Unsupervised Classification Learning from Cross-Modal Environmental Structure

by

Virginia Ruth de Sa

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Dana H. Ballard

Department of Computer Science
College of Arts and Science

University of Rochester
Rochester, New York

1994

Acknowledgments

My first thanks go to my advisor Dana Ballard for giving me the freedom and support to do something different (and often taking heat for it). His ability to look at things from a broad perspective and energetic presentation style are things I will always strive for. I am also greatly indebted to the other members of my committee. Robbie Jacobs provided both the necessary depth, and general perspective of the neural network field. Randal Nelson's critical mind has on several occasions provided new insights and been a good sounding board for technical ideas. John Maunsell set me straight on the neurobiology and his nose for good science helped in the early inception of the thesis. Mary Hayhoe provided the psychologist's view and many times helped to make the work more interesting to a larger audience.

I owe a huge thank you to Ramprasad Polana for performing the visual analysis for the data in Chapter 4. Also thanks to Ramesh Sarukkai and Brad Miller for assistance with the auditory analysis. Many thanks to the "subjects" Marc Light, Michael Marchetti, Eric Ringger, Randal Nelson, Jim Müller and especially my patient pilot subject Jeff Schneider. I'd like to thank Steve Nowlan for sending me the Peterson-Barney vowel dataset used in the bench-marking experiments and Leonidas Kontothanassis for his help in automating the analysis of the results. Also many, many thanks to Sue Becker for running the IMAX experiments with the same dataset. And a special thank you to the "volunteer software support group" of Lawrence Crowl, George Ferguson, Leonidas, Jonas Karlsson and Massimo Poesio, who always had time to solve all my technical emergencies.

For their help in the adjustment to Computer Science and Ph.D. research, many thanks to Jack Veenstra, Lambert Wixson and especially Alan Cox. I'm sure that I learnt more from them than from all the books and courses. I'd also like to thank Barak Pearlmutter, Steve Nowlan and Kevin Lang for bringing me into the neural network fold and for many interesting late-night discussions. For helpful comments on papers and ideas along the way, I am very grateful to Leonidas, Jeff, Andrew McCallum, Justinian Rosca and the rest of the Rochester Envision group. Also many thanks to the people in the Center for Visual Science that provided a stimulating environment for discussing vision, attention and neural processing. I'd especially like to thank Ken McRae and Michael Spivey-Knowlton for the interesting collaborations I've had with them.

The department at Rochester is one of the most closely knit and friendly groups. It would be impossible to mention all the people that made Rochester a fantastic place to

be. I would like to especially thank Alan, Jack, Lambert, Janet Hitzeman, Jim, Ramprasad, Srilatha Polana, George, Polly Pook and Marc Light, "the illegal aliens" Mike Swain, Leo Hartman, and Massimo and "the house" Bill Garrett, Leonidas, Andrew, Bob Wisniewski, and Raj Rao.

My final thanks are for the people who have enriched my life in numerous ways. To John Cross and Leonidas who have done more for me than I could ever recount. To my parents and grandparents for teaching me to work hard and giving me the support and confidence to try anything. And to Jeff for brightening my days with his love, support, and understanding.

The simulations in this dissertation benefited greatly from the neural network simulator SN2.5 from Neuristique inc. as well as the LVQ_PAK and SOM_PAK code from the Helsinki University of Technology, Labratory of Computer and Information Science.

This research was generously supported by a Canadian NSERC 1967 Science and Engineering Scholarship, NSF research grant no. IRI-8903582, NIH/PHS research grant no. 1 R24 RR06853-02, and a Human Science Frontiers Program research grant.

Abstract

This dissertation addresses the problem of unsupervised learning for pattern classification or category learning. A model that is based on gross cortical anatomy and implements biologically plausible computations is developed and shown to have classification power approaching that of a supervised discriminant algorithm.

The advantage of supervised learning is that the final error metric is available during training. Unfortunately, when modeling human category learning, or in constructing classifiers for autonomous robots, one must deal with not having an omniscient entity labeling all incoming sensory patterns. We show that we can substitute for the labels by making use of structure between the pattern distributions to different sensory modalities. For example the co-occurrence of a visual image of a cow with a "moo" sound can be used to simultaneously develop appropriate visual features for distinguishing the cow image and appropriate auditory features for recognizing the moo.

We model human category learning as a process of minimizing the disagreement between outputs of sensory modalities processing temporally coincident patterns. We relate this mathematically to the optimal goal of minimizing the number of misclassifications in each modality and apply the idea to derive an algorithm for piecewise linear classifiers in which each network uses the output of the other networks as a supervisory signal.

Using the Peterson-Barney vowel dataset we show that the algorithm finds appropriate placement for the classification boundaries. The algorithm is then demonstrated on the task of learning to recognize acoustic and visual speech from images of lips and their emanating sounds. Performance on these tasks is within 1-7% of the related supervised algorithm (LVQ2.1).

Finally we compare the algorithm to Becker's IMAX algorithm and give suggestions as to how the algorithm may be implemented in the brain using physiological results concerning the relationship between two types of neural plasticity, LTP and LTD, observed in visual cortical cells. We also show how the algorithm can be used as an efficient method for dealing with learning from data with missing values.

Table of Contents

Acknowledgments ii

Abstract iv

List of Tables vii

List of Figures viii

1 Introduction 1

1.1 Motivation 1
1.2 Neuroscience Background 4
1.3 Neural Network Background 6
1.4 Supervised Learning vs Unsupervised Learning 9
1.5 Contributions 15
1.6 Outline of Dissertation 16

2 Adding Supervision to Competitive Learning 17

2.1 Motivation 17
2.2 Method 1: Supervised Competitive Learning 18
2.3 Method 2: LVQ Algorithms 25
2.4 Summary 32

3 Cross-Modal Information 35

3.1 Motivating Biology 36
3.2 Using Cross-Modality Information for Self-Supervised Learning 39
3.3 A Self-Supervised Piecewise-Linear Classifier 46
3.4 Summary 47

4 Experimental Results 49

4.1 Benchmark Experiments 49
4.2 Cross-Modal Experiments 59

5 Other Theoretical Issues **64**

5.1 Minimizing Disagreement through LTD 65

5.2 Comparison to Maximizing Mutual Information 67

5.3 Minimizing Disagreement as an Efficient Missing Data Solution 74

6 Conclusions **76**

6.1 Summary of Results 77

6.2 Discussion 77

6.3 Contributions 78

6.4 Future Work 79

Bibliography **81**

A Discussion of Rumelhart & Zipser's Supervised Competitive Learning **87**

B Description of the Window Training Procedure from [Wassel & Sklansky, 1972] **90**

C Derivation of the Minimizing Disagreement Algorithm **94**

List of Tables

4.1 Tabulation of performance figures (mean percent correct and sample standard deviation over 60 trials and 2 modalities). 55

List of Figures

1.1 An example neuron showing the major parts. 5
1.2 Computation in the McCulloch-Pitts neuron. 8
1.3 A three layer feed-forward network. 10
1.4 Competitive learning architecture. 12
1.5 Weight movement in competitive learning. 14

2.1 An architecture for adding teaching inputs. 18
2.2 Our supervised competitive learning architecture. 19
2.3 The XOR problem with 4 hidden-layer neurons. 21
2.4 The XOR problem with 3 hidden-layer neurons and no teach input . . . 22
2.5 The XOR problem with 3 hidden-layer neurons and a teach input. . . . 22
2.6 The result of removing the teach input after training as above. 23
2.7 The XOR problem with 2 hidden-layer neurons and a teach input. . . . 23
2.8 The Result of removing the teach input after training as above. 24
2.9 Example input patterns in the square/triangle experiment. 24
2.10 The weights of the hidden layer neurons with no teach input. 26
2.11 The weights of the hidden layer neurons after the addition of teach input and recurrent connections 26
2.12 The problem with the supervised competitive algorithm 27
2.13 A one-dimensional classification problem of two classes. 30
2.14 The driving forces in the LVQ2 and LVQ2.1 algorithms for overlapping distributions. 32
2.15 The driving forces in the LVQ2 and LVQ2.1 algorithms for separable distributions. 33
2.16 Performance of the different algorithms on three different classification problems. 34

3.1 The idea behind the self-supervised algorithm. 36

3.2 Network for learning the labels of the codebook vectors. 40
3.3 An example world as sensed by two different modalities. 41
3.4 Probability distributions used in the calculation of the energy function plots shown in Figures 3.5 , 5.2 , 5.3 , 5.4 , 5.5 and 5.5 42
3.5 The Disagreement energy function as a function of b_1 and b_2 for the particular distributions shown in Figure 3.4 43
3.6 An example of the joint and marginal distributions for the example problem introduced in Figure 3.3 45
3.7 The Minimizing-Disagreement (M-D) algorithm—a self-supervised piecewise-linear classifier. 48

4.1 The Peterson-Barney vowel dataset. 50
4.2 A close-up of part of the Peterson-Barney vowel dataset. 51
4.3 Misclassified patterns after initial labeling algorithm on the Peterson-Barney dataset. 52
4.4 Misclassified patterns after Minimizing-Disagreement algorithm on the Peterson-Barney dataset. 53
4.5 Improvement from running the M-D stage. 54
4.6 The performance (for 30 different initial configurations). 55
4.7 Difference in performance after 1 or 2 iterations of the M-D algorithm. . 56
4.8 Results from different algorithms on the Peterson-Barney Vowel Dataset. 57
4.9 Difference in performance between the supervised and M-D algorithm. . 58
4.10 Example frames showing the clap detection. 59
4.11 Example /ba/ utterance. 60
4.12 Example /va/ utterance. 60
4.13 Example /da/ utterance. 60
4.14 Example /ga/ utterance. 61
4.15 Example /wa/ utterance. 61
4.16 Results on the preliminary cross-modal dataset. 62
4.17 Results on the single-speaker cross-modal dataset. 63

5.1 Connections required to enable appropriate synapse modification in modality 1 according to the Minimizing-Disagreement algorithm 66
5.2 H(Y_1) as a function of b_1 and b_2 for the distributions shown in Figure 3.4 . 69
5.3 H(Y_2) as a function of b_1 and b_2 for the distributions shown in Figure 3.4 . 70
5.4 -$H(Y_1, Y_2)$ as a function of b_1 and b_2 for the distributions shown in Figure 3.4 71
5.5 $I = H(Y_1) + H(Y_2) - H(Y_1, Y_2)$ as a function of b_1 and b_2 for the distributions shown in Figure 3.4 72

5.6 The Disagreement energy function as a function of b_1 and b_2 for the distributions shown in Figure 3.4 73

5.7 Example classification problem after [Ahmad and Tresp, 1993] 75

A.1 Supervised competitive learning architecture proposed in [Rumelhart and Zipser, 1986] 88

A.2 Our supervised competitive learning architecture. 88

1 Introduction

1.1 Motivation

1.1.1 Gaining Insight about the Brain through Modeling

This work is motivated by a desire to understand how humans and animals are able to learn the complicated tasks they do. The particular task of interest in this dissertation is that of learning to classify— learning to recognize that particular patterns belong to the same class which is different from the class of a set of other patterns.

Progress in understanding learning in the brain has benefited from two general approaches. The empirical approach involves examining actual human and animal brains or behaviour. The modeling approach involves constructing models at various levels of abstraction and examining their properties. This dissertation takes a modeling approach to understanding how we learn to classify patterns into different classes.

Modeling is an important complement to empirical studies. While empirical studies suggest theories about the computational processing, models can provide additional insight and point out subtle difficulties that may only be apparent when actually trying to implement the proposed ideas; the implementation task requires that the theory be thoroughly and precisely defined. Models depend on knowledge gained from empirical studies but in turn are able to suggest or predict new experiments that can augment the knowledge base.

In the cognitive science literature there are many models of cognitive processing and it is often hard to discriminate between them. In general, models that account for more of the data or can explain more findings are better. Along these lines, a model that can address the issue of the development or learning of the particular task in addition to providing an explanation for the computational algorithm behind the task should be preferable. Thus a model that describes how we recognize objects may provide more insight if it includes a description of how this ability develops.

The learning question is also extremely important in its own right. It is mathematically impossible that all 10^{15-16} connections in the human brain be individually specified genetically in the DNA. There are also many physiological experiments that show an activity dependence in the patterns of neuronal connectivity and resulting capabilities of animals reared in deprived environments. Thus it seems that useful models for

high level tasks such as object recognition will eventually require a biologically plausible method for learning at least part of the resulting algorithm.

1.1.2 Connectionist Modeling to Address Learning

One useful modeling paradigm that addresses learning is that of connectionist modeling. The goal of connectionist modeling is to explain human learning and cognition in terms of primitives that are abstractions of the brain's neurons. Depending on how abstract the primitives are, these models can also be useful in providing a mapping to the neural substrate that may be investigated experimentally.

There have been many examples of applications of connectionist algorithms that are able to provide insight into the computational processes in the brain. One example is the shape from shading work of [Lehky and Sejnowski, 1988]. Lehky and Sejnowski trained a network to output the curvature of objects presented as two-dimensional images. The trained network developed neurons with response preferences similar to the so-called "edge-detectors" found in the first visual cortical area (area V1) of cats and monkeys (and thought to be present in humans). This work was important because it raised the possibility that the receptive fields in V1 may play a more important role in shape recognition than just detecting edges.

Similarly the work of Zipser and Andersen [Zipser and Andersen, 1988] showed how the cells in another cortical area, area 7a, are useful in the computation of the spatial transformation from retinal centred coordinates to head centred coordinates. Their model gave an explanation for the enigmatic response properties (receptive fields) in this area. These receptive fields are neither fully retinotopic nor fully head-centered and have a non-linear interaction between the visual response and eye position signal. Their model also matched the spatial gain fields — measurements of the variation in response to an optimally placed stimulus (in retinotopic coordinates) — in varying linearly across eye positions.

The technique used in both the above papers was to train a network with many examples of hypothesized inputs and desired outputs and then observe the resulting receptive field structure (as determined by the pattern of connectivity from the input neurons). The logic behind this work is that if a network constructed to do a task using abstract neuronal elements results in receptive fields similar to those observed in the brain, it is possible that the brain is performing a similar computation. While this argument has led to many computational insights such as those mentioned above, the advances in understanding the brain are somewhat diminished in the use of a biologically implausible learning algorithm that can't address the question of how the particular computational network might develop in the brain. The learning algorithm used is known as "back-propagation" [Werbos, 1974; Rumelhart *et al.*, 1986] and requires both that the desired target signal be given for every input pattern, and that the error signal be calculated and sent backwards along the model axons for use by the previous neurons.

Many models of the development of the early areas of the visual system [Linsker, 1986a; Linsker, 1986b; Linsker, 1986c; Miller *et al.*, 1989; Obermayer *et al.*, 1990; Obermayer *et al.*, 1992] use more biologically plausible learning rules. These models

provide a mechanism for the development of the orientation columns, ocular dominance columns and even the particular patterns of interaction observed. The learning algorithm in these models depends only on the correlations between input patterns, and maps very well to neurophysiological ideas about the rules governing neural plasticity. The correspondence between the biologically motivated network models of the first visual area (V1) and the known anatomy and physiology of monkey and cat V1 are extremely striking. However, they have been limited to modeling only this and earlier stages of processing and do not address the particular cognitive task in which an area may be participating. It seems that in order to learn high-level cognitive tasks, some indication of the desired outputs are needed during training.

1.1.3 Self-Supervised Learning for Power and Biological Plausibility

This problem of requiring desired outputs to learn cognitive tasks is not unique to connectionist models, many cognitive models assume that the answer is somehow magically available to the learner during learning. These algorithms are called *supervised* algorithms and are tantamount to assuming that the desired mapping to be learned has already been learned by some entity whose output is available to the output of the training model!

This raises the question — Is there some way to model high level cognitive tasks using more biologically plausible learning algorithm? The goal of my research is to work toward this ideal. The dissertation attempts to address this question for the particular task of object recognition or classification. The major idea is that, while it is implausible that the neuronal output label is available, it is true that the environment is providing extra information not currently considered in most unsupervised algorithms (algorithms that do not provide the target signal).

There are many sources of environmental constraining information but the one emphasized in this dissertation is *cross-modal structure*. By cross- or inter-modal structure we mean the relationship between patterns impinging on the different sensory modalities. For instance the sight of a rose tends to co-occur with a particular scent. Sights of cows often co-occur with "moo" sounds.

These correlations have been noted before and are often used for justification in supervised cognitive models. Thus providing a 'cow' label with cow images (in a model learning to recognize animals) may be justified with the statement that "an infant is told 'cow' when shown a cow." The major point of disagreement of this dissertation from the supervised cognitive models stems from the observation that the spoken word 'cow' is not a useful teaching signal until the auditory system has learned to correctly parse and classify speech signals. This is immediately apparent to those who have tried to build a machine speech recognition system or even observed sound spectrograms of spoken words. Similarly, as computer vision researchers are well aware, the cow picture is not a useful teaching signal for the 'cow' acoustic signal until the visual system has correctly learned to recognize cows. Thus although the world provides extra information in cross-modal structure it is *not* in the form of the correct neuronal target output.

To exploit the structure in the temporally co-occurring cross-modality signals the model must address the question of how both recognizers can develop at the same time. This dissertation develops an algorithm for two (or more) networks to develop together and provide "teaching" information to each other as they learn. Because the networks help each other we refer to this kind of algorithm as *self-supervised*.

Beside offering a model of cognitive development, the algorithm is also important from a machine learning point of view. Current robots and artificial intelligence (AI) application programs are often pre-programmed or trained off-line with pre-collected human-labeled patterns. Pre-programming does not work well when the implementation of a solution is not well understood as is the case for most interesting AI problems. While great strides have been made in speech recognition, computer vision, and natural language understanding, for the most part these tasks that humans perform so easily have eluded comparable computer solutions. As we are not able to describe algorithms in sufficient detail for acceptable computer implementations, approaches in which the computer learns its own algorithm are desirable. Supervised learning, where many input patterns are presented with the correct output solution, is often used and has been reasonably successful but the hand labeling of many patterns is often expensive and in autonomous applications even impossible. The algorithm and ideas developed in this dissertation could be used to allow for self-acquisition and training without human intervention. The algorithm would allow the computer to learn by observing the world from different sensors.

1.2 Neuroscience Background

This section gives a brief overview of some of the relevant information from neuroscience. It will be useful for understanding some of the motivation behind the neural network rules in the next section as well as for understanding the sections that discuss the biological plausibility of different learning algorithms.

The human brain has on the order of 10^{12} neurons [Kandel and Schwartz, 1985] (page 13). The characteristic form of a neuron consists of a soma or cell body between a set of dendrites and an axon (see Figure 1.1). From a very simplified view one can think of the dendrites as the receiving end and the axon as the output channel.

One of the most important properties of a neuron is its potential, or voltage difference between the inside and outside of the cell. This potential is caused by a balance between two forces—diffusion, where ions will tend to move away from areas of high concentration, and electric potential difference, where positive ions tend to move toward areas of more negative potential (and negative ions toward areas of more positive potential). With ion pumps the concentrations of some ions (K+) are kept much higher inside the cell than outside while that of other ions is kept much lower (Na+,Cl-,Ca++). A membrane permeable to only one ion will allow this ion to diffuse across the membrane carrying charge that will tend to change the potential across the membrane. This change in potential will eventually balance the diffusion force and an equilibrium voltage will exist in which there is no net flow of ions across the membrane. A neuron's

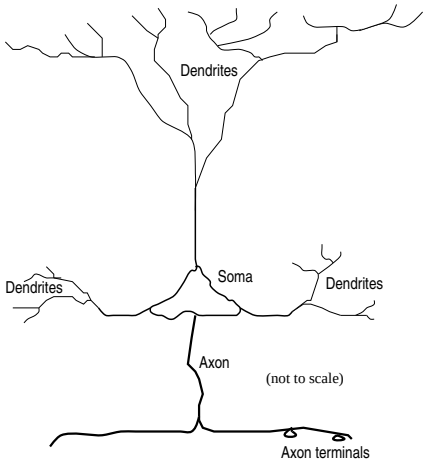

Figure 1.1: **An example neuron showing the major parts.**

equilibrium voltage depends on the concentration differences, the ion charges, and the relative permeabilities of the different ions in the environment. Signaling via membrane potential change could theoretically be accomplished by changing any of these but changing the permeabilities allows for quick voltage changes and is the basis for neuronal communication.

Communication between neurons occurs for the most part at synapses and the most common type of synapse, axodendritic, is one from an axon of one neuron, the *pre-synaptic* neuron, to a dendrite on another, *post-synaptic*, neuron. At a synapse the pre-synaptic neuron releases a neurotransmitter which acts on receptors of the post-synaptic neuron. The activation of the receptor serves to open channels for one or more types of ions which increases the permeability to that ion and tends to move the cell potential more toward the equilibrium potential for that ion. An increase in interior potential is called depolarization and a decrease, hyperpolarization. Changes in membrane potential are summed over space (from different synapses) and time, with some decay, and propagate to the soma and the initial segment of the axon (the axon hillock). Potential above a certain threshold at the axon hillock causes activation of a spike or action potential. This spike travels down the axon. At the ends of the axons, the axon terminals, the action potentials cause release of neurotransmitters which in turn can influence the dendrites of neurons post-synaptic to them.

There are many different types of receptors but one in particular is thought to be important for learning— the NMDA receptor. This receptor is special because it depends not only on the presence of a particular transmitter (glutamate) but also a depolarized membrane to increase its conductance. This means that in order for the channels associated with the NMDA receptor to open, the cell must experience activation from both the cell pre-synaptic to the receptor and also a cell pre-synaptic to another non-NMDA receptor (to allow for the depolarization). There has been much evidence concerning the association between the NMDA receptor and a type of neural plasticity called long term potentiation or LTP—a long-lasting increase in synaptic strength (measured by observing the post-synaptic response from stimulating the pre-synaptic neuron) following

strong stimulation.

Recently more has been discovered about a form of plasticity known as long term depression (LTD). Homosynaptic LTD refers to a long lasting decrease in synaptic strength following stimulation of the pre-synaptic neuron. It has been found[Artola *et al.*, 1990] that this occurs in the same cells for which LTP can be induced where the deciding factor is the post-synaptic potential (and resulting [Ca++] increase). There are two thresholds for changing synapse strength. If the post-synaptic potential is between the two thresholds, long term depression is observed, whereas if the potential is above the highest threshold long term potentiation occurs.

Cortical connectivity is known to be extremely sparse. Most neurons synapse on about 10^4 other neurons—a small fraction of the total. First there is a major division between areas. Axons from the retina synapse on different subcortical structures which in turn project to different cortical areas than axons from the auditory system. Within the different sensory pathways there is also a large degree of segregation. For instance in the visual system the neurons sensitive to colour tend to be separate from others. Within each system there is also a hierarchy of processing with one area projecting mainly to the few areas just above it and projecting back to the few lower areas that feed it. There is also micro-segregation within an area in that nearby neurons have similar response preferences and communicate with each other. The neurons are connected with a network of *lateral inhibition*. They synapse on inhibitory cells that in turn feed back onto neighbouring cells. These kinds of networks are thought to be important in sharpening responses so that fewer stimuli are able to activate each cell. Finally there is columnar organization where the majority of a neuron's connections are to cells in a vertical column with similar response preferences. The column seems to act as a processing unit with some cells responsible for input, others for output, and others for normalization.

To summarize, neurons communicate through firing action potentials, and releasing neurotransmitters. Each neuron performs a spatio-temporal sum of its many incoming signals to determine its potential and firing rate. Changes in the strength of the synapses occur due to relationships between the firing of the pre- and post-synaptic cells. Stronger activation of the post-synaptic cell tends to increase the synaptic strength at active synapses and less activation tends to decrease it. Although there are many neurons in the brain, the connectivity is extremely sparse. Each one tends to synapse on very few other neurons. For the most part neurons tend to communicate with other neurons that are sensitive to the same input dimensions and even more so to those with similar preferences within that signal space.

1.3 Neural Network Background

In this section we introduce the basic notation and ideas from the neural network literature that will be used throughout the thesis. For a thorough introduction to neural networks see [Hertz *et al.*, 1991].

A neural network consists of a set of model neurons with connections between them. Each neuron i has an activation x_i (representing some measure of activation such as potential or firing rate) which is communicated to the other neurons through the connections between them. We represent the connection from neuron i to neuron j with $w_{i,j}$. This weight can be thought of as an abstraction of the strength of all the synapses from neuron i to neuron j.

The model neurons compute a simple function of their input and the connection strengths at the connection from each input. For example, the McCulloch-Pitts neuron [McCulloch and Pitts, 1943] or threshold unit calculates its activation through thresholded spatial summation

$$x_j = \Theta(\sum_i w_{i,j} x_i - t_j) = \Theta(\vec{w_{.,j}} \cdot \vec{x} - t_j) \tag{1.1}$$

where t_j is a bias term associated with each neuron j and $\Theta(\cdot)$ is the threshold function

$$\Theta(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{otherwise} \end{cases} \tag{1.2}$$

The bias term is often considered as a regular weight w_0 connected to an input neuron which receives a constant input x_0 of -1 allowing us to rewrite (1.1) as

$$x_j = \Theta(\sum_i w_{i,j} x_i) = \Theta(\vec{w_{.,j}} \cdot \vec{x}) \tag{1.3}$$

In most of the networks we will be discussing the neurons are arranged in layers with full forward connectivity from the previous layer. The neurons in the first layer have no incoming weights. Their activation is clamped by the input patterns $\vec{\xi}$, one component of the input vector ξ_i for each input neuron i. The final layer contains the output neurons—those we are ultimately interested in observing. We will sometimes refer to the activation of the output neurons as y_j for each output neuron j. There may also be hidden units with activation h_j that are not directly observed but that connect to the output units.

The network computes a function; each input pattern is mapped to an output pattern at the output neurons. For example a McCulloch-Pitts neuron performs a binary classification, dividing the space of inputs into 2 areas with a hyperplane (see Figure 1.2). Input vectors on one side of the hyperplane result in activation 1 and those on the other side result in activation 0. Note that the ratio between w_1 and w_2, the weights from the respective input dimensions, control the orientation of the boundary while the relative magnitude of w_0, the bias weight, determines the displacement from the origin.

1.3.1 Learning by Changing Weights

Changing the strengths of the connections between neurons changes the function that is computed. This is the basis of most neural network learning algorithms. The weight

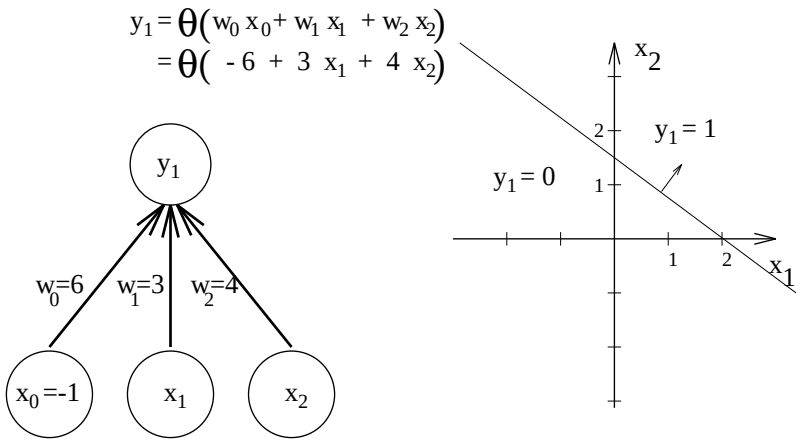

Figure 1.2: **Computation in the McCulloch-Pitts neuron.**

change is usually a function of the activity on both sides of the connection as well as in the case of supervised algorithms the desired output of the receiving neuron.

For example, most unsupervised weight update rules are modifications of one originally proposed by Hebb[Hebb, 1949] as a model of learning in real neurons. The key idea in the Hebbian synapse modification algorithm is to strengthen the connection between a pre- and post- synaptic cell when they exhibit correlated firing as in

$$\Delta w_{i,j} = \epsilon x_i x_j \tag{1.4}$$

where ϵ is the learning rate that reflects the plasticity of the network and is usually a decreasing function of time. This idea is similar to that required in LTP — increase the strength of the connection if the post-synaptic cell has high activation and the pre-synaptic cell was active. The idea is usually generalized to allow for some decay to prevent explosion in the size of the weights. One common method is to weaken the connection for uncorrelated firing [Stent, 1973] as in [Sejnowski, 1977]

$$\Delta w_{i,j} = \epsilon(x_i - \mu_i)(x_j - \mu_j) \tag{1.5}$$

where μ_i is a measure of the average activity of neuron i.

