Department Head

ABSTRACT OF THESIS

ANALYSIS OF LVQ IN THE CONTEXT OF SPONTANEOUS EEG SIGNAL CLASSIFICATION

Learning Vector Quantization (LVQ) has proven to be an effective classification procedure. Since its introduction by Kohonen in 1990 several extensions to the basic algorithm have been proposed. This paper investigates what and how LVQ *learns* in the context of EEG signal classification. LVQ is shown to be comparable with other Neural Network algorithms for the task of classifying electroencephalograph (EEG) signals, yielding approximately 80% classification accuracy for three out of the four subjects tested when differentiating between two different mental tasks.

The best classification accuracy was obtained with unnormalized, sixth-order autoregressive, AR(6), coefficients derived from raw, unfiltered EEG signals. The LVQ2.1 algorithm outperformed any of the other traditional LVQ algorithms tested, yielding a slightly higher classification accuracy than the LVQ3 algorithm. The highest classification accuracy for differentiating between two tasks was obtained using 16 codebook or reference vectors per task and a learning rate, α, of 0.1. The value of the *window width* parameter had no effect on the classification accuracy of LVQ2.1 The window width parameter specifies the width of a window centered around the hyperplane separating the two reference vectors to be updated. Reference vector updates only take place if the data vector currently being considered lies in the area defined by the window.

Initializing reference vectors at random data points resulted in an insignificantly higher classification accuracy than using K-means to initialize the reference vectors. Using the OLVQ algorithm as part of the initialization procedure did not affect the overall classification accuracy. LVQ2.1 exhibited only a moderate degree of over-fitting through the use of an excessive number of reference vectors and showed no signs of over-fitting through excessive training.

It is well known that LVQ2.1 does not continue over time to approximate the class distributions nor the optimal Bayesian decision boundary. This study illustrates this effect in the context of spontaneous EEG signals as well as in a simple artificial problem that provides a clearer insight into this phenomenon. Explanations for this phenomenon are offered based on probabilistic arguments as well as theoretical arguments related to the gradient derivations. The paper concludes with identifying theoretical difficulties associated with LVQ and the differences between LVQ classification and optimal Bayesian classification.

Daniel Kermit Ford
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
Summer, 1996

ACKNOWLEDGEMENTS

I wish to thank my advisor, Dr. Charles Anderson, for his dedicated guidance, encouragement and teachings throughout the completion of this degree. I am also grateful to Dr. Darrell Whitley and Dr. Michael Kirby for serving in my committee. I wish to thank Dr. Doris Flotzinger and Dr. Claudia Diamantini for their help in implementing and analyzing Learning Vector Quantization. I also thank Erik Stolz and Saikumar Viswa Devulapalli for their help with EEG signals, and Michael Schmitt for his discussions and feedback concerning the Vapnik-Chervonenkis dimension of a vector quantization architecture.

I am grateful to Dave Peterson who contributed to this work with numerous discussions and also for reviewing preliminary versions of the manuscript. I am grateful to Zlatko Sijercic and Matt Kretchmar for discussions and their consideration in sharing limited resources. I have also benefited from discussions with Oscar Yanez-Suarez and Doug Hundley. Many other friends and colleagues contributed to this work in different ways.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This work is part of an ongoing project to find representations of electroencephalograph (EEG) signals that can be used to identify which of several mental tasks a subject is performing, the goal being to provide a paralyzed person a means of controlling a device like a wheelchair. The objective of this study is to determine the potential of using learning vector quantization as a method to classify between different spontaneous EEG signals, as well as to analyze learning vector quantization (LVQ) as a general classification method.

Previous EEG signal classification attempts are presented in so far as they relate to learning vector quantization. The classification task was limited to distinguishing between two relatively distinct mental tasks to enable greater insight into the effects of learning vector quantization.The effect of such standard signal preprocessing as filtering out frequencies above 60Hz and normalizing the data vector coefficients was studied. The classification potential of different LVQ algorithms as well as the effect of varying parameter values was explored in great detail. Empirical studies and theoretical analysis were used to illustrate and explain the well known fact that the reference vectors of the LVQ2.1 algorithm do not continue approximating the class distributions over time.

Our research found that sixth-order autoregressive, AR(6), coefficients of the EEG signals were more accurately classified when they were left unnormalized and based on the raw unfiltered signals that were not down-sampled. The specific LVQ algorithm used, the number of reference vectors, and the learning rate, were the only significant factors effecting the classification accuracy. The insensitivity of the LVQ2.1 algorithm to early stopping techniques is explained in terms of its imbalance in attractive and repulsive forces as well as the the peculiar form of the learning curve. The algorithm's unexplained insensitivity to different initialization techniques is also documented.

This study is unique in providing an in depth analysis of learning vector quantization in the context of the difficult real world problem of spontaneous EEG signal classification. The classification results obtained demonstrate that LVQ is competitive with using an artificial neural network as a method for classifying spontaneous EEG signals. Arguments describing the benefits of using a vector quantization architecture are presented, as well as theoretical reasons that the existing LVQ algorithms are likely to be able to be improved even more. Most notably, theoretical causes have been discovered to explain why LVQ2.1 does not continue over time to approximate the optimal Bayesian decision boundary. This phenomenon is related to the goal of LVQ as expressed by the formula that it is designed to approximate as well as difficulties in implementing the gradient movements derived from the formula.

The remaining chapters in this paper cover the following topics. Chapter 2 summarizes the most relevant previous research attempts in the field of EEG signal classification. Chapter 3 introduces the various learning vector quantization (LVQ) algorithms that have been developed. Chapter 4 provides the underlying theory of learning vector quantization as well as some preliminary work at formal calculations of the complexity of vector codebook representations. Chapter 5 describes the experimental method used for the classification experiments. Chapter 6 presents the classification results. Chapter 7 analyzes the positive and negative aspects of the LVQ2.1 algorithm as a vector quantization classifier. Chapter 8 reanalyzes learning vector quantization in the context of minimizing misclassification error probability.

Chapter 2

PREVIOUS EEG CLASSIFICATION RESULTS

The vast majority of the current EEG classification research has focused solely on *event related potentials* as opposed to *spontaneous EEG signals.* Event related potentials are commonly abbreviated ERP. This chapter introduces these two types of electrical brain recordings and the previous classification efforts for each.

Event-related potentials (ERP) are the EEG signals that occur in response to some specific stimulus that occurs at a precise moment in time. The signals are characterized by the onset of a particular effect of a limited duration that begins shortly after the prompt - the event that the subject has been instructed to respond to.

Spontaneous EEG signals are the EEG signals recorded throughout the performance of a particular mental activity. They are characterized by their lack of punctuality. There is no specific moment of onset of a particular effect. EEG signals are not recorded before the subject begins the mental task nor after the subject finishes. The ongoing thought process is recorded, not the changes associated with either their onset or their conclusion.

Spontaneous EEG signals are inherently more impure, and hence more difficult to classify, than event related signals. It doesn't appear possible to ascertain how well a subject is able to maintain a steady level of concentration on the task at hand throughout the duration of time that the EEG signals are measured. Questions as to the type and diversity of EEG signals that occur during lapses of concentration remain unanswered and fluctuations in the subjects concentration level over time add an extra dimension of complexity to the classification task.

2.1 Review of ERP Classification efforts

The vast majority of experiments with ERP's has been conducted in Graz, Austria under the supervision of Professor Gert Pfurtscheller at the Department of Medical Informatics, Institute of of Biomedical Engineering, Graz University of Technology and Ludwig Boltzmann Institute of Medical Informatics. A summary of the research is divided into three sections: learning vector quantization, classification algorithms versus signal preprocessing, and progress towards a *brain-computer interface* (BCI).

2.1.1 Learning Vector Quantization

LVQ is similar to the data compression technique of vector quantization, which represents the data by a smaller, limited number of codebook vectors. However, learning vector quantization is a classification method whose goal is to use the data samples to position the codebook or reference vectors in such a way that a nearest neighbor classification method will result in the maximum classification accuracy.

Since different event related potential (ERP) recordings are generally not phase-locked, the ERP data of a particular class appears more naturally as a compilation of several subclasses than as one homogeneous unit [34]. By allowing the user to specify the number of codebook or reference vectors per class, learning vector quantization is well suited to representing the phenomena of subclasses where the number of reference vectors per class symbolically represents the number of subclasses for that class.

The number of reference vectors per response to be differentiated is critical. ERP signals perform optimally with a very limited number of reference vectors: one to two per distinguishable response. Using more reference vectors causes over-fitting [12]. Variations in the values of the other parameters have much less effect on the classification accuracy. A common learning rate or step size is $0.01 \leq \alpha \leq 0.05$. A common value for the *window width* parameter is between 0.3 and 0.8. For LVQ3, the value of ϵ is commonly bounded by $0.3 \leq \epsilon \leq 0.8$ and directly related to the value of the window width [28] [10] (for a more detailed discussion of the parameters refer to Chapter 4.)

The potential of classifying spatio-temporal rhythmic cognitive activity within the alpha frequency without resorting to averaging techniques was demonstrated using absolute band power values, the squared amplitudes associated with the lower and upper alpha frequencies, 8-10 and 10-12Hz, and the LVQ2 algorithm initialized by K-means clustering [33]. The classification potential of LVQ using non-averaged data is significant since averaging assumes a stationary signal, whereas EEG signals, due to fluctuations in the state of wakefulness, attention, ..., are never stationary. Classification accuracy was generally increased when K-means initialization was performed with K = 3 as opposed to K = 2. Other important classification factors were: the number of reference vectors per class, the localization of the time interval prior to movement, nd the number and type of frequency bands (lower or upper alpha band). The classification accuracy for the three subjects tested was 64%, 74% and 85% [33].

Pregenzer developed a modified learning vector quantization method, Distinction Sensitive Learning Vector Quantization (DSLVQ) as a feature selector [39]. Its usefulness as a feature selection signal preprocessor was demonstrated on Breiman's waveform data and Kohonen's 'hard' classification task. A major disadvantage of LVQ is that all elements of the input vector have the same influence on classification. DSLVQ eliminates this disadvantage by weighting vector components according to how much they increase the classification accuracy. DSLVQ is more reliable and consistent than event-related desynchronization (ERD) difference maps in analyzing cortical activity patterns related to different brain states [34]. Another study comparing DSLVQ to a genetic algorithm (GA) demonstrated that though they perform essentially the same filtering, DSLVQ required an order of magnitude less computing time [13]. DSLVQ also provides a list of the importance of each feature which can serve as a guide for future experiments, i.e. which electrodes should be used for which EEG patterns.

2.1.2 Classification Algorithm versus Signal Preprocessing

A comparison of Back Propagation (BP), Partially Recurrent (PR) and Cascade-Correlation (CC) neural networks demonstrated the superiority of CC networks for classifying ERP [30]. The BP and PR networks performed poorly and unreliably - they didn't converge on the test data until they were *trimmed* so that the number of processing elements in the first hidden layers was reduced to two. In contrast the CC network successfully classified all data with similar architectures for all subjects: the network generated two more hidden nodes for subject three then for subjects one and two. Additionally the CC network learned twenty times faster than the BP and PR networks. The CC network performed better with an asymmetric as opposed to a symmetric sigmoid transfer function: only half the number of hidden units, 3, a little less than half the number of training epochs, 460, were required, and a slightly higher classification accuracy, 80% on an untrained test set, was achieved with the asymmetric sigmoid transfer function.

Three signal preprocessing feature extraction techniques were compared:

- A matrix of band power values of EEG data was created. The columns represented the individual channels; the rows successive time points. The mean subtracted row/column values were divided by its standard deviation.
- The mean subtracted input vector component values were divided by the features standard deviation.
- The input vector component values were linearly interpolated to the range [0,1].

A linear interpolation of the input vector component values worked best. This method resulted in greater classification accuracy than the other two methods and required fewer training epochs. The choice of a classification algorithm appeared much less important than the signal preprocessing technique used [30].

A similar study comparing a multi-layer perceptron, a partially recurrent network, a cascade correlation network, and learning vector quantization, showed no difference in the classification results of one classification algorithm versus any other [35]. This same study revealed significant differences in the classification accuracy depending on the signal preprocessing used. Feature extraction methods included:

- simple band power values, requiring only the squaring of each sample,
- Complex Demodulation, and

- Hilbert Transform.

Band power values yielded the worst results, 70-80% classification accuracy, and the Hilbert Transform yielded the best, approximately 90%. Complex Demodulation and the Hilbert Transform are methods designed to locate the envelope of the signal. Optimizing the time interval around the minimum envelope value, i.e. the maximum ERD, improved the classification accuracy by another 10-15% compared to using a fixed time interval beginning 0-5 sec after the movement stimulus.

A study on differentiating between the mental preparation involved in finger, toe and tongue movements revealed that the most reactive frequency bands were 10-12 Hz, 30-33Hz, and 38-40 Hz for finger, toe and tongue movement respectively [36]. Discrimination between right and left index finger movements using only the 10-12 Hz was possible with 83% accuracy. Discrimination between all 4 different movements yielded between 51-58% accuracy with only alpha band components and 62-70% including the gamma band. The gamma band must thus be related to planning of one or two of the finger, toe or tongue movements that were compared.

Another study comparing the classification capabilities of various combinations of self-organising feature maps (SOFM), learning vector quantization (LVQ3), back-propagation (BP), and K-means, confirmed the previous findings that the selection of relevant features was more important than the type of classifier used [32]. LVQ3 initialized with SOFM, was compared to K-means, BP, and BP initialized with K-means on classifying non-averaged ERP. The SOFM/LVQ3 combination yielded roughly 90% classification accuracy compared to approximately 87.5% for the K-means/BP combination, and 80% for the K-means and BP algorithms when used separately. The classification accuracy of the SOFM/LVQ3 combination deteriorated when the number of reference vectors per class increased beyond one or two: the algorithm appeared to over-fit the training data, creating overly complex decision borders. The BP algorithm was relatively insensitive to the number of hidden units used, but more sensitive to signal preprocessing than LVQ3. Though BP yielded a higher accuracy than K-means on the training data, their classification accuracy on the testing data was practically identical.

The use of various spectral parameters derivable from EEG signals, e.g. amplitude, frequency, phase and power, as features requires at least one second of time stationarity [32]. Since the time stationarity of ERP data is far less than one second, time invariant AR models were used. AR models were chosen over Fourier transformation because they give equivalent resolution and require fewer samples. An order five AR model (AR5) worked best. More electrode signals did not improve classification accuracy. Using all frequencies and power values as input features to the classifier yielded the poorest results. Omitting the frequency values improved classification success, and using only those power values that corresponded to the maximum energy within a pre-specified frequency band yielded the best results.

Still another study compared three signal preprocessing techniques: common average reference, local average reference, and weighted average reference [12]. Common average reference subtracts the average of the potentials of all electrodes from each measured potential. Local average reference weights the potential of up to eight neighboring electrodes by their inverse linear distance to the current electrode [15, 14]. This is an estimate the gradient of the potential field of that electrode which is an approximation to the radial source current density [37]. Weighted average reference combines the other two methods: instead of only using neighboring electrodes all electrodes are inversely linearly weighted.

Preprocessing always improved classification results; however, no significant difference was discernible from the three tested. The motor, somato-sensory, and pre-motor areas associated with the planning of movement involve different neuronal structures and systems. Electrodes located near the area associated with hand movement will not only pick up signals from this area but from neighboring and sub-cortical areas as well. With signal preprocessing, not only is this problem, known as the reference problem, solved, but the EEG spatial resolution can also be improved. Three different time resolutions of the power estimates were also compared. 125 and 250 ms time resolutions resulted in similar performance, both superior to that obtained with 500 ms resolution. The length of recording time of the EEG signals was also experimented with, and it was demonstrated that a longer recording period resulted in higher classification accuracy [12].

A comparison of a Cascade Correlation (CC) network and LVQ3 on simulated and real EEG data contrasted with previous experiments involving Hilbert Transform preprocessing [33, 30]. The highest ERD values were obtained from non-averaged EEG signals (in general, averaging reduces the amount of time-variant ERD information while improving the signal-to noise ratio). Provided that the ERD effect is contained somewhere within the sampling window, the noise-insensitive CC network treats less than optimal window positioning as an insignificant amount of additional noise. In contrast, both LVQ3 and the CC network were hyper-sensitive to the relative distance between reactive and non-reactive alpha frequency components. LVQ3 performed best with 3 reference vectors per class [31].

2.1.3 Brain Computer Interface (BCI)

It has been shown that learning vector quantization is particularly well suited as an on-line classifier primarily due to its algorithmic speed [11]. BCI I trained LVQ3 on preparatory movements of the left and right hand. Testing involved moving a screen cursor left or right based solely on the subjects mental activity. Four reference vectors per class were found to yield the highest accuracy. Feedback to the subject helped to improve classification results, resulting in approximately 75.5% correct classification [20].

BCI II is a continuation of BCI I with the addition of *foot flexions* to the left and right hand movements as a third type of movement [19]. BCI II attempts to differentiate between three distinct movements as opposed to BCI which only differentiated between two. The results are much less clear with BCI II than with BCI I. Nonetheless, the performance of all subjects was greater than random selection: more trials were classified correct than were classified incorrect. The fact that 20% of all trials were not classified at all indicates that additional improvements can be expected through further modification of the threshold. In a session where no physical activity took place, three out of the four subjects had the majority of trials correctly classified. Potential sources of improvement include [12, 19]: optimal frequency band selection (e.g. addition of the beta band), optimal electrode placement, and more sessions.

2.1.4 Summary

Feedforward, partially recurrent, LVQ and cascade correlation networks and K-means methods have been found to be able to correctly distinguish between two ERPs with roughly equal accuracy 87-90% [30, 35, 12], where LVQ and cascade correlation networks are perhaps slightly better than the others [30, 12]. K-means clustering has been shown to be an effective initialization technique for LVQ [10, 33]. LVQ has not been nearly as successful at distinguishing between three responses: the results are closer to random chance than to the 90% accuracy achieved when distinguishing between two responses [20]. ERP classification accuracy is subject dependent [33] and much more sensitive to the preprocessing technique used to extract the relevant features that comprise the data vectors submitted to the classifier than to the classification algorithm employed [30, 35, 12].

Studies have shown mean subtracted potentials to yield higher classification accuracy than raw potentials. However multiplying the effect of each potential by a weight related to the distance to the potential in question produces no clear improvement [12]. Filtering out frequencies outside of the range of 5 to 40 Hz and using order 5 autoregressive models, AR(5), are common techniques to improve classification accuracy [20, 35]. Using power values instead of frequency values has improved results, especially if only those power values that correspond to the maximum energy within a pre-specified frequency band are used [12]. Linearly interpolating the power values to fall within the range [0,1] appears to work better than mean subtracting them [30]. Comparisons of power values, complex demodulation and Hilbert transforms have shown the former to yield the worst results and the later to yield the best results [35].

2.2 Previous Spontaneous EEG Signal Classification Efforts

Pattern classification of spontaneous EEG signals produced from different mental states is complicated by several sources of noise. The surface electrodes used to record the signals introduce noise through movements on the scalp and pick up the interference caused by the signals passing through the various layers of the cranium. Muscular activity like eye blinks produce signals of much greater amplitude than those produced by the cortical activity that we would like to measure. Cognitive factors such as the variance of concentration over time are another immeasurable source of noise.

The data used for this study was obtained from recordings done previously in connection with the research of Keirn and Aunon [22, 21]. The EEG signals from seven subjects were recorded while the subjects performed each of five mental tasks. Keirn and Aunon focused on classifying between pairs of tasks using a Bayesian classification method. They were able to obtain between 70% to 100% correct classification when using a frequency-based signal representation.

Keirn and Aunon's results were promising; however, they were also very limited. Only one single quarter-second or two-second segment was chosen from each 10 second recording session. This segment was chosen such that it was free of eye blinks and near the middle of the 10 second recording session: it was assumed that the subject was most likely to be concentrating on the requested mental task during the middle of the session. A second limitation involves their use of a quadratic Bayesian classifier. A Bayesian classifier requires that the classes have a Gaussian distribution and thus do not involve any complex, nonlinear relationships.

In another study Lin, Tsai, and Liou [29] used EEG signals collected in a manner similar to that of Keirn and Aunon. Lin, et al., used Kohonen's self organizing map (SOM) algorithm [24] to separate the collection of EEG signals into five clusters of similar patterns associating each cluster with a specific task. The algorithm was trained on the data of one subject during one recording session for all five tasks. The resulting map was used to classify the EEG signals from the same subject during another session and different subjects. The poor classification results of this study attest to the difficulty of this problem involving the simultaneous classification of five tasks combined with inter subject testing. The tasks appeared to vary in terms of difficulty of classification, with the math task (Section 5.1.1) yielding the best results.

