Visualization and Analysis of 3D Functional Brain Images

Finn Årup Nielsen

29. February 1996

Abstract

The long title of the thesis is: 3D-visualization and neural network analysis of single subject fMRI 3D functional brain images.

A single subject functional magnetic resonance imaging (fMRI) study is analyzed using a preproccessing of singular value decomposition principal component analysis and a feed forward neural network analysis, using the entropic costfunction. The analysis is a classification of brain scans into classes according to an induced paradigm: sequential finger-to-thumb opposition. The analysis is "backwarded" to construct a saliency map giving the neural networks answer to what it finds important in the brain scans to explain the paradigm.

The hidden units of the neural network are given special attention, and are used to reveal non-paradigm structure in the data. An analysis strategy is presented based on second moment statistics.

The analysis results has been 3D visualized. A world has been build in the description language VRML containing Talairach grid lines and labels, where isosurface brain images — anatomical as well as functional — can be put.

Contents

Preface

This report is describing the work done in connection with the "Afgangsprojekt". It was carried with Lars Kai Hansen as supervisor on the Department of Mathematical Modelling, Section for Digital Signal Processing, former Electronics Institute, Technical University of Denmark. The work was carried on between 1. September 1995 and 29. February 1996.

I would to thank the people in the Section for Digital Signal Processing for useful as well as useless discussions. This includes the brain room people of Ulrik Kjems, Niels Mørch and Mads Hintz Madsen, along with Morten With Pedersen and my room mates Bent and Benny Lundsager and Lønstrup Kristensen, respectively. I would also like to thanks the people on the other end of internet: Stephen Strother, Jon Anderson and Nick Lange. Lastly I would like to thank Ian Law, Claus Svarer and my supervisor Lars Kai Hansen.

Lyngby, 29. February 1996 Finn Årup Nielsen

Symbol Table

To denote ordinals the Danish system has been used: Putting a period after the number. For example: "the 5. scan" is "the fifth scan". Bold letters have been used for vectors and matrices: small letters have denote vectors, and capitals denote matrices.

$\sim$:	Asymptotic
$\approx$:	Approximately
$\|\|\|_2$:	Two norm
$\|\|\|_\Sigma^2$:	Mahalanobis distance.
$()_{pq}$:	The element in the p'th row and q'th column.
$\varDelta$:	Forward difference.
∇:	First order derivative.
∇^2:	Second order derivative.
$(12+1)\times(3+1)\times 1$:	Example of a neural network with 12 inputs and 1 input threshold, 3 hidden units plus 1 hidden threshold, and 1 output.
α:	Weight decay *a*daption parameter. Momentum.
η:	Gradient step.
Γ:	Singular value matrix. "True" generalization error.
γ:	Regularization parameter. γ is used in [28] in this context.
Λ:	Eigenvalues for the dispersion matrix; The Wilk's Lambda distribution.
μ:	The (true) mean value.
M:	(My) Eigenvalues for the dispersion matrix.
ν:	The diagonal (pseudo) approximation of the second order derivative for the hidden weights normalized with the number of patterns.
Σ:	Dispersion (multi dimensional variance).
σ:	Variance
ω:	The diagonal (pseudo) approximation of the second order derivative for the output weights normalized with the number of patterns.
B:	The PCA *b*asis or the eigenvectors of the dispersion matrix.
C:	*C*ostfunction, that is the errorfunction with regularization. The SVD-PCA basis (Next letter after B).
C:	The Costfunction (scalar value), that is the the error and the regularization.
D:	*D*esired Output (the target for the supervised learning).
E:	*E*rror (LS or entropy), that is the cost function without regularization.
$\mathcal{F}$:	*F*ourier transformation.

G: *G*eneralization error.

H: *H*idden units value after the activation function. *H*essian, that is the second order derivative.

h: A *h*idden unit.

$_h = 1..n_h$: Index for the *h*idden units.

I: *I*nput to the neural entwork. Not the identity matrix.

i: A *i*nput value to the neural network.

$_i = 1..n_i$: Index for *i*nput to the neural network.

M: Masking matrix. (See also My).

n_g: *N*umber of groups (clusters).

n_I: *N*umber of *i*nput units to the neural network.

n_i: *N*umber of *i*nput units to the neural network.

n_h: *N*umber of *h*idden units in the neural network.

n_o: *N*umber of *o*utput units in the neural network.

n_p: *N*umber of *p*atterns (other words: examples, scanned subjects, samples).

n_{p_g}: *N*umber of *p*atterns within a group (a cluster).

n_u: *N*umber of all weights, that is the number of actual parameters in the neural network.

n_u, eff: *Eff*ective *n*umber of all weights, that is the effective number of parameters in the neural network.

n_v: *N*umber of hidden weights.

n_w: *N*umber of output weights.

O: *O*utput from the neural network (the estimate).

$o = 1..n_o$: Index for the *o*utput of the neural network.

R: *R*egularization.

R: *R*egularization matrix or hessian.

S: The empirical dispersion matrix (S is usually used in this context).

T: The *t*ransposed — or SVD — dispersion matrix. T is the letter after s. The *T*otal dipersion.

u: All weights vectorized, both hidden and output (u is the letter before v and w).

$\hat{\mathbf{u}}$: Estimated weights.

$\mathbf{u}^*$: Optimization optimal weights.

$\mathbf{u}^\diamond$: True weights.

V: Hidden *w*eights, that is the weights between input and hidden units. The size is $n_h \times n_I$ The hidden layer is also the first layer (1. $\rightarrow$ single v).

v: Hidden *w*eights as a vector, that is **V** vectorized.

W: Output *w*eights, that is the weights between hidden and output units: The output layer is also the second layer (2. $\rightarrow$ double v). The *W*ithin dispersion

X: Vo*x*el. x is also usually used as a symbol for the input.

Chapter 1

Introduction

The purpose of the analysis of functional brain scans is to map specific brain areas with specific mental processes — a thought (in broad terms). The purpose with visualization is to make that map directly comprehensible for human beings. *Functional* brain scans mean scans showing the *working* brain, — the thinking brain. *Directly comprehensible* means to present the results in a form that presupposes little and uses few symbols.

Functional brain mapping can be achieved, because the brain has *functional specialization* — by an other word: *segregation* [16]. Mental processes — for example the mental processes controlling the motion of the hand — are not distributed over the entire brain but localized to a specific region.

We might say that brain mapping is asking *where* is *what*. Other questions in relation are how, when and who:

Where	What	How	When	Who
Anatomical	Functional	Physiological	Dynamical	"Subjectical"

The *physiological* question comes into the functional anatomic brain mapping with the physical measurement of the mental activity in the subject. In brain mapping these are with rather a low resolution: We can not see the single brain cell working. The *dynamics* of the brain are concerned with the temporal responses of the brain activity, and the "subjectical" matter is describing the difference between subjects: for example sexes, and laterality.

In functional brain imaging the mental processes — the *what* — are induced with what is usually referred to as paradigms. These paradigms are constructed to affect only a small area in the brain. The mental processes, the paradigms are aiming at, can be put in 3 groups:

- **Motor processes**, where the brain is working as an output unit, sending signals to the muscles. An example of this is the finger tapping paradigms: the fingers of a hand performing simple arpeggio motions.

- **Perceptive processes (or sensory processes),** where the brain is working as an input unit, receiving signal from the sense. Here is an example the saccadic eye movement study.

- **Cognitive processes,** where the brain is neither outputting nor inputting, just thinking.

These are often mixed: For example in a saccadic eye movement study, where the eyes are to move in relation to a blinking light array. Here the brain must first *percepts* the light blinking then *moves* the eyes. An other example is verb generation from a presented noun: Acoustic perception, internal noun generation, and vocal output.

To get the *where* we must have a scanner. The PET (positron emission tomography) scanner has in a long time been used as the main functional image provider. The MR (magnetic resonance) scanners have since the seventies been used as a purely anatomical scanner, but recently it was discovered that it also could be used as a functional scanner: functional magnetic resonance imaging — fMRI.

Neuroscape [61] has summarized the advantage with fMRI over other modalities with:

> fMRI has the advantage of the highest available spatial resolution (near millimeter) of any non-invasive brain imaging method. Of the other imaging methods, (such as PET and MEG) fMRI is generally more available and has the advantage of providing the anatomic image for reference. Experimental design can be very much like traditional behavioral design methodology, i.e. testing several different conditions and the non-invasiveness of the technique safely allows repeated scanning sessions within subjects.

Before the advent of functional brain scanners, mappers of the human brain were confined to investigating patients with a brain damage. The damage could for example be cerebral thrombosis or a direct damage such as the famous case of Phineas Gage, who in connection with explosion during railway work, literally got a hole in his brain. The analysis of the damage is one-way: From the area to the mental process. That is the opposite of what functional analysis of brain scans does: From an induced mental process to (the activation of) brain area. An other difference is that damage is hitting randomly with no control of what area is affected.

A functional study consists of two or more scans: At least one with the paradigm, and one without. These are then "subtracted" to point out the paradigm relevant area. Because of noise an ordinary subtraction is not the optimal. Several other methods have been developed and used. Neural network methods are a new suggestion:

Neural networks are a group of mathematical methods, that have been inspired by the way the brain cells are connected. Compared to normal statistical tools the neural networks are *nonlinear* and they do not test in the same manner. Because of their fitting capabilities they can easily overfit. This has to be avoided by monitoring the generalization and optimizing the architecture.

The two-layer feed-forward network is a general purpose vehicle that has the linear model as a subset. Thus the neural network has potential for being as good as or better than any linear model to the analysis of the functional brain images.

For the visualization of the functional patterns the medical research workers have been using two dimensional slice plots. The rapidly growing capabilities of computer graphics systems masks it now possible to render the functional patterns in its right dimension: 3D. A new standard on the internet is a description language for 3D objects. Aiming the 3D visualization to this language will make functional brain results quickly available — and for a broad audience.

The purpose of brain mapping is general scientific curiosity. Later brain mapping can be used in for example neurosurgery, where individual functional brain maps can guide the neurosurgeon — so-called presurgical mapping. Anatomical brain images already assist the neurosurgeon in the case of tumors [11]. Instead of locking the head with a frame having a reference coordinate system, 3D visualization on a computer together with a digitizing arm helps the "free-hand" surgery: "bringing spatial information right down to the surgical level" [11]. Individual functional brain maps can map important areas for the everyday life of the patient, for example the motor area for the hand and foot together with the language centers. The neurosurgeon should navigate past these areas.

1.1 Overview

The report will focus on a neural network analysis with singular value decomposition / principal component analysis preprocessing for the analysis of 3D fMRI brain images.

The first two chapters will be an introduction into the human brain and the functional magnetic resonance technique.

After this will the theory of functional brain image analysis be presented. The main effort will be laid on principal component preprocessing and the neural network theory, which is treated general — not exclusively for functional brain image analysis. Of new theory will be the saliency map for the entropic errorfunction and a suggestion for the investigation of the representation in the hidden units.

A brief chapter will introduce 3D visualization.

The last two chapters will contain the actual analysis with a discussion of the results: First a toy brain investigating the saliency map, and then an analysis of data from a real fMRI brain scanning session: A single subject performing a sequential finger opposition task.

In the appendix are some derivations kept, and a supplement to the VRML language standard.

Chapter 2

The Human Brain

2.1 Anatomy

2.1.1 The "Big" Brain

The brain consists of many parts, the most conspicuous being the 2 hemispheres, which forms the cortex. The hemispheres can be divided into *lobes*: a *temporal* (sides), *occipital* (back), *parietal* (top), and *frontal*. A brain structure in the cortex sometimes referred to as the limbic lobe surrounds some of the central brain structures and is not visible from the outside.

The lobes themselves can be divided further into the smaller foldings: *gyri* (gyrus). The gyri are split by *fissures* or *sulci* (sulcus). The line that divides the 2 hemispheres is called the *interhemispheric fissure*. An other pronounced line is the *sulcus lateralis* (Sylvian fissure) cutting through between the temporal and frontal lobe. Less identifiable is the *sulcus centralis* (Rolandic fissure or Rolandic sulcus) which divides the frontal and parietal lobe, running from ear to ear via the top of the head. The depth of the sulci is 10–20 mm [62].

In a functional brain scan — PET or fMRI — the interhemispheric fissure and a bit of the sulcus lateralis are usually visible, while other individual sulci are not. In a normal high resolution anatomical MR scan the sulci are visible.

The cortex is folded to get more surface. It is at the surface that the cortex brain cell bodies are especially situated, while the internal parts of cortex carry the connections between the cells. The division of brain cell bodies and their connection causes the cortex to be either *white matter* (connections) or *gray matter* (brain bodies). The blood flow of the gray matter is larger than the white matter.

The cortex is fueled by 3 arteries: the *anterior cerebral artery*, covering the frontal and most of the top of the medial area; the *middle cerebral artery*, covering the largest area especially the lateral surface; lastly the *posterior cerebral artery* covering the back and most of the bottom of the medial area.

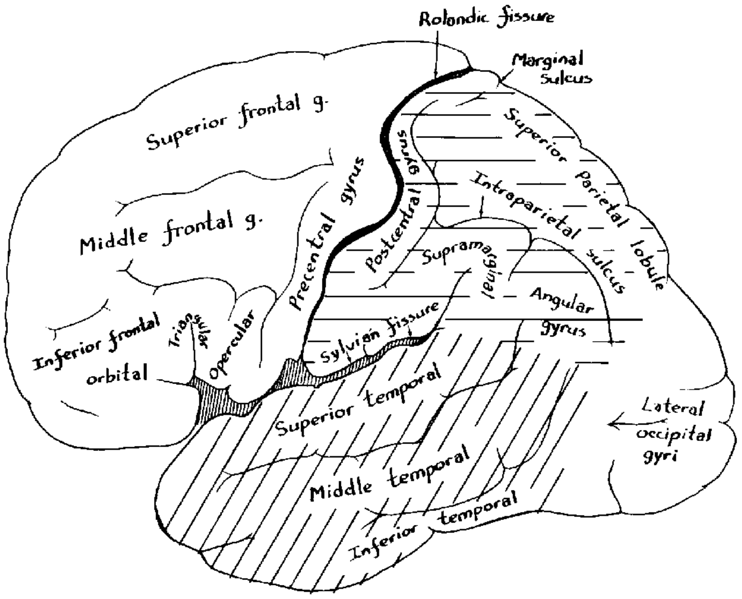


Figure 2.1: Lateral view of the brain anatomy from the left. From [53].

2.1.2 The smaller things

The 2 hemispheres are separated by a structure called the falx cerebri, which begins at the top of the head and continues until the interruption of the *corbus callosum*: the bridge connecting the 2 hemispheres.

The brain is bathed in the *cerebrospinal fluid* — CSF — a nutritive and protective fluid [13], not in direct contact with the blood. The CSF is also occupying the *ventricles*.

The center of the brain contains a lot of other small brain structures, for example the putamen, talamus and amygdala (see figure 2.1).

2.1.3 Orientation

Orientation within the human brain is not easy. The BrainMap program[37] uses the following 5 methods:

- **Bicommissural coordinate space**. Behind these word lay an ordinary coordinate system: The origo is in the middle of the brain, x is positive to the right, y is positive to the front (the face), and z is positive to the top of the head. The offset in the x-direction is easily agreed to be the middle of the interhemispheric fissure: the *midsagital* plane. The offset in the z-direction is defined to be through the *anterior and posterior commissure*. The last offset is through the *anterior commissure*. While the bicommissural coordinate space is objective, it has a problem with the foldings of the human brain: The same coordinate in 2 different brains can for example constitutes 2 different lobes.

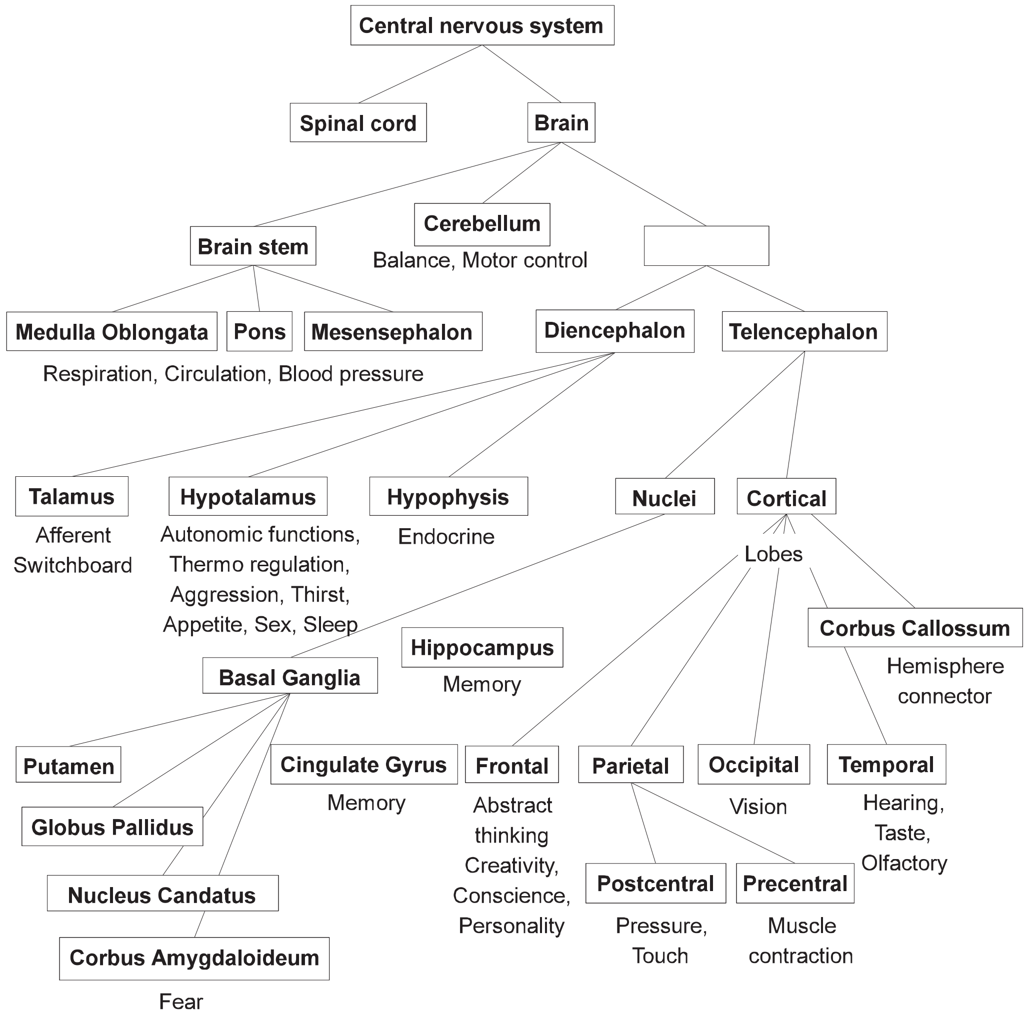


Figure 2.2: Some of the structures in the human brain.

- **Brodmann's areas**. This is a so-called *cytoarchitectural map* of the *cerebral cortex*, that is, areas are identified by the thickness and types of cells within them [72]. Brodmann areas are referred to by numbers from 1 to 47. The Brodmann areas are only defined for the cortex. The appearance of the brain cells is related to their functions, so the Brodmann map is a functional map.

- **Functional area**. Is here the abbreviations especially used in neurobiology: for example V1 meaning the primary visual cortex.

- **Conventional name**. Examples of this is: Broca area, caudate, Wernicke's area, amygdala, and inferior temporal gyrus.

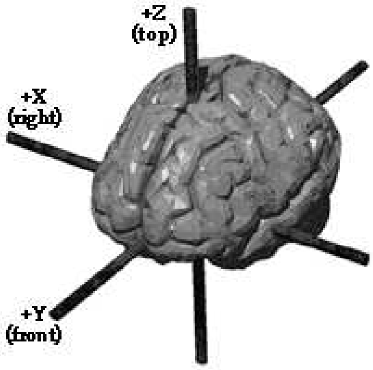


Figure 2.3: The bicommissural brain axes. From [15]

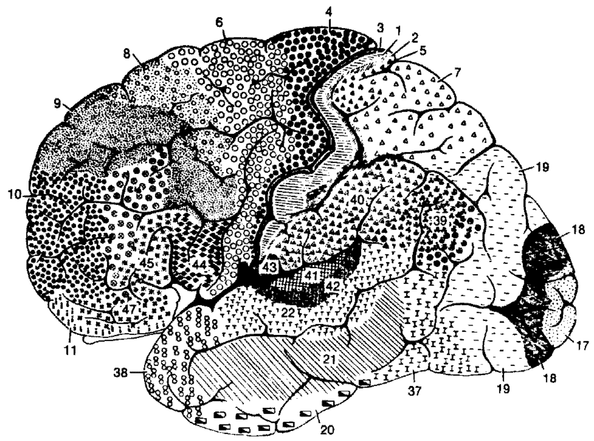


Figure 2.4: Lateral view of the Brodman aras. From [53].

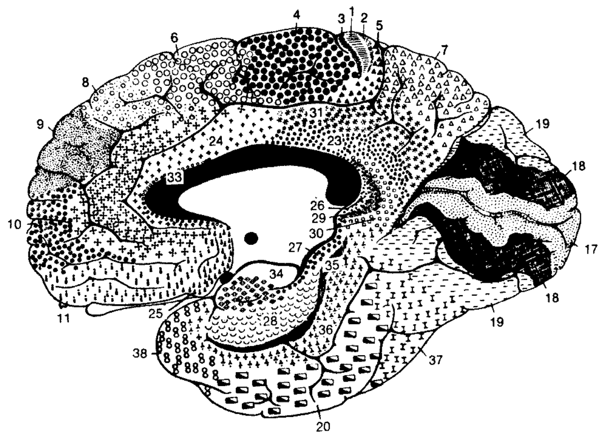


Figure 2.5: Medial view of the Brodman areas. From [53].

- **Lobar outline**. Here is the *cortex* divided according to the lobes: Frontal, limbic, occipital, parietal, temporal, and the cerebellum is along too. These areas are divided further:

  The division of the *Surface* is done by considering a hemisphere as a quarter of a layer cake, laying on a side when it is viewed coronally.

  - **Basal** is the side against the plate, that is the bottom side of the cortex.
  - **Medial** is the raised side, that is the side at the interhemispheric fissure.
  - **Lateral** is the round side with the whipped cream: The side against the cranium.

  Apart from the surface division the lobes are further divided by axis comparable to the axis of the bicommissural system. The location is not so precise.

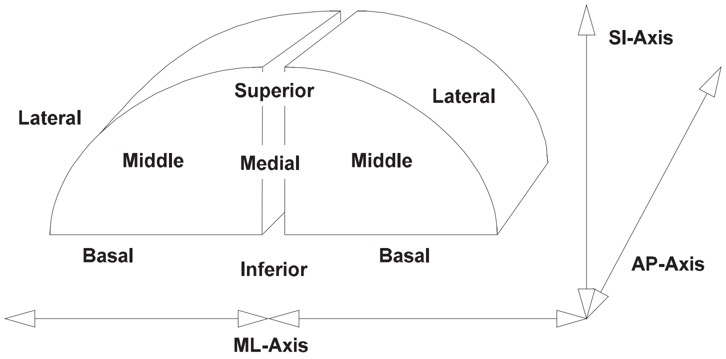


Figure 2.6: The anatomical axis of the brain

The *AP-axis* is the same as the y-axis of the bicommissural coordinate system:

- **Anterior**. Front. Sometimes the prefix "pre-" is used.
- **Middle.**
- **Posterior**. Back. Here is "Post-" used as a prefix, for example postcentral.

The up-down z-axis is called the *SI-axis*:

- **Superior**, meaning the upper.
- **Middle**.
- **Inferior** meaning the lower.

The *ML-axis* is a little different from the bicommissural x-axis, because its reference is one hemisphere and not the whole cortex:

- **Medial**. Medial is referring to the middle of the whole brain, that is at the interhemispheric fissure with the falx cerebri.
- **Middle**. Middle, however, refers to the middle of a hemisphere.
- **Lateral**. Is either the most right or the most left of the whole cortex.

This axis naming is combined with the name of the lobe, when the conventional name for gyri and sulci are formed: For example "inferior temporal gyrus".

Having a brain with no orientation clues, the z-direction of the brain is very easily identified: The top of the brain is round, while the bottom of the brain is bumpy, both in functional and anatomical images. The other directions are harder. If the cerebellum is along the images the back of the head can relatively easily be identified from this. If this is missing the inferior part of the temporal lobe together with the some of the Sylvian fissure can be used. Usually the brain is (laterally) thicker to the back end (posterior) than to the front end. It is very hard to distinguish between the right and the left side of the brain. Transversal slices can be left/right flipped producing a mirrored brain, that cannot directly be discovered.

How to show a 2D brain image

There are different traditions when showing brain images in 2D. First of all we might *project* the brain image unto a plane, or we could just make a *cut*, showing only the image parts in that plane. Two other proposals could be called *view* and *cutview;* the first one showing the whole brain image; the second showing only the brain image from a certain slice and onwards.

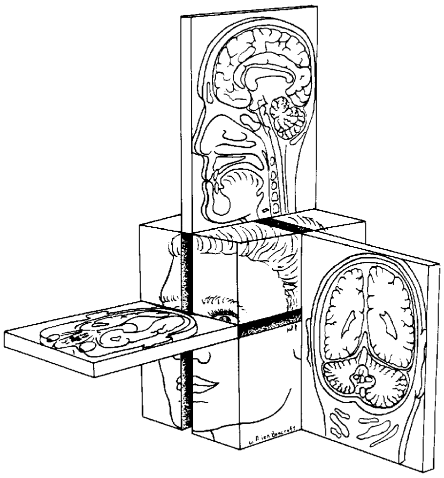


Figure 2.7: Planar slices. From [53].

Orthogonal to the types of view is the question of the viewing direction: These are usually *planar*, meaning planar to the bicommissural coordinates. The names for the planes are: *Sagittal* for a plane with an ear-to-ear normal, *coronal* for a plane with a front to back normal, and *axial* (CT tradition), *horizontal* (neuroanatomical tradition) or *transversal* for the last plane with bottom-to-top normal. The sagittal and coronal always have the top of the head upwards, and the sagittal usually has the noise to the left. The transversal/horizontal plane usually has the eyes upwards. There is less agreement on the rest of the orientations. With the coronal view some people will rather see from the front to the back. With the transversal/horizontal view the argument is about seeing the brain from the top or the bottom.

Atlases

There has been made anatomical atlases of the human brain. One of them is the **Talairach** [76]. This describes the brain in 27 transversal slices. The map is colored according to the brain structures, and especially the inner structures are defined. There is also a grid with the origo defined as the bicommissural origo. The lobes are defined and the Brodmann numbers has been used to clarify them. The cerebellum is not defined.

The gyri and sulci structures are highly varying between subject. A gyrus to gyrus comparison with the Talairach atlas will be hard. The atlas of **Ono** [62] is describing with probabilities the variation in the sulci and gyri.

2.2 Functional Anatomy

Functional anatomy is describing the link between brain area and a cognitive or sensorimotor mental process. The precondition for the functional anatomy is the functional specialization of the brain: the segregation. For example the mental processes, that are controlling the hand, are not using the whole brain but a smaller area in the middle of the AP-axis of the brain.

The anatomic structures (figure 2.1) typically define the borders for the functional anatomic areas. The Brodmann areas also define functional areas.

There is a functional difference between the two hemispheres: The left mainly handles language, calculation, speech, writing and the right mainly handles spatial construction and non-verbal ideation [5]. An other difference is that the right hemisphere contains many long fibers connecting widely separated areas, whereas the left has many short fibers, working in a local area [5].

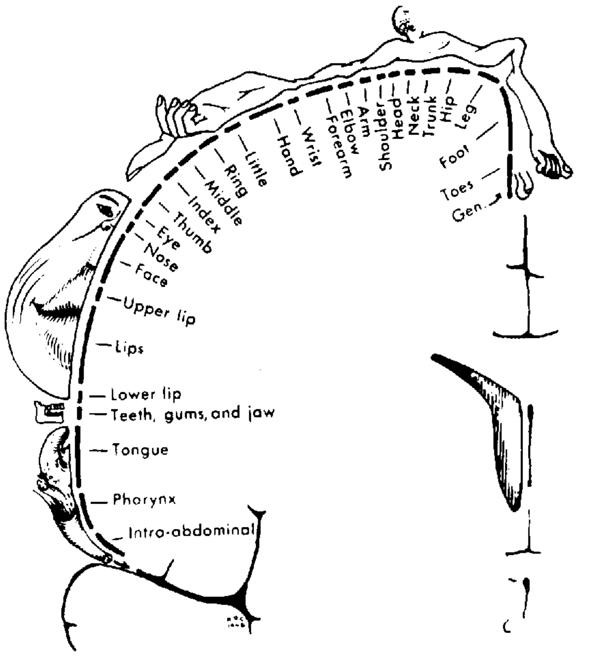


Figure 2.8: The somatosensory Homunculus of Penfield and Rasmussen. From [72].

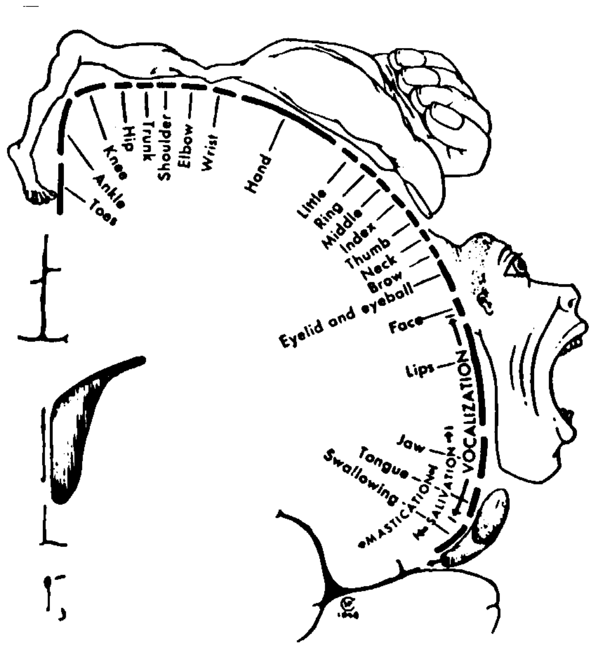


Figure 2.9: The motor Homunculus. From [72].

Two of the most well-defined areas are the primary motor area and primary somatosensory area (heat, cold, touch, pain and sense of body movement [5]) in the precentral and the postcentral gyrus just at the Rolandic sulcus. The body parts have each its own projection onto an area in the gyri (figures 2.8 and 2.9). Other projection areas are the visual and the auditory. The vision projects onto the occipital lobe; the auditory onto the temporal lobe.

The projection areas are connecting further to the association cortex. The frontal association cortex is especially concerned with higher mental processes: planning, problem solving.

The nerves from the body cross, so that the left part of the body receives and transmits to the right hemisphere, and vice versa. Though this is not complete: 90% of the nerve fibers crosses (contralateral), the rest 10% stays on the same side (ipsilateral) [43].

2.3 Physiology

A mental process must depend on a physical phenomenon: the complex pattern the activation potential the brain cells create.

Brain cells receive signals — nerve pulses — through its dendrites. These dendrites are connected to other brain cells, which in turn is connected further on; lastly connecting through afferent nerves to the senses. The brain cells transmit their signal via the axon and its branches: the collaterals. The connection between two cells is called a synapse. It can either be excitatory or inhibitory.

The strength of the signal is not coded as level information but rather time coded: many signals per time unit constitute a large signal.

Between the outside and the inside of a brain cell is there an electrical potential — the *membrane potential* [13] [72]. This is established by the active transport mechanism of a pump — ATPase — pumping Na+ and K+ ion between the outside and the inside of the brain cells; Na+ outnumbering on the outside, and K+ outnumbering on the inside. When enough nerve pulses arrive to a brain cell within a limited time window, the membrane potential is changed a bit, possibly exciting the *threshold potential*. When this happen the conductance of the membrane is suddenly changed, and Na+ ions begin to diffuse into the brain cell. The electrical difference between the outside and the inside of brain cell is thus changed more drastically: From being around –75mV at rest, the electrical difference becomes less negative — *depolarizing*. When the depolarization is at its highest the membrane potential can be +50mV. From that on is the Na+ conductance decreased, while the K+ conductance begins to increase *repolarizing* the membrane, though latter decreases too. The ATPase pump restores the resting membrane potential.

The ATPase pumps are working continuously, and are the main consumers of the energy. When there is a high electrical activity in a brain cell, this requires extra energy. The energy deliverance is in the hands of *metabolism* working with glucose and oxygen: The electrical activity increases *oxygenation* and the *glucose metabolism*. As the blood is transported past the brain cells needing energy, it delivers the hemoglobin attached oxygen, becoming deoxygenated. The normal blood flow cannot supply all the oxygen,

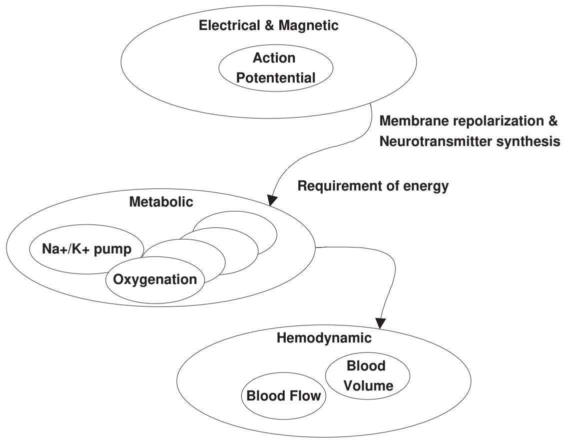


Figure 2.10: The different responses in the brain when activated

so the blood flow is increased. This is done by expanding the cross section of the blood vessels. The mental processes are now spread from the electrical domain — via the metabolism — to the *hemodynamic*.

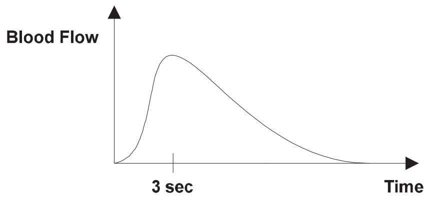


Figure 2.11: The Hemodynamic Response.

The further we go down the chain of figure 2.10 the lesser spatial and temporal resolution we get. The electrical response is in the milliseconds' area: The time at the synapse between a stimulus a the maximum of the postsynaptic signal is around 5 ms.

The metabolism of the oxygenation and glucose consume is relatively fast. The hemodynamic response however is a thousand times slower: The maximum of the cerebral blood flow is around 3 seconds after the stimulus.

Chapter 3

Functional 3D Brain Scanning and Matching

Functional brain scanning breaks in one of the links at the chain of figure 2.10. EEG — electroencephalogram — and MEG — magnetoencephalogram — measures on the very first part of the chain, and thus have the highest temporal resolution. PET (Positron emission tomography) and fMRI breaks in at the lower levels of the chain: The metabolism and the hemodynamics.

Usually, in brain mapping using PET and fMRI, one talks of 4 different physiological parameters of interest that can be measured [15]:

- **CBV — Cerebral Blood Volume.** This method measures the amount of blood in the brain. The unit could be m^3
- **CBF — Cerebral Blood Flow.** This method measures the how much blood gets through the brain. The flow could be measured in $\mathrm{m}^3/\,\mathrm{sec}$. The imaging of macrovascular (large vessel) and microvascular blood flow is different: microvascular blood flow is intravoxel.
- **CMR — Cerebral Metabolism Rate**. This denotes the variety of metabolic parameters one can measure. The one often referred to is the CMRGlu: The glucose metabolism, which is measured in the PET-scanner with the 2-deoxyglucose.
- **OC — Oxygen Consumption**.

Before the abbreviation can be put a prefix "r", for example rCMRGlu. This stands for *regional*. if "rel" is the prefix it denotes *relative*.

A functional brain mapping study consists of a group of scans. Rottenberg [71] defines the following words for the grouping of the scans:

- **Session**. One session is the time a single subject lays in the scanner.
- **Run**. For fMRI the session is divided into runs. Each run usually consists of 50–100 scans, and with around 10 runs per session.

- **Series**. For PET the session is divided into series, with each series consisting of around 10 scans.
- **Scan**. This is the individual image, full volume or slice.

3.1 fMRI

This section will describe magnetic resonance used for imaging and functional imaging. MRI is further described in [3] [1] [9] [19] [35]and [51]. A description of fMRI is contained in [6] [33] and [61].

3.1.1 Magnetic Resonance

Magnetic resonance MR — also called nuclear magnetic resonance NMR — is a physical property in the atomic nucleus: Every nucleus not having an even number of protons or an even number of neutrons has a *spin*. This spin generates a magnetic field, and the atomic nucleus is thus having a little magnetic dipole moment. The spin can due to quantum mechanics only have some specific values. These are for the proton (the hydrogen ^{1}H nucleus) $+\frac{1}{2}$ and $-\frac{1}{2}$. The spins are normally aligned in all different directions, but when introduced in a strong magnetic field, they align themselves parallel and antiparallel with that field. The distribution between the parallel — low energy — and antiparallel — high energy — states is determined by the thermodynamics: the Boltzmann law, with the lower energy states heavily outnumbering the higher.

Under the influence of the strong magnetic field, the nuclei begin to, what within the classical physical could be called precess. The frequency of this precession is called the Larmor frequency,and is dependent upon the strength of the magnetic field and the gyromagnetic ratio of the material. The precessions of the nuclei are not in phase, but rather equally distributed over the cone of the precession.

Having the spins aligned in a strong magnetic field these can be excited by an electromagnetic impulse: The energy of the photon in the impulse (the Planck equation) must be in accordance with the difference in energy between the spin states. Those nuclei hit by the RF pulse will flip to the higher energy states. Because the RF pulse will interact differently with the different phases in the cone of precession, a magnetic field will be created orthogonal to the static field [3].

By the influence of the RF pulse, the ensemble magnetic field from a collection of nuclei, will now have a smaller parallel[1] field (due to the exchange between the lower and higher energy states), and it will have an orthogonal component. Both will decay. The parallel field will decay (or rather restore) back to the thermodynamic equilibrium — with a constant that has been denoted T_1. This is a rather slow decay.

The decay of the orthogonal component is more rapid. It is caused by local differences in the field strength. These can come from inhomogeneities in the static field or the different magnetic properties of the "material" measured (the brain). The constant associated

[1] With respect to the static field

with the static field (we might denote it T_i^*) is dominating over the material constant T_2, so the decay that is immediately observable is mostly a scanner specific constant T_2^*.

Const	Name	Measures	Comes from...
T_1	Spin-lattice	Realignment	Molecular interaction: restoration to thermodynamic equilibrium
T_2	Spin-spin	Lose of phase coherence	Local fields
T_2^*	Observed spin-spin	Nothing(!?)	Static magnetic field inhomogenities

Table 3.1: Decay constants.

As the nuclei dephase, they emit an RF pulse with the decay of T_2^*. This is called the free induction decay — FID. The amplitude of this signal is related to the spin density in the material. To acquire the interesting decay constants — T_1 and T_2 — different pulse sequences must be applied, for example:

- **SR** — saturation recovery — (T_1-weighted) has high resolution of the arteries, veins, and venous sinuses [19]. The SR method continuously interrogates the spins with a 90° pulse, after they have been put in saturation (with respect to the ratio between high and low energy states). The amplitude of the FIDs will be related to the to the decay from saturation: the T_1 constant. The time between two consecutive 90° pulses is called T_R

- **IR** — inversion recovery (T_1-weighted). It has a sharp contrast between the gray and white matter, but a lower spatial resolution than the SR [19]. The IR is similar to the SR method, except that a 180° pulse is used initially.

- **SE** — spin echo — (T_2-weighted). There exist different extensions to this method: CPSE — Carr-Purcell spin echo, and CPMG — Meiboom-Gill modification. To extract the T_2 decay from the T_2^* decay the SE pulse sequence rephases the spin in the middle of a decay. Because the inhomogeneities of the static field is static the decay will then build up in an echo. The amplitude of the echo will decay with the T_2 constant. The time between the rephasing pulses is called T_E

The fact that only odd proton-numbered or neutron-numbered atoms have spins, leaves out the important ^{16}O and ^{12}C, but keeps among clinical relevant nuclei ^{31}P, ^{13}C (1.1% isotope occurrence), ^{23}Na, ^{19}F, and most interesting the already mentioned ^{1}H.

The strong static magnetic field can either be made with resistive, permanent or superconducting magnet. It is only the superconducting magnet that is able to come up in the Tesla area, while the others stay in the "deciTesla" area. To make the coil superconducting helium is used to cool it down. To be economical with the helium, this can be cooled down by nitrogen!

The magnetic field in connection with magnetic resonance seems not to be a safety problem. Weissman [80] mentions that the RF energy deposition can be a problem, especially in "undervascularized, sensitive tissue", such as the lens of the eye. He also draws attention to "ballistic interaction": Ferromagnetic material in the nearby area of the strong field interact very actively!

3.1.2 Magnetic Resonance Imaging

The nuclear signal has a large bandwidth —tens of meters — making it very hard to deduct any directional information using the spatial domain. When spatial information must be obtained in magnetic resonance *imaging*, the direction can be code in the *frequency* of the nuclear signal. As stated before the Larmor frequency is dependent upon the strength of the magnetic field. By changing the *magnetic gradient spatially* spatial information is put in the frequency domain — in the MRI society called "k-space". By Fourier transformation we can come back in the spatial (x-y-z) domain. This gradient method has been called *zeugmatography*. While the order of the large static field still is in the order of Tesla, the strength of the little magnetic gradient field is in the order of milliTesla, for example 0.005 T/m.

The interaction between the gradient magnet and the static causes so much noise in the scanner room, that the subject most wear earplugs. To produce the RF pulses is a bird cage coil typically used.

If the magnetic field is not homogeneous the frequency will immediately be affected, and this will be pipe further on to the spatial domain where the image will be non-linear distorted.

There are 4 different ways of doing the 3-D scanning [35] [51]: point scanning, line scanning, plane imaging and volume imaging. These can be further divided into subgroups, depending on the precise use of the RF pulse and the gradients: changing the gradient field spatial and temporal, injecting wideband — WB — or narrow band —NB — RF pulses, and making them either 90°, 180°, thus they are closely related to the pulse sequence.