1.3.2 Error Functions and Minimization

Often one can write the desired task of the network as a function for the network to optimize. In these cases the goal is for the learning algorithm to minimize the function with respect to the weights of the network. In most supervised algorithms, the function to be minimized is the sum of some measure of the difference between the actual and desired outputs. The usual mathematical form is given by

$$E(W) = \frac{1}{2} \sum_{\xi} \sum_{i} (y_i^{\xi} - y_i^{*\xi})^2 \tag{1.6}$$

where y_i^{ξ} and $y_i^{*\xi}$ are the actual and desired output respectively of the i^{th} neuron when the network is presented with pattern $\vec{\xi}$.

gradient descent

One common technique for minimizing an energy function is called gradient descent. For neural network learning this involves calculating the gradient of the function with respect to the weights and taking a small step in the opposite direction. The idea is that in order to find the minimum one should proceed downhill. Note that this technique is not designed to find the global minimum but only a local minimum. If the energy function is given by $E(w_{i,j})$, the gradient descent weight update rule is

$$\Delta w_{i,j} = -\epsilon \frac{\partial E}{\partial w_{i,j}} \tag{1.7}$$

stochastic approximation

Calculating the gradient of energy functions that are defined in terms of some function of each pattern in the training set (as for instance with the sum of squared error function above), requires obtaining statistics over a presentation of all the input patterns before the gradient can be calculated. This kind of learning is called batch learning.

Arguably one might prefer a more incremental or on-line learning algorithm both for decreased memory constraints and biological plausibility. In these cases however the exact value of the gradient will not be available. Note that if we took the derivative of just the term related to the presented pattern, we would expect, over time, to do approximately the same thing. This is formalized in the stochastic approximation[Robbins and Monro, 1951] method. The idea is similar to that for gradient descent except that the step is not necessarily in the negative gradient direction but the expected direction of the step is in the negative gradient direction. That is for the Energy function $E(w_{i,j})$, the stochastic approximation rule is

$$\Delta w_{i,j} = -\epsilon Z \tag{1.8}$$

where $Z(n)$ is a random variable at time n whose expected value is equal to the partial derivative with respect to the weight. Mathematically,

$$E[Z(n)|w_{i,j}(n)] = \frac{\partial E}{\partial w_{i,j}(n)} \tag{1.9}$$

1.4 Supervised Learning vs Unsupervised Learning

The principal dichotomy in neural network learning algorithms is between supervised and unsupervised learning. Supervised algorithms are defined as ones in which extra information about the desired output is given. This extra information can be in the

form of the desired output, or in the case of reinforcement learning algorithms a correct/incorrect signal. The most well-known supervised algorithm is the aforementioned back-propagation algorithm.

In unsupervised algorithms only input patterns are received and the algorithm does its best to find intrinsic structure in the pattern distribution. The unsupervised algorithm of interest to us is the competitive learning algorithm [Grossberg, 1976a; Grossberg, 1976b; Kohonen, 1982; Rumelhart and Zipser, 1986]. The next two sections provide descriptions of the back-propagation and competitive learning algorithms and highlight their differences.

1.4.1 Back-Propagation

The back-propagation algorithm [Werbos, 1974; Rumelhart *et al.*, 1986] is the most commonly used supervised algorithm. This algorithm is usually applied to feed-forward networks of three or more layers. (An example of a three layer feed-forward network is shown in Figure 1.3). The algorithm is designed to minimize the squared error energy function given by

$$E(W) = \frac{1}{2} \sum_{\xi} \sum_{i} (y_i^{\xi} - y_i^{\xi *})^2 \tag{1.10}$$

So that the function can be differentiated with respect to the weights, the neurons do not calculate a hard threshold of their inputs as with the McCulloch-Pitts neurons but instead a smoothed approximation, a sigmoidal function, is applied to the weighted sum of the inputs to determine the activation. That is (1.3) and (1.2) are replaced with

$$x_j = f(\sum_i w_{i,j} x_i) \tag{1.11}$$

and

$$f(z) = \frac{1}{1 + e^{-2\beta z}}$$

where β is a parameter that affects the steepness of the sigmoid.

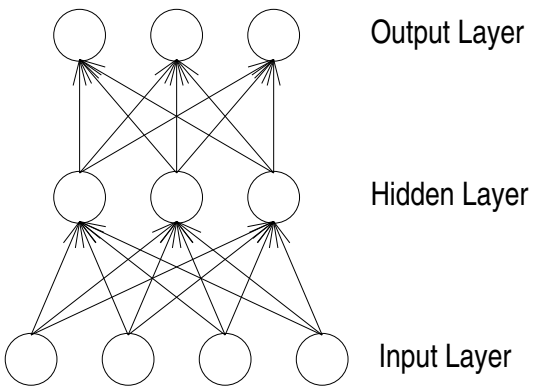

Figure 1.3: **A three layer feed-forward network.**

The algorithm proceeds in two passes. The forward pass propagates the activation according to the smoothed version of the threshold rule given in (1.11). An input pattern vector $\vec{\xi}$ is presented to the input layer and the hidden unit activations are calculated according to

$$\begin{aligned} h_i &= f(\sum_i w_{k,i}\xi_k) \\ &= f(r_i) \end{aligned} \tag{1.12}$$

The output unit activations can then be calculated similarly

$$\begin{aligned} y_j &= f(\sum_j w_{i,j}h_i) \\ &= f(s_j) \end{aligned} \tag{1.13}$$

where we have substituted r_i for the weighted sum of inputs into hidden neuron i and s_j for the weighted sum of inputs into output neuron j. These substitutions are used below.

Once the activation of the output neurons has been determined in the forward pass, the weight updates can be calculated from the derivatives of the energy function. These derivatives depend on the difference between the calculated output $\vec{y}$ and the desired output $\vec{y*}$. The equation for updating the weights from hidden neuron i to output neuron j is

$$\begin{aligned} \Delta w_{i,j} &= -\epsilon\frac{\partial E}{\partial w_{i,j}} \\ &= -\epsilon\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{i,j}} \\ &= \epsilon\sum_\xi (y_j^* - y_j)f'(s_j)h_i \end{aligned} \tag{1.14}$$

Letting

$$\delta_j = f'(s_j)(y_j^* - y_j)$$

we can write

$$\Delta w_{i,j} = \epsilon\sum_\xi \delta_j h_i$$

The δ_j term also appears in the derivative with respect to the weight from input unit k to hidden neuron i

$$\Delta w_{k,i} = -\epsilon\frac{\partial E}{\partial w_{k,i}} \tag{1.15}$$

$$= -\epsilon\sum_\xi\sum_j \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial h_i}\frac{\partial h_i}{\partial w_{k,i}} \tag{1.16}$$

$$= \epsilon \sum_{\xi} \sum_{j} (y_j^* - y_j) f'(s_j) w_{i,j} f'(r_j) \xi_k \tag{1.17}$$

$$= \epsilon \sum_{\xi} \sum_{j} \delta_j w_{i,j} f'(r_i) \xi_k \tag{1.18}$$

$$= \epsilon \sum_{\xi} \delta_i \xi_k \tag{1.19}$$

where

$$\delta_i = f'(r_i) \sum_{j} \delta_j w_{i,j} \tag{1.20}$$

As the weight update for the input to hidden neuron connections depends on the value calculated at each of the output neurons multiplied by the weights from the hidden to the output neurons, they can be efficiently calculated by propagating the δ's backwards along the forward weights. The partial derivative with respect to each weight can be calculated with one backwards sweep of activation (hence the origin of the name—back-propagation). The algorithm is usually applied using the on-line stochastic approximation version of the rules applied after each pattern presentation.

Application of the back-propagation algorithm given a sequence of input-output pairs results in weight vectors that give a local minimum in the mean square error function.

1.4.2 Competitive Learning

Competitive learning [Grossberg, 1976a; Grossberg, 1976b; Kohonen, 1982; Rumelhart and Zipser, 1986] is an unsupervised algorithm that is usually used on 2-layer fully connected feed-forward architectures such as that shown in Figure 1.4.

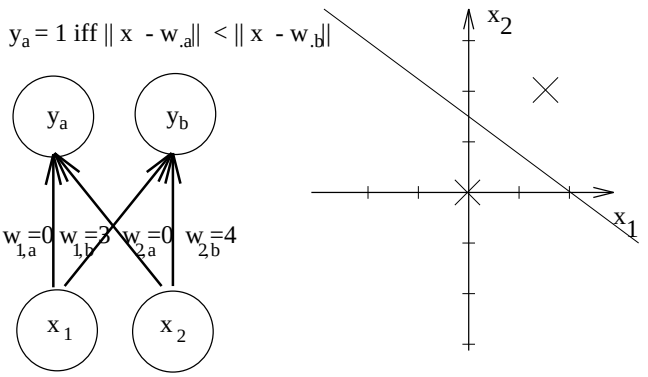

Figure 1.4: **Competitive learning architecture.** The positions of the weights implicitly give the boundaries.

Competitive learning divides the space differently. Instead of each neuron dividing the space with a hyperplane, together the neurons tessellate the space. The space to which a neuron responds depends not only on the value of its weight vector but also on

those of neighbouring weight vectors in the input space. Each neuron responds only to those patterns that are closer to its weight vector than to all the other weight vectors and thus this partitioning of the space is often called a winner-take-all encoding. The borders formed in this way are known as a Voronoi tessellation; an example tessellated space is shown in Figure 1.5. The resulting classifier is often called a *piecewise-linear classifier* due to the shape of the boundaries.

Competitive learning is a clustering algorithm. Whereas the objective with back-propagation is to learn an input-output mapping, the goal of competitive learning is to represent the data more compactly. The idea is to save storage space by instead of storing each individual data pattern storing the values of a smaller number of reference vectors and for each data point storing only the name of the closest reference vector. The goal then is to position the reference vectors to get the smallest square reconstruction error over all the patterns. Mathematically, the learning algorithm performs stochastic approximation with respect to the weight vectors on this discretization error or square reconstruction function [Ritter and Schulten, 1988]

$$\sum_{\vec{\xi}} \|\vec{\xi} - \vec{w}_{j^*(\vec{\xi})}\|^2. \tag{1.21}$$

where $\vec{\xi}$ are the input patterns and $\vec{w}_j$ are the reference or weight vectors. $j^*(\vec{\xi})$ is the closest weight vector defined by $\|\vec{\xi} - \vec{w}_{j^*(\vec{\xi})}\| \leq \|\vec{\xi} - \vec{w}_j\|$ (for all j).

Like back-propagation, the algorithm has a pattern activation stage followed by weight updates. Unlike back-propagation, the output activation is in terms of the winner-take-all encoding — one neuron[1] has activation 1 (or some other fixed value) and the others have activation 0. The neuron with positive activation is the "winning" neuron, the one whose weight vector is closest to the data vector. That is

$$y_{j^*} = \begin{cases} 1 & \text{if } ||\vec{w}_{j^*} - \vec{\xi}|| \leq ||\vec{w}_j - \vec{\xi}|| \text{ for all j} \\ 0 & \text{otherwise} \end{cases} \tag{1.22}$$

For normalized weight vectors the condition for activation 1 is equivalent to the more biologically plausible calculation

$$\vec{w}_{j^*} \cdot \vec{\xi} \geq \vec{w}_j \cdot \vec{\xi} \text{ for all j} \tag{1.23}$$

The weight updates result from the descent of the energy function in 1.21. The winning neuron updates its weights according to:

$$\Delta w_{i,j^*} = \epsilon(\xi_i - w_{i,j^*}) \tag{1.24}$$

where w_{i,j^*} represents the weight from input neuron i to the winning output neuron j^*. This is shown in Figure 1.5. The update rule results in the winning neuron becoming more selective for the input that activated it by adjusting its weights (or synaptic connections) in a Hebb-like [Hebb, 1949] way. Different neurons become responsive to different input patterns. The weights converge to vectors such that the distance between the patterns and their closest weight vector is minimized.

[1](The algorithm can be extended to networks with more than one set of independent WTA groups (see [Rumelhart and Zipser, 1986]))

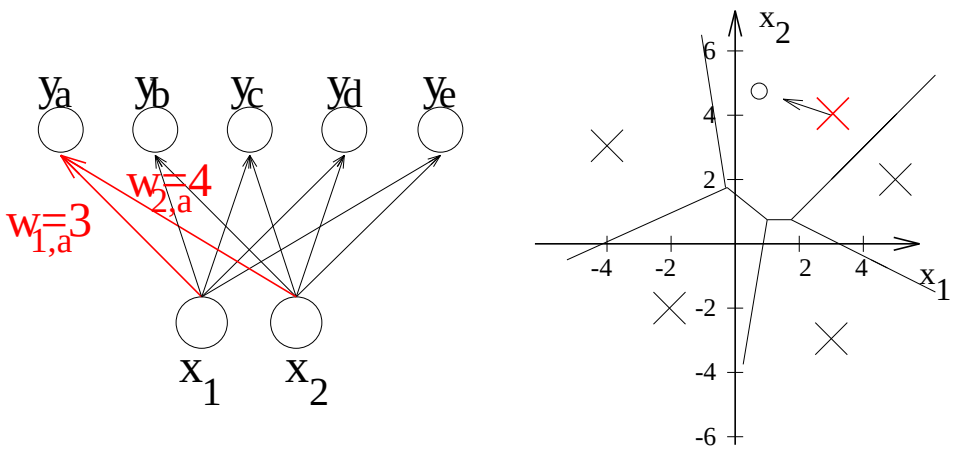

Figure 1.5: **Weight movement in competitive learning.** The circles represents the current data sample. The X's represent the weight vectors of competitive neurons. The input space is divided between the weight vectors such that data points are assigned to the closest weight vector. The closest weight vector is moved toward the input pattern.

Kohonen learning

Kohonen Feature Mapping [Kohonen, 1982] is similar to competitive learning except that the output neurons have a specified positional relationship to each other in their own output space. The topology of the output space, motivated by cortical architecture, is usually a 2-dimensional grid. The weights of the output neurons define positions in the input space as in competitive learning, but the effect of the output topology can be observed by drawing lines between the position of this neuron and its four grid neighbours. These "connections" between weight vectors define a mesh in the input space which is relevant for the weight updates. Beside the winning neuron updating its weights, neighbouring neurons in the neuron topology update their weights with a distance dependent attenuated version of the weight change. The idea is that the imaginary connections between output neurons are elastic and try to keep neighbouring neurons (in the output space) nearby in the input space. This results in nearby neurons responding to similar patterns. The weight update formula in this case is

$$\Delta w_{i,j} = \Lambda(j, j^*)\epsilon(\xi_i - w_{i,j}) \text{ for all j}$$

where $\Lambda(j, j^*)$ is a monotonic decreasing function of $||r_j - r_{j^*}||$, j^* is the index of the winning neuron and r_j gives the position of the neuron with index j in the topology within its layer.

Beside the topological benefits, this modification has the advantage of preventing dead units — units that never get close enough to the input patterns to win any. This is because at the beginning all weight vectors are updated to some degree on each update so that all the weight vectors come to fall in the input pattern space. There are also other solutions to this problem[Rumelhart and Zipser, 1986].

Competitive learning as a plausible cortical learning model

There are two major requirements in the competitive learning algorithms: the detection and activation of the neuron with the closest weight vector and the update of its weights.

While Kohonen learning and competitive learning in their current instantiation require global communication to discover the maximally activated neuron, they are computational simplifications of networks of laterally inhibited neurons in which local computation is sufficient to result in only one neuron being active. Competitive learning simulates a network of lateral inhibition between the neurons and the Kohonen learning rule simulates a lateral interconnection network where there are excitatory (positive) connections to nearby neurons in addition to inhibitory (negative) connections to farther ones. Models that have replaced the implausible rules and explicitly modeled the lateral inhibitory connections in both straight competitive [Coultrip *et al.*, 1992] and Kohonen style algorithms [Miikkulainen, 1991] have resulted in qualitatively similar results.

We saw that the calculation of the maximally active neuron could be done with the easily neurally implemented dot product if the weight vectors and input patterns are normalized. The weight updates in this case are also the neurally plausible Hebbian update rules thought to be implemented by NMDA receptors. Thus we need to hypothesize a network to normalize input activity and a method of normalizing weights. The former could again be done with a lateral inhibitory network and the latter by hypothesizing a resource limitation on the number of synapses a cell can support.

1.4.3 Discussion

As a model of human learning, the back-propagation algorithm has two major drawbacks. The first is the necessity of the desired output y^* with each input pattern. The second is the requirement of the "back-propagation" of the error measures to neurons in the hidden layers.

In comparison the competitive learning algorithm as a two-layer unsupervised algorithm trivially has neither of these disadvantages. Also the computation involved is more biologically plausible and is in fact very similar to the lateral inhibitory networks seen throughout the mammalian nervous system. Its disadvantage is that the lack of external inputs means there is no incorporation of task-specific information.

1.5 Contributions

This dissertation addresses the question of how humans learn invariants for pattern classification by developing and exploring a model of classification learning. The model is based on gross cortical anatomy and achieves powerful classification results using biologically plausible computations.

We first clarify the inherent weakness of unsupervised algorithms for classification and discuss various versions of adding supervision to the basic competitive algorithm.

We provide new insights into some of these algorithms and clearly illustrate their differences and advantages with respect to each other.

We argue that the world provides much more information at a global level than is available to any one sensory modality and that this information can be used to learn classifications within each modality. We use this to address the problem of required supervision in developing the *Minimizing-Disagreement* algorithm. The algorithm does not require the correct output signal (class label) to learn appropriate classification boundaries but still performs comparably to supervised algorithms on both a standard dataset and a new cross-modality dataset. The power is derived from modeling the sensory input from a more global level.

From a practical point of view, the algorithm is useful for self-acquisition and training of robots without human intervention. It would allow autonomous robots to learn complicated classifications through unsupervised interaction with their environment, thereby eliminating the need for costly collection of labeled data.

Prior to this work, we know of only one such unsupervised algorithm [Becker and Hinton, 1992; Becker, 1992; Becker, 1993] of this power. Becker's algorithm relies on back-propagation of error signals for optimal performance. In contrast, the algorithm proposed in this dissertation keeps the computations as simple and as biologically plausible as possible while obtaining better performance on the standard dataset tested.

1.6 Outline of Dissertation

The next chapter explores the inherent weakness of an unsupervised algorithm for classification. We explore different algorithms that incorporate supervision to produce better classifiers while still maintaining the same local characteristics and plausible methods of weight updates. First a simple and then a more complicated method of incorporating supervision to augment the competitive learning algorithm are examined. Examples are presented to demonstrate the increased classification performance.

Chapter 3 returns to the goal of developing a self-supervised classifier. The chapter starts with some motivating biological examples for the use of cross-modality structure. Then the ideas of the previous chapter are used to develop an unsupervised discriminant classifier.

The fourth chapter gives experimental results of applying the algorithm to the Peterson-Barney vowel dataset and to some data we collected consisting of images of lips and their emanating acoustic signals from people speaking consonant vowel utterances.

Chapter 5 discusses the biological plausibility of the algorithm and relates it to some recent results concerning neural plasticity and the conditions surrounding the change in synapse strength. The algorithm is also compared to that in [Becker, 1993]. The use of the algorithm as a method for learning from reduced data representations is also discussed.

Discussion and Conclusions are given in Chapter 6.

2 Adding Supervision to Competitive Learning

As our ultimate goal is to describe how multiple networks can serve to supervise each other, it is essential to understand the advantages of, and mechanisms for, adding supervision to competitive networks. This chapter gives a description of two methods of adding supervision to the basic competitive learning algorithm. The basic ideas in these algorithms have been previously developed; our aim here is to give a unifying treatment. We derive extensions, give a clear view of how they operate, and discuss their limitations and relationship to each other.

2.1 Motivation

In competitive learning, different neurons become responsive to different input patterns. If there are more patterns than output neurons within a competing group, then similar patterns (where similarity is defined as in equation 1.22) will map to the same competitive neuron. In this way competitive learning algorithms learn to group similar inputs without a supervisory signal.

This natural grouping, however, is a disadvantage in many classification tasks that require the mapping of dissimilar inputs to similar outputs. In order to achieve classification based on a task-dependent semantic similarity as opposed to similarity of the inputs, it is necessary to provide more information. This can be done with a "teaching" input that tells the network which inputs should be mapped to the same output.

The necessity of the teaching input can be observed with the task of distinguishing horizontal from vertical, single pixel wide, lines. (An example of a horizontal line of this type is shown in Figure 2.1.) The horizontal lines are more similar to the vertical lines than to all the other horizontal lines (they have more pixels in common) and thus competitive learning will group some horizontal lines with some vertical lines. Of course we could avoid all grouping by having one neuron per input pattern but we are interested in efficient encoding algorithms with fewer neurons than patterns.

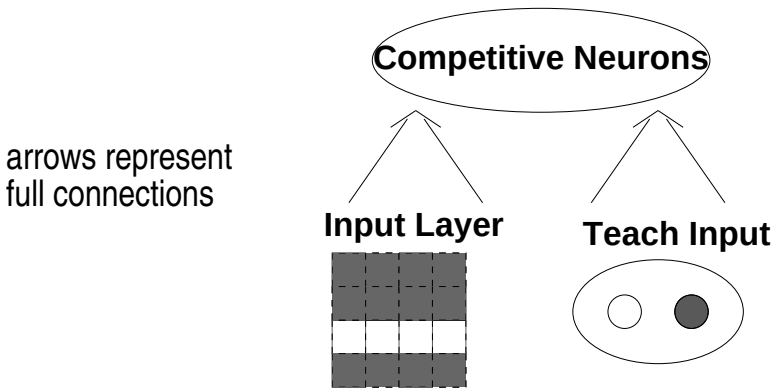

Figure 2.1: **An architecture for adding teaching inputs.**

2.2 Method 1: Supervised Competitive Learning

The first solution that was considered [Rumelhart and Zipser, 1986] is to use the labels as more input dimensions.

Rumelhart and Zipser suggested the architecture shown in Figure 2.1. The competitive neurons, beside receiving connections from the input patterns as before also receive connections from another set of teaching neurons that provide a unique pattern for each class. So, for example, when learning to separate horizontal from vertical lines beside receiving the line the neurons also receive input from two other competitive neurons[1], one representing horizontal lines and one representing vertical lines. The teaching input biases the activations of the neurons so that patterns from the same class are more likely to activate the same neurons. Once a neuron "wins" on a pattern it is more likely to win for that pattern again. In this way we can train the neurons to respond to patterns from the same class so that even with the removal of the teaching input they will respond appropriately.

With the teaching stimulus a layer of four competitive neurons is easily able to separate horizontal and vertical lines. However, to solve the classification goal of having a unique output pattern for each class (in this case one output neuron for horizontal and one for vertical), another layer of neurons is needed. The idea is to add an output layer and connect all the competing neurons from the same class to the same output neuron.

For optimal training of all the neuron weights, we would like to connect the teaching neurons to both the hidden and output neurons. However for large problems, this results in implausible connectivity from the teaching neurons. Rumelhart and Zipser [1986] engineered a solution by duplicating the hidden layer neurons. This solution is discussed in Appendix A.

We have shown [de Sa and Ballard, 1991; de Sa and Ballard, 1992] that we can achieve the same results without the need for duplication of the hidden layer neurons

[1] Rumelhart and Zipser actually used many more but two suffice.

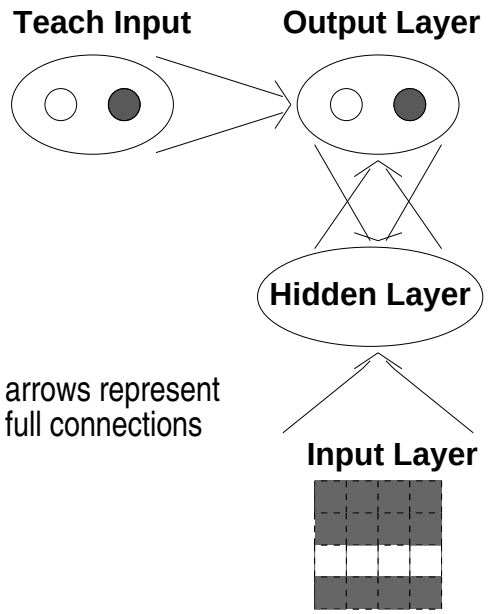

Figure 2.2: **Our supervised competitive learning architecture.**

([de Sa and Ballard, 1991] also see Appendix A) by connecting the teaching input only to the output neurons and influencing the hidden layer neurons through backward projections from the output neurons to the hidden layer.

This architecture is shown in Figure 2.2. A similar network is given in [Fanty, 1988]. It should be emphasized that, unlike the back-propagation algorithms, the projections that propagate the teach information backwards have the same structure (they are actual connections) as the forward projections. Beside reducing the total number of connections and keeping the connectivity from the teaching neurons more manageable, it leads to the more biologically plausible solutions in the next chapter.

2.2.1 Geometric Interpretation

We saw in the previous chapter that the weight vectors of competitive neurons in the hidden layer can be represented as points in the input space (where the input space is simply the vector space of dimensionality $Length(\vec{\xi})$) and that the competitive learning algorithm positions the weights in the input space to minimize the distance between the inputs and their closest weight vector over the distribution of input patterns.

Adding the label increases the dimensionality of the weight space of the competitive neurons. The algorithm still minimizes the distance between the incoming patterns and the weight vectors but this time the space is higher dimensional as the incoming patterns have been augmented with the label signal.

These extra "teaching" dimensions influence the positions of the weight vectors by pulling them to the centres of clusters of augmented inputs. Thus images that are given

similar values in the added teaching dimensions will cluster. The weight vectors will be pulled to the centres of clusters of augmented inputs allowing for similarly taught images to activate the same neuron.

During testing, when only the input dimensions are active, the results can be observed by "projecting" the weights back to the original (feedforward) input space. Thus the purpose of the backward connections from the output unit(s) (reflecting the teaching input) is to move the values of the input (forward) weights so that the appropriate classification is achieved in the absence of teaching input. These ideas can be concretely illustrated with the XOR problem.

The XOR Problem

The XOR problem being a 2-D problem nicely illustrates the effect of the teaching input. Consider a network as in Figure 2.2 with only one output unit, one teach unit and two input units. The number of hidden layer units is varied from 4 to 2 as discussed below. The input patterns are from the two sets of vectors {[-1,-1], [1,1]} and {[1,-1],[-1,1]}. The goal is to have the output neuron respond positively for input from one of the input sets and negatively for input from the other set. This output coding differs from the usual competitive code mentioned earlier but was chosen so that the weights of the hidden units would be 3-dimensional and easily visualized. The activation update and weight update rule for the output layer were changed appropriately to reflect this change.

With four competitive hidden layer neurons, the teaching input feedback from the output to the hidden layer is not required. Each unit will become responsive to one of the input patterns. This is represented in Figure 2.3 where the input patterns are represented by the small black dots and the projections of the weights onto the input space are represented by the larger, open circles. The dashed lines show the partition of the input space among the hidden layer neurons. In this case the only need for a teach unit is to allow the output unit to make the correct positive/negative connections from the hidden layer units.

The problem becomes more interesting with 3 competing hidden layer units. In this case two patterns will have to activate the same unit. In order for the problem to be solved it must be that two patterns with the same required output be grouped (i.e. [1, 1] with [-1, -1] or [1, -1] with[-1, 1]). That is, the two patterns that share a hidden neuron must be from the same class. However these patterns are more distant in the input space. Without teaching input the two patterns that activate the same unit will be patterns that are close in the input space. An example result is shown in Figure 2.4 where inputs [1, 1] and [1, -1] share a hidden-layer neuron. With this weight distribution the problem cannot be solved.

The teaching input solves this problem by making the shared neuron be allotted to two patterns from the same class. Figure 2.5 illustrates how this is accomplished. The teach input has forced the patterns apart along the third dimension. The patterns impinging on the hidden layer neurons are now from the 3-dimensional augmented input space. They are [-1, -1, K],[-1, 1, -K],[1, 1, K],[1, -1, -K] where $\pm$K is the activation of the output neuron. Closest patterns in the new 3-D space will be patterns of the

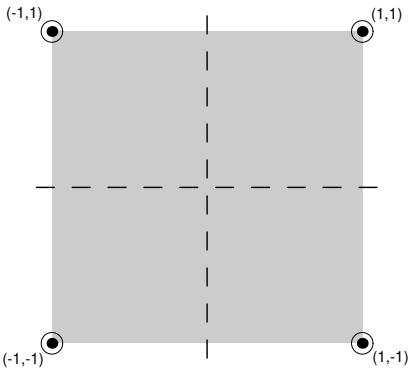

Figure 2.3: **The XOR problem with 4 hidden-layer neurons.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight vector for each input pattern so the dots and circles are superimposed. The dashed lines indicate the partition of the input space among the four neurons.

same class as long as $|K| > 1$ [2]. In this case the output weight vectors converge to positions such as those shown in the figure. Now when the teach signal is removed the hidden layer neurons receive zero activation from the output neurons and the problem is projected back to the original 2 dimensional input space. As shown in Figure 2.6, the network still classifies the patterns correctly as no unit confuses inputs with different output values.

