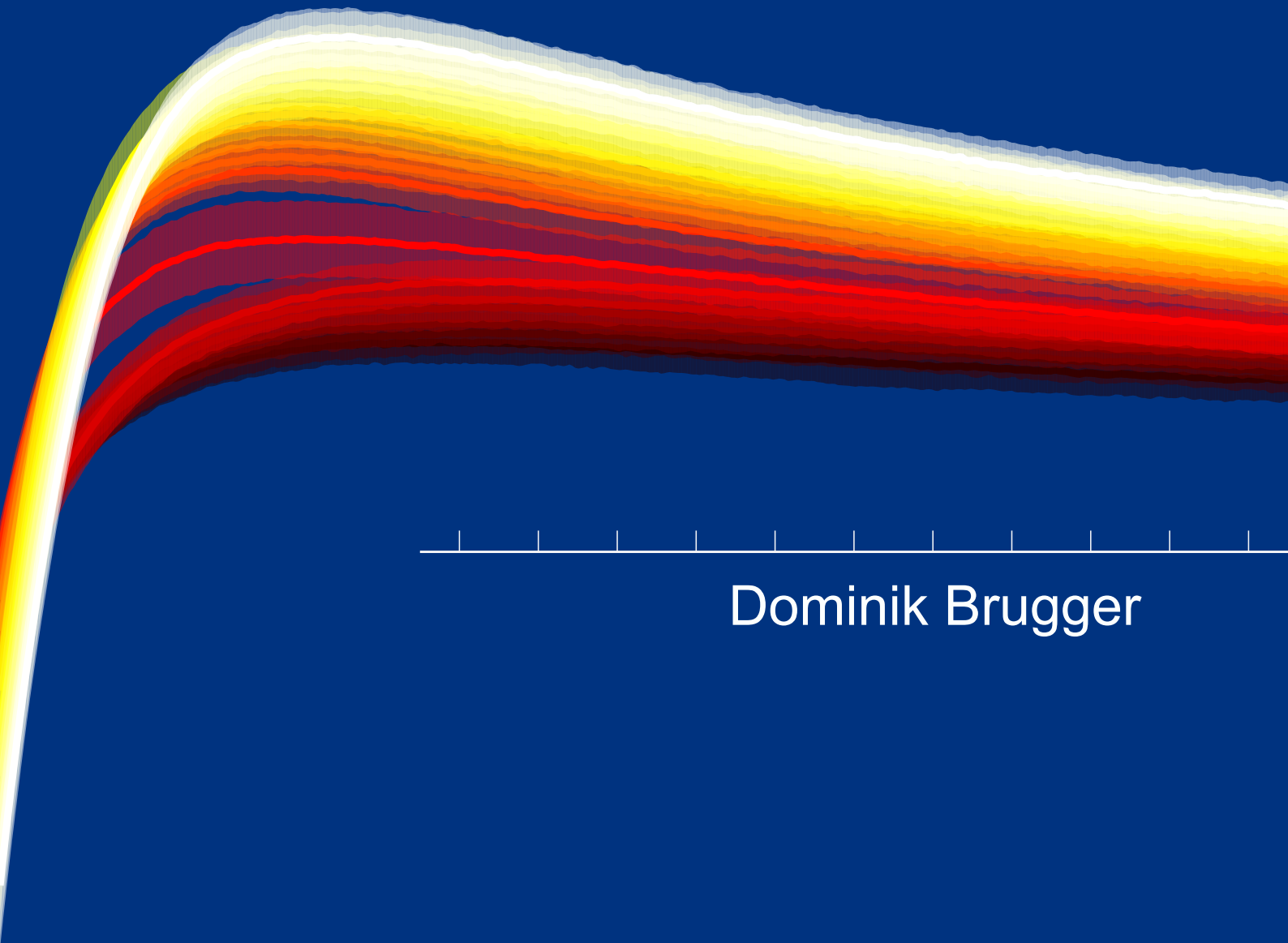
Adaptive microstimulation for stabilizing evoked cortical potentials



Dominik Brugger

Adaptive microstimulation for stabilizing evoked cortical potentials

Dissertation
der Fakultät für Informations- und Kognitionswissenschaften
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dominik Brugger
aus Villingen

Tübingen
2009

Tag der mündlichen Qualifikation:	10.02.2010
Dekan:	Prof. Dr.-Ing. Oliver Kohlbacher
1. Berichterstatter:	Prof. Dr. Wolfgang Rosenstiel
2. Berichterstatter:	PD Dr. Cornelius Schwarz

Acknowledgements

At this point I would like to express my gratitude to all the people that contributed to the success of this thesis' work.

First I would like to thank Prof. Dr. Wolfgang Rosenstiel for giving me the opportunity to work on this challenging topic and for funding the project. Special thanks go to Prof. Dr. Martin Bogdan and PD Dr. Cornelius Schwarz for pointing out the problem of adaptive stimulation, giving continuous encouragement, and engaging in fruitful discussions. I'm also indebted to Dr. Sergejus Butovas for finally "bringing the mammoth down" and inspiring conversations during those seemingly endless lab hours.

Additionally I would like to thank my colleagues at the Department of Computer Engineering for their support, especially my coworkers Mike Bensch and Armin Walter for insightful discussions in our Neuroteam meetings, and Werner Dreher for creating custom electronic devices. Thanks also go to the people from the Hertie Institute, Maik, Petya, Chris, Sinia, Julia, Ursula, and Ute for support and interesting paper discussions as well as to my former students Frank Schulz and Jana Kneslova. For excellent hardware support I would like to thank Hans Löffler, Peter Jesinger, Rainer Mohrlok, and Andreas Möller from Multi Channel Systems.

Finally I would like to thank my family and friends for keeping the faith, in particular Ute, Ruth, Martin, Günther, Barbara, Heribert, Manuel, Katharina, Christoph, Sandra, Frank, Marcus, Nadja, Willi, Hubert, Markus, Melanie, Simone, Rainer, Rosi, Michaela, Kai, Felix, Franziska, Georg, Hans-Jörg, Alexandra, Friedrich, Daniel, Frank, Marion, Steffi, and Steffen.

Abstract

Cortical implants hold the promise to restore lost sensory perceptions, like vision, by using an array of microelectrodes to directly stimulate neural tissue in the corresponding area of the brain. In contrast to retinal implants, cortical implants can aid blind patients even when the information flow from receptors to brain is interrupted in later stages of the visual pathway. Unfortunately evoking stable perceptions by direct stimulation in cortex is currently not possible. One essential unsolved problem is the high variability of evoked cortical potentials caused by an incessantly fluctuating cortical state.

This thesis deals with this problem and proposes to stabilize evoked cortical potentials by adaptive microstimulation, where the intensity of stimulation pulses is continuously adjusted based on the ongoing brain activity. To investigate the feasibility of this approach this work developed an experimental setup with simultaneous recording and stimulation in the barrel cortex of anesthetized rats. A direct and inverse solution using support vector regression is suggested to tackle the control problem associated with adaptive microstimulation. Further algorithmic developments include an application specific kernel function for decoding the cortical state which allows to exploit prior knowledge about the temporal structure of stimulation trials and outperforms other standard kernels. The experimental results recorded in seven animals show for the first time that adaptive microstimulation can stabilize evoked cortical potentials if intensities are chosen from a sub-threshold range. Unfortunately the size of the stabilization effect varies on a time scale of minutes which is due to invalidation of the function learnt by support vector regression. To eliminate the temporal variation in future applications of adaptive microstimulation this work proposes a novel online training algorithm for support vector regression which is suitable for updating the estimated function in a real-time environment and does not require manual tuning of a learning rate. The new algorithm is shown to perform better in terms of convergence speed in comparison to other state of the art algorithms on several benchmark data sets. Together the results presented in this work support the feasibility of adaptive microstimulation and open the perspective to reliably imprint brain activity in future cortical implants.

Zusammenfassung

Die Wiederherstellung einer verlorenen Sinneswahrnehmung, wie zum Beispiel des Sehvermögens, ermöglichen kortikale Implantate. Künstliche Wahrnehmungen werden dabei durch direkte Stimulation des entsprechenden Gehirnareals über eine Reihe von Mikroelektroden hervorgerufen. Im Gegensatz zum Retina-Implantat können kortikale Implantate bei einer blinden Person sogar dann eingesetzt werden, wenn der Informationsfluss zwischen Rezeptor und Gehirn entlang der visuellen Leitungsbahn erst in späteren Stufen unterbrochen ist. Das Erzeugen stabiler Sinneseindrücke durch direkte Stimulation des Gehirns ist derzeit noch nicht möglich. Ein wesentliches Problem dabei ist die starke Schwankung evozierter Potentiale, die durch eine stetig fluktuierende kortikale Aktivität verursacht wird.

Diese Dissertation hat sich mit dieser Schwierigkeit auseinandergesetzt und schlägt zur Stabilisierung evozierter kortikaler Potentiale eine adaptive Mikrostimulation vor, bei der die Intensität der Stromstöße ausgehend von der gegenwärtigen Gehirnaktivität fortlaufend angepasst wird. Um die Machbarkeit dieses Ansatzes zu untersuchen, wurde im Rahmen dieser Arbeit ein experimenteller Aufbau für eine gleichzeitige Aufnahme und Stimulation im Barrel Kortex anästhesierter Ratten entwickelt. Für die Steuerung der Intensitäten werden ein direkter und inverser Lösungsansatz vorgeschlagen und evaluiert, wobei die Schätzung der erforderlichen Funktionen auf Basis experimenteller Daten durch Support Vektor Regression erfolgt. Eine anwendungsspezifische Kern-Funktion, die unter Ausnutzung von Vorwissen über die zeitliche Struktur der Daten, eine Dekodierung der kortikalen Aktivität erlaubt, gehört zu den weiteren algorithmischen Entwicklungen. Im Vergleich mit üblichen Kern-Funktionen erzielt die weiterentwickelte Kern-Funktion eine höhere Präzision bei der Vorhersage der Stimulationsintensität. Die bei sieben Versuchstieren erhobenen experimentellen Ergebnisse zeigen erstmals, dass evozierte Potentiale durch adaptive Mikrostimulation stabilisiert werden können, falls die Stromstöße eine hinreichend geringe Intensität aufweisen. Allerdings schwankt der durch adaptive Mikrostimulation erreichte Effekt innerhalb weniger Minuten, was auf einen Verfall der durch Support Vektor Regression ermittelten Funktion zurückzuführen ist. Zur Vermeidung dieses Verfalls in zukünftigen Anwendungen adaptiver Mikrostimulation schlägt diese Arbeit einen neuen Algorithmus zum Online-Training der Support Vektor Regression vor. Der Algorithmus ist besonders für eine Aktualisierung der geschätzten Funktion in einer Echtzeit Umgebung geeignet und benötigt keine manuelle Einstellung einer Schrittweite. Mit dem neuen Algorithmus lässt sich, bei gleichem Zeitaufwand pro Iteration, im Vergleich mit anderen aktuellen Verfahren eine schnellere Konvergenz der Vorhersagefehlers auf verschiedenen Datensätzen erreichen. Zusammengenommen bestätigen die in dieser Arbeit vorgestellten Ergebnisse die Machbarkeit adaptiver Mikrostimulation. Darüber hinaus eröffnet sich die Perspektive zukünftig stabile Wahrnehmungen mit Hilfe kortikaler Implantate zu erzeugen.

Contents

List of Figures

List of Tables

It is the tension between creativity and skepticism
that has produced the stunning
and unexpected findings of science.

Carl Sagan (1934-1996)

1 Introduction

During their lifetime, humans constantly interact with their environment and other human beings. All these interactions are composed of basic motor actions and sensory impressions. Besides locomotion, the motor actions allow humans to manipulate their environment and to communicate with other individuals. The perception of the surrounding environment that is assembled from the activity of many sensory cells is crucial to judge the effect of motor actions and to control them. Any disruption of sensory input or motor output consequently leads to a severe loss in human quality of life.

The deprivation of sensory and motor interactions can be the result of chronic diseases or traumatic injuries. Patients with stroke, amyotrophic lateral sclerosis, multiple sclerosis, or spinal chord injuries for example suffer a loss of voluntary control over their motor actions. Spinal chord injury is additionally accompanied by a loss of somatosensory input from the paralyzed body parts. Yet, the most widely known sensory deficits are related to the visual and auditory system in humans leading to blindness and deafness.

The diseases and injuries of the nervous system just adverted to are only a small number of examples, but all of these deficits have in common that currently they are at best treatable but not curable. Although a complete cure is certainly the final goal of ongoing medical research, an important option to substantially improve the quality of life in the meantime lies in the restoration of motor and sensory interactions with the help of biomedical devices that can partially replace or bypass the injured parts of the nervous system.

One can distinguish two types of biomedical devices depending on the direction of information flow. Devices that read out brain activity to restore locomotion or communication, and thus provide an alternative efferent pathway from brain to muscles, are commonly called brain machine interfaces [98, 126], or brain computer interfaces [9, 8]. Of concern in this thesis are the second type of devices called sensory neural prostheses that enable information flow in the opposite direction along afferent pathways. Sensory neural prostheses can partially restore lost sensory perceptions like vision [153, 33] and hearing [93] by electrically stimulating particular parts of the central nervous system as illustrated in figure 1.1.