Chapter 3

LVQ THEORY

This chapter is divided into two sections. The first section introduces the theoretical foundations for the Learning Vector Quantization algorithms presented in the following chapter. The second section makes preliminary steps toward formalizing the complexity of the algorithms in an attempt to provide guidelines for the appropriate number of reference vectors necessary for a particular problem.

3.1 Approximation of Optimal Bayes Decision Boundary

Learning Vector Quantization uses a vector quantization architecture to approximate the optimal Bayesian classification boundaries. Traditional classification methods classify directly from the approximated conditional probabilities. LVQ, however, places reference vectors into the data space in such a way that a nearest neighbor classification technique can be used for classification.

This section begins with an introduction of various measures of similarity necessary for the clustering inherent in any classification technique. Different classification methods are then presented followed by an introduction of optimal Bayesian classification. The section concludes with the derivation of the gradients upon which the LVQ algorithms are based.

3.1.1 Measures of Similarity

This section introduces the formulas for the Euclidean distance metric, the Minkowski metric, a continuous valued logic metric, the correlation metric, the Mahalanobis metric, the direction cosine metric, the Tanimoto metric, and the Kullback-Leibler divergence metric.

The Euclidean distance metric [27] is the most common similarity metric for vector spaces. Consider two n-dimensional vectors, $x = (\xi_1, \xi_2, ...\xi_n)$ and $y = (\eta_1, \eta_2, ...\eta_n)$ then their similarity as measured by the Euclidean distance metric is inversely proportional to the Euclidean distance separating the two vectors:

$$d_E(x, y) = \| x - y \| = \sqrt{\sum_{i=1}^{n} (\xi_i - \eta_i)^2}. \tag{3.1}$$

The Minkowski metric [27] is a generalization of the Euclidean distance metric. The Minkowski metric equals the Euclidean metric when $\lambda = 2$:

$$d_M(x, y) = \left(\sum_{i=1}^{n} |\xi_i - \eta_i|^\lambda \right)^{1/\lambda}, \quad \lambda \in R.$$

A metric based on continuous-valued logic [27]. The logical operators *and* and *or* are translated into the respective vector equivalents *minimum* and *maximum*. Thus the following logical relation and the vector equation are equivalent:

$$\begin{aligned} (a \equiv b) &= (\bar{a} \wedge \bar{b}) \vee (a \wedge b) \\ e(\xi_i, \eta_i) &= \max\{\min(\xi_i, \eta_i), \min[(1 - \xi), (1 - \eta)]\}. \end{aligned}$$

This yields a metric whose form is similar to that of the Minkowski metric:

$$S_M(x, y) = \left(\sum_{i=1}^{n} [e(\xi_i, \eta_i)]^\lambda \right)^{1/\lambda}, \quad \lambda \in R.$$

The dot product or correlation metric [27] is well suited for dealing with Gaussian noise and comparing nearly periodic signals. A simple modification of the basic formula allows for phase shifts:

$$\begin{aligned} C &= \sum_{i=1}^{n} \xi_i \eta_i \\ C_m &= \max_k \sum_{i=1}^{n} \xi_i \eta_{i-k}, \qquad k = -n, -n+1, \ldots + n. \end{aligned}$$

The Mahalanobis metric [27] allows the vectors x and y to be mutually dependent. Any noise is assumed to be normally distributed and ψ is the defined as the inverse of the covariance matrix of x and y:

$$d_\psi(x, y) = \parallel x - y \parallel_\psi = \sqrt{(x - y)^T \psi (x - y)},$$

Related to the correlation coefficient of statistics is the Direction Cosine metric [27]. Similarity is inversely related to the degree to which the two vectors are orthogonal to one another. When the vectors are orthogonal $\cos\theta = 0$, and $\cos\theta = 1$ signifies the greatest possible similarity between x and y. Where (x,y) is defined as the dot product the cosine is calculated by:

$$\cos\theta = \frac{(x, y)}{\parallel x \parallel \parallel y \parallel}.$$

From set theory comes the Tanimoto metric [27]. Its most notable success has been in evaluating the similarity of written documents. The metric is based on dividing the number of common elements by the number of elements that are different:

$$S_T(x, y) = \frac{(x, y)}{\parallel x \parallel^2 + \parallel y \parallel^2 - (x, y)}$$

Popular in Information Theory, Physics and Economics is a metric derived from the *minimum relative entropy principle* and known as the Kullback-Leibler divergence metric [23]:

$$d_{re}(x, y) = \sum_{i=1}^{n} x_i \ln \frac{x_i}{y_i}.$$

The above are all distance metrics useful in vector spaces as examples of measures of similarity. Measures of similarity useful for strings or coded patterns include the Hamming distance, the Levenshtein distance, and the related Maximum Posterior Probability distance. Kohonen [27] justifies the use of Euclidean distance metric for LVQ based on its common use by other vector quantization methods, its computationally efficiency, and that for many applications empirical studies have shown that the classification results are quite similar regardless of the similarity metric used. Kohonen, however, does not back up the claim that the different similarity metrics have similar classification potential. And though he indirectly acknowledges that for certain applications this would not be the case, he makes the Euclidean distance metric an integral part of LVQ and does not provide any insight into how these applications for which the Euclidean distance metric and hence LVQ are sub-optimal could be identified.

3.1.2 Classification Methods

A variety of different vector classification methods are possible. Classification methods are usually associated with either one specific or a few highly related distance metrics. Since LVQ uses the popular Euclidean distance metric we focus here on the classification methods usable in combination with this metric.

- Nearest Neighbor method: The simplest classification method is the *Nearest Neighbor method* using the Euclidean distance as a measure of similarity. This method computes the distance between an unknown pattern x and all reference patterns $x_r \in X$ and classifies x based on the closest x_r to x. Note that the amount of calculation required depends directly on the number of reference patterns used [27].

- K-Nearest Neighbor (KNN) method: The KNN method is a generalization of the Nearest Neighbor method designed for the cases where the reference patterns of different classes overlap. Instead of determining the class of an unknown pattern based solely on the nearest neighbor, the unknown pattern is classified according to the class with the most reference patterns among the unknown patterns K nearest neighbors [27].

- Other methods: Many other classification schemes based on the distance metrics discussed above are possible. Classification by the vector direction based on the Direction Cosine metric and a statistical classification metric loosely related to the Mahalanobis distance metric is known as the *Parametric Classification Scheme* are but two possibilities [27].

3.1.3 LVQ approximation of Bayesian Classification

An optimal decision in the context of statistical pattern recognition is most often defined according to Bayes theory of probability. Bayesian Classification assumes that all samples come from a finite set of overlapping classes S_k where the discriminant functions are defined in terms of the conditional probability density, $p(x|x \in S_k)$, and the *a priori* probability of the class S_k:

$$\delta_k(x) = p(x|x \in S_k)P(S_k). \tag{3.2}$$

Unknown samples are optimally classified on average if the class of a sample x is determined by:

$$x \in S_c, \quad where \quad c = arg \max_k \{\delta_k(x)\}.$$

Traditional statistical classification methods approximated the values of $p(x|x \in S_k)$ and $P(S_k)$ and used the approximations to directly classify a pattern based on Equation 3.2. LVQ assigns a subset of codebook vectors to each class S_k, positions the codebook vectors based on the data vectors (i.e. patterns whose class is known *a priori*) and classifies unclassified patterns according to the class of the closest reference vector to the pattern, using the nearest neighbor rule and the Euclidean distance metric. The classification boundaries defined by this technique approximate the optimal Bayesian decision boundary to the extent that the reference vectors accurately approximate the conditional probability density, i.e. $p(x|x \in S_k)$, near the class borders [27].

The optimal Bayesian decision borders are an upper limit on the quality of the class boundaries formed by any classification algorithm. LVQ approximates these borders through the positions of reference vectors in relative close proximity to it. It is evident that the class boundaries formed by the reference vectors are optimal if the reference vectors accurately estimate the conditional probability density, i.e. $p(x|x \in S_k)$, of the data vectors near the class borders. The conditional probability density of the data vectors far away from the decision boundaries is unimportant [27].

3.1.4 Gradient Derivations

Since exact convergence limits for LVQ are not known, convergence is discussed in terms of the observed approximation of $p(x)$, or some monotonic function of it, by VQ methods. The Euclidean distance metric is used to define the error, E, in terms of absence of similarity. Let x represent a data vector and m_i for $i \leq N$ all the reference vectors. The Euclidean error is defined by,

$$E_i = \int \parallel x - m_i \parallel^2 p(x)dx.$$

The most similar reference vector, m_i, to the data vector x is defined as,

$$m_c = \min_i E_i$$

LVQ is derived from approximating a non-negative function $f(x)$ (refer to figure 3.1) based on $\delta_k(x)$. Given the optimal Bayesian boundaries dividing the vector space into classes B_k, defined by equation 3.2, the collection of all the borders, C_B, is defined by:

$$C_B = \{x | f_k(x) = 0 \text{ for all } k\} \tag{3.3}$$

where,

$$f_k(x) = p(x|x \in S_k)P(S_k) - \max_h \{p(x|x \in S_h)P(S_h)\} \quad x \in B_k \text{ and } h \neq k. \tag{3.4}$$

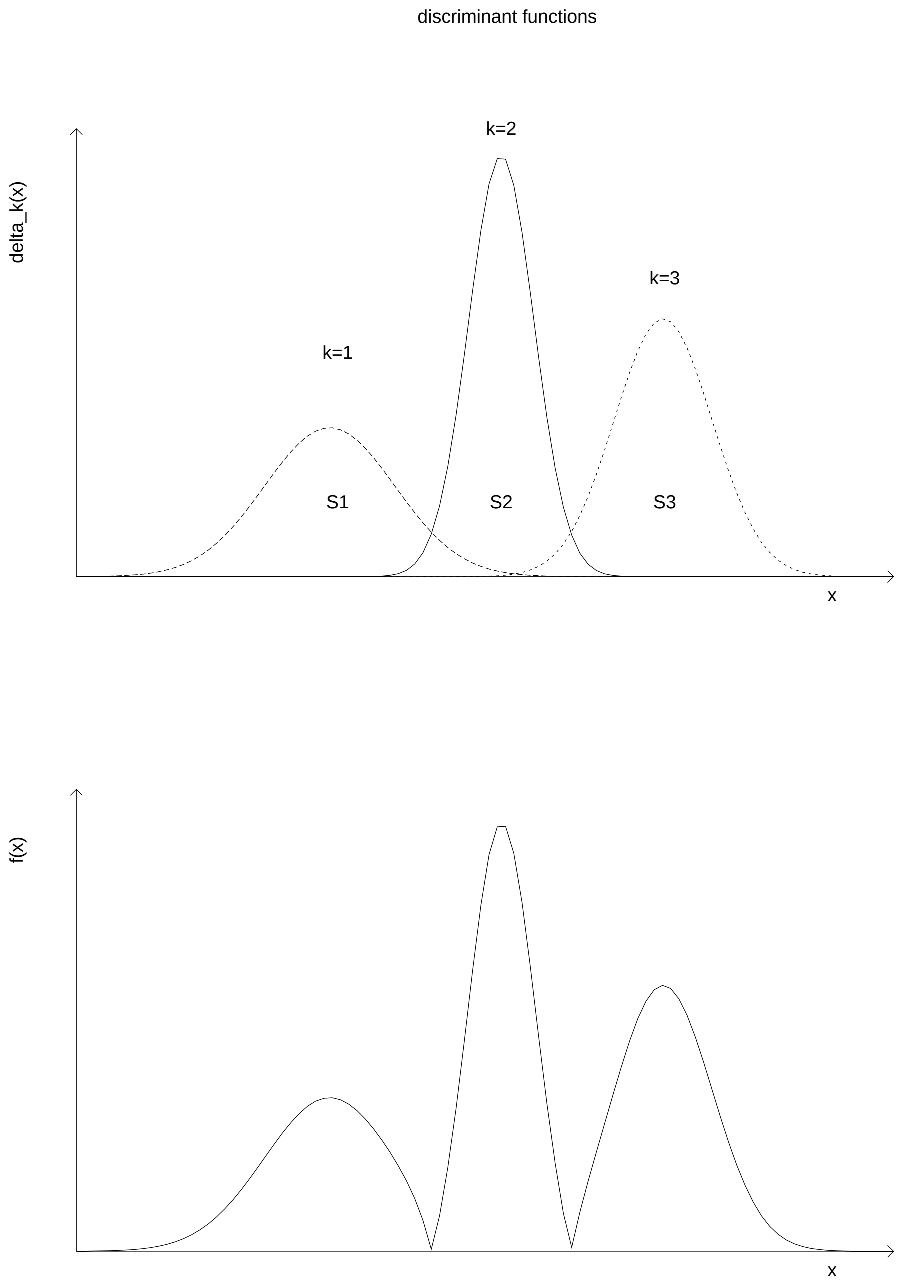


Figure 3.1: The discriminate functions $\delta_k(x)$ and the function $f(x)$ derived from them.

The gradient is obtained by minimizing the average expected quantization error. For each of the original discriminant functions, $\delta_k(x)$ the gradient is defined as:

$$\Delta_{m_i} E = -2 \int \delta_{ci}(x - m_c)\delta_k(x)dx$$

where,

$$\delta_{ci} = \begin{cases} 1 & \text{for } c = i \\ 0 & \text{otherwise,} \end{cases}$$

whereas the gradient for the function $f(x)$ derived from the original $\delta_k(x)$'s is:

$$\Delta_{m_i} E = -2 \int \delta_{ci}(x - m_c)[p(x|x \in S_k)P(S_k) - \max_h\{p(x|x \in S_h)P(S_h)\}dx$$

To illustrate the gradient at a particular time t we define $x(t)$ as simply the data vector presented to the algorithm at time t and $m_i(t)$ as the reference vector m_i at the same time t. $m_c(t)$ is therefore the most similar reference vector, as defined by the Euclidean distance metric (Equation 3.1), to the data vector $x(t)$ at time t. Then the gradient at time t corresponding to the original discriminant functions, $\delta_k(x)$, is:

$$\Delta_{m_c(t)} E = -2\delta_{ci}[x(t) - m_i(t)], \tag{3.5}$$

and the corresponding gradients associated with $f(x)$ which is used as the basis of LVQ are:

$$\Delta_{m_c(t)} E = \begin{cases} -2\delta_{ci}[x(t) - m_i(t)] & \text{for } x(t) \in S_k \\ +2\delta_{ci}[x(t) - m_i(t)] & \text{for } x(t) \in S_r, \end{cases} \tag{3.6}$$

where r signifies the label of the class with the second highest probability density function: $r = arg\max_h\{p(x|x \in S_h)P(S_h)\}$ and $r \neq k$. The following vector update rules for m_i associated with the original discriminant functions, $\delta_k(x)$, follow immediately:

$$m_i(t+1) = m_i(t) + \begin{cases} \alpha[x(t) - m_i(t)] & \text{if } i = c \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

The corresponding update rules for m_i associated with the function $f(x)$ based on the original discriminant functions, $\delta_k(x)$ are the same as those for $\delta_k(x)$ with the addition of the move along the gradient away from the data vector x:

$$m_i(t+1) = m_i(t) + \begin{cases} \alpha[x(t) - m_i(t)] & \text{if } i = c,\ x(t) \in B_k \text{ and } x(t) \in S_k \\ -\alpha[x(t) - m_i(t)] & \text{if } i = c,\ x(t) \in B_k \text{ and } x(t) \in S_r \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Note that the update equations for the function $f(x)$ rely on knowing in which Bayesian region, B_k, a data vector x is located. The Bayesian regions, B_K, must thus be defined *a priori* which is impossible since approximating the Bayesian regions is precisely the goal of the update equations.

3.2 Vapnik-Chervonenkis dimension of LVQ

To talk about the Vapnik-Chervonenkis dimension of LVQ is miss-leading. It does not matter which algorithm one uses to train this architecture. Only the possible output class of the algorithm is relevant. The output class of LVQ is the same as any VQ method: e.g. the codebook of reference vectors.

The complexity of vector quantization (VQ) methods as defined by the Vapnik-Chervonenkis dimension depends on the complexity of the codebook, i.e. the collection of reference vectors. The Vapnik-Chervonenkis dimension of a class of codebooks is the largest number of data vectors that can be *shattered* by codebooks from that class. Applying a particular codebook specifies a specific label for each of the data vectors in a set. Based on the number of data vectors in the set and the number of different possible labels per data vector, there are a limited number of different possible labelings for the set of data vectors. If every possible labeling can be realized by applying a particular codebook from the class of codebooks, then the class of codebooks shatters that set of data vectors.

3.2.1 Problem

The VC dimension of a class of {0,1} valued functions on the n dimension Euclidean space that can be computed by a vector quantizer (VQ) where the reference vectors have labels {0,1}. Initially, we will characterize this class by two parameters:

$$\begin{array}{lll} n & - & \text{the input dimension} \\ k & - & \text{the number of reference vectors} \end{array}$$

The VC dimension is the cardinality of the largest set S that can be shattered by $VQ(n,k)$, i.e. on which $VQ(n,k)$ can compute any function $f : S \rightarrow 0,1$.

3.2.2 Bounds

Trivial lower bounds are immediately evident by noting that $VCdim(VQ(n,k)) \geq k$. Choose an arbitrary set S of k different points and let it be the set of reference vectors. By assigning labels to them in all possible ways we obtain all functions on S.

Furthermore, an $n-1$ dimensional hyperplane can be represented by two reference vectors: Choose two vectors on each side of the hyperplane such that the hyperplane is perpendicular to the line connecting them and intersects it in the middle. The VC dimension of hyperplanes is known to be equal to $n+1$ [2]. This yields the lower bound $VCdim(VQ(n,k)) \geq n+1$.

Upper bounds can be calculated in a variety of ways. One possible method is to design a neural feed-forward architecture that can compute any function in $VQ(n,k)$. Such a net, however, should have hidden units that can compute analog functions because the distances to the reference vectors should be computed somehow inside the net. Upper bounds for the VC dimension of analog (e.g. sigmoidal) neural nets are known but they give only poor upper bounds in the case of $VQ(n,k)$, i.e. approximately equal to the number of weights squared [2]. Tighter upper bounds should be derivable if we stick closer to the problem and think in geometric terms.

Using an approach described in [4], a set S of m vectors can be shattered provided there exist codebooks capable of correctly encoding each of the 2^m subsets of S [3]. Assume,

- there exists a codebook capable of correctly encoding each subset of S and
- no codebook can correctly encode more than one subset of S.

Then, the VC dimension of the class of codebooks capable of shattering m vectors must contain at least 2^m distinct codebooks. Consider the class of N-vector codebooks, with k components in each vector, where each component can be represented in b bits. This class contains at most

$$\binom{(2^b)^k}{N}$$

distinct codebooks. An upper bound on the VC dimension is thus simply the number of bits necessary to represent the codebook.

$$\log_2 \binom{(2^b)^k}{N} \leq bkN.$$

3.2.3 Error estimates based on the VC dimension

Assuming optimal Bayesian classification, the expected worst case generalization error for zero error on training examples is bounded by

$$\epsilon \leq \frac{d}{m}. \tag{3.9}$$

where d is the VC dimension and m the number of data samples. This d/m ratio is known in other fields as the *inverse power law* [4]. Consider applying LVQ using a codebook of 32, 36-dimensional reference vectors, where each coefficient was encoded by 32 bits, to a problem with 480 examples (Section 5.1.4). Assume that the algorithm was able to achieve zero error on training and yielded a codebook that gave a good approximation to an optimal Bayesian decision boundary. Based on Equation 3.9 and the preceding bounds on the VC dimension of VQ methods, the algorithm would have an upper bound on the approximate worst-case generalization error of between $32/480$ and $32^3/480$, e.g. between 7% and 7,000%.

This range is not only quite large, but the lower figure of 7% is useless since it is a lower bound on an upper bound of the expected error, and the upper figure of 7,000% is above 100% and likewise useless. This confirms the findings of Cohn et al. that the theoretical bounds on the generalizable error derivable from estimates of the VC dimension of vector quantizers are far too loose to provide any practical guidance [4].

Chapter 4

LEARNING VECTOR QUANTIZATION ALGORITHMS

Learning Vector Quantization (LVQ) is a supervised Vector Quantization (VQ) method. LVQ focuses on classification [27], unlike unsupervised VQ methods such as Self Organized Maps (SOM) which are useful for discovering hidden structure or relationships between the data vectors, i.e. data clusters. LVQ divides the data space into quantization regions whose borders are hyper-planes. These regions are equivalent to the Voronoi sets in VQ.

