A Probabilistic Frame-work for Classification of Dermatoscopic Images

Ph.D. Thesis

Mads Hintz-Madsen

LYNGBY October 29, 1998

IMM-PHD-1998-

IMM

ISSN 0909–3192

Printed by IMM, Technical University of Denmark

Abstract

This Ph.D. thesis deals with developing methods for an objective and cost-effective tool for diagnosing skin lesions based on digitized dermatoscopic images. This involves feature extracting techniques and the design of robust neural network classifiers.

The main topics covered in the thesis are:

- Distinct visual features in dermatoscopic images of skin lesions are extracted and described. These include measuring skin lesion asymmetry, describing the transition of pigmentation from the skin lesion to the surrounding skin and measuring the color distribution within the skin lesion.

- A probabilistic framework for classification is defined. This includes optimal decision rules, error-reject theory, derivation of error functions, model complexity control and assessment of generalization performance. A novel part of the framework is the inclusion of an outlier model capable of reducing undesired effects from outliers.

- Methods for designing neural network classifiers are proposed. This includes novel methods for estimating regularization parameters and outlier model parameters. An important part is the removal of the inherent redundancy in the commonly used softmax normalization scheme.

- The effects of the outlier model is investigated on an artificial classification problem. It is shown that is possible to estimate the outlier probability and that undesired effects from outliers can be suppressed resulting in robust classifiers and more accurate empirical generalization error estimates.

- Neural network classifiers are designed for the skin lesion classification problem. It is shown that 73.2% of benign skin lesions and 75.0% of malignant skin lesions in a test set are detected. As a result of the neural classifier design process, it is possible to rank the extracted dermatoscopic features according to their importance. The three most important features are found to be shape asymmetry and the amount of blue and black color present within a skin lesion.

While the developed feature extraction techniques are closely linked to the skin lesion application, the probabilistic framework and the neural classifier design methods may be applied to a wide range of pattern recognition tasks.

Resumé

(Abstract in Danish)

Nærværende Ph.D. afhandling omhandler udvikling af metoder til at foretage en objektiv og simpel diagnostisering af dermatoskopiske modermærke billeder. Dette indebærer metoder til *feature* udtrækning og til design af neurale netværk til robust klassifikation.

Hovedemnerne i afhandlingen er:

- Karakteristiske visuelle *features* i dermatoskopiske billede af modermærker udtrækkes og beskrives. Dette indebærer måling af et modermærkes asymmetri, beskrivelse af pigmenteringens overgang fra modermærke til den omkringliggende hud og beskrivelse af farvefordelingen i modermærker.

- Et fundament til klassifikation baseret på sandsynlighedsprincipper defineres. Dette inkluderer optimale beslutningsregler, *error-reject* teori, udledning af fejl funktioner, kompleksitetskontrol af modeller og måling af generaliseringsevne. Nyt er en *outlier* model, som er i stand til at undertrykke uønskede påvirkninger fra *outlier'ere*.

- Metoder til design af neurale netværk til klassifikation foreslås. Dette inkluderer nye metoder til at estimere regulariseringsparametre og *outlier* sandsynlighed. En vigtig del er definitionen af en modificeret *softmax* normalisering, som fjerner den indbyggede redundans, der er tilstede i den sædvanlige *softmax* implementation.

- *Outlier* modellens indvirken undersøges med et kunstigt genereret klassifikationsproblem. Det vises, at det er muligt at estimere *outlier* sandsynligheden, og at uønsket indflydelse fra *outlier'ere* undertrykkes. Det giver et mere robust klassifikationsværktøj og mere præcise empiriske generaliseringsfejlsestimater.

- Neurale netværk designes til modermærke problemstillingen. Det vises, at 73.2% af benigne modermærker og 75.0% af maligne modermærker detekteres i et test sæt. En sidegevinst ved design processen er, at det er muligt at sortere de udtrukne dermatoskopiske *features* efter betydning. De tre vigtigste *features* viser sig at være asymmetri og det relative areal af blå og sort farve i et modermærke.

De udviklede *feature* udtrækningsteknikker er tæt knyttet til modermærke applikationen, mens klassifikationsdelen er anvendelig for en bred vifte af klassifikationsproblemer.

Preface

This thesis serves as partial fulfillment of the requirements for the Ph.D. degree. The work has been carried out at the Department of Mathematical Modelling at the Technical University of Denmark and was supervised by associate professor Lars Kai Hansen and associate professor Jan Larsen. The work was initiated in March 1995 and completed in October 1998.

Acknowledgments

I would like to express my gratitude to the many people who, in one way or another, have helped with the work documented in this thesis. I would especially like to thank Lars Kai Hansen and Jan Larsen for their supervision and support as well as M.D. K.T. Drzewiecki, Rigshospitalet, M.D. Eric Olesen, Vejle Sygehus, and photographer Kirstine Saad, Rigshospitalet, for their initiation of and involvement with the malignant melanoma project.

I'd would also like to thank fellow Ph.D. student Lars Nonboe Andersen for his participation in the development of the neural classifier framework and the people at the Section for Digital Signal Processing including my former roommates Ulrik Kjems, Niels Mørch and Peter Alshede Philipsen for providing lots of fun and inspiring moments.

Lyngby, October 29, 1998.

Mads Hintz-Madsen

Nomenclature

The notation used throughout this text generally follows the standard conventions of mathematics and statistics. All symbols are introduced when encountered in the text, thus, only a few general notational conventions are mentioned here.

All vectors are column vectors and are denoted by boldface lowercase letters, e.g.,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [x_1 \; x_2 \; \cdots \; x_n]^{\mathrm{T}},$$

where $^{\mathrm{T}}$ denotes the vector/matrix transpose. Matrices are denoted by boldface uppercase letters, e.g.,

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix},$$

where m and n is the number of rows and columns in $\mathbf{X}$, respectively.

A few commonly used notational symbols are listed below:

$\delta_{i,j}$	Kronecker delta: $\delta_{i,j} = 1$, if $i = j$, otherwise $\delta_{i,j} = 0$
$\delta(x)$	Dirac delta: $\delta(x) = 0$ for $x \neq 0$. $\int \delta(x) dx = 1$
$p(\cdot)$	probability distribution
$P(\cdot)$	probability
$\langle \cdot \rangle_{p(\cdot)}$	expectation w.r.t. the probability distribution $p(\cdot)$
log	logarithm to base e

Contents

Chapter 1

Introduction

This chapter provides a short introduction to classification systems, lists the main objectives with the work documented in this thesis and presents an overview of the contents of the remaining chapters and appendices.

1.1 Components of a classification system

A large part of this thesis deals with automatic classification or *pattern recognition* with the task of determining which class an object belongs to based on some of its properties. Pattern recognition applications encompass a wide range of problems including character and speech recognition, fault detection and medical diagnosis. The main application considered in this work concerns the classification of dermatoscopic images of skin lesions and stems from the medical diagnosis field. This application will be introduced in chapter 2.

A typical classification system may be divided into 3 sub-tasks as shown in figure 1.1. These are,

Preprocessing The main purpose of data preprocessing is to facilitate the feature extraction process. For image data, the preprocessor could consist of, e.g., noise suppression, removal of unwanted objects, dimensionality reduction or color-space transformation.

Feature extraction After appropriate preprocessing, a feature extractor is employed in order to describe characteristic properties of the data. Again using image data as an example, characteristic features could be, e.g., color information, shape descriptors or texture properties.

Classification The final task consists of the actual classifier. It is presented with the extracted features of an object and decides which class the object is likely to belong to. The classifier could be, e.g., a parametric model, a k nearest neighbor classifier or a probabilistic neural network.

For a classification system to be successful, all 3 sub-tasks must be performed with care. In this thesis, all 3 tasks are addressed.

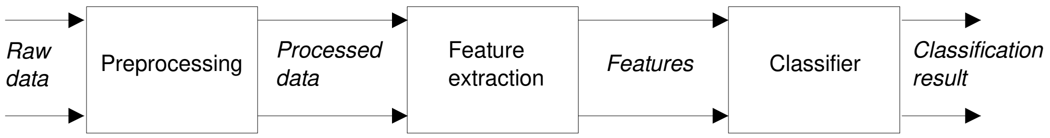

Figure 1.1: An example of a classification system consisting of a preprocessing part, a feature extraction part and a classification part.

1.2 Main objectives

There are three main objectives with the work presented in this thesis. They are as follows,

Develop methods for an objective and cost-efficient tool for diagnosing skin lesions
: This involves extracting relevant information from dermatoscopic images in the form of dermatoscopic features[1] and designing reliable classifiers.

Gain insight into the importance of dermatoscopic features
: The importance of dermatoscopic features is still very much a matter of research. Any additional insight into this area is desirable.

Develop a robust probabilistic neural classifier design framework
: In order to obtain reliable classification systems based on neural networks, a probabilistic approach robust against outliers[2] should be followed.

These objectives should be kept in mind when reading the remainder of this thesis.

1.3 Thesis overview

The thesis is organized into seven chapters and seven appendixes. The contents of the remaining chapters and the appendices are briefly reviewed:

Chapter 2 introduces the malignant melanoma problem. The historic evolution of this deadly form of skin cancer as well as the two most commonly used imaging techniques for recording skin lesions are presented with focus on the dermatoscopic imaging technique. Distinct visual characteristics present in dermatoscopic images are described.

Chapter 3 describes the process of extracting and describing distinct dermatoscopic features from digitized dermatoscopic images. The image processing techniques involved including techniques for image preprocessing and image segmentation are presented. Results of the the feature extraction/description process are shown.

Chapter 4 defines a probabilistic framework suitable as a foundation for modeling and evaluating non-linear classifiers. The use of a probabilistic approach enables us to derive Bayes decision rules and sound cost functions based on *maximum a posterior* parameter estimation. Methods for controlling model complexity in form of parameter priors and pruning techniques are presented along with methods for assessing a

[1]The different dermatoscopic features will be introduced in section 2.4.
[2]The concept of outliers will be introduced in section 4.5.

models generalization performance by using empirical and algebraic generalization error estimates. A key element of the probabilistic framework is the definition of an outlier model capable of reducing undesired effects of possible outliers.

Chapter 5 is concerned with modeling feed-forward multi-layer perceptrons for classification tasks based on the probabilistic framework defined in chapter 4. This includes the definition of a model architecture with a modified softmax output normalization ensuring that model outputs may be interpreted as posterior probabilities and that the inherent redundancy in the standard softmax normalization is avoided. Methods for estimating network weights, appropriate model complexity and outlier model parameters are derived.

Chapter 6 presents results for neural classifiers designed for solving an artificial classification problem with focus on the beneficial effects of the outlier model. Results are also presented for neural classifiers designed for solving the malignant melanoma classification problem. Through network pruning, the importance of the individual dermatoscopic features are estimated.

Chapter 7 summarizes important conclusions drawn from the work presented in the thesis and suggests directions for future work.

Appendix A shows the detailed derivations of the first and second derivatives of the error function with incorporated outlier model w.r.t. the network weights for a modified softmax normalized neural classifier. The Gauss-Newton approximation to the second derivative is also derived.

Appendix B shows the detailed derivation of the gradient of the validation error w.r.t. the model complexity parameters and the outlier model parameters.

Appendices C-G contain reprints of papers authored and co-authored during the Ph.D. study.

Chapter 2

The malignant melanoma problem

In this chapter the malignant melanoma problem will be introduced. We will look at the historic evolution of this deadly form of skin cancer as well as introduce image acquisition techniques used for diagnostic purposes. In particular the dermatoscopic imaging technique will be presented along with a description of distinct visual characteristics present in images acquired with this technique.

2.1 Malignant melanoma

Malignant melanoma is the deadliest form of skin cancer and arises from cancerous growth in pigmented skin lesions. The cancer can be removed by a fairly simple surgical incision if it has not entered the blood stream. It is thus vital that the cancer is detected at an early stage in order to increase the probability of a complete recovery. Skin lesions may be grouped into three classes:

- *Benign nevi* is a common name for all healthy skin lesions. These have no increased risk of developing cancer.
- *Atypical nevi* are also healthy skin lesions but have an increased risk of developing into cancerous lesions. The special type of atypical nevi called *dysplastic nevi* have the highest risk and are, thus, often referred to as precursors of malignant melanoma.
- *Malignant melanoma* are as already mentioned cancerous skin lesions.

When a dermatologist inspects a skin lesion and finds it suspect, the dermatologist will remove the skin lesion and have a biopsy performed in order to determine the exact type of skin lesion. If the lesion is found to be malignant, a larger part of the surrounding skin will be removed depending on the degree of malignancy. If a lesion is not considered to be suspect, it is usually not removed unless there is some cosmetic reason to do so.

It is not an easy task for dermatologists visually to determine whether a skin lesion is or might be malignant, though. A study at *Karolinske Hospital, Stockholm, Sweden* has shown that newly educated dermatologists with less than 1 year of experience detects 31% of the melanoma cases they are presented with while dermatologists with more than 10 years of experience are able to detect 63% [Lindelöf and Hedblad, 1994]. Another study shows that experienced dermatologist are capable of detecting 75% of cancerous skin lesions [Koh et al., 1989].

Malignant melanoma is usually only seen in Caucasians.

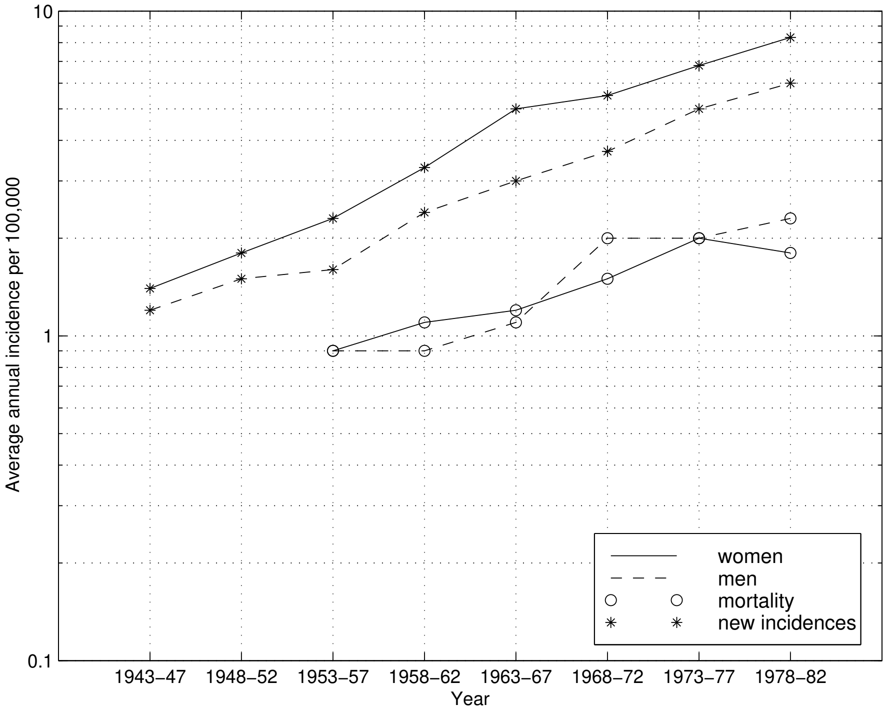

Figure 2.1: The development of the average annual incidence and mortality rates of malignant melanoma in Denmark from 1943 to 1982 [Østerlind, 1990].

2.2 Evolution of malignant melanoma

The incidence of malignant melanoma in Denmark has increased 5- to 6-fold from 1942 to 1982 while the mortality rate has been doubled from 1955 to 1982 [Østerlind, 1990]. Currently, approximately 800 cases of malignant melanoma are reported in Denmark every year. In Germany $9000-10000$ new cases are expected every year with an annual increase of $5-10\%$ [Rassner, 1988].

Due to the rather steep increase in the number of reported malignant melanoma cases, it is becoming increasingly important to develop methods capable of diagnosing malignant melanoma that are simple, objective and preferably non-invasive. Today the only accurate diagnostic technique is a biopsy and a histological analysis of the skin tissue sample. This is an expensive procedure as well as an uncomfortable experience for the patient. For patients with many skin lesions or *dysplastic nevus syndrome*[1], this is clearly not a feasible diagnostic technique. Contributing to the problem is the increasing awareness of skin cancer among the general public. People are consulting dermatologists more often which again calls for a simple and accurate diagnostic technique.

[1]People with *dysplastic nevus syndrome* have multiple dysplastic nevi - often dozens or even hundreds.

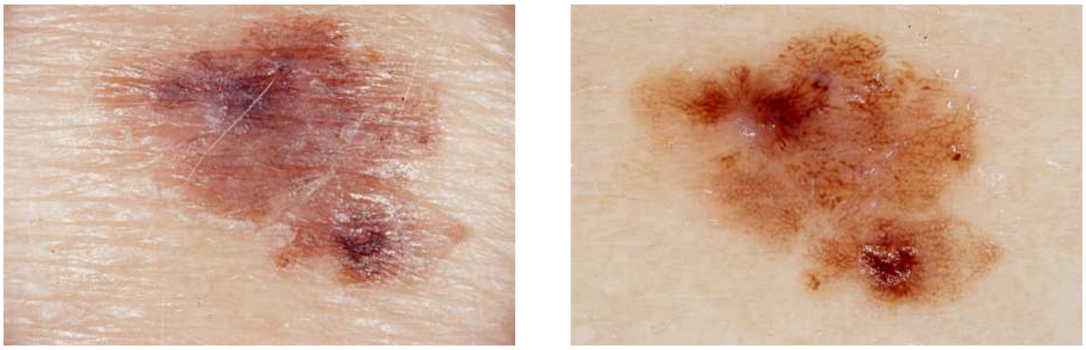

Figure 2.2: Example of pigmented skin lesion. Left: Traditional imaging technique. Right: Dermatoscopy imaging technique.

2.3 Image acquisition techniques

2.3.1 Traditional imaging

In larger dermatological clinics, records of the patients skin lesions are kept in form of a diagnosis and one or more traditional photographs of the lesion. Some patients may be predisposed to melanoma due to, e.g., cancer in the family or dysplastic nevus syndrome. These patients will often be regularly checked in order to detect any changes in their skin lesions. Photographs taken at each check-up are compared and any change is an indication of a possible malignancy. In this case, the lesion is removed and a biopsy performed.

It is mainly for this monitoring over time that traditional imaging is used today. An example of a traditional photograph is shown in figure 2.2.

2.3.2 Dermatoscopic imaging

Since traditional imaging is just a recording of what the human eye sees, it does not reveal any information unavailable to the eye. *Dermatoscopy* also known as *epiluminescence microscopy*, on the other hand, is an imaging technique that enables creating direct links between biological behavior and distinct visual characteristics.

Dermatoscopy is a non-invasive imaging technique that renders the *stratum corneum*[2] translucent and makes subsurface structures of the skin visible. The technique is fairly simple and involves removing reflections from the skin surface. This is done by applying immersion oil onto the skin lesion and pressing a glass plate with the same reflection index as the stratum corneum onto the lesion. The oil ensures that small cavities between the skin and the glass plate are filled in order to reduce reflections. With a strong lightsource, usually a halogen lamp, it is now possible to see skin structures below the skin surface. Usually the glass plate and lightsource are integrated into devices like a *dermatoscope* or a *dermatoscopic camera* as shown in figure 2.3. Both of these have lenses allowing a 10x magnification of pigmented skin lesions. In figure 2.2 an example of a skin lesion, recorded by the dermatoscopic imaging technique, is shown.

Although this imaging technique is not new, it is only in the last decade that the technique has been thoroughly investigated, especially in Western Europe [Argenyi, 1997]. It is still, though, not a widely used technique primarily due to the lack of formal training

[2]The top layer of the skin.

Figure 2.3: Examples of dermatoscopic imaging devices. Left panel: Dermatoscope without image recording capability. Right panel: Dermatoscopic camera.

in evaluating and understanding the visual characteristics in the images. Some of these characteristics will be briefly described in the next section.

A few studies concerning processing and analysis of digital dermatoscopic images have been published. In [Fischer et al., 1996] and [Schmid and Fischer, 1997], results of color segmentation techniques based on fuzzy c-means clustering are shown. Preliminary results using a minimum-distance classifier for discriminating between benign nevi, dysplastic nevi and malignant melanoma are presented in [Ganster et al., 1995]. Based on features describing various properties including shape and color, they were able to classify 56% of skin lesions in a test set correctly.

2.4 Dermatoscopic features

The dermatoscopic imaging technique produces images that are quite different from traditional images. Several visual characteristics have been defined and analyzed in recent studies, e.g., [Steiner et al., 1993], [Stolz et al., 1994] and [Stanganelli et al., 1995]. In the rest of this thesis, these visual characteristics will be called *dermatoscopic features* or just *features* for short.

Table 2.1 lists the most important dermatoscopic features together with a short description. The features all describe specific biological behavior, see, e.g., [Stolz et al., 1994] for a more detailed description. In figures 2.4, 2.5 and 2.6 several dermatoscopic features are shown on pigmented skin lesions.

As can be seen in, e.g., figure 2.4, there is one prominent artifact due to the use of immersion oil. Small air bubbles occur in the oil layer and appear as small white circles or ellipses. This artifact can be avoided if the oil is carefully applied. Usually the area occupied by air bubbles is very small but important features like, e.g., black dots or pseudopods may be obscured by air bubbles.

Table 2.1: Definition of dermatoscopic features

Feature	Description
Asymmetry	An asymmetric shape is the result of different local growth rates. This indicates malignancy. Asymmetry may be defined in numerous ways, though. In section 3.4.1, one such definition is presented.
Edge abruptness	A sharp abrupt edge suggests melanoma while a gradual fading of the pigmentation indicates a benign lesion.
Color distribution	Six different colors may be observed: Light-brown, dark-brown, white, red, blue and black. A large number of colors present indicates melanoma.
Pigment network	Areas with honeycomb-like pigmentation. A regular network usually indicates a benign lesion. A network with varying mesh size suggests an atypical/dysplastic nevus or a melanoma.
Structureless area	Areas with pigmentation but without any visible network. Unevenly distributed areas indicate melanoma.
Globules	Nests with a diameter of more than $0.1mm$ of heavily pigmented melanocytic cells. These may be brown or black. If evenly distributed, it indicates a benign lesion.
Black dots	Heavily pigmented melanocytic cells with a diameter less than $0.1mm$. If located close to the perimeter, it suggests an atypical lesion or a melanoma.
Pseudopods	Large "rain-drop" shaped melanoma nests located at the edge of the lesion. A very strong indicator of malignant melanoma.
Radial streaming	Radial growth of melanoma. Looks like streaks. Very indicative of malignant melanoma.
Blue-white veil	Areas with a blue-white shade of color. Indicates melanocytic cells located deep in the skin. An indicator of melanoma.
Depigmentation	Loss of pigmentation. An indicator of melanoma.

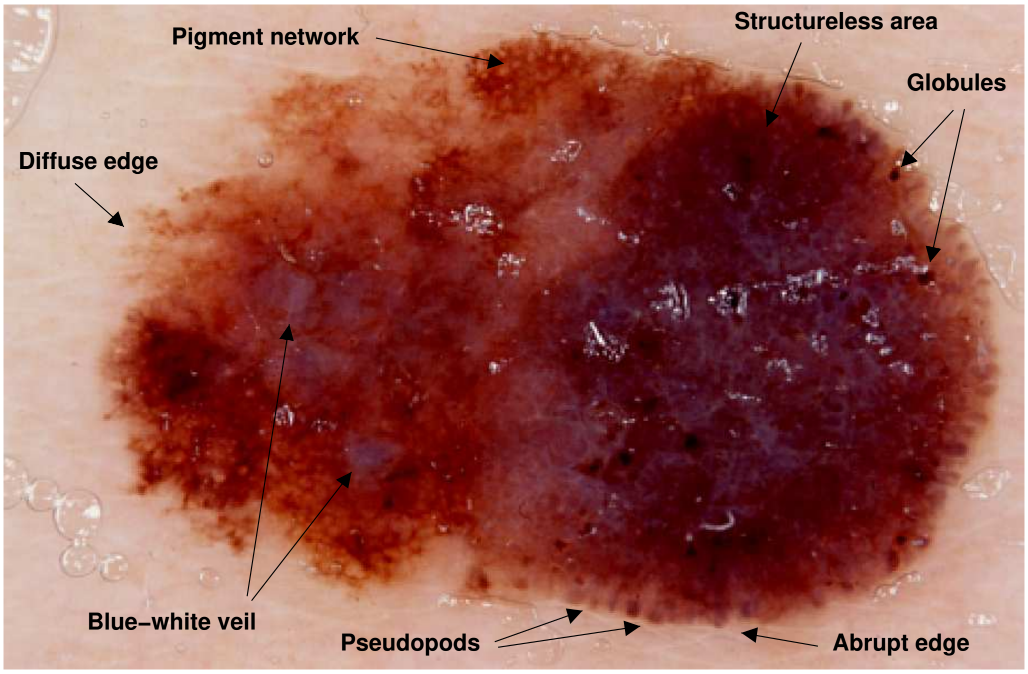

Figure 2.4: Pigmented skin lesion with several dermatoscopic features.

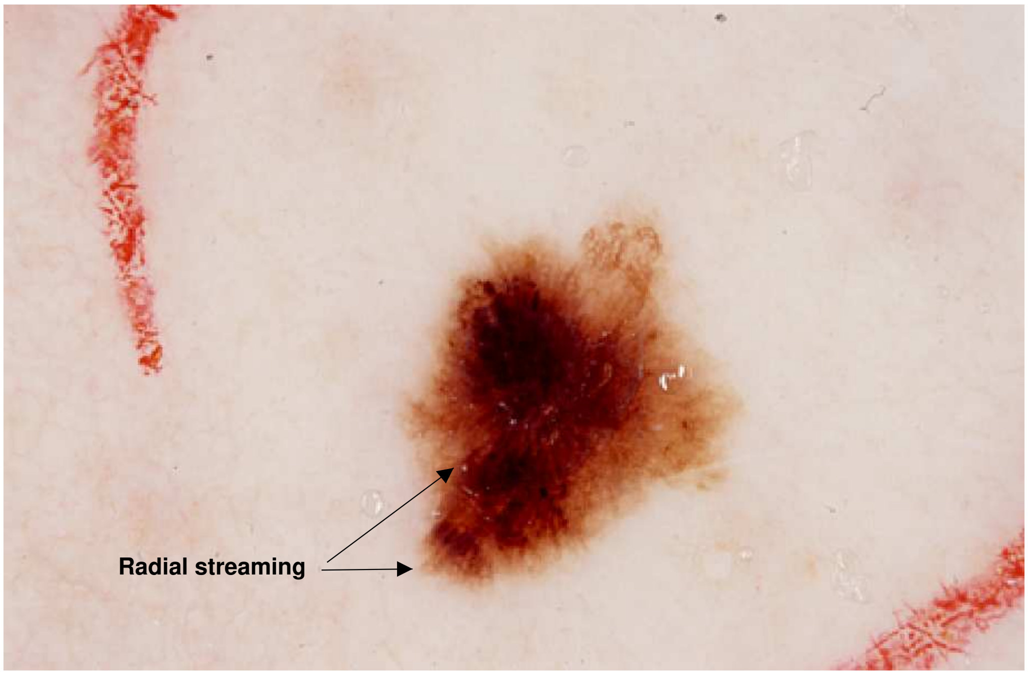

Figure 2.5: Pigmented skin lesion with radial streaming.

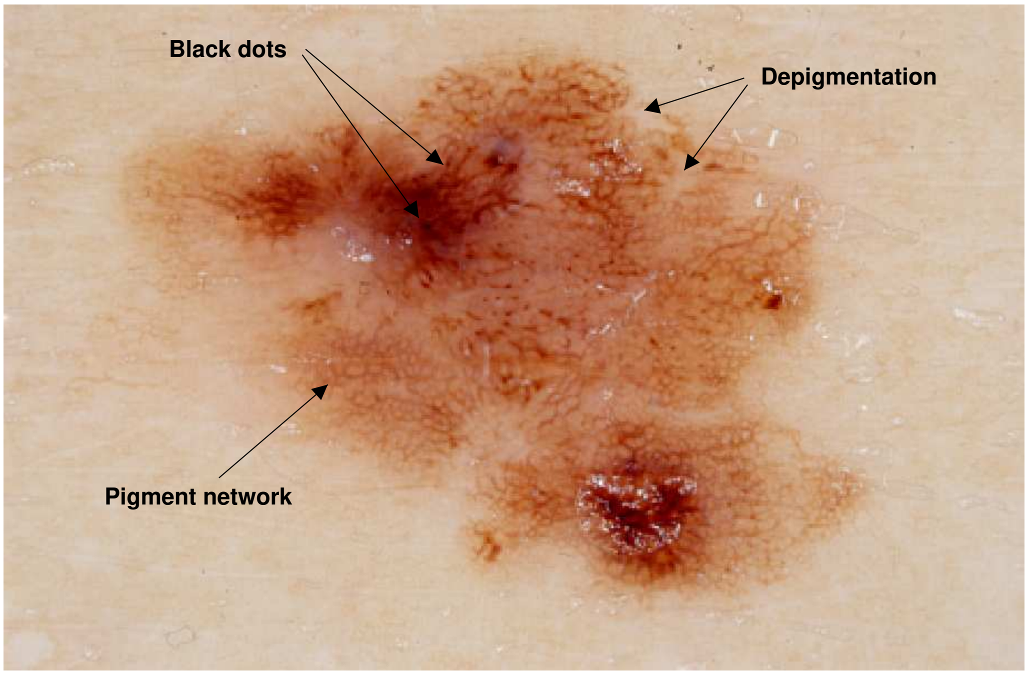

Figure 2.6: Pigmented skin lesion with several dermatoscopic features.

Chapter 3

Feature extraction in dermatoscopic images

In the previous chapter, dermatoscopic images and features were introduced. In this chapter, we will describe the image processing techniques used in order to extract and describe dermatoscopic features in an appropriate way.

In figure 3.1, a flowchart describing the feature extraction process is shown. The four

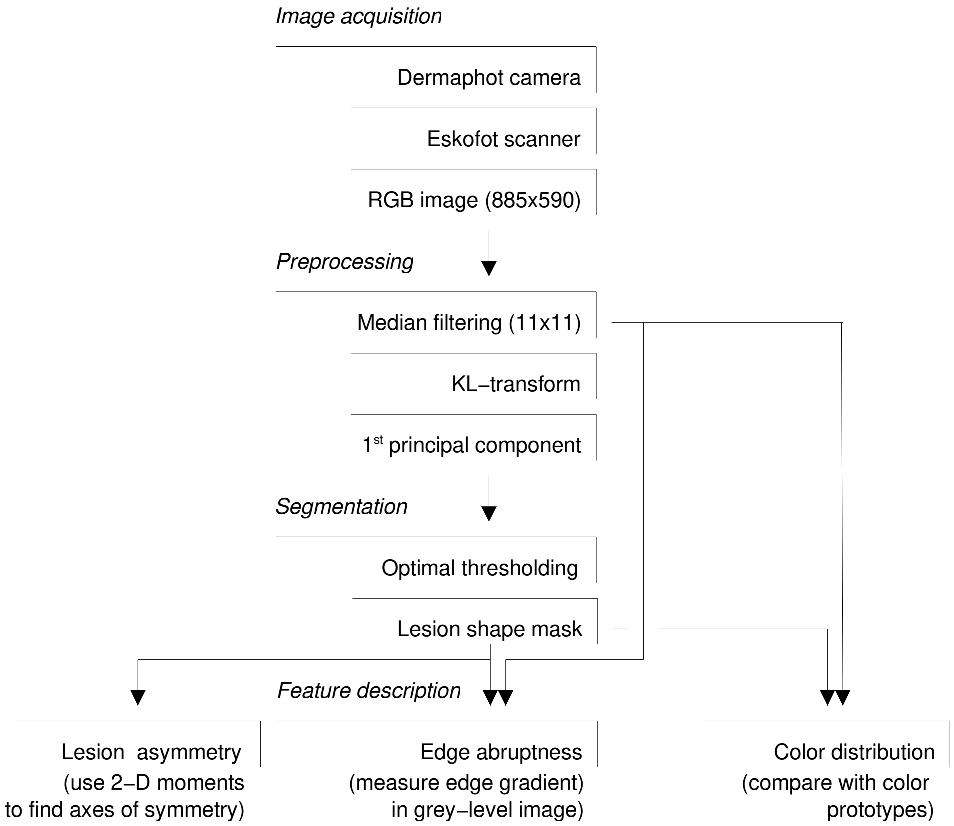

Figure 3.1: Feature extraction flowchart showing the four main processing blocks, image acquisition, preprocessing, segmentation and feature description.

main blocks, image acquisition, preprocessing, segmentation and dermatoscopic feature description, are described in the next sections.

3.1 Image acquisition

All dermatoscopic images used in this study are acquired at *Rigshospitalet, Copenhagen, Denmark* using a *Dermaphot* camera manufactured by *Heine Optotechnik* as seen in figure 2.3.

The images are developed as slides and digitalized with a resolution of 1270 dots per inch and 24 bit color[1] using an *Eskoscan 2540* color scanner from *Eskofot*. The image resolution has later digitally been reduced by a factor 2 in order to limit the computational resources needed for processing the images, thus reducing the size of each image to 885x590.

3.2 Image preprocessing

The first step in the feature extraction process is preprocessing of images with the purpose of reducing noise and facilitating image segmentation by using median filtering and the Karhunen-Loéve transform.

Now, let us first define a grey-level image of size MxN as a sequence of numbers,

$$z(m,n), \quad 1 \leq m \leq M, 1 \leq n \leq N, \tag{3.1}$$

where $z(m,n)$ is the luminance of pixel (m,n). If we are dealing with an 8-bit grey-level image, then each element, $z(m,n)$, will be an integer in the interval $[0;255]$. In any processing of 8-bit images, we will abandon the integer restriction and process the image in a floating point representation in order to minimize quantization effects.

Next, we define a color image of size MxN as 3 sequences, $r(m,n), g(m,n)$ and $b(m,n)$, with $r(m,n)$ representing the red color component, $g(m,n)$ the green color component and $b(m,n)$ the blue color component. The individual color components are typically representated by 8-bit but again any processing will be done using floating point precision.

3.2.1 Median filtering

As noted in section 2.4, the immersion oil used in the dermatoscopic imaging technique may produce small air bubbles manifestating themselves as small white ellipses, lines or dots. This artifact can be considered as impulsive noise and may thus be reduced using a median filter given by

$$\begin{aligned} z_{\text{med}}(m,n) = \text{median}\{z(m-k,n-l)| -\frac{N_{\text{med}}-1}{2} \leq k,l \leq \frac{N_{\text{med}}-1}{2} \\ \wedge 1 \leq m-k \leq M \wedge 1 \leq n-l \leq N\}, \end{aligned} \tag{3.2}$$

where N_{med} is odd[2] and indicates the size of the two-dimensional median filter. Note that we only consider a square median filter kernel. We may in fact consider any shape

[1] 8 bit for each of the color channels red, green and blue.

[2] If the median kernel size is even, there will be two middle values. One could then define the median as the mean of these two values.

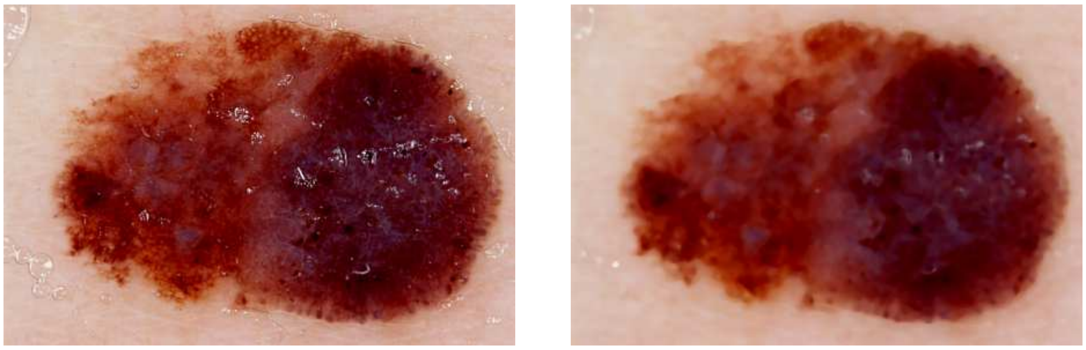

Figure 3.2: The effect of filtering a 885x590 dermatoscopic image with a 11x11 median filter. Left: Original image. Right: Filtered image. Notice how the air bubble artifacts have been reduced, especially around the lesion edge in the upper right hand corner.

of filter kernel if desirable. Equation (3.2) is valid for a grey-level image. When working with color images, one should apply the same median filter to all 3 color components.

Some of the properties of the non-linear median filter are [Jain, 1989], [Sonka et al., 1993]:

- The median filter is very effective in removing noise spikes that only cover a few pixels compared to the kernel neighborhood size. Thus, it can applied for removing thin lines, e.g. human hair, or isolated islands of pixels, e.g. small air bubbles.
- The median filter preserves spatial resolution well, i.e., it reduces blurring of edges compared to linear convolution filtering with Gaussian or averaging kernels.
- If the number of noise pixels inside the kernel neighborhood is greater than half the number of kernel neighborhood pixels, the median filter performs poorly.
- The median may damage sharp corners in the image.

Skin lesion specific comments

The main purpose of filtering dermatoscopic images is to reduce impulsive-like artifacts while at the same time preserving edges. In figure 3.2, the results of applying a 11x11 median filter to a dermatoscopic image is shown. This kernel size is used for all median filtering in this thesis.

3.2.2 Karhunen-Loève transform

The next preprocessing stage aims at facilitating the segmentation process by enhancing the edges in the image. For this purpose, we will consider the *Karhunen-Loève* (KL) transform also known as the *Hotelling* transform or the method of principal components [Hotelling, 1933], [Karhunen, 1947], [Lòeve, 1948].

The KL transform is a linear transformation that uncorrelates the input variables by employing an orthonormal basis found by an eigenvalue decomposition of the sample covariance matrix for the input variables.

In image processing applications, the KL transformation is often applied to the 2-D image domain. Here we will apply the transformation to the 3-D color space spanned by $r(m,n)$, $g(m,n)$ and $b(m,n)$.

Now, let us define the following $3 \times MN$ matrix containing all pixels from the 3 color channels,

$$\mathbf{V} = \begin{bmatrix} r(1,1) & r(1,2) & \dots & r(1,N) & r(2,1) & \dots & r(M,N) \\ g(1,1) & g(1,2) & \dots & g(1,N) & g(2,1) & \dots & g(M,N) \\ b(1,1) & b(1,2) & \dots & b(1,N) & b(2,1) & \dots & b(M,N) \end{bmatrix}, \tag{3.3}$$

where we view $[r(m,n)\ g(m,n)\ b(m,n)]^{\mathrm{T}}$ as a sample of a stochastic variable.

Let $\bar{\mathbf{v}}$ contain the sample mean of the 3 color components,

$$\bar{\mathbf{v}} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \begin{bmatrix} r(m,n) \\ g(m,n) \\ b(m,n) \end{bmatrix}. \tag{3.4}$$

The sample covariance matrix is now given by

$$\mathbf{C} = \frac{1}{MN} \mathbf{V}\mathbf{V}^{\mathrm{T}} - \bar{\mathbf{v}}\bar{\mathbf{v}}^{\mathrm{T}}, \tag{3.5}$$

that can be eigenvalue decomposed, so that

$$\mathbf{C} = \mathbf{E}\boldsymbol{\Lambda}\mathbf{E}^{\mathrm{T}}, \tag{3.6}$$

where $E = [\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3]$ is a matrix containing the eigenvectors of $\mathbf{C}$ and $\boldsymbol{\Lambda}$ a diagonal matrix containing the corresponding eigenvalues of $\mathbf{C}$ in decreasing order: $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$.

The KL transformation is now defined as

$$\mathbf{z} = \mathbf{E}^{\mathrm{T}} (\mathbf{v} - \bar{\mathbf{v}}) \tag{3.7}$$

where $\mathbf{v}$ is a column vector in $\mathbf{V}$ and $\mathbf{z}$ contains what is known as the *principal components*. Some properties of the principal components are [Mardia et al., 1979],

- $\langle z_i \rangle = 0$
- $\mathrm{Var}[z_i] = \lambda_i$
- $\mathrm{Cov}[z_i, z_j] = 0, \quad i \neq j.$

Due to the decreasing ordering of the eigenvalues and the corresponding eigenvectors, the first principal component will contain the maximum variance. In fact, no other linear transformation using unit length basis vectors can produce components with a variance larger than λ_1 [Mardia et al., 1979].

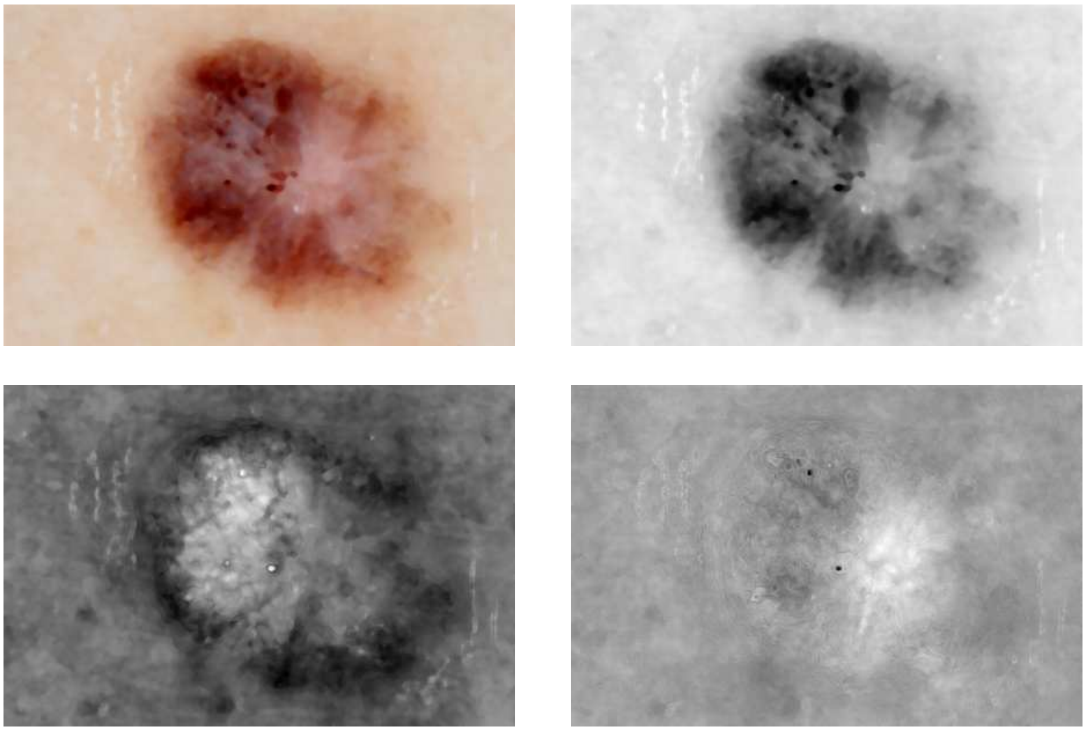

Figure 3.3: Results of the Karhunen-Loève transform applied to the 3-D color space. Upper left: The original median filtered color image. Upper right: The first principal component accounting for 98.7% of the total variance. Lower left: The second principal component - 1.2% variance Lower right: The third principal component - 0.1% variance. Note, that all principal component images have been rescaled to enhance the visual quality.

Skin lesion specific comments

For median filtered dermatoscopic images, the first principal component will typically account for more than 95% of the total variance. Since most variation occur at edges between regions with similar luminance levels, the first principal component is a natural choice for segmentation. Another study also shows that the Karhunen-Loève transform is appropriate for segmenting dermatoscopic images [Fischer et al., 1996].

In figure 3.3, a Karhunen-Loève transformation of a median filtered color image is shown.

3.3 Image segmentation

The next step in the feature extraction process is *image segmentation.* The main goal is to divide an image into regions of interests from which appropriate features can be extracted. Here, we will consider a complete segmentation that divides the entire image into disjoint regions. Denoting the image, $\mathcal{R}$, and the N regions, $\mathcal{R}_i,\ i = 1, 2, \ldots, N$, this may be formalized as

$$\mathcal{R} = \bigcup_{i=1}^{N} \mathcal{R}_i, \qquad \mathcal{R}_i \cap \mathcal{R}_j = \emptyset,\ i \neq j. \tag{3.8}$$

The regions are usually constructed so that they are homogeneous with respect to some chosen property like, e.g., luminance, color or context. We will now consider the case where the aim is to group pixels containing the same approximate luminance level.

3.3.1 Optimal thresholding

Thresholding is a very simple segmentation method based on using thresholds on the luminance level of pixels in order to determine what region a pixel belongs to. Denoting the non-negative luminance of a pixel, $z(m,n)$, a thresholding process using $N-1$ thresholds to divide an image into N regions may be written as

$$z(m,n) \in \begin{cases} \mathcal{R}_1 & \text{if } z(m,n) < T_1 \\ \mathcal{R}_2 & \text{if } T_1 \leq z(m,n) < T_2 \\ \vdots & \vdots \\ \mathcal{R}_i & \text{if } T_{i-1} \leq z(m,n) < T_i \\ \vdots & \vdots \\ \mathcal{R}_N & \text{if } T_{N-1} \leq z(m,n) \end{cases}, \tag{3.9}$$

where T_i is the threshold separating pixels in region $\mathcal{R}_i$ from pixels in region $\mathcal{R}_{i+1}$. To simplify the notation in the following, we introduce $T_0 = -\infty$ and $T_N = \infty$.

Let us consider the luminance level, $z(m,n)$, to be a sample of a stochastic variable, z, and let the conditional luminance probability distribution be denoted by $p(z|\mathcal{R}_i)$ and the prior region probability by $P(\mathcal{R}_i)$. Assuming we know $p(z|\mathcal{R}_i)$ and $P(\mathcal{R}_i)$, we may view the problem of selecting the thresholds as a classification problem and use Bayesian decision theory to minimize the probability of misclassifying a pixel. Bayesian decision theory will be discussed in greater detail in chapter 4.1.

We may now write the probability of misclassifying a pixel in region $\mathcal{R}_i$ using the thresholds $T_1, T_2, \ldots, T_{N-1}$[3] as

$$P_{\mathrm{e}}\left(T_1, T_2, \ldots, T_{N-1} | \mathcal{R}_i\right) \quad = \quad \int_{-\infty}^{T_{i-1}} p(z|\mathcal{R}_i)dz + \int_{T_i}^{\infty} p(z|\mathcal{R}_i)dz \tag{3.10}$$

where $P_{\mathrm{e}}(\cdot)$ denotes the probability of misclassifying a pixel and $i = 1, 2, \ldots, N$. The combined misclassification probability is

$$P_{\mathrm{e}}\left(T_1, T_2, \ldots, T_{N-1}\right) = \sum_{i=1}^{N} P_{\mathrm{e}}\left(T_1, T_2, \ldots, T_{N-1} | \mathcal{R}_i\right) P(\mathcal{R}_i). \tag{3.11}$$

In order to minimize $P_{\mathrm{e}}\left(T_1, T_2, \ldots, T_{N-1}\right)$, we compute its derivative w.r.t. T_i and find the zeros of the derivative:

[3]Recall, T_0 and T_N are fixed at $-\infty$ and ∞, respectively, and thus not of interest to us.

$$\begin{aligned}\frac{\partial P_{\mathrm{e}}\left(T_1, T_2, \ldots, T_{N-1}\right)}{\partial T_i} &= -p(T_i|\mathcal{R}_i)P(\mathcal{R}_i) + p(T_i|\mathcal{R}_{i+1})P(\mathcal{R}_{i+1}) = 0 \qquad (3.12)\\ &\Downarrow \\ p(T_i|\mathcal{R}_i)P(\mathcal{R}_i) &= p(T_i|\mathcal{R}_{i+1})P(\mathcal{R}_{i+1}) \qquad (3.13)\end{aligned}$$

where $i = 1, 2, \ldots, N-1$.

If we know the conditional luminance probability distributions, $p(z|\mathcal{R}_i)$, and the prior region probabilities, $P(\mathcal{R}_i)$, we can find the optimal[4] thresholds by solving equation (3.13).

Let us consider the case where the conditional luminance probability distributions are Gaussian of the form

$$p(z|\mathcal{R}_i) = \frac{1}{\sqrt{2\pi}\sigma_{\mathcal{R}_i}} \exp\left(-\frac{(z-\mu_{\mathcal{R}_i})^2}{2\sigma_{\mathcal{R}_i}^2}\right), \qquad (3.14)$$

where $\mu_{\mathcal{R}_i}$ is the mean of the pixel luminance in region $\mathcal{R}_i$, $\sigma_{\mathcal{R}_i}^2$ the corresponding variance and $i = 1, 2, \ldots, N$.

Inserting equation (3.14) into equation (3.13) and applying the logarithm, we obtain

$$\log \sigma_{\mathcal{R}_i} + \log P(\mathcal{R}_{i+1}) + \frac{(T_i - \mu_{\mathcal{R}_i})^2}{2\sigma_{\mathcal{R}_i}^2} = \log \sigma_{\mathcal{R}_{i+1}} + \log P(\mathcal{R}_i) + \frac{(T_i - \mu_{\mathcal{R}_{i+1}})^2}{2\sigma_{\mathcal{R}_{i+1}}^2}, \qquad (3.15)$$

where $i = 1, 2, \ldots, N-1$. To simplify matters, we assume $\sigma = \sigma_{\mathcal{R}_1} = \sigma_{\mathcal{R}_2}$. This yields the following closed-form solution for the optimal thresholds,

$$T_i = \frac{\mu_{\mathcal{R}_i} + \mu_{\mathcal{R}_{i+1}}}{2} + \frac{\sigma^2}{\mu_{\mathcal{R}_i} - \mu_{\mathcal{R}_{i+1}}} \log \frac{P(\mathcal{R}_{i+1})}{P(\mathcal{R}_i)}, \qquad (3.16)$$

where $i = 1, 2, \ldots, N-1$. Assuming the prior probabilities, $P(\mathcal{R}_i)$, are equal, equation (3.16) reduces to

$$T_i = \frac{\mu_{\mathcal{R}_i} + \mu_{\mathcal{R}_{i+1}}}{2}. \qquad (3.17)$$

A simple iterative scheme based on equation (3.17) for estimating the $N-1$ optimal thresholds and the N luminance means is [Ridler and Calvard, 1978]

1. Initialize thresholds, so that $T_1 < T_2 < \ldots < T_{N-1}$.
2. At time step t, compute the luminance region means

$$\mu_{\mathcal{R}_i}^{(t)} = \frac{\sum_{(m,n)\in\mathcal{R}_i^{(t)}} z(m,n)}{N_{\mathcal{R}_i}^{(t)}}, \qquad (3.18)$$

where $N_{\mathcal{R}_i}^{(t)}$ is the number of pixels in region $\mathcal{R}_i$ at time step t and $i = 1, 2, \ldots, N$.

[4] In the sense that the thresholds minimize the probability of misclassification.

