Linköping Studies in Science and Technology. Dissertations
No. 531

Learning Multidimensional Signal Processing

Magnus Borga

Department of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden

Linköping 1998

Learning Multidimensional Signal Processing

Department of Electrical Engineering
Linköping University
S-581 83 Linköping
Sweden

ISBN 91-7219-202-X ISSN 0345-7524

Abstract

The subject of this dissertation is to show how learning can be used for multidimensional signal processing, in particular computer vision. Learning is a wide concept, but it can generally be defined as a system's change of behaviour in order to improve its performance in some sense.

Learning systems can be divided into three classes: supervised learning, reinforcement learning and unsupervised learning. Supervised learning requires a set of training data with correct answers and can be seen as a kind of function approximation. A reinforcement learning system does not require a set of answers. It learns by maximizing a scalar feedback signal indicating the system's performance. Unsupervised learning can be seen as a way of finding a good representation of the input signals according to a given criterion.

In learning and signal processing, the choice of signal representation is a central issue. For high-dimensional signals, dimensionality reduction is often necessary. It is then important not to discard useful information. For this reason, learning methods based on maximizing mutual information are particularly interesting.

A properly chosen data representation allows local linear models to be used in learning systems. Such models have the advantage of having a small number of parameters and can for this reason be estimated by using relatively few samples. An interesting method that can be used to estimate local linear models is canonical correlation analysis (CCA). CCA is strongly related to mutual information. The relation between CCA and three other linear methods is discussed. These methods are principal component analysis (PCA), partial least squares (PLS) and multivariate linear regression (MLR). An iterative method for CCA, PCA, PLS and MLR, in particular low-rank versions of these methods, is presented.

A novel method for learning filters for multidimensional signal processing using CCA is presented. By showing the system signals in pairs, the filters can be adapted to detect certain features and to be invariant to others. A new method for local orientation estimation has been developed using this principle. This method is significantly less sensitive to noise than previously used methods.

Finally, a novel stereo algorithm is presented. This algorithm uses CCA and phase analysis to detect the disparity in stereo images. The algorithm adapts filters in each local neighbourhood of the image in a way which maximizes the correlation between the filtered images. The adapted filters are then analysed to find the disparity. This is done by a simple phase analysis of the scalar product of the filters. The algorithm can even handle cases where the images have different scales. The algorithm can also handle depth discontinuities and give multiple depth estimates for semi-transparent images.

To Maria

Acknowledgements

This thesis is the result of many years work and it would never have been possible for me to accomplish this without the help, support and encouragements from a lot of people.

First of all, I would like to thank my supervisor, associate professor Hans Knutsson. His enthusiastic engagement in my research and his never ending stream of ideas has been absolutely essential for the results presented here. I am very grateful that he has spent so much time with me discussing different problems ranging from philosophical issues down to minute technical details.

I would also like to thank professor Gösta Granlund for giving me the opportunity to work in his research group and for managing a laboratory it is a pleasure to work in.

Many thanks to present and past members of the Computer Vision Laboratory for being good friends as well as helpful colleagues.

In particular, I would like to thank Dr. Tomas Landelius with whom I have been working very close in most of the research presented here as well as in the (not yet finished) systematic search for the optimum malt whisky. His comments on large parts of the early versions of the manuscript have been very valuable.

I would also like to thank Morgan Ulvklo and Dr. Mats Andersson for constructive comments on parts of the manuscript. Dr. Mats Anderson's help with a lot of technical details ranging from the design of quadrature filters to welding is also very appreciated.

Finally, I would like to thank my wife Maria for her love, support and patience. Maria should also have great credit for proof-reading my manuscript and helping me with the English. All remaining errors are to be blamed on me, due to final changes.

The research presented in this thesis was sponsored by NUTEK (Swedish National Board for Industrial and Technical Development) and TFR (Swedish Research Council for Engineering Sciences), which is gratefully acknowledged.

Contents

Chapter 1

Introduction

This thesis deals with two research areas: learning and multidimensional signal processing. A typical example of a multidimensional signal is an image. An image is usually described in terms of pixel[1] values. A monochrome TV image has a resolution of approximately 700×500 pixels, which means that it is a 350,000-dimensional signal. In computer vision, we try to instruct a computer how to extract the relevant information from this huge signal in order to solve a certain task. This is not an easy problem! The information is extracted by estimating certain local features in the image. What is "relevant information" depends, of course, on the task. To describe what features to estimate and how to estimate them are possible only for highly specific tasks, which, for a human, seem to be trivial in most cases. For more general tasks, we can only define these feature detectors on a very low level, such as line and edge detectors. It is commonly accepted that it is difficult to design higher-level feature detectors. In fact, the difficulty arises already when trying to define what features are important to estimate.

Nature has solved this problem by making the visual system adaptive. In other words, we learn how to see. We know that many of the low-level feature detectors used in computer vision are similar to those found in the mammalian visual system (Pollen and Ronner, 1983). Since we generally do not know how to handle multidimensional signals on a high level and since our solutions on a low level are similar to those of nature, it seems rational also on a higher level to use nature's solution: learning.

Learning in artificial systems is often associated with artificial neural networks. Note, however, that the term "neural network" refers to a specific type of architecture. In this work we are more interested in the learning capabilities than the hardware implementation. What we mean by "learning systems" is discussed in the next chapter.

[1] Pixel is an abbreviation for Picture Element.

The learning process can be seen as a way of finding adaptive models to represent relevant parts of the signal. We believe that local low-dimensional linear models are sufficient and efficient for representation in many systems. The reason for this is that most real-world signals are (at least piecewise) continuous due to the dynamic of the world that generates them. Therefore it can be justified to look at some criteria for choosing low-dimensional linear models.

In the field of signal processing there seems to be a growing interest in methods related to independent component analysis. In the learning and neural network society, methods based on maximizing mutual information are receiving more attention. These two methods are related to each other and they are also related to a statistical method called canonical correlation analysis, which can be seen as a linear special case of maximum mutual information. Canonical correlation analysis is also related to principal component analysis, partial least squares and multivariate linear regression. These four analysis methods can be seen as different choices of linear models based on different optimization criteria.

Canonical correlation turns out to be a useful tool in several computer vision problems as a new way of constructing and combining filters. Some examples of this are presented in this thesis. We believe that this approach provides a basis for new efficient methods in multidimensional signal processing in general and in computer vision in particular.

1.1 Contributions

The main contributions in this thesis are presented in chapters 3, 4, 6 and 7. Chapters 2 and 5 should be seen as introductions to learning systems and computer vision respectively. The most important individual contributions are:

- A unified framework for principal component analysis (PCA), partial least squares (PLS), canonical correlation analysis (CCA) and multiple linear regression (MRL) (chapter 4).
- An iterative gradient search algorithm that successively finds the eigenvalues and the corresponding eigenvectors to the generalized eigenproblem. The algorithm can be used for the special cases PCA, PLS, CCA and MLR (chapter 4).
- A method for using canonical correlation for learning feature detectors in high-dimensional signals (chapter 6). By this method, the system can also learn how to combine estimates in a way that is less sensitive to noise than the previously used vector averaging method.
- A stereo algorithm based on canonical correlation and phase analysis that

can find correlation between differently scaled images. The algorithm can handle depth discontinuities and estimate multiple depths in semi-transparent images (chapter 7).

The TD-algorithm presented in section 3.6.2 was presented at ICANN'93 in Amsterdam (Borga, 1993). Most of the contents in chapter 4 have been submitted for publication in *Information Sciences* (Borga et al., 1997b, revised for second review). The canonical correlation algorithm in section 4.7.3 and most of the contents in chapter 6 were presented at SCIA'97 in Lappeenranta, Finland (Borga et al., 1997a). Finally, the stereo algorithm in chapter 7 has been submitted to ICIPS'98 (Borga and Knutsson, 1998).

Large parts of chapter 2 except the section on unsupervised learning (2.5), most of chapter 3 and some of the theory of canonical correlation in chapter 4 were presented in "Reinforcement Learning Using Local Adaptive Models" (Borga, 1995, licentiate thesis) .

1.2 Outline

The thesis is divided into two parts. Part I deals with learning theory. Part II describes how the theory discussed in part I can be applied in computer vision.

In chapter 2, learning systems are discussed. Chapter 2 can be seen as an introduction and overview of this subject. Three important principles for learning are described: reinforcement learning, unsupervised learning and supervised learning.

In chapter 3, issues concerning information representation are treated. Linear models and, in particular, local linear models are discussed and two examples are presented that use linear models for reinforcement learning.

Four low-dimensional linear models are discussed in chapter 4. They are low-rank versions of principal component analysis, partial least squares, canonical correlation and multivariate linear regression. All these four methods are related to the generalized eigenproblem and the solutions can be found by maximizing a Rayleigh quotient. An iterative algorithm for solving the generalized eigenproblem in general and these four methods in particular is presented.

Chapter 5 is a short introduction to computer vision. It treats the concepts in computer vision relevant for the remaining chapters.

In chapter 6 is shown how canonical correlation can be used for learning models that represent local features in images. Experiments show how this method can be used for finding filter combinations that decrease the noise-sensitivity compared to vector averaging while maintaining spatial resolution.

In chapter 7, a novel stereo algorithm based on the method from chapter 6 is presented. Canonical correlation analysis is used to adapt filters in a local image

neighbourhood. The adapted filters are then analysed with respect to phase to get the disparity estimate. The algorithm can handle differently scaled image pairs and depth discontinuities. It can also estimate multiple depths in semi-transparent images.

Chapter 8 is a summary of the thesis and also contains some thoughts on future research.

Finally there are two appendices. Appendix A contains definitions. In appendix B, most of the proofs have been placed. In this way, the text is hopefully easier to follow for the reader who does not want to get too deep into mathematical details. This also makes it possible to give the proofs space enough to be followed without too much effort and to include proofs that initiated readers may consider unnecessary without disrupting the text.

1.3 Notation

Lowercase letters in italics (x) are used for scalars, lowercase letters in boldface ($\mathbf{x}$) are used for vectors and uppercase letters in boldface ($\mathbf{X}$) are used for matrices. The transpose of a real valued vector or a matrix is denoted $\mathbf{x}^T$. The conjugate transpose is denoted $\mathbf{x}^*$. The *norm* $\|\mathbf{v}\|$ of a vector $\mathbf{v}$ is defined by

$$\|\mathbf{v}\| \equiv \sqrt{\mathbf{v}^*\mathbf{v}}$$

and a "hat" ($\hat{\mathbf{v}}$) indicates a vector with unit length, i.e.

$$\hat{\mathbf{v}} \equiv \frac{\mathbf{v}}{\|\mathbf{v}\|}.$$

$E[\cdot]$ means *expectation value* of a stochastic variable.

Part I

Learning

Chapter 2

Learning systems

Learning systems is a central concept in this dissertation and in this chapter, three different principles of learning are described. Some standard techniques are described and some important issues related to machine learning are discussed. But first, what is learning?

2.1 Learning

According to Oxford Advanced Learner's Dictionary (Hornby, 1989), learning is to

> "gain knowledge or skill by study, experience or being taught."

Knowledge may be considered as a set of rules determining how to act. Hence, knowledge can be said to define a *behaviour* which, according to the same dictionary, is a "way of acting or functioning." Narendra and Thathachar (1974), two learning automata theorists, make the following definition of learning:

> "Learning is defined as any relatively permanent change in behaviour resulting from past experience, and a learning system is characterized by its ability to improve its behaviour with time, in some sense towards an ultimate goal."

Learning has been a field of study since the end of the nineteenth century. Thorndike (1898) presented a theory in which an association between a stimulus and a response is established and this association is strengthened or weakened depending on the outcome of the response. This type of learning is called *operant conditioning*. The theory of *classical conditioning* (Pavlov, 1955) is concerned with the case when a natural reflex to a certain stimulus becomes a response of a second stimulus that has preceded the original stimulus several times.

In the 1930s, Skinner developed Thorndike's ideas but claimed, as opposed to Thorndike, that learning was more "trial and success" than "trial and error" (Skinner, 1938). These ideas belong to the psychological position called *behaviourism*. Since the 1950s, *rationalism* has gained more interest. In this view, intentions and abstract reasoning play an important role in learning. In this thesis, however, there is a more behaviouristic view. The aim is not to model biological systems or mental processes. The goal is rather to make a machine that produces the desired results. As will be seen, the learning principle called *reinforcement learning* discussed in section 2.4 has much in common with Thorndike's and Skinner's operant conditioning. Learning theories have been thoroughly described for example by Bower and Hilgard (1981).

There are reasons to believe that "learning by doing" is the only way of learning to produce responses or, as stated by Brooks (1986):

> "These two processes of learning and doing are inevitably intertwined; we learn as we do and we do as well as we have learned."

An example of "learning by doing" is illustrated in an experiment (Held and Bossom, 1961; Mikaelian and Held, 1964) where people wearing goggles that rotated or displaced their fields of view were either walking around for an hour or wheeled around the same path in a wheel-chair for the same amount of time. The adaptation to the distortion was then tested. The subjects that had been walking had adapted while the other subjects had not. A similar situation occurs for instance when you are going somewhere by car. If you have driven to a certain destination before, instead of being a passenger, you probably will find your way easier the next time.

2.2 Machine learning

We are used to seeing humans and animals learn, but how does a machine learn? The answer depends on how knowledge or behaviour is represented in the machine.

Let us consider knowledge to be a rule for how to generate responses to certain stimuli. One way of representing knowledge is to have a table with all stimuli and corresponding responses. Learning would then take place if the system, through experience, filled in or changed the responses in the table. Another way of representing knowledge is by using a parameterized model, where the output is obtained as a given function of the input $\mathbf{x}$ and a parameter vector $\mathbf{w}$:

$$\mathbf{y} = f(\mathbf{x}, \mathbf{w}). \tag{2.1}$$

Learning would then be to change the model parameters in order to improve the performance. This is the learning method used for example in neural networks.

Another way of representing knowledge is to consider the input space and output space together. Examples of this approach are an algorithm by Munro (1987) and the Q-learning algorithm (Watkins, 1989). Another example is the prediction matrix memory described in section 3.3.1. The combined space of input and output can be called the *decision space*, since this is the space in which the combinations of input and output (i.e. stimuli and responses) that constitute *decisions* exist. The decision space could be treated as a table in which suitable decisions are marked. Learning would then be to make or change these markings. Or the knowledge could be represented in the decision space as distributions describing suitable combinations of stimuli and responses (Landelius, 1993, 1997):

$$p(\mathbf{y}, \mathbf{x}, \mathbf{w}) \tag{2.2}$$

where, again, $\mathbf{y}$ is the response, $\mathbf{x}$ is the input signal and $\mathbf{w}$ contains the parameters of a given distribution function. Learning would then be to change the parameters of these distributions through experience in order to improve some measure of performance. Responses can then be generated from the conditional probability function

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}). \tag{2.3}$$

The issue of representing knowledge is further discussed in chapter 3.

Obviously a machine can learn through experience by changing some parameters in a model or data in a table. But what is the experience and what measure of performance is the system trying to improve? In other words, *what* is the system learning? The answers to these questions depend on what kind of learning we are talking about. Machine learning can be divided into three classes that differ in the external feedback to the system during learning:

- Supervised learning
- Reinforcement learning
- Unsupervised learning

The three different principles are illustrated in figure 2.1.

In the following three sections, these three principles of learning are discussed in more detail. In section 2.6, the relations between the three methods are discussed and it is shown that the differences are not as great as they may seem at first.

2.3 Supervised learning

In supervised learning there is a teacher who shows the system the desired responses for a representative set of stimuli (see figure 2.1). Here, the experience

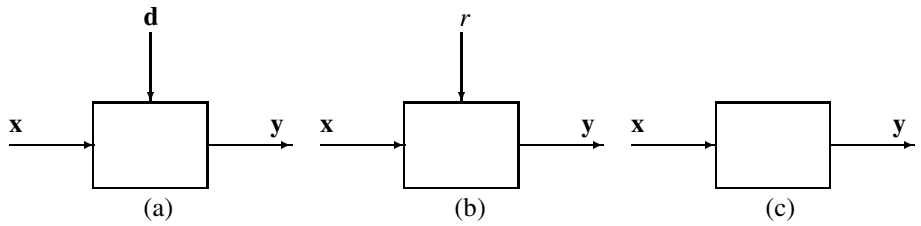

Figure 2.1: The three different principles of learning: Supervised learning (a), Reinforcement learning (b) and Unsupervised learning (c).

is pairs of stimuli and desired responses and improving performance means minimizing some error measure, for example the mean squared distance between the system's output and the desired output.

Supervised learning can be described as function approximation. The teacher delivers samples of the function and the algorithm tries, by adjusting the parameters $\mathbf{w}$ in equation 2.1 or equation 2.2, to minimize some cost function

$$\mathcal{E} = E[\varepsilon], \tag{2.4}$$

where $E[\varepsilon]$ stands for the expectation of costs ε over the distribution of data. The instantaneous cost ε depends on the difference between the output of the algorithm and the samples of the function. In this sense, regression techniques can be seen as supervised learning. In general, the cost function also includes a *regularization* term. The regularization term prevents the system from what is called *over-fitting*. This is important for the *generalization* capabilities of the system, i.e. the performance of the system for new data not used for training. In effect, the regularization term can be compared to the polynomial degree in polynomial regression.

2.3.1 Gradient search

Most supervised learning algorithms are based on *gradient search* on the cost function. Gradient search means that the parameters w_i are changed a small step in the opposite direction of the gradient of the cost function $\mathcal{E}$ for each iteration of the process, i.e.

$$w_i(t+1) = w_i(t) - \alpha \frac{\partial \mathcal{E}}{\partial w_i}, \tag{2.5}$$

where the update factor α is used to control the step length. In general, the negative gradient does of course not point exactly towards the minimum of the cost

function. Hence, a gradient search will in general not find the shortest way to the optimum.

There are several methods to improve the search by using the second-order partial derivatives (Battiti, 1992). Two well-known methods are Newton's method (see for example Luenberger, 1969) and the conjugate-gradient method (Fletcher and Reeves, 1964). Newton's method is optimal for quadratic cost functions in the sense that it, given the Hessian (i.e. the matrix of second order partial derivatives), can find the optimum in one step. The problem is the need for calculation and storage of the Hessian and its inverse. The calculation of the inverse requires the Hessian to be non-singular which is not always the case. Furthermore, the size of the Hessian grows quadratically with the number of parameters. The conjugate-gradient method is also a second-order technique but avoids explicit calculation of the second-order partial derivatives. For an n-dimensional quadratic cost function it reaches the optimum in n steps, but here each step includes a line search which increases the computational complexity in each step. A line search can of course also be performed in first-order gradient search. Such a method is called *steepest descent*. In steepest descent, however, the profit from the line search is not so big. The reason for this is that two successive steps in steepest descent are always perpendicular and, hence, the parameter vector will in general move in a zigzag path.

In practice, the true gradient of the cost function is, in most cases, not known since the expected cost $\mathcal{E}$ is unknown. In these cases, an instantaneous sample $\varepsilon(t)$ of the cost function can be used and the parameters are changed according to

$$w_i(t+1) = w_i(t) - \alpha \frac{\partial \varepsilon(t)}{\partial w_i(t)}. \tag{2.6}$$

This method is called *stochastic* gradient search since the gradient estimate varies with the (stochastic) data and the estimate improves on average with an increasing number of samples (see for example Haykin, 1994).

2.3.2 Adaptability

The use of instantaneous estimates of the cost function is not necessarily a disadvantage. On the contrary, it allows for system adaptability. Instantaneous estimates permit the system to handle non-stationary processes, i.e. the cost function is changing over time.

The choice of the update factor α is crucial for the performance of stochastic gradient search. If the factor is too large, the algorithm will start oscillating and never converge and if the factor is too small, the convergence time will be far too long. In the literature, the factor is often a decaying function of time. The intuitive reason for this is that the more samples the algorithm has used, the closer

the parameter vector should be to the optimum and the smaller the steps should be. But, in most cases, the real reason for using a time-decaying update factor is probably that it makes it easier to prove convergence.

In practice, however, choosing α as a function of time only is not a very good idea. One reason is that the optimal rate of decay depends on the problem, i.e. the shape of the cost function, and is therefore impossible to determine beforehand. Another important reason is adaptability. A system with an update factor that decays as a function of time only cannot adapt to new situations. Once the parameters have converged, the system is fixed. In general, a better solution is to use an adaptive update factor that enables the parameters to change in large steps when consistently moving towards the optimum and to decrease the steps when the parameter vector is oscillating around the optimum. One example of such methods is the *Delta-Bar-Delta* rule (Jacobs, 1988). This algorithm has a separate adaptive update factor α_i for each parameter.

Another fundamental reason for adaptive update factors, not often mentioned in the literature, is that the step length in equation 2.6 is proportional to the *norm* of the gradient. It is, however, only the *direction* of the gradient that is relevant, not the norm. Consider, for example, finding the maximum of a Gaussian by moving proportional to its gradient. Except for a region around the optimum, the step length gets smaller the further we get from the optimum. A method that deals with this problem is the *RPROP* algorithm (Riedmiller and Braum, 1993) which adapts the actual step lengths of the parameters and not just the factors α_i.

2.4 Reinforcement learning

In reinforcement learning there is a teacher too, but this teacher does not give the desired responses. Only a scalar reward or punishment (reinforcement signal) according to the quality of the system's overall performance is fed back to the system, as illustrated in figure 2.1 on page 10. In this case, each experience is a triplet of stimulus, response and corresponding reinforcement. The performance to improve is simply the received reinforcement. What is meant by received reinforcement depends on whether or not the system acts in a closed loop, i.e. the input to the system or the system state is dependent on previous output. If there is a closed loop, an accumulated reward over time is probably more important than each instant reward. If there is no closed loop, there is no conflict between maximizing instantaneous reward and accumulated rewards.

The feedback to a reinforcement learning system is *evaluative* rather than *instructive*, as in supervised learning. The reinforcement signal is in most cases easier to obtain than a set of correct responses. Consider, for example, the situation when a child learns to bicycle. It is not possible for the parents to explain

to the child how it should behave, but it is quite easy to observe the trials and conclude *how good* the child manages. There is also a clear (though negative) reinforcement signal when the child fails. The simple feedback is perhaps the main reason for the great interest in reinforcement learning in the fields of autonomous systems and robotics. The teacher does not have to know *how* the system should solve a task but only be able to decide *if* (and perhaps how good) it solves it. Hence, a reinforcement learning system requires feedback to be able to learn, but it is a very simple form of feedback compared to what is required for a supervised learning system. In some cases, the teacher's task may even become so simple that it can be built into the system. For example, consider a system that is only to learn to avoid heat. Here, the teacher may consist only of a set of heat sensors. In such a case, the reinforcement learning system is more like an *unsupervised* learning system than a supervised one. For this reason, reinforcement learning is often referred to as a class of learning systems that lies in between supervised and unsupervised learning systems.

A *reinforcement*, or reinforcing stimulus, is defined as a stimulus that strengthens the behaviour that produced it. As an example, consider the procedure of training an animal. In general, there is no point in trying to explain to the animal how it should behave. The only way is simply to reward the animal when it does the right thing. If an animal is given a piece of food each time it presses a button when a light is flashed, it will (in most cases) learn to press the button when the light signal appears. We say that the animal's behaviour has been *reinforced*. We use the food as a *reward* to train the animal. One could, in this case, say that it is the food itself that reinforces the behaviour. In general, there is some mechanism in the animal that generates an internal reinforcement signal when the animal gets food (at least if it is hungry) and when it experiences other things that are good for it i.e. that increase the probability of the reproduction of its genes. A biochemical process involving *dopamine* is believed to play a central role in the distribution of the reward signal (Bloom and Lazerson, 1985; Schultz et al., 1997). In the 1950s, experiments were made (Olds and Milner, 1954) where the internal reward system was artificially stimulated instead of giving an external reward. In this case, the animal was even able to learn self destructive behaviour.

In the example above, the reward (piece of food) was used merely to trigger the reinforcement signal. In the following discussion of artificial systems, however, the two terms have the same meaning. In other words, we will use only one kind of reward, namely the reinforcement signal itself, which we in the case of an artificial system can allow us to have direct access to without any ethical considerations. In case of a large system, one would of course want the system to be able to solve different routine tasks besides the main task (or tasks). For instance, suppose we want the system to learn to charge its batteries. Such a behaviour should

then be reinforced in some way. Whether we put a box into the system that reinforces the battery-charging behaviour or we let the charging device or a teacher deliver the reinforcement signal is a technical question rather than a philosophical one. If, however, the box *is* built into the system, we can reinforce behaviour by charging the system's batteries.

Reinforcement learning is strongly associated with learning among animals (including humans) and some people find it hard to see how a machine could learn by a "trial-and-error" method. To show that machines can indeed learn in this way, a simple example was created by Donald Michie in the 1960s. A pile of match-boxes that learns to play noughts and crosses illustrates that even a very simple machine can learn by trial and error. The machine is called MENACE (Match-box Educable Noughts And Crosses Engine) and consists of 288 match-boxes, one for each possible state of the game. Each box is filled with a random set of coloured beans. The colours represent different moves. Each move is determined by the colour of a randomly selected bean from the box representing the current state of the game. If the system wins the game, new beans with the same colours as those selected during the game are added to the respective boxes. If the system loses, the beans that were selected are removed. In this way, after each game, the possibility of making good moves increases and the risk of making bad moves decreases. Ultimately, each box will only contain beans representing moves that have led to success.

There are some notable advantages with reinforcement learning compared to supervised learning, besides the obvious fact that reinforcement learning can be used in some situations where supervised learning is impossible (e.g. the child learning to bicycle and the animal learning examples above). The ability to learn by receiving rewards makes it possible for a reinforcement learning system to become more skilful than its teacher. It can even improve its behaviour by training itself, as in the backgammon program by Tesauro (1990).

2.4.1 Searching for higher rewards

In reinforcement learning, the feedback to the system contains no gradient information, i.e. the system does not know in what direction to search for a better solution. For this reason, most reinforcement learning systems are designed to have a stochastic behaviour. A stochastic behaviour can be obtained by adding noise to the output of a deterministic input-output function or by generating the output from a probability distribution. In both cases, the output can be seen as consisting of two parts: one deterministic and one stochastic. It is easy to see that both these parts are necessary in order for the system to be able to improve its behaviour. The deterministic part is the optimum response given the current knowledge. Without the deterministic part, the system would make no sensible

decisions at all. However, if the deterministic part was the only one, the system would easily get trapped in a non-optimal behaviour. As soon as the received rewards are consistent with current knowledge, the system will be satisfied and never change its behaviour. Such a system will only maximize *the reward predicted by the internal model* but not *the external reward actually received*. The stochastic part of the response provides the system with information from points in the decision space that would never be sampled otherwise. So, the deterministic part of the output is necessary for generating good responses with respect to the current knowledge and the stochastic part is necessary for gaining more knowledge. The stochastic behaviour can also help the system avoid getting trapped in local maxima.

The conflict between the need for exploration and the need for precision is typical of reinforcement learning. The conflict is usually referred to as the *exploration-exploitation dilemma*. This dilemma does not normally occur in supervised learning.

At the beginning when the system has poor knowledge of the problem to be solved, the deterministic part of the response is very unreliable and the stochastic part should preferably dominate in order to avoid a misleading bias in the search for correct responses. Later on, however, when the system has gained more knowledge, the deterministic part should have more influence so that the system makes at least reasonable guesses. Eventually, when the system has gained a lot of experience, the stochastic part should be very small in order not to disturb the generation of correct responses. A constant relation between the influence of the deterministic and stochastic parts is a compromise which will give a poor search behaviour (i.e. slow convergence) at the beginning and bad precision after convergence. Therefore, many reinforcement learning systems have noise levels that decays with time. There is, however, a problem with such an approach too. The decay rate of the noise level must be chosen to fit the problem. A difficult problem takes longer time to solve and if the noise level is decreased too fast, the system may never reach an optimal solution. Conversely, if the noise level decreases too slowly, the convergence will be slower than necessary. Another problem arises in a dynamic environment where the task may change after some time. If the noise level at that time is too low, the system will not be able to adapt to the new situation. For these reasons, an adaptive noise level is to prefer.

The basic idea of an adaptive noise level is that when the system has a poor knowledge of the problem, the noise level should be high and when the system has reached a good solution, the noise level should be low. This requires an internal quality measure that indicates the average performance of the system. It could of course be accomplished by accumulating the rewards delivered to the system, for

instance by an iterative method, i.e.

$$p(t+1) = \alpha p(t) + (1-\alpha) r(t), \tag{2.7}$$

where p is the performance measure, r is the reward and α is the update factor, $0 < \alpha < 1$. Equation 2.7 gives an exponentially decaying average of the rewards given to the system, where the most recent rewards will be the most significant ones.

A solution, involving a variance that depends on the predicted reinforcement, has been suggested by Gullapalli (1990). The advantage with such an approach is that the system might expect different rewards in different situations for the simple reason that the system may have learned some situations better than others. The system should then have a very deterministic behaviour in situations where it predicts high rewards and a more exploratory behaviour in situations where it is more uncertain. Such a system will have a noise level that depends on the *local skill* rather than the average performance.

Another way of controlling the noise level, or rather the standard deviation σ of a stochastic output unit, is found in the REINFORCE algorithm (Williams, 1988). Let μ be the mean of the output distribution and y the actual output. When the output y gives a higher reward than the recent average, the variance will decrease if $|y-\mu| < \sigma$ and increase if $|y-\mu| > \sigma$. When the reward is less than average, the opposite changes are made. This leads to a more narrow search behaviour if good solutions are found close to the current solution or bad solutions are found outside the standard deviation and a wider search behaviour if good solutions are found far away or bad solutions are found close to the mean.

Another strategy for a reinforcement learning system to improve its behaviour is to differentiate a model of the reward with respect to the system parameters in order to estimate the gradient of the reward in the system's parameter space. The model can be known a priori and built into the system, or it can be learned and refined during the training of the system. To know the gradient of the reward means to know in which direction in the parameter space to search for a better performance. One way to use this strategy is described by Munro (1987) where the model is a secondary network that is trained to predict the reward. This can be done with back-propagation, using the difference between the reward and the prediction as an error measure. Then back-propagation can be used to modify the weights in the primary network, but here with the aim of maximizing the prediction done by secondary network. A similar approach was used to train a pole-balancing system (Barto et al., 1983). Other examples of similar strategies are described by Williams (1988).

Adaptive critics

When the learning system operates in a dynamic environment, the system may have to carry out a sequence of actions to get a reward. In other words, the feedback to such a system may be infrequent and delayed and the system faces what is known as the *temporal credit assignment problem* (see section 2.7.2 on page 35). Assume that the environment or process to be controlled is a *Markov process*. A Markov process consists of a set S of states $\mathbf{s}_i$ where the conditional probability of a state transition only depends on a finite number of previous states. The definition of the states can be reformulated so that the state transition probabilities only depend on the current state, i.e.

$$P(\mathbf{s}_{k+1} \mid \mathbf{s}_k, \mathbf{s}_{k-1}, \ldots, \mathbf{s}_1) = P(\mathbf{s'}_{k+1} \mid \mathbf{s'}_k), \tag{2.8}$$

which is a *first order* Markov process. Derin and Kelly (1989) present a systematic classification of different types of Markov models.

Suppose one or several of the states in a Markov process are associated with a reward. Now, the goal for the learning system can be defined as maximizing the total accumulated reward for all future time steps. One way to accomplish this task for a *discrete* Markov process is, like in the MENACE example above, to store all states and actions until the final state is reached and to update the state transition probabilities afterwards. This method is referred to as *batch learning*. An obvious disadvantage with batch learning is the need for storage which will become infeasible for large dimensionalities of the input and output vectors as well as for long sequences.

A problem occurring when only the final outcome is considered is illustrated in figure 2.2. Consider a game where a certain position has resulted in a loss in 90% of the cases and a win in 10% of the cases. This position is classified as a *bad* position. Now, suppose that a player reaches a *novel* state (i.e. a state that has not been visited before) that inevitably leads to the *bad* state and finally happens to lead to a win. If the player waits until the end of the game and only looks at the result, he would label the novel state as a *good* state since it led to a win. This is, however, not true. The novel state is a *bad* state since it *probably* leads to a loss.

Adaptive critics (Barto, 1992) is a class of methods designed to handle the problem illustrated in figure 2.2. Let us, for simplicity, assume that the input vector $\mathbf{x}_k$ uniquely defines the state $\mathbf{s}_k$[1]. Suppose that for each state $\mathbf{x}_k$ there is a value $V_g(\mathbf{x}_k)$ that is an estimate of the expected future result (e.g. a weighted sum of the accumulated reinforcement) when following a policy g, i.e. generating the output as $\mathbf{y} = g(\mathbf{x})$. In adaptive critics, the value $V_g(\mathbf{x}_k)$ depends on the value

[1] This assumption is of course not always true. When it does not hold, the system faces the *perceptual aliasing problem* which is discussed in section 2.7.1 on page 33.

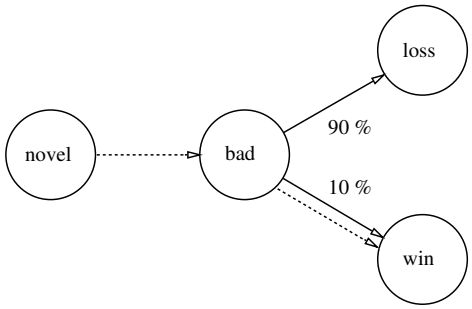

Figure 2.2: An example to illustrate the advantage of adaptive critics. A state that is likely to lead to a loss is classified as a *bad* state. A novel state that leads to the bad state but then happens to lead to a win is classified as a *good* state if only the final outcome is considered. In adaptive critics, the novel state is recognized as a bad state since it *most likely* leads to a loss.

$V_g(\mathbf{x}_{k+1})$ and not only on the final result:

$$V_g(\mathbf{x}_k) = r(\mathbf{x}_k, g(\mathbf{x}_k)) + \gamma V_g(\mathbf{x}_{k+1}), \tag{2.9}$$

where $r(\mathbf{x}_k, g(\mathbf{x}_k))$ is the reward for being in the state $\mathbf{x}_k$ and generating the response $\mathbf{y}_k = g(\mathbf{x}_k)$. This means that

$$V_g(\mathbf{x}_k) = \sum_{i=k}^{N} \gamma^{i-k} r(\mathbf{x}_k, g(\mathbf{x}_k)), \tag{2.10}$$

i.e. the value of a state is a weighted sum of all future rewards. The weight $\gamma \in [0,1]$ can be used to make rewards that are close in time more valuable than rewards further away. Equation 2.9 makes it possible for adaptive critics to improve their predictions during a process without always having to wait for the final result.

Suppose that the environment can be described by the function f so that $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{y}_k)$. Now equation 2.9 can be written as

$$V_g(\mathbf{x}_k) = r(\mathbf{x}_k, g(\mathbf{x}_k)) + \gamma V_g\left(f(\mathbf{x}_k, g(\mathbf{x}_k))\right). \tag{2.11}$$

The optimal response y_* is the response given by the optimal policy g_*:

$$y_* = g_*(\mathbf{x}) = \arg\max_{\mathbf{y}} \{r(\mathbf{x}, \mathbf{y}) + V_*(f(\mathbf{x}, \mathbf{y}))\}, \tag{2.12}$$

where V_* is the value of the optimal policy (Bellman, 1957).

In the methods of *temporal differences* (TD) described by Sutton (1988), the value function V is estimated using the difference between the values of two consecutive states as an internal reward signal. Another well known method for adaptive critics is *Q-learning* (Watkins, 1989). In Q-learning, the system is trying to estimate the Q-function

$$Q_g(\mathbf{x},\mathbf{y}) = r(\mathbf{x},\mathbf{y}) + V_g(f(\mathbf{x},\mathbf{y})) \tag{2.13}$$

rather than the value function V itself. Using the Q-function, the optimal response is

$$y_* = g_*(\mathbf{x}) = \arg\max_{\mathbf{y}}\{Q_*(\mathbf{x},\mathbf{y})\}. \tag{2.14}$$

This means that a model of the environment f is not required in Q-learning in order to find the optimal response.

In control theory, an optimization algorithm called *dynamic programming* is a well-known method for maximizing the expected total accumulated reward. The relationship between TD-methods and dynamic programming has been discussed for example by Barto (1992), Werbos (1990) and Whitehead et al. (1990). It should be noted, however, that maximizing the expected accumulated reward is not always the best criterion, as discussed by Heger (1994). He notes that this criterion of choice of action

- is based upon long-run consideration where the decision process is repeated a sufficiently large number of times. It is not necessarily a valid criterion in the short run or one-shot case, especially when the possible consequences or their probabilities have extreme values.

- assumes the subjective values of possible outcomes to be proportional to their objective values, which is not necessarily the case, especially when the values involved are large.

As an illustrative example, many people occasionally play on lotteries in spite of the fact that the expected outcome is negative. Another example is that most people do not invest all their money in stocks although such a strategy would give a larger expected payoff than putting some of it in the bank.

The first well-known use of adaptive critics was in a checkers playing program (Samuel, 1959). In that system, the value of a state (board position) was updated according to the values of future states likely to appear. The prediction of future states requires a model of the environment (game). This is, however, not the case in TD-methods like the *adaptive heuristic critic* algorithm (Sutton, 1984) where the feedback comes from *actual* future states and, hence, prediction is not necessary.

Sutton (1988) has proved a convergence theorem for one TD-method[2] that states that the prediction for each state asymptotically converges to the maximum-likelihood prediction of the final outcome for states generated in a Markov process. Other proofs concerning adaptive critics in finite state systems have been presented, for example by Watkins (1989), Jaakkola et al. (1994) and Baird (1995). Proofs for continuous state spaces have been presented by Werbos (1990), Bradtke (1993) and Landelius (1997).

Other methods for handling delayed rewards are for example *heuristic dynamic programming* (Werbos, 1990) and *back-propagation of utility* (Werbos, 1992).

Recent physiological findings indicate that the output of dopaminergic neurons indicate errors in the predicted reward function, i.e. the internal reward used in TD-learning (Schultz et al., 1997).

2.4.2 Generating the reinforcement signal

Werbos (1990) defines a reinforcement learning system as

> "any system that through interaction with its environment improves its performance by receiving feedback in the form of a scalar reward (or penalty) that is commensurate with the appropriateness of the response."

The goal for a reinforcement learning system is simply to maximize the reward, for example the accumulated value of the reinforcement signal r. Hence, r can be said to define the problem to be solved and therefore the choice of reward function is very important. The reward, or reinforcement, must be capable of evaluating the overall performance of the system and be informative enough to allow learning.

In some cases, how to choose the reinforcement signal is obvious. For example, in the pole balancing problem (Barto et al., 1983), the reinforcement signal is chosen as a negative value upon failure and as zero otherwise. Many times, however, how to measure the performance is not evident and the choice of reinforcement signal will affect the learning capabilities of the system.

The reinforcement signal should contain as much information as possible about the problem. The learning performance of a system can be improved considerably if a pedagogical reinforcement is used. One should not sit and wait for the system to attain a perfect performance, but use the reward to guide the system to a better performance. This is obvious in the case of training animals and

[2] In this TD-method, called TD(0), the value V_k only depends on the following value V_{k+1} and not on later predictions. Other TD-methods can take into account later predictions with a function that decreases exponentially with time.

humans, but it also applies to the case of training artificial systems with reinforcement learning. Consider, for instance, an example where a system is to learn a simple function $y = f(x)$. If a binary reward is used, i.e.

$$r = \begin{cases} 1 & if \quad |\tilde{y} - y| < \varepsilon \\ 0 & else \end{cases}, \tag{2.15}$$

where $\tilde{y}$ is the output of the system and y is the correct response, the system will receive no information at all[3] as long as the responses are outside the interval defined by ε. If, on the other hand, the reward is chosen inversely proportional to the error, i.e.

$$r = \frac{1}{|\tilde{y} - y|} \tag{2.16}$$

a relative improvement will yield the same relative increase in reward for all output. In practice, of course, the reward function in equation 2.16 could cause numerical problems, but it serves as an illustrative example of a well-shaped reward function. In general, a smooth and continuous function is preferable. Also, the derivative should not be too small, at least not in regions where the system should not get stuck, i.e. in regions of bad performance. It should be noted, however, that sometimes there is no obvious way of defining a continuous reward function. In the case of pole balancing (Barto et al., 1983), for example, the pole either falls or not.

A perhaps more interesting example where a pedagogical reward is used can be found in a paper by Gullapalli (1990), which presents a "reinforcement learning system for learning real-valued functions". This system was supplied with two input variables and one output variable. In one case, the system was trained on an XOR-task. Each input was 0.1 or 0.9 and the output was any real number between 0 and 1. The optimal output values were 0.1 and 0.9 according to the logical XOR-rule. At first, the reinforcement signal was calculated as

$$r = 1 - |+\varepsilon|, \tag{2.17}$$

where ε is the difference between the output and the optimal output. The system sometimes converged to wrong results, and in several training runs it did not converge at all. A new reinforcement signal was calculated as

$$r' = \frac{r + r_{\text{task}}}{2}. \tag{2.18}$$

[3] Well, almost none in any case, and as the number of possible solutions which give output outside the interval approaches infinity (which it does in a continuous system), the information approaches zero.

The term r_{task} was set to 0.5 if the latest output for similar input was less than the latest output for dissimilar input and to -0.5 otherwise. With the reinforcement signal in equation 2.18, the system began by trying to satisfy a weaker definition of the XOR-task, according to which the output should be higher for dissimilar inputs than for similar inputs. The learning performance of the system improved in several ways with the new reinforcement signal.

Another reward strategy is to reward only improvements in behaviour, for example by calculating the reinforcement as

$$r = p - \bar{r}, \tag{2.19}$$

where p is a performance measure and $\bar{r}$ is the mean reward acquired by the system. Equation 2.19 gives a system that is never satisfied since the reward vanishes in any solution with a stable reward. If the system has an adaptive search behaviour as described in the previous section, it will keep on searching for better and better solutions. The advantage with such a reward is that the system will not get stuck in a local optimum. The disadvantage is, of course, that it will not stay in the global optimum either, if such an optimum exists. It will, however, always return to the global optimum and this behaviour can be useful in a dynamic environment where a new optimum may appear after some time.

Even if the reward in the previous equation is a bit odd, it points out the fact that there might be negative reward or punishment. The pole balancing system (Barto et al., 1983) is an example of the use of negative reinforcement and in this case it is obvious that it is easier to deliver punishment upon failure than reward upon success since the reward would be delivered after an unpredictably long sequence of actions; it would take an infinite amount of time to verify a success! In general, however, it is probably better to use positive reinforcement to guide a system towards a solution for the simple reason that there is usually more information in the statement "this was a good solution" than in the opposite statement "this was not a good solution". On the other hand, if the purpose is to make the system avoid a particular solution (i.e. "Do anything but this!"), punishment would probably be more efficient.

2.4.3 Learning in an evolutionary perspective

In this section, a special case of reinforcement learning called *genetic algorithms* is described. The purpose is not to give a detailed description of genetic algorithms, but to illustrate the fact that they are indeed reinforcement learning algorithms. From this fact and the obvious similarity between biological evolution and genetic algorithms (as indicated in the name), some interesting conclusions can be drawn concerning the question of learning at different time scales.

A genetic algorithm is a stochastic search method for solving optimization problems. The theory was founded by Holland (1975) and it is inspired by the theory of natural evolution. In natural evolution, the problem to be optimized is how to survive in a complex and dynamic environment. The knowledge of this problem is encoded as *genes* in the individuals' *chromosomes*. The individuals that are best adapted in a population have the highest probability of *reproduction*. In reproduction, the genes of the new individuals (children) are a mixture or *crossover* of the parents' genes. In reproduction there is also a random change in the chromosomes. The random change is called *mutation*.

A genetic algorithm works with coded structures of the parameter space in a similar way. It uses a population of coded structures (individuals) and evaluates the performance of each individual. Each individual is reproduced with a probability that depends on that individual's performance. The genes of the new individuals are a mixture of the genes of two parents (crossover), and there is a random change in the coded structure (mutation).

Thus, genetic algorithms learn by the method of trial and error, just like other reinforcement learning algorithms. We might therefore argue that the same basic principles hold both for developing a system (or an individual) and for adapting the system to its environment. This is important since it makes the question of what should be built into the machine from the beginning and what should be learned by the machine more of a practical engineering question than a principal one. The conclusion does not make the question less important though; in practice, it is perhaps one of the most important issues.

Another interesting relation between evolution and learning on the individual level is discussed by Hinton and Nowlan (1987). They show that learning organisms evolve faster than non-learning equivalents. This is maybe not very surprising if evolution and learning are considered as merely different levels of a hierarchical learning system. Then the convergence of the slow high-level learning process (corresponding to evolution) depends on the adaptability of the faster low-level learning process (corresponding to individual learning). This indicates that hierarchical systems adapt faster than non-hierarchical systems of the same complexity.

More information about genetic algorithms can be found for example in the books by Davis (1987) and Goldberg (1989).

2.5 Unsupervised learning

In unsupervised learning there is no external feedback at all (see figure 2.1 on page 10). The system's experience mentioned on page 9 consists of a set of signals and the measure of performance is often some statistical or information theoretical

property of the signal. Unsupervised learning is perhaps not learning in the word's everyday sense, since the goal is not to learn to produce responses in the form of useful actions. Rather, it is to learn a certain representation which is thought to be useful in further processing. The importance of a good representation of the signals is discussed in chapter 3.

Unsupervised learning systems are often called *self-organizing systems* (Haykin, 1994; Hertz et al., 1991). Hertz et al. (1991) describe two principles for unsupervised learning: *Hebbian learning* and *competitive learning*. Also Haykin (1994) uses these two principles but adds a third one that is based on *mutual information*, which is an important concept in this thesis. Next, these three principles of unsupervised learning are described.

2.5.1 Hebbian learning

Hebbian learning originates from the pioneering work of neuropsychologist Hebb (1949). The basic idea is that when one neuron repeatedly causes a second neuron to fire, the connection between them is strengthened. Hebb's idea has later been extended to include the formulation that if the two neurons have uncorrelated activities, the connection between them is weakened. In learning and neural network theory, Hebbian learning is usually formulated more mathematically. Consider a linear unit where the output is calculated as

$$y = \sum_{i=1}^{N} w_i x_i. \tag{2.20}$$

The simplest Hebbian learning rule for such a linear unit is

$$w_i(t+1) = w_i(t) + \alpha x_i(t) y(t). \tag{2.21}$$

Consider the expected change $\Delta\mathbf{w}$ of the parameter vector $\mathbf{w}$ using $y = \mathbf{x}^T\mathbf{w}$:

$$E[\Delta\mathbf{w}] = \alpha E[\mathbf{x}\mathbf{x}^T]\mathbf{w} = \alpha\mathbf{C}_{xx}\mathbf{w}. \tag{2.22}$$

Since $\mathbf{C}_{xx}$ is positive semi-definite, any component of $\mathbf{w}$ parallel to an eigenvector of $\mathbf{C}_{xx}$ corresponding to a non-zero eigenvalue will grow exponentially and a component in the direction of an eigenvector corresponding to the largest eigenvalue (in the following called a *maximal eigenvector*) will grow fastest. Therefore we see that $\mathbf{w}$ will approach a maximal eigenvector of $\mathbf{C}_{xx}$. If $\mathbf{x}$ has zero mean, $\mathbf{C}_{xx}$ is the covariance matrix of $\mathbf{x}$ and, hence, a linear unit with Hebbian learning will find the direction of *maximum variance* in the input data, i.e. the *first principal component* of the input signal distribution (Oja, 1982). Principal component analysis (PCA) is discussed in section 4.2 on page 64.

A problem with equation 2.21 is that it does not converge. A solution to this problem is Oja's rule (Oja, 1982):

$$w_i(t+1) = \alpha y(t)\left(x_i(t) - y(t)w_i(t)\right). \tag{2.23}$$

This extension of Hebb's rule makes the norm of $\mathbf{w}$ approach 1 and the direction will still approach that of a maximal eigenvector, i.e. the first principal component of the input signal distribution. Again, if $\mathbf{x}$ has zero mean, Oja's rule finds the one-dimensional representation y of $\mathbf{x}$ that has the maximum variance under the constraint that $\|\mathbf{w}\| = 1$.