The Voronoi sets, commonly called Voronoi tessellations, are formed by calculating the hyperplane the bisects the line segment connecting every pair of neighboring reference vectors (Section 7.1.1, Figure 7.1). The overlapping portions of the hyperplanes are truncated to leave every reference vector encapsulated in a region defined by interconnecting hyperplane segments that define the set of points equal distance between the two closest reference vectors.

The piecewise linearity of the classification border is evident from the fact that it is entirely composed of those tessellations that separate the Voronoi sets into different classes. With enough input vectors and the correct number of reference vectors, the class borders of LVQ eventually approximate the optimal Bayes decision borders [27]. There currently exists a number of variations of the basic LVQ algorithm.

4.1 LVQ1

LVQ1 was the original *learning vector quantization* algorithm developed by Kohonen in 1989 as a classification method using a vector quantization architecture in combination with labeled vectors and supervised training based on a system of rewards and punishments [27]. The algorithm is designed to approximate the optimal Bayesian decision boundary by approximating a function derived from the conditional probabilities of the differently labeled data vectors (Section 3.1.4, Equation 3.4).

Let m_c denote the closest among N reference vectors to a particular data vector x, where $c = argmin_i\{\|x - m_i\|\}$, for $i \in \{1, 2, ..., N\}$. If m_c belongs to the same class as x, m_c is moved closer to it to increase the future probability of correct classification:

$$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]. \tag{4.1}$$

If m_c belongs to a different class than x, m_c is moved farther away from it to decrease the future probability of misclassification:

$$m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)] \tag{4.2}$$

All other reference vectors, m_i where $i \neq c$, remain unchanged, $m_i(t+1) = m_i(t)$. The learning rate, $\alpha(t)$, decreases monotonically over time, where $0 \leq \alpha(t) < 1$ for all t [27].

4.2 Optimized LVQ1

Optimized LVQ1 (OLVQ1) is a computationally fast approximation to LVQ1 [27]. It is based on the observation that if independent learning rates, $\alpha_i(t)$, are assigned to each reference vector m_i, the $\alpha_i(t)$ values resulting in the fastest convergence of equations 4.1 and 4.2 can be calculated from the recursive equation:

$$\alpha_i(t) = \frac{\alpha_i(t-1)}{1 + s(t)\alpha_i(t-1)},$$

where m_c is the closest reference vector to the data vector x, $L(x)$ is the label of x, i.e. the class that x belongs to, and,

$$s(t) = \begin{cases} +1 & \text{if } L(m_c) = L(x) \\ -1 & \text{otherwise} \end{cases}$$

This computational optimization is based on the approximately optimal statistical accuracy of the trained codebook vectors when every data vector is used equally: relative to the final positions of the reference vectors, the effects of changes in their positions made due to comparisons with each data vector are approximately equal in magnitude. It is derived by first combining 4.1 and 4.2 into one equation yielding:

$$m_c(t+1) = [1 - s(t)\alpha_c(t)]m_c(t) + s(t)\alpha_c(t)x(t).$$

From this equation it is evident that the influence of the current data vector, $x(t)$, is scaled by $\alpha_c(t)$, the influence of the previous data vector, $x(t-1)$, is scaled by $[1-s(t)\alpha_c(t)]\alpha_c(t-1)$, and so on. For the effects on the reference vectors of these two data vectors to be equal their scaling must also be equal:

$$\alpha_c(t) = [1 - s(t)\alpha_c(t)]\alpha_c(t-1).$$

Extending this requirement to all t yields equation 4.2.

4.3 LVQ2

LVQ2 was created as a more accurate method of approximating the gradient movement away from a data vector by a reference vector belonging to the *runner-up* class: the class with the second highest conditional probability density at the point specified by the data vector (Section 3.1.4, Equation 3.6). The LVQ1 algorithm specifies a movement along this gradient every time that the label of the closest reference vector is different from that of the data vector regardless of whether the reference vector belongs to the runner-up class or not [27].

Note that this is only a problem when there exists more than two classes of vectors. The class of reference vector with the same label as the data vector is always assumed to be the winner. If there are only two classes then the class of a reference vector with a label different from that of the data vector is by default the runner-up class.

A secondary improvement of LVQ2 compared to LVQ1 is that LVQ2 is more computationally efficient than LVQ1. By updating the two closest, differently-labeled, reference vectors to a particular point, the two reference vectors are guaranteed to define part of a decision boundary. In LVQ2 the two closest reference vectors, m_i and m_j, to a particular data vector, x are updated simultaneously if they satisfy two constraints:

1. One of the reference vectors must belong to the same class as the data vector, m_i, and the other must belong to a different class m_j.

2. Let d_i and d_j represent the corresponding Euclidean distance between the data vector x and the corresponding reference vectors m_i and m_j. $d_i \geq d_j$ and the ratio of d_i to d_j is constrained by a *window-width* parameter $0 \leq w \leq 1$ such that:

$$\frac{d_i}{d_j} > \frac{1-w}{1+w} \ .$$

The reference vector of the same class, m_i, is moved towards the data vector according to equation 4.1 and the reference vector of the other class, m_j, is moved away according to equation 4.2 [25].

The *window width* parameter, w, creates an area (i.e. window) of a particular width defined by w centered on the hyperplane representing the decision boundary between the two reference vectors. The data vector under consideration only has an influence on the two reference vector positions via Equations 4.1 and 4.2 if the data vector lies within the area defined as the window.

4.4 LVQ2.1

LVQ2.1 is the current version of LVQ2. The only difference between LVQ2 and LVQ2.1 is that LVQ2.1 makes no distinction between which of the two reference vectors, m_i and m_j, is the closest to the data vector, x [25]. LVQ2 requires that the reference vector with the same label as the data vector, e.g. m_i, be farther away from the data vector, x, than the reference vector with a different label, m_j. The window width constraint of LVQ2 is thus modified to:

$$\min\{\frac{d_i}{d_j}, \frac{d_j}{d_i}\} > \frac{1-w}{1+w} \ . \tag{4.3}$$

This relaxation of LVQ2 was motivated by the observation that there is an imbalance between the movements toward the data vector and the movements away. LVQ2 conducts simultaneously pair-wise updates such that at every time t every movement toward the data vector $\Delta m_i(t)$ is associated with a corresponding movement away $\Delta m_i(t)$. The movements $\Delta m_i(t)$ and $\Delta m_j(t)$ are directly correlated to the distances between the corresponding reference vectors, $m_i(t)$ and $m_j(t)$, and the data vector $x(t)$, i.e. $d_i(t)$ and $d_j(t)$. Since $d_i(t) \geq d_j(t)$ from Condition (2) in Section 4.3 above, $\Delta m_i(t) \geq \Delta m_j(t)$. This results in an overall imbalance in favor of movements toward the data vectors as well a monotonic decrease in the distance $\| m_i - m_j \|$ [25].

Note that unlike LVQ1 the learning rate of the two LVQ2 algorithms cannot be optimized, i.e. it's not possible to construct an OLVQ2 algorithm. Due to inaccuracies in LVQ2's approximation of Equation 3.6, on average α_i would not decrease and the algorithm would not converge [27].

Kohonen points out that both of versions of LVQ2 may result in an asymptotic equilibrium at suboptimal reference vector positions and cites this as the motivation for the LVQ3 algorithm to ensure that reference vectors continue at least a rough approximation of the class distributions [25]. A detailed discussion of this topic is the subject of Chapter 7.

4.5 LVQ3

With LVQ2.1 the reference vectors are not guaranteed to continue approximating the class distributions and the algorithm does not continue over time to approximate the optimal Bayesian decision boundaries. LVQ3 addresses this fact by adding an update to the LVQ2.1 algorithm in the case where the two reference vectors lie within the *window* centered around the midpoint between m_i and m_j whose width is defined by w, but where they both belong to the same class as the data vector [27]. If m_i, m_j and x belong to the same class, then update both m_i and m_j identically:

$$m_k(t+1) = m_k(t) + \epsilon\alpha(t)[x(t) - m_k(t)], \quad \text{for } k \in \{i, j\}. \tag{4.4}$$

The additional parameter $0 \leq \epsilon \leq 1$ varies the step size of this move as a percentage of the standard step size, $\alpha(t)$. It is evident that LVQ2.1 is a specific instance of LVQ3 where $\epsilon = 0$. If $\epsilon = 0$, Equation 4.4 becomes $m_k(t+1) = m_k(t) + 0$. Since this equation is the only difference between LVQ3 and LVQ2.1, LVQ3 with $\epsilon = 0$ is the same algorithm as LVQ2.1.

4.6 DSLVQ

Distinction Sensitive Learning Vector Quantization (DSLVQ) was designed to reduce the high feature selection dependence of LVQ. DSLVQ addresses the problem in LVQ that all the elements, i.e. features, that comprise the reference and input vectors are given equal importance. DSLVQ is identical to LVQ3 with the addition of a weight, w_i, for each feature or vector element. This weight is dynamically adjusted during learning based on its classification contribution. The weight value is increased for features which improve classification accuracy and decreased for those that are inconsistent, frequently leading to misclassifications [39].

When calculating the closest reference vector to an input vector, DSLVQ uses a weighted distance function instead of the standard Euclidean distance function. The distance between two vectors x and y with the weight vector w is given by:

$$DSLVQdist(x, y, w) = \sqrt{\sum_{h=1}^{n} (w_h[x_h - y_h])^2}$$

$$w_h(t+1) = N\left[T\left(w_h(t) + \alpha_{fw}(t)\left(wn_h(t) - w_h(t)\right)\right)\right] \ ,$$

where the function $N[x]$ is a normalizing function defined as,

$$N[x] = x \Big/ \sum_{h=1}^{n} |x_h| \ , \quad \text{where } n = \text{number of features},$$

and the threshold function $T(x)$ is defined as,

$$threshold(x) = \begin{cases} 0.0001 & \text{if } x \le 0.0001 \\ 1 & \text{if } x \ge 1 \\ x & \text{otherwise} \ , \end{cases}$$

the weighted normal function $wn(t)$ is defined as,

$$wn(t) = N\left[d_i(t) - d_j(t)\right] \ ,$$

and the function $d_k(t)$ defines the Euclidean distance at time t between the the reference vector m_k and the data vector x,

$$d_k(t) = \| x(t) - m_k(t) \|, \quad \text{for } k \in \{i, j\} \ .$$

The $\alpha_{fw}(t)$ in Equation 4.6 is the feature weights' learning rate. Since the feature weights will be updated N times more often than the average reference vector, where N represents the number of reference vectors, $\alpha_{fw}(t)$ should generally be a factor of N less in magnitude than the α used for training the reference vectors.

DSLVQ can be used autonomously or as a noisy feature filter preprocessor for another classifier. When used autonomously, the influence of *noisy* features are reduced as their weight values decrease. When used as a preprocessor, a threshold value is applied to the feature weight values to identify the least important features and they are removed from the original input vectors, leaving only the most important features as input to the final classifier [39]. The usefulness of DSLVQ as a feature selection signal preprocessor was demonstrated on Breiman's waveform data and Kohonen's 'hard' classification task [39]. When used as a feature selection signal preprocessor DSLVQ performed as well as genetic algorithms [13].

4.7 DVQ

Dynamic Vector Quantization (DVQ) guarantees that the closest reference vector to a data vector of the same class will not be farther away from the data vector than α. Whenever the distance between the two vectors is greater than α a new reference vector is created and initialized to be at most α away from the data vector [38]. DVQ methods are the only LVQ methods able to actively identify isolated pockets of data vectors of the same class. Other LVQ methods rely on good initializing techniques to identify these clusters. By consequence, however, DVQ has a greater sensitivity to outlying data points caused by noise.

Poechmueller [38] adds pruning methods to DVQ to eliminate excess reference vectors. The first method, *cleaning*, addresses over generalization, retaining only those centrally located reference vectors of a class, by requiring a reference vector to have at least a minimum percentage of its neighbors which belong to the same class as itself. The second method, *reduction*, addresses the efficient use of reference vectors, eliminating any reference vectors whose Voronoi tessellation does not contribute to defining the class boundaries.

4.8 MEP

The Minimum Error Probability (MEP) algorithm is the result of Diamantini's observation [8] that the underlying goal of LVQ1 is not precisely minimizing error probability. Kohonen claims [25], p.1470 that the goal of LVQ1 is:

$$\min_{m_i} \left\{ \sum_{i=1}^{n} \int_{B_i} \| x - m_i \|^2 \, |f(x)| dx \right\} \quad i = 1, 2, ..., N,$$

where $f(x) = P(A)p(x|A) - p(B)p(x|B)$, where A is the label for reference vectors associated with the Bayesian region, B_i, and B the label of the others. Diamantini, however, shows that the actual goal is to maximize Equation 4.5 for those reference vectors with the same label as the region B_i and minimize the equation for those reference vectors with a different label:

$$\frac{1}{2} \sum_{i=1}^{N} \int_{B_i} \| x - m_i \|^2 \, f(x) dV_x + K, \tag{4.5}$$

where dV_x is the differential volume in the data space and K is a constant. Diamantini also proves that there is no guarantee that LVQ1 is able to find a stable solution [9].

That the goal of LVQ1 is not purely the approximation of Equation 3.4 (Section 3.1.4) is evident from the lack of vector labels in Equation 3.4 and their crucial role in the LVQ1 algorithm (Section 4.1) [9]. This observation is also true for all the other LVQ algorithms in this section since they all rely on vector labels and claim to be derived from Equation 3.4.

The MEP algorithm is most closely related to LVQ2.1. In MEP the two closest reference vectors, m_i and m_j, to a particular data vector, x are updated simultaneously if they satisfy two constraints:

1. One of the reference vectors must belong to the same class as the data vector, m_i, and the other must belong to a different class m_j.

2. Let x_{ij} represent the projection of the data vector x onto the boundary surface between the classification regions represented by m_i and m_j, i.e. the decision surface separating the data space into a region where the conditional probability density of data vectors of the same class as the reference vector m_i is highest, and a region

where the conditional probability density of data vectors of the same class as the reference vector m_j is highest. The reference vector values are only updated if for some small parametric constant Δ:

$$\| x - x_{ij} \| \leq \frac{\Delta}{2} \tag{4.6}$$

The reference vector of the same class, m_i, is moved towards the data vector, $x(t)$, according to an equation similar to Equation 4.1,

$$m_i(t+1) = m_i(t) + \frac{\alpha(t)}{\| m_i(t) - m_j(t) \|}[x(t) - m_i(t)]. \tag{4.7}$$

and the reference vector of the other class, m_j, is moved away according an equation similar to Equation 4.2,

$$m_j(t+1) = m_j(t) - \frac{\alpha(t)}{\| m_i(t) - m_j(t) \|}[x(t) - m_i(t)] \tag{4.8}$$

Since the term $1/ \parallel m_i(t) - m_j(t) \parallel$ cannot be absorbed into the step size, all the other LVQ algorithms in this chapter do not minimize the error probability [9]. Diamantini has thus shown that the goal of the other LVQ algorithms in this chapter is neither to approximate Equation 3.4 nor to minimize the misclassification error probability (see Section 8.4.2 for a more in depth analysis of this algorithm and its importance in the field of Learning Vector Quantization.)

4.9 Additional LVQ Algorithms

DeSieno's Conscience Learning has been combined with LVQ to yield a version of LVQ called Conscience Learning Vector Quantization (CLVQ) Though CLVQ approximates the overall data vector density more accurately than LVQ, it does not directly address the critical issue of improving the approximation to the optimal Bayes decision boundaries [38, 5]. Nevertheless, it has been reported to yield a slight improvement in classification accuracy.

Learning Vector Classification (LVC), proposed by Verleysen et al. [40], is based on the realization that reference vectors that are relatively far away from the class boundaries, whose Voronoi tessellation do not touch these boundaries, do not contribute to the definition of the class boundaries. Likewise, data vectors relatively far away from the decision boundaries are much less likely to contribute to the definition of the class boundary than are those more proximal to it. LVC selects only those vectors that are not farther than α away from the currently estimated class borders. Data vectors that fall outside the region are simply ignored, and reference vectors falling outside the region are replaced by new reference vectors randomly initialized to fall within the region. By gradually decreasing the value of α, and hence the size of the region encompassing the estimated decision boundaries the resulting reference vectors positions are optimally positioned to be in close proximity to the decision boundaries.

Chapter 5

EXPERIMENTAL METHOD

This chapter describes the method used to obtain the results presented in the following chapter. The first section describes the data collection method. This section includes definitions of the two mental tasks that are to be differentiated, the procedure used for recording the EEG signals, including the conditions the subject was placed in, the length and number of recording sessions, the positioning of the electrodes, the sampling frequency, and the detection and handling of artifacts, i.e. eye blinks. The AR modeling section describes the specifics of the AR model used to pre-process the raw EEG signals and the motivations for using this model. The final section describes the technique used to divide AR coefficients from different recording sessions up into multiple training, cross-validation, and testing data suites.

The second section describes the results of experiments on common preprocessing techniques. Two signal preprocessing techniques were experimented with: normalizing the AR coefficients, and low pass filtering combined with down-sampling the raw EEG signals before AR modeling. Since both of these common preprocessing techniques resulted in poorer classification accuracy, neither were used for the remainder of the classification experiments.

The final section describes the principle suite of experiments conducted whose results are presented in the following chapter. The parametric search method is presented, as well as the motivation for conducting each set of experiments.

5.1 Data Collection

The data consists of ten sessions of ten seconds each per subject recorded on two separate days. The coefficients from an AR(6) model of the original signal were calculated and then balanced so that corresponding sessions of different tasks contained an equal number of data samples. This data was then divided up into separate training, cross-validation and testing sets.

5.1.1 Mental Tasks

The two mental tasks described below were selected from among the five tasks used in a previous study [22]:

Base The subject was to relax as much as possible with their eyes open. This baseline task was designed to measure the alpha waves and asymmetries naturally present across different electrodes and EEG bands during relaxation.

Math The subject is given a non-trivial multiplication problem to solve, e.g. 49 times 78. The problems do not repeat and are complicated enough that the answer is not immediately evident. The subject is instructed not to vocalize or make any extraneous movements while working and to verify at the end whether or not they were able to solve the problem.

5.1.2 EEG Signal Recording

Subjects were seated in a sound-proof, dimly-lit, room. EEG signals were recorded from six electrodes in standard positions referred to as C3, C4, O1, O2, P3, and P4 in the 10-20 system of electrode placement [17]. The electrodes were connected to Grass 7P511 amplifiers that bandpass filtered the signals at 0.1-100 Hz. The EEG signals were sampled at 250 samples per second and digitized with 12 bits of accuracy.

Data was recorded from each subject for a duration of 10 seconds while the subject was performing a single task with their eyes open and each task was repeated five times per session. Each session resulted in 250 samples/second x 10 seconds x 6 channels, or 15,000 values. Most subjects attended two such sessions which were recorded on separate weeks. Repeating the tasks several times on different occasions gives a measure of intra-subject variability. Each 10 second recording session was divided into half-second windows with a quarter-second of overlap.

All half-second windows of data recorded during an eye blink were removed from the data set. Eye blinks were detected by means of a separate channel of data recorded from two electrodes placed above and below the subject's left eye. An eye blink was said to have occurred if a change in magnitude greater than 100 μVolts occurred within a time of 10 milliseconds.

5.1.3 AR Modeling

Parametric methods are often used to model EEG signals to reduce the effect of noise in the signals on further analysis. AR models are a common technique. The vast majority of work in AR modeling of EEG has involved scalar AR models of single channels of EEG data. Jansen [16] provides a brief tutorial on applying scalar AR methods to biological signals.

Two previous studies reported good performance with order six, scalar AR models [22, 1]. To facilitate comparison with these earlier studies, order six, scalar AR models were also used for these experiments. The scalar AR coefficients were calculated using the Burg method. Coefficients were calculated for each of the half-second windows that were free of eye blinks.

For all trials, the order one coefficients have the largest magnitude, showing that the linear relationship between the signals' values and their past values is strongest, and usually positive, for the immediately preceding values. The relationship at larger orders is relatively weak.

5.1.4 Training and Testing Data Sets

The removal of recording segments with eye blinks resulted in a different number of data vectors per recording session per task. For unbiased classification results it is helpful to have an equal number of data vectors for each class. Inter-trial classification required the pairing of data vectors resulting from a single recording of the base task with an equal number of data vectors resulting from a single recording of the math task. In the case where the number of vectors in each of the two tasks designated to form a pair were not equal, *excess* vectors from the end of the task with the smaller number of eye blinks were

Subject	Number of Data Vectors
1	277
3	238
4	208
6	241

Table 5.1: Total number of data vectors available per subject per task.

removed until the two tasks contained the same number of vectors. The number of data vectors available per individual task ranged from 35 to 4. The total number of data vectors available per subject per task is given in Table 5.1.