The example of point scanning is where the magnet gradient is varied in all 3 coordinates at the same time. The phase and the frequency of the magnetic field are controlled such that only one voxel will be washed out having a stationary field. The frequency of a narrow band RF pulse is aimed at the Larmor frequency in that voxel, triggering and measuring only that voxel. For every voxel this has to be repeated to yield a volume.

An example of plane scanning is: A specific plane is selected with an NB pulse and a gradient in the normal of that plane. A line in that plane is selected by varying the gradient in an other coordinate, and fitting the SE rephasing pulse to frequency in the target line. By varying the gradient in the last coordinate during the SE signal, the position in the line can be determined.

The whole volume imaging can be done by injecting a WB RF pulses and setting the magnetic gradient in a specific direction (projection reconstruction). This is done over and over again while the gradient is changed — in for example spherical coordinates. The final result is a batch of planar projection that can be reconstruction with a sort of Radon

Method	**Technique**
Point scanning	Sensitive point
	Focused nuclear resonance
Line scanning	Selective excitation
	Sensitive Line methods
	Multiple sensitive point
Plane imaging	Projection reconstruction (PR)
	Fourier imaging
	Planar imaging
	Echo-planar imaging (EPI)
	Rotating frame
Volume imaging	Projection reconstruction (PR)
	Fourier imaging
	Echo planar imaging (EPI)

Table 3.2: Some of the different MR imaging techniques. Assembled from [77]

transformation. In the spherical coordinates the space is isotropic: The z-axis has just as high resolution as the other axis. Furthermore the slices can be acquired "skewed" in relation to the x-y-z directions.

3.1.3 Functional Magnetic Resonance Imaging

Magnetic resonance has mainly been used to acquire anatomical images. But recently techniques have been developed to detect the functional image:

- **fMRI-CBV**. The fMRI technique to get the CBV uses a paramagnetic contrast agent (for example Cd-DTPA — gadolinium diethylene-triamine-penta-acetic acid). The injected paramagnetic material remains in the blood — remains intravascular, and cause disturbances in the magnetic field. The T_2 and T_2^* images can be obtained 1 per second.

- **fMRI-CBF**. There exist different techniques to get the blood flow. One of them is to label arterial blood before it enters the brain. Due to the relatively long T_1 decay time the label blood maintains the magnetization. The problem with the existing fMRI-CBF techniques are their long acquisition time. For a slice this is 2–10 seconds [33]

- **BOLD**. This measures a mixture of metabolic (OC) and hemodynamic (CBF) parameters resting on the BOLD effect, and is the most important method for fMRI:

  > [BOLD] fMRI takes advantage of the fact that deoxyhemoglobin is paramagnetic relative to oxyhemoglobin and surrounding brain tissue. Cerebral

> blood flow studies have suggested that a local increase in oxygen delivery beyond metabolic demand occurs in active cerebral tissue which results in a higher concentration of oxygenated blood and a decrease in deoxyhemoglobin in metabolically active brain areas [61].

The change in the susceptibility is in the connection with the iron in the hemoglobin.

The functional part of the fMRI signal is around 1%–5%, and as explained in the chapter 2.3 the metabolic response is quicker than the hemodynamic, so before the overregulation with a *decrease* of deoxygenated blood, there is actually an *increase*. Furthermore show scans of children that they have a reverse BOLD effect [70].

The imaging technique used for fMRI is usually the EPI: echo-planar imaging. This technique brings the acquisition time for a slice down to 50 ms, so that a whole brain image can be acquired within seconds [33].

Because the analysis of functional images works by "subtracting" a non-activation image from an activation image, the T_2^* decay can actually be used for fMRI.

3.2 Matching

Functional studies contain more than just one scan. To compare these they must be matched in accordance with each order. Wedekind [78] has divided the matching in 5 groups 3.1.

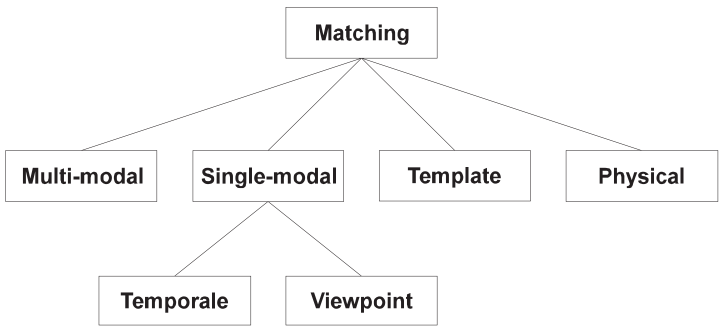


Figure 3.1: Medical Image Matching. From [78].

Multi-modal matching is connecting different modalities, for example functional with structural : PET, fMRI with MRI and CT. **Physical** matching positions the image in relation to the subject. This is needed for stereotactic operations.

With **single-modal viewpoint** matching is meant the compensation for the movement of the head during the scanning session. The motion can be detected with a sensor. Images can also be aligned by comparing them, and performing an affine transformation (table C.2) on each of the images ([44] citing [81]).

Although one adjust for the motion of the head, it can still be the cause to the largest variation in single subject functional studies.

Template matching is where scans from a single subject are matched to a reference, — either an other subject or an atlas. The SPM program (p. 28) contains template matching to the Talairach atlas (p. 16). This is done with a quadratic transformation. The template matching can also be done with an elastic volume: warping [34].

The template matching is less unambiguous than the other matchings because the gyri and lobes may vary from the subject to the template. A way of overcoming the difference in gyral anatomy is through filtering, for example with a gaussian convolution [43].

Chapter 4

Analysis

4.1 Introduction

When a brain is induced with a paradigm the functional scanning image does not just show the paradigm dependent activation. The brain image also shows activation from "idle" mental processes along with noise. It is the analysis job to extract the paradigm image from the activation image, while ignoring the noise and the idle mental processes

The analysis of functional brain scans to extract the functional activation can be divided in 2 groups:

- **Voxel-based methods**, where each voxel is analyzed by itself.
- **Global methods**, where dependencies among voxels bind — that is, help — the analysis. The dependencies need not necessarily be with a spatial prior.

In the first case univariate statistical analysis is to be used: Each voxel is treated as a stochastic variable with no connections to other voxels. After the univariate analysis we end up with a value stated the voxel connections to the paradigm. Each value from a single voxel can be combined with the value from the other voxels in the volume, and this assembled image can then be analyzed, so that spatial priors can be used. This last step is necessary because the voxel-based significance maps are usually too varying between neighboring voxels, due to the noise in the original scanning data. Regions in the assemble image can be characterized by their *critical height* and their *critical size.*

In the case of global methods we have to use multivariate statistics. The multivariate statistics can take spatial dependencies into account, and use this as extra information.

When performing spatial filtering before a voxel-based method — for example with a gaussian filter, one is actually using some global information, saying that neighboring voxels is likely to behave functionally the same.

Some analysis methods can be regarded as having an input and an output. In such a case an other way of dividing the analysis technique is through the direction of the data flow (all though it is just a question of mathematics for a connectionist):

- **Voxel-to-paradigm**, where the "input" to the analysis is the voxel and the "output" is to "predict" the paradigm
- **Paradigm-to-voxel**, where the "input" is the paradigm and the voxel is the *response variable.*

Although what we are after is a functional reference image, the scanning acquisition method, the subject and the situation with the subject differs so much, that every functional study must make its own reference image. Every functional study must have at least 2 image to get one degree of freedom.

4.1.1 Examples

Subtraction

The most simple analysis method would be to subtract — voxel by voxel — a rest reference image from an image that contains the paradigm. This method does not take account for the variance in the voxels [33]. If a voxel in a brain area, independent of the paradigm, is having a large variance, this voxel is likely to be non-zero in the subtraction image, that otherwise should only contain the paradigm.

If the variance of a voxel is estimated from a sample of images, and the subtraction method is still used, we end up with the Student's t test. Contrary to ordinary subtraction we get a statistic significance value — a *p-value.* An example of an extension to simple t test is the SPM:

SPM — Statistical parametric mapping

An example of a paradigm-to-voxel with a combination of voxel-based/global method is the early versions of the program and method **Statistical Parametric Map — SPM** [16]. It uses a subset of multivariate regression analysis — the general linear model, where the input is a so-called design matrix $\mathbf{G}$ which contains the paradigm (as an *interesting* effect) along with *uninteresting* effects — both continuous ("covariates") and discrete ("level"). The analysis targets the voxels[1] $\mathbf{X}$ by adjusting the weight in parameter matrix $\mathbf{B}$:

$$\mathbf{X} = \mathbf{GB} + \mathbf{E} \tag{4.1}$$

This is done to the least square with the error matrix $\mathbf{E}$. The multivariate gaussian second moment statistics are continued by finding the sum of square errors $\mathbf{R}(\Omega)$, using all the parts of the design matrix and its corresponding part of the parameter matrix. Furthermore is the sum of square errors $\mathbf{R}(\Omega_0)$ found using only the uninteresting parts of the design matrix and parameter matrix. If the interesting parts of the design matrix are significant for the paradigm and the uninteresting parts are not, the $\mathbf{R}(\Omega)$ should be "small" and the $\mathbf{R}(\Omega_0)$ should be "large".

[1] The $\mathbf{X}$ matrix is lying — that is size $n_p \times n_x$ down compared to the standard in this report.

The 2 error matrices are however huge and singular, so at this point the early version of SPM switches to univariate statistics, beginning statistical tests using only the diagonal of the error matrices. An F-distributed variable is coming out if the 2 corresponding diagonal elements are subtracted and divided. This leads to the so-called SPM{F} map. A t-distributed variable is popping up if the design matrix is replaced by a *contrast* vector, masking for example activation/non-activation through. This brings the SPM{t} map.

The SPM{t} map — or t-field — is then transformed using a probability integral transform into a gaussian field. With spatial priors this field can be assembled to a SPM{Z} map.

A newer version of SPM [17] transforms the $\mathbf{X}$ matrix from its voxel dimensions down to a lower dimensional signal space (see below). By this transformation the error matrices of $\mathbf{R}(\Omega)$ and $\mathbf{R}(\Omega_0)$ becomes smaller and non-singular, and they can be compared in the Wilk's Λ test. Thus this newer version is a global method, yet still a paradigm-to-voxel method.

Neural Network Regression Analysis

An example of a voxel-based analysis is neural network analysis where scans from an fMRI run is treated as a time series input to a neural network who is targeting at the next scan [64].

Temporal correlation methods

In the case of fMRI the images can be seen as a time series [33] [39]. When the scanning is performed so fast that the hemodynamic has not reach equilibrium just after a shift in activation state, the images would represent a smooth curve. Using this smooth curve as a convolution c^p for the time series of a voxel x_x^p we would end up with a "smooth subtraction image" (In the case were the convolution c^p is just a "hard" on/off switching function in accordance with the paradigm, we end up with a normal subtraction image).

In the time domain this would be:

$$s_x = x_x * c = \sum_p^{n_p} x_x^p c^p \tag{4.2}$$

The analysis could also be perform in the frequency domain

$$S_x = X_x C \tag{4.3}$$

where $S_x = \mathcal{F}(s_x), X_x = \mathcal{F}(x_x)$ and $C = \mathcal{F}(c)$ are the Fourier transformed time series.

Simultaneously fitting the hemodynamic convolution c^p would reveal the dynamics in the brain. Along this road would it be interesting to use a spatial varying convolution c_x^p. This type of analysis has been done by Lange [39].

If the convolution function is a sinus (cosines), it is equivalent to do a Fourier analysis with just one frequency.

It must be remembered that whatever form one gives the convolution functions, this type of analysis remains voxel-based (if not other techniques are not used to included spatial information).

Spatial Correlation Methods

As we can use the temporal dependencies among the scans, we can also use the spatial correlation among the voxels.

With prior knowledge of anatomy the brain can be divided into regions of interest (ROI), thus presuming spatial correlation in the areas.

One method not using anatomical information that could be suggested [21] is to deliberately map the correlation, marching through the voxels, and possibly also involving the time domain.

An other popular method is to use principal component analysis. The subprofile scaling model (SSM) [54] [55] is an extension to the principal component analysis, which scale according to subject and mean for the voxels or region of interest.

The spatial correlation methods do preprocess rather than analyze. To analyze the canonical variable analysis can be applied.

4.2 The Data Space

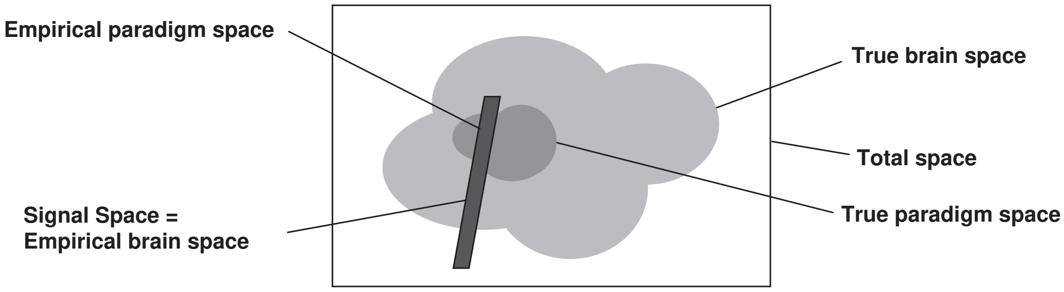


Figure 4.1: The spaces.

Consider the spaces we are working with: We start up with the *total space* which can contain all possible values of every voxel in the scanning volume. The *true brain space* is the space where it is probable to find a brain image: for example it is not likely to have a voxel outside the brain with activation. All images with activation in a voxel outside the brain are not in this space (if we look apart from reconstruction noise and other artifacts).

Within all possible brain images we may find the image that are likely to pop up, when a subject is induced with a paradigm: the *true paradigm space*, the images we are after. This image still spans the total space. It is not 3-dimensional — although we may take the most probable image from the true paradigm space and transform it to a volume.

Within the functional brain study the true brain space is sampled, in what could be called the *empirical brain space*. This is also the **signal space**: The space we obtain information (signal) from. The subject being scanned is performing the task, so ideally the patterns (samples) should lie within the true paradigm space. Uninteresting mental processes (flying thoughts of the scanned subjects) move the patterns "out" of the blurred paradigm space, and noise of all sorts and flavor even moves the patterns "outside" the true brain space. Anyway some of the paradigm space is within the empirical brain space: This could be called the *empirical paradigm space*. Only images within this span are what can be inferenced from a particular study, — without any prior tricks.

The empirical data space — the signal space — can be represented as a matrix: a *datamatrix*. The volume is reshaped to a long vector $\mathbf{x}$. Every scan generates one vector (pattern) $\mathbf{x}^p$. For a study these can be collected in a matrix — the *datamatrix*:

$$\mathbf{X} = \left[\begin{array}{cccc} \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^{n_p} \end{array} \right] = \left[\begin{array}{cccc} x_1^1 & x_1^2 & \cdots & x_1^{n_p} \\ x_2^1 & \ddots & & \\ \vdots & & \ddots & \\ x_{n_x}^1 & \cdots & \cdots & x_{n_x}^{n_p} \end{array} \right] \tag{4.4}$$

The number of voxels in a scan n_x is for fMRI at present moment in the order of some 10'000's, while the number of scan is in the order of some 100's. The direction of the datamatrix is the same as in [59] which is different from the transposed datamatrix others use [52].

4.3 Masking and Normalization

4.3.1 Masking

The voxels in the datamatrix are usually representing a regular grided box: The brain is not a box, so with a whole brain scan some extras come along:

- "Air": Voxels outside the head. There is of course no blood flow here at all, but the ordinary reconstruction will give a low signal here.
- Non-brain anatomy: The eyes are an example of anatomical part that lights up in a MR scan.
- Uninteresting brain: The ventricles are usually of no interest for the analysis.

Voxels representing these extra parts should be masked away for 2 reasons:

- **Reduction** of the datamatrix, taking a burden from the computation,
- **Removal** of, what with a priori knowledge is, totally pure noise which only disturbs the analysis.

The application of the mask is thus a simple transformation from the *total space* to the *true brain space* (figure 4.1).

The determination of the mask can be done in different ways, for example:

- Find the maximum value in the scans, and set a threshold at for example 30 % of that value. All voxel with a value over that are masked through.

- Adjust the brain to an atlas, and use the atlas to decide what is interesting brain and what is not.

To keep track of the mask we might store the index to the used elements in the original scan, or we can apply a non-square masking matrix:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Here are the 1. and 3. voxel masked through, while the 2. voxel is left out.

The mask from the original pattern $\mathbf{x}$ is then easily transformed into the masked pattern $\mathbf{x}^{\odot}$ through an ordinary matrix multiplication:

$$\mathbf{x}^{\odot} = \mathbf{M}\mathbf{x}, \tag{4.6}$$

and the demasking is:

$$\mathbf{x}' = \mathbf{M}^{\top}\mathbf{x}^{\odot} \tag{4.7}$$

The mask matrix $\mathbf{M}$ should of course be sparse to get computation and storage efficiency.

The masking can bring the size of the virtual scan, one is working with, down to a fourth.

4.3.2 Normalization

The scans can contain variations that have no relation to the mental processes. In a PET-study the amount of radioactivity in the blood may vary from session to session. A normalization procedure to take this variation into account should be perform.

4.4 The SVD-PCA/NN analysis

The method I will use is a global method with a voxel-to-paradigm direction, where singular value decomposition and principal component analysis — SVD-PCA — preprocess and a two-layer feed-forward neural network — NN — takes care of the actual analysis. The method was first described in [42] [59].

The method can be seen as a series of transformations: First the voxel space is affine transformed — centered plus rotated — and unimportant dimensions are cut away. Then the resulting space is non-linear transformed by the first layer in the neural network and the activation function in the hidden units. Lastly the space of hidden units is either linear or non-linear transformed into a space, that directly can be compared with the paradigm.

After the information is forwarded from the voxels to the paradigm, the process is reversed to make the image — the saliency map — representing the information the neural network has found important for the paradigm.

The **advantages** of this method could be summarized:

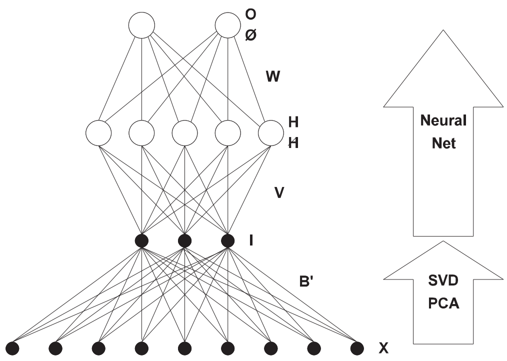


Figure 4.2: PCA and neural network

Apart from the centering of each voxel the SVD *preserves all information.* It is mere a matter of representation. The SVD drastically *reduces the degrees of freedom.* The SVD-PCA favors voxels that behave the same, — voxels with the same kind of *second moment* behavior are put in the same principal component. The new representation is *sorted* according to the second moment behavior. Unfortunately the sorting is not performed according to the paradigm, but rather to the overall activation in the image. The 2-layer neural network with no restriction on the number of hidden units is able to *approximate any continuous function* (A 3-layer neural network is able to approximate any function) [12] [28], that is, the neural network is able to *find non-linear* information in the scanning images.

The **disadvantages** are that, as it is, this analysis technique *does not use any temporal information.* The neural network can — due to its approximation capability — *overfit*, that is, begin to regard the noise as information: The neural network has more degrees of freedom a normal linear model. On the other hand a neural network with a limited number of hidden units is *restricted in its model capabilities.* Finally the value in the saliency map is not a statistical test size: We are not getting "The grand Truth". The ordinary least square linear network has one local minimum, the neural network has many local minima.

4.5 SVD-PCA

Normal principal component analysis finds the direction in which the variation — the spread of data points — is largest, then it finds the orthogonal direction with the second largest variation. This continues until the dimension in the space is used up. The directions describe a new coordinate system, that the data points can be referred to. The PCA is thus a rotation to new coordinates and a sorting with respect to the variance: A shift of basis.[59]

In the case of brain image analysis the dimension of the total space is much larger than the signal space. One scan describes a point the total space. With for example 100 scans the span of the empirical signal space is "only" 100 dimensional. SVD-PCA performs the usual PCA within that little space.

There are no second order dependencies among the principal components, but there can be higher order dependencies. These will be passed on to the neural network, and the neural network can, as a nonlinear modeler, use these dependencies.

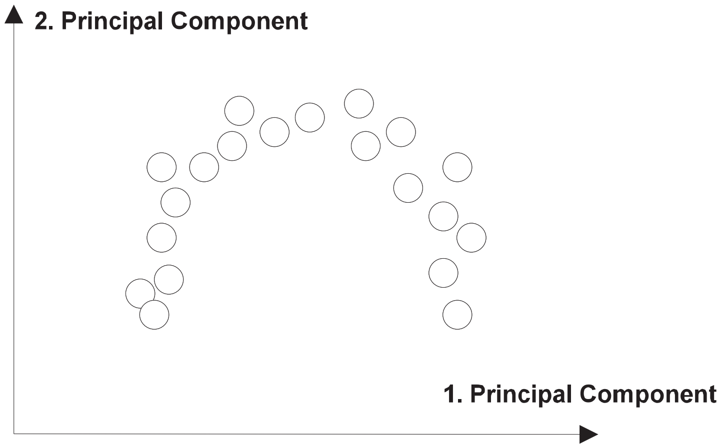


Figure 4.3: Nonlinear data with the principal component: The two principal components are dependent.

There is no data reduction involved in the SVD-PCA, but with the sorting that is included in the PCA, the higher principal components will contain little information, so that the data could be reduced.

The SVD-PCA can be said to produce a distributed representation of the input, compared to a localized: Each input pattern is represented in every component [29].

The SVD-PCA transformation is linear, so to handle data that "is affine" the volume must first be centered:

$$\tilde{\mathbf{x}}^p = \mathbf{x}^p - \bar{\mathbf{x}} \tag{4.8}$$

I will assume that the centering has been done, and call the centered volume $\mathbf{x}$ form now on.

4.5.1 Dispersion matrices

The dispersion matrix[2] measures the covariance — the second moment — among all elements[3]:

$$\mathbf{S} = \mathbf{X}\mathbf{X}^\top \tag{4.9}$$

[2]By some called the covariance matrix

[3]Here the dispersion matrix is non-normalized.

$$\mathbf{T} = \mathbf{X}^\top \mathbf{X} \tag{4.10}$$

The outer product $\mathbf{S}$ and the inner product $\mathbf{T}$ dispersion matrices measure the correlation between the voxels and the correlation between the scans respectively.

The $\mathbf{S}$ matrix is a huge $n_x \times n_x$ matrix. The $\mathbf{T}$ is the much smaller $n_p \times n_p$ matrix.

4.5.2 Eigenvectors and values

From linear algebra comes the fact that a semi-positive definite symmetric matrix can be split up in a eigenvalue and eigenvector — or basis. That is for the inner and outer product dispersion matrices:

$$\mathbf{S} = \mathbf{B}\mathbf{\Lambda}\mathbf{B}^\top \tag{4.11}$$

$$\mathbf{T} = \mathbf{C}\mathbf{M}\mathbf{C}^\top \tag{4.12}$$

Both dispersion matrices do not have full rank: The number of eigenvalues different from zero will only be the smallest number of the two n_x and n_p minus. That is in this case $n_p - 1$.

If the dispersion matrices are viewed as hyperellipsoides the eigenvalue problem here means: To find scale factors — the eigenvalues — and a rotation — the eigenvectors — to a unit hyperellipsoide (hypersphere) so it becomes the dispersion matrix.

The graphical interpretation for the definiteness of a dispersion matrix is the following: positive definiteness means that the dispersion describes a hyperellipsoide in all dimensions. If the dispersion matrix is null-definite in some direction the hyperellipsoide will be collapsed in that direction.

4.5.3 SVD

What we are after is a projection for the voxels, not the scans. So we will need the eigenvalues and eigenvectors of the $\mathbf{S}$ matrix. A straight direct solution to this is a huge overdoing: The $\mathbf{S}$ dispersion matrix describes a small dimensional hyperellipsoide in a huge dimensional space. There will only be $n_p - 1$ axis's lengths — eigenvalues — which is different from zero.

Singular value decomposition is a technique that uses the small dimensional hyperplane instead of the whole space. The singular value decomposition theorem states that a matrix can be transformed to 2 rotations and a matrix $\mathbf{\Gamma}$ that represents the length of the hyperellipsoide axis.

$$\mathbf{X} = \mathbf{B}\mathbf{\Gamma}\mathbf{C}^\top \tag{4.13}$$

The $\mathbf{B}$ and $\mathbf{C}$ are the same as in 4.11 because of the following connections:

$$\begin{aligned} \mathbf{S} &= \mathbf{X}\mathbf{X}^\top = \mathbf{B}\mathbf{\Gamma}^2\mathbf{B}^\top \\ \mathbf{T} &= \mathbf{X}^\top\mathbf{X} = \mathbf{C}\mathbf{\Gamma}^2\mathbf{C}^\top \end{aligned} \tag{4.14}$$

Combining these equations the "big" eigenvalue $\mathbf{\Lambda}$ and eigenvector $\mathbf{B}$ can be found through the "little" eigenvalue $\mathbf{M}$ and eigenvector $\mathbf{C}$:

$$\mathbf{\Lambda} = \mathbf{M} \tag{4.15}$$

$$\mathbf{B} = \mathbf{X}\mathbf{C}\mathbf{M}^{-\frac{1}{2}} \tag{4.16}$$

Each of the eigenvectors in $\mathbf{B}$ is an *eigenpicture* showing a map of where the principal component gets its information from.

4.5.4 Projections

To pipe the signal further on to the neural network the, the rotation must be applied[4]:

$$\mathbf{I} = \mathbf{B}^{\top}\mathbf{X} \tag{4.17}$$

When projecting the data used in the calculation of the transformation one does not have to involve the large volume matrix $\mathbf{X}$. An expression involving smaller matrices can be derived (A.4):

$$\mathbf{I} = \mathbf{M}^{\frac{1}{2}}\mathbf{C}^{\top} \tag{4.18}$$

4.5.5 Projection test

Those volumes $\mathbf{x}^p$ not involved in the calculation of the SVD–PCA transformation is not on the SVD-hyperplane. These scans — these points in the "total space" — will be projected on a point in the hyperplane. The difference from the scan point and the hyperplane can be represented with a plane-orthogonal component $\mathbf{x}^{\perp}$, and the difference between the origo and the projection point can be represented by a plane-parallel component $\mathbf{x}^{\|}$. The original point can now be represented by the 2 *katetes*[5]:

$$\mathbf{x} = \mathbf{x}^{\|} + \mathbf{x}^{\perp} \tag{4.19}$$

If one wants to find out the size of the orthogonal component q_p, this can by done be *including* it in the datamatrix [59]:

$$q_p = \left\|\mathbf{x}_p^{\perp}\right\|_2^2 = \frac{1}{\left((\mathbf{X}^{\top}\mathbf{X})^{-1}\right)_{pp}} \tag{4.20}$$

The angle between the hypotenuse to the original scanning point and the katete to the projection point is called the *angle of erection*:

$$\sin^2\left(\phi_{pp}\right) = \frac{1}{\left((\mathbf{X}^{\top}\mathbf{X})_{pp}\,(\mathbf{X}^{\top}\mathbf{X})_{pp}^{-1}\right)} \tag{4.21}$$

[4] $\mathbf{I}$ is used not as the identity matrix but as the neural network input.

[5] The smaller sides in the right-angled triangle,

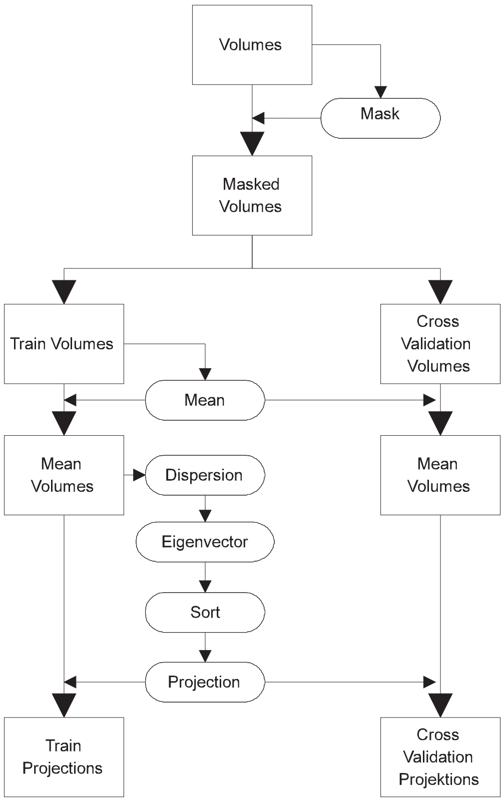


Figure 4.4: PCA flow

4.5.6 Partitioning of the Variation within a Principal Component

The variation in a principal component can be partitioned in for example intersubject and interrun variation ([42] citing [73]). It is done as an univariate technique keeping the principal component separate, although it can be done in parallel.

The total variation in the principal components should be:

$$\sigma^2_{total} = \sum_p^{n_p} \left(\mathbf{i}^p - \bar{\mathbf{i}}\right)^2 \tag{4.22}$$

This is indeed just the eigenvalues.

Imaging that a study consisted of a number of runs n_r, each with an on/off activation $n_a = 2$, the total variation could be partitioned into:

$$\sigma^2_{run} = \sum_r^{n_r} \left(\bar{\mathbf{i}}^r - \bar{\bar{\mathbf{i}}}\right)^2 \tag{4.23}$$

$$\sigma^2_{activation} = \sum_a^{n_a} \left(\bar{\mathbf{i}}^a - \bar{\bar{\mathbf{i}}}\right)^2 \tag{4.24}$$

If the partitioning sets are "orthogonal" ($n_r \times n_a = n_p$), the variation can be summed up:

$$\sigma^2_{total} = \sigma^2_{run} + \sigma^2_{activation} + \sigma^2_{residual} \tag{4.25}$$

4.6 Neural Network

4.6.1 Neural Network Function

I will work with two-layers[6] feed-forward neural networks exclusively [75]. In the spirit of Occam Razor this is the natural choice. As stated before these types of networks approximate any continuous function (though it may need infinitely many hidden units). At the same type they are relatively easy to optimize.

The precondition for the neural networks performances is the hidden units non-linear *activation functions*. The hyperbolic tangents function is a good choice for that: Around zero it has an approximately linear behavior, and in its extremities a binary behavior.

The neural network function, that describes the output as a function of the weight and the input, is for the linear output neural network this:

$$\mathbf{O} = \mathbf{W} \tanh\left(\mathbf{VI}\right). \tag{4.26}$$

Here is $\mathbf{I}$ the symbol for the input not for the identity matrix. It contains n_p number of patterns stored as vectors with the size of n_i. The inputs are send through the first layer of weights: the hidden weights $\mathbf{V}$ by an ordinary matrix multiplication. The hyperbolic tangent is applied for each single element in the resulting matrix with the size of $n_h \times n_i$. Lastly the second layer weights: the output weights $\mathbf{W}$ is used to get the output $\mathbf{O}$.

In an other notation the neural network function is:

$$o^p_o = \sum_{h=1}^{n_h} w_{oh} \tanh\left(\sum_{i=1}^{n_i} v_{hi} i^p_i + v_{h0}\right) + w_{o0} \tag{4.27}$$

The v_{h0} and w_{o0} are the bias. The function of the bias in a digital sense is to imitate *and* gates — not just *or* gates (for binary type of input).

For classification purposes putting a hyperbolic tangent on the output will prove to be a good choice:

$$\mathbf{O} = \tanh\left(\mathbf{W} \tanh\left(\mathbf{VI}\right)\right) \tag{4.28}$$

and with non-matrix notation the function is:

$$o^p_o = \tanh\left(\sum_{h=1}^{n_h} w_{oh} \tanh\left(\sum_{i=1}^{n_i} v_{hi} i^p + v_{h0}\right) + w_{o0}\right) \tag{4.29}$$

[6] "Two-layers" mean 2 layer of weights.

4.6.2 Neural Network Errorfunction

The function associated with the supervised learning of the neural network can be of any form, as long as it minimizes the error between the desired output — by others called the target — and the actual output from the neural network:

$$e_o^p = d_o^p - o_o^p \tag{4.30}$$

Two usually considered functions are the square error:

$$E_{sq} = \sum_{p=1}^{n_p} \sum_{o=1}^{n_o} (d_o^p - o_o^p)^2 \tag{4.31}$$

and the so-called entropic function [28] [50]:

$$E_{entr} = \sum_{p=1}^{n_p} \sum_{o=1}^{n_o} \left[\frac{1}{2}(1 + d_o^p) \ln \frac{1 + d_o^p}{1 + o_o^p} + \frac{1}{2}(1 - d_o^p) \ln \frac{1 - d_o^p}{1 - o_o^p} \right] \tag{4.32}$$

The square error measure is used in the connection with the linear output type of network for continuous valued output. The costfunction assumes additive gaussian noise. Targeting with the square function is also known as *least square*.

The entropic error is used with the hyperbolic tangent as the output activation function, and is used in connection with classification.

Sometimes the errorfunctions are normalized for example by the number of patterns n_p yielding the arithmetic mean, and perhaps also by the variance of the desired output to get a consistent standard second moment measurement for the performance of the network. This of course not standardizing for higher moments.

The two presented errorfunctions both sum over the patterns and the outputs:

$$E = \sum_p \sum_o [\ldots] \tag{4.33}$$

getting a handy additive separable (in p and o) expression. This is not the one and only way. Instead of using a multiply output neural network, multiply separable network could be used. The question whether one should use one single large or many small networks rather depend upon the statistic dependencies among the signals on the outputs: If there are large dependencies the network might take advantage of common hidden weights helping with decreasing the number of parameters the training has to adjust: If there are no dependencies the common hidden weights will just be a constraint.

A second moment method for including the dependencies would be to estimate the covariance matrix for the output signals, and then use this information when computing the costfunction.

Instead of the usual errorfunction (in matrix notation):

$$E = \sum_p \mathbf{e}_p^\top \mathbf{e}_p \tag{4.34}$$

the errorfunction would look like this:

$$E = \sum_p \mathbf{e}_p^\top \mathbf{\Sigma} \mathbf{e}_p \tag{4.35}$$

where $\mathbf{\Sigma}$ is the covariance matrix, or correlation matrix — normalization does not matter.

This was the $\sum_o$ of the errorfunction. The summation $\sum_p$ brings us to robust statistics:

Robust Statistics

A problem arises if not all patterns are equally good: There may be outliers which both affect the training set and test set in a negative direction. With the entropic errorfunction a single outlier can have a dramatic impact. If a single pattern is put in the wrong class and the classifier is a hard classifier the errorfunction would be infinitely large. The infinite value will not so easily crop up when using the square errorfunction and continuous output.

A way of overcoming the outlier problem is by *robust statistics* [20]. Robust meaning that it is statistic where estimators is not so much affected by bad patterns.

A simple robust estimator for the mean is the median or the 10%-trimmed mean where the 10% largest and 10% smallest patterns are removed. The median is actually the 50%-trimmed mean.

An other mean estimator is the Hodges-Lehmann estimator: H/L is the median of all the pairwise averages.

Unfortunately are the robust mean estimators rather nonlinear, making ordinary tests invalid.

Derivatives

We will need the first and second order derivatives of errorfunctions to the optimization. The square errorfunction uses the linear output activation function, while the entropic error function uses the hyperbolic tangent output activation function. The derivations can be found in A.2. The first order derivatives of the squared errorfunction are:

$$\frac{\partial E_{sq}}{\partial w_{oh}} = -2 \sum_p^{n_p} \left(d_o^p - o_o^p\right) h_h^p \tag{4.36}$$

$$\frac{\partial E_{sq}}{\partial v_{hi}} = -2 \sum_o^{n_o} w_{oh} \sum_p^{n_p} \left(d_o^p - o_o^p\right) \left[1 - \left(h_h^p\right)^2\right] i_i^p \tag{4.37}$$

The advantage of using the entropic errorfunction together with the hyperbolic tangent shows up in the simple first order derivative:

$$\frac{\partial E_{entr}}{\partial w_{oh}} = - \sum_p^{n_p} \left(d_o^p - o_o^p\right) h_o^p \tag{4.38}$$

$$\frac{\partial E_{entr}}{\partial v_{hi}} = - \sum_p \sum_o \left(d_o^p - o_o^p\right) w_{oh} \left(1 - \left(h_h^p\right)^2\right) i_i^p \tag{4.39}$$

The second order derivative – called the Hessian – can be approximated in many ways. We define u to be any weight, both a hidden and output:

$$u_u = \begin{cases} v_{hi} & \text{for hidden weights} \\ w_{oh} & \text{for output weights} \end{cases} \tag{4.40}$$

The hidden and output weights can not be represented with an ordinary matrix. Instead it is represented with the weight matrices vectorized, that is by stacking the columns:

$$\mathbf{u} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{\vee} \\ \mathbf{W}^{\vee} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} v_{00} & v_{10} & \dots & v_{n_h 0} & v_{11} & \dots & v_{n_h n_i} \end{bmatrix}^{\top} \\ \begin{bmatrix} w_{00} & w_{10} & \dots & w_{n_o 0} & w_{11} & \dots & w_{n_o n_h} \end{bmatrix}^{\top} \end{bmatrix} \tag{4.41}$$

Then the "all-weight" Hessian is becoming:

$$\frac{\partial^2 E}{\partial \mathbf{u} \partial \mathbf{u}^{\top}} = \begin{bmatrix} \frac{\partial^2 E}{\partial \mathbf{w} \partial \mathbf{w}^{\top}} & \frac{\partial^2 E}{\partial \mathbf{w} \partial \mathbf{v}^{\top}} \\ \frac{\partial^2 E}{\partial \mathbf{v} \partial \mathbf{w}^{\top}} & \frac{\partial^2 E}{\partial \mathbf{v} \partial \mathbf{v}^{\top}} \end{bmatrix} \tag{4.42}$$

For both the square and the entropic errorfunction the Hessian might be approximated in several ways. The approximation might be structural: a thinning of the elements, or it might be working on the single element.

An approximation working on the single element is the **gauss-newton** approximation, which is also known as the Levenberg-Marquardt approximation[7]. This approximation uses that a term within the derivative normally should be negligeable compared to the other term. This term is the difference between the desired output and the neural network output $d_o^p - o_o^p$, and it pops up both with the square and the entropic costfunction in the hidden layer derivatives. An advantage with this approximation is that it forces the Hessian to be semi-positive definite.

Structural approximation can be reached through ignoring of elements (figure 4.5), most radical being the pseudo-newton where only the diagonal is being maintained (a of figure 4.5). By ignoring the dependence between weights in different layers, the Hessian becomes a block matrix (d). Possibly will the norm of the blocks be rather different, where an blockwise approximation could be useful (b and c). With a single unit output the number of weights in the output layer will be smaller than in the hidden layer. This means that the computation of the hidden layer part of the Hessian will take rather long time compared to the output part. By diagonal approximation only in the hidden layer time can be saved, while still maintaining some accuracy.

For the *square errorfunction* the full second order derivative for the output layer is:

$$\frac{\partial^2 E_{sq}}{\partial w_{o_2 h_2} \partial w_{o_1 h_1}} = 2 \sum_{p}^{n_p} h_{h_1}^p h_{h_2}^p \tag{4.43}$$

[7]The Levenberg-Marquardt approximation is different from the Levenberg-Marquardt optimization method.

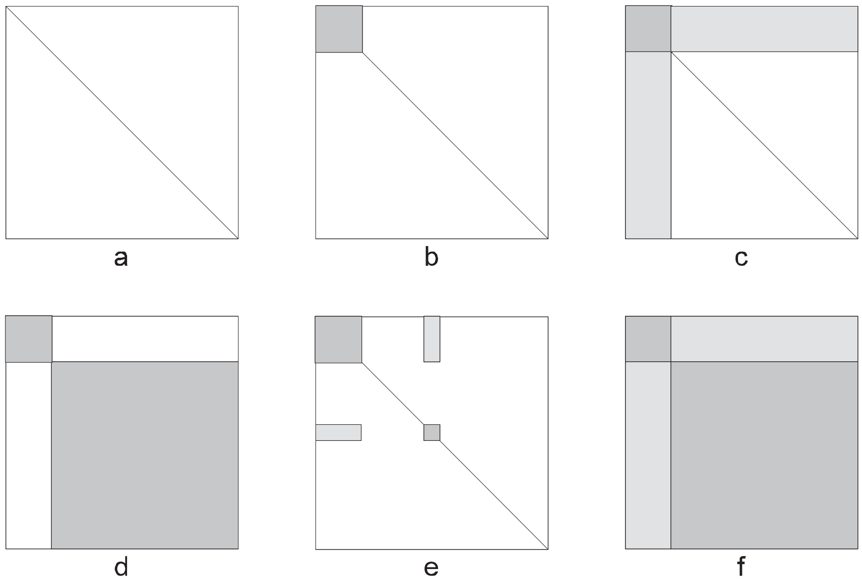


Figure 4.5: Hessian structural approximations

The full derivative for the hidden layer is:

$$\frac{\partial^2 E_{sq}}{\partial v_{h_2 i_2} \partial v_{h_1 i_1}} = 2 \sum_{o}^{n_o} \sum_{p}^{n_p} w_{oh_1} i^p_{i_1} \left[\left[1 - \left(h^p_{h_1}\right)^2\right] w_{oh_2} \left[1 - \left(h^p_{h_2}\right)^2\right] i^p_{i_2} - \begin{cases} (o^p_o - d^p_o)\, 2h^p_h \left[1 - (h^p_h)^2\right] i^p_{i_2} & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \right] \tag{4.44}$$

The upper line is the gauss-newton part. Ignoring the non-gauss-newton part and taking only the diagonal elements the derivative becomes the pseudo-gauss-newton:

$$\frac{\partial^2 E_{sq}}{\partial v^2_{hi}} = 2 \sum_{o}^{n_o} w^2_{oh} \sum_{p}^{n_p} (i^p_i)^2 \left[1 - h^{p\,2}_h\right]^2 \tag{4.45}$$

With the *entropic errorfunction* the full output layer derivative becomes:

$$\frac{\partial^2 E_{entr}}{\partial w_{o_1 h_1} \partial w_{o_2 h_2}} = \left[\sum_{p}^{n_p} h^p_{h_1} \left(1 - o^{p\,2}_o\right) h^p_{h_2} \right]_{o = o_1 = o_2} \tag{4.46}$$

Both the output layer derivative and hidden layer derivative for the entropic errorfunction resemble the square errorfunction derivatives, looking apart from the output activation function derivative: $(1 - o^{p\,2}_o)$. The hidden layer derivative becomes:

$$\frac{\partial^2 E_{entr}}{\partial v_{h_2 i_2} \partial v_{h_1 i_1}} = \sum_{p} \sum_{o} \left[w_{oh_1} w_{oh_2} i^p_{i_1} i^p_{i_2} \left(1 - \left(h^p_{h_1}\right)^2\right) \left(1 - \left(h^p_{h_2}\right)^2\right) \left(1 - \left(o^p_{o_2}\right)^2\right) \right. \tag{4.47}$$

$$- \left\{ \begin{array}{cc} 2\left(o_o^p - d_o^p\right) w_{oh} i_{i_1}^p i_{i_2}^p h_h^p \left(1 - (h_h^p)^2\right) \Big] & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{array} \right. \Bigg]$$

The top line again being the gauss-newton part. The pseudo-gauss-newton is:

$$\frac{\partial^2 E_{entr}}{\partial v_{hi}^2} = \sum_p \sum_o w_{oh}^2 \left(i_i^p\right)^2 \left(1 - (h_h^p)^2\right)^2 \left(1 - \left(o_{o_2}^p\right)^2\right) \tag{4.48}$$

For the cross derivative there is no diagonal approximation, but the derivative can still be split in a gauss-newton and a non-gauss-newton part:

$$\begin{aligned} \frac{\partial^2 E_{entr}}{\partial v_{h_2 i_2} \partial w_{o_1 h_1}} &= \sum_p^{n_p} \left[h_{h_1}^p \left(1 - \left(o_{o_1}^p\right)^2\right) w_{o_1 h_2} \left(1 - \left(h_{h_2}^p\right)^2\right) i_{i_2}^p \right. \\ &\quad + \left\{ \begin{array}{cc} \left(o_{o_1}^p - d_{o_1}^p\right) \left(1 - (h_h^p)^2\right) i_{i_1}^p & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{array} \right. \Bigg] \end{aligned} \tag{4.49}$$

4.6.3 Neural Network Regularization

We may need to augment the errorfunction with an additive term, getting the costfunction:

$$C = E + R \tag{4.50}$$

The reason for doing this is explained later in section 4.6.6.