In order to find more than one principal component, Oja (1989) proposed a modified learning rule for N units:

$$w_{ij}(t+1) = \alpha y_i(t)\left(x_j(t) - \sum_{k=1}^{N} y_k(t)w_{kj}(t)\right), \tag{2.24}$$

where w_{ij} is the weight j in unit i. A similar modification for N units was proposed by Sanger (1989), which is identical to equation 2.24 except for the summation that ends at i instead of N. The difference is that Sanger's rule finds the N first principal components (sorted in order) whereas Oja's rule finds N vectors spanning the same subspace as the N first principal components.

A note on correlation and covariance matrices

In neural network literature, the matrix $\mathbf{C}_{xx}$ in equation 2.22 is often called a correlation matrix. This can be a bit confusing, since $\mathbf{C}_{xx}$ does not contain the correlations between the variables in a statistical sense, but rather the expected values of the products between them. The correlation between x_i and x_j is defined as

$$\rho_{ij} = \frac{E[(x_i - \bar{x}_i)(x_j - \bar{x}_j)]}{\sqrt{E[(x_i - \bar{x}_i)^2]E[(x_j - \bar{x}_j)^2]}} \tag{2.25}$$

(see for example Anderson, 1984), i.e. the covariance between x_i and x_j normalized by the geometric mean of the variances of x_i and x_j ($\bar{x} = E[x]$). Hence, the correlation is bounded, $-1 \leq \rho_{ij} \leq 1$, and the diagonal terms of a *correlation matrix*, i.e. a matrix of correlations, are one. The diagonal terms of $\mathbf{C}_{xx}$ in equation 2.22 are the second order *origin* moments, $E[x_i^2]$, of x_i. The diagonal terms in a *covariance matrix* are the variances or the second order *central* moments, $E[(x_i - \bar{x}_i)^2]$, of x_i.

The maximum likelihood estimator of ρ is obtained by replacing the expectation operator in equation 2.25 by a sum over the samples (Anderson, 1984). This estimator is sometimes called the *Pearson correlation coefficient* after Pearson (1896).

2.5.2 Competitive learning

In competitive learning there are several computational units competing to give the output. For a neural network, this means that among several units in the output layer only one will fire while the rest will be silent. Hence, they are often called *winner-take-all* units. Which unit fires depends on the input signal. The units specialize to react on certain stimuli and therefore they are sometimes called *grandmother cells*. This term was coined to illustrate the lack of biological plausibility for such highly specialized neurons. (There is probably not a single neuron in your brain waiting just to detect your grandmother.) Nevertheless, the most well-known implementation of competitive learning, the *self-organizing feature map* (SOFM) (Kohonen, 1982), is highly motivated by the topologically organized feature representations in the brain. For instance, in the visual cortex, line detectors are organized on a two-dimensional surface so that adjacent detectors for the orientation of a line are sensitive to similar directions (Hubel and Wiesel, 1962).

In the simplest case, competitive learning can be described as follows: Each unit gets the same input $\mathbf{x}$ and the winner is unit i if $\|\mathbf{w}_i - \mathbf{x}\| < \|\mathbf{w}_j - \mathbf{x}\|$, $\forall j \neq i$. A simple learning rule is to update the parameter vector of the winner according to

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(\mathbf{x}(t) - \mathbf{w}_i(t)), \tag{2.26}$$

i.e. to move the winning parameter vector towards the present input. The rest of the parameter vectors are left unchanged. If the output of the winning unit is one, equation 2.26 can be written as

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha y_i(\mathbf{x}(t) - \mathbf{w}_i(t)) \tag{2.27}$$

for all units (since $y_i = 0$ for all losers). Equation 2.27 is a modification of the Hebb rule in equation 2.21 and is identical to Oja's rule (equation 2.23) if $y_i \in \{0, 1\}$ (Hertz et al., 1991).

Vector quantization

A rather simple, but important, application of competitive learning is *vector quantization* (Gray, 1984). The purpose of vector quantization is to quantize a distribution of vectors $\mathbf{x}$ into N classes so that all vectors that fall into one class can be represented by a single prototype vector $\mathbf{w}_i$. The goal is to minimize the distortion between the input vectors $\mathbf{x}$ and the prototype vectors. The distortion measure is usually defined using a Euclidean metric:

$$D = \int_{\mathbb{R}^N} p(\mathbf{x}) \|\mathbf{x} - \mathbf{w}\|^2 d\mathbf{x}, \tag{2.28}$$

where $p(\mathbf{x})$ is the probability density function of $\mathbf{x}$.

Kohonen (1989) has proposed a modification to the competitive learning rule in equation 2.26 for use in classification tasks:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) \begin{cases} +\alpha(\mathbf{x}(t) - \mathbf{w}_i(t)) & \text{if correct classification} \\ -\alpha(\mathbf{x}(t) - \mathbf{w}_i(t)) & \text{if incorrect classification.} \end{cases} \tag{2.29}$$

The need for feedback from a teacher means that this is a *supervised* learning rule. It works as the standard competitive learning rule in equation 2.26 if the winning prototype vector represents the desired class but moves in the opposite direction if it does not. The learning rule is called *learning vector quantization* (LVQ) and can be used for classification. (Note that several prototype vectors can belong to the same class.)

Feature maps

The *self-organizing feature map* (SOFM) (Kohonen, 1982) is an unsupervised competitive learning rule but without winner-take-all units. It is similar to the vector quantization methods just described but has local connections between the prototype vectors. The standard update rule for a SOFM is

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha h(i,j)(\mathbf{x}(t) - \mathbf{w}_i(t)), \tag{2.30}$$

where $h(i,j)$ is a *neighbourhood function* which is dependent on the distance between the current unit vector i and the winner unit j. A common choice of $h(i,j)$ is a Gaussian. Note that the distance is not between the parameter vectors but between the units in a network. Hence, a topological ordering of the units is implied. Note also that all units, and not only the winner, are updated (although some of them with very small steps). The topologically ordered units and the neighbourhood function cause nearby units to have more similar prototype vectors than units far apart. Hence, if these parameter vectors are seen as feature detectors (i.e. filters), similar features will be represented by nearby units.

Equation 2.30 causes the parameter vectors to be more densely distributed in areas where the input probability is high and more sparsely distributed where the input probability is low. Such a behaviour is desired if the goal is to keep the distortion (equation 2.28) low. The density of parameter vectors is, however, *not* strictly proportional to the input signal probability (Ritter, 1991), which would minimize the distortion.

Higher level competitive learning

Competitive learning can also be used on a higher level in a more complex learning system. The function of the whole system is not necessarily based on unsu-

pervised learning. It can be trained using supervised or reinforcement learning. But the system can be divided into subsystems that specialize on different parts of the decision space. The subsystem that handle a certain part of the decision space best will gain control over that part. An example is the *adaptive mixtures of local experts* by Jacobs et al. (1991). They use a system with several local experts and a gating network that selects among the output of the local experts. The whole system uses supervised learning but the gating network causes the local experts to compete and therefore to try to take responsibility for different parts of the input space.

2.5.3 Mutual information based learning

The third principle of unsupervised learning is based on the concept of *mutual information*. Mutual information is gaining an increased attention in the signal processing society as well as among learning theorists and neural network researchers. The theory, however, dates back to 1948 when Shannon presented his classic foundations of information theory (Shannon, 1948).

A piece of information theory

Consider a discrete random variable $\mathbf{x}$:

$$\mathbf{x} \in \{\mathbf{x}_i\},\ i \in \{1, 2, \ldots, N\}. \tag{2.31}$$

(There is, in practice, no limitation in $\mathbf{x}$ being discrete since all measurements have finite precision.) Let $P(\mathbf{x}_k)$ be the probability of $\mathbf{x} = \mathbf{x}_k$ for a randomly chosen $\mathbf{x}$. The *information* content in the vector (or symbol) $\mathbf{x}_k$ is defined as

$$I(\mathbf{x}_k) = \log\left(\frac{1}{P(\mathbf{x}_k)}\right) = -\log P(\mathbf{x}_k). \tag{2.32}$$

If the basis 2 is used for the logarithm, the information is measured in *bits*. The definition of information has some appealing properties. First, the information is 0 if $P(\mathbf{x}_k) = 1$; if the receiver of a message knows that the message will be $\mathbf{x}_k$, he does not get any information when he receives the message. Secondly, the information is always positive. It is not possible to lose information by receiving a message. Finally, the information is additive, i.e. the information in two independent symbols is the sum of the information in each symbol:

$$\begin{aligned} I(\mathbf{x}_i, \mathbf{x}_j) &= -\log\left(P(\mathbf{x}_i, \mathbf{x}_j)\right) = -\log\left(P(\mathbf{x}_i)P(\mathbf{x}_j)\right) \\ &= -\log P(\mathbf{x}_i) - \log P(\mathbf{x}_j) = I(\mathbf{x}_i) + I(\mathbf{x}_j) \end{aligned} \tag{2.33}$$

if $\mathbf{x}_i$ and $\mathbf{x}_j$ are statistically independent.

The information measure considers each *instance* of the stochastic variable $\mathbf{x}$ but it does not say anything about the stochastic variable itself. This can be accomplished by calculating the average information of the stochastic variable:

$$H(\mathbf{x}) = \sum_{i=1}^{N} P(\mathbf{x}_i) I(\mathbf{x}_i) = -\sum_{i=1}^{N} P(\mathbf{x}_i) \log(P(\mathbf{x}_i)). \tag{2.34}$$

$H(\mathbf{x})$ is called the *entropy* of $\mathbf{x}$ and is a measure of *uncertainty* about $\mathbf{x}$.

Now, we introduce a second discrete random variable $\mathbf{y}$, which, for example, can be an output signal from a system with $\mathbf{x}$ as input. The *conditional entropy* (Shannon, 1948) of $\mathbf{x}$ given $\mathbf{y}$ is

$$H(\mathbf{x}|\mathbf{y}) = H(\mathbf{x},\mathbf{y}) - H(\mathbf{y}). \tag{2.35}$$

The conditional entropy is a measure of the average information in $\mathbf{x}$ given that $\mathbf{y}$ is known. In other words, it is the remaining uncertainty of $\mathbf{x}$ after observing $\mathbf{y}$. The *average mutual information*[4] $I(\mathbf{x};\mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is defined as the average information about $\mathbf{x}$ gained when observing $\mathbf{y}$:

$$I(\mathbf{x};\mathbf{y}) = H(\mathbf{x}) - H(\mathbf{x}|\mathbf{y}). \tag{2.36}$$

The mutual information can be interpreted as the difference between the uncertainty of $\mathbf{x}$ and the remaining uncertainty of $\mathbf{x}$ after observing $\mathbf{y}$. In other words, it is the reduction in uncertainty of $\mathbf{x}$ gained by observing $\mathbf{y}$. Inserting equation 2.35 into equation 2.36 gives

$$I(\mathbf{x};\mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}) - H(\mathbf{x},\mathbf{y}) = I(\mathbf{y};\mathbf{x}) \tag{2.37}$$

which shows that the mutual information is symmetric.

Now let $\mathbf{x}$ be a continuous random variable. Then the *differential entropy* $h(\mathbf{x})$ is defined as (Shannon, 1948)

$$h(\mathbf{x}) = -\int_{\mathbb{R}^N} p(\mathbf{x}) \log p(\mathbf{x})\, d\mathbf{x}, \tag{2.38}$$

where $p(\mathbf{x})$ is the probability density function of $\mathbf{x}$. The integral is over all dimensions in $\mathbf{x}$. The average information in a continuous variable would of course be infinite since there are an infinite number of possible outcomes. This can be seen if the discrete entropy definition (eq. 2.34) is calculated in limes when x approaches a continuous variable:

$$H(x) = -\lim_{\delta x \to 0} \sum_{i=-\infty}^{\infty} p(x_i)\delta x \log\left(p(x_i)\delta x\right) = h(x) - \lim_{\delta x \to 0} \log \delta x, \tag{2.39}$$

[4] Shannon (1948) originally used the term *rate of transmission*. The term *mutual information* was introduced later.

where the last term approaches infinity when δx approaches zero (Haykin, 1994). But since mutual information considers the difference in entropy, the infinite term will vanish and continuous variables can be used to simplify the calculations. The mutual information between the continuous random variables $\mathbf{x}$ and $\mathbf{y}$ is then

$$I(\mathbf{x};\mathbf{y}) = h(\mathbf{x}) + h(\mathbf{y}) - h(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^N}\int_{\mathbb{R}^M} p(\mathbf{x},\mathbf{y}) \log\left(\frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right) d\mathbf{x}d\mathbf{y}, \quad (2.40)$$

where N and M are the dimensionalities of $\mathbf{x}$ and $\mathbf{y}$ respectively.

Consider the special case of Gaussian distributed variables. The differential entropy of an N-dimensional Gaussian variable $\mathbf{z}$ is

$$h(\mathbf{z}) = \frac{1}{2}\log\left((2\pi e)^N |\mathbf{C}|\right) \quad (2.41)$$

where $\mathbf{C}$ is the covariance matrix of $\mathbf{z}$ (see proof B.1.1 on page 153). This means that the mutual information between two N-dimensional Gaussian variables is

$$I(\mathbf{x};\mathbf{y}) = \frac{1}{2}\log\left(\frac{|\mathbf{C}_{xx}|\,|\mathbf{C}_{yy}|}{|\mathbf{C}|}\right), \quad (2.42)$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{bmatrix}.$$

$\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ are the within-set covariance matrices and $\mathbf{C}_{xy} = \mathbf{C}_{yx}^T$ is the between-sets covariance matrix. For more details on information theory, see for example Gray (1990).

Mutual information based learning

Linsker (1988) showed that Hebbian learning gives maximum mutual information between the input and the output in a simple case with a linear unit with noise added to the output. In a more advanced model with several units, he showed that there is a tradeoff between keeping the output signals uncorrelated and suppressing the noise. Uncorrelated output signals give more information (higher entropy) on the output, but redundancy can help to suppress the noise. The principle of maximizing the information transferred from the input to the output is by Linsker (1988) called the *infomax principle*.

Linsker has proposed a method, based on maximum mutual information, for generating a topologically ordered feature map (Linsker, 1989). The map is similar to the SOFM mentioned in section 2.5.2 (page 27) but in contrast to the SOFM, Linsker's learning rule causes the distribution of input units to be proportional to the input signal probability density.

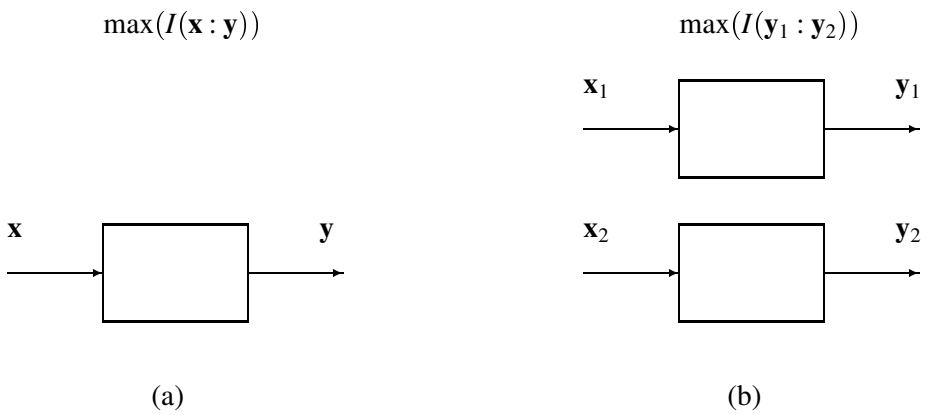

Figure 2.3: The difference between infomax (a) and Imax (b).

Bell and Sejnowski (1995) have used mutual information maximization to perform blind separation of mixed unknown signals and blind deconvolution of a signal convolved with an unknown filter. Actually, they maximize the entropy in the output signal $\mathbf{y}$ rather than explicitly maximizing the mutual information between $\mathbf{x}$ and $\mathbf{y}$. The results are, however, the same if there is independent noise in the output but no known noise in the input[5]. To see that, consider a system where $\mathbf{y} = f(\mathbf{x}) + \eta$ where η is an independent noise signal. The mutual information between $\mathbf{x}$ and $\mathbf{y}$ is then

$$I(\mathbf{x};\mathbf{y}) = h(\mathbf{y}) - h(\mathbf{y}|\mathbf{x}) = h(\mathbf{y}) - h(\eta), \tag{2.43}$$

where $h(\eta)$ is independent of the parameters of f.

Becker and Hinton (1992) have used mutual information maximization in another way than Linsker and Bell and Sejnowski. Instead of maximizing the mutual information between the input and the output they maximize the mutual information between the output of different units, see figure 2.3. They call this principle *Imax* and have used it to estimate disparity in random-dot stereograms (Becker and Hinton, 1992) and to detect depth discontinuities in stereo images (Becker and Hinton, 1993). A good overview of Imax is given by Becker (1996).

Among other mutual information based methods of unsupervised learning are Barlow's *minimum entropy coding* that aims at minimizing the statistical dependence between the output signals (Barlow, 1989; Barlow et al., 1989; Földiák, 1990) and the Gmax algorithm (Pearlmutter and Hinton, 1986) that tries to detect statistical dependent features in the input signal.

[5] "No known noise" means that the input cannot be divided into a signal part $\mathbf{x}$ and a noise part η . The noise is an indistinguishable part of the input signal $\mathbf{x}$.

The relation between mutual information and correlation

There is a clear relation between mutual information and correlation for Gaussian distributed variables. Consider two one-dimensional random variables x and y. Equations 2.42 and 2.25 then gives

$$I(x;y) = \frac{1}{2}\log\left(\frac{\sigma_x^2\sigma_y^2}{\sigma_x^2\sigma_y^2 - (\sigma_{xy})^2}\right) = \frac{1}{2}\log\left(\frac{1}{1-\rho_{xy}^2}\right), \tag{2.44}$$

where σ_x^2 and σ_y^2 are the variances of x and y respectively, σ_{xy} is the covariance between x and y and ρ_{xy} is the correlation between x and y. The extension of this relation to multidimensional variables is discussed in chapter 4.

This relationship means that for a single linear unit with Gaussian distributed variables, the mutual information between the input and the output, i.e. the amount of transferred information, is maximized if the correlation between the input and the output is maximized.

2.6 Comparisons between the three learning methods

The difference between supervised learning, reinforcement learning and unsupervised learning may seem very fundamental at first. But sometimes the distinction between them is not so clear and the classification of a learning method can depend upon the view of the observer.

As we have seen in section 2.4, reinforcement learning can be implemented as a supervised learning of the reward function. The output is then chosen as the one giving the maximum value of the approximation of the reward function given the present input.

Another way of implementing reinforcement learning is to use the output of the system as the desired output in a supervised learning algorithm and weight the update step with the reward (Williams, 1988).

Furthermore, supervised learning can emerge as a special case of reinforcement learning where the system is forced to give the desired output while receiving maximum reward. Also, a task for a supervised learning system can always be reformulated to fit a reinforcement learning system simply by mapping the error vectors to scalars, for example as a function of the norm of the error vectors.

Also unsupervised learning can sometimes be formulated as supervised learning tasks. Consider, for example, the PCA algorithms (section 2.5.1) that find the maximal eigenvectors of the distribution of $\mathbf{x}$. For a single parameter vector $\mathbf{w}$ the problem can be formulated as minimizing the difference between the signal $\mathbf{x}$ and

the output $\mathbf{y} = \mathbf{x}^T \hat{\mathbf{w}}\, \hat{\mathbf{w}}$, i.e.

$$\begin{aligned} \frac{1}{2} E\left[\|\mathbf{x} - \mathbf{x}^T \hat{\mathbf{w}}\, \hat{\mathbf{w}}\|^2\right] &= E\left[\mathbf{x}^T\mathbf{x} - \hat{\mathbf{w}}^T \mathbf{x}\mathbf{x}^T \hat{\mathbf{w}}\right] \\ &= \mathrm{tr}(\mathbf{C}) - \hat{\mathbf{w}}^T \mathbf{C}\hat{\mathbf{w}} = \sum_i \lambda_i - \hat{\mathbf{w}}^T \mathbf{C}\hat{\mathbf{w}}, \end{aligned} \tag{2.45}$$

where $\mathbf{C}$ is the covariance matrix of $\mathbf{x}$ (assuming $\bar{\mathbf{x}} = \mathbf{0}$) and λ_i are the eigenvalues of $\mathbf{C}$. Obviously, the best choice of $\mathbf{w}$ is the maximal eigenvector of $\mathbf{C}$. The output is a reconstruction of $\mathbf{x}$ and the desired output is the same as the input. Another example is the methods described in chapter 4 and by van der Burg (1988).

Finally, there is a similarity between all three learning principles in that they all generally try to optimize a scalar measure of performance, for example mean square error, accumulated reward, variance, or mutual information.

A good example illustrating how similar these three methods can be is the *prediction matrix memory* in section 3.3.1

2.7 Two important problems

There are some important fundamental problems in learning systems. One problem, called *perceptual aliasing*, deals with the problem of consistency in the internal representation of external states. Another problem is called the *credit assignment problem* and deals with the problem of distribution of the feedback in the system during learning. These two problems are discussed in this section. A third important problem is how to represent the information in a learning system, which is discussed in chapter 3.

2.7.1 Perceptual aliasing

Consider a learning system that perceives the external world through a sensory subsystem and represents the set of external states S_E by an internal state representation set S_I. This set can, however, rarely be identical to the real external world state set S_E. To assume a representation that completely describes the external world in terms of objects, their features and relationships, is unrealistic even for relatively simple problem settings. Furthermore, the internal state is inevitably limited by the sensor system, which leads to the fact that there is a many-to-many mapping between the internal and external states. That is, a state $\mathbf{s}_e \in S_E$ in the external world can map into several internal states and, what is worse, an internal state $\mathbf{s}_i \in S_I$ could represent multiple external world states. This phenomenon has been termed *perceptual aliasing* (Whitehead and Ballard, 1990a).

Figure 2.4 illustrates two cases of perceptual aliasing. One case is when two external states $\mathbf{s}_e^1$ and $\mathbf{s}_e^2$ map into the same internal state $\mathbf{s}_i^1$. An example is when

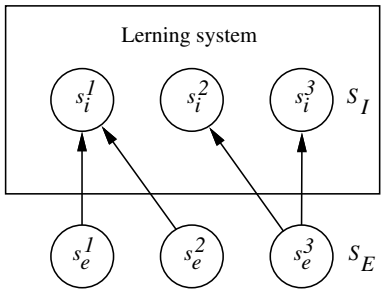

Figure 2.4: Two cases of perceptual aliasing. Two external states $\mathbf{s}_e^1$ and $\mathbf{s}_e^2$ are mapped into the same internal state $\mathbf{s}_i^1$ and one external state $\mathbf{s}_e^3$ is mapped into two internal states $\mathbf{s}_i^2$ and $\mathbf{s}_i^3$.

two different objects appear as identical to the system. This is illustrated in view 1 in figure 2.5. The other case is when one external state $\mathbf{s}_e^3$ is represented by two internal states $\mathbf{s}_i^2$ and $\mathbf{s}_i^3$. This happens, for instance, in a system consisting of several local adaptive models if two or more models happen to represent the same solution to the same part of the problem.

Perceptual aliasing may cause the system to confound different external states that have the same internal state representation. This type of problem can cause a response generating system to make the wrong decisions. For example, let the internal state $\mathbf{s}_i$ represent the external states $\mathbf{s}_e^a$ and $\mathbf{s}_e^b$ and let the system generate an action $\mathbf{a}$. The expected reward for the decision $(\mathbf{s}_i, \mathbf{a})$ to generate the action $\mathbf{a}$ given the state $\mathbf{s}_i$ can now be estimated by averaging the rewards for that decision accumulated over time. If $\mathbf{s}_e^a$ and $\mathbf{s}_e^b$ occur approximately equally often and the actual accumulated reward for $(\mathbf{s}_e^a, \mathbf{a})$ is greater than the accumulated reward for $(\mathbf{s}_e^b, \mathbf{a})$, the expected reward will be underestimated for $(\mathbf{s}_e^a, \mathbf{a})$ and overestimated for $(\mathbf{s}_e^b, \mathbf{a})$, leading to a non-optimal decision policy.

There are cases when the phenomenon is a feature, however. This happens if all decisions made by the system are *consistent*. The reward for the decision $(\mathbf{s}_i, \mathbf{a})$ then equals the reward for all corresponding actual decisions $(\mathbf{s}_e^k, \mathbf{a})$, where k is an index for this set of decisions. If the mapping between the external and internal worlds is such that all decisions are consistent, it is possible to collapse a large actual state space into a small one where situations that are invariant to the task at hand are mapped onto one single situation in the representation space. For a system operating in a large decision space, such a strategy is in fact necessary in order to reduce the number of different states. The goal is then to find a representation of the decision space such that consistent decisions can be found. The simplest example of such a deliberate perceptual aliasing is quantization. If

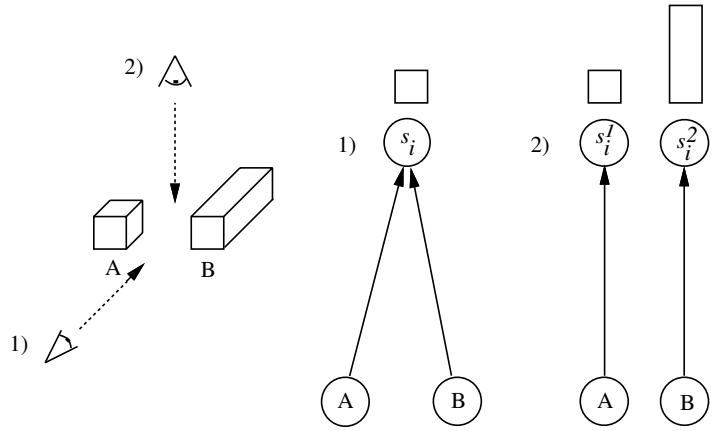

Figure 2.5: Avoiding perceptual aliasing by observing the environment from another direction.

the quantization is properly designed, the decisions will be consistent within each quantized state.

Whitehead and Ballard (1990b) have presented a solution to the problem of perceptual aliasing for a restricted class of learning situations. The basic idea is to detect inconsistent decisions by monitoring the estimated reward error, since the error will oscillate for inconsistent decisions as discussed above. When an inconsistent decision is detected, the system is guided (e.g. by changing its direction of view) to another internal state uniquely representing the desired external state. In this way, more actions will produce consistent decisions (see figure 2.5). The guidance mechanisms are not learned by the system. This is noted by Whitehead who admits that a dilemma is left unresolved:

> "In order for the system to learn to solve a task, it must accurately represent the world with respect to the task. However, in order for the system to learn an accurate representation, it must know how to solve the task."

The issue of information representation is further discussed in chapter 3.

2.7.2 Credit assignment

In all complex control systems, there probably exist some uncertainty of how to distribute credit (or blame) for the control actions taken. This uncertainty is called the *credit assignment problem* (Minsky, 1961, 1963). Consider, for example, a political system. Is it the trade politics or the financial politics that deserves credit

for the increasing export? We may call this a *structural* credit assignment problem. Is it the current government or the previous one that deserves credit or blame for the economic situation? This is a *temporal* credit assignment problem. Is it the management or the staff that should be given credit for the financial result in a company? This is what we may call a *hierarchical* credit assignment problem. These three types of credit assignment problems are also encountered in the type of control systems considered here, i.e. learning systems.

The structural credit assignment problem occurs, for instance, in a neural network when deciding which weights to alter in order to achieve an improved performance. In supervised learning, the structural credit assignment problem can be handled by using back-propagation (Rumelhart et al., 1986) for instance. The problem becomes more complicated in reinforcement learning where only a scalar feedback is available. In section 3.4, a description is given of how the structural credit assignment problem can be handled by the use of local adaptive models.

The temporal credit assignment problem occurs when a system acts in a dynamic environment and a sequence of actions is performed. The problem is to decide which of the actions taken deserves credit for the result. Obviously, it is not certain that it is the final action taken that deserves all the credit or blame. (For example, consider the situation when the losing team in a football game scores a goal during the last seconds of the game. It would not be clever to blame the person who scored that goal for the loss of the game.) The problem becomes especially complicated in reinforcement learning if the reward occurs infrequently. The temporal credit assignment problem is thoroughly investigated by Sutton (1984).

Finally, the hierarchical credit assignment problem can occur in a system consisting of several levels. Consider, for example, the *Adaptive mixtures of local experts* (Jacobs et al., 1991). That system consists of two levels. On the lower level, there are several subsystems that specialize on different parts of the input space. On the top level, there is a supervisor that selects the proper subsystem for a certain input. If the system makes a bad decision, it can be difficult to decide if it was the top level that selected the wrong subsystem or if the top level made a correct choice but the subsystem that generated the response made a mistake. This problem can of course be regarded as a type of structural credit assignment problem, but to emphasize the difference we call it a hierarchical credit assignment problem. Once the hierarchical credit assignment problem is solved and it is clear on what level the mistake was made, the structural credit assignment problem can be dealt with to alter the behaviour on that level.

Chapter 3

Information representation

A central issue in the design of learning systems is the representation of information in the system. The algorithms treated in this work can be seen as signal processing systems, in contrast to AI or expert systems that have symbolic representations[1]. We may refer to the representation used in the signal processing systems as a *continuous representation* while the symbolic approach can be said to use a *string representation*. Examples of the latter are the *Lion Algorithm* (Whitehead and Ballard, 1990a), the *Reinforcement Learning Classifier Systems* (Smith and Goldberg, 1990) and the MENACE example in section 2.4. The genetic algorithms that were described in section 2.4.3 are perhaps the most obvious examples of string representation in biological reinforcement learning systems.

The main difference between the two approaches is that a continuous representation has an implicit metric, i.e. there is a continuum of states and there exist meaningful interpolations between different states. One can say that two states are more or less similar. Interpolations are important in a learning system since they make it possible for the system to make decisions in situations never experienced before. This is often referred to as *generalization*. In a string representation there is no implicit metric, i.e. there is no unambiguous way to tell which of two strings is more similar to a third string than the other. There are, however, also advantages with string representations. Today's computers, for example, are designed to work with string representations and have difficulties in handling continuous information in an efficient way. A string representation also make it easy to include a priori knowledge in terms of explicit rules.

An approach that can be seen as a mix of symbolic representation and continuous representation is *fuzzy logic* (Zadeh, 1968, 1988). The symbolic expressions in fuzzy logic include imprecise statements like "many", "close to", "usually",

[1] By "symbolic", a more abstract representation is referred to than just a digitalization of the signal; a digital signal processing system is still a signal processing system.

etc. This means that statements need not be true or false; they can be somewhere in between. This introduces a kind of metric and interpolation is possible (Zadeh, 1988). Lee and Berenji (1989) describe a rule-based fuzzy controller using reinforcement learning that solves the pole balancing problem.

Ballard (1990) suggests that it is unreasonable to suppose that peripheral motor and sensory activity are correlated in a meaningful way. Instead, it is likely that abstract sensory and motor representations are built and related to each other. Also, combined sensory and motor information must be represented and used in the generation of new motor activity. This implies a learning hierarchy and that learning occurs on different temporal scales (Granlund, 1978, 1988; Granlund and Knutsson, 1982, 1983, 1990). Hierarchical learning system designs have been proposed by several other researchers (e.g. Jordan and Jacobs, 1994).

Both approaches (signal and symbolic) described on the preceding page are probably important, but on different levels in hierarchical learning systems. On a low level, the continuous representation is probably to prefer since signal processing techniques have the potential of being faster than symbolic reasoning as they are easier to implement with analogue techniques. On a low level, interpolations are meaningful and desirable. In a simple control task for instance, consider two similar[2] stimuli $\mathbf{s}_1$ and $\mathbf{s}_2$ which have the optimal responses $\mathbf{r}_1$ and $\mathbf{r}_2$ respectively. For a novel stimulus $\mathbf{s}_3$ located between $\mathbf{s}_1$ and $\mathbf{s}_2$, the response $\mathbf{r}_3$ could, with large probability, be assumed to be in between $\mathbf{r}_1$ and $\mathbf{r}_2$.

On a higher level, on the other hand, a more symbolic representation may be needed to facilitate abstract reasoning and planning. Here, the processing speed is not as crucial and interpolation may not even be desirable. Consider, for instance, the task of passing a tree. On a low level, the motor actions are continuous and meaningful to interpolate and they must be generated relatively fast. The higher level decision on which side of the tree to pass is, however, symbolic. Obviously, it is not successful to interpolate the two possible alternatives of "walking to the right" and "walking to the left". Also, there is more time to make this decision than to generate the motor actions needed for walking.

The choice of representation can be crucial for the ability to learn. Geman et al. (1992) argue that

> "the fundamental challenges in neural modelling are about representation rather than learning per se."

Furthermore, Hertz et al. (1991) present a simple but illustrative example to emphasize the importance of the representation of the input to the system. Two tasks are considered: the first one is to decide whether or not the input is an odd number;

[2] *Similar* means here that they are relatively close to each other in the given metric compared to the variance of the distribution of stimuli.

the second is to decide if the input has an odd number of prime factors. If the input has a binary representation, the first task is extremely simple: the system just has to look at the least significant bit. The second task, however, is very difficult. If the base is changed to 3, for instance, the first task will be much harder. And if the input is represented by its prime factors, the second task will be easier. Hertz et al. (1991) also prove an obvious (and, as they say, silly) theorem:

"learning will always succeed, given the right preprocessor."

In the discussion above, representation of two kinds of information is actually treated: the information entering the system as input signals (signal representation) and the information in the system about how to behave, i.e. knowledge learned by the system (model representation). The representations of these two kinds of information are, however, closely related to each other. As we will see, a careful choice of input signal representation can allow for a very simple representation of knowledge.

In the following section, a special type of signal representation called the *channel representation* is presented. It is a representation that is biologically inspired and which has several computational advantages. The later sections will deal more with model representations. The probably most well-known class of model representations among learning systems, *neural networks*, is presented in section 3.2. They can be seen as global non-linear models. In section 3.3 is shown how the channel representation makes it possible to use a simple linear model. In section 3.4 is argued that low-dimensional linear models are sufficient if they are local enough and the adaptive distribution of such models is briefly discussed in section 3.5. The chapter ends with simple examples of reinforcement learning systems solving the same problem but with different representations.

3.1 The channel representation

As has been discussed above, the internal representation of information may play a decisive role for the performances of learning systems. The representation that is intuitively most obvious in a certain situation, for example a scalar t for temperature or a three dimensional vector $\mathbf{p} = (x\ y\ z)^T$ for a position in space, is, however, in some cases not a very good way to represent information. For example, consider an orientation in $\mathbb{R}^2$ which can be represented by an angle $\varphi \in [-\pi, \pi]$ relative to a fix orientation, for example the x-axis. While this may appear as a very natural representation of orientation, it is in fact not a very good one since it has got a discontinuity at π which means that an orientation average cannot be consistently defined (Knutsson, 1989).

Another, perhaps more natural, way of representing information is the *channel representation* (Nordberg et al., 1994; Granlund, 1997). In this representation, a

set of channels is used where each channel is sensitive to some specific feature value in the signal, for example a certain temperature t_i or a certain position p_i. In the example above, the orientation in $\mathbb{R}^2$ could be represented by a set of channels evenly spread out on the unit circle, as proposed by Granlund (1978). If three channels of the shape

$$c_k = \cos^2\left(\frac{3}{4}(\varphi - p_k)\right), \tag{3.1}$$

where $p_1 = \frac{2\pi}{3}, p_2 = 0$ and $p_3 = -\frac{2\pi}{3}$, are used (Knutsson, 1982), the orientation can be represented continuously by the channel vector $\mathbf{c} = (c_1 \; c_2 \; c_3)^T$ which has a constant norm for all orientations. The reason to call this a more natural representation than for instance the angle φ, is that the channel representation is frequently used in biological systems, where each nerve cell responds strongly to a specific feature value. One example of this is the orientation sensitive cells in the primary visual cortex (Hubel and Wiesel, 1959; Hubel, 1988). This representation is called *value encoding* by Ballard (1987) who contrasts it with *variable encoding* where the activity is monotonically increasing with some parameter.

Theoretically, the channels can be designed so that there is one channel for each feature value that can occur. A function of these feature values would then be implemented simply as a look-up table. In practice, however, the range of feature values is often continuous (or at least quantized finely enough to be considered continuous). Each channel can be seen as a response of a filter that is tuned to some specific feature value. The coding is then designed so that the channel has its maximum value (for example one) when the feature and the filter are exactly tuned to each other, and decreases to zero in a smooth way as the feature and the filter become less similar. This is similar to the magnitude representation proposed by Granlund (1989).

The channel representation increases the number of dimensions in the representation. It should, however, be noted that an increase in the dimensionality does not have to lead to increased complexity of the learning problem. A great advantage of the channel representation is that it allows for simple processing structures. To see this, consider any continuous function $y = f(x)$. If x is represented by a sufficiently large number of channels c_k of a suitable form, the output y can simply be calculated as a weighted sum of the input channels $y = \mathbf{w}^T\mathbf{c}$ however complicated the function f may be. This implies that by using a channel representation, linear operations can be used to a great extent; this fact is used further in this chapter.

It is not obvious how to choose the shape of the channels. Consider, for example, the coding of a variable x into channels. According to the description above, each channel is positive and has its maximum for one specific value of x and it decreases smoothly to zero away from this maximum. In addition, to enable representation of all values of x in an interval, there must be overlapping channels on

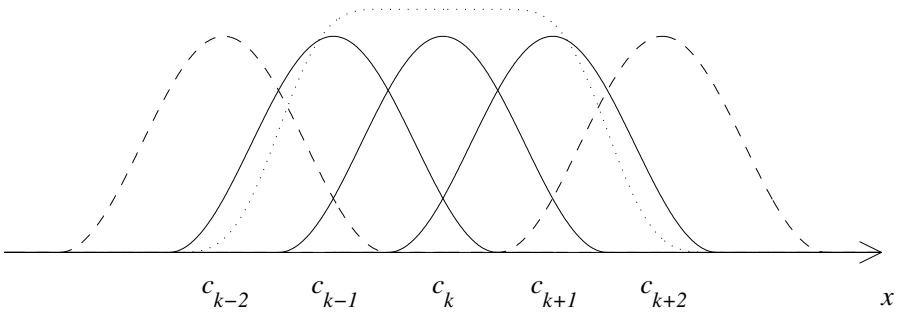

Figure 3.1: A set of $\cos^2$ channels. Only three channels are activated simultaneously. The sum of the squared channel outputs c_{k-1}, c_k and c_{k+1} is drawn with a dotted line.

this interval. It is also convenient if the norm of the channel vector is constant so that the feature value is only represented by the *orientation* of the channel vector. This enables the use of the scalar product for calculating the similarity between values. It also makes it possible to use the norm of the channel vector to represent some other entity related to the measurement, for instance the energy or the certainty of the measurement. One channel form that fulfils the requirements above is:

$$c_k = \begin{cases} \cos^2\left(\frac{\pi}{3}(x-k)\right) & |x-k| < \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

(see figure 3.1). This set of channels has a constant norm (see proof B.2.1 on page 154). It also has a constant square sum of its first derivatives (see proof B.2.2 on page 155) (Knutsson, 1982, 1985). This means that a change Δx in x always gives the same change $\Delta\mathbf{c}$ in $\mathbf{c}$ for any x. Of course, not only scalars can be coded into vectors with constant norm. Any vector $\mathbf{v}$ in a vector space of $(N-1)$ dimensions can be transformed into the orientation of a unit-length vector in an N-dimensional space. This was used, for example, by Denoeux and Lengellé (1993) in order to keep the norm of the input vectors constant and equal to one while preserving all the information. By using this new input representation, a scalar product could be used for calculating the similarity between the input vectors and a set of *prototype* vectors.

The channel vectors described above only exist in a small limited number of dimensions at a time; the channels in all other dimensions are zero. The number of simultaneously active channels is called *local dimensionality*. In the example in figure 3.1, the local dimensionality is three. This means that the vector moves along a curve as in figure 3.2 (left) as x changes. If we look at channels far apart,

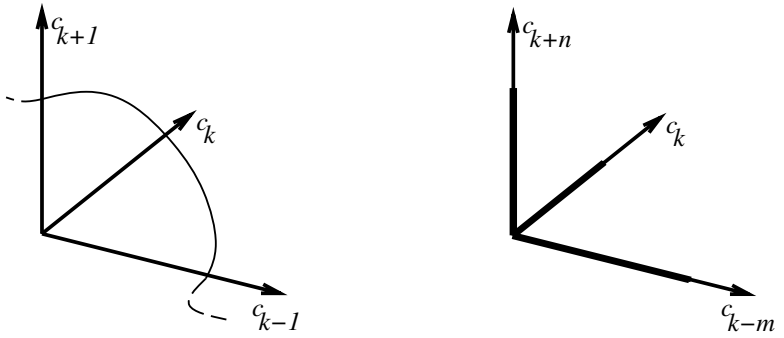

Figure 3.2: **Left:** The curve along which a channel vector can move in a subspace spanned by three neighbouring channels. The broken part of the curve illustrates the proceeding of the vector into other dimensions. **Right:** The possible channel vectors viewed in a subspace spanned by three distant non-overlapping channels.

only one of these channels is active at a time (figure 3.2, right); the activity is *local*. We call this type of channel vector a *pure* channel vector. The pure channel vector can be seen as an extreme of the *sparse distributed coding* (Field, 1994). This is a coding that represents data with a minimum number of *active* units in contrast to *compact coding* that represents data with a minimum number of units.

In general, the input to a system cannot be a pure channel vector. Consider, for example, a system that uses visual input, i.e. images. It is obvious that the dimensionality of the space of pure channel vectors that can represent all images would be far to large to be of practical interest. The input should rather consist of many sets of channels where each set measures a local property in the image, for example local orientation. Each set can be a pure channel vector, but the total input vector, consisting of several concatenated pure channel vectors, will not only have local activity. We call this type of vector, which consists of many sets of channels, a *mixed* channel vector.

The use of mixed channel vectors is not only motivated by limited processing capacity. Consider, for example, the representation of a two-dimensional variable $\mathbf{x} = (x_1\ x_2)^T$. We may represent this variable with a pure channel vector by distributing on the X-plane overlapping channels that are sensitive to different $\mathbf{x}_i$, as in figure 3.3 (left). Another way is to represent $\mathbf{x}$ with a mixed channel vector by using two sets of channels as in figure 3.3 (right). Here, each set is only sensitive

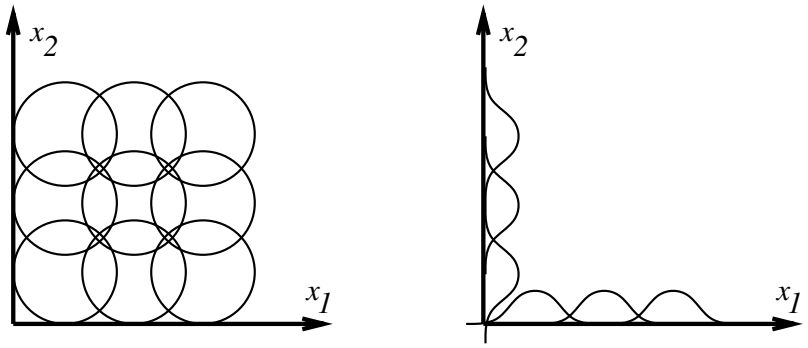

Figure 3.3: **Left:** Representation of a two-dimensional variable with one set of channels that constitute a pure channel vector **Right:** Representation of the same variable with two sets of channels that together form a mixed channel vector.

to one of the two parameters x_1 and x_2 and it does not depend on the other parameter at all; the channel vector $\mathbf{c}_1$ on the x_1-axis is said to be *invariant* with respect to x_2. Invariance can be seen as a deliberate perceptual aliasing as discussed in section 2.7.1. If x_1 and x_2 represent different *properties* of $\mathbf{x}$, for instance colour and size, the invariance can be a very useful feature. It makes it possible to observe one property independently of the others by looking at a subset of the channels. Note, however, that this does not mean that *all* multidimensional variables should be represented by mixed channel vectors. If, for example, $(x_1\ x_2)^T$ in figure 3.3 represents the two-dimensional position of a physical object, it does not seem useful to see the x_1 and x_2 positions as two different properties. In this case, the pure channel vector (left) might be a proper representation.

The use of mixed channel vectors offers another advantage compared to using the original variables, namely the simultaneous representation of properties which belong to different objects. Consider a one-dimensional variable x representing a position of an object along a line and compare this with a channel vector $\mathbf{c}$ representing the same thing. Now, if *two* objects occur at different positions, a mixed channel vector allows for the positions of *both* objects to be represented. This is obviously not possible when using the single variable x. Note that the mixed channel vector discussed here differs from the one described previously which consists of two or more concatenated pure channel vectors. In that case, the mixed channel vector represents several features and one instance of each feature. In the case of representing two or more positions, the mixed channel vector represents several

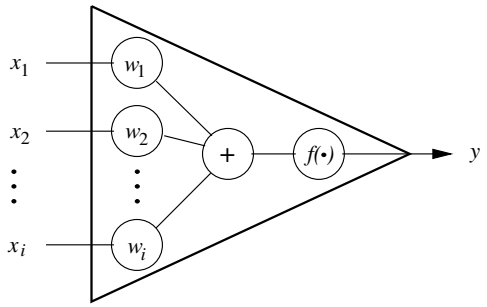

Figure 3.4: The basic neuron. The output y is a non-linear function f of a weighted sum of the inputs x.

instances of the same feature, i.e. multiple events. Both representations are, however, mixed channel vectors in the sense that they can have simultaneous activity on channels far apart as opposed to pure channel vectors.

3.2 Neural networks

Neural networks are perhaps the most popular and well-known implementations of artificial learning systems. The concept is so popular that it is often used synonymous with machine learning, which sometimes can be a bit misleading. There is no unanimous definition of neural networks, but they are usually characterized by a large number of massively connected relatively simple processing units. Learning capabilities are often understood even if they are not explicit. One could of course imagine a hard-wired neural network incapable of learning. Neural networks can be seen as global parameterized non-linear models.

The processing units in a neural network are often called *neurons* (hence, the name neural network) since they were originally designed as models of the nerve cells (neurons) in the brain. In figure 3.4, an artificial neuron is illustrated. This basic model of an artificial neuron was proposed by McCulloch and Pitts (1943) where the non-linear function f was a Heaviside (unit step) function, i.e.

$$f(x) = \begin{cases} 0 & x < 0 \\ (1/2 & x = 0) \\ 1 & x > 0 \end{cases}. \tag{3.3}$$

An example of a neural network is the *two-layer perceptron* illustrated in figure 3.5 which consists of neurons like the one described above connected in a feed-forward manner. The neural network is a parameterized model and the parameters

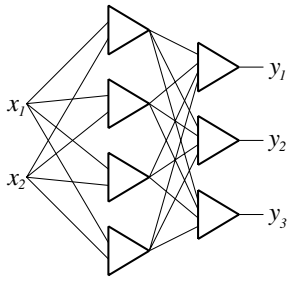

Figure 3.5: A two-layer perceptron with a two-dimensional input and a three-dimensional output.

are often called *weights*. Rosenblatt (1962) presented a supervised learning algorithm for a single layer perceptron. Later, however, Minsky and Papert (1969) showed that a single layer perceptron failed to solve even some simple problems, for example the Boolean exclusive-or function. While it was known that a three-layer perceptron can represent *any* continuous function, Minsky and Papert doubted that a learning method for a multi-layer perceptron would be possible to find. This finding almost extinguished the interest in neural networks for nearly two decades until the 1980s when learning methods for multi-layer perceptrons were developed. The most well-known method is *back-propagation* presented in a Ph.D. thesis by Werbos (1974) and later presented by Rumelhart et al. (1986).

The solution to the problem of how to update a multi-layer perceptron was to replace the Heaviside function (equation 3.3) with a differentiable nonlinear function, usually a sigmoid function. Examples of common sigmoid functions are $f(x) = \tanh(x)$ and the Fermi function:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{3.4}$$

The sigmoid function can be seen as a *basis function* for the internal representation in the network. Another choice of basis function is the *radial-basis function* (RBF), for example a Gaussian, that is used in the input layer in RBF networks (Broomhead and Lowe, 1988; Moody and Darken, 1989). The RBFs can be seen as a kind of channel representation.

The feed-forward design in figure 3.5 is, of course, not the only possible arrangement of neurons in a neural network. It is also possible to have connections from the output back to the input, so called *recurrent networks*. Two famous examples of recurrent networks are the *Hopfield network* (Hopfield, 1982) and the *Boltzmann machine* (Hinton and Sejnowski, 1983, 1986).

