# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

# NEURAL NETWORK SENSOR FUSION:

# CREATION OF A VIRTUAL SENSOR FOR CLOUD-BASE HEIGHT ESTIMATION

By

HUGH JOSEPH CHRISTOPHER PASIKA

M.ENG, B.ENG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

# NEURAL NETWORK SENSOR FUSION

| | |
|---|---|
| Doctor of Philosophy (1999) | McMaster University |
| (Electrical Engineering) | Hamilton, Ontario |

| | |
|---|---|
| TITLE: | Neural Network Sensor Fusion: Creation of a Virtual Sensor for Cloud-Base Height Estimation |
| AUTHOR: | Hugh Joseph Christopher Pasika |
| | M.Eng (McMaster), B.Eng. (McMaster) |
| SUPERVISOR: | Prof. Simon Haykin |
| NUMBER OF PAGES: | xii, 147 |

# Dedication

To my father. He has been missed in a way that defies description. I can only aspire to keep his scientific mind, natural curiosity and belief in principles alive.

To my mother who has always championed my efforts, been a source of stability and instilled a desire to know myself in a way that brings out the mysticism of the world.

Together, they made me who I am today and for that I am thankful.

## Abstract

Sensor fusion has become a significant area of signal processing research that draws on a variety of tools. Its goals are many, however in this thesis, the creation of a virtual sensor is paramount. In particular, neural networks are used to simulate the output of a LIDAR (LASER RADAR) that measures cloud-base height. Eye-safe LIDAR is more accurate than the standard tool that would be used for such measurement; the ceilometer.

The desire is to make cloud-base height information available at a network of ground-based meteorological stations without actually installing LIDAR sensors. To accomplish this, fifty-seven sensors ranging from multispectral satellite information to standard atmospheric measurements such as temperature and humidity, are fused in what can only be termed as a very complex, nonlinear environment. The result is an accurate prediction of cloud-base height. Thus, a virtual sensor is created.

A total of four different learning algorithms were studied; two global and two local. In each case, the very best state-of-the-art learning algorithms have been selected. Local methods investigated are the regularized radial basis function network, and the support vector machine. Global methods include the standard backpropagation with momentum trained multilayer perceptron (used as a benchmark) and the multilayer perceptron trained via the Kalman filter algorithm. While accuracy is the primary concern, computational considerations potentially limit the application of several of the above techniques. Thus, in all cases care was taken to minimize computational cost. For example in the case of the support vector machine, a method of partitioning the problem in order to reduce memory requirements and make the optimization over a large data set feasible was employed and in the Kalman algorithm case, node-decoupling was used to dramatically reduce the number of operations required.

Overall, the methods produced somewhat equivalent mean squared errors indicating that the descriptive capacity of the data had been reached. However, the support vector machine was the clear winner in terms of computational complexity. As well, through its ability to determine its own dimensionality it is able to relate information about the physics of the problem back to the user.

This thesis, contributes to the literature on three fronts. First, it demonstrates the concept of creating of a virtual sensor via sensor fusion. Second, in the remote-sensing field where focus has typically been on pattern classification tasks, this thesis provides an in-depth look at the use of neural networks for tough regression problems. And lastly, it provides a useful tool for the meteorological community in creating the ability to add large-scale, cloud-field information to predictive models.

## Acknowledgments

First and foremost, I would like to thank Dr. Simon Haykin. Through his infectious enthusiasm for new ideas and scientific investigation, he has provided insights that have contributed significantly to this work.

I would also like to thank the members of my supervisory committee, Dr. Suzanna Becker and Dr. Anthony Vaz, for their helpful comments and advice.

Dr. Eugene Clothiaux has gone far beyond the call of duty in helping piece together the data set and define the problem. He too was a motivator with abundant energy.

Gint Puskorious and Dr. Edgar Osuna have my gratitude for providing key pieces of software and being most helpful in assisting me in firing them up.

The supporting cast is large. They have all been parts of fruitful discussions (and many unfruitful ones as well :) that benefited both my research and my state of mind. They are: Dr. Sadasavan Puthusserypady, Dr. Tim Davidson, Dr. Andrew Ukrainec, Dr. Tarun Bhattacharya, Vytas Kezys and Kaywan Afkhamie.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Sensor Fusion

Sensor fusion involves combining information from dissimilar sensors in order to form a decision or an estimate of the environment that is beyond the capability of any one of the individual sensors acting on its own. In recent years, sensor fusion has become a pervasive information processing concept that finds application in a diversity of fields ranging from military decision making (Hall and Llinas, 1997) to grading fruit (Ozer et al., 1994). Benefits that may be realized by this process include:

- Reduced sensor uncertainty
- Fault tolerance through redundancy
- Meta-feature discovery - these are higher-order features formed from combined pattern spaces of the different sensing modalities
- Creation of *virtual sensors* - using a less expensive, or an already established sensor network to perform the same task as another entirely different type of sensor

This thesis is concerned with the latter, the creation of a *virtual sensor* — one that does not physically exist. In particular, neural networks are used to fuse the information contained in fifty-seven remote-sensing variables and thereby simulate the output of a LIDAR (LASER RADAR) that is used by meteorologists to measure cloud-base height (CBH). This work is the first such application of neural networks to perform sensor fusion and predict cloud-base height and is significant on the following four fronts:

1. It explores in depth the use of neural networks for sensor fusion.
2. It is ground-breaking in the sense that the remote-sensing literature has typically focused only on neural networks as applied to pattern classification and not regression.
3. It uses a complex, highly-nonlinear, real-life data set spanning more than a year that is composed of over fifty different sensors, each with a distinct dynamic range.
4. It covers the current state-of-the-art in neural network technologies and studies their use for this unique regression problem.

The thesis can be considered to be composed of two parts. The first contains chapters one through three and discusses the fundamentals of both the problem and the algorithms employed. The second half, chapters four through seven, contains the details of the data set and the results obtained.

This chapter concludes with a discussion of the meteorologic problem and its importance to global climate by elucidating the complex role that clouds play in climate prediction. Moving on to engineering aspects of the problem, chapter two details the current literature on sensor fusion and describes some of the confounding mathematical issues involved in this particular problem. Chapter three elucidates the theory of the neural networks chosen to fuse the data. Here, a multilayer perceptron trained with the back-propagation (BP) algorithm serves as a benchmark and another global method, the node-decoupled extended Kalman filter (NDEKF) algorithm is outlined as are two local methods: the regularized

radial-basis function (RBF) network and the support vector machine (SVM). Moving on to the second half of the thesis, formulation of the data set is discussed in chapter four. This includes the origin of the data and preprocessing issues. Chapter five discusses the results achieved while the penultimate chapter probes the support vector machine results more fully and the overall suitability of the SVM for the cloud-base height estimation problem. The concluding chapter summarizes the findings of this thesis.

## 1.2 Meteorologic Problem

During the latter half of the twentieth century, levels of atmospheric carbon dioxide have increased from approximately 310 ppmv (parts per million volume) to 360 ppmv. Since carbon dioxide interacts much more strongly with terrestrial radiation than solar radiation, the initial effect of increasing levels of atmospheric carbon dioxide is a net heating of the earth system. Since the earth system is comprised of a complex set of interactions between its different components, for example, between the atmosphere and the oceans, the amount of heating that the earth undergoes for various loadings of atmospheric carbon dioxide is a subject of considerable debate (Houghton et al., 1996).

One component of the earth system that is a source of uncertainty in how climate will evolve with increasing levels of atmospheric carbon dioxide, is the cloud system. The properties of clouds are connected in an intricate way to the constituents of the atmosphere, as well as to the dynamic and thermodynamic fields of wind, temperature, air density, and moisture. Furthermore, the impact of clouds on the transfer of solar and terrestrial radiation through the atmosphere is perhaps just as complicated, as it is dependent both on the composition of the cloud particles and on the evolution of the horizontal and vertical distributions of cloud particles in time (Stephens and Webster, 1981; Slingo, 1990; Stephens

et al., 1990). Thus clouds are a key component in the earth's radiation budget and play a significant role in affecting large-scale climatology (Ramanathan et al., 1989). An ability to model their properties would allow us to better predict and adapt to climatic changes.

In most, if not all, computer simulations of weather and climate to date, the treatment of clouds has been relatively simplistic. Grid boxes in these models range in horizontal size from 1 km for numerical weather prediction models to 100 km for global climate models and in vertical size from a few tens of meters to a few tens of kilometers. Generally clouds are treated as a homogeneous slab throughout the grid box. This treatment of clouds is known to be deficient and only recently have more complicated treatments of clouds been attempted. For example, in a series of model simulations by Stubenrauch et al. (1997) and Liang and Wang (1997), the clouds were allowed to be nonhomogeneous within a grid box and to vary with certain statistical properties. Not surprisingly, these more complicated treatments of clouds had a significant impact on the evolution of the numerical models' dynamic and thermodynamic fields. The statistics of the cloud fields that these two model studies incorporated, however, were obtained from a single cloud study performed by Tian and Curry (1989) over the North Atlantic Ocean.

To begin to obtain a better observational data base on the statistics of clouds, the International Satellite Cloud Climatology Project (ISCCP) was initiated in the early 1980s (Schiffer and Rossow, 1983). As Rossow and Garder (1993) later demonstrated, satellites can be effectively used to map the horizontal locations of clouds; however, inferring the vertical distribution of clouds from conventional satellites has turned out to be a difficult problem (Baum et al., 1995). In an attempt to more accurately depict the vertical distribution of clouds in a few distinct climate regimes, the Department of Energy's Atmospheric Radiation Measurement (ARM) Program (Stokes and Schwartz, 1994) has deployed a suite of LIDARs developed by Spinhirne (1993) and millimeter-wave cloud radars developed by

Moran et al. (1998) at its sites. Ceilometers are the common instrument for measuring cloud-base height but unfortunately, their range is limited, being effective to an altitude of approximately four kilometers. LIDAR overcomes this limitation and can be further used to study the aerosol properties of clouds. These instruments currently represent the state-of-the-art in vertical cloud detection; however, their horizontal coverage is limited to those clouds that advect through their vertically oriented pencil beams.

As the observational field of cloud detection now stands, satellites can fairly accurately map the horizontal distribution of clouds globally and at least at a few sites, RADAR/LIDAR pairs can fairly accurately map the vertical distribution of clouds. However, the ability to map the full three-dimensional structure of clouds is limited. Satellites can only infer the cloud top heights directly from radiance measurements, while installations such as the ARM sites can only map the vertical distribution of clouds in a small vertical cone. Thus methods need to be developed that allow the inference of the three-dimensional cloud structure from a combined analysis of the satellite and surface data. This need for new observational methods is the motivation for this thesis.

One important cloud quantity that is difficult to estimate from satellite data is the cloud-base height since it is obscured by the overlying clouds. However, surface-based LIDARs make accurate measurements of cloud-base height, at least to within the spatial resolution of the instrument. Consideration of the types of data that are available motivates the simple question: Can measurements of the atmospheric radiation field and other meteorologic variables be used to estimate cloud-base height? In the approach undertaken, measurements of the radiation field are provided by the Geostationary Orbiting Environmental Satellite (GOES) and ARM surface based radiometers. Other standard meteorologic variables are added via ground-based sensors. Combined, these measurements are then used to predict the cloud-base height using sensor fusion approaches. The targets that are

being predicted are the cloud-base heights recorded by the ARM micropulse LIDAR. In this approach, absolutely no *a priori* knowledge about how these quantities are related is built in in order to more rigorously test the methodology itself. If this approach were to prove successful, it would provide cloud-base height estimates at a much larger number of locations. For example, the ARM site in Oklahoma has only one micropulse LIDAR, but it has approximately 25 surface radiometers and associated ground-based sensors spread across northern Oklahoma and southern Kansas.

### 1.2.1 Earth's Radiation Budget

All meteorological phenomena are driven by radiation from the sun. In the absence of clouds, the amount of direct sunlight hitting the earth (*insolation*) would be relatively constant and would vary only with solar angle. Clouds however, play a significant role in determining climate by modulating solar radiation and creating nonhomogeneity in the way radiation is distributed on the earth's surface. Winds and storms result from atmospheric circulation brought about by this energy imbalance on both regional and global scales. Similarly, the driving force of the sun's radiation leads to oceanic currents.

Clouds result from an extremely dynamic interplay of physical properties and as such, are one of the principal sources of uncertainty when predicting climatic changes. Thus their study is considered of high priority in climate research in order to accurately map their position and determine their characteristics. In order to understand how central their role is in determining climate, it is first necessary to investigate the earth's *radiation budget*.

The sun's surface radiates energy at approximately $6000°C$. By Planck's law (equation 1.1)[1] , the emissions peak at the wavelength of $0.5\mu m$ which is the center of the visible

[1]In equation 1.1 the variables represent the following quantities: $h$ - Planck's constant $(J \cdot s)$, $c$ - speed of light $(m/s)$, $\lambda$ - wavelength $(m)$, $k$ - Boltzmann's constant $(J/K)$, $T$ - temperature $(K)$.

spectrum as illustrated in figure 1.1.

$$I(\lambda, T) = \frac{2\pi hc^2}{\lambda^5(e^{hc/\lambda kT} - 1)} \tag{1.1}$$

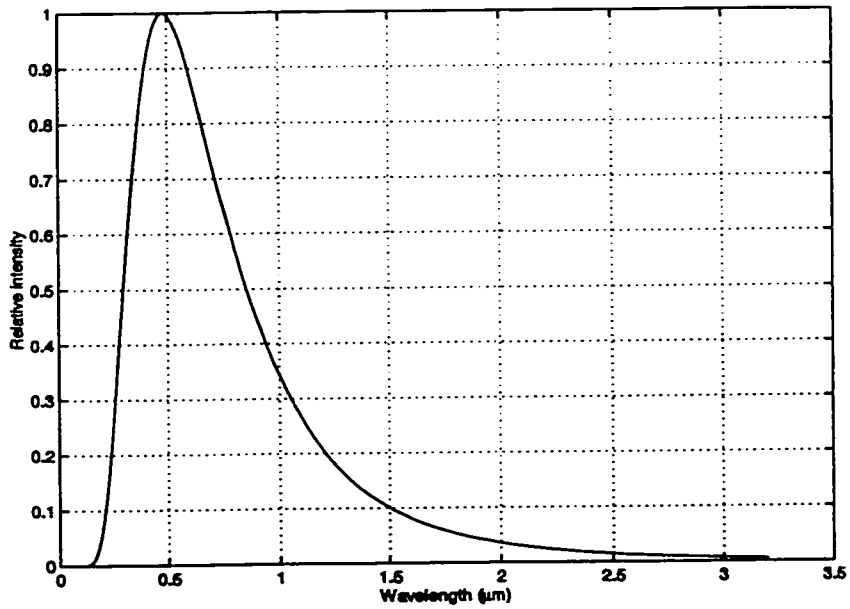



Figure 1.1: Wavelength of radiation emitted by the sun as determined by Planck's law.

These emissions are termed *shortwave radiation*. On its travel to earth, this radiation can be affected in three ways. First, it may be reflected. The term *albedo* is used to quantify the percentage of the radiation that is reflected. Table 1.1 cites the albedo of typical surfaces that incoming solar radiation may encounter. Secondly, radiation may be scattered by airborne particles whose size determine the wavelengths maximally affected. Finally, selective absorption by atmospheric gases can occur.

Figure 1.2(a) shows the typical scenario for earth-bound radiation. The figures

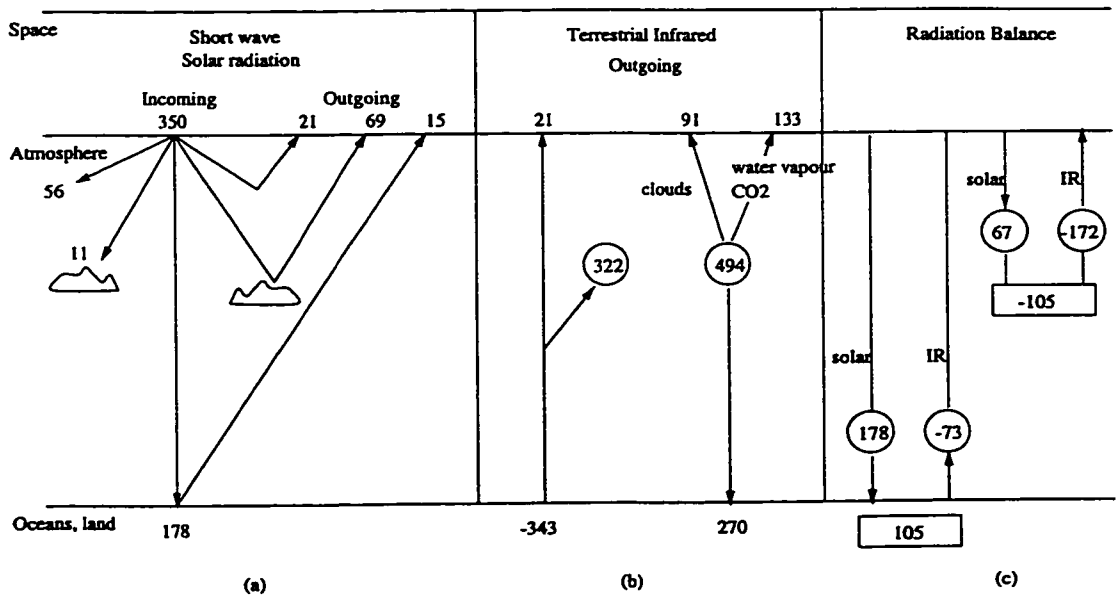



Figure 1.2: Radiation budget. All numbers are in $W/m^2$. (a) Incoming solar radiation (b) Infrared radiation (c) Radiation balance. From (Battan, 1983)

are given in $W/m^2$ but the discussion is simplified here by using percentages. A total of thirty percent of the solar radiation is reflected; 6% by the atmosphere, 20% by clouds and 4% by the earth. Nineteen percent is absorbed via water vapour and atmospheric gases, most notably carbon dioxide (16%), and 3% by clouds. The remaining fifty-one percent is absorbed by the earth.

When the earth absorbs solar radiation, it heats up and re-emits the radiation at a much lower wavelength. Planck's law states that these emissions peak at $10\mu m$. Whereas clouds and the atmosphere are largely transparent to the shortwave radiation, they more willingly absorb invisible, infrared radiation. Hence, the atmosphere is actually heated only indirectly from the sun and more precisely from the ground up. Figure 1.2(b) illustrates this effect. Of the 343 $W/m^2$ given up by the earth, 21 $W/m^2$ escapes to space with the

atmosphere and clouds capturing 322 $W/m^2$. These, in turn, release 494 $W/m^2$ with the earth absorbing 270 $W/m^2$. The combination of figures 1.2(a) and 1.2(b) yield figure 1.2(c). Here, all of the incoming and outgoing radiation fluxes balance and illustrate the earth's *radiation budget.*

Over the long term this equilibrium will be reached and will result in relatively constant climatic conditions. However, this balance does not exist on regional scales. Though some disturbances come about through changes in the earth's vegetation, the earth's ice and the presence of "greenhouse gases", by far the greatest factor leading to radiation imbalances is the presence of clouds. The process through which clouds exert their modulating effect on solar and terrestrial radiation is referred to as *cloud forcing.*

### 1.2.2 Cloud Forcing

Generally, a cloud has a higher albedo than the ground below (see table 1.1). Because of this, it will reflect a greater amount of radiation back to space and hence less energy is incident upon the earth's surface. This is termed *negative forcing* and tends to have a cooling effect. However, this is not the only modulating effect clouds have. The cloud will then absorb the infrared radiation emitted by the earth and if the negative forcing is disregarded, the trapped energy will have a heating effect (*positive forcing*). This may be compounded by the fact that the higher a cloud is, the colder is its top and the less longwave radiation it will emit to space.

Typically three cloud forcing scenarios exist and are governed by the types of clouds involved. Table 1.2 describes the four main classes of clouds: high, with cloud base above 6 km, middle layer clouds with cloud base between 3 and 6 km, low clouds whose bottoms exist below 3 km and clouds with vertical development which might have a base as low as 0 km and a top as high as 10 km.

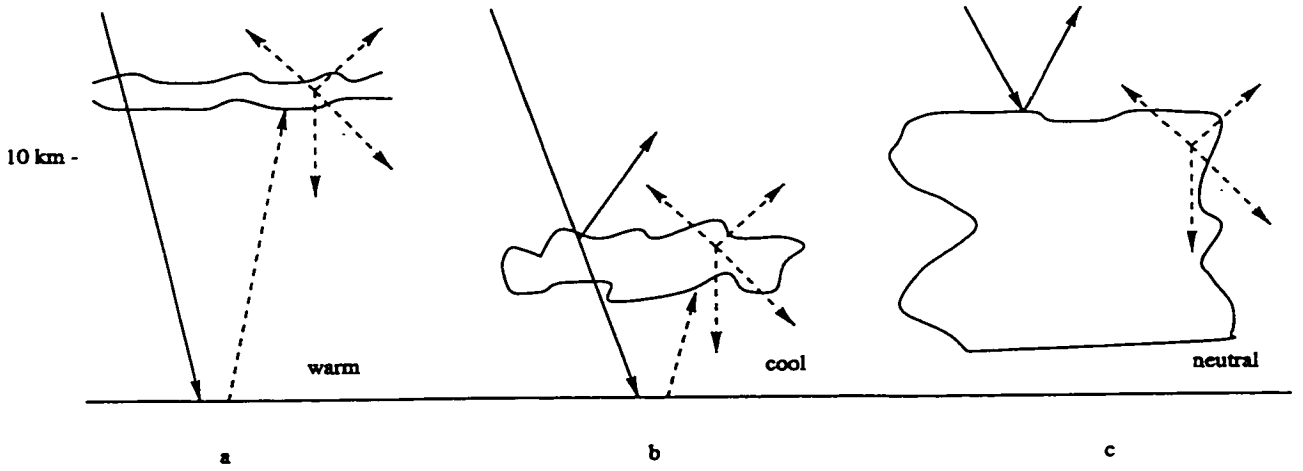



Figure 1.3: Example of cloud forcing. Solid line represents shortwave solar radiation and the broken line represents longwave emitted infrared radiation. (a) High clouds - net effect is atmospheric heating. (b) Low clouds - net effect is atmospheric cooling. (c) Deep convective clouds - net effect is neutral.

Illustrated in figure 1.3(a) is an overall positive forcing event. In this case high cirrus clouds are virtually transparent to incoming solar radiation and allow it to pass relatively unhindered. Upon heating the ground, the emitted longwave radiation is absorbed by the clouds and re-emitted, mostly towards the ground owing to the cold cloud tops. Very little is released into space and the result is a net warming effect. Figure 1.3(b) illustrates a net cooling or negative forcing event. The amount of solar radiation reflected by a cloud is referred to as *cloud albedo forcing*. Here, the billowy cloud results in a high cloud albedo forcing and due to the relatively warm top, it emits an almost equal portion of the longwave radiation to both space and the earth. Thus, the thin layer of air close to the earth's surface is warmed and offsets the cloud albedo forcing. However, since there is a lower amount of solar radiation striking the earth's surface, a net cooling effect predominates. Middle layer clouds will have an effect somewhere between the examples given in figures (a) and

(b). Finally, in figure 1.3(c), vertically developed clouds, that can be several kilometers in thickness, reflect an extremely large amount of shortwave radiation. Despite the fact that the cloud tops are cold and most of the re-emitted energy goes earthward, there is greatly reduced shortwave radiation to be absorbed by the clouds and the net effect is neutral; neither heating nor cooling occur.

## 1.3 Summary

In this chapter, the scientific importance of the cloud-base height estimation problem has been addressed and clouds are seen to have a major impact on climate. This is due to the fact that the models used in forecasting do not take into account the modulating effects of clouds on both solar and terrestrial radiation. Primarily, this is due to the lack of information regarding the three-dimensional distribution of clouds.

This thesis proposes the use of neural network techniques to fuse a suite of satellite and ground based sensors with an eye to providing cloud-base height information on a large scale thus enriching climate prediction.

| Surface | Albedo | Surface | Albedo |
|---|---|---|---|
| Grass | 20-25 | Sea ice | 30-40 |
| Deciduous forest | 15-20 | Fresh snow | 75-95 |
| Coniferous forest | 5-15 | Old snow | 40-70 |
| Dry earth | 15-25 | Glacier ice | 20-40 |
| Wet earth | 10 | Water (high sun) | 3-10 |
| Crops | 15-25 | Water (low sun) | 10-100 |
| Tundra | 15-20 | Thick cloud | 70-80 |
| Desert | 20-30 | Thin cloud | 35-50 |
| Asphalt | 5-10 | Earth and atmosphere | 35 |

Table 1.1: Typical values of albedo. Based on Battan (1983) and Lutgens (1982).

| Name | Cloud-Base Height (km) | Characteristics |
|---|---|---|
| High Clouds | | |
| Cirrus | > 6 | Thin with delicate fibrous ice crystals |
| Cirrostratus | > 6 | Thin, white rows or globules of ice crystals |
| Cirrocumulus | > 6 | Thin sheet of white ice crystals |
| Middle Clouds | | |
| Altostratus | 2-6 | Stratified veil |
| Altocumulus | 2-6 | White or gray globules like a sheep's back |
| Low Clouds | | |
| Stratocumulus | 0-2 | Soft, gray globular patches or rows |
| Stratus | 0-2 | Uniform layer similar to fog |
| Nimbostratus | 0-4 | Dark, produce heavy precipitation |
| Vertical Clouds | | |
| Cumulus | 0-3 | Billowy and dense, often with flat bases |
| Cumulonimus | 0-3 | Ominous and towering, may spread out at the top |

Table 1.2: The four basic cloud types (Lutgens, 1982).

# Chapter 2

# Sensor Fusion Literature

## 2.1 Sensor Fusion

Sensor fusion technology is employed in numerous applications. Seemingly, each application demands a different technique. Tracing sensor fusion's emergence as a research area over the last decade can be somewhat confusing as until recently, a common taxonomy did not exist. Recent reviews of the field have done well to bring order (Hall, 1992; Abidi, 1989; Dasarathy, 1990, 1997; Hall and Llinas, 1997) but still can only discuss the field in broad terms. This chapter reviews the field's literature in order to bring focus and understanding. Through this discussion, sensor fusion's pervasiveness, importance to the signal processing community and diversity are emphasized.

Two main motivations for sensor fusion research exist. On one hand, cognitive scientists and researchers in neurocomputation look at the problem from the standpoint of the human brain as being the ultimate model of a sensor fusion system. Their efforts attempt to tie research back to strong neurobiological principles. On the other hand, application-oriented engineers are less concerned about this aspect and rely on whatever tool works for

the task at hand. The two motivations are far from exclusive and are slowly converging towards a more principled approach to sensor fusion.

This chapter first discusses some of the neurobiological principles behind sensor fusion. As will later be discussed, the examples presented are more in the vein of *low level* (section 2.3.2) fusion systems. Higher cognitive centers are responsible for tying together the output from these systems to form concepts. (A discussion on this level of neuro-processing would be rather extensive and not appropriate here.) Following examples from biology, the chapter then shifts attention to engineering aspects of sensor fusion and discussion of the technology's benefits, the different manners in which sources of information interact and the taxonomy of the field.

After this, a generic sensor fusion platform is presented. This model reflects an amalgamation of several of the more prominent works in the field and possesses a distinct engineering flair. Its modules for extracting information and guiding sensors in their task are discussed using signal processing, machine learning, and control concepts. Discussion then turns to the more common methods of performing fusion and areas of application. As neural networks performing sensor fusion in a remote-sensing environment are of particular interest to this thesis, literature in this field is then discussed. In the next section, attention turns to mathematical considerations that drive the selection of methods in chapter 3. This involves differentiating between pattern classification and regression tasks where the difficulty of the latter is noted. As well, the problematic nature of ill-posed problems is discussed. Finally, the concepts introduced in chapters one and two, are used to more formally describe the problem that will be discussed for the remainder of this thesis by putting it in the context of the literature reviewed.

## 2.2 Neurobiological Principles

### 2.2.1 Biological Motivations for Sensor Fusion

It is easy to understand why many references extol the virtues of the human sensory processing system and hold it as the paradigm to which sensor fusion systems should be compared. A biological system is capable of not only fusing the inputs from our five senses but, it also manages to incorporate a vast wealth of information previously collected and stored. It is widely believed that the brain is constantly forming hypotheses against which incoming data are measured until either a fit occurs and the hypothesis is validated or a novelty is detected and a new class of information is declared. The role of the cortex in performing these two main tasks, correlation and novelty detection, is very effectively argued by Eggermont (1990). In essence, this is what a sensor fusion system does: it assimilates correlated information and forms an opinion on the state of the environment, perhaps using prior information.

The brain performs this function at many different levels. It fuses raw sensor information, and percolates this to progressively more complex levels resulting in extremely complicated concepts and thought processes. With so many levels of processing, at this stage, it is appropriate to introduce the ideas of *low-level* and *high-level* fusion. Generally, high-level fusion is considered to involve the integration of concepts and data that have undergone a fair degree of preprocessing. Thus features, rules, concepts, and goals are fused to formulate a plan of action. Low-level fusion, on the other hand, involves the merging of data that is at its most basic level and the process is generally located at the forefront of the entire sensory processing system. At such a location, generally little preprocessing exists.

Research on biological instantiations of fusion systems mostly involve fusing optical

data with some other sensing modality. This is not surprising as animals generally use vision as their primary sense and the addition of information from other sensors serves to augment the optical data. Examples along these lines include fusion of optical information with: electric field sensors in fish (Bullock, 1982), infrared heat sensors in the pit viper (Hartline et al., 1978) and auditory information in the owl (Knudsen et al., 1987) and cat (Meredith and Stein, 1986; Peck et al., 1993). In all cases, similar mechanisms and anatomical elements are used.

One of the most lucid and well-researched examples of sensor fusion is found in the fusing of visual and auditory information in the owl. Here, ears and eyes jointly focus attentional mechanisms in order to facilitate hunting. For the owl, this is of particular importance due its nocturnal nature. Features extracted via the interaural phase and intensity differences code a two-dimensional representation of the world allowing placement of an acoustic source in a two-dimensional frame. Thus an *acoustic retina* is formed. In the physiological literature, this type of representation is referred to as a *computational map* and it shares the principle of topographic organization with the neural network called the self-organizing map (SOM) (Kohonen, 1995). The acoustic information is then fed to the optic tectum where it is fused with data from the visual retina, thereby enhancing detection of targets in low-lighting conditions. Figure 2.1 illustrates this process and sensor fusion studies mimicking this behaviour of coregistering SOMs may be found in (Gelfand et al., 1992, 1993; Lau, 1993).

Similar studies have been performed in the cat (Meredith and Stein, 1986). In mammals, the equivalent of the optic tectum is the superior colliculus. It plays a similar role in orienting eyes to interesting stimuli. In these studies, close to 70% of the neurons in the superior colliculus have been determined to receive multisensory input. Furthermore, studies have shown that the resulting output from a multisensory neuron receiving input

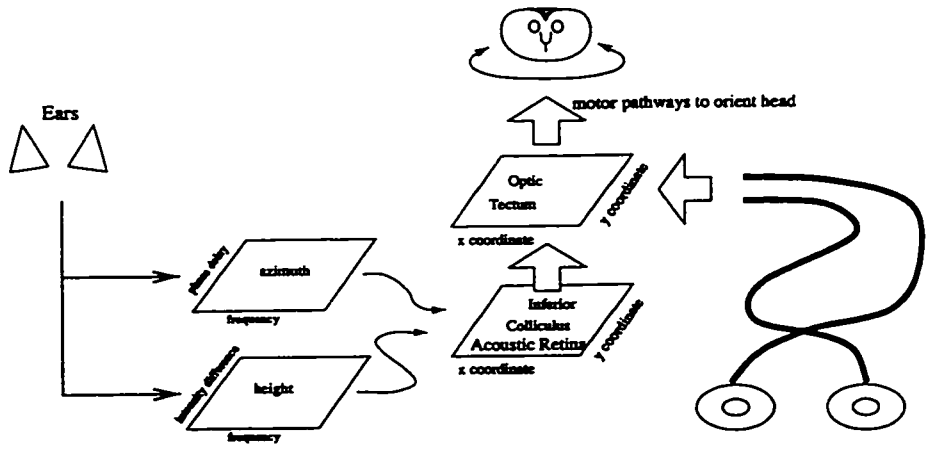



Figure 2.1: Sensor fusion in the barn owl.

from both sensors is much greater than if only one sensor is providing input, even if the single input is very strong. Thus more emphasis is placed on stimuli that trigger both sensors.

### 2.2.2 Psychophysical Studies: Marks' Unity of the Senses

In the realm between the neurophysiological studies of animal sensor fusion and an engineering approach to sensor fusion, lie psychophysical studies that summarize the results found in the animal kingdom's wetware. A concise synopsis of these studies has been formulated by Marks (1978) and is presented as the four doctrines of sensor fusion. These are:

> *Equivalent information nodes*: Frequently the same precept can be inferred from individual sensors each observing a different, nonintersecting set of attributes.
>
> *Analogous attributes and qualities*: All senses have some set of common stimulus properties, such as intensity, duration, size, form, and number.
>
> *Neural Correspondence*: There exists a special neural mechanism to integrate multisensory information, and for this to exist, the different senses use a common format, or knowledge representation.

*Unity of the Senses*: A combination of the other doctrines suggests that, for the different senses to be so similar, they should be interpreted as modalities of a general sense.

The fundamental point that emerges from this discussion is that sensor specific representations are converted to a common representation during fusion. Not mentioned in the doctrines but also discussed by Marks is the fact that a rather significant amount of sensor interaction often occurs. Context plays a key role in modulating perception whereby one sensor can recalibrate others and form feedback control loops. However, while a general framework of sensor fusion emerges from the above points, this is as far as psychophysical studies can take us as they do little to explain the mechanisms of fusion or knowledge representation.