Quadratic Weight Decay

The most used regularization method is the quadratic weight decay:

$$R_{qua} = \gamma_w \sum_{o=1}^{n_o} \sum_{h=1}^{n_h} w_{oh}^2 + \gamma_v \sum_{h=1}^{n_h} \sum_{i=1}^{n_i} v_{hi}^2 = \gamma_w \left\| \mathbf{w} \right\|_2^2 + \gamma_v \left\| \mathbf{v} \right\|_2^2 \tag{4.51}$$

This weight decay can be viewed as the energy of a spring with a spring constant of γ and extended or extracted with a length of w — or v. The advantage of this regularization is that the derivatives are quite simple:

$$\frac{\partial R_{qua}}{\partial u_u} = 2\gamma_u u_u \tag{4.52}$$

$$\frac{\partial^2 R_{qua}}{\partial u_u \partial u_u} = 2\gamma_u \tag{4.53}$$

The quadratic weight is actually making a prior assumption that the weight sizes are normal distributed [67].

A reason for separating the hidden and output weight decay parameters is that the ratio between the value of the hidden and output weights regulates the nonlinearity of the neural network.

The quadratic weight decay puts a high penalty on large weights.

Modified Quadratic Weight Decay

An other form of weight decay that do not punish large weights as much is this:

$$R_{modified} = \gamma_w \sum_{o=1}^{n_o} \sum_{h=1}^{n_h} \frac{w_{oh}^2}{1+w_{oh}^2} + \gamma_v \sum_{h=1}^{n_h} \sum_{i=1}^{n_i} \frac{v_{hi}^2}{1+v_{hi}^2} \tag{4.54}$$

The derivatives are unfortunately more complex:

$$\frac{\partial R_{modified}}{\partial u_u} = 2\gamma_u \frac{u_u}{\left(1+u_u^2\right)^2} \tag{4.55}$$

$$\frac{\partial^2 R_{modified}}{\partial u_{u_1} \partial u_{u_2}} = 2\gamma_u \frac{-u_u^2\left(3u_u^2+2\right)+1}{\left(1+u_u^2\right)^4} \tag{4.56}$$

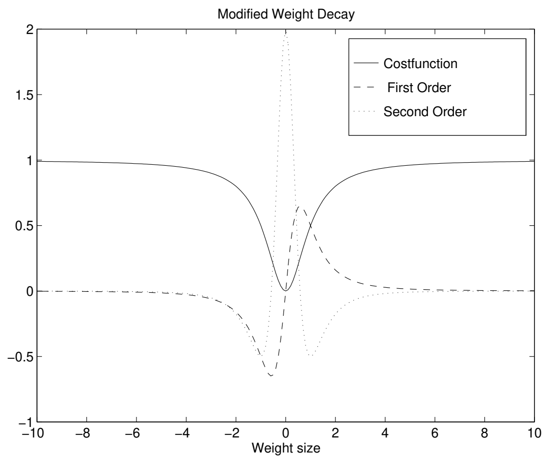


Figure 4.6: The modified weight decay.

None of the 2 above mentioned regularization methods are able to *crossbind* the weights: They bind the individual weight with no respect to the context. The weights are also *biased* against the value zero.

Moody Regularization

Moody has suggested an other type of regularization, that does not bias the weights against the value zero, but rather work on the look of the output function and "bias" against a smooth function, for example a constant function:

$$R_{moody,const} = \gamma_u \sum_{o=1}^{n_o} \sum_{u=1}^{n_u} \int_{\mathbf{i}} d\mathbf{i} P(\mathbf{i}) \left(\frac{\partial O_o(\mathbf{u},\mathbf{i})}{\partial \mathbf{i}}\right)^2 \tag{4.57}$$

or a linear function:

$$R_{moody,linear} = \gamma_u \sum_{o=1}^{n_o} \sum_{u=1}^{n_u} \int_{\mathbf{i}} d\mathbf{i} P(\mathbf{i}) \left\| \frac{\partial^2 O_o(\mathbf{u},\mathbf{i})}{\partial \mathbf{i} \partial \mathbf{i}^\top} \right\|_{ii}^2 \tag{4.58}$$

where $P(\mathbf{i})$ is the input density. This can be the empirical input density:

$$P_{empir}(\mathbf{i}) = \frac{1}{n_p} \sum_{p=1}^{n_p} \delta\left(\mathbf{i} - \mathbf{i}^p\right) \tag{4.59}$$

or a function that smoothes the "delta function bump".

From the formulas it can be seen that the output function is differentiated with respect to the input. The derivative terms are weighted according to their occurrence, for example: There is no need to punish the neural network for not being smooth were input patterns will never occur. Then the whole output function is integrated to get a number for the smoothness of the whole function.

With a square errorfunction and a Moody regularization as an additive term, the penalty to the feed forward becomes [82]:

$$R_{moody} = \gamma \left\|W\right\| \left\|V\right\| \tag{4.60}$$

4.6.4 Optimization

The problem of optimizing the neural network costfunction is usually a *nonlinear static unconstrained optimization problem.*

Nonlinear meaning that it is usually not possible to solve the problem directly, — instead *iterative* schemes must be applied, training epoch after epoch.

There is usually no temporal state involved in the feed-forward neural network. The patterns are treated as independent. Therefore the problem is *static.*

Unconstrained meaning that the variables to be optimized is not bounded, — for example: that they should be positive. This is in regard to the costfunction. Optimization of the errorfunction may be regarded as a constrained optimization where the constraints are imposed by the regularization. Fortunately the constraints are differentiable, so the constraints are easily added to the errorfunction giving the costfunction.

$$\min_{\mathbf{u}} (C) = \mathbf{u}^* \tag{4.61}$$

The optimization in connection with neural network usually called *training* or *learning.* Each of the iterative steps is called an epoch.

For an optimization method, 3 efficiency measures may be given [68]:

- **Robustness**. This concerns the methods' ability to *converge* and to converge to the global minimum: *global convergence.* In neural network optimization the global minimum is not exclusive. A "good" local minimum is sufficient.

- **Speed of convergence**. This describes how much the optimization is improving the weights from epoch to epoch. Often this is denoted by the terms: *Linear*, *superlinear* and *quadratic* [68].

$$\begin{array}{rll} \text{Linear:} & \left\|\mathbf{u}_{new} - \mathbf{u}^*\right\| \leq \beta_e \left\|\mathbf{u}_{old} - \mathbf{u}^*\right\| & 0 \leq \lim_{e\to\infty} \beta_e < 1 \\ \text{Superlinear:} & \left\|\mathbf{u}_{new} - \mathbf{u}^*\right\| \leq \beta_e \left\|\mathbf{u}_{old} - \mathbf{u}^*\right\| & \lim_{e\to\infty} \beta_e = 0 \\ \text{Quadratic:} & \left\|\mathbf{u}_{new} - \mathbf{u}^*\right\| \leq \gamma \left\|\mathbf{u}_{old} - \mathbf{u}^*\right\|^2 & \gamma > 0 \end{array} \tag{4.62}$$

One may say that linear convergence is where the number of determined ciphers increase linearly with the epochs, and that quadratic convergence is where the number increase quadratic.

- **Complexity of computation**. This is a measure of what the single epoch — a single step — of the iterative optimization takes of time. The value can be given in the big-O notation. The complexity of computation is first of all dependent upon the *number of variables* under optimization and the *required functions*: the ordinary function value, the first order derivative, and the second order derivative. An other complexity measure for the optimization is the store complexity, but in neural network optimization the use of storage is usually of no importance.

The speed of convergence and the complexity of computation usually work opposite — it is just a question of where to put the epoch. If the two values are "multiplied" together we get the time consume of the algorithm. Often the robustness fights against the time consume. This is because a robust algorithm would often work more globally and use several different points where the function is evaluated.

Some algorithms are only robust with a convex costfunction, that is where the Hessian is positive definite. The neural network costfunction — especially a unregularized — is not convex. To get both robustness and a fair time consume optimization methods can be mixed forming a hybrid optimization methods.

Some of the different optimization methods can be lined up:

- **Exhaustive search**. By exhaustive search I mean the systematic search of all possible states. If the variables are unlimited the number of states must be restricted be the help of some prior knowledge. With continuous valued weights a finite number of states can be obtained with a *grid search* strategy. After the computation of the function values in the grid points, we might interpolate and find a better minimum between the gridlines [49].

- **Random search**. The technique of random search is also called monte carlo: Start out with a number of random weights and hope that some of them *wins* — having a low function value. If the best is picked and mutated, that is some noise is added to the weights, we end up with a new group of weights in the surroundings of the ancestor. Hopefully one of them is better. This is then used in a new mutation. Critical is the size of the standard deviation of the noise used in the mutation.

- **Amoeba.** The Nelder-May method [66] uses only the functional value, — no derivatives. If the weight space under optimization is of dimension n_u it will need $n_u + 1$ function evaluations. These are the vertices in a simplex — an irregular shaped sort of prism. The vertex having the largest function value is mirrored in the hyperplane the other vertices span. This creates a new simplex, where a new vertex with the highest function value now can be mirror. The simplex — also called amoeba — should gradually move downhill.

- **Gradient descent**. If the costfunction is decreasing in a direction — the negative gradient direction — a step can be taking a that direction (of course!) If the step is kept fixed at a value η we arrive at the ordinary *backpropagation*:

$$\Delta \mathbf{u} = -\eta \frac{\partial C}{\partial \mathbf{u}} \tag{4.63}$$

  The problem with backpropagation is that if the step η is too small, the convergence is to slow, and if it is to large, it is not certain that the costfunction will decrease: We might land on the opposite side of the costfunction valley. By using *soft line search*, where we start off with a large η and gradually decrease it until the costfunction is decreased, we can get both improvements in the robustness and the speed of convergence, but at the cost of more complexity of computation. If the line search is *hard*, that is the optimal step size η^* is found, we end up with *steepest descent*[8]:

$$\Delta \mathbf{u} = -\eta^* \frac{\partial C}{\partial \mathbf{u}} \tag{4.64}$$

  This is usually not worth the effort, compared to soft line search.

  Some curvature information can be obtained if the old step is used in *backpropagation with momentum*:

$$\Delta \mathbf{u} = -\eta \frac{\partial C}{\partial \mathbf{u}} + \alpha \Delta \mathbf{u}_{old} \tag{4.65}$$

  A problem with the gradient descent is that all directions share the same step size. This is a problem with *stiff* functions, that is, functions with a long shallow valley in one direction and a short steep valley in an other. In mathematical terms this means that the ratio between the largest and the smallest Hessian eigenvalue is big. The step size has to be fitted to the short steep valley, not to bump into its' valley side.

  Neural network costfunction are often stiff because the inputs have different importance for the outputs. Consider the neural network input of principal components: The first should be of importance, while the higher should be noise that should be ignored.

  The gradient descent algorithms have linear convergence.

- **Conjugate gradient methods** and **Quasi-newton methods**. These 2 methods try to bend the gradient direction by remembering the gradient in some form from epoch to epoch. Some of the algorithms in this domain is Fletcher-Reeves, Polak-Ribiere, BFGS, DFP and scaled conjugate gradient [56] [66] [68]. These types of optimization have superlinear convergence.

- **Newton methods**. Newton methods are where the second order derivative - or part of it — is computed directly, that is not by approximations found through the use

[8]Some call gradient descent for steepest descent.

of the first order derivative. The pure newton multiplies the inverted costfunction Hessian with the costfunction gradient:

$$\Delta\mathbf{u} = -\left(\nabla^2_{\mathbf{uu}} C\right)^{-1} \nabla_{\mathbf{u}} C \tag{4.66}$$

The pure newton is not robust, because the Hessian might not be positive definite, that is the cost function is not fully convex. If the costfunction is concave in one dimension it will take a step uphill. Using the gauss-newton approximation the Hessian automatically becomes positive definite (or at least semi-positive definite). If the pseudo approximation is used the matrix inversion of 4.66 is replaced by an element-by-element division:

$$\Delta u_u = -\frac{\partial C}{\partial u_u} \bigg/ \frac{\partial^2 C}{\partial^2 u_u}$$

If the costfunction surface is "hyperparaboloidic", one newton step will immediately bring the weights to their optimal values. The costfunction surface has to have axis-parallel cross sections of the hyperparaboloide for that to happen with the pseudo-newton approximation.

Usually the costfunction is certainly not hyperparaboloidic and we have to take several steps. The costfunction can be of such a type, that the newton methods take a too large step, crossing the valley, and getting a worser costfunction value. In parallel with the gradient descent methods, some of the cure for this behavior is soft line search. To make the newton method yet more robust the failed newton step can be followed by a gradient descent step.

The Levenberg-Marquardt method uses the gauss-newton approximation step and the usual gradient descent step in a combination:

$$\Delta\mathbf{u} = -\left(\nabla^2_{\mathbf{uu}} C + \lambda\mathbf{I}\right)^{-1} \nabla_{\mathbf{u}} C \tag{4.67}$$

The $\mathbf{I}$ is here the identity matrix. Thus the Levenberg-Marquardt approximation is an interpolation between the gauss-newton step and an infinitely small gradient step with the parameter λ, as a sort of line search parameter — stored from epoch to epoch.

It seems to me that this method requires matrix inversions in the line search.

The pure newton has quadratic convergence for convex costfunctions [68].

- **Direct methods**. For some of the optimization problems in the neural network there are actually direct methods. For the linear output activation function the output weights are linear parameters, and with the square errorfunction it is the standard problem of least square, where the *normal equation* can be employed [48] [52]:

$$\mathbf{W} = \left[\left(\mathbf{H}\mathbf{H}^{\top}\right)^{-1} \mathbf{H}\mathbf{D}^{\top}\right]^{\top} \tag{4.68}$$

This is actually just the pure newton. Correlation among the outputs (see the equation 4.35) is no further problem:

$$\mathbf{W} = \left(\mathbf{H}\mathbf{\Sigma}^{-1}\mathbf{H}^{\top}\right)^{-1}\mathbf{H}\mathbf{\Sigma}^{-1}\mathbf{D}^{\top}$$

Neither is the augmentation of the square weight decay. In this case we end up with the ridge regression:

$$\mathbf{W} = \left[\left(\mathbf{H}\mathbf{H}^{\top} + \mathbf{R}\right)^{-1}\mathbf{H}\mathbf{D}^{\top}\right]^{\top} \tag{4.69}$$

Batch or on-line training

A question of optimization rather orthogonal to the optimization method is whether to use batch (also called off-line) or on-line training. Batch training is when all patterns are presented before a step is taken. On-line training is where one pattern is presented and then a step is taken. When the weights are optimized close to the optimal the on-line training should decrease the step size in an annealing type of fashion.

Initialization

For the iterative optimization an initialization must be determined. There is no definitive initialization. Putting the weights to zero will halt all the gradient dependent optimization techniques.

Usually it is a good idea to fit the weights so that the summation in the receiving unit — hidden or output unit — is around/between –1/+1. That is, adjusting the weights according to the *standard deviation* of the transmitting unit, and the number of transmitting units — the *fan in*.

By random initialization different parts of the weight space can be search, avoiding a local minima.

4.6.5 Pruning of weights

"Pruning of weights" is the removal of unimportant weights. The advantage of weight pruning could be summarized:

- Improving generalization through a decrease in the number of parameters (weights).
- Pinpointing the important inputs.
- Decrease of needed computation.

What weights shall be pruned away? The size of the weights seems not to be good choice. Better is it to look at the effect that the weights have on the neural network performance. The optimal would be to delete a weight and train the neural network several times again. If the several neural networks in average perform better on test sets, then this weight should be deleted. The computational task of such a scheme is too demanding. Instead of this empirical investigation an estimation scheme could be implemented:

To get a better generalization we delete weights. To interfere as little as possible with the training cost function we delete the weight with the smallest influence — the smallest *saliency* — on it.

The influence on the cost function δC when deleting one weight is approximated by the Taylor expansion:

$$\delta C_u \sim \underbrace{\frac{\partial C}{\partial \mathbf{u}^\top}\delta \mathbf{u}_u}_{\text{First order}} + \underbrace{\frac{1}{2}\delta \mathbf{u}_u^\top \frac{\partial^2 C}{\partial \mathbf{u}\partial \mathbf{u}^\top}\delta \mathbf{u}_u}_{\text{Second order}} + \dots \tag{4.70}$$

The cost function is Taylor expanded from the non-pruned architecture, and the point to be estimated is where the $_u$'th weight has been zeroed.

$$\left[\begin{array}{c} \delta \mathbf{u}_u = \mathbf{u}_{u,new} - \mathbf{u}_{u,old} \\ \mathbf{u}_{u,new} = \left[\begin{array}{cccccc} u_1 & u_2 & \dots & 0 & \dots & u_{n_u} \end{array}\right] \end{array}\right] \Rightarrow \delta \mathbf{u}_u = -\mathbf{u}'_{u,old} \tag{4.71}$$

The pruning should take place when the training has reached around the bottom of the cost function valley. There the first order derivative is zero:

$$\frac{\partial C}{\partial \mathbf{u}} \approx \mathbf{0} \tag{4.72}$$

And not taking the higher $o(n^3)$ terms into account the effect of the perturbation ends up with being:

$$\delta C_u \approx \frac{1}{2}\delta \mathbf{u}_u^\top \frac{\partial^2 C}{\partial \mathbf{u}\partial \mathbf{u}^\top}\delta \mathbf{u}_u \tag{4.73}$$

There are now two ways to proceed: The costfunction derivative can be used as the full Hessian. This is called OBS — **Optimal Brain Surgeon** [25]. Or the cost function second order derivative can diagonal approximated. This is called OBD — **Optimal Brain Damage** [45] [75]. With OBD there is no cross effects between the weights: The perturbation $\delta \mathbf{u}_u$ can now be sat equal for all the weights:

$$\delta \mathbf{u} = \delta \mathbf{u}_u = -\mathbf{u}_{u,old} \tag{4.74}$$

This means that the saliency δC only needs to be calculated once with a single weight perturbation vector $\delta \mathbf{u}$:

$$\delta C_{\mathbf{u}} \approx \frac{1}{2}\delta \mathbf{u}^\top \frac{\partial^2 C}{\partial \mathbf{u}^2}\delta \mathbf{u} \tag{4.75}$$

In theory one should not delete more than one weight with just one saliency calculation, because the deletion of the weight with the lowest saliency could change the other low saliency weights. In practice the one-deletion scheme is too slow, so ignoring the cross-effects one can delete a set of the lowest weights with the size of for example 5%

The pruning can be viewed as the removal of a weight or a removal of an input channel to the weight. In connection with the costfunction derivatives the pruning should also be in the input channel.

Generalization based pruning

With [65] has proposed to obtain the saliency using a generalization error estimate (section 4.6.6. This can both be implemented with the OBD and the OBS.

4.6.6 Generalization

Generalization is the ability of a model to predict a new unseen pattern. Statistically the generalization is a combination of the probability of a pattern and the conditioned probability of the size of the error with that pattern:

A fundamental observation with neural network is that *a costfunction minimum* for a finite training set is *not* the *minimum for the generalization error.* This is due to *overfitting*: where the model has fitted the noise in the training set, or has made an extraordinary complex model (figure 4.8).

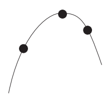


Figure 4.7: A probable fit for noise-free data.

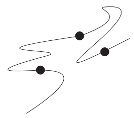


Figure 4.8: A rather daring noise-free interpretation: probably overinterprented.

The components of the generalization can be represented in an equation [41] (This is for a square error network):

$$\Gamma = \sigma_\epsilon^2 + MSME + WFP \tag{4.76}$$

σ_ϵ^2 represents the inherent noise, that can never be accounted for. This is setting the floor for what can achieved of classification/prediction. Patterns are struck with different amount of noise, so if the generalization is measured on a limited test set, the empirical generalization estimate might be smaller than the true.

MSME stands for the mean square model error: The limitation of the model. For example, fitting a linear model to a theoretical model which is parabolic, will cause this type of error.

WFP means the weight fluctuation penalty. This is the loss of generalization ability due to the overfitting.

Comparing a linear model with the non-linear model of the neural network, the neural network has a smaller *MSME*, while the *WFP* potentially is bigger.

There are 2 problems associated with generalization:

- **Methods** for obtaining good generalization. This usually means to restrict the *WFP*, while maintaining as much as possible of the modelling abilities (keeping the *MSME* low).

- **Measures (assessments)** for generalization. This means to estimate the true generalization Γ.

The 2 problems are of course somehow associated with each other: If one is able to measure the generalization one can train to minimize it.

Obtaining good generalization

In [79] the following methods for obtaining good generalization are outlined:

- **Early Stop**. By starting out with a simple network, and stopping early with just a few epochs of training the generalization can be better than the fully trained network, because the neural network does not get to learn all the subtle "noise patterns" in each of the patterns.

- **Penalize network complexity**. This is precisely what the regularization does (p. 43). The quadratic weight decay penalizes large weights, as they — to a certain extent — make the neural network more complex.

- **Prune weights**. Pruning is a more drastic move against the network complexity. The pruning can be regarded as a very hard regularization on a weight.

- **Add tasks**. Letting the neural network predict against several similar targets will restrict the "WFP-space", and increase the generalization. If 2 problems are closely related and there is a limited amount of data, it might be better to use a single neural network with 2 outputs, rather than 2 neural networks with one output each: Common hidden weights reduce the number of parameters.

- **Hints**. By switching between 2 training set — a master and a hint (that is not necessarily completely statistically equivalent) — the generalization might be improved [79].

- **Pseudo-data**. Noise on the input is pseudo-data. The noise prevents the optimization to grow too accustomed to the particular training set under optimization. The noise should carefully reflect the true noise on the input.

  Of these methods is the pruning often the one with the most success for handling a too large neural network.

Measuring generalization

- **Empirical measures**. Empirical measures of the generalization use a limited dataset — called the validation set or the test set — that has not been included in the training and calculates the mean error of this set. If the validation set was infinitely large we should end up with the true generalization Γ.

- **Algebraic measures,** works by constructing an extrapolation formula.

The word "test set" is usually used with a set that is never involved in optimizing the neural network. "Validation set" is used when a set is used to optimize the generalization of the neural network.

By shifting training set with validation set in different training sessions, we get the method of cross-validation. The extreme example of this is the "leave one out" cross-validation scheme, where just one pattern is used as validation. By this method n_p number of neural networks must be trained, if all patterns are to be used as validation "sets". If there is a high error-correlation among the patterns, the leave one out method will fail, because the pattern left out of the training will still be contain in the training set through its highly correlated fellow patterns.

Algebraic Generalization Error Measurement

Suppose we have some generating system [41]:

$$y = g(\mathbf{i}) + \epsilon(\mathbf{i}) \tag{4.77}$$

Under certain assumption [41] this generating system could be modeled with a neural network. If this model is *complete* [40] [41] there exists a *true* weight vector:

$$\forall \mathbf{i}: \quad o(\mathbf{i}, \mathbf{u}^\diamond) \equiv g(\mathbf{i}) \tag{4.78}$$

When training the neural network we want to approximate this function, possibly imitate the true weights: That is why the true weights are sometimes referred to as the *teacher weights*, and the weights from the optimization of the neural network $\hat{\mathbf{u}}$ are called the *student weights*.

The current estimate $\hat{\mathbf{u}}$ of the true weights should, with unbiased optimization, be normal distributed around the true weights $\mathbf{u}^\diamond$ (the big ellipse in figure 4.9) [22] [60]. With $\delta\mathbf{u} = \hat{\mathbf{u}} - \mathbf{u}^\diamond$:

$$\delta\mathbf{u} \in N(\mathbf{0}, \mathbf{\Sigma}^*) \tag{4.79}$$

The variance (dispersion) is determined by the structure of the problem and the neural network model. With the introduction of regularization it is hoped that the size of this variance becomes smaller. Unfortunately the regularization also biases the neural network, and move the "optimization optimal" weight $\mathbf{u}^*$ away from the true weight (The little ellipse in figure 4.9).

One of the methods of to get to the generalization error estimate is to interpolate on the costfunction surface and the generalization surface of figure 4.9, between the true weight $\mathbf{u}^\diamond$ and the current estimate $\hat{\mathbf{u}}$ [22] [60]. An other method is to interpolate on the neural network output function [67].

Square Error An algebraic generalization error measurement exists in Akaike's final prediction error — **FPE** [2]. FPE states, that due to overfitting the average training error will be smaller than the true error variance:

$$\langle E \rangle \approx \left(1 - \frac{n_u}{n_p}\right) \sigma_\epsilon^2 \tag{4.80}$$

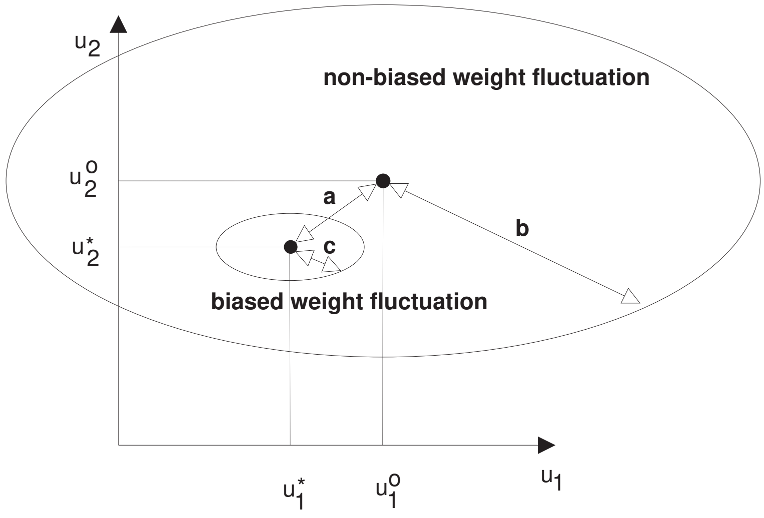


Figure 4.9: Weight fluctuations for the non-biased and biased estimator of the true weights: The generalization seen from the top.

The $\langle\rangle$ should average over all possible training set with the size of n_p with all possible models: $\langle\rangle_{n_p}$. The average test error will have a symmetrically error, that is, it will be larger than the true variance:

$$\langle G\rangle \approx \left(1 + \frac{n_u}{n_p}\right)\sigma_\epsilon^2 \tag{4.81}$$

A specific training error is the known parameter, so with 4.80 and 4.81 combined the estimated average test from the training error becomes[9]:

$$\widehat{\langle G_{FPE}\rangle} = \frac{n_p + n_u}{n_p - n_u} E \tag{4.82}$$

The average test error is only an estimate, because the training error E that can be measure is not an average value over all possible training set with the size of n_p.

With the introduction of regularization, the *effective number of parameters* decreases [57]:

$$\widehat{n}_{u,\text{eff}} = \sum_{oh}^{n_w}\left(\frac{\omega_{oh}}{\omega_{oh} + \frac{2}{n_p}\gamma_w}\right)^2 + \sum_{hi}^{n_v}\left(\frac{\nu_{hi}}{\nu_{hi} + \frac{2}{n_p}\gamma_v}\right)^2 \tag{4.83}$$

Where ω_{oh} and ν_{hi} are the second order derivatives:

$$\omega_{oh} = \frac{1}{n_p}\frac{\partial^2 E}{\partial w_{oh}^2} \tag{4.84}$$

$$\nu_{hi} = \frac{1}{n_p}\frac{\partial^2 E}{\partial v_{hi}^2} \tag{4.85}$$

[9] $\widehat{\langle\rangle}$ is the estimate of the mean generalization, not the mean of the estimates.

The asymptotic behavior of the effective number of parameters should of course be: With no regularization the effective number of parameter is equal to the actual number. And with a hard regularization there is no parameters at all:

$$\begin{aligned} \widehat{n}_{u,\text{eff}}\big|_{\gamma_u \to 0} &= n_w + n_v \\ \widehat{n}_{u,\text{eff}}\big|_{\gamma_u \to \infty} &= 0 \end{aligned} \tag{4.86}$$

With this new value for the number of parameters the FPE can be modified to the so-called generalized prediction error — GPE:

$$\langle \widehat{G_{GPE}} \rangle = \frac{n_p + \widehat{n}_{u,\text{eff}}}{n_p - \widehat{n}_{u,\text{eff}}} E \tag{4.87}$$

This estimate for the generalization error has been used with success in Svarer [75] for a pruning stop criterion. The estimate is not so good with large network, that is, very oversized models.

Entropic Error For the entropic errorfunction exists a similar "FPE" estimate [47] [50]:

$$\langle \widehat{G_{FPEE}} \rangle = E + \frac{n_u}{n_p} \tag{4.88}$$

With the effective number of parameter, taking account for the regularization, the generalization estimate becomes:

$$\langle \widehat{G_{FPEER}} \rangle = E + \frac{\widehat{n}_{u,\text{eff}}}{n_p} \tag{4.89}$$

4.6.7 Adaptive Regularization

When beginning the optimization there is no obvious size the square weight decay should have. We may though say that the regularization should be hard if we are working with a complex neural network on a small noisy training set. But having a weight optimized neural network, the generalization estimates from the previous section could be used, and we might try to minimize the generalization error with respect to the weight decay parameter. The generalization error estimates should of course be the versions that contain adjustment for the regularization.

Square Costfunction

For the square errorfunction the estimate is [24] [67]:

$$G_{sq} \approx \left(1 + \frac{\widehat{n}_{u,\text{eff}}}{n_p}\right)\sigma^2 + \frac{2}{n_p^2}\sum_{oh}\omega_{oh}\left(\frac{\gamma_w w_{oh}^{\diamond}}{\omega_{oh} + \frac{2}{n_p}\gamma_w}\right)^2 + \frac{2}{n_p^2}\sum_{hi}\nu_{hi}\left(\frac{\gamma_v v_{hi}^{\diamond}}{\nu_{hi} + \frac{2}{n_p}\gamma_v}\right)^2 \tag{4.90}$$

The true weights $v_{hi}^{\diamond}$ and $w_{oh}^{\diamond}$ are not known, but can be replaced by the currents weights: w_{oh} and v_{hi}. The variance σ^2 can be estimated through the GPE:

$$\widehat{\sigma}^2 = \left(1 - \frac{\widehat{n}_{u,\mathrm{eff}}}{n_p}\right)^{-1} E \tag{4.91}$$

We might optimize the weight decay parameters γ_v and γ_w with some gradient method. To that we need the derivatives:

$$\begin{aligned} \frac{\partial G_{sq}}{\partial \gamma_w} &= \frac{4}{n_p^2} \sum_{oh} \frac{\left[\gamma_w w_{oh}^2 - \sigma^2\right] \omega_{oh}^2}{\left(\omega_{oh} + \frac{2}{n_p}\gamma_w\right)^3} \\ \frac{\partial^2 G_{sq}}{\partial \gamma_w^2} &= \frac{4}{n_p^2} \sum_{oh} \frac{\left(\omega_{oh} + \frac{2}{n_p}\gamma_w\right)^2 \omega_{oh}^2 \left[\left(\omega_{oh} + \frac{2}{n_p}\gamma_w\right) w_{oh}^2 - \frac{6}{n_p}\gamma_w w_{oh}^2 + 6\frac{\sigma^2}{n_p}\right]}{\left(\omega_{oh} + \frac{2}{n_p}\gamma_w\right)^6} \end{aligned} \tag{4.92}$$

This is for the weight decay parameter for the output weights; the equation in connection with the hidden weights is equivalent.

Entropic Costfunction

The generalization error estimate is for the entropic errorfunction [47]:

$$\begin{aligned} G_{entr} &\approx \left\langle G\left(\mathbf{u}^{\diamond}\right)\right\rangle + \frac{\partial G}{\partial \mathbf{u}} \left\langle \delta \mathbf{u}\right\rangle + \frac{\partial^2 G}{\partial \mathbf{u} \partial \mathbf{u}} \left\langle \delta \mathbf{u} \delta \mathbf{u}\right\rangle \\ &\approx \left\langle G\left(\mathbf{u}^{\diamond}\right)\right\rangle + \frac{\partial E}{\partial \mathbf{u}} \left\langle \delta \mathbf{u}\right\rangle + \frac{\partial^2 E}{\partial \mathbf{u} \partial \mathbf{u}} \left\langle \delta \mathbf{u} \delta \mathbf{u}\right\rangle \end{aligned} \tag{4.93}$$

The pseudo-derivatives for this function is:

$$\begin{aligned} \frac{\partial G_{entr}}{\partial \gamma_w} &= \frac{2}{n_p} \sum_{oh} \frac{\omega_{oh}^2 \left(2\gamma_w w_{oh}^2 - 1\right)}{\left(\omega_{oh}^2 + \frac{2}{n_p}\gamma_w\right)^3} \\ \frac{\partial^2 G_{entr}}{\partial \gamma_w^2} &= \frac{4}{n_p^2} \sum_{oh} \frac{\omega_{oh}^2 \left[\omega_{oh} w_{oh} - \frac{4}{n_p}\gamma_w w_{oh}^2 + \frac{3}{n_p}\right]}{\left(\omega_{oh}^2 + \frac{2}{n_p}\gamma_w\right)^4} \end{aligned} \tag{4.94}$$

Gradient descent

To optimize with respect to γ the ordinary gradient descent (page 47) can be used. For the output weights the formula is:

$$\gamma_{w,new} = \gamma_{w,old} - \alpha \frac{\partial G\left(\gamma_{w,old}\right)}{\partial \gamma_w} \tag{4.95}$$

This method has been used in [24].

Newton Step from Zero

An other optimization method could be the newton methods (p. 47). Possibly will the calculation be to complicated for numerical stability. To make the derivatives simpler the weight decay parameter could be put at zero: $\gamma = 0$, and take the newton step from there. The optimization step becomes equal to the optimization target:

$$\gamma_{new} = \Delta\gamma \tag{4.96}$$

The simpler derivatives for the square errorfunction:

$$\begin{aligned} \left.\frac{\partial G_{sq}}{\partial \gamma_w}\right|_{\gamma_w=0} &= -\frac{4}{n_p^2}\sum_{oh}\frac{\sigma^2}{\omega_{oh}} \\ \left.\frac{\partial^2 G_{sq}}{\partial \gamma_w^2}\right|_{\gamma_w=0} &= \frac{4}{n_p^2}\sum_{oh}\frac{w_{oh}^2 + \frac{6}{n_p}\sigma^2}{\omega_{oh}^2} \end{aligned} \tag{4.97}$$

The newton step is:

$$\begin{aligned} \gamma_{w,sq,new} &= \frac{\sum_{oh}\frac{\sigma^2}{\omega_{oh}}}{\sum_{oh}\frac{\omega_{oh}w_{oh}^2+\frac{6}{n_p}\sigma^2}{\omega_{oh}^2}} \\ &= \sum_{oh}\frac{1}{\omega_{oh}}\sum_{oh}\frac{\omega_{oh}^2}{\frac{\omega_{oh}w_{oh}^2}{\sigma^2}+\frac{6}{n_p}} \end{aligned} \tag{4.98}$$

For the entropic the similar expression is:

$$\begin{aligned} \gamma_{w,entr,new} &= \frac{\sum_{oh}\frac{1}{\omega_{oh}}}{\sum_{oh}\frac{\omega_{oh}w_{oh}^2+\frac{3}{n_p}}{\omega_{oh}^2}} \\ &= \sum_{oh}\frac{1}{\omega_{oh}}\sum_{oh}\frac{\omega_{oh}^2}{\omega_{oh}w_{oh}^2+\frac{3}{n_p}} \end{aligned} \tag{4.99}$$

Direct Solving

The optimization of the generalization might be performed directly by letting the gradient be zero: The summation unfortunately is in its way. However if individual weight decays are used, all terms of the summation should be zero [67]:

$$\begin{aligned} 0 &= \gamma_{w,oh}w_{oh}^2 - \sigma^2 \\ \gamma_{w,oh,sq} &= \frac{\sigma^2}{w_{oh}^2} \end{aligned} \tag{4.100}$$

The same could be done with the entropic function:

$$\begin{aligned} 0 &= \omega_{oh}^2\left(2\gamma_w w_{oh}^2 - 1\right) \\ \gamma_{w,oh,entr} &= \frac{1}{2w_{oh}^2} \end{aligned} \tag{4.101}$$

These steps lead to pruning [23] [67].

4.6.8 Saliency Map

The saliency map [58] is a map of the importance of a dimension in the datamatrix. In the case of brain imaging that is the importance of a voxel or group of voxels. The importance is measured in relation to the generalization error of the neural network. Ideally such a value should be computed by running the neural network through several training sessions on several training and validation sets. This task would be too cumbersome (This method would also run into troubles with correlations), instead the importance with an estimate using some approximations:

- **Using training costfunction**. The generalization is not used directly, instead the training costfunction is used. If the neural network is trained to the best generalization weights and architecture we would expect the ordinals of the saliencies to be almost same, regarding the generalization and the training costfunction.

- **Taylor approximation**. The saliency is not computed directly, but through a Taylor expansion estimation. The terms containing the first order and second order derivatives are used.

When deriving the saliency map there 2 paths that can be followed: Either the costfunction can be expanded with respect to the *voxels*, or it can be expanded with respect to the *basis* summing the changes in the costfunction from all the weights connected to the voxel of interest. In the first case the Taylor permutation has to be $\delta x_x^p = -x_x^p$; in the second case: $\delta b_{ix} = -b_{ix}$. The final result will of course be the same. If the later is used the Taylor expansion will take the form:

$$\delta C_x = \underbrace{\sum_i \frac{\partial C}{\partial \mathbf{b}^\top}\delta\mathbf{b} + \frac{\partial C}{\partial \mathbf{u}^\top}\delta\mathbf{u}}_{\text{First order term}} + \underbrace{\frac{1}{2}\sum_i \delta\mathbf{b}^\top \frac{\partial^2 C}{\partial\mathbf{b}\partial\mathbf{b}^\top}\delta\mathbf{b} + \underbrace{\sum_i \delta\mathbf{b}^\top \frac{\partial^2 C}{\partial\mathbf{b}\partial\mathbf{u}^\top}\delta\mathbf{u}}_{\text{Crossterm}} + \frac{1}{2}\delta\mathbf{u}^\top \frac{\partial^2 C}{\partial\mathbf{u}\partial\mathbf{u}^\top}\delta\mathbf{u}}_{\text{Second order term}} + \ldots \tag{4.102}$$

This equation is approximated further:

- **No retraining**. The removal of one voxel may cause the optimal value of the weights in the neural network to change. No retraining is done. This means that we force $\delta\mathbf{u} = \mathbf{0}$, such that the term containing $\delta\mathbf{u}$ is neglected.

- **Diagonal approximation**. For the second order derivative are the cross dimension dependencies (cross voxel dependencies) ignored: The full Hessian is enormous.

These 2 approximations lead the equation to:

$$\delta C_x = \sum_i \frac{\partial C}{\partial \mathbf{b}^\top}\delta\mathbf{b} + \frac{1}{2}\sum_i \frac{\partial^2 C}{(\partial\mathbf{b}^\top)^2}\delta\mathbf{b}^2 \tag{4.103}$$

There is no derivative term from the regularization if it is additive separable from the errorfunction. The last approximation is:

- **Gauss-newton**. The neural network should be well trained before the saliency calculation, thus the term containing $(d_o^p - o_o^p)$ could be left out.

Finally we end up with:

$$\delta C_x = \sum_i \frac{\partial E}{\partial \mathbf{b}^\top} \delta \mathbf{b} + \frac{1}{2} \sum_i \left\{ \begin{array}{cc} \sum_o \frac{\partial^2 \emptyset_o^p}{\left(\partial \mathbf{b}^\top\right)^2} & \text{for square error} \\ \sum_o \left(1 - (o_o^p)^2\right) \frac{\partial^2 \emptyset_o^p}{\left(\partial \mathbf{b}^\top\right)^2} & \text{for entropic error} \end{array} \right\} \delta \mathbf{b}^2 \quad (4.104)$$

Because the neural network input **i** and the basis **b** are in a linear relationship the derivative to the cost function with respect to the basis and the neural network input will look alike.