Retinal implants as a way to restore vision are currently tested in clinical trials [154, 67, 49]

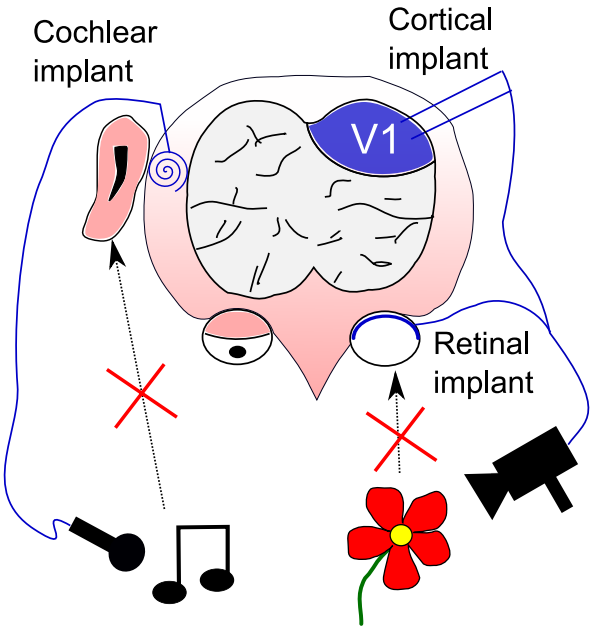


Figure 1.1: *Examples of sensory neural prostheses. For the cochlear implant sounds are recorded over a microphone and transformed to electrical stimulation pulses that are delivered over an electrode placed inside the cochlea. In a similar manner vision can be restored by capturing the scene with a camera and stimulating either the retina in the eye (retinal implant) or primary visual cortex (V1) in the brain (cortical implant).*

while cochlear implants are already routinely used to treat deafness [65, 112]. Unfortunately retinal and cochlear implants require intact optical and auditory nerves respectively – a precondition that is not fulfilled by all patients. If visual deficits are caused by lesions at later stages of the pathway cortical implants still offer the possibility to restore rudimentary perceptions by direct stimulation in primary visual cortex (figure 1.1). Cortical implants have a long history of investigations in humans [14, 43], and both surface electrodes [42] and implanted micro electrodes [122] have been found suitable to elicit light perceptions, called phosphenes, by electrical stimulation.

In contrast to stimulation in the first stages of a sensory system, as done by cochlear and retinal implants, stimulation in cortical areas is complicated by the intricate connectivity of the brain tissue and so far it has proven difficult to create stable visual perceptions over longer time periods. Before cortical implants can be routinely applied there are three fundamental problems that remain to be solved:

1. Temporal dependency of stimulation pulses leading to rapid accommodation. This manifests itself by a gradual decrease of perceived phosphene brightness during continuous stimulation with constant pulse intensity [122].

2. Spatial dependency of stimulation pulses leading to context dependent perception of phosphene properties like color an depth [122].

3. Interference of the background brain activity with evoked cortical potentials [1].

The work described in this dissertation proposes to continuously adjust the stimulation intensity in dependence of the ongoing brain activity in order to stabilize evoked cortical

potentials and reduce the influence of the cortical state. This putative solution to the third fundamental problem mentioned above will be subsequently called "adaptive microstimulation". The idea of adaptive microstimulation is supported by studies showing that the large variability of cortical potentials evoked by stimulation with fixed parameters can be attributed to the ongoing activity of the brain [1, 77, 26]. Further it is known that fluctuating brain activity actually influences and modulates the processing of incoming sensory signals [61, 118, 104] and that changes in local network activity alters the responsiveness of cortical neurons [59]. Even though these findings underpin the principle of adaptive microstimulation it so far remains unclear whether the ongoing brain activity carries sufficient information for establishing closed loop control of stimulation intensities.

The major contribution of this work is an empirical proof that adaptive microstimulation can indeed stabilize evoked cortical potentials in the barrel cortex of anesthetized rats [18]. On the way to a suitable experimental setup several algorithmic problems materialized. The first problem was the lack of prior work on how to solve the stimulus control problem with machine learning algorithms. In addition it was unclear how to decode information about the cortical state from recorded local field potentials and what parts of the pre- and post-stimulus potentials were relevant for the control task. Due to the restricted time of 1-3 minutes between recording and feedback sessions another problem to be addressed was the robust and quick selection of support vector regression (SVR) hyper-parameters. Later, detailed analysis of the stabilization effect revealed a temporal dependence that is probably caused by rapid out-dating of the offline trained SVR model. It was therefore necessary to develop a suitable online SVR training algorithm that can be easily incorporated into the existing experimental setup and does not require manual tuning of the learning rate.

For the stimulus control problem this thesis describes a direct and inverse approach where the associated functions are estimated by SVR [17]. To exploit prior knowledge about the structure of stimulation trials temporal information is incorporated by an application specific ANOVA kernel function and optimal pre- and post-stimulus time windows are identified. It is further proposed to solve the model selection problem by either minimizing the minimum span bound or the cross-validation error by the Quasi-Newton algorithm depending on the set of hyper-parameters to be selected. Finally, this work introduces a novel online training algorithm for SVR, termed PRIONA, based on the idea of solving the primal optimization problem [19]. The PRIONA algorithm is especially suited for real-time environments since it is easy to trade-off between iteration time and convergence speed. As an additional advantage over other state of the art online algorithms PRIONA does not require tuning of a learning rate which facilitates its practical application (figure 1.2).

Besides these algorithmic aspects the time critical adjustment of stimulus intensities led to several technical problems during the development of the experimental setup. Since optimal stimulus intensities change on a millisecond time scale it was necessary to implement the control algorithm in a real-time environment. This in turn required programming of driver software for connecting with the recording and stimulation equipment and development of a short latency serial interface to the stimulator hardware.

This dissertation has the following structure: The basic principles underlying perception, the visual system, current implant technology to restore lost visual perceptions, and rat barrel cortex, which is used as a model sensory system to study adaptive microstimulation are described in chapter 2. After this introduction to the relevant fundamentals of neurobiology chapter 3 explains the SVR algorithm, its formulation as primal and dual optimization problems, and the incorporation of a bias term into the primal SVR formulation. Further, this chapter contains an empirical comparison of l_1/l_2-loss functions and the primal/dual solution approaches on various benchmark data sets. Chapter 4 describes state of the art online training algorithms for SVR and proposes a novel online training algorithm based on the primal SVR formulation. The state of the art algorithms are compared to the new algorithm with respect to convergence speed and prediction precision on different data sets. Chapter 5 deals with the problem of model selection and explores how the SVR hyper-parameters can be selected efficiently by either minimizing a bound on the leave one out error or by directly minimizing the cross validation error. The application of SVR to the problem of adaptive microstimulation is discussed in chapter 6, which comprises a comparison of the direct and inverse modelling approach in conjunction with different feature extraction methods, and a comparison of standard kernel functions with an application-specific kernel. Chapter 7 introduces the experimental setup used for online feedback and presents results that show the feasibility of adaptive microstimulation. Chapter 8 summarizes the contributions of this thesis and gives an outlook on future experimental investigations and algorithmic developments. Finally appendix A contains a formal description of the algorithms and appendix B gives detailed descriptions of all data sets.

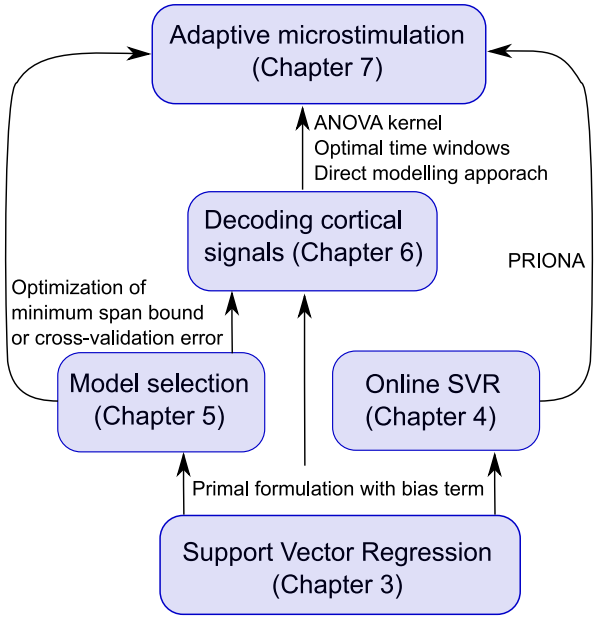


Figure 1.2: *Dependencies of algorithmic solutions.*

In the kingdom of the blind,
the one-eyed man is king.

Desiderius Erasmus
(1466-1536)

2 Perception

Imagine you are on a walk through a forest in autumn, the sight of the colorful leaves in shades of claret-red to golden yellow, the smell of the musty earth and the rustling sound of fallen leaves and branches on the ground disturbed by your stride. These impressions are all examples of sensory perceptions that are made accessible to us by receptor cells which are sensitive to a particular kind of stimulus. From our personal experience perceptions seem like a perfect copy of the surrounding world. But our brain does not work like a movie camera that passively records the environment. Rather it constructs representations of external events based on its functional anatomy and the dynamic activity of populations of nerve cells. These representations are the product of information processing occurring at different stages along the path from receptor cells to the brain. Although the sensory systems responsible for vision, hearing, touch, smell, and taste convey information about different physical stimuli there are common principles for information processing. Understanding the basic principles and the functional properties of neurons in the sensory systems is crucial when it comes to restoring lost sensory functions, which are caused by interruption of information flow at one of the processing stages.

2.1 Principles of sensory processing

Modality, location, intensity and timing are the four basic types of information that all sensory systems transmit upon stimulation, a fact that was revealed by the early work on psychophysics by Weber and Fechner. Bound together these four stimulus attributes yield sensation.

2.1.1 Modality

The five classical major modalities comprise vision, hearing, touch, taste, and smell. More recently the somatic senses of pain, temperature, itch and proprioception and the vestibular sense of balance were added to the classic modalities. Each modality is determined by the type of stimulus energy and the sensory receptors. For example, the only detectors for

the stimulus energy of electromagnetic waves in humans[1)] are the photoreceptors in the retina that give rise to the modality of vision, while there are several receptors for chemical stimulus energy resulting in the modalities of taste, smell, and itch. These receptors form the first stage in each sensory pathway and transduce the stimulus energy into an electrical signal called the receptor potential. Since most sensory receptors are selective only for a single type of stimulus, a property called receptor specificity, modality is represented by a labeled line code inside the nervous system.

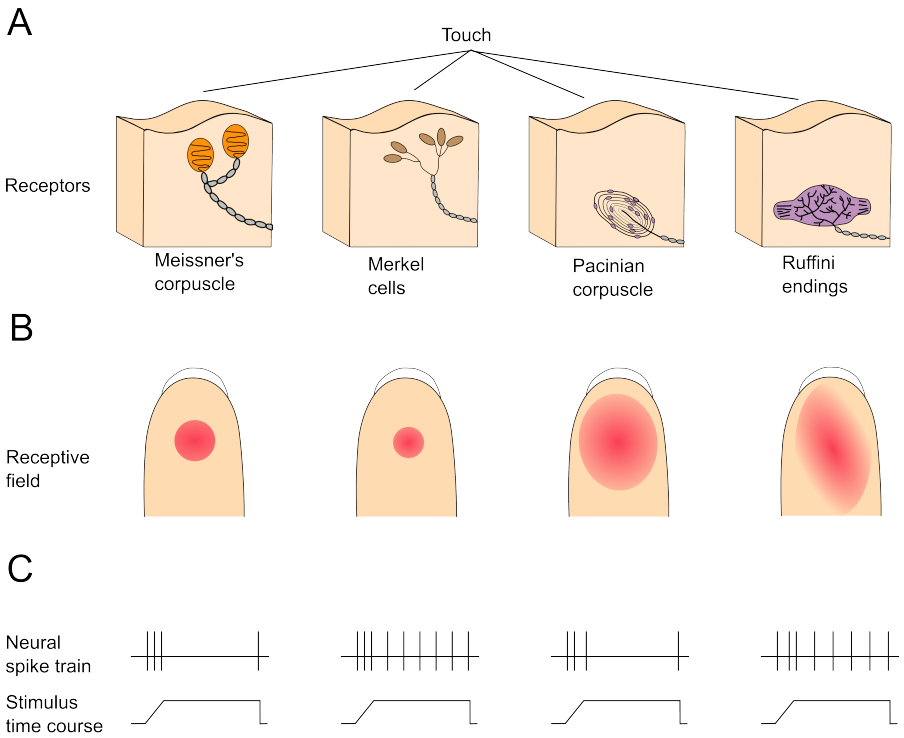


Figure 2.1: *Modality, location, intensity, and timing are four stimulus attributes encoded by a sensory system. These attributes are illustrated for the modality of touch.* ***A:*** *The submodalities of touch are sensed by different mechanoreceptors in the human hand. Activation of Merkel cells and Ruffini endings creates sensations of steady pressure while Meissner's and Pacinian corpuscles convey the sensation of vibration.* ***B:*** *Location and spatial properties are encoded by the spatial distribution of activated receptors. Single receptors fire only when the area of skin, as illustrated by the red shading, is touched. This area differs in size and is called the receptive field.* ***C:*** *Spike trains evoked by touch in the center of the receptive field. The intensity of the stimulus is signaled by the firing rate while the duration is encoded by the time course of the spike train. Adapted from [73].*

[1)] Sharks additionally have receptors for weak electric fields of low frequency (0.1-40Hz) called Lorenzian ampullae [46].