## 2.3 Engineering Aspects of Sensor Fusion

### 2.3.1 Benefits of Sensor Fusion

The benefits brought about by sensor fusion form a lengthy list. It can provide robust operational performance by allowing one sensor to contribute information while others are unavailable, thereby creating fault tolerance via redundancy. It may permit a larger picture of the environment to be formed when one sensor can look where another cannot, it can provide increased confidence, reduced ambiguity and improved system performance. Even if not permitting a final answer, its use may reduce the set of hypotheses about an event. However, one of the most novel applications is to combine old technologies in new ways to create "smart" systems. Regardless of the function of a fusion system, it can be thought of as satisfying one of the four following goals.

*Virtual Sensors*: Used where it is desirable to replace an expensive or nonexistent sensor with several others that are more widely/easily used (Lucas, 1996).

*Redundancy*: Fault tolerant sensor networks can be created that permit constant operation. Systems have been designed that perform better than the best sensor and have been demonstrated to produce a significant performance gain when sensor reliability is anywhere between 0.5 and 0.95 and a moderate number of sensors is used (Nahin and Pokoski, 1980).

*Sensor Degradation*: In a two-sensor system, one sensor may perform adequately in a noise-free environment, however, the other sensor performs inadequately. As sensor one's SNR decreases, additional cues from sensor two allow the system to maintain performance. This is the principle behind audio/visual speech fusion (Yuhas et al., 1990).

*New Features and Feature Discovery*: Used when no existing single sensor can produce the feature of interest as in (Lucas, 1996) or in table 4.2.

With these four goals in mind, it is important to understand the various ways in which information can interact. Information from different streams can be thought of as being combined in four basic ways (Iyengar et al., 1995). (Careful note should be made of the use of the term integration here. While it has mathematical overtones, it is used on par with the term fusion.)

*Competitive Integration*: Fusion of redundant information for devising a fault tolerant system using replicated sensor readings. This is often used to combat the effects of noise.

*Complementary Integration*: Deals with partial or overlapping information and combining it to form a global perspective.

*Cooperative Integration*: Similar to complementary integration but data does not overlap, thus individual sensor data is orthogonal. An example would be factorial distributions (Deco and Obradovic, 1996).

*Independent Integration*: The combining of completely (seemingly?) unrelated data.

### 2.3.2 Three Levels of Fusion

By far, a significant portion of the sensor fusion literature stems from military applications. These generally involve target tracking, threat recognition, and risk assessment. Consequently, the military has driven advances in the field of sensor fusion. Typically, in military scenarios, three levels of fusion are considered to exist. These are data fusion (level 1), situation assessment (level 2), and threat analysis (level 3) (Hall, 1990). Each level reflects an increased cognitive ability and deals with more refined information. Though application of fusion technologies are widespread and include not only military applications, this trilevel definition exists throughout the literature and is expressed as low/mid/high level fusion, data/information/decision (Dasarathy, 1990; Hall, 1990) and sensor/tactical/strategic (dit Neuville, 1996). No matter the specific name given to a level of fusion, in each case, the same concepts apply as the following descriptions illustrate.

> *Low-level/Data Fusion*: Low-level fusion deals with raw data: often directly from the sensors as signals or images. At this stage, the sensors often possess very different characteristics (except in the case of a redundant sensor array). Sensors may be discrete or continuous, possess full or partial access to the environment, vary in dimensionality, sensitivity, tolerance, precision, range and variance.
>
> *Information Fusion*: Information fusion deals with higher-order, more robust features such as categories, properties, primitives (ie. edges or textures in images) or state vectors that are the output of low-level fusion.
>
> *Decision Fusion*: At this stage, data that have flowed through the system are merged with rules, experience or higher-level concepts to decide upon a course of action.

Figure 2.2 demonstrates the relationships between these levels and shows how in a feedforward manner, the heart of a sensor fusion system is constructed. It is important, however, to realize that many variants may exist and data from different levels may interact

as discussed by Dasarathy (1997), here data may be combined with features or feature with decisions as illustrated in figure 2.3.

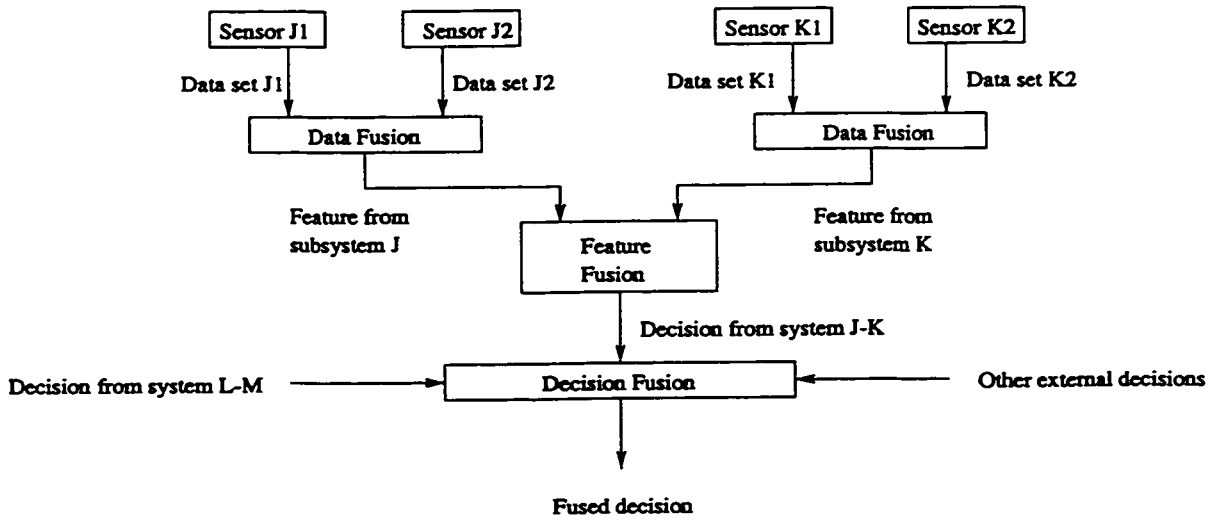



Figure 2.2: Feedforward sensor fusion system.

One final architectural consideration is where in the flow of information the fusion occurs. This allows a further classification. *Centralized fusion* acts on raw data fusing it almost immediately. *Autonomous* systems generate a state vector for each sensor which is subsequently combined. Finally, *hybrid* fusion systems utilize a combination of the two (Heirstrand, 1983). These processes are illustrated in figure 2.4.

### 2.3.3 General Framework for a Sensor Fusion System

Earlier reference was made to the human sensory processing system and the two motivations of sensor fusion research; neurobiological and engineering. Figure 2.5 represents a marriage of these two motivations. In it, a general framework for a sensor fusion system is presented which is an amalgamation of systems presented by Kokar and Tomasik (1994) and Luo and

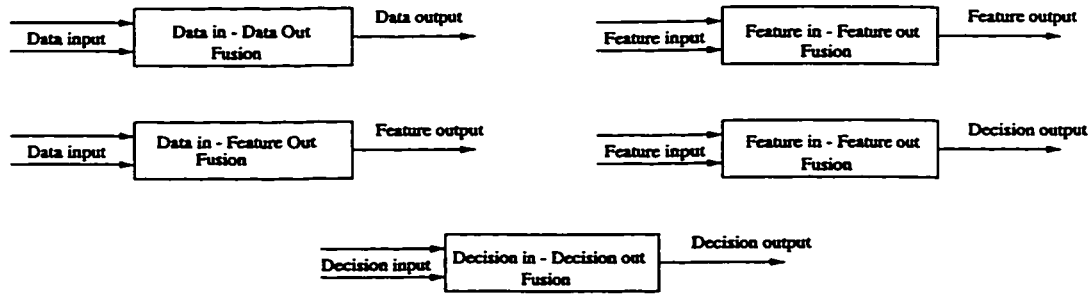



Figure 2.3: Combining information from different levels of fusion (Dasarathy, 1997).

Gonzalez (1992). Here a great deal of emphasis is placed on the supporting structure. Out of the entire diagram, the concepts previously discussed are found only in the block labeled *fusion engine*. The model encompasses all levels of fusion and to some degree, intelligence is built into every module. There are four main sections or types of modules. These are enclosed in the dotted boxes, and are the *ancillary support* algorithms, *guiding/controlling* pathways, *knowledge incorporation* modules and the *fusion engine* itself. Each area is discussed in turn and owing to its importance to this thesis, the fusion engine is discussed in a numbered section of its own.

## Ancillary Support Algorithms

Ancillary support algorithms play a significant role and may require more processing power than the core of the sensor fusion system. Figure 2.6, taken from Hall (1992), outlines the broad categories into which the support algorithms fall. In it there are five basic categories, of which two are of main concern.

*Data alignment*, often referred to as registration, is one of the most important techniques, especially for the fusion of images. Examples occur during the fusion of Computed

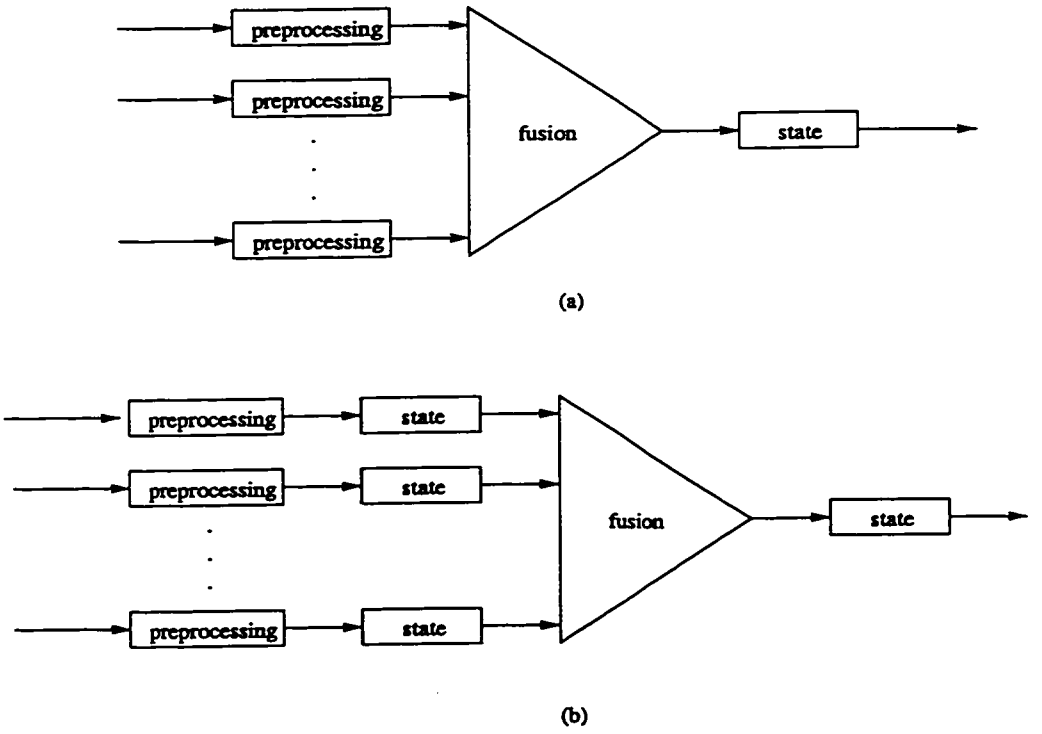



Figure 2.4: Fusion architectures. (a) Centralized fusion. (b) Autonomous fusion.

Tomography images and Magnetic Resonance Imaging images (Hamadeh et al., 1994) or in the case of surgical mapping where three-dimensional registration is required (Bhatia, 1994).

*Preprocessing* techniques include just about any method of signal conditioning or feature extraction. As an example, wavelet decomposition is used by Huntsberger and Jawerth (1993) for image fusion. Here the images are decomposed and subsequent fusion is effected in the wavelet coefficient space via logic operators. These operators are used as opposed to differences, products and probabilistic approaches as these would alter the image content significantly. From this example, it is clear that the types of features available will

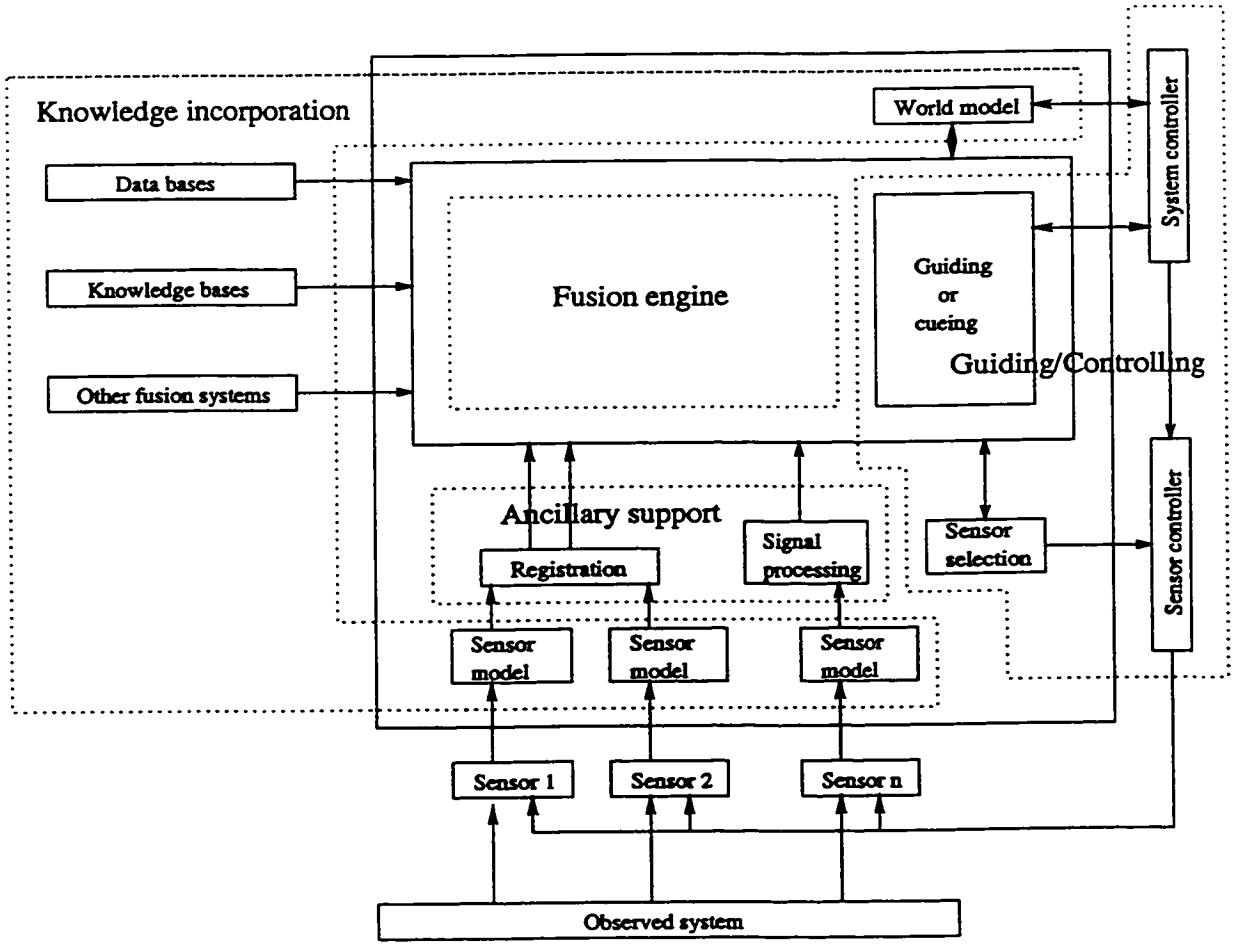



Figure 2.5: A general model of a sensor fusion system. Combined from works by Kokar and Tomasik (1994) and Luo and Gonzalez (1992).

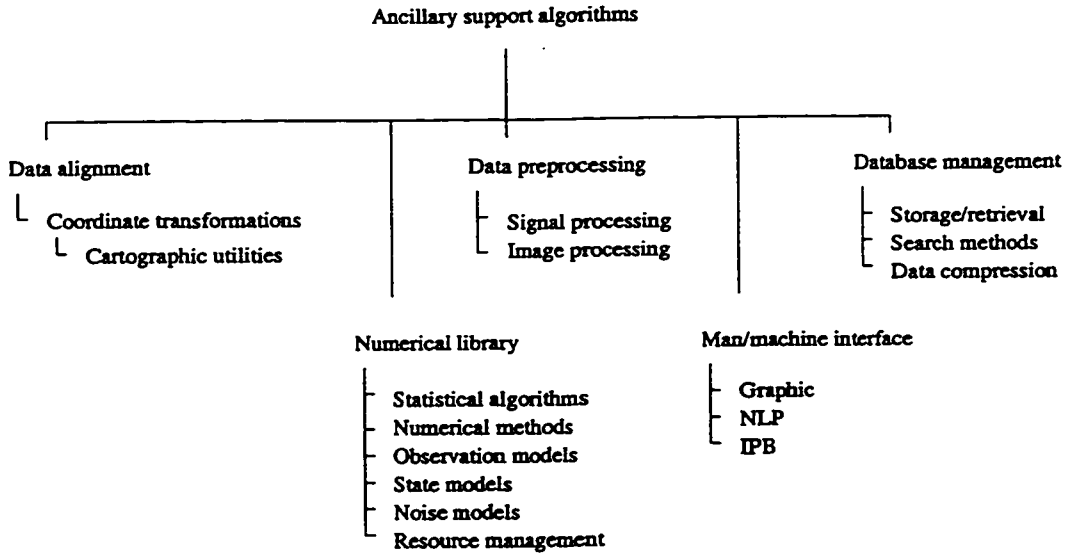



Figure 2.6: Taxonomy of ancillary support algorithms (Hall, 1992).

drive the choice of fusion algorithm. The other three groups outlined in figure 2.6, *database management*, *numerical libraries*, and *man/machine interfaces* play more of a support role as opposed to being information handlers.

## Guiding/Controlling Modules

Modules in figure 2.5 such as *sensor selector*, *guiding/cueing*, *system controller* and *sensor controller* are essential elements in what are referred to as strongly coupled fusion algorithms. In strongly coupled fusion, the operation of one sensory module is affected by the output of another sensory module so that the outputs are no longer independent (Clark and Yuille, 1990). Thus a method of feedback to control the interaction is required. If this were not the case, the system would be said to be weakly coupled where the sensory modules are considered to operate autonomously, independent of the environment and each other.

### Knowledge Incorporation

The *sensor models* provide a measure of a sensor's output quality. This is key in the sensor selection process for redundant systems. Here the system must be able to determine sensor errors and take appropriate action. Methods of exception handling are discussed by Chavez and Murphy (1993) with the goal of performing graceful rather than catastrophic degradation of performance. At a higher level of processing, feature selection can also be monitored. Battiti (1994) has proposed methods based on information theory.

As well as monitoring the quality of a sensor, some algorithms have been proposed to adapt sensor characteristics on the fly. For example, Klassner et al. (1993) modify the short time Fourier transform's parameters (window length, sampling rate, etc.) to enhance performance. Another example occurs where sensing strategies adapt to lighting conditions. Such feedback loops require information in order to direct adaptation. This information is provided by the *world model*. Here, information is stored regarding either the state of the operating environment *a priori*, or recently acquired information. Finally, the other parts of the system are the external inputs themselves. These encompass data bases, knowledge bases and the outputs from any other fusion systems, etcetera.

## 2.3.4 Fusion Engine Methodologies

Inside the fusion engine lies the actual algorithm that combines the information. Hall has broken fusion algorithms into three families (Hall, 1992). These are fusion based on: *physical models*, *feature-based inference techniques*, and *cognitive-based models*. These classifications and their subdivisions are illustrated in figure 2.7

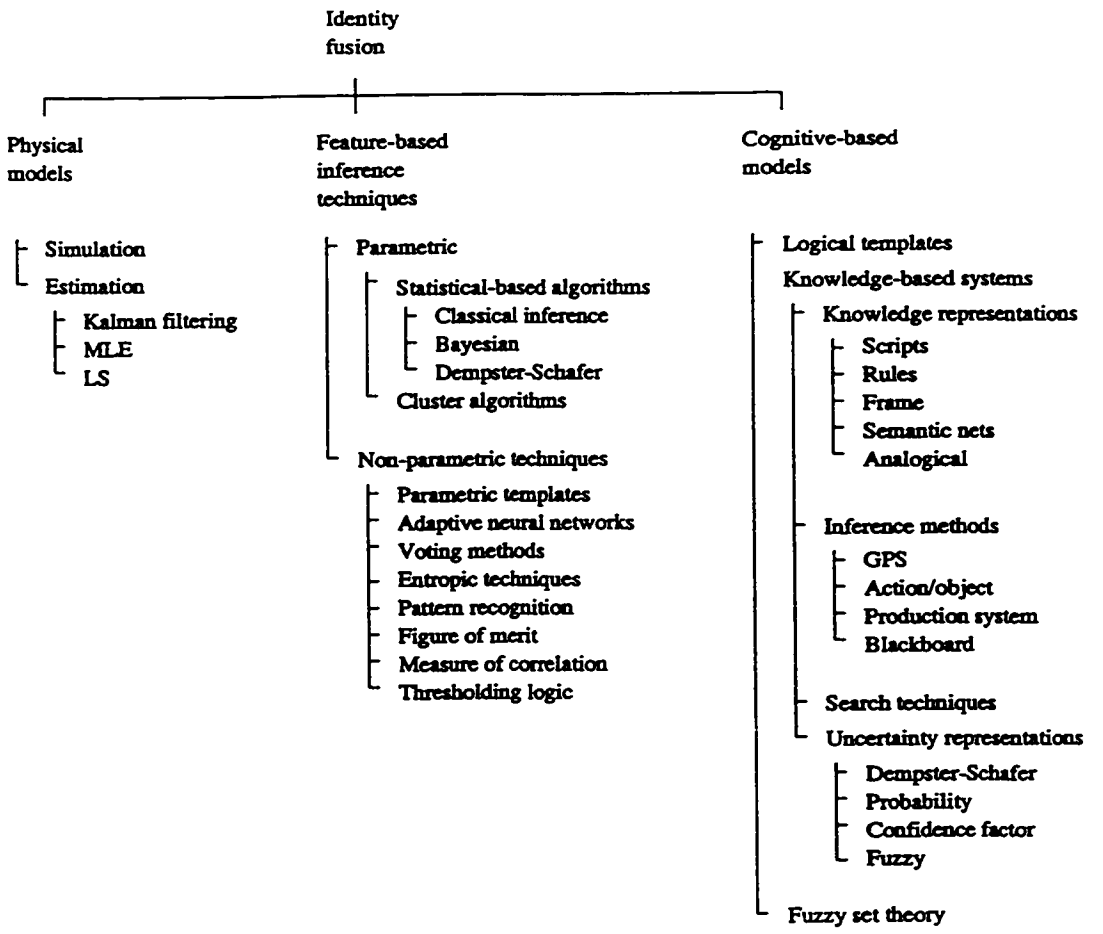



Figure 2.7: Taxonomy of fusion algorithms (Hall, 1992).

## Physical Modeling Methods

Maximum likelihood estimation and least-squares techniques are used for parameter estimation where the parameter of interest is not a random variable.

Another technique of this family that finds wide application is the Kalman filter which is used particularly in robot kinematics and tracking algorithms (Proceedings MFI, 1994, 1996). The Kalman filter forms the heart of one of the leading neural network algorithms and is discussed in chapter three.

## Feature-Based Methods: Parametric

Statistically based algorithms which employ the concept of evidence supporting a hypothesis have been employed for fusion. The underlying assumption is that the fused parameter of interest is a random variable with a known probability density function. The most basic method uses Bayesian theory (Trees, 1968). Given a piece of evidence ($E$) and a hypothesis ($H$), Bayes rule simply states,

$$p(H_i|E) = \frac{p(E|H_i)p(H_i)}{\sum_i p(E|H_i)p(H_i)} \tag{2.1}$$

where the components of the equation bear the following meanings,

| | |
|---|---|
| $p(H_i\|E)$ | *a posteriori* probability of the hypothesis $H_i$ being true, given the evidence $E$ |
| $p(E\|H_i)$ | probability of witnessing evidence, given that $H_i$ is true |
| $p(H_i)$ | *a priori* probability of hypothesis $H_i$ occurring |

It is a satisfying method as the likelihood of a hypothesis is updated given its previous likelihood estimate and new evidence. Once the $p(H_i|E)$ are calculated, decision

logic declares the hypothesis that has the greatest joint probability given all of the evidence to be true. The method also has an intuitive appeal as it bears similarity to scientific experimentation and the process of "building a case". Each observation is treated as an experiment that serves to confirm or deny a hypothesis and thus modifies its probability of being true.

However, certain concerns do exist with Bayesian fusion, the most notable being that the method hinges on the requirement of the *a priori* probabilities $p(H_i)$ of the hypotheses. Also, all hypotheses must be nonoverlapping; for example, the situation where $H_1$ (A is an apple), $H_2$ (A is an orange), $H_3$ (A is a grapefruit) and $H_4$ (A is either an orange or a grapefruit) cannot exist. This is a constraint that is relaxed in the Dempster-Shafer method.

Dempster-Schafer theory overcomes the non-overlapping hypothesis problem and does not require an exhaustive list of hypotheses but more importantly it incorporates uncertainty into the fusion process. Here, the motivation is to assign "belief" in a proposition in a manner similar to how humans ascribe degrees of belief. When fusion of the evidence occurs, a new degree of belief is formed.

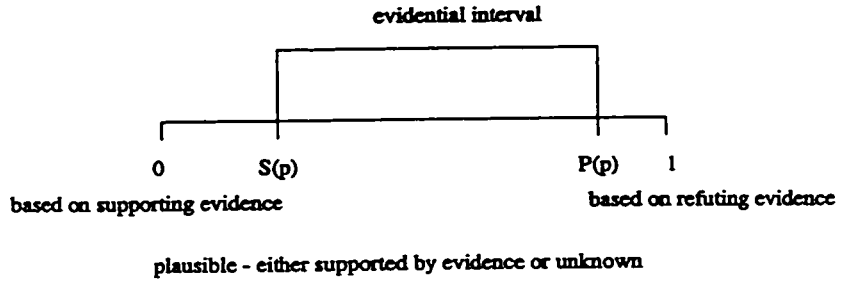



Figure 2.8: Defining the evidential interval.

Development proceeds by assigning an evidential interval to each piece of evidence (feature vector, sensor reading, etc.). This interval lies within the probability interval $(0, 1)$.

The lower bound, termed the *support bound* ($S(p)$) expresses the extent to which the piece of evidence supports the hypothesis while the upper bound, the *plausibility bound* ($P(p)$), relates the degree to which the evidence fails to refute the hypothesis. This is illustrated in figure 2.8. Thus if the evidence expresses total ignorance the evidential interval would be $[0, 1]$. Pieces of evidence are then combined using Dempster's rules and the $S(p)$ and $P(p)$ values may be computed for any hypothesis or proposition (ie. over-lapping hypothesis). Decision logic then chooses the hypothesis or proposition that is maximally supported by joint evidence (high $S(p)$) and is minimally refuted by joint evidence (high $P(p)$). Using this method, a distinction between uncertainty and ignorance is possible. More in depth discussion of this technique may be found in works by Dempster (1968) and Shafer (1976). However, similar to the Bayesian case, the probability masses which give rise to the support and plausibility require definition. These, as in the Bayesian formulation, can be determined via models, statistical databases or may be assigned subjectively.

### Feature-based Methods: Non-parametric

A non-parametric approach involves the estimation of a parameter (regression) or the estimation of decision boundaries (pattern classification) without the requirement of a probability distribution model. Thus, no prior assumptions are made of the data's structure and any results are based on the data alone. In a sense, the data set is said to "speak for itself".

Of particular interest in the non-parametric family of algorithms is the neural network. This type of learning machine is formed by a collection of processing nodes which are highly interconnected in a manner similar to neurons in the brain. The processing nodes (neurons) incorporate nonlinear functions and thereby give rise to complex decision boundaries for pattern classification, or in case of regression problems, to the discovery of highly nonlinear relationships. Figure 2.9 illustrates the typical architecture of a neural network

as embodied by the feedforward multilayer perceptron. It is composed of one input node for each input variable, two hidden layers and a final output layer. Generally, each neuron is connected to all neurons in the previous layer by a series of weights and a bias is also applied via a unity input to each neuron. These are indicated by the square nodes in the figure. Weight $w_{ji}$ connects neuron $j$ to neuron $i$. Thus, the output of a neuron ($y_j$) is determined by the following equation.

$$y_j = \phi(\sum_{i=1}^{p} w_{ji}x_i) \tag{2.2}$$

where $x_i$ are the inputs leading into neuron $j$ multiplied by weight $w_{ji}$. The function $\phi(\cdot)$ is a nonlinear function of the neuron; a common choice for which is the sigmoid, $\phi(\nu) = 1/(1+exp(-\nu))$. Network training is initiated by presenting a pattern vector to the input nodes. Initially, as the multiplicative weights possess small random values, the output value will bear little correspondence to the desired response. However, by determining the direction in which to adjust the weights so that the output will more closely match the desired response, a small adaption of the weights is made. After all patterns have been presented many times over, the network learns the mapping that will produce a desired response for a given input. More details are given in section 3.3

A major distinction between training algorithms is whether they act in a local or a global mode. In chapter 3, two methods of each type are outlined. For global methods, the back-propagation trained multilayer perceptron and the extended Kalman filter trained multilayer perceptron are discussed. These methods construct a global mapping of the input–output relationship. Because of this global nature, their generalization abilities in regions where no training data exist are quite good. However, an argument is made that

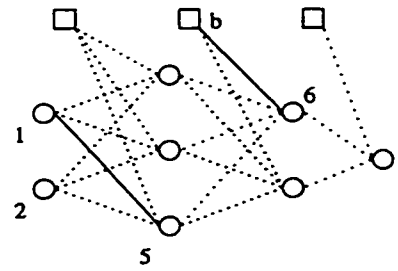



Figure 2.9: Feedforward multilayer perceptron. The leftmost solid line represents weight $w_{5,1}$ while the other solid line represents the weight $w_{6,b}$ from the bias unit to neuron 6.

global methods do not respond well to isolated data points and it is more difficult for them to adapt to differing input densities.

Local methods such as the radial-basis function network and the support vector machine operating in RBF mode, on the other hand, create local mappings based only on the stored points. Generalization outside of these areas may be difficult to achieve and in order to span the entire input–output space, a larger training set is generally required. Local methods, though, have the advantage of more rapid training times and a reduced sensitivity to order of presentation during training (Haykin, 1999).

## Cognitive-Based Methods

This family of algorithms tends to be driven by the fact that human analysts have an ability to compensate for incomplete and sometimes inaccurate data through the use of domain specific knowledge. It is proposed that two types of knowledge exist, declarative and procedural. Unfortunately, most experts "can do" but cannot explain "how to" and the information is fully internalized as rules of thumb thus systems are difficult to implement (Sawaragi et al., 1994). Typically, this strategy is employed during high-level fusion using expert systems.

### 2.3.5 Areas of Applications

One of the classic examples of sensor fusion is that of the *cocktail party problem* where the visual information given by lip movements is fused with the resulting auditory signal in order to enhance speech recognition in a noisy environment. Yuhas et al. (1990) used an approach involving a neural network that performs a mapping from the shape of the lips to the voice's spectral domain. This was then averaged with the spectrum of the speech with the intention that the resulting spectrum would be more easily recognized by a subsequent pattern recognition system. Enhanced performance was only noted in extremely noisy environments and performance was actually lower than for recognition based on the acoustic signal alone when operating in a noise-free environment. Since this original work, more sophisticated approaches have been applied, many of which focus more closely on preprocessing issues. A thorough review of research in the area is made by Chen and Rao (1998).

The military is a strong proponent of sensor fusion research. Given technology's constant forward progress, military strategists are faced with the need for better decisions to be made within the confines of decreased reaction time. As well, there is an ever present desire to improve accuracy and thus increase the lethality of weapons. Furthermore, reduced observability via stealth technology renders standard single sensor platforms inadequate. Sensor fusion is also used where there is increased personal risk and autonomous systems capable of processing in a human framework are required. Basically, the above may be summarized as an increase in the complexity of threats which then require increasingly sophisticated methods of detection. Another desire of sensor fusion systems is to perform *knowledge distillation*. For example, a typical air warfare exercise requires up to fifty decisions to be made per minute. These decisions are based on anywhere from fifty thousand to

one hundred thousand observations from over one hundred and fifty separate sensor platforms that are looking at approximately one thousand hostile targets scattered in a thirty kilometer sided cube (Luo and Gonzalez, 1992): a lot of information to sift through indeed!

Space is another area of application. At issue here is the lack of atmosphere which causes violations of the assumptions that many terrestrial visual recognition techniques are based on. Deeper shadows, apparently missing edges and intense reflections cause standard techniques to fail and thus require additional information to be brought in by other sensors.

The above are far from the only areas that benefit from sensor fusion research. However, they do give a feel for the diversity of applications which benefit from sensor fusion technology. In between lie applications to medical imaging, robotics, process control systems, manufacturing, surveillance, medical imaging and the topic of the next section remote sensing (Hall, 1990).

### 2.3.6 Neural Networks in Remote Sensing for Cloud Studies

For a review of neural networks in sensor fusion, the reader is directed to Atkinson and Tatnall (1997). Here it becomes apparent that by far the vast majority of neural networks in remote sensing have been used for classification problems. Cloud classification (Welch, 1992; Bankert, 1994) and land cover studies (Hepner et al., 1990; Downey, 1992) have shown neural networks to outperform statistical classifiers in both accuracy and execution speed on complex feature spaces (Benediktsson et al., 1993, 1990). On the regression side, they have also been used to estimate surface radiative fluxes from satellite brightness temperatures (Schweiger and Key, 1997). As well, they have been used to allow the embedding of *a priori* information such as realistic physical constraints in problems (Foody, 1995).

Common perceptions of neural networks which are often cited in the literature are their lengthy training times and their "black box" nature. However, as will be seen in this

thesis, both of these problems have been overcome with new developments in the neural network field. As well, the benefit of being able to deal in a distribution-free environment with training targets that may be represented at multiple points in feature space, coupled with the ability to rely on data rather than models easily overcomes former criticisms and demonstrate neural networks to be excellent tools for this field.