The data were used to create thirty independent experimental sets using selection without replacement: one of the ten task pairs was selected for testing, a different task pair was selected for cross validation, and the remaining eight task pairs were used for training. The classification accuracies reported are the average of the classification accuracies over all thirty sets.

In addition to the trial independent training sets, separate training sets were formed where the data from all trials for a particular subject were mixed together. The mixed method combined all the data vectors from each of the 8 pairs of 2 tasks for a particular subject. Selection without replacement was used to randomly select one tenth of the vectors for testing and an equivalent amount of the remaining data vectors for cross validation, leaving the remaining 80% for training.

5.2 Signal Processing

Preliminary tests were conducted using two common signal processing techniques, normalization and low-pass filtering, to determine the best signal representation for the analysis of the classification results using LVQ. Normalizing the data and filtering out frequencies above 60Hz significantly decreased the classification accuracy of LVQ. This section describes the normalization and filtering methods used as well as the basis from which these conclusions were reached.

5.2.1 Normalizing AR Coefficients

The common practice of normalizing the values of the individual features of data vectors is generally inappropriate for LVQ methods. LVQ calculates relative distances between vectors based on a Euclidean coordinate system. In AR models the values of the coefficients reflect the degree of correlation to the original data. For LVQ methods this difference acts as a weighting factor, giving more importance to features that more accurately represent the original data and less importance to those that less accurately represent the original data. Normalizing the data eliminates this weighting factor. Normalizing the AR coefficients reduced the classification accuracy across all subjects by roughly 5%.

In contrast to other neural network methods, LVQ is well-suited to take advantage of the relative differences in magnitudes of the AR coefficients produced by the AR model. Normalizing the data eliminates information inherent in the original AR coefficients related to the amount of representational bias the AR model introduced. LVQ may thus be less sensitive to the order of the AR model used than other neural network methods. The more unbalanced the AR coefficients are the greater the improvement in classification accuracy one would expect to see by using LVQ as opposed to other neural network algorithms that rely on normalized data.

5.2.2 Low pass filtering

Signals of 60Hz as well as those above 60Hz are generally considered artifacts related to muscle movements such as eye blinks when measuring cortical EEG signals. Experiments were run comparing unfiltered data to data filtered to exclude all frequencies of 60Hz and above (Table 5.2). FIR low pass filtering was combined with *down-sampling* by a factor of 2, reducing the effective sampling rate to 125Hz.

Filtering out signal frequencies of 60Hz and above significantly decreased classification accuracy by between 1% and 17% depending on the subject. Filtering decreased the classification accuracy more when the data from separate trials was pooled together than when the data was segregated by trials. In general, the greater the increase in accuracy offered by pooling the various trials together, the greater the decrease in accuracy if the data is low-pass filtered.

Classification Accuracy				
	Un-filtered		Filtered	
Subject	Seg.	Mix.	Seg.	Mix.
1	78% ± 3%	90% ± 1%	72% ± 3%	75% ± 2%
3	73% ± 7%	91% ± 1%	60% ± 4%	74% ± 2%
4	82% ± 4%	85% ± 2%	79% ± 4%	84% ± 1%
6	82% ± 3%	88% ± 1%	73% ± 3%	82% ± 2%

Table 5.2: Classification accuracy of AR(6) coefficients from raw EEG signals versus classification accuracy of AR(6) coefficients from 60Hz low pass filtered EEG signals that were down sampled by a factor of 2.

The results of classification with data from separate trials mixed together was approximately 10% better - 85% versus 75% that results when the data was segregated by trials. There was also significantly less variance in the results from one subject to another when the data from all the trials were mixed together. Using a low pass filter on the data where all trails are pooled together caused a 10% to 15% decrease in accuracy for subjects 1 and 3, a 5% decrease in accuracy for subjects 6 and had no significant effect on subject 4. Using a low pass filter on the data segregated by trials resulted in a 5% to 10% decrease in accuracy for all four subjects.

This peculiar decrease in classification accuracy after low pass filtering confirms the similar findings of Anderson et al. [1] where a neural network with forty hidden nodes was used to classify the same data. Anderson et al. reported $53.2\% \pm 0.6$ correct classification on the raw unfiltered data and $51.1\% \pm 0.6$ correct classification after the data had been filtered and down-sized as described above.

5.3 Overview of Experiments

A preliminary search for the effect on the classification accuracy of varying the number of reference vectors from between 2 and 16 was conducted using OLVQ on segregated and mixed combinations of the raw AR(6) coefficients of subject 3 for up to 50 epochs (an epoch is defined to equal training on each of the 400+ training data vectors available for each of the 30 data sets.)

The same procedure was repeated for LVQ1 for 100 epochs using OLVQ as an initialization procedure for between 5 and 10 epochs and a learning rate of 0.05 (see recommendations in Section 2.1.1 and [10, 28]) and varying the number of reference vectors between 2 and 256. 64 reference vectors yielded the best results, and the alpha rate was varied between 0.0001 and 1.0 using logarithmic steps. A relatively large learning rate, 0.1, produced the highest classification accuracy, which was significantly higher than that obtained with OLVQ alone.

Similar experiments were conducted using LVQ2.1, initially varying the number of reference vectors using a learning rate of 0.1 and a window width of 0.5, then varying learning rate, and then varying the window width between 0.1 and 1.0. 32 reference vectors, a learning rate of 0.1 and a window width of 0.9 yielded the best results.

Parameter searches using the LVQ3 algorithm beginning with $\epsilon = 0.3$ and varying the value of ϵ between 0 and 1.0 confirmed the values of the best parameters found when using the other LVQ versions and yielded the best classification accuracy when $\epsilon = 0$ (LVQ3 with $\epsilon = 0$ is identical to LVQ2.1).

Additional experiments, still concentrating on subject 3, using LVQ2.1 and various combinations of reference vectors, learning rates, window widths near the best values obtained while training for up to 1000 epochs confirmed the optimal values found previously. The same parameter search conducted on the data from the other subjects yielded better results with slightly fewer reference vectors and a slightly lower learning rate.

Experiments with combinations of normalized and filtered data per the procedure in the previous section using the LVQ2.1 algorithm, 16 reference vectors, a learning rate of 0.08 and a window width of 0.8 yielded poorer classification accuracies than the same experiments conducted on the unprocessed data (Section 5.2.2, Table 5.2). Varying the parameters in proximity to the best values did not significantly change this result.

The experiments of the following chapter thus were all conducted using unnormalized, unfiltered, AR(6) coefficients of the data segregated by trials, each base trial paired with a math trial, and organized into training, cross-validation, and testing sets as described in the previous two sections of this chapter.

Throughout the previous preliminary experiments no significant inter-dependence was observed between varying the value of one parameter and the value of another parameter that produced the best results. The accuracy of the piecewise parameter optimization procedure described above was supported by this lack of parameter inter-dependence. The effect of varying a single parameter was clearest when the other parameters were at or near their optimal values. Separate experiments were thus conducted to methodically vary each of the parameter values independently in a way to best illustrate their effect on the classification accuracy of the LVQ2.1 algorithm (Section 6.1, Figures 6.1, 6.2, 6.3, and 6.4.)

LVQ2.1 does not continue to approximate the optimal Bayesian decision boundaries [27]. Yet, it yields a higher classification accuracy than all the other LVQ algorithms tested. The continued approximation of the Bayesian decision boundaries is only theoretically important if the final reference vector positions are used. If learning is stopped at the moment when the best approximation to the Bayesian decision boundary occurs, then it does not matter what happens afterwards. This prompted experiments with LVQ2.1 adapted to include cross-validation as an early stopping technique. The results are presented in Table 6.1. Limited parameter variations were conducted without any significant changes in the results.

The reason behind the failure of early stopping to improve the results of LVQ2.1 was found by analyzing the learning curve of the algorithm using different initial learning rates and different decay methods (see Section 6.4, Figures 6.2 and 6.6). The learning curves demonstrated the peculiar phenomenon of either a stable, non-decreasing cross-validation and testing classification accuracy after a certain time point, or a decreasing cross-validation and testing classification accuracy followed by an increase sometimes to a higher classification accuracy than was obtainable by any other means. This peculiar form of the learning curves motivated an analysis of the effect of using different initialization techniques. Neither the effect of varying the initial learning rate nor the effect of different learning schedules, nor the effect of different initialization procedures provided any insight into the cause of the peculiar shape of the learning curve associated with the optimal learning rate.

An analysis of the final positions of the reference vectors using LVQ2.1 with optimal parameters and no early stopping finally revealed some insight into the peculiar form of the learning curve and the failure of an early stopping technique to improve the classification accuracy of LVQ2.1 (Section 7.1.1, Figure 7.1). The insight into this cause is seen to be related to the failure of LVQ2.1 to continue to approximate the optimal Bayesian decision boundary over time.

Chapter 6

CLASSIFICATION RESULTS

The classification accuracies presented in this paper were obtained by averaging the classification accuracies of each of the 30 independent data sets, and the standard deviations were calculated based on the differences between the classification accuracies of the 30 independent data sets.

Data sets were comprised as described in the previous chapter from unnormalized, AR(6) models of the raw EEG signals for each subject. Except for those in the first section, the results refer to unfiltered data whose data sets were formed from blocks of data segregated by trial, as opposed to the data sets where the data from all trials for a particular subject were pooled together.

The results of an extensive parameter search are followed by a comparison of some of the more common LVQ algorithms. The superior classification accuracy of LVQ2.1 is explored in light of the fact that it does not continue to approximate the optimal Bayesian decision boundary. The final reference vector values are analyzed using Voronoi tessellations, comparing the values of the individual features, and comparing the average values of the vectors of the two different tasks. The ineffectiveness of cross validation to improve the classification accuracy is explained by the unusual learning dynamics observed. The peculiar form of the learning curve is related to the fact that LVQ2.1 does not guarantee a continued approximation over time of the optimal Bayesian decision boundary. The ineffectiveness of more elaborate initializing techniques, i.e. OLVQ and K-means, reinforces the importance of the observed learning dynamics.

6.1 Effects of Parameter Values

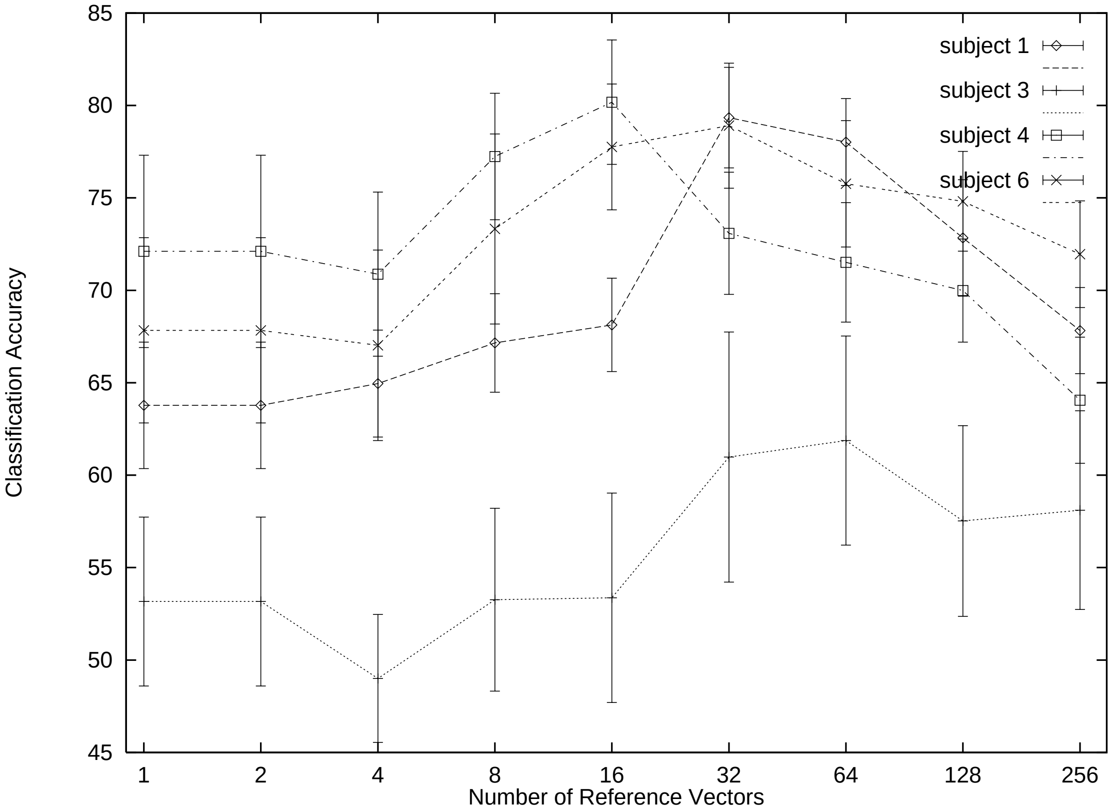


Figure 6.1: LVQ3 classification accuracy sensitivity versus the number of reference vectors for the AR(6) coefficients of the raw EEG signals of all four subjects. The remaining parameters were fixed at their optimal values— $\alpha = 0.08$, a window width equal to 0.8, and $\epsilon = 0$.

An extensive parameter search was performed using LVQ3. The number of reference vectors was varied from 1 to 250, α ranged between 0.0001 and 1.0, the window width ranged from 0.1 to 1.0, and ϵ ranged between 0 and 1.0. The graphs represent the effect of varying one parameter while the others were fixed at their optimal values—32 reference vectors, $\alpha = 0.08$, a window width equal to 0.8 and $\epsilon = 0$. Figures 6.1, 6.2, 6.4, and 6.3 show the effect of varying one parameter while keeping the other parameters fixed at their optimal value. The error bars represent 90% confidence intervals.

When combined with a *reasonable* learning rate, choosing the appropriate number of reference vectors was the most important factor in maximizing the classification accuracy. An extensive search for the optimal number of reference vectors was conducted for combinations of filtered and unfiltered data with data sets segregated by subject and data sets where the data from all the subjects were pooled together. Using different signal

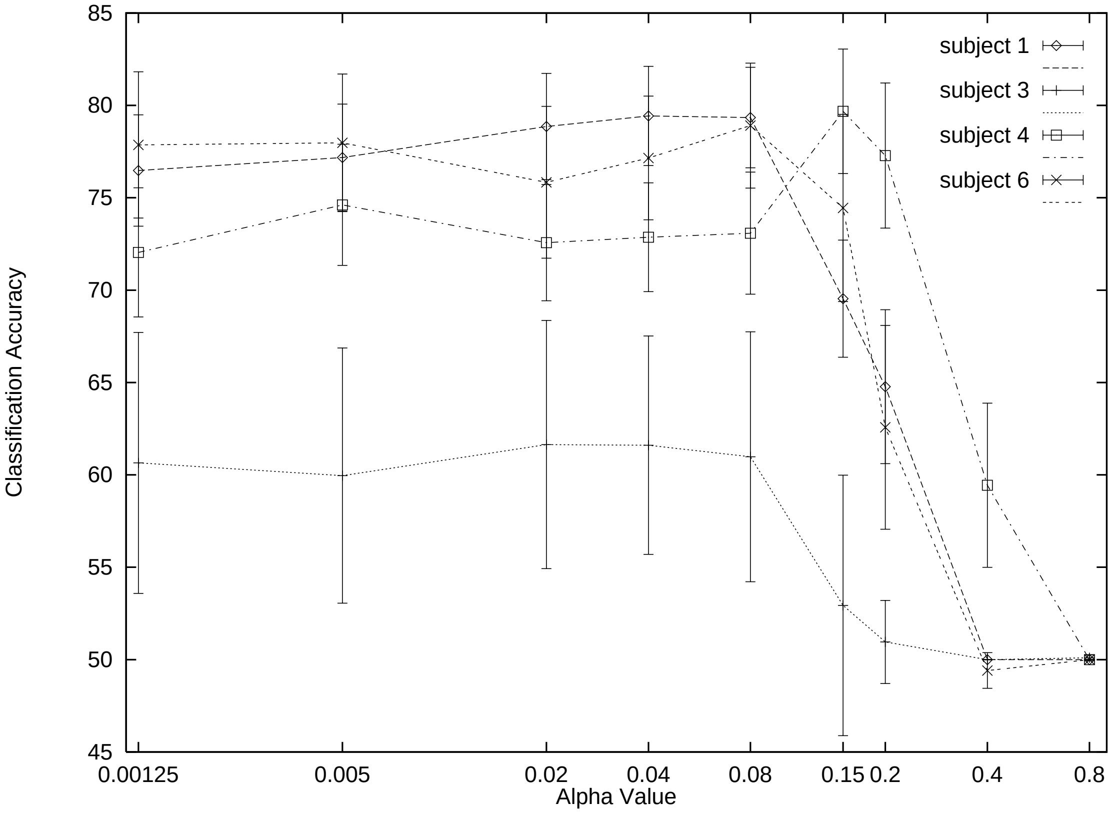


Figure 6.2: LVQ3 classification accuracy sensitivity versus the step size, α, for the AR(6) coefficients of the raw EEG signals of all four subjects. The remaining parameters were fixed at their optimal values— 32 reference vectors, a window width equal to 0.8, and $\epsilon = 0$.

preprocessing techniques in combination with data sets segregated by trials and mixed together did not significantly affect the optimal parameters.

The most significant parametric difference between LVQ classification of event related versus spontaneous EEG data is in the optimal number of reference vectors. Whereas with event related data only one to two reference vectors per distinguishable response was declared optimal [12], the spontaneous data performed best with approximately 16 reference vectors per distinguishable task. The relatively large number of reference vectors necessary for adequate classification accuracy is an indication of a more complex vector space.

The optimal number of reference vectors, 32 (Figure 6.1), is roughly 7.5% of the total number of data vectors used for training. In general the risk of *over-fitting*, defined as an eventual decrease in the classification accuracy on the test data associated with

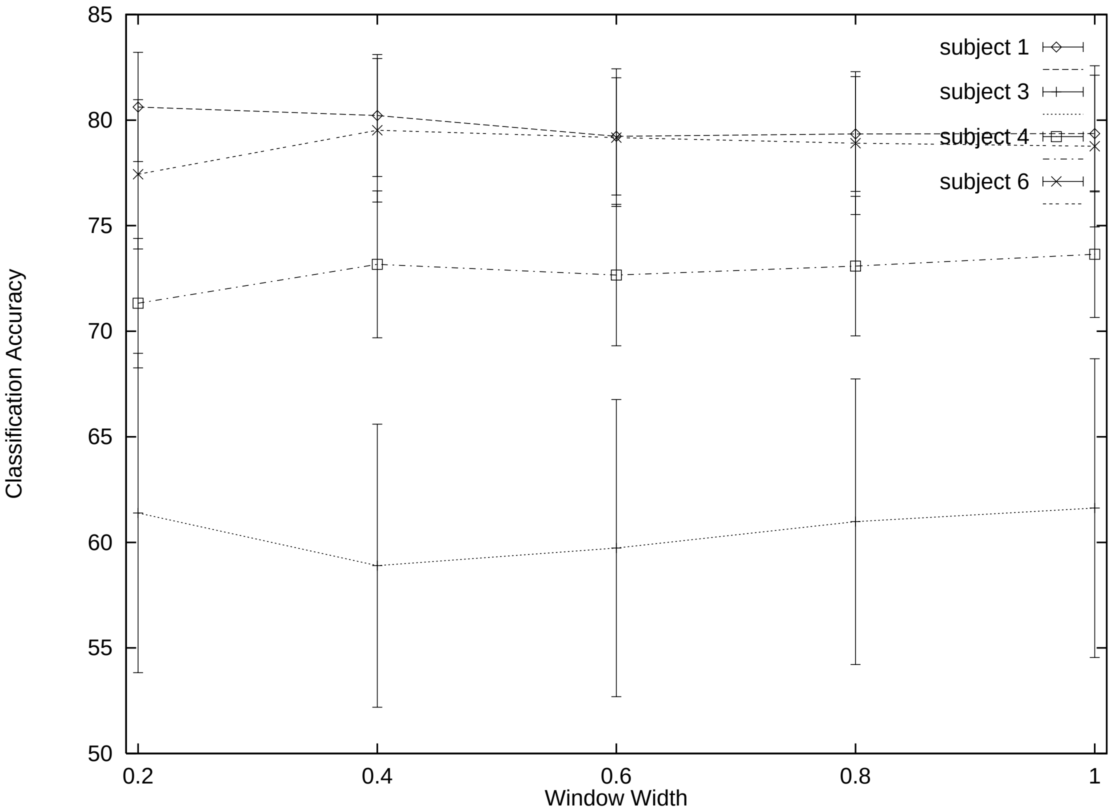


Figure 6.3: LVQ3 classification accuracy sensitivity versus the window width for the AR(6) coefficients of the raw EEG signals of all four subjects. The remaining parameters were fixed at their optimal values— 32 reference vectors, $\alpha = 0.08$, and $\epsilon = 0$.

a continued increase in the classification accuracy on the training data, increases as the number of reference vectors becomes a larger percentage of the number of training data vectors. Data from one of the subjects exhibited signs of over-fitting when more than 8 vectors per class were used. Two other subjects exhibited signs of over-fitting when more than 16 vectors per class were used. The remaining subject's signals were much more difficult to classify and did not exhibit any sign of over-fitting even with 128 vectors per class, approximately one reference vector for every two training data vectors (Figure 6.1).