3.3 Linear models

While neural networks are non-linear models, it could sometimes be sufficient to use a linear model, especially if the representation of the input to the system is chosen carefully. As mentioned above, the channel representation makes it possible to realize a rather complicated function as a linear function of the input channels. In fact, the RBF networks can be seen as a hidden layer creating a channel representation followed by an output layer implementing a linear model.

In this section, a linear model for reinforcement learning called the prediction matrix memory is presented.

3.3.1 The prediction matrix memory

In this subsection, a system that is to learn to produce an output channel vector $\mathbf{q}$ as a function of an input channel vector $\mathbf{v}$ is described. The functions considered here are continuous functions of a *pure* channel vector (see page 42) or functions that are dependent on *one* property while invariant with respect to the others in a mixed channel vector; in other words, functions that can be realized by letting the output channels be linear combinations of the input channels. We call this type of functions *first-order* functions[3]. The order can be seen as the number of events in the input vector that must be considered simultaneously in order to define the output. In practice, this means that, for instance, a first-order function does not depend on any relation between different events; a second-order function depends on the relation between no more than two events and so on.

Consider a first-order system which is supplied with an input channel vector $\mathbf{v}$ and which generates an output channel vector $\mathbf{q}$. Suppose that $\mathbf{v}$ and $\mathbf{q}$ are pure channel vectors. If there is a way of defining a scalar r (the reinforcement) for each decision (**v, q**) (i.e. input-output pair), the function $r(\mathbf{v}, \mathbf{q})$ is a second-order function. The tensor space $\mathcal{Q} \otimes \mathcal{V}$ that contains the outer products $\mathbf{q}\mathbf{v}^T$ we call the *outer product decision space*. In this space, the decision (**v, q**) is *one* event. Hence, r can be calculated as a first-order function of the outer product $\mathbf{q}\mathbf{v}^T$.

In practice, the system will, of course, handle a finite number of overlapping channels and r will only be an approximation of the reward. But if the reward function is continuous, this approximation can be made arbitrarily good by using a sufficiently large set of channels.

[3] This concept of order have similarities to the one defined by Minsky and Papert (1969). In their discussion, the inputs are binary vectors which of course can be seen as mixed channel vectors with non-overlapping channels.

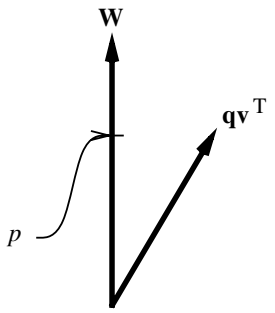

Figure 3.6: The reward prediction p for a certain stimulus-response pair $(\mathbf{v}, \mathbf{q})$ viewed as a projection onto $\mathbf{W}$ in $Q \otimes V$.

Learning the reward function

If supervised learning is used, the linear function could be learned by training a weight vector $\tilde{\mathbf{w}}_i$ for each output channel q_i so that $q_i = \tilde{\mathbf{w}}^T \mathbf{v}$. This could be done by minimizing some error function, for instance

$$\mathcal{E} = E[\|\mathbf{q} - \tilde{\mathbf{q}}\|^2], \tag{3.5}$$

where $\tilde{\mathbf{q}}$ is the correct output channel vector supplied by the teacher. This means, for the whole system, that a matrix $\widetilde{\mathbf{W}}$ is trained so that a correct output vector is generated as

$$\mathbf{q} = \widetilde{\mathbf{W}}\mathbf{v} = \begin{pmatrix} \widetilde{\mathbf{w}}_i^T \mathbf{v} \\ \vdots \end{pmatrix}. \tag{3.6}$$

In reinforcement learning, however, the correct output is unknown; only a scalar r that is a measure of the performance of the system is known (see section 2.4 on page 12). But the reward is a function of the stimulus and the response, at least if the environment is not completely stochastic. If the system can learn this function, the best response for each stimulus can be found. As described above, the reward function for a first-order system can be approximated by a linear combination of the terms in the outer product $\mathbf{q}\mathbf{v}^T$. This approximation can be used as a prediction p of the reward and is calculated as

$$p = \langle \mathbf{W} \mid \mathbf{q}\mathbf{v}^T \rangle, \tag{3.7}$$

see figure 3.6. The matrix $\mathbf{W}$ is therefore called a *prediction matrix memory*. The reward function can be learned by modifying $\mathbf{W}$ in the same manner as in

supervised learning, but here with the aim to minimize the error function

$$\mathcal{E} = E[|r - p|^2]. \tag{3.8}$$

Now, let each triple $(\mathbf{v}, \mathbf{q}, r)$ of stimulus, response, and reward denote an *experience*. Consider a system that has been subject to a number of experiences. How should a proper response be chosen by the system? The prediction p in equation 3.7 can be rewritten as

$$p = \mathbf{q}^T \mathbf{W}\mathbf{v} = \langle \mathbf{q} \mid \mathbf{W}\mathbf{v} \rangle. \tag{3.9}$$

Due to the channel representation, the actual output is completely determined by the *direction* of the output vector. Hence, we can regard the norm of $\mathbf{q}$ as fixed and try to find an optimal direction of $\mathbf{q}$. The $\mathbf{q}$ that gives the highest predicted reward obviously has the same direction as $\mathbf{W}\mathbf{v}$. Now, if p is a good prediction of the reward r for a certain stimulus $\mathbf{v}$, this choice of $\mathbf{q}$ would be the one that gives the highest reward. An obvious choice of the response $\mathbf{q}$ is then

$$\mathbf{q} = \mathbf{W}\mathbf{v} \tag{3.10}$$

which is the same first-order function as $\widetilde{\mathbf{W}}$ suggested for supervised learning in equation 3.6. Since $\mathbf{q}$ is a function of the input $\mathbf{v}$, the prediction can be calculated directly from the input. Equation 3.9 together with equation 3.10 give the prediction as

$$p = (\mathbf{W}\mathbf{v})^T \mathbf{W}\mathbf{v} = \|\mathbf{W}\mathbf{v}\|^2. \tag{3.11}$$

Now we have a very simple processing structure (essentially a matrix multiplication) that can generate proper responses *and* predictions of the associated rewards for any first-order function.

This structure is similar to the *learning matrix* or *correlation matrix memory* described by Steinbuch and Piske (1963) and later by Anderson (1972, 1983) and by Kohonen (1972, 1989). The correlation matrix memory is a kind of linear associative memory that is trained with a generalization of Hebbian learning (Hebb, 1949). An associative memory maps an input vector $\mathbf{a}$ to an output vector $\mathbf{b}$, and the correlation matrix memory stores this mapping as a sum of outer products:

$$\mathbf{M} = \sum \mathbf{b}\mathbf{a}^T. \tag{3.12}$$

The stored patterns are then retrieved as

$$\mathbf{b} = \mathbf{M}\mathbf{a} \tag{3.13}$$

which is equal to equation 3.10. The main difference is that in the method described here, the correlation strength is retrieved and used as a prediction of the reward. Kohonen (1972) has investigated the selectivity and tolerance with respect to destroyed connections in the correlation matrix memories.

The training of the matrix $\mathbf{W}$ is a very simple algorithm. For a certain experience $(\mathbf{v}, \mathbf{q}, r)$, the prediction p should, in the optimal case, equal r. This means that the aim is to minimize the error in equation 3.8. The desired weight matrix $\mathbf{W}'$ would yield a prediction

$$p' = r = \langle \mathbf{W}' \mid \mathbf{q}\mathbf{v}^T \rangle. \tag{3.14}$$

Since this is a linear problem, it could be tempting to solve it analytically. This could be done recursively using the recursive least squares (RLS) method (Ljung, 1987). The problem is that RLS involves the estimation and inversion of a $p \times p$ matrix where $p = \dim(\mathbf{q})\dim(\mathbf{v})$. Since the dimensionalities of $\mathbf{q}$ and $\mathbf{v}$ are high in general due to the channel representation, RLS is not a very useful tool in this case. Instead, we use stochastic gradient search (see section 2.3.1 on page 10) to find $\mathbf{W}'$. From equations 3.7 and 3.8 we get the error

$$\varepsilon = |r - \langle \mathbf{W} \mid \mathbf{q}\mathbf{v}^T \rangle|^2 \tag{3.15}$$

and the gradient is

$$\frac{\partial \varepsilon}{\partial \mathbf{W}} = -2(r-p)\mathbf{q}\mathbf{v}^T. \tag{3.16}$$

To minimize the error, $\mathbf{W}$ should be changed a certain amount a in the direction $\mathbf{q}\mathbf{v}^T$, i.e.

$$\mathbf{W}' = \mathbf{W} + a\mathbf{q}\mathbf{v}^T. \tag{3.17}$$

Equation 3.14 now gives that

$$r = p + a\|\mathbf{q}\|^2\|\mathbf{v}\|^2 \tag{3.18}$$

(see proof B.2.3 on page 156) which gives

$$a = \frac{r-p}{\|\mathbf{q}\|^2\|\mathbf{v}\|^2}. \tag{3.19}$$

To perform stochastic gradient search (equation 2.6 on page 11), we change the parameter vector a small step in the negative gradient direction for each iteration. The update rule therefore becomes

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \Delta\mathbf{W}(t), \tag{3.20}$$

where

$$\Delta \mathbf{W} = \alpha \frac{r-p}{\|\mathbf{q}\|^2 \|\mathbf{v}\|^2} \mathbf{q}\mathbf{v}^T, \tag{3.21}$$

where α is the update factor $(0 < \alpha \leq 1)$ (see section 2.3.2 on page 11). If the channel representation is chosen so that the norm of the channel vectors is constant and equal to one, this equation is simplified to

$$\Delta \mathbf{W} = \alpha(r-p)\mathbf{q}\mathbf{v}^T. \tag{3.22}$$

Here, the difference between this method and the correlation matrix memory becomes clearer. The learning rule in equation 3.12 corresponds to that in equation 3.22 with $\alpha(r-p) = 1$. The prediction matrix $\mathbf{W}$ in equation 3.22 will converge when $r = p$, while the correlation matrix $\mathbf{M}$ in equation 3.12 would grow for each iteration unless a normalization procedure is used.

Here, we can see how reinforcement learning and supervised learning can be combined, as mentioned in section 2.6. By setting $r = p+1$ and $\alpha = 1$ we get the update rule for the correlation matrix memory in equation 3.12, and with $r = 1$ we get a correlation matrix memory with a converging matrix. This means that if the correct response is known, it can be learned using supervised learning by forcing the output to the correct response and setting the parameters $\alpha = 1$ and $r = 1$ or $r = p+1$. When the correct response is *not* known, the system is let to produce the response and the reinforcement learning algorithm described above can be used.

Relation to Q-learning

The description above of the learning algorithm assumed a reinforcement signal as a feedback to the system for each single decision (i.e. stimulus-response pair). This is, however, not necessary. Instead of learning the instantaneous reward function $r(\mathbf{x}, \mathbf{y})$, the system can be trained to learn the Q-function $Q(\mathbf{x}, \mathbf{y})$ (equation 2.13 on page 19), which can be written as

$$Q(\mathbf{x}(t), \mathbf{y}(t)) = r(\mathbf{x}(t), \mathbf{y}(t)) + \gamma Q(\mathbf{x}(t+1), \mathbf{y}(t+1)), \tag{3.23}$$

where γ is a prediction decay factor $(0 < \gamma \leq 1)$ that makes the predicted reinforcement decay as the distance from the actual rewarded state increases. Now the right-hand side of equation 3.23 can be used instead of r in equation 3.22 as the desired prediction. This gives

$$\Delta \mathbf{W} = \alpha(r(t) + \gamma p(t+1) - p(t))\mathbf{q}\mathbf{v}^T. \tag{3.24}$$

This means that the system can handle dynamic problems with infrequent reinforcement signals by maximizing the long-term reward function.

In one sense, this system is better suited for the use of TD-methods than the systems mentioned in section 2.4.1 on page 17, since they have to use separate subsystems to calculate the predicted reinforcement. With the algorithm suggested here, this prediction is calculated by the same system as the response.

3.4 Local linear models

Global linear models (e.g. the prediction matrix memory) can of course not be used for all problems. The number of dimensions required for a pure channel representation would in general be far too high. But a global non-linear model (e.g. a neural network) is in general not a solution. The number of parameters in a global non-linear model would be far too high to be possible to estimate with a low variance using a reasonable number of samples. The rescue in this situation is that we generally do not need a global model at all.

Consider a system with a visual input consisting only of a binary[4] image with 8×8 pixels (picture elements), which is indeed a limited visual sensor. There are $2^{64} > 10^{19}$ possible different binary 8×8 images. If they were displayed with a frame rate of 50 frames per second, it would take about 10 billion years to view them all, a period of time that is about the same as the age of the universe!

It is quite obvious that most of the possible events in a high-dimensional space will never occur during the lifetime of a system. In fact, only a very small fraction of the signal space will ever be visited by the signal. Furthermore, the environment that causes the input signals is limited by the dynamic of the outside world and this dynamic put restrictions on how the input signal can move. This means that the high-dimensional input signal will move on a low-dimensional subspace (Landelius, 1997) and we do not have to search for a global model for the whole signal space (at least if a proper representation is used).

The low dimensionality can intuitively be understood if we consider a signal consisting of N frequency components. Such a signal can span at most a $2N$-dimensional space since each frequency component defines an ellipse and hence spans at most a two-dimensional plane (Johansson, 1997) (see proof B.2.4 on page 156). In the case of images, this is expressed in the *assumption of local one-dimensionality* (Granlund, 1978; Granlund and Knutsson, 1995):

> "The content within a window, measured at a sufficiently small bandwidth, will as a rule have a single dominant component."

[4] Binary, in this case, means that each pixel can only have two different values, e.g. black or white.

By this reasoning, it is sufficient to have a model or a set of models that covers the manifold where the signal exists (Granlund and Knutsson, 1990). If the signal manifold is continuous in space and time (which is reasonable due to the dynamic of the outside world), the low-dimensional manifold could locally be approximated with a linear subspace (Bregler and Omohundro, 1994; Landelius, 1997).

Since we are dealing with learning systems, the local models should be adaptive. In this context, low-dimensional linear local models have several advantages. First of all, the number of parameters in a low-dimensional linear model is low, which reduces the number of samples needed for estimating the model compared to a global model. This is necessary since the locality constraint limits the number of available samples. Moreover, the locality reduces the spatial credit assignment problem (section 2.7.2, page 35) since the adaptation of one local model will in general not have any major effects on the other models (Baker and Farell, 1992).

How the local linear models should be chosen, i.e. according to what criteria the models' adaptation should be optimized, depends of course on the task. A method for estimating local linear models for four different criteria is presented in chapter 4.

3.5 Adaptive model distribution

In the previous section was argued that the signal distribution in a learning system with high-dimensional input should be modelled with local adaptive models. This raises the question of how to distribute these local models. The simplest way is, of course, to divide the signal space into a number of regions (e.g. N-dimensional boxes) and put an adaptive model in each region. Such an approach is, however, not very efficient since, as have been discussed above, most of the space will be empty and, hence, most models will never be used. Moreover, with such an approach, parts of the signal that could be modelled using one single model would make use of several models due to the pre-defined subdivision. This would cause each of these models to be estimated using a smaller number of samples than would be the case if a single model was used and hence this would cause an unnecessary uncertainty in the parameter estimation. Finally, the pre-defined subdivision cannot be guaranteed to be fine enough in areas where the signal has a complicated behaviour.

An obvious solution to this problem is to make the model distribution adaptive. First of all, such an approach would only put models where the signal really exists. Furthermore, an adaptive model distribution makes it possible to distribute models sparsely where the signal has a smooth behaviour and more densely where it has not.

An example of adaptive distribution of local linear models is given by Ritter et al. (1989, 1992) who use a SOMF (Kohonen, 1982) (see section 2.5.2, page 27) to distribute local linear models (Jacobian matrices) in a robot positioning task. Other methods are discussed by Landelius (1997) who sugests linear or quadratic models and Gaussian applicability funcitons organized in a tree structure (see also Landelius et al., 1996). The applicability functions define the regions where the local models are valid. In the system by Ritter et al., the applicability functions are defined by a winner-take-all rule for the units in the SOFM (page 27).

Just as in the case of estimating the model parameters, the adaptive model distribution is task dependent. If, for example, the goal of the system is to achieve maximum reward, the models should be positioned where they are as useful as possible for getting that reward and if the goal is maximum information transmission, the models should be positioned according to this goal. Hence, no general rule can be given for how to adaptively distribute local models. One can only state that the goal must be to optimize the same criteria as the local models are trying to optimize together. This implies that the choice of models and the distribution of them are dependent on each other. The simpler a model is, i.e. the less parameters it has, the smaller the region will be where it is valid and, hence, the larger the number of models required. This does not mean, however, that a small number of more global complex models is as good as a large number of simpler and more local models, even if the total number of parmeters is the same. As mentioned above (section 3.4), the locality in the latter approach reduces the spatial credit assignment problem and, hence, facilitates efficient learning.

3.6 Experiments

This chapter ends with two simple examples of reinforcement learning with different representations. The first one uses the channel representation described in section 3.1 and the prediction matrix memory from section 3.3.1 for learning the Q-function. The second example is a TD-method that uses local adaptive linear models both to represent the input-ouput function and to approximate the V-function. This algorithm was presented at the ICANN'93 in Amsterdam (Borga, 1993).

The experiment is made up of a system that plays "badminton" with itself. For simplicity, the problem is one-dimensional. The position of the shuttlecock is represented by a variable x. The system can change the value of x by adding the output value y to x. A small noise is also added to punish playing on the margin. The reinforcement signal to the system is zero except upon failure when $r = -1$. Failure is the case when x does not change sign (i.e. the shuttlecock does not pass the net), or when $|x| > 0.5$ (i.e. the shuttlecock ends up outside the court).

3.6.1 Q-learning with the prediction matrix memory

The position x is represented by 25 $\cos^2$-channels in the interval $-0.6 < x < 0.6$ and the ouput y is represented by 45 $\cos^2$-channels in the interval $-1.1 < x < 1.1$. The channels have the shape defined in equation 3.2, illustrated in figure 3.1 on page 41. An offset value of one was added to the reinforcement signal, i.e. $r = 1$ except upon failure when $r = 0$, since the prediction matrix memory must contain positive values.

The prediction matrix memory was trained to learn the Q-function as defined in equation 3.23 with the discount factor $\gamma = 0.9$. The matrix was updated according to the update rule in equations 3.20 and 3.24. α was set to a constant value of 0.05.

The output channel vector $\mathbf{q}$ was generated according to equation 3.10. This vector was then decoded into a scalar. As mentioned in section 2.4.1, stochastic search methods are often used in reinforcement learning. Here, this is accomplished by adding Gaussian noise to the output. The variance σ was calculated as

$$\sigma = \max\{0,\ 0.1 \times (10 - p)\} \tag{3.25}$$

which gives a high noise level when the system predicts a low Q-value and a low noise level if the prediction is high. The value 10 is determined by the maximum value of the Q-function for $\gamma = 0.9$ since $\sum_{i=0}^{\infty} \gamma^i = 10$. The max operation is to ensure that the variance does not get negative if the stochastic estimation occasionally gives predictions higher than 10.

A typical run is illustrated to the left in figure 3.7. The graph shows the accumulated reward in a sliding window of 100 iterations. Note that the original reinforcement signal (i.e. -1 for failure) was used. To the right, the contents of the memory after convergence are illustrated. We see that the highest Q-value is predicted for the positions ± 0.2 and the corresponding outputs ∓ 0.4 approximately, which is a reasonable solution.

3.6.2 TD-learning with local linear models

In this experiment, both the predictions p of future accumulated reward and the actions y are linear functions of the input variable x. There is one pair of reinforcement association vectors $\mathbf{v}^i$ and one pair of action association vectors $\mathbf{w}^i, i = \{1, 2\}$. For each model i, the predicted reinforcement is calculated as

$$p^i = v_1^i x + v_2^i \tag{3.26}$$

and the output is calculated as

$$y^i = N(\mu_y^i, \sigma_y), \tag{3.27}$$

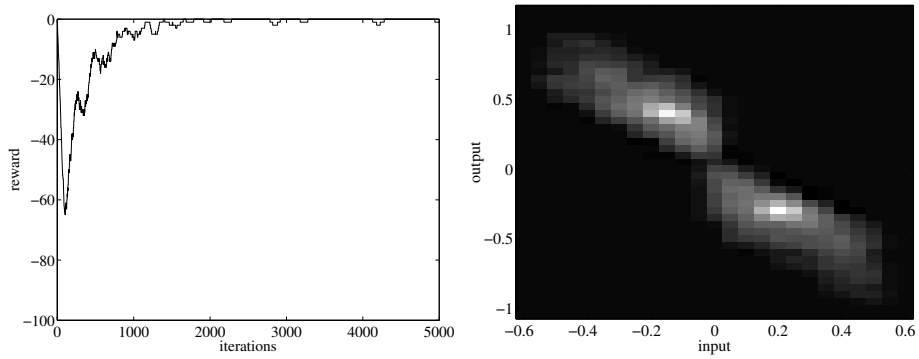

Figure 3.7: **Left:** A typical run of the prediction matrix memory. The graph shows the accumulated reward in a sliding window of 100 iterations. **Right:** The prediction matrix memory after convergence. Black is zero and white is the maximum value.

where

$$\mu_y^i = w_1^i x + w_2^i. \tag{3.28}$$

The system chooses the model c such that

$$p^c = \max_i \{m^i\}, \tag{3.29}$$

where

$$m^i = N(p^i, \sigma_p) \tag{3.30}$$

and generates the corresponding action y^c.

The internal reinforcement signal at time $t+1$ is calculated as

$$\hat{r}[t+1] = r[t+1] + \gamma p^{max}[t, t+1] - p^c[t,t]. \tag{3.31}$$

This is in principle the same TD-method as the one used by Sutton (1984), except that here there are two predictions at each time, one for each model. $p^{max}[t, t+1]$ is the maximum predicted reinforcement calculated using the reinforcement association vector from time t and the input from time $t+1$. If the system fails, i.e. $r = -1$, then $p^{max}[t, t+1]$ is set to zero. $p^c[t,t]$ is the prediction of the selected model.

Learning is accomplished by changing the weights in the reinforcement association vectors and the action association vectors. Only the vectors associated with the chosen model are altered.

The association vectors are updated according to the following rule:

$$\mathbf{w}^c[t+1] = \mathbf{w}^c[t] + \alpha\hat{r}(y^c - \mu_y^c)\mathbf{x} \tag{3.32}$$

and

$$\mathbf{v}^c[t+1] = \mathbf{v}^c[t] + \beta\hat{r}\mathbf{x}\,, \tag{3.33}$$

where c denotes the model choice, α and β are positive learning rate constants and

$$\mathbf{x} = \begin{pmatrix} x \\ 1 \end{pmatrix}.$$

In this experiment, noise is added to the output on two levels. First in the selection of model and then in the generation of output signal. The noise levels are controlled by σ_p and σ_y respectively, as shown in equations 3.27 and 3.30.

The variance parameters are calculated as

$$\sigma_p = \max\{0,\ -0.1 * \max\{p^i\}\} \tag{3.34}$$

and

$$\sigma_y = \max\{0,\ -0.1 * p^c\}. \tag{3.35}$$

The first "max" in the two equations is to make sure that the variances do not become negative. The negative signs are there because of the (relevant) predictions being negative. In this way, the higher the prediction of reinforcement is, the more precision there will be in the output.

The learning behaviour is illustrated to the left in figure 3.8. To the right, the total input-output function is plotted. For each input value, the model with the highest predicted reward has been used. The discrete step close to zero marks the point in the input space where the system switches between the two models. The optimal position for this point is of course zero.

One problem that can occur with this algorithm, and other similar algorithms, is when both models prefer the same part of the input space. This means that the two reinforcement prediction functions predict the same reinforcement for the same inputs and, as a result, both models generate the same actions. This problem can of course be solved if the teacher who generates the external reinforcement signal knows approximately where the breakpoint should be and which model should act on which side. The teacher could then punish the system for selecting the wrong model by giving negative reinforcement. In general, however, the teacher does not know how to divide the problem. In that case, the teacher must try to use a pedagogical reward as discussed in section 2.4.2 on page 20. The teacher could for instance give less reward if the models try to cover the same part of the input space and a higher reward when the models tend to cover different parts of the space.

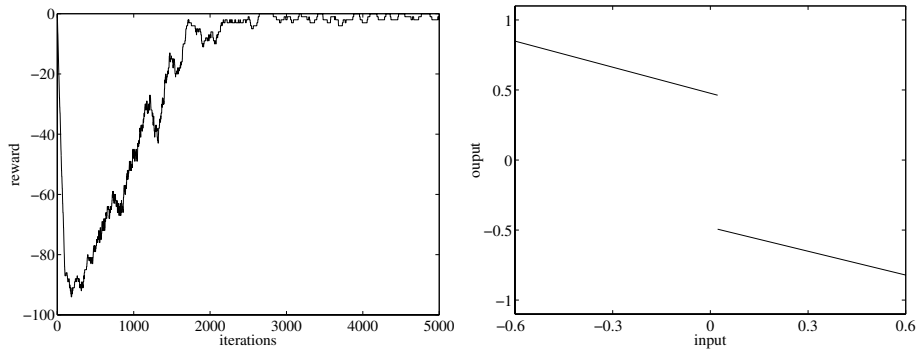

Figure 3.8: **Left:** A typical run of the TD-learning system with two local models. The graph shows the accumulated reward in a sliding window of 100 iterations. **Right:** The total input-output function after convergence.

3.6.3 Discussion

If we compare the contents of the prediction matrix memory to the right in figure 3.7 and the combined function of the linear models in the TD-system plotted to the right in figure 3.8, we see that the two systems implement approximately the same function.

If we compare the learning behaviour (plotted to the left in figure 3.7 and 3.8), the prediction matrix memory appears to learn faster than the TD-method. It should however be noted that each iteration of the prediction matrix memory has a computational complexity of order $O(Q \times V)$, where Q and V are the number of channels used for representing the input and output signals respectively. In this experiment, we used $Q = 25$ and $V = 45$. A larger number of channels enhances the performance when the system has converged but increases the required number of iterations until convergence as well as the computational complexity for each iteration. The computational complexity of the second method is of order $O(N \times (X+1) \times Y)$ per iteration, where N is the number of local models (in this case 2), X is the dimensionality of the input signal (in this case 1) and Y is the dimensionality of the output signal (in this case 1).

The algorithms have not been optimized with respect to convergence time. The convergence speed depends on the setting of the learning rate constants α and β and the modulation of the variance parameters σ_p and σ_y. These parameters have only been tuned to constant values that work reasonably well. Better results can be expected if the learning rates are made adaptive, as discussed in section 2.3.2 on page 11.

Chapter 4

Low-dimensional linear models

As we have seen in the previous chapter (in section 3.4), local low-dimensional linear models is a good way of representing high-dimensional data in a learning system. The linear models can be seen as basis vectors spanning a (local) subspace of the signal space. The signal can then be (approximately) described in this new basis in terms of projections onto the new basis vectors. For signals with high dimensionality, an iterative algorithm for finding this basis must not exhibit a memory requirement nor a computational cost significantly exceeding $O(d)$ per iteration, where d is the dimensionality of the signal. Techniques involving matrix multiplications (having memory requirements of order $O(d^2)$ and computational costs of order $O(d^3)$), quickly become infeasible when signal space dimensionality increases.

The purpose of local models is dimensionality reduction which means throwing away information that is not needed. Hence, the criterion for an appropriate local model is dependent on the application. One criterion is to preserve as much *variance* as possible given a certain dimensionality of the model. This is done by projecting the data on the subspace of maximum data *variation*, i.e. the subspace spanned by the largest *principal components*. This is known as principal component analysis (PCA). There is a number of applications in signal processing where principal components play an important role, for example image coding.

In applications where relations between two sets of data (e.g. process input and output) are considered, PCA or other self-organizing algorithms for representing the two sets of data *separately* are not very useful since such methods cannot separate useful information from noise. Consider, for example, two high-dimensional signals that are described by their most significant principal components. There is no reason to believe that these descriptions of the signals are related in any way. In other words, the signal in the direction of maximum variance in one space may be totally independent of the signal in the direction of

maximum variance in another space, even if there is a strong relation between the signals. The reason for this is that there is no way of finding the relation between two sets of data just by looking at one of the sets. Instead, the two signal spaces must be considered together. One method for doing this is finding the subspaces in the input and the output spaces for which the data *covariation* is maximized. These subspaces turn out to be the ones accompanying the largest singular values of the between-sets covariance matrix (Landelius et al., 1995). A singular value decomposition (SVD) of the between-sets covariance matrix corresponds to *partial least squares* (PLS) (Wold et al., 1984; Höskuldsson, 1988).

In general, however, the input to a system comes from a set of different sensors and it is evident that the range (or variance) of the signal values from a given sensor is unrelated to the importance of the received information. The same line of reasoning holds for the output which may consist of signals to a set of different effectuators. In these cases, the covariances between signals are not relevant. There may, for example, be one pair of directions in the two spaces that has a high covariance due to high signal magnitude but has a high noise level, while another pair of directions has an almost perfect correlation but a small signal magnitude and therefore low covariance. Here, *correlation* between input and output signals is a more appropriate target for analysis since this measure of signal relations is invariant to the signal magnitudes. This approach leads to a *canonical correlation analysis* (CCA) (Hotelling, 1936) of the two sets of signals.

Finally, when the goal is to predict a signal as well as possible in a least square error sense, the basis must be chosen so that this error measure is minimized. This corresponds to a low-rank approximation of *multivariate linear regression* (MLR). This is also known as *reduced rank regression* (Izenman, 1975) or as *redundancy analysis* (van den Wollenberg, 1977).

In general, these four different criteria for selecting basis vectors lead to four different solutions. But, as we will see, the problems are related to each other and can be formulated in very similar ways. An important problem which is directly related to the situations discussed above is the *generalized eigenproblem* or two-matrix eigenproblem (Bock, 1975; Golub and Loan, 1989; Stewart, 1976). In the next section, the generalized eigenproblem is described in some detail and its relation to an energy function called the *Rayleigh quotient* is shown. It is shown that the four important methods discussed above (principal component analysis (PCA), partial least squares (PLS), canonical correlation analysis (CCA) and multivariate linear regression (MLR)) emerge as solutions to special cases of the generalized eigenproblem.

In section 4.7, an iterative $O(d)$ algorithm that solves the generalized eigenproblem by a gradient search on the Rayleigh quotient is presented. The solutions are found in a successive order beginning with the largest eigenvalue and the cor-

responding eigenvector. It is shown how to apply this algorithm in order to obtain the required solutions in the special cases of PCA, PLS, CCA and MLR.

Throughout this chapter, the variables are assumed to be real valued and have zero mean so that the covariance matrices can be defined as $\mathbf{C}_{xx} = E[\mathbf{x}\mathbf{x}^T]$. The zero mean does not impose any limitations on the methods discussed since the mean values can easily be estimated and stored by each local model.

The essence of this chapter has been submitted for publication (Borga et al., 1997b).

4.1 The generalized eigenproblem

When dealing with many scientific and engineering problems, some version of the generalized eigenproblem sometimes needs to be solved along the way:

$$\mathbf{A}\hat{\mathbf{e}} = \lambda\mathbf{B}\hat{\mathbf{e}} \quad \text{or} \quad \mathbf{B}^{-1}\mathbf{A}\hat{\mathbf{e}} = \lambda\hat{\mathbf{e}}. \tag{4.1}$$

(In the right-hand equation, $\mathbf{B}$ is supposed to be non-singular.) In mechanics, the eigenvalues often correspond to modes of vibration. Here, however, the case where the matrices $\mathbf{A}$ and $\mathbf{B}$ consist of components which are expectation values from stochastic processes is considered. Furthermore, both matrices are symmetric and, in addition, $\mathbf{B}$ is positive definite.

The generalized eigenproblem is closely related to the problem of finding the extremum points (i.e. the points of zero derivatives) of a ratio of *quadratic forms*:

$$r = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}, \tag{4.2}$$

where both $\mathbf{A}$ and $\mathbf{B}$ are symmetric and $\mathbf{B}$ is positive definite. This ratio is known as the *Rayleigh quotient* and its critical points correspond to the eigensystem of the generalized eigenproblem. To see this, consider the gradient of r:

$$\frac{\partial r}{\partial \mathbf{w}} = \frac{2}{\mathbf{w}^T\mathbf{B}\mathbf{w}}(\mathbf{A}\mathbf{w} - r\mathbf{B}\mathbf{w}) = \alpha(\mathbf{A}\hat{\mathbf{w}} - r\mathbf{B}\hat{\mathbf{w}}), \tag{4.3}$$

where $\alpha = \alpha(\mathbf{w})$ is a positive scalar. Setting the gradient to $\mathbf{0}$ gives

$$\mathbf{A}\hat{\mathbf{w}} = r\mathbf{B}\hat{\mathbf{w}} \quad \text{or} \quad \mathbf{B}^{-1}\mathbf{A}\hat{\mathbf{w}} = r\hat{\mathbf{w}} \tag{4.4}$$

which is recognized as the generalized eigenproblem (equation 4.1). The solutions r_i and $\hat{\mathbf{w}}_i$ are the eigenvalues and eigenvectors respectively of the matrix $\mathbf{B}^{-1}\mathbf{A}$. This means that the extremum points of the Rayleigh quotient $r(\mathbf{w})$ are solutions to the corresponding generalized eigenproblem. The eigenvalues are the extremum values of the quotient and the eigenvectors are the corresponding

parameter vectors $\mathbf{w}$ of the quotient. A special case of the Rayleigh quotient is *Fisher's linear discriminant function* (Fisher, 1936) used in classification. In this case, $\mathbf{A}$ is the *between-class scatter matrix* and $\mathbf{B}$ is the *within-class scatter matrix* (see for example Duda and Hart, 1973).

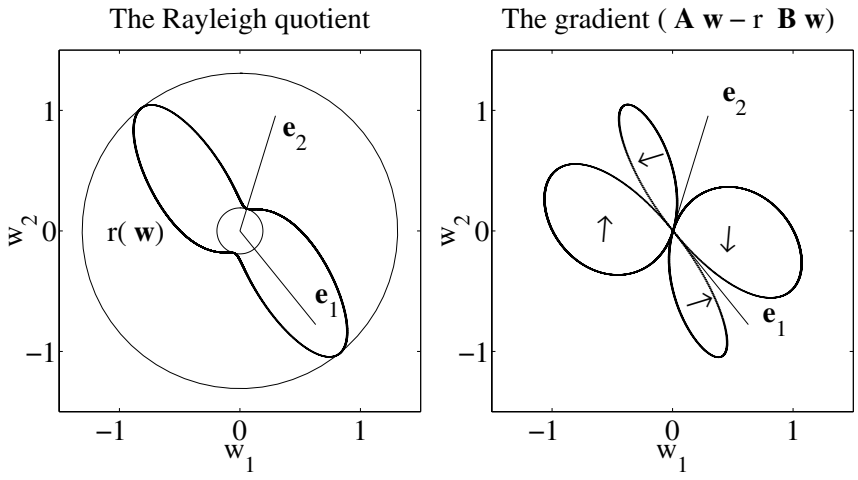

Figure 4.1: **Left:** The Rayleigh quotient $r(\mathbf{w})$ between two matrices $\mathbf{A}$ and $\mathbf{B}$. The curve is plotted as $r\hat{\mathbf{w}}$. The eigenvectors of $\mathbf{B}^{-1}\mathbf{A}$ are marked as reference. The corresponding eigenvalues are marked as the radii of the two circles. Note that the quotient is invariant to the norm of $\mathbf{w}$. **Right:** The gradient of r. The arrows indicate the direction of the gradient and the radii of the blobs correspond to the magnitude of the gradient.

As an illustration, the Rayleigh quotient is plotted to the left in figure 4.1 for two matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 0.25 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}. \tag{4.5}$$

The quotient is plotted as the radius in different directions $\hat{\mathbf{w}}$. Note that the quotient is invariant to the norm of $\mathbf{w}$. The two eigenvalues are shown as circles with their radii corresponding to the eigenvalues. The figure shows that the eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ of the generalized eigenproblem coincide with the maximum and minimum values of the Rayleigh quotient. To the right in the same figure, the gradient of the Rayleigh quotient is illustrated as a function of the direction of $\mathbf{w}$. Note that the gradient is orthogonal to $\mathbf{w}$ (see equation 4.3). This means that a small change of $\mathbf{w}$ in the direction of the gradient can be seen as a rotation of $\mathbf{w}$.

The arrows indicate the direction of this orientation and the radii of the blobs correspond to the magnitude of the gradient. The figure shows that the directions of zero gradient coincide with the eigenvectors and that the gradient points towards the eigenvector corresponding to the largest eigenvalue.

If the eigenvalues r_i are distinct[1] (i.e. $r_i \neq r_j$ for $i \neq j$), the different eigenvectors are orthogonal in the metrics $\mathbf{A}$ and $\mathbf{B}$, i.e.

$$\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = \begin{cases} 0 & \text{for} \quad i \neq j \\ \beta_i > 0 & \text{for} \quad i = j \end{cases} \quad \text{and} \quad \hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = \begin{cases} 0 & \text{for} \quad i \neq j \\ r_i \beta_i & \text{for} \quad i = j \end{cases} \tag{4.6}$$

(see proof B.3.1 on page 157). This means that the $\mathbf{w}_i$s are linearly independent (see proof B.3.2 on page 158). Since an n-dimensional space gives n eigenvectors which are linearly independent, $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ constitutes a *base* and any $\mathbf{w}$ can be expressed as a linear combination of the eigenvectors. Now, it can be proved (see proof B.3.3 on page 158) that the function r is bounded by the largest and the smallest eigenvalue, i.e.

$$r_n \leq r \leq r_1 \tag{4.7}$$

which means that there exists a global maximum and that this maximum is r_1.

To investigate if there are any other local maxima, we look at the second derivative, or the *Hessian* $\mathbf{H}$, of r for the solutions to the eigenproblem,

$$\mathbf{H}_i = \left. \frac{\partial^2 r}{\partial \mathbf{w}^2} \right|_{\mathbf{w}=\hat{\mathbf{w}}_i} = \frac{2}{\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i} (\mathbf{A} - r_i \mathbf{B}) \tag{4.8}$$

(see proof B.3.4 on page 159). The Hessian $\mathbf{H}_i$ have positive eigenvalues for $i > 1$, i.e. there exist vectors $\mathbf{w}$ such that

$$\mathbf{w}^T \mathbf{H}_i \mathbf{w} > 0 \quad \forall\, i > 1 \tag{4.9}$$

(see proof B.3.5 on page 159). This means that for all solutions to the eigenproblem except for the largest root, there exists a direction in which r increases. In other words, all extremum points of the function r are saddle points except for the global minimum and maximum points. Since the two-dimensional example in figure 4.1 only has two eigenvalues, they correspond to the maximum and minimum values of r.

In the following sections is shown that finding the directions of maximum variance, maximum covariance, maximum correlation and minimum square error can be seen as special cases of the generalized eigenproblem.

[1] The eigenvalues will be distinct in all practical applications since all real signals contain noise.

4.2 Principal component analysis

Consider a set of random vectors $\mathbf{x}$ (signals) with a covariance matrix defined by

$$\mathbf{C}_{xx} = E[\mathbf{x}\mathbf{x}^T]. \tag{4.10}$$

Suppose the goal is to find the direction of maximum variation in the signal distribution. The direction of maximum variation means the direction $\hat{\mathbf{w}}$ such that the linear combination $x = \mathbf{x}^T\hat{\mathbf{w}}$ possesses maximum variance. Hence, finding this direction is equivalent to finding the maximum of

$$\rho = E[xx] = E[\hat{\mathbf{w}}^T\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}}] = \hat{\mathbf{w}}^T E[\mathbf{x}\mathbf{x}^T]\hat{\mathbf{w}} = \frac{\mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}}{\mathbf{w}^T\mathbf{w}}. \tag{4.11}$$

This is a special case of the Rayleigh quotient in equation 4.2 on page 61 with

$$\mathbf{A} = \mathbf{C}_{xx} \quad \text{and} \quad \mathbf{B} = \mathbf{I}. \tag{4.12}$$

Since the covariance matrix is symmetric, it is possible to decompose it into its eigenvalues and orthogonal eigenvectors as

$$\mathbf{C}_{xx} = E[\mathbf{x}\mathbf{x}^T] = \sum \lambda_i \, \hat{\mathbf{e}}_i\hat{\mathbf{e}}_i^T, \tag{4.13}$$

where λ_i and $\hat{\mathbf{e}}_i$ are the eigenvalues and the orthogonal eigenvectors respectively. Hence, the problem of maximizing the variance, ρ, can be seen as the problem of finding the largest eigenvalue, λ_1, and its corresponding eigenvector since

$$\lambda_1 = \hat{\mathbf{e}}_1^T\mathbf{C}_{xx}\hat{\mathbf{e}}_1 = \max\frac{\mathbf{w}^T\mathbf{C}_{xx}\mathbf{w}}{\mathbf{w}^T\mathbf{w}} = \max\rho. \tag{4.14}$$

It is also worth noting that it is possible to find the direction and magnitude of maximum data variation for the inverse of the covariance matrix. In this case, we simply identify the matrices in eq. 4.2 on page 61 as $\mathbf{A} = \mathbf{I}$ and $\mathbf{B} = \mathbf{C}_{xx}$.

The eigenvectors $\mathbf{e}_i$ are also known as the *principal components* of the distribution of $\mathbf{x}$. Principal component analysis (PCA) is an old tool in multivariate data analysis. It was used already in 1901 (Pearson, 1901). The projection of data onto the principal components is sometimes called the Hotelling transform after Hotelling (1933) or Karhunen-Loéve transform (KLT) after Karhunen (1947) and Loéve (1963). This transformation is as an orthogonal transformation that diagonalizes the covariance matrix.

PCA gives a data dependent set of basis vectors that is optimal in a statistical mean square error sense. This was shown in equation 2.45 on page 33 for one basis vector and the result can easily be generalized to a set of basis vectors by the following reasoning: Given *one* basis vector, the best we can do is to choose

the maximal eigenvector of the covariance matrix. This basis vector describes the signal completely in that direction. Hence, there is nothing more in that direction to describe and the next basis vector should be chosen orthogonal to the first. Now the same problem is faced again, but in a smaller space where the first principal component of the signal is removed. So the best choice of the second basis vector is a unit vector in the direction of the first principal component in this subspace and that direction corresponds to the second eigenvector[2] of the covariance matrix. This process can be repeated for all basis vectors.

The KLT can be used for image coding (Torres and Kunt, 1996) since it is the optimal transform coding in a mean square error sense. This is, however, not very common. One reason for this is that the KLT is computationally more expensive than the discrete cosine transform (DCT). Another reason is the need for transmission of the data dependent basis vectors. Besides that, in general the mean square error is not a very good error measure for images since two images with a large mean square distance can look very similar to a human observer. Another use for PCA in multivariate statistical analysis is to find linear combinations of variables where the variance is high. Here, it should be noted that PCA is dependent on the units used for measuring. If the unit of one variable is changed, for example from metres to feet, the orientations of the principal components may change. For further details on PCA, see for example the overview by Jolliffe (1986).

When dealing with learning systems, it could be tempting to use PCA to find local linear models to reduce the dimensionality of a high-dimensional input (and output) space. The problem with this approach is that the best representation of the input signal is in general not the least mean square error representation of that signal. There may be components in the input signal that have high variances that are totally irrelevant when it comes to generating responses and there may be components with small variances that are very important. In other words, PCA is not a good tool when analysing the relations between two sets of variables. The need for simultaneous analysis of the input and output signals in learning systems was indicated in the quotation from Brooks (1986) on page 8 and also in the wheel-chair experiment (Held and Bossom, 1961; Mikaelian and Held, 1964) mentioned on the same page.

[2]The somewhat informal notation "second eigenvector" refers to the eigenvector corresponding to the second largest eigenvalue.

4.3 Partial least squares

Now, consider *two* sets of random vectors $\mathbf{x}$ and $\mathbf{y}$ with the between-sets covariance matrix defined by

$$\mathbf{C}_{xy} = E[\mathbf{x}\mathbf{y}^T]. \tag{4.15}$$

Suppose, this time, that the goal is to find the two directions of maximal data *covariation*, by which is meant the directions $\hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}_y$ such that the linear combinations $x = \mathbf{x}^T\hat{\mathbf{w}}_x$ and $y = \mathbf{y}^T\hat{\mathbf{w}}_y$ give maximum covariance. This means that the following function should be maximized:

$$\rho = E[xy] = E[\hat{\mathbf{w}}_x^T\mathbf{x}\mathbf{y}^T\hat{\mathbf{w}}_y] = \hat{\mathbf{w}}_x^T E[\mathbf{x}\mathbf{y}^T]\hat{\mathbf{w}}_y = \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y}{\sqrt{\mathbf{w}_x^T\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y}}. \tag{4.16}$$

Note that, for each ρ, a corresponding value $-\rho$ is obtained by rotating $\mathbf{w}_x$ or $\mathbf{w}_y$ 180°. For this reason, the maximum *magnitude* of ρ is obtained by finding the largest positive value.

This function cannot be written as a Rayleigh quotient. However, the critical points of this function coincide with the critical points of a Rayleigh quotient with proper choices of $\mathbf{A}$ and $\mathbf{B}$. To see this, we calculate the derivatives of this function with respect to the vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ (see proof B.3.6 on page 160):

$$\begin{cases} \frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{1}{\|\mathbf{w}_x\|}(\mathbf{C}_{xy}\hat{\mathbf{w}}_y - \rho\hat{\mathbf{w}}_x) \\ \frac{\partial\rho}{\partial\mathbf{w}_y} &= \frac{1}{\|\mathbf{w}_y\|}(\mathbf{C}_{yx}\hat{\mathbf{w}}_x - \rho\hat{\mathbf{w}}_y). \end{cases} \tag{4.17}$$

Setting these expressions to zero and solving for $\mathbf{w}_x$ and $\mathbf{w}_y$ results in

$$\begin{cases} \mathbf{C}_{xy}\mathbf{C}_{yx}\hat{\mathbf{w}}_x &= \rho^2\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\mathbf{C}_{xy}\hat{\mathbf{w}}_y &= \rho^2\hat{\mathbf{w}}_y. \end{cases} \tag{4.18}$$

This is exactly the same result as that given by the extremum points of r in equation 4.2 on page 61 if the matrices $\mathbf{A}$ and $\mathbf{B}$ and the vector $\mathbf{w}$ are chosen according to:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} \quad , \quad \mathbf{B} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \mu_x\hat{\mathbf{w}}_x \\ \mu_y\hat{\mathbf{w}}_y \end{pmatrix}. \tag{4.19}$$

This is easily verified by insertion of the expressions above into equation 4.4, which results in

$$\begin{cases} \mathbf{C}_{xy}\hat{\mathbf{w}}_y &= r\frac{\mu_x}{\mu_y}\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\hat{\mathbf{w}}_x &= r\frac{\mu_y}{\mu_x}\hat{\mathbf{w}}_y \end{cases}. \tag{4.20}$$

Solving for $\mathbf{w}_x$ and $\mathbf{w}_y$ gives equation 4.18 with $r^2 = \rho^2$. Hence, the problem of finding the direction and magnitude of the largest data covariation can be seen as maximizing a special case of the Rayleigh quotient (equation 4.2 on page 61) with the appropriate choice of matrices.

The between-sets covariance matrix can be expanded by means of singular value decomposition (SVD) where the two sets of vectors $\{\hat{\mathbf{e}}_{xi}\}$ and $\{\hat{\mathbf{e}}_{yi}\}$ are mutually orthogonal:

$$\mathbf{C}_{xy} = \sum \lambda_i \, \hat{\mathbf{e}}_{xi} \hat{\mathbf{e}}_{yi}^T \tag{4.21}$$

where the positive numbers, λ_i, are referred to as the singular values. Since the basis vectors are orthogonal, the problem of maximizing the quotient in equation 4.16 is equivalent to finding the largest singular value:

$$\lambda_1 = \hat{\mathbf{e}}_{x1}^T \mathbf{C}_{xy} \hat{\mathbf{e}}_{y1} = \max \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y}{\sqrt{\mathbf{w}_x^T \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y}} = \max \rho. \tag{4.22}$$

The SVD of a between-sets covariance matrix is directly related to the method of partial least squares (PLS). PLS was developed in econometrics in the 1960s by Herman Wold. It is most commonly used for regression in the field of chemometrics (Wold et al., 1984). For an overview, see for example Geladi and Kowalski (1986) and Höskuldsson (1988). In PLS regression, the principal vectors corresponding to the largest principal values are used as a new, lower dimensional, basis for the signal. A regression of $\mathbf{y}$ onto $\mathbf{x}$ is then performed in this new basis.