The case of 2 hidden layer units displays a limitation of the algorithm. With 2 hidden layer units and large enough K one neuron responds to each output class. However the weights of each neuron will tend to lie midway between the patterns of its class ([0,0,K] and [0,0,-K]). (see Figure 2.7). In this case when we remove the teach input both neurons have the same weights from the input units and we cannot separate the patterns appropriately. (see Figure 2.8). This is a direct result of the competitive approach— the input is represented in terms of which output neuron is activated. As one can consider the weight vectors of two neurons as points in space where the line midway between them partitions the input space between the two neurons, only linearly separable problems can be solved with 2 competitive neurons. Likewise a layer of X competitive neurons can only separate X classes if the classes are pairwise linearly separable.

2.2.2 Another Example

As another illustrative example, we tested the algorithm on the problem of distinguishing squares from triangles. This problem is hard for 15 neurons to separate due to the considerable overlap of patterns from the two classes. There were 25 input patterns—16

[2]For convergence in a finite time it should be reasonably bigger than 1. Empirical tests showed 1.1 to be large enough.

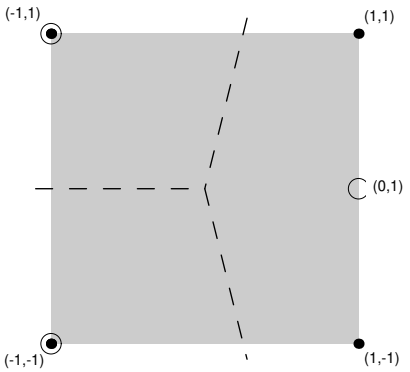

Figure 2.4: **The XOR problem with 3 hidden-layer neurons and no teach input.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight to be shared between two of the closest patterns. The other two weights are allocated to the other patterns. The dashed lines indicate the partition of the input space among the three neurons.

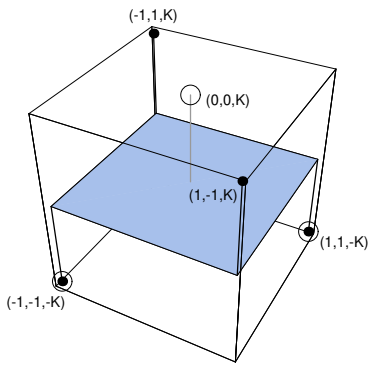

Figure 2.5: **The XOR problem with 3 hidden-layer neurons and a teach input.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The third dimension comes from the feedback from the output layer. The solution has allocated one weight to be shared between two neurons with the same feedback. The other two weights are allocated to the other patterns.

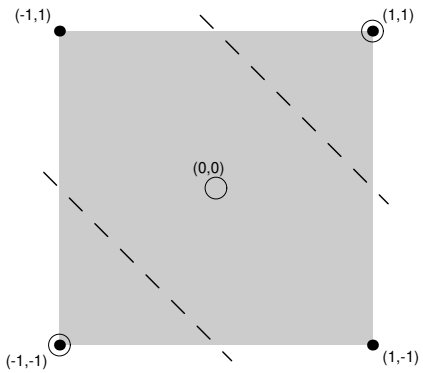

Figure 2.6: **The result of removing the teach input after training as above.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The dashed lines indicate the partition of the input space among the 3 neurons.

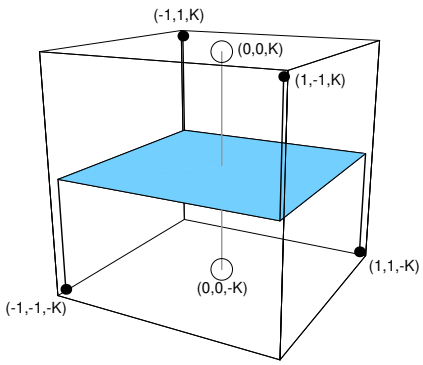

Figure 2.7: **The XOR problem with 2 hidden-layer neurons and a teach input.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. The solution has allocated one weight for each output class.

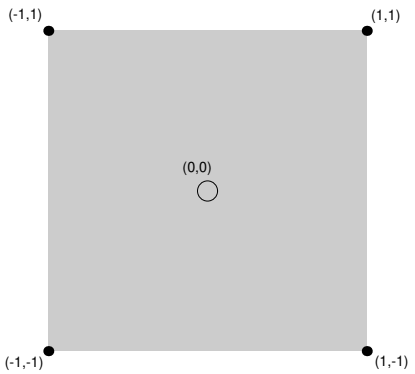

Figure 2.8: **The Result of removing the teach input after training as above.** The input patterns are represented by the small dark dots and the weights are represented by the larger open circles. When the recurrent connections are ignored the two neurons have the same weights so they are drawn superimposed.

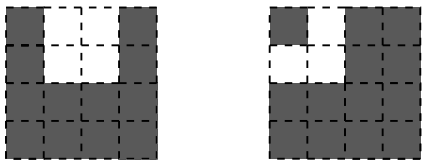

Figure 2.9: **Example input patterns in the square/triangle experiment.** These patterns were placed in all positions and in all orientations on the 4×4 grid with the exception that the right angle of the triangle occurred on one of the four inner squares. The dark squares represent input of -1 and the white input of $+1$.

triangles and 9 squares. Typical input patterns are shown in Figure 2.9. For this task, a 16 unit input layer, a 15 unit hidden layer (arranged in a 5×3 grid), 2 output units and 2 teach units were used. The activation given to the maximally active hidden-layer neuron was 1 and to the maximally active output-layer neuron (K) was 2.5 (All other hidden and output neurons had activation 0). The activation of the teaching units were set to [5,0] and [0,5] for triangles and squares respectively.

We used the Kohonen feature mapping modification to competitive learning to help visualize the influence of the teaching feedback in determining the weight vectors of the hidden layer. We cannot directly visualize the weights in the 18 dimensional space as we could with the 3D weights in the XOR problem. However the topological constraint on the weight vectors results in neighbouring neurons developing similar weights. Therefore the relative positions of the winners in the 2D hidden layer grid gives a crude idea of nearness in the original space. That is, nearby weights on the 2D grid generally correspond to weights nearer in the original high-dimensional space.

Again without the teach input and recurrent projections the neurons confuse the square and triangle inputs. An example of the weights of these neurons is shown in Figure 2.10. Squares and triangles in the appropriate positions have been drawn on top of the representation of the winning neuron for each pattern. Note that there are several neurons that won for both a square and a triangle.

The addition of teach input and recurrent connections resulted in weight vectors such as those shown in Figure 2.11. In this case the units responding maximally to squares are disjoint from those responding maximally to triangles. This allows the task to be solved (with the 15 neurons). In addition due to the topography of the Kohonen mapping we can see that for the most part the "square" neurons are separated from the "triangle" neurons. This is another indication that the teach input was able to warp the grid of weight vectors in a perceptually relevant way.

2.3 Method 2: LVQ Algorithms

In the above method, the teaching input strongly biases the activations of the neurons so that patterns from the same class are more likely to activate the same neurons (those with strong weights from the output neuron for that class). For sufficiently large magnitudes on the output units, these dimensions are dominant and we can consider the teaching input as picking which subset of neurons are able to compete. We can think of assigning a class to each neuron's weight vector and only allowing patterns from the appropriate class to influence each neuron's weight vector. (We could have implemented it not as a network with more weight dimensions but just by labeling the neurons with different classes and moving the closest weight vector whose neuron is *of the correct class*).

Thinking of the algorithm in this more general way allows for the consideration of other algorithms. This is important because the simple approach given above is far from optimal as a means for classification. The positioning of the weight vectors fails to take into account interactions between the classes. Each set of neurons, representing

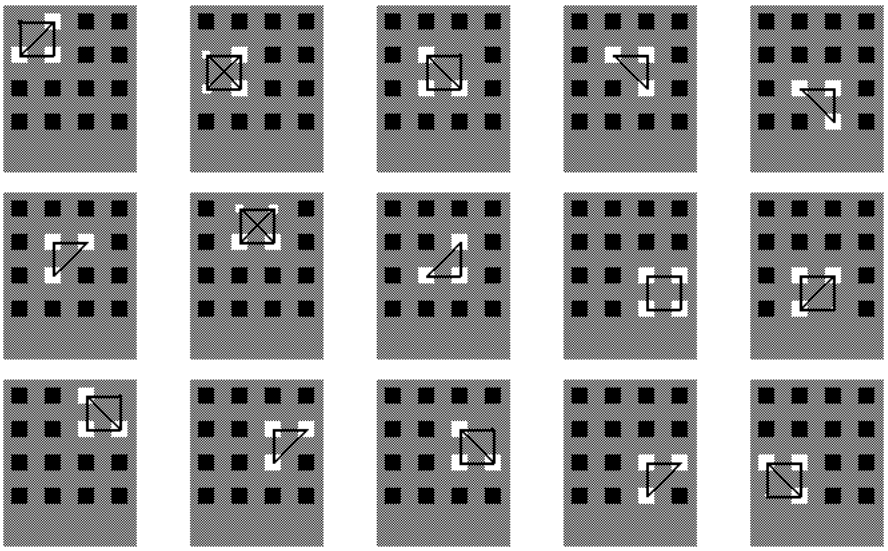

Figure 2.10: **The weights of the hidden layer neurons with no teach input.** Each large gray square represents a hidden layer neuron. The small squares represent the weights from the 4×4 input layer to these neurons. The size of these squares represents the magnitude of the connection; white/black represents positive/negative weights. Squares and triangles have been drawn in the appropriate positions on top of the representation of the winning neuron for the pattern.

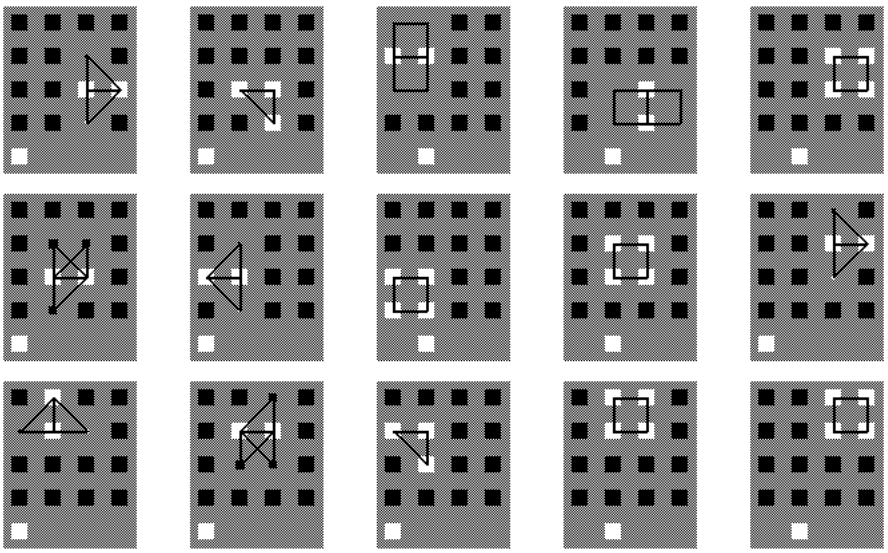

Figure 2.11: **The weights of the hidden layer neurons after the addition of teach input and recurrent connections.** Note the greater segregation of neurons responsive to squares from neurons responsive to triangles. The representation is as in the above Figure. The extra two weights represent the feedback from the two output neurons.

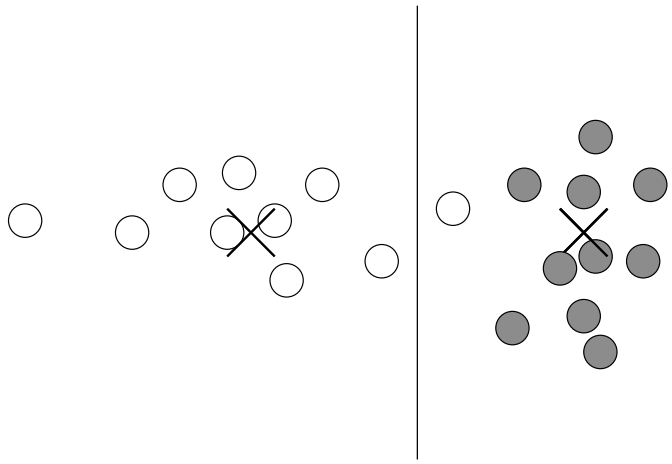

Figure 2.12: **The Problem with the Supervised Competitive Algorithm.** The algorithm has positioned the codebook vector for each class at the centroid of the class distribution. This however does not lead to optimal inducement of the classification boundary (given by the perpendicular bisector of the segment formed by joining the two codebook vector positions).

one class, simply performs the regular unsupervised algorithm on the patterns from their class. The problem arises when the positioning of the weight vectors in this way causes the boundary to be in a nonoptimal place. An example is shown in Figure 2.12. Kohonen addresses this problem in his family of Learning Vector Quantization (LVQ) algorithms.

To be consistent with the notation used by Kohonen, the weight vectors in the input space for the subsequent algorithms discussed in this dissertation will often be called *codebook vectors*. This terminology emphasizes that their purpose is to recode the input and distinguishes them from any other weights such as those from the hidden neurons to the output neurons.

The first algorithm suggested by Kohonen was the LVQ(1) algorithm[Kohonen, 1986]. In this algorithm the closest weight (codebook) vector $\vec{w}_{i^*}$ is pulled toward the pattern if it is of the same class and pushed away if it is from a different class.

$$\vec{w}_i(t+1) = \begin{cases} \vec{w}_i(t)+\epsilon(t)(\vec{\xi}(t) - \vec{w}_i(t)) & \text{if } i = i^* \text{ and } \vec{w}_i, \vec{\xi} \text{ are in the same class} \\ \vec{w}_i(t)-\epsilon(t)(\vec{\xi}(t) - \vec{w}_i(t)) & \text{if } i = i^* \text{ and } \vec{w}_i, \vec{\xi} \text{ are in different classes} \\ \vec{w}_i(t) & \text{otherwise} \end{cases}$$

Note that in this algorithm, unlike the algorithm in the last section, the current performance is affecting the weight changes that are made. The weight change for the closest weight vector is different depending on whether the codebook vector is of the correct or incorrect class. In particular, in this algorithm, the codebook vectors are being affected by the difference in the distributions. Kohonen argues that the codebook

vectors resulting from LVQ(1) tend to approximate for a particular class r,

$$P(x|C_r)P(C_r) - \sum_{s \neq r} P(x|C_s)P(C_s).$$

where $P(C_i)$ is the a priori probability of Class i and $P(x|C_i)$ is the conditional density of Class i. That is, the codebook vectors distribute themselves according to the probability distribution given by the distribution of their class minus the distributions of the other classes.

This approach is still not optimal for classification, as it too addresses the placement of vectors for density representation as opposed to optimal placement of the *borders* (induced by the vectors) for classification [3]. The LVQ2 [Kohonen *et al.*, 1988] and LVQ2.1 [Kohonen, 1990a] algorithms were designed to find optimal borders. The idea behind these algorithms is to move the codebook vectors so that the border between them moves toward the crossing point of the distributions. They do this by sampling from the distributions near the currently estimated border in order to decide which way to move the border.

The original LVQ2 algorithm only makes use of the misclassified patterns in moving the weight vectors. The update rule moves the weight vector from the correct class toward the wrongly classified pattern and the incorrect but closest weight vector away. If the class of the closest weight vector is not incorrect, the class of the second closest weight vector is not correct, or the pattern does not fall into a "window" around the border, no updates are made due to that pattern. Mathematically the update rules are given by

$$\vec{w_i}(t+1) = \vec{w_i}(t) - \epsilon(t)(\vec{\xi}(t) - \vec{w_i}(t))$$

$$\vec{w_j}(t+1) = \vec{w_j}(t) + \epsilon(t)(\vec{\xi}(t) - \vec{w_j}(t))$$

where $\vec{w_i}$ is the nearest codebook vector to x and $\vec{w_j}$ the next nearest and $\vec{\xi}$ belongs to the same class as $\vec{w_j}$ but not $\vec{w_i}$ and $\vec{\xi}$ falls in the "window" between $\vec{w_i}$ and $\vec{w_j}$. The window is defined by

$$\vec{\xi} \in \text{window between } \vec{w_i} \text{ and } \vec{w_j} \text{ if } ||\vec{\xi} - \vec{w_i}||/||\vec{\xi} - \vec{w_j}|| > (1 - \text{win}_{min})/(1 + \text{win}_{min})$$

where win_{min} is the width of the window at the narrowest point (directly between the weight vectors).

The LVQ2.1 algorithm differs from LVQ2 in that weight updates are made even if the current pattern is not misclassified as long as exactly one of the two closest weight vectors is from the correct class. Then, as long as the pattern falls within the window, the weight vector from the correct class is moved toward the pattern and the one from the other class is moved away.

[3] Kohonen [1986] showed this by showing that the use of a "weighted" Voronoi tessellation (where the relative distances of the borders from the codebook vectors was changed) worked better. However no principled way to calculate the relative weights was given and the application to real data used the unweighted tessellation.

$$\vec{w_i}(t+1) = \vec{w_i}(t) - \epsilon(t)(\vec{\xi}(t) - \vec{w_i}(t))$$

$$\vec{w_j}(t+1) = \vec{w_j}(t) + \epsilon(t)(\vec{\xi}(t) - \vec{w_j}(t))$$

where $\vec{w_i}$ and $\vec{w_j}$ are the two closest codebook vectors and $\vec{\xi}$ belongs to C_j but not C_i and $\vec{\xi}$ falls in the "window" between $\vec{w_i}$ and $\vec{w_j}$.

2.3.1 LVQ2.1 Minimizes Misclassifications

The goal in classification tasks is to minimize the numbers of misclassifications of the resultant classifier. That is we want to minimize:

$$E = \sum_j \int_{D_R_j} \sum_{i, i \neq j} P(Class_i) p(x|Class_i) dx \tag{2.1}$$

where $P(Class_i)$ is the a priori probability of $Class_i$ and $P(x|Class_i)$ is the conditional density of $Class_i$ and D_R_j is the decision region for $Class_j$ (which in this case is all $\vec{x}$ such that $||\vec{x} - \vec{w}_k|| < ||\vec{x} - \vec{w}_i||$ (for all i) and $\vec{w}_k$ is a codebook vector for $Class_j$). In this section we show that in the 1-dimensional case a close variant of the LVQ2.1 algorithm minimizes the number of misclassifications.

Consider a One-Dimensional classification problem of two classes and two codebook vectors $w1$ and $w2$ defining a class boundary $b = (w1 + w2)/2$ as shown in Figure 2.13. In this case Equation 2.1 reduces to:

$$E(b) = \int_{-\infty}^{b} P(C_B) p(x|C_B) dx + \int_{b}^{\infty} P(C_A) p(x|C_A) dx\,. \tag{2.2}$$

The derivative of Equation 2.2 with respect to b is

$$dE/db = h(b)$$

where

$$h(x) = P(C_B)P(x|C_B) - P(C_A)P(x|C_A)$$

The window training procedure of [Wassel and Sklansky, 1972] uses a stochastic approximation technique to find a local minimum of E and gives the following iterative calculation for b (see Appendix B for the ideas behind their algorithm)

$$b(n+1) = b(n) + \epsilon(n) Z_n(X_n, Class(n), b(n), c(n)) \tag{2.3}$$

where

$$Z_n = \begin{cases} 1 & \text{for } X_n \in C_A, |X_n - b(n)| < c(n) \\ -1 & \text{for } X_n \in C_B, |X_n - b(n)| < c(n) \\ 0 & \text{otherwise} \end{cases}$$

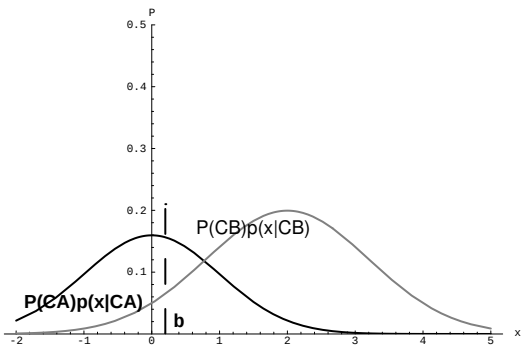

Figure 2.13: **A one-dimensional classification problem of two classes.** The dark curve represents the distribution of the Class A patterns and the lighter curve that of the Class B patterns. The goal is to position the border b to minimize the number of misclassified patterns.

and $c(n)$ defines a "window" in which the probabilities are being sampled, X_n is the sampled data point at time n and $b(n)$ is the border position at time n.

The previous discussion related only to moving one border in a 1-dimensional problem. We now extend the ideas to derive rules for moving codebook vectors around in multi-dimensional space. The first step is to consider the border as the midpoint between two codebook vectors $b = (w1 + w2)/2$ and thus

$$\Delta b = (\Delta w1 + \Delta w2)/2$$

In other words we can induce a border movement by moving both codebook vectors so that their sum movement is twice the desired movement of the border. One simple solution is to move both codebook vectors in the same way as the desired border motion given in Equation 2.3. The algorithm can then be reworded to state:

> If the pattern falls within the window around the border, move the codebook vector of the same class toward the input pattern and the codebook vector of the other class away.

This reformulation has the added benefit that it no longer requires that the class labeled A be toward the left on the x-axis. We can move the codebook vectors according to (instead of equation 2.3).

$$w_i(n+1) = w_i(n) + \epsilon(n)\frac{(X_n - w_i(n-1))}{|X_n - w_i(n-1)|} \tag{2.4}$$

$$w_j(n+1) = w_j(n) - \epsilon(n)\frac{(X_n - w_j(n-1))}{|X_n - w_j(n-1)|} \tag{2.5}$$

(where w_i is the codebook vector of the same class and w_j that for the other class) if X_n lies in a window of width $2c(n)$ centred at $b(n)$, otherwise the weight vectors are not moved.

$$w_i(n+1) = w_i(n), \qquad w_j(n+1) = w_j(n)$$

Expanding the problem to more dimensions, and more classes with more codebook vectors per class, complicates the analysis as a change in two codebook vectors to better adjust their border affects more than just the border between the two codebook vectors. However, ignoring these effects and abstracting from the 1-Dimensional equations (2.5) and (2.4), a first order approximation gives the following update procedure:

$$\vec{w_i}^*(n) = \vec{w_i}^*(n-1) + \epsilon(n)\frac{(X_n - \vec{w_i}^*(n-1))}{||X_n - \vec{w_i}^*(n-1)||}$$

$$\vec{w_j}^*(n) = \vec{w_j}^*(n-1) - \epsilon(n)\frac{(X_n - \vec{w_j}^*(n-1))}{||X_n - \vec{w_j}^*(n-1)||}$$

where X_n is from $Class_i$, and $\vec{w_i}^*$, $\vec{w_j}^*$ are the two nearest codebook vectors, one each from $Class_i$ and $Class_j$ ($j \neq i$) and X_n lies within c(n) of the border between them. (No changes are made if all the above conditions are not true). As above this algorithm assumes that the initial positions of the codebook vectors are such that they will not have to cross during the algorithm.

The above algorithm is identical to Kohonen's LVQ2.1 algorithm (which is performed after appropriate initialization of the codebook vectors) except for the normalization of the step size, the decreasing size of the window width c(n) and constraints on the learning rate ϵ. (In [de Sa and Ballard, 1993], we show empirically that these additions can improve performance slightly.) Thus the LVQ2.1 algorithm can be seen as an approximation of an algorithm that is optimal in the 1-Dimensional case.

The LVQ2 and LVQ2.1 algorithms can be understood by observing the effect of the pattern distributions on the border movement. An example is shown in Figure 2.14. In LVQ2.1 *all* the patterns that fall within the window from Class A push the boundary to the right (increasing the area for Class A) and all the patterns within the window from Class B push the boundary to the left. The net force to the right is thus the excess of patterns from Class A within the window (which will be negative if there are an excess of Class B patterns within the window). The forces will be equal when the probability of patterns from both classes within the window is equal. In order for the border obtained by balancing the forces to correspond with the optimal border (at the crossing point of the distributions) the curvature of both curves (in terms of convexity/concavity) through the window should be similar. For sufficiently small windows the crossing point will correspond with the border obtained by balancing the forces.

The diagram also shows why the LVQ2.1 algorithm is better than LVQ2 for overlapping distributions[4]. In LVQ2 all the patterns that fall within the window and are

[4]Kohonen's original motivation was to counteract the problems brought about by not normalizing, and also that twice as many corrections should increase the statistical accuracy [Kohonen, 1990a]. Knoll and Lo in [Knoll and Lo, 1992] also developed a similar algorithm with the motivation of enabling correctly classified patterns to force out incorrect patterns.

classified incorrectly pull the border toward them. So all the Class B patterns falling to the left of the boundary (but within the window) push the boundary to the left and all the Class A patterns within the window to the right of the boundary push the boundary right. Over time the boundary will tend to the place where the dark stippled area equals the light stippled area. We saw above that if the functions have similar curvature in the window, the LVQ2.1 algorithm places the border optimally. In order for the LVQ2 balance to lead to the optimal border, it is necessary that the slopes of the distributions on the wrong side of the border be nearly equal within the window.

Several groups have found that the LVQ2 style rule works better for artificial, separable problems [Geva and Sitte, 1991; Knoll and Lo, 1992] where the probability distributions for different classes do not overlap. The reason for this is illustrated in Figure 2.15. As the distributions change abruptly the equal curvature assumption is not generally valid, but as there is a border for which there are no misclassifications, the LVQ2 style rule is appropriate.

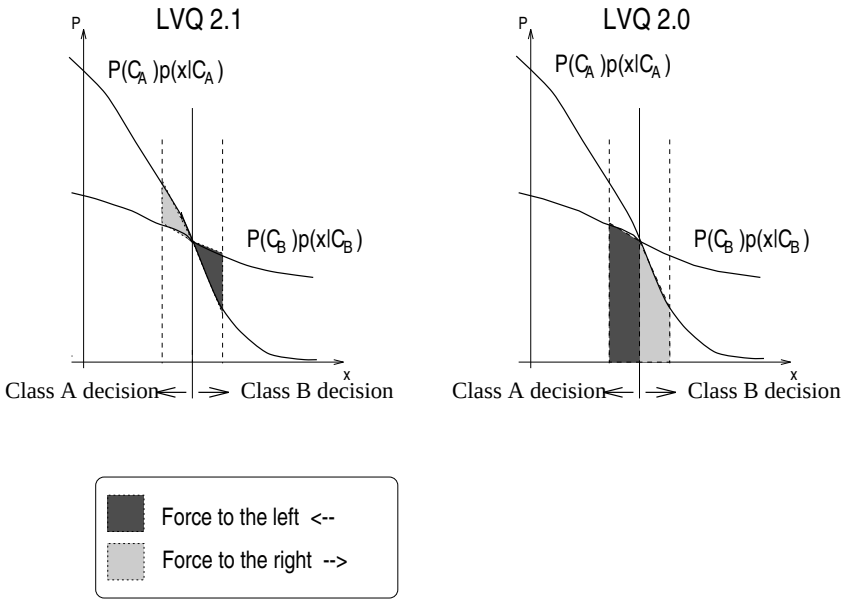

Figure 2.14: **The driving forces in the LVQ2 and LVQ2.1 algorithms for overlapping distributions.** The forces are shown for the optimal border. Note that in the LVQ2.1 case (but not LVQ2) the forces balance at the optimal border. In LVQ2 the border would tend to move toward the left.