There was a very noticeable decay in the classification accuracy for large learning rates, α_0. The classification accuracy of three of the four subjects began to decay for $\alpha_0 > 0.08$ and for $\alpha_0 \geq 0.4$ the results were no greater than chance, 50% (Figure 6.2). Subject 4 showed a marked preference for a higher learning rate, i.e. $\alpha_0 = 0.16$, exhibiting a sharp decay for larger learning rates with classification no better than chance at $\alpha_0 \geq 0.8$.

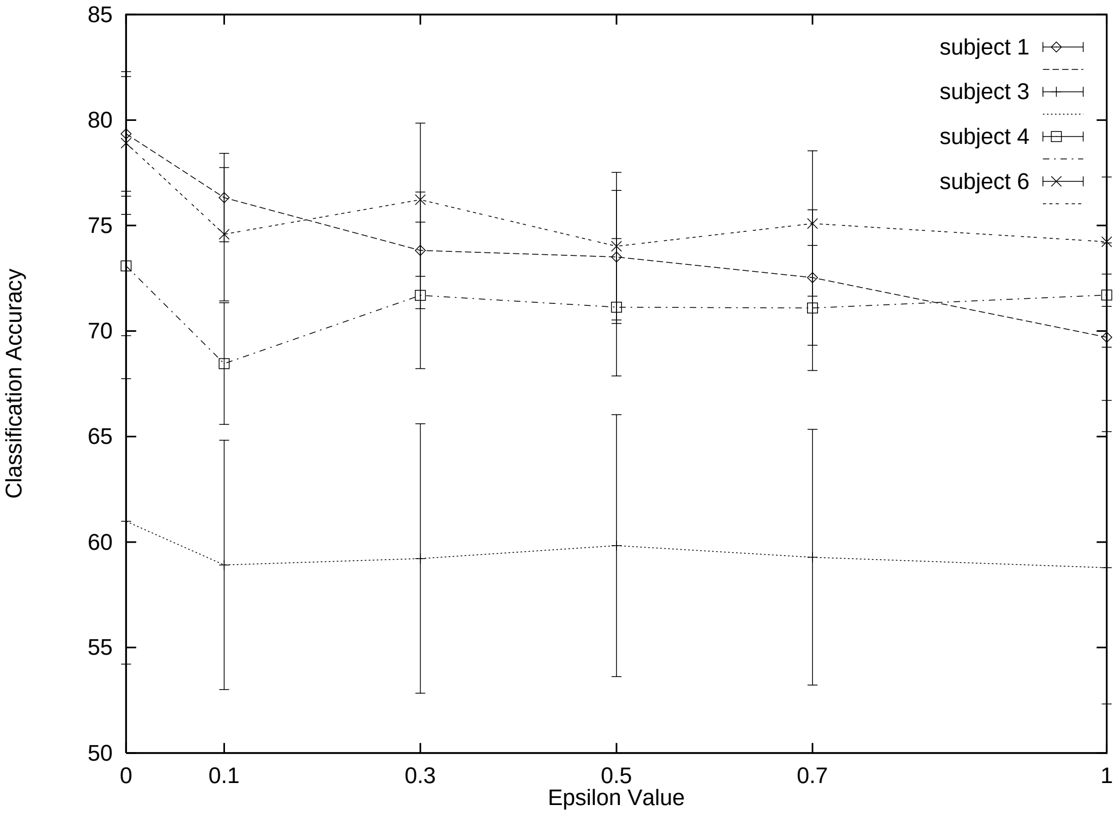


Figure 6.4: LVQ3 classification accuracy sensitivity versus the value of ϵ, for the AR(6) coefficients of the raw EEG signals of all four subjects. Except as indicated the parameter values are fixed at their optimal values— 32 reference vectors, $\alpha = 0.08$, and a window width equal to 0.8

This was the same subject that exhibited signs of overfitting earlier than the others, i.e. more than 8 reference vectors per class (Figure 6.1).

The effect of varying the window width was completely insignificant and inconsistent from one subject to another. A constant classification associated with a large variance makes changes in the values of the window width particularly unimportant. Neither were there any trends toward preferring a larger or a smaller window width and any window width yielding a slightly higher average accuracy for one subject yielded a slightly lower accuracy for another subject (Figure 6.3).

The variance in the classification associated with varying the value of ϵ was also insignificant. In this case, as opposed to variances in the value of the window width, there is a clear trend by two subjects ,subjects 1 and 6, to yield higher classification accuracies with lower values of ϵ (Figure 6.4.) All four subjects yielded the highest classification accuracy with $\epsilon = 0$ (LVQ3 with $\epsilon = 0$ is identical to LVQ2.1.)

6.2 LVQ Variations

Algorithms that only update one vector at a time, e.g. LVQ1 and OLVQ, modify all reference vectors regardless of whether they contribute to the formation of the class boundaries or not. If the two closest reference vectors to a data point are of different classes they must contribute to the definition of the decision boundary. LVQ algorithms that use such pair-wise updates are more efficient than those that update only one vector at a time. It has also been suggested without proof that pair-wise updates lead to more accurate approximations of the Bayes decision boundary [27].

None of the LVQ algorithms introduced to date explicitly ensure that all of the reference vectors that make up the class decision boundaries will be modified. This is, however, guaranteed by any initialization technique that initializes reference vector positions to the positions of data vectors in the training set.

The results of our experiments comparing the classification accuracies of some of the more fundamental LVQ algorithms differs slightly from those previously reported by Kohonen [28, 26, 27]. Our results indicated that LVQ2.1 outperformed LVQ3 by approximately 2%, which was roughly 2% more accurate than LVQ1. LVQ1 was in turn approximately 4% more accurate than OLVQ (where differences greater than 3% were statistically significant with 90% confidence).

6.3 Cross-Validation

The higher classification accuracy of LVQ3 with $\epsilon = 0$ (equivalent to LVQ2.1) when compared with LVQ3 with $\epsilon > 0$, combined with the well known failure of LVQ2.1 to continue to approximate the class distributions over time [25, 27], motivated the introduction of cross-validation as an early stopping method. Incorporating an early stopping method into LVQ2.1 would make irrelevant its failure to continue to approximate the optimal Bayes decision boundary over time. The only important criteria is that the algorithm

Effect of Cross Validation		
Subject	With	Without
1	76% ± 3%	79% ± 3%
3	61% ± 7%	61% ± 7%
4	75% ± 3%	73% ± 3%
6	76% ± 3%	79% ± 3%

Table 6.1: Classification accuracy at the end of 1000 training epochs versus at the optimal early stopping point within 1000 epochs as determined by cross-validation using LVQ2.1 with optimal parameter values as defined in Section 6.1.

at some time gives a good approximation of the optimal generalizable decision boundary. Since the position of the reference vectors corresponding to the best approximation of the optimal generalizable decision boundary are retained as the final result of training, it wouldn't matter whether the best reference vector positions occur at the end of training or sometime earlier.

LVQ2.1 consistently showed near-best results after 50 epochs. When the optimal early stopping point over 1000 epochs was determined by cross-validation the resulting classification accuracy was not any better than that obtained after training for the full 1000 epochs (Table 6.1). LVQ2.1 appears not to be subject to overfitting by excessive training.

6.4 Learning Curves

The failure of cross-validation to improve the classification accuracy of LVQ2.1 prompted investigation of the learning dynamics of the algorithm and further investigation of the effect of the learning rate α. The learning curves were investigated for initial values of alpha, α_0, from 0.001 to 0.8 and for two different scaling methods. Figure 6.5 illustrates the results of using a linear decay to zero,

$$\alpha_t = |S|\alpha_0(t_{max} - t) \quad t_{max} = 400, \quad |S| \approx 400,$$

where $|S|$ is the size of the training set. Figure 6.6 illustrates the results of varying α_t as an inverse of the elapsed time,

$$\alpha_t = \frac{C\alpha_0}{C + t} \quad t_{max} = 400, \quad C = t_{max}/100 = 4$$

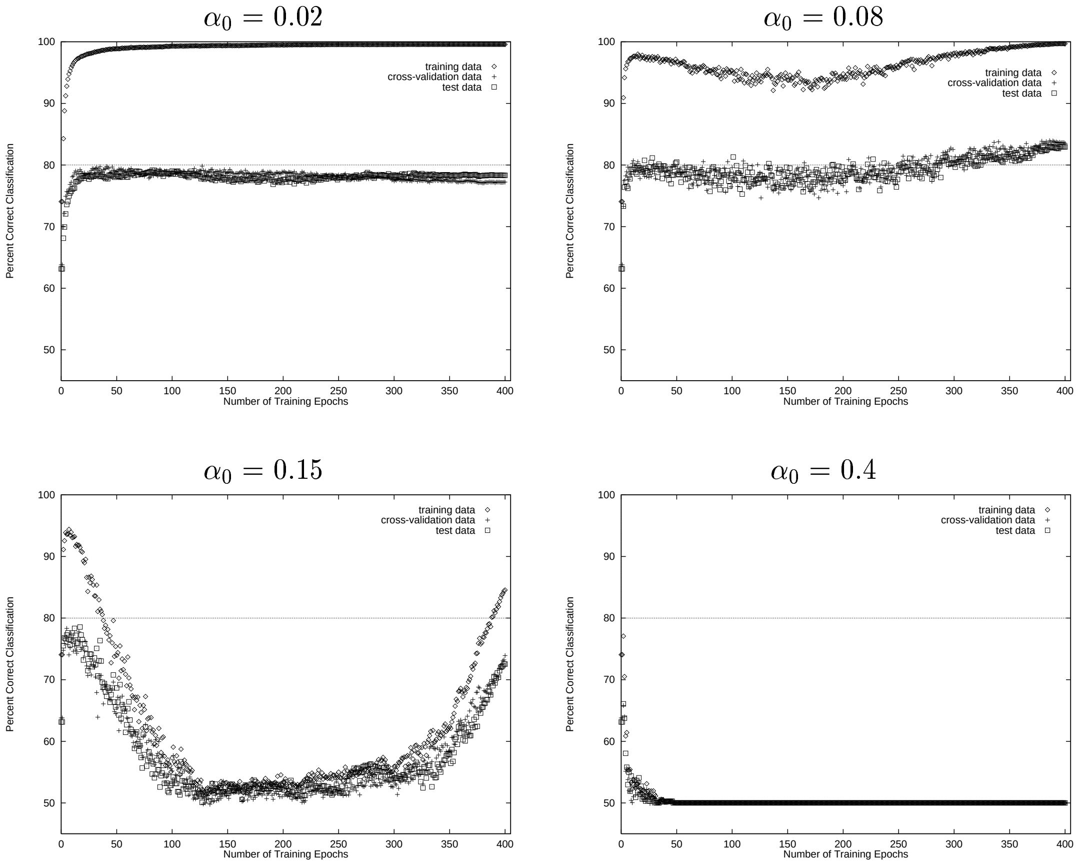


Figure 6.5: Effect of different values of α_0, where alpha decays linearly over time, on the classification accuracy of the training, cross-validation, and test data per training epoch for over 400 epochs.

where C is a constant.

The graphs in Figure 6.5 illustrate the effect on the classification accuracy per epoch for different values of α_0 when the linear decay model was used. The graphs in Figure 6.6 illustrate the same effects for different values of α_0 when $alpha_t$ varies as an inverse of the elapsed time t. A linearly decaying learning rate performed best (Figures 6.5 and 6.6). The plots exhibit no evidence of a tendency to over-fit the training set over time and explain the lack of cross-validation to increase the classification accuracy.

Moderately high values of α of between 0.08 and 0.2 revealed a peculiar phenomena (Figure 6.5): a decrease in classification accuracy and an increase in the variance during the middle of training, followed by an increase in classification accuracy and a decrease

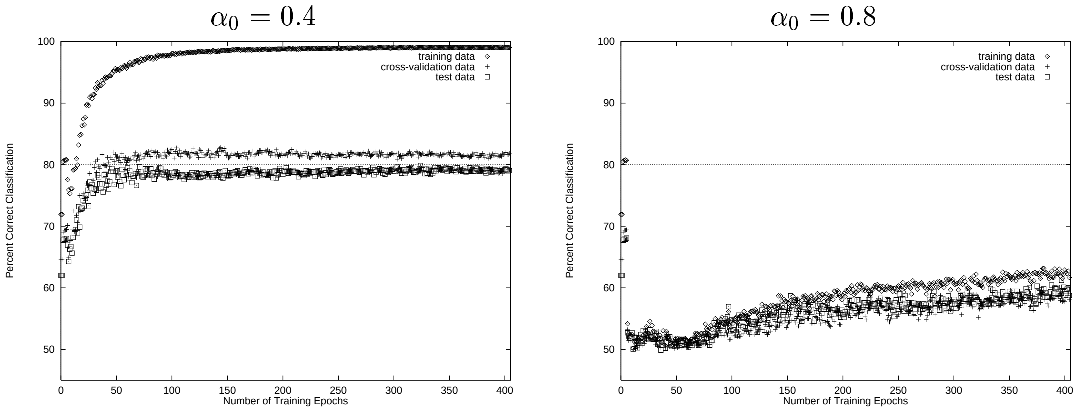


Figure 6.6: Effect of different values of α_0, where alpha is inversely related to time, on the classification accuracy of the training, cross-validation, and test data per training epoch for over 400 epochs.

in the variance at the end of training (Figure 6.5). This unusually shaped learning curve was hypothesized to be the result of poor initialization and exploration for a better local optima as the step size, α, decreases over time.

6.5 Initialization Method

The initialization of reference vectors is critical to guaranteeing that all of the reference vectors that make up the class decision boundaries will be modified. It was hypothesized that the peculiar form of the optimal learning curves, a decrease in accuracy followed by an increase to the highest value (Section 6.4, Figure 6.2), might be the result of a less than optimal initialization procedure. It was further hypothesized that a better initialization procedure might not only improve the overall classification accuracy, but change the learning dynamics as well such that an early stopping method, i.e. cross-validation, might prove useful and yield further improvements. All of the previous experiments had been conducted by initializing the reference vectors at random data points and then running OLVQ for four epochs.

The peculiar form of the optimal learning curves (Section 6.4, Figure 6.2) rendering early stopping methods useless was unchanged by varying the initializing technique. K-means initialization was compared with initialization to random chosen data vectors using the *K-nearest neighbors method* (KNN) alone and in combination with OLVQ. The KNN method initializes the reference vectors at points identical to randomly chosen data vectors as long as the majority of the K nearest neighbors, i.e. data vectors, are of the same class as the data vector whose position the reference vector is adopting.

Initial experiments tested values of K between 1 and 128 inclusive for KNN with no observable trend nor statistically significant difference in the final classification accuracy: $K = 2$ was used as a simple value that appeared to be at least as good as any other. Experiments were run using OLVQ for 100 epochs with no significant change after the first couple epochs. In order to ensure sufficient initialization, OLVQ was generally allowed to run for four epochs. LVQ2.1 was run for 400 epochs. Only the first approximately 100 epochs are presented to better illustrate the effect of the initialization techniques (Figure 6.7). Classification accuracies are plotted on a scale of from 60% to 100% on all three graphs to facilitate comparison.

Overall K-means initialization resulted in a classification accuracy of approximately 73%, OLVQ a final accuracy of approximately 80%, and initialization at random data points a final classification accuracy of approximately 83%. Since in general differences of greater than 6% are statistically significant it appears that K-means initialization gives worse final results than the other two initialization techniques.

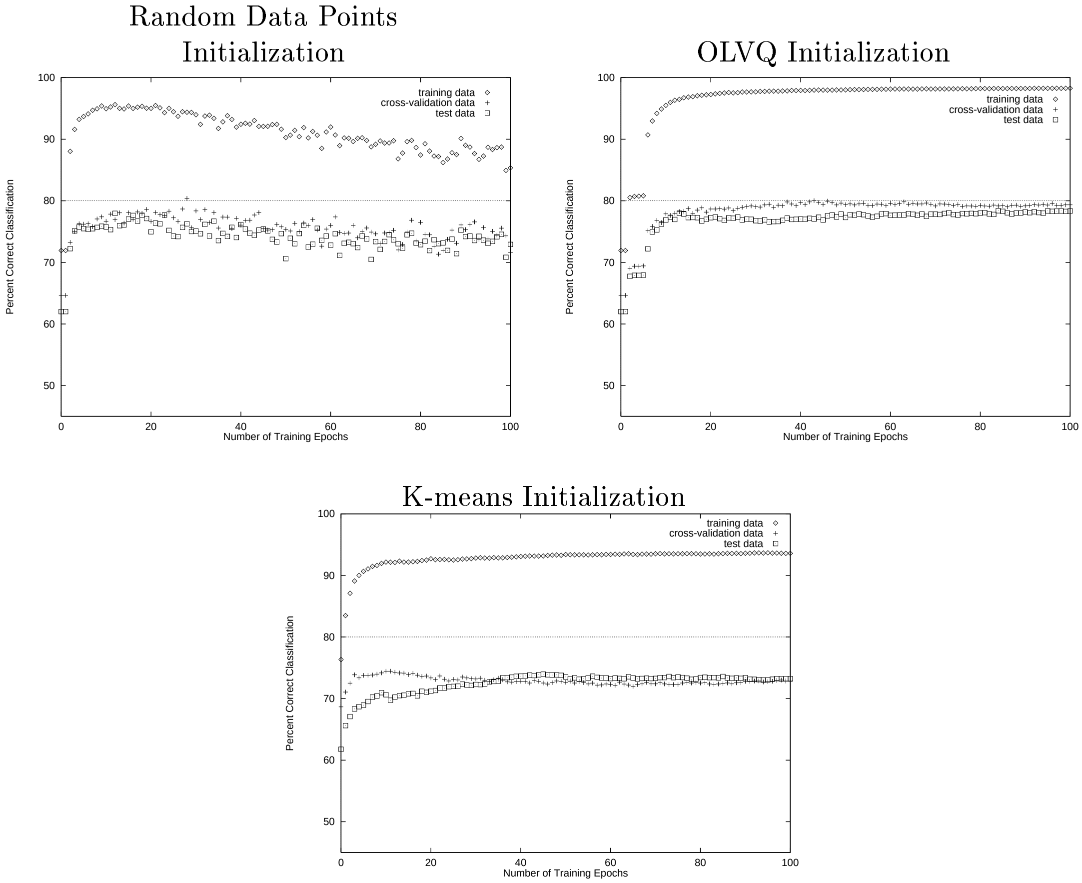


Figure 6.7: Effect of different initialization techniques on the classification accuracy of the training, cross-validation, and test data per training epoch for the first fifty epochs.

Chapter 7

ANALYSIS OF LVQ2.1

An analysis of the final positions of the reference vectors using LVQ2.1 with optimal parameters and no early stopping finally revealed some insight into the peculiar form of the learning curve and the failure of an early stopping to improve the classification accuracy of LVQ2.1 (Figure 7.1). The cause is related to the failure of LVQ2.1 to continue to approximate the class distributions over time as is the inability to optimize step size of LVQ2.1 to produce OLVQ2.1: on average the step size would not decrease and the process would not converge [27] (Section 4.4).

Kohonen [25, 27] takes advantage of two methods to stabilize the reference vector positions while retaining pair-wise reference vector updates of LVQ2.1. The window width constraint limits the effect of the gradient away from the data vector by limiting updates to cases where the two reference vectors are relatively equal distance away from the data vector. In order for reference vectors to continue to be pushed farther away both reference vectors must already be pushed a relatively equal distance away from the data vectors concerned (Section 7.2, Figure 7.4). The second method directly adds an additional force of attraction scaled down by the value of ϵ.

Though these two methods do not directly oppose the gradient calculations of Equation 3.6, neither are they the most accurate approximation of these equations. That their principal purpose is to interact to stabilize the reference vector positions is evident from Kohonen's recommendation to associate larger, more constraining, values of ϵ with larger, less constraining, window widths and vice versa [27]. More accurate gradient approximations can theoretically be implemented through small window widths and large ϵ values.