This means that the axon of each receptor can be seen as a communication channel specific to a certain modality and thus excitation of a sensory neuron, either via a natural stimulus or by electric stimulation, elicits the same sensation. When it comes to restoring auditory perception in patients with damaged receptors in the inner ear, the labeled line code can be exploited, as electrical stimulation of the auditory nerve can be used to signal tones of different frequencies. Since receptors are usually tuned to a narrow range of stimulus intensities, each major modality has several constituent submodalities that signal qualitative aspects of the stimulus. Examples of submodalities for touch are temperature, texture, and, rigidity (see figure 2.1).

2.1.2 Location

Location about a stimulus is conveyed to the nervous system by the spatial layout of receptors within a sensory organ. For somatic sensation and vision it is important to locate the stimulation site on the body or in space, to discriminate the size and shape of objects, and to resolve the fine detail of the stimulus or environment.

The first task is achieved by the receptive field of a sensory neuron, which is the set of all locations in the environment where a stimulus can activate the neuron. In touch this is the area of skin where a tactile stimulus can be conducted to the nerve terminals. Receptive fields also help in distinguishing the size and shape of objects since stimuli larger than the receptive field will activate adjacent sensory neurons. Thus the total number of activated adjacent receptors encodes information about stimulus size and shape. If the density of receptor cells in a given part of the body is high the receptive field of each receptor is small and therefore the population of sensory neurons provide fine spatial resolution. On the fingertips and the central part of the retina spatial discrimination is acute, while it is coarse on the trunk and the outer margins of the retina which is caused by the nonuniform receptor densities in the visual and somatic system. These differences in receptor density and the topographic arrangement of afferent inputs are reflected in the maps of the body present in the central nervous system. Body parts innervated by a high number of sensory neurons are represented by larger cortical areas while sparsely innervated regions occupy smaller areas. Contrary to this topographic arrangement of receptors for vision and somatic sensation, the spatial arrangement of receptors for hearing, taste and smell follows the energy spectrum for these modalities. Thus, the ordering of receptors in the auditory system according to sound frequency leads to a tonotopic map in cortical areas.

2.1.3 Intensity

The relationship between physical intensity of a stimulus and the subjective sense of intensity is described by the laws of psychophysics. From experience we know that it is easy to distinguish 1 kg from 2 kg whereas it is difficult to discern 50 kg from 51 kg. This phenomenon is present in all sensory systems and can be expressed by the following equation

known as Weber's law postulated in 1834:

$$\Delta S = K \cdot S,$$

where ΔS is the minimal difference between a reference Stimulus S and another stimulus that can be discriminated, and K is a constant. By assuming that ΔS corresponds to equal increments in the subjective sense of intensity Fechner extended Weber's law in 1860 to describe the relationship between the stimulus strength S and the intensity of sensation I experienced by a subject:

$$I = K \log S/S_0,$$

where S_0 is the threshold amplitude of the stimulus, and K is a constant. Later Stevens discovered in 1957 that, when subjects are asked to describe subjective sense either by reporting numbers or by selecting an equivalent intensity in another modality, the relationship is better described by power functions:

$$I = K(S - S_0)^n,$$

with modality specific exponent n. Although this seems contradictory, Mac Kay showed that Stevens' power law should be experimentally observable if there is a logarithmic relation between physical stimulus and subjective sense of magnitude, and if the subjective sense of numbers is also logarithmic. More profoundly this implies that there are a great number of laws that lead to emergence of the power law [71]. Clearly this ambiguity cannot be resolved without examination of the internal mechanism, e.g. the neural representation of intensity. This was investigated by Mountcastle [94], who found that the sense for subjective intensity linearly depends on the firing rate of sensory neurons. Stimuli with higher intensity lead to receptor potentials with large amplitude which allows the sensory neuron to reach the firing threshold earlier in the relative refractory period. Besides this rate code stimulus intensity is represented by a population code since strong stimuli activate a greater number of receptor cells. It is important to know these psychophysical laws when sensory perceptions are restored by direct electrical stimulation of neurons. For example, the findings described above imply that a linear range of stimulus intensities is appropriate to cover the full range of subjective sense of intensity.

2.1.4 Timing

Timing properties of a stimulus are encoded by changes in the firing rate of sensory neurons. When the skin is indented by a probe, the firing rate of mechanoreceptors is proportional to the indentation speed and the total amount of applied pressure [94]. But after some time of steady pressure the firing rate decreases to a level that is proportional to skin indentation and stops when the probe is retracted. If a stimulus is presented for several minutes without change in position or amplitude the firing rate ceases, an effect called adaptation. Receptor cells can be distinguished by the speed of adaptation. Slowly adapting receptors have slow inactivating Na^+ or Ca^{2+} channels or calcium-dependent K^+ channels and signal stimulus

magnitude for several minutes by continuous firing of action potentials. Rapidly adapting receptors signal the velocity changes in stimulus intensity. Some of these receptors have a fast inactivation mechanism that prohibits spike generation while for others the anatomical structure of the receptor filters out steady state components, as is the case for the Pacinian corpuscle.

2.1.5 Sensory systems

The view of a painting by Salvador Dalí and the sounds of speech are conveyed to the brain by two sensory systems for the modalities of vision and hearing. Despite the differences in transduction of stimulus energy into an electric signal by the receptors all sensory systems share common functional and organizational principles. Activity in the receptors is transferred to neurons in the relay nuclei of the brain stem and from there to the thalamus and finally to the sensory areas of cortex. This route of information flow is commonly referred to as a sensory pathway.

Although the organizational principles are similar, sensory systems differ in their complexity between different modalities and stages along the sensory pathway. The most intricate sensory system in humans, for example, is the visual system where the complexity of information processing increases along the pathway from retina to primary visual cortex.

To understand the capabilities and limitations of current implant technology for restoring visual perception, described in section 2.3, it is important to know putative stimulation targets on the sensory pathway, as well as the properties and organization of the underlying neural network. Section 2.2 therefore gives a brief introduction to the visual system.

2.2 The visual system

The main parts of the visual system are the retina inside the eyeball, the lateral geniculate nucleus (LGN) in the thalamus, and the primary visual cortex (V1). Each of these parts is involved in visual information processing at different levels of abstraction which starts when light is transduced into an electrical signal by the receptor cells in the retina. Visible light is electromagnetic radiation with a wavelength of 400-700nm and a speed of approximately 300000km/s in vacuum. Physical properties of electromagnetic radiation are intensity, frequency or wavelength, and polarization. The human visual system can only perceive the first two properties, also called luminance and color, while honeybees can additionally perceive the polarization of light and use it for orientation [148].

Light is converted into electric current in a process called phototransduction by rod and cone photoreceptor cells in the retina. In contrast to the auditory system, where hair cells form direct connections with the ganglion cells, the rod an cone cells are connected to the ganglion cells through lateral and vertical pathways consisting of horizontal, amacrine, and bipolar cells. This retinal network gives rise to a concentric receptive field structure

in ganglion cells. When light falls on the center of the receptive field, on-center ganglion cells are excited and off-center ganglion cells inhibited, while light falling on the surround part leads to the reverse response behaviour. The concentric shape of the receptive fields helps to detect changes in stimulus intensity (section 2.1.3). Further one can distinguish magnocellular ganglion cells, or M cells, with large receptive fields specialized on analyzing motion and parvocellular ganglion cells, or P cells, with small receptive fields responsible for perception of color and form [73].

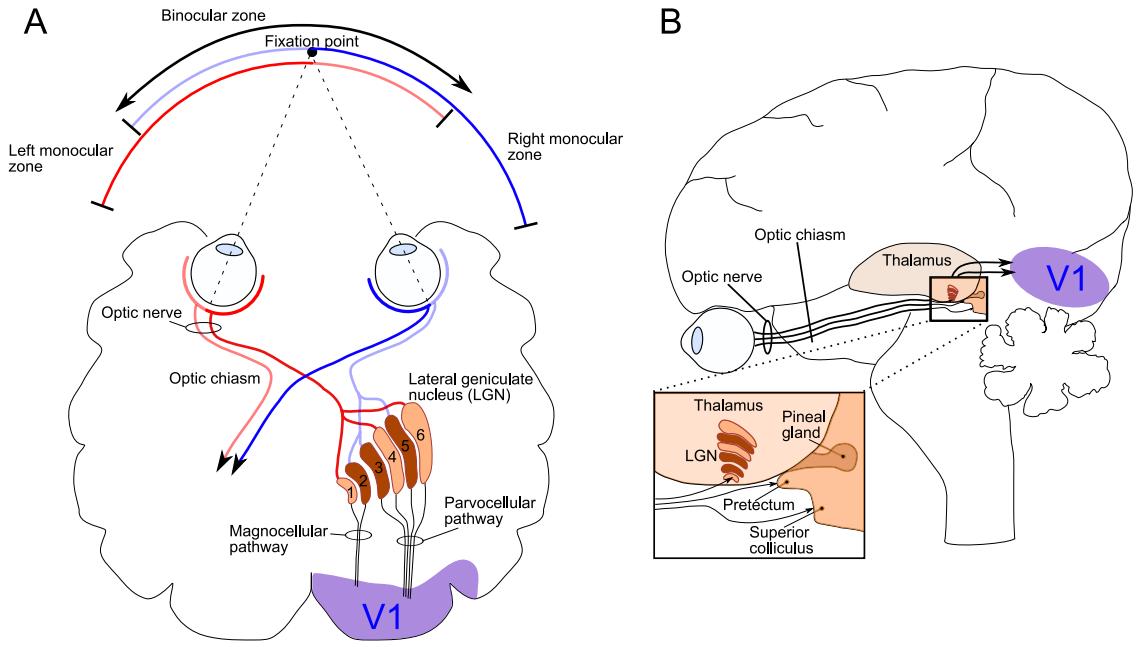


Figure 2.2: *The visual pathway.* ***A:*** *Horizontal view of the visual pathway. The left visual field is projected on the nasal part in the retina of the left eye and the temporal part of the retina in the right eye. Since information about the left visual field is processed in the right cortical hemisphere the optic nerve fibers split at the optic chiasm and cross to the other side. The LGN in the thalamus comprises six layers that receive exclusive input from the ispi- or contra-lateral eye. Magnocellular ganglion cells project to the first two layers and give rise to the magnocellular pathway that terminates in V1. The parvocellular pathway is the second input channel to V1 and starts in the remaining four layers of LGN.* ***B:*** *Sagittal view of the visual pathway. Besides the LGN important other projection targets of the optic nerve are the pretectum and superior colliculus.*

2.2.1 Visual pathways

The optic nerve is formed by the axons of retinal ganglion cells and it contains more than 1 million fibers for each eye. The first station in the visual pathway is the optic

chiasm, where the optic nerve fibers split and cross to the other side of the brain. This crossing is necessary as the left visual field is perceived by the nasal part of the left, or ipsi-lateral, eye and the temporal part of the right, or contra-lateral, eye (figure 2.2A). The next projection targets on the visual pathway are the superior colliculus, concerned with controlling saccadic eye movements and the integration with other sensory inputs, the pretectum, containing the reflex circuit for pupillary constrictions, and the LGN, which is the principal relay station for visual information on its way to V1. After disruption of the LGN pathway visual perception is lost, although movement towards objects in the visual field is still possible. It is speculated that this residual vision, called blind sight, is due to an indirect pathway through the superior colliculus [73].

Throughout the visual pathway to the primary visual cortex the axons from the magno- and parvocellular ganglion cells remain segregated (figure 2.2B) which implies that there are two parallel information channels to the visual cortex called the M and P pathways. Neurons in the LGN have the same concentric structure as the ganglion cells in the retina indicating that the LGN is mainly a relay station for visual information. The primary visual cortex is the first point in the visual pathway where receptive fields are significantly different from those in the retina.

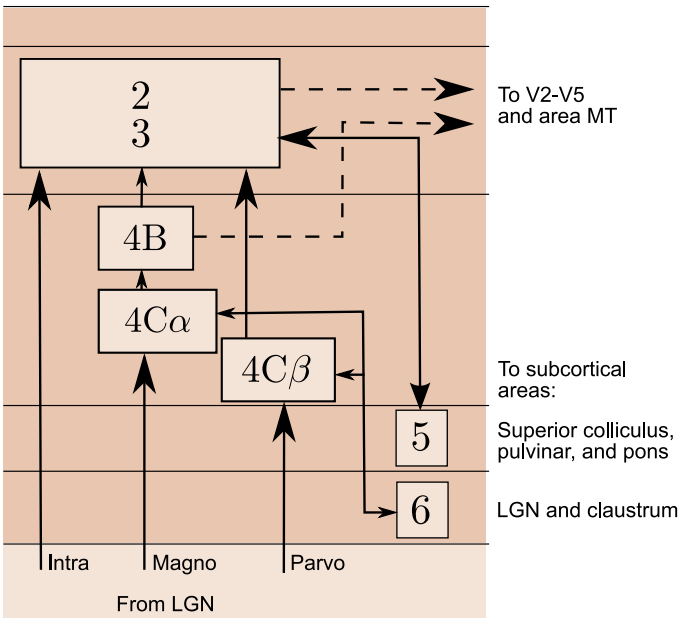


Figure 2.3: *Anatomical connections and information flow in visual cortex. The M and P pathways from LGN are the main inputs that terminate in layers 4Cα and 4Cβ of V1. Cells that lie between the layers in LGN provide input to layer 2/3 in visual cortex. Important intracortical connections are formed by axon collaterals of pyramidal cells in layer 2/3 and layer 5. Further there is a loop back connection from layer 6 to layer 4C. Every layer, except 4C, has output connections to neighboring V1 areas. While cells in layer 2/3 and 4B project to other cortical areas, cells from layer 5/6 project back to sub-cortical areas.*

2.2.2 Cortical processing

The primary visual cortex or visual area 1 (V1), is located in the occipital part of the cortex (figure 2.2B). In humans it is about 2mm thick and contains six layers of cells between the cortical surface and the underlying white matter. In comparison to other

cortical areas it has a prominent layer 4 that is further subdivided into sub-layers 4A, 4B, 4Cα, and 4Cβ. Layer 4 is the principal layer that receives input from the LGN. The axons from the M pathway terminate in layer 4Cα and those from the P pathway in layer 4Cβ (figure 2.3). Axons from cells between the LGN layers form the intra-laminar input to V1 and terminate in layer 2 and 3 in patches of cells called blobs that are mainly concerned with color processing. After processing of visual information in V1 the output is directed to sub-cortical areas like the superior colliculus or LGN via layers 5 and 6, while output to other cortical areas occurs via layer 2 and 3 (figure 2.3).