As in the case of PCA, the scaling of the variables affects the solutions of the PLS. The reason for this is the maximum covariance criteria; the covariance between two variables is proportional to the variances of the variables. Therefore, a scaling of the $\mathbf{x}$ variables to unit variance is sometimes suggested (Wold et al., 1984). Such a solution can of course also amplify the noise which can cause problems in the parameter estimation[3].

4.4 Canonical correlation analysis

Again, consider two random variables $\mathbf{x}$ and $\mathbf{y}$ with zero mean and stemming from a multi-normal distribution with the total covariance matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{bmatrix} = E\left[\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \right]. \tag{4.23}$$

[3] An example of such a problem has been reported from the paper industry (personal communication). In that case, the normalized data had to be filtered to reduce the amplified noise! The filtering will likely introduce new artifacts.

Now, suppose that the goal is to find the directions of maximum data *correlation*. Consider the linear combinations $x = \mathbf{x}^T \hat{\mathbf{w}}_x$ and $y = \mathbf{y}^T \hat{\mathbf{w}}_y$ of the two variables respectively. This means that the function to be maximized is

$$\begin{aligned} \rho &= \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}} = \frac{E[\hat{\mathbf{w}}_x^T \mathbf{x}\mathbf{y}^T \hat{\mathbf{w}}_y]}{\sqrt{E[\hat{\mathbf{w}}_x^T \mathbf{x}\mathbf{x}^T \hat{\mathbf{w}}_x] E[\hat{\mathbf{w}}_y^T \mathbf{y}\mathbf{y}^T \hat{\mathbf{w}}_y]}} \\ &= \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y}{\sqrt{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y}}. \end{aligned} \tag{4.24}$$

Also in this case, since ρ changes sign if $\mathbf{w}_x$ or $\mathbf{w}_y$ is rotated 180°, it is sufficient to find the positive values.

Like equation 4.16, this function cannot be written as a Rayleigh quotient. But also in this case, it can be shown that the critical points of this function coincide with the critical points of a Rayleigh quotient with proper choices of $\mathbf{A}$ and $\mathbf{B}$. The partial derivatives of ρ with respect to $\mathbf{w}_x$ and $\mathbf{w}_y$ are (see proof B.3.7 on page 160)

$$\begin{cases} \frac{\partial \rho}{\partial \mathbf{w}_x} &= \frac{a}{\|\mathbf{w}_x\|} \left(\mathbf{C}_{xy} \hat{\mathbf{w}}_y - \frac{\hat{\mathbf{w}}_x^T \mathbf{C}_{xy} \hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x} \mathbf{C}_{xx} \hat{\mathbf{w}}_x \right) \\ \frac{\partial \rho}{\partial \mathbf{w}_y} &= \frac{a}{\|\mathbf{w}_y\|} \left(\mathbf{C}_{yx} \hat{\mathbf{w}}_x - \frac{\hat{\mathbf{w}}_y^T \mathbf{C}_{yx} \hat{\mathbf{w}}_x}{\hat{\mathbf{w}}_y^T \mathbf{C}_{yy} \hat{\mathbf{w}}_y} \mathbf{C}_{yy} \hat{\mathbf{w}}_y \right), \end{cases} \tag{4.25}$$

where a is a positive scalar. Setting the derivatives to zero gives the equation system

$$\begin{cases} \mathbf{C}_{xy} \hat{\mathbf{w}}_y = \rho \lambda_x \mathbf{C}_{xx} \hat{\mathbf{w}}_x \\ \mathbf{C}_{yx} \hat{\mathbf{w}}_x = \rho \lambda_y \mathbf{C}_{yy} \hat{\mathbf{w}}_y, \end{cases} \tag{4.26}$$

where

$$\lambda_x = \lambda_y^{-1} = \sqrt{\frac{\hat{\mathbf{w}}_y^T \mathbf{C}_{yy} \hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x}}. \tag{4.27}$$

λ_x is the ratio between the standard deviation of y and the standard deviation of x and vice versa. The λs can be interpreted as scaling factors between the linear combinations. Rewriting equation system 4.26 gives

$$\begin{cases} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \hat{\mathbf{w}}_x = \rho^2 \hat{\mathbf{w}}_x \\ \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \hat{\mathbf{w}}_y = \rho^2 \hat{\mathbf{w}}_y. \end{cases} \tag{4.28}$$

Hence, $\hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}_y$ are found as the eigenvectors of the matrices $\mathbf{C}_{xx}^{-1} \mathbf{C}_{xy} \mathbf{C}_{yy}^{-1} \mathbf{C}_{yx}$ and $\mathbf{C}_{yy}^{-1} \mathbf{C}_{yx} \mathbf{C}_{xx}^{-1} \mathbf{C}_{xy}$ respectively. The corresponding eigenvalues ρ^2 are the squared

canonical correlations. The eigenvectors corresponding to the largest eigenvalue ρ_1^2 are the vectors $\hat{\mathbf{w}}_{x1}$ and $\hat{\mathbf{w}}_{y1}$ that maximize the correlation between the *canonical variates* $x_1 = \mathbf{x}^T\hat{\mathbf{w}}_{x1}$ and $y_1 = \mathbf{y}^T\hat{\mathbf{w}}_{y1}$.

Now, if

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix} = \begin{pmatrix} \mu_x\hat{\mathbf{w}}_x \\ \mu_y\hat{\mathbf{w}}_y \end{pmatrix} \tag{4.29}$$

equation 4.4 can be written as

$$\begin{cases} \mathbf{C}_{xy}\hat{\mathbf{w}}_y = r\frac{\mu_x}{\mu_y}\mathbf{C}_{xx}\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\hat{\mathbf{w}}_x = r\frac{\mu_y}{\mu_x}\mathbf{C}_{yy}\hat{\mathbf{w}}_y, \end{cases} \tag{4.30}$$

which is recognized as equation 4.26 for $\rho\lambda_x = r\frac{\mu_x}{\mu_y}$ and $\rho\lambda_y = r\frac{\mu_y}{\mu_x}$. Solving for $\mathbf{w}_x$ and $\mathbf{w}_y$ in equation 4.30, gives equation 4.28 with $r^2 = \rho^2$. This shows that the equations for the canonical correlations are obtained as the result of maximizing the Rayleigh quotient (equation 4.2 on page 61).

Canonical correlation analysis was developed by Hotelling (1936). Some of the results presented here can also be found in (Borga, 1995; Knutsson et al., 1995; Borga et al., 1997a). Although being a standard tool in statistical analysis (see for example Anderson, 1984), where canonical correlation has been used for example in economics, medical studies, meteorology and even in classification of malt whisky (Lapointe and Legendre, 1994) and wine (Montanarella et al., 1995), it is surprisingly unknown in the fields of learning and signal processing. Some exceptions are Becker (1996), Kay (1992), Fieguth et al. (1995), Das and Sen (1994) and Li et al. (1997).

An important property of canonical correlations is that they are invariant with respect to affine transformations of $\mathbf{x}$ and $\mathbf{y}$. An affine transformation is given by a translation of the origin followed by a linear transformation. The translation of the origin of $\mathbf{x}$ or $\mathbf{y}$ has no effect on ρ since it leaves the covariance matrix $\mathbf{C}$ unaffected. Invariance with respect to scalings of $\mathbf{x}$ and $\mathbf{y}$ follows directly from equation 4.24. For invariance with respect to other linear transformations see proof B.3.8 on page 161. Hence, in contrast to PLS, there is no need for normalization of the variables in CCA.

Another important property is that the canonical correlations are uncorrelated for different solutions, i.e.

$$\begin{cases} E[xx] & = E[\mathbf{w}_{xi}^T\mathbf{x}\mathbf{x}^T\mathbf{w}_{xj}] = \mathbf{w}_{xi}^T\mathbf{C}_{xx}\mathbf{w}_{xj} = 0 \\ E[yy] & = E[\mathbf{w}_{yi}^T\mathbf{y}\mathbf{y}^T\mathbf{w}_{yj}] = \mathbf{w}_{yi}^T\mathbf{C}_{yy}\mathbf{w}_{yj} = 0 \\ E[xy] & = E[\mathbf{w}_{xi}^T\mathbf{x}\mathbf{y}^T\mathbf{w}_{yj}] = \mathbf{w}_{xi}^T\mathbf{C}_{xy}\mathbf{w}_{yj} = 0 \end{cases} \quad \text{for} \quad i \neq j, \tag{4.31}$$

according to equation 4.6.

4.4.1 Relation to mutual information and ICA

As mentioned in section 2.5.3, there is a relation between correlation and mutual information (equation 2.44). Since information is additive for statistically independent variables (equation 2.33) and the canonical variates are uncorrelated, the mutual information between $\mathbf{x}$ and $\mathbf{y}$ is the sum of mutual information between the variates x_i and y_i if there are no higher order statistic dependencies than correlation (second-order statistics). For Gaussian variables this means

$$I(\mathbf{x};\mathbf{y}) = \frac{1}{2}\log\left(\frac{1}{\prod_i(1-\rho_i^2)}\right) = \frac{1}{2}\sum_i \log\left(\frac{1}{(1-\rho_i^2)}\right) \tag{4.32}$$

using equation 2.44 on page 32. This is also more formally shown in proof B.3.9 on page 162. Kay (1992)[4] has shown that this relation plus a constant holds for all elliptically symmetrical distributions of the form

$$cf((\mathbf{z}-\bar{\mathbf{z}})^T\mathbf{C}^{-1}(\mathbf{z}-\bar{\mathbf{z}})). \tag{4.33}$$

Minimizing mutual information between signal components is known as independent component analysis (ICA) (see for example Comon, 1994). If there are no higher order statistic dependencies than correlation (e. g. if the variables are jointly Gaussian[5]), the canonical correlates x_i, x_j, $i \neq j$ are independent components since they are uncorrelated.

4.4.2 Relation to SNR

The correlation is strongly related to signal to noise ratio (SNR), which is a more commonly used measure in signal processing. This relation is used later in this thesis.

Consider a signal x and two noise signals η_1 and η_2 all having zero mean[6] and all being uncorrelated with each other. Let $S = E[x^2]$ and $N_i = E[\eta_i^2]$ be the energy of the signal and the noise signals respectively. Then the correlation

[4] There is a difference by a factor 0.5 between equation 4.32 and Kay's equation, due to a typographical error.

[5] The definition of ICA requires that at most one of the source components is Gaussian (Comon, 1994).

[6] The assumption of zero mean is for convenience. A non-zero mean does not affect the SNR or the correlation.

between $a(x+\eta_1)$ and $b(x+\eta_2)$ is

$$\begin{aligned}\rho &= \frac{E\left[a(x+\eta_1)b(x+\eta_2)\right]}{\sqrt{E\left[a^2(x+\eta_1)^2\right]E\left[b^2(x+\eta_2)^2\right]}} \\ &= \frac{E\left[x^2\right]}{\sqrt{\left(E\left[x^2\right]+E\left[\eta_1^2\right]\right)\left(E\left[x^2\right]+E\left[\eta_2^2\right]\right)}} \\ &= \frac{S}{\sqrt{(S+N_1)(S+N_2)}}.\end{aligned} \tag{4.34}$$

Note that the amplification factors a and b do not affect the correlation or the SNR.

Equal noise energies

In the special case where the noise energies are equal, i.e. $N_1 = N_2 = N$, equation 4.34 can be written as

$$\rho = \frac{S}{S+N}. \tag{4.35}$$

This means that the SNR can be written as

$$\frac{S}{N} = \frac{\rho}{1-\rho}. \tag{4.36}$$

Here, it should be noted that the noise affects the signal *twice*, so this relation between SNR and correlation is perhaps not so intuitive. This relation is illustrated in figure 4.2 (top).

Correlation between a signal and the corrupted signal

Another special case is when $N_1 = 0$ and $N_2 = N$. Then, the correlation between a signal and a noise-corrupted version of that signal is

$$\rho = \frac{S}{\sqrt{S(S+N)}}. \tag{4.37}$$

In this case, the relation between SNR and correlation is

$$\frac{S}{N} = \frac{\rho^2}{1-\rho^2}. \tag{4.38}$$

This relation between correlation and SNR is illustrated in figure 4.2 (bottom).

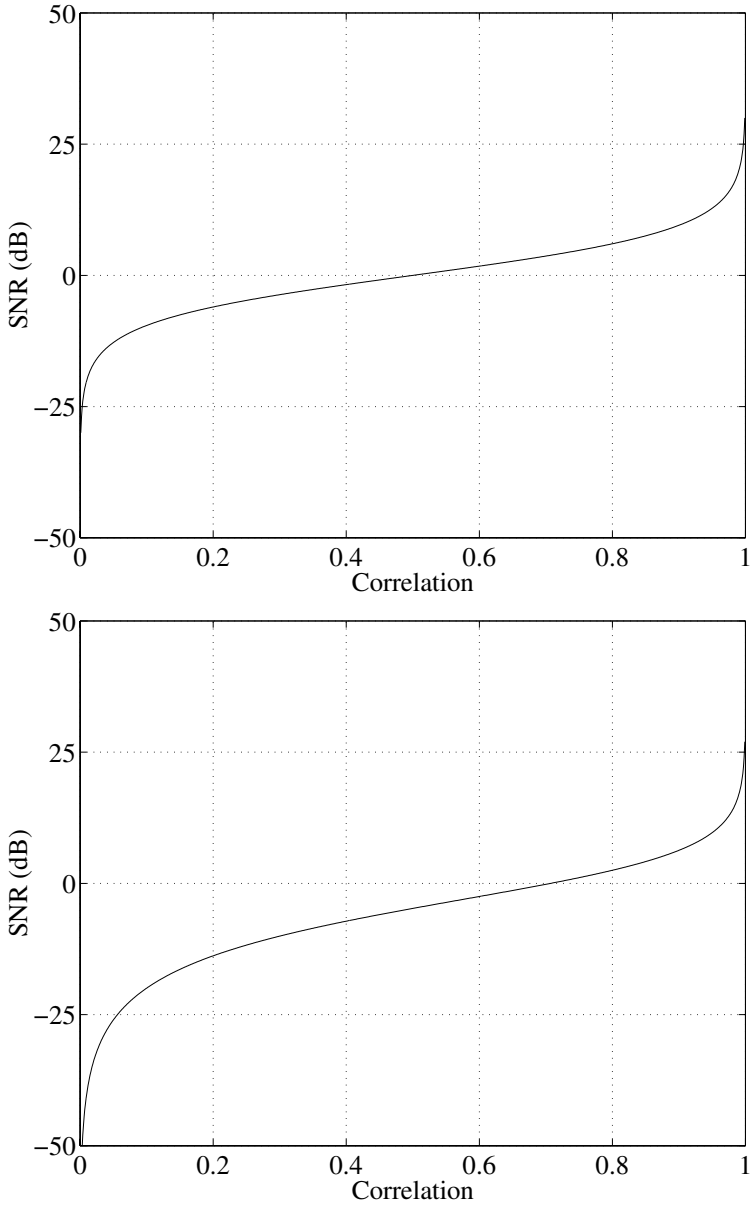

Figure 4.2: **Top:** The relation between correlation and SNR for two signals each corrupted by uncorrelated noise. Both noise signals have the same energy. **Bottom:** The relation between correlation and SNR. The correlation is measured between a signal and a noise-corrupted version of that signal.

4.5 Multivariate linear regression

Again, consider two random variables $\mathbf{x}$ and $\mathbf{y}$ with zero mean and stemming from a multi-normal distribution with covariance as in equation 4.23. In this case, the goal is to minimize the square error

$$\begin{aligned}\varepsilon^2 &= E[\|\mathbf{y} - \beta\mathbf{x}^T\hat{\mathbf{w}}_x\hat{\mathbf{w}}_y\|^2] \\ &= E[\mathbf{y}^T\mathbf{y} - 2\beta\hat{\mathbf{w}}_x^T\mathbf{x}\mathbf{y}^T\hat{\mathbf{w}}_y + \beta^2\hat{\mathbf{w}}_x^T\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}}_x] \\ &= E[\mathbf{y}^T\mathbf{y}] - 2\beta\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y + \beta^2\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x,\end{aligned} \tag{4.39}$$

i.e. a rank-one approximation of the MLR of $\mathbf{y}$ onto $\mathbf{x}$ based on minimum square error. The problem is to find not only the regression coefficient β, but also the optimal basis $\hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}_y$. To get an expression for β, we calculate the derivative

$$\frac{\partial\varepsilon^2}{\partial\beta} = 2\left(\beta\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x - \hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y\right). \tag{4.40}$$

Setting the derivative equal to zero gives

$$\beta = \frac{\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}. \tag{4.41}$$

By inserting this expression into equation 4.39 we get

$$\varepsilon^2 = E[\mathbf{y}^T\mathbf{y}] - \frac{(\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y)^2}{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}. \tag{4.42}$$

Since ε^2 cannot be negative and the left term is independent of the parameters, we can minimize ε^2 by maximizing the quotient to the right in equation 4.42, i.e. maximizing the quotient

$$\rho = \frac{\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\sqrt{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}} = \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y}{\sqrt{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y}}. \tag{4.43}$$

Note that if $\mathbf{w}_x$ and $\mathbf{w}_y$ minimize ε^2, the negation of one or both of these vectors will give the same minimum. Hence, it is sufficient to maximize the positive root.

Like in the two previous cases, this function cannot be written as a Rayleigh quotient, but its critical points coincide with the critical points of a Rayleigh quotient with proper choices of $\mathbf{A}$ and $\mathbf{B}$. The partial derivatives of ρ with respect to $\mathbf{w}_x$ and $\mathbf{w}_y$ are (see proof B.3.10 on page 163)

$$\begin{cases}\frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{xy}\hat{\mathbf{w}}_y - \beta\mathbf{C}_{xx}\hat{\mathbf{w}}_x\right) \\ \frac{\partial\rho}{\partial\mathbf{w}_y} &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{yx}\hat{\mathbf{w}}_x - \frac{\rho^2}{\beta}\hat{\mathbf{w}}_y\right).\end{cases} \tag{4.44}$$

Setting the derivatives to zero gives the equation system

$$\begin{cases} \mathbf{C}_{xy}\hat{\mathbf{w}}_y = \beta\mathbf{C}_{xx}\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\hat{\mathbf{w}}_x = \frac{\rho^2}{\beta}\hat{\mathbf{w}}_y, \end{cases} \tag{4.45}$$

which gives

$$\begin{cases} \mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\mathbf{C}_{yx}\hat{\mathbf{w}}_x = \rho^2\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\hat{\mathbf{w}}_y = \rho^2\hat{\mathbf{w}}_y. \end{cases} \tag{4.46}$$

Now, if we let

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix} = \begin{pmatrix} \mu_x\hat{\mathbf{w}}_x \\ \mu_y\hat{\mathbf{w}}_y \end{pmatrix}, \tag{4.47}$$

equation 4.4 can be written as

$$\begin{cases} \mathbf{C}_{xy}\hat{\mathbf{w}}_y = r\frac{\mu_x}{\mu_y}\mathbf{C}_{xx}\hat{\mathbf{w}}_x \\ \mathbf{C}_{yx}\hat{\mathbf{w}}_x = r\frac{\mu_y}{\mu_x}\hat{\mathbf{w}}_y, \end{cases} \tag{4.48}$$

which is recognized as equation 4.45 for $\beta = r\frac{\mu_x}{\mu_y}$ and $\frac{\rho^2}{\beta} = r\frac{\mu_y}{\mu_x}$. Solving equation 4.48 for $\mathbf{w}_x$ and $\mathbf{w}_y$ gives equation 4.46 with $r^2 = \rho^2$. This shows that the minimum square error in equation 4.39 is found as a result of maximizing the Rayleigh quotient in equation 4.2 on page 61 for the proper choice of matrices $\mathbf{A}$ and $\mathbf{B}$ and regression coefficient β.

So far, the first pair of eigenvectors $\mathbf{w}_{x1}$ and $\mathbf{w}_{y1}$, i.e. a rank-one solution, has been discussed. Intuitively, a rank N regression can be expected to be optimized (in a mean square error sense) if the N first pairs of eigenvectors are used, i.e.

$$\varepsilon^2 = E\left[\|\mathbf{y} - \sum_{i=1}^{N}\beta_i\hat{\mathbf{w}}_{xi}^T\mathbf{x}\hat{\mathbf{w}}_{yi}\|^2\right] \tag{4.49}$$

is minimized if $\mathbf{w}_{xi}$ and $\mathbf{w}_{yi}$ are the solutions to equation 4.46 corresponding to the N largest eigenvalues. To see that this really is the case, note that the eigenvectors $\mathbf{w}_{yi}$ in Y are orthogonal since $\mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}$ in equation 4.46 is symmetric. The orthogonality of the $\mathbf{w}_y$s is explained by the Cartesian separability of the square error; when the error in one direction is minimized, no more can be done in that direction to reduce the error. This means that the minimization of ε^2 in equation 4.49 can be seen as N separate problems that can be solved consecutively beginning with the first solution that minimizes equation 4.39. When the first solution is found, the next solution can be searched for in the subspace orthogonal to $\mathbf{w}_{y1}$.

	A	B
PCA	$\mathbf{C}_{xx}$	$\mathbf{I}$
PLS	$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{pmatrix}$	$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$
CCA	$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{pmatrix}$	$\begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{pmatrix}$
MLR	$\begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{pmatrix}$	$\begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$

Table 4.1: The matrices **A** and **B** for PCA, PLS, CCA and MLR.

Now since $\{\mathbf{w}_{yi}\}$ is orthogonal, the next solution is the second pair of eigenvectors and so on. Since $\{\mathbf{w}_{yi}\}$ is orthogonal, the solutions are not unique; any set of vectors spanning the same subspace in Y can be used to minimize ε^2 in equation 4.49 but, of course, with other $\mathbf{w}_{xi}$s and βs.

If all solutions to the eigenproblem in equation 4.46 and the corresponding β_is are used, a solution for multivariate linear regression (MLR), also known as the *Wiener filter*, is obtained. The mean square sum of the eigenvalues, i.e.

$$\sum_i \rho_i^2 / \dim(Y) = \mathrm{tr}(\mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy})/\dim(Y)$$

is known as the *redundancy index* (Stewart and Love, 1968).

It should be noted that the regression coefficient β defined in equation 4.41 is valid for any choice of $\hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}_y$. In particular, if we use the directions of maximum variance, β is the regression coefficient for *principal components regression* (PCR). For the directions of maximum covariance, β is the regression coefficient for PLS regression.

4.6 Comparisons between PCA, PLS, CCA and MLR

The similarities and differences between the four methods can be seen by comparing the matrices **A** and **B** in the generalized eigenproblem (equation 4.1 on page 61). The matrices are listed in table 4.1.

MLR differs from the other three problems in that it is formulated as a mean square error problem, while the other three methods are formulated as maximi-

sation problems. Reduced rank multivariate linear regression can, for example, be used to increase the stability of the predictors when there are more parameters than observations, when the relation is known to be of low rank or, maybe most importantly, when a full rank solution is unobtainable due to computational costs. The regression coefficients can of course also be used for regression in the first three cases. In the case of PCA, the idea is to separately reduce the dimensionality of the X and Y spaces and to do a regression of the first principal components of Y on the first principal components of X. This method is known as principal components regression. The obvious disadvantage here is that there is no reason to believe that the principal components of X are related to the principal components of Y. To avoid this problem, PLS regression is sometimes used. Clearly, this choice of basis is better than PCA for regression purposes since directions of high covariance are selected, which means that a linear relation is easier to find. However, neither of these solutions results in minimum least squares error. This is only obtained using the directions corresponding to the MLR problem.

It is not only the MLR that can be formulated as a mean square error problem. van der Burg (1988) formulated CCA as a mean square error minimization problem:

$$\text{minimize } \varepsilon^2 = E\left[\sum_{i=1:}^{N} (\mathbf{x}^T \hat{\mathbf{w}}_{xi} - \mathbf{y}^T \hat{\mathbf{w}}_{yi})^2\right], \tag{4.50}$$

where N is the rank of the solution. In this way, CCA can be seen as a supervised learning method as discussed in section 2.6.

PCA differs from the other three methods in that it only concerns one set of variables while the other three concern relations between two sets of variables. The difference between PLS, CCA and MLR can be seen by comparing the matrices in the corresponding eigenproblems (see table 4.1). In CCA, the between-sets covariance matrices are normalized with respect to the within-set covariances in both the $\mathbf{x}$ and the $\mathbf{y}$ spaces. In MLR, the normalization is done only with respect to the $\mathbf{x}$ space covariance while the $\mathbf{y}$ space, where the square error is defined, is left unchanged. In PLS, no normalization is done. Hence, these three cases can be seen as the same problem, covariance maximization, where the variables have been subjected to different, data dependent, scaling.

The main difference between CCA and the other three methods is that CCA is closely related to mutual information as described in section 4.4.1 and, hence, can easily be motivated in information theoretical terms. Because of this relation, it is a bit surprising that canonical correlation seems to be rather unknown in the signal processing, learning and neural networks societies. As an example, a search for "neural network(s)" together with "canonical correlation(s)" in the SciSearch Database of the Institute for Scientific Information, Philadelphia, gave

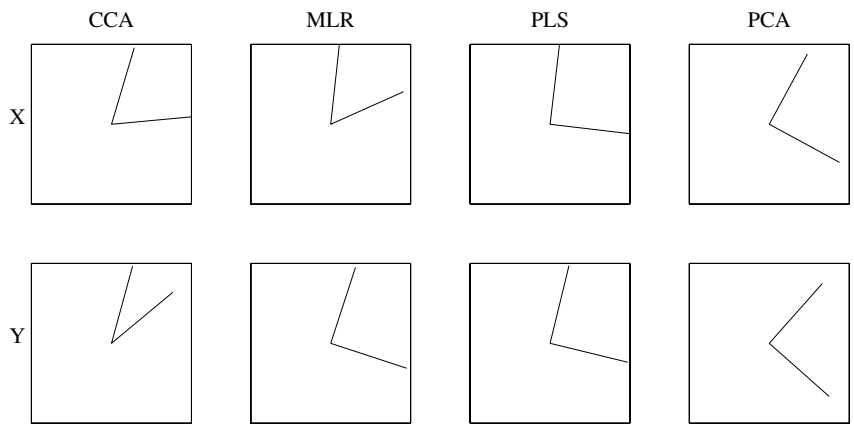

Figure 4.3: Examples of eigenvectors using CCA, MLR, PLS and PCA on the same sets of data.

3 hits. A corresponding search for "partial least square(s)" gave 103 hits, for "linear regression" 212 hits and for "principal component(s)" 287 hits[7]. The same test with "signal processing" instead of "neural networks" gave 2, 5, 18 and 31 hits respectively. This result does not, of course, mean that all articles that matched "principal component(s)" presented learning methods based on PCA. But it may indicate the difference in interest in, or awareness of, the different methods within these fields of research.

To see how these four different special cases of the generalized eigenproblem may differ, the solutions for the same data are plotted in figure 4.3. The data are two-dimensional in X and Y and randomly distributed with zero mean. The top row shows the eigenvectors in X for the CCA, MLR, PLS and PCA respectively. The bottom row shows the solutions in Y. Note that all solutions except the two solutions for CCA and the X-solution for MLR are orthogonal. Figure 4.4 shows the correlation, mean square error, covariance and variance of the data projected onto the first eigenvectors for each method. The figure shows that: the correlation is maximized for the CCA solution; the mean square error is minimized for the MLR solution; the covariance is maximized for the PLS solution; the variance is maximized for the PCA solution.

[7]The search was made on November 4, 1997, through the Norwegian BIBSYS library system (`http://www.bibsys.no`). The "free text" field was used which performs a search in the title, abstract and keywords.

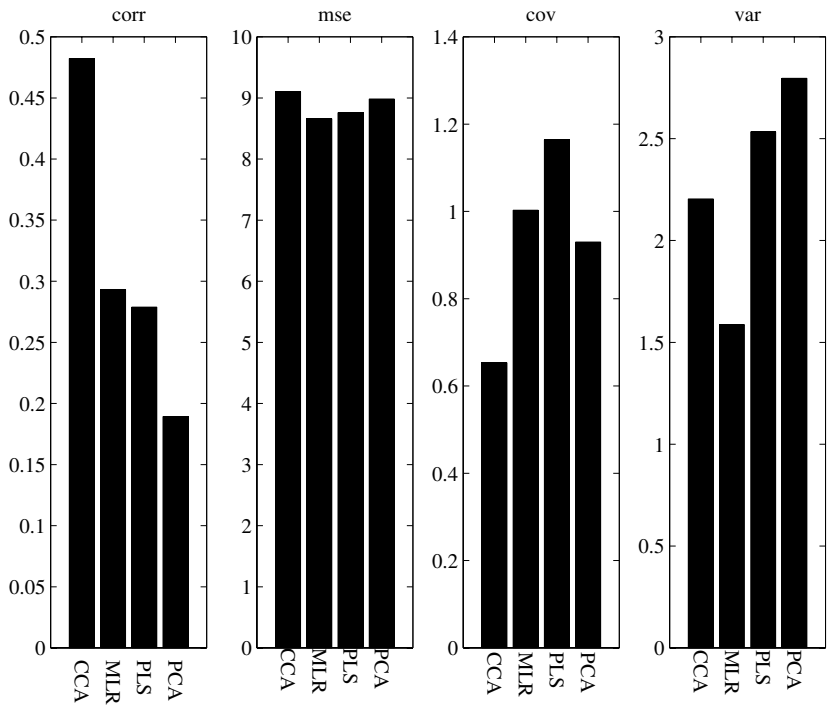

Figure 4.4: The correlation, mean square error, covariance and variance when using the first pair of vectors for each method. The correlation is maximized for the CCA solution. The mean square error is minimized for the MLR solution. The covariance is maximized for the PLS solution. The variance is maximized for the PCA solution. (See section 4.6)

4.7 Gradient search on the Rayleigh quotient

In this section is shown that the solutions to the generalized eigenproblem can be found and, hence, PCA, PLS, CCA or MLR can be performed by a gradient search on the Rayleigh quotient.

Finding the largest eigenvalue

In the previous section was shown that the only stable critical point of the Rayleigh quotient is the global maximum (equation 4.9 on page 63). This means that it should be possible to find the largest eigenvalue of the generalized eigenproblem and its corresponding eigenvector by performing a gradient search on the Rayleigh quotient (equation 4.2 on page 61). This can be done by using an iterative algo-

rithm:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t), \tag{4.51}$$

where the update vector $\Delta\mathbf{w}$, on average, lies in the direction of the gradient:

$$E[\Delta\mathbf{w}] = \beta\frac{\partial r}{\partial \mathbf{w}} = \alpha(\mathbf{A}\hat{\mathbf{w}} - r\mathbf{B}\hat{\mathbf{w}}), \tag{4.52}$$

where α and β are positive numbers. α is the gain controlling how far, in the direction of the gradient, the vector estimate is updated at each iteration. This gain could be constant as well as data or time dependent, as discussed in section 2.3.2.

In all four cases treated here, $\mathbf{A}$ has got at least one positive eigenvalue, i.e. there exists an $r > 0$. An update rule such that

$$E[\Delta\mathbf{w}] = \alpha(\mathbf{A}\hat{\mathbf{w}} - \mathbf{B}\mathbf{w}) \tag{4.53}$$

can then be used to find the positive eigenvalues. Here, the length of the vector represents the corresponding eigenvalue, i.e. $\|\mathbf{w}\| = r$. To see this, consider a choise of $\mathbf{w}$ that gives $r < 0$. Then $\mathbf{w}^T\Delta\mathbf{w} < 0$ since $\mathbf{w}^T\mathbf{A}\mathbf{w} < 0$ and $\mathbf{w}^T\mathbf{B}\mathbf{w} \geq 0$. This means that $\|\mathbf{w}\|$ will decrease until r becomes positive.

The function $\mathbf{A}\hat{\mathbf{w}} - \mathbf{B}\mathbf{w}$ is illustrated in figure 4.5 together with the Rayleigh quotient plotted to the left in figure 4.1 on page 62.

Finding successive eigenvalues

Since the learning rule defined in equation 4.52 maximizes the Rayleigh quotient in equation 4.2 on page 61, it will find the largest eigenvalue λ_1 and a corresponding eigenvector $\hat{\mathbf{w}}_1 = \pm\hat{\mathbf{e}}_1$ of equation 4.1 on page 61. The question that naturally arises is if, and how, the algorithm can be modified to find the successive eigenvalues and vectors, i.e. the successive solutions to the eigenvalue equation 4.1.

Let $\mathbf{G}$ denote the $n \times n$ matrix $\mathbf{B}^{-1}\mathbf{A}$. Then the n equations for the n eigenvalues solving the eigenproblem in equation 4.1 on page 61 can be written as

$$\mathbf{GE} = \mathbf{ED} \quad \Rightarrow \quad \mathbf{G} = \mathbf{EDE}^{-1} = \sum \lambda_i \hat{\mathbf{e}}_i \mathbf{f}_i^T, \tag{4.54}$$

where the eigenvalues and vectors constitute the matrices $\mathbf{D}$ and $\mathbf{E}$ respectively:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_n \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} | & & | \\ \hat{\mathbf{e}}_1 & \cdots & \hat{\mathbf{e}}_n \\ | & & | \end{bmatrix}, \quad \mathbf{E}^{-1} = \begin{bmatrix} - & \mathbf{f}_1^T & - \\ & \vdots & \\ - & \mathbf{f}_n^T & - \end{bmatrix}. \tag{4.55}$$

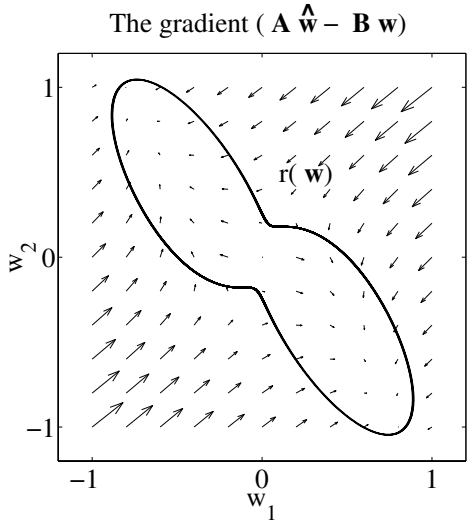

Figure 4.5: The function $\mathbf{A}\hat{\mathbf{w}} - \mathbf{B}\mathbf{w}$, for the same matrices $\mathbf{A}$ and $\mathbf{B}$ as in figure 4.1, plotted for different $\mathbf{w}$. The Rayleigh quotient is plotted as reference.

The vectors, $\mathbf{f}_i$, appearing in the rows of the inverse of the matrix containing the eigenvectors are the *dual vectors* of the eigenvectors $\hat{\mathbf{e}}_i$, which means that

$$\mathbf{f}_i^T \hat{\mathbf{e}}_j = \delta_{ij}. \tag{4.56}$$

$\{\mathbf{f}_i\}$ are also called the *left* eigenvectors of $\mathbf{G}$ and $\{\hat{\mathbf{e}}_i\}$ and $\{\hat{\mathbf{f}}_i\}$ are said to be *biorthogonal*. Remember (from equation 4.6 on page 63) that the eigenvectors $\hat{\mathbf{e}}_i$ are both $\mathbf{A}$ and $\mathbf{B}$ orthogonal, i.e.

$$\hat{\mathbf{e}}_i^T \mathbf{A} \hat{\mathbf{e}}_j = 0 \quad \text{and} \quad \hat{\mathbf{e}}_i^T \mathbf{B} \hat{\mathbf{e}}_j = 0 \quad \text{for} \quad i \neq j. \tag{4.57}$$

Hence, the dual vectors $\mathbf{f}_i$ possessing the property in equation 4.56 can be found by choosing them according to:

$$\mathbf{f}_i = \frac{\mathbf{B}\hat{\mathbf{e}}_i}{\hat{\mathbf{e}}_i^T \mathbf{B} \hat{\mathbf{e}}_i}. \tag{4.58}$$

Now, if $\hat{\mathbf{e}}_1$ is the eigenvector corresponding to the largest eigenvalue of $\mathbf{G}$, the new matrix

$$\mathbf{H} = \mathbf{G} - \lambda_1 \hat{\mathbf{e}}_1 \mathbf{f}_1^T \tag{4.59}$$

has the same eigenvectors and eigenvalues as $\mathbf{G}$ except for the eigenvalue corresponding to $\hat{\mathbf{e}}_1$, which now becomes 0 (see proof B.3.11 on page 164). This means that the eigenvector corresponding to the largest eigenvalue of $\mathbf{H}$ is the same as the one corresponding to the second largest eigenvalue of $\mathbf{G}$.

Since the algorithm starts by finding the vector $\hat{\mathbf{w}}_1 = \hat{\mathbf{e}}_1$, it is only necessary to estimate the dual vector $\mathbf{f}_1$ in order to subtract the correct outer product from $\mathbf{G}$ and remove its largest eigenvalue. In our case, this is a little bit tricky since $\mathbf{G}$ is not generated directly. Instead, its two components $\mathbf{A}$ and $\mathbf{B}$ must be modified in order to produce the desired subtraction. Hence, we want two modified components, $\mathbf{A}'$ and $\mathbf{B}'$, with the following property:

$$\mathbf{B}'^{-1}\mathbf{A}' = \mathbf{B}^{-1}\mathbf{A} - \lambda_1 \hat{\mathbf{e}}_1 \mathbf{f}_1^T. \tag{4.60}$$

A simple solution is obtained if only one of the matrices is modified and the other matrix is kept fixed:

$$\mathbf{B}' = \mathbf{B} \quad \text{and} \quad \mathbf{A}' = \mathbf{A} - \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 \mathbf{f}_1^T. \tag{4.61}$$

This modification can be accomplished by estimating a vector $\mathbf{u}_1 = \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 = \mathbf{B}\mathbf{w}_1$ iteratively as:

$$\mathbf{u}_1(t+1) = \mathbf{u}_1(t) + \Delta\mathbf{u}_1(t) \tag{4.62}$$

where

$$E[\Delta\mathbf{u}_1] = \alpha(r\mathbf{B}\hat{\mathbf{w}}_1 - \mathbf{u}_1). \tag{4.63}$$

Once this estimate has converged, $\mathbf{u}_1 = \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1$ can be used to express the outer product in equation 4.61:

$$\lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 \mathbf{f}_1^T = \frac{\lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T \mathbf{B}^T}{\hat{\mathbf{e}}_1^T \mathbf{B}\hat{\mathbf{e}}_1^T} = \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{e}}_1^T \mathbf{u}_1}. \tag{4.64}$$

Now $\mathbf{A}'$ can be estimated and, hence, a modified version of the learning algorithm in equation 4.52 which finds the second eigenvalue and the corresponding eigenvector to the generalized eigenproblem is obtained:

$$E[\Delta\mathbf{w}] = \alpha \left(\mathbf{A}'\hat{\mathbf{w}} - r\mathbf{B}\hat{\mathbf{w}}\right) = \alpha \left(\left(\mathbf{A} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1}\right)\hat{\mathbf{w}} - r\mathbf{B}\hat{\mathbf{w}}\right). \tag{4.65}$$

The vector $\mathbf{w}_1$ is the solution first produced by the algorithm, i.e. the largest eigenvalue and the corresponding eigenvector.

This scheme can of course be repeated in order to find the third eigenvalue by subtracting the second solution in the same way and so on. Note that this method does not put any demands on the range of $\mathbf{B}$ in contrast to exact solutions involving matrix inversion.

In the following four sub-sections is shown how this iterative algorithm can be applied to the four important problems described in the previous section.

4.7.1 PCA

Finding the largest principal component

The direction of maximum data variation can be found by a stochastic gradient search according to equation 4.53 with $\mathbf{A}$ and $\mathbf{B}$ defined according to equation 4.12:

$$\mathbf{A} = \mathbf{C}_{xx} \quad \text{and} \quad \mathbf{B} = \mathbf{I}. \tag{4.12}$$

This leads to an unsupervised Hebbian learning algorithm that finds both the direction of maximum data variation and the variance of the data in that direction:

$$E[\Delta\mathbf{w}] = \gamma\frac{\partial\rho}{\partial\mathbf{w}} = \alpha\,(\mathbf{C}_{xx}\hat{\mathbf{w}} - \mathbf{w}) = \alpha\,E[\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}} - \mathbf{w}]. \tag{4.66}$$

The update rule for this algorithm is given by

$$\Delta\mathbf{w} = \alpha\,(\mathbf{x}\mathbf{x}^T\hat{\mathbf{w}} - \mathbf{w}), \tag{4.67}$$

where the length of the vector represents the estimated variance, i.e. $\|\mathbf{w}\| = \rho$. (Note that ρ in this case is always positive.)

Note that this algorithm finds both the direction of maximum data variation as well as how much the data vary along that direction. Often algorithms for PCA only find the direction of maximum data variation. If one is also interested in the variation along this direction, another algorithm needs to be employed. This is the case for the well-known PCA algorithm presented by Oja and Karhunen (1985).

Finding successive principal components

In order to find successive principal components, recall that $\mathbf{A} = \mathbf{C}_{xx}$ and $\mathbf{B} = \mathbf{I}$. Hence the matrix $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A} = \mathbf{C}_{xx}$ is symmetric and has orthogonal eigenvectors. This means that the dual vectors and the eigenvectors become indistinguishable and that no other vector than $\mathbf{w}$ itself needs to be estimated. The outer product in equation 4.61 then becomes:

$$\lambda_1\mathbf{B}\hat{\mathbf{e}}_1\mathbf{f}_1^T = \lambda_1\mathbf{I}\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T = \mathbf{w}_1\hat{\mathbf{w}}_1^T. \tag{4.68}$$

This means that the modified learning rule for finding the second eigenvalue can be written as

$$E[\Delta\mathbf{w}] = \alpha\,\left(\mathbf{A}'\hat{\mathbf{w}} - \mathbf{B}\mathbf{w}\right) = \alpha\,\left((\mathbf{C}_{xx} - \mathbf{w}_1\hat{\mathbf{w}}_1^T)\hat{\mathbf{w}} - \mathbf{w}\right). \tag{4.69}$$

A stochastic approximation of this rule is achieved the vector $\mathbf{w}$ is updated by

$$\Delta\mathbf{w} = \alpha\,\left((\mathbf{x}\mathbf{x}^T - \mathbf{w}_1\hat{\mathbf{w}}_1^T)\hat{\mathbf{w}} - \mathbf{w}\right) \tag{4.70}$$

at each time step.

As mentioned in section 4.2, it is possible to perform a PCA on the inverse of the covariance matrix by choosing $\mathbf{A} = \mathbf{I}$ and $\mathbf{B} = \mathbf{C}_{xx}$. The learning rule associated with this behaviour then becomes:

$$\Delta \mathbf{w} = \alpha \, (\hat{\mathbf{w}} - \mathbf{x}\mathbf{x}^T \mathbf{w}). \tag{4.71}$$

4.7.2 PLS

Finding the largest singular value

If the aim is to find the directions of maximum data covariance, the matrices $\mathbf{A}$ and $\mathbf{B}$ are defined according to equation 4.19:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} \quad , \quad \mathbf{B} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \mu_x \hat{\mathbf{w}}_x \\ \mu_y \hat{\mathbf{w}}_y \end{pmatrix}. \tag{4.19}$$

Since $\mathbf{w}$ on average should be updated in the direction of the gradient, the update rule in equation 4.53 gives:

$$E[\Delta \mathbf{w}] = \gamma \frac{\partial r}{\partial \mathbf{w}} = \alpha \left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} \hat{\mathbf{w}} - r \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathbf{w}} \right). \tag{4.72}$$

This behaviour is accomplished if at each time step, the vector $\mathbf{w}$ is updated with

$$\Delta \mathbf{w} = \alpha \left(\begin{bmatrix} \mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0} \end{bmatrix} \hat{\mathbf{w}} - \mathbf{w} \right), \tag{4.73}$$

where the length of the vector at convergence represents the covariance, i.e. $\|\mathbf{w}\| = r = \rho$. This can be done since it is sufficient to search for positive values of ρ.

Finding successive singular values

Also in this case, the special structure of the $\mathbf{A}$ and $\mathbf{B}$ matrices simplifies the procedure of finding the subsequent directions with maximum data covariance. The compound matrix $\mathbf{G} = \mathbf{B}^{-1}\mathbf{A} = \mathbf{A}$ is symmetric and has orthogonal eigenvectors, which are identical to their dual vectors. The outer product for modification of the matrix $\mathbf{A}$ in equation 4.61 is identical to the one presented in the previous section:

$$\lambda_1 \mathbf{B} \hat{\mathbf{e}}_1 \mathbf{f}_1^T = \lambda_1 \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T = \mathbf{w}_1 \hat{\mathbf{w}}_1^T. \tag{4.74}$$

A modified learning rule for finding the second eigenvalue can thus be written as

$$E[\Delta \mathbf{w}] = \alpha \, \left(\mathbf{A}' \hat{\mathbf{w}} - \mathbf{B}\mathbf{w} \right) = \alpha \, \left(\left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} - \mathbf{w}_1 \hat{\mathbf{w}}_1^T \right) \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w} \right). \tag{4.75}$$

A stochastic approximation of this rule is achieved if the vector $\mathbf{w}$ is updated at each time step by

$$\Delta\mathbf{w} = \alpha\left(\left(\begin{bmatrix}\mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0}\end{bmatrix} - \mathbf{w}_1\hat{\mathbf{w}}_1^T\right)\hat{\mathbf{w}} - \mathbf{w}\right). \tag{4.76}$$

4.7.3 CCA

Finding the largest canonical correlation

Again, the algorithm in equation 4.53 for solving the generalized eigenproblem can be used for the stochastic gradient search. With the matrices $\mathbf{A}$ and $\mathbf{B}$ and the vector $\mathbf{w}$ as in equation 4.29:

$$\mathbf{A} = \begin{bmatrix}\mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0}\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}\mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy}\end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix}\mathbf{w}_x \\ \mathbf{w}_y\end{pmatrix} = \begin{pmatrix}\mu_x\hat{\mathbf{w}}_x \\ \mu_y\hat{\mathbf{w}}_y\end{pmatrix} \tag{4.29}$$

the update direction is:

$$E[\Delta\mathbf{w}] = \gamma\frac{\partial r}{\mathbf{w}} = \alpha\left(\begin{bmatrix}\mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0}\end{bmatrix}\hat{\mathbf{w}} - r\begin{bmatrix}\mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy}\end{bmatrix}\hat{\mathbf{w}}\right). \tag{4.77}$$

This behaviour is accomplished if at each time step the vector $\mathbf{w}$ is updated with

$$\Delta\mathbf{w} = \alpha\left(\begin{bmatrix}\mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0}\end{bmatrix}\hat{\mathbf{w}} - \begin{bmatrix}\mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{y}\mathbf{y}^T\end{bmatrix}\mathbf{w}\right). \tag{4.78}$$

Since $\|\mathbf{w}\| = r = \rho$ when the algorithm converges, the length of the vector represents the correlation between the variates.