The derivatives for the squared cost function are:

$$\frac{\partial C_{sq}}{\partial b_{ix}} = -2 \sum_p \sum_o \sum_h (d_o^p - o_o^p)\, w_{oh} \left(1 - (h_h^p)^2\right) v_{h,i} x_x^p \quad (4.105)$$

$$\frac{\partial^2 C_{sq}}{\partial b_{ix}^2} = 2 \sum_p \sum_o \left[\sum_h w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} \right]^2 (x_x^p)^2 \quad (4.106)$$

Using 4.103 the **voxel saliency for the square costfunction** ends up with being:

$$\begin{aligned} \delta C_{sq,x} = {} & 2 \sum_p \sum_o \sum_h \sum_i (d_o^p - o_o^p)\, w_{oh} \left(1 - (h_h^p)^2\right) v_{h,i} b_{ix} x_x^p \\ & + \sum_p \sum_o \sum_i \left[\sum_h w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} \right]^2 b_{ix}^2 (x_x^p)^2 \end{aligned} \quad (4.107)$$

The saliency for the square costfunction has been calculated by Mørch [58]. The entropic cost function will have the following derivatives:

$$\frac{\partial C_{entr}}{\partial b_{ix}} = - \sum_p \sum_o \sum_h (d_o^p - o_o^p)\, w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} x_x^p \quad (4.108)$$

$$\frac{\partial^2 C_{entr}}{\partial b_{ix}^2} = \sum_p \sum_o \left(1 - (o_o^p)^2\right) \left[\sum_h w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} \right]^2 (x_x^p)^2 \quad (4.109)$$

Which brings the **voxel saliency for the entropic cost function** to:

$$\begin{aligned} \delta C_{entr,x} = {} & \sum_p \sum_o \sum_h \sum_i (d_o^p - o_o^p)\, w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} b_{ix} x_x^p \\ & + \frac{1}{2} \sum_p \sum_o \sum_i \left(1 - (o_o^p)^2\right) \left[\sum_h w_{oh} \left(1 - (h_h^p)^2\right) v_{hi} \right]^2 b_{ix}^2 (x_x^p)^2 \end{aligned} \quad (4.110)$$

In relation to weight pruning the saliency map calculation is just the OBD-saliency for the whole SVD/PCA-NN construction.

The value coming out of the saliency map calculation is not a test size — like most other values in the neural network. It depends for example on the type of costfunction and normalization of it.

Negative saliencies can actually arise for some voxels. The interpretation is that these voxels *disturb* the neural network prediction. It is because the principal component does not separate the voxels containing the paradigm from the voxel that contain noise.

Relation to the Eigenpictures

Lets assume that we are working with a single output linear model trained to the square errorfunction. We get the input to this model from the SVD/PCA projection. If this model is not pruned we are able to optimize fully to the desired value, getting the term $(d^p - o^p)$ zero. The saliency of this model will only contain the second order term of equation 4.107, the hidden units will be discarded, and the two weight layers $(u_i \equiv w_h v_{hi})$ in the neural network melt together with the basis:

$$\begin{aligned} \delta C_{sq,x} &= \sum_i \sum_p u_i^2 b_{ix}^2 \left(x_x^p\right)^2 && (4.111) \\ &= \sum_p \left(b'\right)_x^2 \left(x_x^p\right)^2 && (4.112) \end{aligned}$$

Thus we are having a *squared* weight from the voxel $_x$ to the output. Furthermore is the mean[10] *square* value of the voxel added.

If we are just looking at an interesting principal component $_i$ — which is important for the desired output, the "saliency" for this will just be itself:

$$\delta C_{sq,x} = b_{ix} \tag{4.113}$$

Thus with a comparison between the saliency and the basis — the saliency should rather be compared with the square root value.

4.7 The Hidden Units as a State Space

We might describe a state of the neural network in a very simple way: look at the value of the hidden units:

$$h_h^p = \tanh\left(\sum_{i=1}^{n_i} v_{hi} i_i^p + v_{h0}\right) \tag{4.114}$$

A point — a pattern — p is situated in the n_h-dimensional space.

This space is not orthogonal as a canonical variable space. Nor is it deterministic. The nondeterministique arises when the neural network is not trained to the global minimum of the costfunction. The choice of the costfunction is in itself not deterministic: If we restrict the model to a linear model predicting normal distributed data we can solve it deterministically, but even with the introduction of regularization the model becomes nondeterministic, the model becoming dependent on the prior information that is laid in the regularization parameter. Although it might be expected that the state space to some degree is mapping deterministic from training to training.

An other problem with the state space is that there is no ordering of the hidden units. The hidden units can be permuted and map precisely the same.

The state space from a single two-layer neural network is perhaps a rather restricted setup. A "better" way could be to use 2 two-layer neural network in series: The first

[10]...Or rather — with this normalization — the sum.

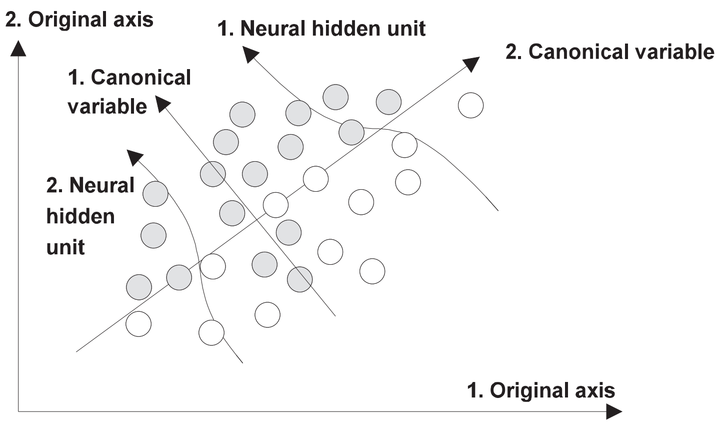


Figure 4.10: The state space projection. The neural network projections are shown bended. They are not bended in the 2D plane, but rather in the third plane.

neural network outputting to the "hidden" units to be used in the state space analysis; the second neural network using these units outputting to the actual desired value. This arrangement may give a not so restricted placement within the state space.

4.7.1 "Backprojektion" from the State Space

As a PCA eigenpicture the projection onto the hidden units has an eigenpicture. The hidden weights are neither normal or orthogonal. To cope with this we must apply the pseudo-inverse:

$$\mathbf{x}^{\mathcal{H}} = \mathbf{B}\left(\mathbf{V}^{\top}\mathbf{V}\right)^{\dagger}\mathbf{V}^{\top} \tag{4.115}$$

Furthermore, we can take a point in the hidden space and backproject it onto the voxels, forming a general image:

$$\mathbf{x}^{\mathcal{H}} = \mathbf{B}\left(\mathbf{V}^{\top}\mathbf{V}\right)^{\dagger}\mathbf{V}^{\top}\,\mathrm{atanh}\left(\mathbf{h}\right) \tag{4.116}$$

The equation is a kind of reverse multivariate regression [52].

4.7.2 "Cluster Analysis" of the State Space

When training the network to a desired value the patterns are weighted equally. The output residual signal of the neural network does not directly tell how much effort has been laid in optimization of a particular pattern. One can investigate the internal state of the neural network. This might reveal that some patterns are treated differently than normal. One might also identify differences among groups of input patterns: Groups that on the output are equal but behave differently within the neural network due to important differences in the input pattern.

Ordinary cluster analysis can separate patterns, but the patterns will be clustered according to the largest variations. In functional brain image analysis the largest variation is usually due to the intersubject and intersession variation and not the variation due to the paradigm. What we need is not this ordinary cluster analysis, which might be called "unsupervised cluster analysis", but rather a "supervised" cluster analysis. The hidden units' state space is the place in the analysis where the data has been trained with supervising, and not yet have been collapsed into the training classes.

In classification problems the patterns will arrange themselves in clusters. The statistical analysis of the patterns and clusters can be divided in two groups:

- **Cluster to cluster**. Here are the test, were the patterns are divided in groups according to the desired value of the neural network, not so interesting: The test should show a difference between the cluster of one group and the cluster of an other, because that is what the optimization of the neural network should do. More interesting is it to look at clusters that describe groups that are targeting at the same desired value.

- **Pattern to cluster**. Here is a pattern compared with its' cluster to see if behave differently than normal.

The hidden units with the hyperbolic tangent activation function have a range from –1 to +1 and therefore cannot be normal distributed, and certainly not if the units are more or less in saturation.

Anyway: if we approximate with the gaussian distribution we are able to use second moment statistics. The clusters will then be turned into ellipsoides.

A central subject for the second moment analysis is the Mahalanobis distance, which is the variance (dispersion) normalized squared distance. Here taken between two means μ_1 and μ_2 with a common dispersion $\mathbf{\Sigma}$:

$$D^2 = \|\mu_1 - \mu_2\|^2_{\mathbf{\Sigma}^{-1}} = (\mu_1 - \mu_2)^\top \mathbf{\Sigma}^{-1} (\mu_1 - \mu_2) \tag{4.117}$$

All formula in this section are from standard multivariate statistic books [10] [52].

Pattern to Cluster

Assuming that the mean vector and the dispersion matrix are infinitely well determined, the variance normalized squared distances will be distributed according to χ^2.

$$(\mathbf{h}^p - \mu)^\top \mathbf{\Sigma}^{-1} (\mathbf{h}^p - \mu) \in \chi^2(n_h) \tag{4.118}$$

However, the mean and dispersion are determined through a finite number of patterns n_p. The empirical mean is normal distributed:

$$\bar{\mathbf{h}} \in N_{n_h}\left(\mu, \frac{1}{n_p}\mathbf{\Sigma}\right) \tag{4.119}$$

and the empirical dispersion matrix will be Wishart distributed:

$$\mathbf{S} \in W_{n_h}\left(n_p - 1, \frac{1}{n_p - 1}\mathbf{\Sigma}\right) \tag{4.120}$$

Cluster to Cluster

The types of test can be classified according to what they test and how many clusters they use in a test:

	One-sample	**Two-sample**	**Multi-sample**	
Equal mean	Hotellings one-sample test	Hotellings two-sample test	Wilk's Λ	Center of the clusters
Equal Dispersion			Box's M, Sen-Puri	Cross sections of the clusters
Complete homogenity				Both center and cross sections
	One cluster against reference	Two cluster compared	Many clusters compared	

Table 4.1: Types of tests

To use the one-sample test one must have an infinitely well determined point working as the reference.

Having more than 2 cluster the two-sample tests can be used between all pairs of clusters. The batch of test then produces a triangular matrix with test values.

Equal two-sample mean The Hotelling's two-sample T^2 test is the test in this domain:

$$\begin{array}{ll} \mathrm{H}_0: & \mu_1 = \mu_2 \\ \mathrm{H}_1: & \mu_1 \neq \mu_2 \end{array} \tag{4.121}$$

The precondition here is that the sample dispersions are equal: $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. The method is to measure the distance between the 2 cluster and then "divide" with the variance:

$$D^2 = \left(\bar{\mathbf{h}}_1 - \bar{\mathbf{h}}_2\right) \mathbf{S}^{-1} \left(\bar{\mathbf{h}}_1 - \bar{\mathbf{h}}_2\right)^\top \tag{4.122}$$

The $\mathbf{S}$ is the "weighted" dispersion matrix:

$$\mathbf{S} = \frac{(n_1 - 1)\,\mathbf{S}_1 + (n_2 - 1)\,\mathbf{S}_2}{n_1 + n_2 - 2} \tag{4.123}$$

The test quantity T^2 becomes:

$$T^2 = \frac{n_1 n_2}{n_1 + n_2} D^2 \tag{4.124}$$

which can be compared to the F-distribution (B.1).

Equal multi-sample mean The multi-sample test for equal means tests the following:

$$\begin{array}{ll} \mathrm{H}_0: & \mu_1 = \mu_2 = \ldots = \mu_{n_g} \\ \mathrm{H}_1: & \exists g_1, g_2\colon\ \mu_{g_1} \neq \mu_{g_2} \end{array} \tag{4.125}$$

A consistent measurement for this hypothesis is the likelihood ratio test of Wilk's lambda Λ — or by an other name Anderson's U:

$$\Lambda = \frac{\det(\mathbf{W})}{\det(\mathbf{T})} \tag{4.126}$$

The $\mathbf{W}$ is the *w*ithin the groups variance:

$$\mathbf{W}_g = \sum_{p_g}^{n_{p_g}} \left(\mathbf{h}_{gp_g} - \bar{\mathbf{h}}_g\right)\left(\mathbf{h}_{gp_g} - \bar{\mathbf{h}}_g\right)^\top \tag{4.127}$$

$$\mathbf{W} = \sum_{g}^{n_g} \mathbf{W}_g \tag{4.128}$$

were n_g is the number of groups, and n_{p_g} is the number of patterns within the group g.

The $\mathbf{T}$ is the *t*otalmatrix:

$$\mathbf{T} = \sum_{g=1}^{n_g} \sum_{p_g=1}^{n_{p_g}} \left(\mathbf{h}_{gp_g} - \bar{\bar{\mathbf{h}}}\right)\left(\mathbf{h}_{gp_g} - \bar{\bar{\mathbf{h}}}\right)^\top \tag{4.129}$$

Equal multi-sample dispersion The hypothesis for equal multi-sample dispersion is constructed the same way as the hypothesis for the multi-sample mean:

$$\begin{array}{ll} \mathrm{H}_0: & \mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = \ldots = \mathbf{\Sigma}_{n_g} \\ \mathrm{H}_1: & \exists g_1, g_2\colon\ \mathbf{\Sigma}_{g_1} \neq \mathbf{\Sigma}_{g_2} \end{array} \tag{4.130}$$

There are different ways of testing this hypothesis. A ordinary likelihood ratio test [52] compares the volumes of the hyperellipsoides of the within and total dispersion:

$$M = \sum_g n_g \log\left(\det\left(\mathbf{S}_g^{-1}\mathbf{S}\right)\right) \tag{4.131}$$

The $\mathbf{S}_g$ and $\mathbf{S}$ are the maximum likelihood estimates of the dispersions, being: $\mathbf{S}_g = \frac{1}{n_{pg}}\mathbf{W}_g$ and $\mathbf{S} = \frac{1}{n_p}\mathbf{W}$ respectively.

An other test is the **Box's M test**, where the estimate of the total dispersion is not biased [52]:

$$M_{box} = \gamma \sum_g (n_g - 1) \log\left(\det\left(\left(\mathbf{S}_g^{\mathrm{u}}\right)^{-1} \mathbf{S}^{\mathrm{u}}\right)\right) \tag{4.132}$$

where the dispersions are the unbiased versions:

$$\mathbf{S}^{\mathrm{u}} = \frac{n_p}{n_p - n_g}\mathbf{S} \tag{4.133}$$

$$\mathbf{S}_g^{\mathrm{u}} = \frac{n_{p_g}}{n_{p_g} - 1}\mathbf{S}$$

and γ is:

$$\gamma = 1 - \frac{2n_h^2 + 3n_h - 1}{6\left(n_h + 1\right)\left(n_g - 1\right)} \left(\sum_g^{n_g} \frac{1}{n_{p_g} - 1} - \frac{1}{n_p - n_g} \right) \tag{4.134}$$

The 2 test sizes are asymptotic χ^2-distributed:

$$M \sim\in \chi^2_{\frac{1}{2}n_h(n_h+1)(n_g-1)} \tag{4.135}$$

This is a particular good approximation if there are many patterns n_p and few clusters n_g and dimensions n_h.

If the representation in the hidden units' space is directional the Box tests are in troubles. We may project the hidden units on principal component and take only those projections that are different from zero (meaning relative high), along to the test. If the space is unidirectional the unbiased Box's M test ends up with up with being the Bartlett test:

$$M_{bartlett} = n_p \log\left(s^2\right) - \sum_g n_g \log\left(s_g^2\right) \tag{4.136}$$

The variances is still the biased versions, and the test size is:

$$M \sim\in \chi^2_{n_g - 1} \tag{4.137}$$

Instead of principal components, just one hidden unit could be piped to the test.

A distribution free dispersion test is the Sen-Puri test.

4.7.3 Directionality of the state space

The directionality of the representation in the state space can be investigated by examining the principal components of the hidden units value. If the inputs project the same way — ignoring the sign — on all the hidden units, the patterns would lie on a line in the hidden space. When projected on the principal component only the first one would have any significant size. So to get a value of the (line-) directionality of the representation in the hidden units' space we could divide the first principal component with the second:

$$D = \frac{PC_1\left(\mathbf{H}\mathbf{H}^\top\right)}{PC_2\left(\mathbf{H}\mathbf{H}^\top\right)} \tag{4.138}$$

4.8 Comparison between maps

As stated before the values that come from the saliency map (and other analysis techniques) are not universal. It is a value that depends on a particular training and neural network. We may however say that the value should be in relation to other analysis value, such that we could get from one to the other by a monotonic transformation. Monotonic equivalence can be measured with rank correlations (B.2).

The rank correlation may however be too pessimistic, and they may emphasize the ordering too much. Among the correlations that do not emphasize the ordering so much is the ordinary correlation coefficient:

$$\rho = \frac{\sigma_{12}^2}{\sigma_1 \sigma_2} \tag{4.139}$$

By some called the Pearson correlation coefficient. This is however rather metric.

An other correlation is the Lin concordance correlation [38] used for reproducibility:

$$\rho_c = \frac{2\rho\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2 + (\mu_1 - \mu_2)^2} \tag{4.140}$$

As the saliency values vary depending on the specific training set and neural network this measure can not be used.

One may argue that the comparison between maps rather should show if there is difference between the blobs (The small areas with the highest saliency): A qualitative description accounting new blobs or blobs with an other form — a higher contrast.

Chapter 5

Visualization

5.1 3D Visualization

3 dimensional visualization can be approached in two different ways: **Volume graphics** and **Surface graphics** [31]. The two methods have the 2D counter parts in vector graphics and raster graphics. The voxel based graphics is more computational expensive and require more storage than surface graphics.

In volume graphics a 3D matrix is storing all voxel values. A method for obtaining the 2D projection of the volume is through *ray tracing*, which comes in two variants [36]: *object space methods*, where the volume is resampled along the rays, and *image space methods*, where the volume first is transformed to be axis aligned before the ray tracing.

The ray in the ray tracing collects voxel values along its line to build up the 2D projection pixel intensity and color. The transformation from voxel values to intensity needs not to be linearly: it is more often a soft threshold function (figure 5.1).

Surface graphics shows an isosurface in the volume. The representation of the surface can be parametric or quadratic, but this is seldom seen. Most 3D surface graphics systems use the polygon as the graphical primitive.

The usual method of constructing the polygons from a volume is the *marching cubes* of Lorensen [46]. This method constructs subvoxel polygons by looking at the value of vertices from the little cube span by 8 neighboring voxels. An other method is the *cubrille* approach, where every voxels with a higher value than the isosurface is boxed with a cube — a cuberille [27]. Joining cuberilles can be melted together to form a bigger cuberille. This approach leads an octree, with a spatial spectrum representation.

The advantage of surface graphics over volume graphics is the smaller memory and processing requirements, and the more immediate concept of *objects*. The disadvantage is the static isosurface and the missing interior [31]. The static isosurface can be overcomed by interactive — data flow — system such as AVS and IRIS Explorer.

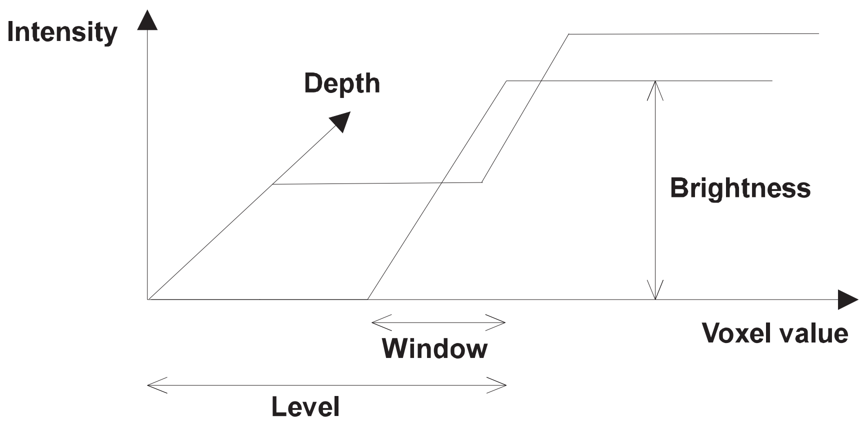


Figure 5.1: One of the threshold function of the `volren` program. Along with the three parameters, level, window and brightness comes a fourth: the transparency. It does not directly translate into an intensity value, but rather indexes into a color table.

5.2 Visualization of Multidimensional Data

In the case of the state space with the neural network hidden units the number of dimensions may be higher than what can be visualized in 2D or 3D displays. There exist a number of methods for squeezing the number of dimensions [18].

We can simply select 2 or 3 axis where the range or variance is the largest. We might also select non-axis dimensions. With the variance used, it brings us to **principal component analysis**. PCA has already been accounted for in section 4.5. Gross [18] mentions a few others: cluster analysis, vector quantization, Voronoi tessellation and scattered data modeling. Neural network analysis can be used for these purposes.

Instead of reducing the dimensions the method of *glyphs* can be used. Glyphs are small graphical objects whose position, shape, color, texture, size, orientation reflects the data [14]. [52] shows a small example where 4D data has been presented in 2D: The single points in the 2D space have each been given two "legs". The size of the legs represents the value of the point in the last two dimensions.

A completely different method is **harmonic curves** [52]: It uses a data point $\mathbf{x}^p$ in an equation 5.1, which then come out with a curve with an independent variable t:

$$f_{\mathbf{x}^p}(t) = \frac{1}{\sqrt{2}} x_1^p + x_2^p \sin(t) + x_3^p \cos(t) + x_4^p \sin(2t) + x_5^p \cos(2t) + \dots \tag{5.1}$$

Data points belonging to the same cluster follow the same path along the t-axis, having the same harmonics and phase. By viewing this 2D plot, clusters can be determined by eye.

5.3 Depth Cues

The perception of depth not just arises from binocular disparity — two-eyed vision, but is also dependent on what we can see with just one eye. If the viewer is a few meters from the object most of the depth information actually comes from the monocular — one-eyed — vision.

Table 5.1 shows some normally considered depth cues. Artificial depth cues can be constructed by lighting and coloration of the objects according to their depth.

Group	Depth Cues	Explanation
Dynamic monocular	Motion parallax	Change in retina image when the object or viewer are moved
Static monocular	Relativ size	Nearer objects are bigger that distant
	Superposition	Distant objects are occluded by nearer
	Relative height	When standing on the ”ground” distant objects are higher in the visual field
	Texture gradient	The cornfield depth cue: Elements are packed closer together
	Perspective	Lines convergence in a point
Binocular	Binocular disparity	Difference in the retina images
	Binocular parallax	The angle of the direction of the eyes

Table 5.1: Depth cues

In science the objects, that are usually visualized, are rather abstract, were the absolute size of the object is hard to judge. This diminishes the depth cue through the relative size.

The superposition cue is unfortunately struggling with the need to see so many of the objects as possible. Consider a surface made of polygons and an other made of lines: In the first case one cannot see all of the surface, in the second case it is hard to judge the depth. An other example is transparency: The more transparent, the objects become the less depth cues they get.

There are 2 methods for projecting 3D objects onto a 2D viewing surface [26]. Either by parallel lines as with a **orthographical** camera or along converging lines as with a **perspective** camera. The orthographical camera preserves the dimension of the object along the depth axis, while the perspective camera destroys it. The advantage with the perspective camera is that it has better depth cues. The orthographical camera is best used for axis parallel views — with brain imaging: transversal, sagittal and coronal — where a precise 2D coordinate determination can be obtained, while the perspective is best suited for free-hand fly around views.

Chapter 6

Toy Brain Analysis

This chapter will contain a brief discussion of a toy brain analysis. It will compare the saliency map with the subtraction method to obtain an understanding for the monotonical relationship between the two.

6.1 Brain Activation

A toy brain with the size of 16×16 was given a paradigm related activation in the area around the 12. pixel (figure 6.1). There was three states: Rest – activation – rest. An activation not related to the paradigm was given around the 5. pixel. The amplitude of this activation was as the same magnitude as the paradigm related signal in the last rest state, and half the size in the activation state. The image was blurred with spatial and temporal independent noise having the standard deviation of 1. Every state contained 10 scan.

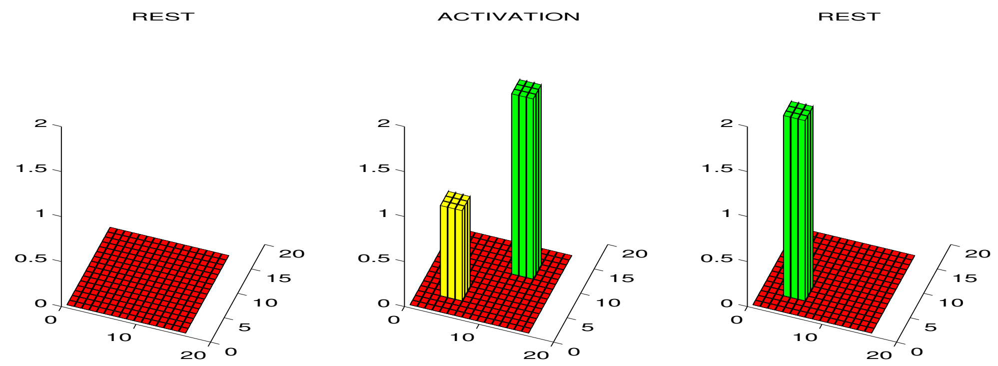


Figure 6.1: An activation of a toy brain.

6.2 Analysis

With spatial independent noise we would not expect global methods (p. 27) to be better than the voxel-based methods.

The subtraction method (p. 28) should cancel out the paradigm unrelated activation, because the mean value of this activation in the subtraction images is zero. This is the case one sees when viewing figure 6.2.

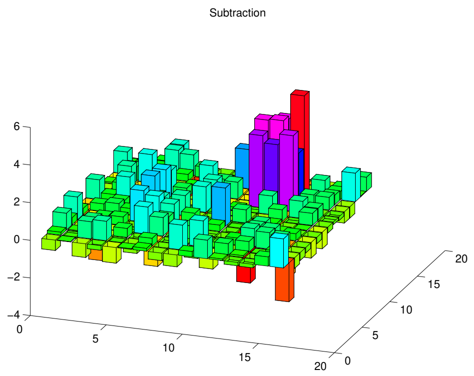


Figure 6.2: Subtraction map.

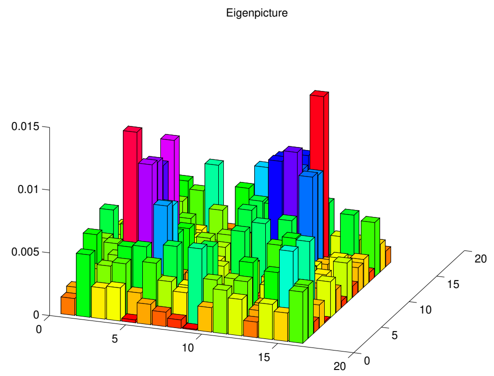


Figure 6.3: Absolute value of the most paradigm relevant principal component.

As discussed earlier (p. 60) the relationship between the saliency map on the one side and the subtraction and eigenpicture on the other, is that the saliency map is "squared". Comparing figure 6.2 and 6.3 with 6.4 and 6.5 the signal to noise ratio is approximated the same if the square root is taken for the saliency maps.

From figure 6.3 we see that the paradigm related and the non-paradigm related activation mixes up in the same principal component. The linear saliency, acquired from an optimization with the normal equation (page 48), cannot distinguish the two activations, while the neural network model is able to extract the paradigm related activation area.

The saliency from the neural network model carries negative saliencies. These are especially situated where the most important principal component has the highest negative values.

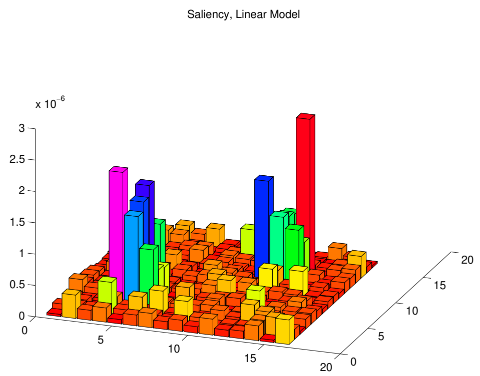


Figure 6.4: Saliency map of a fully connected linear model trained to the square errorfunction.

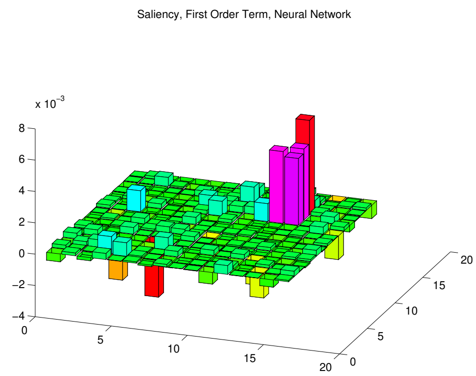


Figure 6.5: First order term saliency from a neural network trained with the entropic errorfunction.

Chapter 7

Sequential Finger Opposition

The test data chosen for an analysis was fMRI data from a study of Woods in Massachusetts General Hospital. The task was sequential finger opposition performed with the left hand. There were original 4 subjects involved, but I used only 1. The subject took part in 8 runs each with 72 volume scans. Thus the number of scans from just one subject is 576. The scanning was performed every 2.5 second, thus one run lasted 3

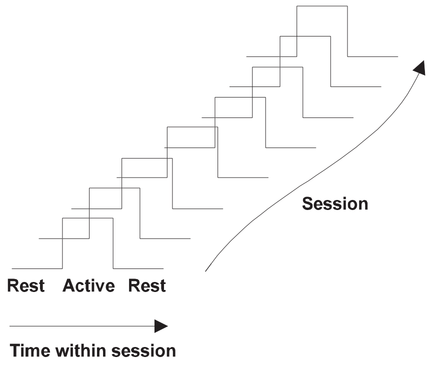


Figure 7.1: SFO protokol

minutes. During the first 24 scans — the first minute — the subject was at "rest" with no sound in the ear, eyes closed (lights off), and performing no task. The next 24 scans were taken with the subject at "activation" with a 1 Hz metronome sound in the ear and the fingers oppositioning at the same rate. The last 24 scans were again taken at rest. The volume was constructed from 14 64×64-sized slices, but resampled to 20 slices. The data was saved using 2 byte unsigned integer. The data I worked with was further modified through an alignment. With 20 slices the number of voxels becomes 81920. Multiplying this with the number of scans, the number of elements in the data matrix becomes almost 42M. If the complete datamatrix is to be stored with 8-byte floating points, we end up with 340Mb.

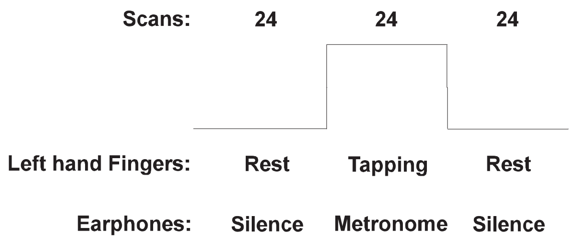


Figure 7.2: One run of the SFO data

We may expect from already known functional anatomy (p. 17, figure 2.9), that a blob in the *right* hemispheric motor area should light up in the functional analysis. Whether the metronome should be identifiable in the functional image is a question. The noise in the scanner room may outweigh the small tick.

”Parameter”	**”Value”**
Scanning site	MGH-NMR, Woods
Data set	”SFO ... aligned”
Subject	Healthy female
Modality	fMRI, 1.5 Tesla
Task	Left hand, Fingers-to-thumb
Dimensions	$64 \times 64 \times 14$ resampled to $64 \times 64 \times 20$
Viewpoint Matching	Yes?
Template Matching	Talairach box
Voxel size	3.125 mm $\times$ 3.125 mm $\times$ 8 mm
Number of runs	8
Run States	Rest — Activation — Rest
Number of scans within a run	24+24+24
Time between scans (T_R)	2.5 sec

Table 7.1: Data for the SFO study.

7.1 Method documentation

The preprocessing of the data was carried out in accordance with figure 4.4.

First the masking was performed using all the scans. For every scan the 30% amplitude level of the maximum value was found. Every voxel above this level was masked through. All image masks were then AND'ed forming the final mask. This masking brought the number of voxel down from 81920 to 15625: More than 80% of the voxels discarded!

To measure the generalization of the neural network empirically, the data was split in test sets and a training set. The test sets where constructed so they hopefully would contain every kind of noise: interrun noise, intrarun noise, "hemodynamic response" noise, and "learning" noise (different behavior at the beginning and the end of the scanning session). All though it could be nice to do some "leave one run out" cross validation scheme, the amount of data made the split of the basis/non-basis sets rather static (table 7.2).

No.	Where	Scans	In the basis
1	2. run	72 (58)	No
2	5. run	72 (58)	No
3	Rise flange of 6. run	$8 = 4 + 4$	No
4	Fall flange of 7. run	$8 = 4 + 4$	No
5	Randomly chosen among the rest	24	Yes
6	Randomly chosen among the rest	24	Yes

Table 7.2: The test sets.

Whether it is statistically correct to have all test sets along in the SVD-PCA basis is not clear. One could argue that generalization is the ability to predict with a completely new scan (with the same subject). In this case the test sets should not be in the basis. On the other hand one could argue that the overfitting only takes place in the neural network, and what happens before this, of split between sets, is of no importance. Then, using more scans for the basis computation, would span the paradigm space better.

Some experiments showed that the hemodynamic response infested the post flange scans, especially the deactivation flange (compare with p. 19). The robust statistic (p. 40) could not ignore the large amount of scans, who carried the hemodynamic response. One could of course try to model this behavior. But this fMRI scanning produced enough scans, so that the number of pattern for the neural network would be sufficient even with a bag of scans omitted: 4 post[1] activation flange scans were omitted from every run, and 10 post deactivation flange scans were left out. This brought the number of scans within a run down to 58.

The fMRI data can be split in many ways: Total numbering, Numbering with a run, run, rest/activation, first rest / activation / second rest, training/test set, hemodynamic mask split. The task of handling these different splittings must not be ignored. Methods to handle the splittings are through a design-matrix-like data structure (see SPM p. 28 for the design matrix) and the masking matrix structure 4.3.1.

After the masking and the splitting of the data, a mean for the training set was calculated to center the training set for the SVD-PCA. The SVD-PCA projected scans were scaled so their range would be less than –1 to +1.

[1] scans just after the shift of state

The weight initialization was such that the weight size was normalized with the square root of the fan-in, and the extremum of the input (for the hidden weights).

The entropic errorfunction was selected with +1/–1 marking rest and activation, respectively. As regularization was the usually square weight decay chosen. The weight decay parameter was varied.

The weight optimization was a hybrid second order method with the structural approximation of the full Hessian for the output weights and diagonal approximation for the hidden weights (b approximation in figure 4.5). The computational complexity for the full Hessian would rise more drastically than the diagonal approximation as the number of weights under optimization is increased.

With and without gauss-newton approximation was tried.

When pruning was done, the neural network was first trained with 1000 or more of weight optimization epochs, then it was OBD-pruned with the ceil of 5% of the remaining weights. After that retrained with 100 or more weight optimization epochs, before the next 5% pruning.

7.2 Principal Component Analysis Results

7.2.1 Dispersion Matrix

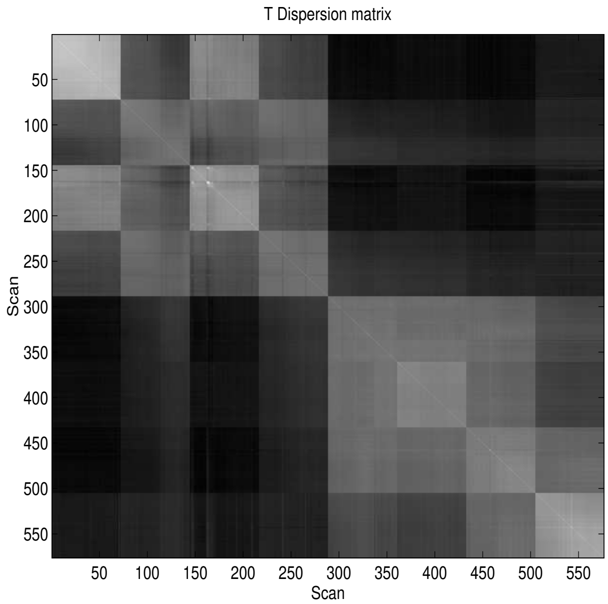


Figure 7.3: The **T** dispersion matrix.

The dispersion from the total data (no test taken out) can be seen in figure 7.3. The dispersion matrix shows a strong intrarun dependence (the 8×8 chessboard), and no apparent variation to the paradigm (no $(3 \times 8) \times (3 \times 8)$ chessboard). This is in accordance with what could be expected: The largest variation does not come from paradigm, but rather from what for the functional analysis point of view is noise. Therefore the effect of ignoring some principal components can not be performed in the usually way [52], where the sizes of the eigenvalues are compared, and those principal components containing for example 90% of the variation, are kept.

A surprising dependency shows up with the 2×2 chessboard: The session seems to be divided into two groups, each having a good deal of within correlation. When reading the scanning journal a little closer, it turns up that the 8 runs were not performed in a row. There was actually 2 different runs: The one used in this study — called A in the journal — with 3 states (rar — Rest–activation–rest), and an other called B, having only 20 seconds between the rest-activation shifts (rarararar). The runs in the session were in this order: AAAABAAAAB. That is, one of the B runs was squeezed in between the A runs. The lag must have caused the difference in correlation between the first and the last runs. The conclusion is that, not only do we have intersubject, intersession and interrun variation, but also intrasession-interrun variation due to the time the session takes.

As far as I can read, the journal does not say anything about the time between the runs. But judging from the dispersion matrix there seems to be no run "overlapping", that is, the last scans in a run do not resembles the first scans in the next run.

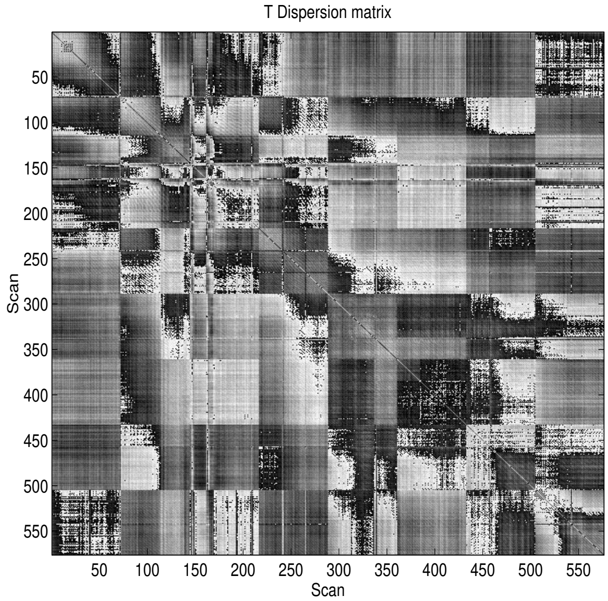


Figure 7.4: The $\mathbf{T}$ dispersion matrix with the colormap manipulated.

There is a disturbance in the session: Some scans show a striking dissimilarity with the other scans in the same run. This is better seen in figure 7.4 where an approximately 10 toothed saw has been used as a grayscaled colormap. The differing scans are: around the shift of the second and third run, especially in the middle of the first state in the third, and at the shifts of state in the fourth run. What these constitute I do not know. The scanning journal does not carry any clues.

7.2.2 Spectrum

That the PCA spectrum is monotonic decreasing, as in figure 7.5, is of no surprise. In the same figure is also shown the spectrum after the hemodynamic response mask has been applied. The spectrum do not changes much after this operation. The figure 7.6 shows that the test set projections do indeed not have a large value in the higher principal components.

Figure 7.7 shows the result of a preliminary training with the neural network sized $(1+1) \times (3+1) \times 1$. The single input is swept through all the principal components with the target of predicting the paradigm. The decrease of the errorfunction is calculated after 25 epochs. This method measures the importance for the paradigm of a stand-alone principal component, without using crossdependencies to other components.

Comparing the "total" spectrum (figure 7.5) with paradigm related spectrum (figure 7.7) it is evident that the largest variation is not related to the paradigm. The first

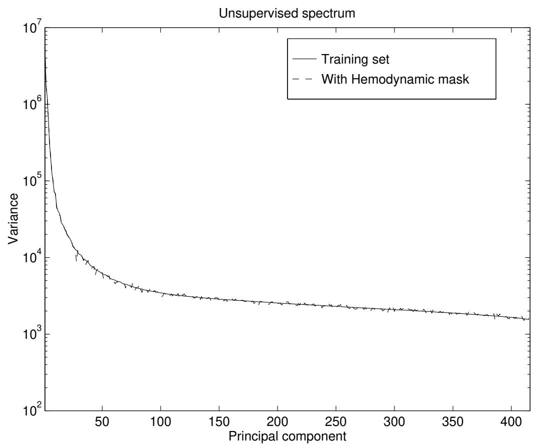


Figure 7.5: The "unsupervised" spectrum.

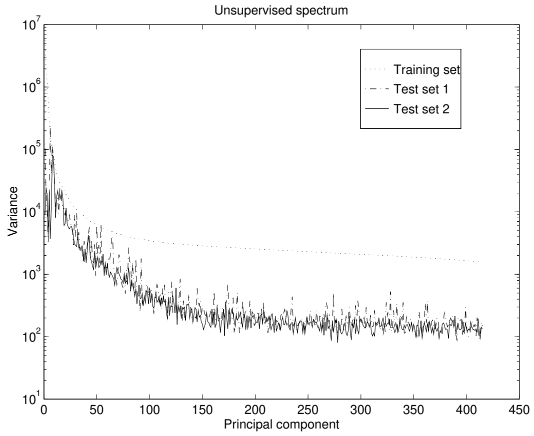


Figure 7.6: The unsupervised spectrum with test sets.

important components are coming after the number of runs. A multi subject study [42] has come to the result, that the most paradigm relevant principal component is the one immediately after the degrees of freedom with respect to the number of subjects. In our particular case the most important principal component is not the one immediately after what could be imaged to be the largest variation: the interrun variation. In the basis there are 6 runs, and the first paradigm relevant principal component is in the 7. This is however not the most important: The 15. principal component is the most important.