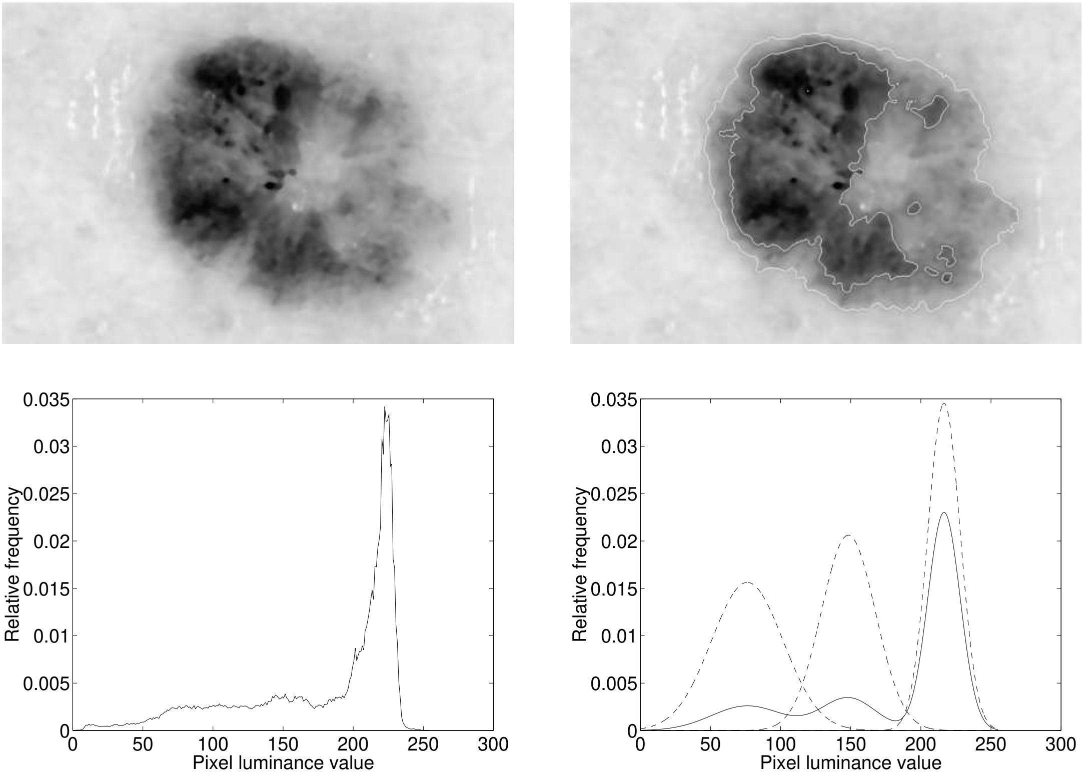

Figure 3.4: Example of results using the optimal thresholding algorithm on the first principal component of a median filtered dermatoscopic image. Upper left: Median filtered first principal component. Upper right: The segmentation result using 2 thresholds to separate 3 regions. The solid white lines indicate region borders. Lower left: The sample histogram of the upper left image. Lower right: Estimated histogram. The dashed lines show the luminance probability densities, $\hat{p}(z|\mathcal{R}_i)$, estimated by the optimal thresholding algorithm. The solid line shows the estimated histogram computed by assuming that the prior probabilities of the 3 regions are $1/6, 1/6$ and $4/6$ from left to right. Note, that the overall shape of the estimated histogram matches the sample histogram fairly well.

3. The thresholds at time step $t+1$ are now computed as

$$T_i^{(t+1)} = \frac{\mu_{\mathcal{R}_i}^{(t)} + \mu_{\mathcal{R}_{i+1}}^{(t)}}{2}, \tag{3.19}$$

   where $i = 1, 2, \ldots, N-1$.

4. If $T_i^{(t+1)} = T_i^{(t)}$ for all $i = 1, 2, \ldots, N-1$, then stop; otherwise return to step 2.

Skin lesion specific comments

All dermatoscopic images in this study have been segmented by the optimal thresholding algorithm using 2 thresholds. A typical first principal component of a median filtered dermatoscopic image consists of a very light background and a dark skin lesion with even darker areas inside. These 3 regions are usually fairly homogeneous making the assumption of Gaussian luminance probability distributions a sound one. The assumptions of equal variances, $\sigma^2_{\mathcal{R}_i}$ and equal priors, $P(\mathcal{R}_i)$, are usually not warranted. Nevertheless, the algorithm provides good results using dermatoscopic images.

Note, the main purpose of segmentation in this application is to find a lesion shape mask defining the edge location of the lesion. Thus, we are only interested in the threshold separating the light skin background and the darker skin lesion. In some cases, the segmentation produces several skin lesion candidates due to other small non-lesion objects. Usually the largest object is the skin lesion and is thus selected for further processing.

In figure 3.4, the results of using the optimal thresholding algorithm on a dermatoscopic image using 2 thresholds to separate 3 regions are shown. Note, the similar shape of the sample histogram and the estimated histogram indicating the usability of the optimal thresholding algorithm in the context of dermatoscopic images.

3.4 Dermatoscopic feature description

The final step in the feature extraction process is the actual extraction and description of features. We will in this section present methods for describing the following skin lesion properties:

- Asymmetry of the lesion border.
- Transition of the pigmentation from the skin lesion to the surrounding skin.
- Color distribution of the skin lesion including blue-white veil.

3.4.1 Asymmetry

An asymmetric skin lesion shape is the result of different local growth rates and may indicate malignancy.

In order to measure asymmetry, we will first look at 2-D moments and how these may be used for describing certain geometrical properties of an object or a region in an image.

3.4.1.1 Moments

Moment representations interpret a normalized grey level image function, $z(x, y)$, as a probability density function of a 2-D stochastic variable. Properties of this variable may thus be described by 2-D moments [Sonka et al., 1993]. For a digital image, $z(m, n)$, the *moment of order* $(p + q)$ is given by

$$m_{pq} = \sum_{m=1}^{M} \sum_{n=1}^{N} m^p n^q z(m, n). \tag{3.20}$$

Translation invariant moments are obtained by considering the *centralized moments*

$$m^c_{pq} = \sum_{m=1}^{M} \sum_{n=1}^{N} (m - m_c)^p (n - n_c)^q z(m, n), \tag{3.21}$$

where (m_c, n_c) is the *center of mass* given by

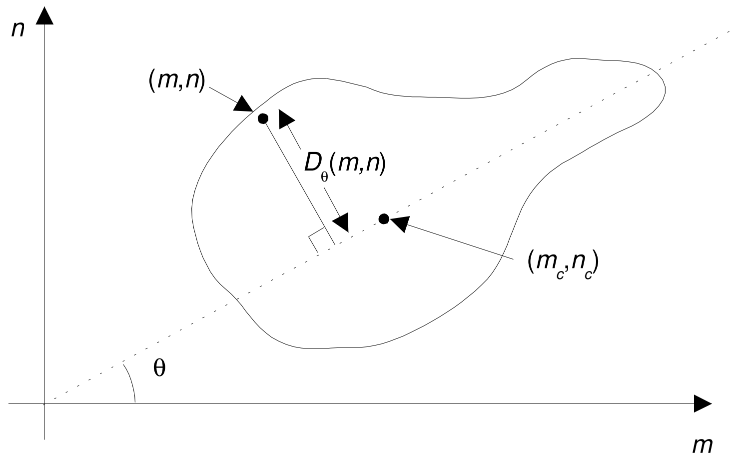

Figure 3.5: The orientation angle, θ_o, of an object is defined as the angle of the axis through the center of mass, (m_c, n_c), that minimizes the moment of inertia, $I(\theta) = \sum_{m=1}^{M}\sum_{n=1}^{N} D_\theta^2(m,n)z(m,n)$.

$$m_c = \frac{m_{10}}{m_{00}}, \quad n_c = \frac{m_{01}}{m_{00}}. \tag{3.22}$$

We will now in the following consider the case where $z(m,n)$ is binary and represents a region, $\mathcal{R}$, so that $z(m,n) = 1$ if $(m,n) \in \mathcal{R}$, otherwise $z(m,n) = 0$. This could, e.g., be the result of a segmentation process.

The moment of inertia for a binary object or region, $\mathcal{R}$, w.r.t. an axis through the center of mass with an angle θ as shown in figure 3.5 is defined as [Jain, 1989]

$$\begin{aligned} I(\theta) &= \sum\sum_{(m,n)\,\in\,\mathcal{R}} D_\theta^2(m,n) && (3.23)\\ &= \sum\sum_{(m,n)\,\in\,\mathcal{R}} \left[-(m-m_c)\sin\theta + (n-n_c)\cos\theta\right]^2, && (3.24)\end{aligned}$$

where $D_\theta(m,n)$ is found by translating the object so that its center of mass coincides with the center of origo of the coordinate system and by rotating[5] the object clockwise by the angle θ so that the n-coordinate of the translated and rotated point (m,n) equals the desired distance $D_\theta(m,n)$.

The orientation of an object is defined as the angle of the axis through the center of mass that results in the least moment of inertia [Jain, 1989]. To obtain this angle, we compute the derivative of equation (3.24) and set it to zero,

[5]Rotation of a point (m,n) clockwise by the angle θ is given by: $(m_r, n_r) = (m\cos\theta + n\sin\theta, -m\sin\theta + n\cos\theta)$.

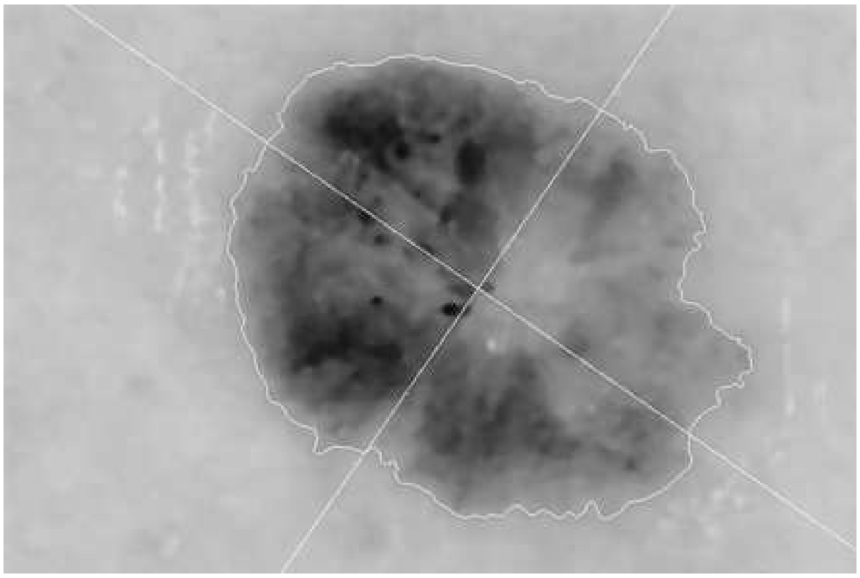

Figure 3.6: Skin lesion showing the edge of the lesion and the two principal axes used for calculating asymmetry. These axes define directions of least and largest moments of inertia. The two asymmetry indexes for this lesion are 0.054 and 0.055, respectively.

$$\frac{\partial I(\theta)}{\partial \theta} = 0 \tag{3.25}$$

$$\Downarrow$$

$$\theta_o = \frac{1}{2}\tan^{-1}\left[\frac{2m_{11}^c}{m_{20}^c - m_{02}^c}\right]. \tag{3.26}$$

The axis through the center of mass defined by θ_o is also known as a *principal axis*. We will refer to this as the *major axis*. All objects have two principal axes[6] where the second principal axis is defined by the angle yielding the largest moment of inertia. This will be referred to as the *minor axis*. The principal axes are orthogonal and will in the next section be used for calculating asymmetry.

In figure 3.6, an example of a skin lesion and its two principal axes are shown.

3.4.1.2 Measuring asymmetry

The principal axes found in the previous section will now be used as axes of symmetry. That is, we will measure how asymmetric the object is with respect to these two axes. This can be done by folding the object about its principal axes and measure the area of the non-overlapping regions relative to the entire object area. Thus, for each principal axis, we define a measure of asymmetry as

$$S_i = \frac{\Delta A_i}{A}, \tag{3.27}$$

[6]Note, a circle has an infinite number of principal axes due to its rotational symmetry.

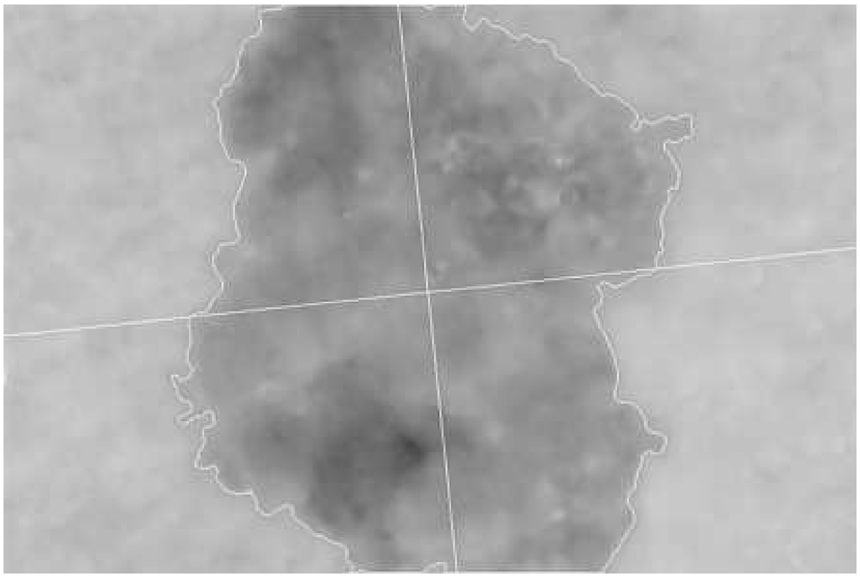

Figure 3.7: An example of a skin lesion larger than the field of view of the camera. The two asymmetry measures for this lesion are both 0.14. Only very large lesions, where the calculation of asymmetry can not be justified, have been omitted from the study.

where $i = 1, 2$ indicates the principal axis, ΔA_i is the corresponding non-overlapping area of the folded object and A is the area of the entire region. For an object completely symmetric about the $i'th$ principal axis, S_i is zero while complete asymmetry yields an asymmetry measure of 1.

Skin lesion specific comments

Several skin lesions included in this study are larger than the field of view of the camera. That is, the entire lesion in not visible in the digitized image. This will introduce an uncertainty in the location of the principal axes and subsequently in the asymmetry measures. See the example in figure 3.7.

Due to the rather limited amount of data available, these have nevertheless been included. Some severe cases, where the calculation of asymmetry could not be justified, have been removed from the data set, though. One could also choose not to compute the asymmetry measures in these cases and subsequently treat them as missing values. Several techniques for dealing with missing values exist, see, e.g., [Ripley, 1996] for an overview.

3.4.2 Edge abruptness

An important feature is the transition of the pigmentation between the skin lesion and the surrounding skin. A sharp abrupt edge suggests malignancy while a gradual fading of the pigmentation indicates a benign lesion.

In order to measure the edge abruptness, let us first estimate the gradient of a grey-level image.

3.4.2.1 Image gradient estimation

In a continuous image, $z(x,y)$, the gradient magnitude, $g(x,y)$ and gradient direction, θ_g, is defined by [Sonka et al., 1993]

$$\begin{aligned} g(x,y) &= \sqrt{\left(\frac{\partial z(x,y)}{\partial x}\right)^2 + \left(\frac{\partial z(x,y)}{\partial y}\right)^2} && (3.28)\\ \theta_g &= \tan^{-1}\left(\frac{\partial z(x,y)}{\partial y} \Big/ \frac{\partial z(x,y)}{\partial x}\right), && (3.29)\end{aligned}$$

and for a digital image, $z(m,n)$,

$$\begin{aligned} g(m,n) &= \sqrt{g_1^2(m,n) + g_2^2(m,n)} && (3.30)\\ \theta_g &= \tan^{-1}\left(\frac{g_2(m,n)}{g_1(m,n)}\right). && (3.31)\end{aligned}$$

Here, the partial derivatives have been approximated by the differences $g_1(m,n)$ and $g_2(m,n)$ defined as

$$\begin{aligned} g_1(m,n) &= \sum_i \sum_j h_1(-i,-j) z(m+i, m+j) && (3.32)\\ g_2(m,n) &= \sum_i \sum_j h_2(-i,-j) z(m+i, m+j) && (3.33)\end{aligned}$$

where $g_1(m,n)$ and $g_2(m,n)$ are expressed as convolutions between the image and *gradient operators* denoted by $h_1(i,j)$ and $h_2(i,j)$, $-(N_h-1)/2 \leq i,j \leq (N_h-1)/2$, where N_h is odd and indicates the size of the gradient operators.

Several gradient operators have been suggested, see, e.g., [Jain, 1989]. Here we will use the Sobel gradient operator defined by

$$\mathbf{H}_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{3.34}$$

We will in the following denote the gradient magnitude estimation of a grey-level digital image, $z(m,n)$, using the Sobel gradient operators by

$$g(m,n) = \mathrm{grad}[z(m,n)]. \tag{3.35}$$

3.4.2.2 Measuring edge abruptness

Let us consider the luminance component of a color image given by

$$z(m,n) = \frac{1}{3}[r(m,n) + g(m,n) + b(m,n)], \tag{3.36}$$

which is just an equally weighted sum of the three color components.

We may now estimate the gradient magnitude of the intensity component by computing,

$$g(m,n) = \mathrm{grad}[z(m,n)]. \tag{3.37}$$

If we sample the gradient magnitude, $g(m,n)$, along the edge of the skin lesion, we obtain a set of gradient magnitude values,

$$e(k) = g(m(k),n(k)), \quad k = 0,1,\ldots,K-1, \tag{3.38}$$

where K is the total number of edge samples and $(m(k),n(k))$ the coordinates of the $k'th$ edge pixel.

This set of values describes the transition between the lesion and the skin background in each edge point. In order to describe the general transition or abruptness, we use the sample mean and variance of the gradient magnitude values $e(k)$:

$$m_e = \frac{1}{K}\sum_{k=0}^{K-1} e(k) \tag{3.39}$$

$$v_e = \frac{1}{K}\sum_{k=0}^{K-1} e^2(k) - m_e^2, \tag{3.40}$$

where the sample mean, m_e, describes the general abruptness level and the sample variance, v_e, describes the variation of the abruptness along the skin lesion edge.

In figure 3.8, an example of measuring the abruptness in a dermatoscopic image is shown.

Skin lesion specific comments

As mentioned previously, several skin lesions larger than the field of view of the camera are included in this study. For these lesions the gradient magnitude has not been sampled along false edges. These occur at the boundaries of the image where the skin lesion crosses the image border, see the example in figure 3.7. Thus we assume, that enough edge information is available from the visible part of the skin lesion in order to describe the characteristics of the lesion edge and that we can neglect the contributions outside the field of view.

3.4.3 Color

The color distribution of a skin lesion is another important aspect that may contribute to an accurate diagnosis. Dermatologists have identified 6 shades of color that may be present in skin lesions examined with the dermatoscopic imaging technique. These colors arise due to several biological processes [Stolz et al., 1994]. The colors are: *Light-brown*, *dark-brown*, *white*, *red*, *blue* and *black* [Stolz et al., 1994]. This is a rather vague color description that is likely to cause some discrepancies between how different individuals perceive skin lesion colors. There are especially problems with separating light-brown from

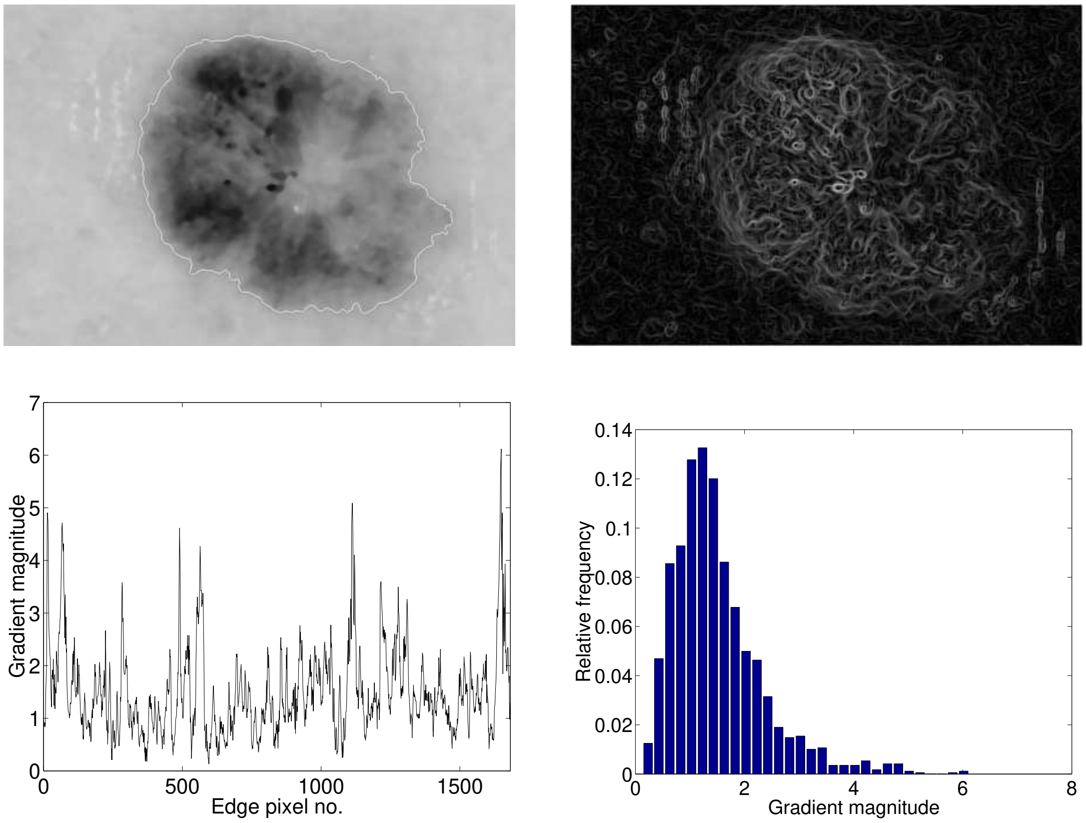

Figure 3.8: Example of measuring edge abruptness in a dermatoscopic image. Upper left: Intensity image showing the lesion edge obtained from the segmentation process. Upper right: Gradient magnitude image. Note: The gradient magnitude range has been compressed by the transformation, $g_c(m,n) = \log(1 + g(m,n))$, in order to enhance the visual quality. Lower left: The gradient magnitude sampled along the lesion edge. Lower right: Histogram of gradient magnitude measured along the lesion edge.

dark-brown but problems also occur with red and dark-brown due to a rather reddish glow of the dark-brown color in skin lesions.

We will nevertheless try to define a consistent method of measuring skin lesion colors that matches dermatologists intuitive perception of colors. This is done by defining color prototypes that are in close correspondence with the color perception of dermatologists and using these prototypes to determine the color contents of skin lesions. As a guideline, a large number of colors is considered to be an indicator of malignancy.

3.4.3.1 Color prototype determination

The color prototypes have been determined from three 2-D histograms[7] of 18 randomly selected skin lesion images combined into one large image. By inspecting the histograms, several clusters matching the color perception of dermatologists have been defined and the perceived cluster centers are used as prototypes. This is shown in figure 3.9. Note, that several shades of light-brown, dark-brown and blue have been identified. No reliable

[7] Red-green, red-blue and green-blue 2-D histograms.

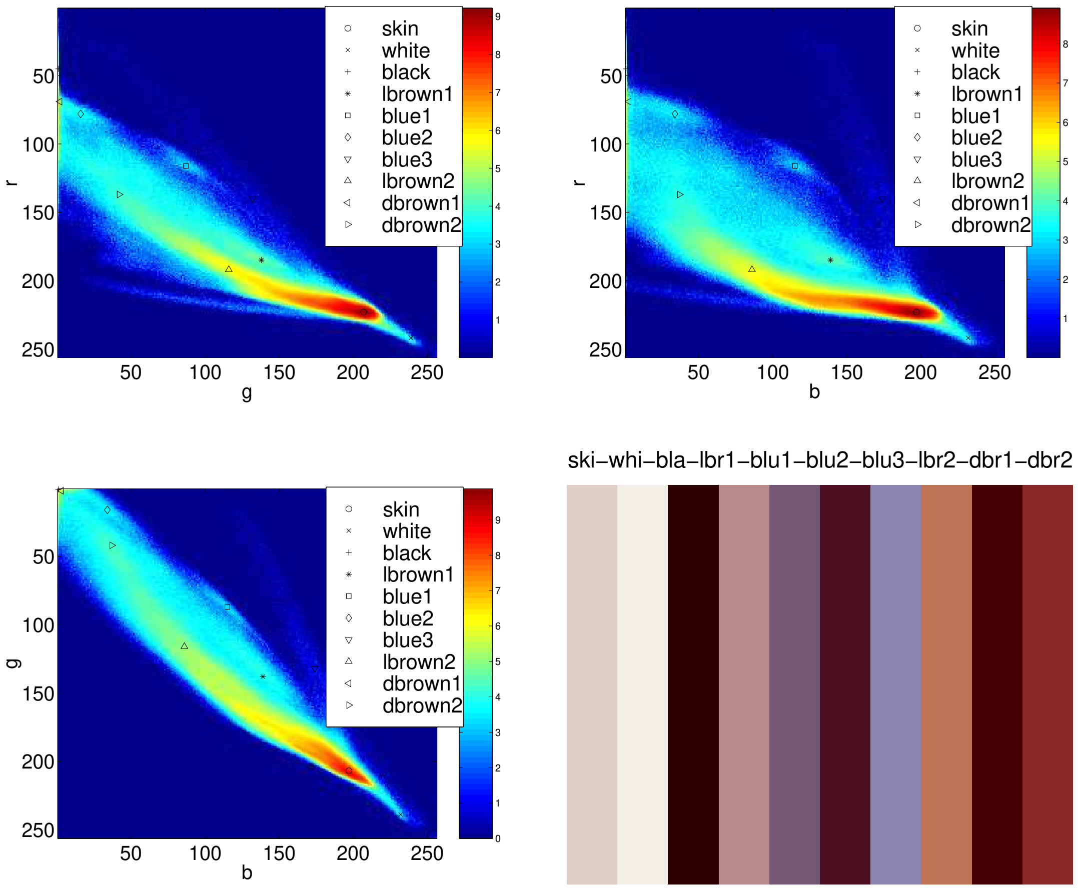

Figure 3.9: Color prototypes have been found manually by inspecting the combined 2-D histograms of 18 randomly selected images. The perceived cluster centers are chosen as prototypes. Upper left: Red-green 2-D histogram. The histogram values, $h(r,g)$, have been compressed by the transformation, $h_c(r,g) = \log(1 + h(r,g))$, in order to enhance the visual quality. Upper right: Red-blue 2-D histogram (log-transformed). Lower left: Green-blue 2-D histogram (log-transformed). Lower right: The determined color prototypes. The *skin* color prototype is left out since it is eliminated by the segmentation process. Only colors inside the lesion are of interest in this study.

prototype for red distinguishing it from dark-brown could be determined. This is a problem also found among dermatologists. One may consider a part of a lesion to be red while another may suggest dark-brown. Due to these difficulties, a red prototype has not been defined.

It is clear that this way of determining prototypes is a very subjective process, yet great care has been taken in order for the prototypes to match the color perception of dermatologists[8].

A standard *k-means* clustering algorithm using the Euclidean distance measure in the RGB color space has also been employed but did not yield acceptable color prototypes. It is obvious from inspecting the 2-D histograms that the Euclidean distance measure is not

[8]The author has spent hour-long sessions with dermatologists viewing and discussing skin lesions in order to gain insight into their color perception.

the most appropriate choice due to the varying shape of the different clusters. It would be beneficial to allow the distance measure to vary between clusters acknowledging that different probability distributions generate the individual clusters. Often these distributions may be considered Gaussian, see e.g. [Scott and Symons, 1971].

Another contributing factor to the failure of the standard *k-means* algorithm is the number of pixels in each cluster. The histograms in figure 3.9 are log-transformed, that is, the dynamic range has been compressed in order to enhance the visual quality. Thus the number of pixels close to the center of some of the clusters seems relative large compared to, e.g., the dominant *skin* color cluster even though the number of pixels in these clusters is in fact rather small. In the standard *k-means* algorithm these clusters are likely to be suppressed by the higher populated dominant clusters resulting in unacceptable results.

Thus in order to overcome these problems and to incorporate the color perception of dermatologists, the manually selected prototypes are used in this study. Note, that 10 color clusters have been defined but only 9 prototypes are used. The *skin* color prototype is left out as this color is eliminated by the segmentation process and normally only found outside the lesion. The 9 color prototypes thus corresponds of *white*, *black*, *light-brown 1*, *light-brown 2*, *dark-brown 1*, *dark-brown 2*, *blue 1*, *blue 2* and *blue 3* representing 5 different colors.

3.4.3.2 Measuring color

The color contents of a skin lesion may be determined by comparing the skin lesion pixels with color prototypes. Here we will use the Euclidean distance measure for comparing colors,

$$d_i^2(m,n) = [r(m,n) - r_i]^2 + [g(m,n) - g_i]^2 + [b(m,n) - b_i]^2, \quad i = 1, 2, \dots 9, \tag{3.41}$$

where $d_i(m,n)$ is the distance in RGB colorspace from pixel (m,n) to the $i'th$ color prototype defined by $cp_i = [r_i \; g_i \; b_i]^{\mathrm{T}}$.

Every skin lesion pixel can now be assigned a prototype color by selecting the shortest distance. That is, the pixel (m,n) should be assigned the prototype color cp_i if

$$d_i(m,n) < d_j(m,n) \quad \text{for all } i \neq j. \tag{3.42}$$

We may now describe the color contents of a skin lesion as a set of relative areas - one for each color prototype. This may be written as

$$a_i = \frac{A_{cp_i}}{A}, \tag{3.43}$$

where A is the area of the skin lesion, A_{cp_i} the area inside the skin lesion occupied by pixels close to prototype color cp_i as defined by equation (3.42) and a_i the relative measure of the color content of the prototype color cp_i. Since we do not wish to distinguish between different shades of the same color, the color content of light-brown is defined as the sum of a_i for the two light-brown color shades. The same applies to the blue and dark-brown color shades.

As mentioned in the previous section, the choice of distance measure is not trivial. The most appropriate distance measure in this context would be one that takes the color perception of dermatologists into account. The CIE[9] has proposed the perceptually uniform colorspaces, CIE-Lab and CIE-Luv, in which the Euclidean distance measure matches the average humans perception of color differences [Skarbek and Koschan, 1994]. In order to transform pixels in RGB colorspace to either CIE-Luv or CIE-Lab colorspace, one must first empirically determine a linear 3×3 transformation matrix for the complete imaging system[10] that transforms the RGB colorspace of the imaging system to the standardized CIE-RGB colorspace, see e.g. [Wyszecki and Stiles, 1982]. The CIE-RGB values may then be converted through a non-linear transformation into either CIE-Luv or CIE-Lab values [Jain, 1989]. Using the Euclidean distance measure in either of these colorspaces for comparing colors may yield results corresponding better with the color perception of dermatologists.

Examples of skin lesion comparison with the color prototypes are shown in figure 3.10.

Skin lesion specific comments

Note, that the use of color prototypes requires that the conditions of the imaging system are very controlled in order to achieve color consistency. This involves camera, lighting conditions, film type, film development process and scanner. We will briefly comment on each of these points:

Camera The same *Dermaphot* camera from *Heine Optotechnik* has been used for all imaging as seen in figure 2.3.

Lighting conditions The ring formed lightsource is together with the camera lens integrated into a closed container that only lets in light through the small glass plate at the front of the *Dermaphot* camera. The glass plate is pressed onto the oily lesion making any contribution from external lightsources negligible. Thus, the lighting conditions may be considered consistent from image to image.

Film type The same type of Fuji film has been used on all images included in this study.

Film development process The same developer studio was used for developing all films. The exact development process is not known and may be a source for introducing color inconsistency even though this does not seem to be a problem.

Scanner The *Eskoscan 2540* scanner from *Eskofot* was calibrated according to the manufacturers instructions before each scanning session was initiated ensuring scanning consistency.

Considering the conditions mentioned above, we do not expect color inconsistency to be a problem.

3.5 Dermatoscopic feature material

A total of 109 dermatoscopic images of skin lesions have been collected. Unfortunately, 51 images had to be excluded from this study due to the following reasons:

[9]Commission Internationale de L'Eclairage - the international committee on color standards.

[10]The imaging system in this application consists of camera, film, development process and image scanning.

- 16 skin lesions had not been histologically analyzed and thus had no diagnosis.
- 14 skin lesions were significantly larger than the field of view of the camera.
- 14 skin lesions had been photographed using a different type of Fuji film. The color sensation obtained from these images were significantly different compared to the rest of the images.
- 6 skin lesions were not considered segmented correctly. This was mainly due to other large dark objects being present in the images, e.g., black marker lines surrounding the lesions.
- 1 lesion image was severely overexposed.

The remaining 58 images are used for this study and are distributed in the 3 skin lesion categories as: *Benign nevi*: 25, *atypical nevi*: 11 and *malignant melanoma*: 22.

A total of 9 features are extracted from each image as described in the previous sections. In summary, these are: 2 asymmetry measures, 2 edge abruptness measures and 5 color measures. The distributions of the individual features within each skin lesion class are shown in figure 3.11 and 3.12. Some discrimination between skin lesion classes seems promising even considering only single features. Especially, high scores for the *edge abruptness* measures, the *blue color* measure, the *black color* measure and the *dark-brown color* measure and low scores for the *light-brown* color measure seem to indicate malignancy. When the features are combined, these may not be the most significant features, though. We will gain more insight into the importance of the individual features in section 6.2.3.

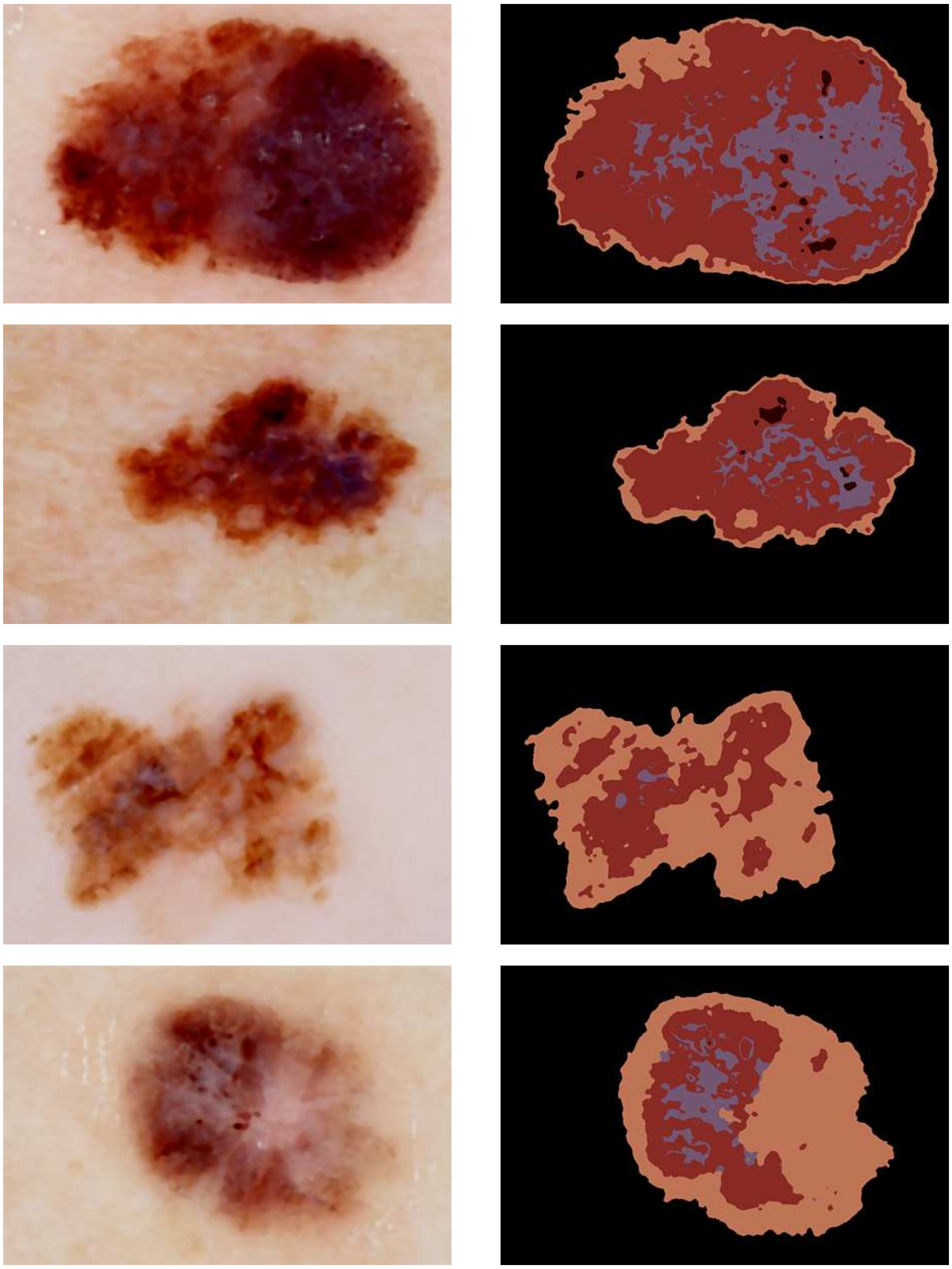

Figure 3.10: Examples of color detection in dermatoscopic images. Left column: Original median filtered images. Right column: Results of comparing the skin lesion images in the left column with color prototypes in the RGB colorspace using the Euclidean difference measure. Note, that all shades of blue are representated by the *blue1* prototype seen in figure 3.9, all shades of dark-brown by *dbrown2* and all shades of light-brown by *lbrown2*. The example in the fourth row shows some problems with detecting the milky-pink area in the center of the lesion. Dermatologists would consider this area to be either white or blue.

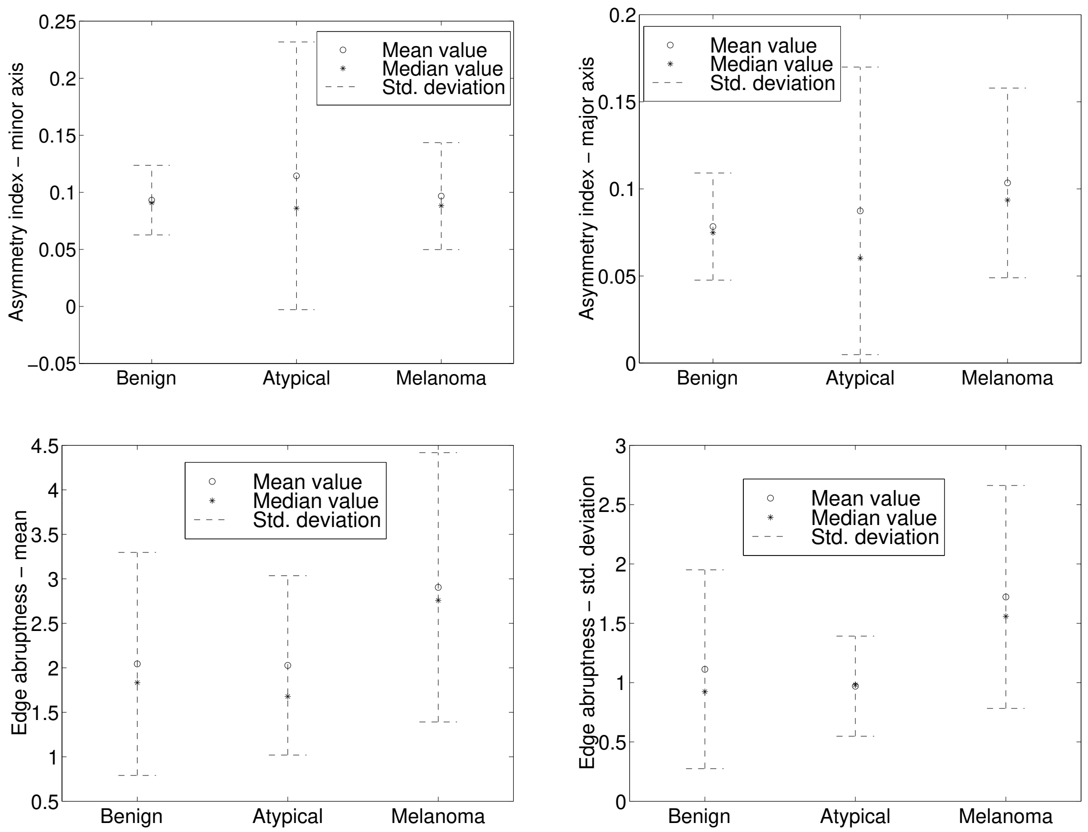

Figure 3.11: Plots showing the mean, median and standard deviation of the dermatoscopic features within each skin lesion class. Upper left: Asymmetry measure w.r.t. the minor principal axis. Upper right: Asymmetry measure w.r.t. the major principal axis. Lower left: The mean edge abruptness. Lower right: The standard deviation of the edge abruptness.

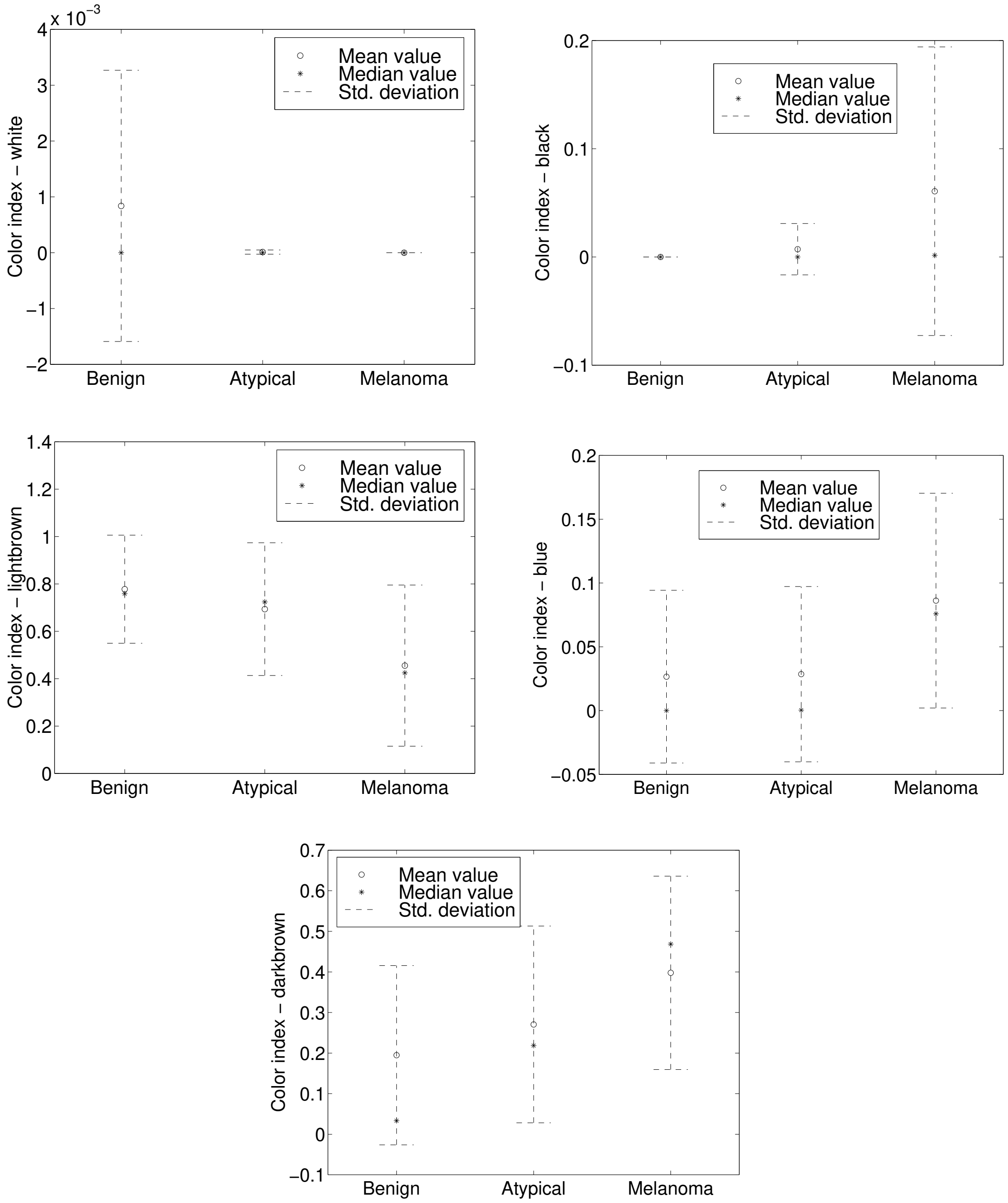

Figure 3.12: Plots showing the mean, median and standard deviation of the dermatoscopic features within each skin lesion class. Upper left: White color measure. Upper right: Black color measure. Center left: Light-brown color measure. Center right: Blue color measure. Lower center: Dark-brown color measure.

Chapter 4

A probabilistic framework for classification

In this chapter, a probabilistic framework for classification will be defined. The framework will be the foundation for designing neural network classifiers in chapter 5.

4.1 Bayes decision theory

Bayes decision theory is based on the assumption that the classification problem at hand can be expressed in probabilistic terms and that these terms are either known or can be estimated.

Suppose the classification problem is to map an input pattern $\mathbf{x}$ into a class $\mathcal{C}_l$ out of $n_\mathcal{C}$ classes where $l = 1, 2, \ldots, n_\mathcal{C}$. We can now define several probabilistic terms that are related through *Bayes' theorem* [Duda and Hart, 1973],

$$P(\mathcal{C}_l|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_l)P(\mathcal{C}_l)}{p(\mathbf{x})}. \tag{4.1}$$

$P(\mathcal{C}_l)$ is the class *prior* and reflects our prior belief of an unobserved pattern $\mathbf{x}$ belonging to class $\mathcal{C}_l$. $p(\mathbf{x}|\mathcal{C}_l)$ is the *class-conditional probability density function* and describes the probability characteristics of $\mathbf{x}$ once we know it belongs to class $\mathcal{C}_l$. The *posterior* probability is denoted by $P(\mathcal{C}_l|\mathbf{x})$ and is the probability of an observed pattern $\mathbf{x}$ belonging to class $\mathcal{C}_l$. The unconditional probability density function, $p(\mathbf{x})$, describing the density function for $\mathbf{x}$ regardless of the class, is given by

$$p(\mathbf{x}) = \sum_{l=1}^{n_\mathcal{C}} p(\mathbf{x}|\mathcal{C}_l)P(\mathcal{C}_l). \tag{4.2}$$

In short, Bayes' theorem shows how the observation of a pattern $\mathbf{x}$ changes the prior probability $P(\mathcal{C}_l)$ into a posterior probability $P(\mathcal{C}_l|\mathbf{x})$.

A classification system usually divides the input space into a set of $n_\mathcal{C}$ decision regions, $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{n_\mathcal{C}}$, so that a pattern, $\mathbf{x}$, located in $\mathcal{R}_l$ is assigned to class $\mathcal{C}_l$. The boundaries between the regions are called *decision boundaries*. Often the aim of a classifier is to minimize the probability of error, that is, to minimize the probability of classifying a pattern $\mathbf{x}$ belonging to class $\mathcal{C}_l$ as a different class due to $\mathbf{x}$ not being in decision region $\mathcal{R}_l$. The probability of error can be written as [Duda and Hart, 1973]

$$\begin{aligned} P(\text{error}) &= 1 - P(\text{correct}) && (4.3)\\ &= 1 - \sum_{l=1}^{n_\mathcal{C}} P(\mathbf{x} \in \mathcal{R}_l, \mathcal{C}_l) && (4.4)\\ &= 1 - \sum_{l=1}^{n_\mathcal{C}} P(\mathbf{x} \in \mathcal{R}_l | \mathcal{C}_l) P(\mathcal{C}_l) && (4.5)\\ &= 1 - \sum_{l=1}^{n_\mathcal{C}} \int_{\mathcal{R}_l} p(\mathbf{x}|\mathcal{C}_l) P(\mathcal{C}_l) d\mathbf{x}. && (4.6) \end{aligned}$$

In order to minimize the probability of error, the decision regions should be chosen so that each pattern, $\mathbf{x}$, is assigned to the class that has the integrand with the maximum value. That is, the pattern should be assigned to class $\mathcal{C}_l$ if

$$p(\mathbf{x}|\mathcal{C}_l)P(\mathcal{C}_l) > p(\mathbf{x}|\mathcal{C}_m)P(\mathcal{C}_m) \qquad \text{for all } l \neq m. \tag{4.7}$$

Rewriting the equation using Bayes' theorem (4.1) and the fact that $p(\mathbf{x})$ is independent of the class, we obtain *Bayes' minimum-error decision rule*,

$$P(\mathcal{C}_l|\mathbf{x}) > P(\mathcal{C}_m|\mathbf{x}) \qquad \text{for all } l \neq m, \tag{4.8}$$

saying, we should assign a pattern, $\mathbf{x}$, to the class with the highest posterior probability.

As already mentioned, Bayes' minimum-error decision rule assumes that the aim is to minimize the probability of error. This makes sense if every possible error is associated with the same cost. If this is not the case, one could adopt a *risk-based approach* as described in section 4.1.1. It may also be appropriate not to divide the entire input space into $n_\mathcal{C}$ decision regions. If a pattern has a low posterior probability for all classes, it may be beneficial to reject the pattern, rather than assigning it to a class. *Rejection thresholds* will be discussed in section 4.1.2.

4.1.1 Risk-based classification

Let us consider the malignant melanoma classification problem for a moment (see chapter 2 for an introduction). It is obvious that diagnosing a benign lesion as melanoma does not have the same consequences for the patient as diagnosing a melanoma as a benign lesion. In the first case, a biopsy of the lesion will be made and a correct diagnosis based on a histological analysis can be made. In the latter case, the cancer is not detected and may be left to develop to the extent where the patient can not be cured.

This is an example where it is not appropriate to just minimize the probability of error. An approach, where the different costs of misclassification are taken into account, would be desirable.

Let us denote the *loss* associated with assigning a pattern to class $\mathcal{C}_m$ when it belongs to class $\mathcal{C}_l$ with L_{lm}. For all patterns belonging to class $\mathcal{C}_l$, the expected loss is given by [Bishop, 1995]

$$R_l = \sum_{m=1}^{n_\mathcal{C}} L_{lm} \int_{\mathcal{R}_m} p(\mathbf{x}|\mathcal{C}_l) d\mathbf{x}. \tag{4.9}$$

The combined loss or *risk* for patterns from all classes is

$$\begin{aligned} R &= \sum_{l=1}^{n_\mathcal{C}} R_l P(\mathcal{C}_l) & (4.10) \\ &= \sum_{l=1}^{n_\mathcal{C}} \sum_{m=1}^{n_\mathcal{C}} \int_{\mathcal{R}_m} L_{lm} p(\mathbf{x}|\mathcal{C}_l) P(\mathcal{C}_l) d\mathbf{x} & (4.11) \\ &= \sum_{m=1}^{n_\mathcal{C}} \int_{\mathcal{R}_m} \left[\sum_{l=1}^{n_\mathcal{C}} L_{lm} p(\mathbf{x}|\mathcal{C}_l) P(\mathcal{C}_l) \right] d\mathbf{x} & (4.12) \end{aligned}$$

In order to minimize the risk, the decision regions should be chosen so that each pattern, $\mathbf{x}$, is assigned to the class that has the integrand with the minimum value. That is, the pattern should be assigned to class $\mathcal{C}_m$ if

$$\sum_{l=1}^{n_\mathcal{C}} L_{lm} p(\mathbf{x}|\mathcal{C}_l) P(\mathcal{C}_l) < \sum_{l=1}^{n_\mathcal{C}} L_{ln} p(\mathbf{x}|\mathcal{C}_l) P(\mathcal{C}_l) \qquad \text{for all } m \neq n. \tag{4.13}$$

Rewriting the equation using Bayes' theorem (4.1) and the fact that $p(\mathbf{x})$ is independent of the class, we obtain the *Bayes minimum-risk decision rule*,

$$\sum_{l=1}^{n_\mathcal{C}} L_{lm} P(\mathcal{C}_l|\mathbf{x}) < \sum_{l=1}^{n_\mathcal{C}} L_{ln} P(\mathcal{C}_l|\mathbf{x}) \qquad \text{for all } m \neq n, \tag{4.14}$$

which says, that a pattern, $\mathbf{x}$, should be assigned to the class with lowest risk. This rule reduces to Bayes' minimum-error decision rule if we assign a loss of 0 if the pattern is classified correctly and a loss of 1 if the pattern is assigned to a wrong class.