Finding successive canonical correlations

In the two previous cases it was easy to cancel out an eigenvalue because the matrix $\mathbf{G}$ was symmetric. This is not the case for canonical correlation. In this case

$$\mathbf{G} = \mathbf{B}^{-1}\mathbf{A} = \begin{bmatrix}\mathbf{C}_{xx}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{C}_{xx}^{-1}\mathbf{C}_{xy} \\ \mathbf{C}_{yy}^{-1}\mathbf{C}_{yx} & \mathbf{0}\end{bmatrix}. \tag{4.79}$$

Because of this, it is necessary to estimate the dual vector $\mathbf{f}_1$ corresponding to the eigenvector $\hat{\mathbf{e}}_1$, or rather the vector $\mathbf{u}_1 = \lambda_1\mathbf{B}\hat{\mathbf{e}}_1$ as described in equation 4.63:

$$E[\Delta\mathbf{u}_1] = \alpha\,(\mathbf{B}\mathbf{w}_1 - \mathbf{u}_1) = \alpha\left(\begin{bmatrix}\mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy}\end{bmatrix}\mathbf{w}_1 - \mathbf{u}_1\right). \tag{4.80}$$

A stochastic approximation of this rule is given by

$$\Delta \mathbf{u}_1 = \alpha \left(\begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{y}\mathbf{y}^T \end{bmatrix} \mathbf{w}_1 - \mathbf{u}_1 \right). \tag{4.81}$$

With this estimate, the outer product in equation 4.61 can be used to modify the matrix $\mathbf{A}$:

$$\mathbf{A}' = \mathbf{A} - \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 \mathbf{f}_1^T = \mathbf{A} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1}. \tag{4.82}$$

A modified version of the learning algorithm in equation 4.78 which finds the second largest canonical correlation and its corresponding directions can be written on the following form:

$$E[\Delta \mathbf{w}] = \alpha \left(\mathbf{A}' \hat{\mathbf{w}} - \mathbf{B}\mathbf{w} \right) = \alpha \left(\left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1} \right) \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{bmatrix} \mathbf{w} \right). \tag{4.83}$$

Again to get a stochastic approximation of this rule, the update at each time step is performed according to:

$$\Delta \mathbf{w} = \alpha \left(\left(\begin{bmatrix} \mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0} \end{bmatrix} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1} \right) \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{y}\mathbf{y}^T \end{bmatrix} \mathbf{w} \right). \tag{4.84}$$

Note that this algorithm simultaneously finds both the directions of canonical correlations *and* the canonical correlations ρ_i in contrast to the algorithm proposed by Kay (1992), which only finds the directions.

4.7.4 MLR

Finding the directions for minimum square error

Also here, the algorithm in equation 4.53 can be used for a stochastic gradient search. With the $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{w}$ according to equation 4.47:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \mathbf{w}_x \\ \mathbf{w}_y \end{pmatrix} = \begin{pmatrix} \mu_x \hat{\mathbf{w}}_x \\ \mu_y \hat{\mathbf{w}}_y \end{pmatrix}, \tag{4.47}$$

the update direction is:

$$E[\Delta \mathbf{w}] = \gamma \frac{\partial r}{\partial \mathbf{w}} = \alpha \left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} \hat{\mathbf{w}} - r \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \hat{\mathbf{w}} \right). \tag{4.85}$$

This behaviour is accomplished if the vector $\mathbf{w}$ at each time step is updated with

$$\Delta \mathbf{w} = \alpha \left(\begin{bmatrix} \mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0} \end{bmatrix} \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w} \right). \tag{4.86}$$

Since $\|\mathbf{w}\| = r = \rho$ when the algorithm converges, the regression coefficient is obtained as $\beta = \|\mathbf{w}\| \frac{\mu_x}{\mu_y}$.

Finding successive directions for minimum square error

Also in this case, the dual vectors must be used to cancel out the detected eigenvalues. The non-symmetric matrix $\mathbf{G}$ is

$$\mathbf{G} = \mathbf{B}^{-1}\mathbf{A} = \begin{bmatrix} \mathbf{C}_{xx}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{C}_{xx}^{-1}\mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix}. \tag{4.87}$$

Again, the vector $\mathbf{u}_1 = \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1$ is estimated as described in equation 4.63:

$$E[\Delta \mathbf{u}_1] = \alpha \ (\mathbf{B}\mathbf{w}_1 - \mathbf{u}_1) = \alpha \left(\begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w}_1 - \mathbf{u}_1 \right). \tag{4.88}$$

A stochastic approximation for this rule is given by

$$\Delta \mathbf{u}_1 = \alpha \left(\begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w}_1 - \mathbf{u}_1 \right). \tag{4.89}$$

With this estimate, the outer product in equation 4.61 can be used to modify the matrix $\mathbf{A}$:

$$\mathbf{A}^{'} = \mathbf{A} - \lambda_1 \mathbf{B}\hat{\mathbf{e}}_1 \mathbf{f}_1^T = \mathbf{A} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1}. \tag{4.90}$$

A modified version of the learning algorithm in equation 4.86 which finds the successive directions of minimum square error and their corresponding regression coefficient can be written on the following form:

$$E[\Delta \mathbf{w}] = \alpha \left(\mathbf{A}^{'} \hat{\mathbf{w}} - \mathbf{B}\mathbf{w} \right) = \alpha \left(\left(\begin{bmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{bmatrix} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1} \right) \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w} \right). \tag{4.91}$$

Again, to get a stochastic approximation of this rule, the update at each time step is performed according to:

$$\Delta \mathbf{w} = \alpha \left(\left(\begin{bmatrix} \mathbf{0} & \mathbf{x}\mathbf{y}^T \\ \mathbf{y}\mathbf{x}^T & \mathbf{0} \end{bmatrix} - \frac{\mathbf{u}_1 \mathbf{u}_1^T}{\hat{\mathbf{w}}_1^T \mathbf{u}_1} \right) \hat{\mathbf{w}} - \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{w} \right). \tag{4.92}$$

As mentioned earlier, the $\mathbf{w}_y$s are orthogonal in this case. This means that this method can be used for successively building up a low-rank approximation of MLR by adding a sufficient number of solutions, i.e.

$$\tilde{\mathbf{y}} = \sum_{i=1}^{N} \beta_i \mathbf{x}^T \hat{\mathbf{w}}_{xi} \hat{\mathbf{w}}_{yi}, \tag{4.93}$$

where $\tilde{\mathbf{y}}$ is the estimated $\mathbf{y}$ and N is the rank.

4.8 Experiments

The memory requirement as well as the computational cost per iteration of the presented algorithm is of order $O(Nd)$, where N is the number of estimated models, i.e. the rank of the solution, and d is the dimensionality of the signal. This enables experiments in signal spaces having dimensionalities which would be impossible to handle using traditional techniques involving matrix multiplications (having memory requirements of order $O(d^2)$ and computational costs of order $O(d^3)$).

This section presents some experiments using the algorithm for analysis of stochastic processes. First, the algorithm is employed to perform PCA, PLS, CCA, and MLR. Here, the dimensionality of the signal space is kept reasonably low in order to make a comparison with the performance of an *optimal* (in the sense of maximum likelihood (ML)) deterministic solution which is calculated for each iteration, based on the data accumulated so far.

In the final experiment, the algorithm is applied to a process in a high-dimensional (1,000 dimensions) signal space. In this case, the update factor is made data dependent and the output from the algorithm is post-filtered in order to meet requirements of quick convergence together with algorithm robustness.

The errors in magnitude and angle were calculated relative the correct answer $\mathbf{w}_c$. The same error measures were used for the output from the algorithm as well as for the ML estimate:

$$\varepsilon_m(\mathbf{w}) = \|\mathbf{w}_c\| - \|\mathbf{w}\| \tag{4.94}$$

$$\varepsilon_a(\mathbf{w}) = \arccos(\hat{\mathbf{w}}^T \hat{\mathbf{w}}_c). \tag{4.95}$$

4.8.1 Comparisons to optimal solutions

The test data for these four experiments were generated from a 30-dimensional Gaussian distribution such that the eigenvalues of the generalized eigenproblem decreased exponentially from 0.9:

$$\lambda_i = 0.9 \left(\frac{2}{3}\right)^{i-1}.$$

The two largest eigenvalues (0.9 and 0.6) and the corresponding eigenvectors were simultaneously searched for. In the PLS, CCA and MLR experiments, the dimensionalities of the signal vectors belonging to the $\mathbf{x}$ and $\mathbf{y}$ parts of the signal were 20 and 10 respectively.

The average angular and magnitude errors were calculated based on 10 different runs. This computation was made for each iteration, both for the algorithm

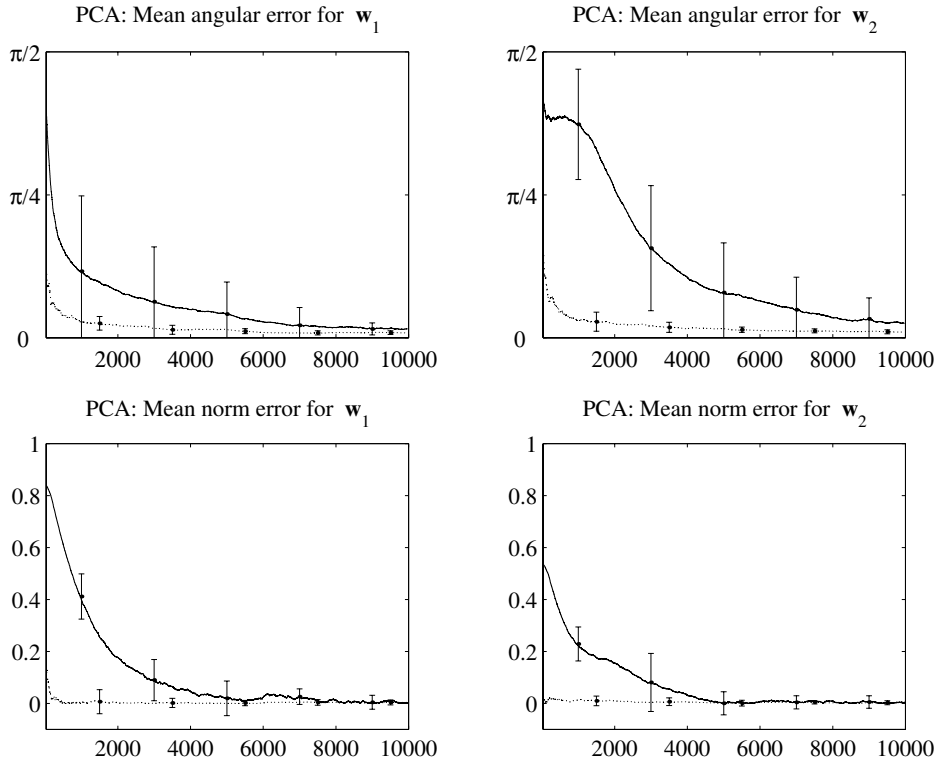

Figure 4.6: Results for the PCA case.

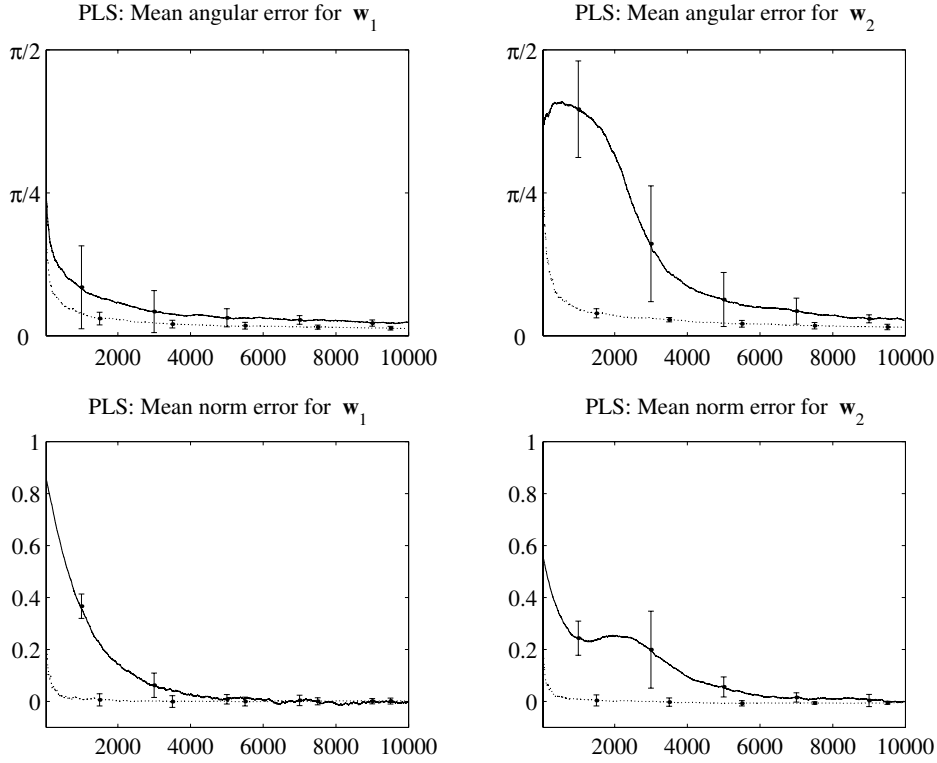

Figure 4.7: Results for the PLS case.

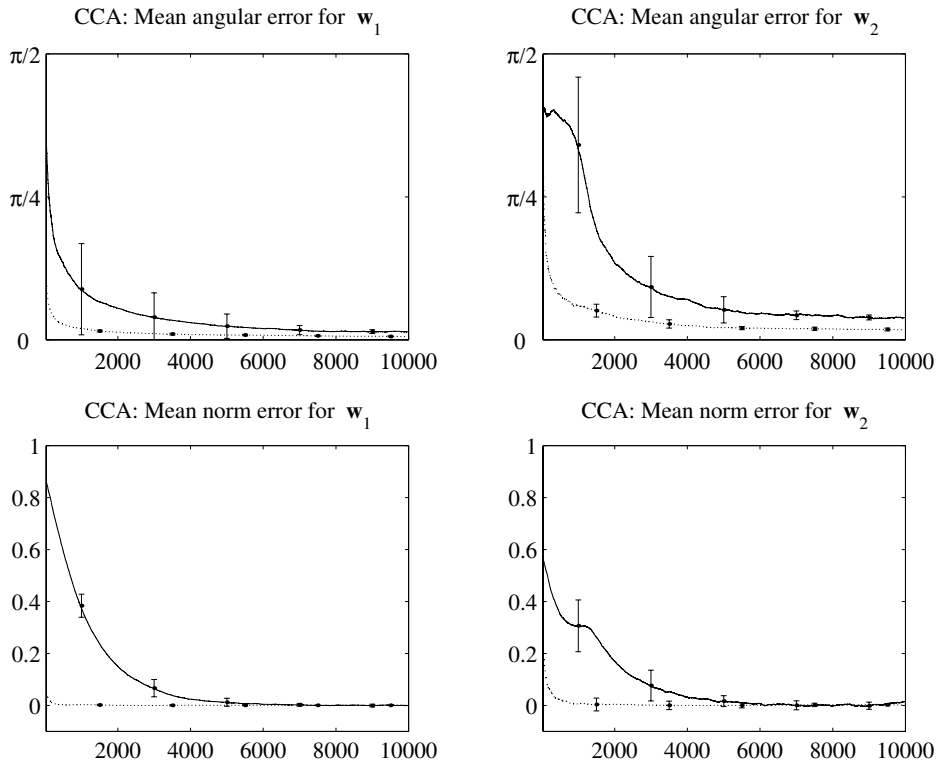

Figure 4.8: Results for the CCA case.

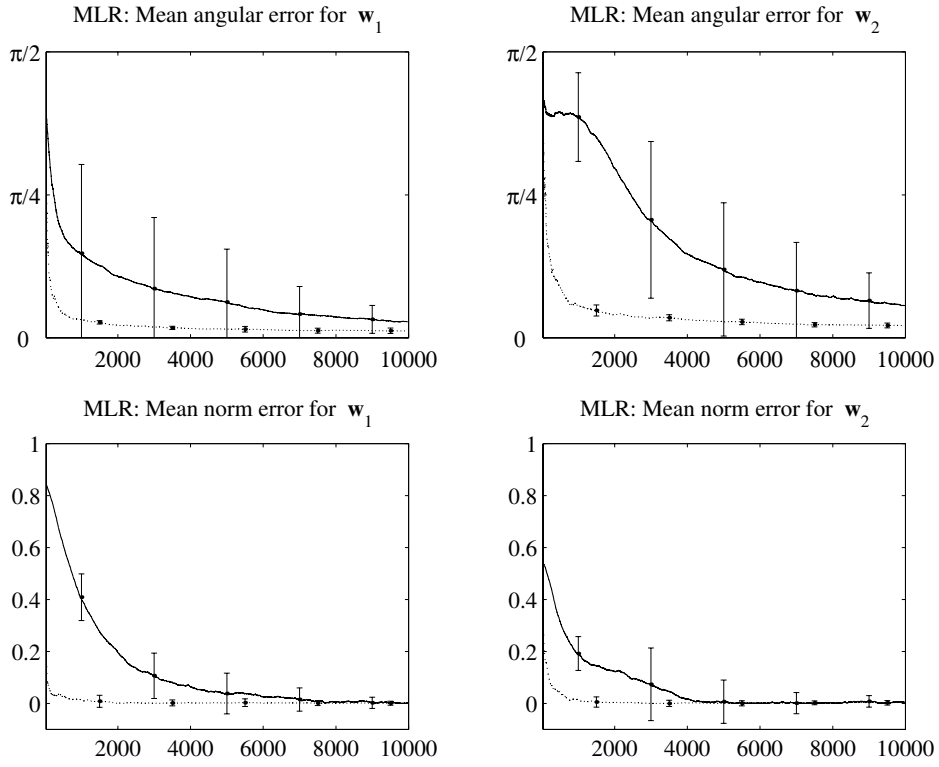

Figure 4.9: Results for the MLR case.

and for the ML solution. The results are plotted in figures 4.6, 4.7, 4.8 and 4.9 for PCA, PLS, CCA and MLR respectively.

The errors of the algorithm are drawn with solid lines and the errors of the ML solution are drawn with dotted lines. The vertical bars show the standard deviations. Note that the angular error is always positive and, hence, does not have a symmetrical distribution. However, for simplicity, the standard deviation indicators have been placed symmetrically around the mean. The first 30 iterations were omitted to avoid singular matrices when calculating matrix inverses for the ML solutions.

No attempt was made to find an optimal set of parameters for the algorithm. Instead, the experiments and comparisons were carried out only to display the behaviour of the algorithm and to show that it is robust and converges to the correct solutions. Initially, the estimate was assigned a small random vector. A constant gain factor of $\alpha = 0.001$ was used throughout all four experiments.

4.8.2 Performance in high-dimensional signal spaces

The purpose of the methods discussed in this chapter is dimensionality reduction in high-dimensional signal spaces. We have previously shown that the proposed algorithm has the computational capacity to handle such signals. This experiment illustrates that the algorithm behaves well also in practice for high-dimensional signals. The dimensionality of $\mathbf{x}$ is 800 and the dimensionality of $\mathbf{y}$ is 200, so the total dimensionality of the signal space is 1,000. The object in this experiment is CCA.

In the previous experiment, the algorithm was used in its basic form with constant update rates set by hand. In this experiment, however, a more sophisticated version of the algorithm is used where the update rate is adaptive and the vectors are averaged over time. The details of this extension of the algorithm are numerous and beyond the scope of this thesis. Only a brief explanation of the basic structure of the extended algorithm is given here. The algorithm can be described in terms of four blocks as illustrated in figure 4.10.

The first block, $\Delta\mathbf{w}$, calculates the delta-vectors according to

$$\begin{cases} \Delta\mathbf{w}_x &= (\mathbf{y}^T\hat{\mathbf{w}}_y - \mathbf{x}^T\mathbf{w}_x)\mathbf{x} \\ \Delta\mathbf{w}_y &= (\mathbf{x}^T\hat{\mathbf{w}}_x - \mathbf{y}^T\mathbf{w}_y)\mathbf{y}. \end{cases} \tag{4.96}$$

The difference between this update rule and the update rule in 4.78 on page 84 is that here the two delta-vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ are calculated separately. But the update rule can still be identified as the gradient of ρ in equation 4.25 on page 68 for $\mathbf{w}_x = \frac{\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}\hat{\mathbf{w}}_x$ and $\mathbf{w}_y = \frac{\hat{\mathbf{w}}_y^T\mathbf{C}_{yx}\hat{\mathbf{w}}_x}{\hat{\mathbf{w}}_y^T\mathbf{C}_{yy}\hat{\mathbf{w}}_y}\hat{\mathbf{w}}_y$.

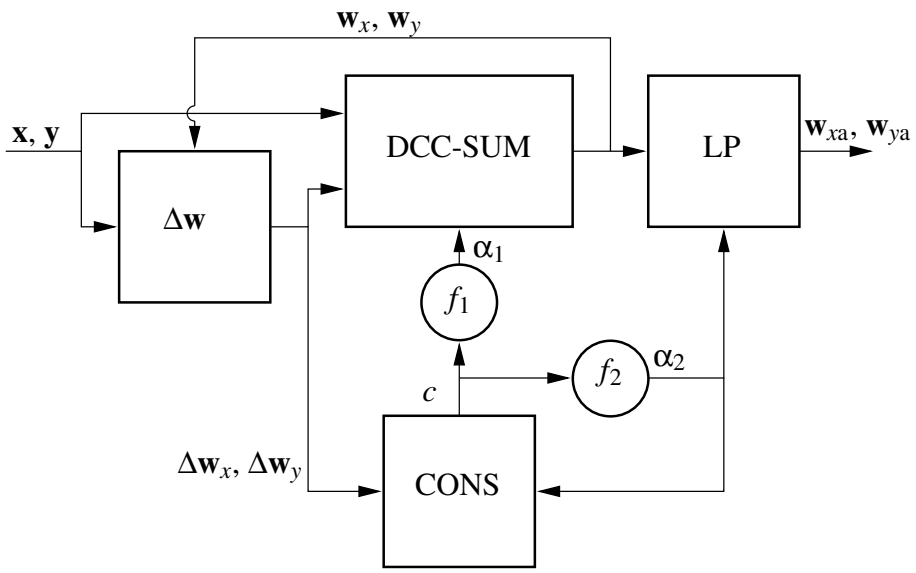

Figure 4.10: The extended CCA-algorithm. See the text for explanations.

The delta vectors are then accumulated in the DCC-SUM block in a way that compensates for the influence of the DC-component of the sample data:

$$\begin{cases} \mathbf{w}_x & = \sum \alpha_{1x} \Delta \mathbf{w}_x - \frac{\sum \alpha_{1x} \mathbf{x} \ \sum \alpha_{1x} (\mathbf{y}^T \hat{\mathbf{w}}_y - \mathbf{x}^T \mathbf{w}_x)}{\sum \alpha_{1x}} \\ \mathbf{w}_y & = \sum \alpha_{1y} \Delta \mathbf{w}_y - \frac{\sum \alpha_{1y} \mathbf{y} \ \sum \alpha_{1y} (\mathbf{x}^T \hat{\mathbf{w}}_x - \mathbf{y}^T \mathbf{w}_y)}{\sum \alpha_{1y}}. \end{cases} \tag{4.97}$$

Note that the sums can be accumulated on-line. Also note that the update factor α_1 can be different for $\mathbf{w}_x$ and $\mathbf{w}_y$.

Finally, the weighted averages $\mathbf{w}_{x\mathrm{a}}$ and $\mathbf{w}_{y\mathrm{a}}$ of $\mathbf{w}_x$ and $\mathbf{w}_y$ respectively are calculated:

$$\begin{cases} \mathbf{w}_{x\mathrm{a}}(t+1) & = \mathbf{w}_{x\mathrm{a}}(t) + \alpha_{2x}(\mathbf{w}_x(t) - \mathbf{w}_{x\mathrm{a}}(t)) \\ \mathbf{w}_{y\mathrm{a}}(t+1) & = \mathbf{w}_{y\mathrm{a}}(t) + \alpha_{2y}(\mathbf{w}_y(t) - \mathbf{w}_{y\mathrm{a}}(t)). \end{cases} \tag{4.98}$$

Adaptability is necessary for a system without a pre-specified (time dependent) update rate α. Here, the adaptive update rate is dependent on the consistency of the change of the vector. The consistency is calculated in the CONS block as

$$c = \|\widetilde{\Delta \mathbf{w}_x}\|, \tag{4.99}$$

where $\widetilde{\Delta \mathbf{w}_x}$ is an estimate of the normalized average delta vector:

$$\widetilde{\Delta \mathbf{w}_x}(t+1) = \widetilde{\Delta \mathbf{w}_x}(t) + \alpha_{2x}\left(\frac{\Delta \mathbf{w}_x}{\|\Delta \mathbf{w}_x\|} - \widetilde{\Delta \mathbf{w}_x}(t)\right). \tag{4.100}$$

A similar calculation of c is made for $\mathbf{w}_x$.

The functions f_1 and f_2 map the consistency c in a suitable way. f_2 increases the sensitivity to changes in c ($\alpha_2 \sim c^2$) and f_1 decreases the sensitivity ($\alpha_1 \sim c^{1/2}$).

When there is a consistent change in $\mathbf{w}$, c is large and the averaging window is short which makes $\mathbf{w_a}$ follow $\mathbf{w}$ quickly. When the changes in $\mathbf{w}$ are less consistent, the window gets longer and $\mathbf{w_a}$ is the average of an increasing number of instances of $\mathbf{w}$. This means, for example, that if $\mathbf{w}$ is moving symmetrically around the correct solution with a constant variance, the error of $\mathbf{w_a}$ will still tend towards zero (see figure 4.11).

The experiment was carried out using a randomly chosen distribution of a 800-dimensional $\mathbf{x}$ variable and a 200-dimensional $\mathbf{y}$ variable. Two $\mathbf{x}$ and two $\mathbf{y}$ dimensions were correlated. The other 798 dimensions of $\mathbf{x}$ and 198 dimensions of $\mathbf{y}$ were uncorrelated. The variances in the 1000 dimensions were of the same order of magnitude.

The upper plot in figure 4.11 shows the estimated first canonical correlation as a function of number of iterations (solid line) and the true correlation in the current directions found by the algorithm (dotted line). Note that each iteration gives one sample.

The lower plot in figure 4.11 shows the effect of the adaptive averaging. The two upper noisy curves show the logarithms of angular errors of the 'raw' estimates $\mathbf{w}_x$ and $\mathbf{w}_y$ and the two lower curves show the angular errors for $\mathbf{w}_{x\mathrm{a}}$ (dashed) and $\mathbf{w}_{y\mathrm{a}}$ (solid). The angular errors of the smoothed estimates are much more stable and decrease more rapidly than the 'raw' estimates. The errors after 2×10^5 samples are below one degree. (It should be noted that this is extreme precision as, with a resolution of 1 degree, a low estimate of the number of different orientations in a 1000-dimensional space is 10^{2000}.) The angular errors were calculated as the angle between the vectors and the exact solutions $\hat{\mathbf{e}}$ (known from the $\mathbf{x}\,\mathbf{y}$ sample distribution), i.e.

$$Err[\hat{\mathbf{w}}] = \arccos(\hat{\mathbf{w}}_{\mathrm{a}}^T \hat{\mathbf{e}}).$$

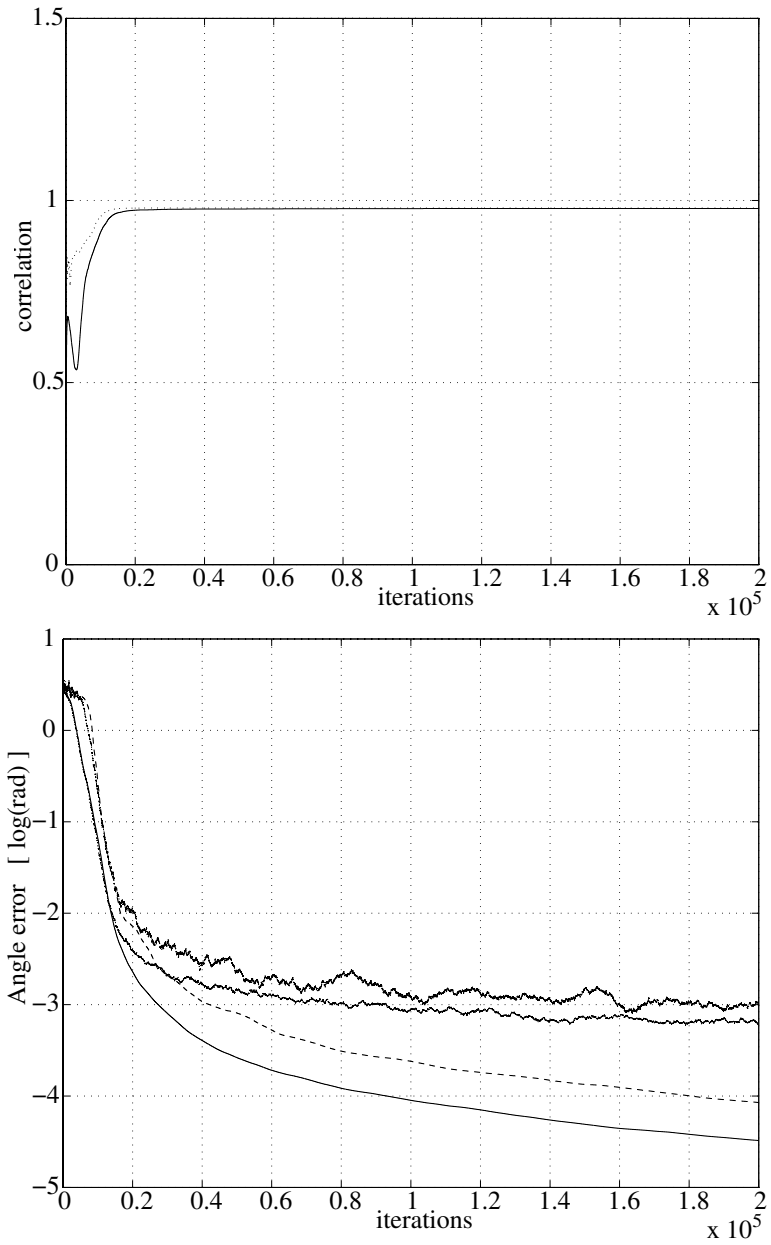

Figure 4.11: **Top:** The estimated first canonical correlation as a function of number of iterations (solid line) and the true correlation in the current directions found by the algorithm (dotted line). The dimensionality of one set of variables is *800* and of the second set *200*. **Bottom:** The logarithm of the angular error as a function of number of iterations.

Part II

Applications in computer vision

Chapter 5

Computer vision

In this part of this dissertation is shown how local linear adaptive models based on canonical correlation can be used in computer vision. This chapter serves as an introduction by giving a brief overview of the parts of the theory and terminology of computer vision relevant to the remaining chapters. For an extensive treatment of this subject, see (Granlund and Knutsson, 1995).

5.1 Feature hierarchies

An image in a computer is usually represented by an array of picture elements (pixels), each one containing a gray level value or a colour vector. The images referred to in this thesis are gray scale images. The pixel values can be seen as image features on the lowest level. On a higher level, there are for example the *orientation* and *phase* of one-dimensional events such as lines and edges. On the next level, the *curvature* describes the change of orientation. On still higher levels, there are features like shape, relations between objects, disparity et cetera. It is, of course, not obvious how to sort complex features into different levels. But, in general, it can be assumed that a function that estimates the values of a certain feature uses features of a lower level as input. High-level features are often estimated on a larger spatial scale than low-level features.

Low-level features (e.g. orientation) are usually estimated by using fairly simple combinations of linear filter outputs. These filter outputs are generated by *convolving* (see for example Bracewell, 1986) the image with a set of filter kernels. The filter coefficients can be described as a vector and so can each region of the image. Hence, for each position in the image, the filter output can be seen as a scalar product between a (reversed) filter vector and a signal vector.

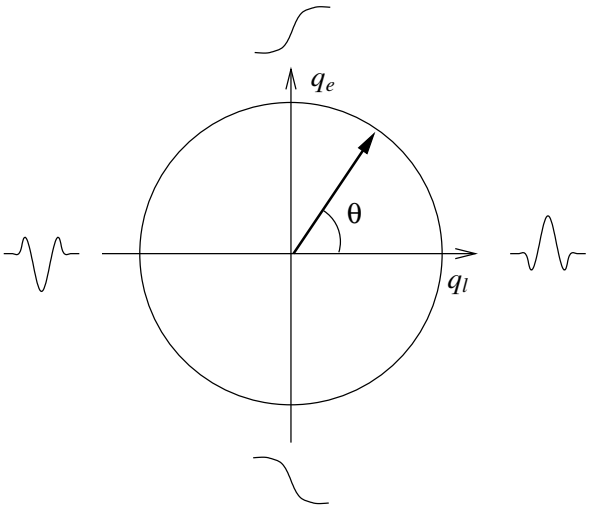

Figure 5.1: The phase representation of a line/edge event.

5.2 Phase and quadrature filters

Consider a half period of a cosine wave. It can illustrate the cross section of a white[1] line if it is centred around 0 and a dark line if it is centred around π. If it is centred around $\pi/2$ or $3\pi/2$ it can illustrate lines of opposite slopes. This leads to the concept of phase. To represent the kind of line/edge event in question, a phase angle θ can be used as illustrated in figure 5.1. If the line and edge filters are designed so that they are orthogonal, their outputs, q_l and q_e respectively, can be combined geometrically so that the magnitude

$$|q| = \sqrt{q_l^2 + q_e^2} \tag{5.1}$$

indicates the presence of a line or an edge of a certain orientation and the argument

$$\theta = \arctan(q_e/q_l) \tag{5.2}$$

represents the kind of event in question, i.e. the phase.

A filter that fits this representation can be obtained as a complex filter consisting of a real-valued line filter and an imaginary edge filter:

$$q = q_l + iq_e. \tag{5.3}$$

[1] White is here represented by the highest value and black is represented by the lowest value.

The magnitude is then the magnitude of the complex filter output q and the phase is the complex argument of q. If the magnitude is invariant with respect to the phase when applied on a pure sine wave function, the filter is said to be a *quadrature filter*. A quadrature filter has zero DC component and is zero in one half-plane in the frequency domain. An example of a quadrature filter is shown in figure 7.5 on page 130. It should be noted that the phase can only be defined after defining a direction in which to measure the phase.

5.3 Orientation

According to the assumption of local one-dimensionality (see page 52), it can be assumed that a small region of an image generally contains at most one dominant orientation. This orientation can be detected by using a set of at least three quadrature filters evenly spread out over all orientations (Knutsson, 1982). Here, the channel representation discussed in section 3.1 can be recognized. The orientation can be represented by a pure channel vector. If four filter orientations are used, the pure channel vector is

$$\mathbf{q} = \begin{pmatrix} |q_1| \\ |q_2| \\ |q_3| \\ |q_4| \end{pmatrix}. \tag{5.4}$$

By choosing a $\cos^2$ shape with proper width of the filter functions as described in section 3.1, the channel vector has a constant norm for all orientations. If four filter orientations are used, each channel looks like

$$|q_k| = d\cos^2(\varphi_k - \phi), \quad \varphi_k = (k-1)\frac{\pi}{4}, \tag{5.5}$$

where φ_k is the filter orientation, ϕ is the line or line or edge orientation and d is an orientation invariant component. By using this set of channels, a more compact orientation vector can be composed:

$$\mathbf{z} = \begin{pmatrix} |q_1| - |q_3| \\ |q_2| - |q_4| \end{pmatrix}. \tag{5.6}$$

Inserting equation 5.5 into equation 5.6 gives

$$\mathbf{z} = a \begin{pmatrix} \cos(2\phi) \\ \sin(2\phi) \end{pmatrix}, \tag{5.7}$$

where a is an orientation invariant component. This orientation representation is called *double angle representation* (Granlund, 1978). The advantage with this

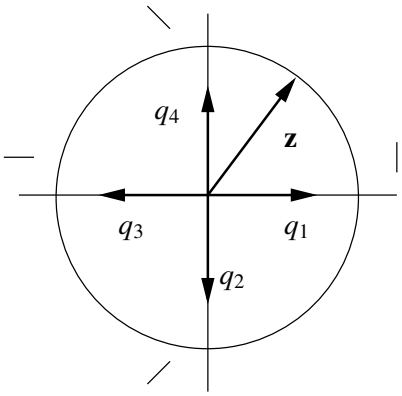

Figure 5.2: The double angle representation.

representation can be seen when considering the rotation of a line. A line is identical if it is rotated 180°. Since $\mathbf{z}$ rotates 360° as a line rotates 180°, this gives an unambiguous and continuous representation of the orientation as illustrated in figure 5.2. The norm and the orientation of the orientation vector $\mathbf{z}$ represent different independent features. While the argument of $\mathbf{z}$ represents the orientation of the signal, the norm depends on the energy of the signal in the passband of the filters.

The double angle representation enables *vector averaging* of the orientation estimates. Vector averaging is usually performed to get a more robust orientation estimate in a larger region of the image. Vector averaging is a geometrical summation of the vectors followed by a normalization:

$$\bar{\mathbf{v}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i. \tag{5.8}$$

The sum of inconsistently oriented vectors is shorter than the sum of vectors with similar directions. This means that the norm of the average vector can be interpreted as a kind of variance, or certainty, measure. This is an important difference between vector averaging and an ordinary geometric scalar average. If the vector average is normalized using the average norm, i.e.

$$\bar{\mathbf{v}} = \frac{\sum_{i=1}^{n}\mathbf{v}_i}{\sum_{i=1}^{n}\|\mathbf{v}_i\|}, \tag{5.9}$$

the certainty measure lies between 0 and 1 where 1 means that all vectors have the same orientation.

5.4 Frequency

Since the norm of **z** depends on the frequency content of the signal, it can be used for estimating local (spatial) frequency. While frequency is only strictly defined for stationary signals, which do not hold for most physical signals, the concept of *instantaneous frequency* (Carson and Fry, 1937; van der Pol, 1946) is usually defined as the rate of change of the phase of the *analytical signal* (see for example Bracewell, 1986; Granlund and Knutsson, 1995).

The instantaneous frequency can be estimated using the ratio between the output of two *lognormal* quadrature filters (Knutsson, 1982). The radial function of a lognormal filter is defined in the frequency domain by

$$R_i(f) = e^{-C_B \ln^2(f/f_i)}, \tag{5.10}$$

where $f = \|\mathbf{u}\|$ is the norm of the frequency vector, f_i is the centre frequency and $C_B = 4/(B^2 \ln 2)$ where B is the 6 dB relative bandwidth. Function 5.10 is a Gaussian on a logarithmic scale. The instantaneous frequency can now be estimated as

$$\omega_i = \frac{|q_{i+1}|}{|q_i|}, \tag{5.11}$$

where $q_i = \|\mathbf{q}_i\|$ is the (orientation invariant) norm of the quadrature filter vector of centre frequency f_i and the difference between f_i and f_{i+1} is one octave (i.e. a factor two). An example of this is illustrated in figure 5.3 where the frequency function of two such lognormal filters are plotted (solid curves) together with the quotient in equation 5.11 (dashed line). See Granlund and Knutsson (1995) for further details.

To estimate local frequencies in a wider range than that covered by the passbands of two filters, a weighted sum of instantaneous frequencies can be used:

$$\tilde{f} = \left(\sum_{i=0}^{N-1} |q_i| \right)^{-1} \sum_{i=0}^{N-1} |q_{i+1}| \sqrt{f_i f_{i+1}}, \tag{5.12}$$

where $f_{i+1} = 2 f_i$.

Also the frequency can be represented by a vector as illustrated in figure 5.4. This enables vector averaging of the frequency estimates too.

5.5 Disparity

An important feature of binocular vision systems is *disparity*, which is a measure of the shift between two corresponding neighbourhoods in a pair of stereo images.

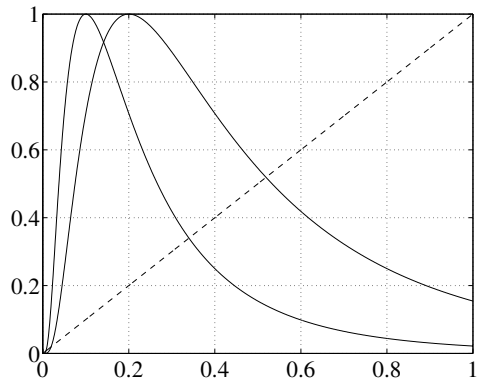

Figure 5.3: The local frequency (dashed line) estimated as a quotient between the magnitude of two lognormal quadrature filter outputs. The centre frequencies of the filters differ one octave.

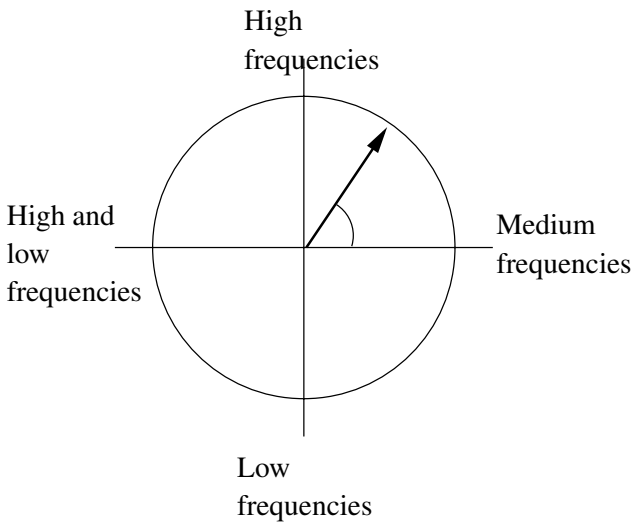

Figure 5.4: The vector representation of frequency.

The disparity is related to the angle the eyes (cameras) must be be rotated relative to each other in order to focus on the same point in the 3-dimensional outside world. The corresponding process is known as *vergence*.

The problem of estimating disparity between pairs of stereo images is not a new one (Barnard and Fichsler, 1982). Early approaches often used matching of some feature in the two images (Marr, 1982). The simplest way to calculate the disparity is to correlate a region in one image with all horizontally shifted regions on the same vertical position and then to find the shift that gave maximum correlation. This is, however, a computationally very expensive method. Since vergence implies a vision system acting in real time, other methods must be employed.

Later approaches have been more focused on using the phase information given by for example Gabor or quadrature filters (Sanger, 1988; Wilson and Knutsson, 1989; Jepson and Fleet, 1990; Westelius, 1995). An advantage of phase-based methods is that phase is a continuous variable that allows for sub-pixel accuracy. In phase-based methods, the disparity can be estimated as a ratio between the phase difference between corresponding vertical line/edge filter outputs from the two images and the instantaneous frequency:

$$\Delta x = \frac{\Delta\phi}{\phi'}, \tag{5.13}$$

where $\phi' = \omega$ is the instantaneous frequency.

Phase-based stereo methods require the filters to be large enough to cover the same structure in the two images, i.e. the shift must be small compared to the wavelength of the filter. Otherwise, the phase difference $\Delta\phi$ will not be related to the shift. On the other hand, if the shift is too small compared to the wavelength of the filter, the resolution becomes poor which leads to a bad disparity estimate. The disparity algorithm proposed by Wilson and Knutsson (1989) handles this problem by working on a *scale pyramid* of different image resolutions. It starts by estimating the disparity on a coarse scale, which corresponds to using low-frequency filters, and adjusting the cameras to minimize this disparity. This process is then iterated on consecutively finer scales.

A problem that is not solved by that approach is when the observed surface is tilted in depth so that the depth varies along the horizontal axis. In this situation, the surface will be viewed at different scales by the two cameras as illustrated in figure 5.5. This means that phase information on one scale in the left image must be compared with phase information on another scale in the right image. In most stereo algorithms, this problem cannot be handled in a simple way.

Another problem that most stereo algorithms are faced with occurs at vertical depth discontinuities (but see Becker and Hinton (1993)). Around the discontinuity there is a region where the algorithm either will not be able to make an estimate

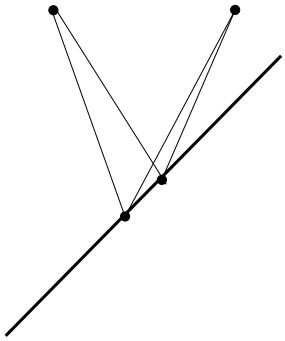

Figure 5.5: Scaling effect when viewing a tilted plane.

at all, or the estimate will be some average between the two correct disparities, indicating a slope rather than a step.

Chapter 6

Learning feature descriptors

In this chapter is shown how canonical correlation analysis can be used to find models that represent local features in images. Such models can be seen as filters that describe a particular feature in an image. The filters can also be forced to be invariant with respect to certain other features. The features to be described are learned by giving the algorithm examples that are presented in pairs. The pairs are arranged in such a way that the property of a certain feature, for example the orientation of a line, is equal for each pair while other properties, for example phase, are presented in an unordered way. This method was presented at SCIA'97 (Borga et al., 1997a).

The idea behind this approach is to use CCA to analyse two signals where the common signal components are due to the feature that is to be represented, as illustrated in figure 6.1. The signal vectors fed into the CCA are image data mapped through some function f. If f is the identity operator (or any other full-rank linear function), the CCA finds the linear combinations of pixel data that have the highest correlation. In this case, the canonical correlation vectors can be seen as linear filters. In general, f can be any vector-valued function of the image data, or even different functions f_x and f_y, one for each signal space. The choice of f can be seen as the choice of representation of input data for the canonical correlation analysis. As discussed in chapter 3, the choice of representation is very important for the ability to learn.

The canonical correlation vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ together with the functions f_x and f_y can be seen as filters. The filters that are developed in this way have the property of maximum correlation between their outputs when applied to two image patches where the represented feature varies simultaneously. In other words, the filters maximize the signal to noise ratio between the desired feature and other signals (see section 4.4.2 on page 70).

A more general approach is to try to maximize mutual information instead

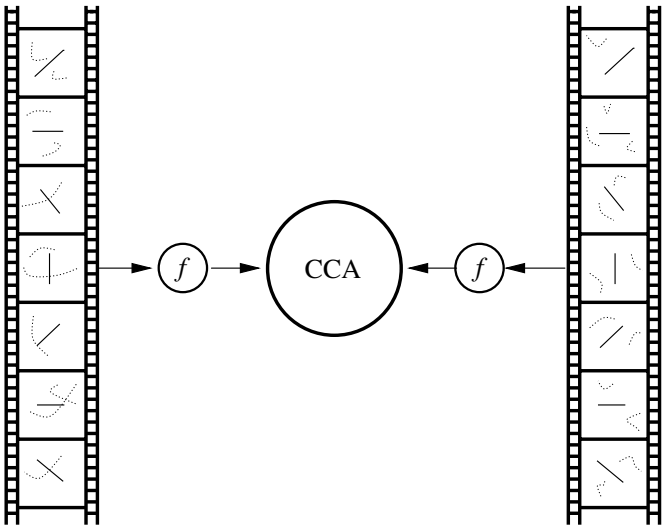

Figure 6.1: A symbolic illustration of the method of using CCA for finding feature detectors in images. The desired feature (here illustrated by a solid line) is varying equally in both image sequences while other features (here illustrated with dotted curves) vary in an uncorrelated way. The input to the CCA is a function f of the image.

of canonical correlation. This could be accomplished by changing not only the linear projection in the CCA, but also the functions f_x and f_y until the maximum correlation ρ is found. This approach relies on the relation between canonical correlation and mutual information discussed in section 4.4.1. The maximum mutual information approach is illustrated in figure 6.2. If f_x and f_y are parameterized functions, the parameters can be updated in order to maximize ρ. This is related to the work of Becker and Hinton (1992) where f was implemented as neural networks with a single neuron in the output layers. The cost function in their approach was the quotient between the variance of the sum and the variance of the difference of the network outputs. The approach illustrated in 6.2 allows f_x and f_y to be implemented as neural networks with several units in their output layers.

In the work presented here, however, the functions are fixed and identical for x and y. In this chapter, f is the outer product of pixel data or the outer product of quadrature filter outputs. But projection operators can also be useful, as will be seen in chapter 7. The motive for choosing non-linear functions here is that we want to find feature descriptors with useful invariance properties. Of course, also a linear filter is invariant to several changes of the signal. It is, for example,

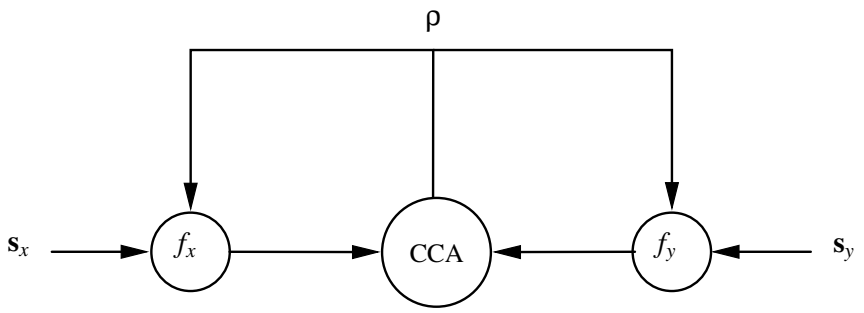

Figure 6.2: A general approach for finding maximum mutual information.

easy to design a linear filter that is invariant with respect to the mean intensity of the image. But higher-order functions can have more interesting invariance properties, as discussed by Giles and Maxwell (1987) and Nordberg et al. (1994). To see this, consider the output q of a linear filter $\mathbf{f}$ for a signal $\mathbf{s}$ in one point: $q = \mathbf{s}^T\mathbf{f}$. The invariance of this filter can be defined as

$$dq = d\mathbf{s}^T\mathbf{f} = 0. \tag{6.1}$$

This means that the changes $d\mathbf{s}$ of the signal for which the linear filter is invariant must be orthogonal to the filter.