## 2.4 Mathematical Considerations

At the heart of this thesis is a very difficult learning problem. Not only are relationships between parameters controlling the climate highly nonlinear, but the type of problem being tackled, regression, is significantly more difficult than pattern classification. As well, the problem is ill-posed which further complicates the situation. The next two sections discuss these concepts.

### 2.4.1 Pattern Classification vs. Regression

The point has been made that sensor fusion pattern classification problems dominate over regression problems in the literature. Though both types of problems bear similarity in that they are mappings from one domain to another, regression problems are inherently more difficult. The intuitive argument is intellectually satisfying and proceeds from the discussion below, further details of which may be found in (Devroye et al., 1996).

Suppose that learning machine $\mathcal{D}$ has been trained on a sequence of data. Thus, given a set of training data vectors which form the set $\mathbf{X}$ and the set of associated targets $\mathbf{y}$ that the individual pattern vectors ($\mathbf{x}_n \in \mathbf{X}$) map into, the trained machine can be represented as $\mathcal{D}((\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n))$. Once trained, $\mathcal{D}$ produces estimates of the *a posteriori* probabilities $\hat{\eta}_n(\mathbf{x}_n)$ where the true probabilities are denoted $\eta_n$. In the case of regression,

this is the estimate of the corresponding $y_n$. In the case of pattern classification, these $\hat{\eta}_n$ are thresholded via a mapping function $g(\cdot)$ (sometimes referred to as a discriminant function) to determine the $y_n$ which are the class memberships. For a binary classification problem, the mapping function which is alternately referred to as the classification rule could look something like equation 2.3.

$$g(\mathbf{x}_n) = \begin{cases} 0 & \text{if } \hat{\eta}_n(\mathbf{x}_n) \leq 1/2 \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

Theoretically, the optimal $g(\cdot)$ is Bayes rule and other classification rules may only approach its performance. Classification error is defined as:

$$L = L(g) = P(g(\mathbf{X}, \mathcal{D}) \neq \mathbf{y}|\mathcal{D}) \tag{2.4}$$

where the Bayes classification error is annotated $L^*$. A classifier is said to be consistent if $L$ gets arbitrarily close to $L^*$ with large probability.

$$\mathbf{E}[L] = P(g(\mathbf{X}, \mathcal{D}) \neq \mathbf{y}) \rightarrow L^* \quad \text{as n} \quad \rightarrow \quad \infty \tag{2.5}$$

or alternatively stated,

$$\lim_{n \rightarrow \infty} P(L - L^* > \epsilon) = 0 \tag{2.6}$$

Now, considering the regression problem, the goal is to determine the $\hat{\eta}_n$ as accu-

rately as possible. If accurate approximation of $\eta_n$ is possible, the classification problem is trivial and it can be shown that the regression error may be bounded as follows:

$$\mathbf{E}[L] - L^* \leq 2\sqrt{\mathbf{E}\{(\hat{\eta}(\mathbf{X}) - \eta(\mathbf{X}))^2\}} \tag{2.7}$$

thus implying that $L_2$ consistent estimation of the regression function will result in consistent classification.

If a ratio is formed as in equation 2.8 between the classification error and the regression error, and the limit taken, it will converge to zero as the numerator more quickly approaches zero than does the denominator.

$$\lim_{n\to\infty} \frac{\mathbf{E}L_n - L^*}{\sqrt{\mathbf{E}[(\eta_n(X) - \eta(X))^2]}} \tag{2.8}$$

Thus, given a finite sample size, classification can be performed more accurately than can regression. This result is intuitively satisfying since the required output of regression estimation is an exact value whereas classification only requires the value to fall within a range.

### 2.4.2 Ill-Posed problems

The problem of mapping the sensor outputs into their corresponding cloud-base height is confounded by the fact that it is fundamentally ill-posed. Following the properties laid out by Haykin (1999), the reconstruction of a mapping $F : \mathbf{x} \to \mathbf{y}$ where $\mathbf{x}$ has domain $\mathcal{X}$ and $\mathbf{y}$ has range $\mathcal{Y}$ is said to be *well-posed* if the following three criteria are met.

*Existence*: For every input vector $\mathbf{x} \in \mathcal{X}$, there exists a $\mathbf{y} = F(\mathbf{x})$ where $\mathbf{y} \in \mathcal{Y}$.

*Uniqueness*: For a pair of vectors $\mathbf{x}, \mathbf{t} \in \mathcal{X}$, $F(\mathbf{t}) = F(\mathbf{x})$ if, and only if, $\mathbf{x} = \mathbf{t}$. This is also sometimes referred to as *invertibility*.

*Continuity*: For any $\varepsilon > 0$ there exists $\delta = \delta(\varepsilon)$ such that $\rho_X(\mathbf{x}, \mathbf{t}) < \delta$ implies that $\rho_Y(F(\mathbf{x}), F(\mathbf{y})) < \varepsilon$ where $\rho(\cdot, \cdot)$ is a distance measure in a given metric space.

The learning problem being discussed in this thesis is inherently ill-posed and clearly violates the uniqueness criteria for two reasons. The first is the fact that the cloud-base height is quantized in 300 meter intervals. Thus several $\mathbf{x}$ that lead to identical cloud-base heights that fall within the 300 meter range between the forty-four measurements that span the 13 km of the atmosphere, will map to the same height. Secondly, several extremely different sets of conditions could lead to the same cloud base. For example one may consider seasonal variations. In both winter and summer there exist instances of clouds with bases at any given height. At the very least, surface temperature would be different leading to a many to one mapping. The well-posedness issue is further compounded by sensor noise which exacerbates the uniqueness requirement and can lead to a violation of the continuity requirement.

## 2.5 Discussion

This chapter has outlined the current state of the field of sensor fusion. It has covered taxonomy, architectural issues and types of fusion algorithms. While giving an indication of the pervasiveness of the technology via the diversity of problems it has been applied to, emphasis has been placed on neural network approaches to remote sensing problems. As well, the difficulty of performing regression problems and data concerns has been elucidated.

With these discussions in mind, it should now be clear what is meant by the formal statement that this thesis deals with low-level, centralized, data fusion that works on essentially raw data with little conditioning. Furthermore, the choice of a non-parametric fusion technique, the neural network, is justified on the basis of the complex nature of the data and uniqueness of the problem where no models exist.

Finally, a large hole exists in the remote sensing literature in terms of using neural networks as sensor fusion engines for regression problems. This thesis serves as a pioneering work in this area. In particular, the problem at hand deals with a real and extremely large data set that comprises over fifty-seven variables from different sensors and as will be seen, provides solid results and an extremely relevant tool to the meteorological community.

# Chapter 3

# Nonlinear Regression Via a Learning Approach

Now that the problem has been described, this chapter shifts the focus to the neural network techniques that will be employed. The basic problem is an inverse problem that, despite not being continuous in the output, is best described as regression. Most importantly, there is a lack of physical laws to provide a mathematical basis for the estimation, and hence the need for developing an empirical model exists. Typically in the remote sensing literature, linear approaches have been used for modeling and this chapter begins with a description of how to form a linear solution. The question that will be answered is: Can nonlinear regression perform better? To find the answer, this chapter reviews the theory behind four nonlinear machine-learning techniques; the multilayer perceptron trained with the back-propagation (BP) algorithm, the node-decoupled extended Kalman filter (NDEKF) trained multilayer perceptron, the radial-basis function (RBF) network, and the support vector machine (SVM).

## 3.1 Linear Regression

Linear regressions fits a linear model to the data in order to derive a relationship between the input variables $x_n(i)$ and the output, $y(i)$. Thus, the goal is to find the coefficients $a_n$ where $n : 1 \rightarrow N + 1$ that satisfy the equation.

$$a_1 x_1(i) + a_2 x_2(i) + \ldots + a_n x_n(i) + a_{N+1}(1) = y(i) \quad \forall \quad i \tag{3.1}$$

Included is the term $a_{N+1}$ which simply provides a bias. Typically, determination of these weights involves solution of the following equation.

$$\mathbf{Xa} = \mathbf{b} \tag{3.2}$$

Here the matrix $\mathbf{X}$ is composed of the pattern vectors with a column of ones appended to account for the bias inputs, and $\mathbf{a}$ is the vector, of dimension $N + 1$, of coefficients to be determined. A simple matrix inversion is all that is required to calculate the coefficients and then an inner product with a test pattern vector provides the estimated output of $y$.

Another linear method of interest is the use of *correlation coefficients*. These relate the degree to which a particular input variable is responsible, in a linear sense, for the output. Coefficients with a value of zero are independent with respect to the output. Those with a value of one can account for all of the variation in the output. Denoted by the symbol $\rho_n$, they are determined using the following equation:

$$\rho_n = \frac{\sum(\mathbf{x_n} - \bar{x}_n)(\mathbf{y} - \bar{y})}{\sqrt{\sum(\mathbf{x_n} - \bar{x}_n)^2 \sum(\mathbf{y} - \bar{y})^2}} \tag{3.3}$$

Use of $\rho_n$ is made in chapter five to illustrate contributions from individual sensors and also to reveal the gains achieved by implementing nonlinear methods.

## 3.2 Introduction to Neural Network Methods

A total of four different learning algorithms were studied; two global and two local. In each case, the very best state-of-the-art learning algorithms were selected. Global methods include the back-propagation with momentum trained multilayer perceptron (Hagiwara, 1992), but more significantly, the node-decoupled extended Kalman filter trained MLP (Puskorious and Feldkamp, 1991). Local methods investigated are the regularized radial-basis function network (Haykin, 1999), and the support vector machine (Vapnik, 1998). All techniques use the feedforward architecture seen earlier and reproduced here for convenience (figure 3.1). These methods are described briefly below and more detail is given in sections devoted to each of the algorithms.

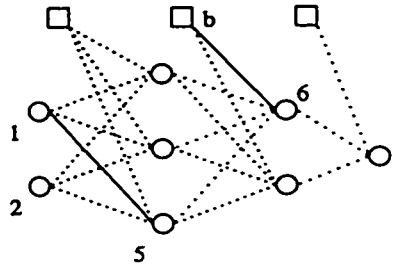



Figure 3.1: Feedforward multilayer perceptron. The leftmost solid line represents weight $w_{5,1}$ while the other solid line represents the weight $w_{6,b}$ from the bias unit to neuron 6.

The back-propagation algorithm used in training the multilayer perceptron architecture continuously computes an error gradient upon presentation of each training pattern and adjusts the network weights to minimize the error between the network output and the desired response. Once training is complete, the network can be used in a strictly feedforward mode, being fed input patterns and producing the output variable of interest. Uses of this algorithm have been widespread and well-documented as have its shortcomings (Hagan et al., 1996). Despite these shortcomings, it is included due to its pervasiveness in the literature and general acceptance as a benchmark.

Extended Kalman filter (EKF) principles may be used to train neural networks by considering the weights to be the state of a system. Using the familiar feedforward architecture, this method computes a recursive estimate of the gradient, not instantaneous as does the back-propagation algorithm. This results in training that is less prone to becoming trapped in local minima. Though training complexity may be extremely large, a modification of the global EKF (GEKF) training algorithm permits EKF techniques to be used on large networks by making the assumption that weights form logical groupings and that these groupings may be updated independently. This is the node-decoupled EKF (NDEKF) algorithm which keeps computational complexity to a moderate level.

Typically, the above global algorithms generate different results depending on initial conditions. For this reason, a *committee machine* was used in both cases to mitigate the effects of random initialization which lead to the network settling in different local minima. Such a technique involves training numerous networks with the same data and the same training parameters but each network is initialized differently and their outputs are then simply averaged to overcome the *bias/variance dilemma* (Haykin, 1999).

Hearkening back to the discussion of the ill-posedness of this learning problem, two methods have been chosen with intent to assuage this condition. They are the regularized

radial-basis function network and the support vector machine. Both of these methods allow the embedding of prior information into the solution of the problem in the form of a smoothness constraint.

Radial-basis function networks contain a single hidden layer which consists of locally sensitive units whose response decreases as a function of distance from the center. Hence, RBF networks can be described as a decomposition of the input into basis functions which are then multiplied by a series of weights to form the output. Of all the algorithms, the RBF is particularly appealing from a connectionist standpoint as the nervous system exhibits evidence of neurons that are locally "tuned" to some point of the input space and RBF networks are built upon this principle (Poggio and Girosi, 1990). After selection of the RBF centers, training involves computation of the weights. This is generally a simple task due to the linearity of the outputs and it is accomplished via application of the pseudoinverse function to $A$ in the general algebraic problem $\mathbf{Ax} = \mathbf{b}$. To enhance performance, tuning of the training may be performed. For example, regularization can be used to mitigate the effects of noise and the usual ill-posed nature of these types of problems (Hansen, 1994). As well, different kernels and the norms they utilize may be employed.

Finally, support vector machines define the learning problem in a formal optimization framework. The machine is based on the concepts of structural and empirical risk minimization. Benefits of the process include relaxing concern about local minima and bypassing the *curse of dimensionality* associated with large problems by performing the optimization in the input space. Another strong feature of the process is its ability to determine its own dimensionality. This is unlike the other networks where considerable user input is required.

## 3.3 Back-Propagation Trained Multilayer Perceptrons

The back-propagation algorithm is the most widely used of all neural network training methods. It is implemented in feedforward multilayer perceptrons (MLPs) and is a gradient descent type of algorithm. The derivations for the weight adjustment equations are quite well published (Haykin, 1999) and only a cursory treatment is made here.

In figure 3.1, suppose that an input vector $\mathbf{x}$ is presented to the network. With this input present, each node produces an output ($y_j$) according to equation 3.4 where the function $\phi(\nu)$ is typically the sigmoidal nonlinearity $\phi(\nu) = 1/(1 + \exp(-\nu))$.

$$y_j = \phi(\nu_j) = \phi(\sum_{i=1}^{p} w_{ji} x_i) \tag{3.4}$$

In such a fashion, the input flows through the network until ultimately, the output layer produces the final output. This is a supervised method and since the desired outputs ($d_j(n)$) of neurons in the output layer are known, it is a simple matter to generate an error signal for these neurons as follows.

$$e_j(n) = d_j(n) - y_j(n) \tag{3.5}$$

The goal is to minimize the average squared error of the network. Given $N$ training examples, this may be expressed as follows.

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(n) \tag{3.6}$$

Here, $C$ includes all neurons in the output layer and $N$ is the training set size. The variable parameters in the network are the weights and liberal usage of the chain rule allows derivation, in terms of usable network values, of how the average error changes with regard to the weights.

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial \nu_j(n)}\frac{\partial \nu_j(n)}{\partial w_{ji}(n)} \tag{3.7}$$

Details presented in (Haykin, 1999) eventually lead to the equation for the weight update.

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{3.8}$$

where $\eta$ is a user specified *learning rate* parameter and $\delta$ is the *local gradient* of neuron $j$.

Two cases for performing the weight updates exist and differ only in the computation of the local gradient $\delta_j(n)$. These cases occur when neuron $j$ is either an output neuron or when neuron $j$ is located in the hidden layer. This distinction arises since, in the case of an output neuron, the desired response is known. However, no simple specification exists for what the output of a hidden layer neuron should be. Thus, the following two equations dictate the local gradients for output and hidden neurons respectively. In the second equation, $\sum_k$ is a summation of the $\delta$ terms from the previous layer (when traversing the network from back to front during the back-propagation phase).

$$\delta_j(n) = e_j(n)\phi'(\nu_j(n)) \tag{3.9}$$

$$\delta_j(n) = \phi'(\nu_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{3.10}$$

Learning may be accelerated by the use of a *momentum term* ($\alpha$), which decreases the chance of becoming trapped in local minima. This benefit arises as the momentum term increases the degree of the weight correction if the previous weight corrections were in the same direction. Thus it possesses a stabilizing effect. Modifying equation 3.8, the weight update with the momentum term is as follows.

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{3.11}$$

Now that the learning properties have been established, one critical consideration is the point at which training is ceased. The danger in continual training is that the network will eventually begin to learn the inconsistencies and noise in the data and even though training error will continue to decrease, subsequent generalization performance will suffer. Cross-validation is an accepted heuristic for determining the optimal stopping point. It uses a second data set that is passed through the network at periodic intervals in order to test the network's generalization ability. When the generalization error of this set begins to increase, over-training is said to be occurring and training is halted. This phenomena is illustrated in figure 3.2.

Many modifications have been proposed to the back-propagation algorithm. Weight pruning to eliminate unnecessary weights, the addition of a constant value to the derivative

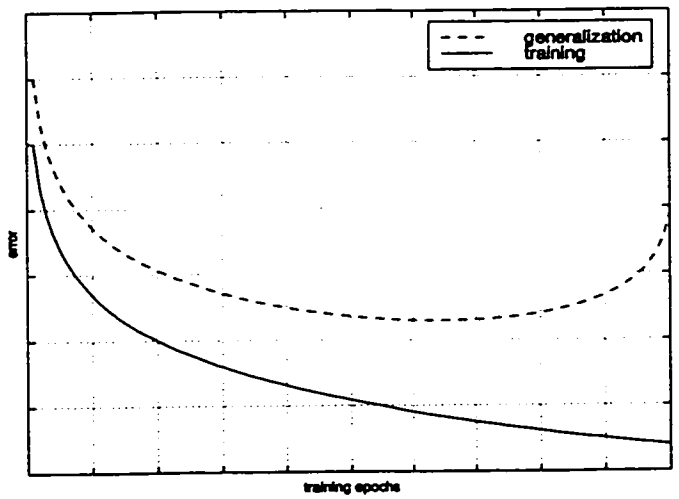



Figure 3.2: Generalization error with the test set begins to increase as the network learns inconsistencies in the data. However, training error continues to decrease.

of the activation function in order to encourage the network to pass over flat spots on the error surface and the use of curvature information (Quickprop) all have merits, however, the standard back-propagation with momentum algorithm remains the standard method of network training.

**Summary of the Back-Propagation Algorithm**

1. Select network parameters (architecture, learning rate $\eta$, momentum term $\alpha$, nonlinearity $\phi$).
2. Initialize weights to small random values.
3. Feed one pattern through and compute the outputs using the following equation on each neuron. The $x_i$ are inputs to neuron $j$ via the weights $w_{ji}$.

$$y_j = \phi(\nu_j) = \phi(\sum_i w_{ji}x_i)$$

4. Compute the errors in the neurons of the output layer:

$$e_j(n) = d_j(n) - y_j(n)$$

5. Update the weights via the following equation:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

where,

$$\begin{aligned} \delta_j(n) &= e_j(n)\phi'(\nu_j(n)) && \text{for output layer neurons} \\ \delta_j(n) &= \phi'(\nu_j(n)) \sum_k \delta_k(n) w_{kj}(n) && \text{for hidden layer neurons} \end{aligned}$$

6. Go to step three until all patterns have been presented. Once completed, this constitutes one epoch of training.
7. Either use cross-validation and repeat training until validation error begins to rise or train for a fixed number of epochs.

## 3.4 Node-Decoupled Extended Kalman Filter Trained Multilayer Perceptrons

One serious concern with the back-propagation algorithm is its reliance on the estimation of the instantaneous gradient to drive the learning process. With this dependence, no memory (save the case where momentum is added) is built into the system and it becomes more susceptible to local minima.

In this section, the extended Kalman filter (EKF) is considered as a method of training the feedforward architecture where the weights of a network are considered to be the state of a system. Its usage of second order statistics and recursive estimate of the gradient leave it less susceptible to local minima. Discussion first develops the extended Kalman filter and then it's application to MLP training is elucidated. The section closes with a modification that permits its use on large networks by keeping computational complexity to a moderate level.

### 3.4.1 EKF Filter Derivation

Consider a nonlinear finite-dimensional system where $\mathbf{x}(n)$ is the state of the system and is to be estimated. It is determined by:

$$\hat{\mathbf{x}}(n+1) = f_n(\mathbf{x}(n)) + g_n(\mathbf{x}(n))\mathbf{w}(n) \tag{3.12}$$

Here, the future state, $\mathbf{x}(n+1)$, is a nonlinear function $f_n(\cdot)$, of the current state plus some nonlinear function $g_n(\cdot)$, of the present state multiplied by the current process noise $\mathbf{w}(n)$.

In practical terms, the state is not observable and the only information available is an observation $\mathbf{d}(n)$, which is a nonlinear function $h_n(\cdot)$, of the current state plus measurement noise $\mathbf{v}(n)$.

$$\mathbf{d}(n) = h_n(\mathbf{x}(n)) + \mathbf{v}(n) \tag{3.13}$$

Given sufficient smoothness, the nonlinear functions can be expanded via a Taylor series so that:

$$\begin{aligned}
\mathbf{x}(n+1) &= \mathbf{F}(n)\mathbf{x}(n) + \mathbf{G}(n)\mathbf{w}(n) + \mathbf{u}(n) && (3.14)\\
\mathbf{z}(n) &= \mathbf{H}^T(n)\mathbf{x}(n) + \mathbf{v}(n) + \mathbf{y}(n) && (3.15)\\
\mathbf{u}(n) &= f_n(\hat{\mathbf{x}}(n|n)) - \mathbf{F}(n)\hat{\mathbf{x}}(n|n) && (3.16)\\
\mathbf{y}(n) &= h_n(\hat{\mathbf{x}}(n|n-1)) - \mathbf{H}^T(n)\hat{\mathbf{x}}(n|n-1) && (3.17)
\end{aligned}$$

where,

$$\begin{aligned}
\mathbf{H}^T(n) &= \frac{\partial h_n(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}(n|n-1)} && (3.18)\\
\mathbf{F}(n) &= \frac{\partial f_n(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}(n|n)} && (3.19)\\
\mathbf{G}(n) &= g_n(\hat{\mathbf{x}}(n|n)) && (3.20)
\end{aligned}$$

By truncating the series, a recursion can be developed that leads to the estimation of $\hat{\mathbf{x}}(n+1|n)$ (Anderson and Moore, 1979). Truncation after the second term leads to what

are formally termed second-order extended Kalman filters. In theory, higher-order Kalman filters may be generated; however, in most cases, the second-order system suffices. The steps in this recursion are laid out in equations 3.21 to 3.25.

$$\hat{\mathbf{x}}(n+1) = f_n(\hat{\mathbf{x}}(n|n)) \tag{3.21}$$

$$\hat{\mathbf{x}}(n|n) = \hat{\mathbf{x}}(n|n-1) + \mathbf{k}(n)[\mathbf{d}(n) - h_n(\hat{\mathbf{x}}(n|n-1))] \tag{3.22}$$

$$\mathbf{k}(n) = \mathbf{P}(n|n-1)\mathbf{H}(n)[\mathbf{R}(n) + \mathbf{H}^T(n)\mathbf{P}(n|n-1)\mathbf{H}(n)]^{-1} \tag{3.23}$$

$$\mathbf{P}(n+1|n) = \mathbf{F}(n)\mathbf{P}(n|n)\mathbf{F}^T(n) + \mathbf{G}(n)\mathbf{Q}(n)\mathbf{G}^T(n) \tag{3.24}$$

$$\mathbf{P}(n|n) = \mathbf{P}(n|n-1) - \mathbf{k}(n)\mathbf{H}^T(n)\mathbf{P}(n|n-1) \tag{3.25}$$

In this recursion, $\hat{\mathbf{x}}(n+1)$ approaches $E[\mathbf{x}(n+1)|\mathbf{d}(0), ..., \mathbf{d}(n)]$ for large $n$ and is optimal for the linear case but only loosely optimal for the nonlinear case.

### 3.4.2 Multilayer Perceptron Training (GEKF)

Singhal and Wu (1989) have presented a method for training multilayer perceptrons using the Kalman filter algorithm. The important feature that permits application of this technique to neural networks is that a description of the network via the weights can be considered to be the state of the network. Consider a fully connected network with $L$ layers where the input layer is layer 1 and the output layer is labeled $L$. Each layer has $\mathcal{N}(n)$ weights (where $n$ is the index of the layer) and a bias input, thus the total number of weights is determined via the following equation.

$$\mathcal{M} = \sum_{n=1}^{L-1} (\mathcal{N}(n)+1)\mathcal{N}(n+1) \tag{3.26}$$

These weights are arranged into a single state vector $\mathbf{x}(n)$ which is not dynamic in that the weights of the network are fixed and once trained will not change with time. Thus, the system model expressed in equation 3.12 becomes $\hat{\mathbf{x}}(n+1|n) = \mathbf{x}(n)$ or more simply, by dropping the conditional notation, $\mathbf{x}(n+1) = \mathbf{x}(n)$. Also, letting $\mathbf{o}^k$ denote the output of layer k, the observation in equation 3.13 becomes:

$$\mathbf{d}(n) = \mathbf{o}^L(n) + \mathbf{v}(n) = h_n(\mathbf{x}(n), \mathbf{o}^1) + \mathbf{v}(n) \tag{3.27}$$

Finally, the following recursion expresses the training process. Note that the input vectors only appear through the output and its associated error vector $\boldsymbol{\xi}$.

$$\mathbf{A}(n) = [\mathbf{R}(n) + \mathbf{H}^T(n)\mathbf{P}(n)\mathbf{H}(n)]^{-1} \tag{3.28}$$

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{H}(n)\mathbf{A}(n) \tag{3.29}$$

$$\hat{\mathbf{x}}(n+1) = \hat{\mathbf{x}}(n) + \mathbf{k}(n)\boldsymbol{\xi}(n) \tag{3.30}$$

$$\mathbf{P}(n+1) = \mathbf{P}(n) - \mathbf{k}(n)\mathbf{H}^T(n)\mathbf{P}(n) \tag{3.31}$$

where,

| | |
|---|---|
| $\mathbf{R}(n)$ | $\mathcal{N}$ by $\mathcal{N}$ measurement noise |
| $\mathbf{H}(n)$ | $\mathcal{M}$ by $\mathcal{N}$ gradient of weights with respect to the outputs |
| $\mathbf{P}(n)$ | $\mathcal{M}$ by $\mathcal{M}$ weight error covariance matrix |
| $\mathbf{A}(n)$ | $\mathcal{N}$ by $\mathcal{N}$ scaling matrix |
| $\mathbf{k}(n)$ | $\mathcal{M}$ by $\mathcal{N}$ Kalman gains |
| $\hat{\mathbf{x}}(n)$ | $\mathcal{M}$-dimensional weight vector |
| $\boldsymbol{\xi}(n)$ | $\mathcal{M}$-dimensional error vector of network output layer |

**Initialization**

Both $\mathbf{P}(n)$ and $\mathbf{x}(n)$ require initialization and can express *a priori* information about the weights. In the absence of any such information, $\mathbf{P}(0)$ is initialized as a diagonal matrix with elements in the order of $10^2$, $10^4$ and $\mathbf{x}(0)$ is simply assigned small random values.

### 3.4.3 Node Decoupled Extended Kalman Filter Trained MLP

When networks get to be of moderate size, training of the EKF MLP becomes computationally expensive. For the network used later in chapter 5, there are two hidden layers of thirty and twenty neurons, respectively. Applying equation 3.26 results in a calculation for the total number of weights being 2381. Thus, in this case of what Puskorious and Feldkamp (1991) refer to as the global extended Kalman filter (GEKF) the $\mathbf{P}$ matrix is 2381 square and not only the number of calculations but also the storage requirements become a concern.

The node-decoupled EKF (NDEKF) algorithm is a simplification of the GEKF case that is based on ignoring the interdependence of mutually exclusive groups of weights. The use of decoupling permits adjustment of computational complexity in order to suit the available computational resources and is accomplished by modifying the recursion as

follows:

$$\mathbf{A}(n) = \left[\mathbf{R}(n) + \sum_{i=1}^{g} \mathbf{H}_i^T(n)\mathbf{P}_i(n)\mathbf{H}_i(n)\right]^{-1} \tag{3.32}$$

$$\mathbf{k}_i(n) = \mathbf{P}_i(n)\mathbf{H}_i(n)\mathbf{A}(n) \tag{3.33}$$

$$\hat{\mathbf{x}}(n+1) = \hat{\mathbf{x}}(n) + \mathbf{k}(n)\boldsymbol{\xi}(n) \tag{3.34}$$

$$\mathbf{P}_i(n+1) = \mathbf{P}_i(n) - \mathbf{k}_i(n)\mathbf{H}_i^T(n)\mathbf{P}_i(n) \tag{3.35}$$

Here the $\mathbf{P}$ matrix is broken into a block diagonal form by asserting that the weights of a network may be grouped such that it is possible to ignore the elements of $\mathbf{P}(n)$ that correspond to correlations between different weight groups. If the number of groups is high, a significant reduction in the complexity of $\mathbf{P}(n)$ results. Figure 3.3 shows how the weight groups are formed. A group's membership is composed of all weights leading into a neuron from a previous layer or a bias unit. The $\mathbf{P}(n)$ matrix that results from the network in figure 3.3 is illustrated in figure 3.4. Here, while the matrix is still twenty by twenty, only sixty-eight elements are nonzero as compared to four hundred in the GEKF case. Storage requirements for $\mathbf{P}_i(n)$ drop from $O(M)$ in GEKF to $O(\sum_{i=1}^{g} m_i^2)$ where $m_i$ is the dimension of the submatrix.

One point that must be adhered to is the keeping of $\mathbf{P}(n)$ positive definite. Without this constraint, singularity of the matrix may result. Puskorius proposes a heuristic solution by including an artificial noise process, a diagonal matrix $\mathbf{Q}_i(n)$, as seen in equation 3.36 which replaces equation 3.35.

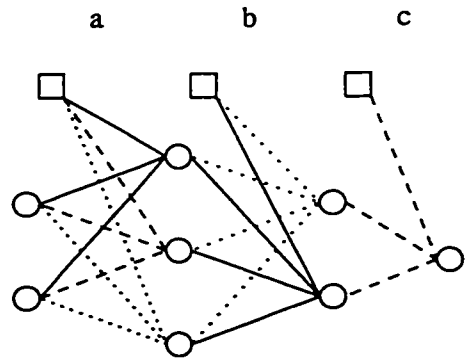



Figure 3.3: Two-layer multilayer perceptron showing groupings for the node-decoupling. Each set of distinct lines leading to a node identifies a group of weights. Bias units are squares.

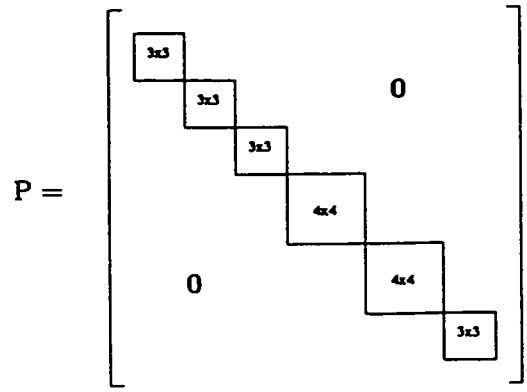



Figure 3.4: Weight error covariance matrix structure for network shown in figure 3.3.

$$\mathbf{P}_i(n+1) = \mathbf{P}_i(n) - \mathbf{k}_i(n)\mathbf{H}_i^T(n)\mathbf{P}_i(n) + \mathbf{Q}_i(n) \tag{3.36}$$

Typically, values in the range $10^{-2}$ to $10^{-6}$ are used and an annealing schedule may be set up. Overall, the claim is made that in addition to circumventing numerical instability, convergence is accelerated and local minima are avoided. Results shown by Puskorius reveal that training the NDEKF is generally an order of magnitude faster than its global counterpart. Puskorious and Feldkamp (1991) propose further modifications which results in a more efficient form.

---

**Summary of Node-Decoupled Extended Kalman Filter Training Algorithm**

---

1. Initialize network parameters ($\mathbf{Q}$, $\mathbf{R}$, $\mathbf{P}, \mathbf{x}$).
2. Perform the following recursion

$$\begin{aligned}
\mathbf{A}(n) &= \left[\mathbf{R}(n) + \sum_{i=1}^{g} \mathbf{H}_i^T(n)\mathbf{P}_i(n)\mathbf{H}_i(n)\right]^{-1} \\
\mathbf{k}_i(n) &= \mathbf{P}_i(n)\mathbf{H}_i(n)\mathbf{A}(n) \\
\hat{\mathbf{x}}(n+1) &= \hat{\mathbf{x}}(n) + \mathbf{k}(n)\boldsymbol{\xi}(n) \\
\mathbf{P}_i(n+1) &= \mathbf{P}_i(n) - \mathbf{k}_i(n)\mathbf{H}_i^T(n)\mathbf{P}_i(n) + \mathbf{Q}_i(n)
\end{aligned}$$

3. Go to step two until all patterns have been presented. Once completed, this constitutes one epoch of training.
4. Either use cross-validation and repeat training until validation error begins to rise or train for a fixed number of epochs.

---

## 3.5 Radial-Basis Function Networks

Radial-basis function (RBF) networks are built upon the principle of locally tuning neurons to points in the input space. They use the standard feedforward architecture but contain a single hidden layer of nonlinear neurons and a single linear output neuron. Hence, RBF networks may be described as a decomposition of the input into basis functions followed by a weighted summation to form the output. The training procedure is formulated by first assuming that the relationship underlying the data is as follows:

$$y(i) = f(\mathbf{x}(i) + \vartheta(i)), \qquad i = 1, 2, \ldots, N \tag{3.37}$$

where $\vartheta$ is additive noise. An approximation to the function $f(\cdot)$ can be expressed as follows.

$$\hat{f}(\cdot) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x}_j - \mathbf{x}_i||) = \mathbf{w}^T \Phi(\mathbf{x}_j, \mathbf{x}_i) \tag{3.38}$$

where $\phi$ is a nonlinear radial basis function and the matrix $\Phi$ is referred to as the interpolation matrix.