It can be a difficult task to quantify the loss. In, e.g., the melanoma classification problem, we would probably have to set the loss values manually, based on the views of experienced dermatologists. In other applications, it may be possible to set the loss values in a more systematic manner.

4.1.2 Rejection thresholds

Misclassification errors are most likely to occur in the proximity of areas with a large class overlap, i.e., where the largest posterior probability is relatively low compared to the class with the second largest posterior probability. This is often in the vicinity of the decision boundaries. Acknowledging this, it may be appropriate not to classify a pattern if the largest posterior probability is less than some rejection threshold. This could be the case in, e.g., the melanoma classification problem where we may not wish to rely on a classifier making a diagnosis in doubtful cases. This can be expressed by the following rejection rule [Duda and Hart, 1973],

$$\text{if} \quad P(\mathcal{C}_l|\mathbf{x}) \quad \begin{cases} \geq \tau, & \text{then classify } \mathbf{x} \\ < \tau, & \text{then reject } \mathbf{x} \end{cases}, \quad l = \arg\max_m \left[P(\mathcal{C}_m|\mathbf{x})\right], \tag{4.15}$$

where τ is the rejection threshold. Note that $\max_m \left[P(\mathcal{C}_m|\mathbf{x})\right] \geq \frac{1}{n_\mathcal{C}}$, so a rejection threshold smaller than $\frac{1}{n_\mathcal{C}}$ has no effect.

In [Hansen et al., 1997], it is argued that the majority of classification problems, that can be learned to a high degree of proficiency, exhibit an *effectively binary* character for most patterns, i.e., for a pattern, there is at most one predominantly alternative to the true classification class. This leads to the following error-reject relationship for near-optimal classifiers [Hansen et al., 1997],

$$\frac{E(R)}{E(0)} = \sqrt{1 + \left(\frac{R}{2E(0)}\right)^2} - \frac{R}{2E(0)}, \tag{4.16}$$

where $E(R)$ is the classification error for rejection rate[1] R and $E(0)$ the classification error for no rejects.

If we do not know the true posterior probabilities but only have an estimate of the posterior probabilities, this error-reject relationship can give us some indication on how accurate our estimates are. If our estimates are near-optimal, we should expect an error-reject relationship as described by equation (4.16).

In [Hintz-Madsen et al., 1995], an application, using the rejection theory described here, is presented.

4.2 Measuring model performance

Up until now, we have assumed that we either know the true posterior probabilities for the classes or that we have some estimate of the posterior probabilities. We will now introduce the notion of a model producing estimates of the posterior probabilities.

Assume we have a data set, $\mathcal{D}$, which we shall call a *training set*, consisting of $q_\mathcal{D}$ input-output pairs drawn from the joint probability distribution $p(\mathbf{x}, \mathbf{y})$

$$\mathcal{D} = \{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, \ldots, q_\mathcal{D}\}, \tag{4.17}$$

where $\mathbf{x}$ is an input pattern vector and $\mathbf{y}$ is an output vector containing the corresponding class label: $\mathbf{y}^\mathrm{T} = (y_1, y_2, \ldots, y_{n_\mathcal{C}})$ with $y_l = 1$, if $\mathbf{x} \in \mathcal{C}_l$, otherwise $y_l = 0$. This class labeling scheme is known as *1-of-$n_\mathcal{C}$* coding.

Let us also assume, we have a model, $\mathcal{M}$, parameterized by a vector, $\mathbf{u}$, that is estimated on the basis of the training set, $\mathcal{D}$, and let the model be capable of producing estimates of the posterior probabilities for the classes,

$$\mathcal{M}(\mathbf{u}) : \mathbf{x} \curvearrowright \hat{\mathbf{y}}, \tag{4.18}$$

[1] The fraction of rejected patterns.

where $\hat{\mathbf{y}}^{\mathrm{T}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{nc})$ contains estimates of the true posterior probabilities, i.e., $\hat{y}_l = \hat{P}(\mathcal{C}_l|\mathbf{x})$.

We can now use Bayes' theorem (4.1) to define several probabilistic terms for the model $\mathcal{M}$,

$$p(\mathbf{u}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{u})p(\mathbf{u})}{p(\mathcal{D})}. \tag{4.19}$$

$p(\mathbf{u})$ is the *parameter prior* and reflects our prior knowledge of the model parameters before observing any data. $p(\mathcal{D}|\mathbf{u})$ is the *likelihood of the model* and describes how probable it is that the data, $\mathcal{D}$, is generated by the model parameterized by $\mathbf{u}$. The *posterior parameter distribution* is denoted by $p(\mathbf{u}|\mathcal{D})$ and quantifies the probability distribution of the model parameters once the data has been observed. The unconditional probability distribution, $p(\mathcal{D})$, is a normalization factor and is given by

$$p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{u})p(\mathbf{u})d\mathbf{u}. \tag{4.20}$$

Now, in order to design a model as close to the true underlying model as possible, we may find the parameters that maximize the posterior parameter distribution,

$$\begin{aligned} \hat{\mathbf{u}}^{\mathrm{MAP}} &= \arg\max_{\mathbf{u}} \left[p(\mathbf{u}|\mathcal{D})\right] && (4.21)\\ &= \arg\max_{\mathbf{u}} \left[p(\mathcal{D}|\mathbf{u})p(\mathbf{u})\right]. && (4.22)\end{aligned}$$

This is known as *maximum a posteriori* (MAP) estimation.

If we have a uniform parameter prior, $p(\mathbf{u})$, the MAP estimate reduces to the *maximum likelihood* (ML) estimate,

$$\hat{\mathbf{u}}^{\mathrm{ML}} = \arg\max_{\mathbf{u}} \left[p(\mathcal{D}|\mathbf{u})\right]. \tag{4.23}$$

The MAP and ML estimate is based on the assumption that there is one near-optimal model matching the true model the best. Bayesians argue that one should use the entire posterior parameter distribution as a description of the model when doing output predictions. Examples of Bayesian approaches include David MacKay's *Bayesian framework for classification* based on approximating the posterior weight distribution [MacKay, 1992a], [MacKay, 1992b], [Thodberg, 1993] and *Markov Chain Monte Carlo* schemes based on sampling the posterior weight distribution [Duane et al., 1987], [Neal, 1994]. We will pursue the ML principle.

A complete probabilistic description of a classification problem is governed by the joint input-output probability distribution, $p(\mathbf{x}, \mathbf{y})$, that may be written as

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{4.24}$$

Since the model, $\mathcal{M}(\mathbf{u})$, approximates the true system, we have

$$p(\mathbf{y}, \mathbf{x}|\mathbf{u}) \approx p(\mathbf{y}, \mathbf{x}). \tag{4.25}$$

Assuming that the individual samples in $\mathcal{D}$ are drawn independently, the likelihood of the model can be written as

$$\begin{aligned} p(\mathcal{D}|\mathbf{u}) &= \prod_{\mu=1}^{q_{\mathcal{D}}} p(\mathbf{y}^{\mu}, \mathbf{x}^{\mu}|\mathbf{u}) && (4.26) \\ &= \prod_{\mu=1}^{q_{\mathcal{D}}} p(\mathbf{y}^{\mu}|\mathbf{x}^{\mu}, \mathbf{u}) p(\mathbf{x}^{\mu}). && (4.27) \end{aligned}$$

Instead of maximizing the likelihood, we may choose to minimize the negative logarithm[2] of the likelihood

$$-\log p(\mathcal{D}|\mathbf{u}) = -\sum_{\mu=1}^{q_{\mathcal{D}}} \left[\log p(\mathbf{y}^{\mu}|\mathbf{x}^{\mu}, \mathbf{u}) + \log p(\mathbf{x}^{\mu})\right]. \tag{4.28}$$

Since $p(\mathbf{x})$ is independent of the parameter vector, $\mathbf{u}$, we can discard this term from equation (4.28) and minimize the following function instead,

$$\begin{aligned} E_{\mathcal{D}}(\mathbf{u}) &= -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \log p(\mathbf{y}^{\mu}|\mathbf{x}^{\mu}, \mathbf{u}) && (4.29) \\ &= \frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} e(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}, \mathbf{u}), && (4.30) \end{aligned}$$

where $E_{\mathcal{D}}(\mathbf{u})$ is called an *error function* and $e(\mathbf{x}, \mathbf{y}, \mathbf{u})$ a *loss function*. Note, that the negative log-likelihood has been normalized with the number of samples in the training set $\mathcal{D}$, thus making $E_{\mathcal{D}}(\mathbf{u})$ an expression of the average pattern error.

Now, let us return to the MAP technique. As with the ML estimate, instead of maximizing the posterior parameter distribution, we can choose to minimize the negative logarithm of the posterior parameter distribution

$$-\log p(\mathcal{D}|\mathbf{u}) - \log p(\mathbf{u}) = -\sum_{\mu=1}^{q_{\mathcal{D}}} \left[\log p(\mathbf{y}^{\mu}|\mathbf{x}^{\mu}, \mathbf{u}) + \log p(\mathbf{x}^{\mu})\right] - \log p(\mathbf{u}). \tag{4.31}$$

Again we note that $p(\mathbf{x})$ is independent of $\mathbf{u}$, so we may discard this term and minimize the following function instead,

$$-\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \log p(\mathbf{y}^{\mu}|\mathbf{x}^{\mu}, \mathbf{u}) - \frac{1}{q_{\mathcal{D}}} \log p(\mathbf{u}). \tag{4.32}$$

[2]Since the logarithm is a monotonic function, the two approaches lead to the same results.

This function that we wish to minimize can now be written as

$$C(\mathbf{u}) = E_{\mathcal{D}}(\mathbf{u}) + R(\mathbf{u}), \tag{4.33}$$

where $C(\mathbf{u})$ is called a *cost function* and $R(\mathbf{u}) \propto -\frac{1}{q_{\mathcal{D}}} \log p(\mathbf{u})$ a *regularization function.* The latter is determined by the parameter prior and we shall return to this subject in section 4.4.1.

In the next section, we will derive a loss function for multiple-class problems based on the ML principle.

4.2.1 Cross-entropy error function for multiple classes

We will now consider the case where we have multiple exclusive classes, i.e., a pattern belongs to one and only one class. As in section 4.2, we assume that we have a model capable of producing estimates of the true posterior probabilities for the classes: $\hat{y}_l = \hat{P}(\mathcal{C}_l|\mathbf{x})$, we use a 1-of-$n_{\mathcal{C}}$ coding scheme for the class labeling and the distributions of the different class labels, y_l, are independent. The probability of observing a class label, $\mathbf{y}$, given a pattern, $\mathbf{x}$, is $\hat{P}(\mathcal{C}_l|\mathbf{x})$, if the true class is $\mathcal{C}_l$, which can be written as

$$p(\mathbf{y}|\mathbf{x}, \mathbf{u}) = \prod_{l=1}^{n_{\mathcal{C}}} (\hat{y}_l)^{y_l} . \tag{4.34}$$

Inserting equation (4.34) in equation (4.29), we obtain the following error function,

$$E_{\mathcal{D}}(\mathbf{u}) = -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{l=1}^{n_{\mathcal{C}}} y_l^{\mu} \log \hat{y}_l^{\mu}, \tag{4.35}$$

which is known as the *cross-entropy* error function [Bishop, 1995].

One of the characteristics of the cross-entropy error function is that it is very sensitive to *outliers*[3]. From the expression in equation (4.35), it is seen that a pattern, with a posterior probability close to zero for the true class, will give a very large contribution to the combined error due to the nature of the logarithmic function. To overcome this, we will in section 4.5 derive an outlier model and incorporate it into the cross-entropy error function, thus providing a more robust error function.

4.3 Measuring generalization performance

As already mentioned, we would like our model to be as close as possible to the true model described by $p(\mathbf{x}, \mathbf{y})$. In order to measure this, we define the *generalization ability* of a model as its ability to predict the output of the true model. Thus, the *generalization error* of a model can be defined as

[3] An outlier may be defined as a pattern with an erroneous class label or a very unlikely pattern, i.e., a pattern with a very low posterior probability for the true class.

$$
\begin{aligned}
G(\mathbf{u}) &= \langle e(\mathbf{x}, \mathbf{y}, \mathbf{u}) \rangle_{p(\mathbf{x},\mathbf{y})} &(4.36)\\
&= \int e(\mathbf{x}, \mathbf{y}, \mathbf{u}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} &(4.37)
\end{aligned}
$$

where the loss function, $e(\mathbf{x}, \mathbf{y}, \mathbf{u})$, could be, e.g, the cross-entropy error. The lower bound of $G(\mathbf{u})$ is $G(\mathbf{u}^\star)$, where $\mathbf{u}^\star$ denotes the parameters of the true model.

In the limit of an infinite training set, $\mathcal{D}$, the training error converges to the generalization error,

$$
\begin{aligned}
\lim_{q_\mathcal{D} \to \infty} E_\mathcal{D}(\mathbf{u}) &= \lim_{q_\mathcal{D} \to \infty} \frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} e(\mathbf{x}^\mu, \mathbf{y}^\mu, \mathbf{u}) &(4.38)\\
&= \int e(\mathbf{x}, \mathbf{y}, \mathbf{u}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}. &(4.39)
\end{aligned}
$$

Note that $G(\mathbf{u})$ is dependent on the training set through the model parameters $\mathbf{u}$. We may remove this dependency by defining the *expected generalization error* as the average generalization error w.r.t. all possible training sets of size $q_\mathcal{D}$,

$$
\begin{aligned}
\bar{G} &= \langle G(\mathbf{u}) \rangle_{p(\mathcal{D})} &(4.40)\\
&= \int G(\mathbf{u}) p(\mathcal{D}) d\mathcal{D}. &(4.41)
\end{aligned}
$$

Here we have acknowledged that the generalization error itself is a stochastic variable and defined the expected or average generalization error. We could equally well have defined other interesting measures like, e.g., the median. See [Larsen and Hansen, 1995] for a discussion of different generalization error statistics.

Usually, we do not know the true joint input-output distribution, $p(\mathbf{x}, \mathbf{y})$, and thus cannot determine neither $G(\mathbf{u})$ or $\bar{G}$. Instead, we can compute either empirical or algebraic estimates of these quantities which we shall discuss in the next two sections.

4.3.1 Empirical estimates

Sine we usually cannot assess the true joint input-output distribution, $p(\mathbf{x}, \mathbf{y})$, we may resolve to using empirical estimates of this distribution. As a first resort, we may consider the empirical distribution governed by the training set $\mathcal{D}$,

$$
\hat{p}_\mathcal{D}(\mathbf{x}, \mathbf{y}) = \frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \delta(\mathbf{x} - \mathbf{x}^\mu, \mathbf{y} - \mathbf{y}^\mu) \tag{4.42}
$$

where $\delta(\cdot, \cdot)$ is the Dirac delta function. Inserting equation (4.42) into equation (4.37), we obtain the following generalization error estimate based on the training set,

$$\hat{G}_{\mathcal{D}}(\mathbf{u}) = \int e(\mathbf{x},\mathbf{y},\mathbf{u})\hat{p}_{\mathcal{D}}(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} \tag{4.43}$$

$$= \frac{1}{q_{\mathcal{D}}}\sum_{\mu=1}^{q_{\mathcal{D}}} e(\mathbf{x}^{\mu},\mathbf{y}^{\mu},\mathbf{u}) \tag{4.44}$$

$$= E_{\mathcal{D}}(\mathbf{u}). \tag{4.45}$$

We have already seen that the training error, $E_{\mathcal{D}}(\mathbf{u})$, converges to the generalization error, $G(\mathbf{u})$, when the size of the training set is infinite. Now, in the case of a limited training set, it is usually a poor estimate of the generalization error. This is due to the model parameters being estimated on the basis of the training set which typically makes the training error underestimate the generalization error. In fact, we can always achieve a zero training error by employing a very flexible model even though we know that the minimal generalization error is $G(\mathbf{u}^{\star})$ which typically is not zero.

A better estimator can be achieved by employing a data set that is independent of the training set but drawn from the same true distribution $p(\mathbf{x},\mathbf{y})$. We call this a *test set*,

$$\mathcal{T} = \{(\mathbf{x}^{\mu},\mathbf{y}^{\mu}|\mu = 1,2,\ldots,q_{\mathcal{T}}). \tag{4.46}$$

If we use the empirical joint input-output distribution, $\hat{p}_{\mathcal{T}}(\mathbf{x},\mathbf{y})$, based on the test set, we may now use the *test error* as an estimate of the generalization error,

$$\hat{G}_{\mathcal{T}}(\mathbf{u}) = \frac{1}{q_{\mathcal{T}}}\sum_{\mu=1}^{q_{\mathcal{T}}} e(\mathbf{x}^{\mu},\mathbf{y}^{\mu},\mathbf{u}). \tag{4.47}$$

As with the training error, $\hat{G}_{\mathcal{T}}(\mathbf{u})$ converges to the generalization error, $G(\mathbf{u})$, when the test set, $\mathcal{T}$, is infinite.

Now, we would like the training set to be as large as possible in order to create an accurate model. At the same time the test set should be large in order to get a reliable estimate of the generalization ability of the model. Unfortunately, the available data is usually rather limited, so we have to deal with a trade-off between having a large training set and a large test set. A method trying to overcome this trade-off is called *cross-validation* [Stone, 1974], [Toussaint, 1974]. The idea of cross-validation is based on training and testing on disjunct subsets of data resampled from the available database. If we split the database up into K disjunct data sets, we may estimate a model using $K-1$ sets and evaluate its performance on the remaining set. This can be done K times resulting in K different models with K measures of the generalization performance. The cross-validation error is then defined as

$$\hat{G}_{\mathrm{CV}} = \frac{1}{K}\sum_{i=1}^{K} \hat{G}_{\mathcal{T}^{(i)}}(\mathbf{u}^{(i)}) \tag{4.48}$$

where $\hat{G}_{\mathcal{T}^{(i)}}(\mathbf{u}^{(i)})$ is the test error defined by equation(4.47) and i the split label. This provides us with an estimate of the expected generalization error defined by equation(4.40).

If each of the K disjunct data sets only contains one pattern, we obtain the special case called *leave-one-out cross-validation.*

Cross-validation has one major drawback, though, and that is the high computational costs involved. K models have to be estimated which for leave-one-out cross-validation corresponds to estimating as many models as there are available patterns in the data set. A scheme trying to remedy this based on linear unlearning of patterns has been proposed in [Hansen and Larsen, 1996]. An application using this technique is presented in [Sørensen et al., 1996].

4.3.2 Algebraic estimates

Empirical generalization error estimates require a fraction of the available data to be set aside thus reducing the amount of data available for the training set. And as stated previously, we would prefer a large training set in order to model the true model as accurately as possible.

In order to maximize the size of the training set, we will now consider an algebraic estimate of the average generalization error based only on the data in the training set. We will assume the following:

- Independence of input and error on output.
- There exists a set of parameters, $\mathbf{u}^\star$, that implements the true model, i.e., the chosen model architecture should be capable of implementing the true model.
- The number of patterns in the training set is large.

Under these assumptions, the following estimate of the average generalization error can be derived [Murata et al., 1991], [Amari and Murata, 1993], [Murata et al., 1994],

$$\langle G(\mathbf{u}) \rangle_{p(\mathcal{D})} \approx \langle E_{\mathcal{D}}(\mathbf{u}) \rangle_{p(\mathcal{D})} + \frac{n_{\text{eff}}}{q_{\mathcal{D}}}. \tag{4.49}$$

The effective number of parameters, n_{eff}, is given by

$$n_{\text{eff}} = \operatorname{tr}\left[\mathbf{J}^{\star-1}\mathbf{Q}^\star\right], \tag{4.50}$$

where $\mathbf{J}^\star$ is the Hessian matrix for the regularized cost function evaluated with the true model parameters $\mathbf{u}^\star$:,

$$\mathbf{J}^\star = \frac{\partial^2 C(\mathbf{u}^\star)}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} = \frac{\partial^2 E_{\mathcal{D}}(\mathbf{u}^\star)}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} + \frac{\partial^2 R(\mathbf{u}^\star)}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}, \tag{4.51}$$

and $\mathbf{Q}^\star$ is *Fishers information matrix* [Mardia et al., 1979],

$$\mathbf{Q}^\star = \left\langle \frac{\partial e(\mathbf{x}, \mathbf{y}, \mathbf{u}^\star)}{\partial \mathbf{u}} \frac{\partial e(\mathbf{x}, \mathbf{y}, \mathbf{u}^\star)}{\partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathcal{D})}, \tag{4.52}$$

where $e(\cdot)$ is the negative log-likelihood error for the individual patterns.

In a real world situation, we usually only have access to a single training set and we may thus replace the average training error, $\langle E_{\mathcal{D}}(\mathbf{u})\rangle_{p(\mathbf{D})}$, with the training error, $E_{\mathcal{D}}(\mathbf{u})$, for a single training set, replace[4] $\mathbf{Q}^\star$ with $\mathbf{H}$ and $\mathbf{J}^\star$ with $\mathbf{J}$,

$$\langle G(\mathbf{u})\rangle_{p(\mathcal{D})} \approx E_{\mathcal{D}}(\mathbf{u}) + \frac{1}{q_{\mathcal{D}}}\mathrm{tr}\left[\mathbf{J}^{-1}\mathbf{H}\right]. \tag{4.53}$$

This estimate may be used to select an optimal model among a hierarchy of models with decreasing complexity, i.e., every model should be a sub model of the previous model in the hierarchy [Murata et al., 1994].

For other texts on algebraic generalization error estimates, see, e.g., [Akaike, 1974], [Ljung, 1987], [Moody, 1992], [Larsen, 1993].

4.4 Controlling model complexity

When estimating models, we face the problem of choosing a model that has an appropriate complexity. That is, the model should be flexible enough to adequately model the underlying function of the true model. At the same time, we should ensure that the model is not too flexible in order not to capture the noise in the data. The latter case is known as *overfitting* [Hertz et al., 1991], [Bishop, 1995].

In brief, the purpose with controlling the model complexity is to maximize the generalization performance of the model. We will in the next two sections consider two such techniques based on parameter regularization and parameter pruning, respectively. Both methods are based on the assumption that the model is too complex.

4.4.1 Regularization techniques

As we saw in section 4.2, the MAP technique involves a prior for the model parameters and the cost function could thus be written as

$$C(\mathbf{u}) = E_{\mathcal{D}}(\mathbf{u}) + R(\mathbf{u}) \tag{4.54}$$

where $R(\mathbf{u}) \propto -\frac{1}{q_{\mathcal{D}}}\log p(\mathbf{u})$ is called a *regularization function.* We will now look at a particular choice of parameter prior.

4.4.1.1 Weight decay

In order to avoid overfitting, we should consider a prior that has the potential of limiting the model complexity by ensuring that the decision boundaries are smooth. One such prior that favors small parameters[5] is a zero mean Gaussian parameter prior with the individual parameters being independent,

$$p(u_k) = \frac{1}{\sqrt{(2\pi)1/\alpha_k}}\exp\left[-\frac{1}{2}\alpha_k u_k^2\right], \tag{4.55}$$

[4] Using Fisher's property: $\left\langle \partial^2 e(\mathbf{x},\mathbf{y},\mathbf{u})/\partial\mathbf{u}\partial\mathbf{u}^{\mathrm{T}}\right\rangle_{p(\mathcal{D})} = \left\langle \partial e(\mathbf{x},\mathbf{y},\mathbf{u})/\partial\mathbf{u}\,\partial e(\mathbf{x},\mathbf{y},\mathbf{u})/\partial\mathbf{u}^{\mathrm{T}}\right\rangle_{p(\mathcal{D})}$ [Seber and Wild, 1995]

[5] Here we assume that small parameters lead to very constrained models while large parameters allow very flexible models which will be the case for the neural network models considered in chapter 5.

where α_k is the inverse prior parameter variance that can be used for controlling the range of u_k. We can now write the normalized negative logarithm of the parameter prior as

$$-\frac{1}{q_{\mathcal{D}}}\log p(\mathbf{u}) = -\frac{1}{q_{\mathcal{D}}}\sum_{k=1}^{n_u}\log p(u_k) \tag{4.56}$$

$$= -\frac{1}{q_{\mathcal{D}}}\sum_{k=1}^{n_u}\left[-\frac{1}{2}\alpha_k u_k^2 - \log\frac{2\pi}{\alpha_k}\right], \tag{4.57}$$

where n_u is the total number of parameters.

We have seen from the MAP estimate that $R(\mathbf{u})$ should really equal $-\frac{1}{q_{\mathcal{D}}}\log p(\mathbf{u})$, but since the second term in equation (4.57), $\log\frac{2\pi}{\alpha_k}$, doesn't depend on $\mathbf{u}$ and we want to minimize $C(\mathbf{u}) = E_{\mathcal{D}}(\mathbf{u}) + R(\mathbf{u})$ with respect to $\mathbf{u}$, we may discard this term and define the regularization function as

$$R(\mathbf{u}) = \frac{1}{2q_{\mathcal{D}}}\sum_{k=1}^{n_u}\alpha_k u_k^2 \tag{4.58}$$

$$= \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}, \tag{4.59}$$

where $\mathbf{R}$ is a diagonal positive semidefinite matrix with elements $\alpha_k/q_{\mathcal{D}}$ in the diagonal. This particular form of the regularization function is called *weight decay* in the neural network community since it penalizes large parameters or weights whereas it for regression problems in traditional statistics is known as *ridge regression* when all α_k's are equal [Hoerl and Kennard, 1970]. The regularization parameters, α_k, are also known as *hyperparameters* since they themselves control other parameters, in this case, the model parameters.

We will later in chapter 5 discuss how we may determine the regularization parameters, α_k, in order to achieve an appropriate model complexity, by minimizing some estimate of the generalization error with respect to the regularization parameters. As a side effect, we will also see how regularization can suppress potential numerical problems that may occur when performing matrix inversions.

4.4.1.2 Other regularizers

The weight decay regularizer punishes large parameters severely which is not always desirable. To remedy this, a modification of the traditional weight decay regularizer has been suggested [Hertz et al., 1991]:

$$R(\mathbf{u}) = \frac{1}{2q_{\mathcal{D}}}\sum_{k=1}^{n_u}\alpha_k\frac{u_k^2}{1+u_k^2}. \tag{4.60}$$

This regularization function lets small parameters decay towards zero faster than large parameters [Hertz et al., 1991].

Another regularization function corresponding to a laplacian parameter prior is given by [Williams, 1995],

$$R(\mathbf{u}) = \frac{1}{2q_{\mathcal{D}}} \sum_{k=1}^{n_u} \alpha_k |u_k|. \tag{4.61}$$

For a thorough investigation into this regularization function showing that it effectively works more like a pruning technique than a regularization technique, see [Goutte, 1996], [Goutte and Hansen, 1997].

4.4.2 Pruning techniques

As we saw previously, we could limit the effect of a parameter or implicitly remove it by setting its regularization parameter, i.e., the inverse parameter variance, to a very large value. We could instead explicitly remove a parameter using one of several pruning techniques.

These methods are often based on computing the importance of each parameter by estimating the increase in an error measure that the removal of a parameter causes. All parameters are then ranked according to their importance denoted *saliency* and a percentage of the parameters with the lowest saliencies can be removed. The model is then re-estimated and the procedure is repeated until no parameters remain. This results in a family of models with decreasing complexity. For each model, an estimate of the generalization error may be computed and used for selecting the optimal model.

We will consider two variations of the pruning technique called *optimal brain damage* [Cun et al., 1990], that are both based on the following assumptions:

- The regularized cost function is at a minimum.
- The terms of third and higher order in a Taylor expansion of the error and regularized cost function can be neglected.
- The off-diagonal elements in the Hessian matrix can be neglected if more than one parameter is removed.

4.4.2.1 Training set based optimal brain damage

Let us assume that we have a model that has been optimized using the regularized cost function, $C(\mathbf{u})$, so that

$$\frac{\partial C(\hat{\mathbf{u}})}{\partial \mathbf{u}} = \frac{\partial E_{\mathcal{D}}(\hat{\mathbf{u}})}{\partial \mathbf{u}} + \mathbf{R}\hat{\mathbf{u}} = 0, \tag{4.62}$$

where $\hat{\mathbf{u}}$ is the minimum of the regularized cost function.

A second order Taylor expansion of the unregularized error function, $E_{\mathcal{D}}(\mathbf{u})$, around $\hat{\mathbf{u}}$ may be written as

$$E_{\mathcal{D}}(\mathbf{u}) = E_{\mathcal{D}}(\hat{\mathbf{u}}) + (\mathbf{u} - \hat{\mathbf{u}})^{\mathrm{T}} \frac{\partial E_{\mathcal{D}}(\hat{\mathbf{u}})}{\partial \mathbf{u}} + \frac{1}{2}(\mathbf{u} - \hat{\mathbf{u}})^{\mathrm{T}} \frac{\partial^2 E_{\mathcal{D}}(\hat{\mathbf{u}})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}(\mathbf{u} - \hat{\mathbf{u}}). \tag{4.63}$$

Inserting equation (4.62) in equation (4.63) yields

$$E_{\mathcal{D}}(\mathbf{u}) - E_{\mathcal{D}}(\hat{\mathbf{u}}) = -(\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}}\mathbf{R}\hat{\mathbf{u}} + \frac{1}{2}(\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}}\frac{\partial^2 E_{\mathcal{D}}(\hat{\mathbf{u}})}{\partial\mathbf{u}\partial\mathbf{u}^{\mathrm{T}}}(\mathbf{u}-\hat{\mathbf{u}}). \tag{4.64}$$

If we neglect[6] the off-diagonal elements of the Hessian matrix, utilize that $\mathbf{R}$ is a diagonal matrix with elements $\alpha_k/q_{\mathcal{D}}$ and set $(\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}} = (0,\ldots,-\hat{u}_k,\ldots,0)$, we can determine the saliency for the removed parameter $\hat{u}_k$,

$$s_k^{\mathrm{OBD}} = \left(\frac{\alpha_k}{q_{\mathcal{D}}} + \frac{1}{2}\mathbf{H}_{kk}\right)\hat{u}_k^2, \tag{4.65}$$

where $\mathbf{H}_{kk}$ is the $k'th$ diagonal element of the Hessian matrix.

4.4.2.2 Validation set based optimal brain damage

The saliency defined by equation (4.65) is computed by estimating the increase in training error when the parameter is removed. Since we are concerned with designing models with a high generalization ability, it's a natural progression to modify this approach in order to estimate the increase in generalization error and remove parameters exhibiting the smallest generalization error increase instead. An approach using an empirical estimate of the generalization error is called vOBD [Larsen et al., 1996].

vOBD is based on an estimate of the generalization error in form of a single validation set,

$$\mathcal{V} = \{(\mathbf{x}^\mu, \mathbf{y}^\mu) | \mu = 1, 2, \ldots, q_{\mathcal{V}}\}, \tag{4.66}$$

where $q_{\mathcal{V}}$ is the number of patterns in the validation set. The generalization error estimate is

$$\hat{G}_{\mathcal{V}}(\mathbf{u}) = \frac{1}{q_{\mathcal{V}}}\sum_{\mu=1}^{q_{\mathcal{V}}} e(\mathbf{x}^\mu, \mathbf{y}^\mu, \mathbf{u}). \tag{4.67}$$

As with the training error, we do a second order Taylor expansion of the validation error around the regularized cost function minimum $\hat{\mathbf{u}}$:

$$\hat{G}_{\mathcal{V}}(\mathbf{u}) = \hat{G}_{\mathcal{V}}(\hat{\mathbf{u}}) + (\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}}\frac{\partial \hat{G}_{\mathcal{V}}(\hat{\mathbf{u}})}{\partial\mathbf{u}} + \frac{1}{2}(\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}}\frac{\partial^2 \hat{G}_{\mathcal{V}}(\hat{\mathbf{u}})}{\partial\mathbf{u}\partial\mathbf{u}^{\mathrm{T}}}(\mathbf{u}-\hat{\mathbf{u}}). \tag{4.68}$$

Again neglecting[6] the off-diagonal elements in the validation set based Hessian and setting $(\mathbf{u}-\hat{\mathbf{u}})^{\mathrm{T}} = (0,\ldots,-\hat{u}_k,\ldots,0)$, we can estimate the change in validation error due to the removal of parameter $\hat{u}_k$ as

$$s_k^{\mathrm{vOBD}} = -\hat{u}_k\frac{\partial \hat{G}_{\mathcal{V}}(\hat{\mathbf{u}})}{\partial u_k} + \frac{1}{2}\mathbf{H}_{kk}^{\mathrm{v}}\hat{u}_k^2, \tag{4.69}$$

where $\mathbf{H}^{\mathrm{v}}$ is the validation set based Hessian matrix.

[6] If only one parameter is removed, this assumption is not necessary.

4.4.2.3 Other pruning techniques

The vOBD method was based on estimating the change in an empirical generalization error estimate when removing a parameter. One could also consider using an algebraic estimate of the generalization error like the one defined by equation (4.49). Such an approach called γOBD is described in [Pedersen et al., 1996] for regression problems.

Recall that one of the assumptions in the OBD and vOBD derivation, is that the off-diagonal elements in the Hessian matrix can be neglected if more than one parameter is removed, i.e., the removal of a parameter does not affect the saliencies of the remaining parameters. This assumption often doesn't hold. An extension of OBD called *optimal brain surgeon* (OBS) does not make this assumption [Hassibi et al., 1992]. OBS also estimates how the remaining parameters should be changed when removing a parameter, thus reducing the need for re-estimating the model. Experience shows that OBD and OBS lead to similar performance, though. In [Pedersen, 1997], it is shown that OBD outperforms OBS in some cases due to the second order Taylor expansion being too crude leading to underestimation of the OBS saliencies.

4.5 Defining an outlier model

We will now introduce the notion of an *outlier* as a pattern, $\mathbf{x}$, with a low posterior probability for the class defined by $\mathbf{y}$. This situation may arise if the class label, $\mathbf{y}$, is wrong due to, e.g., a mistake in the data sampling process[7].

We will define the probability of an outlier, $\epsilon \in [0;1]$, as the probability of a class label, $\mathcal{C}_l$, erroneously changing to one of the other classes, $\mathcal{C}_k$, where $k \neq l$, and we will define $\beta = \epsilon/(n_{\mathcal{C}}-1) \in [0; 1/(n_{\mathcal{C}}-1)]$ as the probability of a class label, $\mathcal{C}_l$, changing specifically to $\mathcal{C}_k$. We will refer to ϵ as the outlier probability and to β as the scaled outlier probability.

Assuming that the process generating outliers is independent of input location and class label, we may view the outlier process as an extra input-output independent error source as opposed to the errors occurring due to the overlap in class posterior probabilities.

Denoting the posterior probability in the case of zero outlier probability $P_0(\mathcal{C}_l|\mathbf{x})$, we can now redefine the posterior probability by incorporating the outlier probability as

$$P(\mathcal{C}_l|\mathbf{x}) = P_0(\mathcal{C}_l|\mathbf{x})(1-\epsilon) + \sum_{k=1,k\neq l}^{n_{\mathcal{C}}} P_0(\mathcal{C}_k|\mathbf{x})\beta. \tag{4.70}$$

The first term is the probability that $\mathbf{x}$ isn't an outlier, i.e., the posterior probability for zero outlier probability times the probability that an outlier does not occur. The second term sum is the probability that $\mathbf{x}$ is an outlier and consists of the posterior probability for zero outlier probability times the probability that $\mathcal{C}_k$ has changed specifically to $\mathcal{C}_l$.

Using $\sum_{k=1,k\neq l}^{n_{\mathcal{C}}} P_0(\mathcal{C}_k|\mathbf{x}) = 1 - P_0(\mathcal{C}_l|\mathbf{x})$, we may rewrite equation (4.70) as

$$P(\mathcal{C}_l|\mathbf{x}) = P_0(\mathcal{C}_l|\mathbf{x})(1-\beta n_{\mathcal{C}}) + \beta, \tag{4.71}$$

where $\beta \leq P(\mathcal{C}_l|\mathbf{x}) \leq 1-\epsilon$ and $\sum_{l=1}^{n_{\mathcal{C}}} P(\mathcal{C}_l|\mathbf{x}) = 1$, since $0 \leq P_0(\mathcal{C}_l|\mathbf{x}) \leq 1$ and $\sum_{l=1}^{n_{\mathcal{C}}} P_0(\mathcal{C}_l|\mathbf{x}) = 1$. The potential risk of outliers thus introduces a lower and upper bound on the posterior probabilities.

[7] Note, a low posterior probability for the class defined by $\mathbf{y}$ does not necessarily mean that the class label is wrong. It may just as well be a very unlikely pattern.

Note, that the introduction of the outlier probability does not change the decision boundaries for $\beta < 1/n_{\mathcal{C}}$, since Bayes' minimum-error decision rule in this case yields the same results when using either $P(\mathcal{C}_l|\mathbf{x})$ or $P_0(\mathcal{C}_l|\mathbf{x})$:

$$\begin{aligned} &\mathbf{x} \in \mathcal{C}_l, \text{ if for all } l \neq m \\ P(\mathcal{C}_l|\mathbf{x}) &> P(\mathcal{C}_m|\mathbf{x}) && (4.72) \\ &\Updownarrow \\ P_0(\mathcal{C}_l|\mathbf{x})(1 - n_{\mathcal{C}}\beta) + \beta &> P(\mathcal{C}_m|\mathbf{x})(1 - n_{\mathcal{C}}\beta) + \beta && (4.73) \\ &\Updownarrow \\ P_0(\mathcal{C}_l|\mathbf{x}) &> P_0(\mathcal{C}_m|\mathbf{x}). && (4.74) \end{aligned}$$

For $1/n_{\mathcal{C}} < \beta \leq 1/(n_{\mathcal{C}} - 1)$, the Bayes' minimum-error decision boundaries still don't change but the actual decision does,

$$\begin{aligned} &\mathbf{x} \in \mathcal{C}_l, \text{ if for all } l \neq m \\ P(\mathcal{C}_l|\mathbf{x}) &> P(\mathcal{C}_m|\mathbf{x}) && (4.75) \\ &\Updownarrow \\ P_0(\mathcal{C}_l|\mathbf{x}) &< P_0(\mathcal{C}_m|\mathbf{x}), && (4.76) \end{aligned}$$

since $(1 - n_{\mathcal{C}}\beta) < 0$. This is very unlikely to occur in practical applications, though. If, e.g., we have a problem with $n_{\mathcal{C}} = 3$ classes, then $1/n_{\mathcal{C}} < \beta \leq 1/(n_{\mathcal{C}} - 1)$ corresponds to $2/3 < \epsilon \leq 1$. That is, the probability of a pattern being an outlier is at least 0.66 thus indicating that the data set is severely corrupted and unsuitable for modeling.

4.5.1 Outlier-modified cross-entropy error function

With the introduction of outliers, we will now modify the cross-entropy error function introduced in section 4.2.1 to incorporate the outlier probability.

Recall, that the cross-entropy error function is defined as

$$E_{\mathcal{D}}(\mathbf{u}) = -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{l=1}^{n_{\mathcal{C}}} y_l^{\mu} \log \hat{y}_l^{\mu}, \tag{4.77}$$

where $\hat{y}_l = \hat{P}(\mathcal{C}_l|\mathbf{x})$ is an estimate of the posterior probability that $\mathbf{x}$ belongs to class $\mathcal{C}_l$ and $\mathbf{u}$ is the model providing posterior probability estimates. As can be seen from equation (4.77), an outlier (a small posterior probability $\hat{y}_l$ for $y_l = 1$) can give a very large contribution to the cross-entropy error, $E_{\mathcal{D}}(u)$, due to the nature of the logarithmic function.

To remedy this, let $\hat{z}_l = \hat{P}_0(\mathcal{C}_l|\mathbf{x})$ denote the estimate of the posterior probability in the case of zero outlier probability, and let us define the modified *cross-entropy error function* by inserting equation (4.71) into equation (4.77):

$$E_{\mathcal{D}}(\mathbf{u}) = -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{l=1}^{n_{\mathcal{C}}} y_l^{\mu} \log \left[\hat{z}_l^{\mu}(1 - \beta n_{\mathcal{C}}) + \beta \right]. \tag{4.78}$$

As can be seen, there is a lower bound, β, on the argument to the logarithmic function. This effectively downweights large error contributions from outliers to maximum of $\log \beta$, thus providing a better estimate of the cross-entropy error for the non-outlier posterior probability distributions. We will in chapter 6 see how this helps designing accurate neural networks classifiers and generating robust empirical generalization error estimates.

In figure 4.1, an example of the effect of incorporating the outlier probability into the cross-entropy error function is shown.

4.5.2 Detecting outliers

The outlier framework enables a method for detecting outliers. Suppose we want to know how probable it is that a pattern, $\mathbf{x}$, with label $\mathcal{C}_l$ is an outlier. Let us first define a binary variable, O, that indicates if a pattern is an outlier or not,

$$O = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is an outlier} \\ 0 & \text{otherwise} \end{cases}. \tag{4.79}$$

We may now write the probability, that a pattern, $\mathbf{x}$, with label $\mathcal{C}_l$ is an outlier, as

$$\begin{aligned} P_{\text{outlier}} &= P(O = 1|\mathbf{x}, \mathcal{C}_l) && (4.80) \\ &= \frac{P(O = 1, \mathcal{C}_l|\mathbf{x})}{P(\mathcal{C}_l|\mathbf{x})}, && (4.81) \end{aligned}$$

where we have used the probability rule, $P(a, b|c) = P(a|b, c)P(b|c)$. The nominator in equation (4.81) is the posterior probability in the case of an outlier and is given by the second term in equation (4.70). The denominator is the posterior probability given by equation (4.71). We thus have

$$P_{\text{outlier}} = \frac{\beta(1 - P_0(\mathcal{C}_l|\mathbf{x}))}{P_0(\mathcal{C}_l|\mathbf{x})(1 - \beta n_{\mathcal{C}}) + \beta}. \tag{4.82}$$

In a modeling context, we only have estimates of $P_0(\mathcal{C}_l|\mathbf{x})$ and β and may use these to estimate how probable it is that labeled data are outliers. Note, that we need a class label for a pattern, $\mathbf{x}$, in order to estimate the probability that the pattern is an outlier. For future patterns without a class label, we have no way of determining if the pattern is an outlier or not in this framework. We do though take the outlier process into account during modeling which results in models that are likely to generalize better, i.e., perform better on future patterns.

The probability, P_{outlier}, may be used for inspecting the labeled data. Patterns that are very likely to be outliers may be relabeled or one may discover atypical patterns that may contribute to a better understanding of the problem at hand.

In figure 4.2, an example illustrating the effects different outlier percentages have on P_{outlier} is shown.

In appendix F, an example of using equation (4.82) for identifying outliers is shown.

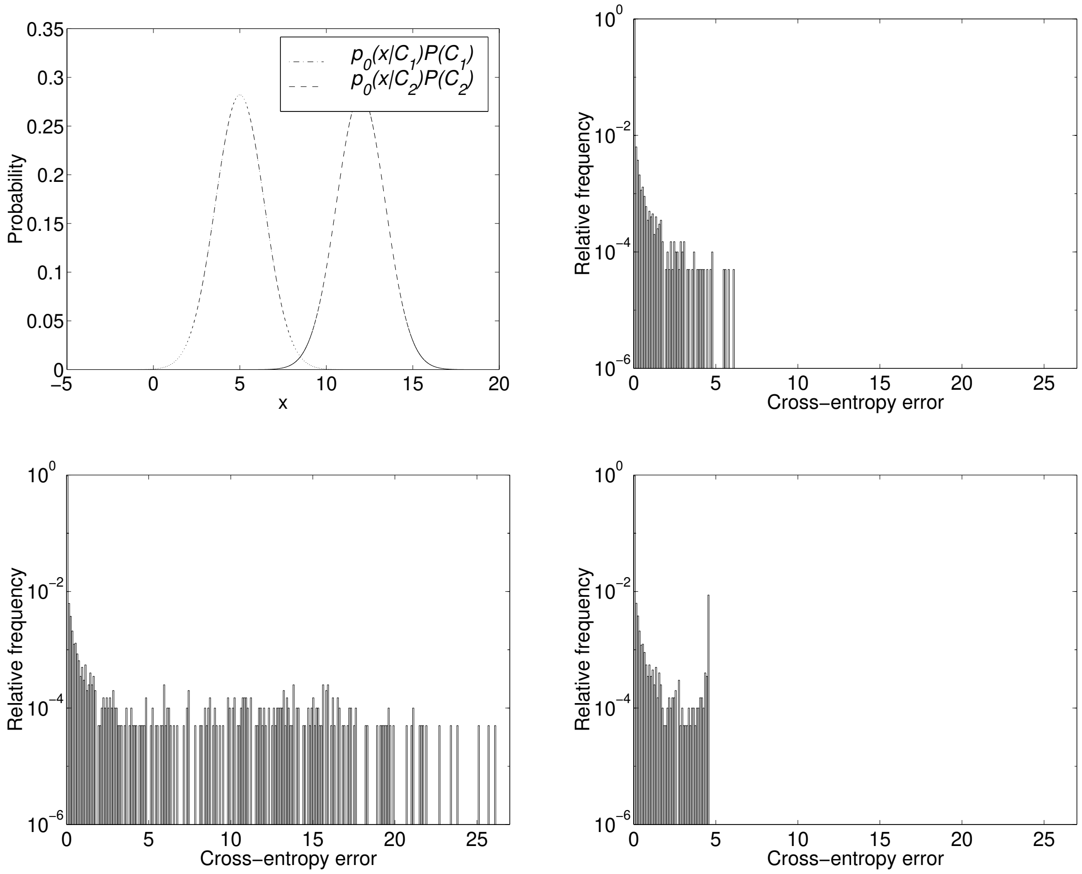

Figure 4.1: Artificial two class problem showing the effect of outliers and the modified cross-entropy error function. Upper left: The probability distributions, $p_0(x|\mathcal{C}_1)P(\mathcal{C}_1)$ and $p_0(x|\mathcal{C}_2)P(\mathcal{C}_2)$, for zero outlier probability are shown, where $p_0(x|\mathcal{C}_1)$ and $p_0(x|\mathcal{C}_2)$ are Gaussians with means 5 and 12 and variances 2 and 2, respectively, and $P(\mathcal{C}_1) = P(\mathcal{C}_2) = 0.5$. Upper right: Histogram of the cross-entropy error for the individual samples (10000 in each class) using the true posterior distributions for zero outlier probability and the true class labels. The mean cross-entropy error is $E_{\mathcal{D}}(\mathbf{u}) = 0.018$ where $\mathbf{u}$ indicates the true underlying model generating posterior probabilities in the case of zero outlier probability. Lower left: Histogram of the individual cross-entropy errors when 1% ($\epsilon = \beta = .01$) of the class labels have been changed to the wrong class and again using the true posterior distributions for zero outlier probability. Note how the outliers introduce very large errors affecting the mean cross-entropy error, $E_{\mathcal{D}}(\mathbf{u}) = 0.146$, significantly. Lower right: Histogram of the individual cross-entropy errors when 1% of the class labels have been changed to the wrong class but this time using the outlier-modified posterior probabilities, i.e., equation (4.78) resulting in a mean cross-entropy error of $E_{\mathcal{D}}(\mathbf{u}) = 0.072$. Note, how the maximum error of a sample now is $-\log \beta = 4.6$ corresponding to the right peak in the histogram thus effectively downweighting the error contribution from outliers.

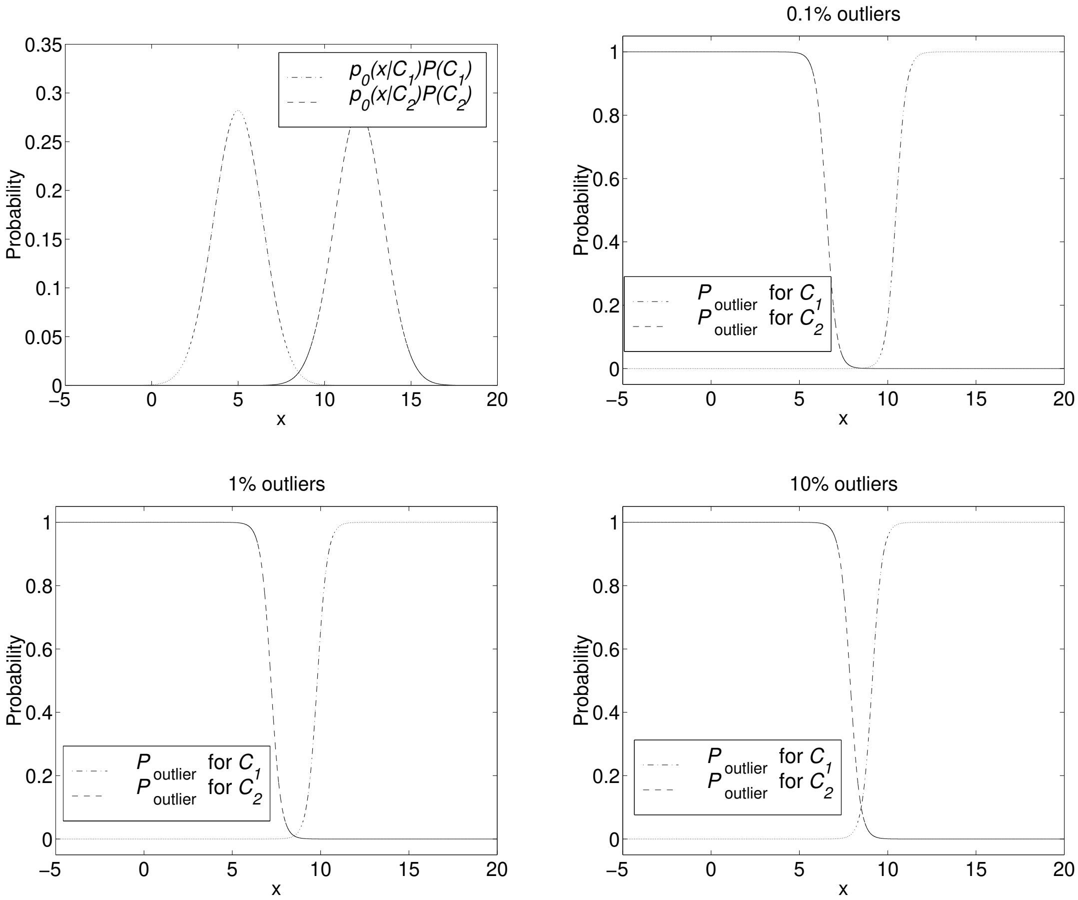

Figure 4.2: Artificial two class problem showing the effect that various outlier percentages have on the probability that a pattern, $\mathbf{x}$, labeled $\mathcal{C}_l$ is an outlier. Upper left: The probability distributions, $p_0(x|\mathcal{C}_1)P(\mathcal{C}_1)$ and $p_0(x|\mathcal{C}_2)P(\mathcal{C}_2)$, for zero outlier probability are shown, where $p_0(x|\mathcal{C}_1)$ and $p_0(x|\mathcal{C}_2)$ are Gaussians with means 5 and 12 and variances 2 and 2, respectively, and $P(\mathcal{C}_1) = P(\mathcal{C}_2) = 0.5$. Upper right: The probability that $\mathbf{x}$ is an outlier for class $\mathcal{C}_1$ and $\mathcal{C}_2$ in the case of 0.1% outliers ($\beta = \epsilon = .001$). Note, there is a rather wide region around the optimal class decision boundary where P_{outlier} is close to zero. This is the region where errors are more likely to occur due to class overlap than due to outliers. This explains the low probability. Lower left: 1% outliers. The region with P_{outlier} close to zero is now narrower due to the larger outlier percentage. Lower right: 10% outliers. Note, that for a two-class problem, it is easy to show that at the point where P_{outlier} is the same for the two classes, P_{outlier} will be equal to β.