Since the invariance properties of linear filters are very limited, it is natural to try second-order functions, which means that f is an outer product of the pixel data. For a quadratic function $\mathbf{F}$, the output can be written as $q = \mathbf{s}^T\mathbf{F}\mathbf{s}$. Here, the invariance is defined by

$$dq = 2d\mathbf{s}^T\mathbf{F}\mathbf{s} = 0. \tag{6.2}$$

This expression can, for example, include the invariances in the linear case if $\mathbf{F} = \mathbf{f}\mathbf{f}^T$. But the quadratic filter can also have invariance properties that depend on the signal $\mathbf{s}$ and not only on the change $d\mathbf{s}$ as in the linear case. An example illustrating the differences between invariances of linear and quadratic functions is illustrated in figure 6.3. In the linear case, the invariances define lines in the two-dimensional case (hyper-planes in general). The lines are orthogonal to $\mathbf{f}$. In the quadratic case, the invariances can define, for example, hyperbolic or parabolic surfaces or ellipsoids. One example of interesting invariance properties of second-order functions is shift or phase invariance when the filter is applied on a sine wave pattern. This is the case for the norm of the output from a pair of quadrature filters, which is a quadratic function of the pixel data.

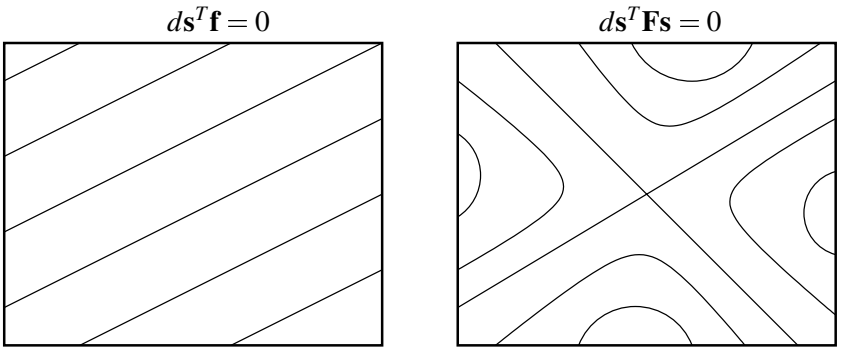

Figure 6.3: Examples of invariances for linear (left) and quadratic (right) two-dimensional functions. The lines are iso-curves on which the function is constant. A change of the parameter vector $\mathbf{s}$ along the lines will not change the output.

6.1 Experiments

If f is an outer product and the image pairs contain sine wave patterns with equal orientations but different phase, the CCA should find a linear combination of the outer products that is sensitive with respect to the orientation and invariant with respect to phase. As illustrated in the experiments below, this is also what happens. The outer products weighted by the canonical correlation vectors can be interpreted as outer products of linear filters. As shown in the experiment, these linear filters are approximately quadrature filters, which explains the phase invariance of the product. The findings of quadrature filters in the interpretation of the result of the CCA can serve as a motive for trying products of quadrature filter outputs as input to CCA on a higher level.

To simplify the description, two functions are used to reshape a matrix into a vector and the other way around: $\mathrm{vec}(\mathbf{M})$ transforms (flattens) an $m \times n$ matrix $\mathbf{M}$ into a vector with mn components (see definition A.1) and $\mathrm{mtx}(\mathbf{v}, m, n)$ reshapes the vector $\mathbf{v}$ into an $m \times n$ matrix (see definition A.2). In particular, for an $m \times n$ matrix $\mathbf{M}$,

$$\mathrm{mtx}(\mathrm{vec}(\mathbf{M}), m, n) = \mathbf{M}. \tag{6.3}$$

6.1.1 Learning quadrature filters

The first experiment shows that quadrature filters are found by the method discussed above when products of pixel data are presented to the algorithm.

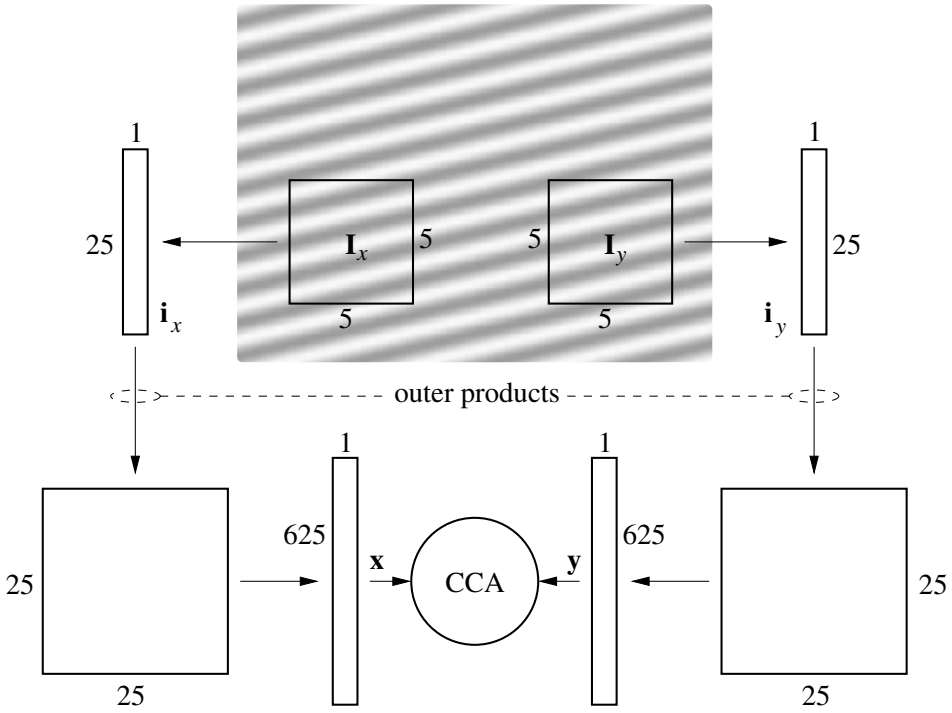

Figure 6.4: Illustration of the generation of input data vectors **x** and **y** as outer products of pixel data. See the text for a detailed explanation.

Let $\mathbf{I}_x$ and $\mathbf{I}_y$ be a pair of 5×5 image patches. Each image consists of a sine wave pattern with a frequency of $\frac{2\pi}{5}$ and additive Gaussian noise. A sequence of such image pairs is constructed so that, for each pair, the orientation is equal in the two images while the phase differs in a random way. The images have independent noise. Each image pair is described by vectors $\mathbf{i}_x = \mathrm{vec}(\mathbf{I}_x)$ and $\mathbf{i}_y = \mathrm{vec}(\mathbf{I}_y)$.

Let **x** and **y** be vectors describing the outer products of the image vectors, i.e. $\mathbf{x} = \mathrm{vec}(\mathbf{i}_x \mathbf{i}_x^T)$ and $\mathbf{y} = \mathrm{vec}(\mathbf{i}_y \mathbf{i}_y^T)$. This gives a sequence of pairs of 625-dimensional vectors describing the products of pixel data from the images. This scheme is illustrated in figure 6.4.

The sequence consists of 6,500 examples, i.e. 20 examples per degree of freedom. (The outer product matrices are symmetric and, hence, the number of free parameters is $\frac{n^2+n}{2}$ where n is the dimensionality of the image vector.) For a signal to noise ratio (SNR) of 0 dB, there were 6 significant[1] canonical correlations and

[1] By significant, we mean that they differ from the random correlations caused by the limited

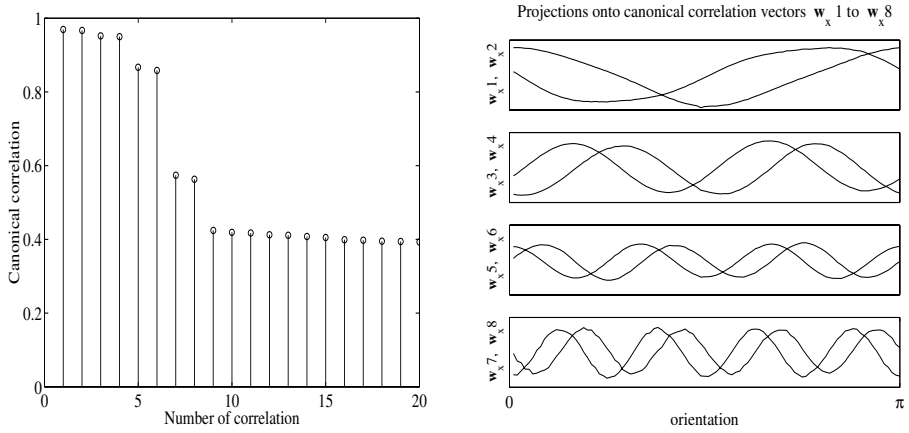

Figure 6.5: **Left:** The 20 largest canonical correlations for the 10 dB SNR sequence. **Right:** Projections of outer product vectors $\mathbf{x}$ onto the 8 first canonical correlation vectors.

for an SNR of 10 dB there were 8 significant canonical correlations. The canonical correlations are plotted to the left in figure 6.5. The two most significant correlations for the 0 dB case were both 0.7 which corresponds to an SNR[2] of 3.7 dB. For the 10 dB case, the two highest correlations were both 0.989, corresponding to an SNR of 19.5 dB.

The projections of image signals $\mathbf{x}$ for orientations between 0 and π onto the 8 first canonical correlation vectors $\mathbf{w}_x$ from the 10 dB case are shown to the right in figure 6.5. The test signals were generated with random phase and without noise. As seen in the figure, the filters defined by the first two canonical correlation vectors are sensitive to the double angle of the orientation of the signal and invariant with respect to phase. The two curves are 90° out of phase and, hence, generate double angle representation (see figure 5.2 on page 102). The following curves show the projections onto the successive canonical correlation vectors with lower canonical correlations. The filters defined by these vectors are sensitive to the fourth, sixth and eighth multiples of the orientation.

set of samples. The random correlations, in the case of 20 samples per degree of freedom, is approximately 0.4 (given by experiments).

[2]The relation between correlation and SNR in this case is defined by the correlation between two signals with the same SNR, i.e. $\text{corr}(s+\eta_1, s+\eta_2)$. (See section 4.4.2.)

Interpretation of the result

It is not easy to interpret the 625 coefficients in each canonical correlation vector. But since the data actually were generated as outer products, i.e. 25×25 matrices, the interpretation of the resulting canonical correlation vectors can be facilitated by writing them as 25×25 matrices, $\mathbf{W}_x = \text{mtx}(\mathbf{w}_x, 25, 25)$. This means that the projection of $\mathbf{x}$ onto a canonical correlation vector $\mathbf{w}_x$ can be written as

$$\mathbf{x}^T \mathbf{w}_x = \mathbf{i}_x^T \mathbf{W}_x \mathbf{i}_x, \tag{6.4}$$

where $\mathbf{i}_x$ is the pixel data vector. By an eigenvalue decomposition of $\mathbf{W}_x$, this projection can be written as

$$\mathbf{x}^T \mathbf{w}_x = \mathbf{i}_x^T \left(\sum_j \lambda_j \mathbf{e}_j \mathbf{e}_j^T \right) \mathbf{i}_x = \sum_j \lambda_j \left(\mathbf{i}_x^T \mathbf{e}_j \right)^2, \tag{6.5}$$

i.e. a square sum of the pixel data vector projected onto the eigenvectors of $\mathbf{W}_x$ weighted with the corresponding eigenvalues. This means that the eigenvectors $\mathbf{e}_j$ can be seen as liner filters and the curves plotted to the right in figure 6.5 are weighted square sums of the pixel data vectors projected onto the eigenvectors of the matrices $\mathbf{W}_{xi}$.

It turns out that only a few of the eigenvectors give significant contributions to the projection sum in equation 6.5. This can be seen if the terms in the sum are averaged over all orientations of the signal:

$$m_j = E[\lambda_j \left(\mathbf{i}_x^T \mathbf{e}_j \right)^2]. \tag{6.6}$$

The coefficients m_j measure the average energy picked up by the corresponding eigenvectors and can therefore be seen as significance measures for the different eigenvectors. In figure 6.6, the significance measures m_j for the 25 eigenvectors are plotted for the two first canonical correlation vectors $\mathbf{w}_{x1}$ and $\mathbf{w}_{x2}$.

Since the projections of $\mathbf{x}$ onto the canonical correlation vectors $\mathbf{w}_x$ can be described in terms of projections of pixel data $\mathbf{i}_x$ onto a few 25-dimensional eigenvectors $\mathbf{e}_j$, these eigenvectors can be used to interpret the canonical correlation vectors. Since the image data $\mathbf{i}_x$ are collected from 5×5 neighbourhoods $\mathbf{I}_x$, it is logical to view also the eigenvectors $\mathbf{e}_j$ as 5×5 matrices $\mathbf{E}_j$. These matrices can be called *eigenimages*. The process of extracting eigenimages from a canonical correlation vector is illustrated in figure 6.7. In figure 6.8, the four most significant eigenimages are shown for the first (top) and second (bottom) canonical correlations respectively.

The eigenimages can be interpreted as quadrature filter pairs, i.e. filter pairs that have the same spectrum and differ 90° in phase (see section 5.2). For $\mathbf{w}_{x1}$,

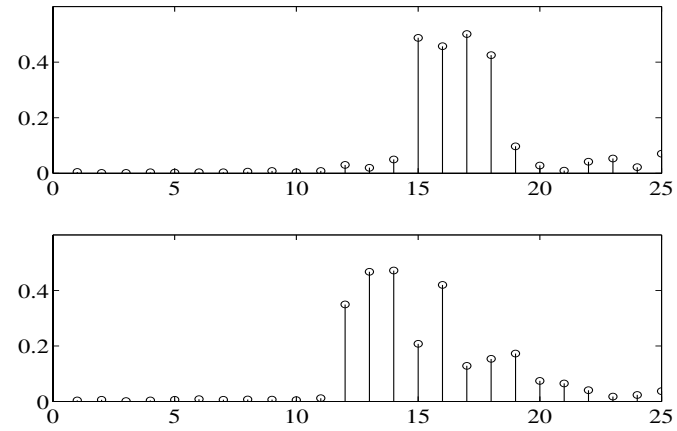

Figure 6.6: The significance measures m_j for the 25 eigenvectors for the two first canonical correlation vectors $\mathbf{w}_{x1}$ (top) and $\mathbf{w}_{x2}$ (bottom).

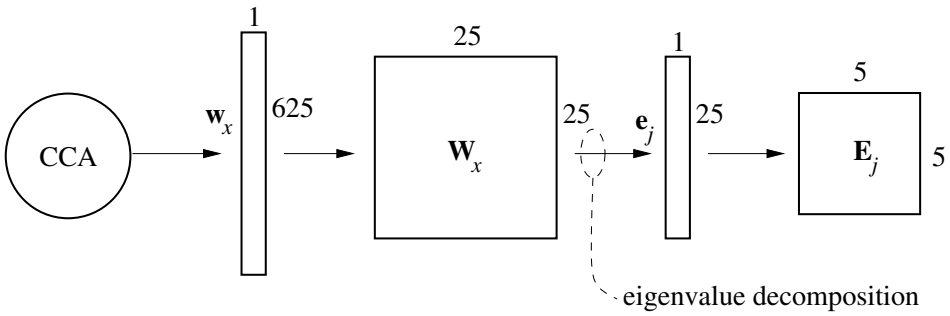

Figure 6.7: Illustration of the extraction of 5×5 eigenimages $\mathbf{E}_j$ from a 625-dimensional canonical correlation vector $\mathbf{w}_x$.

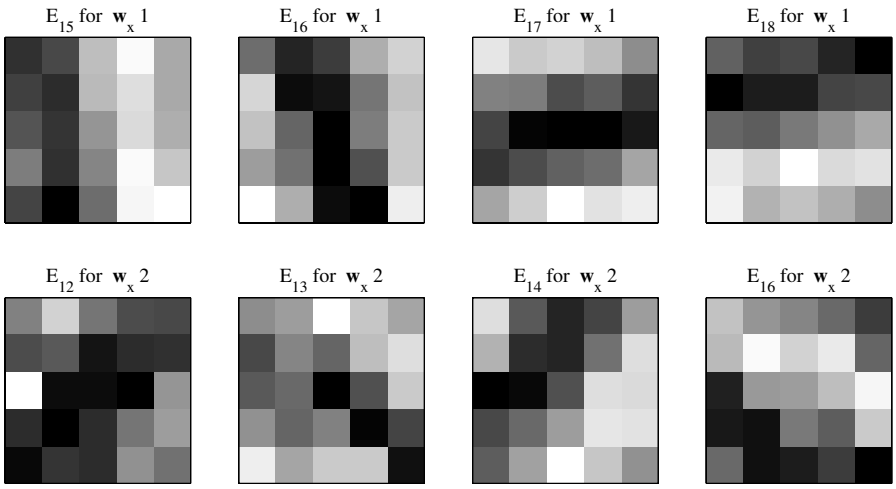

Figure 6.8: The four most significant eigenimages are shown for the first (top row) and second (bottom row) canonical correlations respectively.

eigenimages $\mathbf{E}_{15}$ and $\mathbf{E}_{16}$ form a quadrature pair in one direction and eigenimages $\mathbf{E}_{17}$ and $\mathbf{E}_{18}$ form a quadrature pair in the perpendicular direction. The same interpretation can be made for $\mathbf{w}_{x2}$. To see more clearly that this interpretation is correct, the eigenimage pairs can be combined in the same way as complex quadrature filters, i.e. as one real filter and one imaginary filter with a phase difference of 90°, by multiplying one of the filters[3] with i (see section 5.2, page 101). The spectra of the combinations $\mathbf{E}_{15} + i\mathbf{E}_{16}$ and $\mathbf{E}_{17} + i\mathbf{E}_{18}$ for $\mathbf{w}_{x1}$ are shown in the upper row in figure 6.9. In the lower row, the spectra of the combinations $\mathbf{E}_{12} + i\mathbf{E}_{14}$ and $\mathbf{E}_{13} + i\mathbf{E}_{16}$ for $\mathbf{w}_{x2}$ are shown. The DC-component is in the centre of the spectrum. The white circle illustrates the centre frequency of the training signal. The blobs in the figure show that these eight eigenvectors can be interpreted as four quadrature filter pairs in four different directions.

6.1.2 Combining products of filter outputs

In this experiment, outputs from neighbouring sets of quadrature filters are used rather than pixel values as input to the algorithm. The experimental result shows that canonical correlation can find a way of combining filter outputs from a local neighbourhood to get orientation estimates that are less sensitive to noise than the

[3] Usually, the real filter is symmetric and the imaginary filter is anti-symmetric. However, the choice of offset phase does not matter as long as the filters differ 90° in phase.

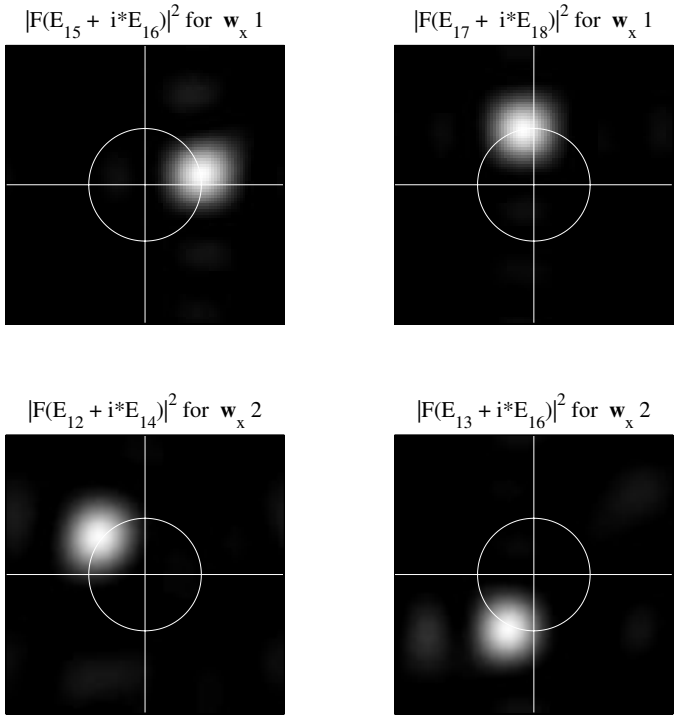

Figure 6.9: Spectra for the eigenimages interpreted as complex quadrature filter pairs.

vector averaging method (see section 5.3 on page 102).

Let $\mathbf{q}_{xi}$ and $\mathbf{q}_{yi}$, $i \in \{1..25\}$, be 4-dimensional complex vectors of filter responses from four quadrature filters from each of 25 different positions in a 5×5 neighbourhood. The quadrature filters used here have kernels of 7×7 pixels, a centre frequency of $\frac{\pi}{2\sqrt{2}}$ and a bandwidth of two octaves. Let $\mathbf{X}_i = \mathbf{q}_{xi}\mathbf{q}_{xi}^*$ and $\mathbf{Y}_i = \mathbf{q}_{yi}\mathbf{q}_{yi}^*$ be the outer products of the filter responses in each position for each image. Finally, all products are gathered into two 400-dimensional vectors:

$$[ht]\mathbf{x} = \begin{pmatrix} \mathrm{vec}(\mathbf{X}_1) \\ \mathrm{vec}(\mathbf{X}_2) \\ \vdots \\ \mathrm{vec}(\mathbf{X}_{25}) \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} \mathrm{vec}(\mathbf{Y}_1) \\ \mathrm{vec}(\mathbf{Y}_2) \\ \vdots \\ \mathrm{vec}(\mathbf{Y}_{25}) \end{pmatrix}. \tag{6.7}$$

This scheme is illustrated in figure 6.10.

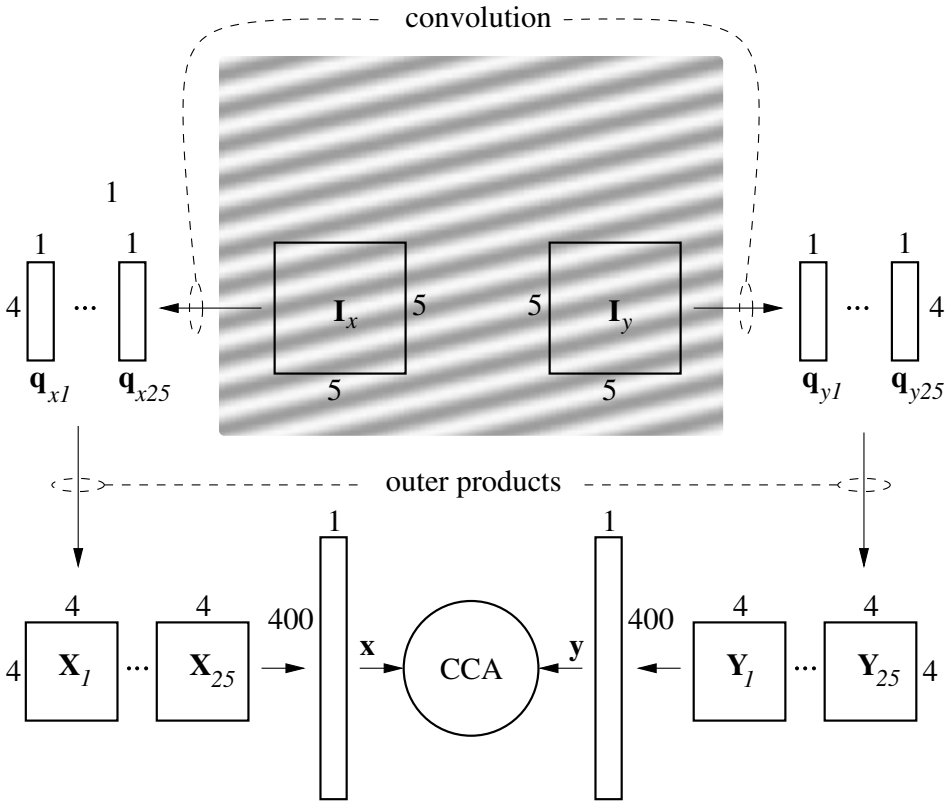

Figure 6.10: Illustration of the generation of input data vectors $\mathbf{x}$ and $\mathbf{y}$ as outer products of quadrature filter response vectors from 5×5 neighbourhoods.

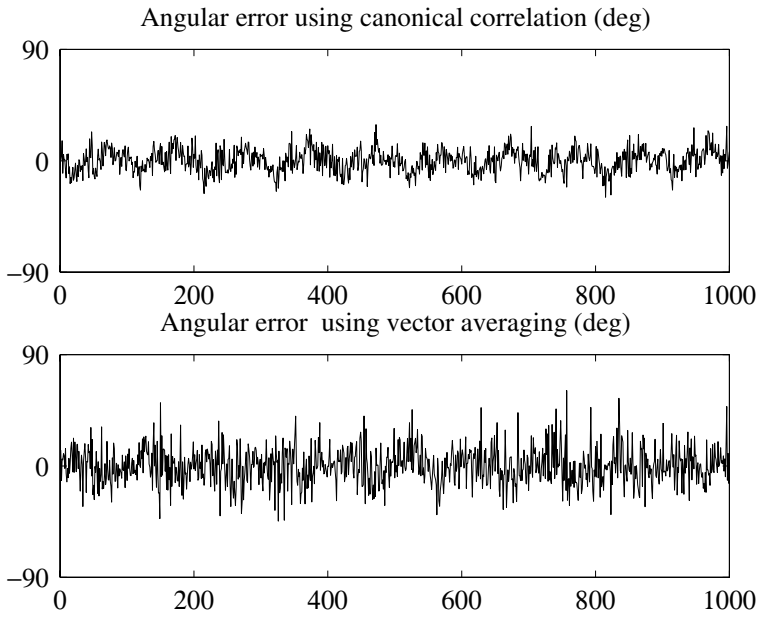

Figure 6.11: Angular errors for 1,000 different samples using canonical correlations (top) and vector averaging (bottom).

8,000 pairs of vectors were generated. For each pair of vectors, the local orientation was equal while the phase and noise differed randomly. Gaussian noise was added to the images giving 0 dB SNR. The data set was analysed using CCA. The two largest canonical correlations were both 0.85. The corresponding vectors detected the double angle of the orientation invariant with respect to phase.

New test data were generated using a rotating sine-wave pattern with an SNR of 0 dB and projected onto the first two canonical correlation vectors. The angular error[4] is shown in the upper plot in figure 6.11. The lower plot shows the angular error using vector averaging on the same data. The standard deviation of the angular error was 9.4° with the CCA method and 14.8° using vector averaging. This is an improvement of the SNR with 4dB compared to the result when using vector averaging on the same neighbourhood size.

[4]The mean angular error is not relevant since it only depends on a reference orientation. The reference orientation can be arbitrarily chosen and, hence, it has been chosen so that the mean angular error is zero.

6.2 Discussion

In this chapter has been shown how a system can learn image feature descriptors by using canonical correlation. A nice property of the method is that the training is done by giving examples of what the user defines as being "equal". In the experiments, sine wave patterns were considered to be "equal" if they had the same orientation, irrespectively of the phase. This was presented to the system as a set of examples and the user did not have to figure out how to represent orientation and phase. In the first experiment, the system developed a phase invariant double angle orientation representation.

This type of learning is of course more useful for higher-level feature descriptors, were it can be difficult to define proper representations of features. Such features are for example corners and line crossings.

In the next chapter, another application of this method is presented, namely disparity estimation, where the horizontal displacement between the images is equal within the training set.

Chapter 7

Disparity estimation using CCA

An important problem in computer vision that is suitable to handle with CCA is stereo vision, since data in this case naturally appear in pairs. In this chapter, a novel stereo vision algorithm that combines CCA and phase analysis is presented. The algorithm has been presented in a paper submitted to ICIPS'98 (Borga and Knutsson, 1998).

For a learning system, the stereo problem is difficult to solve; for small disparities, the high-frequency filters will give the highest accuracy, while for large disparities, the high-frequency filters will be uncorrelated with the disparity and only the low-frequency filters can be used. So the choice of which filters to use for the disparity estimate must be based on a disparity estimate! Furthermore, a general learning system cannot be supposed to know beforehand which inputs come from a certain scale[1]. A solution to this problem is to let the system adapt filters to fit the disparity in question instead of using fixed filters.

The algorithm described here consists of two parts: CCA and phase analysis. Both are performed for each disparity estimate. Canonical correlation analysis is used to create adaptive linear combinations of quadrature filters. These linear combinations are new quadrature filters that are adapted in frequency response and spatial position in order to maximize the correlation between the filter outputs from the two images.

These new filters are then analysed in the phase analysis part of the algorithm. The coefficients given by the canonical correlation vectors are used as weighting coefficients in a pre-computed table that allows for an efficient phase-based search for disparity.

In the following two sections, the two parts of the stereo algorithm are de-

[1] This problem is similar to the problem a system faces when learning to interpret numbers that are represented by one digit on each input. Only the most significant digit will have any correlation with the correct number but the correlation will be weak due to the coarse quantization. Only after this digit is identified, it is possible to detect the use of the next digit.

scribed in more detail. In section 7.3 some experiments are presented to illustrate the performance of the proposed method. Finally, the method is discussed in section 7.4.

7.1 The canonical correlation analysis part

The input $\mathbf{x}$ and $\mathbf{y}$ to the CCA come from the left and right images respectively. Each input is a vector with outputs from a set of quadrature filters:

$$\mathbf{x} = \begin{pmatrix} q_{x1} \\ \vdots \\ q_{xN} \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} q_{y1} \\ \vdots \\ q_{yN} \end{pmatrix}, \tag{7.1}$$

where q_i is the (complex) filter output for the ith quadrature filter in the filter set. The quadrature filters can bee seen as the functions f in figure 6.1 on page 108. In this case, f is a complex vector-valued linear function, i.e. a complex matrix. In the implementation described here, the filter set consists of two identical one-dimensional (horizontal) quadrature filters with two pixels relative displacement. (Other and larger sets of filters can be used including, for example, filters with different bandwidths, different centre frequencies, different positions, etc.)

The data are sampled from a neighbourhood $\mathcal{N}$ around the point of the disparity estimate. The choice of neighbourhood size is a compromise between noise sensitivity and locality. The covariance matrix $\mathbf{C}$ is calculated using the vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathcal{N}$. The fact that quadrature filters have zero mean simplifies this calculation to an outer product sum:

$$\mathbf{C} = \sum_{j \in \mathcal{N}} \begin{pmatrix} \mathbf{x}_j \\ \mathbf{y}_j \end{pmatrix} \begin{pmatrix} \mathbf{x}_j \\ \mathbf{y}_j \end{pmatrix}^* . \tag{7.2}$$

If a rectangular neighbourhood is used, this calculation can be made efficient by a Cartesian separable summation of the outer products as illustrated in figure 7.1. First the outer products are summed in a window moving horizontally for each row. Then this result is summed again by using a window moving vertically for each column. This scheme requires $2 \times m \times n$ additions and subtractions of outer products where $m \times n$ is the size of the image (except for the borders that are not reached by the centre of the neighbourhood). This can be compared to a straightforward summation over each neighbourhood that requires $N \times m \times n$ additions of outer products, where N is the size of the neighbourhood. Hence, for a neighbourhood of 10×10, the separable summation is 50 times faster than a straightforward summation over each neighbourhood.

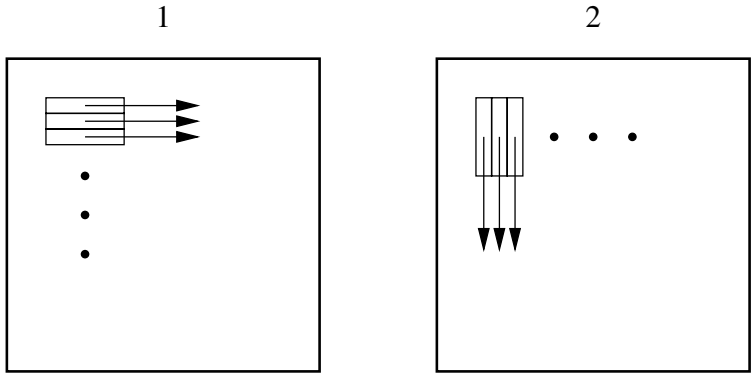

Figure 7.1: Cartesian separable summation of the outer products.

The first canonical correlation ρ_1 and the corresponding (complex) vectors $\mathbf{w}_x$ and $\mathbf{w}_y$ are then calculated. If the set of filters is small, this is done by solving equation 4.26 on page 68. In the case where only two filters are used, this calculation is very simple. If very large sets of filters are used, an analytical calculation of the canonical correlation becomes computationally very expensive. In such a case, the iterative algorithm presented in section 4.7.3 can be used.

The canonical correlation vectors define two new filters:

$$\mathbf{f}_x = \sum_{i=1}^{M} w_{xi}\mathbf{f}_i \quad \text{and} \quad \mathbf{f}_y = \sum_{i=1}^{M} w_{yi}\mathbf{f}_i, \tag{7.3}$$

where $\mathbf{f}_i$ are the basis filters, M is the number of filters in the filter set and w_{xi} and w_{yi} are the components in the first pair of canonical correlation vectors. Due to the properties of canonical correlation, the new filters, $\mathbf{f}_x$ and $\mathbf{f}_y$, have outputs with maximum correlation over $\mathcal{N}$, given the set of basis filters $\mathbf{f}_i$.

7.2 The phase analysis part

The key idea of this part is to search for the disparity that corresponds to a real-valued correlation between the two new filters. This idea is based on the fact that canonical correlations are real valued (see proof B.4.1 on page 165). In other words, find the disparity δ such that

$$\mathrm{Im}\left[\mathrm{Corr}\left(q_y(\xi+\delta)\ ,\ q_x(\xi)\right)\right] = \mathrm{Im}\left[c(\delta)\right] = 0, \tag{7.4}$$

where q_x and q_y are the left and right filter outputs respectively and ξ is the spatial (horizontal) coordinate. There does not seem to exist a well-established definition of correlation for complex variables. The definition used here (see definition A.3) is a generalization of correlation for real-valued variables similar to the definition of covariance for complex variables.

A calculation of the correlation over $\mathcal{N}$ for all δ would be very expensive. A much more efficient solution is to assume that the signal $\mathbf{s}$ can be described by a covariance matrix $\mathbf{C}_{ss}$. Under this assumption, the correlation between the left filter convolved with the signal $\mathbf{s}$ and the right filter convolved with the same signal shifted a certain amount δ can be measured. But convolving a filter with a shifted signal is the same as convolving a shifted filter with the non-shifted signal. Hence, the correlation $c(\delta)$ can be calculated as the correlation between the left filter convolved with $\mathbf{s}$ and a shifted version of the right filter convolved with the same signal $\mathbf{s}$.

Under the assumption that the signal $\mathbf{s}$ has the covariance matrix $\mathbf{C}_{ss}$, the correlation in equation 7.4 can be written as

$$\begin{aligned} c(\delta) &= \frac{E[q_x^* q_y(\delta)]}{\sqrt{E[|q_x|^2]E[|q_y|^2]}} \\ &= \frac{E\left[(\mathbf{s}^*\mathbf{f}_x)^*(\mathbf{s}^*\mathbf{f}_y(\delta))\right]}{\sqrt{E\left[(\mathbf{s}^*\mathbf{f}_x)^*(\mathbf{s}^*\mathbf{f}_x)\right]E\left[(\mathbf{s}^*\mathbf{f}_y)^*(\mathbf{s}^*\mathbf{f}_y)\right]}} \\ &= \frac{E\left[\mathbf{f}_x^*\mathbf{s}\,\mathbf{s}^*\mathbf{f}_y(\delta)\right]}{\sqrt{E\left[\mathbf{f}_x^*\mathbf{s}\mathbf{s}^*\mathbf{f}_x\right]E\left[\mathbf{f}_y^*(\delta)\mathbf{s}\mathbf{s}^*\mathbf{f}_y(\delta)\right]}} \\ &= \frac{\mathbf{f}_x^*\mathbf{C}_{ss}\mathbf{f}_y(\delta)}{\sqrt{\mathbf{f}_x^*\mathbf{C}_{ss}\mathbf{f}_x\,\mathbf{f}_y^*\mathbf{C}_{ss}\mathbf{f}_y}}, \end{aligned} \tag{7.5}$$

where $\mathbf{f}_y(\delta)$ is a shifted version of $\mathbf{f}_y$. Remember that the quadrature filter outputs have zero mean, which is necessary for the first equality. Note the similarity between the last expression and the expression for canonical correlation in equation 4.24 on page 68.

A lot of the computations needed to calculate $c(\delta)$ can be saved since

$$\begin{aligned} \mathbf{f}_x^*\mathbf{C}_{ss}\mathbf{f}_y(\delta) &= \left(\sum_{i=1}^{M} w_{xi}\mathbf{f}_i\right)^* \mathbf{C}_{ss}\left(\sum_{j=1}^{M} w_{yj}\mathbf{f}_j(\delta)\right) \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} w_{xi}^* w_{yj}\mathbf{f}_i^*\mathbf{C}_{ss}\mathbf{f}_j(\delta) = \sum_{ij} v_{ij}g_{ij}(\delta), \end{aligned} \tag{7.6}$$

where

$$g_{ij}(\delta) = \mathbf{f}_i^*\mathbf{C}_{ss}\mathbf{f}_j(\delta). \tag{7.7}$$

The function $g_{ij}(\delta)$ does not depend on the result from the CCA and can therefore be calculated in advance for different disparities δ and stored in a table. The denominator in equation 7.5 can be treated in the same way but does not depend on δ:

$$\mathbf{f}_x^* \mathbf{C}ss\mathbf{f}_x = \sum_{ij} v_{ij}^x g_{ij}(0) \quad \text{and} \quad \mathbf{f}_y^* \mathbf{C}ss\mathbf{f}_y = \sum_{ij} v_{ij}^y g_{ij}(0), \tag{7.8}$$

where $v_{ij}^x = w_{xi}^* w_{xj}$ and $v_{ij}^y = w_{yi}^* w_{yj}$. Note that the filter vectors $\mathbf{f}$ must be padded with zeros at both ends to enable the scalar product between a filter and a shifted filter δ. (The zeros do not, of course, affect the result of equation 7.6.) In the case of two basis filters, the table contains four rows and eight constants.

Hence, for a given disparity a (complex) correlation $c(\delta)$ can be computed as a normalized weighted sum:

$$c(\delta) = \frac{\sum_{ij} v_{ij} g_{ij}(\delta)}{\sqrt{\sum_{ij} v_{ij}^x g_{ij}(0) \ \sum_{ij} v_{ij}^y g_{ij}(0)}}. \tag{7.9}$$

The aim is to find the δ for which the correlation $c(\delta)$ is real valued. This is done by finding the zero crossings of the phase of the correlation. A very coarse quantization of δ can be used in the table since the phase is, in general, rather linear near the zero crossing (as opposed to the imaginary part which in general is not linear). Hence, first a coarse estimate of the zero crossing is obtained. Then the derivative of the phase at the zero crossing is measured, using two neighbouring samples. Finally, the error in the coarse estimate is compensated for by using the actual phase value and the phase derivative at the estimated position:

$$\delta = \delta_c - \frac{\varphi(\delta_c)}{\partial\varphi/\partial\delta}, \tag{7.10}$$

where δ_c is the coarse estimate of the zero crossing and $\varphi(\delta_c)$ is the complex phase of $c(\delta_c)$ (see figure 7.2 on the next page).

7.2.1 The signal model

If the signal model is uncorrelated white noise, $\mathbf{C}_{ss}$ is the identity matrix and the calculations of the values in the table reduce to a simple scalar product: $g_{ij}(\delta) = \mathbf{f}_i^* \mathbf{f}_j(\delta)$. There is no computational reason to choose white noise as signal model if there is a better model, since the table is calculated only once. But it can still be interesting to compare the correlation for white noise with the correlation for another signal model in order to get a feeling for the algorithm's sensitivity with respect to the signal model. In other words, how does the choice of model affect the position of the zero phase of $c(\delta)$?

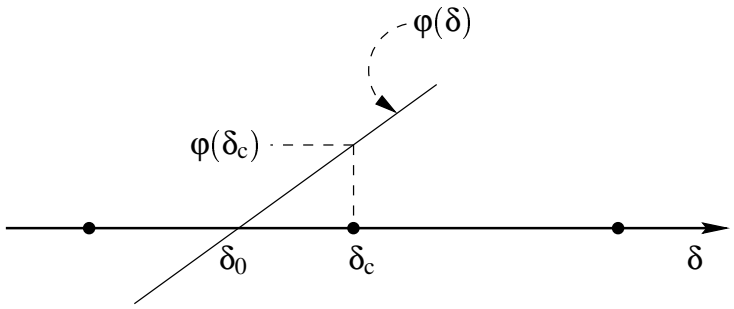

Figure 7.2: The estimation of the coordinate δ_0 of the phase zero crossing using the coarse estimate δ_c of the zero crossing, the phase value $\varphi(\delta_c)$ and the derivative at the coarse estimate. The black dots illustrate the sampling points of the phase given by the table $g_{ij}(\delta)$.

First of all, it should be noted that the denominator in equation 7.5 is real valued and, hence, does not affect the complex phase of $c(\delta)$. So, only the numerator

$$c'(\delta) = \mathbf{f}_x^* \mathbf{C}_{ss} \mathbf{f}_y(\delta) \tag{7.11}$$

has to be considered. In general, $\mathbf{C}_{ss}$ is a Toeplitz matrix (i.e. $C_{ij} = C(i-j)$) with the columns (and rows) containing shifted versions of the (non-normalized) autocorrelation function $\mathbf{c}_s$ of the signal $\mathbf{s}$. This means that $\tilde{\mathbf{f}}_x^* = \mathbf{f}_x^* \mathbf{C}_{ss}$ can be seen as a convolution of $\mathbf{f}_x^*$ with the autocorrelation function $\mathbf{c}_s$. But

$$c'(\delta) = \tilde{\mathbf{f}}_x^* \mathbf{f}_y(\delta) \tag{7.12}$$

can be seen as a convolution between $\tilde{\mathbf{f}}_x^*$ and $\mathbf{f}_y^-$, where $\mathbf{f}_y^-$ is $\mathbf{f}_y$ reversed, since δ only causes a shift of $\mathbf{f}_y$. This means that $c'(\delta)$ can be written as

$$c'(\delta) = (\mathbf{f}_x^* * \mathbf{c}_s) * \mathbf{f}_y^-, \tag{7.13}$$

where $(*)$ denotes convolution. Since the order of convolutions does not matter (convolution is commutative and associative), $c'(\delta)$ can be written as

$$c'(\delta) = (\mathbf{f}_x^* * \mathbf{f}_y^-) * \mathbf{c}_s = (\mathbf{f}_x^* \mathbf{f}_y(\delta)) * \mathbf{c}_s, \tag{7.14}$$

i.e. the convolution between $\mathbf{f}_x^*$ and $\mathbf{f}_y^-$ can be calculated first. This function can then be convolved with the autocorrelation function to get the correct c' for the model.

This means that the difference between the correlation $c(\delta)$ calculated for white noise (i.e. $\mathbf{C}_{ss}$ is the identity matrix) and the correlation calculated by using

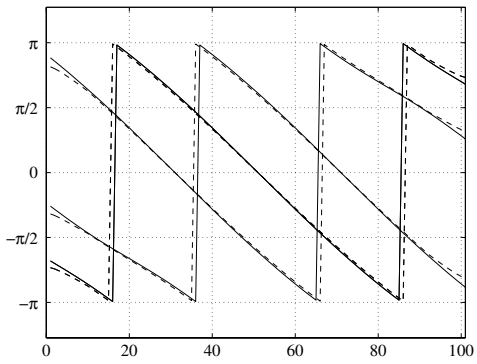

Figure 7.3: The phase of the four rows of the table containing $g_{ij}(\delta)$ without convolution (solid line) and with convolution with the autocorrelation function $1 - |\xi|$.

another signal model is given by the convolution of $c(\delta)$ with the autocorrelation function of the signal model (and an amplitude scaling that does not affect the phase). Hence, if the phase around the zero crossing is anti-symmetric (e.g. linear) in an interval that is large compared to the autocorrelation function of the signal model, the result will be very similar to that obtained for a white noise model. Another lax interpretation of the reasoning above is that as long as the phase does not have bad behaviour around zero, the choice of signal model is not critical.

As an example, the phases of the four rows in a table $g_{ij}(\delta)$ are plotted in figure 7.3 with and without convolution with the autocorrelation function $1 - |\xi|$. (Note that two of the rows, g_{11} and g_{22}, are equal, which means that only three curves for each case are visible.) This autocorrelation function is usually assumed for natural images.

7.2.2 Multiple disparities

If more than one zero crossing are detected, the magnitudes of the correlations can be used to select a solution. Since the CCA searches for maximum correlation, the zero crossing with maximum correlation $c(\delta)$ is most likely to be the best estimate. If two zero crossings have approximately equal magnitude (and the canonical correlation ρ is high), both disparity estimates can be considered to be correct within the neighbourhood, which indicates either a depth discontinuity or that there really exist two disparities.

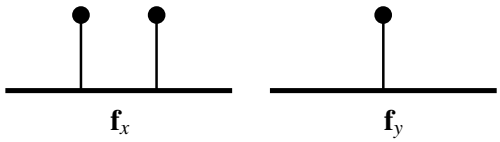

Figure 7.4: A simple example of a pair of filters that have two correlation peaks.

The latter is the case for semi-transparent images, i.e. images that are sums of images with different depths. Such images are typical of many medical applications such as x-ray images. An every-day example of this kind of image is obtained by looking through a window with reflection. (The effect on the intensity of a light- or X-ray when passing two objects is in fact multiplicative, but a logarithmic transfer function is usually applied when generating X-ray images which makes the images additive.)

Note that both disparity estimates are represented by the same canonical correlation solution. This means that the CCA must generate filters that have correlation peaks for *two* different disparities. To see how this can be done, consider the simple filter pair illustrated in figure 7.4. The autocorrelation function (or convolution) between these two filters is identical to the left filter, which consists of two impulses. The example is much simplified, but illustrates the possibility of having two filters with two correlation peaks. If the CCA was used directly on the pixel data instead of on the quadrature filter outputs, such a filter pair *could* develop. In the present method, the image data are represented by using other basis functions (the quadrature filters of the basis filter set) but it is still possible to construct filters with two correlation peaks.

7.2.3 Images with different scales

If the images are differently scaled, the CCA will try to create filters scaled correspondingly. In order to improve the disparity estimates in these cases, the table can be extended with scaled versions of the basis filters:

$$g_{ij}(\sigma,\delta) = \mathbf{f}_i^* \mathbf{C}_{ss} \mathbf{f}_j(\sigma,\delta), \tag{7.15}$$

where $\mathbf{f}_j(\sigma,\delta)$ is a scaled and shifted version of $\mathbf{f}_j$. The motive for this is that a scaled signal convolved with a certain filter gives the same result as the non-scaled signal convolved with a reciprocally scaled filter. The CCA step is not affected by this and the phase analysis is performed as described above on each scale. The correct scale is indicated by having the maximum real-valued correlation.

The resolution in scale can be very coarse. In the experiments presented in the following section, the filters have been scaled between +/- one octave in steps of a quarter of an octave, which seems to be a quite sufficient resolution.

It should be noted that the disparity estimates measured in pixels will differ in the two images since one of the images has a scaled filter as reference. But given the filter scales, the interpretations in terms of depth are of course the same in both images.

7.3 Experiments

In this section, some experiments are presented to illustrate the performance of the stereo algorithm. Some results on artificial data are shown. Finally, the algorithm is applied to two real stereo image pairs, both common test objects for stereo images.

In all experiments presented here, a basis filter set consisting of two one-dimensional horizontally oriented quadrature filters, both with a centre frequency of $\pi/4$ and a bandwidth of two octaves has been used. The filters have 15 coefficients in the spatial domain and are shifted two pixels relative to each other. The frequency function is approximately a squared cosine on a log scale:

$$F(u) \approx \cos^2\left(k \ln(u/u_0)\right), \tag{7.16}$$

where $k = \pi/(2\ln(2))$ and $u_0 = \pi/4$. The actual filter functions are illustrated in figure 7.5.