The empirical superiority of the LVQ2.1 algorithm is puzzling in light of its failure to continue to approximate the class distributions. Its superiority over LVQ1 appears to be due to the method of pair-wise updates introduced with LVQ2. The less significant superiority of LVQ2.1 over LVQ3 must be the result of Equation 4.4. This equation is the only difference between LVQ2.1 and LVQ3 and was introduced to assure at least a rough approximation of the class distributions over time.This suggests that either continued approximation of the class distributions is unnecessary or that the Equation 4.4 is inappropriate.

This chapter focuses on analyzing the failure of LVQ2.1 to continue to approximate the class distributions. The values of the reference vectors after training are analyzed first in relation to the values of the data vectors followed by an analysis of the average coefficient values for the two different classes, math and base (Section 5.1.1). The failure of LVQ2.1 to continue to approximate the class distributions is explained and illustrated by a a simple example. A discussion of the sub-optimality of these final reference vector positions is followed by a discussion of Kohonen's solution to this problem, LVQ3.

7.1 Empirical Observations

What LVQ2.1 *learned* during training was investigated by analyzing the values of the reference vectors after training. Two techniques were used to aid in visualizing the reference vectors. *Sammon Mapping* [18] is a non-linear dimensionality reduction technique that strives to preserve the same relative distance between reference vectors in two dimensions that existed in the original k dimensions, where $k > 2$. This permits the comparison of reference vectors among themselves as well as to the data vectors, allowing the visualization of the Voronoi tessellations and the resultant classification boundary. *Averaging* was used to combine the reference vectors to analyze the average values of the coefficients of the reference vectors.

7.1.1 Sammon Mapping

To aid in the graphical visualization of Voronoi tessellations, the Sammon mapping [18] method of dimensionality reduction was used. This is a non-linear dimensionality reduction algorithm that over repeated cycles adjusts the positions of the data points in the lower dimensionality data set to more accurately represent the relative positions of the data points in the higher dimensionality set [18].

Let d^*_{ij} represent the distance between two reference vectors i and j in the original higher dimensional space, d_{ij} represent the distance between two reference vectors in the desired lower dimensional space and N the number of reference vectors in each of the two spaces. The dimensionality reduction or mapping error is:

$$E = \frac{1}{\sum_{i=1}^{N}\sum_{j=i+1}^{N}[d^*_{ij}]}\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{[d^*_{ij}-d_{ij}]^2}{d^*_{ij}}.$$

The error is minimized though a stochastic approximation method using gradient descent. Let r equal the dimensionality of the reduced dimensionality space, then at time t,

$$d_{ij}(t) = \sqrt{\sum_{k=1}^{r}[y_{ik}(t)-y_{jk}(t)]^2}.$$

The vector update equation for the reduced dimensionality space is,

$$y_{lk}(t+1) = y_{lk}(t) - \alpha\Delta_{lk}(t)$$

where

$$\Delta_{lk}(t) = \frac{\partial E(t)}{\partial y_{lk}(t)} \Bigg/ \left|\frac{\partial^2 E(t)}{\partial y_{lk}(t)^2}\right|.$$

Though Sammon advises $\alpha \approx 0.3$ or 0.4, we used $\alpha = 0.2$ per Kohonen [28]. The partial derivatives are,

$$\frac{\partial E}{\partial y_{lk}} = \frac{-2}{c}\sum_{\substack{j=1\\ j\neq l}}^{N}\left[\frac{d^*_{lj}-d_{lj}}{d_{lj}d^*_{lj}}\right](y_{lk}-y_{jk})$$

and

$$\frac{\partial^2 E}{\partial y_{lk}^2} = \frac{-2}{c}\sum_{\substack{j=1\\ j\neq l}}^{N}\frac{1}{d_{lj}d^*_{lj}}\left[(d^*_{lj}-d_{lj}) - \frac{(y_{lk}-y_{jk})}{d_{lj}}\left(1+\frac{d^*_{lj}-d_{lj}}{d_{lj}}\right)\right],$$

where

$$c = \sum_{i<j}^{N}[d^*_{ij}].$$

Simplifying,

$$\Delta_{lk}(t) = \sum_{\substack{j=1\\ j\neq l}}^{N}\left[y_{lk}(t)-y_{jk}(t) - \frac{(d^*_{lj}-d_{lj}(t))}{(y_{lk}(t)-y_{jk}(t))}\frac{d^2_{lj}(t)}{d^*_{lj}}\right].$$

It was observed through Sammon mapping that at the end of 400 epochs the LVQ2.1 reference vectors formed a ring around the data vectors. All reference vectors were more or less equal distance from the center of the data cloud. The reference vectors of one class were grouped together to form half the ring and those of the other class formed the other half (Mapping Error = 0.17).

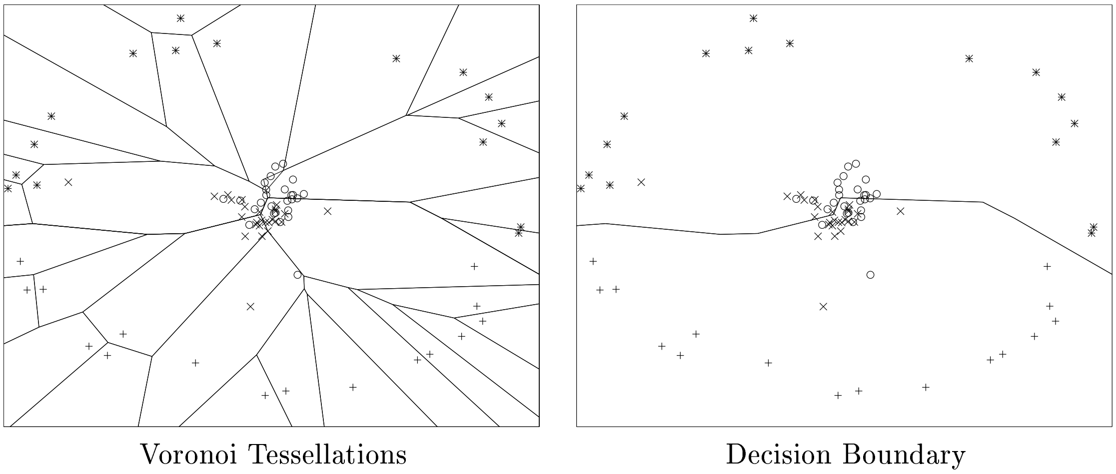


Figure 7.1: Sammon Mapping of LVQ2.1 reference vectors after training. (x's represent data vectors for the math task, o's represent data vectors for the base task, +'s represent reference vectors for the math task, *'s represent reference vectors for the base task.)

Figure 7.1 as well as all subsequent similar plots show the relative position of the reference vectors and corresponding data vectors from the test set. The size of the test set comprises approximately one tenth of the total number of data vectors available. The plot represents the vectors from one of the thirty examples sets of Subject 1, chosen arbitrarily (Section 5.1.4). All thirty of the experimental sets exhibited the same phenomenon, as did the reference vector mappings for the other subjects. Verleysen et al. [40] claim that such reference vector positions are limited in the data distributions that they are able to accurately classify (see Section 7.3 for a more detailed discussion of this point).

7.1.2 Averaging

After training the thirty different training sets using LVQ2.1 with optimal parameters, the thirty different sets of final reference vectors were averaged together to produce 32 average reference vectors, 16 base vectors and 16 math vectors. Figure 7.2 displays the values of each of the 36 features of the 32 reference vectors plotted in Figure 7.1. Each line represents a reference vector. The first six columns correspond to the coefficients of the first order or the sixth order AR model, the next six columns the second order and so on (Section 5.1.3). Black or solid boxes represent positive values and white or empty boxes negative values. The size of the boxes is directly proportional to the magnitude of the feature. Thus, very small boxes of different colors represent values close to zero, and very large boxes of different colors represent the two extremes of possible values.

The reference vectors are organized by position beginning with the base reference the farthest to the right and continuing counter-clockwise around the circle of reference vectors ending with the math vector the farthest to the right. According to the Sammon mapping dimensionality reduction procedure used to produce Figure 7.1, neighboring reference vectors in Figure 7.2 are more similar to each other than more distant reference vectors. Due to the circular nature of the reference vectors the same is true of the vectors at the two extremes. The only significant factor we are able to see in Figure 7.2 is the general difference between the values of the reference vectors representing base tasks and those representing math tasks.

An important requirement of any averaging technique is the consistency of what is being averaged. In our example it was essential for the resulting average to have any meaning that those reference vectors that represented the same section of the data space be averaged together and that reference vectors that represented different sections of the data space *not* be averaged together. This task was made easier in the case of LVQ2.1, by the fact that Sammon mapping was able to completely separate the base and math vectors in two dimensions (Figure 7.1).

The consistent pattern of a ring of reference vectors surrounding the data space consisting of two distinct arcs, one of math vectors and the other of base vectors made correlating vectors across the various reference vectors sets for averaging a fairly simple and accurate method of generalizing the results. The Sammon mapping separation of the reference vectors into two distinct clusters also supports the validity of comparing the average of all the base vectors to the average of all the math vectors.

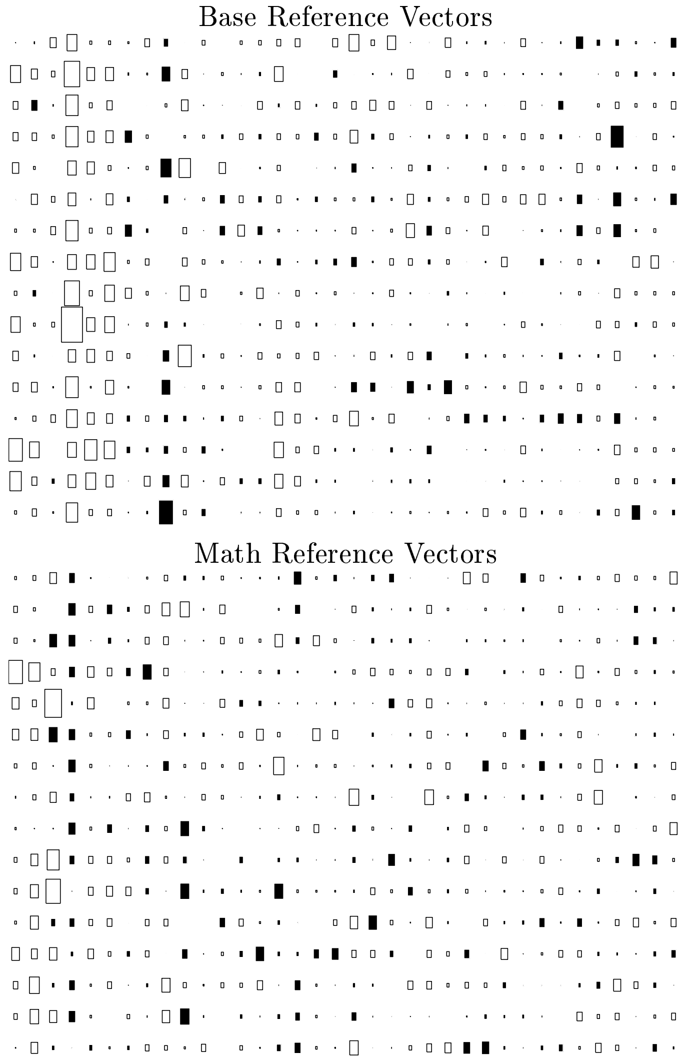


Figure 7.2: Reference vector feature coefficients. Columns correspond to first through sixth order AR coefficients for channels C_3, followed by similar coefficients from channels C_4, O_1, O_2, P_3, and P_4.

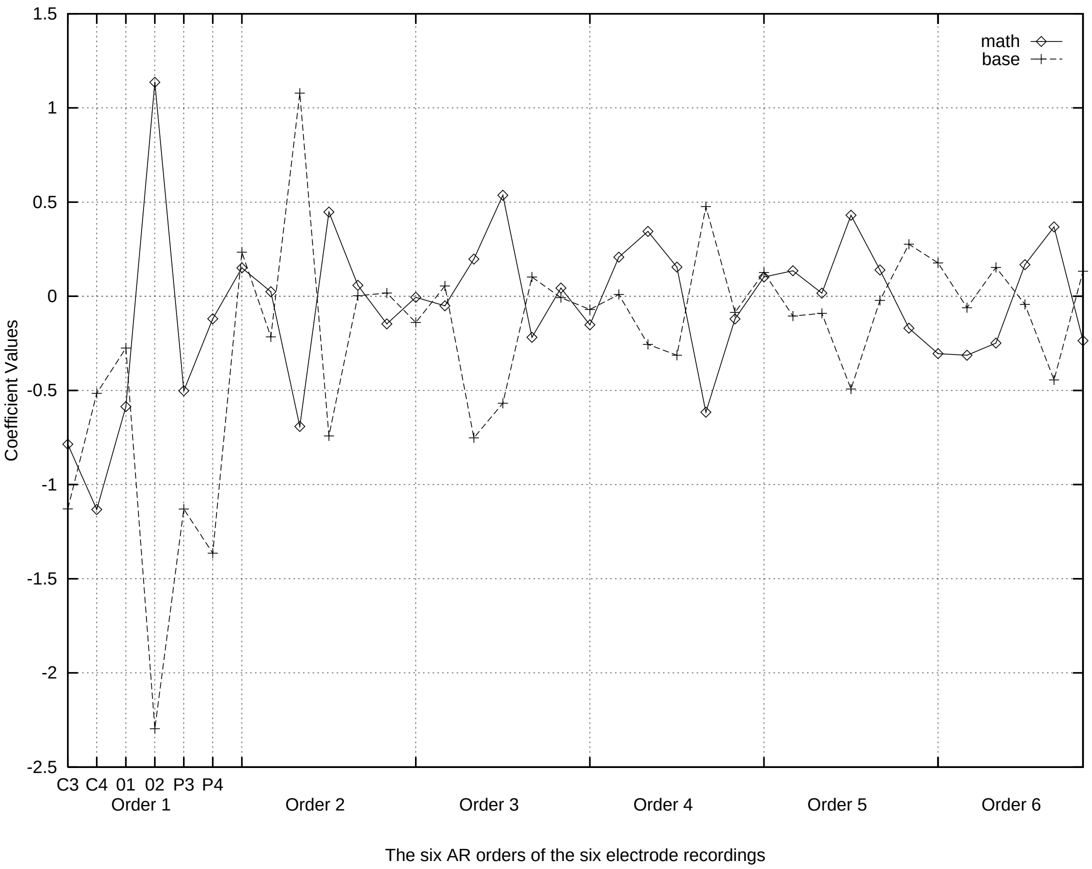


Figure 7.3: Average feature coefficients of reference vector sample.

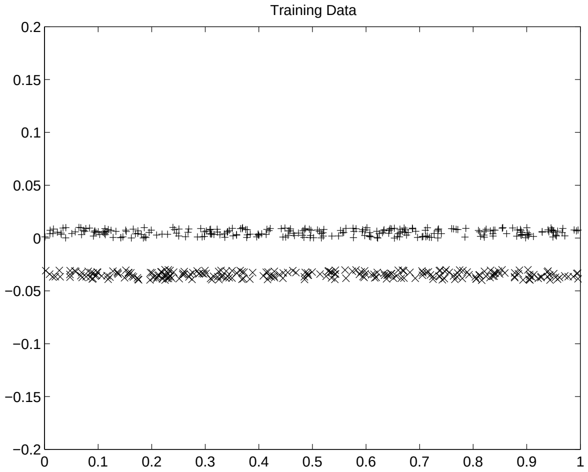
The comparison of the average of all the base vectors to the average of all the math vectors (Figure 7.1.2) makes the differences more clear. It is evident from the average magnitudes of the coefficients that the classification potential of the coefficients decreases as the AR order of the coefficients increases. There are clear differences, however, between the average math and base vectors coefficients for all AR-orders. The coefficients obtained when averaging the sixteen math and base vectors (Figure 7.1.2) correlate strongly to the coefficients of the math and base vectors when LVQ2.1 was run with only two reference vectors.

7.2 Statistical Observations

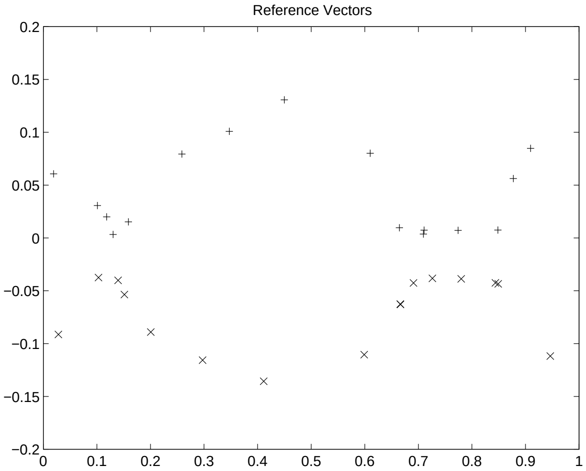
The observed consistent long term effect of LVQ2.1 to move reference vectors completely out of the data vector space and their resulting pattern of two arcs, one for each class, appears to be undesirable [40]. The analysis of the coefficients of the reference vectors by averaging techniques provides no insight into the reasons behind this peculiar phenomenon. An explanation for this result, however, is available based on probabilities, specifically the probability that the closest reference vector to a data vector will be of the same class as the data vector.

If an LVQ algorithm is classifying better than chance, then most often the closest reference vector to a data vector will be of the same class. LVQ1 only updates the closest reference vector, which will most often be of the same class as the data vector and hence over time the updates will move the reference vectors closer to the data vectors.

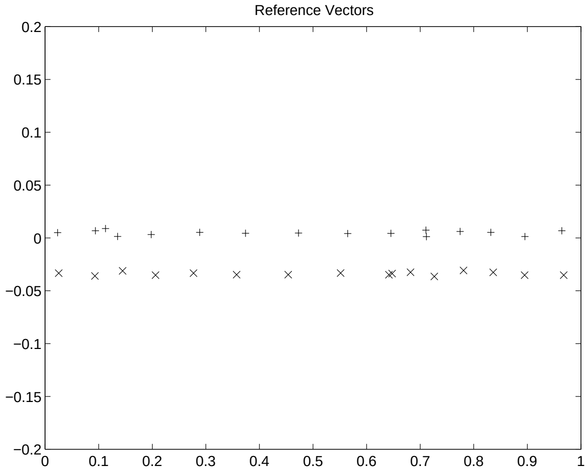
LVQ2.1 determines the closest two reference vectors to a particular data vector—if these two vectors are labeled differently, e.g. base and math, then they are updated. The reference vector with the label the same as the data vector is moved toward the data vector: α times the distance between it and the data vector. The other reference vector is moved away from the data vector: $-\alpha$ times the distance between it and the data vector. Thus if LVQ2.1 is classifying better than chance, then over time the updates will move the reference vectors away from the data vectors, as was observed in our experiments with LVQ2.1 (Figure 7.1) where the reference vectors were pushed outside the data space. The



a. Data



b. LVQ2.1



c. LVQ2.1 Without Repulsion

Figure 7.4: Effect of repulsive update on LVQ2.1 after training for five epochs with 32 reference vectors, $\alpha = 0.08$, and a window width equal to 0.8

bounds on this effect are realized when the two closest reference vectors to a data vector belong to the same class as the data vector.

A simple example was constructed to more clearly illustrate this effect (Figure 7.4a). Two non-overlapping bands of differently labeled data in close proximity to each other were generated. The result of applying LVQ2.1 to this problem clearly illustrates the long term effects of the repulsive reference vector updates after as few as five training cycles. The difference between the standard LVQ2.1 algorithm, and the LVQ2.1 algorithm without repulsion is clear (Figure 7.4b&c).

As was seen with the experimental EEG data (Section 7.1.1, Figure 7.1) the overall repulsive effect on data segregated into two parallel bands is symmetrical (Figure 7.4b). Thus, though LVQ2.1 does not approximate the class distributions, it does not always adversely effect the decision boundary and the resulting classification accuracy (Section 6.2).

7.3 Sub-optimality of final LVQ2.1 reference vector positions

It is quite easy to see that any decision boundary formed by reference vectors in a ring around the data vector space can be duplicated by reference vectors positioned within the data vector space. It is equally evident that the reverse is not true. The position of reference vectors outside the data vector space reduces the number of different decision boundaries possible for a given number of reference vectors. The assumption of the sub-optimality of LVQ2.1 rest first of all on the assumption that there exist certain problems for which the best decision boundary formed by reference vectors outside the data space is significantly worse than the best decision boundary formed by reference vectors within the data space.

Secondly, if a method of early stopping, e.g. cross-validation, is used in conjunction with LVQ2.1, it must also be assumed that the best decision boundary formed during the movement of the reference vectors outside the data space is significantly worse than the best decision boundary formed when the reference vectors remain in the decision boundary. This assumes that the algorithm imposes a particular path on the movement of the reference vectors, thereby limiting their possible positions within the data vector space before they are eventually expelled.