The complete training cycle involves determination of the RBF unit centers ($\mathbf{x}_j$) and computation of the weight vector $\mathbf{w}$. The latter is a relatively simple task due to the linearity of the outputs and is accomplished using the pseudoinverse function. Center selection may be undertaken in a variety of ways. Self-organized selection, which places more centers in areas of higher density, has been proposed as has the random selection of centers. Both of these methods attempt to use a representative set of centers thus cutting down the size of the interpolation matrix and hopefully preventing ill-conditioning. This

thesis uses *strict interpolation* where each data point in the training set becomes a center. While this yields as precise a local representation as possible, the resulting interpolation matrix can be quite large and bring about certain computational difficulties.

## Kernel Selection and Norm Types

Each interpolating basis function $\phi$ is generally a Gaussian kernel that has been translated in the data space. However, Gaussian kernels are not the only possibility. Powell (1987) makes an argument that theoretically and experimentally, the choice of basis function bears little on the performance of the RBF network. But, as will be seen in chapter 5, this is not necessarily true, particularly in the case of nonlinear regression, as recent work on optimal kernel selection has pointed out (Ormoneit and Hastie, 1999).

As examples of other kernel options, equations 3.39, 3.40 and 3.41 respectively define the following three kernels: Gaussian, inverse multiquadric, and a variant of the Gaussian where the exponential term is first square rooted (herein referred to as the SRG kernel). In these equations, $\sigma$, $N$ and $c$ are user defined and $r$ is a norm. Typically, the $L_2$ norm is used but nothing limits the use of the $L_1$ or any other norm for that matter, save performance.

$$\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{3.39}$$

$$\phi(r) = \frac{1}{(r^2+c^2)^{\frac{1}{2}}} \tag{3.40}$$

$$\phi(r) = \exp\left(-\frac{1}{N}\sqrt{r^2}\right) \tag{3.41}$$

The Gaussian and SRG kernels are illustrated in figure 3.5.

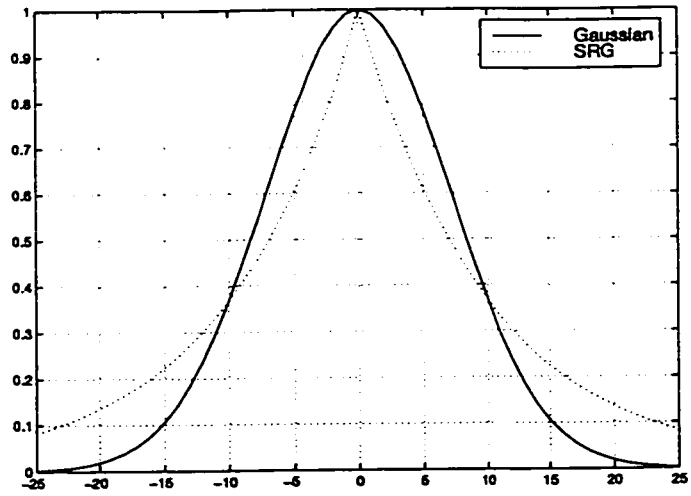



Figure 3.5: Two different kernels used in the radial-basis function network.

### 3.5.1 Regularization of the RBF

RBF training requires the solution of the problem $\Phi \mathbf{w} = \mathbf{y}$. Though a solution may be reached through simple inversion of the matrix $\Phi$, the solution for $\mathbf{w}$ may not always be as expected. A good example of this phenomenon is demonstrated by Hansen (1994) where noise in the vector $\mathbf{y}$ and numerical instability lead to different solutions depending on the method chosen. Unfortunately, the ill-posedness of these problems and corruption by noise leads to erroneous results as not only the fundamental relationship is learned but also, a model of the noise is trained into the network. Regularization smooths the function approximation and mitigates the effects of noise in the data. This requires a modification of the training procedure above. Instead of equation 3.38, the solution to equation 3.42 is computed. Here, $\lambda$ is the regularization parameter and is determined by the user in order to build a prescribed degree of smoothness into the solution.

$$(\Phi + \lambda \mathbf{I})\mathbf{w} = \mathbf{y} \tag{3.42}$$

Fundamental to the regularization concept is the singular value decomposition which is used to perform the matrix inversion. The decomposition of $\Phi$ may be expressed as follows:

$$\Phi = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \mathbf{u}_i \sigma_i \mathbf{v}_i^T \tag{3.43}$$

Here, the right singular vector (columns of U) and left singular vectors (columns of V) form orthonormal bases. The matrix $\mathbf{\Sigma}$ is zero except for the diagonal which denotes the contributions of the singular vectors to the matrix $\Phi$. The inversion of $\Phi$ is given by:

$$\Phi^{-1} = \mathbf{V}\frac{1}{\mathbf{\Sigma}}\mathbf{U}^T = \sum_{i=1}^{n} \mathbf{v}_i \frac{1}{\sigma_i} \mathbf{u}_i^T \tag{3.44}$$

To test the ill-posedness of the problem and suitability for regularization several criteria exist. These are given by Hansen (1994) and are enumerated below. A problem may be ill-posed if

1. Its singular values decay gradually to zero.
2. The ratio of largest to smallest singular values (condition number $\kappa$) is large.
3. The left and right singular vectors tend to have more sign changes as index $i$ increases and $\sigma_i$ decreases.

Point one implies that there are no nearby problems with a well conditioned coefficient matrix and well determined numerical rank. Item two refers to the condition number

($\kappa$) and implies sensitivity to perturbations in the target output **y**. Point three implies high-frequency content in the rightmost singular vectors. As these have corresponding small singular values, upon inversion via equation 3.44, it can be seen that the originally dominant singular vectors' contributions are diminished and the contribution from the singular vectors corrupted with high frequency noise are amplified.

Though regularization is generally implemented via equation 3.42, one could simply effect regularization via truncation. Here, the singular values and vectors associated with the part of the decomposition that deals with noise are simply left out. However, use of this method requires an operator decision for the cut-off point. As well, while these components typically convey mostly noise information, it is possible that they still encode information about the system and thus, complete truncation disposes of useful information. Fortunately, the more sophisticated regularization method does not possess this negative trait.

### Filter Factors and L - curves

One method of viewing the effect of regularization is through the use of filter factors. Regularization dampens or filters out the contributions corresponding to small singular values. Thus filter factors determine to what extent a singular value is to be kept in the solution. Tikhonov regularization results in filter factors of the following form:

$$f_i = \frac{\sigma_i^2}{(\sigma_i^2 + \lambda)^2} \tag{3.45}$$

Filtering occurs primarily for the $\sigma_i$ less than the regularization parameter $\lambda$. The effect of varying $\lambda$ is illustrated in figure 3.6 where an interpolation matrix for a RBF network involving 4281 centers is considered. Here, a regularization parameter of 0.1 has little effect

on the dominant singular vectors and only significantly attenuates singular vectors with an index greater than 2500. The attenuation becomes progressively more pronounced as $\lambda$ increases. The line marked *optimal via gcv* gives the filter factors that are optimal for this particular problem and shows a distrust of most of the upper singular vectors. The $\lambda$ leading to these filter factors is determined via generalized cross-validation (GCV) which will be discussed later.

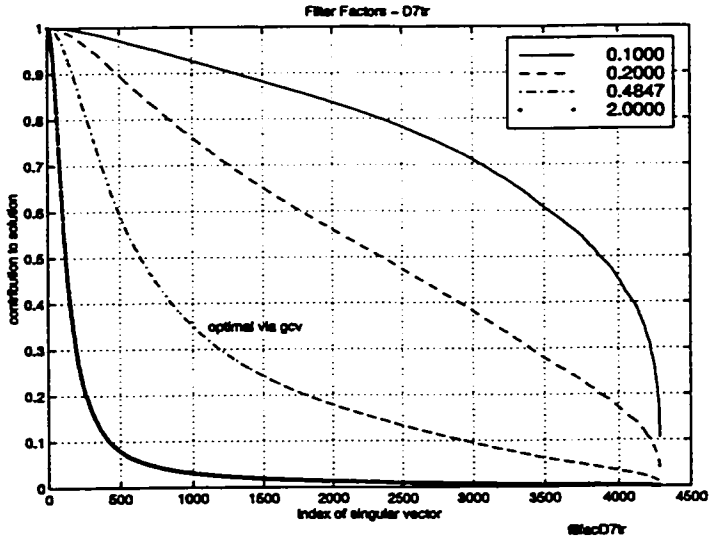



Figure 3.6: Filter factors for a large interpolation matrix.

The question arises as to how to choose the regularization parameter. To gain insight into this problem, the L-curve provides an intuitive tool (Hansen and O'Leary, 1993). This is a plot over all valid regularization parameters of the norm of regularized solution versus the corresponding residual norm. It shows that the best regularization parameter is one that results in a compromise between minimizing these two norms. A typical L-curve is illustrated in figure 3.7. The figure may be broken into two main sections.

*Vertical Area*: Here, the perturbation in the target vector error dominates the

regularized solution $\mathbf{w}_{reg}$, and the solution encodes information about system noise.

*Horizontal Area*: In this region, regularization error dominates as the solution is smoothed to greatly. This results in the loss of information that characterizes the problem.

It is impossible to construct a solution lying below the curve and the optimal $\lambda$ that makes the best trade-off against the perturbation and regularization errors lies in the region of maximum curvature.

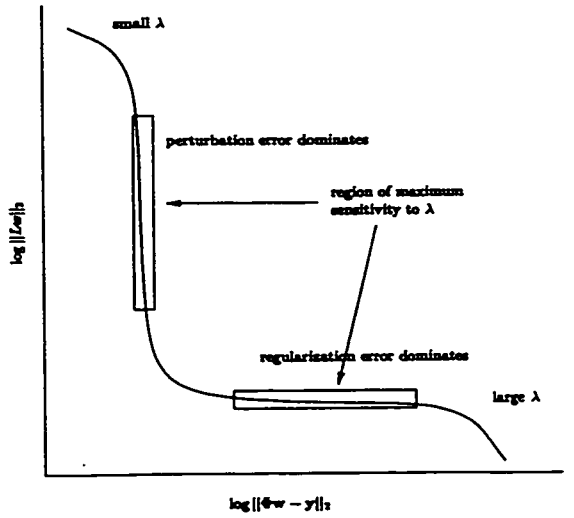



Figure 3.7: L-curve

## Implementing Regularization

The dominating approach to regularization requires that the 2-norm of the solution be small. Thus a theoretical framework for regularization involves minimizing this norm $\Omega(w) = ||L(\mathbf{w} - \mathbf{w}^*)||_2$ and then balancing the result with the residual norm $||\Phi\mathbf{w} - \mathbf{y}||_2$. The most common and well studied method of regularization is based on this approach and is called

Tikhonov regularization. The optimal weight vector achieved with this technique satisfies the following equation.

$$\mathbf{w}_\lambda = argmin||\Phi\mathbf{w} - \mathbf{y}||_2^2 + \lambda^2||L(\mathbf{w} - \mathbf{w}^*)||_2^2 \tag{3.46}$$

### Choosing the Regularization Parameter, GCV

Generalized cross-validation is the most common technique of selecting the optimal regularization parameter and is not to be confused with the method of cross-validation previously discussed and used for determining the optimal stopping point during back-propagation training. It is based on the notion that if an arbitrary element of $\mathbf{b}$ is left out, a well regularized solution should still be able to accurately predict that value. The optimal $\lambda$ is determined by the following equation where, $\Phi^I$ is the matrix that yields the regularized solution $\mathbf{w}_{reg} = \Phi^I\mathbf{y}$.

$$\lambda_{opt} = G_{min}\frac{||\Phi\mathbf{w}_{reg} - \mathbf{y}||_2^2}{trace(\mathbf{I}_m - \Phi\Phi^I)^2} \tag{3.47}$$

Further discussion of this method may be found in (Hansen, 1992). In this reference, it is shown that the method balances the same perturbation and regularization errors that are fundamental to the L-curve.

## 3.5.2 Numerical Stability of the Matrix Inversion

In dealing with matrices that exhibit a large condition number ($\kappa$), the use of standard methods to invert a matrix such as the SVD can lead to numerical instability. Techniques

exist which can overcome the instability of the system as well as significantly reduce computation. One technique that may be employed is inversion via the Cholesky decomposition.

Cholesky factorization of a positive definite matrix $\Phi$ results in a matrix $\mathbf{C}$ such that $\Phi = \mathbf{C}\mathbf{C}^T$. For the RBF case, the positive definite constraint is not a concern since the interpolation matrix is positive definite by construction. Using this factorization, $\Phi\mathbf{w} = \mathbf{y}$ becomes $\mathbf{C}\mathbf{C}^T\mathbf{w} = \mathbf{y}$. Thus an intermediate vector may be formed where $\mathbf{k} = \mathbf{C}^T\mathbf{w}$. The system is now reduced to $\mathbf{C}\mathbf{k} = \mathbf{y}$ which may easily be solved via substitution since $\mathbf{C}$ is lower triangular. Now that numerical values for $\mathbf{k}$ exist, the same method of substitution can be used to solve $\mathbf{C}^T\mathbf{w} = \mathbf{k}$ and thus obtain values for $\mathbf{w}$.

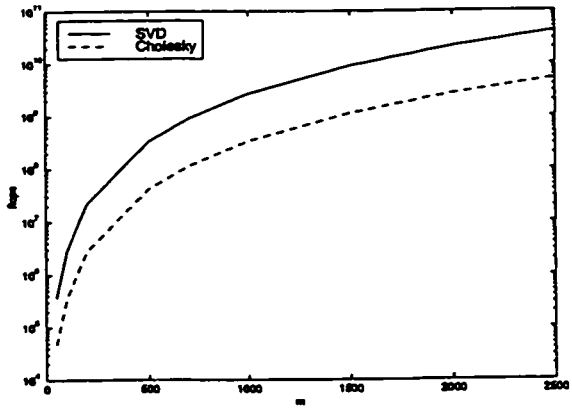



Figure 3.8: FLOPS required for SVD and Cholesky decomposition on an $m$ by $m$ matrix.

Both the SVD and the Cholesky methods require $O(n^3)$ operations (Golub and Loan, 1990). To get a solid feel for the computational requirements, figure 3.8 shows the demands made while calculating the SVD and the Cholesky decomposition of the same matrix as its size increases. As a trend, the Cholesky factorization requires an order of magnitude less floating-point operations (FLOPS ) and additionally will require one quarter the memory since only half of one $m^2$ matrix has to be stored whereas the SVD requires

two full $m^2$ matrices: one for the right singular vectors and another for the left.

---

**Summary of Radial-Basis Function Training**

---

1. Initialize parameters (kernel selection, kernel parameters).
2. Determine training centers. Use all for strict interpolation.
3. Form the interpolation matrix $\Phi(\mathbf{x}_j, \mathbf{x}_i)$ .
4. Determine the optimal $\lambda$ using generalized cross-validation.
5. Solve for the weights.

$$\mathbf{w} = (\Phi + \lambda \mathbf{I})^{-1}\mathbf{y}$$

---

## 3.6 Support Vector Machine

Based on the concepts of structural and empirical risk minimization, support vector machines (Vapnik, 1998) define the learning problem in a formal optimization framework. Benefits of the SVM are the ability to relax the typical degree of concern about local minima and the ability to bypass the *curse of dimensionality* associated with large problems by performing the optimization in the input space. And, unlike the other networks where considerable user input is required, another strong feature of the process is its ability to determine its own dimensionality.

### 3.6.1 Formulating the Regression Problem

Central to the design of a support vector machine for regression are two important concepts:

1. The inputs are mapped into a high-dimensional feature space and thus a simpler regression function may be found. This makes use of Cover's theorem (Cover and Thomas, 1991).
2. The solution is optimal and arrived at by solving a quadratic optimization problem with linear constraints. This is more sophisticated than other methods which require significant user input.

Development of the theory begins by considering the linear regression case and the process may be expressed as follows. Find the function $f(\cdot)$ in

$$y = f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b \tag{3.48}$$

with the constraint that all estimates of $y$ are within some precision $\epsilon$. Since noise is usually present in the data, the training problem finds an approximation to $y$ called $\hat{y}$.

The first step involves the definition of a loss function as illustrated in figure 3.9 and expressed mathematically in equation 3.49.

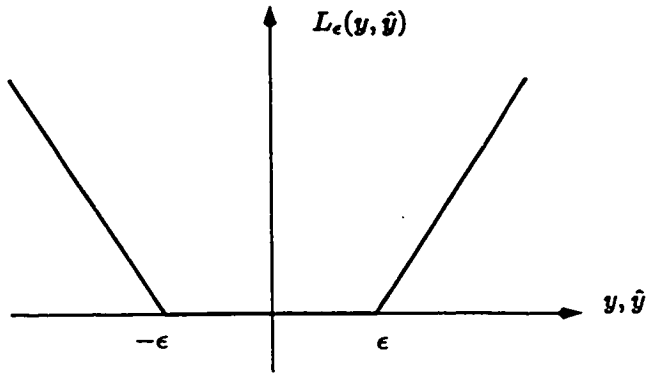



Figure 3.9: $\epsilon$-insensitive loss function for nonlinear regression.

$$L_\epsilon(y,\hat{y}) = \begin{cases} 0 & \text{if } |y-\hat{y}| \leq \epsilon \\ |y-\hat{y}| - \epsilon & \text{otherwise} \end{cases} \tag{3.49}$$

From figure 3.9 and equation 3.49, it can be seen that as long as the approximating function produces an answer that lies within the bounds of epsilon, the loss function declares that there is zero error. Thus, $L_\epsilon$ is referred to as the $\epsilon$-insensitive loss function. Should $f(\mathbf{x})$ stray outside the $\epsilon$-boundary, a penalty is incurred and summation of these penalties allows the beginning of the derivation of the optimization problem. Thus, an *empirical risk* ($R_{emp}$) is declared:

$$R_{emp} = \frac{1}{N}\sum_{i=1}^{N} L_\epsilon(y_i,\hat{y}_i) \tag{3.50}$$

Equation 3.50 can simply be thought of as a figure of merit as to the success of the regression. Minimization of this risk factor results in the optimal weight vector w. Development of the SVM algorithm proceeds by transforming $R_{emp}$ into an optimization problem by defining two slack variables to bring any estimate outside the epsilon boundary to its edge. These variables $(\xi_i, \xi_i')$ may then be incorporated into the problem definition using the following equations.

$$\mathbf{w}^T\mathbf{x}_i - y_i + b \leq \varepsilon + \xi_i, \qquad \xi_i \geq 0, \qquad i = 1,2,\ldots,N \tag{3.51}$$

$$y_i - \mathbf{w}^T\mathbf{x}_i - b \leq \varepsilon + \xi_i', \qquad \xi_i' \geq 0, \qquad i = 1,2,\ldots,N \tag{3.52}$$

Minimizing the Euclidean norm of the weight vector is equivalent to determining an optimally flat regression function. With this in mind, setting up the optimization problem results in,

$$W(\mathbf{w}, \xi, \xi') = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i') \tag{3.53}$$

Thus, the optimization process creates a trade-off between model complexity and the number of points lying outside the zero region of the loss function and thus requiring a positive slack variable. Therefore $C$ can be considered to play the role of a regularization parameter. To find the solution, the Lagrangian is formed using Lagrange multipliers $\alpha_i$, $\alpha_i'$, $\gamma_i$ and $\gamma_i'$.

$$\begin{aligned} J(\mathbf{w}, \xi, \xi', b, \alpha, \alpha') = \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i') \\ & - \sum_{i=1}^{N}\alpha_i[\mathbf{w}^T\mathbf{x}_i - y_i + b + \epsilon + \xi_i] \\ & - \sum_{i=1}^{N}\alpha_i'[y_i - \mathbf{w}^T\mathbf{x}_i - b + \epsilon + \xi_i'] - \sum_{i=1}^{N}(\gamma_i\xi_i + \gamma_i'\xi_i') \end{aligned} \tag{3.54}$$

Next, the saddle point of this Lagrangian is determined by satisfying the following three conditions of optimality which lead to several important relationships.

$$\frac{\partial J}{\partial \mathbf{w}} = 0 \qquad \rightarrow \qquad \mathbf{w} = \sum_{i,j=1}^{N} (\alpha_i - \alpha_i')\mathbf{x}_i \tag{3.55}$$

$$\frac{\partial J}{\partial \xi} = 0 \qquad \rightarrow \qquad \gamma_i = C - \alpha_i \tag{3.56}$$

$$\frac{\partial J}{\partial \xi'} = 0 \qquad \rightarrow \qquad \gamma_i = C - \alpha_i \tag{3.57}$$

Substitution of these equations into equation 3.54 yields the final form of the optimization problem.

$$\text{Minimize} \quad W(\alpha_i, \alpha_i') = \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i') - \epsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i') + \frac{1}{2} \sum_{i,j=1}^{N} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j')\langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{3.58}$$

subject to

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0 \tag{3.59}$$

$$0 \leq (\alpha_i, \alpha_i') \leq C \tag{3.60}$$

The vectors for which $\alpha_i \neq \alpha_i'$ are the support vectors for the problem and the unique solution results in a weight vector that is expressed as:

$$\mathbf{w} = \sum_{i=1}^{l} (\alpha_i - \alpha_i')\mathbf{x}_i \tag{3.61}$$

Here, the $\alpha_i$ and $\alpha_i'$ can be interpreted as forces pushing and pulling the estimate $f(\mathbf{x}_i)$ towards $y_i$. If the problem is well behaved, the number of support vectors will be significantly less than the number of initial pattern vectors.

One criticism of the $\epsilon$-insensitive loss function is that it tends to underfit the data if $\epsilon$ is chosen too large thus introducing a *systematic bias* (Muller et al., 1999). Other loss function's, such as Huber's, overcome this bias at the cost of significantly increasing the number of support vectors.

Central to the entire process are the Karush-Kuhn-Tucker (KKT) conditions of optimality. These lead to the statement that the only $\alpha_i$ or $\alpha_i'$ that may have non-zero values are those that lie on the boundary of the epsilon-insensitive function.

In order to determine the value of $b$, exploitation of the KKT conditions show that:

$$b = y_i - \langle w, x_i \rangle - \epsilon \quad \text{for} \quad \alpha_i \in (O, C) \tag{3.62}$$

$$b = y_i - \langle w, x_i \rangle + \epsilon \quad \text{for} \quad \alpha_i' \in (O, C) \tag{3.63}$$

Other modifications involve not specifying $\epsilon$ *a priori* but rather auto-computing $\epsilon$ by changing the objective function to allow specification of the percentage of support vectors and then optimizing over $\epsilon$. See (Schölkopf et al., 1999) for details.

## Equivalency with Neural Networks

Mercer's theorem permits substitution of an inner product kernel $\Phi(\mathbf{x}_i, \mathbf{x}_j)$ into (3.58) for the term $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. This permits the process of mapping the inputs into the nonlinear feature space mentioned earlier. What's more, it allows the optimization to be performed in the input space rather than the feature space. In effect, this uses Cover's theorem to increase

| network type | kernel |
|---|---|
| RBF network | $\exp(-\frac{1}{2\sigma^2}\|\|\mathbf{x}_j - \mathbf{x}_i\|\|^2)$ |
| perceptron | $\tanh(\beta \mathbf{x}_j^T \mathbf{x}_i + c)$ |

Table 3.1: Kernels that yield popular neural network architectures.

the dimensionality of the problem and facilitate the finding of linear solution in this space without unduly complicating the problem. Popular choices for the inner product kernel are shown in table 3.1 and will generate the corresponding neural network.

### 3.6.2 Reducing Run-Time Complexity

The optimization process for the regression machine involves a quadratic problem with boundary constraints and one linear equality constraint. Equation 3.58 may be rewritten to emphasize this quadratic nature. Doing so produces,

$$\text{Minimize} \quad W(\alpha, \alpha') = \mathbf{y}^T(\alpha - \alpha') - \epsilon(\alpha + \alpha') + \frac{1}{2}(\alpha - \alpha')^T \mathbf{Q}(\alpha - \alpha') \tag{3.64}$$

subject to

$$(\alpha - \alpha')1 = 0 \tag{3.65}$$

$$0 \leq \quad (\alpha, \alpha') \leq C \tag{3.66}$$

where, $\mathbf{Q}_{ij} = \Phi(\mathbf{x}_i, \mathbf{x}_j)$.

Typically, optimization involves determining the Hessian of $\mathbf{Q}$ and searching along

its eigenvectors for a minimum. In a sizable problem, with a large number ($\ell$) of training examples, computation of the Hessian is expensive and memory requirements are extensive. Several heuristics have been applied to the optimization process in order to make large problems manageable. Generally, these involve exploiting certain known properties of the result and decomposing the problem into a series of smaller subproblems.

One such method is the chunking algorithm. Training proceeds by first optimizing the problem over an arbitrary subset of the original training vectors. Once this has been completed, the $q$ nonsupport vectors are removed from the subset and $q$ pattern vectors are added to the training set from the vectors not considered in the first pass. Optimization commences again and the cycle is iterated until the all pattern vectors have been added to the training problem. This process does not converge to the same solution as would optimization over the entire training set but it does derive a relationship that is quite close.

Another method, and the one used to calculate the results in chapter 5, involves dividing the data and their associated parameters $(\alpha, \alpha')$ into an active set $(B)$ and an inactive set $(N)$ where the set $B$ is much smaller than $\ell$. Thus, the following partitions are declared,

$$\alpha = \begin{vmatrix} \alpha_B \\ \alpha_N \end{vmatrix} \quad \alpha' = \begin{vmatrix} \alpha'_B \\ \alpha'_N \end{vmatrix} \quad \mathbf{y} = \begin{vmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{vmatrix} \quad \mathbf{Q} = \begin{vmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{NB} \\ \mathbf{Q}_{BN} & \mathbf{Q}_{NN} \end{vmatrix} \tag{3.67}$$

First, the original optimization problem of equation 3.58 is solved over the active set and the points in $N$ are tested for compliance with the KKT. Should any of them fail, any pair of $\alpha$ from $B$ are replaced and a new subproblem is defined and optimized. Considering that $\mathbf{Q}_{BN} = \mathbf{Q}_{NB}^T$, this new subproblem is defined by first expanding equation 3.58 as

follows.

$$
\begin{aligned}
W(\boldsymbol{\alpha}_B, \boldsymbol{\alpha}'_B) \;=\; & \epsilon(\boldsymbol{\alpha}_B - \boldsymbol{\alpha}'_B) + \epsilon(\boldsymbol{\alpha}_N - \boldsymbol{\alpha}'_N) \\
& -\mathbf{y}_B^T(\boldsymbol{\alpha}_B - \boldsymbol{\alpha}'_B) - \mathbf{y}_N^T(\boldsymbol{\alpha}_N - \boldsymbol{\alpha}'_B) \\
& +\frac{1}{2}(\boldsymbol{\alpha}_B - \boldsymbol{\alpha}'_B)^T\mathbf{Q}_{BB}(\boldsymbol{\alpha}_B - \boldsymbol{\alpha}'_B) \\
& +(\boldsymbol{\alpha}_B - \boldsymbol{\alpha}'_B)^T\mathbf{Q}_{BN}(\boldsymbol{\alpha}_N - \boldsymbol{\alpha}'_N) \\
& +\frac{1}{2}(\boldsymbol{\alpha}_N - \boldsymbol{\alpha}'_N)^T\mathbf{Q}_{NN}(\boldsymbol{\alpha}_N - \boldsymbol{\alpha}'_N)
\end{aligned} \tag{3.68}
$$

Since the terms that deal strictly with the set $N$ are static, they may be removed and the new problem becomes that expressed in step 4 of the algorithm's summary. The process continues by solving this subproblem and testing for violating data points in the set $N$. These are then used to replace data points in B and the subproblem is solved again. The process is stopped when set $N$ contains no KKT violating points.

Originally, Osuna showed that as long as at least one KKT violating example is added to the examples for the previous sub-problem, the sequence of QP sub-problems will asymptotically converge to the optimal solution (Osuna, 1999). Subsequent work by Keerthi et al. (1999) has shown that the claim must be weakened somewhat to state that the value of the objective function will monotonically decrease and approach the optimal solution. The main point to remember however is that throughout optimization, a constant matrix size is maintained with the benefit that by shrinking the optimization problem this way, memory requirements become linear.

**Summary of SVM Training using Osuna's Decomposition**

1. Arbitrarily choose $B$ points from the data set and make the following partitions.

$$\alpha = \begin{vmatrix} \alpha_B \\ \alpha_N \end{vmatrix} \quad \alpha' = \begin{vmatrix} \alpha'_B \\ \alpha'_N \end{vmatrix} \quad \mathbf{y} = \begin{vmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{vmatrix} \quad \mathbf{Q} = \begin{vmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{NB} \\ \mathbf{Q}_{BN} & \mathbf{Q}_{NN} \end{vmatrix}$$

2. Perform the following optimization over the variables in $B$.

$$\begin{aligned} \text{Minimize} \quad & W(\alpha, \alpha') && = \mathbf{y}^T(\alpha - \alpha') - \epsilon(\alpha + \alpha') \\ & && \quad + \frac{1}{2}(\alpha - \alpha')^T \mathbf{Q}(\alpha - \alpha') \\ \text{subject to} & \\ & (\alpha - \alpha')\mathbf{1} && = 0 \\ & 0 && \leq (\alpha, \alpha') \leq C \\ \text{where} & \\ & \mathbf{Q} && = \Phi(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

3. If there exists some $j \in N$, such that:
   - $\alpha'_j = 0$, $\alpha_j = 0$ and $|g(\mathbf{x}_j) - y_j| > \epsilon$
   - ($\alpha'_j = C$ or $\alpha_j = C$), and $|g(\mathbf{x}_j) - y_j| < \epsilon$
   - ($0 < \alpha'_j < C$ or $0 < \alpha_j < C$), and $|g(\mathbf{x}_j) - y_j| \neq \epsilon$,

   where

$$g(\mathbf{x}_j) = \sum_{i=1}^{\ell} (\alpha'_i - \alpha_i)\Phi(\mathbf{x}_i, \mathbf{x}_j) + b,$$

   replace any pair $(\alpha'_i, \alpha_i)$, $i \in B$, with $(\alpha'_j, \alpha_j)$

4. Solve the new subproblem defined below.

$$
\begin{aligned}
\text{Minimize} \quad W(\alpha_B, \alpha'_B) &= \epsilon(\alpha_B - \alpha'_B)^T 1 - (\alpha_B - \alpha'_B)^T (\mathbf{y} - \mathbf{q}_{BN}) \\
&\quad + \frac{1}{2}(\alpha_B - \alpha'_B)\mathbf{Q}_{BB}(\alpha_B - \alpha'_B)
\end{aligned}
$$

subject to

$$
\begin{aligned}
(\alpha_B - \alpha'_B)1 &= -(\alpha_N - \alpha'_N)1 \\
0 &\leq (\alpha, \alpha') \leq C
\end{aligned}
$$

where

$$
(\mathbf{q}_{BN})_i = \sum_{j \in N} (\alpha'_j - \alpha_j)\Phi(\mathbf{x}_i, \mathbf{x}_j) \qquad i \in B
$$

5. Repeat steps three and four until criteria of step three are met.

6. Determine the weight vector and the bias via the following equations

$$
\begin{aligned}
\mathbf{w} &= \sum_{i=1}^{l} (\alpha_i - \alpha'_i)\Phi(\mathbf{x}_i) \\
b &= y_i - \langle w, \Phi(x_i) \rangle - \epsilon \qquad \text{for } \alpha_i \in (0, C) \\
b &= y_i - \langle w, \Phi(x_i) \rangle + \epsilon \qquad \text{for } \alpha'_i \in (0, C)
\end{aligned}
$$

---

## 3.7 Summary

In this chapter four neural network training methods were discussed. None of the four methods are without concerns and consequent remedies. The two local methods, the back-propagation trained MLP and the node-decoupled extended Kalman filter MLP , are both known to be susceptible to local minima though the NDEKF is prone to this problem to a lesser extent. A committee machine has been used to reduce this susceptibilty. Of the local methods, the radial-basis function network suffers from poor conditioning and computationally expensive matrix inversion. Regularization and Cholesky factorization can

overcome these issues. Finally, in the case of the support vector machine, a sizable problem will require large amounts of memory and a significant amount of time for optimization. Again a solution exists, this time by decomposing the problem into subproblems and only optimizing over a subset of the variables at any given time.

With these descriptions in hand, the discussion in the next chapter turns to the origin of the data that will be fused by these methods to produce cloud-base height estimates.

# Chapter 4

# Data Set

Collection, understanding, and preprocessing of data are extremely important processes that ensure the integrity of the overall system. With these points in mind, the following chapter details formation of the data set. Discussion first revolves around the data's origin and then moves to descriptions of the individual sensors and the variables that they produce. The chapter concludes with a description of preprocessing issues such as outliers, variance normalization, and interpolation.

## 4.1 Data Set - The Southern Great Plains (SGP) Site

The data set was aquired through the department of meteorology at Pennsylvania State University and had been has been collected as part of the Atmospheric Radiation Measurement program. Of the several sites available, the geographic point of interest is located in the southern great plains. Being centered almost exactly between between Wichita, Kansas and Oklahoma City, Oklahoma, its map coordinates are 36.6 deg north latitude and 97.5 deg west longitude. Its selection is ideal because it plays host to both cold, dry winters and hot,

humid summers. Thus, in the course of a year, it provides data that lead to virtually every kind of cloud system. After careful consideration by meteorologists, fifty-seven sensors from six sensor suites that observe conditions at this site were chosen as they were felt to most closely relate to cloud formation. These covered radiation measurements, standard surface meteorological observations, satellite imagery, and of course the LIDAR measurements themselves.

## 4.2 Sensors Used

A complete listing of all sensors used is given in table 4.3 and is located at the end of this chapter. This section gives background on the sensors, revealing the reasons for their selection and the physics on which they are based.

### Micropulse LIDAR

The micropulse LIDAR is the target value which the neural network will attempt to estimate. Its operation is based on the principle that light is scattered and attenuated by atmospheric particles: either dust (aerosols) or water/ice (cloud), and the intensity of the LASER beam diminishes due to scattering as it travels. At each range gate, the backscattered light can be plugged into a series of equations that dictate the distance to the scatterer and a profile of the atmosphere results. Analysis can then yield information about aerosols or, in this case, cloud-base height. In 1993, four sophisticated LIDARs operating at 523 nanometers were constructed by the Goddard Space Flight Center (Spinhirne, 1993). There are three important features of these time gated, incoherent detection units that distinguish them from their predecessors.