The cortical processing reflects itself in the receptive field structure of neurons. By studies in layer 4 of visual cortex in cats Hubel and Wiesel [66] identified two different types of neurons: simple cells and complex cells. Simple cells are only excited by light bars with a certain orientation and their receptive fields consequently have rectangular inhibitory and excitatory zones (figure 2.4A).

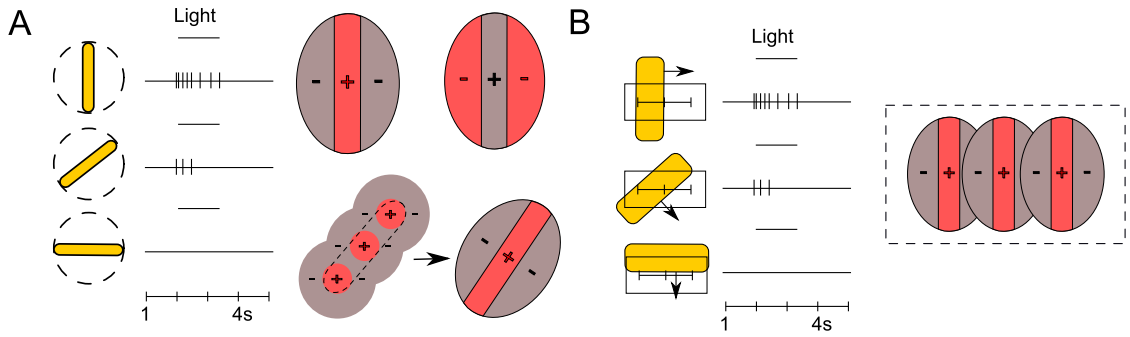


Figure 2.4: *Receptive fields of simple and complex cells.* ***A:*** *The receptive field structure of simple cells in primary visual cortex is revealed by presentation of light bars with different orientation and observation of a cells response. Simple cells are best activated by light bars with a specific orientation. The elongated excitatory and flanking inhibitory zones of simple cell receptive fields may result from the superposition of several center surround receptive fields of ganglion cells.* ***B:*** *Complex cells respond with high firing rates to light bars that move in a certain direction. The receptive field of complex cells could result from the superposition of simple cell receptive fields.*

On a larger scale V1 is organized into columns that extend from the cortical surface to the white matter and contain cells with similar receptive fields. Orientation columns are 30 to 100μm wide and 2 mm deep and contain cells in layer 4C with concentric receptive fields. Below and above this layer there are simple cells with identical axes of orientation that are responsive to stimuli at a particular position in the visual field. A complete set of orientation columns that represent all orientation angles between 0 and 180° for the same visual position are arranged in a circular structure like a pinwheel, termed hyper-column. The arrangement of hyper-columns is occasionally interrupted by patches of cells called blobs that are not orientation sensitive but respond to different color stimuli. In addition

to orientation columns and blobs, V1 is organized in ocular dominance columns, where cells receive exclusive input from the ispi- or contra-lateral eye.

Cortical columns with similar function for different spatial positions are linked through horizontal connections. To represent the location attribute of a stimulus (section 2.1.2), the visual system uses a place code since neighboring hyper-columns in V1 are activated by stimuli in neighboring positions of the visual field. This place code gives rise to a retinotopic map that can be exploited to restore visual perception by cortical implants, as described in section 2.3.2. In the retinotopic map the lower part of the visual field is mapped to the cortical area above the calcarine fissure and vice versa. The region of the fovea is represented by a large portion of the cortical surface and is mapped to the lateral part of V1 (figure 2.5).

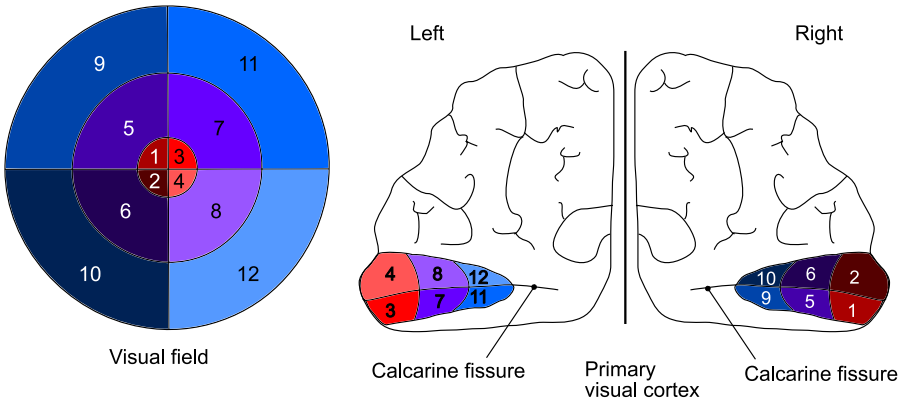


Figure 2.5: *Representation of the visual field on the surface of primary visual cortex. Due to the crossing of fibers in the optic chiasm, the left visual field is mapped to the right cortical hemisphere and vice versa. The calcarine fissure separates the representation for the upper and lower half of the visual field. The region around the fovea in the visual field is highly magnified and occupies about half of the cortical surface area in V1.*

2.3 Current implant technology

Worldwide there are 37 million blind people, with more than 90% of the world's visual impaired living in developing countries. On a global scale cataract, an opacity of the lens, is the main cause for blindness followed by glaucoma, which involves loss of retinal ganglion cells, and age-related macular degeneration, where abnormal blood vessel growth under the central retina leads to degeneration of cells. While age-related macular degeneration ranks third globally, it is the main cause for visual impairment in the most developed countries, due to the growing number of people over 70 years of age [150]. Current clinical trials that attempt to restore visual perception with retinal implants, described in section 2.3.1, focus on patients with age-related macular degeneration or patients with retinitis pigmentosa, a

hereditary disease that leads to a loss of photoreceptors. Retinal implants, the analogue to cochlear implants in the auditory systems, bridge the first stage of visual perception by direct stimulation of bipolar of ganglion cells in the retina. This requires an intact retinal network, or at least undamaged ganglion cells. For patients with strokes of the optic chiasm and optic nerve atrophy the normal flow of visual information is disrupted at later stages of the visual pathways which renders retinal implants useless. This is also the case for patients with end-stage retinitis pigmentosa, as it involves abnormal connections formed by amacrine or horizontal cells [33]. Restoring visual perception in these patients is possible by electrical stimulation of primary visual cortex, an approach that is pursued by cortical implants described in section 2.3.2.

2.3.1 Retinal implants

Electric stimulation of the human eye was already known in the beginning of the 18th century to elicit artificial sensation of light called phosphenes. The first retinal prosthesis proposed in a patent by Tassiker [135] consisted of a light-sensitive selenium photodiode cell placed behind the retina. Nowadays retinal implants under research can be distinguished by the placement of the stimulation electrodes. In epiretinal implants electrodes are located on top of the ganglion cells, while subretinal implants place electrodes above the pigment epithelium (figure 2.6).

For epiretinal implants the visual scene is captured by a small camera mounted on glasses or by a field sensor situated in an intra-ocular plastic lens. A video processor converts the video stream into a train of stimulation pulses that are delivered over an electrode array attached to the inner retinal surface separating the ganglion cell layer from the vitreous body of the eye (figure 2.6). Communication between the video processor and the electrode array either occurs over shielded wires or trans-cutaneous radio frequency telemetry [87]. In clinical trials [67, 49] with epiretinal implants phosphene perception could be evoked by biphasic current pulses of 24-702μA amplitude and 1ms duration. Non-flickering perceptions were achieved by stimulation frequencies between 40 and 50Hz and some patients were able to detect movements. Compared to subretinal implants the epiretinal approach is less invasive, does not occlude retinal vasculature, can be monitored ophthalmoscopically, and does not require intact optics, like a clear lens [33]. On the downside epiretinal implants complicate the encoding of visual information and cannot guarantee a consistent relationship between the phosphene map and electrode positions, by stimulating the ganglion cell layer instead of the remaining retinal network [33].

In the subretinal implant light entering the eye is transformed into stimulation currents by microphotodiode array located on top of the pigment epithelium (figure 2.6). The generated electric current stimulates the overlying bipolar cells and leads to a more natural excitation of ganglion cells through the remaining retinal network [153, 52]. A subretinal implant comprising 1500 microphotodiodes, amplifiers, and a 4x4 array of stimulation electrodes was chronically implanted in two retinitis pigmentosa patients in a recent clinical trial [154].

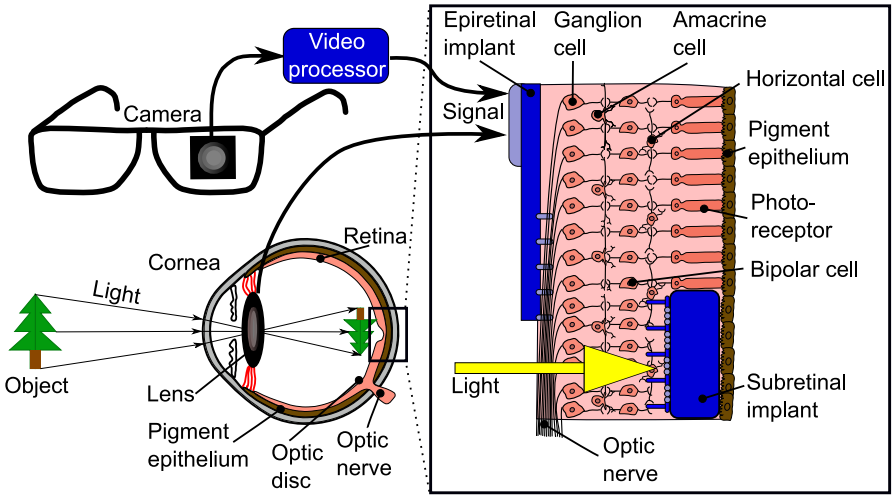


Figure 2.6: *Two types of retinal implants. For epiretinal implants light is either captured by a camera mounted on glasses or a field sensor that replaces the natural lens. The encoded visual information is used to directly stimulate ganglion cell axons by electrodes placed on top of the retina. The subretinal implant – a silicon plate carrying thousands of microphotodiodes and stimulation electrodes – is located in front of the pigment epithelium and replaces lost photoreceptors. Light reaching the photodiodes is converted into electric currents to stimulate cells of the retinal network.*

Depending on the spatiotemporal activation pattern of the electrodes patients were able to perceive single phosphenes, lines or squares and could distinguish between lines having vertical or horizontal orientation. Furthermore patients were able to correctly describe the direction of dot movements [154].

2.3.2 Cortical implants

Cortical implants attempt to restore visual perception by direct stimulation of primary visual cortex. Because of the retinotopic map (figure 2.5) electric stimulation of neighboring cortical areas elicits perception of phosphenes at neighboring locations in the visual field. Direct stimulation of visual cortex, as opposed to stimulation of the retina, has the advantage that the cortical area occupied by the central visual field is highly magnified. Two degrees of the central visual field occupy about 1mm^2 on the retina but approximately 2000mm^2 on the cortex [33]. Unfortunately a part of the central visual field lies hidden in the calcarine fissure (figure 2.5), and is not accessible to surface stimulation. There are two different types of cortical implants that either use surface electrodes or intracortical electrodes for stimulation.

Surface electrodes

The use of surface electrodes for cortical stimulation has been pioneered by Brindley [14] and later Dobelle [43]. Initially Brindley used an array of 80 square silicon-insulated platinum electrodes with an area of 0.64mm^2. In later studies the number of electrodes was increased to 151. These arrays were implanted above the pial surface of visual cortex and were connected through wires to the extra-cranial part of the implant, which comprised an array of radio receivers. Electromagnetic induction was used to activate single receivers and stimulate a single electrode. Upon stimulation of single electrodes implanted patients commonly perceived single phosphenes with constant position in the visual field, although stimulation via some electrodes led to perception of several phosphenes or diffuse clouds of light points. Furthermore patients could distinguish the position of two phosphenes when the corresponding stimulation electrodes were between 2 and 4mm apart.