For the experiments on the artificial data, the neighbourhood for the CCA has been chosen to fit the problem reasonably well. This means that the neighbourhood is longer in the direction of constant disparity than in the direction where the disparity changes. In the real images, a square neighbourhood has been used. How the choice of neighbourhood can be made adaptive is discussed in section 7.4.

7.3.1 Discontinuities

The first experiment illustrates the algorithm's ability to handle depth discontinuities. The test image is made of white noise shifted so that the disparity varies between $+/-d$ along the horizontal axis and d varies as a ramp from -5 pixels to $+5$ pixels along the vertical axis in order to get discontinuities between $+/-10$ pixels. A neighbourhood $\mathcal{N}$ of 13×7 pixels (horizontal $\times$ vertical) was used for the CCA. Figure 7.6 shows the estimated disparity for this test image. Disparity estimates with corresponding canonical correlations less than 0.7 have been removed. In figure 7.7, two lines of the disparity estimate are shown. To

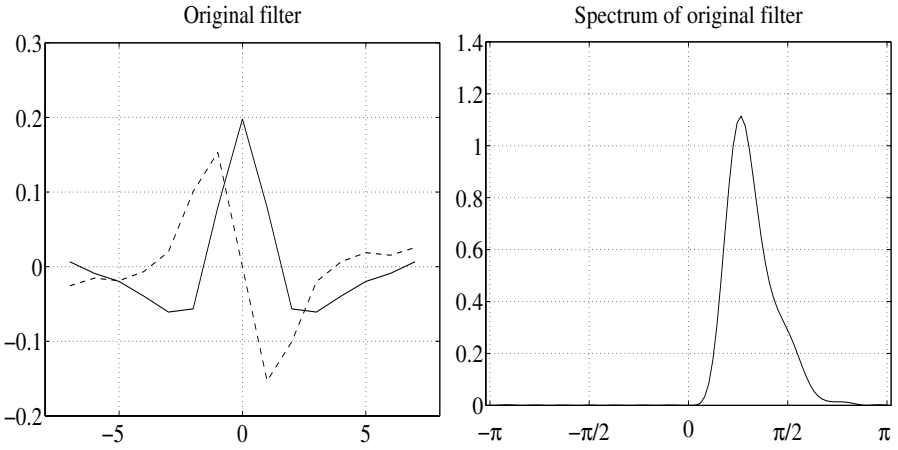

Figure 7.5: The filter in the basis filter set.

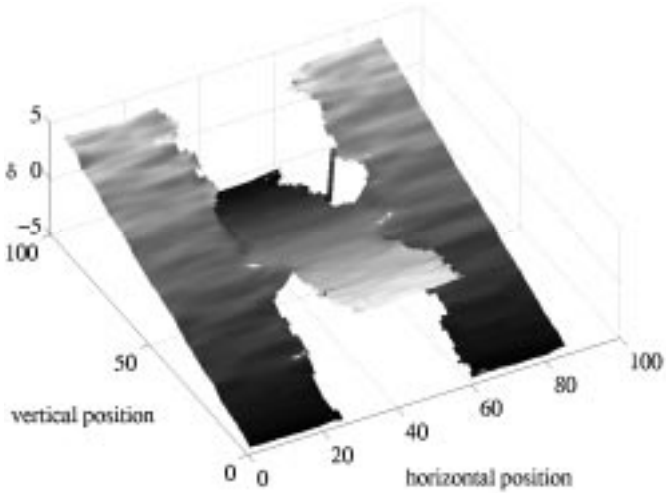

Figure 7.6: Disparity estimate for different depth discontinuities.

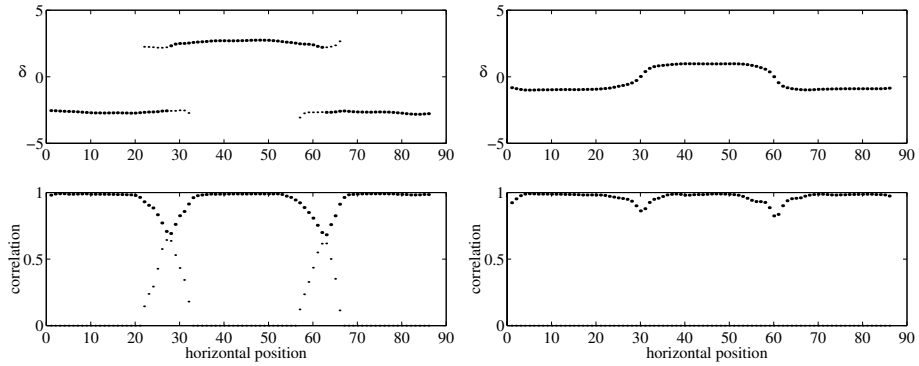

Figure 7.7: **Top:** Line 20 (left) and line 38 (right) from the disparity estimates in figure 7.6 on the facing page. The small dots indicate the disparity estimates with the second strongest correlations. **Bottom:** The corresponding correlations.

the left, line 20 with a disparity of $+/-2.5$ pixels is shown and to the right, line 38 with a disparity of 1 pixel is shown. The figures at the top show the most likely (large dots) and second most likely (small dots) disparity estimates along these lines. The bottom figures show the corresponding canonical correlations at the zero crossings. Figures 7.6 and 7.7 show that for small discontinuities, the algorithm interpolates the estimates while for large discontinuities, there are two overlapping estimates.

An interpolation or fusion for small disparity differences is also performed by the human visual system. The depth interval for which all points are fused into a single image is called Panum's area (see for example Coren and Ward, 1989).

7.3.2 Scaling

The second experiment shows that the algorithm can estimate disparities between images that are differently scaled. The test image here is white noise warped to form a ramp along the horizontal axis. The warping is made so that the right image is scaled to 50% of the original size which means that there is a scale difference of one octave. For a human, this corresponds to looking at a point on a surface with its normal rotated 67° from the observer at a distance of 20 centimetres. In this experiment, a neighbourhood $\mathcal{N}$ of 3×31 pixels was used. In figure 7.8 on the next page the results are shown for the basic algorithm without the scaling parameter (left) and for the extended algorithm that searches for the

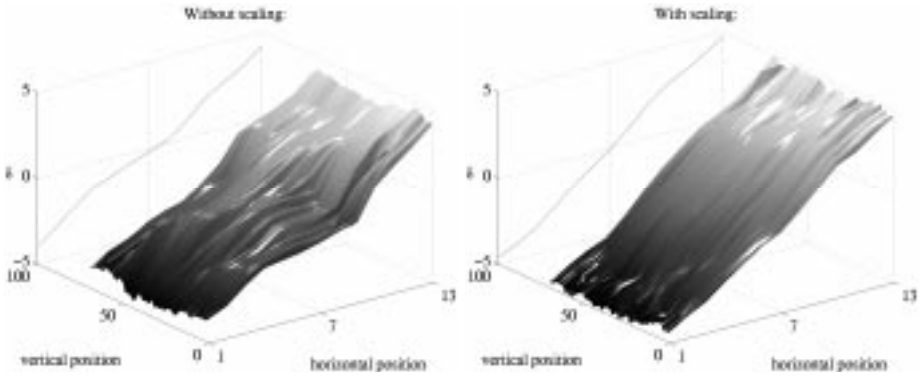

Figure 7.8: Disparity estimate for a scale difference of one octave between the images without scale analysis (left) and with scale analysis (right).

optimal scaling (right). The lines at the back of the graphs show the mean value.

The filters created by the CCA are illustrated in figure 7.9. The left-hand plots show the filters in the spatial domain and the right-hand plots show them in the frequency domain.

7.3.3 Semi-transparent images

This experiment illustrates the algorithm's capability of multiple disparity estimates on semi-transparent images. The test images in this experiment were generated as a sum of two images with white uncorrelated noise. The images were tilted in opposite directions around the horizontal axis. The disparity range was $+/-5$ pixels. Figure 7.10 illustrates the test scene. The stereo pair is shown in figure 7.11 on page 134. Here, the averaging or fusion performed by the human visual system for small disparities can be seen in the middle of the image. A neighbourhood $\mathcal{N}$ of 31×3 pixels was used for the CCA. The result is shown in figure 7.12. In figure 7.13 on page 136, the estimates are projected along the horizontal axis. The results show that the disparities of both the planes are approximately estimated. In the middle, where the disparity difference is small, the result is an average between the two disparities in accordance with the results illustrated in figure 7.7 on the preceding page.

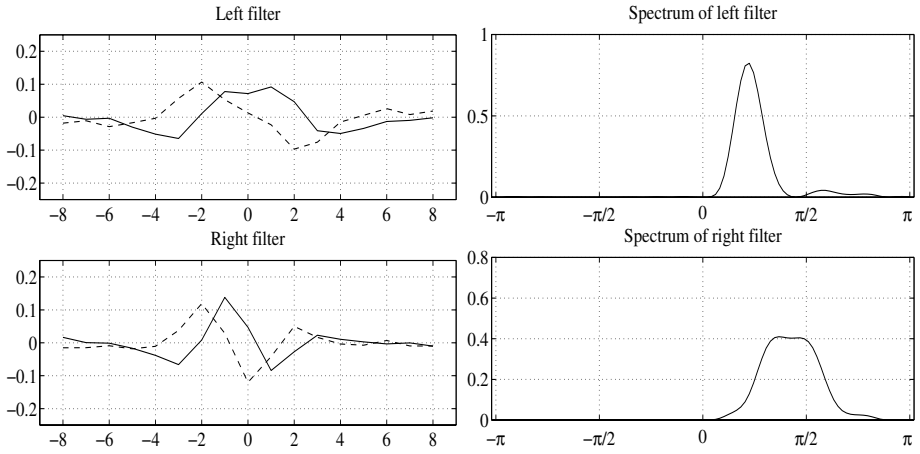

Figure 7.9: The filter created by CCA. Solid lines show the real parts and dashed lines show the imaginary parts.

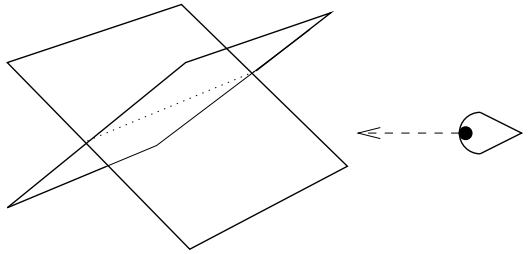

Figure 7.10: The test image scene for semi-transparent images.

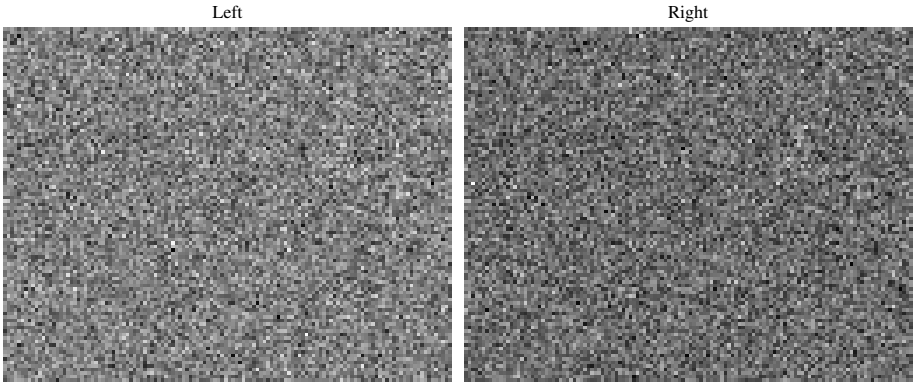

Figure 7.11: The stereo image pair for the semi-transparent images.

7.3.4 An artificial scene

This experiment tries to simulate a slightly more realistic case where both the discontinuity problem and the scale problem are present. The scene can be thought of as a pole or a tree in front of a wall. Figure 7.14 on page 136 illustrates the scene from above. The distance from the wall to the centre of the tree was 2, the radius of the tree was 1, the distance from the wall to the cameras was 5 and the distance between the cameras was 0.4 length units. A texture of white noise was applied on the wall and on the tree and a stereo pair of images was generated. Each image had the size 200×31 pixels. The generated stereo images are shown in figure 7.15. The disparity was only calculated for one line. Also in this case, a neighbourhood $\mathcal{N}$ of 3×31 pixels was used for the CCA. The algorithm was run 100 times on different noise images. The result is illustrated in figure 7.16 on page 137. Close to the edges of the tree, the images are differently scaled. In figure 7.17 on page 138, the average scale difference used by the algorithm is plotted. The scaling can be done in nine steps between $+/-$ one octave and in the figure, the average scaling (in octaves) is plotted. The plot illustrates how the algorithm scales the images relative to each other in one way near the left edge of the tree and in the opposite way at the other edge as expected. There is no scale difference on the background and in the middle of the tree.

7.3.5 Real images

The two final experiments illustrate how the algorithm works on real stereo image pairs. In both experiments, a neighbourhood $\mathcal{N}$ of 7×7 pixels was used.

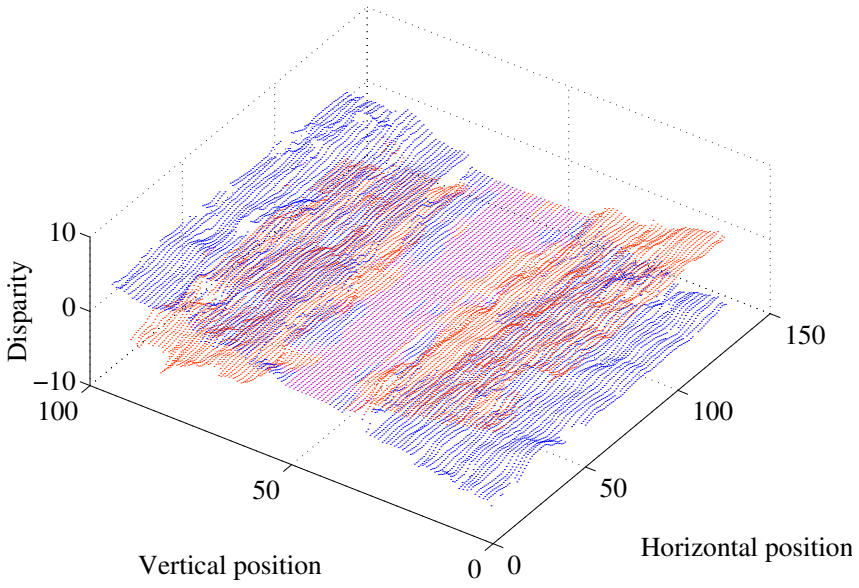

Figure 7.12: The result for the semi-transparent images. The disparity estimates are coloured to simplify the visualization.

The first stereo pair is two air photographs of Pentagon (see figure 7.18 on page 139, upper row). The result is shown in the bottom row of the same figure. To the left, the disparity estimates are shown. White means high disparity and black means low disparity. The lower-right image shows a certainty image calculated from the canonical correlation in each neighbourhood. The certainty used here is the logarithm of the SNR according to equation 4.38 on page 71 plus an offset in order to make it positive.

The second stereo pair is two images from a well-known image sequence, the Sarnoff tree sequence. This stereo pair is shown in top of figure 7.19 on page 140 and the result and the certainty image are shown at the bottom of the same figure.

The results are also illustrated in colour in figures 7.20 on page 141 and 7.21 on page 142. The images at the top are generated so that the colour represents disparity and the intensity represents the original (left) image. The images at the bottom are 3-dimensional surface plots with height and colour representing the disparity estimates. Note that the walls in the pentagon result are depth discontinuities and not just steep slopes.

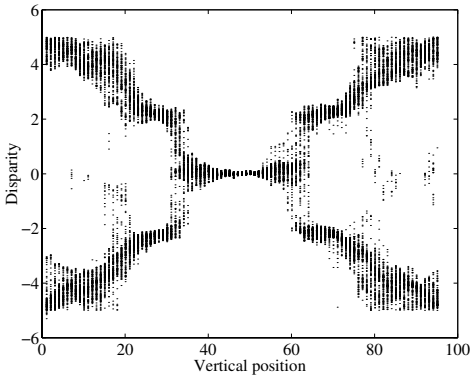

Figure 7.13: A projection along the horizontal axis of the estimates in figure 7.12 on the preceding page.

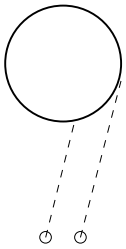

Figure 7.14: The artificial tree scene from above.

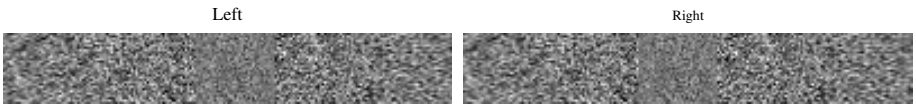

Figure 7.15: The stereo pair generated from the artificial tree scene.

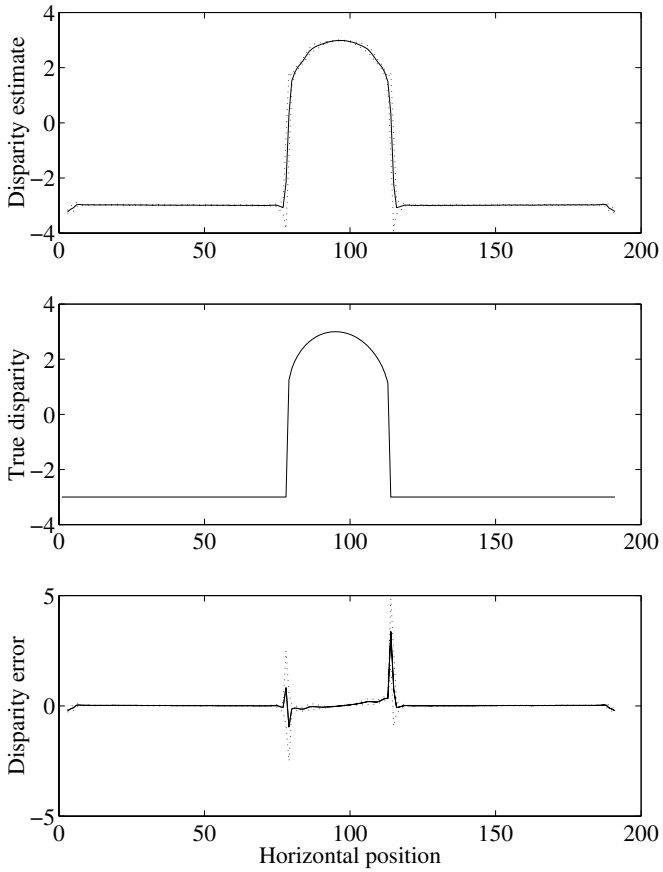

Figure 7.16: The result for the artificial test scene. The top graph shows the average disparity estimate. The dotted lines show the standard deviation. The middle graph shows the true disparity. The bottom graph shows the mean disparity error and the standard deviation of the disparity error (dotted line).

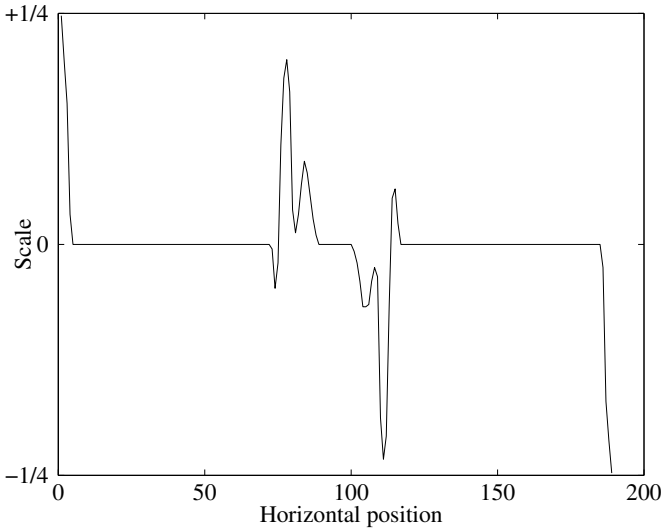

Figure 7.17: The average scaling performed by the algorithm. 0 means no scaling, $+1/4$ means that the left image is scaled $+1/4$ of an octave, i.e. made smaller compared to the right image.

7.4 Discussion

The stereo algorithm described in this chapter is rather different from most image processing algorithms. Common procedures are first to optimize a set of filters and then to use these filters to analyse the image or performing statistical analysis directly on the pixel data. This algorithm, however, first adapts the filters to a local region in the image and then analyses the *adapted filters*.

In all the experiments presented in the preceding section, a filter set of two filters differing only by a shift was used. A larger filter set with filters in different scales would be able to handle larger disparity ranges. If such a set of filters was used, the algorithm would simply select the proper scale to work on, i.e. the scale that has the highest correlation. In general, a larger filter set offers the shapes of the adapted filters more freedom. Hence, a larger filter set should make it easier to handle multiple disparities, depth discontinuities and scale differences if the filter set is chosen properly. A larger filter set covering a wider range of frequencies would also reduce the risk of the signal in a region giving very weak filter output because the filter does not fit the signal. With a larger filter set, the CCA would only use the filters that have a high SNR for the current signal.

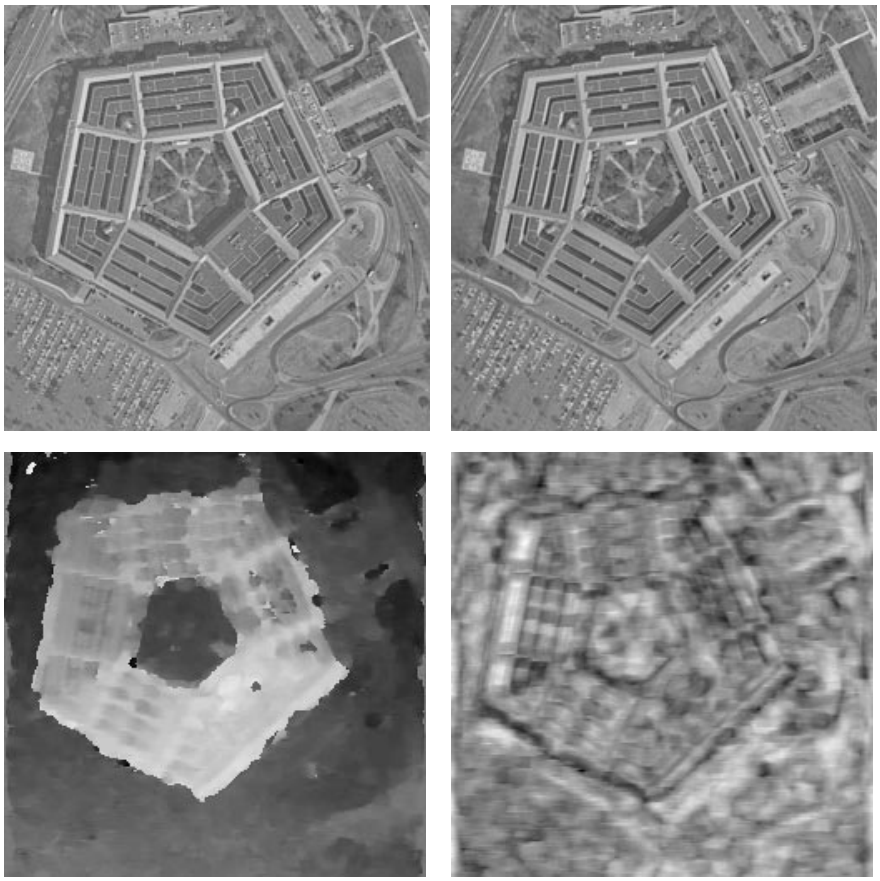

Figure 7.18: **Upper row:** Stereo pair of Pentagon. **Lower left:** Resulting disparity estimates. **Lower right:** Certainty image of the estimates.

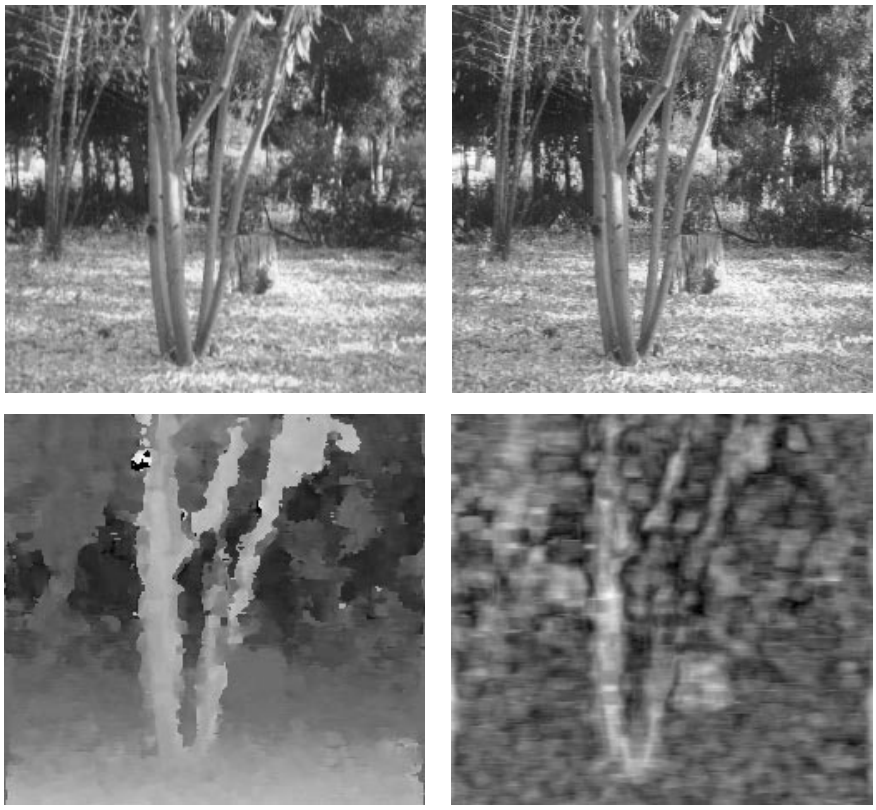

Figure 7.19: **Upper row:** Stereo pair of the tree scene. **Lower left:** Resulting disparity estimates. **Lower right:** Certainty image of the estimates.

The filter set can be seen as the basis functions used for representing the signal. The simplest choice of basis functions is the pixels themselves. The canonical correlation vector will then define the filters directly in the pixel base. A disadvantage with such an approach is that the analysis of the filter becomes expensive. The canonical correlation vectors in the experiments presented here were two-dimensional since there were two basis filters. If the pixel basis is used, the dimensionality is equal to the size of the filters that the algorithm is to construct. This means, for example, that if the algorithm should be able to use 1×15 filter kernels, the canonical correlation vectors become 15-dimensional. In other words, the pixel basis is not a good choice of signal representation in this problem (see the discussion in chapter 3). Since we know a better representation for this

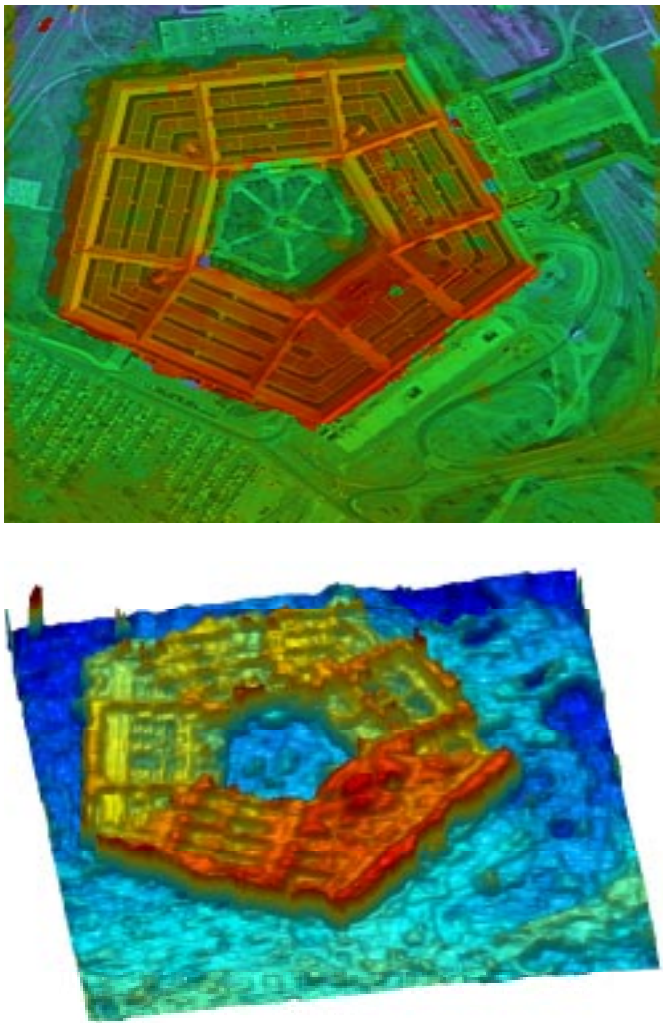

Figure 7.20: Result for the pentagon images in colour. The upper image displays the disparity estimate as the colour overlaid on the original intensity image.

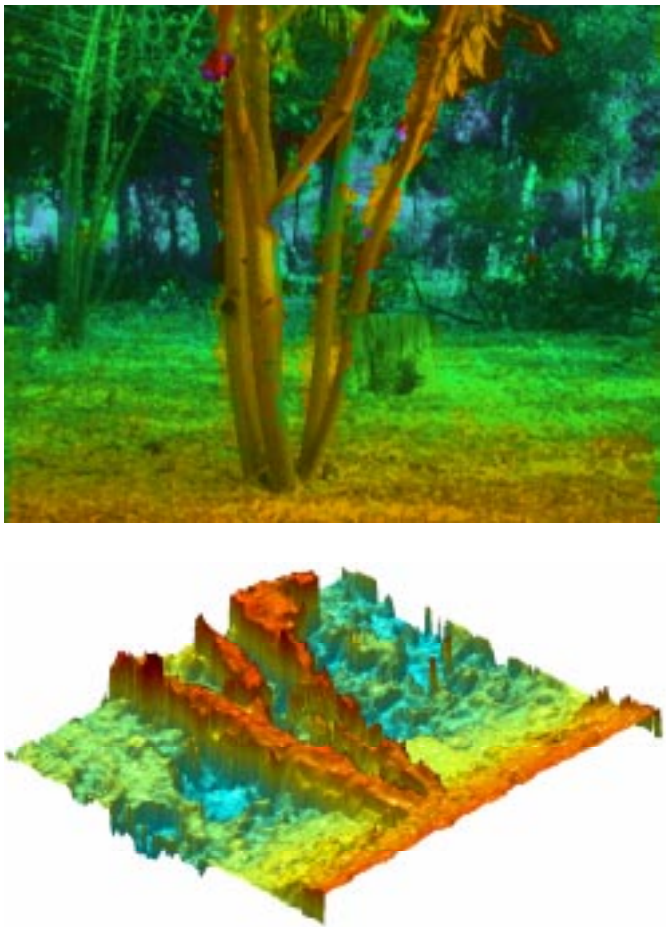

Figure 7.21: Result for the tree images in colour. The upper image displays the disparity estimate as the colour overlaid on the original intensity image.

problem (i.e. quadrature filters), it would be unwise not to use it.

The choice of neighbourhood for the CCA is of course important for the result. If there is a priori knowledge of the shape of the regions that have relatively constant depths, the neighbourhood should, of course, be chosen accordingly. This means that if the disparity is known to be relatively constant along the vertical axis, for example, the shape of the neighbourhood should be elongated vertically, as in the experiments on artificial data in the previous section. It is, however, possible to let the algorithm select a suitable neighbourhood shape automatically. This may be done in two ways.

One way is to measure the canonical correlation for a few different neighbourhood shapes. These shapes could be, for example, one horizontally elongated, one vertically elongated and one square. The algorithm should then use the result from the neighbourhood that gave the highest canonical correlation to estimate the disparity.

Another way to automatically select neighbourhood shape is to begin with relatively small square-shaped neighbourhoods to get a coarse disparity estimate. Then the disparity estimates are segmented. A second run of the algorithm can then use neighbourhood shapes selected according to the shape of the segmented regions. It should be noted that the neighbourhoods can be arbitrarily shaped and even non-connected. The only advantage with a rectangular neighbourhood is that it is computationally efficient when calculating the covariance matrices for the CCA. But if this is utilized in the first run and the covariance matrices are stored, they can simply be added when forming the new larger neighbourhoods in the second run. On the tree image for example, this approach would give vertically elongated neighbourhoods on the tree and horizontally elongated neighbourhoods on the ground.

Chapter 8

Epilogue

In this final chapter, the thesis is summed up and discussed. To conclude, some ideas for future research are presented.

8.1 Summary and discussion

The thesis started with a discussion of learning systems. Three different principles of learning were described. Supervised learning can be seen as function approximation. The need for a training set that has an associated set of desired output restricts its use to tasks where such training data can be obtained. Reinforcement learning, on the other hand, is more general than supervised learning and we believe that it is an important general learning principle in complex systems. Its relation to learning among animals and to evolution can support this position. Unsupervised learning is a way of finding a data dependent representation of the signals that is useful according to some criterion. We do not believe that unsupervised learning is the highest general learning principle, since the performance measure these methods are trying to maximize is related only to the internal data representation and has nothing to do with the actual performance of the system in terms of actions. Unsupervised learning can, however, be an important component which helps the system find a good signal representation. It should again be pointed out that the difference between the three learning principles is not so clear as it might seem at first, as discussed in section 2.6.

For unsupervised learning, we believe that methods based on maximizing information are important. If nothing else is known about the optimum choice of representation, it is probably wise to preserve as much information (rather than variance, for example) as possible. It is, however, not only the amount of information that is important. The information to be represented must be relevant for the task. In other words, it must be related to information about possibly success-

ful responses; otherwise it is not useful. This makes methods based on maximum mutual information good candidates.

The signal representation needs a model for the represented information. A complex global model is not a realistic choice for large systems that have high-dimensional input and output signals. The number of parameters to estimate would be far too large and the structural credit assignment problem would be unsolvable (see section 2.7.2). We believe that local low-dimensional linear models should be used. One reason for this is that only a small fraction of a high-dimensional signal space will ever be visited by the signal. Furthermore, this signal is (at least) piecewise continuous because of the dynamic of the real world, which means that the signal can be represented arbitrarily well with local linear models. How to distribute these models is only briefly mentioned in this thesis (section 3.5). The interested reader is referred to the PhD thesis by Landelius (1997) for a detailed investigation of this subject.

The choice of local linear models can be made according to different criteria depending on the task. If maximum mutual information is the criterion, canonical correlation analysis is a proper method for finding local linear models. CCA is related to PCA, PLS and MLR which maximize other criteria, statistical or mean square error (see chapter 4). An iterative algorithm for these four methods was presented. The algorithm is more general and actually finds the solutions to the generalized eigenproblem. An important feature of the proposed algorithm is that it finds the solutions successively, beginning with the most significant one. This enables low-rank versions of the solutions of the four methods which is necessary if the signal dimensionality is high. Another nice feature is that the algorithm gives the eigenvector *and* the corresponding eigenvalues and not only the normalized eigenvectors as is the case with many other iterative methods.

It was shown that CCA can be used for learning feature descriptors for computer vision. The proposed method allows the user to define what is equal in two signals by giving the system examples. If other features are varied in an uncorrelated way, the feature descriptors become invariant to these features. An experiment showed that the system learned quadrature filters when it was trained to represent orientation invariant to phase. When quadrature filter outputs are used as input to the system, it learns to combine them in a way that is less sensitive to noise than vector averaging without losing spatial resolution. For a 5×5 neighbourhood, the angular error of the orientation estimate was reduced by 4 dB which is quite a substantial improvement. This method will most likely replace vector averaging in many applications where there is a conflict between the need for noise reduction and spatial resolution.

Another application of CCA in computer vision is stereo. A novel stereo algorithm was presented in chapter 7. The algorithm is a bit unusual since first it

adapts filters to an image neighbourhood an then it analyses *the resulting filters*. A more common approach in computer vision is first to optimize filters and then to use these filters to analyse the *image*. Some interesting features of the proposed algorithm are that it can handle depth discontinuities, multiple depths in semi-transparent images and image pairs that are differently scaled. Although only one basis filter set with two shifted identical filters has been tested, the results look very promising both on real an artificial images. We believe that the proposed method can be useful also in motion estimation, in particular on x-ray images where there are multiple motions in semi-transparent images.

8.2 Future research

There is a number of ideas left open for future research. One interesting question is how to combine reinforcement learning and mutual information based unsupervised learning.

A rather ad hoc modification of the canonical correlation algorithm that can handle very high-dimensional signals was presented. Other methods for handling adaptive update factors should be investigated. Preliminary investigations indicate that the *RPROP* algorithm (Riedmiller and Braum, 1993) can be modified to fit our algorithm. Since the purpose of a gradient based algorithm is to handle very high-dimensional signals, it is important that the algorithm is optimized to handle such cases.

The theory for the gradient search method presented in chapter 4 was developed for real valued signals. In chapters 6 and 7, however, we have seen that canonical correlation is useful also when analysing complex-valued signals. Hence, an extension of the theory in chapter 4 to include complex-valued signals is desirable.

One of the most interesting issues for future research based on this work is to investigate how canonical correlation can be used in multidimensional signal processing. The experiments in chapter 6 show that phase invariant orientation filters can be learned by using this method. The use of this algorithm for detecting other, higher-level, features should be investigated. Examples of such features are line crossings, corners and even texture. Consider a pair of local neighbourhoods with a given spatial relation as illustrated in figure 8.1. The spatial relation is defined by a displacement vector $\mathbf{r}$. If data are collected from such neighbourhood pairs in a larger region of the image, a CCA would give the linear combination of one neighbourhood that is the most predictable and at the same time the linear combination of the other neighbourhood that is the best predictor. For each displacement, this would give a measure of the best linear relation between the image patches and a description of that relation. This can be performed directly

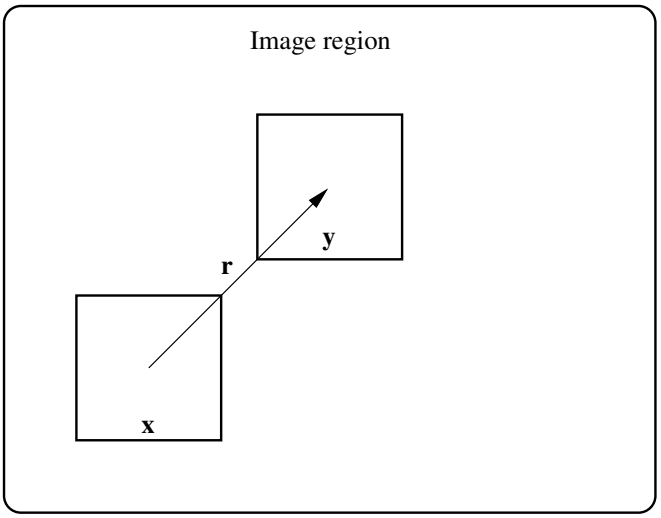

Figure 8.1: Illustration of how CCA can be used for generating a texture descriptor by analysing the linear relation between two neighbourhoods **x** and *by* with a spatial relationship defined by the displacement vector **r**.

on the pixel data or on a filtered image. Consider, for example, a sine wave pattern without noise. The canonical correlation for such an image would be one for all displacements between the neighbourhoods. This is logical, since the pattern is totally predictable. An ordinary correlation analysis, however, would give zero correlations where the phase of the patterns differs 90°. A matrix containing the largest canonical correlations for different neighbourhood displacements defines the displacement vectors for which the patterns are linearly predictable. Instead of the matrix, a tensor that contains the canonical correlation vectors can be used. Such a tensor would be a descriptor of the texture. The use of such descriptors in texture analysis should be investigated.

The generalization of the canonical correlation method aiming to find maximum mutual information as illustrated in figure 6.2 on page 109 should be investigated. The non-linear functions f_x and f_y can be implemented as neural networks with, for example, sigmoid or radial-basis functions. The neural networks are trained, for example using back-propagation, in order to maximize the canonical correlation ρ.

The method in chapter 6 is, of course, not limited to image data. Another interesting application is speech recognition, where it is important to be invariant with respect to how the words are pronounced.

Another very interesting issue is the extension of the stereo algorithm in chap-

ter 7 in order to estimate both *vertical* and horizontal shifts, i.e. two-dimensional translations of the image. If the neighbourhoods are taken from different frames in a temporal image sequence, the extended algorithm could be used for motion estimation. The capability of handling multiple estimates in semi-transparent images would make this method interesting in medical applications. The problem of estimating multiple motions exists for example in x-ray image sequences, where different parts of the body move in different ways. The capability of handling scaling between the images would make it possible to handle motions more complex than pure translations, for example 3-dimensional rotations and deformations.

Appendix A

Definitions

In this appendix, some useful non-standard functions are defined. "$\triangleq$" means "equal by definition".

A.1 The vec function

Consider an $m \times n$ matrix M:

$$\mathbf{M} = [\mathbf{m}_1 \ \mathbf{m}_2 \ \ldots \ \mathbf{m}_n] \tag{A.1}$$

where the columns $\mathbf{m}_i$ are m-dimensional vectors. Then

$$\mathbf{v} = \mathrm{vec}(\mathbf{M}) \triangleq \begin{pmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{pmatrix} \tag{A.2}$$

A.2 The mtx function

Consider an mn-dimensional vector $\mathbf{v}$. Then

$$\mathbf{M} = \mathrm{mtx}(\mathbf{v}, m, n) \triangleq [\mathbf{m}_1 \ \mathbf{m}_2 \ \ldots \ \mathbf{m}_n] \tag{A.3}$$

where the columns $\mathbf{m}_i$ are m-dimensional vectors.

A.3 Correlation for complex variables

Consider $x, y \in \mathbb{C}^1$ with mean $\bar{x}$ and $\bar{y}$ respectively. The correlation between x and y is defined as

$$\mathrm{Corr}(x,y) = \frac{E[(x-\bar{x})(y-\bar{y})^*]}{\sqrt{E[|x-\bar{x}|^2]\,E[|y-\bar{y}|^2]}} \tag{A.4}$$

Appendix B

Proofs

This appendix contains all the proofs referred to in the text.

B.1 Proofs for chapter 2

B.1.1 The differential entropy of a multidimensional Gaussian variable

$$h(\mathbf{z}) = \frac{1}{2}\log\left((2\pi e)^N|\mathbf{C}|\right), \tag{2.41}$$

where $|\mathbf{C}|$ is the determinant of the covariance matrix of $\mathbf{z}$ and N is the dimensionality of $\mathbf{z}$.

Proof: The Gaussian distribution for an N-dimensional variable $\mathbf{z}$ is

$$p(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^N|\mathbf{C}|}}\, e^{-\frac{1}{2}\mathbf{z}^T\mathbf{C}^{-1}\mathbf{z}} \tag{B.1}$$

The definition of differential entropy (equation 2.38 on page 29) then gives

$$\begin{aligned} h(\mathbf{z}) &= -\int_{\mathbb{R}^N} p(\mathbf{z})\log p(\mathbf{z})\, d\mathbf{z} \\ &= \int_{\mathbb{R}^N} p(\mathbf{z})\left(\log\left(\sqrt{(2\pi)^N|\mathbf{C}|}\right) + \frac{1}{2}\mathbf{z}^T\mathbf{C}^{-1}\mathbf{z}\right) d\mathbf{z} \\ &= \log\sqrt{(2\pi)^N|\mathbf{C}|} \;+\; \frac{N}{2} = \frac{1}{2}\log\left((2\pi e)^N|\mathbf{C}|\right). \end{aligned} \tag{B.2}$$

Here, we have used the fact that

$$\int_{\mathbb{R}^N} p(\mathbf{z})\mathbf{z}\mathbf{C}^{-1}\mathbf{z}\, d\mathbf{z} = E[\mathbf{z}^T\mathbf{C}^{-1}\mathbf{z}] = E[\mathrm{tr}(\mathbf{z}\mathbf{z}^T\mathbf{C}^{-1})] = N \tag{B.3}$$

B.2 Proofs for chapter 3

B.2.1 The constant norm of the channel set

$\sum_{\forall k} |c_k|^2 = constant$ where

$$c_k = \begin{cases} \cos^2\left(\frac{\pi}{3}(x-k)\right) & \text{if} \quad |x-k| < \frac{3}{2} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

(page 41).

Proof: Consider the interval $-\frac{\pi}{6} < x \leq \frac{\pi}{6}$. On this interval, all channels are zero except for $k = -1, 0, 1$. Hence, it is sufficient to sum over these three channels.

$$\begin{aligned}
\sum |c_k|^2 &= \cos^4\left[\frac{\pi}{3}(x-1)\right] + \cos^4\left[\frac{\pi}{3}x\right] + \cos^4\left[\frac{\pi}{3}(x+1)\right] \\
&= \frac{1}{4}\left(\left(1+\cos\left[\frac{2\pi}{3}x\right]\right)^2 + \left(1+\cos\left[\frac{2\pi}{3}(x-1)\right]\right)^2 + \left(1+\cos\left[\frac{2\pi}{3}(x+1)\right]\right)^2\right) \\
&= \frac{1}{4}\left(3+\cos^2\left[\frac{2\pi}{3}x\right] + \cos^2\left[\frac{2\pi}{3}(x-1)\right] + \cos^2\left[\frac{2\pi}{3}(x+1)\right]\right. \\
&\quad \left. + 2\cos\left[\frac{2\pi}{3}x\right] + 2\cos\left[\frac{2\pi}{3}(x-1)\right] + 2\cos\left[\frac{2\pi}{3}(x+1)\right]\right) \\
&= \frac{1}{4}\left(3+\cos^2\left[\frac{2\pi}{3}x\right] + \left(\cos\left[\frac{2\pi}{3}x\right]\cos\left[\frac{2\pi}{3}\right] + \sin\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}\right]\right)^2\right. \\
&\quad + \left(\cos\left[\frac{2\pi}{3}x\right]\cos\left[\frac{2\pi}{3}\right] - \sin\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}\right]\right)^2 + 2\cos\left[\frac{2\pi}{3}x\right] \\
&\quad + 2\left(\cos\left[\frac{2\pi}{3}x\right]\cos\left[\frac{2\pi}{3}\right] + \sin\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}\right]\right) \\
&\quad \left. + 2\left(\cos\left[\frac{2\pi}{3}x\right]\cos\left[\frac{2\pi}{3}\right] - \sin\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}\right]\right)\right) \\
&= \frac{1}{4}\left(3+\cos^2\left[\frac{2\pi}{3}x\right] + \left(-\frac{1}{2}\cos\left[\frac{2\pi}{3}x\right] + \frac{\sqrt{3}}{2}\sin\left[\frac{2\pi}{3}x\right]\right)^2\right. \\
&\quad + \left(-\frac{1}{2}\cos\left[\frac{2\pi}{3}x\right] - \frac{\sqrt{3}}{2}\sin\left[\frac{2\pi}{3}x\right]\right)^2 + 2\cos\left[\frac{2\pi}{3}x\right] \\
&\quad \left. + 2\left(-\frac{1}{2}\cos\left[\frac{2\pi}{3}x\right] + \frac{\sqrt{3}}{2}\sin\left[\frac{2\pi}{3}x\right]\right) + 2\left(-\frac{1}{2}\cos\left[\frac{2\pi}{3}x\right] - \frac{\sqrt{3}}{2}\sin\left[\frac{2\pi}{3}x\right]\right)\right)
\end{aligned}$$

$$
\begin{aligned}
=&\frac{1}{4}\left(3+\cos^2\left[\frac{2\pi}{3}x\right]+\frac{1}{4}\cos^2\left[\frac{2\pi}{3}x\right]+\frac{3}{4}\sin^2\left[\frac{2\pi}{3}x\right]-\frac{\sqrt{3}}{2}\cos\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}x\right]\right.\\
&+\frac{1}{4}\cos^2\left[\frac{2\pi}{3}x\right]+\frac{3}{4}\sin^2\left[\frac{2\pi}{3}x\right]+\frac{\sqrt{3}}{2}\cos\left[\frac{2\pi}{3}x\right]\sin\left[\frac{2\pi}{3}x\right]+2\cos\left[\frac{2\pi}{3}x\right]\\
&\left.-\cos\left[\frac{2\pi}{3}x\right]+\sqrt{3}\sin\left[\frac{2\pi}{3}x\right]-\cos\left[\frac{2\pi}{3}x\right]-\sqrt{3}\sin\left[\frac{2\pi}{3}x\right]\right)\\
=&\frac{1}{4}\left(3+\cos^2\left[\frac{2\pi}{3}x\right]+\frac{1}{2}\cos^2\left[\frac{2\pi}{3}x\right]+\frac{3}{2}\sin^2\left[\frac{2\pi}{3}x\right]\right)\\
=&\frac{9}{8}.
\end{aligned}
$$

This case can be generalized for any x that is covered by three channels of this shape that are separated by $\frac{\pi}{3}$.