2.4 Summary

To summarize, the standard unsupervised algorithm classifies only based on input similarity. It is oblivious to the particular labels associated with each different task. For the three pattern distributions given at the top of Figure 2.16, each classification task

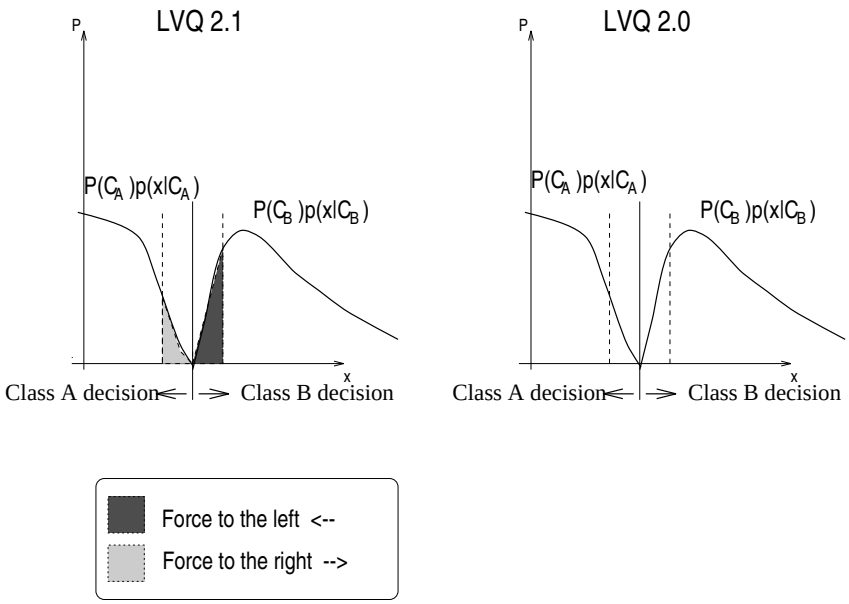

Figure 2.15: **The driving forces in the LVQ2 and LVQ2.1 algorithms for separable distributions.** The forces are shown for an optimal border. Note that here the forces at the optimal border do not balance in the LVQ2.1 case (the border would tend to move to the left). In LVQ2, however, there are no misclassified patterns and hence no unbalanced force at the optimal border.

has the same input patterns (just with different labels) and so it classifies them all the same way. This only results in a correct separation for the first problem.

By adding supervision to the standard competitive learning algorithm we can provide it with task-specific information to allow it to classify the problems in Figure 2.16 differently. For appropriately separated classes, the simple algorithm of competitive learning in the augmented input space (with a dimension for each class) can provide appropriate boundaries. The algorithm is still limited, however, as it is not designed for optimal placement of the boundaries.

For optimal classification with arbitrary class separation (and overlapping classes) the LVQ2.1 algorithm is best as it actually takes into account the current performance in order to improve it.

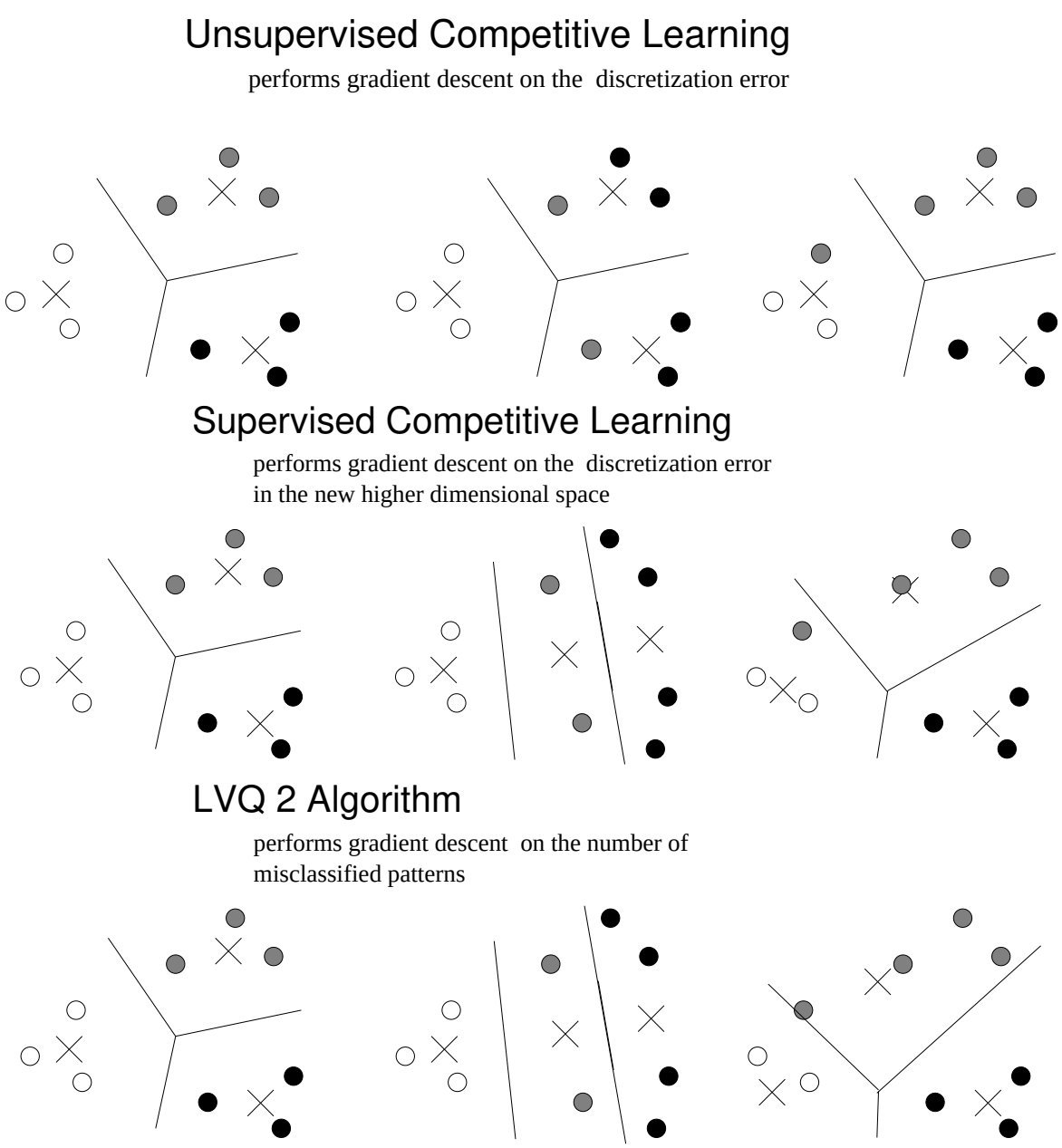

Figure 2.16: **Performance of the different algorithms on three different classification problems.** The three different coloured circles represent three different classes of patterns. Only the LVQ2 algorithm is able to separate the three classes appropriately. See the text for more description.

3 Cross-Modal Information

The previous chapter showed that information about misclassifications can greatly increase the classification abilities of a piecewise-linear classifier. The power of the supervised approach lies in the fact that it directly minimizes its final error measure (on the training set). The positions of the codebook vectors are placed not to approximate the probability distributions but to decrease the number of misclassifications.

Unfortunately, when modeling human category learning, one must deal with the fact that there is no omniscient homunculus (or pre-trained network) correctly labeling all the incoming sensory patterns. The algorithm employed by the brain does not have the correct answer always available, can not measure its classification errors while learning and hence can not directly minimize them.

In this chapter we derive an algorithm that uses the structure between signals from two or more modalities to assist in the development of a piecewise-linear classifier within each modality. It takes advantage of the structure available in natural environments that results in sensations to different sensory modalities (and sub-modalities) that are correlated. For example, hearing "mooing" and seeing cows tend to occur together. So, although the sight of a cow does not come with an internal homuncular "cow" label it does co-occur with an instance of a "moo." The key is to process the "moo" sound to obtain a self-supervised label for the network processing the visual image of the cow and vice-versa. This idea is schematized in Figure 3.1. In lieu of correct labels each network receives the output of the other network. Note that this is a fundamentally different kind of information than the external labels of supervised algorithms. Before the networks develop into good classifiers, they are not providing good labels to each other. We develop an algorithm that allows the networks to develop into better classifiers together.

Most of the chapter deals with the mathematical theory behind the algorithm, but to start we give some of the biological motivation behind the idea of using the cross-modality structure.

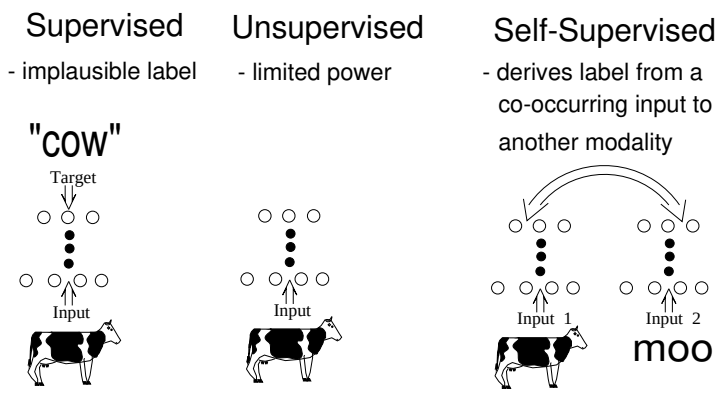

Figure 3.1: **The idea behind the self-supervised algorithm.**

3.1 Motivating Biology

3.1.1 Psychophysics

The coincidence between auditory and visual events is pervasive throughout our experience. From just after birth, babies are able to turn crudely toward sounds. Over time we develop a rather precise auditory-visual spatial map. This enables us to be able to pick up correlations between auditory and visual sensations. For example, with sound localization we are able to notice that mooing comes from cows.

Another example of information picked up in this way is the ability to read lips. Anyone who has conversed in a noisy environment, is aware of the improved speech recognition performance achieved when the speakers face (particularly the lips) is visible. (This has also been demonstrated in more controlled experimental conditions [Sumby and Pollack, 1954].) This is an example of cross-modal integration. The visual signal from the motion of the lips, jaw, and tongue are helping the auditory system to understand the speech.

The effect of lip movement on speech recognition is even more prominent when the stimuli are experimentally manipulated so that the visual and acoustic signals are discordant. In the experiments of [McGurk and MacDonald, 1976; MacDonald and McGurk, 1978] subjects are presented with acoustic stimuli of various consonant-vowel pairs and simultaneously shown images of faces speaking a different consonant with the vowel. Thus for example when presented acoustically with a /ba/ syllable and visually with a face speaking /ga/, 98% of adult subjects hear /da/ [McGurk and MacDonald, 1976]. The result is very striking and not subject to conscious control. It shows that visual and auditory stimuli are able to interact to produce a unified percept, different from the stimuli actually given to either modality. Furthermore it seems that the ability of visual signals to influence acoustic classification is at least partially learned. Pre-

school and school children show significantly less of the effect than do adults [McGurk and MacDonald, 1976].

This ability to recognize relationships between lip movements and emitted sound starts developing early. By four and a half months of age infants are able to recognize that particular lip motions go with particular sounds. Kuhl and Meltzoff showed that infants looked significantly longer [Kuhl and Meltzoff, 1984] at the matching face when presented with the sound /a/ or /i/. Their preference was specific to the actual speech information as they did not show this effect when the speech signals were replaced with tones that followed the duration, amplitude envelope and onset/offset timing of the original speech sounds [Kuhl and Meltzoff, 1984].

Similarly, an example of auditory events influencing visual perception is demonstrated in a novel cross-modal experiment in [Durgin, 1995]. The experiment involved repeated brief presentations of random dot patterns in two rectangular areas of a screen. On each presentation, one of the two areas received 25 dots/deg^2 and the other 2 dots/deg^2. The visual presentations were paired with auditory tone stimuli such that the pitch of the tone was perfectly correlated with the side of the denser dot pattern. After 180 flashed presentations, a staircase procedure was used to determine the perceived density equivalence (for test patterns with dot densities between the two trained densities) between the two areas when presented with each of the two tones. The experiment showed that there was a significant effect of the tone on the perceived density relationship between the patterns in the two areas. The simultaneous presentation of the tone associated with a denser texture in one area during training, lead to an impression of greater dot density in that area during testing. To match a constant density, the difference between the density required in the presence of the high pitch and that with a low pitch was 10% [Durgin, 1995].

3.1.2 Neurobiology

The previous section examined results showing that information from different sensory modalities is combined in determining our perception. Often, as for the McGurk effect mentioned above, the combination is not subject to conscious control. It is as if the results are not simply being combined at a high-level output stage but are able to influence each other in the individual processing stages. This is corroborated by neurophysiological studies which have found responses of cortical cells in primary sensory areas that respond to features from other sensory modalities. For example [Spinelli *et al.*, 1968] found sound frequency specificity in cells in primary *visual* cortex of the cat and [Fishman and Michael, 1973] found that these bimodal cells tend to be clustered together. As support for the unified percept observed in psychophysical studies, the stimuli are able to affect the same cell. In fact acoustic responses in a single cell could be inhibited by inhibitory visual stimuli [Morrell, 1972].

[Sams *et al.*, 1991] have also shown effects of visual input on auditory processing in humans. Using magnetoencephalographic (MEG) recordings, they showed that although a visual signal by itself did not result in a response over the auditory cortical area, different visual signals changed the response to the auditory signal. They again used the

McGurk effect stimuli. Subjects were trained with a higher percentage of either agreeing or disagreeing stimuli. Significantly different neuromagnetic measurements were made to the frequent and infrequent stimuli. As no similar difference occurred when two different light stimuli occurred with the sounds, they argue that this shows that the visual information from the face is reaching the auditory cortex.

The neurophysiological and psychophysical evidence must be reconciled with the fact that anatomically the information from the different sensory modalities goes to spatially separate, segregated cortical areas. Retinal input goes to occipital cortex whereas auditory input goes to auditory cortex in the temporal lobe. Even within the auditory and visual cortex there are many different areas which seem to be specialized to processing different parts of the signal. For instance colour processing seems to be mostly separate from motion processing [Merigan and Maunsell, 1993; Desimone and Ungerleider, 1989]. There is a significant restriction on the amount of cross-modality interaction that can occur. This is thought to be due to restrictions on connectivity; it is not physically possible to have all neurons connected to all other neurons (or even any significant fraction). Therefore input from each modality and submodality must be processed separately with little cross-talk.

As there are no direct afferent (feed-forward) connections from one input modality to another, the information from other modalities could either be coming bottom-up from shared subcortical structures such as the superior colliculus or alternatively top-down from the multi-sensory integration areas such as entorhinal cortex and other limbic polymodal areas. This idea is suggested in [Rolls, 1989] and seems to be supported by the evidence from visual cortex. As stated by [Spinelli *et al.*, 1968]

> non-visual stimuli affect the activity of ganglion cells only minutely [Spinelli *et al.*, 1965; Spinelli and Weingarten, 1966; Spinelli, 1967]; they affect that of the geniculate cells to a greater extent [Meulders *et al.*, 1965] and very markedly affect cortical cells [Murata *et al.*, 1965]. Even more interaction appears to be present in prestriate cortex [Buser and Borenstein, 1959].

Thus we know from psychophysical studies that information from different modalities is combined and that information from one modality can assist or interfere with classification in another. The physiological evidence supports this finding in showing that input to other modalities can influence processing in another sensory pathway. This combined with the anatomical evidence that shows no direct input from one modality's transducers to another pathway, suggests that perhaps this information is coming top-down through feedback pathways from multi-sensory areas. Furthermore we suggest that this integration may not just affect the properties of developed systems but play an important role in the learning process itself. Just as lip-reading is a learned classification ability, correlations between inputs to different sensory modalities may affect other classification learning in the individual modalities. In this chapter we will investigate the power that this kind of integration might provide to learning classifiers in the individual modalities.

3.2 Using Cross-Modality Information for Self-Supervised Learning

Motivated by the data presented in the last section, we hypothesize that one way in which animals may compensate for a lack of labeling signals is through using information in other modalities to assist in learning to classify within another modality.

Following the anatomical evidence presented earlier and acting under the assumption that it is infeasible to have neurons receiving input from the sensory transducers of all the senses, we propose that useful information between modalities is communicated higher up. We propose an architecture such as that schematized in Figure 3.1 in which each modality has its own processing stream (or classification network) but access to each other's output at a high level. This information can reach the lower levels within each processing stream through feedback within the stream.

In deriving an algorithm to work with this architecture there are two issues to consider. The first is learning the labels of the codebook vectors and the second is positioning them in the input space. This can be made more concrete by considering the network shown in Figure 3.2. In this network the weights to the hidden layer competitive neurons represent the codebook vectors and the labels associated with them are given by the weights from the hidden to output neurons. The label of a codebook vector is given by the output neuron to which it has the strongest weight.

3.2.1 Cross-Modal Information for Determining Weight Vector Labels

One way to make use of the cross-modality structure is to use it for deriving labels for the codebook vectors (after they have been positioned either by random initialization or an unsupervised algorithm). The labels can be learned with a competitive learning algorithm using a network such as that shown in Figure 3.2. Presentation of the paired patterns results in activation of the closest codebook vectors in each modality (and 0's elsewhere). Co-occurring codebook vectors will then increase their weights to the same competitive output neuron. After several iterations the codebook vectors are given the (arbitrary) label of the output neuron to which they have the strongest weight. We will refer to this as the "labeling algorithm."

3.2.2 Cross-Modal Information for Better Weight Vector Placement

A more powerful use of the extra information is for better placement of the codebook vectors themselves. In this section we derive an algorithm that minimizes the disagreement between the outputs of two modalities. The derivation is very similar to the discussion in Section 2.3.1. As in that section the algorithm is first derived not as a piecewise-linear classifier but as a method for moving boundaries for the case of two classes and an agent with two 1-Dimensional sensing modalities (see Figure 3.3.).

Each class has a particular probability distribution for the sensation received by each modality. If modality 1 experiences a sensation from its pattern A distribution,

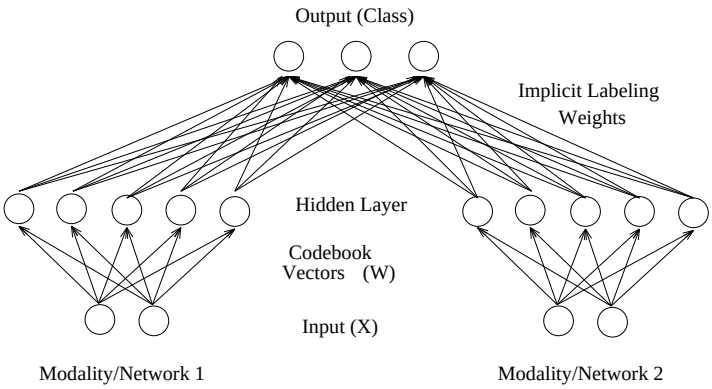

Figure 3.2: **Network for learning the labels of the codebook vectors.** The weight vectors of the hidden layer neurons represent the codebook vectors while the weight vectors of the connections from the hidden layer neurons to the output neurons represent the output class that each codebook vector currently represents. In this example there are 3 output classes and two modalities each of which has 2-D input patterns and 5 codebook vectors.

.

modality 2 experiences a sensation from its own pattern A distribution. That is, the world presents patterns from the 2-D joint distribution shown in Figure 3.6a) but each modality can only sample its 1-D marginal distribution (shown in Figure 3.3 and Figure 3.6a)).

In Section 2.3.1, we showed that for the one-dimensional problem of two classes in Figure 3.3 it is possible to iteratively learn the optimal class boundaries for distinguishing Class A from Class B when the labels of the sample patterns are given. Letting the current boundary for modality j be represented with b_j (see Figure 3.3), the supervised goal was to minimize the probability of misclassified patterns or each modality. That is, to minimize

$$E = \int_{b_j}^{\infty} P(C_A)p(x_j|C_A)dx_j + \int_{-\infty}^{b_j} P(C_B)p(x_j|C_B)dx_j \tag{3.1}$$

for both modalities ($j = 1, 2$) where $P(C_i)$ is the a priori probability of Class i and $p(x_j|C_i)$ is the conditional density of Class i for modality j.

Minimizing Disagreement

This formulation is explicitly supervised in that the estimation of the conditional probabilities depends explicitly on class information. In order to estimate the term $P(C_A)p(x|C_A)$ it is necessary to know which patterns are from Class A (and which are from Class B). An unsupervised error function must depend only on the whole pattern distribution $P(C_A)p(x|C_A) + P(C_B)p(x|C_B)$. We propose the *Disagreement Error*— the number of patterns classified differently by the two networks. The Disagreement Error can be written mathematically as:

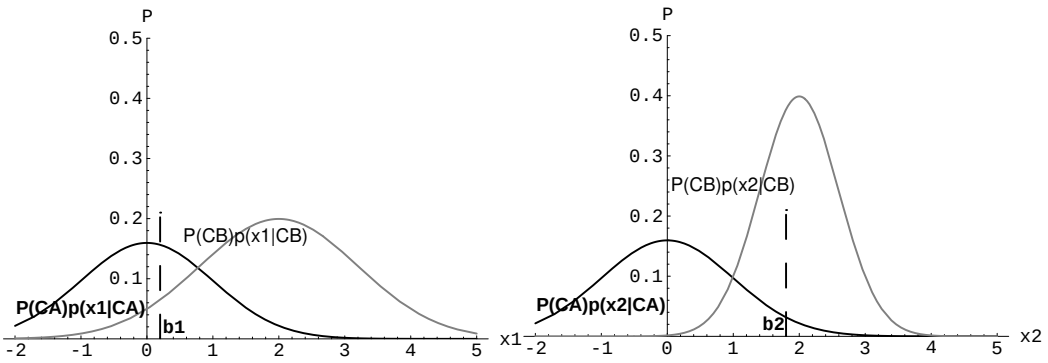

Figure 3.3: **An example world as sensed by two different modalities.** If modality 1 receives a pattern from its Class A distribution, modality 2 receives a pattern from its own class A distribution (and the same for Class B). Without receiving information about which class the patterns came from, they must try to determine appropriate placement of the boundaries b_1 and b_2. $P(C_i)$ is the prior probability of Class i and $p(x_j|C_i)$ is the conditional density of Class i for modality j

$$E(b_1, b_2) = Pr\{x_1 < b_1 \ \& \ x_2 > b_1\} + Pr\{x_1 > b_1 \ \& \ x_2 < b_2\} \tag{3.2}$$

That is,

$$\begin{aligned} E(\ \ b_1, b_2) = & \qquad (3.3) \\ & P(C_A) \quad \left[\int_{-\infty}^{b_1} p(x_1|C_A)dx_1 \int_{b_2}^{\infty} p(x_2|C_A)dx_2 + \int_{b_1}^{\infty} p(x_1|C_A)dx_1 \int_{-\infty}^{b_2} p(x_2|C_A)dx_2 \right] \\ + \quad & P(C_B) \quad \left[\int_{-\infty}^{b_1} p(x_1|C_B)dx_1 \int_{b_2}^{\infty} p(x_2|C_B)dx_2 + \int_{b_1}^{\infty} p(x_1|C_B)dx_1 \int_{-\infty}^{b_2} p(x_2|C_B)dx_2 \right] \end{aligned}$$

Note that (3.4) does not depend on the class information as can be seen by rewriting it as

$$E(b_1, b_2) = \int_{-\infty}^{b_1} \int_{b_2}^{\infty} f(x_1, x_2)dx_1dx_2 + \int_{b_1}^{\infty} \int_{-\infty}^{b_2} f(x_1, x_2)dx_1dx_2 \tag{3.4}$$

where

$$f(x_1, x_2) = P(C_A)p(x_1|C_A)p(x_2|C_A) + P(C_B)p(x_1|C_B)p(x_2|C_B) \tag{3.5}$$

As $f(x_1, x_2)$ is the joint density of all inputs (over both classes), it can be sampled without requiring labeled patterns.

A graphical example of the Energy function as a function of b_1 and b_2 for particular distributions is shown in Figure 3.5. The figure actually shows the negative of the Energy function for easier viewing. The small bump in the middle is the local minimum

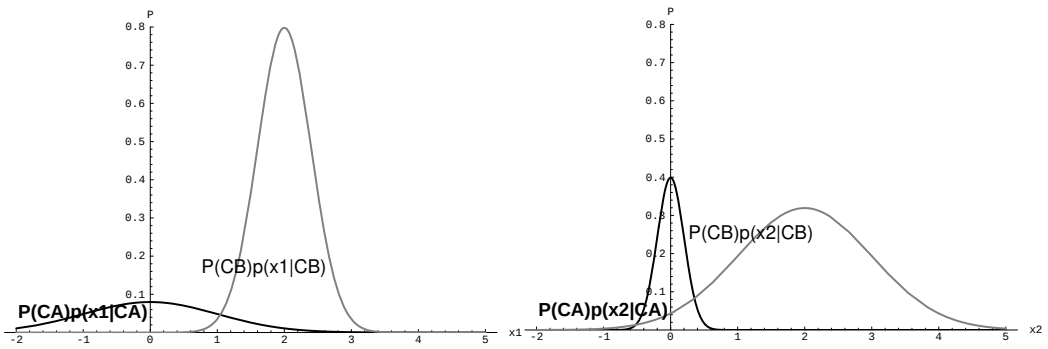

Figure 3.4: **Probability distributions used in the calculation of the energy function plots shown in Figures 3.5, 5.2, 5.3, 5.4, 5.5 and 5.5.**

in Disagreement Error that the algorithm is designed to seek. This is one neural network algorithm in which we want to find a local minimum, instead of the global minimum. (In our case the global minima correspond to both networks setting their borders at very large values and always outputting $ClassA$, or both setting them at very large negative numbers and always outputting $ClassB$.) Such a local minimum exists when there exists a pair (b_1^*, b_2^*) such that $\int_{-\infty}^{b_2^*} f(b_1^*, x_2)dx_2 = \int_{b_2^*}^{\infty} f(b_1^*, x_2)dx_2$ and $\int_{-\infty}^{b_1^*} f(x_1, b_2^*)dx_1 = \int_{b_1^*}^{\infty} f(x_1, b_2^*)dx_1$ and $\partial^2 E(b_1^*, b_2^*)/\partial b_1^2 \times \partial^2 E(b_1^*, b_2^*)/\partial b_2^2 - (\partial^2 E(b_1^*, b_2^*)/\partial b_1 \partial b_2)^2 > 0$ and $\partial^2 E(b_1^*, b_2^*)/\partial b_1^2 > 0$. A local minimum is not guaranteed to exist even for distributions for which the optimal boundary could be found with a supervised algorithm. This issue will be addressed more in Chapter 5.

The derivative of (3.4) with respect to b_1 is

$$\partial E/\partial b_1 = \int_{b_2}^{\infty} f(b_1, x_2)dx_2 - \int_{-\infty}^{b_2} f(b_1, x_2)dx_2 \tag{3.6}$$

and similarly with respect to b_2

$$\partial E/\partial b_2 = \int_{b_1}^{\infty} f(x_1, b_2)dx_1 - \int_{-\infty}^{b_1} f(x_1, b_2)dx_1 \tag{3.7}$$

Using an extension by Wassel and Sklansky [1972], to the stochastic approximation method [Robbins and Monro, 1951] gives the following iterative calculation for b_1 (see Appendix C for the derivation) where the hypothesized output class from the other modality network replaces the correct label.

$$b_1(n+1) = b_1(n) + \epsilon(n) Z_n(X_1(n), X_2(n), b_1(n), b_2(n), c(n)) \tag{3.8}$$

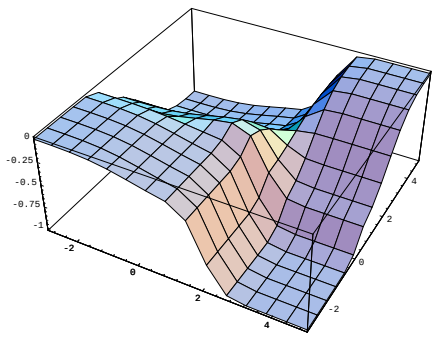

Figure 3.5: **The Disagreement energy function as a function of b_1 and b_2 for the particular distributions shown in Figure 3.4.** The function is actually the negative of the Disagreement energy function for easier viewing.

with

$$Z_n = \begin{cases} 1 & \text{for } X_2(n) < b_2(n), |X_1(n) - b_1(n)| < c(n) \\ -1 & \text{for } X_2(n) > b_2(n), |X_1(n) - b_1(n)| < c(n) \\ 0 & \text{otherwise} \end{cases}$$

The rule for the movement of b_2 is analogous.

Intuitively we can understand that this update rule minimizes the disagreement as follows. If Network 2 outputs Class A ($X_2(n) < b_2(n)$) then Network 1 increases its future probability of saying Class A by moving its border b_1 to the right ($b_1(n+1) = b_1(n)+\epsilon(n)$). If Network 2 outputs Class B ($X_2(n) > b_2(n)$) then Network 1 decreases its chance of saying Class A by moving its border to the left ($b_1(n+1) = b_1(n)-\epsilon(n)$).