Figure 7.7 shows the difference between neural network testing and ordinary second moment statistics testing: The 160. principal component can explain some of the paradigm judging from the upper panel. With ordinary statistics we would calculate some level for the decrease of the errorfunction. Everything above that line would be significant: either a large random error or paradigm related. We would not be able to settle the question. The neural network technique is to use one or more test sets. The results from the first and second test set (table 7.2) show that the neural network has overfitted: There is an *increase* of their errors.

The lower panel of figure 7.7 shows that some components have a reverse behavior with the two test sets: The two test set is projected differently on the principal components.

7.2.3 Eigenpictures

The eigenpictures shows where the principal components look for information in the voxel space.

The images in this section have been shown as projected maps onto the cortex contours of the Talairach atlas (p. 16) found in the SPM program (p. 28). Red is to be considered most positive, and blue most negative. The scans were fitted to the Talairach contours by eye.

From the spectrum of figure 7.7 it is known that the first principal components do not

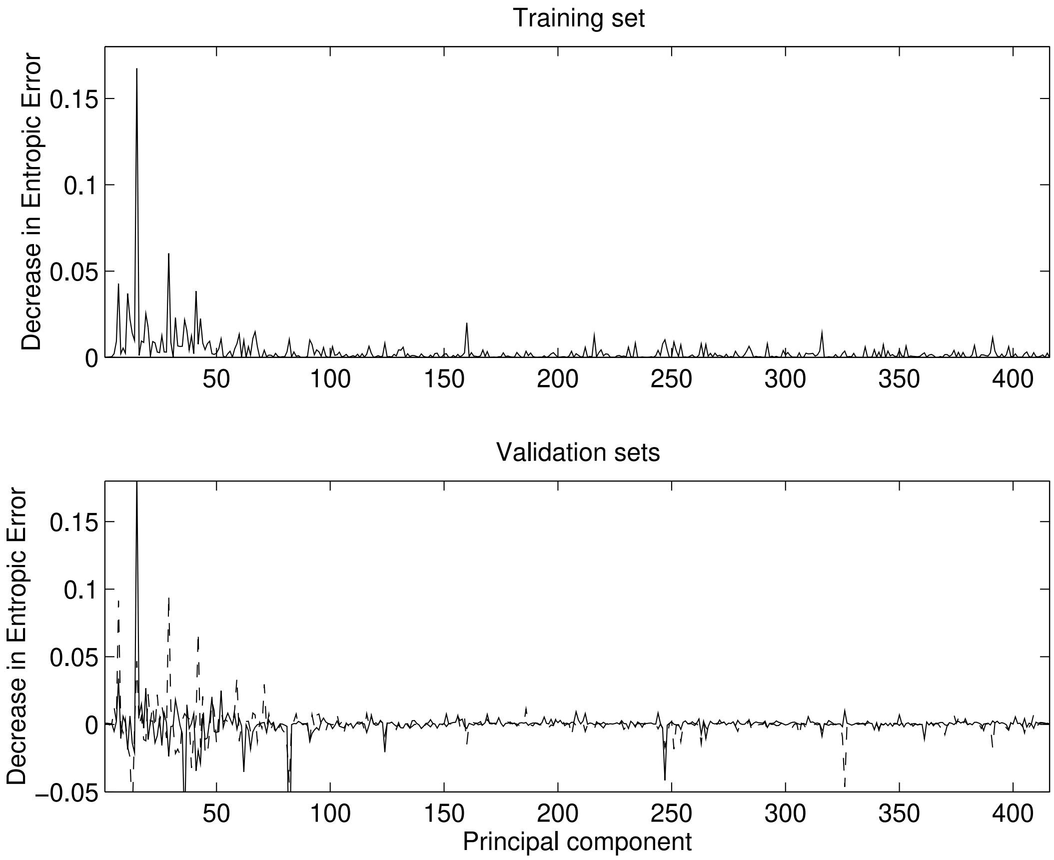


Figure 7.7: The "supervised" spectrum. The upper panel is for the training set and the lower panel is for the 1. and 2. test set (table 7.2)

contain the paradigm. An eigenpicture from such a component can be seen in figure 7.8. The sagittal view clearly shows that the first principal component explains a sort of chess pattern in the scans. This is perhaps coming from the resampling of the scans.

The 15. principal component is known to explain the paradigm better than any other stand alone principal component. Figure 7.9 shows that it projects onto an area in the right hemisphere that is in accordance with the motor and somatosensory area of Penfield (figures 2.8 and 2.9). This is seen most easily in the coronal projection, with the blue spot (remembering that the sign of a principal component is not important).

The other areas in this principal component *might* be noise, that is just having the same kind of variation as the paradigm relevant area.

Whereas the low principal components with paradigm irrelevance show large areas with the approximately same value, the highest is just representing pure noise that is scattered all over the image. Figure 7.10 shows the very last principal component: The spatial correlation is very low.

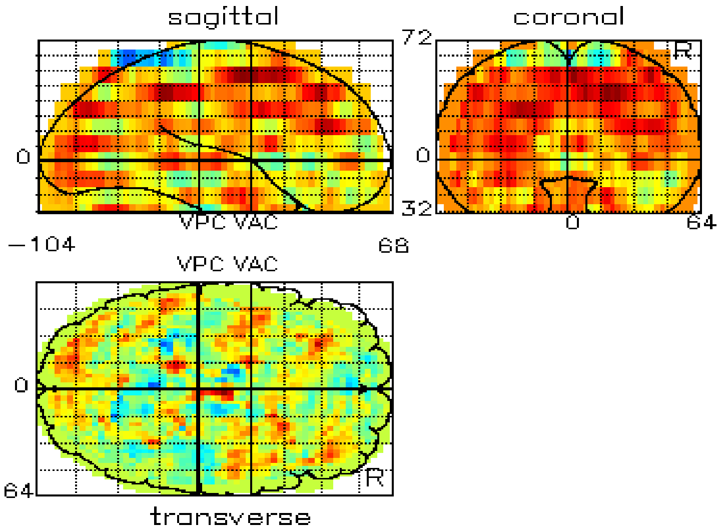


Figure 7.8: The eigenpicture with the first principal component.

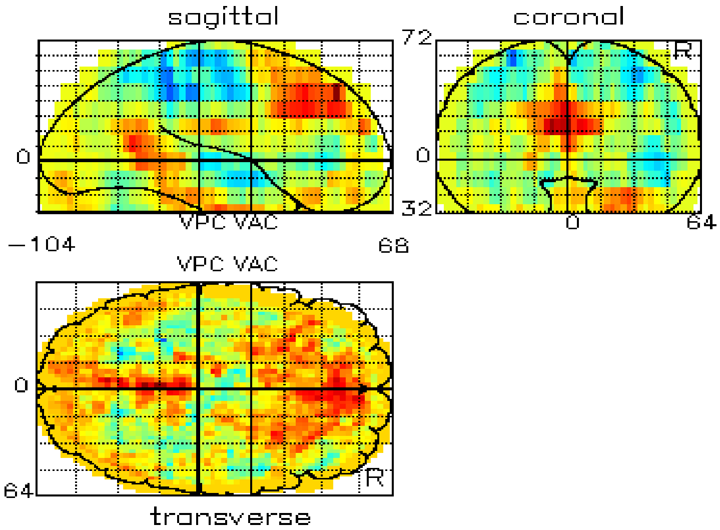


Figure 7.9: The eigenpicture with the 15. principal component: The one known from the PCA sweep to contain a lot of the paradigm

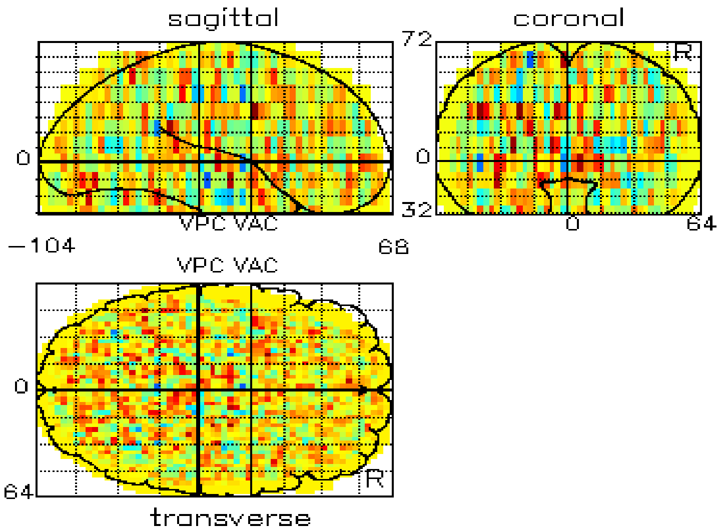


Figure 7.10: The eigenpicture with the last principal component: Pure noise.

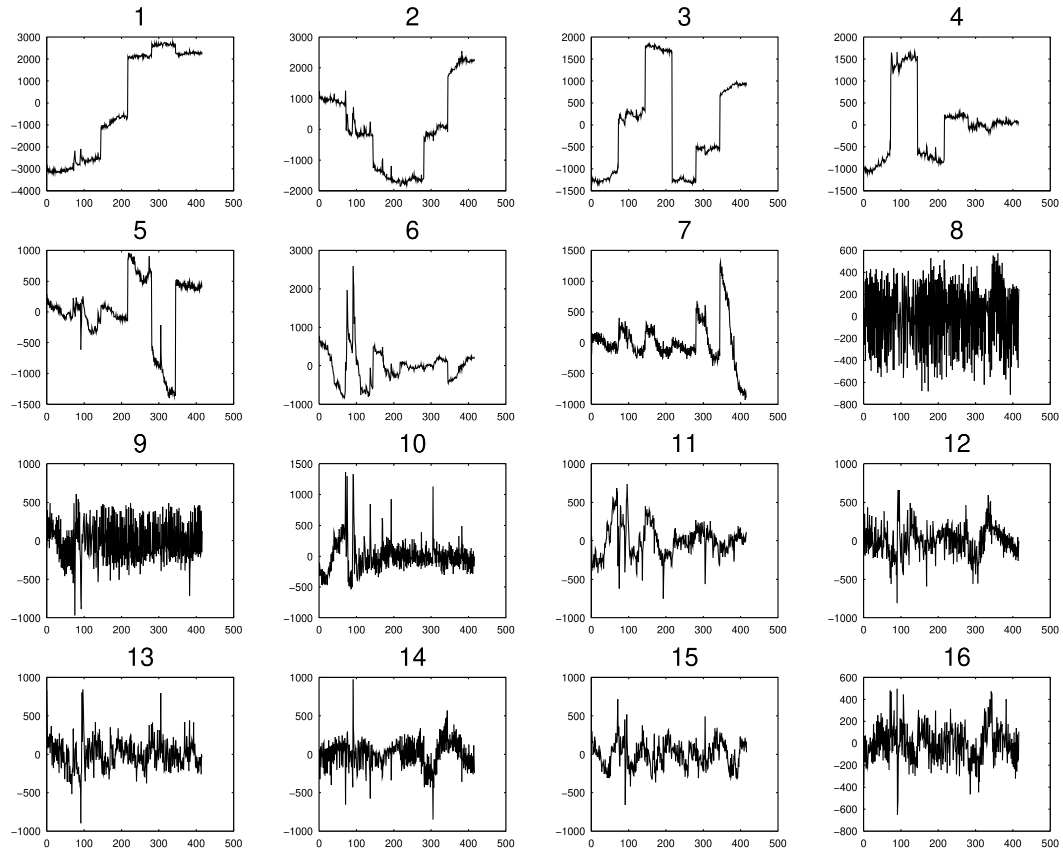


Figure 7.11: The PCA projection onto the neural network input. The basis contains 6 runs.

7.3 Neural Network Analysis Results

7.3.1 Optimization

The heterogenous errors in the data — due to interrun noise and intrarun-interscan noise — causes the test error to behave rather irregularly with overfitting: Usually the 1. and 2. test sets[2] have errors that are higher than the 5. and 6. test sets, firstly because they are not in the basis — they may span differently —, and secondly: test set 5 and 6 do not contain interrun noise that is not already in the training set. This behavior can be studied when overfitting with the lower principal components (that is up till 50). But if all principal components are piped to the neural network the sizes of the test errors reverses: The test set 5 and 6 get the highest errors. This might be explained in the following way: The highest principal components span the intrarun noise of the in-the-basis test sets. Test set 5 and 6 have a significant contribution on these principal components, while the out-of-basis test sets do not have a significant span in these components. As the pruning takes care of removing the principal components which contain little information, the errors of the test set 5 and 6 will cross the errors of test set 1 and 2.

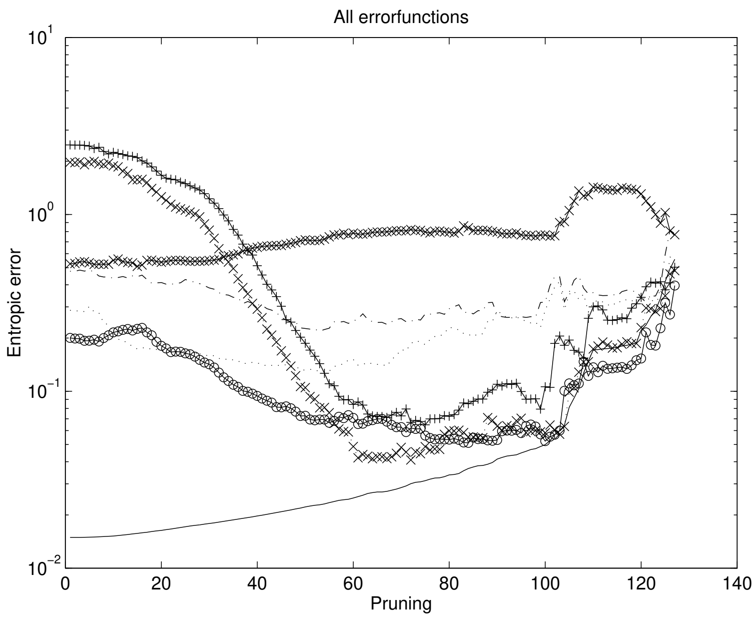


Figure 7.12: Behavour of the errors of the test sets. — Learning set, • Test set 1, •– Test set 2, ∘ Test set 3 (rise flange), × Test set 4 (fall flange), + Test set 5, × Test set 6

The test test set at the fall flange (test set 4 table 7.2) will have a reverse type of behavior. This test set is right at the deactivation flange with the hemodynamic response.

[2]refer to table 7.2 for the set splitting

As noted in section 7.1, I was not able to obtain any good results with training and test sets that contained the hemodynamic response. These were masked out. Test set 4 shows the effect the hemodynamic response has on the neural network: The neural network will correctly(!) misclassify these first post flange scans, and due to the entropic errorfunction penalizing total misclassifications hard, the error will be large. In fact the best neural network (with respect to intrarun generalization) will be the one with the largest error on the flange test test set.

The test error can be studied in figure 7.12, showing the error during pruning from a large network sized $(416+1) \times (20+1) \times 1$, regularized heavily. The 2 in-basis intrarun test sets starts out with the highest error, then as the pruning proceeds taking a deep duck, crossing the out-of-basis test sets.

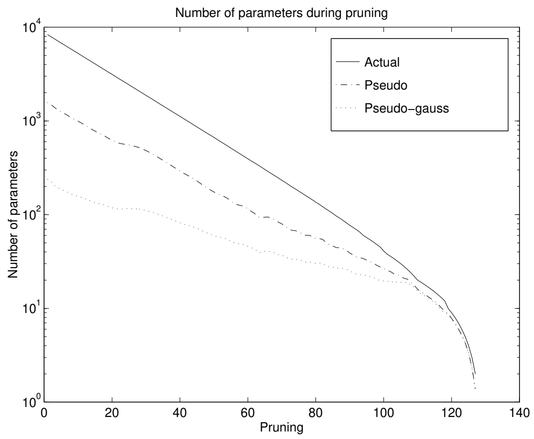


Figure 7.13: The effective number of parameters during pruning.

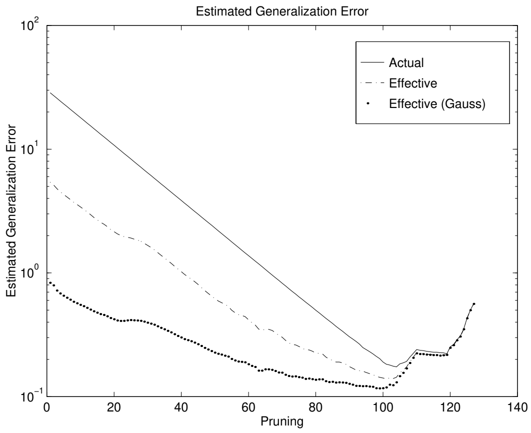


Figure 7.14: Final Prediction Error estimate.

The neural network is very large, but also regularized very hard, being able to bring number of parameters down to under the number of patterns, watching figures 7.13 and 7.14. The generalization error estimates do not fit with the test set 1 and 2 minimum, but rather resemble the minimum of the test set 5 and 6.

The figures 7.13 and 7.14 also show a difference between the generalization error estimates and effective number of parameter estimates, when these are calculated with and without the gauss approximation of the second order derivative. This is only for the estimates early in the pruning session. The estimate using the pseudo-newton is lower than than the estimate using the pseudo-gauss-newton derivative. This is dependent on the regularization parameter. Their minima are approximately the same. The generalization error estimates do usually not perform very well with a model that is much too large, so the difference between the pseudo-gauss-newton and the pseudo-newton estimate might not be of any importance.

The behavour of the test set was consistent through the weight decay space. Figures 7.15 and 7.16 show a neural network sized $(200+1) \times (2+1) \times 1$, trained with a weight decay of 10^{-7}.

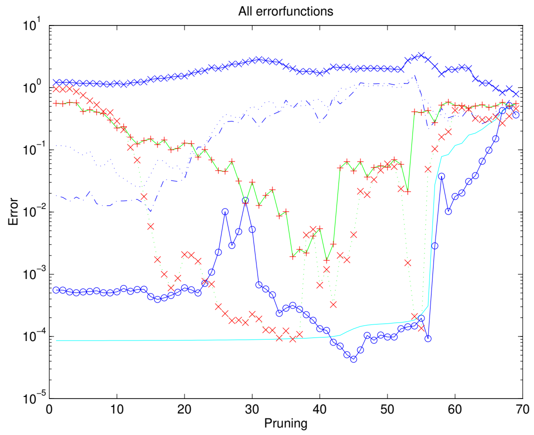


Figure 7.15: The errorfunctions. Same legend as figure 7.12

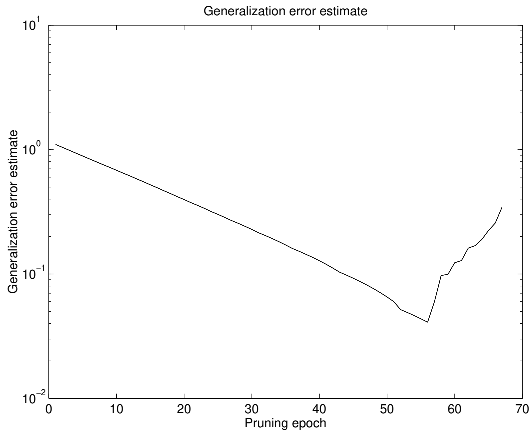


Figure 7.16: Generalization estimates

The size of the neural network with the optimal architecture, chosen with the intrarun test set, depend not on the starting architecture or much on the regularization parameter. For the intrarun generalization the effective number of parameters is around 20. For the interrun out-of-basis generalization the architecture is 10 times greater, 200 effective parameters or less.

The generalization behavior is indeed different than what usually is observed. Training and test patterns that are further away in the input space usually require a smaller network to show good generalization. Here it is not the case. The test sets have little span in the higher principal components, and the projection onto these axes is rather low.

Although there are little span in the higher principal component looking at the test sets (figure 7.6), it proved to be important not to leave to many principal components out. With 50 principal component I was not able to have generalization into the interrun out-of-basis test sets. The first 150–200 principal component was important for this generalization.

It was also important to have the principal component normalized, so their range would be of equal size. Neural networks that were just feed the principal component projection could not generalize on the test sets.

7.3.2 Mean and Median of Patterns

The skewness of the entropic costfunction can be examined with a comparison of the mean against the median (p. 40). Viewing figure 7.17 one sees that the mean value of the entropy for every set — learning set and test set — always lies above the median (the H/L median in this case). This means that the entropy has skewness to larger values. For the square errorfunction there should be no skewness as it is trained with the prior of gaussian noise.

The upper skewness comes when some scans are being totally misclassified. These errors are being punished hard by the entropic error, — more than the right classifications

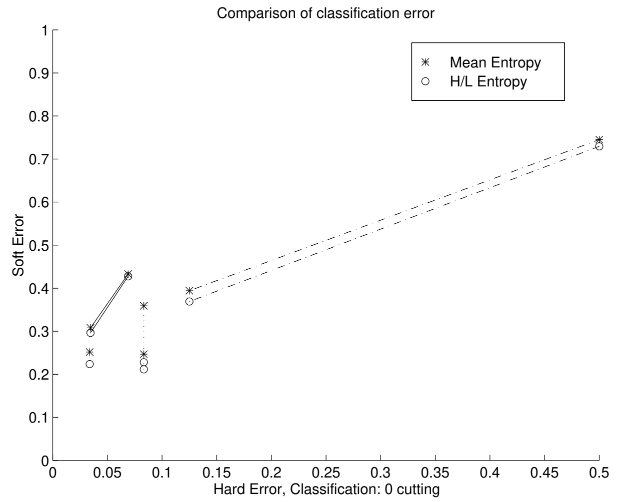


Figure 7.17: Comparison of the classification error. Mean entropy is arithmetic mean over the patterns. H/L entropy is the Hodge/Leeman estimated mean. The solid line is connecting the first and second test sets. The dash-dot line is for the flange test sets. The dot-dot line is for the in-basis test sets. The stand-alone dot is for the learning set.

can outbalance.

Figure 7.17 shows the "soft" errors (median and mean entropy) as a function of the "hard" classification errors (sign function). In this particular case with this size of the test sets, there is not a monotonic relationship between the hard and the soft classification error. The hard cutting classification error should be considered *the* measure for the error.

7.3.3 Weight Decay

To further investigate the generalization errors for the different test sets, a batch of training sessions was made with fully connected neural network, varying the weight decay parameter.

The figures 7.18 and 7.19 show that there is a significant difference in the generalization estimates with the pseudo-gauss-newton derivative and the pseudo-newton derivation not using the gauss approximation. This difference is happening where the neural network is nonlinear, while in other states there is not a very large variation between the two.

One should imaging from the theory that the pseudo-newton estimate is the best since it take the most terms along. But comparing the algebraic estimates of the generalization error with the empirical estimates — test errors, it looks as if the pseudo-gauss-newton estimate is better. The pseudo-newton is estimating for some weight decay parameters very wrong. This is though for the fully connected neural network: Usually the algebraic generalization error estimates do not work with a too overparametrized model.

The area where the weight decay parameter of both hidden and output weights are large will cause all the weights of the neural network to come close to zero, — except

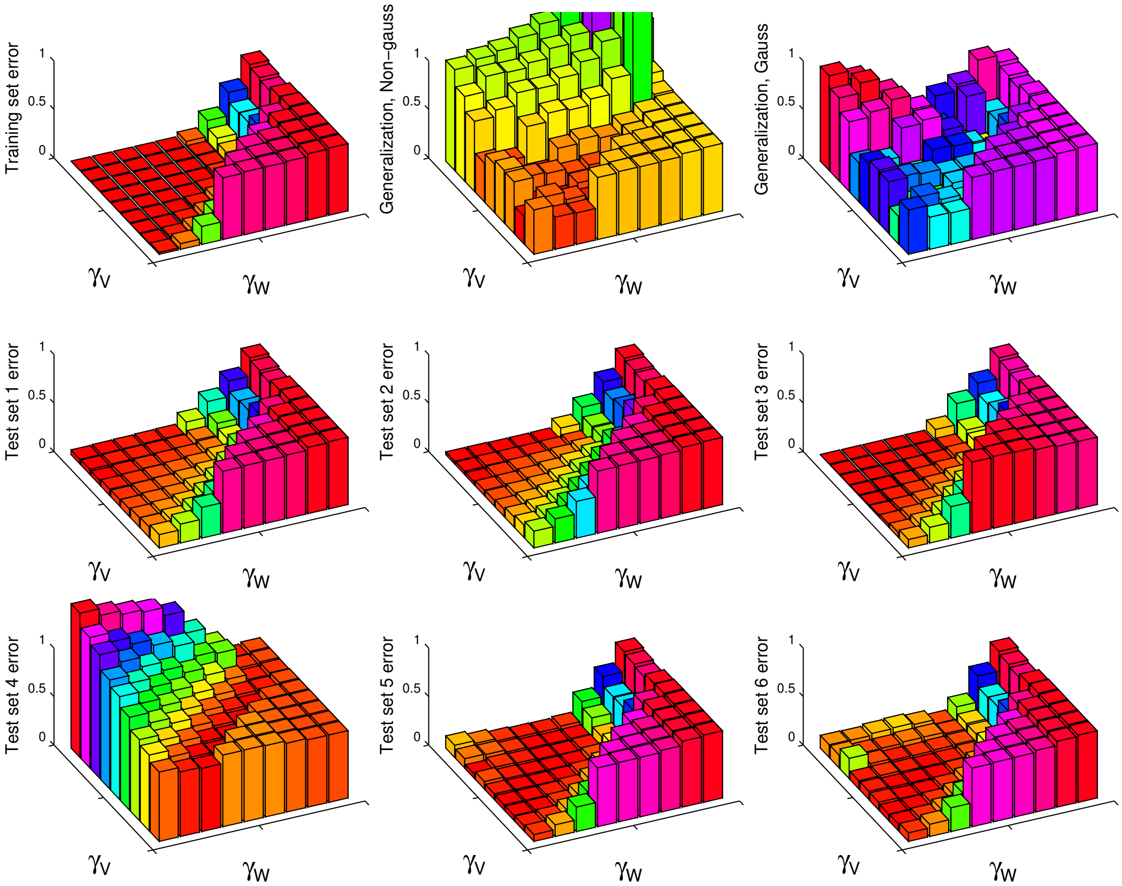


Figure 7.18: Training set, generalization estimate and test set errors for a network sized: $(100 + 1) \times (5 + 1) \times 1$. See figure 7.20 for axes labeling.

for the output threshold, which is used to compensate for the unbalanced classification classes.

For non of the pair of weight decay parameters is there an accordance between minima of the test set 1 and 2 errors and other estimates of the generalization. There is an correspondence between the test set 5 and 6 and the algebraic generalization error estimates. What we are after is, unfortunately, not the intrarun generalization — rather the interrun generalization. If these algebraic measures do not reflect the right type of generalization, we cannot use the adaptive methods to obtain an optimal weight decay parameter.

As a general result of the influence of the regularization can figure 7.20 be regarded: It shows on the left panel the fraction between the norm of the hidden units and the norm of the output weights, and on the right panel the product. To no surprise is an increase in the weight decay parameters imposing the sizes of the weights to decrease.

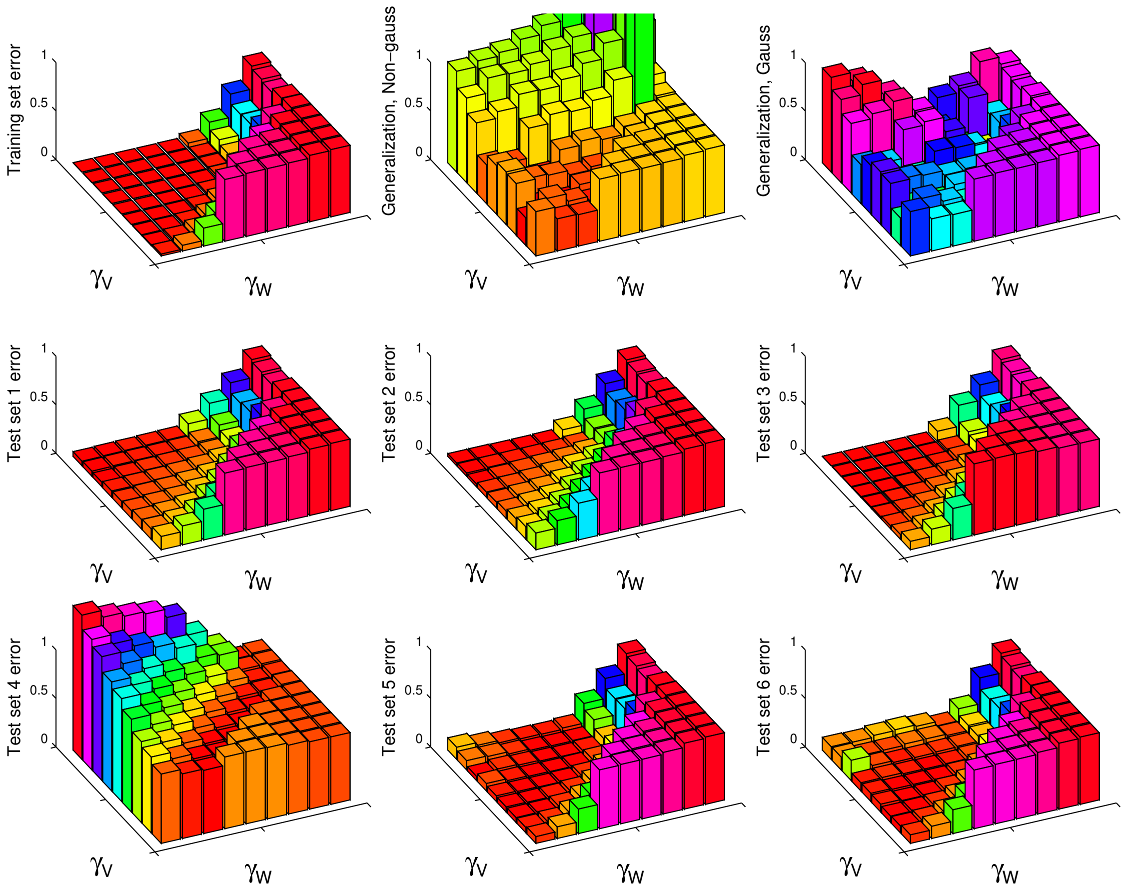


Figure 7.19: Training set, generalization estimate and test set errors for a network sized: $(100+1)\times(5+1)\times 1$

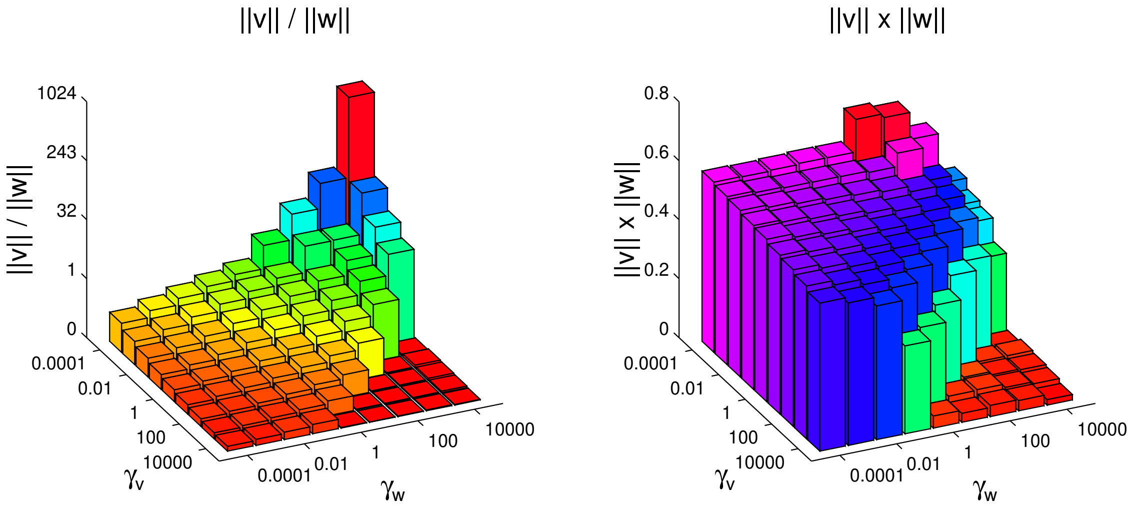


Figure 7.20: The fraction and product of hidden and output weights norm, for a neural network sized: $(100+1) \times (5+1) \times 1$. Compare with the Moody regularization (p. 44).

7.3.4 Hidden Unit Space

The hidden unit space could in this case with the single subject multi-run study be used to examine if there is difference among the states. There is of course no meaning with describing the difference between the rest and activation states: These are trained to be different. But the two rest states are trained to equal output, so it might be interesting to describe if there is any generalizable difference between them.

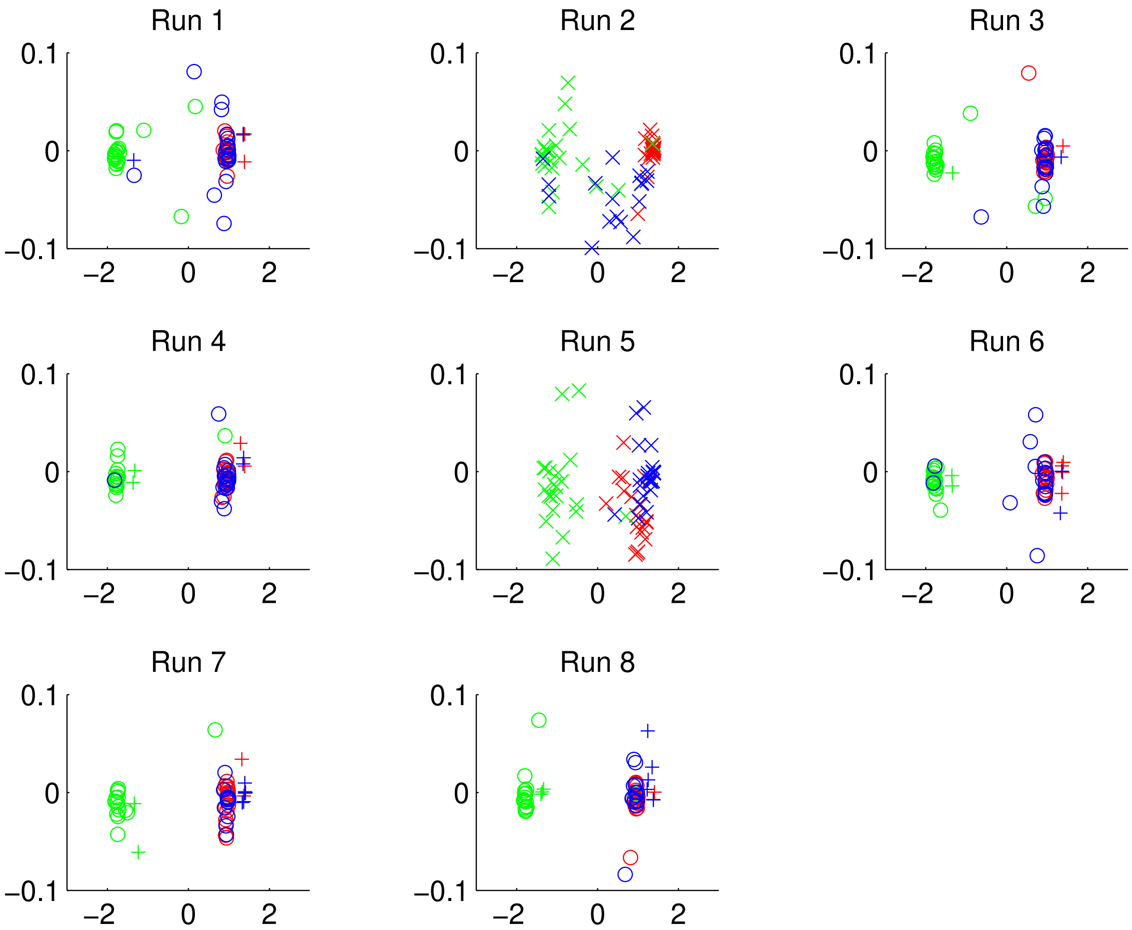


Figure 7.21: Hidden unit state space. Neural network sized $(200 + 1) \times (2 + 1) \times 1$. The neural network is the 15. pruning of figure 7.15. Rings are training set and scans masked out by the hemodynamic mask. Plus is test sets 5 and 6. Crosses are test set 1 and 2. Red first rest, green activation, and red second rest. The hidden space has been PCA rotated.

Figure 7.21 shows the hidden space[3] of a medium sized neural network with a relatively

[3]Throughout this section will red be used for the first rest state, green for the activation state, and blue for the last rest.

low weight decay. The neural network has been trained with the hemodynamic mask, but all scans are shown. On can actually see the hemodynamic response: some blue and green states are in the "wrong" cluster.

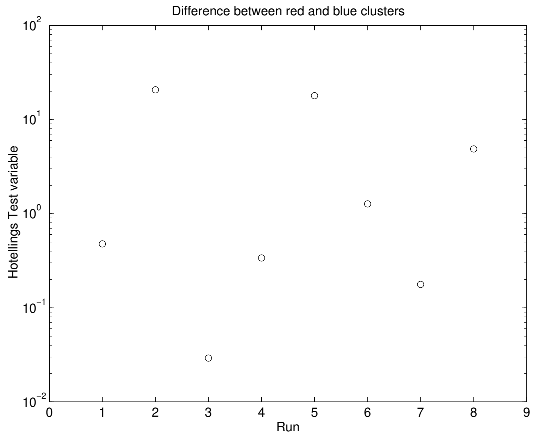


Figure 7.22: Hotelling's test variable on the centers of the red and blue clusters. The comparable the 5% significance level is approximately 3.5.

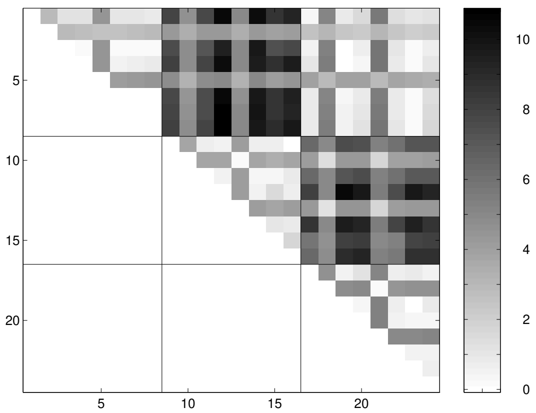


Figure 7.23: The complete Hotelling's two sample test on the cluster centers. The values have been scaled with $\log x + 1$.

By visual inspection there seems not to be any difference between the red and blue states for the training set, though it seems that the test sets have a difference. To test this the Hotelling's two sample test is needed. The figure 7.22 seems to support the visual impression from figure 7.21.

Other investigations of the hidden space show a higher separation between red and blue states (figures 7.24, 7.25 and 7.26). Tough there is a difference within run between red and blue state, this could perhaps be explained by the hemodynamic response. The red and blue states have not a complete general state. For some runs the cluster center is situated different than the centers of the rest. However, for this particular neural network the Hotelling's two sample test gives that the red are so different from the blue, that this will only happen in one out of 55 cases, if the were equal.

Figure 7.27 shows a piece of the pruning history. Between the 89. and the 90. pruning is one of the two remaining hidden units pruned away. Watching the figure it must be clear that this should happen: The second weight seems not to contain any information relation to the paradigm. It is though interesting that in this case the second weight seems to distinguish between the first and the second rest state. Figure 7.28 shows the mean value of the three states in each of the two weights. For a long time during the pruning the two weights are remarkable alike: The hidden unit space seems to be *directional*. Then they split up and there begins to be a difference between the red and blue clusters for the decaying weight.

To investigate the question about this decaying weight beginning to have importance for the distinction between the red and blue state, we can look at the figure of 7.29.

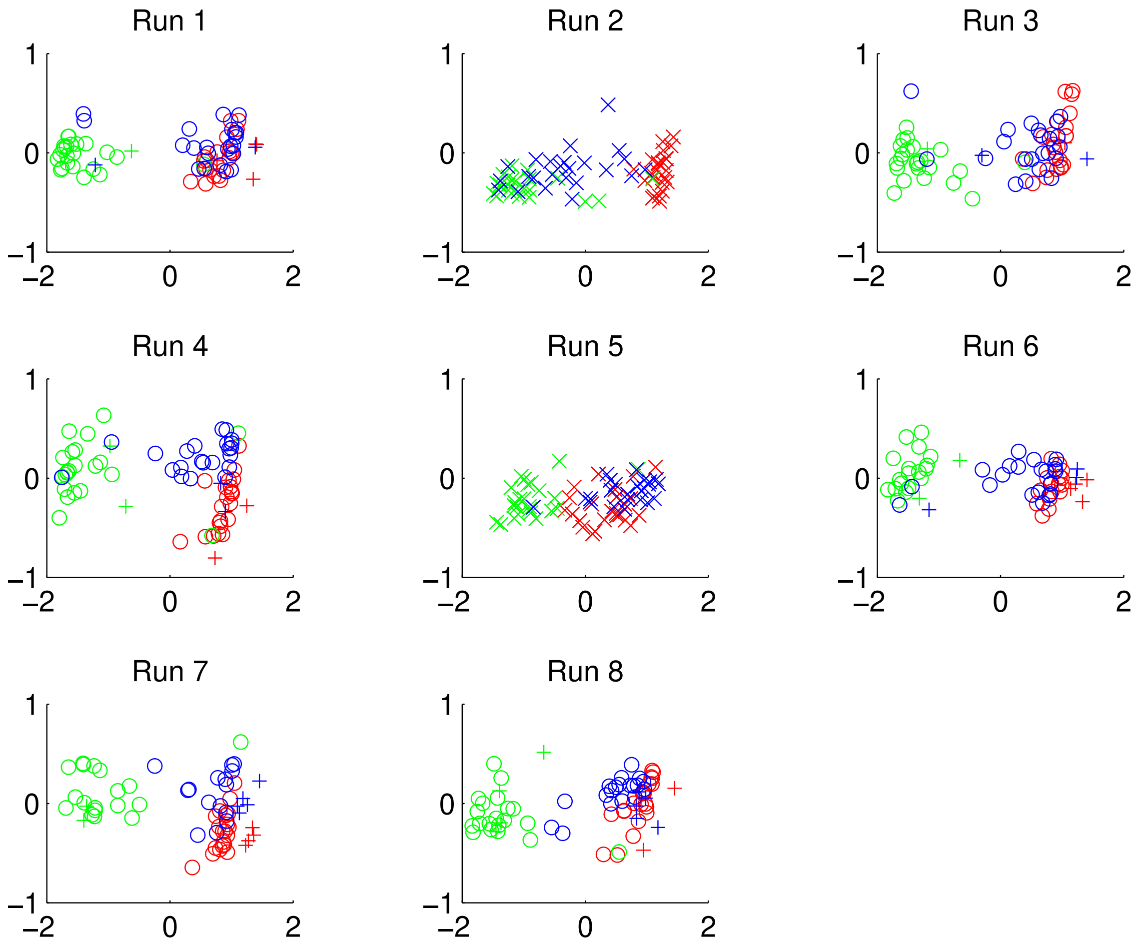


Figure 7.24: Hidden unit state space. Neural network sized $(416 + 1) \times (20 + 1) \times 1$. The neural network is the 98. pruning of figure 7.12, where there are 3 hidden units back. These have been PCA reduced to 2 dimensions. The 8. run can also be viewed in figure 7.61.