B.2.2 The constant norm of the channel derivatives

$\sum_{\forall k}|\frac{d}{dx}c_k|^2 = \textit{constant}$ (page 41).

Proof: The derivative of a channel k with respect to x is

$$\frac{d}{dx}c_k = -\frac{2\pi}{3}\cos\left[\frac{\pi}{3}(x-k)\right]\sin\left[\frac{\pi}{3}(x-k)\right].$$

The sum is then

$$\begin{aligned}
\sum \left| \frac{d}{dx} c_k \right|^2 &= \left(\frac{2\pi}{3}\right)^2 \left(\cos^2\left[\frac{\pi}{3}x\right] \sin^2\left[\frac{\pi}{3}x\right] \right. \\
&\quad + \cos^2\left[\frac{\pi}{3}(x-1)\right] \sin^2\left[\frac{\pi}{3}(x-1)\right] \\
&\quad \left. + \cos^2\left[\frac{\pi}{3}(x+1)\right] \sin^2\left[\frac{\pi}{3}(x+1)\right] \right) \\
&= \left(\frac{\pi}{3}\right)^2 \left(\sin^2\left[\frac{2\pi}{3}x\right] + \sin^2\left[\frac{2\pi}{3}(x-1)\right] + \sin^2\left[\frac{2\pi}{3}(x+1)\right] \right) \\
&= \left(\frac{\pi}{3}\right)^2 \left(\sin^2\left[\frac{2\pi}{3}x\right] + \left(\sin\left[\frac{2\pi}{3}x\right] \cos\left[\frac{2\pi}{3}\right] - \cos\left[\frac{2\pi}{3}x\right] \sin\left[\frac{2\pi}{3}\right] \right)^2 \right. \\
&\quad \left. + \sin^2\left[\frac{2\pi}{3}x\right] + \left(\sin\left[\frac{2\pi}{3}x\right] \cos\left[\frac{2\pi}{3}\right] + \cos\left[\frac{2\pi}{3}x\right] \sin\left[\frac{2\pi}{3}\right] \right)^2 \right) \\
&= \left(\frac{\pi}{3}\right)^2 \left(\sin^2\left[\frac{2\pi}{3}x\right] + 2\left(\sin^2\left[\frac{2\pi}{3}x\right] \cos^2\left[\frac{2\pi}{3}\right] + \cos^2\left[\frac{2\pi}{3}x\right] \sin^2\left[\frac{2\pi}{3}\right] \right) \right) \\
&= \left(\frac{\pi}{3}\right)^2 \left(\sin^2\left[\frac{2\pi}{3}x\right] + 2\left(\frac{3}{4}\cos^2\left[\frac{2\pi}{3}x\right] + \frac{1}{4}\sin^2\left[\frac{2\pi}{3}x\right] \right) \right) \\
&= \left(\frac{\pi}{3}\right)^2 \frac{3}{4} \left(\cos^2\left[\frac{2\pi}{3}x\right] + \sin^2\left[\frac{2\pi}{3}x\right] \right) \\
&= \frac{\pi^2}{12}.
\end{aligned}$$

B.2.3 Derivation of the update rule for the prediction matrix memory

$$r = p + a\|\mathbf{q}\|^2\|\mathbf{v}\|^2 \tag{3.18}$$

Proof: By inserting equation 3.17 on page 49 into equation 3.14 on page 49, we get

$$\begin{aligned}
r &= \langle \mathbf{W} + a\mathbf{q}\mathbf{v}^T \mid \mathbf{q}\mathbf{v}^T \rangle \\
&= \langle \mathbf{W} \mid \mathbf{q}\mathbf{v}^T \rangle + a\langle \mathbf{q}\mathbf{v}^T \mid \mathbf{q}\mathbf{v}^T \rangle \\
&= p + a(\mathbf{q}^T\mathbf{q}\mathbf{v}^T\mathbf{v}) \\
&= p + a\|\mathbf{q}\|^2\|\mathbf{v}\|^2.
\end{aligned}$$

B.2.4 One frequency spans a 2-D plane

One frequency component defines an ellipse and, hence, spans a two-dimensional plane (page 51).

Proof: Consider a signal with frequency ω in an n-dimensional space:

$$\begin{pmatrix} a_1 \sin(\omega t + \alpha_1) \\ a_2 \sin(\omega t + \alpha_2) \\ \vdots \\ a_n \sin(\omega t + \alpha_n) \end{pmatrix} = \sin(\omega t) \begin{pmatrix} a_1 \cos\alpha_1 \\ a_2 \cos\alpha_2 \\ \vdots \\ a_n \cos\alpha_n \end{pmatrix} + \cos(\omega t) \begin{pmatrix} a_1 \sin\alpha_1 \\ a_2 \sin\alpha_2 \\ \vdots \\ a_n \sin\alpha_n \end{pmatrix} \tag{B.4}$$

$$= \mathbf{v}_1 \sin(\omega t) + \mathbf{v}_2 \cos(\omega t)$$

Remark: It should be noted that the two-dimensionality is caused by the different phases α_i. If all components have the same phase, the signal spans only one dimension.

B.3 Proofs for chapter 4

B.3.1 Orthogonality in the metrics A and B

$$\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = \begin{cases} 0 & \text{for} \quad i \neq j \\ \beta_i > 0 & \text{for} \quad i = j \end{cases} \quad \text{and} \quad \hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = \begin{cases} 0 & \text{for} \quad i \neq j \\ r_i \beta_i & \text{for} \quad i = j \end{cases} \tag{4.6}$$

Proof: For solution i we have

$$\mathbf{A} \hat{\mathbf{w}}_i = r_i \mathbf{B} \hat{\mathbf{w}}_i. \tag{B.5}$$

The scalar product with another eigenvector gives

$$\hat{\mathbf{w}}_j^T \mathbf{A} \hat{\mathbf{w}}_i = r_i \hat{\mathbf{w}}_j^T \mathbf{B} \hat{\mathbf{w}}_i \tag{B.6}$$

and of course also

$$\hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = r_j \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j. \tag{B.7}$$

Since $\mathbf{A}$ and $\mathbf{B}$ are Hermitian we can change positions of $\hat{\mathbf{w}}_i$ and $\hat{\mathbf{w}}_j$ which gives

$$r_j \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = r_i \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j \tag{B.8}$$

and hence

$$(r_i - r_j) \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = 0. \tag{B.9}$$

For this expression to be true when $i \neq j$, we have that $\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = 0$ if $r_i \neq r_j$. For $i = j$ we now have that $\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i = \beta_i > 0$ since $\mathbf{B}$ is positive definite. In the same way we have

$$\left(\frac{1}{r_i} - \frac{1}{r_j} \right) \hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = 0, \tag{B.10}$$

which means that $\hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = 0$ for $i \neq j$. For $i = j$ we know that $\hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_i = r_i \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i = r_i \beta_i$.

B.3.2 Linear independence

$\{\mathbf{w}_i\}$ are linearly independent.

Proof: Suppose $\{\mathbf{w}_i\}$ are *not* linearly independent. This would mean that we could write an eigenvector $\mathbf{w}_k$ as

$$\hat{\mathbf{w}}_k = \sum_{j \neq k} \gamma_j \hat{\mathbf{w}}_j. \tag{B.11}$$

This means that for $j \neq k$,

$$\mathbf{w}_j^T \mathbf{B} \mathbf{w}_k = \gamma_j \mathbf{w}_j^T \mathbf{B} \mathbf{w}_j \neq 0 \tag{B.12}$$

which violates equation 4.6 on page 63. Hence, $\{\mathbf{w}_i\}$ are linearly independent.

B.3.3 The range of r

$$r_n \leq r \leq r_1 \tag{4.7}$$

Proof: If we express a vector $\mathbf{w}$ in the base of the eigenvectors $\hat{\mathbf{w}}_i$, i.e.

$$\mathbf{w} = \sum_i \gamma_i \hat{\mathbf{w}}_i, \tag{B.13}$$

we can write

$$r = \frac{\sum \gamma_i \hat{\mathbf{w}}_i^T \mathbf{A} \sum \gamma_i \hat{\mathbf{w}}_i}{\sum \gamma_i \hat{\mathbf{w}}_i^T \mathbf{B} \sum \gamma_i \hat{\mathbf{w}}_i} = \frac{\sum \gamma_i^2 \alpha_i}{\sum \gamma_i^2 \beta_i}, \tag{B.14}$$

where $\alpha_i = \hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_i$ and $\beta_i = \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i$, since $\hat{\mathbf{w}}_i^T \mathbf{A} \hat{\mathbf{w}}_j = \hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_j = 0$ for $i \neq j$. Now, since $\alpha_i = \beta_i r_i$ (see equation 4.6 on page 63), we get

$$r = \frac{\sum \gamma_i^2 \beta_i r_i}{\sum \gamma_i^2 \beta_i}. \tag{B.15}$$

Obviously this function has the maximum value r_1 when $\gamma_1 \neq 0$ and $\gamma_i = 0 \,\forall\, i > 1$ if r_1 is the largest eigenvalue. The minimum value, r_n, is obtained when $\gamma_n \neq 0$ and $\gamma_i = 0 \,\forall\, i < n$ if r_n is the smallest eigenvalue.

B.3.4 The second derivative of r

$$\mathbf{H}_i = \left. \frac{\partial^2 r}{\partial \mathbf{w}^2} \right|_{\mathbf{w}=\hat{\mathbf{w}}_i} = \frac{2}{\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i} (\mathbf{A} - r_i \mathbf{B}) \tag{4.8}$$

Proof: From the gradient in equation 4.3 on page 61 we get the second derivative as

$$\frac{\partial^2 r}{\partial \mathbf{w}^2} = \frac{2}{(\mathbf{w}^T \mathbf{B} \mathbf{w})^2} \left[\left(\mathbf{A} - \frac{\partial r}{\partial \mathbf{w}} \mathbf{w}^T \mathbf{B} - r\mathbf{B} \right) \mathbf{w}^T \mathbf{B} \mathbf{w} - (\mathbf{A}\mathbf{w} - r\mathbf{B}\mathbf{w}) 2\mathbf{w}^T \mathbf{B} \right]. \tag{B.16}$$

If we insert one of the solutions $\hat{\mathbf{w}}_i$, we have

$$\left. \frac{\partial r}{\partial \mathbf{w}} \right|_{\mathbf{w}=\hat{\mathbf{w}}_i} = \frac{2}{\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i} (\mathbf{A}\hat{\mathbf{w}}_i - r\mathbf{B}\hat{\mathbf{w}}_i) = \mathbf{0} \tag{B.17}$$

and hence

$$\left. \frac{\partial^2 r}{\partial \mathbf{w}^2} \right|_{\mathbf{w}=\hat{\mathbf{w}}_i} = \frac{2}{\hat{\mathbf{w}}_i^T \mathbf{B} \hat{\mathbf{w}}_i} (\mathbf{A} - r_i \mathbf{B}). \tag{B.18}$$

B.3.5 Positive eigenvalues of the Hessian

There exists a $\mathbf{w}$ such that

$$\mathbf{w}^T \mathbf{H}_i \mathbf{w} > 0 \quad \forall\, i > 1 \tag{4.9}$$

Proof: If we express a vector $\mathbf{w}$ as a linear combination of the eigenvectors we get

$$\begin{aligned}\frac{\beta_i}{2}\mathbf{w}^T\mathbf{H}_i\mathbf{w} &= \mathbf{w}^T(\mathbf{A}-r_i\mathbf{B})\mathbf{w}\\ &= \mathbf{w}^T\mathbf{B}(\mathbf{B}^{-1}\mathbf{A}-r_i\mathbf{I})\mathbf{w}\\ &= \sum\gamma_j\hat{\mathbf{w}}_j^T\mathbf{B}(\mathbf{B}^{-1}\mathbf{A}-r_i\mathbf{I})\sum\gamma_j\hat{\mathbf{w}}_j\\ &= \sum\gamma_j\hat{\mathbf{w}}_j^T\mathbf{B}\left(\sum r_j\gamma_j\hat{\mathbf{w}}_j-\sum r_i\gamma_j\hat{\mathbf{w}}_j\right)\\ &= \sum\gamma_j\hat{\mathbf{w}}_j^T\mathbf{B}\sum(r_j-r_i)\gamma_j\hat{\mathbf{w}}_j\\ &= \sum\gamma_j^2\beta_j(r_j-r_i),\end{aligned} \tag{B.19}$$

where $\beta_i=\hat{\mathbf{w}}_i^T\mathbf{B}\hat{\mathbf{w}}_i>0$. Now, $(r_j-r_i)>0$ for $j<i$ so if $i>1$ there is at least one choice of $\mathbf{w}$ that makes this sum positive.

B.3.6 The partial derivatives of the covariance

$$\begin{cases}\frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{1}{\|\mathbf{w}_x\|}(\mathbf{C}_{xy}\hat{\mathbf{w}}_y-\rho\hat{\mathbf{w}}_x)\\ \frac{\partial\rho}{\partial\mathbf{w}_y} &= \frac{1}{\|\mathbf{w}_y\|}(\mathbf{C}_{yx}\hat{\mathbf{w}}_x-\rho\hat{\mathbf{w}}_y).\end{cases} \tag{4.17}$$

Proof: The partial derivative of ρ with respect to $\mathbf{w}_x$ is

$$\begin{aligned}\frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{\mathbf{C}_{xy}\mathbf{w}_y\|\mathbf{w}_x\|\|\mathbf{w}_y\|-\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y\|\mathbf{w}_x\|^{-1}\mathbf{w}_x\|\mathbf{w}_y\|}{\|\mathbf{w}_x\|^2\|\mathbf{w}_y\|^2}\\ &= \frac{\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\|\mathbf{w}_x\|}-\frac{\rho\mathbf{w}_x}{\|\mathbf{w}_x\|^2}\\ &= \frac{1}{\|\mathbf{w}_x\|}(\mathbf{C}_{xy}\hat{\mathbf{w}}_y-\rho\hat{\mathbf{w}}_x)\end{aligned}$$

The same calculations can be made for $\frac{\partial\rho}{\partial\mathbf{w}_y}$ by exchanging x and y .

B.3.7 The partial derivatives of the correlation

$$\begin{cases}\frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{xy}\hat{\mathbf{w}}_y-\frac{\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}\mathbf{C}_{xx}\hat{\mathbf{w}}_x\right)\\ \frac{\partial\rho}{\partial\mathbf{w}_y} &= \frac{a}{\|\mathbf{w}_y\|}\left(\mathbf{C}_{yx}\hat{\mathbf{w}}_x-\frac{\hat{\mathbf{w}}_y^T\mathbf{C}_{yx}\hat{\mathbf{w}}_x}{\hat{\mathbf{w}}_y^T\mathbf{C}_{yy}\hat{\mathbf{w}}_y}\mathbf{C}_{yy}\hat{\mathbf{w}}_y\right)\end{cases} \tag{4.25}$$

Proof: The partial derivative of ρ with respect to $\mathbf{w}_x$ is

$$\begin{aligned}
\frac{\partial \rho}{\partial \mathbf{w}_x} &= \frac{(\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y)^{1/2} \mathbf{C}_{xy} \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y} \\
&\quad - \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y (\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y)^{-1/2} \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y} \\
&= (\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{C}_{yy} \mathbf{w}_y)^{-1/2} \left(\mathbf{C}_{xy} \mathbf{w}_y - \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x} \mathbf{C}_{xx} \mathbf{w}_x \right) \\
&= \|\mathbf{w}_x\|^{-1} (\underbrace{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x \hat{\mathbf{w}}_y^T \mathbf{C}_{yy} \hat{\mathbf{w}}_y}_{\geq 0})^{-1/2} \left(\mathbf{C}_{xy} \hat{\mathbf{w}}_y - \frac{\hat{\mathbf{w}}_x^T \mathbf{C}_{xy} \hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x} \mathbf{C}_{xx} \hat{\mathbf{w}}_x \right) \\
&= \frac{a}{\|\mathbf{w}_x\|} \left(\mathbf{C}_{xy} \hat{\mathbf{w}}_y - \frac{\hat{\mathbf{w}}_x^T \mathbf{C}_{xy} \hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x} \mathbf{C}_{xx} \hat{\mathbf{w}}_x \right), \qquad a \geq 0.
\end{aligned}$$

The same calculations can be made for $\frac{\partial \rho}{\partial \mathbf{w}_y}$ by exchanging x and y .

B.3.8 Invariance with respect to linear transformations

Canonical correlations are invariant with respect to linear transformations.

Proof: Let

$$\mathbf{x} = \mathbf{A}_x \mathbf{x}' \quad \text{and} \quad \mathbf{y} = \mathbf{A}_y \mathbf{y}', \tag{B.20}$$

where $\mathbf{A}_x$ and $\mathbf{A}_y$ are non-singular matrices. If we denote

$$\mathbf{C}'_{xx} = E[\mathbf{x}' \mathbf{x}'^T], \tag{B.21}$$

the covariance matrix for $\mathbf{x}$ can be written as

$$\mathbf{C}_{xx} = E[\mathbf{x}\mathbf{x}^T] = E[\mathbf{A}_x \mathbf{x}' \mathbf{x}'^T \mathbf{A}_x^T] = \mathbf{A}_x \mathbf{C}'_{xx} \mathbf{A}_x^T. \tag{B.22}$$

In the same way we have

$$\mathbf{C}_{xy} = \mathbf{A}_x \mathbf{C}'_{xy} \mathbf{A}_y^T \quad \text{and} \quad \mathbf{C}_{yy} = \mathbf{A}_y \mathbf{C}'_{yy} \mathbf{A}_y^T. \tag{B.23}$$

Now, the equation system 4.26 on page 68 can be written as

$$\begin{cases} \mathbf{A}_x^T \mathbf{C}'_{xy} \mathbf{A}_y \hat{\mathbf{w}}_y &= \rho \lambda_x \mathbf{A}_x^T \mathbf{C}'_{xx} \mathbf{A}_x \hat{\mathbf{w}}_x \\ \mathbf{A}_y^T \mathbf{C}'_{yx} \mathbf{A}_x \hat{\mathbf{w}}_x &= \rho \lambda_y \mathbf{A}_y^T \mathbf{C}'_{yy} \mathbf{A}_y \hat{\mathbf{w}}_y \end{cases} \tag{B.24}$$

or

$$\begin{cases} \mathbf{C}'_{xy}\hat{\mathbf{w}}'_y &= \rho\lambda_x \mathbf{C}'_{xx}\hat{\mathbf{w}}'_x \\ \mathbf{C}'_{yx}\hat{\mathbf{w}}'_x &= \rho\lambda_y \mathbf{C}'_{yy}\hat{\mathbf{w}}'_y, \end{cases} \tag{B.25}$$

where $\hat{\mathbf{w}}'_x = \mathbf{A}_x^T \hat{\mathbf{w}}_x$ and $\hat{\mathbf{w}}'_y = \mathbf{A}_y^T \hat{\mathbf{w}}_y$. Obviously this transformation leaves the roots ρ unchanged. If we look at the canonical variates,

$$\begin{cases} x' &= \mathbf{w}'^T_x \mathbf{x}' = \mathbf{w}_x^T \mathbf{A}_x \mathbf{A}_x^{-1}\mathbf{x} = x \\ y' &= \mathbf{w}'^T_y \mathbf{y}' = \mathbf{w}_y^T \mathbf{A}_y \mathbf{A}_y^{-1}\mathbf{y} = y, \end{cases} \tag{B.26}$$

we see that these too are unaffected by the linear transformation.

B.3.9 Relationship between mutual information and canonical correlation

$$I(\mathbf{x};\mathbf{y}) = \frac{1}{2}\log\left(\frac{1}{\prod_i (1-\rho_i^2)}\right), \tag{4.32}$$

where $\mathbf{x}$ and $\mathbf{y}$ are N-dimensional Gaussian variables and ρ_i are the canonical correlations.

Proof: The differential entropy of a multidimensional Gaussian variable is

$$h(\mathbf{z}) = \frac{1}{2}\log\left((2\pi e)^N |\mathbf{C}|\right), \tag{B.27}$$

where $|\mathbf{C}|$ is the determinant of the covariance matrix of $\mathbf{z}$ and N is the dimensionality of $\mathbf{z}$ (see proof B.1.1 on page 153). If $\mathbf{z} = \binom{\mathbf{x}}{\mathbf{y}}$, the covariance matrix $\mathbf{C}$ can be written as

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{xx} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{C}_{yy} \end{bmatrix}. \tag{B.28}$$

By using the relation

$$|\mathbf{C}| = |\mathbf{C}_{xx}|\,|\mathbf{C}_{yy} - \mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}| \tag{B.29}$$

(Kailath, 1980, page 650) and equation 2.42 on page 30, we get

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &= \frac{1}{2}\log\left(\frac{|\mathbf{C}_{xx}|\,|\mathbf{C}_{yy}|}{|\mathbf{C}|}\right) \\ &= -\frac{1}{2}\log\left(\frac{|\mathbf{C}_{yy} - \mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}|}{|\mathbf{C}_{yy}|}\right) \\ &= -\frac{1}{2}\log\left(|\mathbf{I} - \mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}|\right) \end{aligned} \tag{B.30}$$

assuming the covariance matrices $\mathbf{C}_{xx}$ and $\mathbf{C}_{yy}$ being non-singular. The eigenvalues to $\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}$ are the squared canonical correlations (see equation 4.28 on page 68). Hence, an eigenvalue decomposition gives

$$\begin{aligned} I(\mathbf{x};\mathbf{y}) &= -\frac{1}{2}\log\left|\mathbf{I} - \begin{bmatrix} \rho_1^2 & & 0 \\ & \rho_2^2 & \\ & \ddots & \\ 0 & & \rho_n \end{bmatrix}\right| = -\frac{1}{2}\log\prod_i(1-\rho_i^2) \\ &= \frac{1}{2}\log\left(\frac{1}{\prod_i(1-\rho_i^2)}\right) \end{aligned} \tag{B.31}$$

since the eigenvalue decomposition does not change the identity matrix.

B.3.10 The partial derivatives of the MLR-quotient

$$\begin{cases} \frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{xy}\hat{\mathbf{w}}_y - \beta\mathbf{C}_{xx}\hat{\mathbf{w}}_x\right) \\ \frac{\partial\rho}{\partial\mathbf{w}_y} &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{yx}\hat{\mathbf{w}}_x - \frac{\rho^2}{\beta}\hat{\mathbf{w}}_y\right). \end{cases} \tag{4.44}$$

Proof: The partial derivative of ρ with respect to $\mathbf{w}_x$ is

$$\begin{aligned} \frac{\partial\rho}{\partial\mathbf{w}_x} &= \frac{(\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y)^{1/2}\mathbf{C}_{xy}\mathbf{w}_y}{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y} \\ &\quad - \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y(\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y)^{-1/2}\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y}{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y} \\ &= (\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{w}_y)^{-1/2}\left(\mathbf{C}_{xy}\mathbf{w}_y - \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y}{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x}\mathbf{C}_{xx}\mathbf{w}_x\right) \\ &= \|\mathbf{w}_x\|^{-1}(\underbrace{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x\hat{\mathbf{w}}_y^T\hat{\mathbf{w}}_y}_{\geq 0})^{-1/2}\left(\mathbf{C}_{xy}\hat{\mathbf{w}}_y - \frac{\hat{\mathbf{w}}_x^T\mathbf{C}_{xy}\hat{\mathbf{w}}_y}{\hat{\mathbf{w}}_x^T\mathbf{C}_{xx}\hat{\mathbf{w}}_x}\mathbf{C}_{xx}\hat{\mathbf{w}}_x\right) \\ &= \frac{a}{\|\mathbf{w}_x\|}\left(\mathbf{C}_{xy}\hat{\mathbf{w}}_y - \beta\mathbf{C}_{xx}\hat{\mathbf{w}}_x\right), \qquad a \geq 0. \end{aligned}$$

The partial derivative of ρ with respect to $\mathbf{w}_y$ is

$$\begin{aligned}
\frac{\partial \rho}{\partial \mathbf{w}_y} &= \frac{(\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y)^{1/2} \mathbf{C}_{yx} \mathbf{w}_x}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y} \\
&\quad - \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y (\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y)^{-1/2} \mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y} \\
&= (\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y)^{-1/2} \left(\mathbf{C}_{yx} \mathbf{w}_x - \frac{\mathbf{w}_x^T \mathbf{C}_{xy} \mathbf{w}_y \mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x}{\mathbf{w}_x^T \mathbf{C}_{xx} \mathbf{w}_x \mathbf{w}_y^T \mathbf{w}_y} \mathbf{w}_y \right) \\
&= \|\mathbf{w}_y\|^{-1} (\underbrace{\hat{\mathbf{w}}_x^T \mathbf{C}_{xx} \hat{\mathbf{w}}_x}_{\geq 0})^{-1/2} \left(\mathbf{C}_{yx} \hat{\mathbf{w}}_x - \hat{\mathbf{w}}_x^T \mathbf{C}_{xy} \hat{\mathbf{w}}_y \hat{\mathbf{w}}_y \right) \\
&= \frac{a}{\|\mathbf{w}_x\|} \left(\mathbf{C}_{yx} \hat{\mathbf{w}}_x - \frac{\rho^2}{\beta} \hat{\mathbf{w}}_y \right), \qquad a \geq 0.
\end{aligned}$$

B.3.11 The successive eigenvalues

$$\mathbf{H} = \mathbf{G} - \lambda_1 \hat{\mathbf{e}}_1 \mathbf{f}_1^T \tag{4.59}$$

Proof: Consider a vector $\mathbf{u}$ which is expressed as the sum of one vector parallel to the eigenvector $\hat{\mathbf{e}}_1$, and another vector $\mathbf{u}_o$ that is a linear combination of the other eigenvectors and, hence, orthogonal to the dual vector $\mathbf{f}_1$.

$$\mathbf{u} = a\hat{\mathbf{e}}_1 + \mathbf{u}_o, \tag{B.32}$$

where

$$\mathbf{f}_1^T \hat{\mathbf{e}}_1 = 1 \quad \text{and} \quad \mathbf{f}_1^T \hat{\mathbf{u}}_o = 0.$$

Multiplying $\mathbf{H}$ with $\mathbf{u}$ gives

$$\begin{aligned}
\mathbf{H}\mathbf{u} &= \left(\mathbf{G} - \lambda_1 \hat{\mathbf{e}}_1 \mathbf{f}_1^T \right) (a\hat{\mathbf{e}}_1 + \mathbf{u}_o) \\
&= a(\mathbf{G}\hat{\mathbf{e}}_1 - \lambda_1 \hat{\mathbf{e}}_1) + (\mathbf{G}\mathbf{u}_o - \mathbf{0}) \\
&= \mathbf{G}\mathbf{u}_o.
\end{aligned} \tag{B.33}$$

This shows that $\mathbf{G}$ and $\mathbf{H}$ have the same eigenvectors and eigenvalues except for the largest eigenvalue and eigenvector of $\mathbf{G}$. Obviously the eigenvector corresponding to the largest eigenvalue of $\mathbf{H}$ is $\hat{\mathbf{e}}_2$.

B.4 Proofs for chapter 7

B.4.1 Real-valued canonical correlations

The canonical correlations ρ_i are real valued.

Proof: The canonical correlations are eigenvalues to the matrix $\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}$:

$$\mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}\mathbf{w}_x = \rho_i \mathbf{w}_x \tag{B.34}$$

$\mathbf{A} = \mathbf{C}_{xx}^{-1}$ is Hermitian and positive definite. $\mathbf{B} = \mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}$ is Hermitian. Then $\mathbf{AB} = \mathbf{C}_{xx}^{-1}\mathbf{C}_{xy}\mathbf{C}_{yy}^{-1}\mathbf{C}_{yx}$ is Hermitian (see proof B.4.2) and, hence, got real-valued eigenvalues.

It should be noted that if $\mathbf{A} = \mathbf{C}_{xx}^{-1}$ only is positive semidefinite, $\mathbf{A}$ and $\mathbf{B}$ can be projected into a subspace spanned by the eigenvectors of $\mathbf{A}$ corresponding to the non-zero eigenvalues. This will give two new matrices $\mathbf{A}'$ and $\mathbf{B}'$ with the same non-zero eigenvalues as $\mathbf{A}$ and $\mathbf{B}$ but with $\mathbf{A}'$ positive definite. In this way it can be shown that all non-zero correlations are real valued.

B.4.2 Hermitian matrices

If $\mathbf{A}$ is Hermitian and positive definite and $\mathbf{B}$ is Hermitian then $\mathbf{AB}$ is Hermitian.

Proof: By writing the singular value decomposition $\mathbf{A} = \mathbf{U}^*\mathbf{D}\mathbf{U}$ we see that also

$$\mathbf{C} = \mathbf{U}^*\mathbf{D}^{1/2}\mathbf{U} = \mathbf{A}^{1/2} \tag{B.35}$$

is Hermitian and positive definite. Then

$$\mathbf{CBC} = \mathbf{C}^*\mathbf{B}^*\mathbf{C}^* = (\mathbf{CBC})^* \tag{B.36}$$

is Hermitian. But $\mathbf{CBC}$ and $\mathbf{AB}$ has got the same eigenvalues since

$$\mathbf{AB} = \mathbf{C}^2\mathbf{B} = \mathbf{C}(\mathbf{CBC})\mathbf{C}^{-1} \tag{B.37}$$

is only a change of basis which does not change the eigenvalues.

Bibliography

Anderson, J. A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14:197–220.

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. *IEEE Transactions on Systems, Man, and Cybernetics*, 14:799–815.

Anderson, T. W. (1984). *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, second edition.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, CA. Armand Prieditis and Stuart Russell, eds.

Baker, W. L. and Farell, J. A. (1992). *Handbook of intelligent control*, chapter An introduction to connectionist learning control systems, pages 35–63. Van Nostrand Reinhold, New York.

Ballard, D. H. (1987). *Vision, Brain, and Cooperative Computation*, chapter Cortical Connections and Parallel Processing: Structure and Function. MIT Press. M. A. Arbib and A. R. Hanson, Eds.

Ballard, D. H. (1990). *Computational Neuroscience*, chapter Modular Learning in Hierarchical Neural Networks. MIT Press. E. L. Schwartz, Ed.

Barlow, H. (1989). Unsupevised learning. *Neural Computation*, 1:295–311.

Barlow, H. B., Kaushal, T. P., and Mitchson, G. J. (1989). Finding minimum entropy codes. *Neural Computation*, 1:412–423.

Barnard, S. T. and Fichsler, M. A. (1982). Computational Stereo. *ACM Comput. Surv.*, 14:553–572.

Barto, A. G. (1992). *Handbook of Intelligent Control*, chapter Reinforcement Learning and Adaptive Critic Methods. Van Nostrand Reinhold, New York. D. A. White and D. A. Sofge, Eds.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-13(8):834–846.

Battiti, R. (1992). First and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4:141–166.

Becker, S. (1996). Mutual information maximization: models of cortical self-organization. *Network: Computation in Neural Systems*, 7:7–31.

Becker, S. and Hinton, G. E. (1992). Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(9):161–163.

Becker, S. and Hinton, G. E. (1993). Learning mixture models of spatial coherence. *Neural Computation*, 5(2):267–277.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–59.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bloom, F. E. and Lazerson, A. (1985). *Brain, Mind, and Behavior*. W. H. Freeman and Company.

Bock, R. D. (1975). *Multivariate Statistical Methods in Behavioral Research*. McGraw-Hill series in psychology. McGraw-Hill.

Borga, M. (1993). Hierarchical Reinforcement Learning. In Gielen, S. and Kappen, B., editors, *ICANN'93*, Amsterdam. Springer-Verlag.

Borga, M. (1995). Reinforcement Learning Using Local Adaptive Models. Thesis No. 507, ISBN 91–7871–590–3.

Borga, M. and Knutsson, H. (1998). An adaptive stereo algorithm based on canonical correlation analysis. Submitted to ICIPS'98.

Borga, M., Knutsson, H., and Landelius, T. (1997a). Learning Canonical Correlations. In *Proceedings of the 10th Scandinavian Conference on Image Analysis*, Lappeenranta, Finland. SCIA.

Borga, M., Landelius, T., and Knutsson, H. (1997b). A unified approach to PCA, PLS, MLR and CCA. *Information Sciences*. Submitted. Revised for second review.

Bower, G. H. and Hilgard, E. R. (1981). *Theories of Learning*. Prentice–Hall, Englewood Cliffs, N.J. 07632, 5 edition.

Bracewell, R. (1986). *The Fourier Transform and its Applications*. McGraw-Hill, 2nd edition.

Bradtke, S. J. (1993). Reinforcement learning applied to linear quadratic regulation. In *Advances in Neural Information Processing Systems 5*, San Mateo, CA. Morgan Kaufmann.

Bregler, C. and Omohundro, S. M. (1994). Surface learning with applications to lipreading. In *Advances in Neural Information Processing Systems 6*, pages 43–50, San Francisco. Morgan Kaufmann.

Brooks, V. B. (1986). *The Neural Basis of Motor Control*. Oxford University Press.

Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Carson, J. and Fry, T. (1937). Variable frequency electric circuit theory with application to the theory of frequency modulation. *Bell Syste Tech. J.*, 16:513–540.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314.

Coren, S. and Ward, L. M. (1989). *Sensation & Perception*. Harcourt Brace Jovanovich, Publishers, San Diego, USA, 3rd edition. ISBN 0–15–579647–X.

Das, S. and Sen, P. K. (1994). Restricted canonical correlations. *Linear Algebra and its Applications*, 210:29–47.

Davis, L., editor (1987). *Genetic Algorithms and Simulated Anealing*. Pitman, London.

Denoeux, T. and Lengellé, R. (1993). Initializing back propagation networks with prototypes. *Neural Networks*, 6(3):351–363.

Derin, H. and Kelly, P. A. (1989). Discrete-index markov-type random processes. In *Proceedings of IEEE*, volume 77.

Duda, R. O. and Hart, P. E. (1973). *Pattern classification and scene analysis*. Wiley-Interscience, New York.

Fieguth, P. W., Irving, W. W., and Willsky, A. S. (1995). Multiresolution model development for overlapping trees via canonical correlation analysis. In *International Conference on Image Processing*, pages 45–48, Washington DC. IEEE.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*. in press.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7(Part II):179–180. Also in *Contributions to Mathematical Statisitcs (John Wiley, New York, 1950)*.

Földiák, F. (1990). Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *Computer Journal*, 7:149–154.

Geladi, P. and Kowalski, B. R. (1986). Parial least-squares regression: a tutorial. *Analytica Chimica Acta*, 185:1–17.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

Giles, G. L. and Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, 26(23):4972–4978.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Golub, G. H. and Loan, C. F. V. (1989). *Matrix Computations*. The Johns Hopkins University Press, second edition.

Granlund, G. H. (1978). In search of a general picture processing operator. *Computer Graphics and Image Processing*, 8(2):155–178.

Granlund, G. H. (1988). Integrated analysis-response structures for robotics systems. Report LiTH–ISY–I–0932, Computer Vision Laboratory, Linköping University, Sweden.

Granlund, G. H. (1989). Magnitude representation of features in image analysis. In *The 6th Scandinavian Conference on Image Analysis*, pages 212–219, Oulu, Finland.

Granlund, G. H. (1997). From multidimensional signals to the generation of responses. In Sommer, G. and Koenderink, J. J., editors, *Algebraic Frames for the Perception-Action Cycle*, volume 1315 of *Lecture Notes in Computer Science*, pages 29–53, Kiel, Germany. Springer-Verlag. International Workshop, AFPAC'97, invited paper.

Granlund, G. H. and Knutsson, H. (1982). Hierarchical processing of structural information in artificial intelligence. In *Proceedings of 1982 IEEE Conference on Acoustics, Speech and Signal Processing*, Paris. IEEE. Invited Paper.

Granlund, G. H. and Knutsson, H. (1983). Contrast of structured and homogenous representations. In Braddick, O. J. and Sleigh, A. C., editors, *Physical and Biological Processing of Images*, pages 282–303. Springer Verlag, Berlin.

Granlund, G. H. and Knutsson, H. (1990). Compact associative representation of visual information. In *Proceedings of The 10th International Conference on Pattern Recognition*. Report LiTH–ISY–I–1091, Linköping University, Sweden, 1990.

Granlund, G. H. and Knutsson, H. (1995). *Signal Processing for Computer Vision*. Kluwer Academic Publishers. ISBN 0-7923-9530-1.

Gray, R. M. (1984). Vector quaantization. *IEEE ASSP Magazine*, 1:4–29.

Gray, R. M. (1990). *Entropy and Information Theory*. Springer-Verlag, New York.

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company.

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York.

Heger, M. (1994). Consideration of risk in reinforcent learning. In Cohen, W. W. and Hirsh, H., editors, *Proceedings of the 11th International Conference on Machine Learning*, pages 105–111, Brunswick, NJ.

Held, R. and Bossom, J. (1961). Neonatal deprivation and adult rearrangement. Complementary techniques for analyzing plastic sensory–motor coordinations. *Journal of Comparative and Physiological Psychology*, pages 33–37.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley.

Hinton, G. E. and Nowlan, S. J. (1987). How learning can guide evolution. *Complex Systems*, pages 495–502.

Hinton, G. E. and Sejnowski, T. J. (1983). Optimal perceptual inference. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 448–453, Washington DC.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In Rummelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in Microstructures of Cognition*. MIT Press, Cambridge, MA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational capabilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558.

Hornby, A. S. (1989). *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press, Oxford, fourth edition. A. P. Cowie (ed.).

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 498–520.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28:321–377.

Höskuldsson, A. (1988). PLS regression methods. *Journal of Chemometrics*, 2:211–228.

Hubel, D. H. (1988). *Eye, Brain and Vision*, volume 22 of *Scientific American Library*. W. H. Freeman and Company. ISBN 0–7167–5020–1.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148:574–591.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's striate cortex. *J. Physiol.*, 160:106–154.

Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5:248–264.

Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6:1185–1201.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaption. *Neural Networks*, 1:295–307.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Jepson, A. D. and Fleet, D. J. (1990). Scale-space singularities. In Faugeras, O., editor, *Computer Vision-ECCV90*, pages 50–55. Springer-Verlag.

Johansson, B. (1997). Multidimensional signal recognition, invariant to affine transformation and time-shift, using canonical correlation. Master's thesis, Linköpings universitet. LiTH-ISY-EX-1825.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214.

Kailath, T. (1980). *Linear Systems*. Information and System Sciences Series. Prentice-Hall, Englewood Cliffs, N.J.

Karhunen, K. (1947). Uber lineare methoden in der Wahrsccheilichkeitsrechnung. *Annales Academiae Scientiarum Fennicae, Seried A1: Mathematica-Physica*, 37:3–79.

Kay, J. (1992). Feature discovery under contextual supervision using mutual information. In *International Joint Conference on Neural Networks*, volume 4, pages 79–84. IEEE.

Knutsson, H. (1982). *Filtering and Reconstruction in Image Processing*. PhD thesis, Linköping University, Sweden. Diss. No. 88.

Knutsson, H. (1985). Producing a continuous and distance preserving 5-D vector representation of 3-D orientation. In *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management - CAPAIDM*, pages 175–182, Miami Beach, Florida. IEEE. Report LiTH–ISY–I–0843, Linköping University, Sweden, 1986.

Knutsson, H. (1989). Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland. Report LiTH–ISY–I–1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.

Knutsson, H., Borga, M., and Landelius, T. (1995). Learning Canonical Correlations. Report LiTH-ISY-R-1761, Computer Vision Laboratory, S–581 83 Linköping, Sweden.

Kohonen, T. (1972). Correlation matrix memories. *IEEE Trans.s on Computers*, C-21:353–359.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Kohonen, T. (1989). *Self-organization and Associative Memory.* Springer–Verlag, Berlin, third edition.

Landelius, T. (1993). Behavior Representation by Growing a Learning Tree. Thesis No. 397, ISBN 91–7871–166–5.

Landelius, T. (1997). *Reinforcement Learning and Distributed Local Model Synthesis*. PhD thesis, Linköping University, Sweden, S–581 83 Linköping, Sweden. Dissertation No 469, ISBN 91–7871–892–9.

Landelius, T., Borga, M., and Knutsson, H. (1996). Reinforcement Learning Trees. Report LiTH-ISY-R-1828, Computer Vision Laboratory, S–581 83 Linköping, Sweden.

Landelius, T., Knutsson, H., and Borga, M. (1995). On-Line Singular Value Decomposition of Stochastic Process Covariances. Report LiTH-ISY-R-1762, Computer Vision Laboratory, S–581 83 Linköping, Sweden.

Lapointe, F. J. and Legendre, P. (1994). A classification of pure malt scotch whiskies. *Applied Statistics*, 43(1):237–257.

Lee, C. C. and Berenji, H. R. (1989). An intelligent controller based on approximate reasoning and reinforcement learning. *Proccedings on the IEEE Int. Symposium on Intelligent Control*, pages 200–205.

Li, P., Sun, J., and Yu, B. (1997). Direction finding using interpolated arrays in unknown noise fields. *Signal Processing*, 58:319–325.

Linsker, R. (1988). Self-organization in a perceptual network. *Coputer*, 21(3):105–117.

Linsker, R. (1989). How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Computation*, 1:402–411.

Ljung, L. (1987). *System Identification*. Prentice-Hall.

Loéve, M. (1963). *Probability Theory*. Van Nostrand, New York.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods*. Wiley, New York.

Marr, D. (1982). *Vision*. W. H. Freeman and Company, New York.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Mikaelian, G. and Held, R. (1964). Two types of adaptation to an optically–rotated visual field. *American Journal of Psychology*, 77:257–263.

Minsky, M. L. (1961). Steps towards artificial intelligence. In *Proceedings of the Institute of Radio Engineers*, volume 49, pages 8–30.

Minsky, M. L. (1963). *Computers and Thought*, chapter Steps Towards Artificial Intelligence, pages 406–450. McGraw–Hill. E. A. Feigenbaum and J. Feldman, Eds.

Minsky, M. L. and Papert, S. (1969). *Perceptrons*. M.I.T. Press, Cambridge, Mass.

Montanarella, L., Bassani, M. R., and Breas, O. (1995). Chemometric classification of some European wines using pyrolysis mass spectometry. *Rapid Communications in Mass Spectrometry*, 9(15):1589–1593.

Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–293.

Munro, P. (1987). A dual back-propagation scheme for scalar reward learning. In *Proceedings of the 9th Annual Conf. of the Cognitive Science Society*, pages 165–176, Seattle, WA.

Narendra, K. S. and Thathachar, M. A. L. (1974). Learning automata - a survey. *IEEE Trans. on Systems, Man, and Cybernetics*, 4(4):323–334.

Nordberg, K., Granlund, G., and Knutsson, H. (1994). Representation and Learning of Invariance. Report LiTH-ISY-I-1552, Computer Vision Laboratory, S–581 83 Linköping, Sweden.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *J. Math. Biology*, 15:267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68.

Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84.

Olds, J. and Milner, P. (1954). Positive reinforcement produced by electrical stimulation of septal area and other regions of rat brain. *J. comp. physiol. psychol.*, 47:419–427.

Pavlov, I. P. (1955). *Selected Works*. Foreign Languages Publishing House, Moscow.

Pearlmutter, B. A. and Hinton, G. E. (1986). G-maximization: An unsupervised learning procedure for discovering regularities. In *Neural Networks for Computing: American Institute of Physics Conference Proceedings*, volume 151, pages 333–338.

Pearson, K. (1896). Mathematical contributions to the theory of evolution–III. Regression, heridity and panmixia. *Philosophical Transaction of the Royal Society of London, Series A*, 187:253–318.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572.

Pollen, D. A. and Ronner, S. F. (1983). Visual cortical neurons as localized spatial frequency filters. *IEEE Trans. on Syst. Man Cybern.*, 13(5):907–915.

Riedmiller, M. and Braum, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA.

Ritter, H. (1991). Asymptotic level density for a class of vector quantization processes. *IEEE Transactions on Neural Networks*, 2:173–175.

Ritter, H., Martinetz, T., and Schulten, K. (1989). Topology conseving maps for learning visuomotor-coordination. *Neural Networks*, 2:159–168.

Ritter, H., Martinetz, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps*. Addison-Wesley.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, D.C.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Res. Develop.*, 3(3):210–229.

Sanger, T. D. (1988). Stereo disparity computation using gabor filters. *Biological Cybernetics*, 59:405–418.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, 12:459–473.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.

Shannon, C. E. (1948). A mahtematical theory of communication. *The Bell System Technical Journal*. Also in N. J. A. Sloane and A. D. Wyner (ed.) *Claude Elwood Shannon Collected Papers*, IEEE Press 1993.

Skinner, B. F. (1938). *The Behavior of Organisms: An Experimental Analysis*. Prentice–Hall, Englewood Cliffs, N.J.

Smith, R. E. and Goldberg, D. E. (1990). Reinforcement learning with classifier systems. *Proceedings. AI, Simulation and Planning in High Autonomy Systems*, 6:284–192.

Steinbuch, K. and Piske, U. A. W. (1963). Learning matrices and their applications. *IEEE Transactions on Electronic Computers*, 12:846–862.

Stewart, D. K. and Love, W. A. (1968). A general canonical correlation index. *Psychological Bulletin*, 70:160–163.

Stewart, G. W. (1976). A bibliographical tour of the large, sparse generalized eigenvalue problem. In Bunch, J. R. and Rose, D. J., editors, *Sparse Matrix Computations*, pages 113–130.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Tesauro, G. (1990). Neurogammon: a neural network backgammon playing program. In *IJCNN Proceedings III*, pages 33–39.

Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *Psychological Review*, 2(8). Monogr. Suppl.

Torres, L. and Kunt, M., editors (1996). *Video Coding: The Second Generation Approach*. Kluwer Academic Publishers.

van den Wollenberg, A. L. (1977). Redundancy analysis: An alternative for canonical correlation analysis. *Psychometrika*, 36:207–209.

van der Burg, E. (1988). *Nonlinear Canonical Correlation and Some Related Techniques*. DSWO Press.

van der Pol, B. (1946). The fundamental principles of frequency modulation. *Proceedings of the IEEE*, 93:153–158.

Watkins, C. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University.

Werbos, P. (1992). *Handbook of Intelligent Control*, chapter Approximate dynamic programming for real-time control and neural modelling. Van Nostrand Reinhold. D. A. White and D. A. Sofge, Eds.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavoral Sciences*. PhD thesis, Harvard University.

Werbos, P. J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3:179–189.

Westelius, C.-J. (1995). *Focus of Attention and Gaze Control for Robot Vision*. PhD thesis, Linköping University, Sweden, S–581 83 Linköping, Sweden. Dissertation No 379, ISBN 91–7871–530–X.

Whitehead, S. D. and Ballard, D. H. (1990a). Active perception and reinforcement learning. *Proceedings of the 7th Int. Conf. on Machine Learning*, pages 179–188.

Whitehead, S. D. and Ballard, D. H. (1990b). Learning to perceive and act. Technical report, Computer Science Department, University of Rochester.

Whitehead, S. D., Sutton, R. S., and Ballard, D. H. (1990). Advances in reinforcement learning and their implications for intelligent control. *Proceedings of the 5th IEEE Int. Symposium on Intelligent Control*, 2:1289–1297.

Williams, R. J. (1988). On the use of backpropagation in associative reinforcement learning. In *IEEE Int. Conf. on Neural Networks*, pages 263–270.

Wilson, R. and Knutsson, H. (1989). A multiresolution stereopsis algorithm based on the Gabor representation. In *3rd International Conference on Image Processing and Its Applications*, pages 19–22, Warwick, Great Britain. IEE. ISBN 0 85296382 3 ISSN 0537–9989.

Wold, S., Ruhe, A., Wold, H., and Dunn, W. J. (1984). The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5(3):735–743.

Zadeh, L. A. (1968). Fuzzy algorithms. *Information and Control*, 12:94–102.

Zadeh, L. A. (1988). Fuzzy logic. *Computer*, pages 83–93.

Author index