Chapter 5

Neural classifier modeling

In this chapter, we will suggest algorithms for designing neural networks for supervised classification based on the probabilistic framework described in chapter 4. In particular, we will focus on feed-forward multi-layer perceptrons with appropriate output normalization in order to enable the classifier outputs to be interpreted as posterior probabilities.

5.1 Introduction

The traditional approach to classification is statistical and concerns the modeling of stationary class-conditional probability distributions by a set of basis functions, e.g., Parzen windows or Gaussian mixtures [Duda and Hart, 1973], [Bishop, 1995], [Ripley, 1996].

Neural networks have in the last decade been employed extensively for classification applications. The two most common neural network architectures for supervised classification are the multi-layer perceptron and the radial basis function network with two layers of weights. These may be represented by the network diagram in figure 5.1. The radial

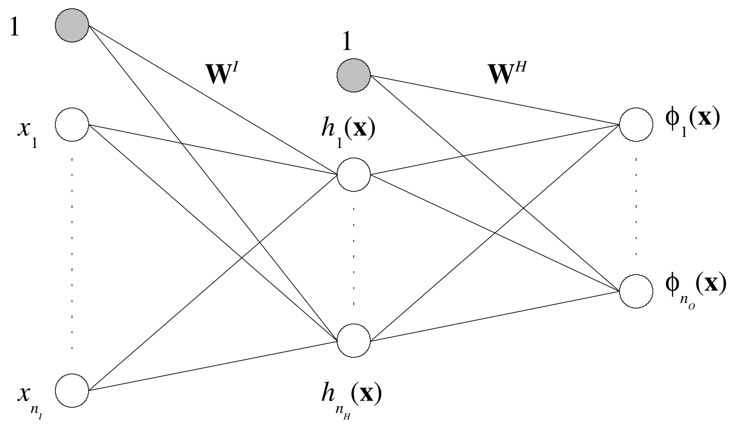

Figure 5.1: Two-layer neural network structure. By choosing appropriate basis or activation functions for the hidden units, $h_j(\mathbf{x})$, and output units, $\phi_i(\mathbf{x})$, the radial basis function network or the multi-layer perceptron may be defined. The shaded units are threshold units, $\mathbf{W}^I$ is a matrix of weights connecting input units with hidden units and $\mathbf{W}^H$ and $\mathbf{W}^H$ is a matrix of weights connecting hidden units with output units.

basis function network is typically defined by hidden unit basis functions measuring some distance between the input vector and a prototype vector and by linear output functions whereas the multi-layer perceptron typically is defined by sigmoidal hidden unit basis functions and by linear or non-linear output functions. We will consider the multi-layer perceptron architecture in greater detail in the next section.

Both classes of neural networks possess the important *universal approximation* capability, i.e., they may approximate any given function[1] with arbitrary precision as long as the number of hidden units are large enough [Park and Sandberg, 1991],[Ripley, 1996]. Since neural networks *learn by example*, they are particular effective in situations where no suitable traditional statistic model may be identified, i.e., knowledge about the true data-generating system is poor.

Radial basis function networks will not be discussed any further. For a more thorough introduction, see, e.g., [Bishop, 1995].

5.2 Multi-layer perceptron architecture

We will now focus on two-layer perceptrons and define a particular model architecture that is used throughout the rest of this thesis.

The hidden unit activation function used is the hyperbolic tangent function. Thus, the output of the hidden units for a pattern, $\mathbf{x}^\mu$, may be written as

$$h_j(\mathbf{x}^\mu) = \tanh\left(\sum_{k=1}^{n_I} w_{jk}^I x_k^\mu + w_{j0}^I\right), \quad j = 1, 2, \ldots, n_H, \tag{5.1}$$

where w_{jk}^I is the weight connecting input k and hidden unit j, w_{j0}^I is the threshold for hidden unit j, n_I is the number of inputs and n_H is the number of hidden units.

The hidden unit outputs are weighted and summed, yielding the following unbounded network outputs,

$$\phi_i(\mathbf{x}^\mu) = \sum_{j=1}^{n_H} w_{ij}^H h_j(\mathbf{x}^\mu) + w_{i0}^H, \quad i = 1, 2, \ldots, n_O, \tag{5.2}$$

where w_{ij}^H is the weight connecting hidden unit j and the unbounded output unit i, w_{i0}^H is the threshold for the unbounded output unit i and n_O is the number of unbounded output units.

In order to employ the probabilistic framework derived in chapter 4, the neural classifier outputs must be normalized so that the classifier may be used for estimating posterior probabilities. We will now consider two slightly different normalization schemes and discuss their properties.

5.2.1 Softmax normalization

The standard way of ensuring that network outputs may be interpreted as probabilities is by using the normalized exponential transformation know as *softmax* [Bridle, 1990],

[1] If the network output function imposes bounds on the the output values, the networks can of course only approximate equally bounded functions.

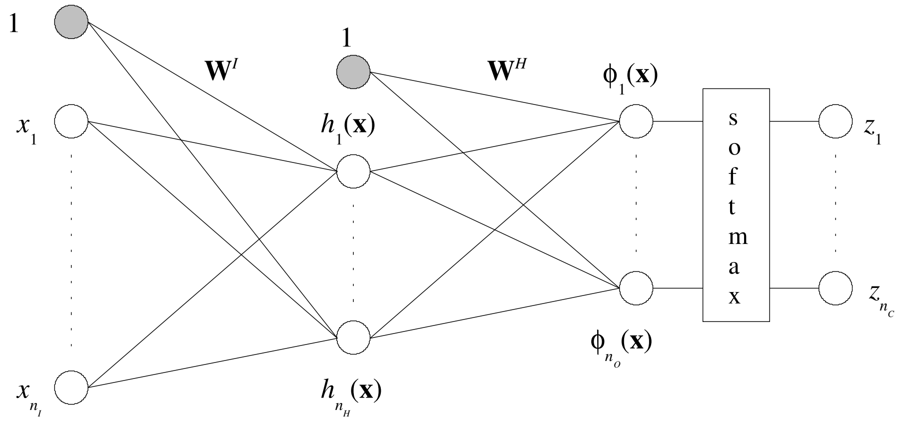

Figure 5.2: The standard two-layer softmax network. This has an inherent output redundancy yielding the unregularized Hessian singular for a well-trained network.

$$\hat{z}_i^\mu = \hat{P}(\mathcal{C}_i|\mathbf{x}^\mu) = \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{\sum_{i'=1}^{n_O} \exp[\phi_{i'}(\mathbf{x}^\mu)]}, \tag{5.3}$$

where $\hat{z}_i^\mu$ is short for the estimated posterior probability that the pattern, $\mathbf{x}^\mu$, belongs to class $\mathcal{C}_i$. We thus have the following properties

$$0 \leq \hat{z}_i^\mu \leq 1, \qquad \sum_{i=1}^{n_O} \hat{z}_i^\mu = 1. \tag{5.4}$$

As can be seen, the softmax normalization introduces a redundancy in the output representation due to the property that the posterior probability estimates for a pattern sum to one.

An effect of this is that the unregularized Hessian matrix for a well-trained network will be singular due to the output redundancy resulting in a dependency between the weights going to one output unit and the weights going to the other output units. This effectively reduces the rank of the unregularized Hessian matrix by the number of hidden units plus one (threshold unit). Any computations involving the inverse Hessian matrix will be affected by this. The problem is reduced by employing regularization since this usually reestablishes the full rank of the regularized Hessian.

The standard softmax network is shown in figure 5.2.

5.2.2 Modified softmax normalization

In order to remove the output redundancy introduced by the standard softmax normalization, we may remove one of the unbounded outputs. This yields the following *modified softmax* normalization,

$$\hat{z}_i^\mu = \begin{cases} \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{1+\sum_{i'=1}^{n_\mathcal{C}-1} \exp[\phi_{i'}(\mathbf{x}^\mu)]}, & \text{for } i = 1, 2, \dots, n_\mathcal{C}-1 \\ 1 - \sum_{i'=1}^{n_\mathcal{C}-1} \hat{z}_{i'}^\mu, & \text{for } i = n_\mathcal{C} \end{cases}, \tag{5.5}$$

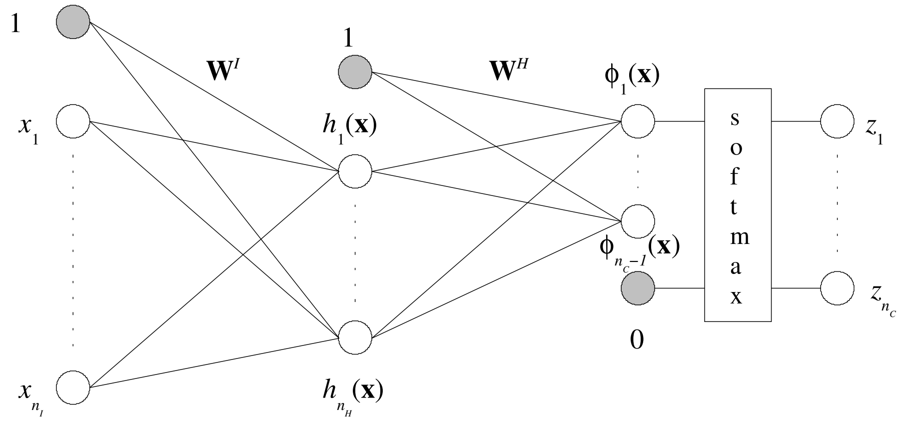

Figure 5.3: The modified two-layer softmax network. This does not have the inherent output redundancy.

where $n_{\mathcal{C}}$ is the number of classes.

Another way of obtaining this modification is by removing all input connections to the unbounded output $\phi_{n_{\mathcal{C}}}(\cdot)$ and setting $\phi_{n_{\mathcal{C}}}(\cdot)$ to zero for all input patterns. Using the standard softmax normalization (5.3), we now effectively obtain the modified softmax normalization. This is illustrated in figure 5.3.

The modified softmax normalization has several benefits compared to the standard softmax normalization. With a certain number of hidden units, the modified softmax normalization reduces the network complexity, i.e., the number of weight parameters is reduced by the number of hidden units plus one. This improves the number of training patterns per weight relationship. The dependency between weights is removed, thus improving the performance of algorithms based on the computation of the inverse Hessian, e.g., the Newton scheme of updating weights that will be discussed in section 5.3.1.2. Another example where the modified softmax normalization may be beneficial is in MacKay's Bayesian framework for classification [MacKay, 1992a], [MacKay, 1992b]. This framework approximates the posterior probability distribution of the weights by a Gaussian distribution centered on the MAP solution of the weights and with the inverse Hessian as covariance matrix. The posterior class probabilities are then found by using the entire posterior weight distribution. Any inaccuracies in the Hessian may in this framework affect the results considerably.

The modified softmax scheme is recommended for output normalization.

5.3 Estimating model parameters

With the neural classifier architecture in place, we now need to address the task of estimating the model parameters. We will pursue the MAP approach with some weight prior and with incorporated outlier model. As we recall from chapter 4, this yields the following cost function

$$C(\mathbf{u}) = E_{\mathcal{D}}(\mathbf{u}) + R(\mathbf{u}), \tag{5.6}$$

where $\mathbf{u}$ is a column vector containing all n_u network weights and thresholds, $E_{\mathcal{D}}(\mathbf{u})$ the outlier-modified cross-entropy error function (4.78) and $R(\mathbf{u})$ a regularization function proportional to the log weight prior.

Given a training set, $\mathcal{D}$, and a validation set, $\mathcal{V}$, we will now suggest methods for estimating the model parameters, i.e., the weights and thresholds, the regularization parameters and the outlier probability.

5.3.1 Weight parameters

We will first address the problem of finding the MAP solution for the network weights and thresholds using a training set, $\mathcal{D}$, of size $q_{\mathcal{D}}$. The optimization methods will be based on gradient and curvature information.

Common for these approaches is an iterative weight updating scheme that may be formulated as

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} + \Delta\mathbf{u}^{(t)}, \tag{5.7}$$

where t indicates the iteration timestep and $\Delta\mathbf{u}^{(t)}$ the weight parameter change.

In the following sections, we will need the first and second derivatives of the outlier-modified cross-entropy error function w.r.t. the weights. Note, if the modified softmax normalization is used then $\phi_{n_{\mathcal{C}}}(\mathbf{x}^\mu) = 0$ in the following.

The gradient is

$$\frac{\partial E_{\mathcal{D}}(\mathbf{u})}{\partial u_j} = -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{1}{\hat{y}_i^\mu} y_i^\mu (1-\beta n_{\mathcal{C}}) \frac{\partial \hat{z}_i^\mu}{\partial u_j}, \tag{5.8}$$

and the Hessian,

$$\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u})}{\partial u_j \partial u_k} = -\frac{(1-\beta n_{\mathcal{C}})}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{y_i^\mu}{\hat{y}_i^\mu} \left[-\frac{(1-\beta n_{\mathcal{C}})}{\hat{y}_i^\mu} \frac{\partial \hat{z}_i^\mu}{\partial u_k} \frac{\partial \hat{z}_i^\mu}{\partial u_j} + \frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} \right], \tag{5.9}$$

where the derivative of the non outlier-modified posterior probability w.r.t. the weights is given by

$$\frac{\partial \hat{z}_i^\mu}{\partial u_j} = \hat{z}_i^\mu \sum_{i'=1}^{n_{\mathcal{C}}} (\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}, \tag{5.10}$$

and the second derivative is given by

$$\begin{aligned} \frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} \quad = \quad & \sum_{i'=1}^{n_{\mathcal{C}}} \left[\left((\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial \hat{z}_i^\mu}{\partial u_k} - \hat{z}_i^\mu \frac{\partial \hat{z}_{i'}^\mu}{\partial u_k} \right) \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \right. \\ & \left. + \hat{z}_i^\mu (\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial^2 \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j \partial u_k} \right]. \end{aligned} \tag{5.11}$$

Note, that we have expressed the derivatives as a function of the derivatives for a standard neural network with linear outputs: $\partial\phi_{i'}(\mathbf{x}^\mu)/\partial u_j$ and $\partial^2\phi_{i'}(\mathbf{x}^\mu)/\partial u_j\partial u_k$.

It is often desirable for computational reasons to use the Gauss-Newton approximation of the Hessian instead,

$$\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u})}{\partial u_j \partial u_k} = \frac{(1-\beta n_{\mathcal{C}})^2}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{1}{\hat{y}_i^\mu} \frac{\partial \hat{z}_i^\mu}{\partial u_k} \frac{\partial \hat{z}_i^\mu}{\partial u_j}. \tag{5.12}$$

An important property of this approximation is that the Hessian is guaranteed to be positive semi-definite, thus ensuring that a Newton step is a descent direction. The Newton algorithm will be described in section 5.3.1.2.

The detailed derivations of equation (5.8)-(5.12) may be found in appendix A.

5.3.1.1 Gradient descent optimization

One of the simplest optimization algorithms is *gradient descent* also known as *steepest descent*. It is based on iteratively updating the weight vector so that we move in the direction of the largest rate of decrease of the cost function, i.e., in the direction of the negative gradient of the cost function evaluated at timestep t. This may be written as

$$\begin{aligned} \Delta\mathbf{u}^{(t)} &= -\eta \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}} && (5.13)\\ &= -\eta \left(\frac{\partial E_{\mathcal{D}}(\mathbf{u}^{(t)})}{\partial \mathbf{u}} + \frac{\partial R(\mathbf{u}^{(t)})}{\partial \mathbf{u}} \right). && (5.14) \end{aligned}$$

If we insert equation (5.13) into a first order Taylor expansion of the cost function around the weight vector $\mathbf{u}^{(t)}$ at timestep t, we obtain,

$$\begin{aligned} C(\mathbf{u}^{(t+1)}) &\approx C(\mathbf{u}^{(t)}) + \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}^{\mathrm{T}}} \Delta\mathbf{u}^{(t)} && (5.15)\\ &\approx C(\mathbf{u}^{(t)}) - \eta \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}^{\mathrm{T}}} \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}}, && (5.16) \end{aligned}$$

where η is called a *learning rate* that ensures that the cost error decreases for each iteration when η is sufficiently small in order not to seriously invalidate the first order approximation.

It is clear that a too small learning rate will result in slow convergence while a too large learning rate will yield the first order approximation inadequate which may result in an error increase. A simple approach for iteratively adapting the learning rate is described in the following first order optimization scheme with fixed regularization parameters and outlier probability[2]:

1. Initialize weights, e.g., uniformly over $[-0.5; 0.5]$.
2. Compute $C(\mathbf{u}^{(t)})$, initialize[3] the learning rate, η and compute the weight parameter change, $\Delta\mathbf{u}^{(t)} = -\eta\partial C(\mathbf{u}^{(t)})/\partial\mathbf{u}^{(t)}$.

[2] We will later in section 5.3.2 see how we may estimate these.

[3] Through initial experiments, a suitable value may be found.

3. Update the weights, $\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} + \Delta\mathbf{u}^{(t)}$, and compute $C(\mathbf{u}^{(t+1)})$.

4. If $C(\mathbf{u}^{(t+1)}) > C(\mathbf{u}^{(t)})$, then set $\eta = \eta/2$ and goto step 3.

5. If the convergence criteria[4] is not met, then set $t = t+1$ and goto step 2.

This simple gradient descent scheme is not very efficient but it may be employed when more sophisticated optimization schemes are not applicable. This could, e.g., be the case in the startup phase for optimization algorithms based on a second order Taylor expansion where the quadratic approximation initially may be poor. That is, the gradient descent algorithm may be applied as initialization for more advanced optimization schemes.

5.3.1.2 Newton optimization

There are several optimization algorithms based on a second order Taylor expansion of the cost function. We will focus on the *Newton* optimization method [Hertz et al., 1991].

Let us consider a second order Taylor expansion of $C(\mathbf{u})$ at timestep t around the weight vector $\mathbf{u}^{(t)}$,

$$\begin{aligned} C(\mathbf{u}^{(t+1)}) \approx C(\mathbf{u}^{(t)}) &+ \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}^{\mathrm{T}}}(\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)}) \\ &+ \frac{1}{2}(\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)})^{\mathrm{T}} \frac{\partial^2 C(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}(\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)}). \end{aligned} \tag{5.17}$$

If the quadratic approximation is exact, we may find the weight vector $\mathbf{u}^{(t+1)}$ that minimizes the cost function by setting the derivative of equation. (5.17) equal to zero,

$$\frac{\partial C(\mathbf{u}^{(t+1)})}{\partial \mathbf{u}} = \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}} + \frac{\partial^2 C(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}(\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)}) = 0 \tag{5.18}$$

$$\Downarrow$$

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \left(\frac{\partial^2 C(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}\right)^{-1} \frac{\partial C(\mathbf{u}^{(t)})}{\partial \mathbf{u}} \tag{5.19}$$

$$= \mathbf{u}^{(t)} - \left(\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} + \frac{\partial^2 R(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}\right)^{-1} \left(\frac{\partial E_{\mathcal{D}}(\mathbf{u}^{(t)})}{\partial \mathbf{u}} + \frac{\partial R(\mathbf{u}^{(t)})}{\partial \mathbf{u}}\right) \tag{5.20}$$

If the quadratic approximation is poor, the full weight update step may not result in a cost error decrease. As with the first order approximation, we therefore introduce a stepsize parameter, η, that can be used for ensuring a cost error decrease,

$$\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)} - \eta \left(\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} + \frac{\partial^2 R(\mathbf{u}^{(t)})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}\right)^{-1} \left(\frac{\partial E_{\mathcal{D}}(\mathbf{u}^{(t)})}{\partial \mathbf{u}} + \frac{\partial R(\mathbf{u}^{(t)})}{\partial \mathbf{u}}\right). \tag{5.21}$$

The Newton algorithm may be formulated as the following iterative scheme for fixed regularization parameters and outlier probability:

[4] This could, e.g., be when the 2-norm of the gradient is below some small value.

1. Initialize weights, use, e.g., the gradient descent scheme in section 5.3.1.1.

2. Compute $C(\mathbf{u}^{(t)})$ and initialize the step size, η, to 1.

3. Update the weights according to eq. (5.21) and compute $C(\mathbf{u}^{(t+1)})$.

4. If $C(\mathbf{u}^{(t+1)}) > C(\mathbf{u}^{(t)})$, then set $\eta = \eta/2$ and goto step 3.

5. If the convergence criteria[5] is not met, then set $t = t + 1$ and goto step 2.

The Newton algorithm converges in very few iterations but may be computational expensive due to the need for computing and inverting the regularized Hessian.

5.3.2 Regularization parameters and outlier probability

We have now discussed methods for estimating the weights and thresholds of the model. The next step is to define methods for estimating the remaining model parameters, i.e., the regularization parameters and the outlier probability. We will employ an independent validation set, $\mathcal{V}$, of size $q_{\mathcal{V}}$ for this purpose. That is, we want to minimize the validation error w.r.t. the remaining model parameters. As noted in section 4.3.1, an independent data set may be used as an estimate of the true generalization error, $G(\mathbf{u})$,

$$\hat{G}_{\mathcal{V}}(\mathbf{u}) = \frac{1}{q_{\mathcal{V}}} \sum_{\mu=1}^{q_{\mathcal{V}}} e(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}, \mathbf{u}), \tag{5.22}$$

where $e(\mathbf{x}^{\mu}, \mathbf{y}^{\mu}, \mathbf{u})$ is the outlier-modified cross-entropy error for a single pattern. Thus, this approach aims at minimizing an estimate of the generalization error.

Let $\boldsymbol{\theta}^{\mathrm{T}} = [\boldsymbol{\alpha}^{\mathrm{T}}\ \beta]$ be a column vector containing all regularization parameters, $\boldsymbol{\alpha}$, and the scaled outlier probability, β. We will only consider the case where the regularization function is linear in the regularization parameters,

$$R(\mathbf{u}) = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{r}(\mathbf{u}) = \sum_{i=1}^{n_\alpha} \alpha_i r_i(\mathbf{u}), \tag{5.23}$$

where n_α is the number of regularization parameters and $r_i(\mathbf{u})$ some function of the weights and thresholds, e.g., $1/(2q_{\mathcal{D}})u_i^2$, which is the standard weight decay regularizer described in section 4.4.1.1.

We suggest using the *conjugate gradient* optimization scheme that will be described in the next section. This requires the gradient of the validation error w.r.t. the regularization parameters and the scaled outlier probability, $\partial \hat{G}_{\mathcal{V}}(\mathbf{u})/\partial \boldsymbol{\theta}$. Note, that the network weights, $\mathbf{u}$, are implicitly dependent on the regularization parameters and the scaled outlier probability. To be notational correct, we should thus use $\mathbf{u}(\boldsymbol{\theta})$ to represent the weights but for convenience this notation is not adopted.

In vector notation, the gradient w.r.t. the regularization parameters is (see appendix B for details)

[5] This could, e.g., be when the 2-norm of the gradient is below some small value.

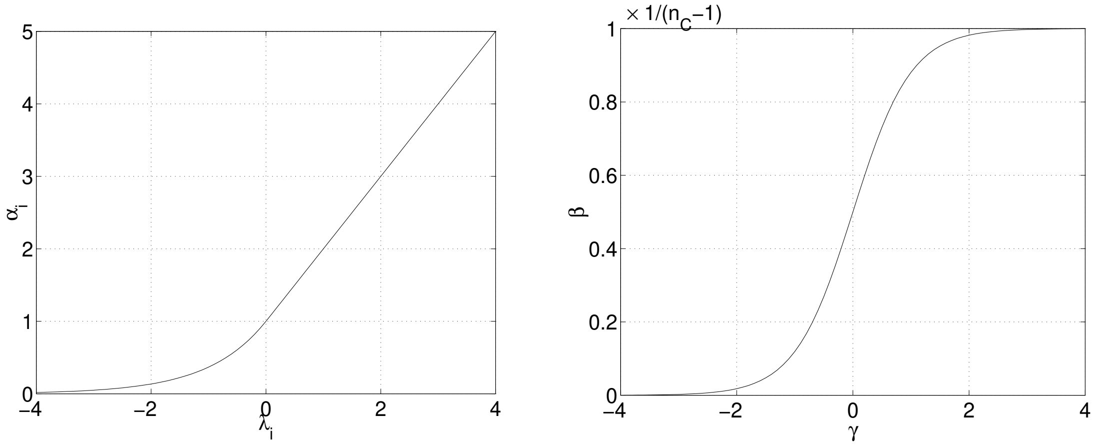

Figure 5.4: Reparameterization functions for the regularization parameters (left) and the scaled outlier probability (right). These functions ensure that the parameters during minimization of the validation error are confined to valid regions in parameter space, i.e., $\alpha_i \geq 0$ and $0 \leq \beta \leq 1/(n_{\mathcal{C}}-1)$.

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \boldsymbol{\alpha}} = -\frac{\partial \mathbf{r}(\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \left(\frac{\partial^2 C(\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}\right)^{-1} \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \mathbf{u}}, \tag{5.24}$$

where $\partial \hat{G}_{\mathcal{V}}(\mathbf{u})/\partial \mathbf{u}$ is the gradient defined in equation (5.8) but evaluated on the validation set.

The gradient w.r.t. the scaled outlier probability is given by (see appendix B for details)

$$\begin{aligned} \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \beta} &= -\frac{1}{q_{\mathcal{V}}} \sum_{\mu=1}^{q_{\mathcal{V}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{1 - n_{\mathcal{C}} \hat{z}_i^{\mu}}{\hat{y}_i^{\mu}} y_i^{\mu} \\ & \quad - \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \left(\frac{\partial^2 C(\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}}\right)^{-1} \left(\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{y_i^{\mu}}{(\hat{y}_i^{\mu})^2} \frac{\partial \hat{z}_i^{\mu}}{\partial \mathbf{u}}\right), \end{aligned} \tag{5.25}$$

where $\partial \hat{z}_i^{\mu}/\partial \mathbf{u}$ is given by equation (5.10).

In order to ensure that the regularization parameters are non-negative and thus can be interpreted as inverse weight prior variances, we perform a reparameterization,

$$\alpha_i = \begin{cases} \exp(\lambda_i), & \lambda_i < 0 \\ \lambda_i + 1, & \lambda_i \geq 0 \end{cases}. \tag{5.26}$$

Recall from section 4.5 that $0 \leq \beta \leq 1/(n_{\mathcal{C}}-1)$. To ensure that this holds, we do a reparameterization of β,

$$\beta = \frac{1 + \tanh(\gamma)}{2(n_{\mathcal{C}}-1)}. \tag{5.27}$$

Optimization may now be done w.r.t. the new parameters, $\boldsymbol{\lambda}$ and γ, collected in the column vector $\boldsymbol{\xi}^{\mathrm{T}} = [\boldsymbol{\lambda}^{\mathrm{T}}\ \gamma]$ instead. The gradient w.r.t. the new parameters is given by the simple expression,

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \xi_i} = \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \theta_i} \frac{\partial \theta_i}{\partial \xi_i}, \tag{5.28}$$

where we have already computed $\partial \hat{G}_{\mathcal{V}}(\mathbf{u})/\partial \theta_i$ and where $\partial \theta_i/\partial \xi_i$ are simple derivatives of the reparameterization equations.

We now have gradient expressions and by using the reparameterization we have ensured that the regularization parameters and the scaled outlier probability are confined to valid regions in parameter space during optimization. Next, we will consider using the conjugate gradient algorithm for optimizing these parameters.

5.3.2.1 Conjugate gradient optimization

The *conjugate gradient* optimization algorithm increases the optimization efficiency compared to the gradient descent method by ensuring that successive search directions, $\mathbf{h}^{(t+1)}$ and $\mathbf{h}^{(t)}$, are conjugate. That is, using a second order Taylor approximation of the error function, the following should hold,

$$(\mathbf{h}^{(t+1)})^{\mathrm{T}} \mathbf{H}^{(t+1)} \mathbf{h}^{(t)} = 0, \tag{5.29}$$

where $\mathbf{H}^{(t+1)}$ is the Hessian of the error function at timestep $t+1$.

Once a search direction is found, the parameters may be updated by the following iterative scheme,

$$\boldsymbol{\xi}^{(t+1)} = \boldsymbol{\xi}^{(t)} + \eta_{\min} \mathbf{h}^{(t)}, \tag{5.30}$$

where $\eta_{\min}$ is found by doing a line search that minimizes $\hat{G}_{\mathcal{V}}(\mathbf{u}^{(t)})$ along the direction of $\mathbf{h}^{(t)}$.

Let us denote the gradient of the validation error w.r.t. the regularization parameters and the scaled outlier probability by

$$\mathbf{g}^{(t)} = \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}^{(t)})}{\partial \boldsymbol{\xi}}, \tag{5.31}$$

where we recall that $\mathbf{u}^{(t)}$ is an implicit function of $\boldsymbol{\xi}$.

We may now write the update equation for the search direction[6], $\mathbf{h}^{(t)}$, as the sum of the negative gradient direction and a fraction of the previous search direction:

$$\mathbf{h}^{(t)} = -\mathbf{g}^{(t)} + \tau_{t-1} \mathbf{h}^{(t-1)}, \tag{5.32}$$

where τ_{t-1} is given by the *Polak-Ribiere* form [Bishop, 1995],

[6]For the first iteration, the search direction is set to the negative gradient direction.

$$\tau_{t-1} = \frac{(\mathbf{g}^{(t)})^{\mathrm{T}}(\mathbf{g}^{(t)} - \mathbf{g}^{(t-1)})}{(\mathbf{g}^{(t-1)})^{\mathrm{T}}\mathbf{g}^{(t-1)}}. \tag{5.33}$$

There are other forms for τ_{t-1} that are identical if the error is exactly quadratic. The Polak-Ribiere form is generally preferred since, if the algorithm is making little progress, so that successive gradient directions are similar, then τ_i will be very small, thus resetting the search direction to the negative gradient direction.

One of the properties of the conjugate gradient algorithm is that it guarantees to the find the exact minimum of a n_ξ-dimensional quadratic error function in at most n_ξ iterations. We rarely have a quadratic error function but nevertheless normally experience a significant speed increasement compared to the gradient descent scheme due to the error function being approximatively quadratic locally. Another property is that the algorithm does not require the computation of the Hessian that in some cases may be computational prohibitive.

The conjugate gradient algorithm for optimizing the regularization parameters and scaled outlier probability may be formulated as the following scheme where we employ a simple approximative line search method:

1. Initialize $\boldsymbol{\xi}^{(t)}$, e.g., set regularization parameters and the scaled outlier probability close to zero.

2. Train the network with fixed $\boldsymbol{\xi}^{(t)}$ using, e.g., the Newton algorithm in order to find the optimal weights, $\mathbf{u}^{(t)}$.

3. Compute $\hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t)})$, $\mathbf{g}^{(t)}$ and $\mathbf{h}^{(t)}$ using eq. (5.31)-(5.33). Initialize line search parameter, $\eta_{\min}$, to 1.

4. Update parameters, $\boldsymbol{\xi}^{(t+1)} = \boldsymbol{\xi}^{(t)} + \eta_{\min}\mathbf{h}^{(t)}$. Find optimal weights, $\mathbf{u}^{(t+1)}$, and compute $\hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t+1)})$.

5. If $\hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t+1)}) < \hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t)})$, then repeat:

   - $\eta_{\min} = \eta_{\min} \cdot 2$,
   - do step 4,

   until $\hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t+1)}) \geq \hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t)})$. Restore previous $\mathbf{u}^{(t+1)}$ and $\boldsymbol{\xi}^{(t+1)}$, and goto step 7.

6. Repeat:

   - $\eta_{\min} = \eta_{\min}/2$,
   - do step 4,

   until $\hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t+1)}) < \hat{G}_{\mathcal{V}}(\boldsymbol{\xi}^{(t)})$. Goto step 7.

7. If the convergence criteria[7] is not met, then t=t+1. Goto step 3.

Note, the search direction, $\mathbf{h}^{(t)}$, tends to deteriorate as the algorithm progresses. Thus, it is beneficial to reset the search direction to the negative gradient direction every, e.g., $2n_\xi$ iterations, where n_ξ is the dimension of the $\boldsymbol{\xi}$ vector.

[7] This could, e.g., be when the 2-norm of the gradient, $\mathbf{g}^{(t+1)}$, is below some small value.

5.4 Design algorithm overview

Based on the optimization algorithms described in this chapter and the probabilistic framework described in chapter 4, we will suggest two different schemes for designing neural network classifiers with appropriate model complexity.

5.4.1 Algorithm 1: Adaptive optimization of regularization parameters and outlier probability

This algorithm deals with adaptive estimation of the regularization parameters in order to achieve an appropriate model complexity and with adaptive estimation of the outlier probability in order to reduce the influence of possible outliers. As described in section 5.3.2, this requires a validation set. Thus, a rather large amount of data should be available for this algorithm to be successful.

A flowchart of the algorithm is shown in figure 5.5 and is also described in section 5.3.2. We will briefly comment on some of the components of the algorithm:

Model initialization The available data should be split into a training, validation and test set. The split ratios are often chosen so that the training and validation set have the same size and the test set has the same size as the combined training and validation set. See, e.g., [Goutte and Larsen, 1998] for an empirical assessment of the effects of different split rations. The regularization parameters should be initialized to small values, yet large enough to avoid numerical instabilities when doing the first Newton optimization step (see section 6.1.2 for details).

Model complexity optimization The suggested line search method used for the conjugate gradient optimization of the regularization parameters and the outlier probability is only an approximate line search method. In brief, the approximate line search method takes as large a step as possible in the conjugate gradient direction by doubling or bisecting the line search parameter. Thus, we do not do a thorough search for computational reasons. Recall, that for each update of the regularization parameters and the outlier probability, we need to do a full Newton optimization of the weights.

Model retraining After the determination of the regularization parameters and the outlier probability, it may be beneficial to retrain the weights on the combined training and validation set with the regularization parameters and the outlier probability fixed to their determined optimal values.

After designing a classifier using this algorithm, Bayes minimum-risk decision rule and rejection thresholds may be applied as described in section 4.1.1 and 4.1.2. It is also possible to inspect the training and validation set for outliers as described in section 4.5.2.

The algorithm will be used for an artificial generated classification problem in section 6.1 in order to investigate the effects of incorporating the outlier model.

In the paper in appendix E, an example of using the algorithm extended with validation set based optimal brain damage pruning but without the outlier model is shown.

5.4.2 Algorithm 2: Adaptive optimization of network architecture

This algorithm deals with adaptive estimation of the network architecture by using the optimal brain damage pruning technique described in section 4.4.2. The regularization parameters and outlier probability are fixed throughout the pruning scheme. The algebraic test error estimate described in section 4.3.2 is used for selection of the optimal network architecture. This algorithm is appropriate when the amount of data is limited, i.e., when holding out data for a validation set for model selection or estimation of regularization parameters and outlier probability can not be justified.

A flowchart of the algorithm is shown in figure 5.6. We will briefly comment on some of the components of the algorithm:

Model initialization The available data should be split into a training and a test set. The split ratio is often chosen so that the training and test set have the same size. The regularization parameters and outlier probability should be fixed throughout the algorithm. These parameters may be found, e.g., by sampling the algebraic test error estimate as a function of these parameters and choose those that minimize the algebraic test error estimate. An example of this is shown in the paper in appendix G.

Model complexity optimization The optimal brain damage saliencies are computed from the training set. After the removal of a weight, only a few Newton iterations are necessary in order to find a new minimum.

Optimal model selection The model with the lowest algebraic test error estimate is selected. Recall, that the algebraic test error estimate is an asymptotic estimate. Thus, for small training sets, the validity of the estimate may be questionable.

After designing a classifier using this algorithm, Bayes minimum-risk decision rule and rejection thresholds may be applied.

The algorithm will be used for the malignant melanoma problem in section 6.2.

In the papers in appendix C and D, examples of using the algorithm are shown.

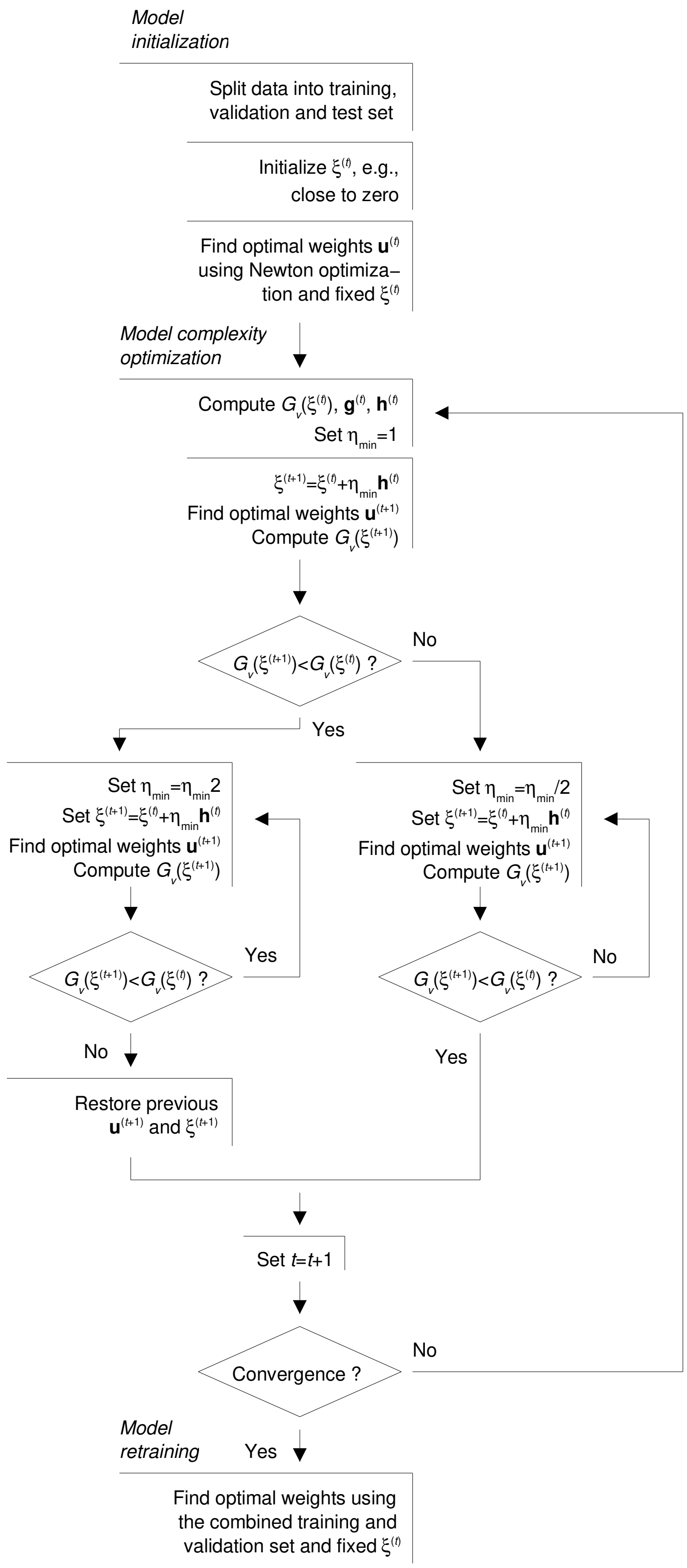

Figure 5.5: Scheme for designing neural classifiers with adaptive estimation of regularization parameters and outlier probability. This algorithm requires a large data set.

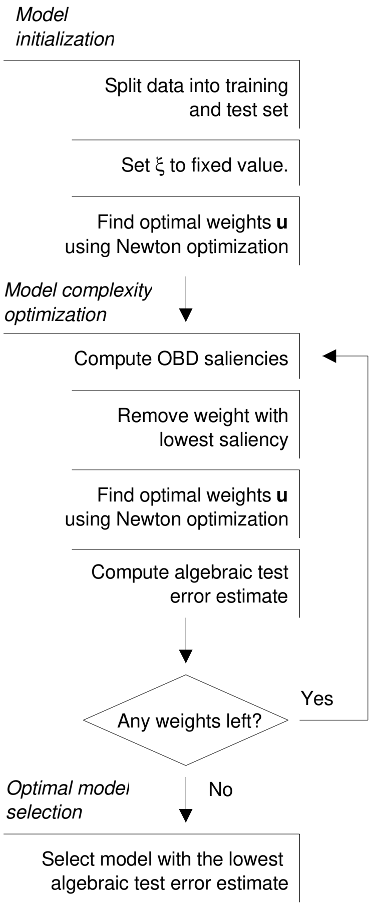

Figure 5.6: Scheme for designing neural classifiers using optimal brain damage pruning. This algorithm is useful when dealing with a limited amount of data where holding out data for a validation set is not feasible.

Chapter 6

Experiments

In this chapter, we will present experiments based on the design algorithms described in section 5.4. In particular, we will illustrate the benefits of the outlier model introduced in section 4.5 using an artificial problem and, finally, present results for neural classifiers designed for solving the malignant melanoma problem.

6.1 Application to artificial Gaussian problem

To illustrate the design algorithm described in section 5.4.1 using adaptive estimation of the regularization parameters and the outlier probability, the algorithm has been applied to an artificial generated classification problem. We will in particular focus on the effects of incorporating the outlier probability in the model framework.

6.1.1 Dataset description

The artificial Gaussian problem has 3 classes in a 2-D input space with equal prior class probabilities, $P(\mathcal{C}_l) = 1/3$. The class-conditional probability distributions are given by

$$p(\mathbf{x}|\mathcal{C}_l) = \frac{\mathcal{N}(\boldsymbol{\mu}_l, \mathbf{I}) + \mathcal{N}(-\boldsymbol{\mu}_l, \mathbf{I})}{2}, \quad l = 1, 2, 3 \, , \tag{6.1}$$

where $\mathcal{N}(\boldsymbol{\mu}_l, \mathbf{I})$ is a 2-D Gaussian distribution with mean vector $\boldsymbol{\mu}$ and identity covariance matrix $\mathbf{I}$. The mean vectors are given by

$$\boldsymbol{\mu}_l = 3 \left[\cos\left(\pi(2l-1)/6\right) \;\; \sin\left(\pi(2l-1)/6\right) \right]^{\mathrm{T}}, \quad l = 1, 2, 3 \, . \tag{6.2}$$

Note, that the class-conditional probability distributions have a significant overlap which is also reflected by the minimal Bayes probability of misclassification being 0.213.

300 patterns were generated and split into a training set of size $q_{\mathcal{D}} = 150$ and a validation set of size $q_{\mathcal{V}} = 150$ respecting the equal class prior property. In addition, a test set consisting of $q_{\mathcal{T}} = 600$ patterns were generated. In all 3 sets, outliers were introduced with probability $\epsilon = 0.08$, i.e., 8% of all patterns had their class label changed randomly.

In figure 6.1, the class-conditional probability distributions are shown.

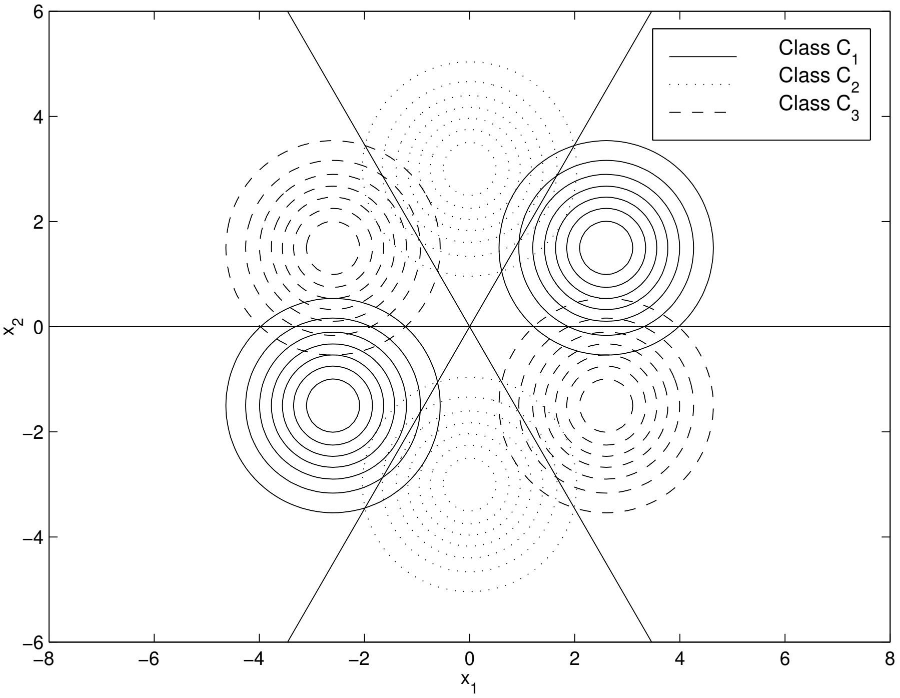

Figure 6.1: The artificial Gaussian problem. This is a 3 class problem where each class is a mixture of two Gaussians. The plot shows contours of constant class-conditional probability in the 2-D input space as well as the optimal Bayes decision boundaries.

6.1.2 Experimental setup

The used network architecture consists of 2 inputs, 5 hidden units and 2 output units[1] with 4 regularization parameters for the following 4 weights groups: input to hidden unit weights, input bias to hidden unit connections, hidden to output unit weights and hidden bias to output unit connections.

The network weights are initialized uniformly over $[-0.5; 0.5]$, the regularization parameters set to zero and the outlier probability, ϵ, set to 0.01. 30 gradient descent iterations for updating the weights are performed followed by a re-initialization of the regularization parameters to $\lambda_{\max}/10^4$, where $\lambda_{\max}$ is the maximum eigenvalue of the Hessian matrix for the cost function. This prevents initial numerical stability problems when inverting the Hessian. The Newton algorithm[2] is then used for finding a minimum of the cost function w.r.t. the weight parameters. Matrix inversion is done using the Moore-Penrose pseudo inverse (see e.g. [Seber and Wild, 1995]) ensuring that the eigenvalue spread[3] is less than 10^8.

Next the regularization parameters and the outlier probability are adaptively esti-

[1] Note, we have 3 classes but only 2 output units with weight connections due to the modified softmax normalization.

[2] Training is stopped when the 2-norm of the gradient of the training error w.r.t. the weights is below 10^{-5} or the maximum number of allowed iterations is reached.

[3] Eigenvalue spread should not be larger than the square root of the machine precision [Dennis and Schnabel, 1983].

Cross-entropy error	**Initial neural classifier**	**Optimal neural classifier**†	**Optimal neural classifier**‡
Training	0.379 ± 0.014	0.466 ± 0.028	0.466 ± 0.025
Validation	0.905 ± 0.082	0.785 ± 0.132	0.657 ± 0.029
Test	0.867 ± 0.033	0.837 ± 0.159	0.684 ± 0.030
Test after retraining	0.762 ± 0.017	0.830 ± 0.108	0.671 ± 0.022

† Evaluated using the standard cross-entropy error.

‡ Evaluated using the outlier-modified cross-entropy error.

Table 6.1: Cross-entropy error for the artificial Gaussian problem with the true outlier probability being $\epsilon = 0.08$. The averages and standard deviations over 10 runs are reported. Initial and optimal neural classifiers refers to using initial and optimized values of $\boldsymbol{\alpha}$ and ϵ, respectively. The outlier probability was estimated to $\hat{\epsilon} = 0.097 \pm 0.018$.

mated[4] as shown in the *model complexity optimization* part of figure 5.5. Finally, the weights are retrained on the combined training and validation set using the optimized regularization parameters and outlier probability.

6.1.3 Results

A total of 10 classifiers are designed as described in the previous section. In table 6.1 the mean and standard variation of the cross-entropy error for the 10 classifiers are listed. The first column shows the performance before the regularization parameters and the outlier parameters are adapted, i.e., the classifiers are trained with the regularization parameters set to initial values that ensures numerical stability and the outlier probability set to zero. The results show the problems with overfitting and with the large contribution to the validation and test error from the outliers. Note, that retraining on the combined training/validation set reduces the test error significantly. This indicates that the overfitting problem is reduced due to the larger training set. The second column in table 6.1 shows the results from the optimal classifiers evaluated using the standard cross-entropy error function. That is, the classifiers are designed using adaptive regularization and outlier probability estimation by using the outlier-modified cross-entropy error function. The final results, though, are reported using the standard cross-entropy error function in order to appreciate the effects the outlier probability model has on the empirical generalization error estimates. The third column shows the results of the exact same optimal classifiers but here the results are reported using the outlier-modified cross-entropy error function. Comparing the two last columns, we see that the contribution of the outliers to the validation and test error are suppressed by using the outlier-modified cross-entropy error function yielding more robust empirical generalization error estimates. Note, that retraining on the combined training/validation set does not yield a significant improvement indicating that overfitting is not a problem, i.e., the estimated regularization parameters have eliminated overfitting.

Table 6.2 shows the corresponding classification errors. Recall, that using the posterior probabilities from a classifier directly or using the outlier-modified posterior probabilities do not change the decision boundaries and the classification results (see section 4.5). Thus

[4] Adaptation is stopped when the gradient of the validation error w.r.t. the regularization parameters and the outlier probability is below 10^{-2} or the maximum number of allowed iterations is reached.

Probability of misclassification	Initial neural classifier	Optimal neural classifier	Optimal Bayes decisions
Training	0.141 ± 0.009	0.145 ± 0.011	0.160
Validation	0.263 ± 0.022	0.241 ± 0.012	0.240
Test	0.273 ± 0.009	0.250 ± 0.009	0.222
Test after retraining	0.268 ± 0.015	0.249 ± 0.012	0.222

Table 6.2: Probability of misclassification for the artificial Gaussian problem with the true outlier probability being $\epsilon = 0.08$. The averages and standard deviations over 10 runs are reported. Initial and optimal neural classifiers refers to using initial and optimized values of $\boldsymbol{\alpha}$ and ϵ. The optimal Bayes decision results are found by using the true optimal decision boundaries when there are no outliers. For an infinite data set without outliers, the minimal Bayes error is 0.213. The outlier probability was estimated to $\hat{\epsilon} = 0.097 \pm 0.018$.

the classification results for the two last columns in table 6.1 are the same and are shown in the second column in table 6.2. The third column indicates the classification results if the true Bayes decision boundaries are used. The minimal Bayes classification error for an infinite data set without outliers is 0.213. Note, that when using the true Bayes decision boundaries, the classification error for the training set is significantly smaller than the classification error on the validation and test set as well as the minimal Bayes classification error. This is probably an effect of the rather small training set.

In figure 6.2, the development of the classifier design is shown for an *optimal neural classifier* before retraining on the combined training/validation set. For this data set, 20 to 40 iterations are typically needed for the algorithm to converge. Note, the large difference between the training error and validation/test error. This is not due to significant overfitting but rather due to the small training set not representing the true data distribution very well. In fact, if we look at the classification results using the true decision boundaries in table 6.2, we actually expect this discrepancy.

Figure 6.3 shows an example of the posterior probabilities for a classifier designed using adaptive regularization and retrained on the combined training/validation set but with the outlier probability fixed to zero, i.e., the outlier model is not used for this example. Comparing the decision boundaries qualitatively with the optimal Bayes decision boundaries in figure 6.1, it seems that the outliers have had a significant adverse effect on the location of the decision boundaries. Figure 6.4, on the other hand, shows the posterior probabilities for a classifier designed using the outlier model. Here we see, that the decision boundaries seem to correspond better with the optimal Bayes decision boundaries. Thus, the outlier model has suppressed the outliers during modeling resulting in a classifier that better describes the true non outlier-infected data set. Note, that the smallest and largest possible posterior probability is $\hat{\epsilon}/(n_C - 1) = 0.049$ and $1 - \hat{\epsilon} = 0.903$, respectively, due to the lower and upper bound imposed by the outlier model (see section 4.5).