1. **Pulse Characteristics:** The pulse repetition factor (PRF) is much higher being in the kilohertz range rather than Hertz. This allows the pulse energies to be much lower. Only ten microjoules are used rather than millijoules. This is the feature that leads to eye safety.

2. **Solid State:** Conventional lasers are flashlamp pumped. The MPL is diode pumped leading to more efficient operation and a smaller footprint.

3. **Signal Detection:** Photon counting is employed which is generally considered the most accurate method.

These LIDARs are capable of three hundred meter resolution and provide a much greater operating range than conventional ceilometers. Average vertical profiles are recorded every minute and clear sky is indicated by a cloud-base height of zero kilometers.

## Pyran/Pyrge/Pyrheli-ometer

The standard instruments for measuring the amount of electromagnetic energy at the surface of the earth are *pyranometers* and *pyrgeometers*. Both of these instruments consist of flat plates that absorb the radiation incident upon them and output a proportional voltage. An estimate of the total radiant energy incident on the plates, called the irradiance or flux (Watts/meter$^2$), is obtained by calibrating the output voltage to the incident power. The difference between the two instruments is the wavelength of the photons to which their detectors are sensitive. *Pyranometers* measure the irradiance in the wavelength interval of 0.3 to 4.0 microns, whereas *pyrgeometers* measure the irradiance in the wavelength interval of 4.0 to 50.0 microns. An instrument, which points only at the sun, that directly measures the solar direct beam irradiance is a *pyrheliometer* or normal incidence pyrheliometer. These sensors are summarized in table 4.1.

| Type of *ometer* | Measures | Wavelength ($\mu m$) |
|---|---|---|
| pyranometers | solar irradiance | 0.3 - 4.0 |
| pyrgeometers | earth emitted photons | 4.0 - 50.0 |
| pyrheliometers | solar direct beam irradiance | 0.3 - 4.0 |

Table 4.1: Solar and terrestrial radiation measuring sensors.

Three important scientific facts are:

1. The bulk of the energy coming from the sun is in the wavelength interval from 0.3 to 4.0 microns,
2. The bulk of the energy emitted from the constituents of the atmosphere, the ocean water and the land surface is in the wavelength interval from 10.0 to 20.0 microns,
3. The majority of the photons in the atmosphere are in the 0.3 to 4.0 and 4.0 to 50.0 micron wavelength intervals and have their origin from the sun and earth, respectively.

Therefore, *pyranometers* measure the irradiances due to solar (broadband shortwave or visible to near-infrared) photons and *pyrgeometers* measure the irradiances due to earth-emitted (broadband longwave or infrared or thermal) photons.

The photons from the sun are approximately in a parallel beam of radiation at the top of the atmosphere. Once the photons enter the atmosphere of the earth, they can be scattered, absorbed or transmitted without any interaction with the surface of the earth. On clear-sky days with no clouds, the probability of a photon being scattered increases as one over the fourth power of the photon wavelength, or $1/(\lambda^4)$, and is called Rayleigh scattering (blue photons are scattered more efficiently than red photons). Photons are rarely absorbed at visible/near-infrared wavelengths between 0.3 and 1.0 microns; however,

beyond 1.0 microns water vapor and carbon dioxide in the atmosphere are very effective at absorbing solar photons in certain wavelength bands.

Cloud drops efficiently scatter radiation of all wavelengths and absorb significant amounts of photons above and beyond what the clear sky does. Cloud drop absorption, however, is not completely independent of water vapor absorption since the same molecule is doing the absorbing, just in a different macroscopic state, i.e., vapor versus liquid.

The solar photons that reach the surface without undergoing any interaction make up what is called the direct beam irradiance or normal solar irradiance. When a solar photon is scattered, it is no longer considered as part of the direct beam; rather, it becomes part of the diffuse field irradiance. Note that solar photons that are scattered in the same direction as the direct beam, i.e., in the forward direction, are also considered as part of the diffuse field. In summary, all photons that are scattered back to space are diffuse field photons since they have undergone at least one scattering event, while the photons that reach the surface can be part of either the direct beam or diffuse field.

The sum of the direct beam irradiance plus the diffuse field irradiance makes up the total shortwave irradiance at the surface, which is the quantity measured by an unshaded pyranometer. A shaded pyranometer, shaded from the direct beam of the sun that is, measures the diffuse field irradiance. The difference between the unshaded pyranometer irradiance and the shaded pyranometer irradiance yields an estimate of the direct beam irradiance.

## Multi-Filter Rotating Shadowband Radiometer (MFRSR)

The multi-filter rotating shadowband radiometer (MFRSR) is like an unshaded pyranometer and a shaded pyranometer rolled into one instrument. It measures the solar irradiance in only a few, narrowband wavelengths from about 0.4 to 1.0 microns.

Photons that emanate from some material in the earth/atmosphere system are all considered to be diffuse field photons. In this case the longwave surface irradiance has no direct beam contribution. The pyrgeometer, which measures the longwave surface irradiance, is simply measuring the diffuse field photons that emanate from the earth and are incident on its surface.

The microwave radiometer measures the number of photons originating from the water vapor and liquid water in the atmosphere above the instrument. Using these photon counts, one is able to infer the actual atmospheric column integrated amounts of water vapor and cloud liquid water. If there are no clouds above the instrument, the integrated cloud liquid water is zero. If big rain clouds are above the instrument, then the integrated cloud liquid water is large. On clear dry days, when there is not much water vapor in the atmosphere, the integrated water vapor amount should be low, but not zero. On the other hand, on humid days, just after thunder showers for example, the integrated amount of water vapor in the atmosphere is rather large. In addition to the microwave radiometer, there is a simple infrared radiometer that effectively measures photons that are produced by clouds, but does not very efficiently measure those produced by water vapor. Therefore, when clouds are present, it measures a significant amount of photons; furthermore, as the base of the cloud is closer to the ground and warmer, the photon count goes up. These photon counts are transformed to a temperature scale; warmer temperatures indicate clouds with bases that are closer to the ground. Note that ice crystals do not emit any significant amount of energy at wavelengths that the microwave radiometer operates. Therefore, the microwave radiometer is insensitive to ice. Furthermore, if there is any dew or rain water on the antennas of the microwave radiometer or infrared radiometer, their measurements are useless since the instruments are responding to the water on the antennas and not the water vapor and cloud liquid water up in the atmosphere.

### Surface Meteorological Observation System (SMOS) Instruments

The Surface Meteorological Observation System (SMOS) uses conventional sensors to obtain one minute and thirty minute averages of surface wind speed, wind direction, air temperature, relative humidity, barometric pressure, and precipitation.

### Satellite Data

Of central importance to this thesis is the Geosynchronous Operational Environmental Satellite 8 (GOES-8), which provides two-dimensional cloud and temperature imagery in both visible and infrared spectra. In total, five different channels of data are available covering various wavelengths and thus provide differing views of the atmosphere. Cloud data are provided as cloud top temperatures, which can then be calibrated to give the height of the cloud. Resolutions are one kilometer patches for the visual channel and four kilometer patches for the infrared. The channels, their wavelengths and utility are contained in table 4.2, which also shows the synergies of some of the channels and how combinations yield different modalities otherwise unavailable.

All the surface-based instruments simply produce a measurement exactly at the grid location however, some uncertainty as to the positioning of the satellite can occur. In order to overcome this and also provide some idea of cloud texture around the site, the four pixels surrounding the location were averaged and their standard deviations were included as features.

| Channel | Wavelength | Features |
|---|---|---|
| 1 | 0.52 - 0.72 mm | The detection of pollution, haze and smoke, more accurate cloud height measurements. |
| 2 | 3.78 - 4.03 mm | Distinguish between water and ice clouds during daytime hours (used in synthesis with channels one and four). |
| 3 | 6.47 - 7.02 mm | Improved analysis of synoptic-scale features. |
| 4 | 10.20 - 11.20 mm | Improved low level moisture identification in combination with channel five. |
| 5 | 11.50 - 12.50 mm | More accurate low-level moisture measurements. |

Table 4.2: GOES 8 channels.

### Microwave Water Vapor Radiometer

These instruments measure the total amount of water vapour and liquid water from the surface to the top of the atmosphere. This is basically the same as the reading one would achieve by condensing all the water vapour to liquid and collecting it in a rain gauge. It will not however, detect ice which typically occurs above five kilometers.

## 4.3 Formation of the Data Set

The data set spans the time period of May 1, 1996 to August 31, 1997. In total, over thirty gigabytes of information were sifted through via a rather extensive set of programs in order to obtain the final data set. Each pattern vector was composed of the surface measurements at a given time and the corresponding satellite values. As the satellite information was the most sparse (occuring roughly each half hour), it was used as the reference time point and pattern vectors were formed by obtaining readings from the other sensors that were within one minute of the satellite reading. If this was not possible, all data associated with that

time stamp were discarded.

With such a large suite of sensors, difficulties arise in that should any one sensor become inoperational, it would render that time point invalid. At worst, this would occur for a day or two and, in general, the data were well distributed in time over the full year and four months. As well, only daylight hours were considered since visual information is available during this period.

Another form of preprocessing involved the examination of the raw data for obvious outliers. For example, in figure 4.1, data for all fifty-seven sensors are illustrated. In several boxes, obvious outliers are evident as exhibited by spikes. These are obvious sensor errors and the pattern vectors for these time points were discarded.

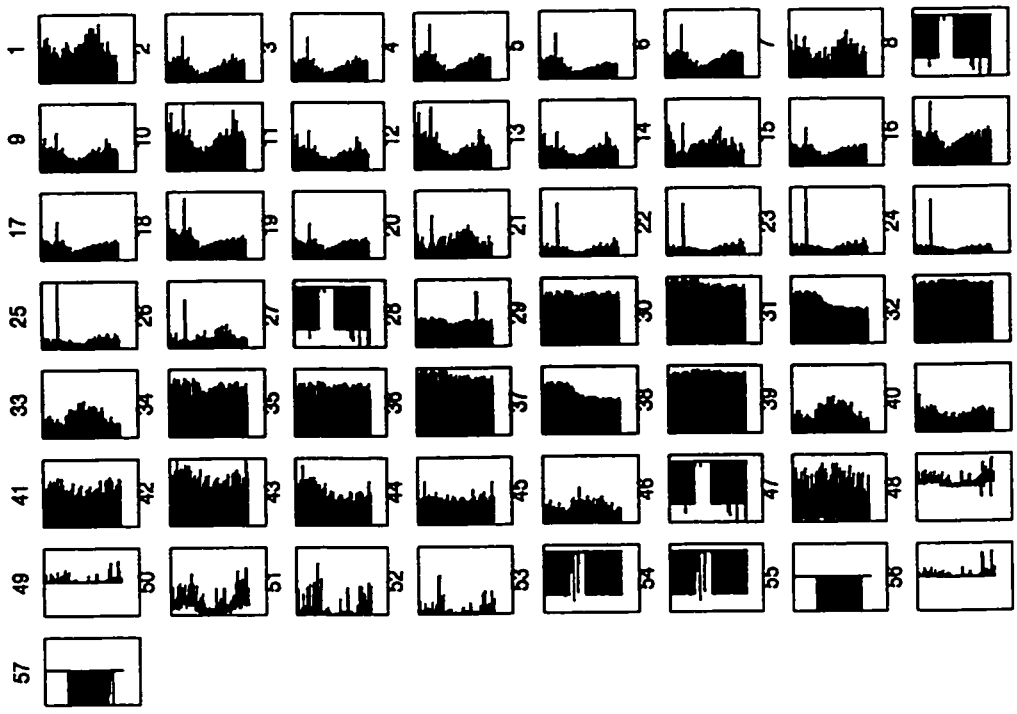



Figure 4.1: Columns of data matrix demonstrating outliers.

Variance normalization was another point that required addressing. The multitude of ground-based sensors and the satellite data, possess different dynamic ranges. This often causes problems during training as relative variances between components can have a significant affect on a learning algorithm's performance. Generally, the sensors with high variance will tend to dominate. The severity of this situation is illustrated in figure 4.2. The typical solution involving removal of the mean and normalization of each sensor's variance to unity was employed.

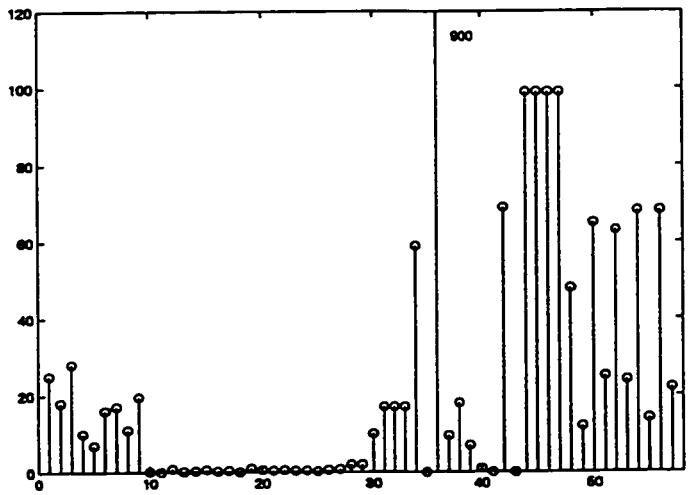



Figure 4.2: Variances of the variables. The y-axis denotes the variance and the x-axis in the number of the sensor (see table at the end of the chapter).

The final form of preprocessing was strict linear interpolation of the data to a regular time grid. Generally, this process is performed to align data for incorporation into models and is also seemed to have a slightly beneficial effect during processing — this is most likely due to a slight regularizing (smoothing) effect. One could argue that the process could have generated false data points that, by virtue of the interpolative process, would become easy to predict but this was not the case. Care was taken to only consider days where large

amounts of data were available. Specifically, days with roughly continuous data between the hours of 15:15 and 22:15 hours were used. Interpolation was performed over this interval commencing at 15:15 in half-hour increments. This corresponds to the theoretical times of the satellite images and, in reality, the data were not greatly displaced from their original time stamp. Before interpolation, the data set contained 3103 pattern vectors. After the interpolative process, it contained 3405. Thus, the number of data points that were "filled" in was not actually that large. The resulting distribution of cloud-base heights is shown in figure 4.3. Finally, the data set was shuffled and divided into a training set of 2724 pattern vectors and 681 test vectors. Shuffling of the data before division ensured that seasonal variations were distributed among the two sets.

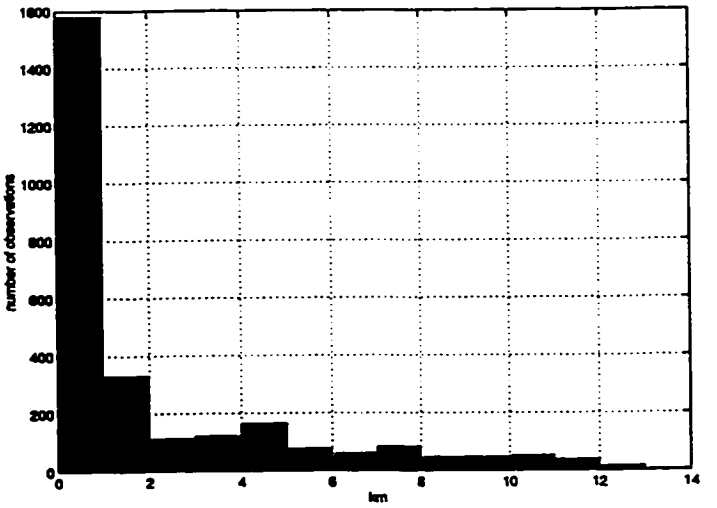



Figure 4.3: Distribution of cloud-base heights in data set.

The entire process is summarized as follows:

1. Obtain data from all sensors at the time designated by the satellite measurements.
2. Strip outliers.

3. Normalize the variance.
4. Keep days of data if measurements are continuous between 15:15 and 22:15 hours.
5. Linear interpolate data to a regular time grid.
6. Shuffle pattern vectors and divide into training and test sets.

## 4.4 Summary

The data set used in this thesis is extremely rich and spans roughly one and half years. It is comprised of fifty-seven different sensor variables which were chosen due to their close relationships to cloud formation. Owing to the diversity of sensors used, and the difficulty in keeping all of them completely operational, care was taken in building the data set. As well, preprocessing due to the varying dynamic ranges had to be taken into account. Thus preprocessing included removal of outliers, normalization of variance, and interpolation of the data to a regular time grid.

| ID | Name | Description or units |
|---|---|---|
| **SIROS - Solar and Infrared Observing System** | | |
| 1 | hemisp_broadband | Hemispheric Irradiance, MFRSR |
| 2 | diffuse_hemisp_broadband | Diffuse Hemispheric Irradiance, MFRSR |
| 3 | direct_norm_broadband | Direct Normal Irradiance, MFRSR |
| 4 | up_long_hemisp | 10 meter Upwelling Longwave Hemispheric Irradiance, Pyrgeometer |
| 5 | down_long_diffuse_hemisp | Downwelling Longwave Diffuse Hemispheric Irradiance, Ventilated Pyrgeometer |
| 6 | up_short_hemisp | 10 meter Upwelling Shortwave Hemispheric Irradiance, Pyranometer |
| 7 | down_short_hemisp | Downwelling Shortwave Hemispheric Irradiance, Ventilated Pyranometer |
| 8 | down_short_diffuse_hemisp | Downwelling Shortwave Diffuse Hemispheric Irradiance, Ventilated Pyranometer |
| 9 | short_direct_normal | Shortwave Direct Normal Irradiance, Pyrheliometer |
| 10-15 | hemisp_narrowband | Hemispheric Irradiance, MFRSR |
| 16-21 | diffuse_hemisp_narrowband | Diffuse Hemispheric Irradiance, MFRSR |
| 22-27 | direct_norm_narrowband | Direct Normal Irradiance, MFRSR |
| **SMOS - Surface Meteorological Observing System** | | |
| 28 | wspd | m/s |
| 29 | wspd_vs | m/s |
| 30 | wdir | deg |
| 31 | temp | C |
| 32 | rh | % |
| 33 | vap_pres | kPa |
| 34 | bar_pres | kPa |
| 35 | precip | mm |
| **BSRN - Baseline Surface Radiation Network** | | |
| 36 | nip | $w/m^2$ Normal Incidence Pyrheliometer |
| 37 | psp1 | $w/m^2$ Precision Spectral Pyranometer |
| 38 | psp2 | $w/m^2$ Precision Spectral Pyranometer |
| 39 | psig | $w/m^2$ |
| **MWRLOS - Microwave Radiometer Line of Sight** | | |
| 40 | vap | (cm) total water vapor along LOS path |
| 41 | liq | (cm) total liquid water along LOS path |
| 42 | ir_temp | K |
| 43 | wet_window | unitless |
| **MWRAVG - Microwave Radiometer - Averaged** | | |
| 44 | vap | Averaged total water vapor along LOS path |
| 45 | liq | Averaged total liquid water along LOS path |
| 46 | water_flag_fraction | Fraction of data in averaging interval with water on Teflon window |
| 47 | water_flag_periods | Number of contiguous periods in averaging interval with water on Teflon window |
| **Satellite** | | |
| 48 | ir ch1 mean | 0.52 - 0.72 mm |
| 49 | ir ch1 stdev | |
| 50 | ir ch2 mean | 3.78 - 4.03 mm |
| 51 | ir ch2 stdev | |
| 52 | ir ch3 mean | 6.47 - 7.02 mm |
| 53 | ir ch3 stdev | |
| 54 | ir ch4 mean | 10.2 - 11.2 mm |
| 55 | ir ch4 stdev | |
| 56 | ir ch5 mean | 11.5 - 12.5 mm |
| 57 | ir ch5 stdev | |

Table 4.3: Variables used in the study.

# Chapter 5

# Results

This chapter discusses results derived via the four types of networks described in chapter three. It begins by first describing the manner in which error will be assessed and then presents results achieved by pure linear regression in order to guage improvements brought about by the nonlinear learning paradigm. In this framework, it also analyses the contributions made by each sensor and thus illustrates the need for the combination of all sensors in order to realize adequate performance. Analysis with the regularization tools presented in section 3.5.1 is performed and show the linear problem to be fairly well behaved and the result achieved to be the best that this technique can provide. It is presented mainly to demonstrate the large gains in accuracy achieved by using a nonlinear learning techniques.

Next, results and learning curves for the neural networks trained with the back-propagation and the NDEKF algorithms, in both stand-alone and in committee machine configuration, are presented. Radial-basis function network results with and without regularization demonstrate the power of regularization and the benefit gained by the interpolation of the data as well as the importance of kernel selection. Section 5.6 reviews the support vector machine's output. Trends noticed during training with all methods indicated that

the greatest sources of error are regions where little data exist. Thus the four algorithms are rerun with a slightly reduced data set [1] that focuses concern on the region where data are most prevalent (under 8.0 km). In doing this, significant performance improvements are achieved. The chapter concludes with a discussion of the complexity of these different methods and a summary that elucidates why the SVM is the method of choice.

## 5.1 Error Definition

The output of the learning algorithms is a continuous variable that, depending on the ability of the regression function, could take on virtually any value. This presents a problem since the learning problem tackled in this thesis is actually a quantized regression problem, that is to say, that the target values themselves are quantized in 300 meter intervals and thus form a set ($\mathcal{T}$) with 44 members. To make a fair assessment of the output error, outputs during the generalization phase were assigned the closest value in $\mathcal{T}$. Subsequently, all results are reported as mean squared error (MSE) between the target values and the quantized outputs of the networks.

## 5.2 Linear Regression

By solving the equation $\mathbf{Ax} = \mathbf{b}$, a purely linear relationship is established between the input variables (row-wise pattern vectors of $\mathbf{A}$) and the cloud-base height ($\mathbf{b}$). This relationship will yield a method of determining the mapping of sensor variables to CBH but, additional information may be obtained through the determination of the correlation coefficients as

[1] The term *reduced set* is sometimes used in the SVM literature to refer to a process of reducing the number of support vectors required for definition of decision boundaries or regression functions. In this thesis' case, it refers to truncating the data set in order to ensure that there are sufficient numbers of training patterns for each cloud-base height so that meaningful relationships may be drawn.

defined by equation 3.3. With these it may be determined to what degree each input variable contributes to the solution.

Figure 5.1(a) demonstrates this principle. Here, the 58 (57 sensors plus the cloud-base height itself) variables are numbered as in table 4.3 and the correlation coefficient of each variable with respect to the cloud-base height is determined. Plotting the absolute values of these coefficients forms a symmetric matrix where white represents a high correlation and dark a low correlation. The final vertical column represents the correlation coefficient of each sensor and cloud-base height and is plotted in figure 5.1(b). Much intuitive information may be gathered from this type of plot in order to determine the best features to use for training. For example, it can be seen that sensors 10-15 correlate well with sensors 22-27. This makes somewhat obvious sense since they represent the same type of information (radiometer). However, they do not correlate with cloud-base height as well as sensors 16-21 from the same suite as evidenced by the stronger correlations determined by looking at figure 5.1(b). As a final point, feature selection based on these linear relationships can be extended to nonlinear training as it stands to reason that data with a high nonlinear correlation will often also exhibit linear relationships.

By using the correlation coefficients, a table illustrating the contribution of each sensor to the accuracy of the prediction function may be generated (table 5.1). Once sorted and added to the regression equation one by one, there is an almost monotonically decreasing relationship between MSE and the addition of each sensor. Results are also given in this table for one run using the support vector machine. These may be ignored for the time being but will be referred to later in the chapter.

As was discussed during development of the RBF trained network, regularization can lead to better solutions of the $\mathbf{Ax} = \mathbf{b}$ problem (see section 3.5.1). Since these techniques will be applied to help solve for the weights in the RBF network, it seems fair to incor-

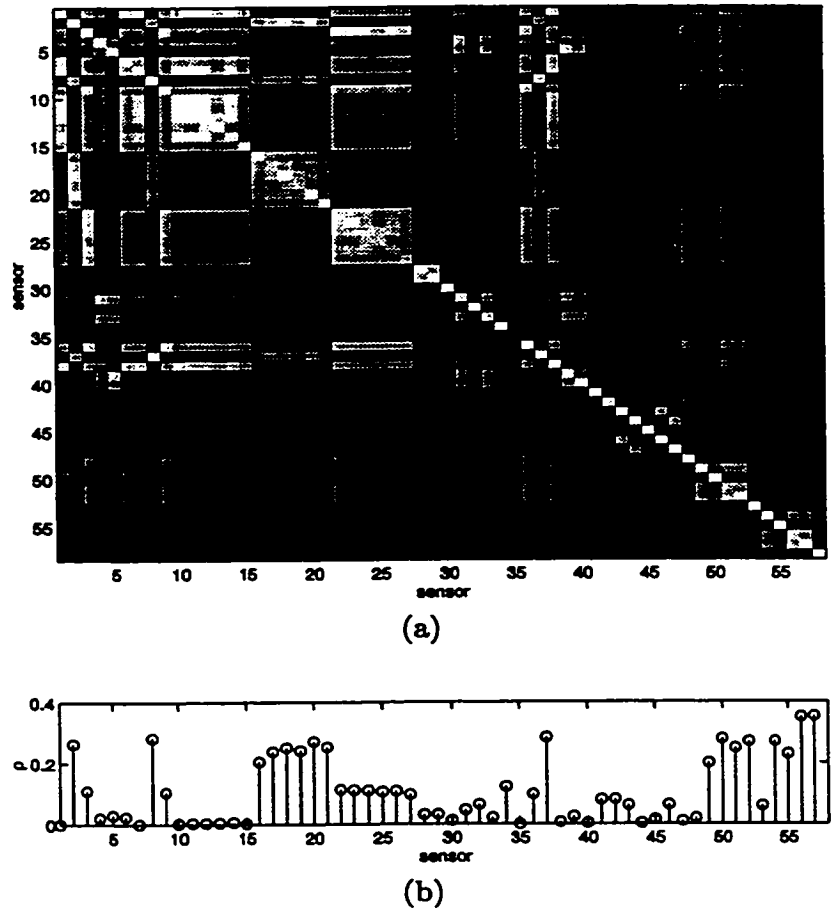



Figure 5.1: Correlogram. (a) Correlation coefficients between all variables. Sensor numbers correspond to those given in table 4.3. (b) Correlation coefficients between each sensor and cloud-base height.

porate regularization into the linear regression solution. Hence, the optimal regularization parameter ($\lambda_{opt}$) must be determined. Generalized cross-validation (GCV) determines this value by plotting the value of the GCV function against the range of valid $\lambda$. As illustrated in figure 5.2 no clear minimum emerges and for all purposes, a regularization parameter of zero, which corresponds to no regularization, is just as adequate as a small degree of regularization. In fact, any substantial amount of regularization only leads to a degenerate solution.

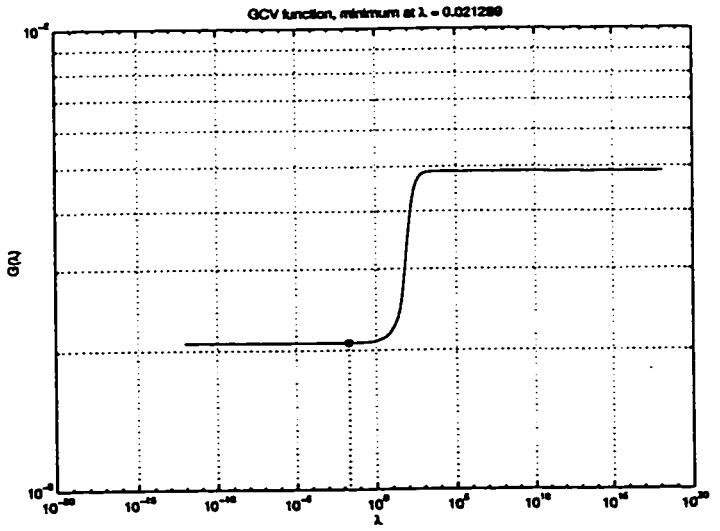



Figure 5.2: Generalized cross-validation for linear regression.

Finally, the accuracy of the linear regression approach is reported in table 5.2. The last column cites the overall correlation coefficient between the target and predicted values. The fact that the regularized solution, where $\lambda = 0.0213$ as determined in figure 5.2, is virtually identical to the unregularized solution shows that the result achieved by straight matrix inversion is quite stable.

| comp | index | reg $\rho$ | sensor name | linear regression MSE | support vector machine MSE | num SVs |
|---|---|---|---|---|---|---|
| 1 | 57 | 0.36 | ch5 stdev (sat) | 8.07 | 53.81 | 1367 |
| 2 | 56 | 0.36 | ch5 mean (sat) | 8.05 | 31.93 | 1356 |
| 3 | 50 | -0.30 | ch2 mean (sat) | 7.27 | 15.64 | 1281 |
| 4 | 37 | 0.29 | pep1 (smos) | 6.80 | 9.35 | 1203 |
| 5 | 8 | 0.29 | down.short.diff.hem (siros) | 6.77 | 9.25 | 1197 |
| 6 | 52 | -0.28 | ch3 mean (sat) | 6.77 | 8.59 | 1152 |
| 7 | 54 | 0.27 | ch4 mean (sat) | 6.77 | 7.36 | 1146 |
| 8 | 20 | 0.27 | diff.hem.nb (mwravg) | 6.79 | 7.01 | 1137 |
| 9 | 2 | 0.26 | diff.hem.bb (siros) | 6.84 | 6.97 | 1141 |
| 10 | 51 | -0.26 | ch2 stdev (sat) | 6.62 | 6.77 | 1131 |
| 11 | 55 | 0.25 | ch4 stdev (sat) | 6.58 | 6.53 | 1137 |
| 12 | 21 | 0.25 | diff.hem.nb (mwravg) | 6.43 | 5.92 | 1098 |
| 13 | 18 | 0.24 | diff.hem.nb (mwravg) | 6.44 | 5.84 | 1105 |
| 14 | 17 | 0.23 | diff.hem.nb (mwravg) | 6.47 | 5.62 | 1100 |
| 15 | 19 | 0.23 | diff.hem.nb (mwravg) | 6.43 | 5.58 | 1109 |
| 16 | 49 | -0.21 | ch1 stdev (sat) | 6.38 | 5.43 | 1120 |
| 17 | 16 | 0.20 | diff.hem.nb (mwravg) | 6.37 | 5.24 | 1100 |
| 18 | 34 | -0.12 | bar.pres (smos) | 6.23 | 4.38 | 1094 |
| 19 | 22 | -0.12 | direct.norm.nb (mwravg) | 6.25 | 4.24 | 1113 |
| 20 | 23 | -0.12 | direct.norm.nb (mwravg) | 6.07 | 4.13 | 1101 |
| 21 | 24 | -0.12 | direct.norm.nb (mwravg) | 6.02 | 4.03 | 1109 |
| 22 | 3 | -0.12 | direct.norm.bb (siros) | 5.92 | 3.99 | 1110 |
| 23 | 26 | -0.11 | direct.norm.nb (mwravg) | 5.91 | 3.96 | 1115 |
| 24 | 9 | -0.11 | short.direct.normal (siros) | 5.90 | 3.98 | 1123 |
| 25 | 25 | -0.11 | direct.norm.nb (mwravg) | 5.89 | 3.94 | 1127 |
| 26 | 36 | -0.11 | nip (smos) | 5.88 | 3.89 | 1126 |
| 27 | 27 | -0.10 | direct.norm.nb (mwravg) | 5.76 | 3.88 | 1134 |
| 28 | 42 | -0.08 | ir.temp (mwrios) | 5.65 | 3.60 | 1065 |
| 29 | 46 | -0.07 | water.flag.fraction (mwrios) | 5.60 | 3.64 | 1084 |
| 30 | 43 | -0.07 | wet.window (mwrios) | 5.62 | 3.65 | 1084 |
| 31 | 41 | -0.07 | liq (mwrios) | 5.59 | 3.60 | 1082 |
| 32 | 32 | -0.07 | rel humid (smos) | 5.61 | 3.43 | 1070 |
| 33 | 53 | 0.05 | ch3 stdev (sat) | 5.45 | 3.41 | 1067 |
| 34 | 31 | 0.04 | temp (smos) | 5.40 | 3.22 | 1018 |
| 35 | 5 | -0.04 | down.long.diff.hem (siros) | 4.98 | 3.22 | 1009 |
| 36 | 39 | -0.03 | psig (smos) | 4.96 | 3.20 | 1011 |
| 37 | 48 | -0.02 | ch1 mean (sat) | 4.89 | 3.23 | 994 |
| 38 | 6 | 0.02 | up.short.hem (siros) | 4.82 | 3.23 | 1005 |
| 39 | 29 | 0.02 | wspd.va (smos) | 4.81 | 3.24 | 997 |
| 40 | 45 | -0.02 | liq (mwrios) | 4.81 | 3.27 | 989 |
| 41 | 28 | 0.02 | wspd (smos) | 4.80 | 3.28 | 1014 |
| 42 | 4 | 0.02 | up.long.hem (siros) | 4.81 | 3.28 | 1012 |
| 43 | 33 | 0.01 | vap.pres (smos) | 4.70 | 3.22 | 993 |
| 44 | 13 | -0.01 | hem.nb (mwravg) | 4.67 | 3.20 | 996 |
| 45 | 40 | -0.01 | vap (mwrios) | 4.70 | 3.18 | 994 |
| 46 | 47 | 0.01 | water.flag.periods (mwrios) | 4.71 | 3.21 | 1001 |
| 47 | 12 | -0.01 | hem.nb (mwravg) | 4.66 | 3.20 | 1003 |
| 48 | 11 | -0.01 | hem.nb (mwravg) | 4.68 | 3.20 | 1005 |
| 49 | 10 | -0.01 | hem.nb (mwravg) | 4.67 | 3.22 | 1011 |
| 50 | 44 | -0.01 | vap (mwrios) | 4.68 | 3.23 | 1012 |
| 51 | 7 | -0.00 | down.short.hem (siros) | 4.73 | 3.23 | 1013 |
| 52 | 1 | -0.00 | hem.bb (siros) | 4.73 | 3.25 | 1018 |
| 53 | 14 | 0.00 | hem.nb (mwravg) | 4.72 | 3.23 | 1023 |
| 54 | 30 | -0.00 | wdir (smos) | 4.73 | 3.30 | 1024 |
| 55 | 15 | -0.00 | hem.nb (mwravg) | 4.67 | 3.30 | 1042 |
| 56 | 38 | 0.00 | pep2 (smos) | 4.63 | 3.29 | 1046 |
| 57 | 35 | 0.00 | precip (mm) | 4.62 | 3.29 | 1045 |

Table 5.1: Mean squared error determined by adding one sensor at a time to the regression function with order of addition determined by the magnitude of the sensor's correlation coefficient. The linear regression MSE exhibits a constant decrease in MSE with the addition of each sensor. The support vector machine results will be discussed later in the chapter.

| $\lambda$ | MSE | $\rho$ |
|---|---|---|
| 0.0000 | 4.63 | 0.71 |
| 0.0213 | 4.64 | 0.71 |

Table 5.2: MSE for linear regression. $\lambda$ - regularization parameter, $\rho$ - correlation between predicted output and the target value.