This shows the values of all the output weight (20 — quite many). The value of the weights decreases before the weights are pruned. But what is most remarkable is that the Hotteling's test variable (The Mahalanobis distance between the clusters) seems to be dependent upon the pruning: Just before the weights are pruned — where they have a low value — the Hotteling's test variable rises.

I have three answers to this strange behavior:

- The small weights contain a true difference between the red and blue clusters.

- Before they are pruned, weights contain relatively few inputs. This makes them sensitive to errors. Any temporal difference (errors) between red and blue will light up in this unit. There is, however, no clue to why all the red cluster from all the runs should group in one "hypercluster" and the blue in an other "hypercluster".

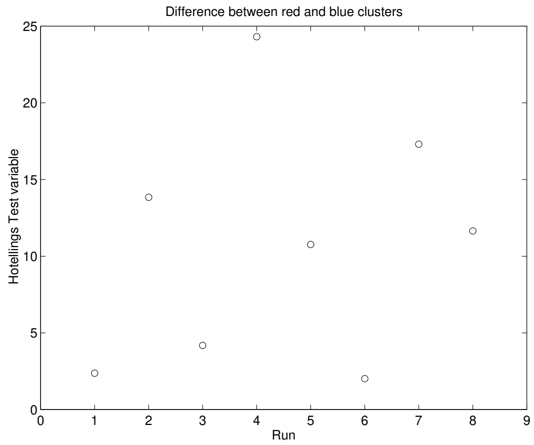


Figure 7.25: Hotelling's test variable on the centers of the red and blue clusters. The comparable the 5% significance level is approximately 3.5.

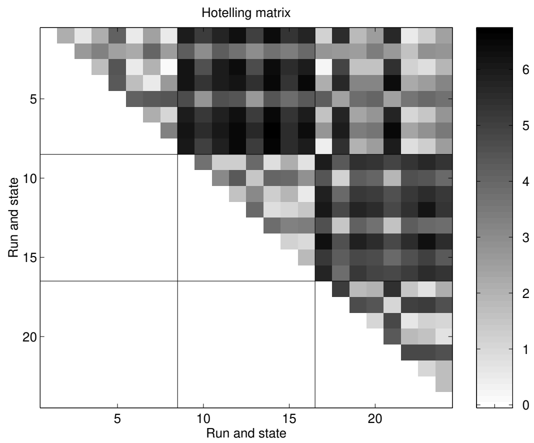


Figure 7.26: The complete Hotelling's two sample test on the cluster centers. The values have been scaled with $\log x + 1$.

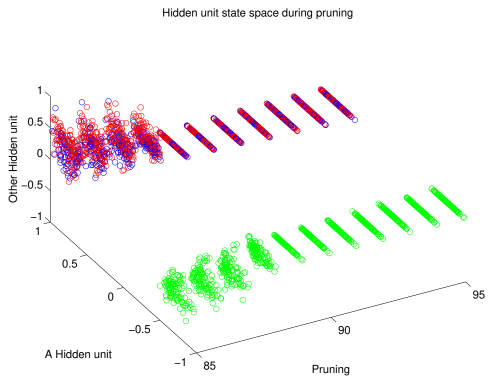


Figure 7.27: Hidden unit state space during a piece of the pruning.

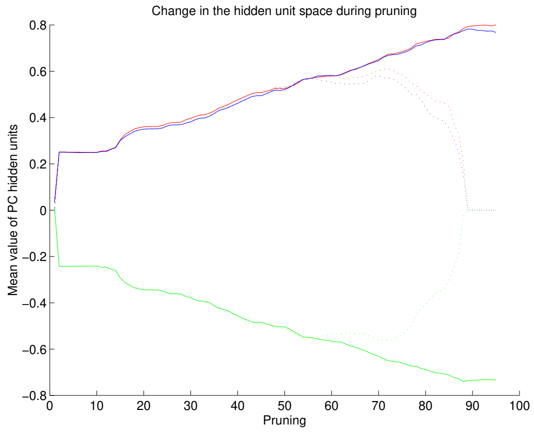


Figure 7.28: The value of two hidden units during pruning. The mean value has been taken for the three states. The solid line represents the first weight; the dotted represents the second weight, which is pruned in epoch 90.

- The Hotelling's two sample test is sensitive to highly collapsed direction. Small differences will bring numerical errors. A method that could be suggested [21] to overcome this problem, is not to use the noncollapsed directions — possibly only the direction with the largest difference between the red and blue state. There is a problem, however, if the state difference lies in the highly collapsed direction (which seems to be the case, judging from figure 7.27).

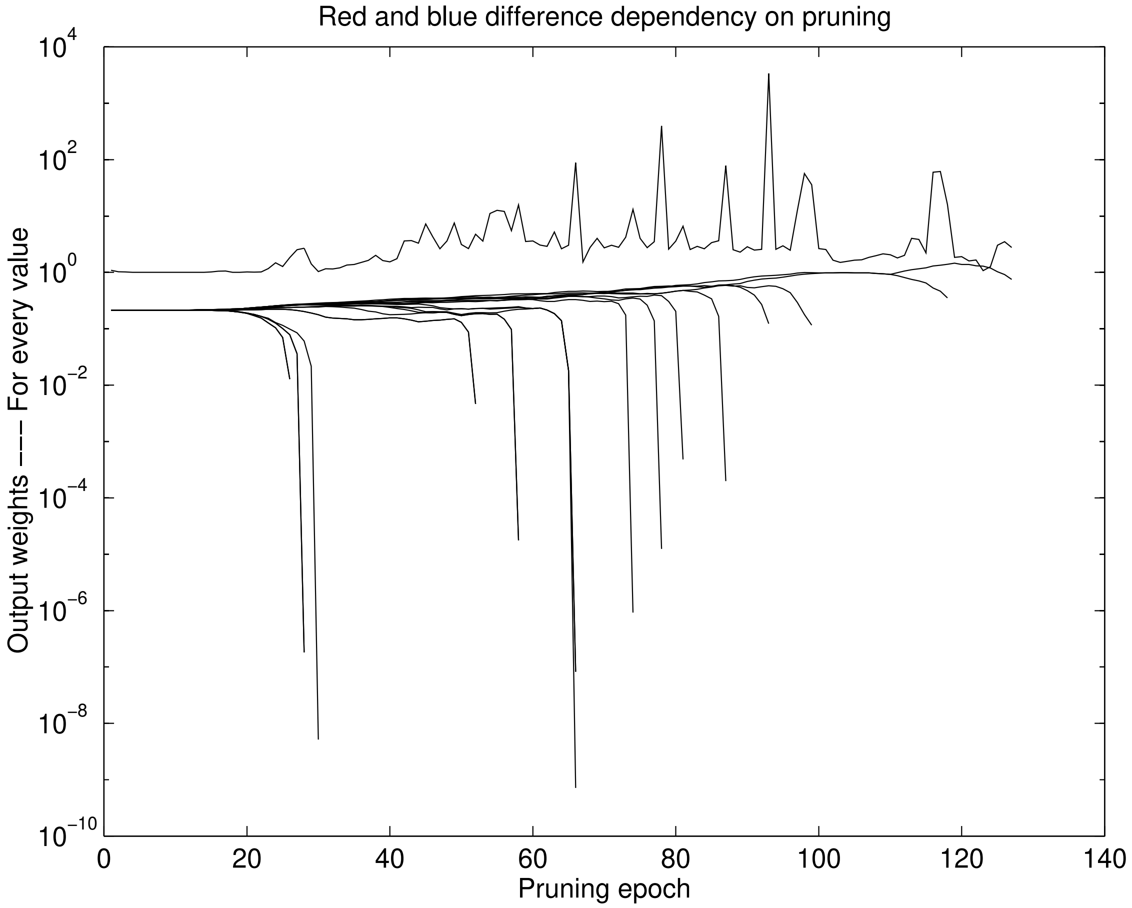


Figure 7.29: The red and blue clusters separation dependency on the pruning. Upper line is the Hotelling's two sample test variable translated to a *for every* value, meaning "one out of ..." will behave such significantly if the H_0 hypothesis was true. Lower lines are the value of each of the output weights.

Whether of the three explanations are right, it is though an interesting dependency.

For the state space in figure 7.24 the red and blue states was backprojected and the difference was visualized. The most significant blob was in the left hemisphere (see figure 7.60).

Directionality

As explained under section 4.7 the hidden units are not working orthogonal, as for example the canonical variables. To investigate the directionality of the representation in the hidden units we can apply the equations from section 4.7.3, comparing the ratio between the first two principal components of the hidden units' space.

This has been done with fully connected network, sweeping the weight decay parameter for the hidden and output layer, figures 7.30, 7.31 and 7.32.

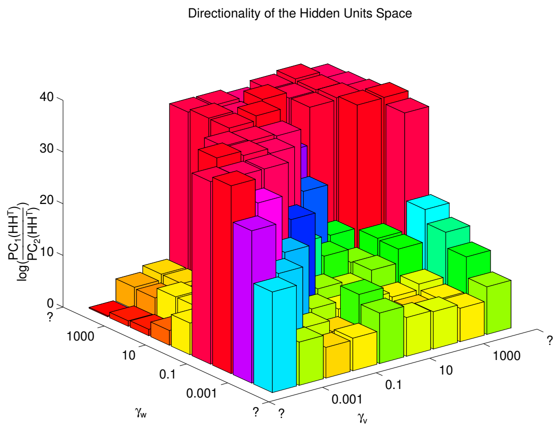


Figure 7.30: Directionality with an extra large net. $(415+1)\times(5+1)\times 1$

The first thing that should catch the eye, is the terrible uniformity that the hidden units are working with. The figures 7.30, 7.31 and 7.32 is plotted with logarithm of the directionality 4.138. Most of the hidden units' space is only utilized in one dimension.

When the network architecture is changed, the influence of the weight decay parameter changes.

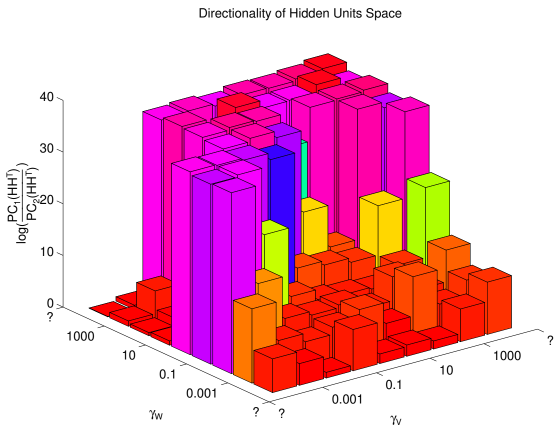


Figure 7.31: Directionality with a large net. $(100+1)\times(5+1)\times 1$

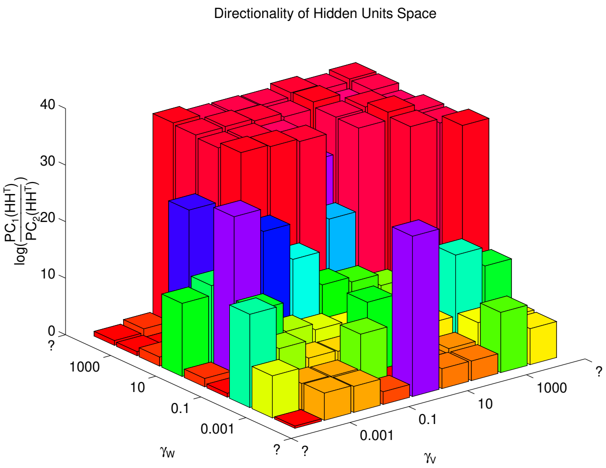


Figure 7.32: Directionality with a medium sized net. $(50+1)\times(3+1)\times 1$

It might be the weight decay that imposes the directionality. Lets presume that the weight decay penalty is the dominating factor in the costfunction, and the weights work somewhat linear. Considering the (hyper-)rectangle the weights span, we would then need a certain amount of perimeter for the amplification of the input signal to the desired output. The weight decay penalty biases this rectangle to a square:

$$\begin{aligned} u + u &= \frac{1}{2}u + \frac{3}{2}u \\ u^2 + u^2 &< \left(\frac{1}{2}u\right)^2 + \left(\frac{3}{2}u\right)^2 \end{aligned}$$

Thus the network would rather have two hidden units working the same than two unbalanced units.

This seems however not to be the case if one look at figure 7.34, which shows a pruning session with a very low weight decay parameter (10^{-7}). This too shows a high directionality in the beginning of the pruning session, decreasing until just before the pruning of the hidden unit.

The directionality might rather be explained by the hidden units competing to be the best: Each wants to explain as much as possible.

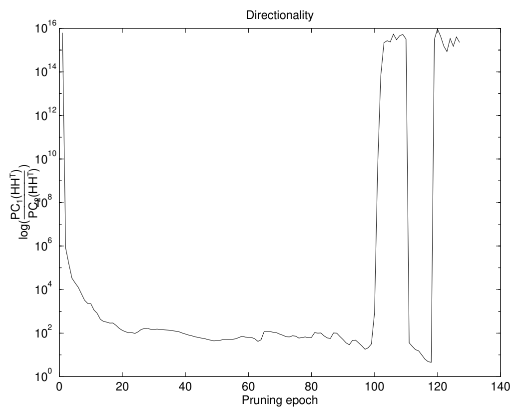


Figure 7.33: Directionality for a pruning session with a huge net: $(415 + 1) \times (20 + 1) \times 1$. The net is the same as in figure 7.12

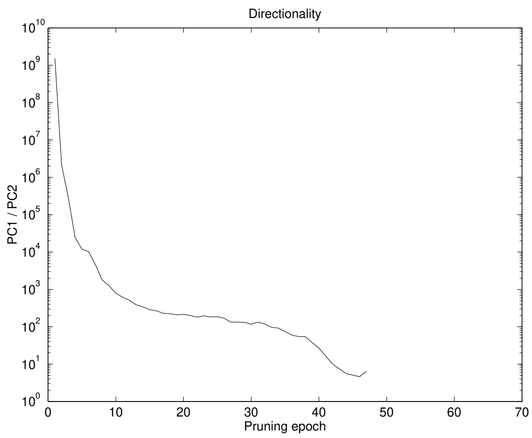


Figure 7.34: Directionality for a smaller sized net: $(200 + 1) \times (2 + 1) \times 1$ with a very low weight decay

7.4 "Brain" Results

In this section will the saliency maps appear. An area in the contralateral sensorimotor cortex, slice 17, has been given special attention: It is likely that the most paradigm relevant activation will be there (p. 17). The particular area, with the size of 16×16 has been used as a benchmark in a comparison between analysis methods [38] [71].

7.4.1 Saliency maps

After the neural network has been trained, regularized and pruning to yield the best generalization the saliency map can be calculated. The equation 4.110 has been used for that.

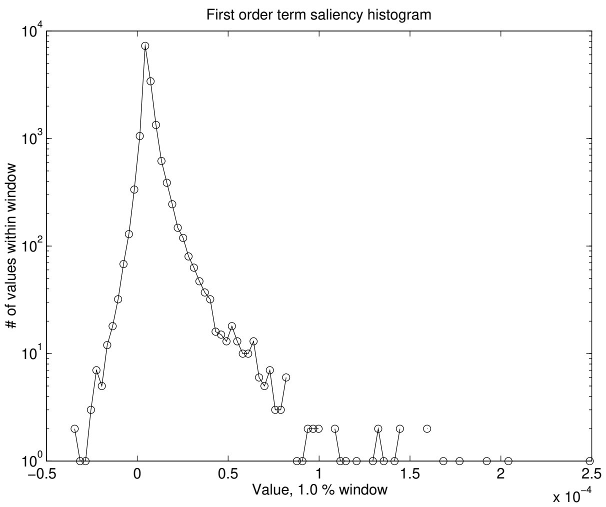


Figure 7.35: Histogram of first order term saliency

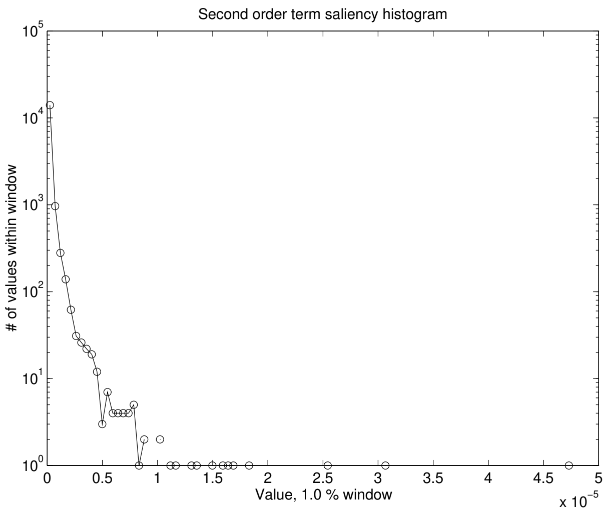


Figure 7.36: Histogram of second order term of saliency

The saliency histogram of figures 7.35, 7.36 and the saliency map of figure 7.38 is computed from the 15 pruning of the optimization session depicted in the figure 7.15 and 7.16. The histograms show, that the second order term does not contribute as much to the saliency as the first order term. This is in accordance with the original study of the saliency [58]. The most pronounced blob in the second order term is in the left hemisphere (see figure 7.60).

Figure 7.37 shows the saliency map put into the SPM Talairach contours. Figure 7.38 shows the 16×16 area in slice 17. The saliency was also rendered in 3D with the definition language VRML (appendix C. The result can be seen in section 7.5.

The salient areas in figure 7.38 have been identified by [43]: Upper left area: some of the supplementary motor area (SMA). Lower right area: The somatosensory area of the left hand. The less pronounced middle right area: motor area for the left hand. The biggest blob is not in the motor area but rather in the somatosensory area. According

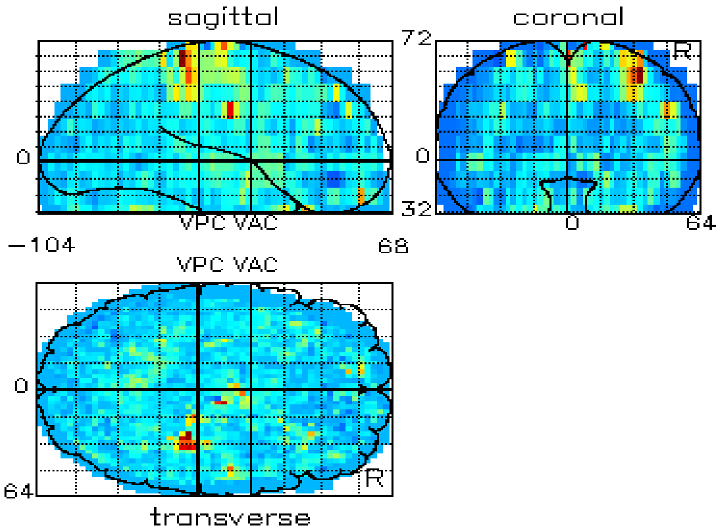


Figure 7.37: Saliency map for the 55 pruning of the neural network of figure 7.12.

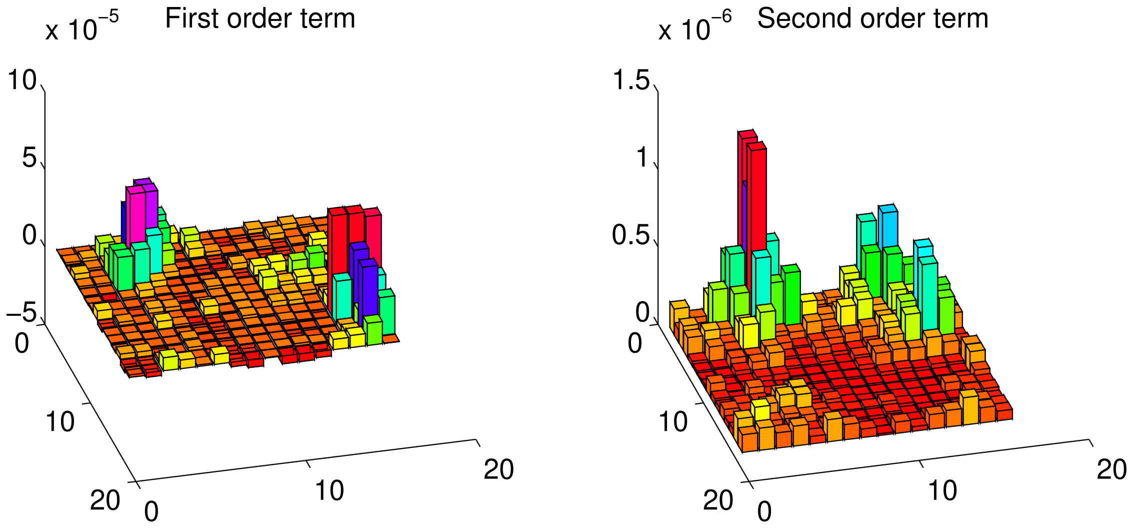


Figure 7.38: Saliency map close-up of a 16×16 area in the 17. slice: the "contralateral sensorimotor cortex". The total saliency is not shown, rather the two contributions: the first order derivative term to the left and the second order derivative term to the right.

to Law [43] this is not a surprising result: The tapping of the fingers also requires the information of the spatial position of the fingers, and the tapping itself creates "touch" activation. The supplementary motor "participate in the earlier stages of motor planning" [72], and it is neither surprising to find activity here.

7.4.2 Simple Fourier analysis

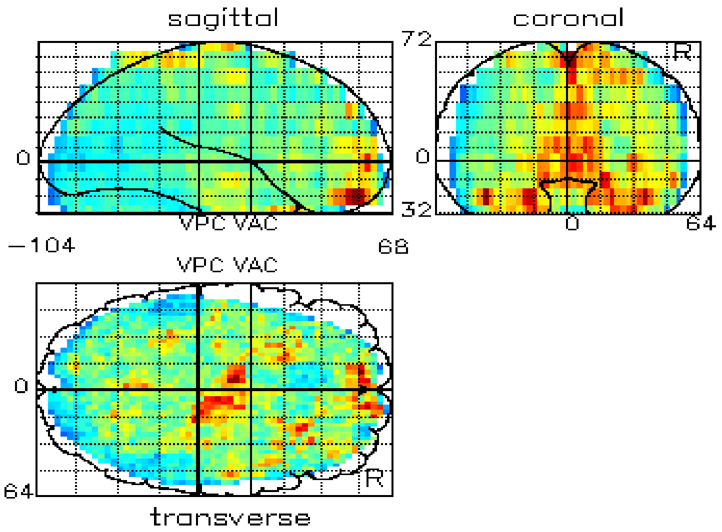


Figure 7.39: Saliency map of a simple Fourier analysis.

For a volumetric comparison with the neural network saliency map my own very simple analysis has been made using the Fourier domain: The whole session has been regarded as one time serie and then Fourier transformed voxel by voxel (p. 29). There are 8 runs, and one paradigm in every run, so the paradigm should be present in the 8. frequency. This is rather inefficient — not using the overtones of the paradigm convolution.

It is interesting to note that the 16×16 area (figure 7.40) resembles the saliency map (by the site of the blobs), while the volume (figure 7.39) looks rather different: The clear blob in the somatosensory area is not present.

7.4.3 Subtraction analysis

A more efficient voxel-based analysis, than the single Fourier component analysis, is the subtraction method. The result from such an analysis is seen in figure 7.41. Before the subtraction the hemodynamic mask (section 7.1) was applied.

With the knowledge from the toy brain of section 6, we known that the saliency is more upper skew. To increase the contrast in the subtraction map we may simple square it. This is done with figure 7.42: The activation areas from the saliency map of figure 7.37 do indeed appear as yellow spot, but the picture is dominated by an activation in

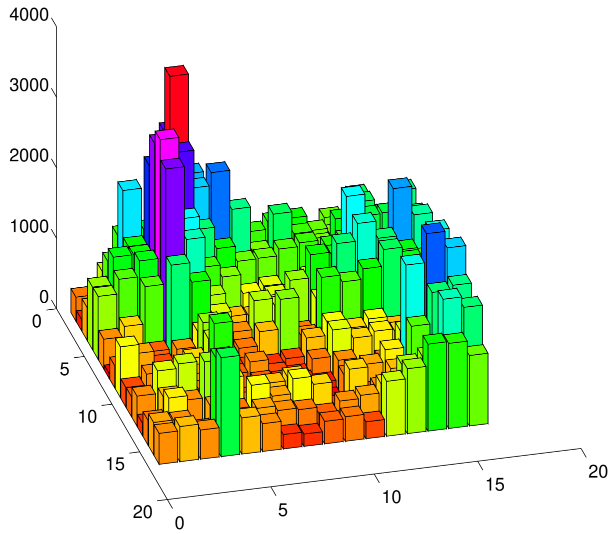


Figure 7.40: The 16×16 area in the contralateral sensorimotor cortex. Results from the simple Fourier method.

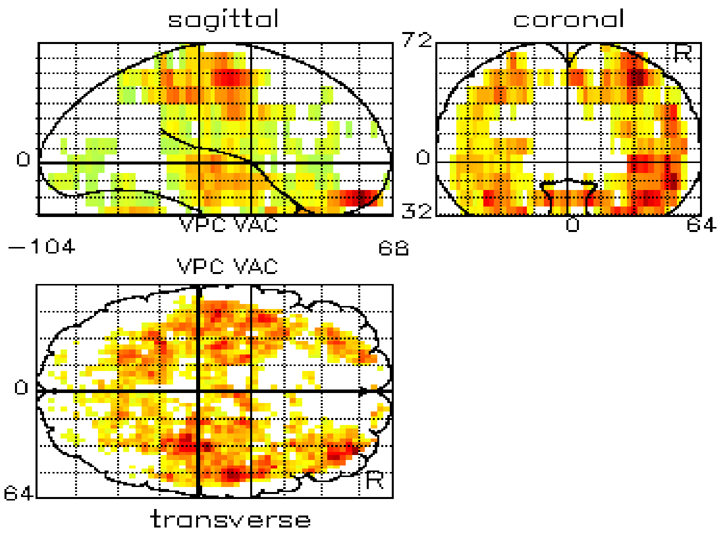


Figure 7.41: "Saliency" map of a subtraction analysis.

the front of the brain. There is certainly a qualitative difference in the volume between the subtraction and the neural network saliency analysis.

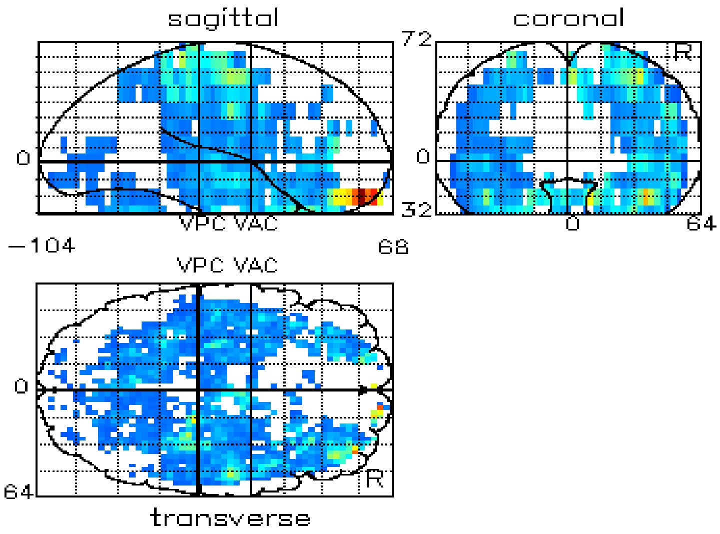


Figure 7.42: "Saliency" map of a subtraction analysis with the square root taken.

7.4.4 Comparison

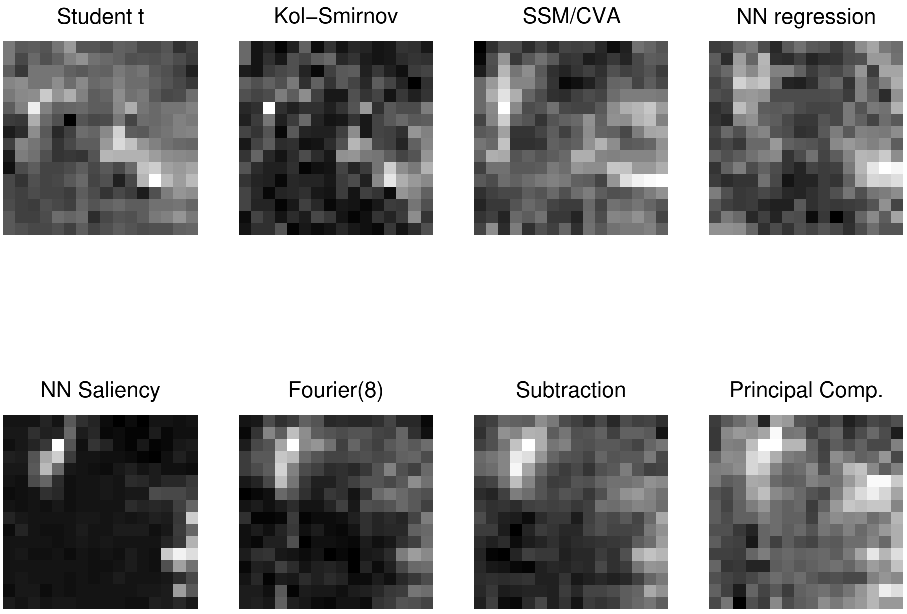


Figure 7.43: Activation measures. The principal component is the important 15.

The saliency compares well with other studies described in [38] [71]: Student t-test [38], Kolmogorov-Smirnov [38], PCA with canonical variable analysis (SSM/CVA), neural network regression analysis [63] (p. 29). A quantitative measure of the correlation among the analysis methods can be seen in table 7.3, and a qualitative may be obtained looking at low resolution images in figure 7.44.

The two methods, that the result of the neural network saliency method resembles most, are strangely enough the two simple voxel–based methods. This is though only for this slice.

Viewing figure 7.44 one sees that all methods more or less find two activated areas: The upper left and the lower right. Thus in this simple case with a slice that as been selected with prior knowledge of the functional anatomy, there is not qualitative difference for the tops of the blobs. The borders of the activation area are however not similar. It might be explained by the methods' different approach to handle the dependencies among the voxels.

The analysis with the subtraction and the simple Fourier methods gave the clue that larger differences rather lay in the volumetric images (figures 7.39 and 7.41): That the analysis methods may give approximately equal results in a local area, but when the volume is considered as a whole, the areas are weighted differently. In table 7.4 is shown the ordinary correlation coefficient for the masked volume and for the whole volume.

It is interesting to note that the neural network has found the 15. principal component important, through the PCA sweeping (section 7.2.2). But measuring the correlation

	Stud.	KolSmi	PCA/CVA	NNreg	NNsal	$Four_8$	Subtr.	15. PC
Student		0.43	0.31	0.19	0.27	0.24	0.33	–0.26
KolSmi	0.72		0.30	0.25	0.29	0.20	0.28	–0.23
PCA/CVA	0.56	0.55		0.20	0.24	0.09	0.18	–0.27
NNreg	0.35	0.44	0.42		0.26	0.27	0.32	–0.25
NNsal	0.37	0.46	0.45	0.48		0.32	0.43	–0.39
$Fourier_8$	0.28	0.27	0.22	0.46	0.66		0.64	–0.43
Subtraction	0.40	0.40	0.35	0.50	0.76	0.87		–0.58
15. PC	–0.34	–0.34	–0.43	–0.39	–0.62	–0.70	–0.82	

Table 7.3: Correlation between different analysis methods. Lower left: ordinary correlation coefficient. Upper right Kendall τ rank correlation.

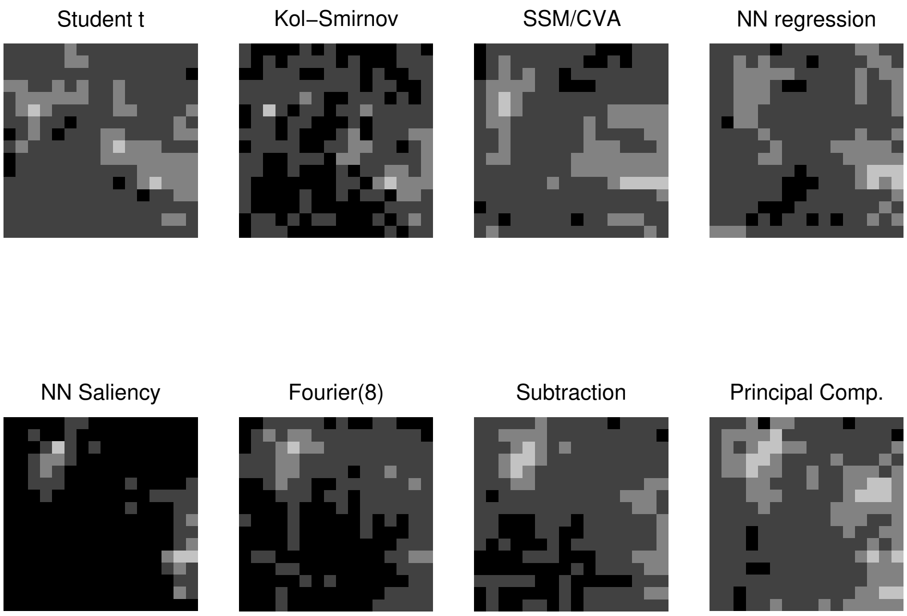


Figure 7.44: 4 levels images of the activation measures.

between that principal component and the saliency map, yield that they are not so correlated (see also figure 7.9 compared with 7.37; and figure 7.55). The other principal component may balance out some of the blobs from the principal component.

The saliency map is much more selective than any other of the analysis methods: The histogram (figure 7.45) has the largest and thinnest upper tail.

As described in section 4.6.8 and tested under the analysis of the toy brain (section 6), the saliency is rather a squared activation compared to the results from the other analysis methods. But an other approach to explain the contrast, is to see what the neural network

	NNsal	Fourier_8	Subtraction	15. PC
NNsal		0.35	0.21	–0.17
Fourier_8	0.27		0.15	–0.11
Subtraction	0.18	0.08		–0.44
15. PC	–0.15	–0.07	-0.44	

Table 7.4: Volume correlation coefficient. Lower left: masked volume. Upper right: Full volume.

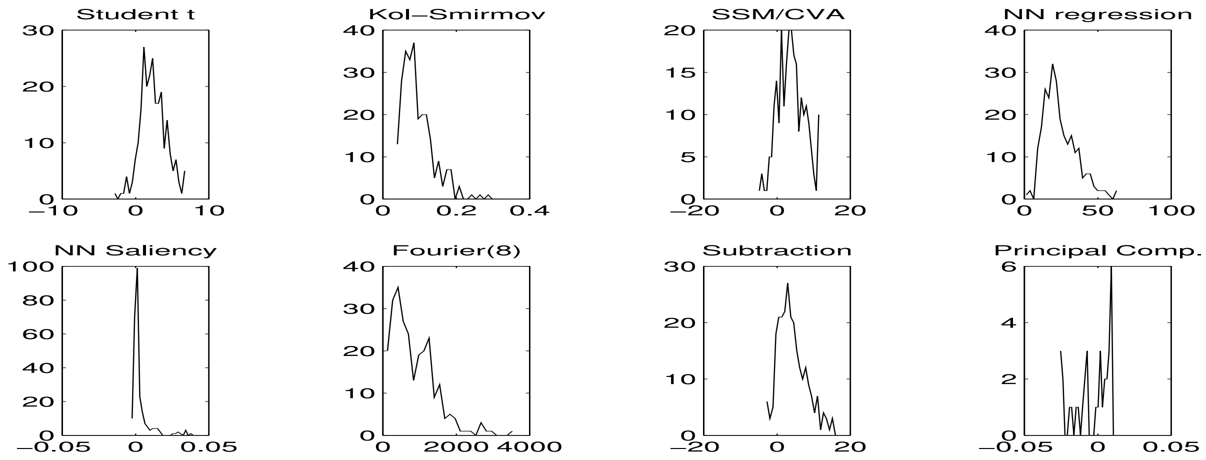


Figure 7.45: Histograms for the activation measures.

actually needs to predict the paradigm. If there is just one noiseless voxel containing the paradigm, the neural network needs only use this — presuming that the preprocessing does not affect the signal (this is though unlikely). This voxel is then the only voxel in the saliency map. The neural network saliency map is one of the few methods to have this capability.

It might be asked whether it is preferable to have this elitarian contrasting: Those blobs which have been found important could however be deleted, and a new hunt for the second most salient area could be started, yielding an iterative scheme.

7.4.5 Reproducibility

The neural network saliency map should of course be reproducible. A comparison between two neural networks, with a different startup architecture, is shown at table 7.5.

The second order term saliency shows a correlation about 0.99 for both the volume and the 16×16 area. The first and the second order term do not need to have relation to each other. Their correlation should be low: For the two different neural networks in

Architecture and selection criteria		
$(415+1) \times (20+1) \times 1$		0.96
$(200+1) \times (2+1) \times 1$	0.80	

Table 7.5: The reproducibility of the saliency map. Lower left: volume, Upper right: The 16×16 area. The neural networks have been selected so that the out-of-basis test set generalization error was lowest.

table 7.5 it is 0.14 and 0.27.

The correlation is though still for the same basis, and training set. Furthermore the directionality that the neural network is working under might impose constraints on the possible outcomes.

7.4.6 Hemodynamic response

To investigate the dynamics in the brain, we can regard the 8 runs as 8 time series. The neural network does not directly model the dynamics, but the output of the neural network can be used as a simple measure for the influence of the hemodynamic response. The neural network measure is not local, but working on the whole image of the brain — though paradigm-weighted.

In figure 7.46 is shown the output from the neural network (remember the hemodynamic mask has been applied, and that the 2., the 5., the activation flange of the 6. and the deactivation flange of the 7. run are test sets).

Comparing the activation flange with the deactivation flange, two remarks can be made: The activation flange is *shorter* than the deactivation flange:

$$Rise\,time < Fall\,time \tag{7.1}$$

...and the length of the deactivation flange is more varied than the length of the activation flange.

The "equation" 7.1 is in accordance with the findings of Strother [71] [74]. Because of the non-balanced hemodynamic mask (4 activation flange taken out, 10 deactivation flange taken out), it might however be argued, that the neural networks fitting of time dependent noise can cause the output after the flanges to behave differently in relation to each other.

Watching the activation flanges of figure 7.46 they appear to be relatively sharp compared to the deactivation flange, and the start of the flanges is not varying so much.

If the hemodynamic response should be easy to model (p. 29), then the flange should be soft (else it would not be worth the effort), and the flange should not vary between runs. The length of the deactivation flange has a large variation between the runs, so it might be hard to model this flange.

Instead of looking at the output of the neural network, we can use the hidden unit state space with the scan to cluster distance (figures 7.47 and 7.48). In this particular case

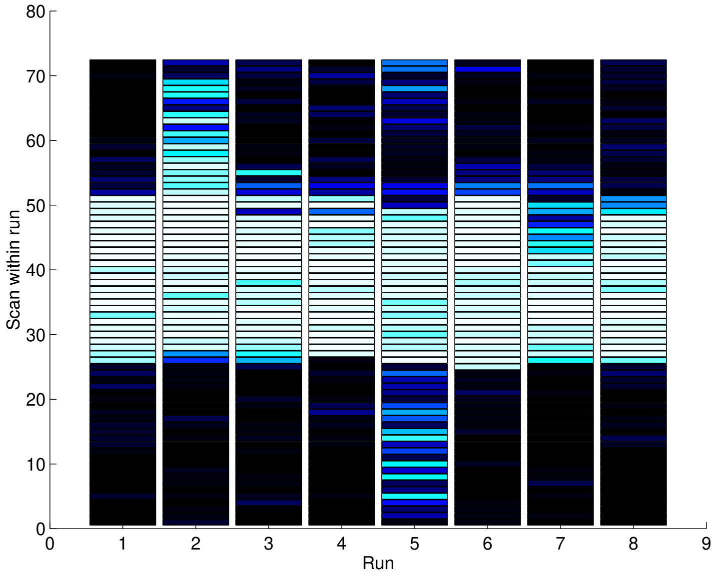


Figure 7.46: Output from the 98. pruning of the neural network of figure 7.12. The split of the sets is as in table 7.2.

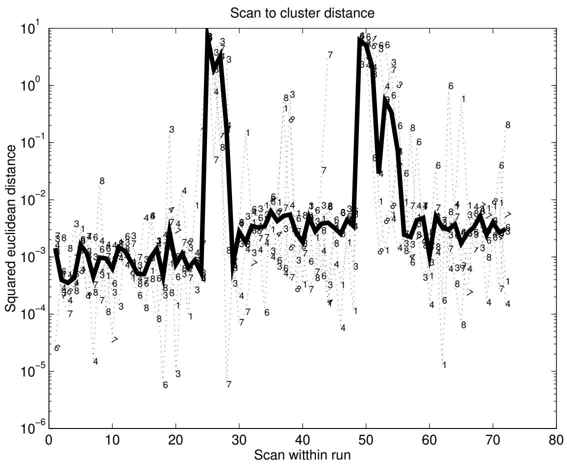


Figure 7.47: The scan to cluster squared euclidean distance. Cluster centers are found with the hemodynamic masked scans. The tick line is the median for all runs.