6.2 Application to the malignant melanoma problem

Due to the limited amount of available data for the malignant melanoma problem, we can not justify holding out data for a validation set in order to adaptively estimate the regularization parameters and outlier probability.

Instead, we employ the design algorithm described in section 5.4.2 using fixed values

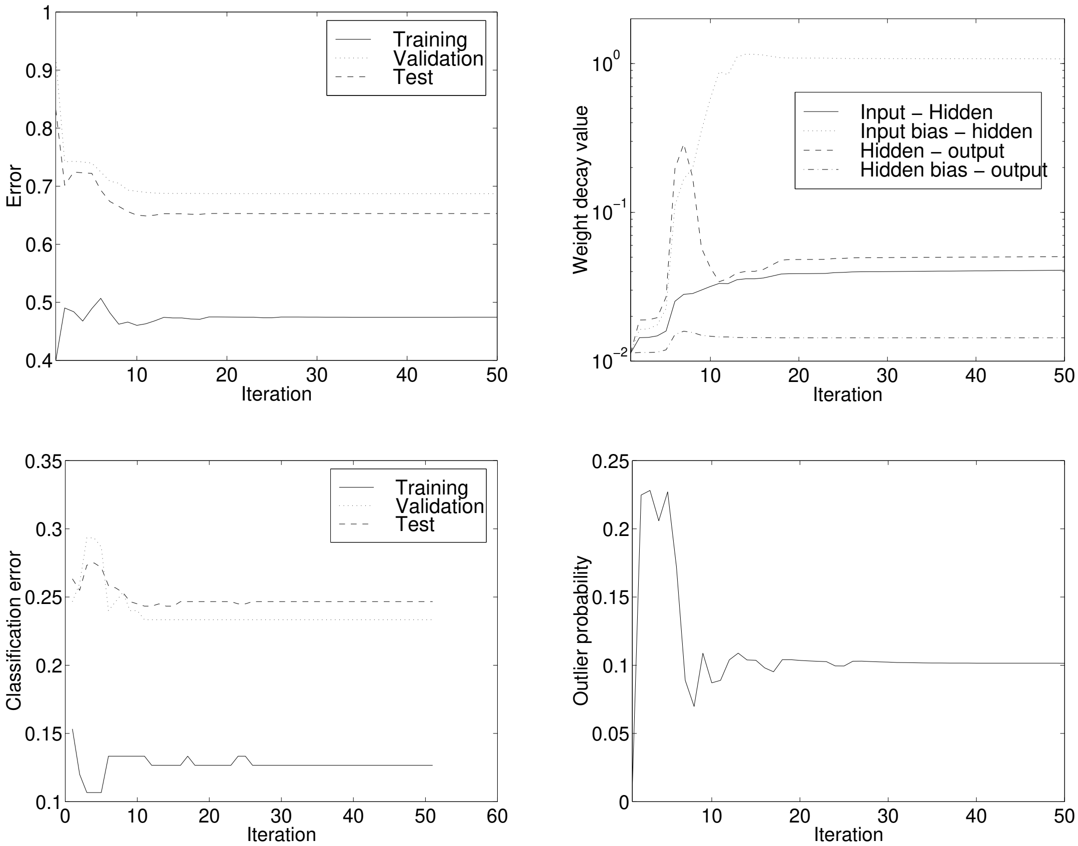

Figure 6.2: Results of a run of the design algorithm for the artificial Gaussian problem. An iteration consists of weight parameter optimization w.r.t. the training set and regularization and outlier probability parameter optimization w.r.t. the validation set. Upper left: The development of the outlier-modified cross-entropy error. Upper right: The development of the weight decay parameters. Lower left: The development of the classification error. Lower right: The development of the estimated outlier probability.

of the regularization parameters and outlier probability combined with network pruning. Of particular interest is the pruning of dermatoscopic input features.

6.2.1 Dataset description

As mentioned in section 3.5, we have a total of 58 dermatoscopic images distributed in 3 skin lesion categories as: *Benign nevi*: 25, *atypical nevi*: 11 and *malignant melanoma*: 22. For each image, 9 features have been extracted. In summary, these are: 2 asymmetry measures, 2 edge abruptness measures and 5 color measures (see chapter 3 for details).

One approach for attempting to overcome the limited data problem, would be to employ bootstrapping methods for increasing the training set size, see e.g. [Young, 1994], [Efron and Tibshirani, 1986], [Efron and Tibshirani, 1993].

We will use the empirical leave-one-out test error estimator described in section 4.3.1 for evaluating the designed classifiers. This gives us 58 training sets each with 57 patterns and 58 test sets with 1 pattern. Thus, in order to design a complete classifier for solving

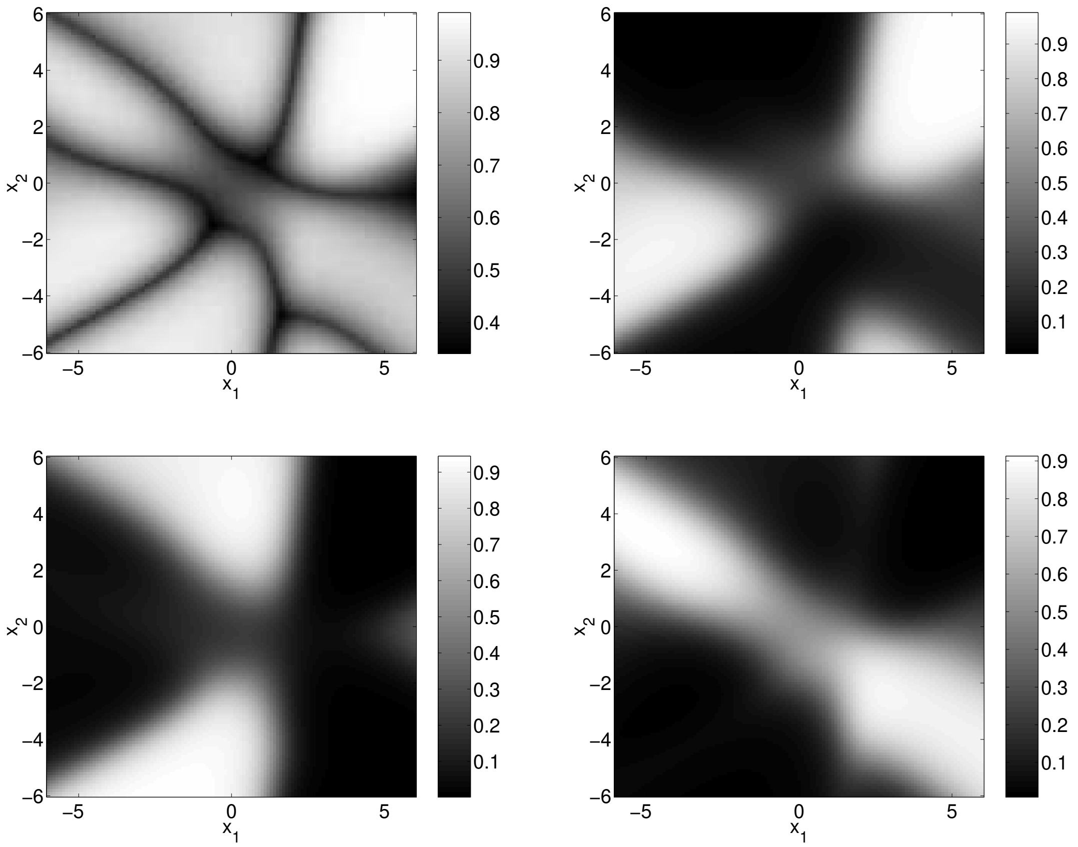

Figure 6.3: Example of plots of the posterior probabilities for a classifier retrained on the combined Gaussian outlier-infected training/validation set without using the outlier probability framework, i.e., the outlier probability, ϵ, is fixed to zero during modeling. Upper left: The distribution of the maximum of the posterior probability for the 3 classes. The dark lines corresponds to decision boundaries. Upper right: The posterior probability distribution for class $\mathcal{C}_1$. Lower left: The posterior probability distribution for class $\mathcal{C}_2$. Lower right: The posterior probability distribution for class $\mathcal{C}_3$.

the malignant melanoma problem, we need to design 58 classifiers for the 58 training sets.

6.2.2 Experimental setup

The used network architecture consists of 9 inputs, 4 hidden units and 2 output units with 2 regularization parameters, α_{w^I} and α_{w^H}, for the weights/biases in the input layer and the weights/biases in the output layer, respectively.

The network weights are initialized uniformly over $[-0.5; 0.5]$ and the regularization parameters are set to $\alpha_{w^I} = 0.5$ and $\alpha_{w^H} = 0.9$. These are chosen in order to prevent significant overfitting of the training data. A more systematic approach for determining the regularization parameters without the use of a validation set is to sample the algebraic test error estimate as a function of the regularization parameters and use the regularization parameters that minimize the algebraic test error estimate. Examples of this are shown in the paper in appendix G. The outlier probability ϵ is set to zero. Thus, we do not use the outlier model for this application since we do not have enough data for estimating

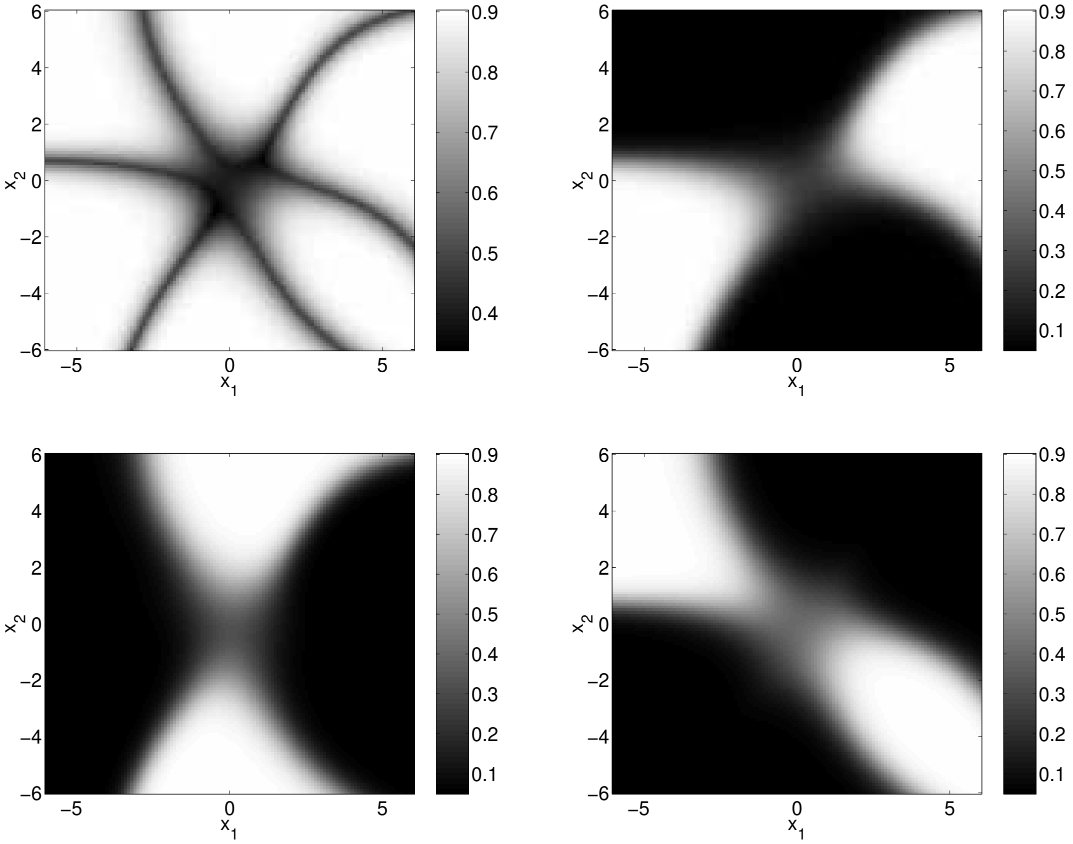

Figure 6.4: Example of plots of the posterior probabilities for a classifier retrained on the combined Gaussian outlier-infected training/validation set when using the outlier probability framework. The outlier probability was in this case estimated to $\hat{\epsilon} = 0.097$. Upper left: The distribution of the maximum of the posterior probability for the 3 classes. The dark lines corresponds to decision boundaries. Upper right: The posterior probability distribution for class $\mathcal{C}_1$. Lower left: The posterior probability distribution for class $\mathcal{C}_2$. Lower right: The posterior probability distribution for class $\mathcal{C}_3$.

the outlier probability. 30 gradient descent iterations are performed prior to using the Newton algorithm[5] for locating a cost function minimum. Matrix inversion is done using the Moore-Penrose pseudo inverse (see e.g. [Seber and Wild, 1995]) ensuring that the eigenvalue spread[6] is less than 10^8. This is not a problem for this application due to the rather large regularization parameters.

Next, the network is pruned as shown in the *model complexity optimization* part of figure 5.6 and the optimal pruned model is selected as the model with the lowest algebraic test error estimate. Recall, that this is an asymptotic estimate. Thus, its use may be questionable in an application with only 57 training patterns. During pruning, the *training patterns per weight* relationship improves, thus hopefully improving the validity of the

[5] Training is stopped when the 2-norm of the gradient of the training error w.r.t. the weights is below 10^{-5} or the maximum number of allowed iterations is reached.

[6] Eigenvalue spread should not be larger than the square root of the machine precision [Dennis and Schnabel, 1983].

Cross-entropy error	Non-pruned neural classifier	Pruned neural classifier
Training	0.689 ± 0.002	0.757 ± 0.003
Test	1.022 ± 0.016	1.007 ± 0.006

Table 6.3: Cross-entropy error for the malignant melanoma problem. The averages and standard deviations over 10 runs are reported. One run is a full leave-one-out scheme using 58 training sets.

Probability of misclassification	Non-pruned neural classifier	Pruned neural classifier
Training	0.273 ± 0.004	0.306 ± 0.001
Test	0.441 ± 0.023	0.400 ± 0.007

Table 6.4: Probability of misclassification for the malignant melanoma problem. The averages and standard deviations over 10 runs are reported.

estimator.

Since we employ the leave-one-out empirical test error estimator for model evaluation, the full classifier consists of 58 pruned networks.

6.2.3 Results

A total of 10 classifiers each consisting of 58 pruned networks as described in the previous section are designed. All results reported are the averages and standard deviations for the 10 classifiers.

6.2.3.1 Classifier results

Table 6.3 lists the cross-entropy error rates for the training and test set before and after pruning. As expected, the training error increases as a result of pruning due to the reduced network complexity while the test error decreases only slightly. Recall, that we do not use the outlier model, thus improvement in generalization ability may be disguised by a few patterns dominating the cross-entropy error as illustrated by the example in figure 4.1.

The corresponding classification[7] results are shown in table 6.4. Here we see a more noticeable decrease of the test error from 0.441 ± 0.023 to 0.400 ± 0.007 after pruning. Note, that there is still some discrepancy between the training error and test error suggesting that we are still overfitting the training set somewhat.

While the cross-entropy error and the classification error yield some insight into the performance of a classifier, it is of great interest to see how the classification errors are distributed in the 3 classes. This information is contained in the *confusion matrices* for the training and test set.

In table 6.5 and 6.6, the confusion matrices for the training set before and after pruning are shown. We see that the performance for the *atypical nevi* class is rather poor before

[7]Following Bayes minimum-error decision rule as described in section 4.1, the network output with the highest probability determines the class. One could also adopt Bayes minimum-risk decision rule as described in section 4.1.1.

Confusion matrix for training set	**Non-pruned neural classifier**		
	Benign nevi	**Atypical nevi**	**Melanoma**
Benign nevi†	0.921 ± 0.002	0.474 ± 0.015	0.208 ± 0.004
Atypical nevi†	0.003 ± 0.002	0.163 ± 0.014	0.004 ± 0.001
Melanoma†	0.076 ± 0.001	0.363 ± 0.001	0.788 ± 0.003

† indicates the estimated output classes.

Table 6.5: Confusion matrix for the training set using non-pruned networks. The averages and standard deviations over 10 runs are reported.

Confusion matrix for training set	**Pruned neural classifier**		
	Benign nevi	**Atypical nevi**	**Melanoma**
Benign nevi†	0.922 ± 0.002	0.632 ± 0.004	0.210 ± 0.003
Atypical nevi†	0.000 ± 0.000	0.012 ± 0.004	0.014 ± 0.003
Melanoma†	0.078 ± 0.002	0.356 ± 0.003	0.776 ± 0.001

† indicates the estimated output classes.

Table 6.6: Confusion matrix for the training set using pruned networks. The averages and standard deviations over 10 runs are reported.

pruning and even worse after pruning. This indicates that the flexibility of the classifier is rather low, thus reducing the risk of overfitting. This is mainly due to the rather large regularization parameters. With smaller regularization parameters, we can easily obtain a zero classification error rate for the training set resulting in significant overfitting of the training set. The reason, that the *atypical nevi* class suffers, is the lower class prior[8] compared to the *benign nevi* and *melanoma* class. Thus, the error contribution from the *atypical nevi* class is relatively small making it fairly inexpensive to ignore this class during training. A method for minimizing the risk of completely ignoring a class is to weight each error contribution from a pattern in the cross-entropy error function with the inverse class prior. This corresponds to creating equal class priors. In order to take the real imbalanced priors into account, the network outputs should be reweighted with the real imbalanced class priors divided by the balanced class priors (see, e.g., [Bishop, 1995]). This approach has not been employed in this study. It is interesting to note that the majority of the *atypical nevi* before and after pruning are assigned to the *benign nevi* class when recalling that the *atypical nevi* are in fact healthy.

Table 6.7 and 6.8 show the confusion matrices for the test set before and after pruning. Again, we note that the *atypical nevi* class seems to be ignored. In fact, for the pruned classifiers none of the *atypical nevi* in the test set are classified correctly. $72.7\% \pm 0.0\%$ are actually classified as benign. This suggests that the information in the extracted dermatoscopic features is not adequate for distinguishing the *benign nevi* from the *atypical nevi* but is more appropriate for separating healthy lesions, i.e. *benign nevi* and *atypical nevi*, from cancerous lesions. Acknowledging this, we might be able to obtain a higher detection of the *melanoma* lesions by considering only these two categories of lesions when designing the classifiers. This has not been attempted, though. If we compare the test set results before and after pruning, we note that pruning has improved the detection

[8]Recall, only 11 of 58 lesions in the training set are atypical.

Confusion matrix for test set	Non-pruned neural classifier		
	Benign nevi	**Atypical nevi**	**Melanoma**
Benign nevi†	0.684 ± 0.058	0.709 ± 0.038	0.273 ± 0.000
Atypical nevi†	0.108 ± 0.033	0.018 ± 0.038	0.041 ± 0.014
Melanoma†	0.208 ± 0.041	0.273 ± 0.000	0.686 ± 0.014

† indicates the estimated output classes.

Table 6.7: Confusion matrix for the test set using non-pruned networks. The averages and standard deviations over 10 runs are reported.

Confusion matrix for test set	Pruned neural classifier		
	Benign nevi	**Atypical nevi**	**Melanoma**
Benign nevi†	0.732 ± 0.019	0.727 ± 0.000	0.241 ± 0.037
Atypical nevi†	0.032 ± 0.017	0.000 ± 0.000	0.009 ± 0.019
Melanoma†	0.236 ± 0.013	0.273 ± 0.000	0.750 ± 0.024

† indicates the estimated output classes.

Table 6.8: Confusion matrix for the test set using pruned networks. The averages and standard deviations over 10 runs are reported.

of the *benign nevi* and the *melanoma* lesions significantly. In fact, a detection rate of $75.0\% \pm 2.4\%$ for the *melanoma* lesions are comparable with the detection rates of very experienced dermatologists [Koh et al., 1989].

In figure 6.5, the results of a typical run of the design algorithm is shown. For the non-pruned networks, the cross-entropy test error and classification test error exhibit only very little overfitting. Notice, how the Newton optimization sets in after 30 iterations. If smaller regularization parameters were used, the effects would have been a lot more dramatic. The pruning plots show that the decrease of the cross-entropy test error and classification test error occurs at the end of the pruning session, i.e., when only 12 to 20 weights remain. Note, that the minimum of the algebraic test error estimate coincides fairly well with the region where the test error is lowest.

For comparison a standard *k-nearest-neighbor*[9] (k-NN) classification was performed. The training error may be computed from the training set by including each training pattern in the majority vote. The *leave-one-out* test error is computed by excluding each training pattern from the vote. Figure 6.6 shows the classification error on the training and test set as a function of k. We see that for a wide range of k-values, the k-NN classifier has similar classification error rates on the test set compared with the non-pruned and pruned neural classifiers suggesting that the k-NN classifier and the neural classifiers perform similarly. If we inspect the confusion matrix for the test set for a 15-NN classifier shown in table 6.9, we see that they classify quite differently despite having approximately the same overall classification error rate. The 15-NN classifier performs much better for the *benign nevi* class at the expense of the *melanoma* class. This is very unfortunate since the cancerous lesions are our major concern. From a medical point of view, it is significantly more expensive classifying a cancerous lesion as healthy as is the opposite case. Again,

[9] Within a k-NN, a pattern is classified according to a majority vote among its k nearest neighbors using the Euclidean metric, see, e.g., [Duda and Hart, 1973].

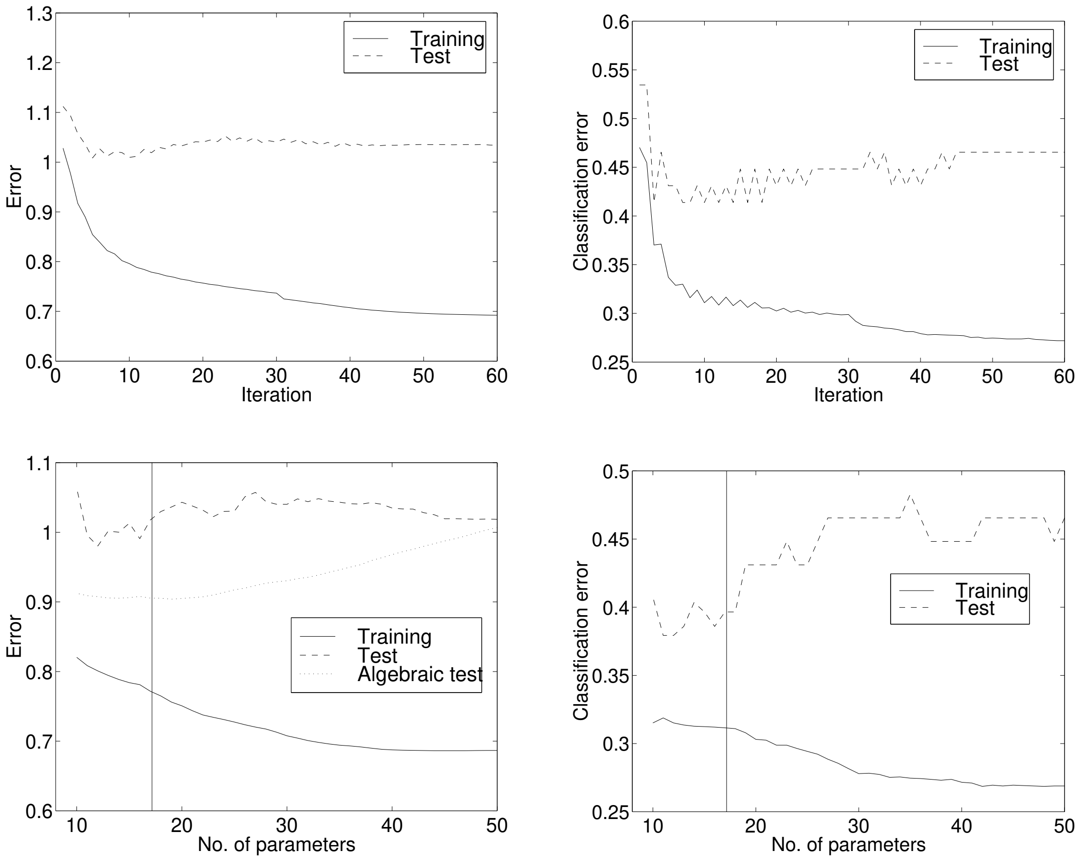

Figure 6.5: Results of a run of the design algorithm for the malignant melanoma problem. Each run consists of 58 networks. Upper left: The development of the cross-entropy error during training of the non-pruned networks. Gradient descent is used for the first 30 iterations, thereafter Newton optimization is used. Upper right: The development of the classification error during training of the non-pruned networks. Lower left: The development of the cross-entropy error during pruning. The vertical line indicates the mean location of the minimum of the estimated test error. Lower right: The development of the classification error during pruning.

we note that a large majority of the *atypical nevi* are classified as *benign nevi* supporting our earlier statement concerning the discriminating power of the extracted dermatoscopic features.

6.2.3.2 Dermatoscopic feature importance

One of the most interesting effects of pruning is that it may provide information about the importance of the input variables. This is of particular interest for this application where the discriminating power of the dermatoscopic features is still rather unclear. Figure 6.7 shows an example of a pruned network selected by the minimum of the algebraic test error estimate. Two inputs have been completely removed by the pruning process. For this particular network, it is the *minor axis asymmetry* measure and the *dark-brown color* measure. These are in fact the two most commonly removed dermatoscopic input features

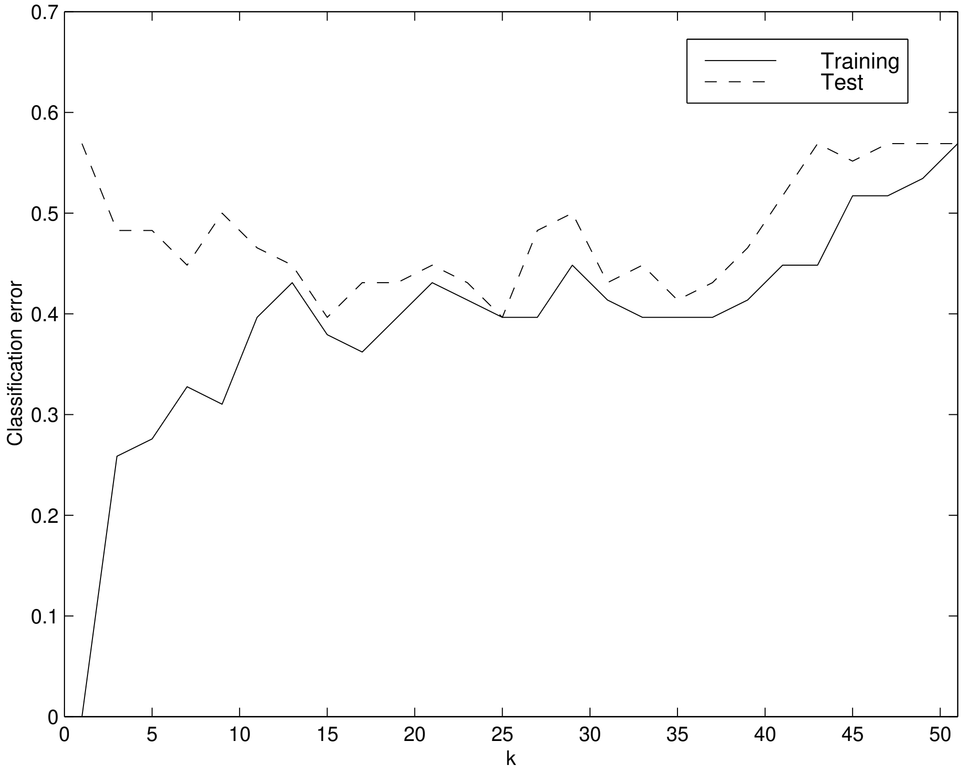

Figure 6.6: Classification results for a k-NN classifier as a function of k. Note, that for a wide range of k-values, the k-NN classifier performs similar to the non-pruned and pruned neural classifiers when comparing the classification rates.

Confusion matrix for test set	**k-NN classifier ($k = 15$)**		
	Benign nevi	**Atypical nevi**	**Melanoma**
Benign nevi†	0.920	0.818	0.455
Atypical nevi†	0.000	0.000	0.000
Melanoma†	0.080	0.182	0.545

† indicates the estimated output classes.

Table 6.9: Confusion matrix for the test set using a 15-NN classifier. Note, that the classifier favors the *benign nevi* class, thus making costly errors in the *melanoma* class from a medical point of view.

as can be seen in table 6.10. The table shows how often the individual dermatoscopic features have been completely removed during the runs of the design algorithm. Recall, that each run results in 58 pruned networks. Thus, for each run the number of times a feature has been removed is computed relative to the maximum number of times it could have been removed (58). This enables us to compute the mean and standard deviation over 10 runs and sort the features according to their importance[10]. Two features were

[10] Assuming that the number of times a feature has been removed is inversely proportional to its importance.

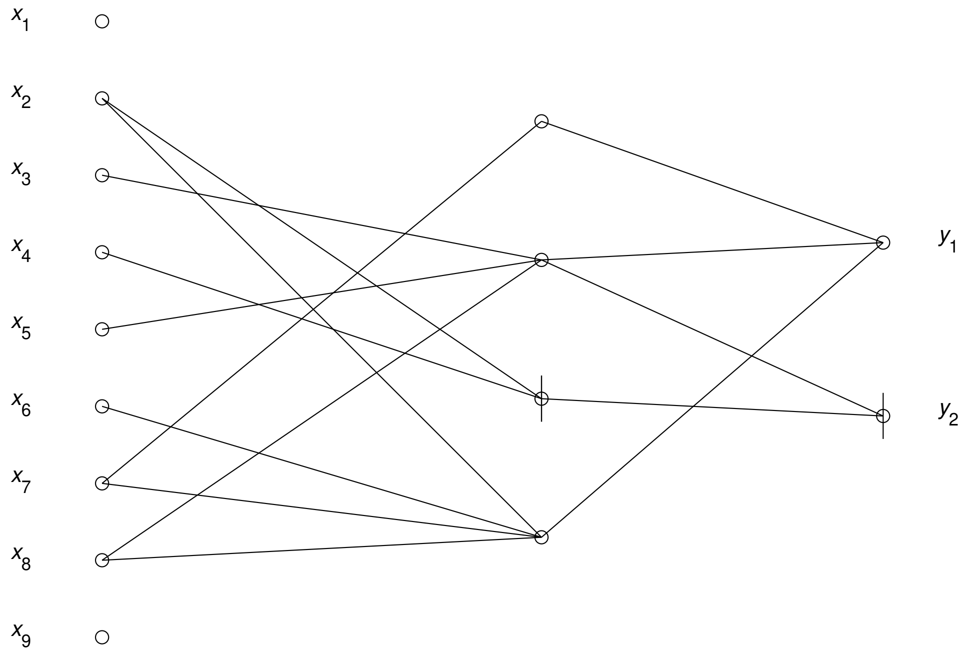

Figure 6.7: Example of a pruned malignant melanoma network with 17 weights. A vertical line through a node indicates a bias. The two pruned dermatoscopic input features are the *minor axis asymmetry* measure and the *dark-brown color* measure. These are the two most commonly pruned input features. Recall, that we only have two network outputs with weight connections due to the modified softmax normalization.

never pruned: The *major axis asymmetry* measure and the *blue color* measure. We know that the presence of blue color in a lesion indicates *blue-white veil* and thus malignancy. So this is an expected result. We would also expect asymmetry to be important since this indicates different local growth rates in the lesion and thus malignancy. It is interesting to note that while the *major axis asymmetry* measure seems very important, the *minor axis asymmetry* measure is nearly always removed. The reason for this is probably that these two measures often are very similar which is also indicated by the skin lesion examples in figure 3.6 and 3.7. That is, they both contain the same information, thus only one asymmetry measure is needed. The *dark-brown color* measure is the most often pruned feature. This is a bit surprising since the number of different colors present in a skin lesion normally is considered to correlate with the degree of malignancy. The removal of this feature could be due to the fact that the 5 color measures sum to 1 for a skin lesion. Thus, it is possible to infer a missing color measure from the remaining 4. We also note that the *white color* measure is often removed. This could invalidate the explanation of the inference of a missing color measure but as can be seen from the distribution of the *white color* measure in figure 3.12, the amount of white color, if present, is typically under 0.5%. That is, the *white color* measure could easily be ignored in the inference of the missing *dark-brown color* measure.

In summary, the 3 most important dermatoscopic features seem to be the *major axis*

Feature importance	**Pruning index**	**Feature importance**	**Pruning index**	**Feature importance**	**Pruning index**
Asymmetry: Major axis	0.000 $\pm$ 0.000	**Edge abrupt.: Std. dev.**	0.053 $\pm$ 0.025	**Color: White**	0.272 $\pm$ 0.031
Color: Blue	0.000 $\pm$ 0.000	**Edge abrupt.: Mean**	0.083 $\pm$ 0.021	**Asymmetry: Minor axis**	0.772 $\pm$ 0.048
Color: Black	0.022 $\pm$ 0.008	**Color: Light-brown**	0.097 $\pm$ 0.023	**Color: Dark-brown**	0.783 $\pm$ 0.054

Table 6.10: Table showing how often the individual dermatoscopic features have been completely pruned during the 10 runs. A zero pruning index for a feature indicates that it was never removed while a pruning index of 1 indicates that the feature was always removed. The averages and standard deviations over 10 runs are reported.

asymmetry measure and the *blue and black color* measures while the 3 least important are the *dark-brown and white color* measures and the *minor axis asymmetry* measure.

Chapter 7

Conclusion

In this thesis, we have proposed a probabilistic framework for classification based on neural networks and we have applied the framework to the problem of classifying skin lesions.

This involved extracting relevant information from dermatoscopic images, defining a probabilistic framework robust against outliers, proposing methods for optimizing neural networks capable of estimating posterior class probabilities and applying the methods to the malignant melanoma classification problem.

Dermatoscopic feature extraction

The extraction of dermatoscopic features involved measuring the skin lesion asymmetry, the transition of pigmentation from the skin lesion to the surrounding skin and the color distribution within the skin lesion. The latter involved determining color prototypes by inspecting 2-D color histograms and by using knowledge of dermatologists color perception. No reliable red prototype color could be identified, though, partially due to a strong reddish glow of the dark-brown color in skin lesions. It was seen that some of the extracted dermatoscopic features singlehandedly showed potential for separating in particular the malignant lesions from the healthy lesions.

Probabilistic framework for classification

The defined probabilistic framework for classification included optimal decision rules, derivation of error functions, model complexity control, assessment of generalization performance and an outlier model. We noted that the derived cross-entropy error function is very sensitive to outliers due to the nature of the logarithmic function, thus inspiring the development of a novel outlier model. Incorporating the outlier model in the probabilistic framework, results in a modified cross-entropy error function effectively downweighting error contributions from outliers. This renders model optimization schemes more robust and yields more accurate empirical generalization error estimates.

Neural classifier modeling

The proposed schemes for designing neural network classifiers involved defining a two-layer feed-forward network architecture, methods for optimizing network weights and methods for estimating regularization parameters and the outlier probability. Traditionally, a standard softmax output normalization scheme is employed in order to ensure that model outputs may be interpreted as posterior probabilities. This normalization scheme has an inherent redundancy due to the property that the posterior probability output estimates sum to one. This redundancy is generally ignored and results in weight dependencies in the output layer and, thus, a singular unregularized Hessian matrix. In order to overcome this, a modified softmax output normalization scheme removing the redundancy has been suggested. A novel method for adaptively estimating regularization parameters and the outlier probability has been derived. It is based on employing a validation set and acknowledging that the network weights are implicitly dependent on the regularization parameters and the outlier probability. Using an artificially generated classification problem plagued with outliers, we were able to approximately determine the true outlier probability. We also saw that the inclusion of the outlier model reduced the influence from outliers on the empirical generalization error estimates and on the decision boundaries.

The malignant melanoma classification problem

The neural classifier framework was applied to the malignant melanoma classification problem using the extracted dermatoscopic features and results from histological analyzes of skin tissue samples. The adaptive estimation of regularization parameters and outlier probability was not employed due to the very limited amount of data available. Instead, optimal brain damage pruning and model selection using an algebraic generalization error estimate was employed. In a leave-one-out test set, we were able to detect $73.2\% \pm 1.9\%$ of benign lesions and $75.0\% \pm 2.4\%$ of malignant lesions. None of the atypical lesions were classified correct. We argued that this probably is due to the fact that the atypical lesion class has a small prior and thus is ignored during model estimation. $72.7\% \pm 0.0\%$ of the atypical lesions were classified as benign lesions. Recalling, that atypical lesions are in fact healthy indicates that the extracted dermatoscopic features are effective only for separating healthy lesions from cancerous lesions, i.e., the features do not possess adequate information for discriminating between benign and atypical lesions. As a result of the pruning process, it was possible to rank the dermatoscopic features according to their importance. We found that the three most important features are shape asymmetry and the amount of blue and black color present within a skin lesion.

Suggestions for further work

The extracted dermatoscopic features only represent a few of the features defined in table 2.4. A natural progression would be to extract some or all of the remaining features which would hopefully provide us with additional discriminating power. Collecting more skin lesion data should also have a high priority. This would enable us to estimate the regularization parameters and the outlier probability adaptively and inspect the labeled data for outliers or very unusual skin lesions.

More experimental work investigating and validating the beneficial effects of the outlier model is recommended. Of special interest is how the regularization parameters and the outlier probability interact during modeling.

Appendix A

Computation of the gradient and Hessian of the training error

In this appendix, the detailed derivation of the first and second derivatives of the outlier-modified cross-entropy error function w.r.t. the networks weights for a softmax or modified softmax normalized classifier is shown (see section 5.3.1). We will express the derivatives as functions of the first and second derivatives of the traditional neural network with linear outputs.

Note, if the modified softmax normalization is used then $\phi_{n_\mathcal{C}}(\mathbf{x}^\mu) = 0$ in the following.

A.1 Gradient

Recall, that the outlier-modified cross-entropy error is

$$E_\mathcal{D}(\mathbf{u}) = -\frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} y_i^\mu \log\left[\hat{z}_i^\mu (1 - \beta n_\mathcal{C}) + \beta\right], \tag{A.1}$$

where we will let $\hat{y}_i^\mu$ denote the outlier-modified posterior probability $\hat{z}_i^\mu(1 - \beta n_\mathcal{C}) + \beta$. The first derivative of equation (A.1) w.r.t. a weight or threshold parameter, u_j, is

$$\frac{\partial E_\mathcal{D}(\mathbf{u})}{\partial u_j} = -\frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} \frac{y_i^\mu}{\hat{y}_i^\mu}(1 - \beta n_\mathcal{C}) \frac{\partial \hat{z}_i^\mu}{\partial u_j}, \tag{A.2}$$

where

$$\frac{\partial \hat{z}_i^\mu}{\partial u_j} = \sum_{i'=1}^{n_\mathcal{C}} \frac{\partial \hat{z}_i^\mu}{\partial \phi_{i'}(\mathbf{x}^\mu)} \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \tag{A.3}$$

$$= \sum_{i'=1}^{n_\mathcal{C}} \frac{\delta_{i,i'} \exp[\phi_i(\mathbf{x}^\mu)] \sum_{i''=1}^{n_\mathcal{C}} \exp[\phi_{i''}(\mathbf{x}^\mu)] - \exp[\phi_i(\mathbf{x}^\mu)] \exp[\phi_{i'}(\mathbf{x}^\mu)]}{\left(\sum_{i''=1}^{n_\mathcal{C}} \exp[\phi_{i''}(\mathbf{x}^\mu)]\right)^2} \cdot \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \tag{A.4}$$

$$= \sum_{i'=1}^{n_\mathcal{C}} \delta_{i,i'} \hat{z}_i^\mu - \hat{z}_i^\mu \hat{z}_{i'}^\mu \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \tag{A.5}$$

$$= \hat{z}_i^\mu \sum_{i'=1}^{n_\mathcal{C}} (\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}. \tag{A.6}$$

With zero outlier probability, the gradient reduces to

$$\frac{\partial E_\mathcal{D}(\mathbf{u})}{\partial u_j} = -\frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} \frac{y_i^\mu}{\hat{z}_i^\mu} \frac{\partial \hat{z}_i^\mu}{\partial u_j}. \tag{A.7}$$

Note, that only when i equals the correct class for pattern $\mathbf{x}^\mu$, is the contribution to the gradient from the inner sum in equation (A.2) and equation (A.7) non-zero.

A.2 Hessian

The second derivative of equation (A.2) is given by

$$\frac{\partial^2 E_\mathcal{D}(\mathbf{u})}{\partial u_j \partial u_k} = -\frac{(1-\beta n_\mathcal{C})}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} \left[-\frac{y_i^\mu}{(\hat{y}_i^\mu)^2} \frac{\partial \hat{y}_i^\mu}{\partial u_k} \frac{\partial \hat{z}_i}{\partial u_j} + \frac{y_i^\mu}{\hat{y}_i^\mu} \frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} \right] \tag{A.8}$$

$$= -\frac{(1-\beta n_\mathcal{C})}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} \frac{y_i^\mu}{\hat{y}_i^\mu} \left[-\frac{(1-\beta n_\mathcal{C})}{\hat{y}_i^\mu} \frac{\partial \hat{z}_i^\mu}{\partial u_k} \frac{\partial \hat{z}_i^\mu}{\partial u_j} + \frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} \right], \tag{A.9}$$

where $\partial \hat{z}_i^\mu / \partial u_j$ is given by equation (A.6) and $\partial^2 \hat{z}_i^\mu / \partial u_j \partial u_k$ by

$$\frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} = \sum_{i'=1}^{n_\mathcal{C}} \left[\left((\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial \hat{z}_i^\mu}{\partial u_k} - \hat{z}_i^\mu \frac{\partial \hat{z}_{i'}^\mu}{\partial u_k} \right) \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} + \hat{z}_i^\mu (\delta_{i,i'} - \hat{z}_{i'}^\mu) \frac{\partial^2 \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j \partial u_k} \right]. \tag{A.10}$$

With zero outlier probability, the Hessian reduces to

$$\frac{\partial^2 E_\mathcal{D}(\mathbf{u})}{\partial u_j \partial u_k} = -\frac{1}{q_\mathcal{D}} \sum_{\mu=1}^{q_\mathcal{D}} \sum_{i=1}^{n_\mathcal{C}} \frac{y_i^\mu}{\hat{z}_i^\mu} \left[-\frac{1}{\hat{z}_i^\mu} \frac{\partial \hat{z}_i^\mu}{\partial u_k} \frac{\partial \hat{z}_i^\mu}{\partial u_j} + \frac{\partial^2 \hat{z}_i^\mu}{\partial u_j \partial u_k} \right]. \tag{A.11}$$

Note, that only when i equals the correct class for pattern $\mathbf{x}^\mu$, is the contribution to the Hessian from the inner sum in equation (A.9) and equation (A.11) non-zero.

A.2.1 Gauss-Newton approximation of the Hessian

The Gauss-Newton approximation of the Hessian is obtained by using Fisher's property [Seber and Wild, 1995],

$$\left\langle \frac{\partial^2 e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathbf{y}|\mathbf{x})} = \left\langle \frac{\partial e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u}} \frac{\partial e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathbf{y}|\mathbf{x})}, \tag{A.12}$$

where $e(\mathbf{x},\mathbf{y},\mathbf{u}) = -\log p(\mathbf{y}|\mathbf{x},\mathbf{u}) = -\sum_{i=1}^{n_{\mathcal{C}}} y_i \log\left[\hat{z}_i(1-\beta n_{\mathcal{C}})+\beta\right]$ (see section 4.2 and 4.5.1).

Noting, that we have already computed $\partial e(\mathbf{x},\mathbf{y},\mathbf{u})/\partial \mathbf{u}$ as part of the gradient in equation (A.2), we have

$$\begin{aligned}
\left\langle \frac{\partial^2 e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathbf{y}|\mathbf{x})} &= \left\langle \frac{\partial e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u}} \frac{\partial e(\mathbf{x},\mathbf{y},\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathbf{y}|\mathbf{x})} && \text{(A.13)}\\
&= \left\langle \sum_{i=1}^{n_{\mathcal{C}}} \sum_{i'=1}^{n_{\mathcal{C}}} \frac{y_i}{\hat{y}_i} \frac{y_{i'}}{\hat{y}_{i'}} (1-\beta n_{\mathcal{C}})^2 \frac{\partial \hat{z}_i}{\partial \mathbf{u}} \frac{\partial \hat{z}_{i'}}{\partial \mathbf{u}^{\mathrm{T}}} \right\rangle_{p(\mathbf{y}|\mathbf{x})} && \text{(A.14)}\\
&= (1-\beta n_{\mathcal{C}})^2 \sum_{i=1}^{n_{\mathcal{C}}} \sum_{i'=1}^{n_{\mathcal{C}}} \frac{\langle y_i y_{i'} \rangle_{p(\mathbf{y}|\mathbf{x})}}{\hat{y}_i \hat{y}_{i'}} \frac{\partial \hat{z}_i}{\partial \mathbf{u}} \frac{\partial \hat{z}_{i'}}{\partial \mathbf{u}^{\mathrm{T}}} && \text{(A.15)}\\
&= (1-\beta n_{\mathcal{C}})^2 \sum_{i=1}^{n_{\mathcal{C}}} \frac{p(y_i|\mathbf{x})}{\hat{y}_i^2} \frac{\partial \hat{z}_i}{\partial \mathbf{u}} \frac{\partial \hat{z}_i}{\partial \mathbf{u}^{\mathrm{T}}}, && \text{(A.16)}
\end{aligned}$$

where $\partial \hat{z}_i/\partial \mathbf{u}$ is given by equation (A.6) and where we have used

$$\begin{aligned}
\langle y_i y_{i'} \rangle_{p(\mathbf{y}|\mathbf{x})} &= \delta_{i,i'} \left\langle (y_i)^2 \right\rangle_{p(\mathbf{y}|\mathbf{x})} && \text{(A.17)}\\
&= \delta_{i,i'} \langle y_i \rangle_{p(\mathbf{y}|\mathbf{x})} && \text{(A.18)}\\
&= \delta_{i,i'} p(y_i|\mathbf{x}). && \text{(A.19)}\\
& && \text{(A.20)}
\end{aligned}$$

Using the training set for estimating the average in equation (A.16) and using $\hat{y}_i$ as estimate of $p(y_i|\mathbf{x})$, we arrive at the following approximation of the Hessian,

$$\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u})}{\partial u_j \partial u_k} = \frac{(1-\beta n_{\mathcal{C}})^2}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{1}{\hat{y}_i^{\mu}} \frac{\partial \hat{z}_i^{\mu}}{\partial u_k} \frac{\partial \hat{z}_i^{\mu}}{\partial u_j}. \tag{A.21}$$

With zero outlier probability, the Gauss-Newton approximation of the Hessian reduces to

$$\frac{\partial^2 E_{\mathcal{D}}(\mathbf{u})}{\partial u_j \partial u_k} = \frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{1}{\hat{z}_i^{\mu}} \frac{\partial \hat{z}_i^{\mu}}{\partial u_k} \frac{\partial \hat{z}_i^{\mu}}{\partial u_j}. \tag{A.22}$$

Appendix B

Computation of the gradient of the validation error

This appendix shows the detailed derivation of the gradient of the validation error w.r.t. the regularization parameters and the scaled outlier probability (see section 5.3.2).

B.1 Gradient w.r.t. the regularization parameters

We will derive the gradient w.r.t. the regularization parameters along the lines described in [Larsen et al., 1998b].

Let us consider the case where the regularization function is linear in the regularization parameters, i.e.,

$$R(\mathbf{u}) = \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{r}(\mathbf{u}) = \sum_{i=1}^{n_\alpha} \alpha_i r_i(\mathbf{u}), \tag{B.1}$$

where n_α is the number of regularization parameters and $r_i(\mathbf{u})$ some function of the weights and thresholds and let us denote the total number of weights and thresholds by n_U.

We are interested in computing the gradient of the validation error, $\hat{G}_{\mathcal{V}}$, w.r.t. the regularization parameters, $\boldsymbol{\alpha}$. Using the chain rule, we may write the gradient as

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}(\boldsymbol{\alpha}))}{\partial \boldsymbol{\alpha}} = \frac{\partial \mathbf{u}(\boldsymbol{\alpha})^{\mathrm{T}}}{\partial \boldsymbol{\alpha}} \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}(\boldsymbol{\alpha}))}{\partial \mathbf{u}}, \tag{B.2}$$

where $\partial \mathbf{u}(\boldsymbol{\alpha})^{\mathrm{T}}/\partial \boldsymbol{\alpha}$ is a $n_\alpha \times n_U$ derivative matrix and where we acknowledge that $\mathbf{u}$ is an implicit function of $\boldsymbol{\alpha}$. In order to determine the derivative matrix, let us consider a first order Taylor expansion of the cost function around the current estimate $\boldsymbol{\alpha}^{(t)}$ at time step t,

$$C(\boldsymbol{\alpha}^{(t+1)}) \approx C(\boldsymbol{\alpha}^{(t)}) + \frac{\partial C(\boldsymbol{\alpha}^{(t)})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}}(\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^{(t)}). \tag{B.3}$$

The derivative of the expansion w.r.t. the weight parameters is given by

$$\frac{\partial C(\boldsymbol{\alpha}^{(t+1)})}{\partial \mathbf{u}} \approx \frac{\partial C(\boldsymbol{\alpha}^{(t)})}{\partial \mathbf{u}} + \frac{\partial^2 C(\boldsymbol{\alpha}^{(t)})}{\partial \mathbf{u} \partial \boldsymbol{\alpha}^{\mathrm{T}}}(\boldsymbol{\alpha}^{(t+1)} - \boldsymbol{\alpha}^{(t)}). \tag{B.4}$$

Assuming that we at time step t are at a minimum of the cost function w.r.t. the weights and requiring that we at time step $t+1$ also are at a minimum, i.e., the two first derivative expressions in equation (B.4) are zero, we have

$$\frac{\partial^2 C(\mathbf{u}(\boldsymbol{\alpha}^{(t)}), \boldsymbol{\alpha}^{(t)})}{\partial \mathbf{u} \partial \boldsymbol{\alpha}^{\mathrm{T}}} = \mathbf{0}, \tag{B.5}$$

where we acknowledge that the cost function is explicitly dependent on $\boldsymbol{\alpha}$ and implicitly dependent on $\boldsymbol{\alpha}$ through the weights.

Now, let us expand the left hand side of equation (B.5) into a sum of a term implicitly and a term explicitly dependent on $\boldsymbol{\alpha}$ omitting the time step dependency for notational convenience,

$$\begin{aligned}
\frac{\partial^2 C(\mathbf{u}(\boldsymbol{\alpha}), \boldsymbol{\alpha})}{\partial \mathbf{u} \partial \boldsymbol{\alpha}^{\mathrm{T}}} &= \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}(\boldsymbol{\alpha}), \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} \right) & \text{(B.6)} \\
&= \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} + \frac{\partial C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} \right) & \text{(B.7)} \\
&= \frac{\partial^2 C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} + \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} \right) & \text{(B.8)} \\
&= \frac{\partial^2 C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}^{\mathrm{T}}} + \frac{\partial \mathbf{r}(\mathbf{u})^{\mathrm{T}}}{\partial \mathbf{u}}, & \text{(B.9)}
\end{aligned}$$

where we have used the generalized chain rule in equation (B.7) and that only the regularization function contributes to the second expression on the right hand side of equation (B.8). $\partial \mathbf{r}(\mathbf{u})^{\mathrm{T}} / \partial \mathbf{u}$ is a $n_U \times n_\alpha$ matrix. Setting equation (B.9) to zero, yields

$$\frac{\partial \mathbf{u}(\boldsymbol{\alpha})^{\mathrm{T}}}{\partial \boldsymbol{\alpha}} = -\frac{\partial \mathbf{r}(\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \left(\frac{\partial^2 C(\mathbf{u}, \boldsymbol{\alpha})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right)^{-1}. \tag{B.10}$$

Inserting this into equation (B.2) and omitting the $\boldsymbol{\alpha}$ dependency, yields the desired expression for the gradient of the validation error w.r.t. the regularization parameters,

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \boldsymbol{\alpha}} = -\frac{\partial \mathbf{r}(\mathbf{u})}{\partial \mathbf{u}^{\mathrm{T}}} \left(\frac{\partial^2 C(\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right)^{-1} \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \mathbf{u}}. \tag{B.11}$$

B.2 Gradient w.r.t. the scaled outlier probability

The gradient of the validation error w.r.t. the scaled outlier probability, β, may be written as a sum of a term explicitly dependent on β and a term implicitly dependent on β through the weights,

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}(\beta), \beta)}{\partial \beta} = \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}, \beta)}{\partial \beta} + \frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u}, \beta)}{\partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\beta)}{\partial \beta}, \tag{B.12}$$

where we have used the chain rule for the implicit term. The explicit term is easily found to be

$$\frac{\partial \hat{G}_{\mathcal{V}}(\beta)}{\partial \beta} = -\frac{1}{q_{\mathcal{V}}} \sum_{\mu=1}^{q_{\mathcal{V}}} \sum_{i=1}^{n_{\mathcal{C}}} y_i^\mu \frac{1 - n_{\mathcal{C}} \hat{z}_i^\mu}{\hat{y}_i^\mu}, \tag{B.13}$$

where $q_{\mathcal{V}}$ is the number of patterns in the validation set and $n_{\mathcal{C}}$ the number of classes.