3.2.3 Minimizing Disagreement as an Approximation to Minimizing Misclassifications

The previous section developed an algorithm to minimize the disagreement between two networks receiving patterns arising from the same class (but sampled along different dimensions). In this section we show how minimizing the disagreement is related to the classification goal of minimizing the number of misclassified patterns.

Note that the algorithm to minimize the disagreement corresponds to the supervised algorithm in Section 2.3.1 except that the "label" for each modality's pattern is the hypothesized output of the other modality. To understand how making use of this label,

through minimizing the disagreement between the two outputs, relates to the true goal of minimizing misclassifications in each modality, consider the example illustrated in Figure 3.6. As previously mentioned, in the supervised case (Figure 3.6a)) the availability of the actual labels allows sampling of the actual marginal distributions. For each modality, the number of misclassifications can be minimized by setting the boundaries for each modality at the crossing points of their marginal distributions.

However in the self-supervised system, the labels are not available. Instead we are given the output of the other modality. Consider the system from the point of view of modality 2. Its patterns are labeled according to the outputs of modality 1. This labels the patterns in Class A as shown in Figure 3.6b). Thus from the actual Class A patterns, the second modality sees the "labeled" distributions shown. Letting a be the fraction of misclassified patterns from Class A, the resulting distributions of the real Class A patterns seen by modality 2 are $(1-a)P(C_A)p(x_2|C_A)$ and $(a)P(C_A)p(x_2|C_A)$.

Similarly Figure 3.6c) shows the effect on the patterns from class B. Letting b be the fraction of Class B patterns misclassified, the distributions are given by $(1-b)P(C_B)p(x_2|C_B)$ and $(b)P(C_B)p(x_2|C_B)$. Combining the effects on both classes results in the "labeled" distributions shown in Figure 3.6d). The "apparent Class A" distribution is given by $(1-a)P(C_A)p(x_2|C_A)+(b)P(C_B)p(x_2|C_B)$ and the "apparent Class B" distribution by $(a)P(C_A)p(x_2|C_A)+(1-b)P(C_B)p(x_2|C_B)$. The crossing point of these two distributions occurs at the value of x_2 for which $(1-2a)P(C_A)p(x_2|C_A) = (1-2b)P(C_B)p(x_2|C_B)$. Comparing this with the crossing point of the actual distributions that occurs at x_2 satisfying $P(C_A)p(x_2|C_A) = P(C_B)p(x_2|C_B)$ reveals that if the proportion of misclassified patterns from $ClassA$ is the same as the proportion of misclassified patterns from Class B (i.e. $a = b$) the crossing points of the distributions will be identical. This is true even though the approximated distributions will be discrepant for all cases where there are any misclassified patterns ($a > 0$ OR $b > 0$). If $a \approx b$, the crossing point will be close.

Simultaneously the second modality is labeling the patterns to the first modality. At each iteration of the algorithm both borders move according to the samples from the "apparent" marginal distributions.

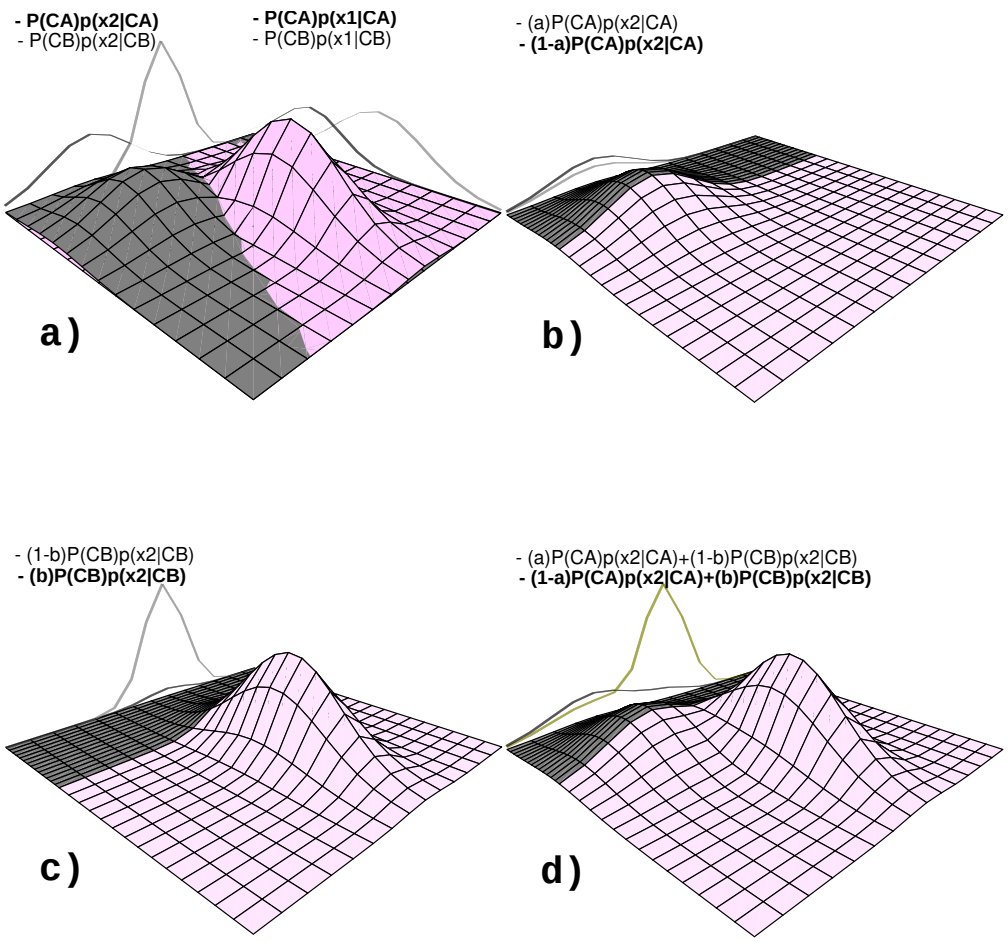

Figure 3.6: **An example of the joint and marginal distributions for the example problem introduced in Figure 3.3.** (For better visualization the scale of the joint distribution is twice that of the marginal distributions.) The darker gray represents patterns labeled "A", while the lighter gray are labeled "B". The dark and light curves are the corresponding marginal distributions with bold and regular labels respectively. a) shows the labeling for the supervised case. b),c) and d) reflect the labels given by modality 1 and the corresponding marginal distributions seen by modality 2. See text for more details

3.3 A Self-Supervised Piecewise-Linear Classifier

We have so far only discussed the moving of borders in a problem with two 1-Dimensional modalities. In this section we extend the ideas, exactly as we did in Section 2.3.1, to derive an algorithm for moving codebook vectors for non-linear classification in multi-dimensional space.

In the 1-D case the borders b_j can be defined as the midpoint $b_j = (w_{j_1} + w_{j_2})/2$, where w_{j_k} is a codebook vector for modality j. Thus

$$\Delta b_j = (\Delta w_{j_1} + \Delta w_{j_2})/2$$

This means we can induce the desired border motion given by (3.8) in terms of the following motion of the codebook vectors

$$w_{1_i}(n+1) = w_{1_i}(n) + \epsilon(n)\frac{(X_1(n) - w_{1_i}(n-1))}{|X_1(n) - w_{1_i}(n)|}$$

$$w_{1_j}(n+1) = w_{1_j}(n) - \epsilon(n)\frac{(X_1(n) - w_{1_j}(n-1))}{|X_1(n) - w_{1_j}(n)|}$$

if $X_1(n)$ lies in a window of width $2c(n)$ centred at $b_1(n)$; w_{1_i} is the codebook vector belonging to the same class as the closest codebook vector to $X_2(n)$ in Modality 2; and w_{1_j} belongs to the other class [1]. Otherwise no updates are made. Note that this formalization removes the assumption that Class A lies to the left of Class B.

Expanding the problem to more dimensions, and more classes with more codebook vectors per class, complicates the analysis as a change in two codebook vectors to better adjust their border affects more than just the border between the two codebook vectors. However ignoring these effects, a first order approximation suggests the algorithm shown in Figure 3.7 which we call the Minimizing-Disagreement (M-D) algorithm. Note that this algorithm is identical to the modified LVQ2.1 [Kohonen, 1990a] algorithm presented in Section 2.3.1 with one important difference — it does not require the pattern labels. The initialization is unsupervised and in the main algorithm the labeling signal is not the actual class but the class hypothesized by the other modality.

3.3.1 Initial Labels for the Codebook Vectors

To complete the algorithm idea, the codebook vectors need to be given initial labels (The derivation assumes that the current labels are correct). In LVQ2.1 the initial codebook vectors are chosen from among the data patterns that are consistent with their neighbours (according to a k-nearest neighbour algorithm); their labels are then taken as the labels of the data patterns. In order to keep our algorithm unsupervised

[1] How the unsupervised labeling initialization of the codebook vectors is accomplished, is discussed at the end of this section.

the "labeling algorithm" described in Section 3.2.1 is used to derive labels for the initial codebook vectors.

Also due to the fact that the codebook vectors may cross borders or may not be accurately labeled in the initialization stage, they are updated throughout the algorithm by increasing the weight to the output class hypothesized by the other modality, from the neuron representing the closest codebook vector. This has been found to make the algorithm more robust because codebook vectors that are not able to find one particular boundary (due perhaps to no local minimum in the Disagreement Error) may be reassigned to play a role in defining another boundary. The final Minimizing-Disagreement (M-D) algorithm is given in Figure 3.7. We will often refer to step 2 of the algorithm as the initial labeling stage and step 3 as the Minimizing-Disagreement stage.

3.4 Summary

This chapter motivated and developed the Minimizing-Disagreement algorithm that makes use of regularities between patterns to two modalities. The algorithm resembles the modified LVQ2.1 algorithm discussed in Section 2.2 except that the label signal from one modality is not provided by a correct external teacher, but by the output of the other network. Instead of minimizing the misclassifications (which can't be done without a measure of the misclassifications provided by the labels), the algorithm is minimizing the disagreement between the outputs of the two networks. This makes the algorithm fundamentally different from supervised algorithms. Before both networks have developed, they are providing bad label estimates — there is no guarantee that this should help the other network. In the next chapter we review some experimental results that show that the networks are able to assist each other and vastly improve their classification abilities over their initial configuration.

1. Randomly choose initial codebook vectors from data vectors
2. Initialize labels of codebook vectors using the labeling algorithm described in Section 3.2.1
3. Repeat for each presentation of input patterns $X_1(n)$ and $X_2(n)$ to their respective modalities
   - Find the two nearest codebook vectors in modality 1 -- $\vec{w}_{1_{i_1^*}}, \vec{w}_{1_{i_2^*}}$, and modality 2 -- $\vec{w}_{2_{k_1^*}}, \vec{w}_{2_{k_2^*}}$ to the respective input patterns
   - Find the hypothesized output class (C_A, C_B) in each modality (as given by the label of the closest codebook vector)
   - For each modality update the weights according to the following rules (Only the rules for modality 1 are given)
     If neither or both $\vec{w}_{1_{i_1^*}}, \vec{w}_{1_{i_2^*}}$ have the same label as $\vec{w}_{2_{k_1^*}}$ or $X_1(n)$ does not lie within c(n) of the border between them no updates are done, otherwise

$$\vec{w}_{1_{i^*}}(n) = \vec{w}_{1_{i^*}}(n-1) + \epsilon(n)\frac{(X_1(n) - \vec{w}_{1_{i^*}}(n-1))}{||X_1(n) - \vec{w}_{1_{i^*}}(n-1)||}$$

$$\vec{w}_{1_{j^*}}(n) = \vec{w}_{1_{j^*}}(n-1) - \epsilon(n)\frac{(X_1(n) - \vec{w}_{1_{j^*}}(n-1))}{||X_1(n) - \vec{w}_{1_{j^*}}(n-1)||}$$

     where $\vec{w}_{1_{i^*}}$ is the codebook vector with the same label, and $\vec{w}_{1_{j^*}}$ is the codebook vector with another label.
   - Update the labeling weights

Figure 3.7: **The Minimizing-Disagreement (M-D) algorithm—a self-supervised piecewise-linear classifier.**

4 Experimental Results

This chapter presents results of running the algorithm from the previous chapter. The first set of experiments were performed using the Peterson-Barney vowel formant dataset and the second set of experiments deal with visual and auditory data collected from several speakers.

4.1 Benchmark Experiments

In order to compare performance between the M-D algorithm and uni-modal supervised and unsupervised algorithms it is important that the difficulty in both modalities of the M-D problem be the same as that of the uni-modal problem. This is because the M-D algorithm's performance within a modality depends not only on the difficulty of the pattern classification within that modality but also on that of the other "labeling" modality; it is this modality that is providing a better or worse teaching signal. For this reason, we ran a set of experiments that applied the M-D algorithm to two networks both simultaneously learning to classify the same dataset. Accuracy was measured individually (on the training set) for both modalities and averaged. These results were then averaged over 60 runs. The results described below are also tabulated in Table 4.1 and displayed in Figure 4.8.

The dataset chosen was the Peterson and Barney vowel formant dataset which consists of the formant frequencies[1] of 10 vowels spoken in a /hVd/ context (e.g. /had/, /hid/,...). We employed a version, used by and obtained from Steven Nowlan, consisting of the frequencies of the first and second formants of vowels spoken by 75 speakers (32 males, 28 females, 15 children). (See Figures 4.1 and 4.2.) Each speaker repeated each vowel twice except 3 speakers that were each missing one vowel. The raw data was linearly transformed to have zero mean and fall within the range $[-3,3]$ in both components.

In the first experiment, the classes were paired so that the modalities received patterns from the same vowel class. If modality 1 received a pattern from the /a/ vowel

[1]Formant frequencies are peak frequencies in the acoustic signal and reflect resonances of the vocal tract. These resonant frequencies are characteristic of different configurations of the vocal tract and are useful features for vowel recognition.

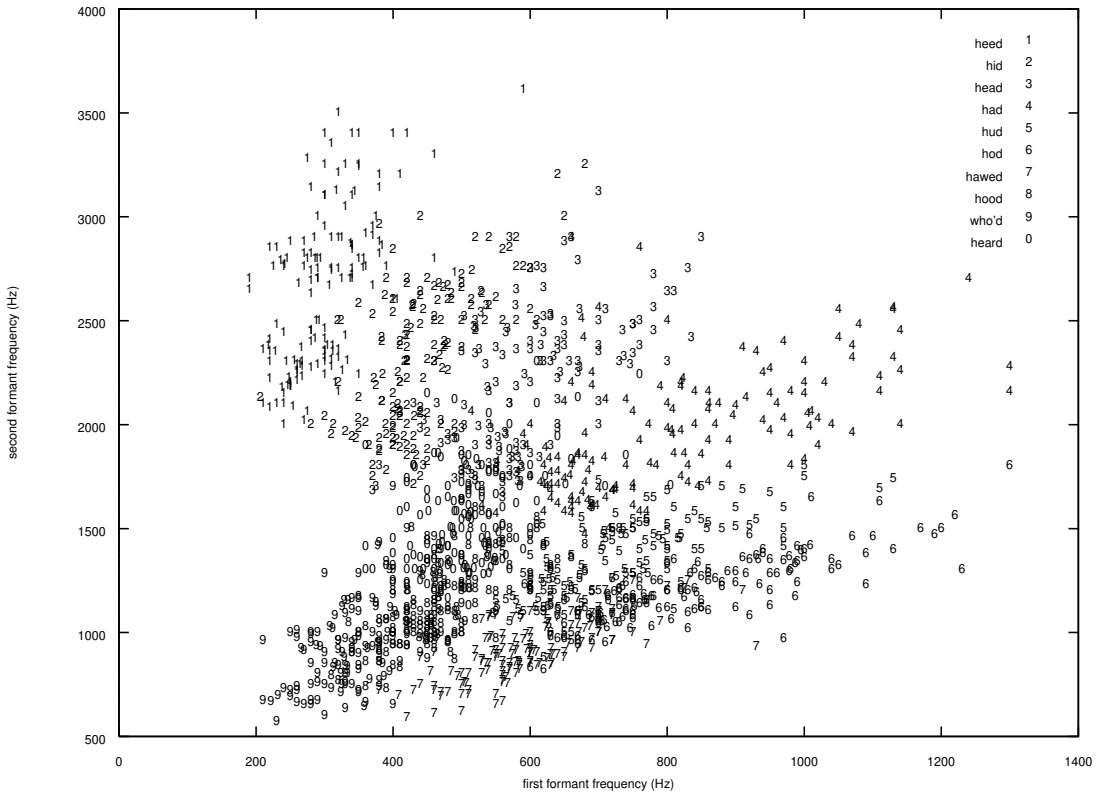

Figure 4.1: **The Peterson-Barney vowel dataset.** Each number (0-9) represents a different vowel class. Each plotted number corresponds to the formants measured from one instance of the uttered vowel plotted in terms of the first two formant frequencies (F1 and F2).

class, modality 2 received an independently chosen pattern from the same (/a/) vowel class and likewise for all the vowel classes (i.e. $p(x_1|C_j) = p(x_2|C_j)$ for all j).

4.1.1 Better Border Placement

After the initial labeling algorithm stage (the second step in Figure 3.7), the accuracy was $60\pm5\%$ reflecting the fact that the initial random placement of the codebook vectors does not induce a good classifier. After application of the Minimizing-Disagreement stage (the third step in Figure 3.7) the accuracy was $75\pm4\%$. At this point the codebook vectors are much better suited to defining appropriate classification boundaries.

The improvement in border placement can be seen by observing the arrangement of misclassified patterns after each stage. Figures 4.3 and 4.4 show the misclassified patterns after the initial labeling and Minimizing-Disagreement stage of the algorithm. These results are averaged over 20 runs in one modality. The results for the other modality are similar. The size of the diamonds corresponds to the fraction of classifiers from the 20 runs that misclassified the pattern. For both stages of the algorithm there

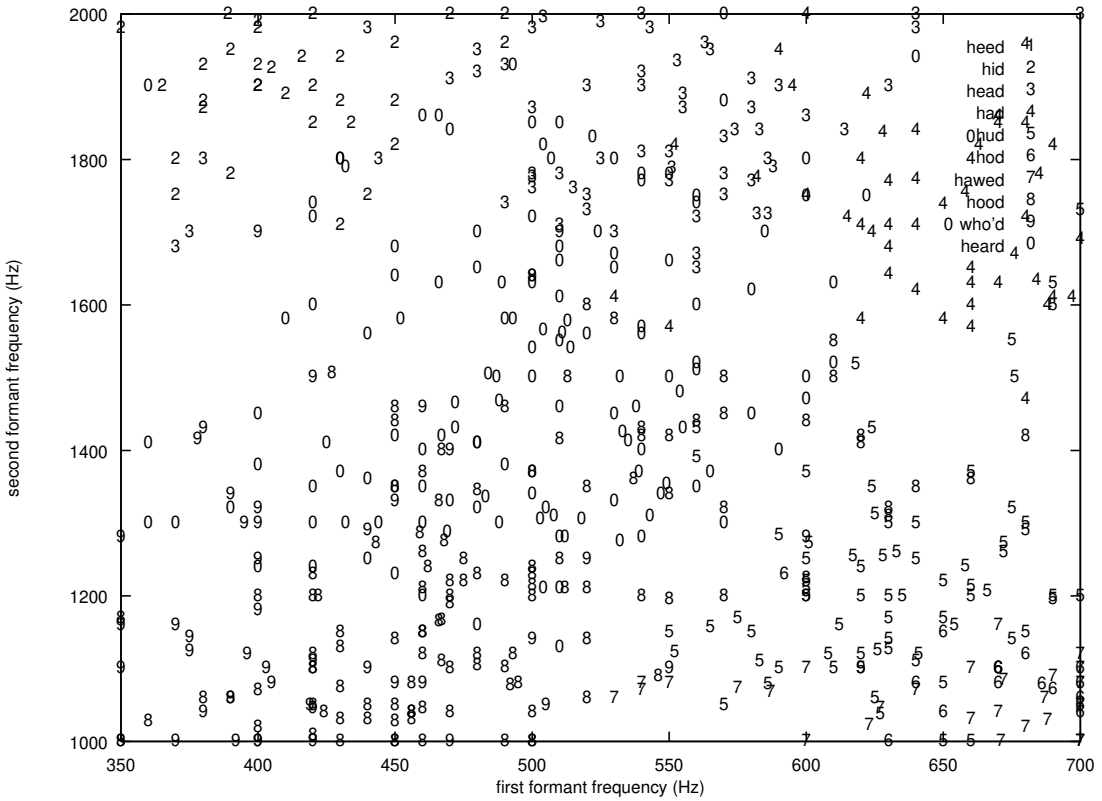

Figure 4.2: **A Close-up of part of the Peterson-Barney vowel dataset.** Note the large amount of overlap between the pattern classes in this central area.

are several patterns that are surrounded by many patterns of the same class and are always classified correctly as well as many patterns that are surrounded by patterns from other classes that are always misclassified. However, after the Minimizing-Disagreement stage many of the patterns between the two extremes, especially those at the edges of the class distributions, are classified more reliably (correctly by more of the classifiers). This shows as a decrease in diamond size from Figure 4.3 to Figure 4.4.

This difference between the two figures is made explicit in Figure 4.5 where the size of the diamonds corresponds to the difference in percentage of classifiers that correctly classified the pattern. Filled diamonds represent positive differences or better classification after the complete Minimizing-Disagreement algorithm and open diamonds better classification after only the initial labeling. The improvements from the M-D stage are in the border regions between classes indicating that minimizing the disagreement tends to find better border placements.

4.1.2 Cycling Helps

The final performance figures were positively correlated with the performance after the label initialization step, which in turn was positively correlated with (and bounded by)

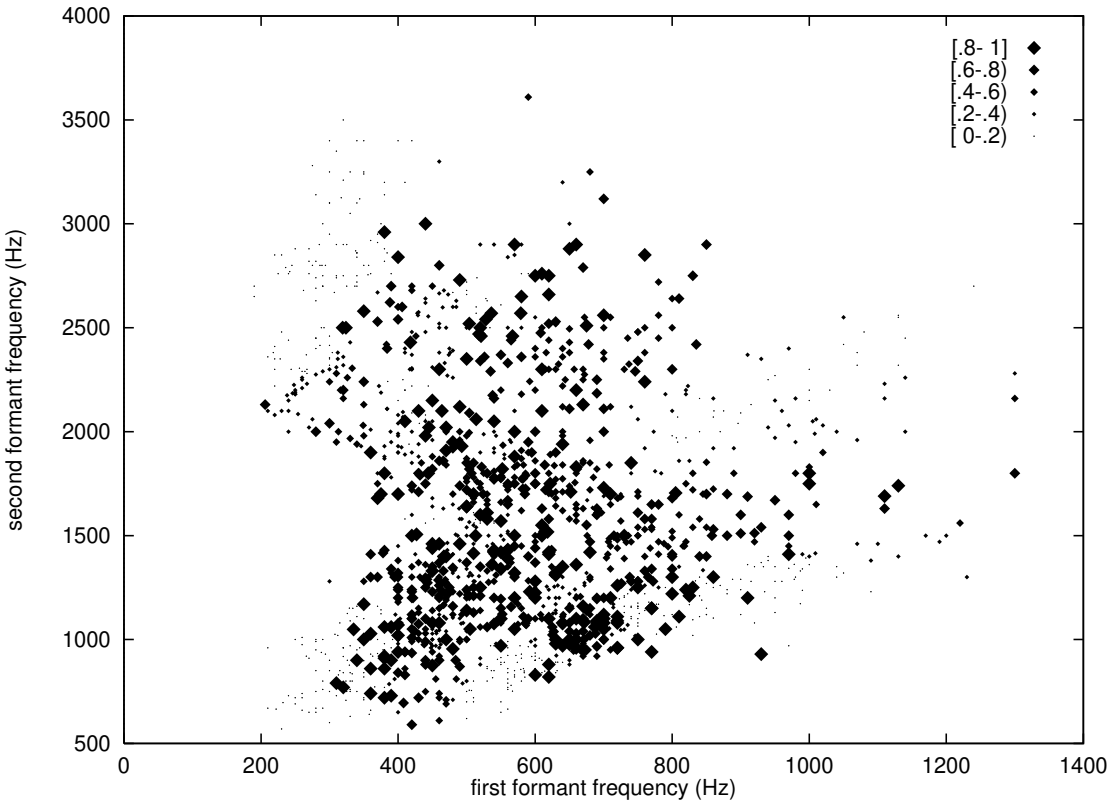

Figure 4.3: **Performance of initial labeling algorithm on the Peterson-Barney dataset.** The size of the diamonds corresponds to the fraction of classifiers that misclassified the pattern.

the best performance possible with the randomly chosen initial codebook vectors (as measured independently with optimal labels). This suggested that improved methods of choosing the initial codebook vector positions and/or labels might result in improved final performance. Thus, we tried using the final codebook vectors from a run of the Minimizing-Disagreement algorithm as the initial codebook vectors for another run (replacing the first step in Figure 3.7). This resulted in improved performance ($73 \pm 4\%$ after step 2 and $76 \pm 4\%$ after step 3). Figure 4.6 shows performance after the initial labeling, first application of the M-D algorithm, and second application of the M-D algorithm for 30 different trials. This cycling of steps 2 and 3 was repeated several more times with no further significant increase in performance (see Table 4.1).

The improvement can be seen in the difference plot in Figure 4.7. Again the size of the diamonds corresponds to the difference in performance. The main area of improvement is in the crowded and very overlapped middle area.

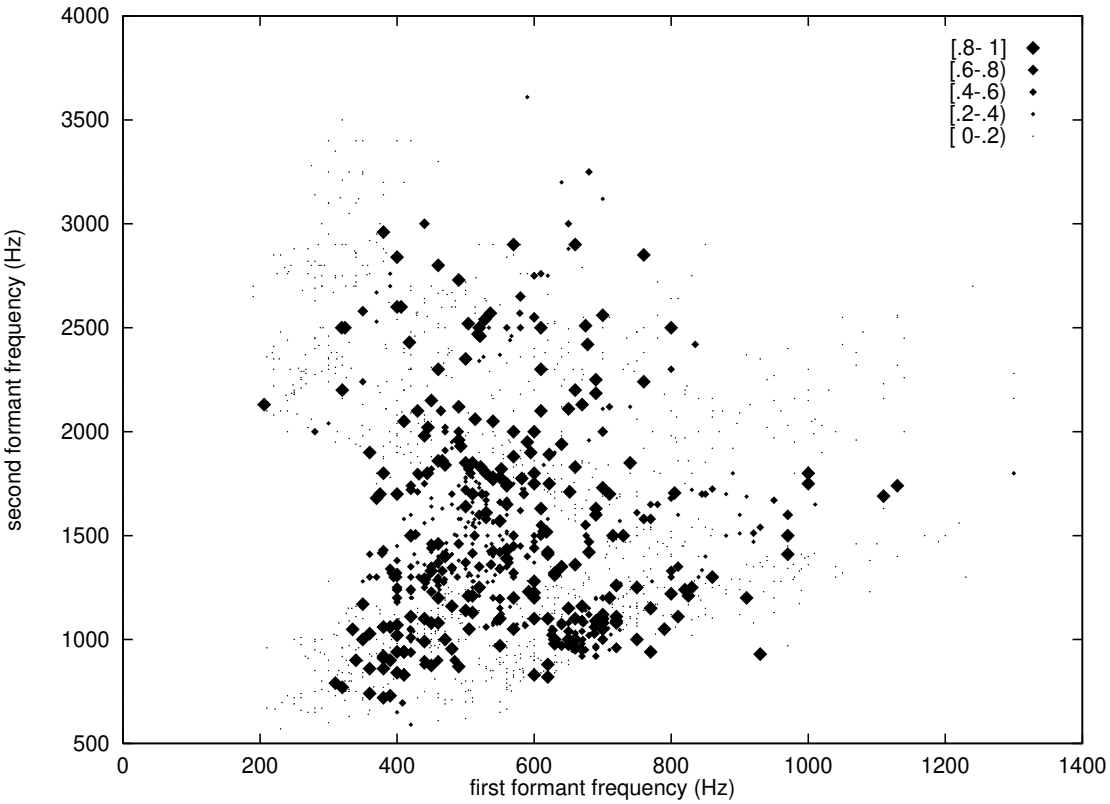

Figure 4.4: **Performance after Minimizing-Disagreement algorithm on the Peterson-Barney dataset.** The size of the diamonds corresponds to the fraction of classifiers that misclassified the pattern.