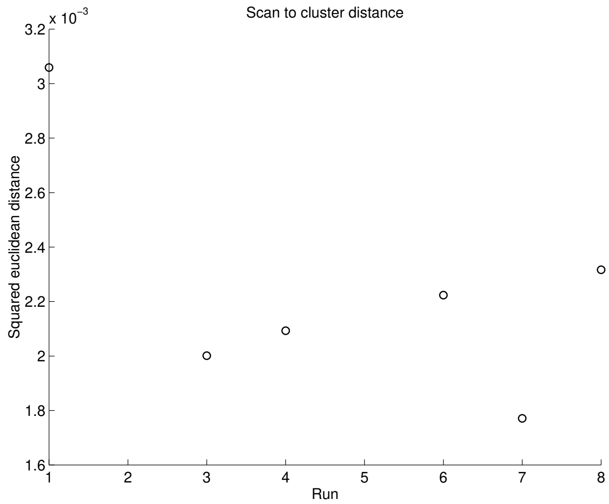


Figure 7.48: Scan to cluster difference.

it does indeed show nothing: There are two remarkable deviations just after the shift of states (figure 7.47). This is of no surprise since non of these scans are in the training set: The test sets are simple more split than the training set. Figure 7.48 shows the distance weighted with respect to the runs. The first run shows that the scans have large distances to the centers. A daring interpretation of this is that the mental processes in the first run is more disturbed.

7.4.7 Blob activation

With the activation patterns found through the saliency map, we might ask what actually happens in the individual blob. In figures 7.49 and 7.50 are the activation in the finger sensoric area of the right hemisphere shown. A cube in this area has been masked out, and the voxel inside this cube has been weighted according to their saliency. In figure

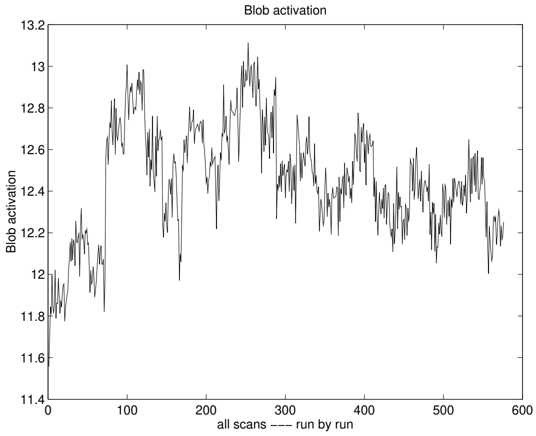


Figure 7.49: The saliency weighted blob activity for the blob in the contralateral finger sensoric area (see figure 7.57).

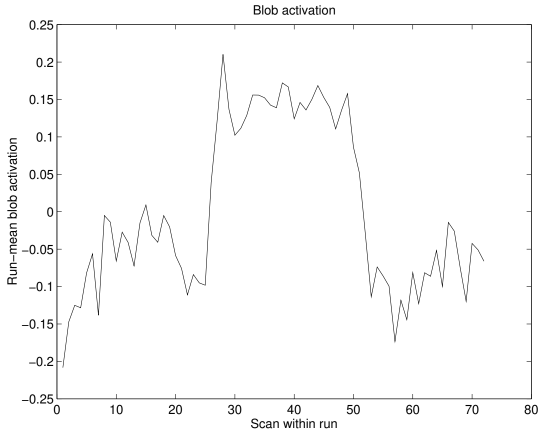


Figure 7.50: Mean activation in the same blob as figure 7.49

7.49 is seen the complete time series of the sessions. One can just make the paradigm variation out, but the main variations are due to the different runs. This run effect can be filtered out: In figure 7.50 every run has been centered, and then the mean of these centered runs has been calculated. This makes the variation due to the paradigm very clear. In this blob the paradigm activation is positive — what of course is of no surprise — from the knowledge of the fMRI.

From figure 7.50 one is also able to judge the blob hemodynamic response: The activation flange does first come to its peak in the scan of number 28. For the deactivation flange the restored state is the 53. Thus the fall time is longer than the rise time.

In comparison with section 7.4.6 the hemodynamic response is somewhat different. The hemodynamic responses in section 7.4.6 are neural network weighted and they are rather an expression for the global inertia in the brain.

An other interesting blob we might investigate, is the one in the left hemisphere that the second order term of the saliency and state space red-blue difference backprojection points out. This is seen in figures 7.51 and 7.52. It seems there is not a difference between the red/blue state and green state, but rather between the red/green state and the blue:

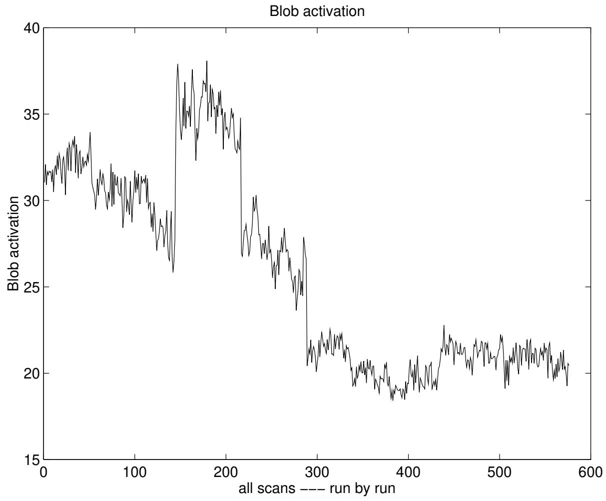


Figure 7.51: The saliency weighted blob activity for the blob in the left hemisphere.

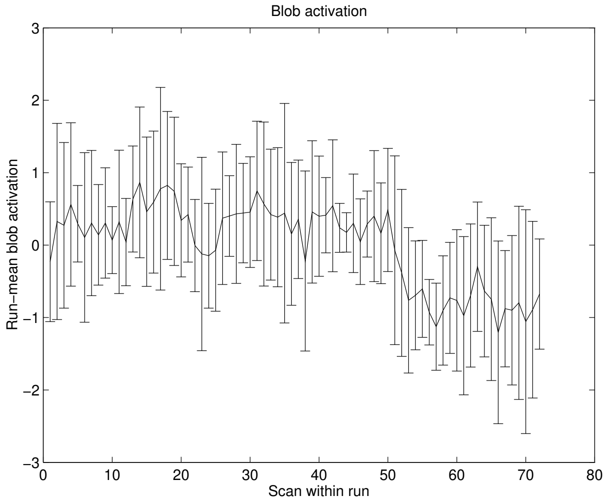


Figure 7.52: Mean activation in the same blob as figure 7.51, and with errorbars for the standard deviation.

thus a difference between the red and blue state as suggested by the state space of the hidden units.

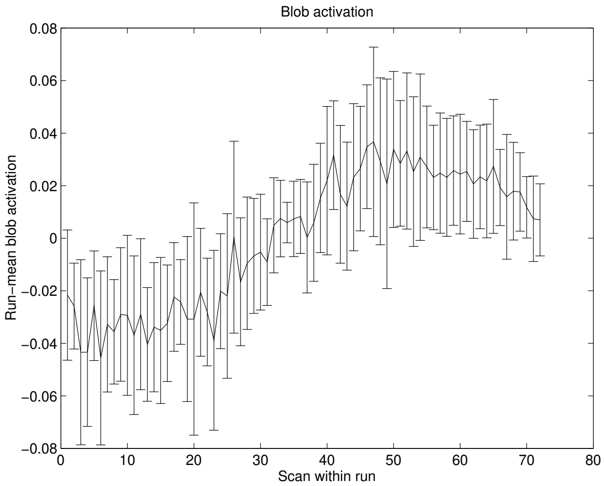


Figure 7.53: The mean activation in a blob in the front of the head — just watchable in figure 7.57 at the bottom edge.

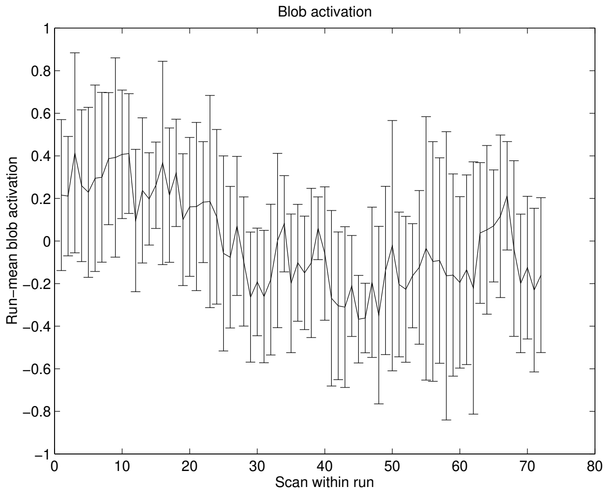


Figure 7.54: Mean in an other front head blob.

Non of the blobs in the later figures show full accordance with the paradigm, — rather a "run general" behavior: If a voxel has an increasing activity in all runs, it is possible for a two hidden units neural network (varying their threshold) to imitate the paradigm.

7.5 3D Views

This section will contain some screenshots from the 3D visualization. The VRML figures (7.55, 7.56, 7.57, 7.58 and 7.60) show the isosurface from volumetric data. The transformation between these two formats has been made with the `polyr` program [30], which uses the marching cubes algorithm (p. 67).

They are but mere screenshot: The important depth cue one gets from moving in relation to the object can not be obtained.

An other compromise one must make is in regard to the transparency: Present day surface-based graphics systems use dithering to obtain the transparency: This decreases the visualization quality and limits the amount of transparency levels one can have.

Most 3D functional brain visualization seems to be presented without any frame of reference, apart from an anatomic overlay (see for example [69]). This is in contrast to 2D plots where axes and grid lines help with orientation clues. In the virtual world build in VRML, I put 3D gridlines and axes in the form of rulers to obtain a more precise orientation.

When put in a public readable place the VRML files are directly includable for internet users. The coordinate system they must confer to is the bicommissural in centimeters. VRML and the bicommissural coordinate system differ in what coordinate is up (Appendix C): The coordinates used is the bicommissural.

A glyph-like presentation of the functional blobs has been made with color and texture.

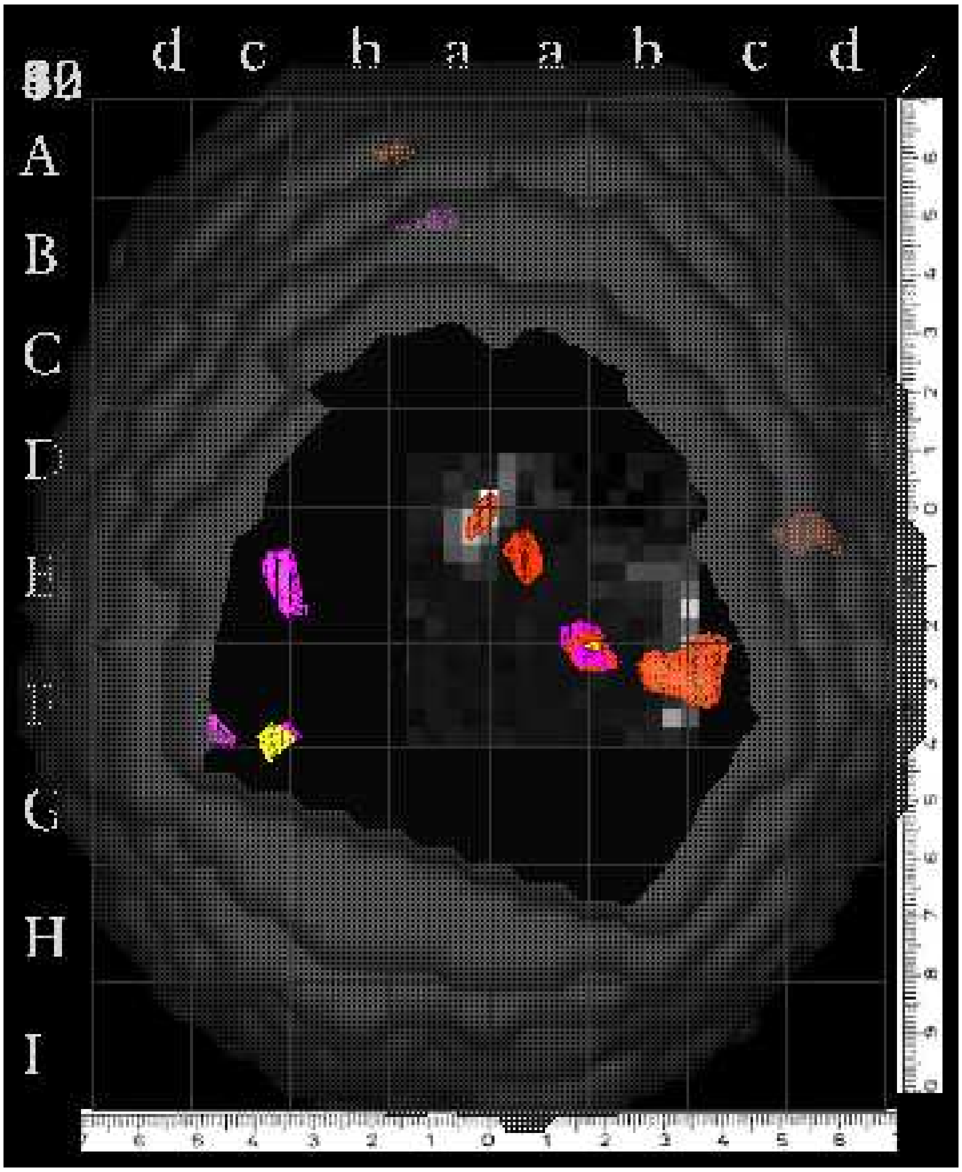


Figure 7.55: Transversal orthogonal view from the top. The orange blobs are the first order saliencies; the yellow are the second order (exclusively). The isosurfaces are not comparable. The magenta blobs are the important 15. principal component. The grid, labels and rulers are in accordance with the Talairach atlas [76]. Also shown is the saliency of the 16×16 area in the 17. slice that has been analyzed with other methods (p. 103) The big convolving surface is one of the scans (the total activity).

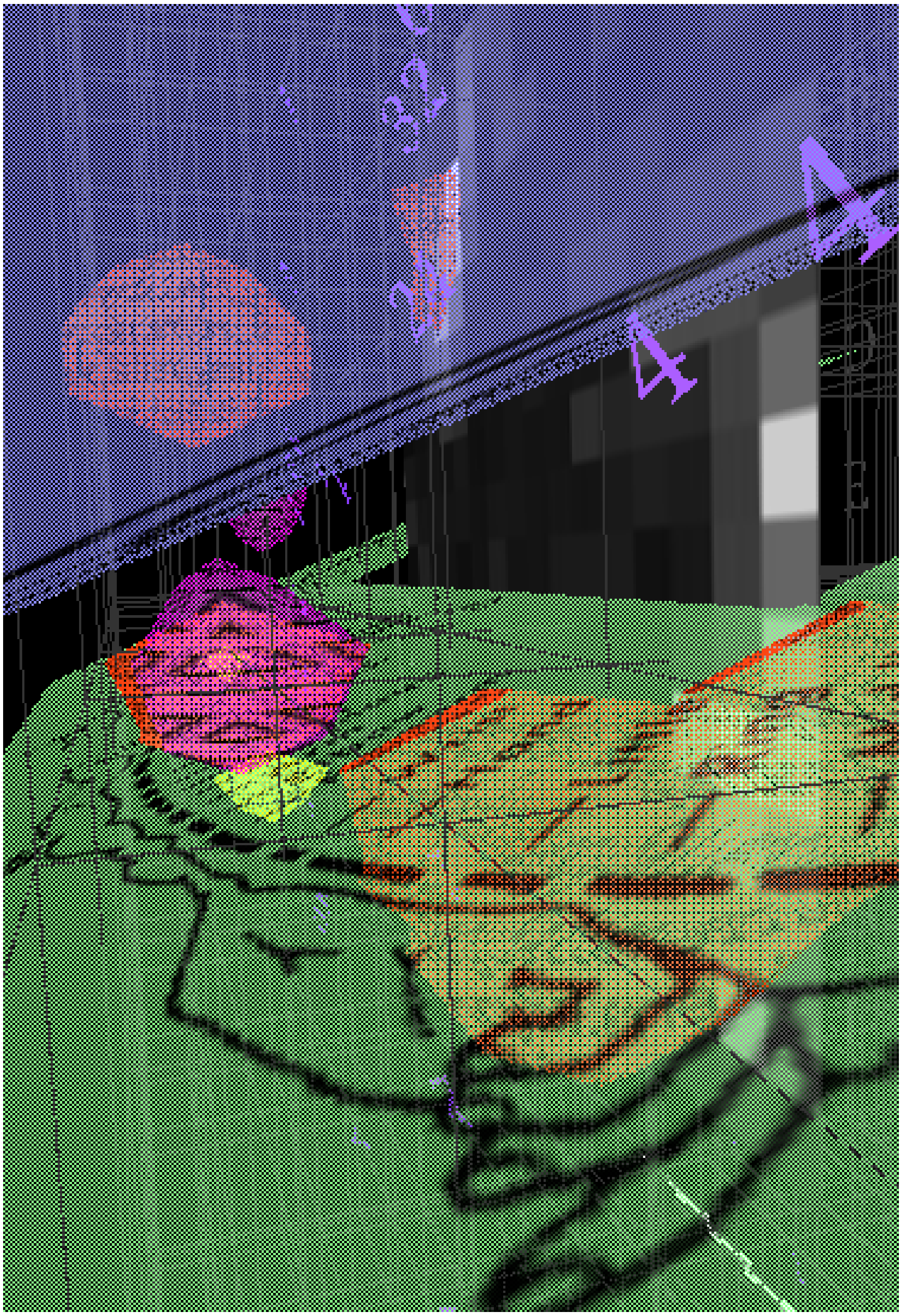


Figure 7.56: Sagittal view from the left on the blobs from the first and second order saliency and the 15. principal component (see the figure 7.55 for legend). The numbers denote the Brodman areas. We are at the Rolandic sulcus with the Penfield somatosensoric map to the left and the motor map to the right.

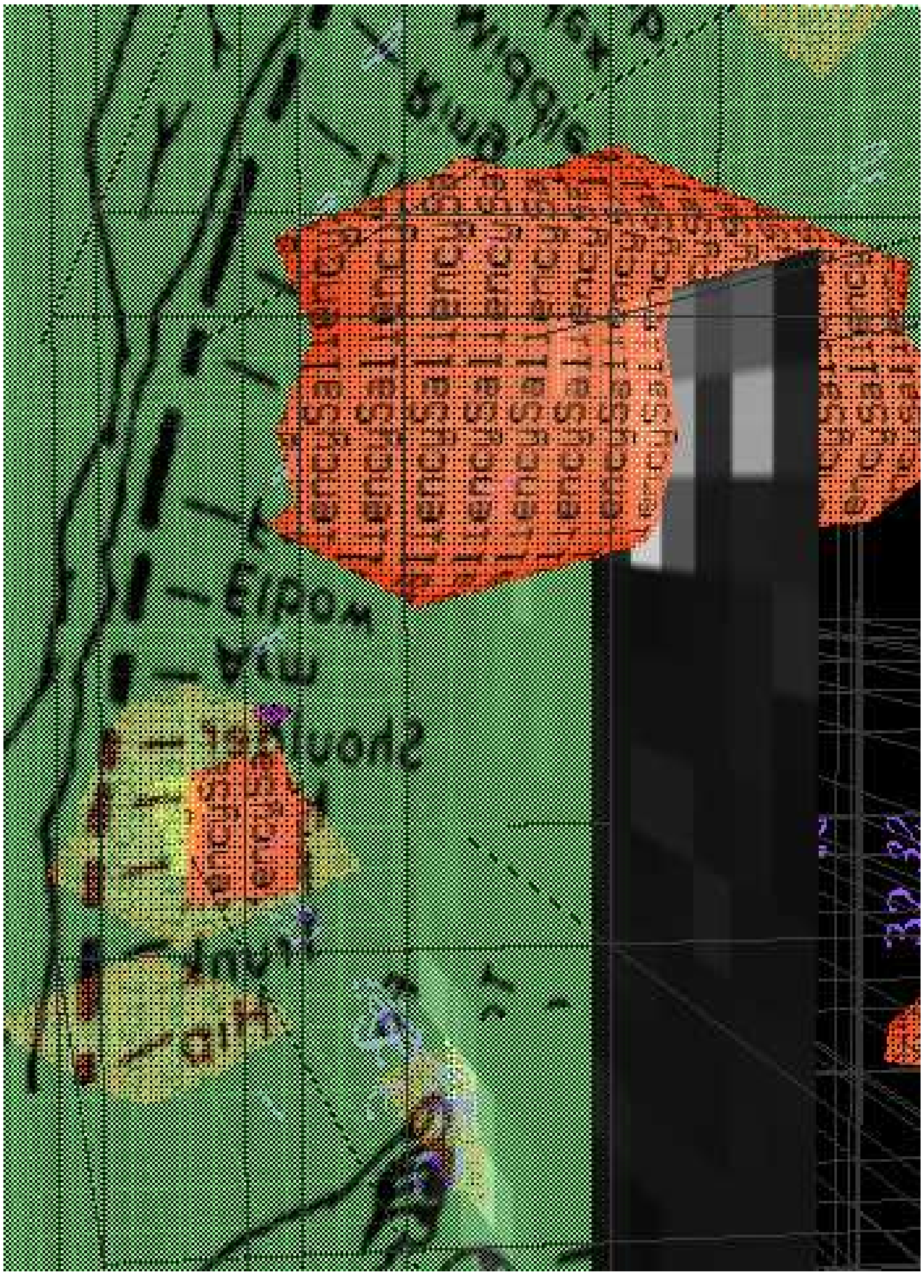


Figure 7.57: Coronal perspective view on the blobs, right into the Penfield somatosensoric map. Refer to figure 7.55

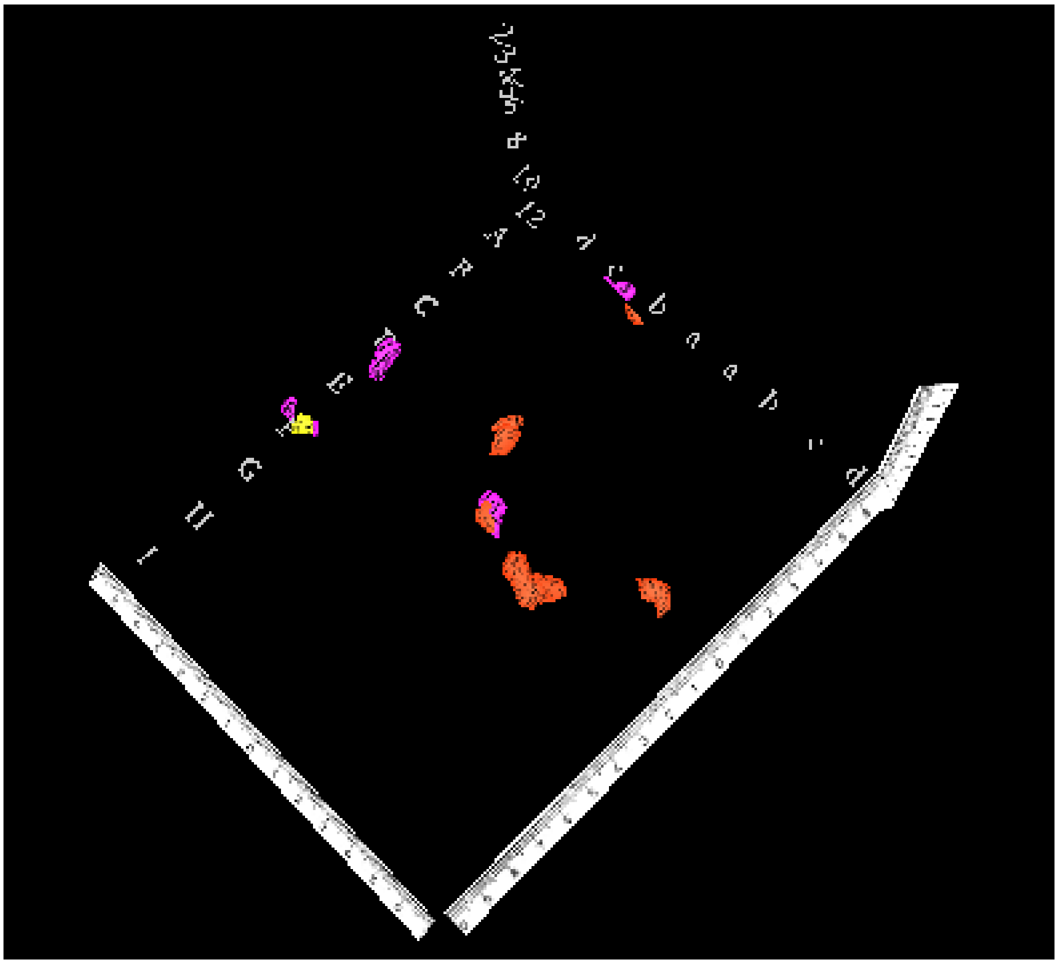


Figure 7.58: Perspective view on the saliency blobs. Compare with figure 7.59.

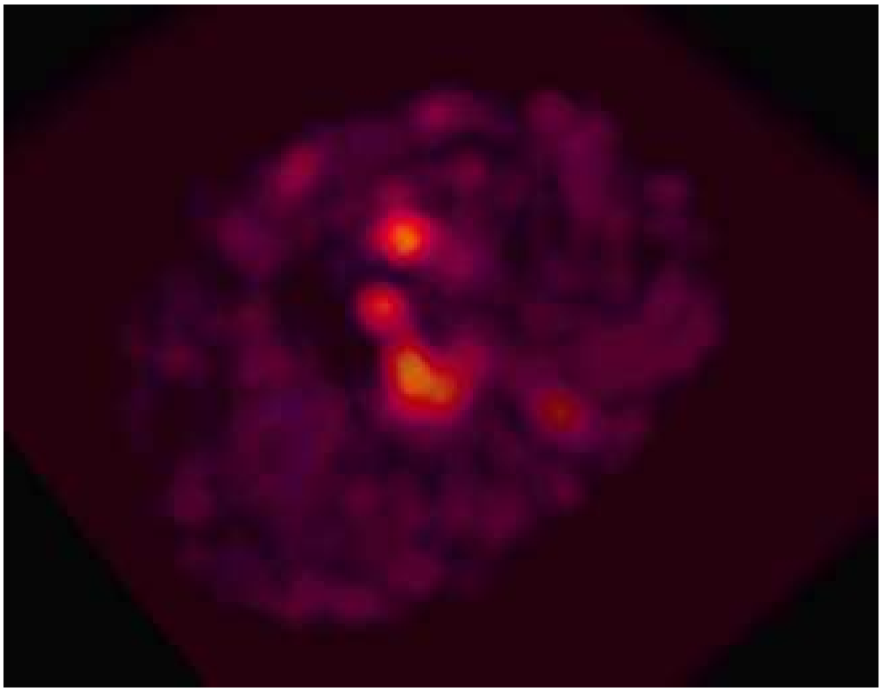


Figure 7.59: Perspective view on a saliency map constructed with a volume-based program. The salient — bright — areas can directly be compared with the orange blobs in the VRML snapshots.

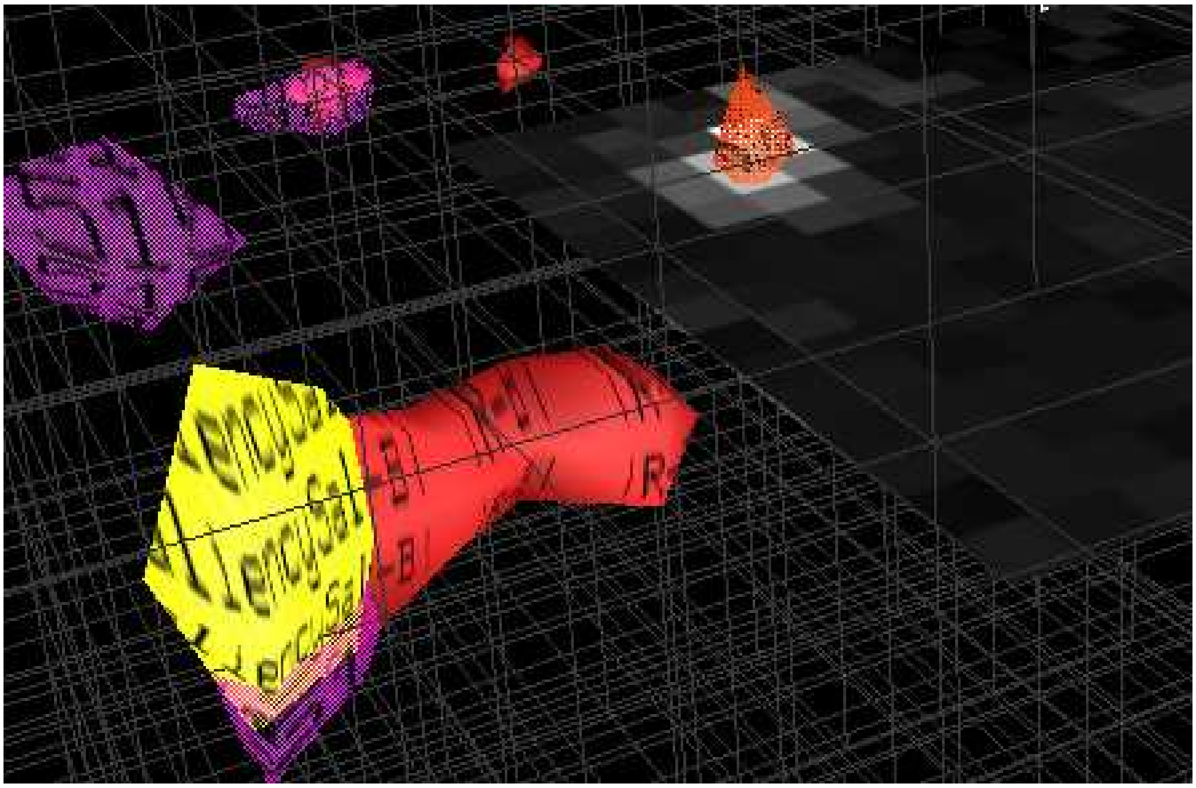


Figure 7.60: Perspective view in the left hemisphere on the second order term saliency (yellow blob), the 15. principal component (transparent magenta blob), and the difference on the backprojected red and blue state (red blob).

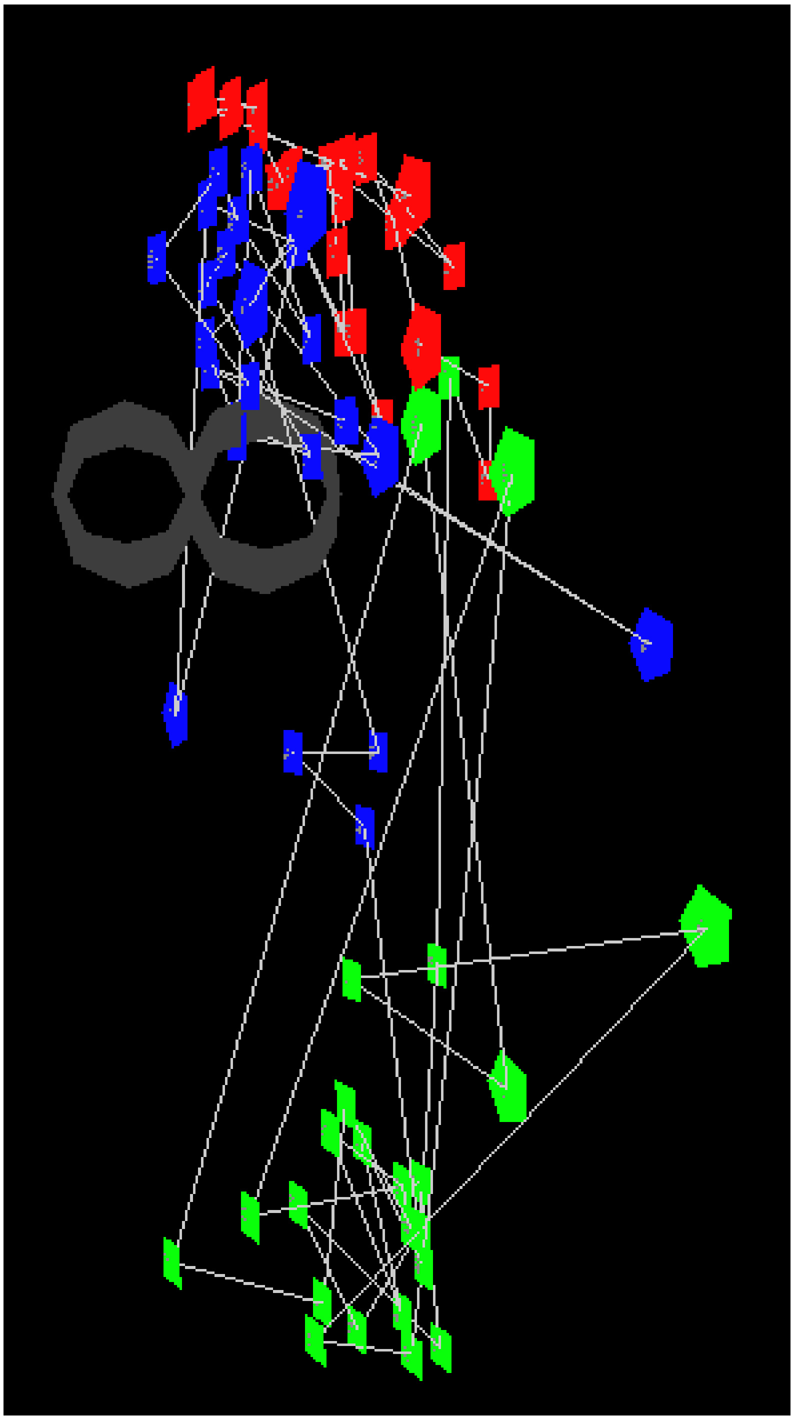


Figure 7.61: Spate space of the hidden units. From the 8. run of the SFO scans. Red are first rest, green are the activation scans, and blue are the last rest. The lines connect consecutive scans.

Chapter 8

Conclusion

3D Functional fMRI brain images have been analyzed with principal components and a neural network, the internal state of the neural network has been investigated, the activation patterns have been found, and the results have been 3D visualized.

The largest variation in the data is not in relation to the paradigm. This is in accordance with reported findings (for example [42]). I find that the paradigm related variation need not to be associated with the principal component corresponding to the number of runs.

The neural network analysis with preprocessing of singular value decomposition / principal component analysis is able to intrasession, interrun, out-of-basis generalize.

The various errors in the signal cause the neural network to require varying network complexity. Intrarun generalization can be obtained to a high degree, and algebraic generalization error estimates usually favor a network comparable to the optimal network for the intrarun test sets. But the optimal architecture for the out-of-basis interrun test sets do not correspond to the algebraic generalization estimates: Here the generalization estimates do not work with the highly clustered data.

The neural network analysis with the entropic costfunction is especially sensitive to presented misclassifications in the training and test sets. The reverse classifications will penalize the entropic function significantly, and the neural network must overfit to cope with these. The hemodynamic response found in fMRI studies is a problem. Robust statistics (for example taking the median value) can not compensate for the large amount of errors. Post flange scan must be masked out. The deactivation flange is especially long, and the variation in its' length makes it hard to model the hemodynamic response.

The saliency map shows good accordance with results from other analysis methods, and it identified blobs that could be explained by prior neurological knowledge. The difference to other methods is, that the neural network saliency map gives an activation pattern with much higher contrast. This is explainable with the way the saliency map is calculated: The squared weights and voxels. It might be said that the neural network gives a too optimistic map for where the neurosurgeon might navigate. The elitarian weightening of the blobs might not reflect all the voxels that participate in the mental process. Though an iterative — but time consuming — scheme could be suggested with the application of voxel deletions.

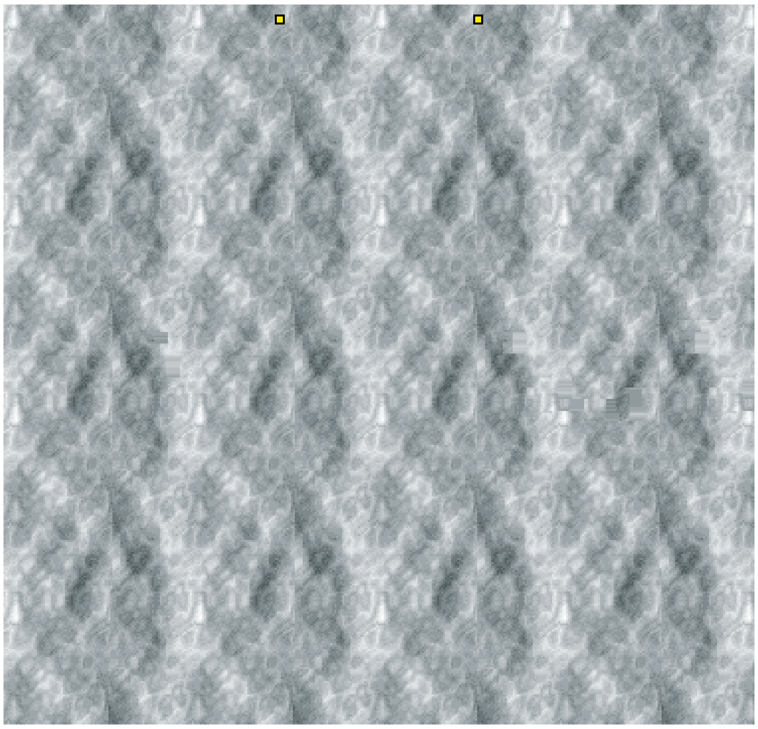
For some of the blobs the signal has been integrated, and the activity in the somatosensoric blob shows a fine accordance with the paradigm and other fMRI studies [70].

The 3D visualizations of the functional analysis results have been among the worlds first serious VRML implementations. Subjective evaluations of the visualizations were positive.

The hidden representation are not orthogonal. In my particular study the hidden units can even melt together: The space they spans can be quite directional.

During a weights' decay of importance on its' way to pruning, the neural network begins to distinguish between groups of input that are trained to the same output.

A new use of the state space was proposed by [43]: Left and right handed seems to have a fundamental different activation. These would be distributed unequally in the state space. I suggest that the state space of the hidden units is used in connection with multi-subject studies, where the subjects are grouped according to for example laterality and sex.



Appendix A

Derivations

A.1 Principal Components Analysis

These calculation were first carried out in [59]:

$$\begin{aligned}
\mathbf{X} &= \mathbf{B\Gamma C}^\top && \text{(A.1)}\\
\mathbf{X} &= \mathbf{B\Gamma C}^{-1} \\
\mathbf{XC} &= \mathbf{B\Gamma} \\
\mathbf{XC} &= \mathbf{BM}^{\frac{1}{2}} \\
\mathbf{B} &= \mathbf{XCM}^{-\frac{1}{2}}
\end{aligned}$$

Firstly a little help:

$$\begin{aligned}
\mathbf{C}^\top\mathbf{T} &= \mathbf{C}^\top\mathbf{CMC}^\top && \text{(A.2)}\\
&= \mathbf{C}^{-1}\mathbf{CMC}^\top \\
&= \mathbf{MC}^\top && \text{(A.3)}
\end{aligned}$$

Then the actual projection:

$$\begin{aligned}
\mathbf{I} &= \mathbf{B}^\top\mathbf{X} && \text{(A.4)}\\
&= \left(\mathbf{XCM}^{-\frac{1}{2}}\right)^\top \mathbf{X} \\
&= \mathbf{M}^{-\frac{1}{2}}\mathbf{C}^\top\mathbf{X}^\top\mathbf{X} \\
&= \mathbf{M}^{-\frac{1}{2}}\mathbf{C}^\top\mathbf{T} \\
&= \mathbf{M}^{-\frac{1}{2}}\mathbf{MC}^\top \\
&= \mathbf{M}^{\frac{1}{2}}\mathbf{C}^\top. && \text{(A.5)}
\end{aligned}$$

A.2 Derivative

Deriving the cost function with respect to the weights is easiest done, when viewing it as an exercise in using the chain rule:

$$\frac{\partial E}{\partial v_{hi}} = \sum_{p}^{n_p} \sum_{o}^{n_o} \frac{\partial E}{\partial o_o^\mu} \frac{\partial o_o^\mu}{\partial \o_o^\mu} \frac{\partial \o_o^\mu}{\partial h_h^\mu} \frac{\partial h_h^\mu}{\partial \hbar_h^\mu} \frac{\partial \hbar_h^\mu}{\partial v_{hi}}. \tag{A.6}$$

A.2.1 The Square Errorfunction

$$\frac{\partial E_{sq}}{\partial v_{hi}} = \sum_{p}^{n_p} \sum_{o}^{n_o} 2\left(o_o^p - d_o^p\right) w_{oh} \left[1 - \left(h_h^p\right)^2\right] i_i^p, \tag{A.7}$$

Were some of the terms can be moved outside the summations, either the term v_{hi} or i_i^p:

$$\frac{\partial E_{sq}}{\partial v_{hi}} = 2\sum_{o}^{n_o} w_{oh} \sum_{p}^{n_p} \left(o_o^p - d_o^p\right) \left[1 - \left(h_h^p\right)^2\right] i_i^p. \tag{A.8}$$

Deriving the first order derivative once again the second order derivative comes out. Two terms are depended upon v_{hi}: o_o^p and h_h^p, so I will use the multiplication rule, getting the full newton derivative:

$$\begin{aligned}
\frac{\partial^2 E_{sq}}{\partial v_{h_2 i_2} \partial v_{h_1 i_1}} &= 2\sum_{o}^{n_o} w_{oh_1} \sum_{p}^{n_p} i_{i_1}^p \quad \Bigg[\left[1 - \left(h_{h_1}^p\right)^2\right] \frac{\partial}{\partial v_{h_2 i_2}} \left(o_o^p - d_o^p\right) \\
&\qquad + \left(o_o^p - d_o^p\right) \frac{\partial}{\partial v_{h_2 i_2}} \left[1 - \left(h_{h_1}^p\right)^2\right]\Bigg] \\
&= 2\sum_{o}^{n_o} w_{oh_1} \sum_{p}^{n_p} i_{i_1}^p \quad \Bigg[\left[1 - \left(h_{h_1}^p\right)^2\right] w_{oh_2} \left[1 - \left(h_{h_2}^p\right)^2\right] i_{i_2}^p \\
&\qquad - \begin{cases} \left(o_o^p - d_o^p\right) 2h_{h_2}^p \left[1 - \left(h_{h_2}^p\right)^2\right] i_{i_2}^p & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \Bigg]
\end{aligned} \tag{A.9}$$

The gauss approximation – or by some called the Levenberg-Marquardt approximation – of the second order derivative states that the term $(o_o^p - d_o^p)$ is comparatively small, and thus this term can be neglected:

$$\frac{\partial^2 E_{sq}}{\partial v_{h_2 i_2} \partial v_{h_1 i_1}} = 2\sum_{i}^{n_o} w_{oh_1} w_{oh_2} \sum_{p}^{p} i_{i_1}^p i_{i_2}^p \left[1 - \left(h_{h_1}^p\right)^2\right] \left[1 - \left(h_{h_2}^p\right)^2\right]. \tag{A.10}$$

Both the full newton and the gauss-newton derivatives are Hessian *matrices* with a size of $n_v \times n_v$. The pseudo-gauss-newton the diagonal approximation only taking elements from the Hessian where $i_1 h_1 = i_2 h_2$:

$$\frac{\partial^2 E_{sq}}{\partial v_{hi}^2} = 2\sum_{i}^{n_o} w_{oh}^2 \sum_{p}^{n_p} \left(i_i^p\right)^2 \left[1 - \left(h_h^p\right)^2\right]^2, \tag{A.11}$$

ending up with a *vector* sized n_v.