Later the Dobelle group [42] combined this basic cortical implant with a camera mounted on a pair of glasses. The data recorded by the camera is processed by a belt-mounted laptop computer that uses the Sobel edge detection algorithm for analyzing the visual scene (figure 2.7). In a first study four blind patients received this implant which was limited to 64 stimulation electrodes and one side of the visual cortex. After ten days of training one patient could recognize letters of 15cm^2 size at a distance of 1.5 meters [42]. In a more recent clinical trial in Portugal in 2002 a group of 16 blind patients received bilateral cortical implants with 72 stimulation electrodes. Although these trials received great media attention [80] including a video showing an implanted patient driving a car around an empty parking lot, there have been no publications about the quantitative visual performance of patients or long-term stability of the visual prosthesis yet.

Besides risks associated with brain surgery and bio-compatibility, surface electrodes are the least invasive cortical implant, but there are also a number of disadvantages. To create perception of phosphenes, surface electrodes require high stimulation currents of 0.5-5mA, which enhances the risk of seizures and necessitates electrodes with large surface area to avoid electrochemical degradation. Large electrode areas in turn lead to a low spatial resolution around 3mm and increased current spread [14]. Since the surface electrodes are placed on top of the pia mater, stimulation can in some cases lead to headaches, presumably by activation of pain fibers in the meninges [33]. Another problem that cortical implants have in common with epiretinal implants is the motion of phosphenes during voluntary eye movements. Interestingly phosphenes retain their spatial position during eye movements caused by the vestibular reflex [14].

Microelectrodes

In comparison with surface electrodes, stimulation of visual cortex via intracortical microelectrodes has the advantage that lower stimulus currents in the range of 10 to 20μA are sufficient to evoke perception of phosphenes. This reduces the risk of causing seizures, allows microelectrodes to be more densely packed on a single array than surface electrodes,

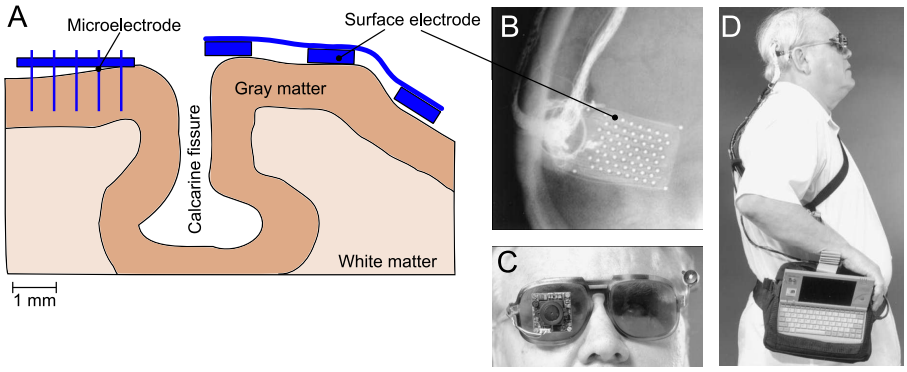


Figure 2.7: *Cortical implants for restoring vision.* ***A:*** *Intracortical implants use microelectrodes with a tip distance of 500μm that penetrate the cortical surface. Surface electrodes used by Dobelle [42] have a diameter of 1mm and are implanted above the pial surface.* ***B:*** *X-ray picture of 64 surface electrodes implanted on the surface of primary visual cortex.* ***C:*** *The visual scene is recorded by a camera mounted on a pair of glasses.* ***D:*** *A belt-mounted laptop computer is used to analyze the camera data and determine appropriate stimulation currents that are delivered via the implanted electrode array.*

and improves the spatial resolution of evoked phosphenes. Experiments with intracortical stimulation in one human subject showed that two different phosphenes could be discriminated when stimulation sites were 500μm apart, which is about five times better than the spatial resolution achieved with surface electrodes [122]. Prior psychophysical studies with healthy human subjects that used a simulated intracortical visual prosthesis with a 25 $\times$ 25 electrode array demonstrated that this resolution is sufficient to restore mobility and reading in blind subjects [24, 25]. The main drawback of intracortical microelectrodes is the higher risk to damage neural tissue, but recently special techniques for safe insertion of electrodes have been developed [99].

So far the feasibility of a visual prosthesis based on intracortical microelectrodes has been tested only in a single patient [122]. The cortical implant consisted of 38 iridium electrodes that were placed in the right visual cortex for a period of 4 months. Similar to the experiments with surface electrodes the patient perceived discrete phosphenes upon stimulation with a single electrode. The positions of phosphenes were consistent with the placement of the electrode array and the retinotopic map in primary visual cortex. Phosphene brightness could be adjusted by changing stimulus amplitude, frequency and the pulse duration, while phosphene size usually decreased with higher stimulation currents and increased with longer stimulation trains. In contrast to surface stimulation, where phosphenes were always described to have a yellowish or grayish color, the patient with the intracortical implant could perceive colored phosphenes when the stimulation amplitude was close to the threshold current. Furthermore phosphenes appeared at different distances from the subject [122]. From these findings one can conclude that stimulation of visual cortex with

microelectrodes can evoke richer visual percepts in comparison to stimulation with surface electrodes.

2.4 Unsolved problems of cortical implants

In contrast to sensory neural prostheses that target the first stage of a sensory system, like retinal implants (section 2.3.1), cortical implants have to cope with problems caused by the inherent complexity of the cortical neural network. The unsolved problems of cortical implants are temporal degradation of phosphenes, interaction of adjacent stimulation sites, and interference of ongoing brain activity with stimulus evoked potentials.

The temporal decrease in phosphene brightness during stimulation over several minutes, reported in [122], is due to an accommodation of cortical neurons to repeated stimuli with fixed parameters. Further, experiments in this study revealed three types of phosphene interaction. First, simultaneous stimulation of adjacent electrodes caused a reduction of threshold currents that are required to produce phosphenes in comparison to the non-simultaneous stimulation. Second, alternating stimulation of an adjacent electrode pair resulted in a loss of individual phosphene attributes like color and form that could be perceived during single electrode stimulation. Third, the apparent depth of a single phosphene changed when additional electrodes in the array were stimulated [122]. For more than two phosphenes similar interactions were found, e.g. the simultaneous activation of six electrodes required the adjustment of single microelectrode currents before the patient could see all phosphenes at the same time. In conclusion these interactions indicate that phosphene creation is highly dependent on the stimulation context in cortical implants.

Beside the interactions of phosphenes that are caused by electric stimulation itself another source that can interfere with the stimulation effect is the background activity of the brain [1, 118, 26]. As described in section 2.2.2 the neurons in visual cortex form intricate local connection patterns and receive input from distant sub-cortical structures. Even without explicit visual input these connections can lead to a dynamically changing excitability of neural elements like cell bodies and axons [116, 39], which will in turn result in different visual perceptions as long as the parameters of the stimulation pulses are held fixed.

On the way to produce a pixelized vision system, like the one envisaged by Cha [24], by implants using intracortical microelectrodes, the problems of accommodation, context dependency, and interference of the background brain activity have to be solved in order to evoke stable visual percepts of simple geometric forms like lines and squares and letters. In the same experiments that revealed the interactions between phosphenes it was observed that manual adjustment of the stimulation parameters could disentangle some of the interactions [122]. This indicates that a stable visual percept might be achieved by continuous adaptation of the stimulation parameters based either on contextual information from evoked potentials of neighboring stimulation electrodes or the information

provided by the background activity of the brain. Subsequently this approach will be termed adaptive microstimulation.

The primary objective of this thesis is to answer the following question: Can adaptive stimulation be used to stabilize sensory percepts? Up to now there has been no previous work investigating adaptive microstimulation and hence the experimental setup for answering above question is chosen to be as simple as possible. As a first simplification the work concentrates on the information that can be extracted from the background brain activity since gathering contextual information requires several stimulation electrodes that are harder to handle experimentally. For contextual data it is also difficult to cover the space of stimulus parameters due to multiple stimulation sites. Therefore the experiments described in chapter 7 are restricted to a single stimulation and recording electrode. Of course these experiments cannot be conducted in the visual system of humans. A suitable model system for testing adaptive stimulation would be the visual system of primates which is one of the most investigated sensory systems in mammals. But primates are elaborate to handle and thus not suited given the explorative nature of the experiment where many animals are expected to be needed. Fortunately all sensory systems share common organizational principles, as described in section 2.1, which gives the opportunity to investigate the adaptive stimulation approach in a sensory system different from the visual system. The sensory system used as a model in the experimental part of this thesis is the somatosensory cortex of rats, called barrel cortex, described in section 2.5. Further, the experiments exploring adaptive microstimulation will use anesthetized animals to avoid any difficulties associated with wake behaving rats, like poor signal quality, artifacts, and limited recording time.

2.5 Model sensory system: rat barrel cortex

The barrel cortex of rats is a special part of the somatosensory cortex where information acquired by the whiskers is processed. With their whiskers rats and other rodents can locate objects, build spatial representations of their environment and discriminate between textures that have small differences in granularity. Except for the latter task humans would rely on visual cues. This explains why the visual system in humans occupies a large portion of the cortical surface. The visual system is less important for rats since they are nocturnal animals that live in tunnels where sensory perceptions picked up by the whiskers are more informative about the environment. Consequently, the somatosensory system of rodents is highly developed and covers a large part of the rodent brain. Woolsey and Van der Loos were the first to discover the remarkable anatomical structure of the primary somatosensory cortex of rats where each whisker is represented by a discrete and well-defined cluster of cells in layer 4 that looked like a barrel [149]. Later barrels were also discovered in other rodents like mice, gerbils and hamsters as well as in rabbits, ferrets and wallabies [50]. In the years following the initial discovery by Woolsey the barrel cortex

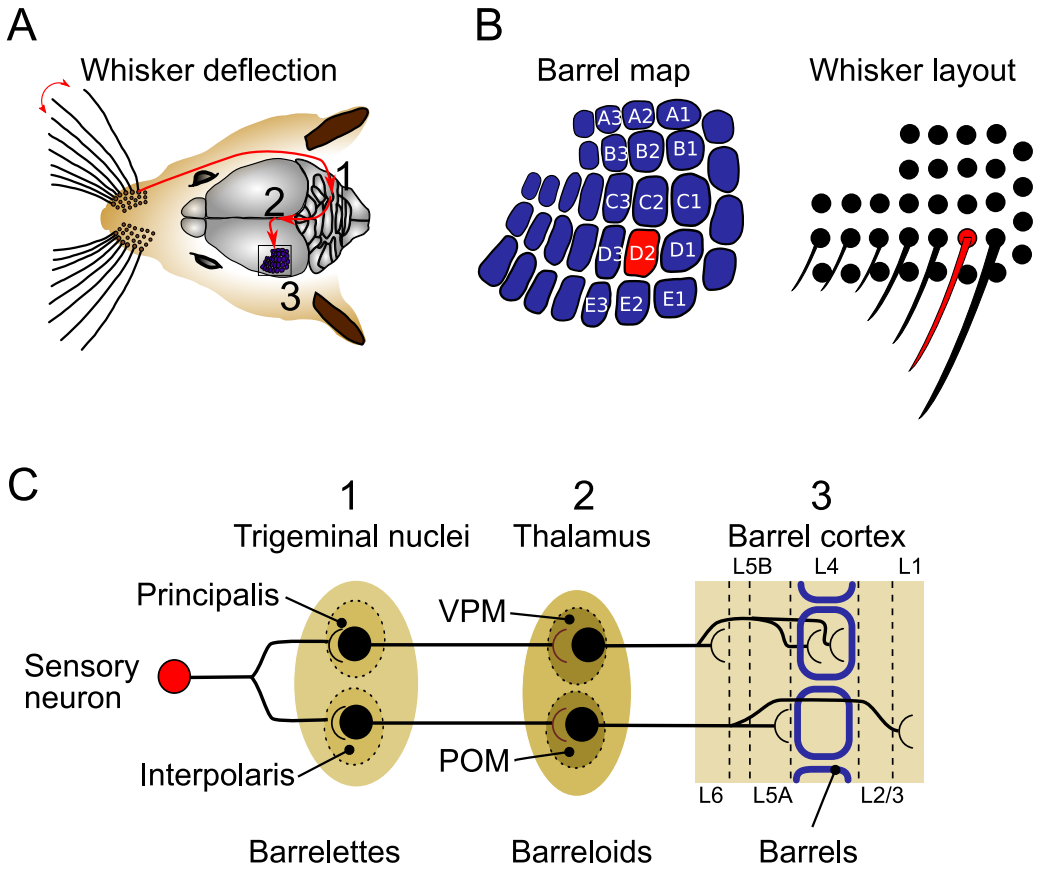


Figure 2.8: *The barrel cortex in rats represents sensory information acquired by the whiskers.* ***A:*** *Deflection of a whisker evokes action potentials in the sensory neuron that is conveyed to the trigeminal nuclei in the brain stem (1). Brain stem neurons project to the thalamus (2) and axons of thalamic neurons finally terminate in the primary somatosensory cortex, termed barrel cortex (3).* ***B:*** *The somatotopic map in barrel cortex shown on the left follows the layout of the whiskers on the snout of the rat. Whiskers and barrel are labeled according to rows (A-E) and arcs (1-3). Deflection of the D2 whisker leads to excitation of neurons in the D2 barrel.* ***C:*** *Pathway from whisker to barrel cortex. The trigeminal nuclei and the thalamus have somatotopic maps called barrelettes and barreloids. Major input to the barrels in cortical layer 4 comes from a pathway over nucleus principalis and the ventral posterior medial thalamus. A secondary pathway over nucleus interpolaris and the posterior medial thalamus provides input to cortical layer 5A and layer 1. For simplicity other trigeminal nuclei and projection targets are omitted.*

became a popular research subject in neuroscience. The reasons for this are the easy experimental access to the barrel cortex and the theoretical connection between barrels and the cortical column that is supposed to form the basic functional unit in cortex. Under the columnar hypothesis the cortex is composed of vertical structures called columns that are aligned orthogonal to the cortical surface and cross the six cortical layers. Each column has a diameter of about 300μm and contains a local neural circuit that is assumed to be identical in all columns. Neighboring columns only differ in the input they receive from the thalamus [50]. So far this hypothesis seems to be valid in the visual system, where neighboring hyper-columns process information from neighboring areas of the retina (section 2.2.2), and in the barrel cortex, where neighboring columns represent neighboring whiskers on the snout (figure 2.8).