4.1.3 Improvement when the Confusable Classes are Different in the Two Modalities

One feature of cross-modality information is that classes that are easily confusable in one modality may be well separated in another. This is evident in lip-reading for example. Consonants /b/ and /p/ which differ only in voicing (the presence of vocal cord vibration) are easily acoustically distinguished even in the presence of noise [Miller and Nicely, 1955]. Visually however they are indistinguishable. On the other hand /b/ and /d/, which differ in their place of articulation, are visually distinct but more acoustically confusable in the presence of noise. This difference in confusable classes between the modalities should lead to improved performance with the Minimizing-Disagreement algorithm. As the "labeling" signal for separating the overlapping classes is likely determined between two non-overlapping distributions in the other modality, it will tend to be more reliable. To explore this, more tests were conducted with random pairing of the vowels for each run. For example, presentation of patterns of /a/ vowels to one modality would be paired with presentation of patterns of /i/ vowels to the other. That is $p(x_1|C_j) = p(x_2|C_{\alpha_j})$ for a random permutation $\alpha_1, \alpha_2..\alpha_{10}$. For the labeling stage

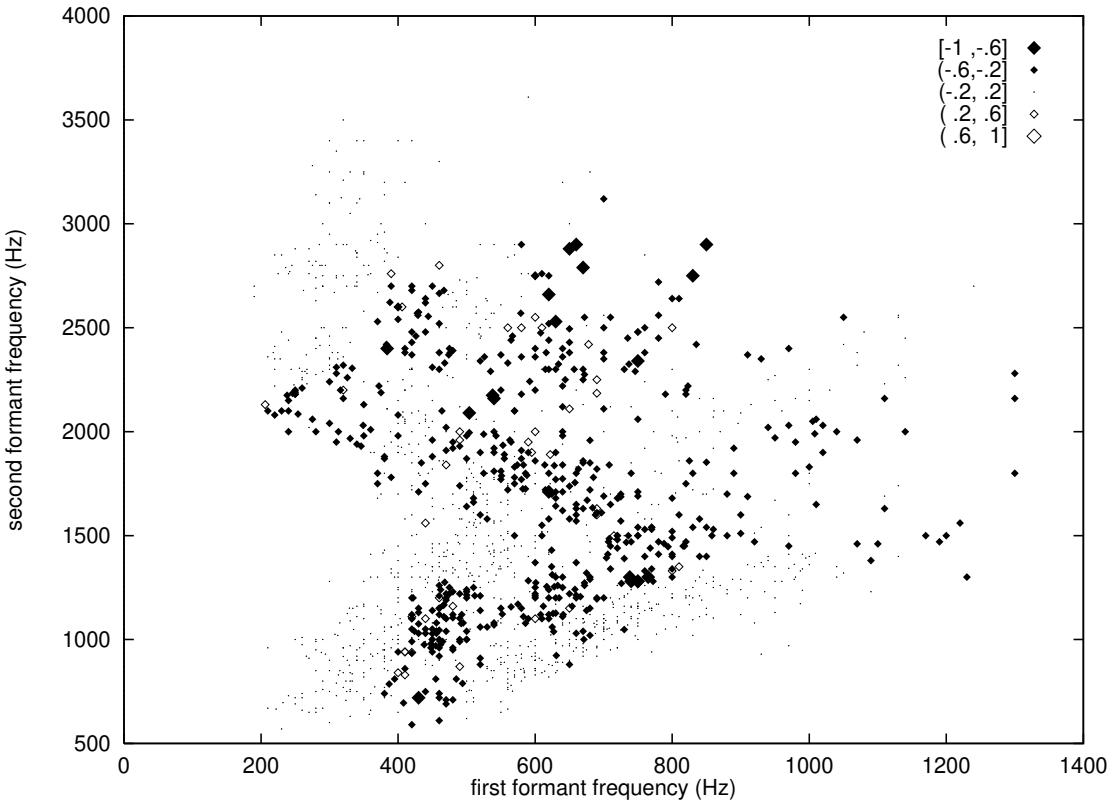

Figure 4.5: **Improvement from running the M-D stage.** The size of the diamonds corresponds to the difference in percentage of classifiers that correctly classified the pattern. Filled diamonds represent better classification after the Minimizing-Disagreement algorithm and open diamonds better classification after only the initial labeling.

the performance was as before ($60 \pm 4\%$) as the difficulty within each modality has not changed. However after the Minimizing-Disagreement algorithm the results were better as expected. After 1 and 2 iterations of the algorithm, $77 \pm 3\%$ and $79 \pm 2\%$ were classified correctly.

4.1.4 Comparison to Other Algorithms

The algorithm compares favourably with the 38% achieved in preliminary experiments with the unsupervised discrete IMAX algorithm [Becker, personal communication] with 30 sigmoidal hidden units for each network (for 6000 iterations of steepest descent). The low performance of the IMAX algorithm results from it settling in local maximums in the mutual information function that don't use all the output classes. The IMAX algorithm is described in Chapter 5.

It also performs better than a hybrid unsupervised-supervised algorithm — Kohonen feature mapping algorithm (with 30 codebook vectors) followed by optimal labeling of the codebook vectors, which achieved 72%. (In fact if the same optimal labeling algo-

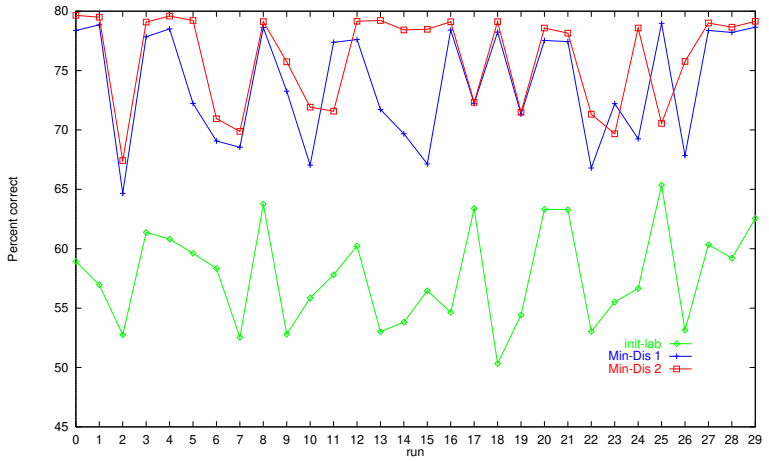

Figure 4.6: **The performance (for 30 different initial configurations).** Performance was measured after: initial labeling (init-lab), one application of the M-D algorithm (Min-Dis 1), and two applications of the M-D algorithm (Min-Dis 2)

Table 4.1: **Tabulation of performance figures (mean percent correct and sample standard deviation over 60 trials and 2 modalities).** The heading $i-j$ refers to performance measured after the j^{th} step during the i^{th} iteration. (Note Step 1 is not repeated during the multi-iteration runs).

	1-2(IL)	1-3(M-D)	2-2	2-3(M-D2)	3-3	4-3	5-3
same-paired vowels	60 ± 5	$\mathbf{75 \pm 4}$	73 ± 4	$\mathbf{76 \pm 4}$	76 ± 4	76 ± 4	76 ± 4
random pairing	60 ± 4	$\mathbf{77 \pm 3}$	77 ± 3	$\mathbf{79 \pm 2}$	79 ± 2	79 ± 2	79 ± 2

rithm is applied to the codebook vectors resulting from the M-D algorithm, an average performance of 76% and 78% (for applying after one or two iterations respectively) results.) Performance is not as good as that of the related fully supervised algorithm LVQ 2.1, which achieved an average performance of 80%, but is comparable to the performance of (supervised) back-propagation (with 25-200 hidden layer neurons) which obtained average performances of 73.4-78.5% [Nowlan, 1991]. Nowlan's more complicated mixture model (supervised) achieved an average performance of 86.1%. These results are all shown in Figure 4.8.

Figure 4.9 is a comparison between the results from the 2-stage Minimizing-Disagreement algorithm (with same-vowel pairs) and the results from using the supervised LVQ2.1 algorithm. Again the size of the diamonds reflects the magnitude of the difference in performance and the colour reflects the sign. The larger number of open diamonds reflects the slight superiority of the supervised algorithm. More strikingly though, this plot emphasizes the borders between the regions. The two algorithms differ so slightly in the resulting classifiers that the only patterns classified differently are right at the borders. The two algorithms tend to pick slightly different borders resulting in some

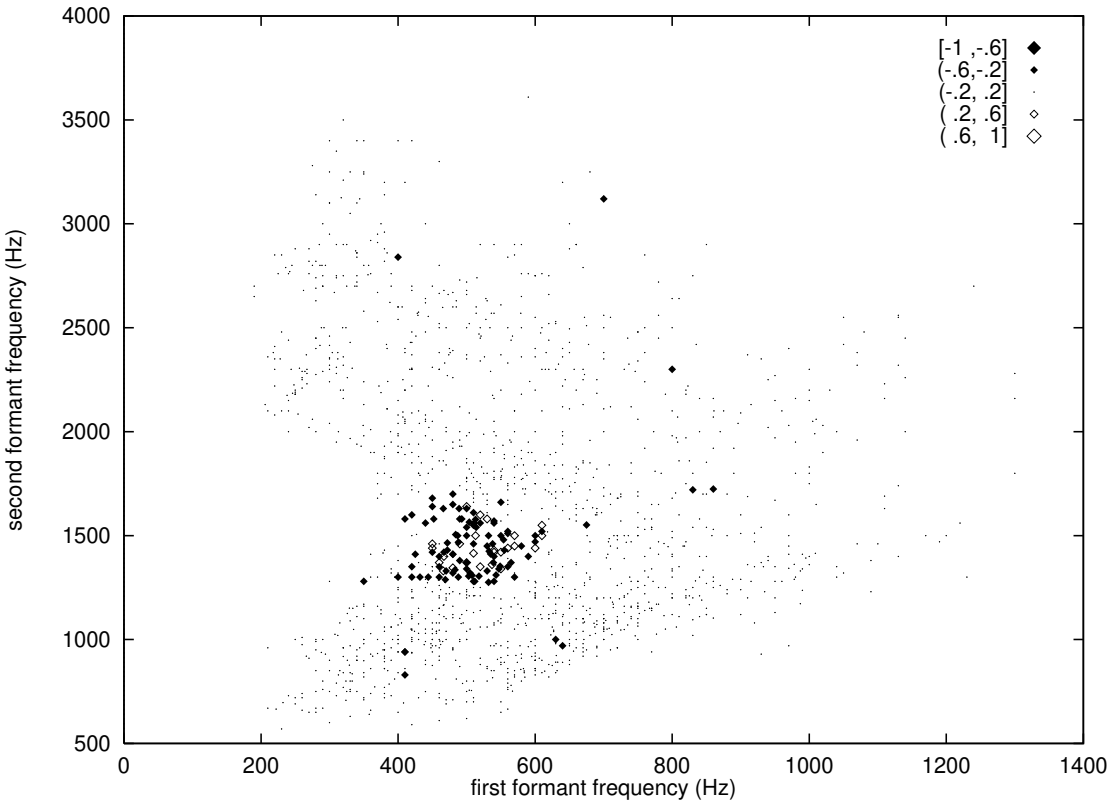

Figure 4.7: **Difference in Performance after 1 or 2 iterations of the M-D algorithm.** The size of the diamonds corresponds to the difference in percentage of classifiers that correctly classified the pattern. Filled diamonds represent better classification after the two iteration algorithm and open diamonds better classification after the first iteration.

border patterns being better classified by one algorithm and others by the other.

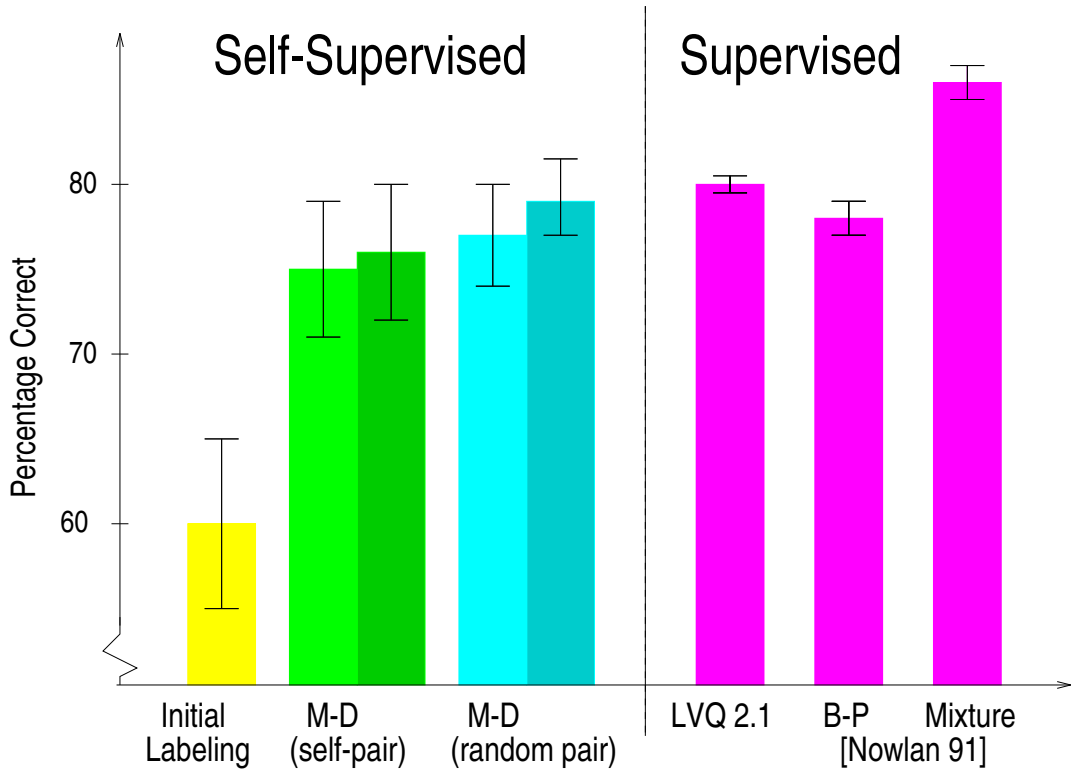

Figure 4.8: **Results from different algorithms on the Peterson-Barney Vowel Dataset.** The double bars for the M-D algorithms represent results from 1 iteration and 2 iterations. The error bars represent 1 standard deviation.

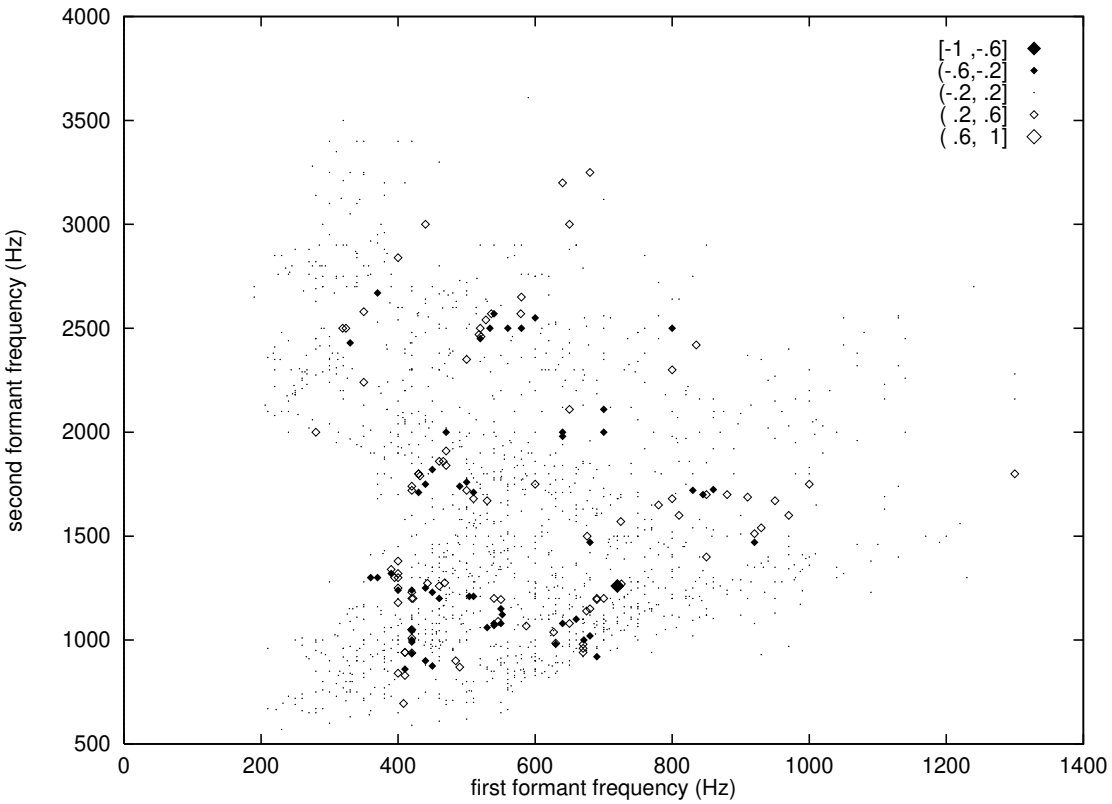

Figure 4.9: **Difference in Performance between the supervised and M-D algorithm.** The size of the diamonds corresponds to the difference in percentage of classifiers that correctly classified the pattern. Filled diamonds represent better classification with the Minimizing-Disagreement algorithm (with two iterations) and open diamonds better classification with the supervised algorithm.

4.2 Cross-Modal Experiments

4.2.1 Data Collection

Data were collected using an 8mm camcorder from 5 male English speakers as they spoke 26 iterations of /ba/ /va/ /da/ /ga/ /wa/ [2]. Each set of 10 utterances (twice through the set) was preceded by a clap using a clapboard arrangement similar to that used in commercial movie production for matching the visual and auditory signals. The camera recorded 30 frames a second and was positioned to view the tip of the nose through chin of the speaker. The audio was recorded through a cardioid microphone positioned approximately 5 inches from the speaker's mouth.

The acoustic data were transferred to a Sparc LX, low-pass filtered and segmented automatically (using time-domain wave magnitude) using the ESPS software from Entropic Research Laboratory, Inc. Each utterance was taken from 50msec before the automatically detected utterance start to 50msec after[3]. These utterances were then encoded using a 24 channel mel code[4] over 20msec windows overlapped by 10msec. This gave a (24*9 = 216) dimension auditory code for each utterance.

The visual data was processed using software designed and written by Ramprasad Polana [Polana, 1994]. The visual frames were digitized as 64×64 8 bit gray-level images using the Datacube MaxVideo system. The video and auditory tracks were aligned using the clapboard arrangement. The visual detection of the clap was performed manually which allowed alignment to within 1 video frame (1/30 second). (For an example of a video sequence showing a clap frame see Figure 4.10). The frame of the clap was matched to the time of the acoustically detected clap allowing the automatic segmentation obtained from the acoustic signal to be used to segment the video. The segments were taken as 6 frames before the acoustically determined utterance onset and 4 after. The normal flow was computed using differential techniques between successive frames. Each pair of frames was then averaged resulting in 5 frames of motion over the 64×64 pixel grid. The frames were then divided into 25 equal areas (5×5) and the motion magnitudes within each frame were averaged within each area. This gave a final visual feature vector of dimension (5 frames * 25 areas)= 125.

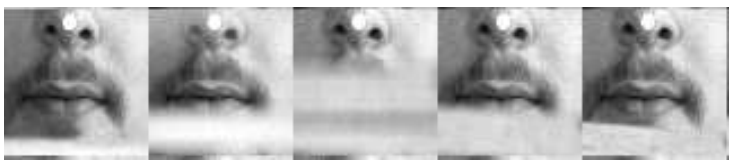

Figure 4.10: **Example frames showing the clap detection.** The clap is in the centre frame.

[2] A few speakers spoke more iterations.

[3] This was to ensure that all the consonantal information was retained.

[4] linear spacing below 1000Hz and logarithmic above 1000Hz

Figure 4.11: **Example /ba/ utterance.**

Figure 4.12: **Example /va/ utterance.**

4.2.2 Results

The data for one speaker was unusable due to a problem with the video tape. The training set was made up of each of the other speakers' first 20 cycles through the utterances (minus a few cycles that could not be used due to lost frames during digitization). The test set was made up of the next 6 cycles [5]. For all experiments 30 codebook vectors were used in the auditory pattern space and 60 in the visual pattern space. As the auditory signal contains more reliable information, the relative contributions from the auditory and visual networks during the labeling algorithm were weighted (1.5 to 1) in favour of the auditory choice.

We first benchmarked the dataset by running the supervised LVQ2.1 algorithm. Using 30 codebook vectors for the auditory patterns we achieved an accuracy of 99% on the training set and 97% on the test set. Using 60 codebook vectors for the visual patterns the performance was 83% on the training set and 60% on the test set. We also ran the unsupervised-supervised algorithm of Kohonen feature mapping followed by optimal labeling of the codebook-vectors. This gave accuracies of 84% and 55% on the two training sets.

Ideally we would like to test the M-D algorithm by presenting to the auditory and visual networks the pairs of patterns that occurred together. However, to get a good covering of the spaces, many utterances need to be collected. Due to the time involved in the current method of synchronizing the audio and video (they are processed separately and synchronized manually through the visual clap detection) it was decided to run preliminary experiments that artificially increase the dataset using the technique employed with the vowel dataset. This technique makes the assumption that within an

[5] For some speakers there were a few extra cycles that were also included.

Figure 4.13: **Example /da/ utterance.**

Figure 4.14: **Example /ga/ utterance.**

Figure 4.15: **Example /wa/ utterance.**

utterance class the exact auditory and visual patterns are independent and thus each auditory pattern can be paired with each visual pattern from the same class (not just the one that it actually co-occurred with). For example, an individual acoustic pattern from a /ba/ utterance is randomly paired with a visual sample from a randomly chosen /ba/ utterance.

For these experiments, the Minimizing-Disagreement algorithm was applied to codebook vectors resulting from the unsupervised Kohonen learning algorithm instead of randomly initialized ones in the respective spaces [6]. The initial labeling algorithm on the codebook vectors resulting from the Kohonen learning algorithm resulted in 72% on the training set and 68% on the test set for the auditory network and 48% (training) and 36% (test) for the visual network. The Minimizing-Disagreement stage was able to greatly increase the classification performance from this initial state to 97% and 72% for the auditory network and 82% and 58% for the visual network. The performance results are summarized in Figure 4.16.

While the previous results were encouraging, it was important to demonstrate the algorithm in the fully unsupervised way, making no assumptions about independence between the modalities, and using only the cross-modality information sampled from the environment. In order to accurately sample the space, we restricted the problem to that of a single speaker. This speaker repeated 120 cycles of /ba/, /va/,/da/,/ga/,/wa/. The first 100 cycles (minus two that lost frames during the digitization) were used as the training set and the last 20 were used as the test set. Again for this cross-modal dataset the M-D algorithm was applied using the results of the Kohonen learning algorithm as the initial codebook vector positions. The algorithm achieved accuracies of 92% (train), 92% (test) on the auditory data and 91% (train),78% (test) on the visual data. For comparison the supervised LVQ2.1 algorithm, as well as the M-D algorithm using full-pairings as before, were also run on this dataset. The supervised results were 99% (aud-train) ,95% (aud-test) and 96% (vis-train), 82% (vis-test). The M-D algorithm using the artificial increased pairing resulted in 98% (aud-train), 95% (aud-test) and 95% (vis-train), 80% (vis-test). These results are displayed in Figure 4.17.

[6]Initial experiments suggested that this might provide better results but later tests indicated there was not much difference.

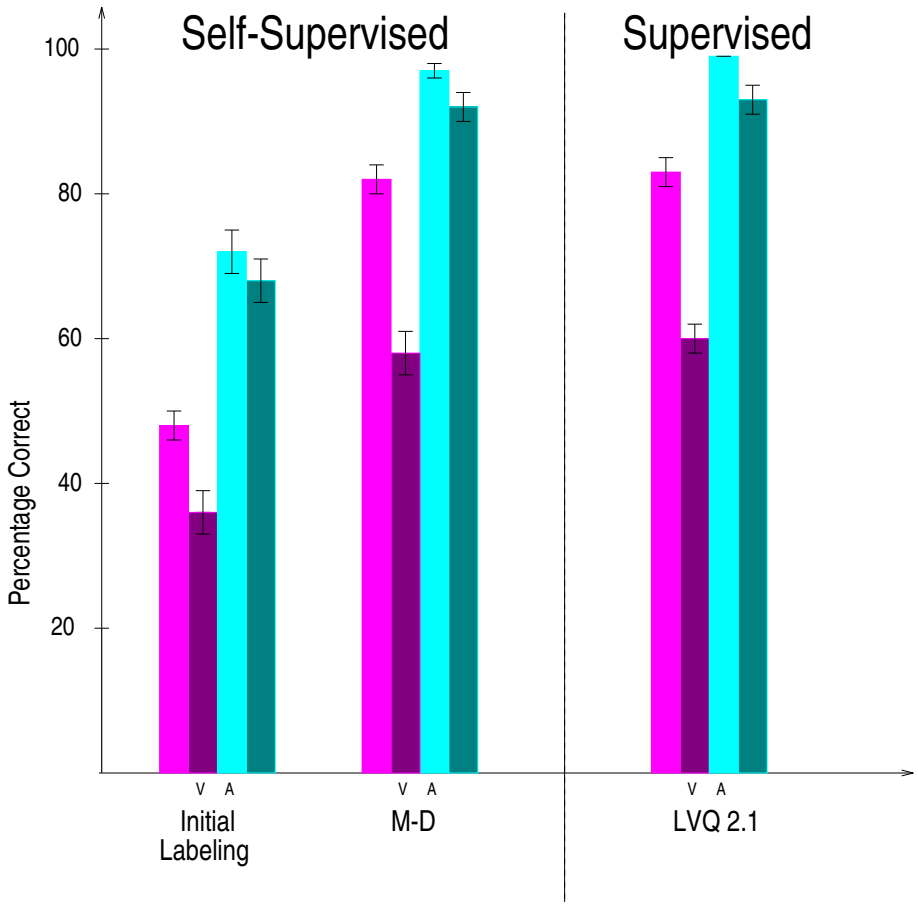

Figure 4.16: **Results on the preliminary cross-modal dataset.** The two leftmost bars in each set of four give the performance of the visual network and the rightmost bars show the auditory network's performance. Within the two bars for each modality, the lighter and leftmost bar represents performance on the training set. The darker, rightmost bars give results on the test set. The error bars represent 1 standard deviation.

The results demonstrate that for one speaker, the natural lip-sound co-occurrences were enough to give performance within 7% percent of the supervised results on the training set and within 4% of the supervised results on the test set. More data collection is needed to determine if the fully unsupervised algorithm would work on the multi-speaker problem.

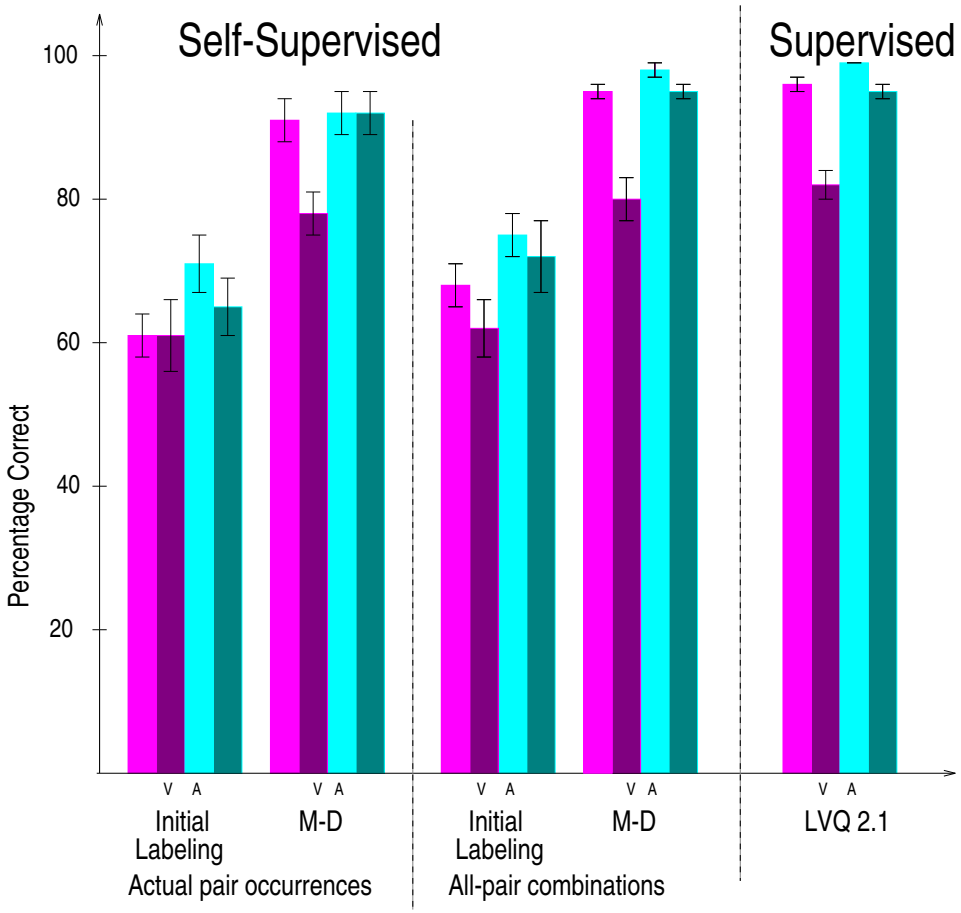

Figure 4.17: **Results on the single-speaker cross-modal dataset.** The two leftmost bars in each set of four give the performance of the visual network and the rightmost bars show the auditory network's performance. Within the two bars for each modality, the lighter and leftmost bar represents performance on the training set. The darker, rightmost bars give results on the test set. The error bars represent 1 standard deviation.