## 5.3 Back-Propagation Trained MLP

The back-propagation (BP) with momentum algorithm was implemented using the University of Stuttgart's Neural Network Simulator (SNNS, 1999). This package provides a flexible and fast method for training with many standard neural network algorithms. During experimentation involving trials with different values for the learning parameters, it was found that a relatively small step size, low momentum term, and small value for flat spot elimination yielded the best results. The training values selected are shown in table 5.3.

| parameter | symbol | value |
|---|---|---|
| learning rate | $\eta$ | 0.001 |
| momentum | $\alpha$ | 0.200 |
| flat spot elimination | | 0.100 |

Table 5.3: Training parameters used in the back-propagation with momentum experiment.

Cross-validation was used throughout training in order to gauge the networks' success. Though training error continues to decrease throughout the epochs, validation error begins to creep up around the 50-epoch mark indicating a classic case of over-training of the network as it begins to learn the noise and idiosyncrasies of the data. This phenomenon is illustrated in figure 5.3(a), which shows both the training error (solid trace) and valida-

| architecture | epochs | MSE (std dev) |
|---|---|---|
| mean of 50 networks 2010 | 50 | 4.6742 (0.4686) |
| mean of 50 networks 2010 | 2000 | 5.5623 (0.5688) |
| committee 2010 | 50 | 3.2360 |
| committee 2010 | 2000 | 2.6778 |
| mean of 50 networks 3020 | 50 | 4.8653 (0.3869) |
| mean of 50 networks 3020 | 2000 | 5.9470 (0.4715) |
| committee of 50 networks 3020 | 50 | 3.2220 |
| committee of 50 networks 3020 | 2000 | 2.8144 |

Table 5.4: Back-propagation training results. 2010 refers to a network with two hidden layers of 20 and 10 neurons. 3020 refers to a network with two hidden layers of 30 and 20 neurons. (std dev) is the standard deviation of the MSEs.

tion error (broken trace). Yet another measure of the success in training is the correlation between output and target values. The trace in the same figure (upper box) shows an immediate peak in the correlation coefficient between the predicted value and the target of approximately 0.78 at fifty epochs that gradually falls off. These traces are averaged over fifty networks.

In terms of regression performance, a single network is able to achieve a mean squared error of approximately 4.67 kilometers which is slightly worse than linear regression. However, using fifty trained networks to implement a simple committee machine model via ensemble averaging of the network outputs, a substantial gain in performance is realized. In fact, the regression error on the test set can be dropped over twenty percent with this method. Figure 5.3(b) illustrates how the test error decreases as networks are added to the committee machine. Little benefit results from increasing the number of networks beyond fifty, thus, it is reasonable to accept this number as the optimal number of networks in terms of both error reduction and computational complexity. To see the stabilizing effect

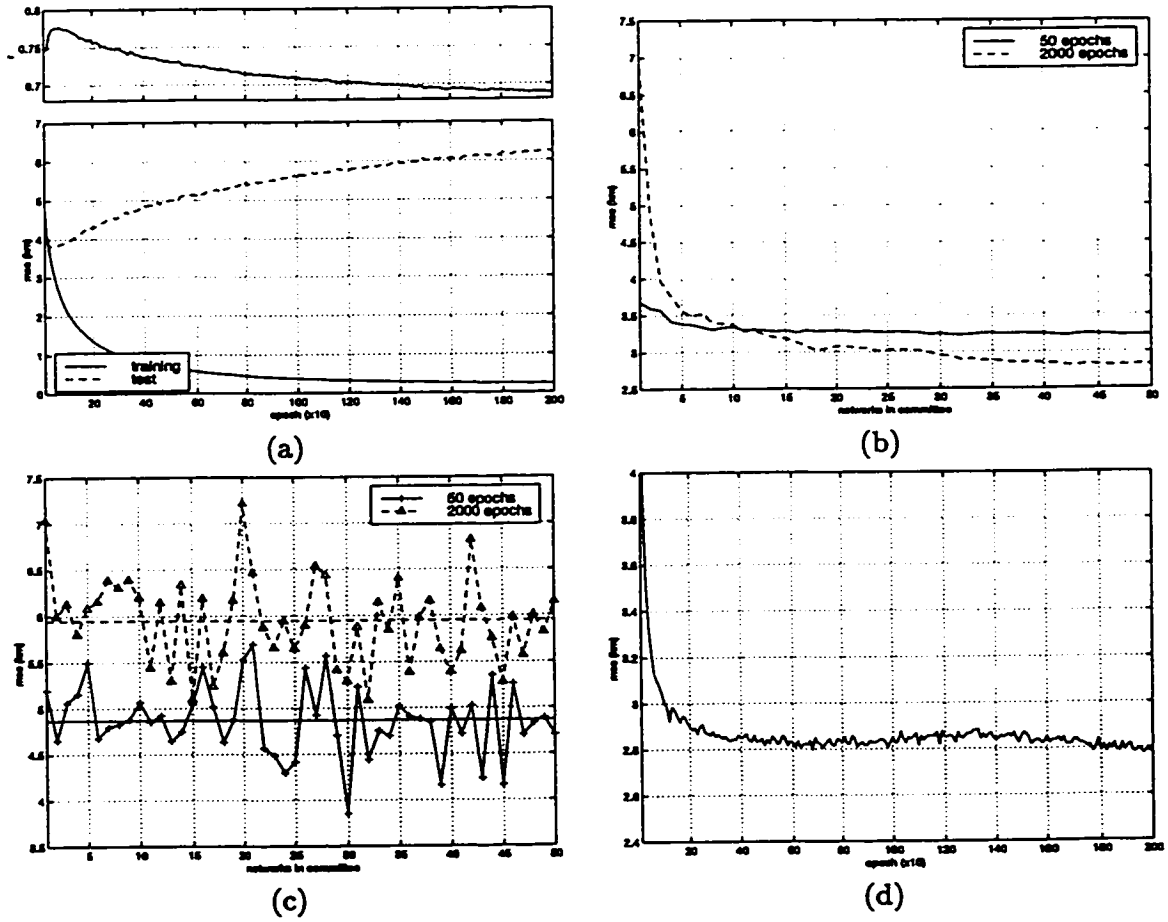



Figure 5.3: Results for back-propagation training - two hidden layers of 30 and 20 neurons. (a) Average training and testing error for the individual networks. Upper box illustrates peaking of the correlation coefficient between the predicted value and its target. (b) Decrease in generalization error as the number of networks in the committee machine increases. (c) Mean squared error of individual networks with means given as straight lines. (d) Decrease in MSE of committee machine with over-training.

of the committee machine, figure 5.3(c) reveals the major downfall of the back-propagation algorithm; susceptibility to entrapment in local minima. Here, the difference in the fifty networks that compose the committee is clearly demonstrated after fifty epochs (solid trace) and two thousand epochs of training (broken trace). Though the two thousand-epoch runs yield a much higher mean MSE per network during generalization, the over-training is actually helpful for committee machines (Haykin, 1999). This point is borne out by the final figure in this series, (d). Here, the efficacy of the committee approach is demonstrated by plotting the committee's MSE against an increasing number of training epochs. There is a marked decrease in MSE as the training extends beyond the logical early stopping point of fifty epochs as determined by cross-validation. Results are presented in table 5.4 where the best result is a MSE of 2.8 km achieved with 50 networks forming a committee after having been trained for 2000 epochs. Since single networks' MSEs have been averaged over 50 networks, the standard deviation has also been given to indicate the variability in performance of individual networks.

## 5.4 Node-Decoupled Extended Kalman Filter Training

The node-decoupled extended Kalman filter trained MLP was configured with the same architecture as used for the BP trained MLP of the previous section. Thus, with two hidden layers of thirty and twenty nodes, there are 2381 weights in the network. As discussed in section 3.4, this number is important as it relates directly to computational complexity. Without the use of the decoupling, each iteration of the algorithm would require the calculation of 5 707 321 partial derivatives. With node-decoupling only 120 581 partial derivative calculations, or 2.11% of those for the GEKF case, are required.

In total, the networks were trained for one hundred and twenty iterations. This

represents a substantially lower number of epochs required for training as compared to the BP trained network. However, it should be kept in mind that the NDEKF algorithm converges much more quickly and again, this constitutes a purposeful over-training of the networks.

Results are presented in the same fashion as were those in the back-propagation case. This facilitates an easy comparison between the two algorithms and in both cases, the trends are identical. Figure 5.4 illustrates this starting with (a) which shows the training and cross-validation errors over the 120 epochs as well as the correlation coefficient. Most notable here is the almost immediate convergence to what would be considered the optimal stopping point in single network training in only five epochs. Pane (b) articulates the increased performance of the committee machine approach with networks trained for 5 and 120 epochs. Here, the additional networks decreased overall MSE . Again, the over-training has a profoundly beneficial effect. This same concept is illustrated in pane (d) where the MSE of the committee machine is plotted against increasing epochs. Finally, figure(c) of this series clearly shows the same susceptibility to local minima that plagued the BP case. Overall, a MSE of 2.69 kilometers is achieved which compares favourably to back-propagation's 2.81 km.

## 5.5 Radial-Basis Function Networks

As noted previously, the training of a RBF network basically involves a one-shot solution of an equation of the form $\mathbf{Ax} = \mathbf{b}$. In this section, in addition to determining the applicability of the RBF network for the sensor fusion task, investigation into kernel selection is performed, with the results clearly showing that kernel selection is an important consideration in RBF design.

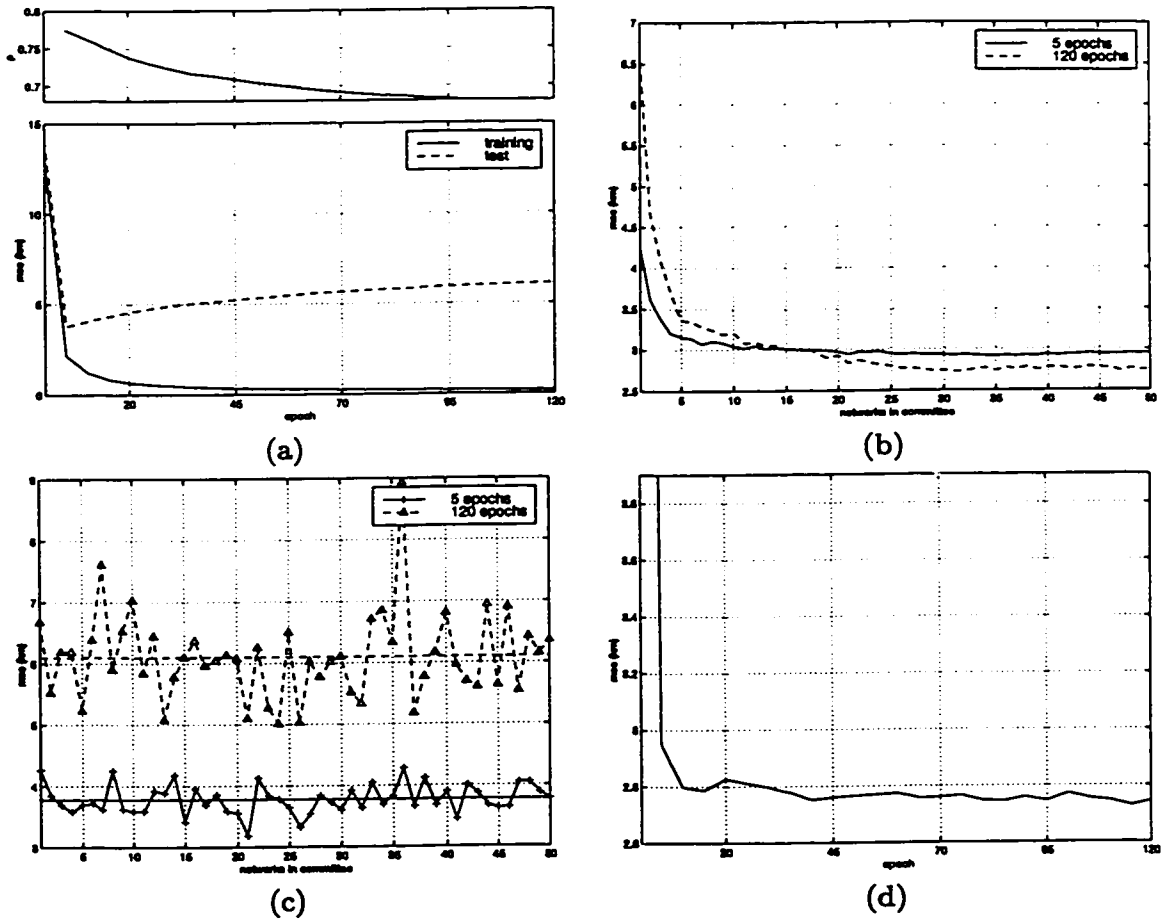



Figure 5.4: Results for NDEKF training - two hidden layers of 30 and 20 neurons. (a) Average training and testing error for the individual networks. The upper box illustrates the immediate peaking of the correlation coefficient between the predicted value and its target. (b) Decrease in generalization error as the number of networks in the committee machine increases. (c) Mean squared error of the individual networks with means given as straight lines. (d) Decrease in MSE of committee machine with over-training.

| architecture | epochs | MSE (std dev) |
|---|---|---|
| mean of 50 networks 2010 | 5 | 3.8778 (0.2492) |
| mean of 50 networks 2010 | 120 | 6.0382 (0.6812) |
| committee of 50 networks 2010 | 5 | 2.9745 |
| committee of 50 networks 2010 | 120 | 2.7867 |
| mean of 50 networks 3020 | 5 | 3.7787 (0.2396) |
| mean of 50 networks 3020 | 120 | 6.0986 (0.7074) |
| committee of 50 networks 3020 | 5 | 2.9511 |
| committee of 50 networks 3020 | 120 | 2.7451 |

Table 5.5: NDEKF results. 2010 refers to a network with two hidden layers of 20 and 10 neurons. 3020 refers to a network with two hidden layers of 30 and 20 neurons. (std dev) is the standard deviation of the MSEs.

| | unregularized | | | regularized | | | |
|---|---|---|---|---|---|---|---|
| $\sigma/N$ | MSE | $\rho$ | $\kappa$ | MSE | $\rho$ | $\kappa$ | $\kappa$ red (%) |
| | | | Gaussian kernel | | | | |
| 1.58 | 9.30 | 0.45 | 2.90e+03 | 5.10 | 0.70 | 2.77e+03 | 95.5 |
| 1.91 | 24.65 | 0.23 | 1.33e+04 | 4.12 | 0.75 | 1.26e+04 | 94.7 |
| 2.24 | 56.30 | 0.12 | 4.73e+04 | 3.99 | 0.76 | 4.56e+04 | 96.4 |
| 2.49 | 53.48 | 0.13 | 1.09e+05 | 4.11 | 0.76 | 1.05e+05 | 96.0 |
| 2.74 | 60.32 | 0.01 | 2.35e+05 | 4.35 | 0.75 | 2.25e+05 | 95.7 |
| | | | square root Gaussian kernel | | | | |
| 0.77 | 4.79 | 0.74 | 8.41e+01 | 4.98 | 0.74 | 6.78e+01 | 19.4 |
| 1.10 | 3.23 | 0.81 | 7.41e+02 | 3.28 | 0.81 | 6.03e+02 | 81.4 |
| 3.30 | 2.90 | 0.83 | 7.17e+04 | 2.91 | 0.83 | 5.86e+04 | 81.7 |
| 17.30 | 2.86 | 0.83 | 2.99e+06 | 2.85 | 0.83 | 2.40e+06 | 80.2 |
| 35.30 | 2.86 | 0.83 | 1.26e+07 | 2.85 | 0.83 | 1.01e+07 | 80.1 |
| 100.00 | 2.86 | 0.83 | 1.01e+08 | 2.85 | 0.83 | 8.12e+07 | 80.1 |

Table 5.6: Radial basis function network results using the standard Gaussian kernel and the square root Gaussian kernel on the interpolated data set. $\sigma$ - bandwidth, $\rho$ - correlation coefficient, $\kappa$ - condition number, $\kappa$ red (%) is the percent reduction in the condition number. (In the case of the modified Gaussian kernel, the value given in column $\sigma$ is actually $N$.)

## Kernel Selection

In section 3.5 it was mentioned that kernel selection is thought to be relatively unimportant in the design of a radial-basis function network. This assertion is refuted by viewing the results contained in table 5.6. Here, using two different kernels in both nonregularized and regularized solutions, results are given for varying bandwidths. The square root Gaussian (SRG) kernel presented in equation 3.41 outperforms the standard Gaussian (SG) kernel (equation 3.40) with a MSE of 2.86 over 3.99. The reasons for this are somewhat difficult to explain. Initially, it would seem that the increased performance could stem from the non-continuous nature of the function to be approximated. Recalling that the values for the cloud-base height are quantized in 300 meter increments, the SRG kernel might provide better localization through its sharper peak as illustrated in figure 3.5. However, performance increases with increasing bandwidth. And eventually, for a large $N$ in equation 3.40 the kernel becomes very nonlocal. Thus, like the multiquadric kernel, this is a case of a nonlocal kernel performing better than a local kernel, which is an argument put forward and verified by Powell (1987).

As well, it can be stated that the SRG's use results in an interpolation matrix that is more descriptive of the underlying problem. In the case of the Gaussian kernel, regularization attenuates the upper singular vectors and significantly improves the solution. However, the SRG kernel performs no better with regularization and thus retains all information contained in the interpolation matrix. This point is illustrated through the viewing of figures 5.6 and 5.5 which show the generalized cross-validation functions, L-curves and filter factors for the Gaussian and square root Gaussian kernels, respectively. In the case of the Gaussian kernel, a pronounced shoulder appears in the L-curve (figure 5.5(a)) revealing a problem that would benefit from regularization. This plot cites an optimal regularization

parameter ($\lambda_{opt}$) of 1.0e-4, which is in close accord with the $\lambda_{opt} = 0.00013$ result given by the clear minimum in the more accurate GCV function illustrated in (b). This value results in an attenuation of the upper singular vectors, as illustrated in pane (c) of the same figure. In contrast, the SRG kernel results in an interpolation matrix that better describes the underlying dynamics. Examination of figure 5.6 shows an L-curve that exhibits no shoulder and a GCV curve without a clear minimum. While the filter factors given by the selected regularization parameter do truncate the upper singular vectors, table 5.6 reveals that, in fact, regularization degrades the solution and that the regularization parameter determined by the GCV algorithm is simply an artifact of the curve's beginning.

The results of this section illustrate a large difference in the generalization ability of RBF networks that is kernel dependent. The Gaussian is the typical choice but in this particular case, however, use of the square root Gaussian kernel with a bandwidth that is quite large making it nonlocal, significantly reduced the MSE .

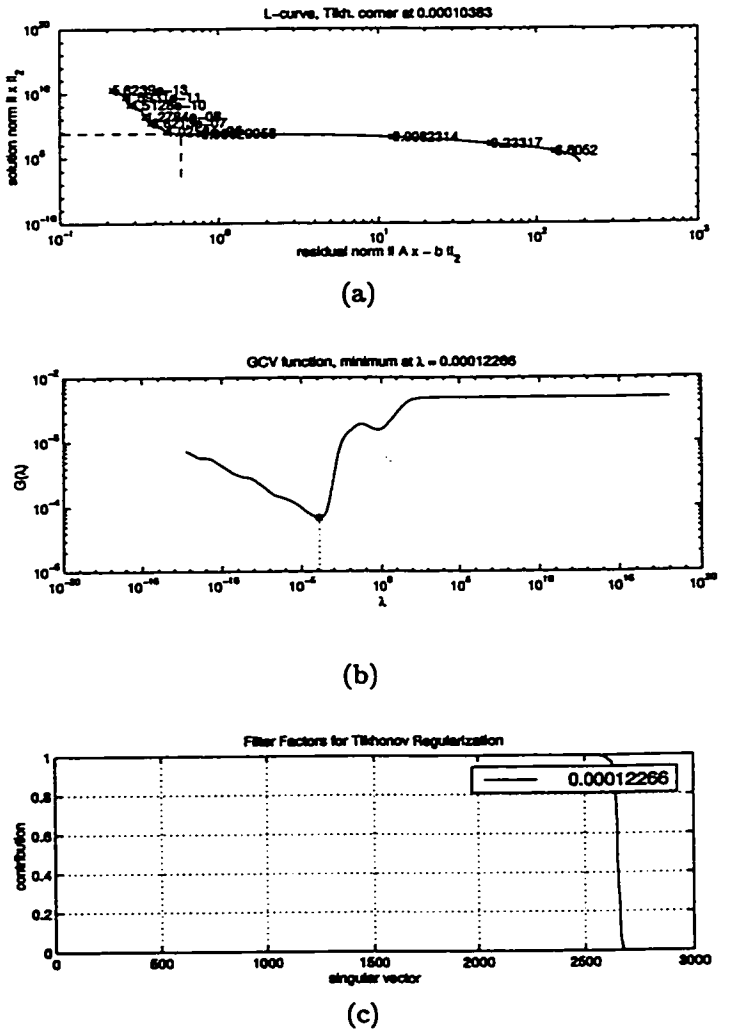
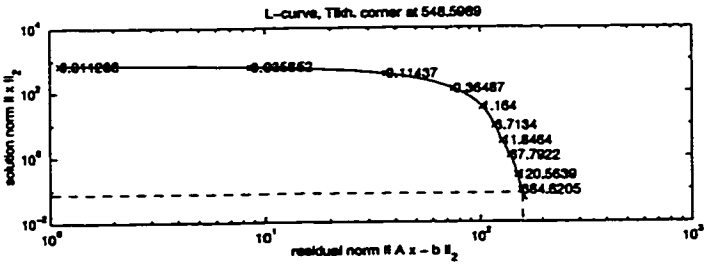
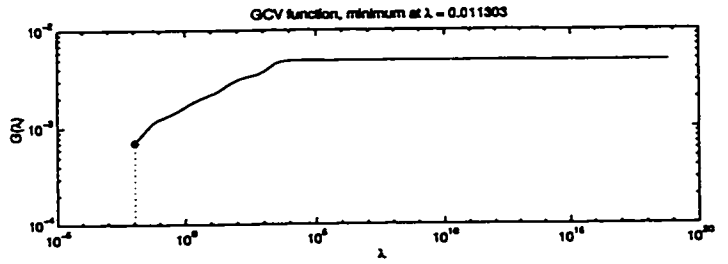


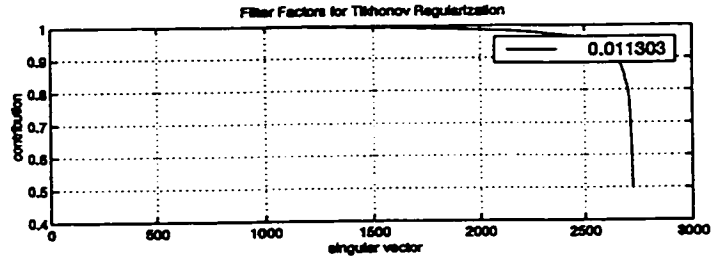

Figure 5.5: Regularization curves for the Gaussian kernel acting on the interpolated data. (a) L-curve showing distinct elbow and optimal trade-off between the solution and residual norms. (b) Generalized cross-validation curve showing distinct minima and hence an optimal value for $\lambda$. (c) Filter factors which show a distrust of the highest singular values.




(a)




(b)




(c)

Figure 5.6: Regularization curves for the square root Gaussian kernel on interpolated data. (a) L-curve showing *no* distinct elbow. (b) Generalized cross-validation curve showing *no* distinct minima. (c) Filter factors. In actuality, the solution is best if no regularization is used and all singular vectors are used to their fullest extent, ie. no filtering.

## 5.6 Support Vector Machine

There are three basic decisions involved in the design of a support vector machine. The first is the operating mode of the SVM and selection of any associated parameters. In this case, RBF mode was selected and consequently, the bandwidth of the centers ($\sigma$ for the Gaussian and $N$ for the SRG) required selection. For these, trial and error are utilized. The second decision is a selection of the rather coarse variable epsilon ($\epsilon$). This controls the approximation accuracy and can simply be started relatively high and reduced until the data can no longer support additional accuracy. (There is however a caveat to this statement which is presented in chapter 6.) Such a condition is evidenced by no further reduction in error. The final variable $C$, however, does have a basis for selection. $C$ controls the tradeoff between tolerating deviations larger than $\epsilon$ and obtaining flatness of the regression function in the high-dimensional space. The argument has been made that since $C$ defines the maximum possible value for the absolute value of the weights, its selection directly affects the output range of the SVM (Mattera and Haykin, 1999). Too small a $C$ limits the output range and too large a value could lead to excessively large weights. Thus, $C$ should be made approximately equal to the maximum of the expected output. Since the maximum value is fourteen kilometers, from the preceding arguments, a value in the neighbourhood of 15 seems reasonable. This is verified experimentally in tables 5.7 and 5.7, where $\epsilon$ is set to one. These tables contain the results for the cases where epsilon equals 1.0 and 0.5 with $C$ varying between 15 and 30. As well, the bandwidth ($\sigma$ for the Gaussian kernel and $N$ for the SRG kernel) are varied. In the tables, boxes indicate the best case MSEs for each parameter group.

In all cases, an epsilon of 25 performed best, though differences between the different epsilon groups were small. This value of 25 is not out of line with the theoretically suggested

value of 15. Most notably however is that unlike the RBF case, there is much less sensitivity to kernel selection. One reason for this is that the optimization process that is inherent to SVM training results in a solution that is unconstrained by prior center selection as in the RBF case. Thus, the SVM has the flexibility to choose the best centers, given any kernel. The performance difference is apparent in looking at the case where the bandwidth is low. For example, consider the case where $C$=15 in table 5.8. In the first line, the number of support vectors is relatively similar for both kernels (1605 and 1612) however, the SRG kernel has a much lower MSE than the Gaussian case: 3.20 and 3.71 respectively.

Overall, the SRG kernel performs slightly better than the Gaussian kernel for a small epsilon; however, the differences are minimal when compared to the radial-basis function case. Thus it can be stated that the support vector machine is somewhat less susceptible to training parameter variation.

| | Gaussian | | | | square root Gaussian | | | |
|---|---|---|---|---|---|---|---|---|
| C | $\sigma$ | MSE | $\rho$ | # SV | $N$ | MSE | $\rho$ | # SV |
| | 8.00 | 4.07 | 0.76 | 1300 | 4.00 | 3.50 | 0.80 | 1281 |
| | 24.00 | 3.41 | 0.80 | 970 | 12.00 | 3.30 | 0.81 | 1086 |
| 15 | 40.00 | 3.34 | 0.80 | 908 | 20.00 | 3.36 | 0.80 | 1025 |
| | 56.00 | 3.38 | 0.80 | 896 | 28.00 | 3.48 | 0.80 | 1006 |
| | 72.00 | 3.43 | 0.80 | 884 | 36.00 | 3.59 | 0.79 | 1003 |
| | 88.00 | 3.47 | 0.79 | 878 | 44.00 | 3.68 | 0.78 | 990 |
| | | | | | 52.00 | 3.79 | 0.78 | 997 |
| | 8.00 | 4.10 | 0.76 | 1313 | 4.00 | 3.55 | 0.80 | 1292 |
| | 24.00 | 3.46 | 0.79 | 979 | 12.00 | 3.31 | 0.81 | 1100 |
| 20 | 40.00 | 3.32 | 0.80 | 914 | 16.00 | 3.27 | 0.81 | 1079 |
| | 56.00 | 3.32 | 0.80 | 890 | 28.00 | 3.37 | 0.80 | 1021 |
| | 72.00 | 3.38 | 0.80 | 893 | 36.00 | 3.46 | 0.80 | 1003 |
| | 88.00 | 3.46 | 0.79 | 880 | 44.00 | 3.53 | 0.79 | 1000 |
| | | | | | 52.00 | 3.62 | 0.79 | 997 |
| | 8.00 | 4.13 | 0.75 | 1324 | 4.00 | 3.54 | 0.80 | 1288 |
| | 24.00 | 3.46 | 0.79 | 997 | 12.00 | 3.27 | 0.81 | 1114 |
| 25 | 40.00 | 3.28 | 0.80 | 925 | 20.00 | 3.26 | 0.81 | 1077 |
| | 56.00 | 3.29 | 0.80 | 895 | 28.00 | 3.33 | 0.81 | 1041 |
| | 72.00 | 3.36 | 0.80 | 896 | 36.00 | 3.36 | 0.80 | 1016 |
| | 88.00 | 3.40 | 0.80 | 892 | 44.00 | 3.45 | 0.80 | 1000 |
| | | | | | 52.00 | 3.49 | 0.80 | 1006 |
| | 8.00 | 4.13 | 0.75 | 1330 | 4.00 | 3.53 | 0.80 | 1294 |
| | 24.00 | 3.48 | 0.79 | 1019 | 12.00 | 3.29 | 0.81 | 1127 |
| 30 | 40.00 | 3.31 | 0.80 | 940 | 20.00 | 3.28 | 0.81 | 1082 |
| | 56.00 | 3.29 | 0.81 | 898 | 28.00 | 3.31 | 0.81 | 1050 |
| | 72.00 | 3.30 | 0.80 | 901 | 36.00 | 3.34 | 0.81 | 1035 |
| | 88.00 | 3.35 | 0.80 | 888 | 44.00 | 3.35 | 0.80 | 1016 |
| | | | | | 52.00 | 3.42 | 0.80 | 1001 |

Table 5.7: Support vector machine operating in RBF mode results for $\epsilon = 1.0$ and the two different kernels.

| | Gaussian | | | | square root Gaussian | | | |
|---|---|---|---|---|---|---|---|---|
| C | $\sigma$ | MSE | $\rho$ | # SV | N | MSE | $\rho$ | # SV |
| | 8.00 | 3.71 | 0.78 | 1605 | 4.00 | 3.20 | 0.82 | 1612 |
| | 24.00 | 3.25 | 0.81 | 1337 | 12.00 | 3.17 | 0.81 | 1467 |
| 15 | 40.00 | 3.37 | 0.80 | 1276 | 20.00 | 3.27 | 0.81 | 1392 |
| | 56.00 | 3.46 | 0.80 | 1255 | 28.00 | 3.41 | 0.80 | 1361 |
| | 72.00 | 3.50 | 0.79 | 1262 | 36.00 | 3.56 | 0.79 | 1355 |
| | 88.00 | 3.59 | 0.79 | 1272 | 44.00 | 3.71 | 0.78 | 1352 |
| | 8.00 | 3.72 | 0.78 | 1615 | 4.00 | 3.22 | 0.81 | 1630 |
| | 24.00 | 3.23 | 0.81 | 1351 | 12.00 | 3.09 | 0.82 | 1493 |
| 20 | 40.00 | 3.28 | 0.81 | 1287 | 20.00 | 3.17 | 0.81 | 1426 |
| | 56.00 | 3.37 | 0.80 | 1279 | 28.00 | 3.29 | 0.81 | 1383 |
| | 72.00 | 3.45 | 0.80 | 1269 | 36.00 | 3.38 | 0.80 | 1362 |
| | 88.00 | 3.50 | 0.79 | 1273 | 44.00 | 3.50 | 0.79 | 1355 |
| | 8.00 | 3.73 | 0.78 | 1630 | 4.00 | 3.21 | 0.81 | 1643 |
| | 24.00 | 3.25 | 0.81 | 1381 | 12.00 | 3.09 | 0.82 | 1514 |
| 25 | 40.00 | 3.22 | 0.81 | 1290 | 20.00 | 3.12 | 0.82 | 1454 |
| | 56.00 | 3.35 | 0.80 | 1292 | 28.00 | 3.19 | 0.81 | 1404 |
| | 72.00 | 3.43 | 0.80 | 1279 | 36.00 | 3.28 | 0.81 | 1376 |
| | 88.00 | 3.46 | 0.80 | 1275 | 44.00 | 3.37 | 0.80 | 1357 |
| | 8.00 | 3.75 | 0.78 | 1642 | 4.00 | 3.22 | 0.81 | 1645 |
| | 24.00 | 3.29 | 0.81 | 1392 | 12.00 | 3.09 | 0.82 | 1531 |
| 30 | 40.00 | 3.28 | 0.81 | 1311 | 20.00 | 3.10 | 0.82 | 1468 |
| | 56.00 | 3.31 | 0.80 | 1292 | 28.00 | 3.17 | 0.81 | 1421 |
| | 72.00 | 3.36 | 0.80 | 1279 | 36.00 | 3.23 | 0.81 | 1395 |
| | 88.00 | 3.43 | 0.80 | 1281 | 44.00 | 3.29 | 0.81 | 1377 |

Table 5.8: Support vector machine operating in RBF mode results for $\epsilon = 0.5$ and the two different kernels.

| range (km) | MSE | (std dev) | # |
|---|---|---|---|
| 0 - 1 | 1.2038 | 1.0275 | 1580 |
| 1 - 2 | 1.0887 | 1.0387 | 330 |
| 2 - 3 | 1.1799 | 1.1786 | 112 |
| 3 - 4 | 1.1382 | 1.1105 | 121 |
| 4 - 5 | 1.6150 | 1.4718 | 163 |
| 5 - 6 | 2.3384 | 1.8323 | 77 |
| 6 - 7 | 2.4374 | 2.0804 | 60 |
| 7 - 8 | 2.6400 | 2.1370 | 82 |
| 8 - 9 | 3.0397 | 2.5192 | 45 |
| 9 - 10 | 3.3348 | 2.7511 | 47 |
| 10 - 11 | 5.0074 | 3.1669 | 51 |
| 11 - 12 | 5.0679 | 3.2029 | 38 |
| 12 - 13 | 5.3202 | 5.7276 | 15 |
| 13 - 14 | 12.3000 | 0 | 3 |

Table 5.9: Evidence for reducing the data set. (std dev) is the standard deviation of the MSEs. # gives the number of pattern vectors contained in each range.