In order to find $\partial \mathbf{u}(\beta)/\partial\beta$ from the implicit term in equation (B.12), we compute the gradient w.r.t. the weights of a first order Taylor expansion of the cost function around the current estimate of β similar to what we did for the regularization parameters in section B.1. Again, requiring that we are at a minimum of the cost function w.r.t. the weights for the current estimate of β and the next estimate of β implies

$$\frac{\partial^2 C(\mathbf{u}(\beta), \beta)}{\partial \mathbf{u} \partial \beta} = \mathbf{0}. \tag{B.14}$$

Expanding the left hand side into a sum of a term implicitly dependent on β through the weights and a term explicitly dependent on β, yields

$$\begin{aligned}
\frac{\partial^2 C(\mathbf{u}(\beta), \beta)}{\partial \mathbf{u} \partial \beta} &= \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}(\beta), \beta)}{\partial \beta} \right) && \text{(B.15)} \\
&= \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}, \beta)}{\partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\beta)}{\partial \beta} + \frac{\partial C(\mathbf{u}, \beta)}{\partial \beta} \right) && \text{(B.16)} \\
&= \frac{\partial^2 C(\mathbf{u}, \beta)}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \frac{\partial \mathbf{u}(\beta)}{\partial \beta} + \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}, \beta)}{\partial \beta} \right), && \text{(B.17)}
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial C(\mathbf{u}, \beta)}{\partial \beta} \right) &= \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial E_{\mathcal{D}}(\mathbf{u}, \beta)}{\partial \beta} \right) && \text{(B.18)} \\
&= \frac{\partial}{\partial \mathbf{u}} \left(-\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} y_i^\mu \frac{1 - n_{\mathcal{C}} \hat{z}_i^\mu}{\hat{y}_i^\mu} \right) && \text{(B.19)} \\
&= -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} y_i^\mu \left(\frac{-n_{\mathcal{C}} \frac{\partial \hat{z}_i^\mu}{\partial \mathbf{u}} \hat{y}_i^\mu - (1 - n_{\mathcal{C}} \hat{z}_i^\mu) \frac{\partial \hat{y}_i^\mu}{\partial \mathbf{u}}}{(\hat{y}_i^\mu)^2} \right) && \text{(B.20)} \\
&= -\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} y_i^\mu \left(\frac{-n_{\mathcal{C}} \frac{\partial \hat{z}_i^\mu}{\partial \mathbf{u}} \hat{y}_i^\mu - (1 - n_{\mathcal{C}} \hat{y}_i^\mu) \frac{\partial \hat{z}_i^\mu}{\partial \mathbf{u}}}{(\hat{y}_i^\mu)^2} \right) && \text{(B.21)} \\
&= \frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{y_i^\mu}{(\hat{y}_i^\mu)^2} \frac{\partial \hat{z}_i^\mu}{\partial \mathbf{u}}. && \text{(B.22)}
\end{aligned}$$

Setting equation (B.17) equal to zero and solving for $\partial \mathbf{u}(\beta)/\partial\beta$, yields

$$\frac{\partial \mathbf{u}(\beta)}{\partial \beta} = -\left(\frac{\partial^2 C(\mathbf{u}, \beta)}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right)^{-1} \left(\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{y_i^\mu}{(\hat{y}_i^\mu)^2} \frac{\partial \hat{z}_i^\mu}{\partial \mathbf{u}} \right). \tag{B.23}$$

Inserting this result and equation (B.13) into equation (B.12) and omitting the β dependency gives us the desired expression for the gradient of the validation error w.r.t. the scaled outlier probability,

$$\frac{\partial \hat{G}_{\mathcal{V}}(\mathbf{u})}{\partial \beta} = -\frac{1}{q_{\mathcal{V}}} \sum_{\mu=1}^{q_{\mathcal{V}}} \sum_{i=1}^{n_{\mathcal{C}}} y_i^{\mu} \frac{1 - n_{\mathcal{C}} \hat{z}_i^{\mu}}{\hat{y}_i^{\mu}} - \left(\frac{\partial^2 C(\mathbf{u})}{\partial \mathbf{u} \partial \mathbf{u}^{\mathrm{T}}} \right)^{-1} \left(\frac{1}{q_{\mathcal{D}}} \sum_{\mu=1}^{q_{\mathcal{D}}} \sum_{i=1}^{n_{\mathcal{C}}} \frac{y_i^{\mu}}{(\hat{y}_i^{\mu})^2} \frac{\partial \hat{z}_i^{\mu}}{\partial \mathbf{u}} \right). \tag{B.24}$$

Appendix C

Contribution to EANN'96

This appendix contains the paper "Detection of Malignant Melanoma using Neural Classifiers" presented at the 1996 International Conference on Engineering Applications using Neural Networks (EANN'96) in London, England [Hintz-Madsen et al., 1996a].

Detection of Malignant Melanoma using Neural Classifiers

Mads Hintz-Madsen, Lars Kai Hansen, and Jan Larsen
Dept. of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark
email: hintz, lkhansen, jlarsen@ei.dtu.dk

Eric Olesen, and Krzysztof T. Drzewiecki
Dept. of Reconstructive Surgery S, The National Hospital of Denmark, DK-2100 Copenhagen, Denmark

Abstract

In this paper we propose a method for design of feed-forward neural classifiers based on regularization and adaptive architectures, and we apply the scheme to the problem of detecting malignant melanoma. Using features acquired from color photographs describing color and texture properties of skin tumors, we are able to detect 76.0±7.8% of melanoma cases in a test set.

1 Introduction

The incidence of malignant melanoma, the most lethal of skin cancers, has risen rapidly during the last 50 years. Fortunately patients can be saved from this life-threatening cancer, if it is detected at an early stage. Thus, in recent years, there has been an increased interest in schemes for automatic and early detection of melanoma. Digital imaging may assist and improve the possibility of such early detections. A recent review of digital imaging in this field was recently published in "Skin Research and Technology" [1]. In this paper we apply a method for design of feed-forward neural classifiers based on regularization and adaptive architectures to the problem of detecting melanoma. Statistical measurements describing color and texture properties are extracted from color photographs of skin tumors and used for classifying tumors.

Key features of the design approach is efficient regularized Gauss-Newton training [2], SoftMax [3] normalization of outputs enabling a probabilistic interpretation of the outputs, architecture optimization and evaluation with Optimal Brain Damage (OBD) [4], and an asymptotic algebraic test error estimate [5].

2 Neural Classifier

Let us assume that we have a training set, D, consisting of q input-output pairs

$$D = \{(\mathbf{x}^{\mu}, y^{\mu}) | \mu = 1, ..., q\}, \tag{1}$$

where $\mathbf{x}$ is an input vector consisting of n_I elements and y is the corresponding class label. In this presentation we will assume that the class label is of the definite form $y = 1, ..., n_O$, with n_O being the number of classes. An alternative soft target assignment might be relevant in some practical contexts where the target could be, e.g., an estimate of class probabilities for the given input.

We aim to model the conditional probability distribution

$$p(i|\mathbf{x}), \qquad i = 1, \ldots, n_O. \tag{2}$$

To represent these distributions we choose the following network architecture:

$$h_j(\mathbf{x}^\mu) = \tanh\left(\sum_{k=0}^{n_I} w_{jk} x_k^\mu\right), \qquad \phi_i(\mathbf{x}^\mu) = \sum_{j=0}^{n_H} W_{ij} h_j(\mathbf{x}^\mu), \tag{3}$$

with n_I input units, n_H hidden units, n_O output units, and parameters $\mathbf{u} = (\mathbf{w}, \mathbf{W})$, where w_{j0} and W_{i0} are thresholds. To ensure that the outputs, $\phi_i(\mathbf{x}^\mu)$, can be interpreted as probabilities, we use the normalized exponential transformation known as SoftMax [3]:

$$\hat{p}(i|\mathbf{x}^\mu) \equiv \frac{\exp(\phi_i(\mathbf{x}^\mu))}{\sum_{i'=1}^{n_O} \exp(\phi_{i'}(\mathbf{x}^\mu))}, \tag{4}$$

where $\hat{p}(i|\mathbf{x}^\mu)$ is the estimated probability, that $\mathbf{x}^\mu$ belongs to class i.

The likelihood of the parameters, $\mathbf{u}$, can now be expressed as

$$P(D|\mathbf{u}) = \prod_{\mu=1}^{q} \prod_{i=1}^{n_O} \hat{p}(i|\mathbf{x}^\mu)^{\delta_{i,y^\mu}}, \tag{5}$$

where δ_{i,y^μ} is the Kronecker delta.

Training is based on the minimization of the negative log-likelihood

$$E(\mathbf{u}) = -\frac{1}{q} \log P(D|\mathbf{u}) = -\frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\sum_{i'=1}^{n_O} \exp(\phi_{i'}(\mathbf{x}^\mu))\right)\right]. \tag{6}$$

In order to eliminate overfitting and ensure numerical stability, we augment the cost function by a regularization term, e.g. a simple weight decay,

$$C(\mathbf{u}) = E(\mathbf{u}) + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}, \tag{7}$$

where $\mathbf{R}$ is a positive definite diagonal matrix with elements $2\frac{\alpha_j}{q}$.

Using matrix/vector notation the Gauss-Newton paradigm of updating the weights can now be computed as [2]

$$\mathbf{u}^{\mathrm{new}} = \mathbf{u} - \eta\,(\mathbf{H} + \mathbf{R})^{-1}\left[\frac{\partial E}{\partial \mathbf{u}} + \mathbf{R}\mathbf{u}\right], \tag{8}$$

where $\mathbf{Ru}$ and $\mathbf{R}$ are the first and second derivatives of the regularization term respectively, $\mathbf{H}$ is the Hessian matrix containing the second derivatives of (6) w.r.t. the weights and η is a parameter, that may be used to ensure a decrease in the cost function, e.g., by line search.

In order to reduce and optimize a networks architecture, we recommend to apply a pruning scheme such as *Optimal Brain Damage* (OBD) [4]. The aim of OBD is to estimate the importance of the weights for the training error and rank the weights according to their importance. If the importance is estimated using a second order expansion of the training error around its minimum, the *saliency* for a weight u_i is [6]

$$s_i = \left(\mathbf{R}_{ii} + \frac{1}{2}\mathbf{H}_{ii}\right) u_i^2, \tag{9}$$

where the Hessian $\mathbf{H}_{ii}$ and $\mathbf{R}_{ii}$ is the i'th diagonal element of $\mathbf{H}$ and $\mathbf{R}$ respectively.

By repeatedly removing weights with the smallest saliencies and retraining the resulting network, a nested family of networks is obtained. To select the network with the best generalization ability,

	Fully connected ANN			**Pruned ANN**		
Conf. mat.	Benign nevi	Dyspl. nevi	Melanoma	Benign nevi	Dyspl. nevi	Melanoma
Benign nevi†	42.4±9.1%	29.0±6.2%	22.0±10.1%	47.6±9.3%	27.5±7.6%	18.5±8.8%
Dyspl. nevi†	24.3±5.2%	49.5±6.9%	5.5±3.7%	22.4±6.0%	53.5±5.8%	5.5±4.4%
Melanoma†	33.3±6.7%	21.5±6.7%	72.5±7.6%	30.0±8.4%	19.0±3.9%	76.0±7.8%

Table 1: Confusion matrix for the test set using fully connected (103 weights) and pruned networks (41-62 weights). Note that the classifier performs best for the critical melanoma class. † indicates the ANN output classes.

we can use an estimate of the test error. Using statistical assumptions, see e.g. [7] and [5], the following estimate of the test error of a network $\mathbf{u}$ estimated on a training set D can be derived [5],

$$\widehat{\langle E_{\text{test}} \rangle} = E_{\text{train}}(\mathbf{u}(D)) + \frac{N_{\text{eff}}}{q}, \tag{10}$$

where $E_{\text{train}}(\mathbf{u}(D))$ is the training error of the model. The effective number of weights is given by $N_{\text{eff}} = \text{Tr}[\mathbf{H}(\mathbf{H}+\mathbf{R})^{-1}]$, where $\mathbf{R}$ is the second derivative of the regularization term in (7).

3 Results

From a collection of color photographs of skin tumors at The National Hospital of Denmark 21 statistical measurements describing color and texture properties have been acquired for each tumor and are used for classification into three groups: Benign nevi (non-cancer), dysplastic nevi (non-cancer, but increased risk of developing cancer) and melanoma. A biopsy has been performed on all skin tumors used in this study, thus giving us the "correct" diagnosis for each tumor. This dataset is considered to be difficult as all tumors were labelled "suspect" by doctors prior to the biopsy, and thus excludes obvious cases of non-cancer tumors.

A total of 180 images with an even split of the three classes were used for training and 60 images were used for testing. Ten feed-forward networks with 21 inputs, 4 hidden units and 3 output units were trained[1] and subsequently pruned; hence resulting in ten nested families of pruned networks. The estimate of the test error for each family was used to select the network with the lowest estimated generalization error. Two weight decays were used: $\alpha_w = 0.1$ and $\alpha_W = 1$ for the weights in the input and output layer respectively. In figure 1 a typical pruning scenario is shown. Table 1 shows the confusion matrix for the test set classified[2] with the fully connected networks and the selected "optimal" networks. The mean and standard deviation of the ten runs are reported. Overall the fully connected networks classified 89.6±1.6% of the training set and 54.6±4.1% of the test set correctly, while the results for the pruned networks are 86.7±3.6% for the training set and 58.9±2.1% for the test set. Thus the pruning has increased the generalization ability of the networks. An important effect of the pruning approach is the selection of input features, that are salient for the classification; thus providing us with information that can be used in clinical dermatology. Of the ten pruned networks, four didn't use input 15 and three didn't use input 16. This is an indication that these two color features (blue color mean and red color variance) are unimportant for the classification task.

For comparison a standard *k-Nearest-Neighbor*[3] (k-N-N) classification was performed. The training error may be computed from the training set by including each training pattern in the majority vote. A *leave-one-out* "validation" error on the training set may be computed by excluding each

[1] Training was stopped when the 2-norm of the gradient vector was below 10^{-5}.

[2] Following Bayes decision theory, the network output with the highest probability determines the class label.

[3] Within k-N-N a pattern is classified according to a majority vote among its k nearest neighbors using the simple Euclidean metric.

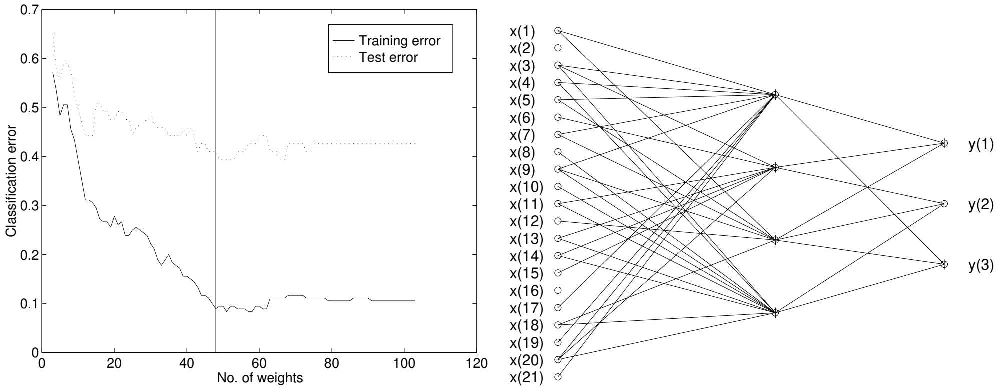

Figure 1: *Left*: Example of a pruning session. The vertical line indicates the "optimal" network selected by the estimated test error. *Right*: Example of a pruned network (from 103 to 48 weights). Note that two inputs are not used.

training pattern from the vote. Finally, the test patterns may be classified by voting among the k nearest neighbors found among the training patterns. Using the validation error we found that $k = 6$ was optimal for this data set. The 6-N-N scheme classified 74.7% of the training set and 57.3% of the test set correctly. Thus the performance of the optimized K-N-N scheme fall in between the pruned and fully connected networks.

4 Conclusion

We have applied a method for design of feed-forward neural classifiers to the problem of detecting melanoma. The approach was shown to be able to design compact neural network models with a generalization ability better than non-pruned network as well as giving us an indication of which input features are unimportant for the classification task; thus enabling us to reevaluate the acquired statistical measurements.

References

[1] W.V. Stoecker, R.H. Moss, F. Ercal, and S.E. Umbaugh. Nondermatoscopic digital imaging of pigmented lesion. *Skin Research and Technology* **1**, pages 7–16, 1995.

[2] C.M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, UK, 1995.

[3] J.S. Bridle. *Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition*, volume 68 of *Neurocomputing - Algorithms, Architectures and Applications*, pages 227–236. Springer-Verlag, Berlin, 1990.

[4] Y. Le Cun, J. Denker, and S. Solla. Optimal Brain Damage. *Advances in Neural Information Processing Systems*, 2:598–605, 1990.

[5] S. Amari and N. Murata. Statistical Theory of Learning Curves under Entropic Loss Criterion. *Neural Computation*, 5:140–153, 1993.

[6] C. Svarer, L.K. Hansen, and J. Larsen. On Design and Evaluation of Tapped-Delay Neural Network Architectures. *The 1993 IEEE International Conference on Neural Networks*, pages 46–51, 1993.

[7] M. Hintz-Madsen, L.K. Hansen, J.Larsen, E. Olesen, and K.T. Drzewiecki. Design and evaluation of neural classifiers - application to skin lesion classification. *The 1995 IEEE Workshop on Neural Networks for Signal Processing (NNSP'95)*, pages 484–493, 1995.

Appendix D

Contribution to NNSP'96

This appendix contains the paper "Design and Evaluation of Neural Classifiers" presented at the 1996 IEEE Workshop on Neural Networks for Signal Processing (NNSP'96) in Kyoto, Japan [Hintz-Madsen et al., 1996b].

DESIGN AND EVALUATION OF NEURAL CLASSIFIERS

Mads Hintz-Madsen, Morten With Pedersen,
Lars Kai Hansen, and Jan Larsen
CONNECT, Dept. of Mathematical Modeling, B. 349
Technical University of Denmark
DK-2800 Lyngby, Denmark

Phone: (+45) 4525 3885, Fax: (+45) 4588 0117
Email: hintz, with, lkhansen, jlarsen@ei.dtu.dk

Abstract - In this paper we propose a method for design of feed-forward neural classifiers based on regularization and adaptive architectures. Using a penalized maximum likelihood scheme we derive a modified form of the entropic error measure and an algebraic estimate of the test error. In conjunction with Optimal Brain Damage pruning the test error estimate is used to optimize the network architecture. The scheme is evaluated on an artificial and a real world problem.

INTRODUCTION

Pattern recognition is an important aspect of most scientific fields and indeed the objective of most neural network applications. Some of the by now classic applications of neural networks like Sejnowski and Rosenbergs "NetTalk" concern classification of patterns into a finite number of categories. In modern approaches to pattern recognition the objective is to produce class probabilities for a given pattern. Using Bayes decision theory, the "hard" classifier selects the class with the highest class probability, hence minimizing the probability of error. The conventional approach to pattern recognition is statistical and concerns the modeling of class probability distributions for patterns produced by a stationary stochastic source by a certain set of basis functions, e.g., Parzen windows or Gaussian mixtures.

In this paper we define and analyze a system for design and evaluation of feed-forward neural classifiers based on regularization and adaptive architectures. The proposed scheme is a generalization of the approach we have suggested for time series processing [1, 2] and for binary classification in the context of a medical application [3]. The key concept of the new methodology for optimization of neural classifiers is an asymptotic estimate of the test error of the classifier providing an algebraic expression in terms of the training error and a complexity estimate. Our approach is a penalized maximum likelihood scheme. The likelihood is formulated using a simple stationary noisy channel model of the pattern source. For any fixed input pattern there can be defined a fixed probability distribution over a fixed finite set of classes. The training set involves simple labelled data, i.e., for each input vector we are provided with a single class label. The task of the network is to estimate the relative frequencies of class labels for a given pattern. In conjunction with SoftMax normalization of the outputs of a standard, computationally universal, feed-forward network we recover a slightly modified form of the so-called entropic error measure [4]. For a fixed architecture the neural network weights are estimated using a Gauss-Newton method [5], while the model architecture is optimized using Optimal Brain Damage [6]. The problem of proper selection of regularization parameters is also briefly discussed, see also [7].

The salient features of the approach are: Efficient Newton optimization, pruning by Optimal Brain Damage and evaluation of network architectures by an algebraic test error estimate.

NEURAL CLASSIFIERS

Let us assume that we have a training set, D, consisting of q input-output pairs

$$D = \{(\mathbf{x}^\mu, y^\mu) | \mu = 1, ..., q\} \tag{1}$$

where $\mathbf{x}$ is an input vector consisting of n_I elements and y is the corresponding class label. In this presentation we will assume that the class label is of the definite form $y = 1, ..., n_O$, with n_O being the number of classes. An alternative soft target assignment might be relevant in some practical contexts where the target could be, e.g., an estimate of class probabilities for the given input.

We aim to model the posterior probability distribution

$$p(y = i|\mathbf{x}), \qquad i = 1, \ldots, n_O. \tag{2}$$

In some applications it might be desirable to use a rejection threshold when classifying, that is if all of the posterior probabilities fall below this threshold then no classification decision is made, see e.g., [3].

To represent these distributions we choose the following network architecture:

$$h_j(\mathbf{x}^\mu) \quad = \quad \tanh\left(\sum_{k=1}^{n_I} w_{jk} x_k^\mu + w_{j0}\right) \tag{3}$$

$$\phi_i(\mathbf{x}^\mu) \quad = \quad \sum_{j=1}^{n_H} W_{ij} h_j(\mathbf{x}^\mu) + W_{i0} \tag{4}$$

with n_I input units, n_H hidden units, n_O output units, and parameters $\mathbf{u} = (\mathbf{w}, \mathbf{W})$, where w_{j0} and W_{i0} are thresholds. To ensure that the outputs, $\phi_i(\mathbf{x}^\mu)$, can be interpreted as probabilities, we use the normalized exponential transformation known as SoftMax [4]:

$$\hat{p}(y^\mu = i|\mathbf{x}^\mu) \equiv \frac{\exp(\phi_i(\mathbf{x}^\mu))}{\sum_{i'=1}^{n_O} \exp(\phi_{i'}(\mathbf{x}^\mu))} \tag{5}$$

where $\hat{p}(y^\mu = i|\mathbf{x}^\mu)$ is the estimated probability, that $\mathbf{x}^\mu$ belongs to class i.

Assuming that the training data are drawn independently, the likelihood of the model can be expressed as

$$P(D|\mathbf{u}) = \prod_{\mu=1}^{q} \prod_{i=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu)^{\delta_{i,y^\mu}} \tag{6}$$

where $\delta_{i,y^\mu} = 1$ if $i = y^\mu$, otherwise $\delta_{i,y^\mu} = 0$.

Training is based on the minimization of the negative log-likelihood

$$E(\mathbf{u}) = -\frac{1}{q} \log P(D|\mathbf{u}) = \frac{1}{q} \sum_{\mu=1}^{q} \epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) \tag{7}$$

where

$$\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) = -\sum_{i=1}^{n_O} \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\sum_{i'=1}^{n_O} \exp(\phi_{i'}(\mathbf{x}^\mu)) \right) \right]. \tag{8}$$

In order to eliminate overfitting and ensure numerical stability, we augment the cost function by a regularization term, e.g., a simple weight decay,

$$C(\mathbf{u}) = E(\mathbf{u}) + \frac{1}{2}\mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u} \tag{9}$$

where $\mathbf{R}$ is a positive definite matrix. In this paper we consider a diagonal matrix with elements $2\alpha_j/q$.

The gradient of (7) is

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right] \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j}. \tag{10}$$

The matrix of second derivatives (the Hessian) can be expressed as

$$H_{jk} \equiv \frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k} = \frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \sum_{i'=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu) \left[\delta_{i,i'} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right] \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_k} \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \tag{11}$$

where we have used a Gauss-Newton like approximation.

Using matrix/vector notation the Gauss-Newton paradigm of updating the weights can now be computed as [5]

$$\mathbf{u}^{\text{new}} = \mathbf{u} - \eta \left(\mathbf{H} + \mathbf{R}\right)^{-1} \left[\frac{\partial E}{\partial \mathbf{u}} + \mathbf{R}\mathbf{u}\right] \tag{12}$$

where $\mathbf{Ru}$ and $\mathbf{R}$ are the first and second derivatives of the regularization term, respectively, and η is a parameter, that may be used to ensure a decrease in the cost function, e.g., by line search.

A natural approach for determining the regularization parameters is by minimizing the test error with respect to the regularization parameters. Here one may use an estimate of the test error as derived in the next section. Let us now consider the case with only two different weight decays: α_w for the input-to-hidden weights and α_W for the hidden-to-output weights. By sampling the space spanned by α_w and α_W with e.g., a 3x3 grid and computing the estimated test error, it is possible to fit e.g., a paraboloid[1] in a least-square sense to the sample points, locate the minimum[2] of the paraboloid and use the weight decays found for the design of the network.

A different approach using a validation set for determining the regularization parameters is described in [7].

Test Error Estimate

One of the main objectives in our approach is to estimate a network model with a high generalization ability. In order to obtain this we need an estimate of the generalization ability of a model. The generalization or test error for a given network $\mathbf{u}$ may be defined as

$$E_{\text{test}}(\mathbf{u}) = \int d\mathbf{x} dy P(\mathbf{x}, y) \epsilon(\mathbf{x}, y, \mathbf{u}) \tag{13}$$

where $P(\mathbf{x}, y)$ is the true underlying distribution of examples and $\epsilon(\mathbf{x}, y, \mathbf{u})$ is the error on example $(\mathbf{x}, y)$. Since the test error involves an average over all possible examples, it is in general not accessible, but it can be estimated by using additional statistical assumptions, see e.g., [3] and [8], thus giving us the following estimate for the average test error of a network $\mathbf{u}$ estimated on a training set D [8],

$$\langle \widehat{E_{\text{test}}} \rangle = E_{\text{train}}(\mathbf{u}(D)) + \frac{N_{\text{eff}}}{q} \tag{14}$$

[1] Paraboloid: $(z - z_0) = (x - x_0)^2/a^2 + (y - y_0)^2/b^2$.

[2] In case the minimum is located outside the sample-grid, one should relocate the grid and find a new minimum.

where $E_{\text{train}}(\mathbf{u}(D))$ is the training error of the model. The effective number of parameters is given by $N_{\text{eff}} = \text{Tr}[\mathbf{H}(\mathbf{H}+\mathbf{R})^{-1}]$, where $\mathbf{R}$ is the second derivative of the regularization term. This estimate of the test error averaged over all possible training sets may be used to select the optimal network e.g., among a nested family of pruned networks; hence, be used as a pruning stop criterion similarly to our procedure for evaluation of function approximation networks [1, 2].

Pruning with Optimal Brain Damage

In order to reduce and optimize a networks architecture, we recommend to apply a pruning scheme such as *Optimal Brain Damage* (OBD) [6]. The aim of OBD is to estimate the importance of the weights for the training error and rank the weights according to their importance. If the importance is estimated using a second order expansion of the training error around its minimum, the *saliency* for a weight u_i is [1]

$$s_i = \left(R_{ii} + \frac{1}{2} H_{ii} \right) u_i^2 \tag{15}$$

where the Hessian H_{ii} is given by (11) and R_{ii} is i'th diagonal element of $\mathbf{R}$.

By repeatedly removing weights with the smallest saliencies and retraining the resulting network, a nested family of networks is obtained. Here we may use the previously derived test error estimate to select the "optimal" network.

EXPERIMENTS

The proposed methodology for designing neural classifiers has been evaluated on the artificial *contiguity problem* and the real world *glass classification problem*. The latter is a part of the Proben1 neural network benchmark collection [9].

The Contiguity Problem

The contiguity problem has in several cases been used for evaluating optimization schemes, see e.g., [10]. The boolean input vector (± 1) is interpreted as a one-dimensional image and connected clumps of $+1$'s are counted. Two classes are defined: those with two and three clumps. We consider the case, where $n_I = 10$. In this case there are 792 legal input patterns consisting of 432 patterns with three clumps and 360 with two clumps. We use a randomly selected training set with 150 patterns and a test set with 510 patterns both containing an even split of the two classes.

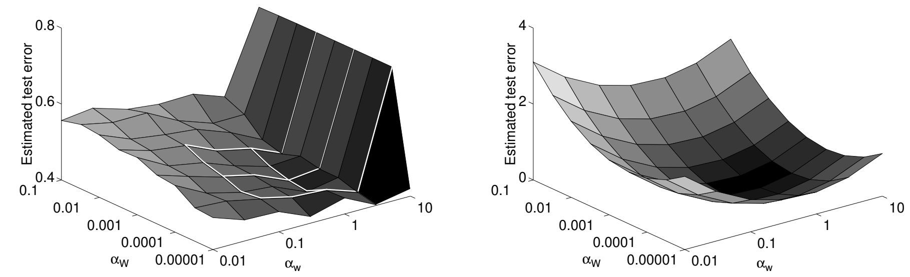

Figure 1: Left: The estimated test error for the *contiguity problem* as function of the weight decay parameters. Right: Paraboloid fitted to the 3x3 grid shown in the left panel. Minimum located at $(\alpha_w, \alpha_W) = (0.68, 2.8 \cdot 10^{-4})$.

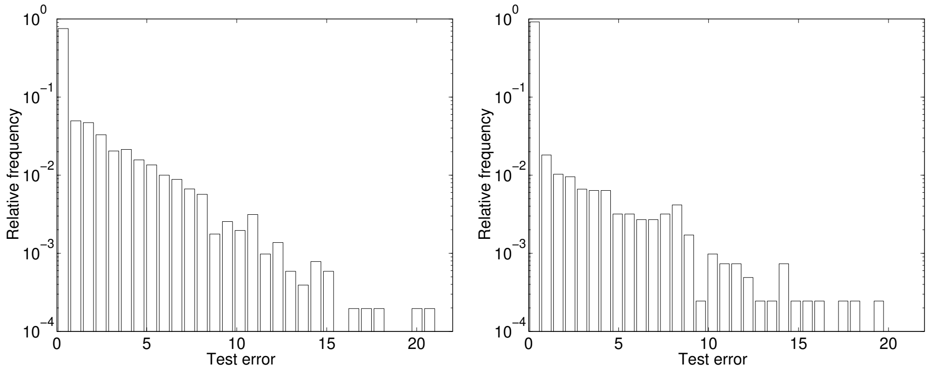

Figure 2: Left: The distribution of the test error for 10 fully connected *contiguity* networks combined. Notice the "long tail" of the distribution resulting in a high mean error (0.94) and a small median error (0.022) i.e., the mean is predominantly driven by a few examples with high error. Right: The distribution of the test error for 10 pruned networks selected by the minimum of the estimated test error combined. The mean error is 0.37 and the median error is 0.0014 showing a significant performance improvement as result of pruning.

Initially a network architecture consisting of 10 input units, 8 hidden units and 2 output units was chosen. The weight decay parameters were estimated by using the previously described sample-grid technique. This is shown in figure 1. Next ten fully connected networks were trained[3] using the estimated weight decay parameters, subsequently pruned using the OBD saliency ranking, removing one weight per iteration. In figure 2 the distribution of the test error is shown. The error distribution shows that the mean error is predominantly driven by a few examples with a high error, thus suggesting that one should monitor the median error as well in order to get a good indication of a network's performance. Seven of the ten pruned networks had a classification[4]

[3]Training was stopped when the 2-norm of the gradient vector was below 10^{-5}.

[4]Following Bayes decision theory, the network output with the highest probability determines the class label.

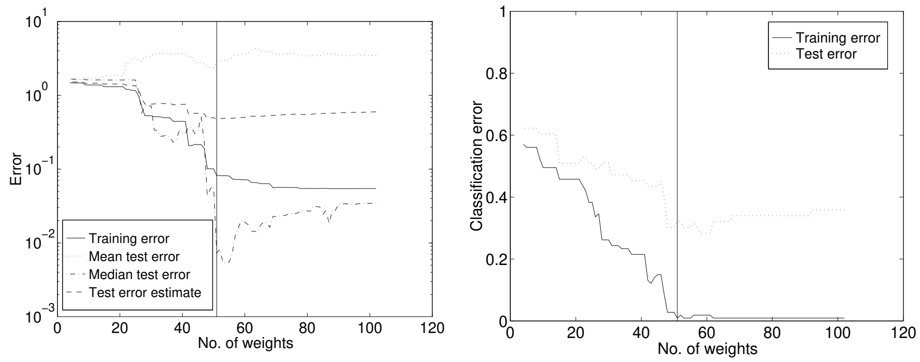

Figure 3: Pruning of a *glass classification* network using the small training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error. Notice the similarity in the development of the median error and the classification error on the test set.

error on the test set between 0% and 3.3%, while three networks had an error of 16-19%. In [10] seven of ten networks had an error of 8-38% using the same size of training set, while three networks had errors around 0%. Compared with these results, our classifier design scheme has a significantly higher yield.

The Glass Classification Problem

The task in the glass classification problem is to classify glass splinters into six classes. The glass splinters have been chemical analyzed and nine different measures have been extracted from the analysis, see [9] for details. The original dataset (*glass1*) consists of 214 examples divided into a training set (107), a validation set (54) and a test set (53). Since our approach doesn't require a validation set, we have used two different training scenarios: one using the original training set and one using a new training set consisting of the original training and validation set.

The initial network architecture chosen consisted of 9 input units, 6 hidden units and 6 output units. We estimated the regularization parameters using the sample-grid technique and the small training set. The parameters were found to be $\alpha_w = 2.2 \cdot 10^{-2}$ and $\alpha_W = 4.7 \cdot 10^{-4}$.

In figure 3 and 4 we show the pruning results of networks trained with the small and large training set, respectively, using the estimated regularization parameters. The "optimal" network found with the small training set had a classification error of 32% on the test set, while the "optimal" network found with the large training set had an error of 28%. In [9] Prechelt reports a test error of 32% for a fixed network architecture using the small training set. The validation set is used to stop training, thus he effectively uses both the training and validation set for training [11]. Our approach using the estimated test error for model selection eliminates the need for a validation

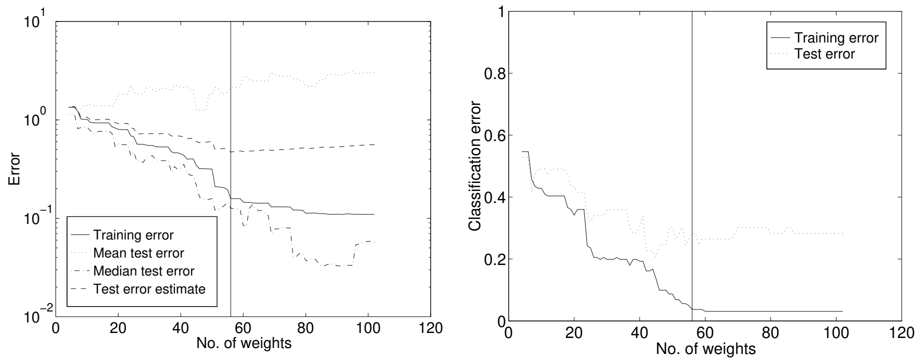

Figure 4: Pruning of a *glass classification* network using the large training set. The vertical line indicates the "optimal" network selected by the estimated test error. Notice the overall lower classification error on the test set compared to figure 3.

set, thus allowing us to use more data for the actual training resulting in a better generalization performance. In a forthcoming paper the problem of comparing the performance of neural network models is addressed [12].

For comparison a standard *k-Nearest-Neighbor*[5] (k-N-N) classification was performed using the large training set. The training error may be computed from the training set by including each training pattern in the majority vote. A *leave-one-out* "validation" error on the training set may be computed by excluding each training pattern from the vote. Finally, the test patterns may be classified by voting among the k nearest neighbors found among the training patterns. Using the *leave-one-out* validation error we found that $k = 2$ was optimal for this data set. The 2-N-N scheme had a classification error of 34% on the test set. Thus the performance of the optimized k-N-N scheme cannot match Prechelt's or our networks.

CONCLUSION

We have developed a methodology for design and evaluation of neural classifiers. The approach was applied to the *contiguity problem* and the *glass classification problem*. It was shown that the test error estimator for classifiers could be used to select optimal networks among families of pruned networks, thus increasing the generalization ability compared to non-pruned networks. Currently, the aim is to establish more empirical data for the validation of the neural classifier design approach.

[5] Within k-N-N a pattern is classified according to a majority vote among its k nearest neighbors using the simple Euclidean metric.

ACKNOWLEDGMENT

This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. Jan Larsen thanks the Radio-Parts Foundation for financial support.

REFERENCES

[1] C. Svarer, L.K. Hansen, and J. Larsen. "On Design and Evaluation of Tapped-Delay Neural Network Architectures". **Proceedings of the 1993 IEEE International Conference on Neural Networks**, pages 46–51, 1993.

[2] C. Svarer, L.K. Hansen, J. Larsen, and C.E. Rasmussen. "Designer Networks for Time Series Processing". **Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing III**, pages 78–97, 1993.

[3] M. Hintz-Madsen, L.K. Hansen, J.Larsen, E. Olesen, and K.T. Drzewiecki. "Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification". **Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V**, pages 484–493, 1995.

[4] J.S. Bridle. "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition". **Neurocomputing - Algorithms, Architectures and Applications**, 6:227–236, 1990.

[5] G.A.F. Seber and C.J. Wild. **Nonlinear Regression**. John Wiley & Sons, New York, New York, 1995.

[6] Y. Le Cun, J. Denker, and S. Solla. "Optimal Brain Damage". **Advances in Neural Information Processing Systems**, 2:598–605, 1990.

[7] J. Larsen, L.K. Hansen, C. Svarer, and M. Ohlsson. "Design and Regularization of Neural Networks: The Optimal Use of A Validation Set". **Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI**, 1996.

[8] S. Amari and N. Murata. "Statistical Theory of Learning Curves under Entropic Loss Criterion". **Neural Computation**, 5:140–153, 1993.

[9] Lutz Prechelt. "PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms". **Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany**, 1994. *Available via anonymous ftp* `ftp.ira.uka.de/pub/papers/techreports/1994/1994-21.ps.Z`.

[10] J. Gorodkin, L.K. Hansen, A. Krogh, C. Svarer, and O. Winther. "A Quantitative Study of Pruning by Optimal Brain Damage". **International Journal of Neural Systems**, 4:159–169, 1993.

[11] J. Sjöberg. "Non-Linear System Identification with Neural Networks". **Ph.D. Thesis no. 381, Department of Electrical Engineering, Linköping University, Sweden**, 1995.

[12] J. Larsen et al. "Empirical Comparison of Neural Network Models". **In preparation**, 1996.

Appendix E

Contribution to NNSP'97

This appendix contains the paper "Adaptive Regularization of Neural Classifiers" presented at the 1997 IEEE Workshop on Neural Networks for Signal Processing (NNSP'97) in Florida, USA [Andersen et al., 1997].

ADAPTIVE REGULARIZATION OF NEURAL CLASSIFIERS

L. Nonboe Andersen, J. Larsen, L.K. Hansen & M. Hintz-Madsen
CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
Phones: +45 4525+ext. 3899,3923,3889,3894
Fax: +45 45872599
emails: lna,jl,lkhansen,mhm@imm.dtu.dk
www: http://eivind.imm.dtu.dk

Abstract. In this paper we present a regularization scheme which iteratively adapts the regularization parameters by minimizing the validation error. It is suggested to use the adaptive regularization scheme in conjunction with Optimal Brain Damage pruning to optimize the architecture and to avoid overfitting. Furthermore, we propose an improved neural classification architecture eliminating an inherent redundancy in the widely used SoftMax classification network. Numerical results demonstrate the viability of the method.

INTRODUCTION

Neural networks are flexible tools for pattern recognition and by expanding the network architecture any relevant target function can be approximated [6]. In this contribution we present an improved version of the neural classifier architecture based on a feed-forward net with SoftMax [2] normalization presented in [7], [8] avoiding an inherent redundant parameterization. The outputs of the network estimate the class conditional posterior probabilities and the network is trained using a maximum a posteriori (MAP) framework.

The associated risk of overfitting on noisy data is of major concern in neural network design [4]. The objective of architecture optimization is to minimize the generalization error. The architecture can be optimized *directly* by e.g., pruning techniques or *indirectly* by using regularization. One might consider various regularization schemes: from adapting a single regularization parameter to individual regularization of the weights in the net. These subjects are further addressed in [9], [10]. We suggest a hybrid approach with Optimal Brain Damage [11] for pruning and an adaptive regularization scheme. The inevitable problem of adapting the *amount* of regularization is solved by minimizing the generalization error w.r.t. regularization parameters. Using the validation error calculated from a single validation set as an estimate of the generalization error, it is possible to formulate an iterative gradient descent scheme for adapting the regularization parameters [9]. The Bayesian way to adapt regularization parameters is to minimize the evidence [1, Ch. 10], [14]; however, the evidence does not, in a simple way, relate to the generalization error which is our primary object of interest.

NETWORK ARCHITECTURE

Suppose that the input (feature) vector is denoted by $\boldsymbol{x}$ with $\dim(\boldsymbol{x}) = n_I$. The aim is to model the posterior probabilities $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c$ where $\mathcal{C}_i$ denotes the i'th class. Then under a simple loss function the Bayes optimal[1] classifier assigns class label $\mathcal{C}_i$ to $\boldsymbol{x}$ if $i = \arg\max_j p(\mathcal{C}_j|\boldsymbol{x})$.

Following [8] (see also [1]), the outputs, $\widehat{y}_i$, of the neural network represent *estimates* of the posterior probabilities, i.e., $\widehat{y}_i = \widehat{p}(\mathcal{C}_i|\boldsymbol{x})$; hence, $\sum_{i=1}^{c} p(\mathcal{C}_i|\boldsymbol{x}) = 1$. That is, we need merely to estimate $c-1$ posterior probabilities, say $p(\mathcal{C}_i|\boldsymbol{x})$, $i = 1, 2, \cdots, c-1$, then the last is calculated as $p(\mathcal{C}_c|\boldsymbol{x}) = 1 - \sum_{i=1}^{c-1} p(\mathcal{C}_i|\boldsymbol{x})$.

Define a 2-layer feed-forward network with n_I inputs, n_H hidden neurons and $c-1$ outputs by:

$$h_j(\boldsymbol{x}) = \tanh\left(\sum_{\ell=1}^{n_I} w_{j\ell}^I x_\ell + w_{j0}^I\right), \quad \phi_i(\boldsymbol{x}) = \sum_{j=1}^{n_H} w_{ij}^H h_j(\boldsymbol{x}) + w_{i0}^H \tag{1}$$

where $w_{j\ell}^I$, w_{ij}^H are the input-to-hidden and hidden-to-output weights, respectively. All weights are assembled in the weight vector $\boldsymbol{w} = \{w_{j\ell}^I, w_{ij}^H\}$.

In order to interpret the network outputs as probabilities a *modified* normalized exponential transformation similar to SoftMax [2] is used,

$$\widehat{y}_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c-1}\exp(\phi_j) + 1}, \quad i = 1, 2, \cdots, c-1, \qquad \widehat{y}_c = 1 - \sum_{i=1}^{c-1} \widehat{y}_i. \tag{2}$$

The modification amounts to fixing $\exp(\phi_c)$ in the standard SoftMax at 1 eliminating the inherent redundancy of the output weights as also mentioned in [18, p. 150]. The redundancy implies that a particular set of outputs, $\widehat{y}_i$, $i = 1, 2, \cdots, c$ induces a one-dimensional sub-manifold in weight space. The network architecture is shown in Fig. 1.

TRAINING AND REGULARIZATION

Assume that we have a training set $\mathcal{T}$ of N_t related input-output pairs $\mathcal{T} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N_t}$ where

$$y_i(k) = \begin{cases} 1 & \text{if } \boldsymbol{x}(k) \in \mathcal{C}_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

[1] That is, each misclassification is equally serious corresponding to minimal probability of misclassification.

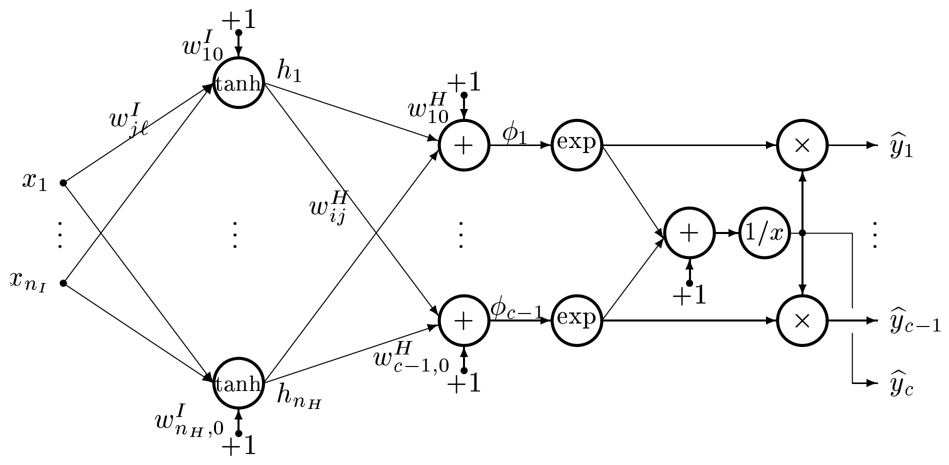

Figure 1: Neural network architecture.

The likelihood of the network parameters is given by (see e.g., [1], [8]),

$$p(\mathcal{T}|\boldsymbol{w}) = \prod_{k=1}^{N_t} p(\boldsymbol{y}(k)|\boldsymbol{x}(k),\boldsymbol{w}) = \prod_{k=1}^{N_t}\prod_{i=1}^{c} \left(\widehat{y}_i(k)\right)^{y_i(k)} \tag{4}$$

where $\widehat{\boldsymbol{y}}(k) = \widehat{\boldsymbol{y}}(\boldsymbol{x}(k),\boldsymbol{w})$ is a function of the input and weight vectors. The training error is the normalized negative log-likelihood

$$S_{\mathcal{T}}(\boldsymbol{w}) = -\frac{1}{N_t}\log p(\mathcal{T}|\boldsymbol{w}) \equiv \frac{1}{N_t}\sum_{k=1}^{N_t} \ell\left(\boldsymbol{y}(k),\widehat{\boldsymbol{y}}(k);\boldsymbol{w}\right) \tag{5}$$

with $\ell(\cdot)$ denoting the loss given by

$$\ell\left(\boldsymbol{y}(k),\widehat{\boldsymbol{y}}(k);\boldsymbol{w}\right) = \sum_{i=1}^{c} y_i(k)\log\left(1+\sum_{j=1}^{c-1}\exp(\phi_j(\boldsymbol{x}(k)))\right) - \sum_{i=1}^{c-1} y_i(k)\phi_i(\boldsymbol{x}(k)). \tag{6}$$

The objective of training is minimization of the regularized cost function[2]

$$C(\boldsymbol{w}) = S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w},\boldsymbol{\kappa}) \tag{7}$$

where the regularization term $R(\boldsymbol{w},\boldsymbol{\kappa})$ is parameterized by a set of regularization parameters $\boldsymbol{\kappa}$. Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$ and is done using a Gauss-Newton scheme,

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} - \eta\cdot\boldsymbol{J}^{-1}(\boldsymbol{w}^{\text{old}})\boldsymbol{\nabla}(\boldsymbol{w}^{\text{old}}) \tag{8}$$

[2]This might be viewed as a maximum a posteriori (MAP) method.

where η is the step-size (line search parameter). For that purpose we require the gradient, $\boldsymbol{\nabla}(\boldsymbol{w}) = \partial C/\partial \boldsymbol{w}$, and the Hessian, $\boldsymbol{J}(\boldsymbol{w}) = \partial^2 C/\boldsymbol{w}\boldsymbol{w}^\top$ of the cost function given by,

$$\frac{\partial C}{\partial \boldsymbol{w}}(\boldsymbol{w}) = -\frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}\left[y_i(k) - \widehat{y}_i(k)\right]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}} + \frac{\partial R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}}, \tag{9}$$

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c-1}\sum_{j=1}^{c-1}\widehat{y}_i(k)\left[\delta_{ij} - \widehat{y}_j(k)\right]\frac{\partial \phi_i(\boldsymbol{x}(k))}{\partial \boldsymbol{w}}\frac{\partial \phi_j(\boldsymbol{x}(k))}{\partial \boldsymbol{w}^\top} + \frac{\partial^2 R(\boldsymbol{w},\boldsymbol{\kappa})}{\partial \boldsymbol{w}\partial \boldsymbol{w}^\top}. \tag{10}$$

Here δ_{ij} is the Kronecker delta and we have used the Gauss-Newton approximation to the Hessian.

ADAPTING REGULARIZATION PARAMETERS

The available data set, $\mathcal{D}$, of N examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ examples for architecture selection and estimation of regularization, and a training set, $\mathcal{T}$, with $N_t = N - N_v$ examples for estimation of network parameters. γ is referred to as the split-ratio.

The validation error of the trained network is given by

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v}\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \tag{11}$$

where the sum runs over the N_v validation examples. $S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$ is thus an estimate of the generalization error defined as the expected loss: $G(\widehat{\boldsymbol{w}}) = E_{\boldsymbol{x},\boldsymbol{y}}\{\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \widehat{\boldsymbol{w}})\}$, where $E_{\boldsymbol{x},\boldsymbol{y}}\{\cdot\}$ denotes the expectation w.r.t. the joint input-output distribution.

Aiming at adapting the regularization parameters $\boldsymbol{\kappa}$ so that the validation error is minimized we can apply the iterative scheme suggested in [9]:

$$\boldsymbol{\kappa}^{\mathrm{new}} = \boldsymbol{\kappa}^{\mathrm{old}} - \mu\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\mathrm{old}})) \tag{12}$$

where μ is a step-size and $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{\mathrm{old}})$ is the estimated weight vector using the regularization parameter $\boldsymbol{\kappa}^{\mathrm{old}}$. Suppose the regularization term is linear in the regularization parameters, i.e.,

$$R(\boldsymbol{w},\boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q}\kappa_i r_i(\boldsymbol{w}) \tag{13}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ the associated regularization functions. The gradient of the validation error then equals [9]:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}). \tag{14}$$

Consider the specific case of weight decay regularization with separate weight decays for input-to-hidden and hidden-to output layers, i.e.,

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa_I \cdot |\boldsymbol{w}^I|^2 + \kappa_H \cdot |\boldsymbol{w}^H|^2 \tag{15}$$

where $\boldsymbol{\kappa} = [\kappa_I, \kappa_H]^\top$ and $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^H]$ with $\dim(\boldsymbol{w}^I) = m_I$, $\dim(\boldsymbol{w}^H) = m_H$ and $\dim(\boldsymbol{w}) = m = m_I + m_H$.