5 Other Theoretical Issues

5.1 Minimizing Disagreement through LTD

In giving a plausible implementation for the competitive learning algorithms we postulated a lateral inhibitory connection network and Hebbian updates for the active neurons. In order to give an implementation for the Minimizing-Disagreement algorithm it is necessary in addition to postulate a mechanism for the anti-Hebbian weight updates required for the weight vector of the closest neuron from the wrong class.

The biophysical properties described in [Artola *et al.*, 1990] from slice recordings in visual cortex of the rat match exactly what is required. Artola, Bröcher and Singer found that regular spiking cells in cortical layers II and III have two thresholds for synaptic modification—a low threshold for the induction of long term depression (LTD) and a higher threshold (which has long been associated with the NMDA receptor) for long term potentiation (LTP). This could be utilized as a substrate for the Minimizing-Disagreement algorithm if we postulate the existence of feedback connections from neurons reflecting the output of the other modality to the codebook vector neurons. The neurons that are active in the codebook vector layer but not receiving the additional reinforcing activation from the output "copy" neurons (because they do not support the same class) would undergo homosynaptic long term depression (LTD) (i.e. the strength of their active synapses would decrease) and the ones receiving the additional activation from the agreeing output would undergo long term potentiation (increase the strength of their active synapses).

As in Competitive Learning the codebook vector neurons must be connected with lateral inhibitory connections such that only the maximally activated neuron is able to significantly influence the output neurons. The basic idea is shown in Figure 5.1. More details are given below.

We don't make any strong claims about any of the other subtle rules of the algorithm. In fact we have previously experimented with an algorithm that is much simpler and only requires the LTP/LTD mechanism mentioned above [de Sa and Ballard, 1992]. However we could speculate on some of the subtle implementation details that would be required to implement the Minimizing-Disagreement algorithm precisely. First, if both codebook vectors are from the same (correct) class no weight changes are desired by the algorithm. To detect this, the network needs a way of having the second most

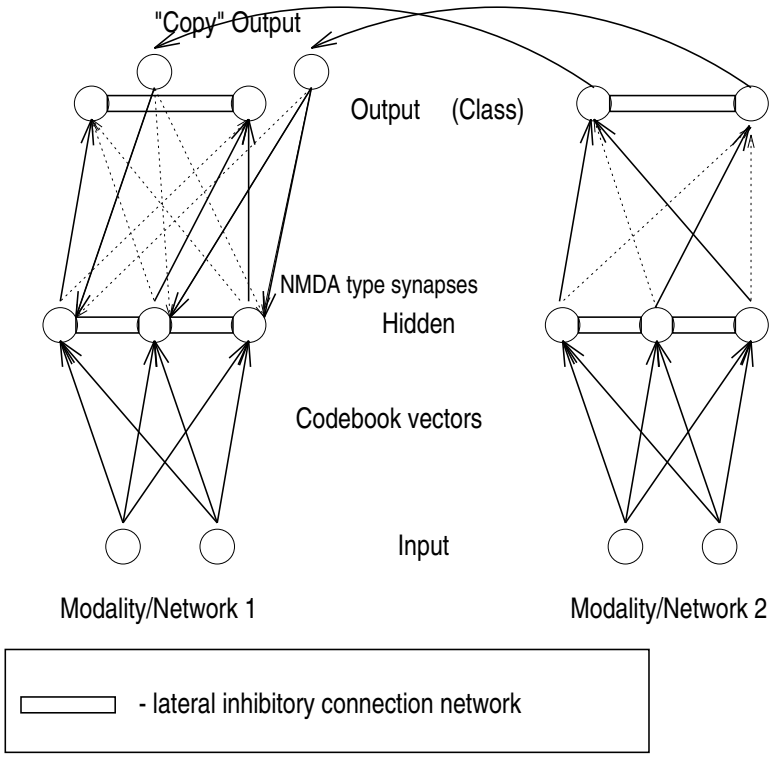

Figure 5.1: **Connections required to enable appropriate synapse modification in modality 1 according to the Minimizing-Disagreement algorithm**

activated neuron able to influence a set of output neurons. This could be done with other sets of neurons closely associated with their counterparts (for example in the same cortical column), codebook vector B and output B neurons, in which the inhibition is less, allowing two neurons to be active. The network must detect that there is only one significantly activated neuron at this second output layer. This could easily be done neurally with a collection of neurons that summed the activity. Alternatively it could be detected through another set of neurons designed to detect agreement. They would be paired with the previous ones (in the same column for example) but have higher thresholds; thus they would only be activated if both codebook vectors had strong activation (i.e. picked the same class). The algorithm also requires inhibition from these cells to some part of the network connecting the output neurons of the other modality to the codebook vectors.

Second, if neither codebook vector is from the correct class the algorithm specifies no weight changes (no LTD). This can be arranged by having neurons that compare the activation of the copy neurons with the output B neurons. These "compare" neurons would have positive connections from the copy and negative connections from the paired B output neuron. These neurons could then provide some sort of input or modulatory signal to offset the LTD that would normally occur.

Third there is the window requirement. The Minimizing-Disagreement algorithm

does not update weights unless the pattern falls in the "window" between two codebook vectors. This can be detected since, if the activations are not close only one neuron in the B codebook vector layer will be activated due to the lateral inhibition. Neurons that calculate the general activation level could detect this and provide some sort of input or modulatory signal to offset any learning. Alternatively with proper thresholds this case may be covered above. If the sole active neuron is from the correct class then LTP will be blocked by the first case above as only one output B neuron will be active. If the sole active neuron is from the wrong class then this will be handled by the second case above.

5.2 Comparison to Maximizing Mutual Information

The Minimizing-Disagreement algorithm is similar to the Discrete IMAX algorithm of [Becker and Hinton, 1992; Becker, 1993]. The basic idea is the same — to take advantage of the fact that the outputs from two (or more) networks receiving related input should agree. Instead of minimizing the disagreement, Becker and Hinton maximize the mutual information between the outputs of the two networks. This section gives a brief description of the IMAX algorithm and then highlights some of the differences from the M-D algorithm.

Mutual information is a measure of the redundancy between variables. The mutual information between two variables Y_1 and Y_2 is defined as

$$I_{Y_1;Y_2} = H(Y_1) + H(Y_2) - H(Y_1, Y_2)$$

where $H(Y) = -E[logP(Y)]$ is the entropy of the distribution of Y and $H(Y_1, Y_2)$ is the entropy of the joint distribution $P(Y_1, Y_2)$. The entropy of a probability distribution measures the uncertainty of the output. $H(Y)$ is minimized if there is only one possible value of Y and is maximized if all outputs are equiprobable.

For n discrete outputs (representing n output classes)

$$H(Y_1) = -\sum_{i=1}^{n} P(Y_1 = C_i) \log P(Y_1 = C_i)$$

and

$$H(Y_1, Y_2) = -\sum_{i=1}^{n}\sum_{j=1}^{n} P(Y_1 = C_i, Y_2 = C_j) \log P(Y_1 = C_i, Y_2 = C_j)$$

where variables Y_1 and Y_2 can take on any of the n discrete values.

The output neurons in the Becker network are not hard competitive neurons as in the M-D algorithm. Instead they are probabilistic units that in the two class case have probability of outputting 1 given by [1]

$$P(Y_1 = 1|\vec{\xi}) \quad = \quad \frac{1}{1 + e^{-s}}$$

[1] for ease of simulation they are simulated deterministically and the output is the expected value

$$
\begin{aligned}
&= \frac{1}{1+e^{-w_1\cdot\vec{\xi}+t_1}} \\
&\equiv g(t_1, w_1, \vec{\xi}) \qquad (5.1)
\end{aligned}
$$

In the n class case the probabilities of the outputs are given by the softmax function[Bridle, 1990]

$$
\begin{aligned}
P(Y_1 = i|\vec{\xi}) &= \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}} \\
&= \frac{e^{w_i\cdot\vec{\xi}+t_i}}{\sum_{j=1}^{n} e^{w_j\cdot\vec{\xi}+t_j}} \qquad (5.2)
\end{aligned}
$$

In order to compare the algorithms, we consider again the 1-Dimensional case introduced in Chapter 2 and again in Section 3.2.2. This problem can be implemented in an IMAX architecture where each network consists of one input and one output neuron. We let Class A be represented by output 1 and Class B be represented by output 0.

$H(Y_1)$ is calculated as

$$H(Y_1) = -P(Y_1 = 0)\log P(Y_1 = 0) - P(Y_1 = 1)\log P(Y_1 = 1)$$

where

$$P(Y_1 = 1)) = \int_{-\infty}^{\infty} [P(C_A)p(x_1|C_A) + P(C_B)p(x_1|C_B)]g(t_1, w_1, x_1)dx_1$$

and

$$P(Y_1 = 0) = \int_{-\infty}^{\infty} [P(C_A)p(x_1|C_A) + P(C_B)p(x_1|C_B)](1 - g(t_1, w_1, x_1))dx_1$$

Note that for this 1-Dimensional case the ratio of the bias term to the input weight (t_i/w_i) for the output neuron corresponds to the border in the Minimizing-Disagreement derivation.

$H(Y_1, Y_2)$ is calculated as

$$
\begin{aligned}
H(Y_1, Y_2) &= -P(Y_1 = 0, Y_2 = 0)\log P(Y_1 = 0, Y_2 = 0) \\
&- P(Y_1 = 0, Y_2 = 1)\log P(Y_1 = 0, Y_2 = 1) \\
&- P(Y_1 = 1, Y_2 = 0)\log P(Y_1 = 1, Y_2 = 0) \\
&- P(Y_1 = 1, Y_2 = 1)\log P(Y_1 = 1, Y_2 = 1)
\end{aligned}
$$

where

$$P(Y_1 = 1, Y_2 = 0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1, x_2)g(t_1, w_1, x_1)(1 - g(t_2, w_2, x_2))dx_1dx_2$$

and the other terms are similarly calculated. $f(x_1, x_2)$ is the joint probability distribution (as given in (3.5)) of inputs x_1, and x_2.

In order to compare the two algorithms we look at the value of their respective energy functions for different values of the borders b_1 and b_2. First note, however, that the joint entropy term in the mutual information energy function depends on both the position and steepness of the sigmoidal border (given by $b_i = t_i/w_i$ and w_i respectively). To produce figures of the energy function dependent only on the position of the border , we consider the case where $w_i, t_i \to \infty$ (with b_i kept constant) and approximate the stochastic neurons with hard threshold units (that output 0 for $x_i < b_i$ and 1 otherwise). Figure 5.2 shows that $H(Y_1)$ is maximized if each of the 2 outputs for network 1 has equal probability. Similarly $H(Y_2)$ is maximized if each of the 2 outputs for network 2 has equal probability (see Figure 5.3). The $H(Y_1, Y_2)$ term is similarly maximized when there is maximal uncertainty about the likely output. Thus $-H(Y_1, Y_2)$ is maximized if only one combination of outputs occurs. As shown in Figure 5.4, the $-H(Y_1, Y_2)$ term is maximized as the borders (b_1, b_2) tend to extreme values $(-\infty, -\infty), (-\infty, \infty), (\infty, -\infty), (\infty, \infty)$. There are four maxima as opposed to two for the M-D algorithm because there is no restriction that Class A for modality 1 has to be paired with Class A for modality 2.

The total mutual information $I(Y_1, Y_2)$ function is the sum of these 3 functions. The $H(Y_1)$ and $H(Y_2)$ terms serve to reduce the energy at the extreme points in the $H(Y_1, Y_2)$ function leaving $I(Y_1, Y_2)$ with a maximum within the input distribution (see Figure 5.5).

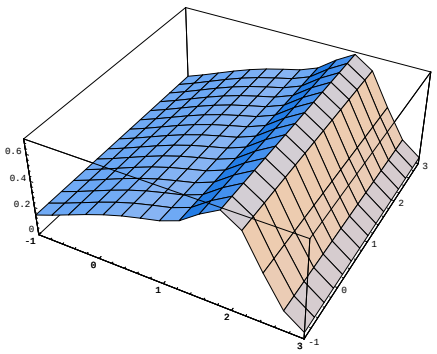

Figure 5.2: **H(Y_1) as a function of b_1 and b_2 for the distributions shown in Figure 3.4.**

For comparison, we have replotted Figure 3.5, the negative of the Minimizing Disagreement energy function, over the same domain. This is shown in Figure 5.6. Comparing Figures 5.5 and 5.6, several differences should be noted. The first difference is that the two algorithms are optimizing different functions and therefore the desired

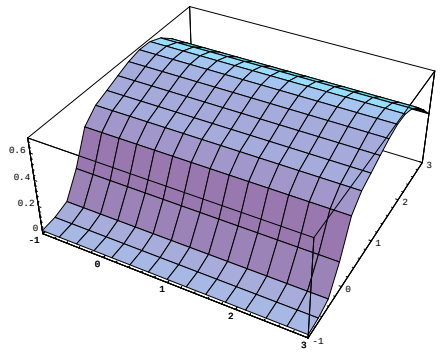

Figure 5.3: **$\mathbf{H}(Y_2)$ as a function of b_1 and b_2 for the distributions shown in Figure 3.4.**

optima differ. The two algorithms find different boundaries in different circumstances. However, a greater difference is that the energy function for IMAX has one global maximum, whereas the energy function for the M-D algorithm has maxima at extreme values of the borders b_1, b_2; the desired point is only a local optimum. In fact as mentioned earlier, depending on the distributions, this local optimum may not exist. The difference is due to the addition of the individual entropy terms $H(Y_1), H(Y_2)$ in the IMAX algorithm which penalize the extreme values. The requirement of these terms in the IMAX algorithm but not the M-D algorithm, is related to the different ways that the two algorithms scale up to non-linear higher dimensional classification problems.

Whereas the M-D algorithm scaled up by abstracting general rules from the 1-D derivation (to move a hidden layer of competitive-style codebook vectors), the IMAX algorithm is derived directly for multi-layer sigmoidal networks. The Mutual information energy function is differentiated with respect to each weight in the network. These derivatives are calculated in the same way as the back-propagation algorithm, where derivatives with respect to the weights to the final layer are calculated first and then communicated backwards for use in calculating derivatives with respect to the earlier weights. Alternatively for a sacrifice of not minimizing the global mutual information, the objective function can be applied in a hierarchical manner to 2-layer networks[Becker, 1992]. In addition, Becker's algorithm has been generalized to continuous output variables [Becker, 1992].

The different architectures mean that the algorithms optimize with respect to different parameters. The IMAX algorithm performs gradient descent on all the weights with respect to the Mutual Information. For this reason, it is imperative that a non-trivial lo-

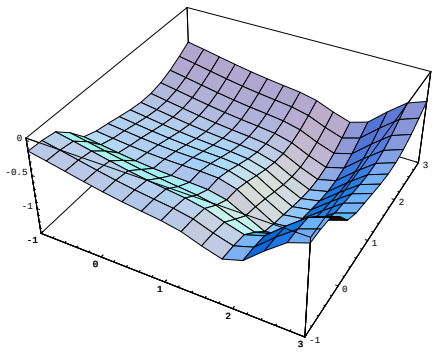

Figure 5.4: **-$H(Y_1, Y_2)$ as a function of b_1 and b_2 for the distributions shown in Figure 3.4.**

cal optimum exist. The M-D algorithm performs gradient descent on the input weights, subject to fixed labels (or fixed hidden to output weights). At the same time the hidden to output weights are updated competitively. In a sense the competitive updating of the labeling weights plays a role similar to that of the individual entropy terms in the mutual information algorithm. In fact the competitive units are a stronger requirement for using all available output labels, as demonstrated with the Peterson-Barney vowel dataset studies. The poor performance of the discrete IMAX algorithm was due to the algorithm finding a local maximum where only four of the output classes were used. In the M-D algorithm, the competitive learning of the output weights ensures that each class receives some codebook vectors. The only way an output class will not be used is if its codebook vectors don't actually win any input patterns [2] Also, as mentioned earlier the ability to change labels while rearranging borders gives the algorithm more robustness.

In terms of biological plausibility, the IMAX algorithm requires the storage of the individual and joint probabilities for the derivative calculations and requires the communication of the mutual information derivatives to the neurons feeding the output (including back-propagation for non-linear classifiers). The M-D algorithm is designed as an on-line algorithm that doesn't require storage of extra state information. Instead of back-propagating the mutual information derivatives, the only communication requirements are for communicating the maximally activated neurons within and between

[2]This could be considered a disadvantage in that it appears that the number of output classes must be pre-specified, however providing extra output neurons should just allow finer divisions within classes (e.g. male /a/ vs female /a/). More work should be done to investigate this as well as the possibility for hierarchical classification.

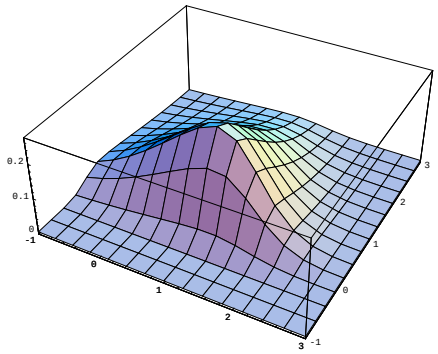

Figure 5.5: $I = H(Y_1 + H(Y_2) - H(Y_1, Y_2)$ **as a function of** b_1 **and** b_2 **for the distributions shown in Figure 3.4.**

layers. Biologically plausible lateral interconnection schemes for this were discussed in the previous Section.

To summarize, the IMAX algorithm is more general particularly given its extension to the continuous case [Becker, 1992]. It is also potentially more robust as its energy function is designed to ensure a finite optimum. The Minimizing-Disagreement algorithm was designed for discrete classification tasks and seems to perform significantly better in this area. It is also less complicated in terms of the computation and communication requirements and thus seems closer to a biologically plausible algorithm. In these ways the relative strengths and weaknesses between the algorithms parallel those between their related supervised algorithms of back-propagation[Rumelhart *et al.*, 1986] and learning vector quantization[Kohonen, 1990b].

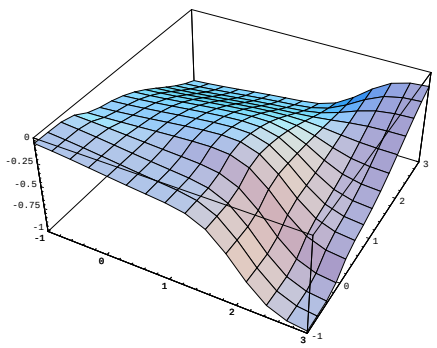

Figure 5.6: **The Disagreement energy function as a function of b_1 and b_2 for the distributions shown in Figure 3.4.** The function is actually the negative of the Disagreement energy function for direct comparison with the IMAX energy function. This graph is the same as that in Figure 3.5 plotted over the same domain as Figures 5.2 through 5.5

5.3 Minimizing Disagreement as an Efficient Missing Data Solution

There are many occasions during learning when some or all of the data encountered contain some missing values. This can occur for example due to occlusion, noise on some sensors, or simply from having unlabeled patterns (where the output category is missing).

It is often worth using the data for the information they do contain. If the particular dimensions are assumed to be missing independently of both their value on that particular dimension and from each other, the optimal Bayes solution is to fill in the data using the other input pattern dimensions together with information about the joint distribution of the input patterns (drawn from experience with the other training patterns). The idea is to integrate the computation over the set of possible values, weighted by their probability given the values on the observed dimensions.

Each given point restricts the area over which the integration is performed (if all dimensions are given we are restricted to that point). For classification we can determine the $P(C_i|x_{observed})$ by integrating the joint probability over the area determined by the observed dimensions [Ahmad and Tresp, 1993].

$$\begin{aligned} P(C_i|x_{observed}) &= \frac{p(C_i, x_{observed})}{p(x_{observed})} \\ &= \frac{\int p(C_i, x_{observed}, x_{missing}) dx_{missing}}{p(x_{observed})} \\ &= \frac{\int P(C_i|x_{observed}, x_{missing}) p(x_{observed}, x_{missing}) dx_{missing}}{p(x_{observed})} \end{aligned} \tag{5.3}$$

To do this properly in a network requires either an approximation to the distribution shapes (e.g. assuming a mixture of gaussians) or a lot of memory storage. Ahmad and Tresp show that an approximation can be obtained without requiring integration using Gaussian basis functions. However there is still significant storage involved in storing the parameters of the basis functions.

An efficient approximation to the above missing features classification problem can be obtained with a supervised version of the M-D algorithm in which there is one LVQ2.1 network for each single dimension. Running the LVQ algorithm on each dimension trades off the distribution modeling for less memory usage while trying to retain the information about the best classification borders. Instead of modeling the joint distribution, it simply keeps track of the best boundaries. It is not necessary to fill in the missing information if there is a decision regarding each feature individually. We don't actually need one network per dimension but could have several networks spanning different combinations of dimensions. The networks whose dimensions were all given would be available for determining the class.

For instance consider the example in Figure 5.7. The Bayesian method used in [Ahmad and Tresp, 1993] involves calculating for each y_0 the projection of the gaussian basis functions onto the missing dimension and evaluating the resulting sums which will approximate $P(C_i|y = y_0)$.

The 2-network LVQ algorithm only keeps track of the border positions for each dimension. It has in a sense already worked out the integral and determined the cross-over points at which $P(C_1|y) = P(C_2|y)$ and $P(C_2|y) = P(C_3|y)$. For each pattern y_0 it simply sees into which partition it falls.

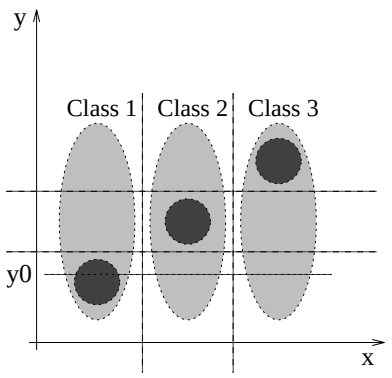

Figure 5.7: **Example classification problem after [Ahmad and Tresp, 1993].** Each oval represents the pattern distribution for a class in the 2-Dimensional space. The shading represents pattern density. Darker areas indicate higher probability regions.

The Minimizing-Disagreement algorithm is an approximation to the supervised case in the case of missing outputs. Thus, the Minimizing-Disagreement algorithm with the addition of making use of the labels when available can be considered as an efficient algorithm for handling missing data (both input and output) during both training and testing.

6 Conclusions

6.1 Summary of Results

Chapter 2 showed that there are many problems that are better classified when the labels of the data points are available during training and that even better classifiers can be developed if the network attempts to minimize the number of misclassifications.

In Chapter 3 we saw that we could approach the performance of the minimizing misclassifications classifier without requiring labeled input patterns. The labels were replaced by a requirement that the world present patterns to different modalities in such a way that patterns from Modality 1's Class A occur with patterns from Modality 2's class A distribution. By iterating the algorithm, the labeling algorithm is able to take advantage of the better codebook vector placement and produce better results allowing the Minimizing-Disagreement algorithm to perform even better. With this two iteration algorithm the performance on the benchmark dataset was within 5% of the analogous supervised algorithm. ncd16.ai:We also saw that the results were better when the confusable classes were different for the two modalities as they could provide better labeling where it was needed most. In this case the performance was within 2% of the supervised algorithm. The experiments on the visual-auditory speech data showed that the algorithm works on higher dimensional "real" patterns and the single-speaker experiments demonstrated the ability of the algorithm to work with co-occurrences present in natural cross-modal signals.

6.2 Discussion

The M-D algorithm was described as an algorithm to minimize the disagreement between the output of two sensory modalities but the basic idea can be viewed more generally. Ideally the algorithm could be applied hierarchically with lower levels minimizing disagreement between submodalities such as colour and form. For example as the lighting changes, the colour of an object may change while its form may stay relatively constant. On the other hand, while moving around the form will change but the colour will stay relatively constant. The M-D algorithm could also be applied hierarchically in a spatial sense. Low-level processing could minimize disagreement between small areas while high-level processing would be minimizing disagreement over larger areas.

Also the algorithm can be generalized to apply to patterns near in time. The two networks would represent delayed versions of the same input stream. The signal to the other "modality" could be a temporally close sample from the same modality. As sensations change slowly over time, the network could learn to classify objects by minimizing the disagreements between outputs for input patterns received close in time. This approach is more powerful than that of [Földiák, 1991] as signals close in time need not be mapped to the *same* codebook vector but the closest codebook vector of the *same class*.

6.3 Contributions

Many have argued that the reliable relationships between the sensations to different modalities may be an important source of "teaching" information; in this dissertation we have embodied the argument in a powerful working model. We have developed an abstract algorithm for making use of the cross-modal information, examined its properties mathematically and demonstrated its performance empirically on real data. Finally, we have given suggestions about how this might be implemented in the brain which agrees nicely with some physiological results concerning the relationship between long term potentiation and long term depression in visual cortical cells [Artola *et al.*, 1990].

The algorithm is similar in general motivation to that of [Becker and Hinton, 1992; Becker, 1993], but differs in the relative emphasis on biological plausibility and generality. This results in significantly different derivations and resulting algorithms. While this algorithm is restricted to classification tasks, it performs better at them and it is easier to imagine how it might be implemented in the brain.

From a biological point of view this dissertation offers an explanation for why cells in one sensory area also respond to inputs to another sensory modality. We have shown that without connecting neurons to all sensory input we can still take advantage of the greater structure available in the higher dimensional total space of inputs. This occurs through integrating the modalities at a higher level and using feedback connections. The work provides an explanation for the ubiquitous back projections in cortex whose purpose is not yet well understood. In fact [Sandell and Schiller, 1982] found that inactivating V2 had little effect on the response properties of cells in V1 indicating that back-projections, or at least the V2-V1 projection has little purpose during regular processing. This dissertation suggests that these back-projections are important for the organization of the incoming sensory stimuli during learning.

From a machine learning point of view, the contribution of this dissertation is a better understanding of the properties of some of the piecewise-linear classification algorithms and the development of a self-supervised algorithm for classification. This algorithm enables trainers to avoid the costly hand-labeling of training data, provided that the learning system has access to information from two or more modalities, submodalities, or points in time, that are providing redundant but not identical information. At a minimum it would allow the learning system to train on patterns labeled verbally by an observer.

6.4 Future Work

While the Minimizing-Disagreement stage of the algorithm is mathematically well motivated, the initial labeling algorithm could be more carefully considered.

Another possible avenue for exploration involves the addition of terms to explicitly prevent the trivial solution of uniform agreement (analogous to the individual entropy terms $H(Y_1), H(Y_2)$ in Becker's IMAX algorithm). Preliminary experiments using terms like this with the Peterson-Barney vowel dataset yielded slightly worse performance, likely because the addition of the terms changes the position of the local minimum (It is no longer minimizing the disagreement). However it is possible that for problems with more overlap between classes, terms like this might help. If not, this also merits more study to determine exactly how the Minimizing-Disagreement algorithm copes with finding a weak or non-existent local minimum. A closer study of the interaction of the competitive learning of the output weights would greatly aid the understanding.

In order to more closely model cortical processing the algorithm should be extended to work in multi-layer networks where neurons within each layer have limited connectivity from neurons in the previous layer. The teaching information from the other modality would also feedback through the layers. With this closer match to the neurobiology we may be able to predict receptive fields in higher areas of the temporal visual pathway as the models of V1 do for striate cortex.