The transfer of sensory information to barrel cortex begins with the deflection of a whisker. Although the precise molecular mechanism is still unknown the deflection is thought to open mechano-gated ion channels in the dendrites of sensory neurons that innervate the hair follicle [103]. The action potential elicited by the depolarization is propagated over the trigeminal nerve to the four trigeminal nuclei in the brain stem, called principalis, oralis, interpolaris, and caudalis. Each sensory neuron only innervates one whisker and forms excitatory glutamatergic synapses in the brain stem nuclei. Neurons in the nucleus principalis are somatotopically arranged into structures called barrelettes, since they resemble the layout in the barrel cortex. Similar to the visual system, where information is transferred separately in the M- and P-pathways, the sensory information that arrives in the trigeminal nucleus is split up into a main pathway over the nucleus principalis that projects to the ventral posterior medial (VPM) nucleus of the thalamus and a secondary pathway over the nucleus oralis that projects to the posterior medial nucleus (POM) of the thalamus. Besides additional projections to the pretectum and superior colliculus the brain stem nuclei provide input to the nucleus facialis that forms a feedback connection to the muscles in the whisker pad. This early feedback connection is important due to a lack of spindle organs in the whisker pad muscles [50].

At the next stage of the pathway in the VPM nucleus of the thalamus the neurons are again somatotopically arranged in anatomical units called barreloids. The axons of neurons in the barreloids finally terminate on neurons in layer 4 of the primary somatosensory cortex that make up the barrels. The spatial arrangement of the barrels is almost identical to the layout of the whiskers on the snout of the rat (figure 2.8). Layer 1 and 5A of the primary somatosensory cortex as well as the secondary somatosensory cortex and the motor cortex receive input from the neurons in the POM nucleus of the thalamus. In anesthetized animals this secondary pathway is unlikely to contribute to sensory processing since neurons in the POM nucleus are inhibited by GABAergic neurons from zona incerta[2)]. Yet, the POM receives strong excitatory input from cortex and the inhibition depends upon the brain state. So the POM pathway may play an important role during active exploration [103].

[2)] A narrow band of gray matter between the subthalamic nucleus and thalamic fasciculus.

At a first glance the pathway from whiskers to the barrel cortex seems to be straightforward but there are important differences in the processing of sensory information between the relay stations and the cortex. First, neurons in the trigeminal nuclei respond with great reliability to whisker deflection whereas neurons in the barrel cortex show huge response variability across trials with identical whisker stimuli. This variability is mainly driven by interactions with the background brain activity [118] which makes the barrel cortex a suitable model system to study adaptive stimulation as outlined in section 2.3.2. Second, the receptive fields of a single whisker are narrow in the trigeminal nucleus and broad in the barrel cortex [103].

Considering the functional organization of barrel cortex one could ask whether there are additional functional maps besides the somatotopic layout in analogy to the visual system with its ocular dominance and orientation selectivity maps (section 2.2.2). Although there are cell clusters in layer 4 of barrel cortex that preferentially respond to similar directions of whisker deflections, the direction tuning does not appear to be organized in an orderly map. But there are indications for a direction preference map within layer 2/3, since cells responding to a given direction of whisker deflection are located closer to the neighboring barrel in direction of the deflection. For example, if the D3 whisker is deflected caudally towards the D2 whisker, then more cells in the half of the D3 barrel closer to the D2 barrel will fire than in the half of the D3 barrel closer to the D4 barrel [103]. The idea of an orientation map in barrel cortex is attractive as it encodes an important stimulus feature that is already represented in the barreloids of the thalamus [137], but it has to be verified by additional studies.

Recent experiments that concentrated on whisker perception in wake behaving rats revealed that the processing of sensory information in barrel cortex depends on the state of the animal. When the whiskers are not moving and the animal is at rest but wakeful, there are slow changes in membrane potential with large amplitude. These slow oscillations of the membrane potential in cells of layer 2/3 disappear as soon as the animal starts active whisking. Interestingly these correlations of membrane potential dynamics with behavior are not observable in the firing of action potentials which on average across cells has a frequency of 1Hz during rest and active whisking [103]. These findings imply that there are sub-threshold changes in membrane potentials and thus changes in the excitability of neurons that are related to the behavioral state of the animal. Further it has been discovered that these changes influence the processing of sensory information in barrel cortex. Passive deflection of a whisker by the experimenter during quiet wakefulness of the animal leads to a strong cortical response, while this response is weak during deflections that occur during active whisking [61]. In addition to a low tactile response amplitude active whisking is characterized by a narrow spatial representation in barrel cortex and elevated background firing as opposed to the wide spatial representation and low background firing of passive whisker contacts. The switching between the active and passive cortical states occurs within 100ms and is unrelated to the alertness of the animal. Since switching between cortical states persists after transection of the infraorbital nerve and substitution of whisker contacts by direct electrical stimulation over a cuff electrode, the modulator

signal presumably has a central origin. One likely central source are the motor commands of the animal that initiate the whisker movement [61]. These observations concerning the state dependent modulation of stimulation evoked responses supports the idea of adaptive microstimulation where decoding of the cortical state and appropriate adjustment of stimulation parameters are hoped to stabilize evoked cortical potentials.

Occurrences in this domain
are beyond the reach of exact prediction
because of the variety of factors in operation,
not because of any lack of order in nature.

Albert Einstein (1879-1955)

3 Support Vector Regression

Biological systems are often governed by large numbers of different factors, which makes it virtually impossible to obtain exact predictions about the quantities of interest. Adaptive stimulation, as described in section 2.3.2, requires that stimulus parameters can be determined on the basis of the ongoing background brain activity, and thus also suffers from this shortcoming, due to the inherent complexity of the brain. Since it is futile to build an explicit model of the brain, the only way to tackle the problem of adaptive stimulation is by learning the needed relationships from example stimulation trials. To be more concrete, one wants to learn a mapping from ongoing brain activity and stimulation response to a particular stimulation parameter, the stimulation intensity for instance. In general these parameters will take on real values on a continuous scale and hence the desired mapping turns out to be the solution of a regression problem.

3.1 State of the art

The Support Vector Machine (SVM), like neural networks is a supervised learning algorithm that can be used to solve both linear and nonlinear classification and regression problems [45]. Historically SVMs were first used to solve linear classification problems [142] and later extended to handle regression. The initial limitation to compute linear relationships was later removed by introducing the concept of kernel functions [124] that implicitly compute a nonlinear mapping to a so called feature space. By using a kernel function, SVMs still compute a linear function, but this functions now resides in the feature space, instead of the input space that is spanned by the training patterns. The feature space is also called reproducing kernel Hilbert space (RKHS). Although neural networks and other supervised learning algorithms can also learn linear and nonlinear relationships based on a set of training examples, and have been successfully applied to difficult real world problems [83, 10], SVMs have gained popularity in recent years since they have some exclusive properties not shared by other supervised algorithms. First, SVMs allow the incorporation of prior knowledge about an application via the kernel function, which leads to a separation between learning algorithm and application-specific extensions. On the one hand this means that problems in new application domains can be solved without changing the

basic algorithm and on the other hand that improvements in the learning algorithm are instantly available to all applications. The development of a special kernel function for adaptive stimulation will be described in section 6.1.4. Second, the solution of the SVM optimization problem can be used to determine bounds on the generalization performance of the algorithm which permits to automatically choose the free hyper-parameters like kernel and loss function parameters. The process of choosing suitable hyper-parameters is called model selection and will be discussed in chapter 5. Third, the SVM optimization problem involves minimization of a convex function which ensures that there is a single, although not unique, global minimum. After training, supervised algorithms often provide a solution that works like a black box when it comes to prediction on unseen data. In some application domains this can be a severe restriction, as the black box prevents interpretation of the learned function, and consequently the improvement of the algorithm. As described later in this chapter, the SVM solution is a linear combination of a subset of the training patterns, termed support vectors, and hence does not have this black box nature. In addition it is possible to directly inspect the SVM weight vector even in the nonlinear case by computing the corresponding pre-image of the weight vector that resides in the RKHS [123]. Taken together this facilitates the interpretation of the SVM solution.

The adaptation of stimulus parameters requires the solution of a regression problem and therefore the rest of this chapter will be concerned with Support Vector Regression (SVR) only. In a regression problem one is given d-dimensional training patterns $x_i \in \mathbb{R}^d$ and target values $y_i \in \mathbb{R}$ and wants to estimate a function $f(x_i) \mapsto y_i$. Assuming for the moment that $f(x_i) = \langle w, x_i \rangle + b$ is a linear function with weight vector $w \in \mathbb{R}^d$ and bias term $b \in \mathbb{R}$ the regression problem is usually solved by minimizing the squared loss $(f(x_i) - y_i)^2$ over all pairs of training patterns and target values (x_i, y_i). The result of this process is called the least-squares solution to the regression problem [54]. In contrast to least-squares regression, SVR is different in two aspects: First, it restricts the solution space of linear functions by regularization and tries to find the flattest function possible; Second, it uses an alternative loss function to measure the error between predicted values $f(x_i)$ and true target values y_i. The ε-insensitive loss function for SVR, shown in figure 3.1B, was introduced by [142] and is defined as follows:

$$l_\varepsilon(y_i - f(x_i)) = \max\{0, |y_i - f(x_i)| - \varepsilon\}^p \tag{3.1}$$

with $p = 1$. This means that training patterns that incur an error smaller than ε do not contribute to the solution and that errors that exceed ε are penalized linearly. A simple one-dimensional example of a regression problem is shown in figure 3.1A, where the circles represent training patterns (x_i, y_i) and the black solid line the function $f(x)$ estimated by SVR. The area that is insensitive to errors forms a tube of width ε around the function $f(x)$ and is indicated by the dotted lines in figure 3.1A. As a consequence, only the training patterns outside this tube (red circles in figure 3.1A) contribute to the SVR solution. This subset of training patterns are the support vectors and coined the algorithm's name 'Support Vector Machine'.

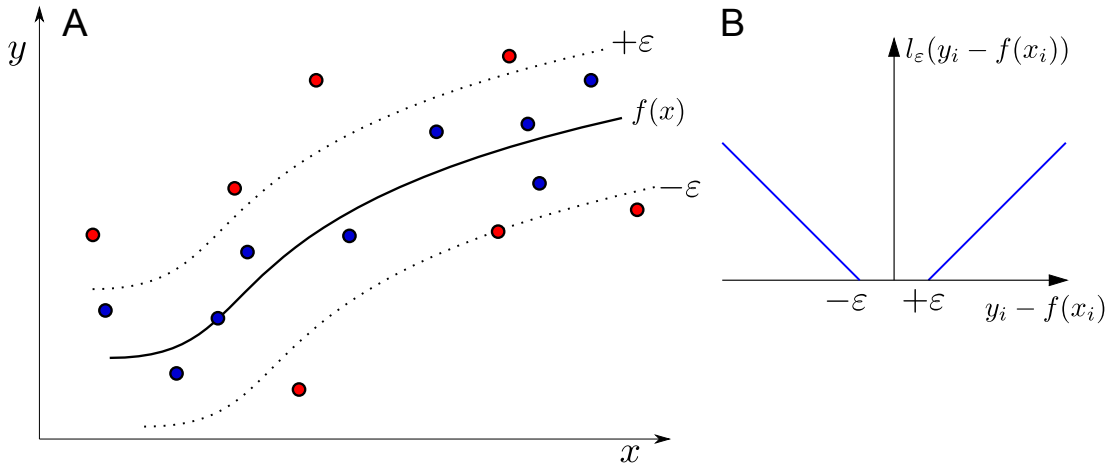


Figure 3.1: *Example of a one-dimensional regression problem.* ***A:*** *Pairs of training patterns x_i and target values y_i are represented by circles. The solid line indicates the function $f(x)$ estimated by SVR. By using the ε-insensitive loss function, only a subset of the training patterns that lie outside the tube indicated by the dotted lines contribute to the SVR solution.* ***B:*** *The ε-insensitive loss function penalizes errors linearly if they exceed the value ε.*

From this example it becomes clear that for a given value of ε one finds the function $f(x) = \langle w, x\rangle + b$, by minimizing the loss over all training patterns. Unfortunately the loss function given by equation (3.1) is not differentiable and hence the loss cannot be minimized by gradient descent algorithms. This problem can be avoided by introducing nonnegative slack variables ξ and ξ^* to rewrite the loss function as inequalities and minimizing the sum of the slack variables instead. In addition to minimizing the loss function SVR regularizes the solution by minimizing the squared norm of the weight vector $\|w\|^2$ and thus tries to find the flattest function possible. It is worthwhile to note that finding the flattest function is not a randomly imposed restriction but instead corresponds to finding a separating hyperplane with maximum margin in Support Vector Classification (SVC) [124]. Putting it all together the SVR problem can be formulated as the following optimization problem, with $p = 1$:

$$\begin{aligned} \min_{w,b,\xi,\xi^*} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i^p + \xi_i^{*p}) \\ \text{subject to} \quad & f(x_i) - y_i \leq \varepsilon + \xi_i \\ & y_i - f(x_i) \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \ \forall i = 1, \ldots, m \ . \end{aligned} \tag{3.2}$$

In equation (3.2) the objective function consists of the sum of the regularization term $\|w\|^2$ and a term involving the sum of slack variables multiplied by a positive parameter $C \in \mathbb{R}^+$. By changing the regularization parameter C, one can thus trade off between the complexity

of the function and the error on the training data. Since in practice it is desirable to have a function with good generalization performance, meaning that it gives accurate predictions on unseen data, both the regularization parameters C and the loss function parameter ε are chosen to optimize the performance as described in chapter 5.