A.2.2 The Entropic Errorfunction

$$E_{entr} = \sum_{p=1}^{n_p} \sum_{o=1}^{n_o} \left[\frac{1}{2} (1 + d_o^p) \ln \frac{1 + d_o^p}{1 + o_o^p} + \frac{1}{2} (1 - d_o^p) \ln \frac{1 - d_o^p}{1 - o_o^p} \right] \tag{A.12}$$

The logarithmic function needs not to be the natural but it makes the derivative simpler.

The complex expression of the network equation and the error function resolves into a simple first order derivative for the output weights:

$$\begin{aligned}
\frac{\partial E_{entr}}{\partial w_{oh}} &= \sum_p \left[\frac{1}{2} (1 + d) \frac{1 + o}{1 + d} \frac{1 + d}{-(1 + o)^2} \left(1 - o^2\right) h \right. \qquad \text{(A.13)} \\
&\quad \left. + \frac{1}{2} (1 - d) \frac{1 - o}{1 - d} \frac{1 - d}{(1 - o)^2} \left(1 - o^2\right) h \right] \\
&= \sum \left[-\frac{1}{2} \frac{1 + d}{1 + o} \left(1 - o^2\right) h + \frac{1 - d}{1 - o} \left(1 - o^2\right) h \right] \\
&= \sum \left[\frac{1}{2} \frac{-(1 + d)(1 - o) + (1 - d)(1 + o)}{(1 + o)(1 - o)} \left(1 - o^2\right) h \right] \\
&= \sum \left[\frac{1}{2} \frac{2o - 2d}{1 - o^2} \left(1 - o^2\right) h \right] \\
&= \sum_p (o_o^p - d_o^p) h_h^p
\end{aligned}$$

The second order derivative becomes:

$$\frac{\partial^2 E_{entr}}{\partial w_{o_2 h_2} \partial w_{o_1 h_1}} = \begin{cases} \sum\limits_p h_{h_1}^p \left(1 - (o_o^p)^2\right) h_{h_2}^p & \text{fo } o_1 = o_2 \\ 0 & \text{fo } o_1 \neq o_2 \end{cases} \tag{A.14}$$

The first order derivative of the hidden layer weights is:

$$\begin{aligned}
\frac{\partial E_{entr}}{\partial v_{hi}} &= \sum_p \sum_o \left[\frac{1}{2} (1 + d) \frac{1 + o}{1 + d} \frac{-(1 + d)}{(1 + o)^2} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \right. \qquad \text{(A.15)} \\
&\quad \left. + \frac{1}{2} (1 - d) \frac{1 - o}{1 - d} \frac{(1 - d)}{(1 - o)^2} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \right] \\
&= \sum \sum \left[-\frac{1}{2} \frac{1 + d}{1 + o} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \right. \\
&\quad \left. + \frac{1}{2} \frac{1 - d}{1 - o} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \right] \\
&= \sum \sum \frac{1}{2} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \frac{(1 - d)(1 + o) - (1 + d)(1 - o)}{(1 - o)(1 + o)} \\
&= \sum \sum \frac{1}{2} \left(1 - o^2\right) w_{oh} \left(1 - h^2\right) i_i^p \frac{2(o - d)}{(1 - o)^2}
\end{aligned}$$

$$= \sum_p \sum_o \left(o_o^p - d_o^p\right) w_{oh} \left(1 - (h_h^p)^2\right) i_i^p$$

Deriving further on to the second order derivative, it can be observed that there are two terms that contains the derivative variables:

$$\sum_p \sum_o (\underbrace{o_o^p}_{v_{hi}} - d_o^p) w_{oh} (1 - \underbrace{(h_h^p)^2}_{v_{hi}}) i_i^p \tag{A.16}$$

So the multiplication rule is used:

$$\begin{aligned} \frac{\partial^2 E_{entr}}{\partial v_{h_2 i_2} \partial v_{h_1 i_1}} &= \sum_p \sum_o \Bigg[w_{oh_1} \left(1 - \left(h_{h_1}^p\right)^2\right) i_{i_1}^p \left(1 - \left(o_{o_2}^p\right)^2\right) w_{oh_2} \left(1 - \left(h_{h_2}^p\right)^2\right) i_{i_2}^p \\ &\quad + \begin{cases} \left(o_o^p - d_o^p\right) w_{oh_1} i_{i_1}^p \,(-1)\, 2 h_{h_1}^p \left(1 - \left(h_{h_2}^p\right)^2\right) i_{i_2}^p & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \Bigg] \\ &= \sum_p \sum_o \Bigg[w_{oh_1} w_{oh_2} i_{i_1}^p i_{i_2}^p \left(1 - \left(h_{h_1}^p\right)^2\right) \left(1 - \left(h_{h_2}^p\right)^2\right) \left(1 - \left(o_{o_2}^p\right)^2\right) \\ &\quad - \begin{cases} 2 \left(o_o^p - d_o^p\right) w_{oh_1} i_{i_1}^p i_{i_2}^p h_{h_1}^p \left(1 - \left(h_{h_2}^p\right)^2\right) & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \Bigg] \end{aligned} \tag{A.17}$$

If the total Hessian is used the cross derivatives between the hidden and output weights must be found. First the short wide[1] derivative:

$$\begin{aligned} \frac{\partial^2 E_{entr}}{\partial v_{h_2 i_2} \partial w_{o_1 h_1}} &= \frac{\partial}{\partial v_{i_2 h_2}} \left[\sum_p \left(o_{o_1}^p - d_{o_1}^p\right) h_{h_1}^p \right] \\ &= \sum_p \Bigg[h_{h_1}^p \left(1 - \left(o_{o_1}^p\right)^2\right) w_{o_1 h_2} \left(1 - \left(h_{h_2}^p\right)^2\right) i_{i_2}^p \\ &\quad + \begin{cases} \left(o_{o_1}^p - d_{o_1}^p\right) \left(1 - (h_h^p)^2\right) i_{i_1}^p & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \Bigg] \end{aligned} \tag{A.18}$$

Then the tall narrow[1] derivative:

$$\begin{aligned} \frac{\partial^2 E_{entr}}{\partial w_{o_2 h_2} \partial v_{h_1 i_1}} &= \frac{\partial}{\partial w_{o_2 h_2}} \left[\sum_p \sum_{o_1} \left(o_{o_1}^p - d_{o_1}^p\right) w_{o_1 h_1} \left(1 - \left(h_{h_1}^p\right)^2\right) i_{i_1}^p \right] \\ &= \sum_p \Bigg[\left[w_{o_2 h_1} \left(1 - \left(h_{h_1}^p\right)^2\right) i_{i_1}^p \left(1 - \left(o_{o_2}^p\right)^2\right) h_{h_2}^p \right] \\ &\quad + \begin{cases} \left[\left(o_{o_2}^p - d_{o_2}^p\right) \left(1 - (h_h^p)^2\right) i_{i_1}^p \right] & \text{for } h_1 = h_2 \\ 0 & \text{for } h_1 \neq h_2 \end{cases} \Bigg] \end{aligned} \tag{A.19}$$

The last 2 should be the same when one of them is transposed.

[1]For a network with fewer outputs than inputs

The "Gauss Newton" Approximation

An approximation similar to the Gauss Newton with the squared error function can be used with the entropic error. If the entropic error function is differentiated just past the output activation function with respect to all weights, the following two additive separable terms comes out:

$$\frac{\partial^2 E_{entr}}{\partial u_{u_1} \partial u_{u_2}} = \sum_p \sum_o \left[\left(1 - {o_o^p}^2\right) \frac{\partial \o_o^p}{\partial u_{u_1}} \frac{\partial \o_o^p}{\partial u_{u_2}} + (o_o^p - d_o^p) \frac{\partial^2 \o_o^p}{\partial u_{u_1} \partial u_{u_2}} \right]$$

Again if the term $(o_o^p - d_o^p)$ is neglected, the result is a simpler derivative.

Appendix B

Statistics

B.1 Multivariate Statistics

	Univariate	Multivariate	
Mean	μ	μ	Mean
Varians	σ^2	$\mathbf{\Sigma}$	Dispersion
Normal distribution	$N(\mu, \sigma^2)$	$N_p(\mu, \mathbf{\Sigma})$	Multi dimensional normal distribution
Students t	$t(m)$	$T^2(p, m)$	Hotellings T^2
Chi squared	$\chi^2(m)$	$W_p(\mathbf{\Sigma}, m)$	Wishart
F distribution	F	Λ	Wilk's Lambda

Table B.1: Comparison of univariate and multivariate statistics

The test distributions are the centered versions. There is an asymmetry between the Students t and the Hotelling's T^2: The T^2 is the multi dimensional equivalence to the t, but also being *squared*. Furthermore is the univariate σ the standard variation and the multivariate Σ is the "standard variation squared".

B.1.1 Conversions

T^2-distribution

With the distributions:

$$\bar{\mathbf{x}} \in N_{n_h}\left(\mu, \frac{1}{n_p}\mathbf{\Sigma}\right)$$

$$\mathbf{S} \in W_{n_h}\left(n_p - 1, \frac{1}{n_p - 1}\mathbf{\Sigma}\right)$$

the one sample test variable T^2 for the distance to the ellipsoidal center can be approximated by:

$$\frac{n_p - n_h}{(n_p - 1)\, n_h} T^2_{n_h, n_p} = F_{n_h, n_p - n_h}$$

and the two sample test looses a degree of freedom:

$$\frac{n_p - n_h - 1}{(n_p - 2)\, n_h} T^2_{n_h, n_p} = F_{n_h, n_p - n_h - 1} \tag{B.1}$$

where the n_p here is the total number of patterns for both samples:

$$n_p = n_{p_1} + n_{p_2}$$

Λ-distribution

The Λ distribution might be approximated by the F distribution — of course: the **W** and **T** matrices are a multidimensional variance:

$$F \approx \frac{1 - \Lambda^{1/t}}{\Lambda^{1/t}} \frac{vt + 1 - \frac{1}{2}pq}{pq}$$

$$F_{pq, vt + 1 - \frac{1}{2}pq}$$

were the t and v helping variables are defined to be:

$$\begin{aligned} t &= \begin{cases} 1, & p^2 + q^2 = 5 \\ \sqrt{\frac{p^2 q^2 - 4}{p^2 + q^2 - 5}}, & p^2 + q^2 \neq 5 \end{cases} \\ v &= \frac{1}{2} (2r + q - p - 1)\,. \end{aligned}$$

"W"-distribution

A grand test size presented in [10] which cites [4] does the Box's M in a little different way:

$$\mathrm{W}_1 = \frac{\prod_{g=1}^{n_g} [\det(\mathbf{W}_g)]^{\frac{1}{2}\left(n_{p_g} - 1\right)}}{[\det(\mathbf{W})]^{\frac{1}{2}(n - n_g)}} \frac{(n_p - n_g)^{\frac{1}{2} n_h (n_p - n_g)}}{\prod_{g=1}^{n_g} \left(n_{p_g} - 1\right)^{\frac{1}{2} n_h \left(n_{p_g} - 1\right)}} \tag{B.2}$$

The log of the size can be compared to the χ^2-distribution:

$$P\{-2\rho \log W \leq z\} \approx P\left\{\chi^2(f) \leq z\right\} + \omega_2 \left[P\left\{\chi^2(f + 4) \leq z\right\} - P\left\{\chi^2(f) \leq z\right\}\right] \tag{B.3}$$

where the different factors are:

$$
\begin{aligned}
f &= \frac{1}{2}\left(n_g - 1\right) n_h \left(n_h - 1\right) && \text{(B.4)}\\
\rho &= 1 - \left(\sum_g \frac{1}{n_{p_g}} - \frac{1}{n_p}\right) \frac{2n_h^2 + 3n_h - 1}{6\left(n_h + 1\right)\left(n_g - 1\right)}\\
\omega_2 &= \frac{1}{48\rho^2} n_h \left(n_h - 1\right)\left[\left(n_h - 1\right)\left(n_h + 2\right)\left(\sum_g \frac{1}{n_g^2} - \frac{1}{n_p^2}\right) - 6\left(n_g - 1\right)\left(1 - \rho\right)^2\right]
\end{aligned}
$$

B.2 Rank Correlations

Rank correlations [32] are correlations in connection with ordinal data. Here I will describe 2 rank correlations which both are particular cases of the general correlation coefficient. The data ranks should here be described by integers from one to the number of elements.

The following example is from [32]:

Boys	A	B	C	D	E	F	G	H	I	J
Math	7	4	3	10	6	2	9	8	1	5
Music	5	7	3	10	1	9	6	2	8	4

(B.5)

Here are 10 boys' abilities in mathematics and music put in 2 ordinal rankings. A rank correlation should tell if a boy being bright in mathematics is likely to be musical compared to the other 9.

B.2.1 Kendall τ rank correlation

The τ uses an S which counts the number of pairs out of order in one ranking when the other ranking is used as ordered reference:

$$\tau = \frac{S}{\frac{1}{2}n\left(n-1\right)} \tag{B.6}$$

The number S is normalized by the number of possible pair permutations: n being the number of individuals: in the example B.5 $n = 10$.

This could also be viewed as the number of bubble sort shiftings.

If there are *tied ranks*, that is, 2 or more have equal rankings there should be an other normalization:

$$\tau_{tied} = \frac{S}{\sqrt{\frac{1}{2}n\left(n-1\right) - T_\tau}\sqrt{\frac{1}{2}n\left(n-1\right) - U_\tau}} \tag{B.7}$$

where the T and U are defined to be:

$$
\begin{aligned}
T_\tau &= \frac{1}{2}\sum_t^{n_t} n_{t_t}\left(n_{t_t} - 1\right) && \text{(B.8)}\\
U_\tau &= \frac{1}{2}\sum_u^{n_u} n_{u_u}\left(n_{u_u} - 1\right)
\end{aligned}
$$

n_{t_t} and n_{u_u} are the number of individuals in a tie t and u in the first and second ranking respectively. The summation is running over all ties (n_t and n_u) in that rank.

The tied rankings of the individuals must be "weighted": Considered the example were the boy A and E are ranked equal, then their rankings should be $6\frac{1}{2}$, while all the other 8 individuals should not get their rankings changed.

B.2.2 Spearman's ρ rank correlation

Spearman's ρ is simply the sum of squared ordinal distance d between the two rankings with a normalization (and an offset):

$$\rho = 1 - \frac{6}{n^3 - n} \sum d^2 \tag{B.9}$$

In the example the first couple of contribution would be:

$$\rho_{example} = 1 - \frac{6}{10^3 - 10} \left[(7-5)^2 + (4-7)^2 + \ldots \right] \tag{B.10}$$

The Spearman ρ also has a tied rank correlation:

$$\rho_{tied} = \frac{\frac{1}{6}\left(n^3 - n\right) - T_\rho - U_\rho - \sum d^2}{\left[\frac{1}{6}\left(n^3 - n\right) - T_\rho - U_\rho\right] \sqrt{1 - \frac{\left(T_\rho - U_\rho\right)^2}{\left[\frac{1}{6}\left(n^3-n\right) - T_\rho - U_\rho\right]^2}}} \tag{B.11}$$

Where T_ρ and U_ρ are defined by:

$$\begin{aligned} T_\tau &= \frac{1}{12} \sum_t^{n_t} \left(n_{t_t}^3 - n_{t_t}\right) \\ U_\tau &= \frac{1}{12} \sum_u^{n_u} \left(n_{u_u}^3 - n_{u_u}\right) \end{aligned} \tag{B.12}$$

B.2.3 Test of significance

For a test of significance the discrete frequency distributions can be approximated to the normal distribution with the variances. For the τ pair count number S the variance is

$$\sigma_S^2 = \frac{1}{18} n \left(n-1\right)\left(2n+5\right) \tag{B.13}$$

For the Spearman's ρ:

$$\sigma_\rho^2 = \frac{1}{n-1} \tag{B.14}$$

which for the sum of squared ordinal distances is:

$$\sigma_{\left(\sum d^2\right)}^2 = \left(\frac{n^3 - n}{6}\right)^2 \frac{1}{n-1} \tag{B.15}$$

The test with Spearman's ρ is also called the Hotelling-Pabst test [8].

Appendix C

VRML

Virtual Reality Modeling Language[1] [7], abbreviated VRML, is a 3D data description format, but is meant to be extended into a virtual reality language that also can contain for example sound. It looks as if the language is going to be the standard language for defining 3D objects, which makes it interesting in connection with visualization of the brain. An other advantage is that the language has 2 definitions that are supposed to work with the internet: a call and transfer to an other link for example an HTML-hypertext page (`WWWAnchor`), and an including (`WWWInline`). With the exception of included files the input format is text oriented with case sensitivity. The language defines only static objects, — behavior is not implemented yet

The data stream can be viewed as describing hierarchical input to a stack machine where *node* states are pushed onto the stack to be saved from modification of lower levels in the hierarchy. Nodes come, in an object oriented manner, with a predefined state through its *fields* and changes to these either accumulate or overwrite if the state is not pushed with the `Separator` node. The state is restored when the end of the separator is reached (a right tuborg).

Nodes can be "physical" objects, that is, geometrical nodes. An example is the polygon. They can also be properties for example fontstyle, material and texture. Other groups of nodes are: Transform nodes, camera nodes, light nodes and group nodes.

The sizes of the objects are changed by redefining the fields of the objects. By default all objects are of the size 1, with the exception of the `AsciiText` which — by mistake — was defined to size 10.

The Complex and text geometrical objects are created at origo. If the object is to be moved it has to be through the already accumulated transformation. The transformation is the ordinary 4×4 3D transformation matrix multiplication [26]. There are predefined "high-level" keywords such as `Rotation`, `Scale`, and `Translation`, but one can also go directly to the matrix transformation matrix with `MatrixTransform`. The coordinate system is right-handed.

Nodes can be given an identifier with the keyword `DEF` which can be used for multiply

[1] VRML is for the moment under definition, especially is behavior to be defined, so everything described here must be regarded as description of the present specification

Type	VRML node	Description
Simple	`IndexedFaceSet`	Polygons. The polygons can have more vertices than 3. The polygons are define through indices into the present `Coordinate3` list.
	`IndexedLineSet`	Polylines. Like the polygon definition the polyline uses the `Coordinate3` list. The thickness of the line is not defined.
	`PointSet`	Points. Takes uses the `Coordinate3` list to define its points
Complex	`Cube`	An ordinary rectangular box.
	`Sphere`	Like the cone and the cylinder the sphere is transformed into a not defined number of polygons before being rendered.
	`Cone`	The bottom and sides can be switch on and off.
	`Cylinder`	The bottom, top, and side can be switch on and off.
Text	`AsciiText`	The letters are automatically transform into either planar polylines or polygons.

Table C.1: VRML geometrical object nodes.

instancing using the `USE` keyword. Funny enough the geometrical nodes are not just defined with the `DEF` keyword but also instanced, that is, rendered. A way of over coming this, only defining the geometrical node, is to use the switch trick:

```
Switch {
        DEF BiggerCube Cube {
                width 10
        }
}
Translation { translation 100 0 0 }
USE BiggerCube
```

The `Switch` `whichChild` field is by default sat to -1, which means that none of the children (here is it the DEF cube) are transversed. Without the `Switch` there would have been rendered 2 `Cube`s: one at origo, the other at 100 units along the x-axis.

In connection with brain imaging the coordinate system must be clarified: In brain imaging bicommissural coordinates the *z-axis* is defined to be *up*. In VRML the *z-axis* codes for the *depth*! Although it is relatively easy to transform between the two, a problem arises with the use of the *walk viewer* which behavior will be unnatural in the bicommissural coordinates:

Transformation	Explanation	Matrix
Scaling	Diagonal matrix	$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation	Extended Givens matrix. A rotation about an arbitrary axis can be made when using the "linear part" ($3x3$ upper left matrix) as a definition of 3 normal vectors	$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation	"Affine" matrix. The "affine part" different from zero. Each of the elements translate along its axis.	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix}$
Reflection	Identity matrix with negated element. The reflection in a plane, here the xy.	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Shear	An asymmetric matrix	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table C.2: Transformation matrix.

Motion of the viewer in relation to the viewed 3D objects, provides the important motion parallax depth cue. Browsers of the VRML files often have 3 different motion modes: The *walk* mode where the viewer can move in the x-z plane and possibly tilt. The *fly* mode where the viewer can move in all directions, and finally the *examine* mode where it is the object that is moved with the possibility of translation and rotation.

Pure volume definition is not implemented in VRML. A poor man's implementation of volume rendering might be reached through the use of dithered `PointSet`. The points should be dithered because regular grided points produce a too plain picture.

C.1 Example

Here I will describe how to convert the state space clusters and cutting plane into VRML ellipsoides and VRML polygons.

C.1.1 Making a Plane from Neural Network Output Weights

The output weights to a single output unit of a classifier neural network describes a cutting plane. How can that be described in VRML:

Consider a weight vector $\mathbf{w} = \begin{bmatrix} 1 & 2 & -3 & 4 \end{bmatrix}^\top$, the last element representing the bias. We start with the VRML file header:

```
#VRML V1.0 ascii
```

After this comes the main separator encapsulating the whole data:

```
Separator {
```

We can render the plane in an intercept form — 3 vertices on the axis. The problem here is that the size of the plane will vary depending how far the plane is from origo. Instead we define a 2×2 sized plane and by rotation and translation place it properly. The transformations must be accumulated before we can define the plane, so we start with the transformations:

```
Rotation {
        rotation -3 0 -1 1.006854
}

Translation {
        translation 0 -1.069045 0
}
```

The transformation has resulted from this: The cutting plane is to be defined in the x-z plane, that is, it will have a normal vector parallel with the y-axis. The first 3 arguments to the `rotation` keyword are to represent a vector to rotate about. This rotation vector must thus be orthogonal to both the cutting plane normal and the y-axis. Such a vector can be constructed with the crossproduct:

$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & -3 \end{bmatrix} = \begin{bmatrix} -3 & 0 & -1 \end{bmatrix}$$

The last value the `rotation` keyword uses is the angle. All angles in VRML are defined to be in radians:

$$\theta = \text{acos}\left(\frac{\mathrm{w}_2}{\sqrt{\mathrm{w}_1^2 + \mathrm{w}_2^2 + \mathrm{w}_3^2}}\right) = \arccos\left(\frac{2}{\sqrt{1^2 + 2^2 + (-3)^2}}\right) = 1.006854\,\text{rad}$$

The translation has to move the plane from origo to the value indicated by the bias weight:

$$d = \frac{-|w_4|}{\sqrt{w_1^2 + w_2^2 + w_3^2}} = \frac{-4}{\sqrt{1^2 + 2^2 + (-3)^2}} = -1.069045$$

Before defining the object we can put some color and make it half-way transparent:

```
        Material {
                emissiveColor 0.5 0.5 0.5
                transparency 0.5
        }
```

Finally we can begin defining the actual geometry. First the vertices are defined in the x-y plane:

```
        Coordinate3 {
                point [
                        -1 0 1,
                        -1 0 -1,
                        1 0 -1,
                        1 0 1,
                ]
        }
```

Then the vertex connections, — zero indexed and with the "-1" polygon termination value.

```
        IndexedFaceSet {
                coordIndex [
                        0, 1, 2, 3, -1
                ]
        }
```

We end with closing the starting `Separator` with a right tuborg:

```
}
```

C.1.2 Making a Ellipsoide from a Normal distribution

A 3D normal distribution can be described by its contour ellipsoide. The VRML ellipsoide is constructed as a `Sphere` which is deformed and moved by the `MatrixTransform`. We consider a normal distribution defined by:

$$\begin{aligned} \mu &= \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^\top \\ \mathbf{\Sigma} &= \begin{bmatrix} 5.3 & -1.3 & -0.8 \\ -1.3 & 0.8 & 1.3 \\ -0.8 & 1.3 & 8.3 \end{bmatrix} \end{aligned}$$

The mean μ goes straight down in the bottom of the transformation matrix. The dispersion matrix first has to be square rooted before it is put into the transformation matrix — the dispersion describes the variance while the contour ellipsoide should describe the standard deviation. Before the geometric definition a scaling is performed to make the "body-tail" area of the 3D normal distribution equal to the 1D normal distribution.

```
Switch {
        DEF Ellipsoide Separator {
                MatrixTransform {
                        2.2593   -0.4234   -0.1274 0,
                        -0.4234    0.7060    0.3497 0,
                        -0.1274    0.3497    2.8568 0,
                        1 2 3 1,
                }
                Scale { scaleFactor 1.55 1.55 1.55 }
                Sphere { }
        }
}
```

This definition of an ellipsoide can later be instantiated with the keyword `USE`. Here first making it red and half-way transparent:

```
Separator {
        Material {
                emissiveColor 1 0 0
                transparency 0.5
        }
        USE Ellipsoide
}
```

Bibliography

[1] Introduction to MRI. Internet, 1996. `http://www.xray.ufl.edu/~rball/mritutor.html`.

[2] H. Akaike. Fitting Autoregressive Models for Prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247, 1969.

[3] Ole Trier Andersen. En Introduktion til NMR-scanning. Elektronics Institute, Technical University of Denmark, Lyngby, Denmark, 1984.

[4] T. W. Anderson. *An Introduction to Multivariate Statistic Analysis*. Wiley, New York, 1958.

[5] Rita L. Atkinson, Richard C. Atkinson, Edward E. Smith, Daryl J. Bem, and Ernest R. Hilgard. *Introduction to Psychology*. Harcourt Brace Jovanovich, San Diego, tenth edition, 1990. ISBN 0–15–543688–0.

[6] Peter A. Bandettini and Eric C. Wong. Echo-Planar Magnetic Resonance Imaging of Human Brain Activation. In Franz Schmitt, Michael Stehling, and Robert Turner, editors, *Echo Planar Imaging*. Springer Verlag, 1. edition, 1996. MGH NMR Center and University of Califonia, email: `pab@nmr.mgh.harvard.edu`.

[7] Gavin Bell, Anthoy Parisi, and Mark Pesce. The Virtual Reality Modeling Language. Specification, Internet, 1995. `http://www.sdsc.edu`.

[8] James V. Bradley. *Distribution-free Statistical Tests*. Prentice-Hall, 1968.

[9] R. Campanella, C. Casieri, F. De Luca, B. C. De Simone, and B. Maraviglia. Basic Principles of NMR Imaging. In R. Guzzardi, editor, *Physics and Engineering of Medical Imaging*, number 119 in NATO ASI: Applied Sciences, pages 489–499. Martinus Nijhoff Publishers, Dordrecht, 1987.

[10] Knut Conradsen. *En Introduktion til Statistik*. Institute of Mathematical Modeling, Technical University of Denmark, Lyngby, 4. edition, 1984.

[11] Douglas Cruickshank. Navigating the brain. *IRIS Universe: The Magazine of Visual Computing*, (25):54–58, 1993.

[12] Georg Cybenko. Continuous Valued Neural Network with Two Hidden Layers are Sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA, 1988.

[13] Agamemnon Despopoulos and Stefan Silbernagl. *Color Atlas of Physiology*. Thieme, Stuttgart, 4. edition, 1991. ISBN 3–13–545004–X.

[14] Jim Foley and Bill Ribarsky. Next-generation Data Visualization Tools. In L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization, Advances and Challenges*, chapter 7, pages 103–127. Academic Press, London, 1994. ISBN 0–12–227742–2.

[15] Peter T. Fox, Mark Stewart, and Thomas Ganslandt. BrainMap Database, Research Imaging Center, 1996. The BrainMap Database is accessable via the internet address: `http://ric.uthscsa.edu`.

[16] Karl J. Friston, A. P. Holmes, K. J. Worsley, J.-P. Poline, C. D. Frith, and R. S. J. Frackowiak. Statistical Parametric Maps in Functional Imaging: A General Linear Approach. email: `karl@cu.rpms.ac.uk`.

[17] Karl J. Friston, J.-P. Poline, S. Strother, A. P. Holmes, C. D. Frith, and R. S. J. Frackowiak. A Multivariate Analysis of PET Activation Studies. email: `karl@cu.rpms.ac.uk`.

[18] Markus H. Gross. Subspace Methods for the visualization of multidimensional data sets. In L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization, Advances and Challenges*, chapter 11, pages 171–184. Academic Press, London, 1994. ISBN 0–12–227742–2.

[19] Charles B. Grossman and Robert T. Anger. Brain Anatomy. In Carlos F. Ganzales, Charles B. Grossman, and Joseph C. Masdeu, editors, *Head and Spine Imaging*, Wiley Medical Publication, chapter 7, pages 133–222. John Wiley & Sons, New York, 1. edition, 1985. ISBN 0–471–89747–7, 617'.510757.

[20] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics, The Approach Based on Influence Functions*. Wiley Series in Probability and Mathematic Statistic. John Wiley and Sons, New York, 1. edition, 1985.

[21] Lars Kai Hansen. Associated professor at DSP IMM DTU. `lkhansen@ei.dtu.dk`.

[22] Lars Kai Hansen and Jan Larsen. Unsupervised Learning and Generalization. ., 1996. `http://www.ei.dtu.dk`.

[23] Lars Kai Hansen and Carl Edward Rasmussen. Pruning from Adaptive Regularization. *(Neural Computation)*, 1993. `http://www.ei.dtu.dk`.

[24] Lars Kai Hansen, Carl Edward Rasmussen, Claus Svarer, and Jan Larsen. Adaptive regularization. In *Proceedings of the 1994 IEEE NNSP Workshop*, 1994. `http://www.ei.dtu.dk`.

[25] B. Hassibi, D. Stock, and G. Wolff. Optimal Brain Surgeon and General Network Pruning. *Neural Computation*, (4):1–8, 1992.

[26] Donald Hearn and M. Pauline Baker. *Computer Graphics*. Prentice Hall International Editions. Prentice Hall, 1986. Features also 3D graphics.

[27] Gabor T. Herman and Jayaram K. Udupa. Display of 3-D Digital Images: Computational Foundations and Medical Applications. *IEEE Computer Graphics and Applications*, 3(5):39+, 1983.

[28] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, Califonia, USA, 1. edition, 1991. Santa Fe Institute.

[29] Georg E. Hinton. How Neural Networks Learn from Experience. *Scientific American*, 267(3):105–109, 1992.

[30] Jesper James Jensen. 3D Visualisering. Technical report, Electronics Institute, Technical University of Denmark, 1995. `http://www.ei.dtu.dk`.

[31] Arie Kaufman. Trends in Volume Visualization and Volume Graphics. In L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization, Advances and Challenges*, chapter 7, pages 103–127. Academic Press, London, 1994. ISBN 0–12–227742–2.

[32] Maurice G. Kendall. *Rank Correlation Methods*. Griffin, London, 4. edition, 1970.

[33] Seong-Gi Kim and Kamil Ugurbil. Functional Magnetic Resonance Imaging of the Human Brain, 1996. Center for Magnetic Resonance Research, Department of Radiology, University of Minnesota Medical School, email `kim@geronimo.drad.umn.edu`.

[34] Ulrik Kjems, Peter A. Philipsen, Lars Kai Hansen, Chin-Tu Chen, and Jon Anderson. A Nolinear 3D MRI Brain Co-registration Method. `http://www.ei.dtu.dk`, 1995.

[35] Carl L. Kramer and Ferdinando S. Buonanno. Physical Principles of Nuclear Magnetic Resonance and its Application to Imaging. In Carlos F. Ganzales, Charles B. Grossman, and Joseph C. Masdeu, editors, *Head and Spine Imaging*, Wiley Medical Publication, chapter 26, pages 859–887. John Wiley & Sons, New York, 1. edition, 1985. ISBN 0–471–89747–7, 617'.510757.

[36] Wolfgang Krüger and Peter Schröder. Data parallel Volume Rendering. In L. Rosenblum, R. A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization, Advances and Challenges*, chapter 7, pages 103–127. Academic Press, London, 1994. ISBN 0–12–227742–2.

[37] John L. Lancaster, Peter T. Fox, Gwendolyn Davis, and Shawn Mikiten. BrainMap: A Database of Human Functional Brain Mapping. In *The Fifth International Conference: Peace through Mind/Brain Science*, Hammamatsu, Japan, February 1994. `http://ric.uthscsa.edu`.

[38] N. Lange, L. K. Hansen, M. W. Pedersen, and R. L. Savoy adn S. C. Strother. A Concordance Correlation Coefficient Reproducibility of Spatial Activation Patterns. In *Human Brain Mapping*, 1996.

[39] N. Lange and S. L. Seger. Non-linear Fourier analysis of magnetic resonance function neuroimage time series. Submitted to *Journal of Royal Statistical Society, Series C.*, 1996.

[40] Jan Larsen. *Design of Neural Network Filters.* PhD thesis, Technical University of Denmark, 1993. `http://www.ei.dtu.dk`.

[41] Jan Larsen and Lars Kai Hansen. Generalization Performance of Regularized Neural Network Models. In *NNSP*, 1994. `http://www.ei.dtu.dk`.

[42] Benny Lautrup, Lars Kai Hansen, Ian Law, Niels Mørch, Claus Svarer, and Stephen C. Strother. Massive Weight Sharing: A Cure for Extremely Ill-posed Problems. ., 1994. `http://ei.dtu.dk`.

[43] Ian Law. Physician at Rigshospitalet.

[44] Ian Law, Claus Svarer, and Oluf B. Paulson. A Characterization of Cerebral Responses during the Performance of Reflexive Predicted and Antisaccadic Eye Movement. Rigshospitalet, Denmark, 1995.

[45] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal Brain Damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, number 2, pages 598–605, San Meteo, Califonia, 1989. Morgan Kauffmann Publishers.

[46] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D surface Construction Algoritm. *Computer graphics*, 21(4):163+, 1987.

[47] Bent Lundsager and Benny Lønstrup Kristensen. Lineær og Ulineær Modellering af Positron Emission Tomografier. Master's project, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, February 1996. `http://www.ei.dtu.dk`.

[48] Henrik Madsen. *Tidsrækkeanalyse.* IMM, DTU, 1989. `http://www.imm.dtu.dk`.

[49] Mads Hintz Madsen. Ph.D. stud. at DSP IMM DTU.

[50] Mads Hintz Madsen, Lars Kai Hansen, and Jan Larsen. Design and Evaluation of Neural Classifiers — Application to Skin Lesion Classification. In *IEEE Workshop on Neural Networks for Signal Processing*, 1995. `http://www.ei.dtu.dk`.

[51] P. Mansfield. Comparative Evaluation of NMR Imaging Techniques. In R. Guzzardi, editor, *Physics and Engineering of Medical Imaging*, number 119 in NATO ASI: Applied Sciences, pages 500–510. Martinus Nijhoff Publishers, Dordrecht, 1987.

[52] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1994.

[53] Joseph C. Masdeu and Charles B. Grossman. Brain Anatomy. In Carlos F. Ganzales, Charles B. Grossman, and Joseph C. Masdeu, editors, *Head and Spine Imaging*, Wiley Medical Publication, chapter 7, pages 133–222. John Wiley & Sons, New York, 1. edition, 1985. ISBN 0–471–89747–7, 617'.510757.

[54] J.R. Moeller and S. C. Strother. A Regional Covariance Approach to the Analysis of Functional Patterns in Positron Emission Tomographic Data. *Journal of Cerebral Blood Flow and Metabolism*, (11):121–135, 1991.

[55] J.R. Moeller, S. C. Strother, J.J.Sidtis, and D. A. Rottenberg. Scaled Subprofile Model: A Statistical Approach to the Analysis fo Functional Petterns in Positron Emission Tomographic Data. *Journal of Cerebral Blood Flow and Metabolism*, (7):649–658, 1987.

[56] Martin Fodslette Møller. A Scaled Conjugate Gradient Algoritm for Fast Supervised Learning. *Neural Networks*, 6:525–533, 1993. University of Aarhus.

[57] John E. Moody. The *Effective* Number of Parameters: An Analysis of Generalization and Regularisation in Nonlinear Learning Systems. In John E. Moody, S. J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, number 4, San Meteo, Califonia, 1992. Morgan Kauffmann Publishers. `http://www.cse.ogi.edu`.

[58] Niels Mørch, Ulrik Kjems, Lars Kai Hansen, Claus Svarer, Ian Law, Benny Lautrup, Steve Strother, and Kelly Rehm. Visualization of Neural Networks Using Saliency Maps. *ICNN 1995*, 1995. `http://www.ei.dtu.dk`.

[59] Niels Jacob Sand Mørch and Jens Rehhoff Thomsen. Statistisk Analyse af Positron Emission Tomografier. Master's project, Technical University of Denmark, Electronics Institute, August 1994. `http://imm.dtu.dk`.

[60] Noboru Murata, Shuji Yoshizawa, and Shun ichi Amari. Network Information Criterion — Determining the Number of Hidden Units for an Artificial Neural Network Model. *IEEE Transaction on Neural Networks*, 5(6):865–872, 1994. Department of

Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo.

[61] Neuroscape: Journal. Internet, 1996. `http://neurocog.lrdc.pitt.edu/neuroscape/journal/fmri/`.

[62] Michi Ono, Stefan Kubik, and Chad D. Abernathey. *Atlas of the Cerebral Sulci.* Georg Thieme Verlag, Stuttgart, 1990. ISBN 3–13–732101–8.

[63] Morten With Pedersen. Ph.D. student at DSP IMM DTU. `with@ei.dtu.dk`.

[64] Morten With Pedersen. Neural Network Study of fMRI data, 1995.

[65] Morten With Pedersen, Lars Kai Hansen, and Jan Larsen. Pruning with Generalization based Weight Saliencies. 1996. `http://www.ei.dtu.dk`.

[66] William H. Press, Brian P. Flannery, Saul A Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, England, 1. edition, 1988.

[67] Carl Edward Rasmussen. Generalization in Neural Networks. Master's project, Technical University of Denmark, Electronics Institute, August 1993. `http://www.ei.dtu.dk`.

[68] Hans Ravn. *Noter til Statisk og Dynamisk Optimering.* Instistute of Mathematical Modeling, Technical University of Denmark, Lyngby, 1. edition, 1994.

[69] Kelley Rehm, Stephen Strother, Jon R. Anderson, Kirt Schaper, and David A. Rottenberg. Display of Merged Multimodality Brain Images Using Interleaved Pixels woth Independent Color Scales. *Journal of Nuclear Medecin*, 35:1815–1821, 1994. `http://pet.med.va.gov:8080/papers/rehm-94a.html`.

[70] Egill Rostrup. Physician at Hvidovre Hospital, Denmark.

[71] Davis A. Rottenberg. Reproducible Features of Functional Neuroimages. Plan for part of the Human Brain Project, January 1996.

[72] Gordon M. Sheperd. *Neurobiology.* Oxford University Press, Oxford, 3. edition, 1994.

[73] S. C. Strother, Jon R. Anderson, K. A. Schaper, J. J. Sidtis andJ. S. Liow, R. P. Woods, and D. A. Rottenberg. Principal Component Analysis and the Subprofile Scaling Model Compared to Intersubject Averaging and Statistical Parametric Mapping: "Functional Connectivity of the Human Motor System studied with [^{15}O] water PET". VA Medical Center, Minneapolis, Minnesota, USA.

[74] S. C. Strother, N. Lange, R. Savoy, J. R. Anderson, J. J. Sidtis, and L. K. Hansen. Multidimensional State-Spaces for fMRI and PET activation studies. ., 1996.

[75] Claus Svarer, Lars Kai Hansen, and Jan Larsen. On the Design and Evaluation of Tapped-delay lines Neural Networks. In *Advances in Neural Information Processing Systems*, number 2, San Meteo, Califonia, 1989. Morgan Kauffmann Publishers.

[76] J. Talairach and P Tournoux. *Co-planar Stereotaxic Atlas of the Human Brain*. Thieme Medical Publisher Inc, New York, 1988.

[77] R. A. Taube and S. J. Adelstein. A Short History of Modern Medical Imaging. In R. Guzzardi, editor, *Physics and Engineering of Medical Imaging*, number 119 in NATO ASI: Applied Sciences, pages 9–40. Martinus Nijhoff Publishers, Dordrecht, 1987.

[78] Katrin Wedekind. Medical Image Matching. Internet, 1996. German `http://mbi.dkfz-heidelberg.de/mbi/research/matching.html`.

[79] Andreas S. Weigned and Hans Georg Zimmermann. Clearning. In *Third International Conference on Neural Networks in the Capital Markets*, London Business School, October 1995. `http://www.cs.colorado.edu`.

[80] Joseph D. Weissman. State of the Art Magnetic Resonance Imaging. In R. Guzzardi, editor, *Physics and Engineering of Medical Imaging*, number 119 in NATO ASI: Applied Sciences, pages 551–560. Martinus Nijhoff Publishers, Dordrecht, 1987.

[81] R. P. Woods, S. R. Cherry, and J. C. Mazziotta. Rapid automated algoritm for aligment and reslicing PET images. *Journal of Computer Assisted Tomography*, 15:634–639, 1991.

[82] Lizhong Wu and John Moody. A Smoothing Regulizer for Feedforward and Recurrent Neural Networks. *Neural Computation*, 8(3), 1996. `http://www.cse.ogi.edu/~moody`.