We are also interested in searching for optimal ways to partition the processing from many sensory inputs. Given a vector of sensations, there are many ways in which it could be partitioned so that one set of dimensions is processed by one network and the rest by another. Introducing the ability to have more than two "modality" networks increases the options further. Intuitively it seems that dimensions that are dependent would benefit from being processed together due to the ability to form more complex boundaries in the space. On the other hand those dimensions that are independent within a class could benefit from the advantages of applying the Minimizing-Disagreement algorithm to two or more networks. Work along these lines may give insight into why particular submodalities fall along the dimensions they do. I'd like to thank Randal Nelson and Mary Hayhoe for suggesting this idea.

Another way to extend this work is to develop more general algorithms that make use of natural teaching information. Other problems in which there is natural teaching information are learning to speak and learning general sensory-motor coordination. In these tasks, the organism's goal is to produce correct motor neuron signals to obtain desired output trajectories. The problem is to determine appropriate future input signals given only the errors in the output space. Algorithms that can intelligently select appropriate control signals in a biologically plausible way may be useful in discovering more about motor control or general sensory-motor coordination.

Bibliography

[Ahmad and Tresp, 1993] Subutai Ahmad and Volker Tresp, "Some Solutions to the Missing Feature Problem in Vision," in C.L. Giles, S.J.Hanson, and J.D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 393—400. Morgan Kaufmann, 1993.

[Artola *et al.*, 1990] A. Artola, S. Bröcher, and W. Singer, "Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex," *Nature*, 347:69–72, September 1990.

[Becker and Hinton, 1992] S. Becker and G. E. Hinton, "A self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, 355:161–163, January 1992.

[Becker, 1992] Suzanna Becker, *An Information-theoretic Unsupervised Learning Algorithm for Neural Networks*, PhD thesis, Department of Computer Science, University of Toronto, 1992.

[Becker, 1993] Suzanna Becker, "Learning to categorize objects using temporal coherence," in C.L. Giles, S.J.Hanson, and J.D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 361—368. Morgan Kaufmann, 1993.

[Bridle, 1990] J.S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems 2*, pages 111–217. Morgan Kaufmann, 1990.

[Buser and Borenstein, 1959] P. Buser and P. Borenstein, "Responses somesthesiques, visuel et auditives, recuellies, au niveau du cortex "associatif" infrasylvien chez le chat curarise non anesthesie," *Electroencephalog. Clin. Neurophysiol.*, 11:285–304, 1959.

[Coultrip *et al.*, 1992] R. Coultrip, R. Granger, and G. Lynch, "A Cortical Model of Winner-Take-All Competition Via Lateral Inhibition," *Neural Networks*, 5:47–54, 1992.

[de Sa and Ballard, 1992] Virginia de Sa and Dana Ballard, "Top-down teaching enables task-relevant classification with competitive learning," in *IJCNN International Joint Conference on Neural Networks*, volume 3, pages III–364—III–371, 1992.

[de Sa and Ballard, 1991] Virginia R. de Sa and Dana H. Ballard, "Top-down Teaching Enables Non-trivial Clustering via Competitive Learning," Technical Report 402, Department of Computer Science, University of Rochester, Rochester, NY 14627-0226, November 1991.

[de Sa and Ballard, 1993] Virginia R. de Sa and Dana H. Ballard, "a note on learning vector quantization," in C.L. Giles, S.J.Hanson, and J.D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 220—227. Morgan Kaufmann, 1993.

[Desimone and Ungerleider, 1989] Robert Desimone and Leslie G. Ungerleider, "Neural Mechanisms of visual processing in monkeys," in F. Boller and J. Grafman, editors, *Handbook of Neuropsychology*, volume 2, chapter 14, pages 267–299. Elsevier Science Publishers B. V., 1989.

[Durgin, 1995] Frank H. Durgin, *Contingent aftereffects of texture density: Perceptual learning and contingency*, PhD thesis, Department of Psychology, University of Virginia, 1995.

[Fanty, 1988] Mark Alan Fanty, *Learning in Structured Connectionist Networks*, PhD thesis, University of Rochester, Dept. of Computer Science, 1988. (Also available as UR CS TR 252).

[Fishman and Michael, 1973] Mark C. Fishman and Charles R. Michael, "Integration of Auditory Information in the Cat's Visual Cortex," *Vision Research*, 13:1415–1419, 1973.

[Földiák, 1991] Peter Földiák, "Learning Invariance from Transformation Sequences," *Neural Computation*, 3(2):194–200, 1991.

[Geva and Sitte, 1991] Shlomo Geva and Joaquin Sitte, "Adaptive Nearest Neighbor Pattern Classification," *IEEE Transactions on Neural Networks*, 2(2):318—322, March 1991.

[Grossberg, 1976a] Stephen Grossberg, "Adaptive Pattern Classification and Universal Recoding: I. Parallel Development and Coding of Neural Feature Detectors," *Biological Cybernetics*, 23:121–134, 1976.

[Grossberg, 1976b] Stephen Grossberg, "Adaptive Pattern Classification and Universal Recoding: II. Feedback, Expectation, Olfaction, Illusions," *Biological Cybernetics*, 23:187–202, 1976.

[Hebb, 1949] D. O. Hebb, *The Organization of Behavior*, Wiley, 1949.

[Hertz *et al.*, 1991] John Hertz, Anders Krogh, and Richard G. Palmer, *Introduction to the Theory of Neural Computation*, volume 1 of *Santa Fe Institute Studies in the Sciences of Complexity Lecture Notes*, Addison-Wesley, 1991.

[Kandel and Schwartz, 1985] Eric R. Kandel and James H. Schwartz, editors, *Principles of Neural Science*, Elsevier, second edition, 1985.

[Knoll and Lo, 1992] Travis Dean Knoll and James Ting-Ho Lo, "Push-and-Pull for Piecewise Linear Machine Training," in *IJCNN International Joint Conference on Neural Networks*, volume 3, pages III–573—III–578, 1992.

[Kohonen, 1982] Teuvo Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, 43:59–69, 1982.

[Kohonen, 1986] Teuvo Kohonen, "Learning Vector Quantization for Pattern Recognition," Technical Report TKK-F-A601, Helsinki University of Technology, Department of Technical Physics, Laboratory of Computer and Information Science, November 1986.

[Kohonen, 1990a] Teuvo Kohonen, "Improved Versions of Learning Vector Quantization," in *IJCNN International Joint Conference on Neural Networks*, volume 1, pages I–545–I–550, 1990.

[Kohonen, 1990b] Teuvo Kohonen, "Statistical Pattern Recognition Revisited," in R. Eckmiller, editor, *Advanced Neural Computers*, pages 137–144. Elsevier Science Publishers, 1990.

[Kohonen *et al.*, 1988] Teuvo Kohonen, György Barna, and Ronald Chrisley, "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies," in *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88*, volume 1, pages I–61—I–68, 1988.

[Kuhl and Meltzoff, 1984] Patricia K. Kuhl and Andrew N. Meltzoff, "The Intermodal Representation of Speech in Infants," *Infant Behavior and Development*, 7:361—381, 1984.

[Lehky and Sejnowski, 1988] S. Lehky and T.J. Sejnowski, "Network model of shape-from-shading: Neural function arises from both receptive and projective fields," *Nature*, 333:452–454, 1988.

[Linsker, 1986b] Ralph Linsker, "From basic network principles to neural architecture: Emergence of orientation-selective cells," *Proc. Natl. Acad. Sci. USA*, 83:8390–8394, November 1986.

[Linsker, 1986c] Ralph Linsker, "From basic network principles to neural architecture: Emergence of orientation columns," *Proc. Natl. Acad. Sci. USA*, 83:8779–8783, November 1986.

[Linsker, 1986a] Ralph Linsker, "From basic network principles to neural architecture: Emergence of spatial-opponent cells," *Proc. Natl. Acad. Sci. USA*, 83:7508–7512, October 1986.

[MacDonald and McGurk, 1978] J. MacDonald and H. McGurk, "Visual influences on speech perception processes," *Perception & Psychophysics*, 24(3):253–257, 1978.

[McCulloch and Pitts, 1943] W. S. McCulloch and W. Pitts, "A Logical Calculus of Ideas Imminent in Nervous Activity," *Bulletin of Mathematical Biophysics*, 5:115—133, 1943.

[McGurk and MacDonald, 1976] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature*, 264:746–748, December 1976.

[Merigan and Maunsell, 1993] W. H. Merigan and J. H. R. Maunsell, "How parallel are the primate visual pathways?," in *Annual Review of Neuroscience*, volume 16, pages 369–402. 1993.

[Meulders *et al.*, 1965] M. Meulders, J. Colle, and N. Biosacq-Schepens, "Macro and microelectrode studies of somatic responses in the lateral geniculate body," in *Proceedings, XXIII International Congress of Physiological Sciences*, page 364, Tokyo, 1965.

[Miikkulainen, 1991] Risto Miikkulainen, "Self-Organizing Process based on Lateral Inhibition and Synaptic Resource Redistribution," in T. Kohonen, K. Makisära, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 415–420. Elsevier Science Publishers, 1991.

[Miller and Nicely, 1955] George A. Miller and Patricia E. Nicely, "An Analysis of Perceptual Confusions Among some English Consonants," *The Journal of the Acoustical Society of America*, 27(2):338–352, March 1955.

[Miller *et al.*, 1989] Kenneth D. Miller, Joseph B. Keller, and Michael P. Stryker, "Ocular Dominance Column Development: Analysis and Simulation," *Science*, 245:605–615, August 1989.

[Morrell, 1972] Frank Morrell, "Visual System's View of Acoustic Space," *Nature*, 238:44–46, July 1972.

[Murata *et al.*, 1965] K. Murata, H. Cramer, and P. Bach y Rita, "Neuronal convergence of noxious, acoustic and visual stimuli in the visual cortex of the cat," *Journal of Neurophysiology*, 28:1233–1239, 1965.

[Nowlan, 1991] Steven J. Nowlan, *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*, PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.

[Obermayer *et al.*, 1990] K. Obermayer, H. Ritter, and K. Schulten, "A principle for the formation of the spatial structure of cortical feature maps," *Proc. Natl. Acad. Sci.*, 87:8345–8349, November 1990.

[Obermayer *et al.*, 1992] K. Obermayer, K. Schulten, and G.G. Blasdel, "A comparison between a neural network model for the formation of brain maps and experimental data," in *Advances in Neural Information Processing Systems 4*, pages 83–90. Morgan Kaufmann, 1992.

[Parzen, 1962] E. Parzen, "On estimation of a probability density function and mode," *Annals of Math. Stat.*, 33:1065—1076, 1962.

[Polana, 1994] Ramprasad Polana, *Temporal Texture and Activity Recognition*, PhD thesis, Department of Computer Science, University of Rochester, 1994.

[Ritter and Schulten, 1988] Helge Ritter and Klaus Schulten, "Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities," in *IEEE International Conference on Neural Networks*, volume 1, pages 109–116, 1988.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro, "A Stochastic Approximation Method," *Annals of Math. Stat.*, 22:400—407, 1951.

[Rolls, 1989] Edmund Rolls, "The Representation and Storage of Information in Neuronal Networks in the Primate Cerebral Cortex and Hippocampus," in Durbin, Miall, and Mitchison, editors, *The Computing Neuron*, chapter 8, pages 125–159. Addison-Wesley, 1989.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–364. MIT Press, 1986.

[Rumelhart and Zipser, 1986] D. E. Rumelhart and D. Zipser, "Feature Discovery by Competitive Learning," in David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2, pages 151–193. MIT Press, 1986.

[Sams *et al.*, 1991] Mikko Sams, Reijo Aulanko, Matti Hämääinen, Riitta Hari, Olli V. Lounasmaa, Sing-Teh Lu, and Juha Simola, "Seeing speech: visual information from lip movements modifies activity in the human auditory cortex," *Neuroscience Letters*, 127:141–145, 1991.

[Sandell and Schiller, 1982] Julie H. Sandell and Peter H. Schiller, "Effects of Cooling Area 18 on Striate Cortex Cells in Squirrel Monkey," *Journal of Neurophysiology*, 48(1):38—48, July 1982.

[Sejnowski, 1977] T.J. Sejnowski, "Storing Covariance with nonlinearly interacting neurons," *Journal of Mathematical Biology*, 4:203–211, 1977.

[Sklansky and Wassel, 1981] Jack Sklansky and Gustav N. Wassel, *Pattern Classifiers and Trainable Machines*, Springer-Verlag, 1981.

[Spinelli, 1967] D.N. Spinelli, "Receptive field organization of ganglion cells in the cat's retina," *Experimental Neurology*, 19:291–315, 1967.

[Spinelli *et al.*, 1965] D.N. Spinelli, K.H. Pribram, and M. Weingarten, "Centrifugal optic nerve responses evoked by auditory and somatic stimulation," *Experimental Neurology*, 12:303–319, 1965.

[Spinelli *et al.*, 1968] D.N. Spinelli, Arnold Starr, and Terence W. Barrett, "Auditory Specificity in Unit Recordings from Cat's Visual Cortex," *Experimental Neurology*, 22:75–84, 1968.

[Spinelli and Weingarten, 1966] D.N. Spinelli and M. Weingarten, "Afferent and efferent activity in single units of the cat's optic nerve," *Experimental Neurology*, 15:347–362, 1966.

[Stent, 1973] G. S. Stent, "A physiological mechanism for Hebb's postulate of learning," *Proceedings of the National Academy of Science*, 70(4):997–1001, 1973.

[Sumby and Pollack, 1954] W.H. Sumby and Irwin Pollack, "Visual Contribution to Speech Intelligibility in Noise," *The journal of the acoustical society of america*, 26(2):212–215, March 1954.

[Wassel and Sklansky, 1972] Gustav N. Wassel and Jack Sklansky, "Training a One-Dimensional Classifier to Minimize the Probability of Error," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(4):533—541, 1972.

[Werbos, 1974] P.J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD thesis, Harvard University, 1974.

[Zipser and Andersen, 1988] D. Zipser and R.A. Andersen, "A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons," *Nature*, 331:679–684, 1988.

A Discussion of Rumelhart & Zipser's Supervised Competitive Learning

As was shown in Chapter 2, segregating the horizontal from vertical lines requires more than 2 competitive neurons and thus an extra layer of neurons is needed between the input and 2 neuron ouput layer. By giving these hidden neurons access to a teaching layer they were able to segregate appropriately. The hidden layer however introduces another problem in that the output layer is not connected to the teaching input and thus has no guidance on how to cluster the hidden units. It has no information to group the two horizontal neurons apart from the two vertical neurons. The output neurons at any one time see only a vector with one 1 and three 0's in it. These patterns are all equidistant providing no information for appropriate grouping.

Rumelhart and Zipser addressed this problem by duplicating the hidden layer neurons to form two clusters and allowing competitition only within each cluster as shown in Figure A.1. The idea of the duplication is to introduce redundancy to provide extra information. The solution depends on the probability that the two sets will not divide the patterns up identically. In this case different members from the same class will activate different combinations of the neurons representing that class (1 from each group). The important point is that the occurrence of activation of all neurons from one class will have overlap with each other but not with the neurons from the other class. Consider a case of two output classes (for horizontal and vertical lines) and 4 neurons in each hidden-layer group (each representing 2 of one kind of line). Each of the horizontal lines will then have hidden layer activations that differ in 0,2, or 4 bits from activations of all other horizontal lines but always in 4 bits from all the vertical lines. Thus the patterns of activation caused by the horizontal lines will be grouped separately from those caused by the vertical lines allowing the output neurons to segregate horizontal from vertical lines.

We can compare the number of neurons and connections required with this solution to that in [de Sa and Ballard, 1991; de Sa and Ballard, 1992], discussed in Section 2.2 and shown in Figure 2.2 duplicated as Figure A.2.

If we let I, H, O, and T represent the number of input layer, hidden layer, output layer and teaching neurons respectively, the number of connections in the Rumelhart and Zipser model is

$$[(I+T)\cdot 2H]+[2H\cdot O]$$

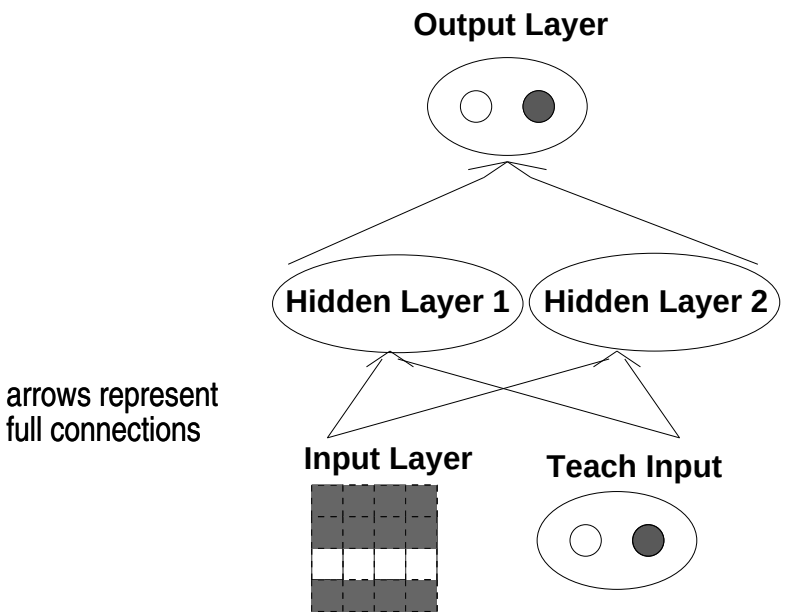

Figure A.1: **Supervised competitive learning architecture proposed in [Rumelhart and Zipser, 1986].**

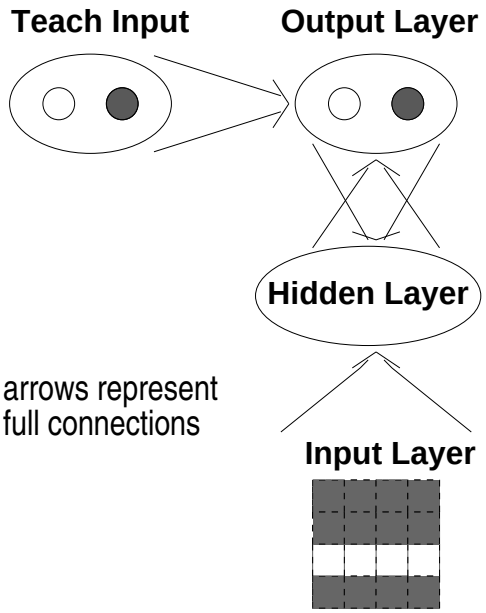

Figure A.2: **Our supervised competitive learning architecture.**

whereas that of [de Sa and Ballard, 1991; de Sa and Ballard, 1992] is

$$(I \cdot H) + [2 \cdot (H \cdot O)] + (T \cdot O)$$

for a connectivity savings of

$$I \cdot H + T(2H - O)$$

Note that since $H > O$, there is always a connectivity saving.

B Description of the Window Training Procedure from [Wassel & Sklansky, 1972]

The window training procedure [Wassel and Sklansky, 1972] is a stochastic approximation algorithm designed to minimize the number of misclassifications in a one-dimensional two pattern classifier [1].

Defining Class B as the one with larger values on the x axis, the function to be minimized is (restating Equation 2.2)

$$E(b) = \int_{-\infty}^{b} P(C_B)p(x|C_B)dx + \int_{b}^{\infty} P(C_A)p(x|C_A)dx. \tag{B.1}$$

and its gradient is given by

$$dE/db = h(b) \tag{B.2}$$

where

$$h(x) = P(C_B)p(x|C_B) - P(C_A)p(x|C_A) \tag{B.3}$$

The idea of the algorithm is to perform stochastic approximation on the Energy function. As discussed in Section 1.3.2, stochastic approximation can be thought of as noisy gradient descent. More specifically, in stochastic approximation iterative approximations are made by adding to the current estimate an appropriately scaled random variable whose expected value is the gradient, at the current estimate, of the function to be minimized.

The problem is that Equation B.2 is not expressible as an expectation of a random variable dependent on the received input pattern. Thus the bulk of [Wassel and Sklansky, 1972] extends the stochastic approximation method to work for functions that are the limit of a sequence of such, *regression* functions. Then the solution to the minimization of (B.1) involves finding a sequence of functions whose limit is given by (B.2) the gradient of E. We describe this part below. (For the proof of the former see [Wassel and Sklansky, 1972].)

They let h(x) be as given above in (B.3) and consider the function

[1] The algorithm has been extended to work in higher dimensions in [Sklansky and Wassel, 1981].

$$\hat{g}(b,c) = \int_{-\infty}^{\infty} \Psi(x-b,c)h(x)dx \tag{B.4}$$

where $\Psi(x-z,c)$ is a Parzen window function [Parzen, 1962] centred at z with width parameter c that satisfies the following conditions

$$\Psi(x-z,c) \geq 0 \quad \forall x,z \quad c>0$$

$$\int_{-\infty}^{\infty} \Psi(x-z,c)dx = 1 \quad \forall z \quad c>0$$

$$c\int_{-\infty}^{\infty} \Psi^2(x-z,c)dx < \infty \quad \forall z \quad c>0$$

$$\lim_{c\to 0} \Psi(x-z,c) = \delta(x-z) \quad \forall x,z \quad c>0$$

then

$$\begin{aligned} \lim_{c\to 0} \hat{g}(b,c) &= h(b) \\ &= dE/db \end{aligned} \tag{B.5}$$

They then show that $\hat{g}(b,c)$ is a regression function by showing that there is a random variable Z such that $\frac{1}{2c}E(Z|B=b) = -\hat{g}(b,c)$

Letting

$$Z_i = 2cS(X(i))\Psi(X(i)-B,c)) \tag{B.6}$$

where $X(i)$ is the sample data point at time i and $S(X(i))$ is $+1$ if $X(i) \in C_A$ and -1 if $X(i) \in C_B$.

gives

$$\begin{aligned} \frac{1}{2c}E(Z|B=b) &= E(S(X)\Psi(X-b,c)) \\ &= \int_{-\infty}^{\infty} (P(C_A)p(x|C_A) - P(C_B)p(x|C_B))\Psi(x-b,c)dx \\ &= -\hat{g}(b,c) \end{aligned}$$

Thus (B.2) is the limit of a sequence of regression functions $(\hat{g}(b,c))$. Then by extending the stochastic approximation method, Wassel and Sklansky prove that the following training procedure converges to the unique optimal border.

$$b(n+1) = b(n) + \epsilon(n)Z_n(X(n),b(n),c(n)) \tag{B.7}$$

where

$$Z_n = \begin{cases} 2c(n)\Psi(X_1(n)-b_1(n),c(n)) & \text{for } X(n) \in C_A \\ -2c(n)\Psi(X_1(n)-b_1(n),c(n)) & \text{for } X(n) \in C_B \end{cases}$$

Using rectangular Parzen window functions, the above simplifies to

$$Z_n = \begin{cases} 1 & \text{for } X(n) \in C_A,\ |X_1(n) - b(n)| < c(n) \\ -1 & \text{for } X(n) \in C_B,\ |X_1(n) - b(n)| < c(n) \end{cases}$$

The proof requires that the distributions $P(C_A)p(x|C_A), P(C_B)p(x|C_B)$: remain finite, have only one crossing point, and that $P(C_A)p(x|C_A > P(C_B)p(x|C_B)$ to the left of the crossing point and $P(C_A)p(x|C_A < P(C_B)p(x|C_B)$ to the right of the crossing point [2] as well as the the following standard convergence restrictions: $\epsilon(n), c(n) > 0$; $\lim_{n\to\infty} \epsilon(n) = 0$; $\lim_{n\to\infty} c(n) = 0$; $\sum_{n=1}^{\infty} \epsilon(n)c(n) = \infty$; and $\sum_{n=1}^{\infty} \epsilon(n)^2 c(n) < \infty$.

[2] This condition is sufficient for the rectangular Parzen window functions used above. See [Wassel and Sklansky, 1972] for the more general condition for any Parzen window functions.

C Derivation of the Minimizing Disagreement Algorithm

The algorithm we derive for minimizing (3.4) is patterned after [Wassel and Sklansky, 1972] and uses their extension to the stochastic approximation method [Robbins and Monro, 1951]. The Wassel and Sklansky [1972] paper developed a method for iteratively minimizing a function whose derivatives can not be expressed as regression functions but that can be expressed as the limits of sequences of regression functions. Thus the solution to the minimization of (3.4) involves finding a sequence of functions whose limit is given by (3.6)(and similarly for (3.7)).

Consider the function

$$\hat{g}(b_1, b_2, c) = \int_{-\infty}^{\infty} \Psi(x_1 - b_1, c) \left[\int_{b_2}^{\infty} f(x_1, x_2) dx_2 - \int_{-\infty}^{b_2} f(x_1, x_2) dx_2 \right] dx_1 \qquad \text{(C.1)}$$

where $\Psi(x-z, c)$ is a Parzen window function [Parzen, 1962] centred at z with width parameter c that satisfies the following conditions

$$\Psi(x - z, c) \geq 0 \quad \forall x, z \quad c > 0$$

$$\int_{-\infty}^{\infty} \Psi(x - z, c) dx = 1 \quad \forall z \quad c > 0$$

$$c \int_{-\infty}^{\infty} \Psi^2(x - z, c) dx < \infty \quad \forall z \quad c > 0$$

$$\lim_{c \to 0} \Psi(x - z, c) = \delta(x - z) \quad \forall x, z \quad c > 0$$

then

$$\begin{aligned} \lim_{c \to 0} \hat{g}(b_1, b_2, c) &= \int_{b_2}^{\infty} f(b_1, x_2) dx_2 - \int_{-\infty}^{b_2} f(b_1, x_2) dx_2 \qquad \text{(C.2)} \\ &= \partial E / \partial b_1 \end{aligned}$$

We now show that $\hat{g}$ is a regression function by showing that there is a random variable Z such that $\frac{1}{2c} E(Z | B_1 = b_1) = -\hat{g}(b_1, b_2, c)$

Let

$$Z_i = 2cS(X_2(i))\Psi(X_1(i) - B_1, c)) \tag{C.3}$$

where $X_j(i)$ is the sample data point at time i presented to modality j, and $S(X_2(i))$ is $+1$ if $X_2(i) < b_2$ and -1 if $X_2(i) > b_2$.

Then,

$$\begin{aligned}
\frac{1}{2c}E(Z|B_1 = b_1) &= E(S(X_2)\Psi(X_1 - b_1, c)) \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_1, x_2)S(x_2)\Psi(x_1 - b_1, c)dx_1dx_2 \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{b_2} f(x_1, x_2)\Psi(x_1 - b_1, c)dx_1 - \int_{-\infty}^{\infty}\int_{b_2}^{\infty} f(x_1, x_2)\Psi(x_1 - b_1, c)dx_1 \\
&= -\hat{g}(b_1, b_2, c)
\end{aligned}$$

Thus (3.6) is the limit of a sequence of regression functions and by the Theorem in [Wassel and Sklansky, 1972] the position of b_1 can be iteratively calculated according to

$$b_1(n+1) = b_1(n) + \alpha(n)Z_n(X_1(n), X_2(n), b_1(n), b_2(n), c(n)) \tag{C.4}$$

where

$$Z_n = \begin{cases} 2c(n)\Psi(X_1(n) - b_1(n), c(n)) & \text{for } X_2(n) < b_2(n) \\ -2c(n)\Psi(X_1(n) - b_1(n), c(n)) & \text{for } X_2(n) > b_2(n) \end{cases}$$

Using rectangular Parzen window functions, the above simplifies to

$$Z_n = \begin{cases} 1 & \text{for } X_2(n) < b_2(n), |X_1(n) - b_1(n)| < c(n) \\ -1 & \text{for } X_2(n) > b_2(n), |X_1(n) - b_1(n)| < c(n) \end{cases}$$

The theorem in [Wassel and Sklansky, 1972] guarantees convergence under the condition that the function to be minimized has a single, minimum; as we are searching for a local minimum, convergence is only guaranteed if (b_1, b_2) stay within the area in which (b_1^*, b_2^*) is a local minimum and the only local extremum. The following conditions are also required : $\alpha(n), c(n) > 0$; $\lim_{n\to\infty}\alpha(n) = 0$; $\lim_{n\to\infty} c(n) = 0$; $\sum_{n=1}^{\infty}\alpha(n)c(n) = \infty$; $\sum_{n=1}^{\infty}\alpha(n)^2c(n) < \infty$; and $p(x_i|C_j) < \infty$, $i = 1, 2; j = A, B$.