## 5.7 Reduced Set Results

The cloud-base height estimations via fusion of the 57 sensor variables have thus far provided very reasonable results. However, inspection of the distribution of the error with regards to cloud-base height indicates an interesting point. Table 5.9 shows the average MSE and its standard deviation for different bands of cloud-base height. As the height is quantized in 300 meter increments, each one kilometer band will typically contain three different cloud-base heights. The fourth column gives the number pattern vectors in the training set within the stated range. This table reveals that a significant jump in average MSE occurs above the 8 km mark and not surprisingly, this corresponds to the point where the cloud-base heights become severely underrepresented (see figure 4.3). Thus, a logical question is: What would be the MSE if more data were available for the underrepresented regions?

One manner of providing insight into this problem is to rebuild the data set leaving

out measurements above 8 kilometers and then to run the same experiments of the previous sections. Having taken this tack, what follows is a series of tables and figures that reproduce all previous experiments but use the reduced data set.

In all cases, there was a fairly substantial decrease in the MSE as table 5.10 demonstrates. This indicates that the collection of additional data would do well in reducing the MSE and that the estimation via fusion results may certainly be improved. In order to provide continuity of the text, the figures and tables are contained at the very end of this chapter. There were no surprises in that the results followed trends identical to their predecessors. Thus only brief descriptions and the final results are given under each heading.

| Method | Full Set | Reduced Set |
|---|---|---|
| linear | 4.63 | 3.37 |
| BP | 2.81 | 1.43 |
| NDEKF | 2.69 | 1.41 |
| RBF | 2.86 | 1.54 |
| SVM | 3.26 | 1.63 |

Table 5.10: Decrease in MSEs achieved by considering only well represented cloud-base heights. The column "Full Set" gives the best case MSE using the interpolated data set. Column "Reduced Set" gives the best case MSE for the height limited set where no cloud-base height exceeds 8 km.

## Back-Propagation Trained MLPs

Figure 5.7 of training curves replaces figure 5.3 and shows the same benefits with respect to over-training and the formation of the committee machine that were previously noted. Table 5.13 of MSEs for various network configurations replaces table 5.4 and shows that the MSE dropped from 2.81 to 1.43.

### NDEKF Trained MLPs

Again the same relationships previously noted were evident. The decreased training time in epochs and the extremely fast convergence to the cross-validation point when compared to the BP case are apparent. Here, figure 5.8 of training curves replaces figure 5.4 and table 5.14 of MSEs for the two network configurations replaces table 5.5. The MSE decreased from 2.69 to 1.41.

### Regularized Radial-Basis Function Networks

Table 5.15 of MSEs replaces table 5.6 and no exceptions to earlier observations are noted. With the square root Gaussian kernel as the best choice, mean squared error was reduced from 2.86 to 1.72.

### Support Vector Machines

Previously, the support vector machine was run with epsilon equal to 1.0 and 0.5 (tables 5.7 and 5.8 respectively). These are replaced with tables 5.16 and 5.17.

Though the reduction in the number of patterns in the training set was only 7.3% the number of support vectors decreased approximately 25% in most cases. This substantial reduction in support vectors can be attributed to the more well-behaved nature of the data.

### Further Data Set Reduction

The idea of dealing with only well distributed data may be taken further by training on a data set that spans 0-5 km. If this is performed, MSE becomes a very low 0.75 km. This value was achieved using the SVM; however no tables of results are presented here since in placing such a large restriction on the data pushes the problem into the realm of only

approximating the output of a ceilometer.

## 5.8 Computational Complexity

Neural network algorithms are not only evaluated on their performance but also in terms of computational complexity. The prime reason for this is that some learning algorithms require extremely powerful computers in terms of both processing ability and memory to execute the training. These requirements are often brought about by the the *curse of dimensionality*. The RBF network is a good example of this. Thus, here, two main criteria are considered for determining computational complexity. The first involves a measure of network complexity in terms of architecture (*structural complexity*) and in essence deals with the number of weights which are alternately referred to as the free parameters in the model. The second deals with the computational requirements of the algorithm (*algorithmic complexity*). Both of these issues are discussed below.

### Structural Complexity

Structural complexity manifests itself mainly as the number of weights, otherwise known as free parameters, in a network. In the cases of the support vector machine and the radial-basis function network, this number is relatively low and is directly proportional to the size of the data set. A support vector machine, with its mathematical basis in structural risk minimization, has the most parsimonious representation since the number of weights will necessarily be less than the number of training examples. In a RBF network, and in particular for a difficult learning problem that would require strict interpolation as has been used in this thesis, the number of weights will equal the number of examples in the training set. This of course equals the number of patterns in the training set. Thus, for the results of

training on the interpolated data set, the RBF network possesses 2724 weights and a typical run of the SVM, would yield a network with approximately 1100 weights. These values contrast sharply to the BP/NDEKF trained networks. As determined via equation 3.26 the number of weights in one of these networks is 2381. Already this is on par with the RBF network; however, when one considers that 50 of these individual networks are required to achieve adequate performance, this results in a requirement of 119 050 weights.

| architecture | weights |
|---|---|
| BP (3020) 50 networks | 119 050 |
| NDEKF (3020) 50 networks | 119 050 |
| RBF | 2525 |
| SVM | 1353 |

Table 5.11: Weights required in each architecture.

Table 5.11 summarizes this relationship and shows that to achieve comparable network performance, the global methods require over 50 times the computational resources of an RBF and 100 times those of the SVM . Generally, the most parsimonious representation for an acceptable level of error is the most appealing.

## Algorithmic Complexity

Another measure of complexity is the time required to train the networks. This is proportional to the algorithmic complexity. The simplest case to consider is the back-propagation trained MLP. Here, an analytic expression for the number of floating-point operation (FLOPS) required to train on one pattern is available. Using the same notation as for equation 3.26 where a network has layers numbered 1 to $L$, and each layer has $\mathcal{N}(n)$ weights where $n$ is the index of the layer, the following expression may be determined for training on one

pattern.

$$\text{FLOPS} = \overbrace{\sum_{n=1}^{L-1} \mathcal{N}_{n+1}(2\mathcal{N}_n + 2)}^{A} + \overbrace{4\mathcal{N}_L + \sum_{n=2}^{L-2} 2\mathcal{N}_n(\mathcal{N}_{n+1})}^{B} + \overbrace{3\sum_{k=1}^{L-1}(\mathcal{N}(n)+1)\mathcal{N}(n+1)}^{C} \tag{5.1}$$

In the above equation, there are three terms: $A$ expresses the FLOPS required for the feedforward phase, $B$ gives the FLOP count for determination of the local gradients, and $C$ specifies the number of FLOPS required for the actual weight updates.

NDEKF training requires fewer training cycles than does back-propagation training at the cost of greater computational complexity per epoch: $O(N_L M^2)$ versus $O(M)$ where $M$ is the number of weights and $N_L$ in the number of output nodes. Due to the fact that there is only one output unit in the network used in this thesis, the inversion of equation 3.33 becomes a simple scalar division and an analytic expression can be formed for the number of FLOPS required,

$$\text{FLOPS} = \overbrace{\sum_{n=1}^{L-1} \mathcal{N}_{n+1}(2\mathcal{N}_n + 2)}^{A} + \overbrace{3\mathcal{N}_L + \sum_{n=2}^{L-2} 2\mathcal{N}_n(\mathcal{N}_{n+1})}^{B} + \overbrace{3\mathcal{M} + 2 + \sum_{n=2}^{L-2} \mathcal{X}^2(2\mathcal{X}(\mathcal{Y}\mathcal{Z}+\mathcal{X}) + \mathcal{Y}(4\mathcal{Z}+3))}^{C} \tag{5.2}$$

where $\mathcal{X}$ is $\mathcal{N}(n)+1$, $\mathcal{Y}$ is $\mathcal{N}(n+1)$, $\mathcal{Z}$ is $\mathcal{N}(L)$ and $\mathcal{M}$ is the number of weights in the

network. Again there are three terms corresponding to feedforward, partial derivative calculation and update costs. These are annotated $A$,$B$ and $C$, respectively. The feedforward cost is identical to the back-propagation cost and the gradient cost differs only by one FLOP for neurons in the output layer. Thus for the 3020 network, 14 011 754 FLOPS are required as compared to 16 873 in the back-propagation case. One must keep in mind however that the NDEKF converges much more quickly and in the examples presented previously only 120 epochs were required as compared to 2000 when training with backpropagation.

However, such a convenient representation is not available for the two local methods presented. For example, the matrix inversion involved in the RBF network is an iterative process that does not require a definite number of steps but can only be very generally stated as having order, $O(M^3)$. As well, the training of a support vector machine is an optimization procedure with a variable number of steps. Because of this, the only way to evaluate training complexity is to state that the number of operations required is proportional to the amount of CPU time. The Unix command "time" allows a measure of the time used during the command (*real*), the time spent in the system (*sys*), and the time spent in execution of the command (*user*) (Sun Microsystems, 1998). Thus by recording the *user* time, the algorithms may be compared. In table 5.12 a relative performance index, normalized with respect to the training time for the support vector machine, is given.

| architecture | performance index |
|---|---|
| BP (30, 20) 50 networks | 73.0 |
| NDEKF (30, 20) 50 networks | 147.0 |
| RBF | 5.5 |
| SVM | 1.0 |

Table 5.12: Training time required in each architecture.

From both table 5.11 and 5.12 the support vector machine provides the most economic representation and the lowest computational complexity.

## 5.9 Summary

From the results presented earlier, it is quite clear that all four architectures performed roughly on par with each other in terms of their MSEs. This indicates that the data can do little more to support additional accuracy and an overall cloud-base height estimation from the fused data is a mean squared error in the neighbourhood of 1.4-1.6 kilometers. As the original LIDAR output was quantized in 300 meter increments, this means that the predicted output is within $\pm$ 4 levels. This was achieved by reducing the data set to exclude pattern vectors that led to cloud-base heights over 8 km. It is important to note that the data set reduction has only to do with the fact that the cloud-base heights above 8 km are severely underrepresented and given the ill-posedness of the problem and the existence of many-to-one mappings, there is no way, short of increasing the amount of data, to overcome this issue.

To further this point, although more than a year's worth of data is used, weather changes fairly slowly and meteorological studies are generally based on much more data. With these points in mind, it is felt that the accuracy can be increased significantly by extending the study over a greater period of time.

Earlier in the chapter, results for linear regression were presented (table 5.1). In this table a column for a typical run of the support vector machine were also included but not discussed. At this point it is appropriate to mention that the processing gain by using nonlinear neural networks is significant as almost immediately, for a given number of sensors, the SVM overtakes the ability of pure linear regression. In fact, the SVM achieves

a degree of accuracy with only 18 sensor variables that is unobtainable even when linear regression uses all 57 sensors.

In terms of a "best method", at first glance, all of the algorithms seem to have performed similarly and the global methods slightly better than the local algorithms. However, the previous section affords an opportunity to make a distinction based on the computational requirements. In all cases, efforts have been made to minimize the required resources. For the extended Kalman filter trained network, the node-decoupling significantly decreased the number of operations and the memory requirements but the use of a committee machine obviates any gain. For the RBF network, expensive matrix inversion is overcome by the Cholesky back-substitution method but only if regularization is not required. This was the situation only when a certain kernel type was chosen as the standard Gaussian kernel significantly benefited from regularization and thus required the full singular value decomposition. As well, the RBF experienced difficulty in converging with interpolation matrices much larger than those employed and thus could not be used for extremely large problems. Finally, the SVM with its partitioning during the optimization process has a training time that was less coupled to the factors that increased computational demands in the other methods. This fact, in concert with the discussion of the next chapter which demonstrates the illustrative power of the support vectors themselves and what they can reveal about the problem, casts the SVM in a very positive light.

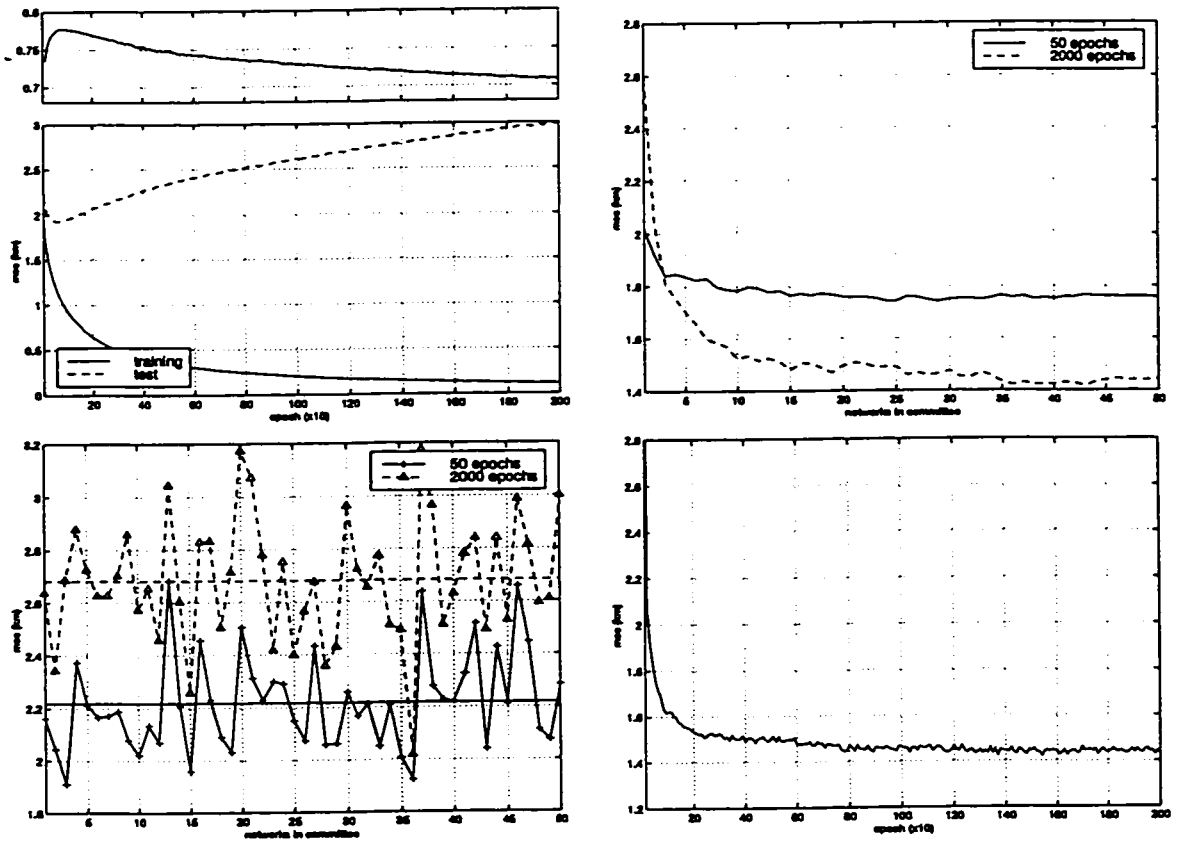



Figure 5.7: Results for back-propagation training - two hidden layers of 30 and 20 neurons on the reduced data set. (a) Overall training and testing error for committee machine. (b)Decrease in generalization error as number of networks in committee machine increases. (c) Increase in individual network error with over-training. (d) Decrease in training error of committee machine as the single networks are over-trained.

| architecture | epochs | MSE (std dev) |
|---|---|---|
| mean of 50 networks 2010 | 50 | 2.2106 (0.1735) |
| mean of 50 networks 2010 | 2000 | 2.7350 (0.2004) |
| committee of 50 networks 2010 | 50 | 1.8143 |
| committee of 50 networks 2010 | 2000 | 1.4255 |
| mean of 50 networks 3020 | 50 | 2.2174 (0.1844) |
| mean of 50 networks 3020 | 2000 | 2.6822 (0.2341) |
| committee of 50 networks 3020 | 50 | 1.7529 |
| committee of 50 networks 3020 | 2000 | 1.4272 |

Table 5.13: Back-propagation trained MLP results for the reduced data set. 2010 refers to a network with two hidden layers of 20 and 10 neurons. 3020 refers to a network with two hidden layers of 30 and 20 neurons. (std dev) is the standard deviation of the MSEs.

| architecture | epochs | MSE (std dev) |
|---|---|---|
| mean of 50 networks 2010 | 5 | 1.9084 (0.0908) |
| mean of 50 networks 2010 | 120 | 2.6895 (0.2259) |
| committee of 50 networks 2010 | 5 | 1.6179 |
| committee of 50 networks 2010 | 120 | 1.4579 |
| mean of 50 networks 3020 | 5 | 1.8450 (0.1405) |
| mean of 50 networks 3020 | 120 | 2.6534 (0.2600) |
| committee of 50 networks 3020 | 5 | 1.5674 |
| committee of 50 networks 3020 | 120 | 1.4138 |

Table 5.14: NDEKF results for reduced data set. 2010 refers to a network with two hidden layers of 20 and 10 neurons. 3020 refers to a network with two hidden layers of 30 and 20 neurons. (std dev) is the standard deviation of the MSEs.

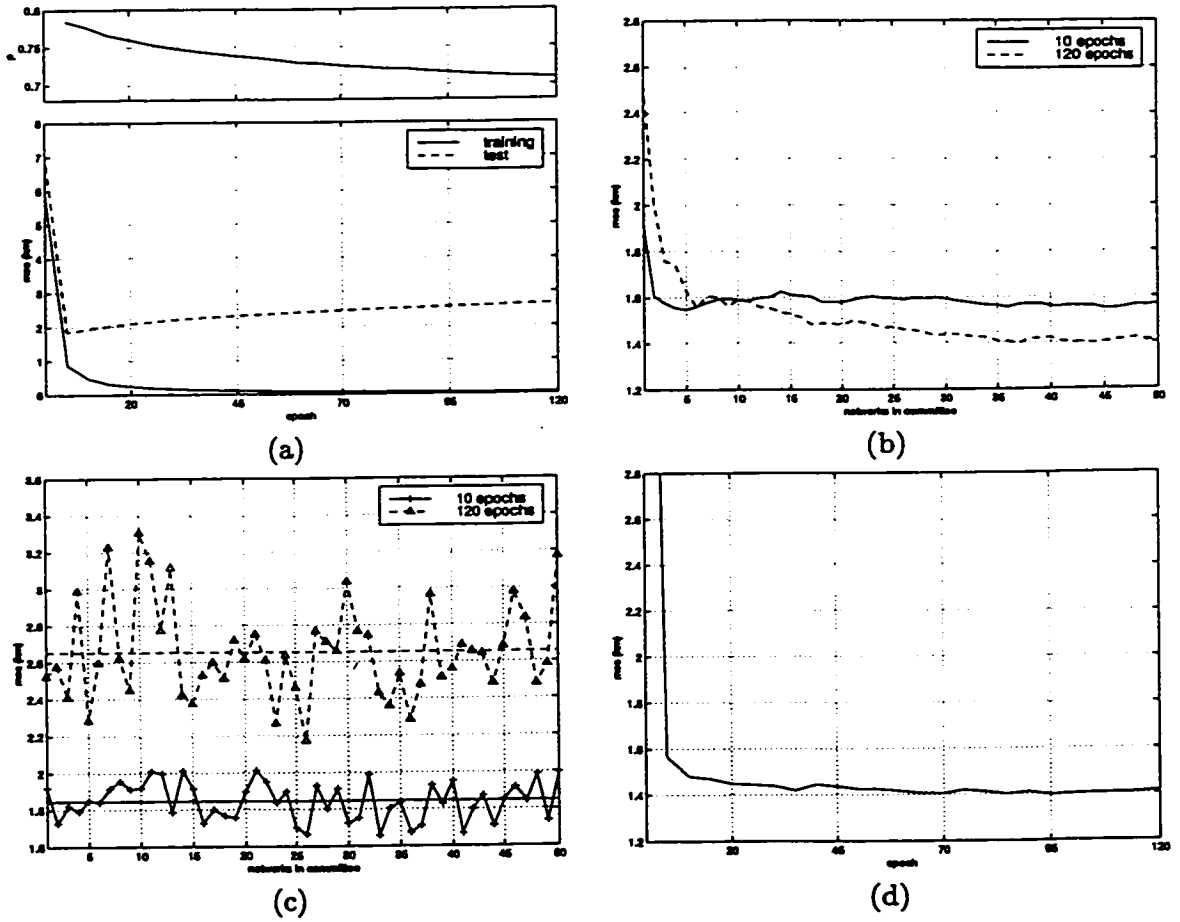



Figure 5.8: Results for NDEKF training - two hidden layers of 30 and 20 neurons on the reduced data set. (a) Overall training and testing error for committee machine. Upper box illustrates peaking of the correlation coefficient between predicted value and target. (b) Decrease in generalization error as number of networks in the committee machine increases. (c) Increase in individual network error with over-training. (d) Training error for one network.

| | unregularized | | | regularized | | | |
|---|---|---|---|---|---|---|---|
| $\sigma$ | MSE | $\rho$ | $\kappa$ | MSE | $\rho$ | $\kappa$ | $\kappa$ red (%) |
| | | | Gaussian kernel | | | | |
| 2.24 | 20.62 | 0.04 | 2.28e17 | 1.96 | 0.77 | 6.42e05 | 0.00 |
| 2.49 | 23.90 | 0.04 | 5.72e17 | 1.87 | 0.78 | 1.83e06 | 0.00 |
| 2.74 | 21.68 | 0.08 | 9.86e17 | 1.86 | 0.78 | 3.68e06 | 0.00 |
| 2.95 | 23.98 | 0.00 | 6.40e18 | 1.85 | 0.78 | 7.41e06 | 0.00 |
| 3.16 | 22.76 | 0.04 | 2.66e18 | 1.92 | 0.78 | 1.49e07 | 0.00 |
| | | | square root Gaussian kernel | | | | |
| 0.77 | 2.74 | 0.71 | 3.08e02 | 2.81 | 0.70 | 1.54e02 | 50.2 |
| 1.10 | 1.81 | 0.79 | 3.00e03 | 1.84 | 0.79 | 1.50e03 | 50.0 |
| 3.30 | 1.56 | 0.82 | 2.99e05 | 1.56 | 0.82 | 1.49e05 | 49.8 |
| 17.30 | 1.54 | 0.82 | 1.25e07 | 1.54 | 0.82 | 5.91e06 | 47.3 |
| 35.30 | 1.54 | 0.82 | 5.27e07 | 1.54 | 0.82 | 2.49e07 | 47.2 |
| 100.00 | 1.54 | 0.82 | 4.24e08 | 1.54 | 0.82 | 2.00e08 | 47.1 |

Table 5.15: Radial basis function network results for the reduced data set using the standard Gaussian kernel and the square root Gaussian kernel. $\sigma$ - bandwidth, $\rho$ - correlation coefficient, $\kappa$ - condition number, $\kappa$ (%) is the percent reduction in the condition number. (In the case of the modified Gaussian kernel, the value given in column $\sigma$ is actually $N$).

| | Gaussian | | | | square root Gaussian | | | |
|---|---|---|---|---|---|---|---|---|
| C | $\sigma$ | MSE | $\rho$ | # SV | $N$ | MSE | $\rho$ | # SV |
| | 8.00 | 2.23 | 0.75 | 1028 | 4.00 | 2.08 | 0.78 | 1064 |
| 15 | 24.00 | 1.81 | 0.79 | 712 | 12.00 | 1.91 | 0.79 | 874 |
| | 40.00 | 1.82 | 0.79 | 650 | 20.00 | 1.90 | 0.79 | 842 |
| | 56.00 | 1.81 | 0.79 | 634 | 28.00 | 1.90 | 0.78 | 827 |
| | 72.00 | 1.82 | 0.78 | 632 | 36.00 | 1.91 | 0.78 | 810 |
| | 88.00 | 1.84 | 0.78 | 626 | 44.00 | 1.94 | 0.78 | 794 |
| | 108.00 | 1.84 | 0.78 | 638 | 52.00 | 1.96 | 0.78 | 787 |
| | 8.00 | 2.22 | 0.75 | 1035 | 4.00 | 2.08 | 0.78 | 1072 |
| 20 | 24.00 | 1.83 | 0.78 | 712 | 12.00 | 1.89 | 0.79 | 888 |
| | 40.00 | 1.80 | 0.79 | 655 | 20.00 | 1.90 | 0.78 | 845 |
| | 56.00 | 1.80 | 0.79 | 633 | 28.00 | 1.88 | 0.79 | 832 |
| | 72.00 | 1.82 | 0.78 | 623 | 36.00 | 1.90 | 0.78 | 824 |
| | 88.00 | 1.82 | 0.79 | 620 | 44.00 | 1.92 | 0.78 | 807 |
| | 108.00 | 1.83 | 0.78 | 620 | 52.00 | 1.93 | 0.78 | 800 |
| | 8.00 | 2.23 | 0.75 | 1039 | 4.00 | 2.08 | 0.78 | 1069 |
| 25 | 24.00 | 1.85 | 0.78 | 717 | 12.00 | 1.90 | 0.79 | 901 |
| | 40.00 | 1.79 | 0.79 | 648 | 20.00 | 1.95 | 0.79 | 872 |
| | 56.00 | 1.80 | 0.79 | 629 | 28.00 | 1.89 | 0.79 | 834 |
| | 72.00 | 1.80 | 0.79 | 624 | 36.00 | 1.88 | 0.79 | 827 |
| | 88.00 | 1.80 | 0.79 | 619 | 44.00 | 1.89 | 0.79 | 819 |
| | 108.00 | 1.81 | 0.79 | 610 | 52.00 | 1.91 | 0.78 | 813 |
| | 8.00 | 2.22 | 0.75 | 1044 | 4.00 | 2.08 | 0.78 | 1069 |
| 30 | 24.00 | 1.84 | 0.78 | 723 | 12.00 | 1.91 | 0.79 | 908 |
| | 40.00 | 1.81 | 0.79 | 660 | 20.00 | 1.89 | 0.79 | 865 |
| | 56.00 | 1.80 | 0.79 | 625 | 28.00 | 1.88 | 0.79 | 844 |
| | 72.00 | 1.80 | 0.79 | 628 | 36.00 | 1.89 | 0.79 | 835 |
| | 88.00 | 1.80 | 0.79 | 627 | 44.00 | 1.89 | 0.79 | 821 |
| | 108.00 | 1.80 | 0.79 | 612 | 52.00 | 1.89 | 0.79 | 819 |

Table 5.16: Support vector machine results for $\epsilon = 1.0$ working on a reduced data set with the two different kernels.

| | Gaussian | | | | square root Gaussian | | | |
|---|---|---|---|---|---|---|---|---|
| C | $\sigma$ | MSE | $\rho$ | # SV | $N$ | MSE | $\rho$ | # SV |
| | 8.00 | 1.89 | 0.78 | 1271 | 4.00 | 1.75 | 0.80 | 1336 |
| | 24.00 | 1.67 | 0.80 | 1025 | 12.00 | 1.67 | 0.81 | 1178 |
| 15 | 40.00 | 1.72 | 0.80 | 979 | 20.00 | 1.68 | 0.80 | 1141 |
| | 56.00 | 1.75 | 0.79 | 968 | | | | |
| | 72.00 | 1.74 | 0.80 | 955 | 36.00 | 1.76 | 0.79 | 1113 |
| | 88.00 | 1.76 | 0.79 | 970 | 44.00 | 1.80 | 0.79 | 1112 |
| | 108.00 | 1.79 | 0.79 | 974 | 52.00 | 1.81 | 0.79 | 1103 |
| | 8.00 | 1.89 | 0.78 | 1284 | 4.00 | 1.75 | 0.80 | 1335 |
| | 24.00 | 1.65 | 0.81 | 1034 | 12.00 | 1.64 | 0.81 | 1190 |
| 20 | 40.00 | 1.71 | 0.80 | 985 | 20.00 | 1.66 | 0.81 | 1153 |
| | 56.00 | 1.74 | 0.80 | 975 | 28.00 | 1.69 | 0.80 | 1132 |
| | 72.00 | 1.74 | 0.80 | 949 | 36.00 | 1.70 | 0.80 | 1122 |
| | 88.00 | 1.75 | 0.80 | 967 | 44.00 | 1.74 | 0.80 | 1111 |
| | 108.00 | 1.76 | 0.79 | 965 | 52.00 | 1.77 | 0.79 | 1111 |
| | 8.00 | 1.90 | 0.78 | 1298 | 4.00 | 1.76 | 0.80 | 1334 |
| | 24.00 | 1.67 | 0.80 | 1055 | 12.00 | 1.64 | 0.81 | 1203 |
| 25 | 40.00 | 1.69 | 0.80 | 992 | 20.00 | 1.63 | 0.81 | 1160 |
| | 56.00 | 1.72 | 0.80 | 985 | 28.00 | 1.67 | 0.80 | 1147 |
| | 72.00 | 1.73 | 0.80 | 952 | 36.00 | 1.69 | 0.80 | 1120 |
| | 88.00 | 1.72 | 0.80 | 955 | 44.00 | 1.70 | 0.80 | 1124 |
| | 108.00 | 1.76 | 0.79 | 953 | 52.00 | 1.73 | 0.80 | 1119 |
| | 8.00 | 1.88 | 0.78 | 1309 | 4.00 | 1.76 | 0.80 | 1332 |
| | 24.00 | 1.67 | 0.80 | 1055 | 12.00 | 1.64 | 0.81 | 1216 |
| 30 | 40.00 | 1.67 | 0.81 | 997 | 20.00 | 1.65 | 0.80 | 1164 |
| | 56.00 | 1.72 | 0.80 | 990 | 28.00 | 1.65 | 0.81 | 1156 |
| | 72.00 | 1.73 | 0.80 | 959 | 36.00 | 1.67 | 0.80 | 1143 |
| | 88.00 | 1.71 | 0.80 | 953 | 44.00 | 1.68 | 0.80 | 1121 |
| | 108.00 | 1.74 | 0.80 | 954 | 52.00 | 1.70 | 0.80 | 1121 |

Table 5.17: Support vector machine results for $\epsilon = 0.5$ working on a reduced data set with the two different kernels.

# Chapter 6

# Support Vector Analysis

## 6.1 What the Support Vectors Tell Us?

In chapter three the basic design of the support vector machine was laid out and the point was made that the SVM is also able to relate information about the underlying problem and not just provide a non-parametric model. This feature is by virtue of the SVM's ability to choose it's own dimensionality. In this chapter, previous SVM results, where the parameters were tuned for maximal performance, have their support vectors analyzed to determine how well-behaved the problem is and to see what information the support vectors can reveal about the underlying physics of cloud formation.

As a starting point, one must geometrically envision the SVM in operation. This is easily accomplished using a simple regression problem involving only single variables $x$ and $y$ which are related via the equation $y = g(x)$. However, it is assumed that $x$ is corrupted with noise and thus the SVM attempts to find a smooth function so that $\hat{y} = f(x)$ is within a degree of accuracy specified by $\epsilon$. Geometrically, this situation is reflected in figure 6.1. Here, the solid line represents the sought after regression function and all points are the

observations $x$.

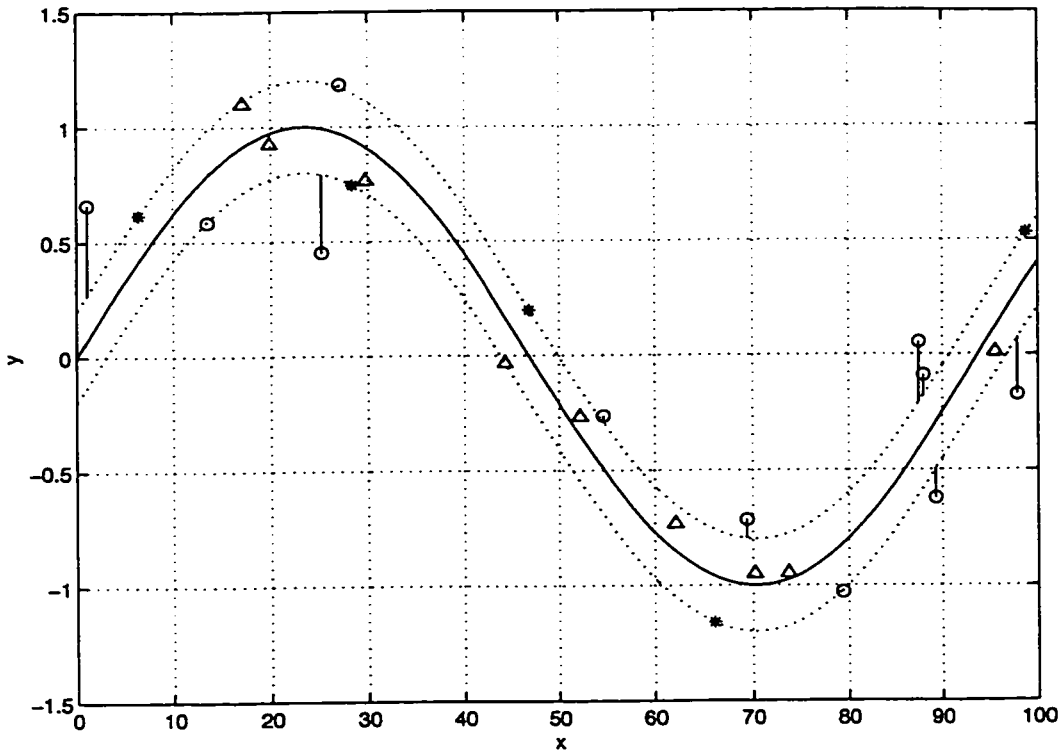



Figure 6.1: Geometric interpretation of the support vector machine's epsilon tube. Triangles are non-support vectors. Circles are support vectors that require a slack variable to bring them to the $\epsilon$-tube. Asterisks are support vectors that lie on the $\epsilon$-boundary.