The classic approach [124] to solve the optimization problem (3.2) is to derive its dual form as described in section 3.1.1 which turns out to be a quadratic programming problem. The connections between the primal and dual formulation of an optimization problem are explored in section 3.1.2 in the context of the regularized least squares algorithm. A recent approach that directly solves the primal SVR formulation without bias term b is presented in section 3.1.3. The extension of this approach to use a bias term b, which is an important part of this thesis' work, is developed in section 3.2. Finally section 3.3 compares the performance of the different solution approaches on various data sets from different application areas.

3.1.1 Dual SVR

The dual SVR formulation is derived from the primal optimization problem given in equation (3.2) by using nonnegative Lagrange multipliers α, α^*, β and β^* to incorporate the constraints into the objective function. The resulting function L also called Lagrangian can be written as follows

$$\begin{aligned} L(w, b, \xi^{(*)}, \alpha^{(*)}, \beta^{(*)}) =& \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) - \sum_{i=1}^{m}\alpha_i(\varepsilon + \xi_i - \langle w, x_i\rangle - b + y_i) \\ & - \sum_{i=1}^{m}\beta_i\xi_i - \sum_{i=1}^{m}\alpha_i^*(\varepsilon + \xi_i^* + \langle w, x_i\rangle + b - y_i) - \sum_{i=1}^{m}\beta_i^*\xi_i^* \ , \end{aligned} \tag{3.3}$$

and is minimized with respect to the primal variables w, b, ξ and ξ^* and maximized with respect to the Lagrange multipliers, or dual variables, α, α^*, β and β^*. As a shorthand $v^{(*)}$ will be used in the following to designate both non-starred v and starred v^* variables. The simultaneous minimization of primal and maximization of dual variables implies that one seeks a saddle point of the Lagrangian function, where the partial derivatives with respect to the primal variables have to vanish:

$$\frac{\partial L}{\partial w} = w + \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)x_i \overset{!}{=} 0 \Leftrightarrow w = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)x_i \tag{3.4}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) \overset{!}{=} 0 \tag{3.5}$$

$$\frac{\partial L}{\partial \xi} = C - \alpha_i^{(*)} - \beta_i^{(*)} \overset{!}{=} 0 \overset{\text{Since } \beta_i^{(*)} \geq 0}{\Leftrightarrow} 0 \leq \alpha_i^{(*)} \leq C \ . \tag{3.6}$$

From equation (3.4) it can be seen that the SVR weight vector is a linear combination of training patterns x_i, as already mentioned in the previous section, and support vectors

are those training patterns for which the difference in dual variables $(\alpha_i - \alpha_i^*)$ is nonzero. Using the expression of w in terms of α and α^* and exploiting the constraints on the dual variables in equations (3.5) and (3.6) it is possible to eliminate the primal variables in the Lagrangian (3.3) to arrive at the dual formulation of the SVR optimization problem:

$$\begin{aligned} \min_{\alpha,\alpha^*} \; & \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\left\langle x_i, x_j\right\rangle + \varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \; & \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0 \\ & 0 \leq \alpha_i^{(*)} \leq C, \; \forall i = 1, \ldots, m \; . \end{aligned} \tag{3.7}$$

The dual formulation of the SVR problem (3.7) is a quadratic program with a linear equality constraint and a box constraint on the dual variables $\alpha^{(*)}$. General quadratic programs are usually solved by interior point algorithms [131, 140], but there exist specific methods to solve the dual SVR formulation more efficiently, like sequential minimal optimization (SMO) [107] and projected gradient descent [151, 152]. The most widely used software implementation to solve the dual formulation, called LibSVM [27], uses a variant of SMO [48]. For more details on the relative benefits of these different solution methods and the description of a parallel dual SVR solver the interested reader is referred to [15].

3.1.2 Primal and dual optimization

Before describing the approach to directly solve the primal SVR problem in subsequent sections, it is fruitful to explore the connection between primal and dual optimization problems and to identify situations where solving the primal (3.2) or the dual (3.7) should be preferred. To simplify the ensuing discussion given in [29], the primal and dual optimization problems will be compared for the regularized least squares (RLS) algorithm. Given a matrix $X \in \mathbb{R}^{m\times d}$ that contains m training patterns with d dimensions the RLS objective function is defined as:

$$\min_{w} \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\|Xw - y\|^2 \; . \tag{3.8}$$

Similar to SVR the training patterns in X are linearly combined by the weight vector $w \in \mathbb{R}^d$ to approximate the targets y and the solution is regularized by the squared norm term $\|w\|^2$, where the regularization strength can be varied by the nonnegative hyper-parameter $\lambda \in \mathbb{R}^+$. The primal objective function (3.8) is minimized for $w = (\lambda I + X^T X)^{-1} X^T y$ and the value of the minimum is then given by:

$$y^T y - y^T X(\lambda I + X^T X)X^T y \; . \tag{3.9}$$

Analogous to the derivation of the dual SVR problem in section 3.1.1 the primal RLS problem is converted into its dual form by introducing slack variables $\xi = Xw - y$ to write

down the Lagrangian with dual variables α:

$$L(w, \xi, \alpha) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\xi^T\xi - \alpha(Xw - y - \xi) \; . \tag{3.10}$$

Again, the partial derivatives with respect to the primal variables have to vanish to fulfill the saddle point condition:

$$\frac{\partial L}{\partial w} = w - X^T\alpha \stackrel{!}{=} 0 \Leftrightarrow w = \frac{1}{\lambda}X^T\alpha \tag{3.11}$$

$$\frac{\partial L}{\partial \xi} = \xi + \alpha \stackrel{!}{=} 0 \Leftrightarrow \xi = -\alpha \; , \tag{3.12}$$

and can be used to establish the relationship between primal and dual solution via equation (3.11) and to eliminate the primal variables from the Lagrangian by employing both, equations (3.11) and (3.12). The result of this substitution is the dual objective function for the RLS algorithm:

$$\max_{\alpha} 2\alpha^T y - \frac{1}{\lambda}\alpha^T(XX^T + \lambda I)\alpha \; , \tag{3.13}$$

which attains its maximum at $\alpha = \lambda(XX^T + \lambda I)^{-1}y$ where the value of the maximum is given by:

$$\lambda y^T(XX^T + \lambda I)^{-1}y \; . \tag{3.14}$$

According to duality theory the minimum value (3.9) of the primal objective and the maximum value of dual objective (3.14) should be equal. That this is actually the case becomes apparent when the inverses of $\lambda I + X^TX$ and $XX^T + \lambda I$ are related by the Sherman-Morrison-Woodbury formula [54]:

$$\lambda(XX^T + \lambda I)^{-1} = I - X(\lambda I + X^TX)^{-1}X^T \; . \tag{3.15}$$

From the viewpoint of computational complexity the primal optimization requires inversion of the matrix $\lambda I + X^TX$ with complexity $\mathcal{O}(md^2 + d^3)$ and the dual optimization requires $\mathcal{O}(m^2d + m^3)$ operations to compute the inverse of matrix $XX^T + \lambda I$. At first sight it therefore seems beneficial to either solve the primal or dual problem depending on whether m is larger or smaller that d in order to achieve the lowest computational complexity. But this argument is flawed since it is always possible to use the Sherman-Morrison-Woodbury formula (4.28) to invert the smaller of the two matrices $\lambda I + X^TX$, or $XX^T + \lambda I$.

So what is the advantage of solving the primal instead of the dual optimization problem? The difference between primal and dual formulations is important when one seeks an approximate solution for the primal optimization problem. To illustrate this, it instructive to optimize both the primal (3.8) and dual (3.13) objective function by the conjugate gradient (CG) method [6] and to observe how the value of the primal objective function decreases as a function of the number of CG iterations. During the optimization of the dual objective function an approximate solution for α can be converted to a primal solution w by using equation (3.11). Figure 3.2 shows the decrease of the primal objective function value

in dependence of the number of conjugate CG iterations for primal and dual optimization on different subsets of the triazines data set. For some cases the difference between primal and dual optimization is small (figure 3.2A) while in other cases the dual optimization converges slower than the primal one (figure 3.2B).

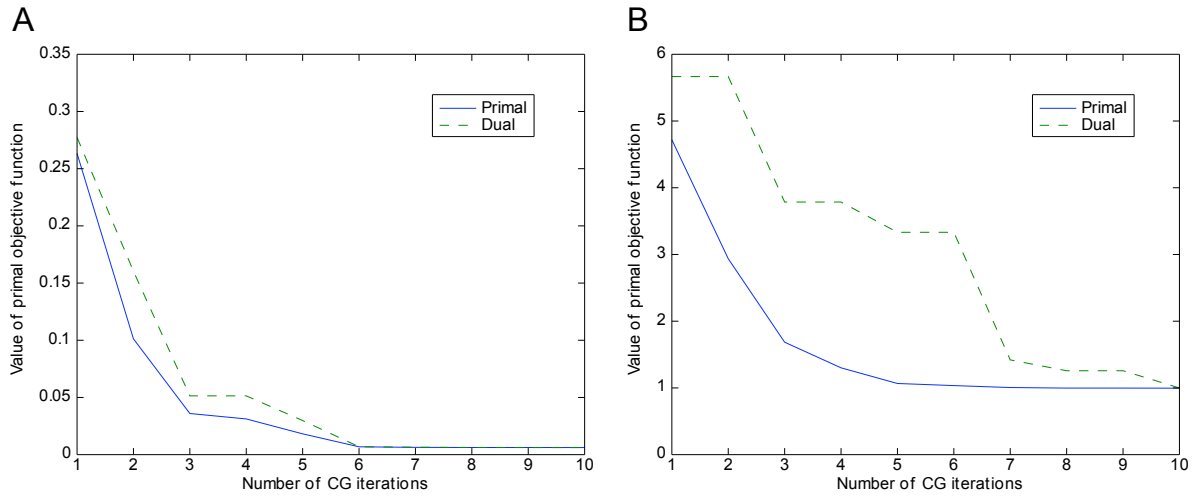


Figure 3.2: *Convergence of the primal and dual solution for a regularized least squares problem on different subsets of the triazines data set. In both experiments the regularization parameter was set to* $\lambda = 0.1$. ***A:*** *For* $m = 10$ *training patterns with* $d = 60$ *dimensions the convergence speed of primal and dual solution is almost identical* ***B:*** *For* $m = 60$ *training patterns with* $d = 10$ *dimensions the dual solution needs more conjugate gradient (CG) iterations than the primal solution to converge to the minimum value of the primal objective function.*

Intuitively it is clear that directly minimizing the quantity of interest, as done in primal optimization, is superior in terms of convergence speed and this is confirmed by the example shown in figure 3.2. But it can be proven that even after a single CG iteration the primal optimization always yields a lower value than the dual optimization [29].

Computing approximate solutions for regression problems is interesting in the context of large scale data sets and there have been suggestions to introduce approximations to the dual SVR formulation [138, 139], although this seems not to be the right approach in light of the preceding discussion. Besides large scale optimization computing approximations for the primal problem plays an essential role in the development of a new online learning algorithm that will be described in chapter 4. It is based on the insight that online learning can be considered as computing an approximation to an offline learning problem since the time for each optimization step and the number of available training patterns in online learning is limited.