Similarly, we must finally assume that the positions that the reference vectors pass through on the way to leaving the data vector space are somehow inferior to the positions they would pass through if they were not being pushed out of the data vector space. However, this assumption is entirely dependent on the assumption that the algorithmic updates which lead to the eventual expulsion of the reference vectors are significantly inferior to some other possible updates that do not have the same long term expulsion effect.

Diamantini [7], actually proves that the LVQ2 algorithms lead to sub-optimal decision boundaries. An example is also given using LVQ2.1 to classify two equiprobable bivariate normal density functions with identical mean and different covariance matrices using four reference vectors. It is evident that there exists a repositioning of the final reference vector positions that would result in a more accurate approximation of the optimal Bayesian decision boundary and hence greater classification accuracy. The same results were also found for LVQ1, [8]. Diamantini [7] also shows an example where her Minimum Error Probability (MEP) algorithm (Section 4.8, Equations 4.7 and 4.8) results in only 1.3% greater error probability than the optimal Bayesian classifier. The error probability of LVQ2.1 given the same classification task is 13.3% greater than that of the Bayesian classifier.

It is significant to note with all three algorithms, LVQ1, LVQ2.1 and MEP, the greater error probability appears to be caused by a LVQ decision boundary that was outside the Bayesian decision boundary [7, 8]. This is clearly what would be expected by the imbalance in favor of repulsive forces in the case of LVQ2.1. It also appears to be a likely result of the MEP algorithm given its similarity to LVQ2.1 (Equations 4.1, 4.7, 4.2, 4.8, 4.3 and 4.6). It is, however, a rather surprising result in the case of LVQ1.

7.4 Continued Approximation of the Bayesian Decision Boundary

The cause behind the failure of LVQ2.1 to continue to approximate the class distributions has been explained through simple statistical observations. It was shown that this cause is a direct result of the relative strengths of the attractive and repulsive forces exerted on the reference vectors. The fact that goal of LVQ1 is not strictly to minimize misclassification error probability, but rather to maximize its cost function with respect to units of the same class and minimize the cost function with respect to units of the other class has been proven by Diamantini [8, 6, 9]. Diamantini also demonstrates that LVQ2.1 doesn't minimize the misclassification error probability either [7].

It appears that errors of LVQ2.1 in approximating the optimal Bayesian decision boundary are due to its failure to continue to approximate the class distributions. Kohonen introduces LVQ3 (Section 4.5) as a method to provide at least a *rough* approximation of the class distributions over time. LVQ3 adds an additional update to the reference vectors, albeit by a lesser amount, a factor of ϵ, when the two closest reference vectors are of the same sign as the data vector. Provided the value of ϵ is between 0.1 and 0.5, the addition of this attractive force to the algorithm tends to keep the reference vectors in the same region as the data [27]. There is no theoretical justification given for this modification. It is simply offered as a solution to the apparent advantage of continued rough approximation of the class distributions.

It is also worth noting that constraint of LVQ2.1 introduced by the notion of a *window width* also helps the algorithm to approximate the class distributions. A data vector must be situated within a certain distance, as specified by the window width parameter, of the class decision boundary separating the two closest reference vectors in order for it to influence the position of the reference vectors. This provides a constraint on the ratio of the relative distances separating the reference vector and the data vector, Equation 4.3, and by consequence a similar constraint is placed on the relative distances that the two reference vectors are moved: the distance that each reference vector is moved is an equal proportion, α, of the distance between them and the data vector. However, this constraint, by itself, is generally not strong enough to guarantee that the reference vectors continue to approximate the class distributions (Figure 7.1 and [27]).

An exception to this was observed with the simple example presented in the preceding section of classifying two non-overlapping bands of differently labeled data. In this case it was found that LVQ2.1 with a window width $w < 0.05$ was sufficient to provide continued approximation of the class distributions. $w < 0.05$ permits reference vectors to be updated only when the minimum of the ratio of the distances between a data vector and the two closest reference vectors, which must also belong to different classes, be greater than 0.9, (Section 4.4, Equation 4.3.) It is evident from Figure 7.4 that this is unlikely to ever be the case, which would mean that the only reason the window width parameter is guaranteeing continued approximation of the class distributions is because it is preventing LVQ2.1 from updating the positions of the reference vectors, leaving the reference vectors at the same positions they were initialized.

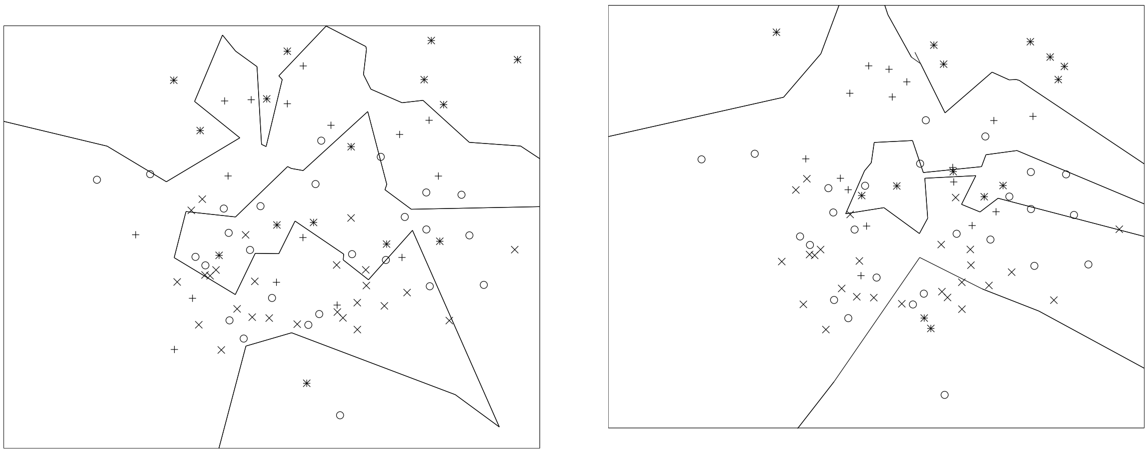


Figure 7.5: LVQ3 $\epsilon = 0.1$ & 0.5: Decision Boundary (Mapping Error =0.15).

Our experiments confirm Kohonen's [28, 27] that the LVQ3 algorithm with ϵ values of between 0.1 and 0.5 adds a force pulling the reference vectors towards the data vectors that is strong enough to contain the reference vector positions within the data space. Figure 7.5 shows representative final reference vector and test data vector positions for two different ϵ values. Previous results showed no significant variance in the classification accuracy for different values of ϵ (Section 6.1, Figure 6.4).

It was for this reason that the algorithm was simplified by eliminating a parameter, choosing $\epsilon = 0$ (LVQ3 with $\epsilon = 0$ is equivalent to LVQ2.1). The impact of the value of the parameter ϵ is also dependent on the number of reference vectors per class. If there is only one reference vector per class then LVQ3 and LVQ2.1 are again equivalent: the reference vector update specified by Equation 4.4 will never be used since the algorithm does not posses two reference vectors of the same class. The impact of the value of ϵ is roughly correlated to the number of reference vectors per class minus one divided by the total number of reference vectors.

Chapter 8

THEORETICAL ANALYSIS OF LVQ

Diamantini [7, 8, 6, 9] proves that the traditional LVQ algorithms proposed by Kohonen neither approximates Equation 3.4, nor minimizes the misclassification error probability. This chapter analyzes some of the basic theoretical concepts of LVQ, focusing on the proposed goal of LVQ (Equation 3.4), its gradient (Equation 3.6), and its approximation (which depends on certain assumptions that are not altogether evident.) These facets of LVQ are analyzed independently and in the context of minimizing misclassification error probability.

The chapter begins with a discussion of the errors of traditional LVQ in approximating Equation 3.4. Of particular interest is the effort to balance the repulsive and attractive forces necessary for continued function approximation over time and the deviation from a theoretically accurate approximation of Equation 3.4 induced by these efforts.

The next section is an *ad-hoc* presentation of alternative methods to balance the repulsive and attractive forces necessary for continued function approximation over time. This section is complemented by occasional preliminary results to demonstrate that radical deviations from a theoretical approximation of Equation 3.4 yield classification results comparable to LVQ3.

The following section returns to Equation 3.4. The question as to why the theoretically more accurate approximation of the equation represented by LVQ2.1 yields poorer function approximation in practice than other LVQ variations including the radical variants presented in the previous section is discussed. Secondly, the relationship between Equation 3.4 and minimizing misclassification error probability is analyzed especially in light of work by Diamantini associated with the MEP algorithm.

The chapter concludes with a discussion of the MEP algorithm and the quest for a classifier based on Vector Quantization that approximates optimal Bayesian Classification.

8.1 Errors in Approximating Equation 3.4

The approximation of Equation 3.6 is based on the Robbins-Monro theory of stochastic approximation [27]. Though the theory is well established and central to the theory of iterative approximations it is often only superficially understood. Unfortunately a detailed expose of the theory is beyond the scope of this study. Instead we will focus on several key assumptions for LVQ regarding the vector updates formulas in approximating gradient descent (Section 3.1.4, Equation 3.6):

1. Every data vector x is assumed to be situated in some Bayesian region B_k where k identifies the label associated with the region, i.e. $x(t) \in B_k$.

2. Assume a specific Bayesian region, B_k. Any update specified by the gradient assumes that the reference vectors to be updated, $m_i(t)$ and $m_j(t)$, and the current data vector, $x(t)$, are in the region B_k (i.e. $m_i(t), m_j(t), x(t) \in B_k$. If a reference vector to be moved, e.g. $m_i(t)$, has the same label as the data vector $x(t)$, then $x(t)$ is assumed to belong to the class with the highest discriminant function in that region, $x(t) \in S_k$ (Section 3.1.3, Figure 3.1 and Equation 3.2). Thus, the label or class of $x(t)$ is assumed to be the same as the class associated with the region, and the gradient specifies moving m_i toward $x(t)$ (Equation 3.6).

3. If a reference vector, m_i, to be moved at time t has a different label than $x(t)$, then $x(t)$ is assumed not to belong to the class with the highest discriminant function in the Bayesian region where it is located, $x(t) \notin S_k$.

4. $x(t) \notin S_k$ implies $x(t) \notin S_r$. In other words, if, per assumption 3, $x(t)$ is assumed not to be a member of the class with the highest discriminant function in the Bayesian region where it is located, then it is assumed that $x(t)$ belongs to the class with the second highest discriminant function in the Bayesian region where it is located. The gradient thus specifies moving m_i away from $x(t)$ (Equation 3.6). When only two classes of items are considered $x(t) \notin S_k$ implies $x(t) \notin S_r$ by default.

In the case of pair-wise updates of differently classed reference vectors, e.g. LVQ2.1, assumption 2 conflicts with assumption 3. The reference vector of the same class as the data vector $x(t)$ leads us to conclude that $x(t) \in S_k$ per assumption 2. At the same time, t, the reference vector of a different class than the same data vector $x(t)$ leads us to conclude that $x(t) \notin S_k$ per assumption 3.

The choice of which gradient direction to move in (Equation 3.6) depends on whether the label of the reference vector matches the label of the Bayesian region, B_k, where it is presently located. After selecting the closest reference vector to a data point, determining which gradient direction to move in, if any, relies on determining whether a data point belonging to a particular conditional probability distribution lies within its Bayesian region B_k or outside of it. This requires that the regions B_k be defined *a priori*. Approximating these regions, however, is the goal of the algorithm.

The notion of a *window width* parameter, introduced in LVQ2 and used in all subsequent versions of LVQ based on LVQ2, provides for a more accurate approximation of the gradient as the value of the parameter decreases. A small window width constrains the second closest reference vector to be nearly as close to the data vector as the closest reference vector. In the limit as the window width approaches 0 the two reference vectors are equal distance from the data vector and are thus both the closest reference vector to the data vector and both subject to be updated per Equation 3.6.

A small window width constrains the region where data vectors are able to influence the reference vector positions to be in close proximity to the hyperplane representing the decision boundary between the two reference vectors. Thus in the LVQ2 algorithms a small window width limits the adjustment of the decision boundaries at a particular time to be the result of influence only from those data vectors situated near the current estimates of the decision boundaries at that time.

The optimal Bayesian decision boundaries are defined at points where the two highest conditional probability densities are equal. Assume that the probability densities at the Bayesian decision boundaries are more accurately approximated by neighboring probability densities the closer they are to actual Bayesian decision boundaries. Then provided we also assume good initialization, a relatively small window width should increase the accuracy of the approximation of the optimal Bayesian decision boundaries.

The use of a small window width also help to balance the attractive and repulsive forces of LVQ2.1 providing for better continued approximation of the class distributions. Preliminary results have indicated that a window width where $w < 0.01$ provides for continued approximation of some simple class distributions that were not approximated for larger values of w. These values of w fall outside the range of previously tested window widths, and the influence of such small window widths on the classification accuracy on more complicated data distributions as well as their success in continued rough approximation of the class distributions remains to be tested.

For small window widths, the modifications of LVQ3 should improve the approximation of Equation 3.6. They do so in a biased way however, since LVQ3 does not update the two closest reference vector positions when the closest reference vectors belongs to the same class as each other but to a different class than the data vector. It would appear that an even more accurate approximation of Equation 3.6 would be to update only the closest reference vectors position when the window width constraint was not satisfied, i.e. the closest reference vector was significantly closer to the data vector than the second closest reference vector.

8.2 LVQ Algorithms not Based on Equation 3.4

Of all the LVQ algorithms investigated, LVQ2.1 results in the worst empirical approximation of the class distributions and hence of Equation 3.4. Yet, it is precisely this algorithm that theoretically appears to give the most accurate approximation of Equation 3.4 [7, 27]. Section 4.8 discusses the related observation that LVQ2.1 yields the worst empirical approximation of Equation 3.4, but the highest classification accuracy.

This section demonstrates that the LVQ3 solution of ensuring a more accurate continued approximation of the class distributions over time than LVQ2.1 is by no means the only possibility. The deviation of LVQ3 from a theoretical approximation of Equation 3.6 motivates the investigation of other deviations. In general the algorithms that follow represent more radical deviations from a theoretical approximation of Equation 3.6 than LVQ3.

Limited empirical studies were conducted based on some of the algorithms to demonstrate that not only are there a wide variety of alternative methods to improve the empirical approximation of the class distributions of LVQ2.1, the classification results of some of those tested are comparable with those obtained with LVQ3. The empirical results of this section, though limited, demonstrate a surprising degree of insensitivity to the implementation of reference vector movements and hence the basis of LVQ as a theoretical approximation of Equation 3.4.

A simple solution to the problem of LVQ2.1 pushing the reference vectors out of the data space is to bound the repulsive movements to be less than or equal to the attractive movements. This bound is directly expressible in terms of the size of the *window width* w, (Section 4.3, Equation 4.3.) The reference vector positions are only modified when the ratio of the distances between the data vector and closest reference vector to that of the distance between the data vector and second closest reference vector is greater than $\frac{1-w}{1+w}$. Multiplying α in Equation 4.2 for LVQ2.1 (Section 4.4) by $\frac{1-w}{1+w}$ guarantees that the sum of all forces pushing reference vectors away from data vectors is less than or equal to the sum of all forces pulling reference vectors toward data vectors. Letting $L(x)$ represent the label of x, this algorithm can be expressed as:

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha(t)[x(t) - m_c(t)] & \text{if } L(m_c) = L(x) \\ m_c(t) - \frac{1-w}{1+w}\alpha(t)[x(t) - m_c(t)] & \text{otherwise} \end{cases}$$

As this bounds the attractive forces to be greater than the repulsive forces, the distance between the reference vectors might decrease asymptotically, though not monotonically as with LVQ2. It is hypothesized that the use of an early stopping technique with this algorithm would increase classification accuracy.

It is also possible to further limit the attractive and repulsive forces to be exactly equal: the distance moved away from a data vector by one reference vector can be constrained to equal the distance moved away from the same data vector by a different reference vector. An equal step size for both movements is made while preserving the gradient direction of each. If the step sizes are made equal along each separate dimension then the direction of the gradients are altered and the expressibility of the collection of reference vectors is limited by holding the center of mass of the reference vectors fixed. Preserving the gradient direction requires a significant amount of additional geometric calculations likely to slow down the algorithm.

Another relatively simple alternative is to eliminate any dependence on the distance between a reference vector and the data vector when moving reference vector away:

$$m_c^{(i)}(t+1) = \begin{cases} m_c^{(i)}(t) + \gamma\alpha(t) & \text{if } m_c^{(i)} \geq x^{(i)} \\ m_c^{(i)}(t) - \gamma\alpha(t) & \text{otherwise} \end{cases} \tag{8.1}$$

A slightly more complicated approach uses a logarithmic scale where movements away from the data vector are inversely proportional to the relative proximity of the reference vector in each of the dimensions of the vector space:

$$m_c^{(i)}(t+1) = \begin{cases} m_c^{(i)}(t) + (V^{(i)} - |x^{(i)} - m_c^{(i)}|)\gamma\alpha(t)) & \text{if } m_c^{(i)} \geq x^{(i)} \\ m_c^{(i)}(t) - (V^{(i)} - |x^{(i)} - m_c^{(i)}|)\gamma\alpha(t)) & \text{otherwise} \end{cases} \tag{8.2}$$

$V^{(i)}$ equals the maximum range of values across each dimension. Indexing V by the dimension preserves the relative importance of different features represented by the scaling of the data vectors across dimensions. The optimal value of γ will need to be determined experimentally. Common sense suggests $0 < \gamma < V$ for Equation 8.1, or $0 < \gamma < \max_i V^{(i)}$ for Equation 8.2.

The results of implementing Equation 8.2 for repulsive updates were statistically equivalent to the classification accuracy obtained with LVQ3. The results of using Equation 8.1 for repulsive updates was statistically equivalent to the results when using either Equation 8.1 or Equation 8.2 for both repulsive as well as attractive updates: the classification accuracy was statistically equivalent to that obtained with LVQ1. It is rather surprising that the reference vector updates either independent of inversely related to the proximity of the reference vectors to the data vector did not significantly decrease classification accuracy.

Replacing moves away from differently classed data vectors with moves towards data vectors of the same class is another alternative. Although this solution resolves the problems of continued approximation of class distributions because of an imbalance between the attractive and repulsive forces of LVQ2.1, it introduces a problem of an attractive force concentrated around a specific data vector as opposed to a cluster of data vectors. Rather than moving toward one particular data vector of the same type it would be better to move toward the center of a cluster of data vectors of the same type. However, since the distribution of data vectors is not known *a priori*, the optimal size of a collection of data vectors cannot be determined.

To reduce the over attraction to the closest data vector of the same class to the detriment of simultaneously representing nearby similarly classed data vectors, the force of this attraction can be scaled down by a factor $\delta \leq 1$. The lower bound on δ can be safely established as equal to the number of data vectors in the sample that are not of a particular class. As a guideline, if δ equals the number of oppositely classed data vectors that exerted a force on it, then the attractive force towards the closest data vector of the same class would be doubled. Preliminary experimental evidence suggests that this alternative also yields statistically equivalent classification accuracy as that obtained with LVQ3.

8.3 Problems with Equation 3.4

The advantage of the Equation 3.4 is that, provided the initialization has separated the reference vectors into appropriate B_k regions, an accurate approximation of the function that preserves this separation of the reference vectors leads to an similarly accurate approximation of the optimal Bayesian decision boundaries. Note that this transformation and the segregation of different reference vectors into different regions is a direct result of the nearest neighbor rule used for classification. The k-nearest neighbor rule is potentially capable of determining which probability density function or which class of reference vectors predominate in a particular region, and thus does not require a complete separation the classes for accurate Bayesian classification.

The decision boundaries are identified by those points where Equation 3.4 is zero. For accurate classification, it is only the regions that define the decision boundaries, where Equation 3.4 equals zero, that we need to accurately approximate. As long as these regions are accurately approximated, including a positive probability density in areas near these regions, then errors in the remainder of the function approximation are irrelevant: values close to zero, or even equal to zero provided the segregation of classed reference vectors remained appropriate, would actually yield to a more efficient algorithm focusing the placement of reference vectors near the decision boundaries where they are needed.

It is immediately evident that the fact that Equation 3.4 is zero at the decision boundaries means that the reduction in the probability density is greatest at these points. A VQ approximation of Equation 3.4 will thus place many fewer reference vectors in this region than would be the case in approximating an accumulation of the discriminant functions. The region in close proximity to the decision boundaries is where the greatest percentage of reference vectors should ideally be located for optimal classification. The problem, is the practical one of insuring that the reference vectors will not cross the Bayesian decision boundaries, while placing the greatest percentage of reference vectors in the closest proximity of these boundaries that simultaneously enables the accurate determination of these boundaries as possible.