The gradient then yields,

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa_I}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^I)^\top \cdot \boldsymbol{g}_{m_I}, \quad \frac{\partial S_{\mathcal{V}}}{\partial \kappa_H}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}^H)^\top \cdot \boldsymbol{g}_{m_H} \tag{16}$$

where $\boldsymbol{g} = [\boldsymbol{g}_{m_I}, \boldsymbol{g}_{m_H}] = \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \partial S_{\mathcal{V}}(\widehat{\boldsymbol{w}})/\partial \boldsymbol{w}$.

In summary the algorithm for adapting regularization parameters consists of the following 8 steps:

1. Choose the split ratio γ between training and validation set sizes.
2. Initialize $\boldsymbol{\kappa}$ and the weights of the network.
3. Train the network with fixed $\boldsymbol{\kappa}$ to achieve $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})$. Calculate the validation error $S_{\mathcal{V}}$.
4. Calculate the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ cf. Eq. (14). Initialize the step-size μ.
5. Update $\boldsymbol{\kappa}$ using Eq. (12).
6. Retrain the network from the previous weights and calculate the validation error $S_{\mathcal{V}}$.
7. If no decrease in validation error then perform a bisection of μ and goto step 5; otherwise, continue.
8. Repeat steps 4–7 until the relative change in validation error is below a small percentage or, e.g., the 2-norm of the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}$ is below a small number.

PRUNING

In order to reduce and optimize the network architecture we suggest to use a pruning scheme, e.g., Optimal Brain Damage (OBD) [11]. An alternative method is Optimal Brain Surgeon (OBS) [5]; however, in a series of experiments we noticed that extreme care is essential in order not to underestimate the saliencies [16]. Thus OBS is less robust than OBD.

OBD ranks the weights according to importance or *saliency*. Here we use the validation error based OBD proposed in [9]. The saliency for weight i is given by

$$\varrho_i = -\widehat{w}_i \frac{\partial S_{\mathcal{V}}}{\partial w_i}(\widehat{\boldsymbol{w}}) + \frac{1}{2}\widehat{w}_i^2 \frac{\partial^2 S_{\mathcal{V}}}{\partial w_i^2}(\widehat{\boldsymbol{w}}). \tag{17}$$

By repeatedly removing weights with small saliencies and retraining the resulting network, a nested family of network architectures is obtained. The validation error (or an alternative measure of generalization performance[3]) is then used for selecting the optimal architecture.

EXPERIMENTS

We test the performance of the adaptive regularization algorithm on a vowel classification problem. The data are based on the Peterson and Barney database [17]. The classes are vowel sounds characterized by the first four formant frequencies. 76 persons (33 male, 28 female and 15 children) have pronounced $c = 10$ different vowels (IY IH EH AE AH AA AO UH UW ER) two times. This results in a data base of totally 1520 examples. The database is the verified database described in [22] where all data[4] are used, including examples where utterance failed of unanimous identification in the listening test (26 listeners). All examples were included to make the task more difficult.

The examples were split into a data set, $\mathcal{D}$, consisting of $N = 760$ examples (16 male, 14 female and 8 children) and an independent test set of the remaining 760 examples. The regularization was adapted by splitting the data set $\mathcal{D}$ equally into a validation set of $N_v = 380$ examples and a training set of $N_t = 380$ examples (8 male, 7 female and 4 children in each set).

Suppose that the network weights are given by $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^I_{\text{bias}}, \boldsymbol{w}^H, \boldsymbol{w}^H_{\text{bias}}]$ where $\boldsymbol{w}^I$, $\boldsymbol{w}^H$ are input-to-hidden and hidden-to-output weights, respectively, and the bias weights are assembled in $\boldsymbol{w}^I_{\text{bias}}$ and $\boldsymbol{w}^H_{\text{bias}}$. In this example, we use the following weight decay regularization term:

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa^I \cdot |\boldsymbol{w}^I|^2 + \kappa^I_{\text{bias}} \cdot |\boldsymbol{w}^I_{\text{bias}}|^2 + \kappa^H \cdot |\boldsymbol{w}^H|^2 + \kappa^H_{\text{bias}} \cdot |\boldsymbol{w}^H_{\text{bias}}|^2. \quad (18)$$

where $\boldsymbol{\kappa} = [\kappa^I, \kappa^I_{\text{bias}}, \kappa^H, \kappa^H_{\text{bias}}]$. We further define the normalized weight decays as $\boldsymbol{\alpha} \equiv \boldsymbol{\kappa} \cdot N_t$. The simulation set-up was:

- Network: 4 inputs, 5 hidden neurons, 9 outputs[5].
- The training input data were normalized to zero mean and unit variance in order to facilitate training and weight initialization.
- Weights were initialized uniformly over $[-0.5, 0.5]$, regularization parameters were initialized at zero. 10 steps in a gradient descent training algorithm (see e.g., [12]) was performed and the weight decays, $\boldsymbol{\kappa}$, were re-initialized at $\lambda_{\max}/10^2$, where $\lambda_{\max}$ is the max. eigenvalue of the Hessian matrix of the cost function. This initialization scheme is motivated by the following observations:
  - Weight decays should be so small that they do not reduce the approximation capabilities of the network significantly.

[3] E.g., the previously suggested algebraic estimate [8], [15].

[4] The database can be retrieved from `ftp://eivind.imm.dtu.dk/dist/data/vowel/PetersonBarney.tar.Z`

[5] We only need 9 outputs since the posterior class probability of the 10th class is given by $1 - \sum_{j=1}^{9} p(\mathcal{C}_j|\boldsymbol{x})$.

	NNet	**KNN** ($K = 9$)
Training	0.105 ± 0.008	0.150
Validation	0.115 ± 0.005	0.158
Test	0.122 ± 0.005	0.199
Test after retrain.	0.119 ± 0.003	0.153

Table 1: Probability of misclassification, *pmc*. For the neural network the averages and standard deviations over 6 runs are reported.

  - They should be so large that the algorithm is prevented from being trapped in a local optimum and numerical instabilities are eliminated.
- Training is now done using a Gauss-Newton algorithm (see e.g., [12]). The Hessian is inverted using the Moore-Penrose pseudo inverse (see e.g., [19]) ensuring that the eigenvalue spread[6] is less than 10^8.
- The regularization step-size η is initialized at 1.
- When the adaptive regularization scheme has terminated we prune 3% of the weights using a validation set based version of the Optimal Brain Damage recipe [9], [11].
- We alternate between pruning and adaptive regularization until the validation error has reached a minimum.
- Finally, remaining weights are retrained on all data using the optimized weight decay parameters.

Table 1 reports the average and standard deviations of the probability of misclassification (*pmc*) over 6 runs for pruned networks using the optimal regularization parameters. Note that retraining on the full data set decreases the test *pmc* slightly on the average; improvement was found in 4 out of 6 runs. For comparison we used a K-nearest-neighbor (KNN) classification, see e.g., [1] and found that $K = 9$ was optimal on the validation set. Note that the neural network performed significantly better. Contrasting the obtained results to other work is difficult. In [20] results on the Peterson-Barney vowel problem are reported, but their data are not exactly the same; only the first 2 formant frequencies were used. Furthermore, different test sets have been used for the different methods presented. The best result reported [13] is obtained by using KNN and reach $pmc = 0.186$ which is somewhat higher than our results.

In Fig. 2 the evolution of the adaptive regularization as well as the pruning algorithm is demonstrated.

CONCLUSIONS

This paper presented a framework for design of neural classifiers which include architecture optimization by pruning and adaptation of regularization

[6]Eigenvalue spread should not be larger than the square root of the machine precision [3].

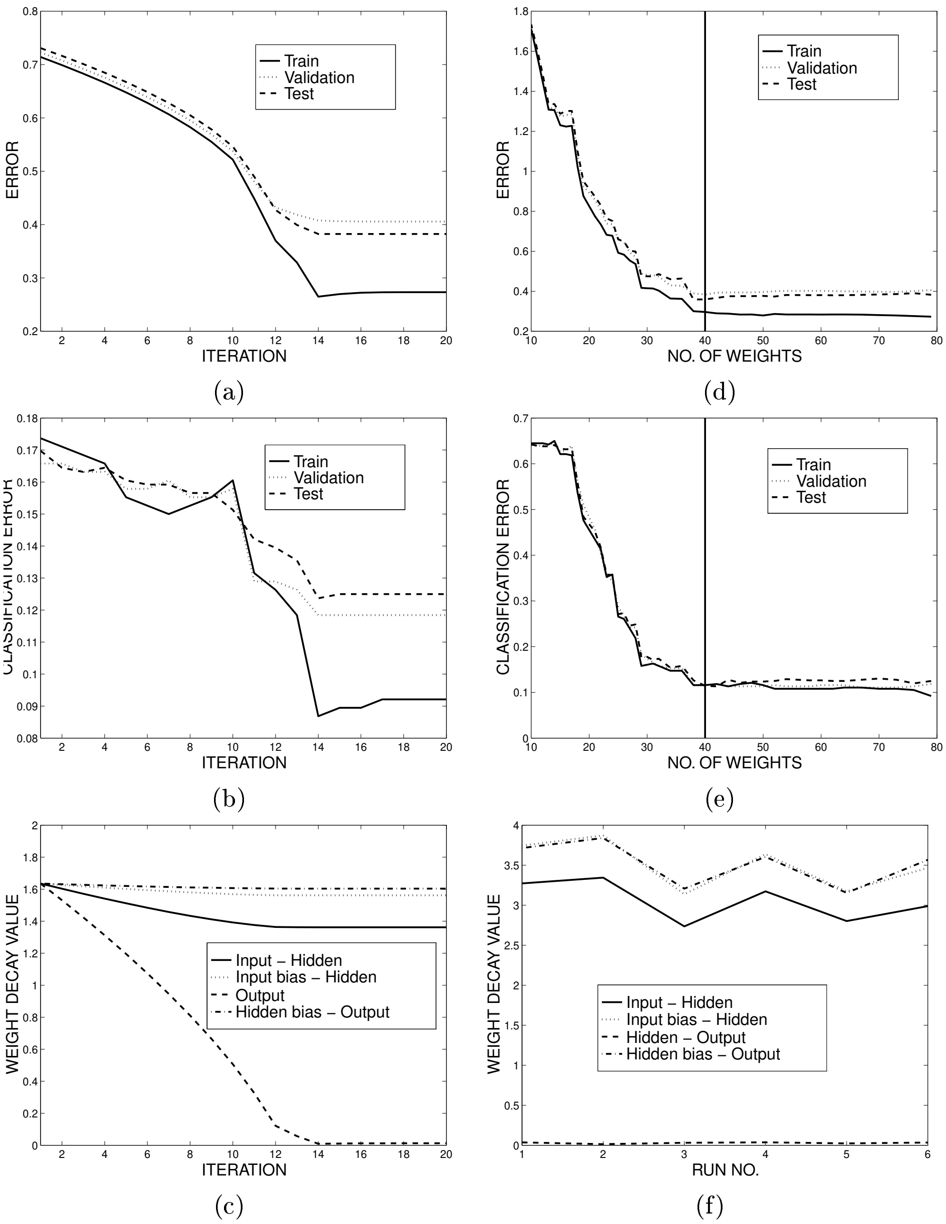

Figure 2: Panels (a), (b) and (c) show the evolution of the adaptive regularization algorithm in a typical run. Optimal weight decays are found by minimizing the validation error in (a). Note that also the test errors decreases. This tendency is also evident in (b) displaying *pmc* even though a small increase is noticed. In (c) the normalized weight decays, $\alpha = \kappa \cdot N_t$, are depicted. (d) and (e) show the evolution of errors and *pmc* during pruning. The optimal network having minimal validation error is indicated by the vertical line. There is only a marginal effect of pruning. Finally, the variation of the optimal normalized weight decays (before pruning) in different runs is shown in (f) and is seen to be relatively small.

parameters. Moreover, an improved neural net architecture was presented. Numerical examples demonstrated the potential of the framework.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support. Allan René Rasmussen is thanked for programming assistance.

REFERENCES

[1] C.M. Bishop: **Neural Networks for Pattern Recognition**, Oxford, UK: Oxford University Press, 1995.

[2] J.S. Bridle: "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition," **Neurocomputing - Algorithms, Architectures and Applications**, Berlin Germany: Springer-Verlag, vol. 6, pp. 227–236, 1990.

[3] J.E. Dennis & R.B. Schnabel: **Numerical Methods for Unconstrained Optimization and Non-linear Equations**, Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

[4] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," **Neural Computation**, vol. 4, pp. 1–58, 1992.

[5] B. Hassibi & D.G. Stork: "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in S.J. Hanson, J. Cowan & C. Giles (eds.) **Advances in Neural Information Processing Systems 5, Proceedings of the 1992 Conference**, San Mateo, California: Morgan Kaufmann Publishers, 1993, pp. 164–171.

[6] K. Hornik: "Approximation Capabilities of Multilayer Feedforward Networks," **Neural Networks**, vol. 4, pp. 251–257, 1991.

[7] M. Hintz-Madsen, L.K. Hansen, J. Larsen, E. Olesen & K.T. Drzewiecki: "Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification," in F. Girosi, J. Makhoul, E. Manolakos & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V**, Piscataway, New Jersey: IEEE, 1995, pp. 484–493.

[8] M. Hintz-Madsen, M. With Pedersen, L.K. Hansen, & J. Larsen: "Design and Evaluation of Neural Classifiers," in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), **Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VI**, Piscataway, New Jersey: IEEE, 1996, pp. 223–232.

[9] J. Larsen, L.K. Hansen, C. Svarer & M. Ohlsson: "Design and Regularization of Neural Networks: The Optimal Use of a Validation Set," in S. Usui, Y. Tohkura, S. Katagiri & E. Wilson (eds.), **Proceedings of**

the IEEE Workshop on Neural Networks for Signal Processing VI, Piscataway, New Jersey: IEEE, 1996, pp. 62–71.

[10] J. Larsen, C. Svarer, L. Nonboe Andersen & L.K. Hansen: "Adaptive Regularization in Neural Network Modelling," preprint IMM, DTU, submitted 1997. Available by `ftp://eivind.imm.dtu.dk/dist/1997/larsen.bot.ps.Z`.

[11] Y. Le Cun, J.S. Denker & S.A. Solla: "Optimal Brain Damage," in D.S. Touretzky (ed.) **Advances in Neural Information Processing Systems 2, Proceedings of the 1989 Conference**, San Mateo, California: Morgan Kaufmann Publishers, 1990, pp. 598–605.

[12] L. Ljung: **System Identification: Theory for the User**, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[13] D. Lowe: "Adaptive Radial Basis Function Nonlinearities and the Problem of Generalisation," **Proceedings of IEE Conference on Artificial Neural Networks**, 1989, pp. 171–175.

[14] D.J.C. MacKay: "A Practical Bayesian Framework for Backprop Networks," **Neural Computation**, vol. 4, no. 3, pp. 448–472, 1992.

[15] N. Murata, S. Yoshizawaand & S. Amari: "Network Information Criterion — Determining the Number of Hidden Units for an Artificial Neural Network Model," **IEEE Transactions on Neural Networks**, vol. 5, no. 6, pp. 865–872, Nov. 1994.

[16] M.W. Pedersen, L.K. Hansen & J. Larsen: "Pruning with Generalization Based Weight Saliencies: γOBD, γOBS," in D.S. Touretzky, M.C. Mozer & M.E. Hasselmo (eds.) **Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference**, Cambridge, Massachusetts: MIT Press, pp. 521–528, 1996.

[17] G.E. Peterson & H.L. Barney: "Control Methods Used in a Study of the Wowels," **JASA** vol. 24, pp. 175–184, 1952.

[18] B.D. Ripley: **Pattern Recognition and Neural Networks**, Cambridge, UK: Cambridge University Press, 1996.

[19] G.A.F. Seber & C.J. Wild: **Nonlinear Regression**, New York, New York: John Wiley & Sons, 1989.

[20] R.S. Shadafan & M. Niranjan: "A Dynamic Neural Network Architecture by Sequential Partitioning of the Input Space," **Neural Computation**, vol. 6, no. 6, pp. 1202–1222, 1994.

[21] C. Svarer, L.K. Hansen & J. Larsen: "On Design and Evaluation of Tapped-Delay Neural Architectures," in **Proceedings of the IEEE International Conference on Neural Networks**, San Francisco, California, USA, vol. 1, pp. 46–51, 1993.

[22] R.L. Watrous: "Current Status of PetersonBarney Vowel Formant Data," **JASA**, vol. 89 pp. 2459–2460, 1991.

Appendix F

Contribution to ICASSP'98

This appendix contains the paper "Design of Robust Neural Network Classifiers" presented at the 1998 International Conference on Acoustics, Speech and Signal Processing (ICASSP'98) in Seattle, USA [Larsen et al., 1998a].

DESIGN OF ROBUST NEURAL NETWORK CLASSIFIERS

Jan Larsen, Lars Nonboe Andersen, Mads Hintz-Madsen and Lars Kai Hansen

CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
emails: jl,lna,mhm,lkhansen@imm.dtu.dk
www: http://eivind.imm.dtu.dk

ABSTRACT

This paper addresses a new framework for designing robust neural network classifiers. The network is optimized using the maximum a posteriori technique, i.e., the cost function is the sum of the log-likelihood and a regularization term (prior). In order to perform robust classification, we present a modified likelihood function which incorporate the potential risk of outliers in the data. This leads to introduction of a new parameter, the outlier probability. Designing the neural classifier involves optimization of network weights as well as outlier probability and regularization parameters. We suggest to adapt the outlier probability and regularization parameters by minimizing the error on a validation set, and a simple gradient descent scheme is derived. In addition, the framework allows for constructing a simple outlier detector. Experiments with artificial data demonstrates the potential of the suggested framework.

1. INTRODUCTION

Neural networks are flexible tools for pattern recognition due to the universal approximation theorems [6]. We consider a neural classifier architecture based on a feed-forward net with a modified SoftMax [3] normalization as presented in [1] (see also, [2], [7]). The outputs of the network estimate the class conditional posterior probabilities and the network is trained using a maximum a posteriori (MAP) framework. Robustness is incorporated via a probabilistic definition of outliers. Thus a given example is considered as an outlier if its class label is changed with a certain probability ε; the *outlier probability*.

The associated risk of overfitting on noisy data is of major concern in neural network design [5]. The objective of network design is to obtain a reliable and minimal generalization error which can be done by constraining the model flexibility and adapting the outlier probability. Model constraints are imposed directly via pruning techniques (see e.g., [1], [9], [10]) or indirectly using regularization. We will merely consider regularization in this presentation.

Based on earlier work [1], [9], [10], we will present an iterative scheme for simultaneously adapting the amount of regularization and outlier probability by minimizing the validation error calculated from a single validation set. Here we take the validation error as an estimate of the true generalization error.

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support.

2. NETWORK ARCHITECTURE

Suppose that the input (feature) vector is denoted by $\boldsymbol{x}$ with $\dim(\boldsymbol{x}) = n_I$ and $\mathcal{C}_i$ denotes the i'th of the mutually exclusive classes, $i = 1, 2, \cdots, c$. The aim is to model the posterior probabilities of the class given the input. Aiming at robustness against an outlier[1], defined as a class label which erroneously is changed to one of the other classes, we introduce the probability of an outlier, $0 \le \varepsilon \le 1$. It is assumed that the occurrence of an outlier is independent of class label and input location. Thus the outlier process acts as an extra noise source independent of input location, as opposed to the error due to overlap in class posterior probabilities. This leads to the definition of posterior probabilities, $p(\mathcal{C}_i|\boldsymbol{x})$, $1 \le i \le c$,

$$p(\mathcal{C}_i|\boldsymbol{x}) = p_0(\mathcal{C}_i|\boldsymbol{x}) \cdot (1-\varepsilon) + \frac{\varepsilon}{c-1} \sum_{j=1, j\neq i}^{c} p_0(\mathcal{C}_j|\boldsymbol{x}). \quad (1)$$

where $p_0(\mathcal{C}_i|\boldsymbol{x})$ is the posterior probability in the case of zero outlier probability. The first term is the posterior probability for $\mathcal{C}_i$ times the probability that an outlier does not occur while the second term is the sum of posterior probabilities for $\mathcal{C}_j \neq \mathcal{C}_i$ times the scaled outlier probability $\beta \equiv \varepsilon/(c-1) \in [0; 1/(c-1)]$ that $\mathcal{C}_i$ has changed specifically to $\mathcal{C}_j$. Under a simple loss function the Bayes optimal[2] classifier assigns class label $\mathcal{C}_i$ to $\boldsymbol{x}$ if $i = \arg\max_j p(\mathcal{C}_j|\boldsymbol{x})$. Note that Eq. (1) can be rewritten as

$$p(\mathcal{C}_i|\boldsymbol{x}) = p_0(\mathcal{C}_i|\boldsymbol{x})(1-\beta c) + \beta. \quad (2)$$

Since $0 \le p_0(\mathcal{C}_i|\boldsymbol{x}) \le 1$ and $\sum_{i=1}^{c} p_0(\mathcal{C}_i|\boldsymbol{x}) = 1$ due to mutual exclusive classes, then $\beta \le p(\mathcal{C}_i|\boldsymbol{x}) \le 1-\varepsilon$ and further $\sum_{i=1}^{c} p(\mathcal{C}_i|\boldsymbol{x}) = 1$.

Define $\widehat{y}_i$ as *estimates* of the posterior probabilities given by $\widehat{y}_i = \widehat{z}_i(1-\beta c) + \beta$ where $\widehat{z}_i$ are estimates of the $\varepsilon = 0$ posterior probabilities, $p_0(\mathcal{C}_i|\boldsymbol{x})$. Following [1], [7] (see also [2]), $\widehat{z}_i$, are taken as outputs of a neural network. Since the posterior probabilities sums to 1, also $\sum_{i=1}^{c} \widehat{z}_i = 1$, i.e., we merely estimate $c-1$ posterior probabilities, say $\widehat{z}_i$, $1 \le i \le c-1$, and calculate the last as $\widehat{z}_c = 1 - \sum_{i=1}^{c-1} \widehat{z}_i$.

Define a 2-layer feed-forward network with n_I inputs, n_H hidden neurons and $c-1$ outputs by:

$$h_j(\boldsymbol{x}) = \tanh\left(\sum_{\ell=1}^{n_I} w_{j\ell}^I x_\ell + w_{j0}^I\right), \quad (3)$$

[1] See [8] for various approaches on robust statistics.

[2] That is, each misclassification is equally weighted corresponding to minimal probability of misclassification.

$$\phi_i(\boldsymbol{x}) = \sum_{j=1}^{n_H} w_{ij}^H h_j(\boldsymbol{x}) + w_{i0}^H \quad (4)$$

where $w_{j\ell}^I$, w_{ij}^H are the input-to-hidden and hidden-to-output weights, respectively. All weights are assembled in the weight vector $\boldsymbol{w} = \{w_{j\ell}^I, w_{ij}^H\}$. In order to interpret the network outputs as probabilities we use a *modified* normalized exponential transformation [1] similar to SoftMax [3],

$$\widehat{z}_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c-1}\exp(\phi_j)+1},\ 1 \le i \le c-1, \quad (5)$$

and $\widehat{z}_c = 1 - \sum_{i=1}^{c-1}\widehat{z}_i$.

3. TRAINING AND REGULARIZATION

Assume that we have a training set $\mathcal{T}$ of N_t related input-output pairs $\mathcal{T} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N_t}$ where

$$y_i(k) = \begin{cases} 1 & \text{if } \boldsymbol{x}(k) \in \mathcal{C}_i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The likelihood of the network parameters is given by (see e.g., [2], [7]),

$$p(\mathcal{T}|\boldsymbol{w}) = \prod_{k=1}^{N_t} p(\boldsymbol{y}(k)|\boldsymbol{x}(k), \boldsymbol{w}) = \prod_{k=1}^{N_t}\prod_{i=1}^{c}(\widehat{y}_i(k))^{y_i(k)} \quad (7)$$

where $\widehat{\boldsymbol{y}}(k) = \widehat{\boldsymbol{y}}(\boldsymbol{x}(k), \boldsymbol{w})$ is a function of the input and weight vectors. The training error is the normalized negative log-likelihood

$$S_{\mathcal{T}}(\boldsymbol{w}) = -\frac{1}{N_t}\log p(\mathcal{T}|\boldsymbol{w}) \equiv \frac{1}{N_t}\sum_{k=1}^{N_t}\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) \quad (8)$$

with $\ell(\cdot)$ denoting the loss given by

$$\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) = \sum_{i=1}^{c} -y_i(k)\log(\widehat{y}_i(k)) \quad (9)$$

Making an comparison with M-estimates considered in robust statistics [8], we note that the loss for a specific example is $\ell = -\log(\widehat{y}_i) = \psi(\ell_0)$ where $\ell_0 = -\log(\widehat{z}_i)$ is the non-robust loss ($\varepsilon = 0$) and $\psi(\cdot)$ is a function which downweights extreme losses[3].

The objective of training is minimization of the regularized cost function[4]

$$C(\boldsymbol{w}) = S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w}, \boldsymbol{\kappa}) \quad (10)$$

where the regularization term $R(\boldsymbol{w}, \boldsymbol{\kappa})$ is parameterized by a set of regularization parameters $\boldsymbol{\kappa}$. Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w})$ and is done using a Gauss-Newton scheme (see e.g., [11]),

$$\boldsymbol{w}^{\text{new}} = \boldsymbol{w}^{\text{old}} - \eta \cdot \boldsymbol{J}^{-1}(\boldsymbol{w}^{\text{old}})\boldsymbol{\nabla}(\boldsymbol{w}^{\text{old}}) \quad (11)$$

where η is the step-size (line search parameter). For that purpose we require the gradient, $\boldsymbol{\nabla}(\boldsymbol{w}) = \partial C/\partial \boldsymbol{w}$, and the Gauss-Newton approximation[5], $\boldsymbol{J}(\boldsymbol{w})$, of the Hessian $\partial^2 C/\boldsymbol{w}\boldsymbol{w}^\top$ which can be written as

$$\frac{\partial C}{\partial \boldsymbol{w}}(\boldsymbol{w}) = \frac{\beta c - 1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c}\frac{\partial \widehat{z}_i}{\partial \boldsymbol{w}}\frac{y_i(k)}{\widehat{y}_i} + \frac{\partial R(\boldsymbol{w}, \boldsymbol{\kappa})}{\partial \boldsymbol{w}}, \quad (12)$$

$$\boldsymbol{J}(\boldsymbol{w}) = \frac{(1-\beta c)^2}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c}\frac{1}{\widehat{y}_i}\frac{\partial \widehat{z}_i}{\partial \boldsymbol{w}}\frac{\partial \widehat{z}_i}{\partial \boldsymbol{w}^\top} + \frac{\partial^2 R(\boldsymbol{w}, \boldsymbol{\kappa})}{\partial \boldsymbol{w}\partial \boldsymbol{w}^\top}. \quad (13)$$

where

$$\frac{\partial \widehat{z}_i}{\partial \boldsymbol{w}} = \widehat{z}_i\sum_{j=1}^{c-1}(\delta_{ij} - \widehat{z}_j)\frac{\partial \phi_j}{\partial \boldsymbol{w}},\ 1 \le i \le c-1, \quad (14)$$

$$\frac{\partial \widehat{z}_c}{\partial \boldsymbol{w}} = -\widehat{z}_c\sum_{j=1}^{c-1}\widehat{z}_j\frac{\partial \phi_j}{\partial \boldsymbol{w}}. \quad (15)$$

By convenience, the dependency of $\widehat{z}_i$, $\widehat{y}_i$ and ϕ_i on $\boldsymbol{x}(k)$ and $\boldsymbol{w}$ is omitted, and δ_{ij} denotes the Kronecker delta.

4. ADAPTING REGULARIZATION PARAMETERS AND OUTLIER PROBABILITY

The available data set, $\mathcal{D}$, of N examples is split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ examples for estimation of regularization and outlier probability, and a training set, $\mathcal{T}$, with $N_t = N - N_v$ examples for estimation of network parameters. γ is referred to as the split-ratio. The validation error of the trained network is given by

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v}\ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \quad (16)$$

where the sum runs over the N_v validation examples. $S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$ is thus an estimate of the generalization error defined as the expected loss: $G(\widehat{\boldsymbol{w}}) = E_{\boldsymbol{x},\boldsymbol{y}}\{\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \widehat{\boldsymbol{w}})\}$, where $E_{\boldsymbol{x},\boldsymbol{y}}\{\cdot\}$ denotes the expectation w.r.t. the joint input-output distribution.

Let $\boldsymbol{\theta} = [\boldsymbol{\kappa}, \beta]$ be the vector of all regularization parameters and the scaled outlier probability. Aiming at adapting $\boldsymbol{\theta}$ so as to minimize the validation error we apply the iterative scheme suggested in [1], [9], [10]:

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \mu\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\theta}}(\widehat{\boldsymbol{w}}(\boldsymbol{\theta}^{\text{old}})) \quad (17)$$

where μ is a step-size and $\widehat{\boldsymbol{w}}(\boldsymbol{\theta}^{\text{old}})$ is the estimated weight vector using $\boldsymbol{\theta}^{\text{old}}$. Suppose that

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q}\kappa_i r_i(\boldsymbol{w}) \quad (18)$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ the associated regularization functions. The gradient of the validation error then equals [9], [10]:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}), \quad (19)$$

[3] $\psi(\ell) = -\log(e^{-\ell}(1-\beta c) + \beta)$.

[4] This might be viewed as a maximum a posteriori (MAP) technique.

[5] This is obtained using Fisher's property: $E[\partial^2 L/\partial \boldsymbol{w}\partial \boldsymbol{w}^\top] = E[\partial L/\partial \boldsymbol{w}\, \partial L/\partial \boldsymbol{w}^\top]$ where $L = -\log p(\mathcal{T}|\boldsymbol{w})$.

$$\frac{\partial S_{\mathcal{V}}}{\partial \beta}(\widehat{\boldsymbol{w}}) = -\frac{1}{N_v}\sum_{k=1}^{N_v}\sum_{i=1}^{c}\frac{1-c\widehat{z}_i(\widehat{\boldsymbol{w}})}{\widehat{y}_i(\widehat{\boldsymbol{w}})}y_i(k)$$
$$-\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}})\cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}})\cdot\frac{1}{N_t}\sum_{k=1}^{N_t}\sum_{i=1}^{c}\frac{\partial \widehat{z}_i}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}})\frac{y_i(k)}{\widehat{y}_i^2(\widehat{\boldsymbol{w}})}. \quad (20)$$

In order to ensure that $\kappa_i \geq 0$ and $0 \leq \beta \leq 1/(c-1)$ we perform a reparameterization,

$$\kappa_i = \begin{cases} \exp(\lambda_i), \lambda_i < 0 \\ \lambda_i + 1 \ \ , \lambda_i \geq 0 \end{cases} \quad \beta = \frac{1+\tanh(\gamma)}{2(c-1)}, \gamma \in \mathbb{R} \quad (21)$$

and carry out the minimization w.r.t. the new parameters $\boldsymbol{\lambda}$ and γ assembled in the vector $\boldsymbol{\xi} = [\boldsymbol{\lambda}, \gamma]$. Note that

$$\frac{\partial S_{\mathcal{V}}}{\partial \xi_i} = \frac{\partial S_{\mathcal{V}}}{\partial \theta_i}\cdot\frac{\partial \theta_i}{\partial \xi_i}. \quad (22)$$

In summary the algorithm for adapting regularization parameters and outlier probability is:

1. Select the split ratio γ and initialize $\boldsymbol{\kappa}$, β and the weights of the network.
2. Train the network with fixed $\boldsymbol{\xi}$ to achieve $\widehat{\boldsymbol{w}}(\boldsymbol{\xi})$. Calculate the validation error $S_{\mathcal{V}}$.
3. Calculate the gradient $\partial S_{\mathcal{V}}/\partial \boldsymbol{\xi}$ cf. Eq. (19), (20). Initialize the step-size μ.
4. Update $\boldsymbol{\xi}$ using Eq. (17), (21), train the network from the previous weights and calculate $S_{\mathcal{V}}$.
5. If $S_{\mathcal{V}}$ decreases repeat: double μ, update $\boldsymbol{\xi}$, retrain weights and recalculate $S_{\mathcal{V}}$ until no decrease is noticed then goto step 7.
6. Repeat: perform bisection of μ, update $\boldsymbol{\xi}$, retrain weights and recalculate $S_{\mathcal{V}}$ until a decrease is noticed, then continue.
7. Repeat steps 3–6 until the relative change in validation error is below a small percentage or, e.g., $\|\partial S_{\mathcal{V}}/\partial \boldsymbol{\kappa}\|$ is below a small number.

5. OUTLIER DETECTION

Once the network is designed, i.e., we have estimates of the weights, regularization parameters and outlier probability[6], it is possible to devise a method for outlier detection. Suppose we want to decide whether an example $\boldsymbol{x}$ with label $\mathcal{C}_i$ is an outlier or not. Define the binary variable O which is 1 if the example is an outlier, and zero otherwise. The probability that the example is an outlier is given as $p_{\text{outlier}} = p(O = 1|\boldsymbol{x}, \mathcal{C}_i)$. Using Bayes rule, $p_{\text{outlier}} = p(O = 1|\boldsymbol{x}, \mathcal{C}_i) = p(O = 1 \wedge \mathcal{C}_i|\boldsymbol{x})/p(\mathcal{C}_i|\boldsymbol{x})$. The denominator is given by Eq. (2) and the numerator is the posterior probability for $\mathcal{C}_i$ in the case of outliers which is equal to the last addend of Eq. (1). Thus[7],

$$p_{\text{outlier}} = \frac{\beta(1-p_0(\mathcal{C}_i|\boldsymbol{x}))}{p_0(\mathcal{C}_i|\boldsymbol{x})(1-\beta c)+\beta}. \quad (23)$$

The estimated probability that the example is an outlier is consequently, $\widehat{p}_{\text{outlier}} = \widehat{\beta}(1-\widehat{z}_i)/\widehat{y}_i$.

[6] In this contribution we do not include network pruning as an element in the design phase; however, this is easily done, see further [1], [9], [10].

[7] Note $p_{\text{outlier}} = 0$ for $\varepsilon = 0$ and $p_{\text{outlier}} = 1$ for $\varepsilon = 1$.

6. EXPERIMENTS

We first test the performance of the algorithm on an artificial example with $c = 3$ classes in a 2D input space. The class conditional probabilities are $p(\boldsymbol{x}|\mathcal{C}_i) = [\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{I}) + \mathcal{N}(-\boldsymbol{\mu}_i, \boldsymbol{I})]/2$ where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I})$ is a 2D Gaussian distribution with mean vector $\boldsymbol{\mu}$ and identity covariance matrix. The mean vectors are given by $\boldsymbol{\mu}_i = 3\cdot[\cos(\pi(2i-1)/6), \sin(\pi(2i-1)/6)]$, $i = 1, 2, 3$. The prior class probabilities are $p(\mathcal{C}_i) = 1/3$. We generated $N = 300$ data and used $N_t = 150$ for training and $N_v = 150$ for validation. In addition we generated a test set of $N_{\text{test}} = 600$. Further, we introduced outliers by changing class labels with probability $\varepsilon = 0.08$. Suppose that the network weights are given by $\boldsymbol{w} = [\boldsymbol{w}^I, \boldsymbol{w}^I_{\text{bias}}, \boldsymbol{w}^H, \boldsymbol{w}^H_{\text{bias}}]$ where $\boldsymbol{w}^I$, $\boldsymbol{w}^H$ are input-to-hidden and hidden-to-output weights, respectively, and the bias weights are assembled in $\boldsymbol{w}^I_{\text{bias}}$ and $\boldsymbol{w}^H_{\text{bias}}$. In this example, we use the following weight decay regularization term:

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \kappa^I|\boldsymbol{w}^I|^2 + \kappa^I_{\text{bias}}|\boldsymbol{w}^I_{\text{bias}}|^2 + \kappa^H|\boldsymbol{w}^H|^2 + \kappa^H_{\text{bias}}|\boldsymbol{w}^H_{\text{bias}}|^2 \quad (24)$$

where $\boldsymbol{\kappa} = [\kappa^I, \kappa^I_{\text{bias}}, \kappa^H, \kappa^H_{\text{bias}}]$. The simulation set-up was:

- Network: 2 inputs, 5 hidden neurons, 2 outputs.
- Weights were initialized uniformly over $[-0.5, 0.5]$, regularization parameters were initialized at zero. 30 steps in a gradient descent training algorithm (see e.g., [11]) was performed and the weight decays, $\boldsymbol{\kappa}$, were re-initialized at $\lambda_{\max}/10^4$, where $\lambda_{\max}$ is the max. eigenvalue of the Hessian matrix of the cost function. This prevents initial numerical stability problems. ε is initialized at 0.01.
- Training is now done using a Gauss-Newton algorithm (see e.g., [11]). The Hessian is inverted using the Moore-Penrose pseudo inverse (see e.g., [11]) ensuring that the eigenvalue spread[8] is less than 10^8.
- The step-size μ in Eq. (17) is initialized at 1 and $\boldsymbol{\xi}$ is adapted until the validation error has reached a minimum.
- Finally, weights are retrained on the combined set of training and validation data using the optimized weight decay parameters and outlier probability.

Table 1 reports the average and standard deviations of the probability of misclassification (*pmc*) over 10 runs using the optimal $\boldsymbol{\kappa}$ and β. Note that retraining on the full data set decreases the test *pmc* slightly on average; improvement was found in 5 out of 10 runs.

In Fig. 1 a typical run of the $\boldsymbol{\theta}$ adaptation algorithm is shown. We tested the possibility to detect whether specific examples in the data set, e.g., the combined training/validation set, are outliers and the result is summarized in Table 2. This technique can e.g., be applied to manual inspection of examples which are likely to be outliers. This might lead to relabeling or discovery of new interesting features of the problem.

7. CONCLUSIONS

This paper presented a new framework for design of robust neural classifiers by invoking a probabilististc model for outliers. We devised an iterative scheme for simultanenous adaptation of regularization parameters and the outlier probabilty. Moreover, we discussed the possibility of detecting outliers. Numerical examples demonstrated the potential of the framework.

[8] Eigenvalue spread should not be larger than the square root of the machine precision [4].

	Initial Neural Net	Optimal Neural Net	Optimal Bayes Decisions
Train.	0.141 ± 0.010	0.145 ± 0.011	0.160
Val.	0.266 ± 0.029	0.241 ± 0.012	0.240
Test	0.278 ± 0.013	0.250 ± 0.009	0.222
Test after retrain.	0.268 ± 0.008	0.249 ± 0.012	0.222

Table 1: Probability of misclassification, when outlier probability is $\varepsilon = 0.08$. For the neural network the averages and standard deviations over 10 runs are reported. Inital and optimal neural net refers to using initial and optimized setting of $\boldsymbol{\kappa}$ and ε. Optimal Bayes decisions boundaries are calculated from the detailed knowledge of the true posterior probabilities. The minimal Bayes error on an infinite set is 0.213. The outlier probability was estimated as $\widehat{\varepsilon} = 0.097 \pm 0.018$.

	True	
Estimated	*not outlier*	*outlier*
not outlier	$a = 0.911 \pm 0.017$	$b = 0.142 \pm 0.022$
outlier	$c = 0.089 \pm 0.017$	$d = 0.858 \pm 0.022$

Table 2: Confusion matrix for outlier detection (over 10 runs) on the combined training/validation set. An example is considered not to be an outlier if $1 - \widehat{p}_{\text{outlier}} > 0.9$. The aim is, e.g., to set the threshold so that the false positive rate $b/(b+d)$ is acceptable small.

8. REFERENCES

[1] L.N. Andersen, J. Larsen, L.K. Hansen & M Hintz-Madsen: "Adaptive Regularization of Neural Classifiers," in J. Principe *et al.* (eds.) *Proc. IEEE Workshop on Neural Networks for Signal Processing VII*, Piscataway, NJ: IEEE, pp. 24–33, 1997.

[2] C.M. Bishop: *Neural Networks for Pattern Recognition*, Oxford, UK: Oxford University Press, 1995.

[3] J.S. Bridle: "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition," *Neurocomp. - Algor. Arch. and Appl.*, Berlin, Germany: Springer-Verlag, vol. 6, pp. 227–236, 1990.

[4] J.E. Dennis & R.B. Schnabel: *Numerical Methods for Unconstrained Optimization and Non-linear Equations*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[5] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[6] K. Hornik: "Approximation Capabilities of Multilayer Feedforward Networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[7] M. Hintz-Madsen, M. With Pedersen, L.K. Hansen, & J. Larsen: "Design and Evaluation of Neural Classifiers," in S. Usui *et al.* (eds.), *Proc. IEEE Workshop on Neural Networks for Signal Processing VI*, Piscataway, NJ: IEEE, 1996, pp. 223–232.

[8] P.J. Huber: *Robust Statistics*, New York, NY: John Wiley & Sons, 1981.

[9] J. Larsen, L.K. Hansen, C. Svarer & M. Ohlsson: "Design and Regularization of Neural Networks: The Optimal Use of a Validation Set," in S. Usui *et al.* (eds.), *Proc. IEEE Workshop on Neural Networks for Signal Processing VI*, Piscataway, NJ: IEEE, 1996, pp. 62–71.

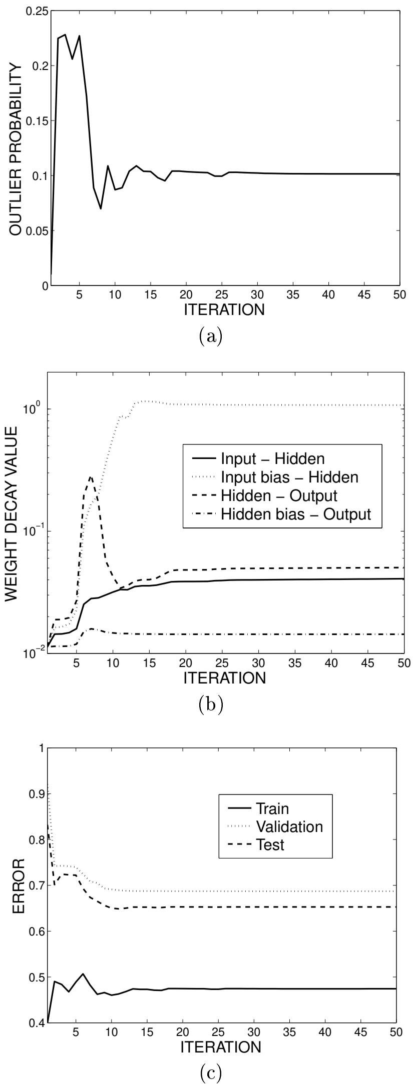

Figure 1: Typical run of the $\boldsymbol{\theta}$ adaptation algorithm. In panel (a) the evolution of the outlier probability $\varepsilon = \beta(c-1)$ is shown. Panel (b) shows the adaptation of the normalized weight decays $\alpha = \kappa \cdot N_t$, and in panel (c) the training, validation and test errors are depicted.

[10] J. Larsen, C. Svarer, L.N. Andersen & L.K. Hansen: "Adaptive Regularization in Neural Network Modeling," appears in G.B. Orr *et al.* (eds.) *"The Book of Tricks"*, Germany: Springer-Verlag, 1997.

[11] G.A.F. Seber & C.J. Wild: *Nonlinear Regression*, New York, New York: John Wiley & Sons, 1989.

Appendix G

Contribution to Neural Networks

This appendix contains a preprint of the paper "Neural Classifier Construction using Regularization, Pruning and Test Error Estimation" [Hintz-Madsen et al., 1998] accepted for publication in Neural Networks.

Neural Classifier Construction Using Regularization, Pruning and Test Error Estimation

Mads Hintz-Madsen, Lars Kai Hansen, Jan Larsen,
Morten With Pedersen, and Michael Larsen
CONNECT, Dept. of Mathematical Modelling, Build. 321,
Technical University of Denmark, DK-2800 Lyngby, Denmark,
Phone: (+45) 4525 3885, Fax: (+45) 4587 2599,
Email: mhm, lkhansen, jl@imm.dtu.dk[1]

[1]This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. Jan Larsen thanks the Radio-Parts Foundation for financial support.

Abstract

In this paper we propose a method for construction of feed-forward neural classifiers based on regularization and adaptive architectures. Using a penalized maximum likelihood scheme, we derive a modified form of the entropic error measure and an algebraic estimate of the test error. In conjunction with Optimal Brain Damage pruning, a test error estimate is used to select the network architecture. The scheme is evaluated on four classification problems.

Keywords: Neural classifiers, Architecture optimization, Regularization, Generalization estimation.

1 INTRODUCTION

Pattern recognition is an important aspect of most scientific fields and indeed the objective of most neural network applications. Some of the classic applications of neural networks like Sejnowski and Rosenbergs "NetTalk" concern classification of patterns into a finite number of categories. In modern approaches to pattern recognition the objective is to produce class probabilities for a given pattern. Using Bayes decision theory, the "hard" classifier selects the class with the highest class probability, hence, minimizing the probability of error. If different costs are associated with the individual classes, a risk-based approach can be adopted (Bishop, 1995),(Ripley, 1996). The conventional approach to pattern recognition is statistical and concerns the modeling of stationary class-conditional probability distributions by a certain set of basis functions, e.g., Parzen windows or Gaussian mixtures (Duda & Hart, 1973),(Bishop, 1995),(Ripley, 1996).

In this paper we define and analyze a system for construction and evaluation of feed-forward neural classifiers based on regularization and adaptive architectures. The proposed scheme is a generalization of the approach we have suggested for time series processing (Svarer, Hansen, & Larsen, 1993; Svarer, Hansen, Larsen, & Rasmussen, 1993) and for binary classification in the context of a medical application (Hintz-Madsen, Hansen, Larsen, Olesen, & Drzewiecki, 1995). The key concept of the new methodology for optimization of neural classifiers is an asymptotic estimate of the test error of the classifier providing an algebraic expression in terms of the training error and a model complexity estimate. Our approach is a penalized maximum likelihood scheme. The likelihood is formulated using a simple stationary noise model of the pattern source. For any given input pattern there can be defined a probability distribution over a fixed finite set of classes. The training set involves simply labeled data, i.e., each input vector is associated with a single class label. The task of the network is to estimate the relative frequencies of class labels for a given pattern. In conjunction with SoftMax normalization of the outputs of a standard, computationally universal, feed-forward network we recover a slightly modified form of the so-called entropic error measure (Bridle, 1990). For a fixed architecture the neural network weights are estimated using a Gauss-Newton scheme (Seber & Wild, 1995), while the model architecture is optimized using Optimal Brain Damage (Cun, Denker, & Solla, 1990). The problem of proper selection of regularization parameters is also briefly discussed, see also (Larsen, L.K. Hansen, Svarer, & Ohlsson, 1996).

While most of the components of our approach have been described in brief conference papers, we have here aimed at a complete account of the computational aspects as well as thorough tests on practical examples.

2 NEURAL CLASSIFIERS

Assume we have a training set, D, consisting of q input-output pairs

$$D = \{(\mathbf{x}^\mu, y^\mu) | \mu = 1, ..., q\} \tag{1}$$

where $\mathbf{x}$ is an input vector consisting of n_I elements and y is the corresponding class label. In this presentation we will assume that the class label is of the definite form $y = 1, ..., n_O$, with n_O being the number of classes. An alternative soft target assignment might be relevant in some practical contexts where the target could be, e.g., an estimate of class probabilities for the given input (Ripley, 1996).

We aim to model the posterior probability distribution

$$p(y = i|\mathbf{x}), \qquad i = 1, \ldots, n_O. \tag{2}$$

In some applications it might be desirable to use a rejection threshold when classifying, that is if all of the posterior probabilities fall below this threshold then no classification decision is made, see e.g., (Duda & Hart, 1973), (Hintz-Madsen et al., 1995).

To represent these distributions we choose the following feed-forward network architecture:

$$h_j(\mathbf{x}^\mu) = \tanh\left(\sum_{k=1}^{n_I} w_{jk} x_k^\mu + w_{j0}\right) \tag{3}$$

$$\phi_i(\mathbf{x}^\mu) = \sum_{j=1}^{n_H} W_{ij} h_j(\mathbf{x}^\mu) + W_{i0} \tag{4}$$

with n_I input units, n_H hidden units, n_O output units, and parameters $\mathbf{u} = (\mathbf{w}, \mathbf{W})$, where w_{j0} and W_{i0} are thresholds. To ensure that the outputs can be interpreted as probabilities, we use the normalized exponential transformation known as SoftMax (Bridle, 1990):

$$\hat{p}(y^\mu = i|\mathbf{x}^\mu) \equiv \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{\sum_{i'=1}^{n_O} \exp[\phi_{i'}(\mathbf{x}^\mu)]} \tag{5}$$

where $\hat{p}(y^\mu = i|\mathbf{x}^\mu)$ is the estimated probability, that $\mathbf{x}^\mu$ belongs to class i. Numerical aspects of equation (5) are discussed in appendix A.

Assuming that the training data are drawn independently, the likelihood of the model can be expressed as

$$P(D|\mathbf{u}) = \prod_{\mu=1}^{q} \prod_{i=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu)^{\delta_{i,y^\mu}} \tag{6}$$

where the Kronecker delta is defined by: $\delta_{i,y^\mu} = 1$ if $i = y^\mu$, otherwise $\delta_{i,y^\mu} = 0$.

Training is based on minimization of the negative normalized log-likelihood

$$E(\mathbf{u}) = -\frac{1}{q} \log P(D|\mathbf{u}) = \frac{1}{q} \sum_{\mu=1}^{q} \epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) \tag{7}$$

where

$$\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) = -\sum_{i=1}^{n_O} \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log \left(\sum_{i'=1}^{n_O} \exp[\phi_{i'}(\mathbf{x}^\mu)] \right) \right]. \tag{8}$$

Numerical aspects of equation (8) are discussed in appendix A.

In order to eliminate overfitting and ensure numerical stability, we augment the cost function by a regularization term, e.g., a simple weight decay, to form a penalized log-likelihood,

$$C(\mathbf{u}) = E(\mathbf{u}) + \frac{1}{2} \mathbf{u}^{\mathrm{T}} \mathbf{R} \mathbf{u} \tag{9}$$

where $\mathbf{R}$ is a positive definite matrix. In this paper we consider a diagonal matrix with elements $2\alpha_j \delta_{j,k}/q$.

The gradient of (7) is

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu) \right] \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j}. \tag{10}$$

See appendix B for details.

The matrix of second derivatives (the Hessian) can be expressed as

$$H_{jk} \equiv \frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k} \approx \frac{1}{q} \sum_{\mu=1}^{q} \sum_{i=1}^{n_O} \sum_{i'=1}^{n_O} \hat{p}(y^\mu = i|\mathbf{x}^\mu) \left[\delta_{i,i'} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu) \right] \frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_k} \frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \tag{11}$$

where we have used a Gauss-Newton like approximation. See appendix B for details.

It is important to notice that the Hessian in (11) is singular everywhere for the SoftMax network. This is due to the redundant output representation (5) which leaves the set of outputs invariant to certain linear transformations of the hidden-to-output weights. The use of regularization, however, ensures that the effects of this symmetry don't interfere with training or evaluation of the network. We are currently working on a modified implementation which explicitly removes the SoftMax redundancy (Andersen, Larsen, Hansen, & Hintz-Madsen, 1997).