Now, recalling that when specifying the epsilon parameter, one is really stating the desired accuracy of the solution, a tube may be drawn around the function $f(\cdot)$. This tube possesses sides that are always exactly epsilon away from $f(\cdot)$ and are represented by the dotted line in the figure.

During training, estimates of the observations are computed and an estimate is considered to be a support vector if one of two conditions exist. Condition one involves

the case where a predicted data point touches the epsilon tube (*ex.* the asterisks at $x = 6.4, 28.4, 46.9, 66.1, 98.8$). These points take on a weight value that is less than $C$. Condition two involves the point falling outside the tube thus requiring a slack variable to bring it to the tube's edge in order that it meet the KKT conditions. These points are labeled with circles (*ex.* $x = 1.1, 13.6, 25.2, 27.1, 54.6, 69.4, 79.4, 87.5, 88.0, 89.3, 97.9$) and result in a corresponding weight value that is equal to $C$. The other points, which are represented as triangles, are points that meet the KKT conditions (without the use of a slack variable) and result in a value $\hat{y}$ which is less than $\epsilon$ away from the regression function. Such points are not considered support vectors and can be said to be "well-behaved".

In viewing figure 6.1, it is fairly easy to conceive two extremes. These are an unnaturally high value for epsilon and an epsilon close to zero. In the former case, optimization would cease almost immediately as a very poor estimate of the regression function could satisfy the condition of keeping all estimates within the wide epsilon tube and the KKT conditions would be easily met without the involvement of any slack variables. In such a case, the number of resulting support vectors would be very low. On the other hand, if $\epsilon$ were very small, additive noise would keep many points outside the epsilon tube. In actual fact, the choice of $\epsilon$ hinges on the type of noise in the data. For example, if the noise follows a Laplacian distribution (one with long tails), it can be shown that $\epsilon = 0$ is the best choice and thus the $L_1$ loss function would be optimal (Schölkopf et al., 1998). It is also of interest to note that the choice of a small $\epsilon$ results in an optimization process that would take much longer and would involve a good number of the slack variables $\xi_i$ and $\xi_i'$ to balance the right hand sides of equations 3.51 and 3.52.

As an illustrative example, the weights of several support vectors resulting from a typical run are given in table 6.1. Here it is demonstrated that support vectors that touch the tube and hence are exactly $\epsilon$ away from the regression function possess a weight where

$|w| < C$ and those points that require a slack variable or sit on the epsilon boundary possess a weight equal to $\pm C$.

With this geometric understanding, one can now ask: What information does the SVM yield about the cloud-base height estimation problem? First, by looking at the number of support vectors and the relative proportions on and outside the tube, one may gauge the success of the regression function. That is to say, that if the number of support vectors is reasonable with respect to the data set and the majority of support vectors are on the tube, the resulting function and choice of epsilon accurately reflect the underlying physics of the problem. If, on the other hand, the majority of support vectors are outside the tube, then, it may be stated that the regression function as determined by the SVM is a poor approximation and that the data cannot support the desired epsilon. Furthermore, by running the support vectors back through the machine, one can determine the degree to which the support vectors are in fact outside the tube. This is a further indication of the degree of fit of the resulting regression function.

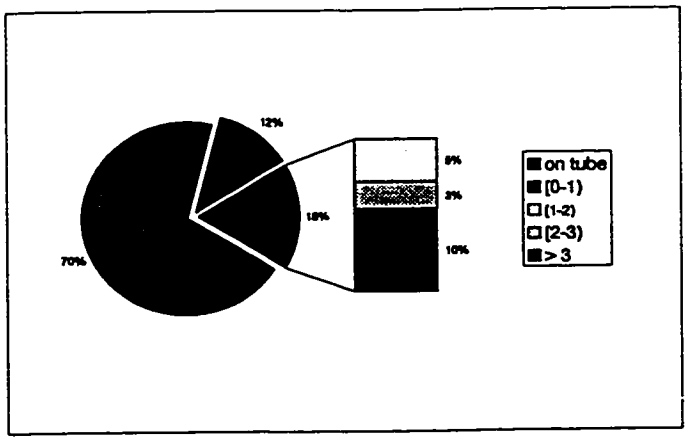
| weight | distance from target value | position WRT epsilon tube |
|---|---|---|
| -20.0000 | 0.9968 | above tube |
| 1.5907 | -0.2500 | on lower boundary of tube |
| 16.6381 | -0.2500 | on lower boundary of tube |
| -1.1434 | 0.2500 | on upper boundary of tube |
| 9.7809 | -0.2500 | on lower boundary of tube |
| -1.7165 | 0.2500 | on upper boundary of tube |
| 20.0000 | -0.2524 | just below tube |
| 2.7641 | -0.2500 | on lower boundary of tube |
| 0.8189 | -0.2500 | on lower boundary of tube |
| 20.0000 | -0.5130 | below tube |
| 20.0000 | -1.8115 | far below tube |
| -6.5404 | 0.2500 | on upper boundary of tube |

Table 6.1: Sampling of support vectors for a run where $\epsilon = 0.25$, $N = 32$ and $C = 20$ using the Gaussian kernel.

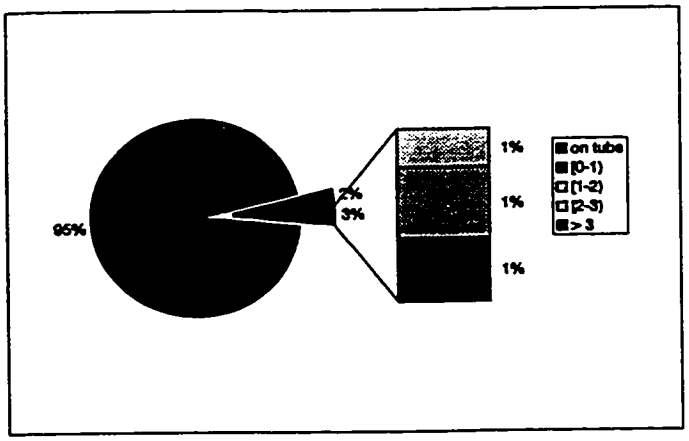
Analyzing the support vectors from a run where, $C = 20$, $N = 32$ on the interpolated data set results in figure 6.2(a). Figure 6.2(b) analyzes the support vectors when training with the same parameters on the reduced data set. These figures reinforce the decision made towards the end of chapter five to use a reduced data set as it leads to a problem that is remarkably more well-behaved and is evidenced by the fact that in figure 6.2(b), the number of support vectors decreased significantly (from 949 to 647) and the proportion of support vectors that lie outside the tube has also dropped remarkably from thirty percent to 5 percent. Moreover, of the five percent that are outside the tube, the majority are quite close - lying within 1 kilometer. This indicates that the sensors selected in chapter three fuse nicely to yield a model for cloud-base height estimation.

Interesting observations result when looking at the breakdown of the support vectors for height ranges. Table 6.2 gives the number of training examples (column 3) for cloud-base height ranges. Column 4 recounts the number of examples that are retained as support vectors and columns 5 to 9 give percentages of support vectors on the $\epsilon$-tube and then in increasing distances from the tube. This same information is represented pictorially in figure 6.3. Clearly the majority of patterns occur in the [0-1) range. As mentioned earlier, this is due to the fact that clear sky observations carry a cloud-base height of 0 km. Again, the underrepresentation above 8 km is evident. Particularly in looking at the bar graph, it is apparent that the most difficulty lies in predicting cloud bases in three height ranges. These are [0-1), [5-6) and [7-8). The first range is explained due to the clear sky observations and the fact that the variables leading to clear sky and a cloud base of, say, 0.5 km are remarkably different however, the output variable of the network is continuous. Thus, this is a data organization issue. The other two ranges are a little more interesting. In discussion with meteorologists, these other two difficult ranges may be explained as follows.

An alternate way of stating that the SVM has difficulty with the [5-6) and [7-8) is




(a)




(b)

Figure 6.2: Support vector breakdown, $\epsilon = 1$ (a) full data set (b) reduced data set.

to say that the routine shows skill in fusing the selected variables into a prediction for the ranges [0-5) and [6-7). General cloud statistics show that, isolated single layer clouds have a thickness of 1-2 km. Two very common types of clouds are boundary layer stratus which have bases in the 0.5 to 2.5 km range and cirrus which possess cloud bases in the 6-7 km range. Since table 5.1 shows that in the top 16 variables, there are eight representatives from the satellite, the algorithm is drawing an association between cloud-top height given by the satellite and statistical average of cloud thickness. This association however does not hold in the regions [5-6) and [7-8) as evidenced by the decreased performance in these ranges. Thus, for a better predictor, additional information is required.

The final table (table 6.3) presents an analysis of the support vectors resulting from training on the height limited data set. Here, epsilon is reduced to 0.25. In doing so, the number of support vectors required in all ranges jumps. As well, the number of support vectors on the $\epsilon$-tube decreases, which indicates that more and more slack variables are coming into play. In viewing the three tables, it seems most reasonable to declare $\epsilon = 0.5$ as the best choice. This claim is justified by the observation that a further decrease in epsilon does not improve the MSE and increases the number of support vectors. Thus, the regression function can be said to be modeling the noise at this point and the network is essentially overtrained. A choice of $\epsilon = 0.5$ however results in the lowest MSE and most parsimonious model.

## Summary

In chapter five, the decreased sensitivity of the SVM to kernel selection and to some extent, training parameter variation, was studied. As well, low memory requirements and greatly reduced training times cast the method in a positive light. In this chapter, analysis of the support vectors permits an intuitive feel for the data and the level of accuracy it can

| $\epsilon$ | height | num pats | % SVs | % on tube | % [0-1) | % [1-2) | % [2-3) | % 3 < |
|---|---|---|---|---|---|---|---|---|
| | [0-1) | 1580 | 25.57 | 72.77 | 16.34 | 5.94 | 2.72 | 2.23 |
| | [1-2) | 330 | 28.18 | 87.10 | 9.68 | 3.23 | 0.00 | 0.00 |
| | [2-3) | 112 | 33.04 | 94.59 | 2.70 | 2.70 | 0.00 | 0.00 |
| | [3-4) | 121 | 33.88 | 92.68 | 2.44 | 4.88 | 0.00 | 0.00 |
| | [4-5) | 163 | 44.17 | 94.44 | 2.78 | 2.78 | 0.00 | 0.00 |
| | [5-6) | 77 | 54.55 | 61.90 | 19.05 | 4.76 | 11.90 | 2.38 |
| 1.0 | [6-7) | 60 | 63.33 | 76.32 | 10.53 | 0.00 | 7.89 | 5.26 |
| | [7-8) | 82 | 68.29 | 58.93 | 10.71 | 14.29 | 3.57 | 12.50 |
| | [8-9) | 45 | 71.11 | 56.25 | 6.25 | 0.00 | 9.38 | 28.12 |
| | [9-10) | 47 | 78.72 | 51.35 | 13.51 | 5.41 | 2.70 | 27.03 |
| | [10-11) | 51 | 84.31 | 37.21 | 9.30 | 4.65 | 4.65 | 44.19 |
| | [11-12) | 38 | 94.74 | 19.44 | 19.44 | 2.78 | 5.56 | 52.78 |
| | [12-13) | 15 | 100.00 | 0.00 | 6.67 | 6.67 | 6.67 | 80.00 |
| | [13-14) | 3 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 |

Table 6.2: Full data set by height range.

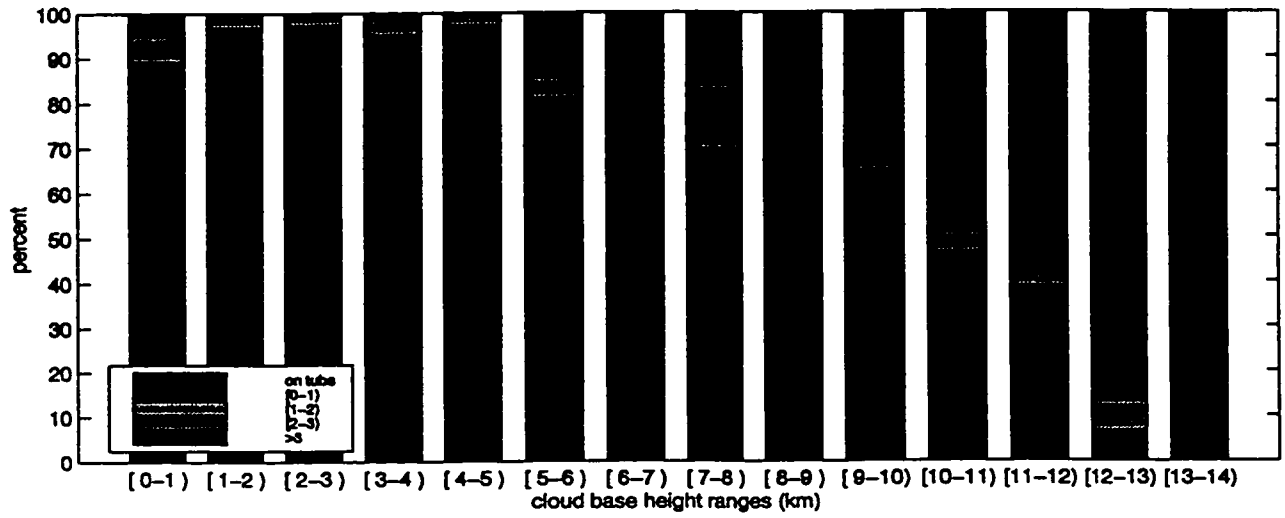



Figure 6.3: Bar graph of the information contained in table 6.2. Decreased performance in the ranges [0-1), [5-6) and [7-8) is apparent.

| $\epsilon$ | height | num pats | % SVs | % on tube | % (0-1) | % [1-2) | % [2-3) | % 3 < |
|---|---|---|---|---|---|---|---|---|
| | [0-1) | 1580 | 22.7 | 98.6 | 0.8 | 0.3 | 0.0 | 0.3 |
| | [1-2) | 330 | 24.9 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | [2-3) | 112 | 22.3 | 96.0 | 4.0 | 0.0 | 0.0 | 0.0 |
| 1.00 | [3-4) | 121 | 33.9 | 92.7 | 4.9 | 2.4 | 0.0 | 0.0 |
| | [4-5) | 163 | 45.4 | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | [5-6) | 77 | 58.4 | 77.8 | 6.7 | 4.4 | 11.1 | 0.0 |
| | [6-7) | 60 | 66.7 | 87.5 | 5.0 | 0.0 | 2.5 | 5.0 |
| | [7-8) | 82 | 80.5 | 78.8 | 7.6 | 1.5 | 4.6 | 7.6 |
| | [0-1) | 1580 | 28.7 | 88.7 | 8.8 | 1.3 | 0.7 | 0.4 |
| | [1-2) | 330 | 42.7 | 90.1 | 8.5 | 0.0 | 1.4 | 0.0 |
| | [2-3) | 112 | 58.9 | 84.9 | 13.6 | 1.5 | 0.0 | 0.0 |
| 0.50 | [3-4) | 121 | 52.9 | 84.4 | 10.9 | 3.1 | 1.6 | 0.0 |
| | [4-5) | 163 | 65.0 | 82.1 | 17.0 | 0.9 | 0.0 | 0.0 |
| | [5-6) | 77 | 67.5 | 57.7 | 11.5 | 11.5 | 5.8 | 13.5 |
| | [6-7) | 60 | 76.7 | 73.9 | 8.7 | 4.4 | 2.2 | 10.9 |
| | [7-8) | 82 | 86.6 | 54.9 | 5.6 | 8.5 | 9.9 | 21.1 |
| | [0-1) | 1580 | 42.2 | 87.1 | 10.3 | 1.7 | 0.5 | 0.5 |
| | [1-2) | 330 | 62.4 | 84.0 | 15.1 | 0.0 | 1.0 | 0.0 |
| | [2-3) | 112 | 71.4 | 77.5 | 17.5 | 3.8 | 1.3 | 0.0 |
| 0.25 | [3-4) | 121 | 69.4 | 79.8 | 15.5 | 2.4 | 1.2 | 1.2 |
| | [4-5) | 163 | 77.9 | 81.1 | 13.4 | 4.7 | 0.8 | 0.0 |
| | [5-6) | 77 | 87.0 | 56.7 | 17.9 | 10.5 | 1.5 | 13.4 |
| | [6-7) | 60 | 83.3 | 68.0 | 14.0 | 2.0 | 4.0 | 12.0 |
| | [7-8) | 82 | 87.8 | 47.2 | 13.9 | 4.2 | 6.9 | 27.8 |

Table 6.3: Height reduced set training results by range for various degrees of epsilon.

support. When coupled with domain knowledge, the support vectors are able to demonstrate relationships that are known or suspected and also identify areas that require more investigation.

# Chapter 7

# Conclusion

## Contributions

With the experiments concluded, these final pages summarizes the contributions of this work. The initial goal was to create a virtual sensor via sensor fusion techniques utilizing data that already exist. This goal has been met and several interesting observations and conclusions have been made along the way. Thus, success may be claimed on several fronts. Overall, the concept of creating something (the virtual sensor), for no additional cost save computational resources is extremely profound. This point and other key components of this thesis are discussed below.

1. *Creation of a virtual sensor.* Without actual deployment of LIDAR instruments, this work has provided a method to produce wide-spread LIDAR output.

2. *Accuracy of estimates.* In speaking with meteorologists, general surprise was expressed at the closeness of the estimates provided by fusing the multisensor data. The results have been deemed useful by meteorologists and can provide a new dimension

to climate prediction models as a description of the vertical distribution of clouds via this technique permits accounting for the radiation modulating properties of clouds. This is something that is absent from current climate modeling techniques and could enhance them significantly.

3. *Uniqueness of problem.* This research has used neural networks to tackle a tough learning problem. In particular, it has used a very complex, highly-nonlinear, real-life data set to take the first positive step in providing a solution to the question posed at the outset. This work has never been attempted previously.

4. *Demonstrated the effectiveness of the nonlinear learning paradigm.* The neural networks' ability to discover the complex nonlinear relationships between the elements has been demonstrated and the results have clearly shown neural networks to provide far superior estimates of cloud-base height than do purely linear methods.

5. *Machine learning discoveries.* Several interesting observations have been noted. In particular, kernel selection has been shown to be an important consideration in RBF network design. As well, the support vector machine has demonstrated a robustness to parameter and kernel variation.

6. *The data mining ability of the SVM has been explored.* A common complaint of neural networks is their "black box" nature. This work has explored the explanatory ability of the support vector machine. Through this process, certain explanations of physical phenomena, such as why certain cloud-base heights are easier to predict than others, have arisen. On the converse side it has prompted challenging research questions such as: What are the more complex interactions that lead to a decreased skill in predicting certain regions of cloud-base height?

7. ***Compared the current state-of-the-art learning techniques in terms of ability and computational complexity.*** Previously the point was made that the remote-sensing literature has been largely devoid of neural network regression applications. While this thesis has brought neural networks into this arena, it has also provided an excellent case study that can be a guide for the application of neural networks to any problem as it has compared the very best machine-learning algorithms in terms of performance and computational complexity.

# Bibliography


Abidi, M., 1989. "Sensor fusion: a new approach and its application." In *Proceedings of the SPIE: Sensor Fusion II Human and Machine Strategies*, pp. 235–246.

Anderson, B. and J. Moore, 1979. *Optimal Filtering*. Prentice Hall.

Atkinson, P. and A. Tatnall, 1997. "Neural networks in remote sensing." *International Journal of Remote Sensing*, 18(4):699–709.

Bankert, R., 1994. "Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network." *Journal of Applied Meteorology*, 33:322–333.

Battan, L., 1983. *Fundamentals of Meteorology*. Prentice-Hall.

Battiti, R., 1994. "Using mutual information for selecting features in supervised neural net learning." *IEEE Transactions on Neural Networks*, 5(4):537–550.

Baum, B., T. Uttal, M. Poellot, T. Ackerman, J. Alvarez, J. Intrieri, D. Starr, J. Titlow, V. Tovinkere, and E. Clothiaux, 1995. "Satellite remote sensing of multiple cloud layers." *J. Atmos. Sci.*, 52(23):4210–4230.

Benediktsson, J., P. Swain, and O. Ersoy, 1990. "Neural network approaches versus statistical methods in classification of multisource remote sensing data." *Transactions on Geoscience and Remote Sensing*, 28:540–552.

Benediktsson, J., P. Swain, and O. Esroy, 1993. "Conjugate gradient neural networks in classification of multisource and very-high-dimensional remote sensing data." *International Journal of Remote Sensing*, 14:2883–2903.

Bhatia, G., 1994. "Practical surface patch registration technique." In *Proceedings of the SPIE: Sensor Fusion VII*, volume 2355, pp. 135–146.

Bullock, T., 1982. "Electroreception." *Annual Review of Neurosciences*, 5:121.

Chavez, G. and R. Murphy, 1993. "Exception handling for sensor fusion." In *Proceedings of the SPIE: Sensor Fusion VI*, volume 2059, pp. 142–153.

Chen, T. and R. Rao, 1998. "Audio-visual integration in multimodal communication." *Proceedings of the IEEE*, 86(5).

Clark, J. and A. Yuille, 1990. *Data Fusion for Sensory Information Processing Systems*. Kluwer Academic Publishers.

Cover, T. and J. Thomas, 1991. *Elements of Information Theory*. Wiley.

Dasarathy, B., 1990. "Paradigms for information processing in multisensor environments." In *Proceedings of the SPIE: Sensor Fusion III*, volume 1306, pp. 69–80.

Dasarathy, B., 1997. "Sensor fusion potential exploitation – Innovative architectures and illustrative applications." *Proceedings of the IEEE*, 85(1):24–38.

Deco, G. and D. Obradovic, 1996. *An Information-Theoretic Approach to Neural Computing*. Spriner-Verlag.

Dempster, A., 1968. "A generalization of Bayesian inference." *Journal of the Royal Statistical Society*, 30:205–247.

Devroye, L., L. Gyorfi, and G. Lugosi, 1996. *A Probabalistic Theory of Pattern Classification*. Springer-Verlag.

dit Neuville, S., 1996. "Intelligence fusion." In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*.

Downey, I., 1992. "A comparison of Landsat Thematic Mapper land cover classification based on neural network techniques and traditional maximum likelihood algorithms and minimum distance algorithms." In *Proceedings of the Annual Conference of the Remote Sensing Society*, pp. 518–528. Nottingham: Remote Sensing Society.

Eggermont, J., 1990. *The Correlative Brain*. Springer-Verlag.

Foody, G. M., 1995. "Using prior knowledge in artificial neural network classification with a minimal training set." *Internation Journal of Remote Sensing*, 16:310–312.

Gelfand, J., M. Flax, R. Endres, S. Lane, and D. Handleman, 1992. "Acquisition of automatic activity through practice: changes in sensory input." In *Proceedings of the Tenth National Conference on Artificial Intelligence*.

Gelfand, J., D. Handleman, S. Lane, and S. Epstein, 1993. *Adapting human functional architectures and behaviours for intelligent machines*, volume 9. Handbook of Neuropsychology.

Golub, G. and C. V. Loan, 1990. *Matrix Computations*. The Johns Hopkins University Press, second edition.

Hagan, M., H. Demuth, and M. Beale, 1996. *Neural Network Design*. PWS Publishing Company.

Hagiwara, M., 1992. "Theoretical derivation of momentum term in back-propagation." In *International Joint Conference on Neural Netowrks*, volume 9, pp. 39–71.

Hall, D., 1990. *Intelligent Data Fusion*, chapter Techniques for identity fusion. Artech House.

Hall, D., 1992. *Mathematical Techniques for Multisensor Data Fusion*. Artech House Inc.

Hall, D. and J. Llinas, 1997. "An introduction to multisensor data fusion." *Proceedings of the IEEE*, 85(1):6–23.

Hamadeh, A., P. Cinquin, and S. Lavallee, 1994. "Anatomy based multimodal medical image registration for computer-integrated surgery." In *Proceedings of the SPIE: Sensor Fusion VII*, volume 2355, pp. 178–188.

Hansen, P., 1992. "Analysis of discrete ill-posed problems by means of the L-curve." *SIAM Review*, 34:561–580.

Hansen, P. C., 1994. "Regularization tools." *Numerical Algorithms*, 6:1–35.

Hansen, P. C. and D. P. O'Leary, 1993. "The use of the L-curve in the regularization of discrete ill-posed problems." *SIAM J. Sci Comput.*, 14:1487–1503.

Hartline, P., L. Kass, and M. Loop, 1978. "Merging of modalities in the optic tectum: infrared and visual integration in rattlesnakes." *Science*, 199:1225–1229.

Haykin, S., 1999. *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

Heirstrand, D., 1983. "An automated threat value model." In *Proceedings of the 50th MORS*.

Hepner, G., T. Logan, N. Ritter, and N. Bryant, 1990. "Artificial neural network classification using a minimal training set: comparison to conventional supervised classification." *Photogrammetric Engineering and Remote Sensing*, 56:469–473.

Houghton, J., L. M. Filho, B. Callander, N. Harris, A. Kattenberge, and K. Maskell, 1996. *Climate Change 1995: The Science of Climate Change*. Published for the Intergovernmental Panel on Climate Change by the Cambridge University Press, Cambridge, England.

Huntsberger, T. and B. Jawerth, 1993. "Wavelet based sensor fusion." In *Proceedings of the SPIE: Sensor Fusion VI*, volume 2059, pp. 488–497.

Iyengar, S., I. Prasad, and H. Min, 1995. *Advances in Distributed Sensor Technology*. Prentice Hall.

Keerthi, S., S. Shevade, C. Bhattacharyya, and K. Murthy, 1999. "Improvements to platt's smo algorithm for svm classifier design." Technical Report CD-99-16, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, http://guppy.mpe.nus.edu.sg/ mpessk/smo_mod.ps.gz.

Klassner, F., V. Lesser, and H. Nawab, 1993. "Fusing multiple reprocessing of signal data." In *Proceedings of the SPIE: Sensor Fusion VI*, volume 2059, pp. 466–475.

Knudsen, E., S. du Lac, and S. Esterly, 1987. "Computational maps in the brain." *Annual Review of Neuroscience*, 10:41–65.

Kohonen, T., 1995. *Self-Organizing Maps*. Springer Verlag.

Kokar, M. and J. Tomasik, 1994. "Towards a formal theory of sensor/data fusion." Technical Report COE-ECE-MMK-1/94, Northeastern University, 330 Snell Building, 360 Huntington Avenue, Boston Massachusetts 02115.

Lau, L., 1993. "A model for sensory fusion."

Liang, X. and W. Wang, 1997. "Cloud overlap effects on general circulation model climate simulations." *J. Geophys. Res.*, 102(D10):11039–11047.

Lucas, M., 1996. "Virtual sensor concept for Eco-Informa 96." In *Proceedings of Eco-Informa 96*, volume 11, pp. 983–988.

Luo, R. and R. Gonzalez, 1992. *Data Fusion in Robotics and Machine Intelligence*, chapter Data Fusion and Sensor Integration: State-of-the-art 1990s, pp. 7–136. Academic Press.

Lutgens, F., 1982. *The atmosphere*. Prentice-Hall.

Marks, L., 1978. *The Unity of the Senses: Interractions Among the Modalities*. New York: Academic Press.

Mattera, D. and S. Haykin, 1999. *Advances in Kernel Methods: Support Vector Learning*, chapter Support Vector Machines for Dynamic Reconstruction of a Chaotic System, pp. 211–242. The MIT Press.

Meredith, M. and B. Stein, 1986. "Visual, auditory, and somatosensory convergence on cells in superior colliculus results in multisensory integration." *Journal of Neurophysiology*, 56(3):640–662.

Moran, K., B. Martner, M. Post, R. Kropfli, D. Welsh, and K. Widener, 1998. "An unattended cloud-profiling radar for use in climate research." *Bull. Amer. Meteor. Soc.*, 79(3):443–455.

Muller, K., A. Smola, G. Ratsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, 1999. *Advances in Kernel Methods: Support Vector Learning*, chapter Using support vector machines for time series prediction, pp. 243–254. The MIT Press.

Nahin, P. and J. Pokoski, 1980. "NCTR plus sensor fusion equals IFFN." In *IEEE Proceedings on Aerospace Electronic Systems*, volume AES-16, pp. 320–327.

Ormoneit, D. and T. Hastie, 1999. "Optimal kernel shapes for local linear regression." Technical report, Department of Computer Science, Technische Universitat Munchen.

Osuna, E., 1999. *Advances in Kernel Methods: Support Vector Learning*, chapter Reducing run-time complexity in support vector machines, pp. 271–284. The MIT Press.

Ozer, N., B. Engel, and J. Simon, 1994. "Ad adaptive technique for multi-sensor fusion and classification of fruit." Technical Report ASAE Paper No. 94, ASAE 2950 Niled Rd. MI.

Peck, C., J. Baro, and S. Warder, 1993. "Sensory integration in the deep layers of the superior colliculus." *Progress in Brain Research*, 95:91–102.

Poggio, T. and F. Girosi, 1990. "Networks for approximation and learning." *Proceedings of the IEEE*, 78:1481–1497.

Powell, M., 1987. "Radial basis function approximations to polynomials." In *Numerical Analysis 1987 Proceedings*, pp. 233–241. Dundee, UK.

Proceedings MFI, 1994. *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.*

Proceedings MFI, 1996. *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.*

Puskorious, G. and L. Feldkamp, 1991. "Decoupled extended Kalman filter training of feedforward layered networks." In *Proceedings of the International Joint COnference on Neural Networks*, volume 1, pp. 771–777.

Ramanathan, V., R. Cess, E. Harrison, P. Minnis, B. Barkstrom, E. Ahmed, and D. Hartmann, 1989. "Cloud-radiative forcing and climate: Results from the earth radiation budget experiment." *Science*, pp. 57–63.

Rossow, W. and L. Garder, 1993. "Cloud detection using satellite measurements of infrared and visible radiances for isccp." *J. Climate*, 6:2341–2369.

Sawaragi, T., J. Umemura, O. Kitai, and S. Iwai, 1994. "Integrating multiple knowledge sources using genetic algorithm applied to hierarchically structured sensor data." In *IEEE International Conference on MultiSensor Fusion and Integration for Intelligent Systems*, pp. 395–402.

Schiffer, A. and W. Rossow, 1983. "The international satellite cloud climatology project (ISCCP): The first project of the world climate research programme." *Bull. Amer. Meteor. Soc*, 64:779–784.

Schölkopf, B., C. Burges, and A. Smola, 1999. *Advances in Kernel Methods: Support Vector Learning*, chapter Introduction to Support Vector Learning, pp. 1–16. The MIT Press.

Schölkopf, B., A. Smola, R. Williamson, and P. Bartlett, 1998. "New support vector algorithms." Technical Report NC2-TR-1998-031, ESPRIT Working Group in Neural and Computational Learning II, http://www.neurocolt.com.

Schweiger, A. and J. Key, 1997. "Estimating surface radiation fluxes in the Arctic from TOVS HIRS and MSU brightness temperatures." *International Journal of Remote Sensing*, 18:955–970.

Shafer, G., 1976. *A Mathematical Theory of Evidence*. Princeton University Press.

Singhal, S. and L. Wu, 1989. *Advances in Neural Information Processing Systems 1*, chapter Training multilayer perceptrons with the extended Kalman algorithm, pp. 133–140. Morgan Kaufmann.

Slingo, A., 1990. "Sensitivity of the earth's radiation budget to changes in low clouds." *Nature*, 343:49–51.

SNNS, 1999. "Stuttgart neural network simulator." ftp://ftp.informatik.uni-stuttgart.de/pub/snns/, Fakultaet Informatik, University of Stuttgart, Germany.

Spinhirne, J., 1993. "Micro pulse lidar." *IEEE Transactions on Geoscience and remote Sensing*, 31(1):48–55.

Stephens, G., S. Tsay, J. Stackhouse, and P. Flatau, 1990. "The relevance of the microphysical and radiative properties of cirrus clouds to climate and climatic feedback." *J. Atmos. Sci.*, 47:1742–1753.

Stephens, G. and P. Webster, 1981. "Clouds and climate: Sensitivity of simple systems." *J. Atmos. Sci.*, 38:235–247.

Stokes, G. and S. Schwartz, 1994. "The atmospheric radiation measurement (ARM) program: Programmatic background and design of the cloud and radiation test bed." *Bull. Amer. Meteor. Soc.*, 75:1201–1221.

Stubenrauch, C., A. D. Genio, and W. Rossow, 1997. "Implementation of subgrid cloud vertical structure inside a gcm and its effect on the radiation budget." *J. Climate*, 10:273–287.

Sun Microsystems, 1998. "Manual page for the *time* command." *Solaris 2.5.1*.

Tian, L. and J. Curry, 1989. "Cloud overlap statistics." *J. Geophys. Res.*, 94(D7):9925–9935.

Trees, H. V., 1968. *Detection, Estimation and Modulation Theory*. J. Wiley and Sons, New York.

Vapnik, V., 1998. *Statistical Learning Theory*. John Wiley and Sons.

Welch, R., 1992. "Polar cloud and surface classification using AVHRR imagery: an intercomparison of methods." *Journal of Applied Meteorology*, 31:405–420.

Yuhas, B., M. Goldstein, T. Sejnowski, and R. Jenkins, 1990. "Neural network models of sensory intergration for improved vowel recognition." *Proceeding of the IEEE*, 78(10):1658–1667.