It is also to be observed that classifying non-overlapping probability distributions can be done by separately approximating the individual probability density functions. The triviality of this situation is interesting only in so far as it provides guidelines on appropriately approximating overlapping conditional probability densities functions (Equation 3.2) or a function derived from them (Equation 3.4). Nevertheless, classes that are completely separable should reduce to a sum of separately approximated densities. If there is no overlap in the probability densities, then there is no difference between the discriminant functions, Equation 3.2 for all S_k and the transformed function, Equation 3.4, for all B_k. No overlap is equivalent to saying that $S_k \in B_k$ for all B_k. Thus for $x \in B_k$ and $h \neq k$, $P(S_k) = 1$ and $P(S_h) = 0$:

$$\begin{aligned} f(x) &= p(x|x \in S_k)P(S_k) - \max_h\{p(x|x \in S_h)P(S_h)\} \\ &= p(x). \end{aligned} \tag{8.3}$$

$$\tag{8.4}$$

This leads to the familiar attractive portion of the LVQ1 update, Equation 4.1. Without any repulsive force it is essential that the algorithm is properly initialized so that the reference vectors are already appropriately separated by the initialization method.

The formula for repulsive updates (Section 3.1.4, second part of Equation 3.4) was based on a function transformation (Sections 3.1.3 and 3.1.4, Equations 3.2 and 3.4, Figure 3.1) that appears unnecessary. The function transformation to Equation 3.4 is formed by subtracting the conditional probability density at x of the class with the highest conditional probability density at x from the conditional probability density at x of the class with the second highest conditional probability density at x.

This transformation is such that $f(x) = 0$ at all Bayesian decision borders. A Bayesian decision boundary is by definition the point at which the conditional probability densities at x of the classes with the two highest conditional probability densities are equal. This provides for a convenient definition for the Bayesian borders, even though it is not theoretically necessary.

There are, however, practical arguments for such a transformation. Any accurate method of approximating this function should discourage reference vectors from crossing the Bayesian decision boundary, since the value of the function at these boundaries is 0. Thus to the extent that the initialization method has successfully segregated the reference vectors into the appropriate regions, approximating the transformed function may well preserve this segregation while providing a more precise estimate of the borders.

This function transformation, however, is at the root of the failure of LVQ2.1 to continue to approximate the class distributions and consequently fail to guarantee an approximation of the Bayesian decision boundaries for all distributions. The gradient calculations of the original discriminant functions specify only an attractive force (Section 3.1.4, Equation 3.7). The gradient calculations of the transformed function, Equation 3.4, specify both an attractive force as well as a repulsive force (Equation 3.8). As previously mentioned (Section 7.4), it is the difficulty in balancing these two forces that causes LVQ2.1 to fail to approximate the class distributions.

Theoretical justifications for this transformation appear lacking. In addition it introduces an additional level of complexity in dealing with a gradient that specifies movements in two diametrically opposed directions (Section 3.1.4, Equation 3.6). It appears, however, that the practical benefits of such a transformation when combined with a nearest-neighbor classification method are quite solid. Empirical studies using LVQ2.1 where the repulsive updates were omitted resulted in a reduction in classification accuracy of approximately 10%, from 82% to 72% correct classification. Additional support for the usefulness of repulsive forces and related function transformation is given by the fact that the MEP algorithm which is theoretically based on minimizing the misclassification error probability is strikingly similar to LVQ2.1.

8.4 Maximizing Classification Accuracy

Kohonen confuses function approximation with classification by failing to give adequate importance to the labels of the reference vectors. Optimal Bayesian classification according to the nearest neighbor rule requires:

1. The reference vectors be segregated by label into Bayesian regions.
2. For every set of two differently labeled reference vectors that are each the closest reference vector of a different label to the other, the midpoint of the line connecting them lies on the Bayesian border.

This section discusses this important distinction in light of the segregation of reference vectors by class or label required by the nearest neighbor classification rule for optimal classification accuracy. The MEP algorithm is then compared with LVQ2.1 and analyzed in light of this distinction.

8.4.1 Segregating Reference Vectors by Label

The function (Equation 3.4) that Kohonen's LVQ algorithms are derived from is completely independent of the labels of the reference vectors. This places a great deal of importance on the initialization method as a means of segregating the differently labeled reference vectors.

The approximation of each of the B_k regions by a *different* collection of reference vectors, each collection identified by the label k is more complicated than approximating the Equation 3.4 by a collection of reference vectors. The former problem is what is required by LVQ: using only one class of reference vectors, or multiple classes or reference vectors that are not separated into separate B_k regions is less than ideal. The sole benefit of using the transformation from the collection of $p(x)$ to $f(x)$ is so that each B_k region contains only those reference vectors appropriate to that region. If this is the case a more precise determination of the optimal Bayesian decision boundary is possible.

Moreover, the Equation 3.4 is a potentially discontinuous combination of k equations, one for each of the B_k regions. $f(x)$ implicitly assumes that the reference vectors repositioned in a particular region B_k are the appropriate reference vectors for that region. There is nothing in the approximation through movements in the direction of the gradient that provides for sorting a mix of differently labeled reference vectors into separately labeled B_k regions. The gradient calculations simply work toward approximating the function as a whole, making no distinction between the reference vectors in one B_k region and those in another.

An *appropriate* approximation of $f(x)$ by LVQ relies on proper initialization of the reference vectors, i.e. the reference vectors must be already sorted into separate B_k regions. Additionally, the stochastic gradient approximation method does not guarantee that the differently labeled reference vectors will not get mixed up during the iterative function approximation.

8.4.2 Discussion of MEP algorithm

The MEP algorithm is quite similar to LVQ2.1. There are only two differences, the window width constraint of LVQ2.1 is replaced by a Δ constraint and the MEP algorithm introduces an additional term in the update equations, $1/ \parallel m_i(t) - m_j(t) \parallel$. The Δ constraint of MEP (Equation 4.6), like the window width constraint of LVQ2.1 places a constraint on the distance between the data vector and the midpoint between the two reference vectors to be updated. The Δ constraint, however, also includes an angular component, placing a constraint on the angle between the data vector and the decision surface. The projection of data vector unto the decision surface is large when the two relatively parallel and close, and small when they are relatively orthogonal and far apart.

The Δ condition is therefore more likely to be satisfied when the data vector is close to and relatively parallel with the decision surface and less likely to be satisfied when the data vector is far away from and relatively orthogonal to it.

The MEP algorithm would appear, like LVQ2.1, to contain an overly powerful repulsive force causing the algorithm under certain data distributions to fail to approximate the class distributions by pushing the reference vectors away from the sections in the data space with high conditional probability densities, creating a biased and sub-optimal decision boundary. The major difference between the two algorithms is the additional term in the update equations of the MEP algorithm, $1/ \parallel m_i(t) - m_j(t) \parallel$. By scaling the step size in inverse proportion to the distance separating the two reference vectors this should slow the movement of reference vectors away from the sections in the data space with high conditional probability densities as the distance between the reference vectors increases. This should place a constraint on the maximum distance between a pair of simultaneously update-able reference vectors. It is not evident, however, that this additional term completely redresses the overly powerful repulsive force of LVQ2.1.

Figure 2 in [7] shows an example comparison of the reference vectors positions after training using the MEP algorithm, the resultant decision boundary, and the optimal Bayesian decision boundary. The task was to differentiate two bivariate normal distributions having the same mean and different covariant matrices. The Bayesian classification error was 2.7%, the MEP classification error was 3.8% (the classification error associated with LVQ2.1 was 5%.) The decision boundary formed by MEP completely encloses the Bayesian decision boundary. The lower classification accuracy of the MEP algorithm compared to Bayesian classification is due to the vectors being pushed too far away from the center where the conditional probability densities are highest. This is exactly what we would also expect with LVQ2.1.

Diamantini demonstrates that LVQ1 is a sub-optimal approximation of the optimal Bayesian decision boundary, by illustrating how a different configuration of the final reference vector positions would result in a higher classification accuracy. Since the decision boundary formed by MEP in the example problem described above completely encloses the Bayesian decision boundary, this same argument also applies to MEP. Thus, in so far as the difference between LVQ2.1 and MEP does not significantly affect the well documented tendency of LVQ2.1 to fail to approximate the class distributions because of an over-powerful repulsive force away from the data vectors, and in so far as this effect is undesirable and results in a sub-optimal reference vector configuration and a lower classification accuracy, the MEP algorithm, for all its possible other benefits, has not resolved the most glaring shortcoming of LVQ2.1.

Chapter 9

CONCLUSION

The objective of this study was to determine the potential of using learning vector quantization as a method to classify between different spontaneous EEG signals, as well as to analyze learning vector quantization as a general classification method.

The best classification accuracy was obtained with unnormalized, AR(6) coefficients derived from raw, unfiltered EEG signals. The fact that a Vector Quantization architecture does not require normalized data allows LVQ to capitalize on any information contained in the differences in the variances of the coefficients of the input vectors. The geometric nature of a Vector Quantization architecture also gives a means of interpreting what the algorithm has learned that is not available with neural network architectures using hidden nodes.

The classification results obtained demonstrate that LVQ is competitive with using an artificial neural network as a method for classifying spontaneous EEG signals. The highest classification accuracy, approximately 80% correct classification, was obtained using LVQ2.1 with 16 reference vectors per class and a learning rate of 0.1. The LVQ2 and LVQ3 algorithms were relatively insensitive to the value of the window width parameter and the method of initialization. LVQ2.1 exhibited only a moderate degree of over-fitting through the use of an excessive number of reference vectors and showed no signs of overfitting through excessive training.

LVQ2.1 yields the highest classification accuracy despite its failure to approximate the class distributions. Variants on LVQ2.1, e.g. LVQ3, successfully guarantee continued rough approximation of the class distributions, but yield the same or worse classification accuracy. The lack of LVQ2.1 to provide long term approximation of the class distributions is the result of the difficulties in balancing the force pushing reference vectors away from data vectors and force pulling them closer. The repulsive force is the direct consequence of a function transformation which appears to be motivated for practical as opposed to theoretical reasons.

The MEP algorithm of Diamantini has demonstrated a much better approximation of the optimal Bayesian classification accuracy than LVQ2.1 for certain distributions. Its not clear, however, that the MEP algorithm provides significantly better long term approximation of the class distributions than LVQ2.1. It is argued that at least a rough approximation of the class distributions is necessary for optimal classification. It is evident that by the same method Diamantini uses to argue that LVQ2.1 is suboptimal because there exists alternate reference vector positions that would more accurately approximate the optimal Bayesian decision boundary, the MEP algorithm can also be seen to be suboptimal, though perhaps to a lesser degree. This observation appears inconsistent with the fact that the MEP algorithm is presumed to be derived with the exact goal of minimizing misclassification error probability. Diamantini has proven this is not the the exact goal of LVQ1, and that the analytic form of LVQ1 is different from that claimed by Kohonen.

One of Kohonen's principal weaknesses in providing theoretical motivations for his algorithms is the lack of importance and rigor that is given to the treatment of labels. It is the vector labels that differentiate the problem of pure function approximation from minimizing misclassification error. Intimately connected with vector labels is the importance of initialization and the fact that LVQ is a local search technique that uses the nearest neighbor classification rule. It is the nearest neighbor classification rule that requires the segregation of the reference vectors by labels at the Bayesian decision boundary. This combined with the the fact that LVQ is a local search technique gives rise to the theoretical importance of initialization. Empirically, however, it was found that initialization had little impact on the classification accuracy.

The classification task was limited to intra-subject classification of AR(6) models of the EEG signals of two of the five mental task collected in association with the research of Keirn and Aunon. One of the most obvious extensions is to attempt a simultaneous classification of all five of the mental tasks. Inter-subject classification is potentially an even more difficult problem. Most EEG classification research has found that the signal preprocessing techniques used are more important in terms of overall classification accuracy then the classification procedure. A great many signal processing techniques exist whose influence in classifying EEG signals is unknown. Neither the use of DSLVQ as a signal preprocessor to filter out noisy coefficients, nor the potential benefit of combining Hidden Markov Models with LVQ was explored. Lastly, very little is known about the optimal conditions for reducing the noise associated with recording cognitive signals. In addition to the position of the electrodes and the conditions under which the recording sessions are performed there also exist other methods of recording cognitive signals that have received far less attention than EEG recordings.

The theoretical analysis of LVQ in this study has been empirically and intuitively motivated without much attention to mathematical rigor. Aside from Kohonen the only critical analysis of the potential and difficulties in supervised vector quantization classification known to this author is the work of Diamantini. There appears to be a great deal of work left in understanding the exact theoretical foundations of the current versions of LVQ, its limitations, and the development of an optimal VQ classification algorithm for minimizing misclassification error probability.

LVQ has been shown to be as successful as other methods for EEG signal classification. It's potential, however, has not yet been reached. Besides its use in combination with signal pre-processing time series modeling techniques, there remain some fundamental questions as to the mathematical goal of the various LVQ algorithms, the application of stochastic approximation methods to a complex gradient with two opposing directions, and the optimal method of using the Vector Quantization architecture to approximate the Bayesian decision boundaries to minimize the misclassification error probability.

REFERENCES

[1] Charles W. Anderson, Saikumar V. Devulapalli, and Erik A. Stolz. EEG signal classification with different signal representations. *Neural Networks for Signal Processing*, V:475-483, 1995.

[2] M. Anthony and N. Briggs. *Computational Learning Theory*. Cambridge University Press, 1992.

[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929-965, October 1989.

[4] David Cohn, Eve Riskin, and Richard Ladner. Theory and practice of vector quantizers trained on small training sets. Technical Report 8, University of Washington, 12 1992.

[5] Duane DeSieno. Adding a conscience to competitive learning. In *Proceedings of the IEEE International Conference on Neural Networks, ICANN88*, volume I, pages 117-124, New York, 1988.

[6] C. Diamantini and A. Spalvieri. Quantizing for bayes risk. In *Proceedings of the Internatinal Symposium on Information Theory and its Applications, ISITA94*, volume II, pages 715-720, Sidney, Australia, November 1994.

[7] C. Diamantini and A. Spalvieri. Vector quantization for minimum error probability. In *Proceedings of the International Conference on Artificial Neural Networks, ICANN94*, volume II, pages 1091-1094, Sorrento, Italy, May 1994.

[8] Claudia Diamantini and Arnaldo Spalvieri. Certain facts about kohonen's LVQ1 algorithm. In *Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS94*, volume VI, pages 427-430, London (UK), June 1994.

[9] Claudia Diamantini and Arnaldo Spalvieri. Certain facts about kohonen's LVQ1 algorithm. *IEEE Transactions on Circuits and Systems*, 43(5):425-427, May 1996.

[10] D. Flotzinger. Neural network-based classification of spatiotemporal EEG-data. Master's thesis, Graz University of Technology, Graz, Austria, October 1991.

[11] D. Flotzinger, J. Kalcher, and G. Pfurtscheller. Suitability of learning vector quatization for on-line learning: A case study of EEG classification. In *Proc. WCNN'93, World Congress on Neural Networks*, volume I, pages 224-227, Hillsdale, NJ, 1993. INNS, Lawrence Erlbaum.

[12] D. Flotzinger, G. Pfurtscheller, Ch. Neuper, J. Berger, and W. Mohl. Classification of non-averaged EEG data by learning vector quantisation and the influence of signal preprocessing. *Medical and Biological Engineering and Computing*, 32:571-576, September 1994.

[13] D. Flotzinger, M. Pregenzer, and G. Pfurtscheller. Feature selection with distinction sensitive learning vector quantisation and genetic algorithms. In *International Conference on Neural Networks*, volume 6, pages 3448-3451, 1994.

[14] Pfurtscheller G. Mapping of event-related desynchronization and type of derivation. *Electroencephalograph and Clinical Neurophysiology*, 70:190-193, 1988.

[15] Thickbroom G.W., Mastaglia F.L., Carroll W.M., and Davies H.D. Source derivation: application to topographic mapping of visual evoked potentials. *Electroencephalograph and Clinical Neurophysiology*, 59:279-285, 1984.

[16] B. H. Jansen. Time series analysis by means of linear modelling. In *Digital Biosignal Processing*, volume 5 of *Techniques in the Behavioral and Neural Sciences*, pages 157–180. Elsevier, Amsterdam, 1991.

[17] H. Jasper. The ten twenty electrode system of the international federation. *Electroencephalographic Clinical Neurophysiology*, 10:371–375, 1958.

[18] John W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401-409, 1969.

[19] J. Kalcher, D. Flotzinger, S. Golly, Ch. Neuper, and G. Pfurtscheller. Graz brain-computer interface (BCI) II. In *Proceedings of the 4th International Conference, ICCHP Vienna*, pages 170-176. ICCHP, 1994.

[20] Joachim Kalcher, Doris Flotzinger, and Gert Pfurtscheller. A new approach to a brain-computer-interface (BCI) based on learning vector quantization (LVQ3). In *IEEE Engineering in Medicine and Biology 14th Annual Conference*, volume 4, pages 1658-1659, 1992.

[21] Zachary A. Keirn. Alternative modes of communication between man and machine. Master's thesis, Purdue University, 1988.

[22] Zachary A. Keirn and Jorge I. Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical Engineering*, 37(12):1209–1214, December 1990.

[23] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical report, Baskin Center for Computer Engineering and Information Sciences University of California, Santa Cruz, June 1994.

[24] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1984.

[25] Teuvo Kohonen. The self-organizing map. In *IEEE Proceedings*, pages 1464-1480, September 1990.

[26] Teuvo Kohonen. Advances in the theory and applications of the self-organizing map. In *Proc. NIPS'94, Neural Information Processing Systems*. IEEE Service Center, 1994.

[27] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin Heidelberg, 1995.

[28] Teuvo Kohonen, Jari Kangas, Jorma Laaksonen, and Kari Torkkola. LVQ_PAK: A program package for the correct application of learning vector quantization algorithms. In *Proc. IJCNN'92, Int. Joint Conf. on Neural Networks*, volume I, pages 725-730, Piscataway, NJ, 1992. IEEE Service Center.

[29] Shiao-Lin Lin, Yi-Jean Tsai, and Cheng-Yuan Liou. Conscious mental tasks and their EEG signals. *Medical & Biological Engineering & Computing*, 31:421–425, 1993.

[30] Nikola Masic and G. Pfurtscheller. Neural network based classification of single-trial EEG data. *Artificial Intelligence in Medicine*, 5:503-513, February 1993.

[31] Nikola Masic, Gert Pfurtscheller, and Doris Flotzinger. Neural network-based predictions of hand movements using simulated and real EEG data. *Neurocomputing*, 7:259-274, February 1995.

[32] M. Peltoranta and G. Pfurtscheller. Neural network based classification of non-averaged event-related EEG responses. *Medical and Biological Engineering and Computing*, 32:189-196, March 1994.

[33] G. Pfurtscheller, D. Flotzinger, W. Mohl, and M. Peltoranta. Prediction of the side of hand movements from single-trial multi-channel EEG data using neural networks. *Electroencephalography and Clinical Neurophysiology*, 82(4):313-315, April 1992.

[34] G. Pfurtscheller, M. Pregenzer, and C. Neuper. Visualization of sensorimotor areas involved in preparation for hand movement based on classification of mu and central beta rhythms in single EEG trials in man. *Neuroscience Letters*, 181:43-46, September 1994.

[35] Gert Pfurtscheller, Doris Flotzinger, and Joachim Kalcher. Brain-computer interface-a new communication device for handicapped persons. *Journal of Microcomputer Applications*, 16:293-299, February 1993.

[36] Gert Pfurtscheller, Doris Flotzinger, and Christa Neuper. Differentiation between finger, toe and tongue movement in man based on 40 hz EEG. *Electroencephalography and clinical Neurophysiology*, 90:456-460, January 1994.

[37] Nunez P.L. Methods to estimate spatial properties of dynamic cortical source activity. In G. Pfurtscheller and F.H. Lobes da Silva, editors, *Functional Brain Imaging*, pages 3-9. Hans Huber, Toronto, 1988.

[38] W. Poechmueller, M. Glesner, and H. Juergs. Is LVQ really good for classification? - an interesting alternative. In *International Conference on Neural Networks*, volume 3, pages 1207-1212, Piscataway, NJ, 1993. IEEE Service Center.

[39] M. Pregenzer, D. Flotzinger, and G. Pfurtscheller. Distinction sensitive learning vector quantisation - a new noise-insensitive classification method. In *International Conference on Neural Networks*, volume 5, pages 2890-2894, 1994.

[40] Michel Verleysen, Philippe Thissen, and Jean-Didier Legat. Linear vector classification: An improvement on lvq algorithms to create classes of patterns. In J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation, Lecture Notes in Computer Science No. 686*, pages 340-345, Berlin, Heidelberg, 1993. Springer.