Using matrix/vector notation the Gauss-Newton paradigm of updating the weights can now be computed as (Seber & Wild, 1995)

$$\mathbf{u}^{\mathrm{new}} = \mathbf{u} - \eta \left(\mathbf{H} + \mathbf{R} \right)^{-1} \left[\frac{\partial E}{\partial \mathbf{u}} + \mathbf{R} \mathbf{u} \right] \tag{12}$$

where $\mathbf{Ru}$ and $\mathbf{R}$ are the first and second derivatives of the regularization term, respectively, and η is a step-size, that may be used to ensure a decrease in the cost function, e.g., by line search.

The determination of regularization parameters is an issue of ongoing research. A natural approach is to minimize the test error with respect to the regularization parameters. Here one may use an estimate of the test error, as derived in the next section. This technique is demonstrated in section 3.1. Another approach which is based on minimization of a validation error has been implemented in (Larsen et al., 1996).

2.1 Test Error Estimate

One of the main objectives in our approach is to estimate a network model with a high generalization ability. In order to obtain this we need an estimate of the generalization ability of a model. The generalization, or test error, for a given network $\mathbf{u}$ may be defined as

$$E_{\text{test}}(\mathbf{u}) = \int \epsilon(\mathbf{x}, y, \mathbf{u}) P(\mathbf{x}, y)\, d\mathbf{x}dy \tag{13}$$

where $P(\mathbf{x}, y)$ is the true underlying distribution of examples and $\epsilon(\mathbf{x}, y, \mathbf{u})$ is the error on example $(\mathbf{x}, y)$. Since the test error involves an average over all possible examples, it is in general not accessible, but it can be estimated by using additional statistical assumptions, thus giving us the following estimate for the average test error of a network $\mathbf{u}$ estimated on a training set D (Murata, Yoshizawa, & Amari, 1994),

$$\widehat{\langle E_{\text{test}} \rangle} = E_{\text{train}}(\mathbf{u}(D)) + \frac{N_{\text{eff}}}{q} \tag{14}$$

where $E_{\text{train}}(\mathbf{u}(D))$ is the training error of the model. The effective number of parameters is given by $N_{\text{eff}} = \text{Tr}[\mathbf{H}(\mathbf{H}+\mathbf{R})^{-1}]$, where $\mathbf{R}$ is the second derivative of the regularization term.

In brief, the following assumptions enter the derivation of (14):

- Independence of input and error on output.
- Many examples per weight: $N_{\text{eff}}/q \to 0$.
- There exists a network, $\mathbf{u}^*$, that implements the true model.

For a detailed discussion on test error estimates and their assumptions, see, e.g., (Larsen, 1992), (Larsen, 1993). This estimate of the test error averaged over all possible training sets may be used to select the optimal network e.g., among a nested family of pruned networks; hence, be used as a pruning stop criterion similarly to our procedure for evaluation of function approximation networks (Svarer et al., 1993, 1993).

2.2 Pruning with Optimal Brain Damage

In order to reduce and optimize a networks architecture, we recommend to apply a pruning scheme such as *Optimal Brain Damage* (OBD) (Cun et al., 1990). The aim of OBD is to estimate the importance of the weights for the training error and rank the weights according to their importance. If the importance is estimated using a second order expansion of the training error around its minimum, the *saliency* for a weight u_i is (Svarer et al., 1993)

$$s_i = \left(R_{ii} + \frac{1}{2} H_{ii} \right) u_i^2 \tag{15}$$

where the Hessian H_{ii} is given by (11) and R_{ii} is the i'th diagonal element of $\mathbf{R}$.

The following assumptions enter the derivation of OBD:

- The training error is at a minimum.
- The terms of third and higher orders in the deleted weights can be neglected.

- The off-diagonal terms in the Hessian, $\frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k}$, can be neglected (if more than one weight is pruned).

By repeatedly removing weights with the smallest saliencies and retraining the resulting network, a nested family of networks is obtained. Here we may use the previously derived test error estimate to select the "optimal" network.

2.3 Recipe Overview

The algorithm can be summarized by the following:

1. Determine regularization parameters (e.g. by using the grid-sampling technique described below in section 3.1).
2. Train/retrain network using Gauss-Newton optimization.
3. Compute the estimated test error.
4. Compute OBD saliencies and remove a percentage of the weights with the smallest saliency. Goto 2, if # of remaining weights > 0.
5. Select the network with the smallest estimated test error as the optimal network.

2.4 Comments on Algorithm Complexity

When choosing an algorithm for solving a particular problem, it is necessary not only to ensure that the algorithm is theoretically well founded, but it should also be applicable in practice. One limiting factor is the available computational resources. Though a diminishing problem, it still needs consideration. In this section we'll briefly discuss the complexity of the proposed algorithm.

For each training iteration it is necessary to compute the regularized Hessian and its inverse. Computing the Hessian is an $O(q{n_O}^2N^2)$ operation, where q is the number of training examples, n_O the number of classes and N the number of weights, while inverting the regularized Hessian is an $O(N^3)$ operation. Since the number of training examples is usually larger than the number of weights, it is the computation of the Hessian that can be a limiting factor.

As a guiding principle, this algorithm will on a computer with performance equivalent to a Pentium 200 MHz machine produce results in hours when dealing with network configurations of hundreds of weights, while using thousands of weights is less feasible and will take considerable longer.

Note, that after each pruning step with OBD, it is usually only necessary to retrain the network for just a few iterations. Because only weights with small saliencies are removed, the minimum of the cost function is only slightly changed.

3 EXPERIMENTS

The proposed methodology for constructing neural classifiers has been evaluated on several problems: the artificial *contiguity problem*, the real world problems of *glass classification*, *bacteria cell classification*, and *skin lesion classification*.

3.1 The Contiguity Problem

The contiguity problem has been used for evaluating optimization schemes, see e.g., (Denker et al., 1987), (Gorodkin, Hansen, Krogh, Svarer, & Winther, 1993). The boolean input vector (± 1) is interpreted as a one-dimensional image and connected clumps of $+1$'s are counted. Two classes are defined: those with two and three clumps. We consider the case, where $n_I = 10$. In this case there are 792 legal input patterns consisting of 432 patterns with three clumps and 360 with two clumps. We use a randomly

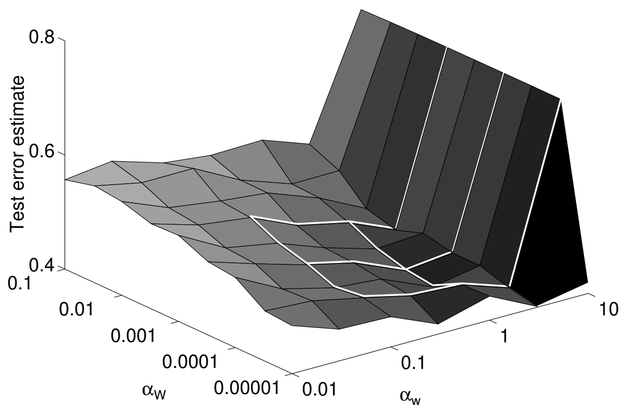

Figure 1: The estimated test error for the *contiguity problem* as function of the weight decay parameters. The grid indicates the points where the estimated test error is sampled.

selected training set with 150 patterns and a test set with 510 patterns both containing an even split of the two classes.

Initially a network architecture consisting of 10 input units, 8 hidden units and 2 output units was chosen. We only employ two different weight decays: α_w for the input-to-hidden weights and α_W for the hidden-to-output weights. By sampling the space spanned by α_w and α_W for non-pruned networks[1] with e.g., a 3×3 grid and computing the estimated test error , it is possible to fit e.g., a diagonal quadratic form[2] in a least-square sense to the sample points, locate the minimum[3] of the quadratic form and use the weight decays found for the design of the network. This is shown in figure 1 and 2. In order to cover a large range of values for the regularization parameters, it's appropriate to use a logarithmic scale for the sampling grid. The values for α_w and α_W should be chosen large enough to ensure numerical stability in equation 12, yet small enough in order not to impose too large a restriction on the number of degrees of freedom. This method is a quick and dirty way to determine the approximate order of magnitude for the regularization parameters and works best when the estimated test error surface is close to being convex. This is typically the case due to the nature of the test error estimate.

Next ten fully connected networks were trained[4] using the estimated weight decay parameters, subsequently pruned using the OBD saliency ranking, removing one weight per iteration. In figure 3 and 4 the distribution of the individual test errors is shown for fully connected networks and pruned networks, respectively. The error distribution shows that the mean error is predominantly driven by a few examples with a high error, thus suggesting that one should monitor the median[5] error as well in order to get a good indication of a network's performance. This problem arises due to the nature of the logarithm in the error function and one should be aware of this property when evaluating the performance.

Seven of the ten pruned networks had a classification[6] error on the test set between 0% and 3.3%, while three networks had an error of $16-19\%$. In (Gorodkin et al., 1993) seven of ten networks had an error of $8-38\%$ using the same size of training set, while three networks had errors around 0%. Compared with these results, our classifier design scheme has a significantly higher yield.

[1] To reduce the computational burden.
[2] Diagonal quadratic form: $(z - z_0) = (x - x_0)^2/a^2 + (y - y_0)^2/b^2$.
[3] In case the minimum is located outside the sample-grid, one should relocate the grid and find a new minimum.
[4] Training was stopped when the 2-norm of the gradient vector was below 10^{-5}.
[5] $E_{median}(\mathbf{u}) = \text{median}\{\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) | \mu = 1, ..., q\}$, where ϵ is the error measure defined by equation (8).
[6] Following Bayes decision theory, the network output with the highest probability determines the class label.

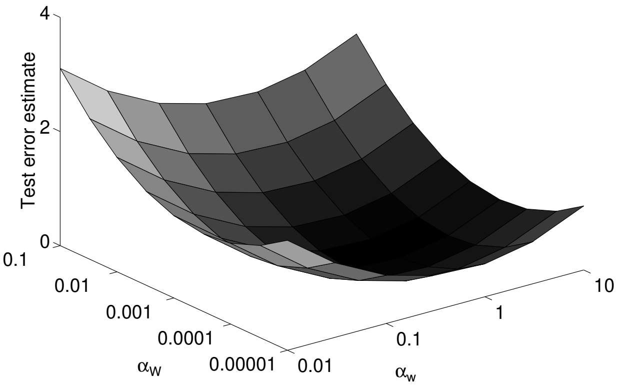

Figure 2: Quadratic form fitted to the 3×3 grid for the *contiguity problem* shown in figure 1. Minimum located at $(\alpha_w, \alpha_W) = (0.68, 2.8 \cdot 10^{-4})$.

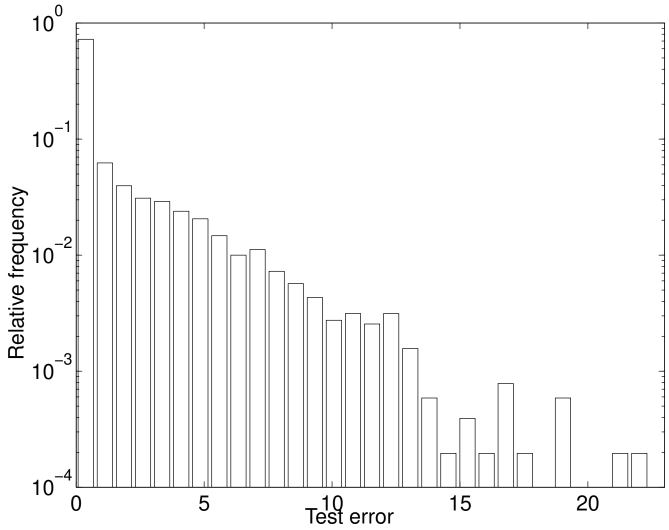

Figure 3: The distribution of the errors of the individual test examples for 10 fully connected *contiguity* networks combined in one pool. Notice the "long tail" of the distribution resulting in a high mean error (0.94) and a small median error (0.022) i.e., the mean is predominantly driven by a few examples with high error.

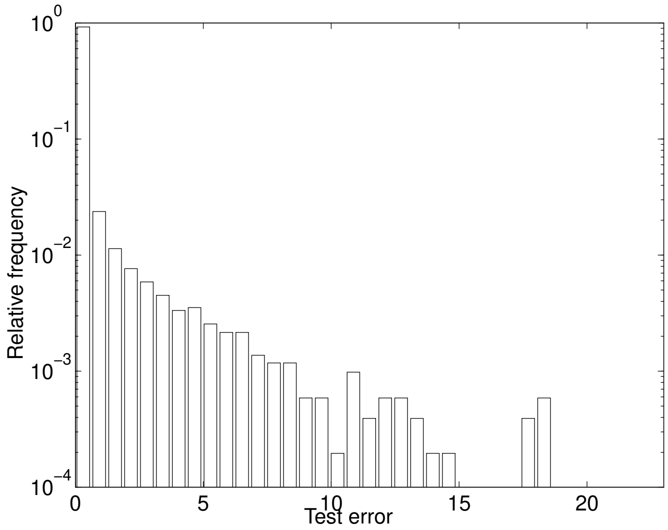

Figure 4: The distribution of the errors of the individual test examples for 10 pruned *contiguity* networks selected by the minimum of the estimated test error combined in one pool. The mean error is 0.37 and the median error is 0.0014 showing a significant performance improvement as result of pruning compared to the fully connected networks in figure 3.

3.2 The Glass Classification Problem

The real world glass classification problem is a part of the Proben1 neural network benchmark collection (Prechelt, 1994). The task is to classify glass splinters into six classes. The glass splinters have been chemical analyzed and nine different measures have been extracted from the analysis, see (Prechelt, 1994) for details. The original dataset (*glass1*) consists of 214 examples divided into a training set (107), a validation set (54) and a test set (53). Since our approach doesn't require a validation set, we have used two different training scenarios: one using the original training set and one using a new training set consisting of the original training and validation set. The initial network architecture chosen consisted of 9 input units, 6 hidden units and 6 output units. We estimated the regularization parameters using the sample-grid technique and the small training set. The parameters were found to be $\alpha_w = 2.2 \cdot 10^{-2}$ and $\alpha_W = 4.7 \cdot 10^{-4}$.

In figure 5-6 and figure 7-8 we show the pruning results of networks trained with the *small* and *large* training set, respectively, using the estimated regularization parameters. The "optimal" network found with the *small* training set had a classification error of 32% on the test set, while the "optimal" network found with the *large* training set had an error of 28%. In (Prechelt, 1994) Prechelt reports a test error of 32% for a fixed network architecture using the *small* training set. The validation set is used to stop training, thus he effectively uses both the training and validation set for training (Sjöberg, 1995). Our approach using the estimated test error for model selection eliminates the need for a validation set, thus allowing us to use more data for the actual training resulting in a better generalization performance. The problem of comparing the performance of neural network models is addressed in (Larsen & Hansen, 1995).

For comparison a standard *k-Nearest-Neighbor*[7] (k-N-N) classification (Duda & Hart, 1973) was performed using the *large* training set. The training error may be computed from the training set by including each training pattern in the majority vote. A *leave-one-out* "validation" error on the training set may be computed by excluding each training pattern from the vote. Finally, the test patterns may be classified by voting among the k nearest neighbors found among the training patterns. Using the *leave-one-out* validation error we found that $k = 2$ was optimal for this data set. The 2-N-N scheme

[7]Within k-N-N a pattern is classified according to a majority vote among its k nearest neighbors using the simple Euclidean metric.

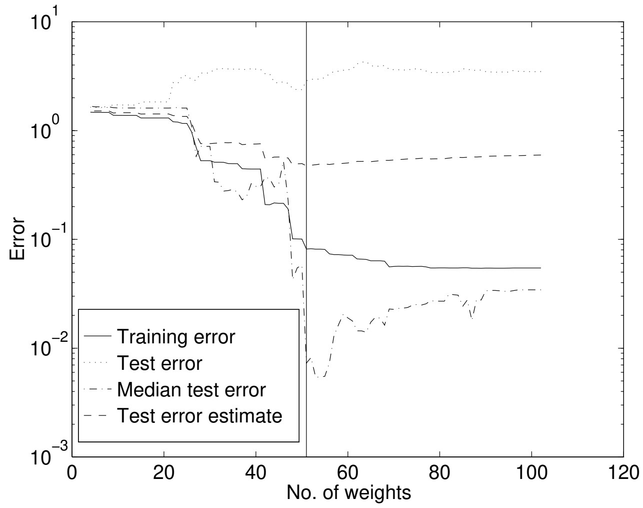

Figure 5: Pruning of a *glass classification* network using the *small* training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error. Note that the test error is very high and evolves quite differently from the classification error on the test set shown in figure 6. The development of the median test error is more similar to the development of the classification error on the test set.

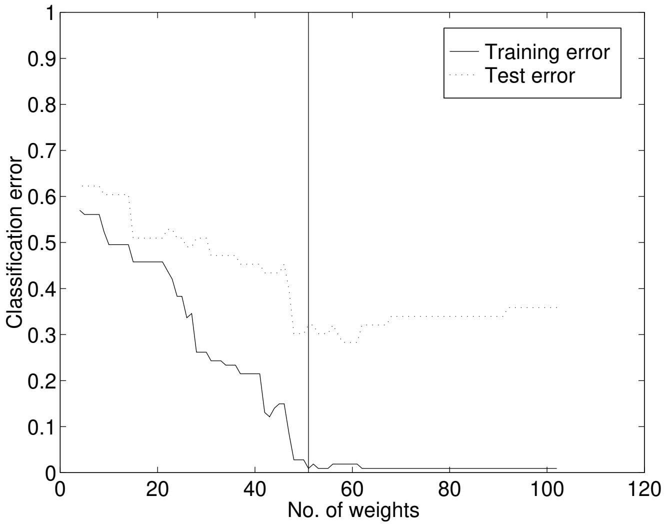

Figure 6: The classification error during pruning of a *glass classification* network using the *small* training set. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error shown in figure 5.

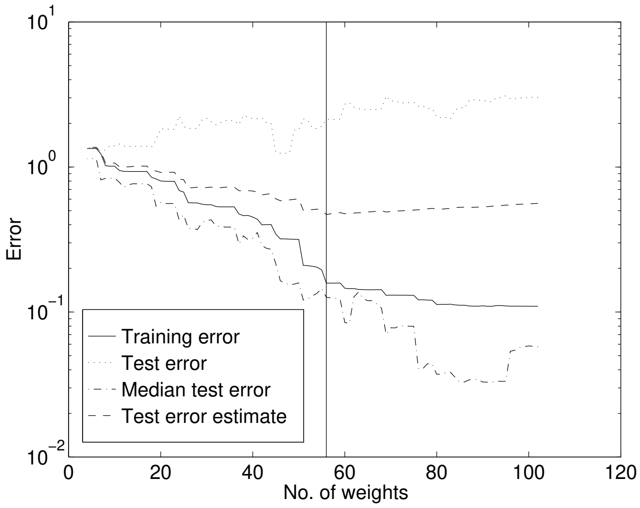

Figure 7: Pruning of a *glass classification* network using the *large* training set. The vertical line indicates the "optimal" network selected by the estimated test error.

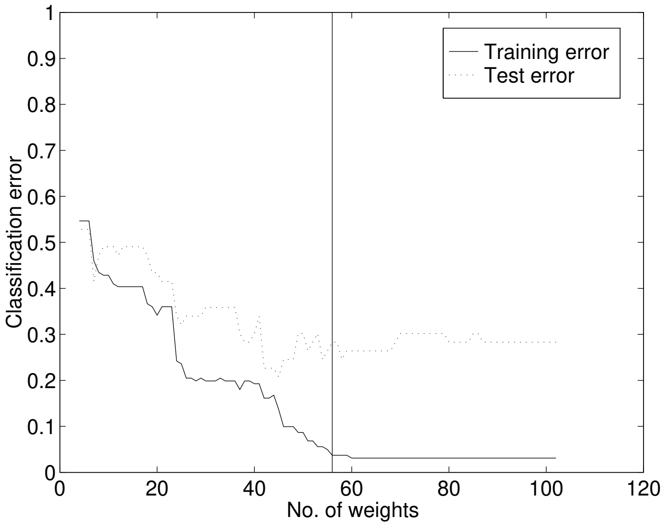

Figure 8: The classification error during pruning of a *glass classification* network using the *large* training set. The vertical line indicates the "optimal" network selected by the estimated test error shown in figure 7. Notice the overall lower classification error on the test set compared to figure 6.

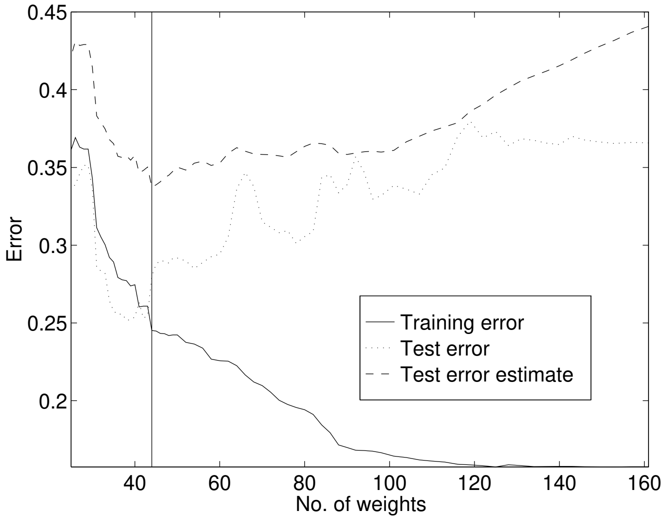

Figure 9: Pruning of a *bacteria cell* network. The vertical line indicates the "optimal" network selected by the minimum of the estimated test error.

had a classification error of 34% on the test set. Thus the performance of the optimized k-N-N scheme cannot match Prechelt's or our networks.

3.3 The Bacteria Cell Classification Problem

In order to evaluate the quality of water in e.g. oceans and lakes, it is desirable to know the type and extent of bacteria cells in water samples. Here we address the problem of classifying cells in microscopic images into five different morphological classes. 398 cells have been detected and their shapes are described by 10 complex *Fourier descriptors* (Granlund, 1972). The dataset is divided into a training set (320) and a test set (78).

Initially a network architecture consisting of 20 input units, 6 hidden units and 5 output units was chosen. The regularization parameters were after preliminary experiments set to $\alpha_w = 0.5$ and $\alpha_W = 0.5$.

In figure 9 a typical pruning scenario for a network is shown. Before pruning this particular network classified 96.9% of the training set and 85.9% of the test set correctly, while the "optimal" pruned network in figure 10 had a classification rate of 92.5% on the training set and 92.3% on the test set. Thus the pruning has increased the generalization ability of the network.

A k-N-N classification was performed for comparison. Using the leave-one-out validation error as described in section 3.2 we found that $k = 3$ was optimal for this data set. The 3-N-N scheme classified 91.0% of the test set correctly. Thus for this data set the performance of the neural and k-N-N classifier is similar.

3.4 The Skin Lesion Classification Problem

The incidence of malignant melanoma, the most lethal of skin cancers, has risen rapidly during the last 50 years. Fortunately patients can be saved from this life-threatening cancer, if it is detected at an early stage. Thus, in recent years, there has been an increased interest in schemes for automatic and early detection of melanoma. Digital imaging may assist and improve the possibility of such early detections. A review of digital imaging in this field was recently published in "Skin Research and Technology" (Stoecker, Moss, Ercal, & Umbaugh, 1995).

From a collection of color photographs of skin tumors at The National University Hospital of Denmark 21 statistical measurements describing color and texture properties have been acquired for each tumor and are used for classification into three groups: Benign nevi (non-cancer), dysplastic nevi (non-cancer, but increased risk of developing cancer) and melanoma.

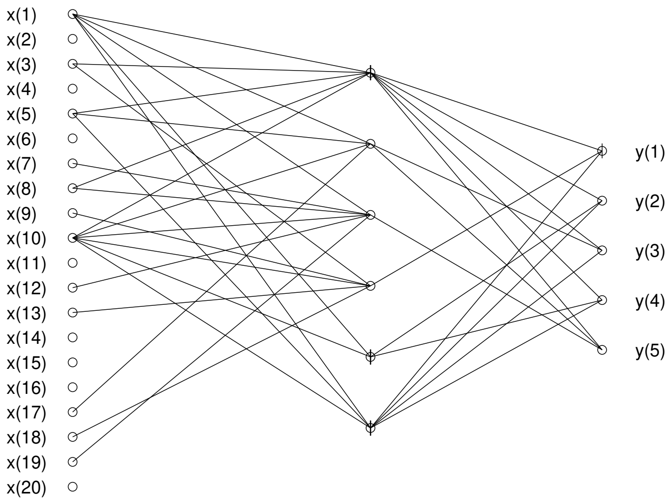

Figure 10: The "optimal" pruned *bacteria cell* network selected by the minimum estimated test error in figure 9. Note that a large number of inputs are not used. A vertical line thorugh a node indicates that a threshold is present.

Table 1: Confusion matrix for the test set using fully connected (103 weights) and pruned networks (41-62 weights). The mean and standard deviation of ten runs are reported. Note that the classifier performs best for the critical melanoma class. † indicates the estimated output classes.

	Fully connected ANN			**Pruned ANN**		
Conf. mat.	Benign nevi	Dyspl. nevi	Melanoma	Benign nevi	Dyspl. nevi	Melanoma
Benign nevi†	42.4±9.1%	29.0±6.2%	22.0±10.1%	47.6±9.3%	27.5±7.6%	18.5±8.8%
Dyspl. nevi†	24.3±5.2%	49.5±6.9%	5.5±3.7%	22.4±6.0%	53.5±5.8%	5.5±4.4%
Melanoma†	33.3±6.7%	21.5±6.7%	72.5±7.6%	30.0±8.4%	19.0±3.9%	76.0±7.8%

A total of 180 images with an even split of the three classes were used for training and 60 images were used for testing. Ten feed-forward networks with an initial architecture consisting of 21 inputs, 4 hidden units and 3 output units were trained and subsequently pruned; hence resulting in ten nested families of pruned networks. The estimate of the test error for each family was used to select the network with the lowest estimated generalization error. The weight decay parameters were after preliminary experiments set to $\alpha_w = 0.1$ and $\alpha_W = 1$.

In figure 11-13 a typical pruning scenario is shown. Table 1 shows the confusion matrix for the test set classified with the fully connected networks and the selected "optimal" networks. The mean and standard deviation of the ten runs are reported. Overall the fully connected networks classified $89.6 \pm 1.6\%$ of the training set and $54.6 \pm 4.1\%$ of the test set correctly, while the results for the pruned networks are $86.7 \pm 3.6\%$ for the training set and $58.9 \pm 2.1\%$ for the test set. Thus the pruning has increased the generalization ability of the networks. An important effect of the pruning approach is the selection of input features, that are salient for the classification; thus providing us with information that can be used in clinical dermatology. Of the ten pruned networks, four didn't use input 15 and three didn't use input 16. Such information can be valuable feedback for the design of future experiments. In the particular case, the two inputs that some networks discard are color variances, suggesting that these do not carry useful information for the classifier. Hence, we might e.g. investigate the color control of the illumination system in order to stabilize color variances.

A k-N-N classification was performed for comparison. Using the validation error as described in section 3.2 we found that $k = 6$ was optimal for this data set. The 6-N-N scheme classified 74.7% of the training set and 57.3% of the test set correctly. Thus the performance of the optimized k-N-N scheme fall in between the pruned and fully connected networks.

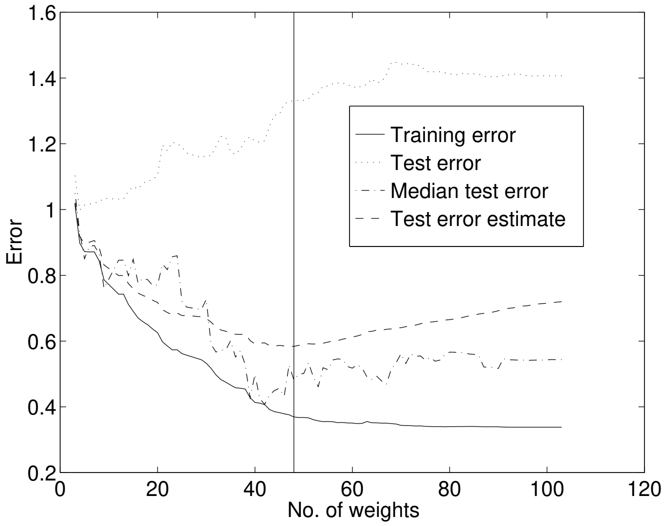

Figure 11: Pruning of a *skin lesion* network. The vertical line indicates the "optimal" network selected by the estimated test error. Notice the big difference between the test error and median test error.

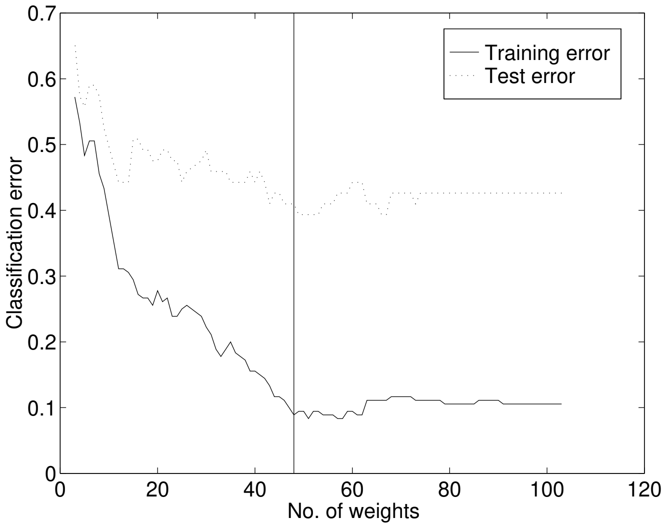

Figure 12: The classification error during pruning of a *skin lesion* network. Note again that the median test error in figure 11 follows the development of the classification test error better than the test error.

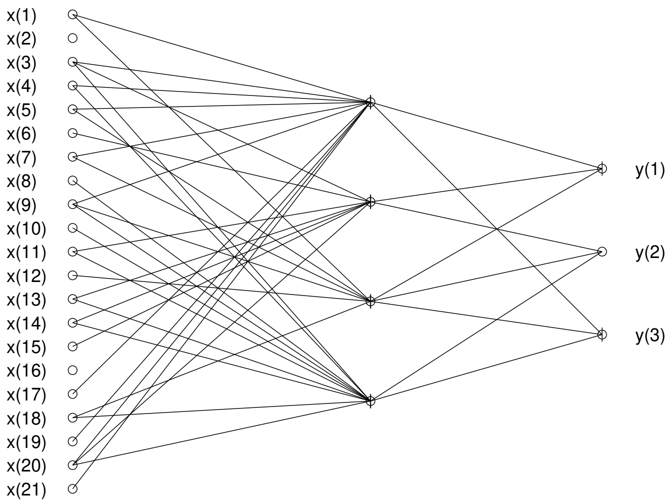

Figure 13: The "optimal" pruned *skin lesion* network (from 103 to 48 weights) selected by the minimum of the estimated test error in figure 11. Note that two inputs are not used.

4 CONCLUSION

We have developed a methodology for construction and evaluation of neural classifiers. Our aim was to present a *practical* approach dealing with the problems of overfitting and model selection without the use of a validation set. The approach was applied to one artificial and three real world problems. It was shown that the test error estimator for classifiers could be used to select optimal networks among families of pruned networks, thus increasing the generalization ability compared to non-pruned networks. Currently, the aim is to establish more empirical data for the validation of the neural classifier construction approach.

A NUMERICAL CONSIDERATIONS

When doing computer simulations, one has to consider the effects of calculations with finite word-length data. Here we rewrite the SoftMax equation (5) and the error measure in equation (8) to prevent overflow caused by the exponential function. The reformulations ensure that the argument to the exponential function is always smaller than or equal to zero. Equation (5) can be rewritten as

$$\begin{aligned}
\hat{p}(y^\mu = i|\mathbf{x}^\mu) &= \frac{\exp[\phi_i(\mathbf{x}^\mu)]}{\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu)]} && (16)\\
&= \frac{\exp[\phi_i(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]}{\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]} && (17)\\
&= \frac{\exp[\phi_{i(\mathbf{x}^\mu)} - \phi_{i_{\max}}(\mathbf{x}^\mu)]}{1 + \sum_{i' \neq i_{\max}} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]} && (18)
\end{aligned}$$

where

$$\phi_{i_{\max}}(\mathbf{x}^\mu) = \max_i \{\phi_i(\mathbf{x}^\mu)\}. \tag{19}$$

Equation (8) is reformulated as

$$\begin{aligned}
\epsilon(\mathbf{x}^\mu, y^\mu, \mathbf{u}) &= -\sum_i \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu)]\right)\right] && (20)\\
&= -\sum_i \delta_{i,y^\mu} \left[\phi_i(\mathbf{x}^\mu) - \log\left(\exp[\phi_{i_{\max}}(\mathbf{x}^\mu)] \sum_{i'} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]\right)\right] && (21)
\end{aligned}$$

$$= -\sum_i \delta_{i,y^\mu}\left[\phi_i(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu) - \log\left(1 + \sum_{i' \neq i_{\max}} \exp[\phi_{i'}(\mathbf{x}^\mu) - \phi_{i_{\max}}(\mathbf{x}^\mu)]\right)\right] \quad (22)$$

where $\phi_{i_{\max}}(\mathbf{x}^\mu)$ is given by (19).

B COMPUTATION OF GRADIENT AND HESSIAN

The gradient of (7) is

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\delta_{i,y^\mu}\left[\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} - \frac{\sum_{i'=1}^{n_O}\left(\exp[\phi_{i'}(\mathbf{x}^\mu)]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}\right)}{\sum_{i''=1}^{n_O}\exp[\phi_{i''}(\mathbf{x}^\mu)]}\right] \quad (23)$$

$$= -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O}\delta_{i,y^\mu}\left[\delta_{i',i}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} - \frac{\exp[\phi_{i'}(\mathbf{x}^\mu)]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j}}{\sum_{i''=1}^{n_O}\exp[\phi_{i''}(\mathbf{x}^\mu)]}\right] \quad (24)$$

$$= -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O}\delta_{i,y^\mu}\left[\delta_{i',i} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_j} \quad (25)$$

where $\delta_{i,y^\mu} = 1$ if $i = y^\mu$, otherwise $\delta_{i,y^\mu} = 0$.

Note that only when the index i equals the correct class for example $\mathbf{x}^\mu$ in (25), is the contribution to the gradient non-zero. This means that equation (25) can be reduced to

$$\frac{\partial E(\mathbf{u})}{\partial u_j} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right]\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (26)$$

The Hessian can be expressed as

$$\frac{\partial^2 E(\mathbf{u})}{\partial u_j \partial u_k} = -\frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\left(\left[\delta_{i,y^\mu} - \hat{p}(y^\mu = i|\mathbf{x}^\mu)\right]\frac{\partial^2 \phi_i(\mathbf{x}^\mu)}{\partial u_j \partial u_k} - \frac{\partial \hat{p}(y^\mu = i|\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j}\right) \quad (27)$$

$$\approx \frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\frac{\partial \hat{p}(y^\mu = i|\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (28)$$

$$= \frac{1}{q}\sum_{\mu=1}^{q}\sum_{i=1}^{n_O}\sum_{i'=1}^{n_O}\hat{p}(y^\mu = i|\mathbf{x}^\mu)\left[\delta_{i,i'} - \hat{p}(y^\mu = i'|\mathbf{x}^\mu)\right]\frac{\partial \phi_{i'}(\mathbf{x}^\mu)}{\partial u_k}\frac{\partial \phi_i(\mathbf{x}^\mu)}{\partial u_j} \quad (29)$$

where we have used the Gauss-Newton approximation (Seber & Wild, 1995). This is motivated by Fisher's argument which is valid when using a log-likelihood cost function.

References

Andersen, L., Larsen, J., Hansen, L., & Hintz-Madsen, M. (1997). Adaptive Regularization of Neural Classifiers. In *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing VII* (pp. 24–33). Piscataway, New Jersey.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press.

Bridle, J. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition. In *Neurocomputing - Algorithms, Architectures and Applications* (Vol. 6, pp. 227–236). Berlin: Springer-Verlag.

Cun, Y. L., Denker, J., & Solla, S. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems, 2*, 598–605.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987). Large Automatic Learning, Rule Extraction and Generalization. *Complex Systems, 1*, 877–922.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: Wiley-Interscience.

Gorodkin, J., Hansen, L., Krogh, A., Svarer, C., & Winther, O. (1993). A Quantitative Study of Pruning by Optimal Brain Damage. *International Journal of Neural Systems*, *4*, 159–169.

Granlund, G. (1972). Fourier Preprocessing for Hand Print Character Recognition. *IEEE Transactions on Computers*, 195–201.

Hintz-Madsen, M., Hansen, L., Larsen, J., Olesen, E., & Drzewiecki, K. (1995). Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification. In *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V* (pp. 484–493). Piscataway, New Jersey.

Larsen, J. (1992). A Generalization Error Estimate for Nonlinear Systems. In *Proceedings of the 1992 IEEE Workshop on Neural Networks for Signal Processing II* (pp. 29–38). Piscataway, New Jersey.

Larsen, J. (1993). *Design of Neural Network Filters.* Unpublished doctoral dissertation, Electronics Institute, Technical University of Denmark.

Larsen, J., & Hansen, L. (1995). Empirical Generalization Assessment of Neural Network Models. In *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V* (pp. 30–39). Piscataway, New Jersey.

Larsen, J., L.K. Hansen, Svarer, C., & Ohlsson, M. (1996). Design and Regularization of Neural Networks: The Optimal Use of A Validation Set. In *Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI* (pp. 62–71). Piscataway, New Jersey.

Murata, N., Yoshizawa, S., & Amari, S. (1994). Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks*, *5*, 865–872.

Prechelt, L. (1994). *PROBEN1 — A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms* (Tech. Rep. No. 21/94). Fakultät für Informatik, Universität Karlsruhe, Germany. (*Available via anonymous ftp* `ftp.ira.uka.de/pub/papers/techreports/1994/1994-21.ps.Z`)

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge: Cambridge University Press.

Seber, G., & Wild, C. (1995). *Nonlinear Regression.* New York, New York: John Wiley & Sons.

Sjöberg, J. (1995). *Non-Linear System Identification with Neural Networks.* Unpublished doctoral dissertation, Department of Electrical Engineering, Linköping University, Sweden.

Stoecker, W., Moss, R., Ercal, F., & Umbaugh, S. (1995). Nondermatoscopic Digital Imaging of Pigmented Lesion. *Skin Research and Technology*, *1*, 7–16.

Svarer, C., Hansen, L., & Larsen, J. (1993). On Design and Evaluation of Tapped-Delay Neural Network Architectures. In *Proceedings of the 1993 IEEE International Conference on Neural Networks* (pp. 46–51). Baltimore.

Svarer, C., Hansen, L., Larsen, J., & Rasmussen, C. (1993). Designer Networks for Time Series Processing. In *Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing III* (pp. 78–97). Piscataway, New Jersey.

Bibliography

[Akaike, 1974] Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

[Amari and Murata, 1993] Amari, S. and Murata, N. (1993). Statistical Theory of Learning Curves under Entropic Loss Criterion. *Neural Computation*, 5:140–153.

[Andersen et al., 1997] Andersen, L., Larsen, J., Hansen, L., and Hintz-Madsen, M. (1997). Adaptive Regularization of Neural Classifiers. In Principe, J., Gile, L., Morgan, N., and Wilson, E., editors, *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing VII*, pages 24–33, New York, New York.

[Argenyi, 1997] Argenyi, Z. (1997). Dermoscopy (Epiluminescence Microscopy) of Pigmented Skin Lesions. *Dermatologic Clinics*, 15(1):79–95.

[Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford.

[Bridle, 1990] Bridle, J. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition. In Fougelman-Soulie, F. and Herault, J., editors, *Neurocomputing - Algorithms, Architectures and Applications*, volume 6, pages 227–236. Springer-Verlag, Berlin.

[Cun et al., 1990] Cun, Y. L., Denker, J., and Solla, S. (1990). Optimal Brain Damage. *Advances in Neural Information Processing Systems*, 2:598–605.

[Dennis and Schnabel, 1983] Dennis, J. and Schnabel, R. (1983). *Numerical Methods for Unconstrained Optimization and Non-linear Equations.* Prentice-Hall, Englewood Cliffs, New Jersey.

[Duane et al., 1987] Duane, S., Kennedy, A., Pendleton, B., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letter B*, 2(195):216–222.

[Duda and Hart, 1973] Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York.

[Efron and Tibshirani, 1986] Efron, B. and Tibshirani, R. (1986). Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1):54–77.

[Efron and Tibshirani, 1993] Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap.* Monographs on Statistics and Applied Probability. Chapman & Hall.

[Fischer et al., 1996] Fischer, S., Schmid, P., and Guillod, J. (1996). Analysis of Skin Lesions with Pigmented Networks. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 323–326.

[Ganster et al., 1995] Ganster, H., Gelautz, M., Pinz, A., Binder, M., Pehamberger, P., Bammer, M., and Krocza, J. (1995). Initial Results of Automated Melanoma Recognition. In Borgefors, G., editor, *Proceedings of The 9th Scandinavian Conference on Image Analysis*, pages 209–218.

[Goutte, 1996] Goutte, C. (1996). On the Use of a Pruning Prior for Neural Networks. In Usui, S., Tohkura, Y., Katagiri, S., and Wilson, E., editors, *Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI*, pages 52–61, New York, New York.

[Goutte and Hansen, 1997] Goutte, C. and Hansen, L. (1997). Regularization with a Pruning Prior. *Neural Networks*, 10(6):1053–1059.

[Goutte and Larsen, 1998] Goutte, C. and Larsen, J. (1998). Optimal Cross-Validation Split Ratio: Experimental Investigation. In Niklasson, L., Bodén, M., and Ziemke, T., editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, volume 2, pages 681–686, Berlin. Springer-Verlag.

[Hansen and Larsen, 1996] Hansen, L. and Larsen, J. (1996). Linear Unlearning for Cross-Validation. *Advances in Computational Mathematics*, 5:269–280.

[Hansen et al., 1997] Hansen, L., Liisberg, C., and Salamon, P. (1997). The Error-reject Tradeoff. *Open Systems & Information Dynamics*, 4:159–184.

[Hassibi et al., 1992] Hassibi, B., Stork, D., and Wolff, G. (1992). Optimal Brain Surgeon and Network Pruning. *Neural Computation*, 4:1–8.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, Massachusetts.

[Hintz-Madsen et al., 1995] Hintz-Madsen, M., Hansen, L., Larsen, J., Olesen, E., and Drzewiecki, K. (1995). Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification. In Girosi, F., Makhoul, J., Manolakos, E., and Wilson, E., editors, *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V*, pages 484–493, New York, New York.

[Hintz-Madsen et al., 1996a] Hintz-Madsen, M., Hansen, L., Larsen, J., Olesen, E., and Drzewiecki, K. (1996a). Detection of Malignant Melanoma using Neural Classifiers. In Bulsari, A., Kallio, S., and Tsaptsinos, D., editors, *Solving Engineering Problems with Neural Networks - Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN'96)*, pages 395–398, Turku, Finland.

[Hintz-Madsen et al., 1998] Hintz-Madsen, M., Hansen, L., Larsen, J., Pedersen, M., and Larsen, M. (1998). Neural Classifier Construction using Regularization, Pruning and Test Error Estimation. *Neural Networks*, In press.

[Hintz-Madsen et al., 1996b] Hintz-Madsen, M., Pedersen, M., Hansen, L., and Larsen, J. (1996b). Design and Evaluation of Neural Classifiers. In Usui, S., Tohkura, Y., Katagiri, S., and Wilson, E., editors, *Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI*, pages 223–232, New York, New York.

[Hoerl and Kennard, 1970] Hoerl, A. and Kennard, R. (1970). Ridge Regression. *Technometrics*, 12:55–82.

[Hotelling, 1933] Hotelling, H. (1933). Analysis of a Complex of Statistical Variables into Principle Components. *Journal of Educational Psychology*, 24:417–441, 498–520.

[Jain, 1989] Jain, A. (1989). *Fundamentals of Digital Image Processing.* Prentice-Hall, New Jersey.

[Karhunen, 1947] Karhunen, H. (1947). Über Lineare Methoden in der Wahrscheinlichkeitsrechnung. *American Academy of Science*, 37:3–17.

[Koh et al., 1989] Koh, H., Lew, R., and Prout, M. (1989). Screening for Melanoma/Skin Cancer. *Journal of American Academy of Dermatology*, 20(2):159–172.

[Larsen, 1993] Larsen, J. (1993). *Design of Neural Network Filters.* PhD thesis, Electronics Institute, Technical University of Denmark.

[Larsen et al., 1998a] Larsen, J., Andersen, L., Hintz-Madsen, M., and Hansen, L. (1998a). Design of Robust Neural Network Classifiers. In *Proceedings of the 1998 International Conference on Acoustics, Speech and Signal Processing*, pages 1205–1208, New York, New York.

[Larsen and Hansen, 1995] Larsen, J. and Hansen, L. (1995). Empirical Generalization Assessment of Neural Network Models. In Girosi, F., Makhoul, J., Manolakos, E., and Wilson, E., editors, *Proceedings of the 1995 IEEE Workshop on Neural Networks for Signal Processing V* , pages 30–39, New York, New York.

[Larsen et al., 1996] Larsen, J., Hansen, L., Svarer, C., and Ohlsson, M. (1996). Design and Regularization of Neural Networks: The Optimal Use of A Validation Set. In Usui, S., Tohkura, Y., Katagiri, S., and Wilson, E., editors, *Proceedings of the 1996 IEEE Workshop on Neural Networks for Signal Processing VI*, pages 62–71, New York, New York.

[Larsen et al., 1998b] Larsen, J., Svarer, C., Andersen, L., and Hansen, L. (1998b). Adaptive Regularization in Neural Network Modeling. In Orr, G., Müller, K., and Caruana, R., editors, *Tricks of the Trade.* Springer-Verlag, Germany.

[Lindelöf and Hedblad, 1994] Lindelöf, B. and Hedblad, M. (1994). Accuracy in the Clinical Diagnosis and Pattern of Malignant Melanoma at a Dermatologic Clinic. *The Journal of Dermatology*, 21(7):461–464.

[Ljung, 1987] Ljung, L. (1987). *System Identification: Theory for the User.* Prentice-Hall, New Jersey.

[Lòeve, 1948] Lòeve, M. (1948). Fonctions Aleatoires de Seconde Ordre. In Levy, P., editor, *Processus Stochastiques et Mouvement Brownien.* Hermann.

[MacKay, 1992a] MacKay, D. (1992a). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472.

[MacKay, 1992b] MacKay, D. (1992b). The Evidence Framework Applied to Classification Networks. *Neural Computation*, 4(5):720–736.

[Mardia et al., 1979] Mardia, K., Kent, J., and Bibby, J. (1979). *Multivariate Analysis.* Academic Press, London.

[Moody, 1992] Moody, J. (1992). The Effective Numbers of Parameters: An Analysis of Generalization and Regularization in Nonlinear Models. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854, San Mateo, California.

[Murata et al., 1991] Murata, N., Yoshizawa, S., and Amari, S. (1991). A Criterion for Determining the Number of Parameters in an Artificial Neural Network Model. In *Artificial Neural Networks*, pages 9–14. Elsevier, Amsterdam.

[Murata et al., 1994] Murata, N., Yoshizawa, S., and Amari, S. (1994). Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transactions on Neural Networks*, 5:865–872.

[Neal, 1994] Neal, R. (1994). *Bayesian Learning for Neural Networks.* PhD thesis, University of Toronto, Canada.

[Østerlind, 1990] Østerlind, A. (1990). *Malignant Melanoma in Denmark.* PhD thesis, Danish Cancer Registry, Institute of Cancer Epidemiology, Denmark .

[Park and Sandberg, 1991] Park, J. and Sandberg, I. (1991). Universal Approximation using Radial-basis-function Networks. *Neural Computation*, 3:246–257.

[Pedersen, 1997] Pedersen, M. (1997). *Optimization of Recurrent Neural Networks for Time Series Modeling.* PhD thesis, Institute of Mathematical Modeling, Technical University of Denmark.

[Pedersen et al., 1996] Pedersen, M., Hansen, L., and Larsen, J. (1996). Pruning with Generalization based Weight Saliencies: γOBD, γOBS. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Proceedings of Advances in Neural Information Processing Systems*, volume 8, pages 521–528, Cambridge, Massachusetts. MIT Press.

[Rassner, 1988] Rassner, G. (1988). Früherkennung des malignen Melanoms der Haut. *Hausartz*, 39:396–401.

[Ridler and Calvard, 1978] Ridler, T. and Calvard, S. (1978). Picture Thresholding using an Iterative Selection Method. *IEEE Transactions on Systems, Man and Cybernetics*, 8(8):630–632.

[Ripley, 1996] Ripley, B. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge.

[Schmid and Fischer, 1997] Schmid, P. and Fischer, S. (1997). Colour Segmentation for the Analysis of Pigmented Skin Lesions. In *Proceedings of the Sixth International Conference on Image Processing and its Applications*, volume 2, pages 688–692.

[Scott and Symons, 1971] Scott, A. and Symons, M. (1971). Clustering Methods based on Likelihood Ratio Criteria. *Biometrics*, 27:387–397.

[Seber and Wild, 1995] Seber, G. and Wild, C. (1995). *Nonlinear Regression.* John Wiley & Sons, New York, New York.

[Skarbek and Koschan, 1994] Skarbek, W. and Koschan, A. (1994). Colour Image Segmentation - A Survey. Technical Report 94-32, Institute for Technical Informatics, Technical University of Berlin, Germany.

[Sonka et al., 1993] Sonka, M., Hlavac, V., and Boyle, R. (1993). *Image Processing, Analysis and Machine Vision.* Chapman & Hall, London.

[Sørensen et al., 1996] Sørensen, P., Nørgård, M., Hansen, L., and Larsen, J. (1996). Cross-Validation with LULOO. In Amari, S., Xu, L., King, I., and Leung, K., editors, *Proceedings of 1996 International Conference on Neural Information Processing*, volume 2, pages 1305–1310, Hong Kong.

[Stanganelli et al., 1995] Stanganelli, I., Burroni, M., Rafanelli, S., and Bucchi, L. (1995). Intraobserver Agreement in Interpretation of Digital Epiluminescence Microscopy. *Journal of the American Academy of Dermatology*, 33(4):584–589.

[Steiner et al., 1993] Steiner, A., Binder, M., Schemper, M., Wolff, K., and Pehamberger, H. (1993). Statistical Evaluation of Epiluminescence Microscopy Criteria for Melanocytic Pigmented Skin Lesions . *Journal of the American Academy of Dermatology* , 29(4):581–588.

[Stolz et al., 1994] Stolz, W., Braun-Falco, O., Bilek, P., Landthaler, M., and Cognetta, A. (1994). *Color Atlas of Dermatoscopy.* Blackwell Science, Oxford, England.

[Stone, 1974] Stone, M. (1974). Cross-validatory Choice and Assesment of Statistical Predictors. *Journal of the Royal Statistical Society*, 36(2):111–147.

[Thodberg, 1993] Thodberg, H. (1993). Ace of Bayes: Application of Neural Networks with Pruning. Technical Report 1132E, The Danish Meat Research Institute, DK-4000, Denmark.

[Toussaint, 1974] Toussaint, G. (1974). Bibliography on Estimation of Misclassification. *IEEE Transactions on Information Theory*, 20(4):472–479.

[Williams, 1995] Williams, P. (1995). Bayesian Regularization and Pruning using a Laplace Prior. *Neural Computation*, 7(1):117–143.

[Wyszecki and Stiles, 1982] Wyszecki, G. and Stiles, W. (1982). *Color Science.* Wiley, New York.

[Young, 1994] Young, G. (1994). Bootstrap: More than a Stab in the Dark? *Statistical Science*, 9(3):382–415.