3.1.3 Primal SVR without bias

This section describes an approach proposed by [11] for solving the primal SVR problem in equation (3.2) directly, but without optimizing the bias term b. As a first step the slack variables ξ_i and ξ_i^* are removed from the objective function by using the definition (3.1) of the ε insensitive loss function. By dividing the resulting primal objective function by C one arrives at the following unconstrained formulation for the primal SVR problem:

$$\min_w L_\varepsilon(w) = \sum_{i=1}^{m} l_\varepsilon(\langle w, x_i\rangle - y_i) + \lambda\|w\|^2 \ , \tag{3.16}$$

where the new regularization parameter is given by $\lambda = \frac{1}{2C}$. So far only linear functions $f(x_i) = \langle w, x_i\rangle$ were considered in this chapter. Nonlinear SVR is obtained by introduction of a kernel function $k(x_i, x_j)$ and an associated Hilbert space $\mathcal{H}$ that fulfills the so called "reproducing property" [2]:

$$f(x_i) = \langle f, k(., x_i)\rangle_{\mathcal{H}} \ . \tag{3.17}$$

In words the reproducing property states that the function f can be evaluated on training pattern x_i by computing the dot product of function f with the kernel function $k(., x_i)$ centered on the pattern x_i in the Hilbert space $\mathcal{H}$. By exploiting the reproducing property, equation (3.16) can be reformulated to deal with nonlinear functions:

$$\min_f L_\varepsilon(f) = \sum_{i=1}^{m} l_\varepsilon(f(x_i) - y_i) + \lambda\|f\|_{\mathcal{H}}^2 \ . \tag{3.18}$$

Equivalent to the linear formulation the second term in the objective function serves to regularize the solution by minimizing the squared norm of the function in the Hilbert space. Before it is possible to derive a closed-form expression for $\|f\|_{\mathcal{H}}^2$ it is necessary to introduce the "representer theorem" [76] given by:

$$f(x) = \sum_{i=1}^{m} \beta_i k(x, x_i) \ . \tag{3.19}$$

According to this theorem each function f in the Hilbert space $\mathcal{H}$ can be expressed as a linear combination of kernel functions $k(., x_i)$ that are centered on the training patterns x_i. When combining the representer theorem (3.19) with the reproducing property (3.17) the squared norm term in the objective function can be expanded to $\|f\|_{\mathcal{H}}^2 = \langle f, f\rangle_{\mathcal{H}} = \sum_{i,j=1}^{m} \beta_i\beta_j k(x_i, x_j)$. From these relationships it becomes clear that function f can be minimized by optimizing its expansion coefficients β_i. Substitution into equation (3.18) yields the following equivalent objective function:

$$\min_\beta L_\varepsilon(\beta) = \sum_{i=1}^{m} l_\varepsilon(\sum_{j=1}^{m} \beta_j k(x_i, x_j) - y_i) + \lambda \sum_{i,j=1}^{m} \beta_i\beta_j k(x_i, x_j) \ . \tag{3.20}$$

By defining the kernel matrix $K \in \mathbb{R}^{m \times m}$ via its entries $K_{ij} = k(x_i, x_j)$ and letting K_i denote the i-th row of the kernel matrix the objective function (3.20) can be written in more compact notation as:

$$\min_{\beta} L_\varepsilon(\beta) = \sum_{i=1}^{m} l_\varepsilon(K_i\beta - y_i) + \lambda\beta^T K\beta \; . \tag{3.21}$$

This is an unconstrained optimization problem that can be solved by gradient descent only, if the loss function l_ε is differentiable. As already mentioned in the introductory section of this chapter the loss function in equation (3.1) is not differentiable for $p = 1$, due to the hinge points that lie at the border of the ε insensitive zone and the linear part of the loss function (figure 3.3). By setting $p = 2$ in definition (3.1) these discontinuities in the first derivative of l_ε vanish due to the quadratic part outside the ε insensitive zone. For $p = 1$ the resulting loss function is alternatively called the l_1 loss and for $p = 2$ the l_2 loss. Thus, by changing the definition of the loss function, the problem (3.1) is actually solvable by gradient descent methods. But are there possible drawbacks when one uses the l_2 loss instead of the l_1 loss? As the shaded area in figure 3.3 indicates the l_2 loss penalizes errors less than the l_1 loss when they cross the ε threshold. Consequently the l_2 loss tends to produce more support vectors than the l_1 loss for a fixed value of the regularization parameter λ. Although this does not pose a problem during the SVR training it can be detrimental for the run time during the prediction on unseen data, since more evaluations of the kernel function are necessary. Yet, it should be noted that this is just a problem for nonlinear SVR functions since the weight vector w can be computed once prior to the prediction for linear functions.

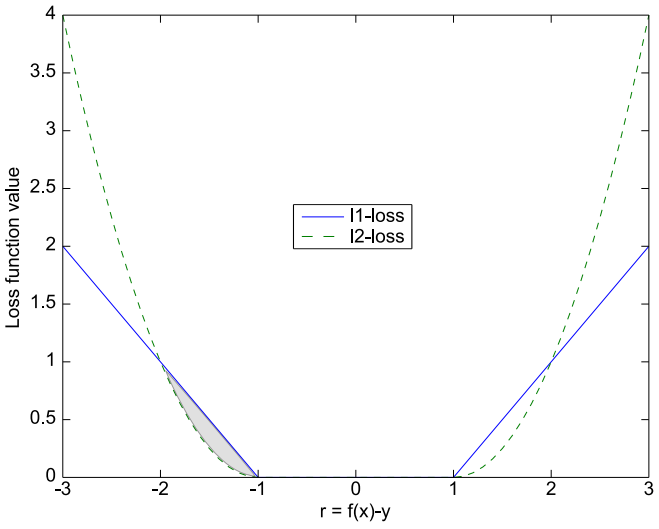


Figure 3.3: *Comparison of l_1 and l_2 loss functions. The l_1 loss is not differentiable due to discontinuities in its first derivative at the border of the ε-insensitive zone. The l_2 loss is differentiable but produces more support vectors since errors that just exceed the threshold ε are penalized less in comparison to the l_1 loss as indicated by the shaded area.*

Another drawback of the l_2 loss function is its tendency to be less robust to outliers in the training data. This is illustrated by a simple 1-dimensional example in figure 3.4. For

this example training patterns were generated by uniformly sampling the domain of the function

$$\operatorname{sinc}(x) = \begin{cases} 1, & x = 0 \\ \frac{\sin(\pi x)}{\pi x}, & x \neq 0 \ , \end{cases} \tag{3.22}$$

and manual addition of two outliers. If the training data does not contain any outliers the SVR solution using l_1 or l_2 loss is identical (figure 3.4A). But after adding the outliers to the training data the SVR solution using the l_2 loss shows large deviations from the true underlying sinc-function (figure 3.4B). Although the l_2 loss is more sensitive to outliers

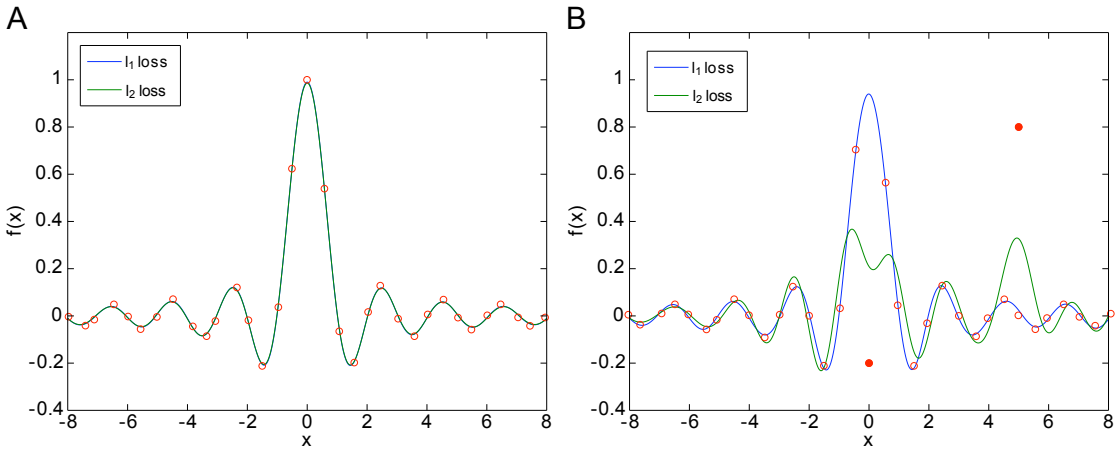


Figure 3.4: *Robustness of l_1 and l_2 loss with respect to outliers.* ***A:*** *When clean training data (red open circles) is created by sampling the* sinc*-function the SVR solutions produced by the l_1 and l_2 loss are indistinguishable.* ***B:*** *After adding two outliers (red closed circles) the function estimated using the l_2 loss strongly deviates from the true* sinc*-function at the corresponding locations.*

than the l_1 loss function, the results of section 3.3.1 imply that the impact on the prediction accuracy of SVR is not as severe as expected when considering real data sets. Nevertheless it is possible to overcome this drawback and still ensure the differentiability of the loss function by introducing the ε-insensitive Huber loss function [11] defined by:

$$l_{\varepsilon,\Delta}(r) = \begin{cases} 0, & |r| \leq \varepsilon \\ (|r| - \varepsilon)^2, & \varepsilon < |r| < \Delta \\ (\Delta - \varepsilon)(2|r| - \Delta - \varepsilon), & |r| \geq \Delta \ . \end{cases} \tag{3.23}$$

Equation (3.23) defines a family of loss functions that are governed by the two nonnegative parameters ε and Δ. Different choices for these parameters yield six distinct types of loss functions that are shown in figure 3.5. In particular, the already introduced l_2 and l_1 loss functions are recovered by setting $\varepsilon > 0, \Delta = +\infty$ or $\varepsilon = 0, \Delta > 0$ respectively (figure 3.5D and F). It is important to note that all loss functions in the left column of figure 3.5 have

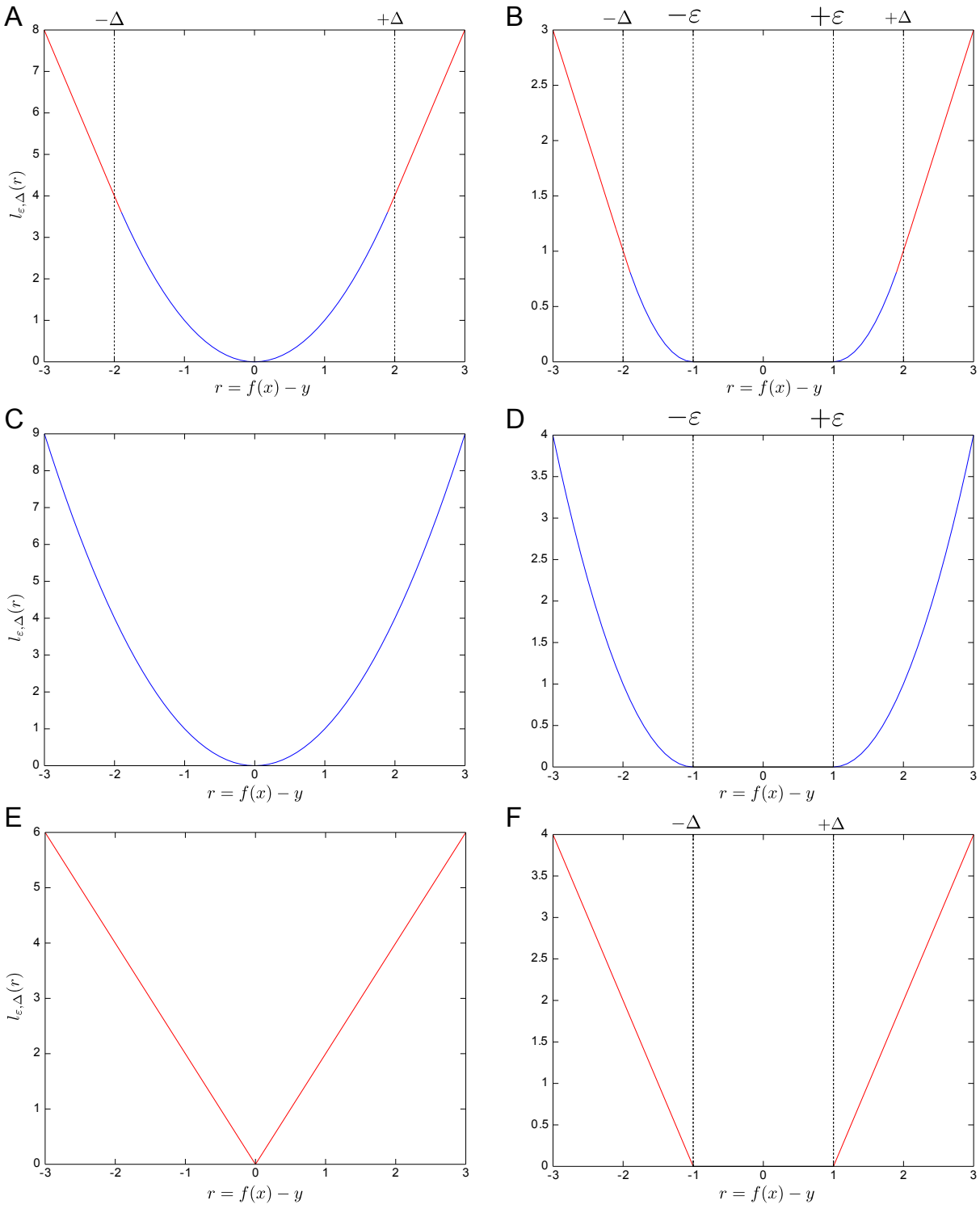


Figure 3.5: *Overview of a family of loss functions parameterized by ε and Δ. Blue lines indicate the quadratic, and red lines the linear parts of the loss function.* ***A:*** *Huber loss.* ***B:*** *ε-insensitive Huber loss.* ***C:*** *Quadratic loss.* ***D:*** *ε-insensitive quadratic (l_2) loss.* ***E:*** *Linear (Laplace) loss.* ***F:*** *ε-insensitive linear (l_1) loss.*

$\varepsilon = 0$ and the resulting algorithm cannot be considered as a support vector algorithm anymore, since all training patterns contribute to the solution. For example the quadratic loss with $\varepsilon = 0, \Delta = +\infty$ leads to the kernel ridge regression algorithm [120]. Further, the regularized least squares algorithm described in section 3.1.2 can be seen as kernel ridge regression with a linear kernel function. But the most important conclusion concerning the ε-insensitive Huber loss is that the non-differentiable l_1 loss can be approximated to arbitrary precision by a differentiable loss function by choosing $\varepsilon > 0$ and letting Δ approach ε.

The primal optimization approach was first introduced in [74] for linear SVC and later extended to nonlinear SVC with and without bias term b in [29]. Building on the results of [29] the primal optimization was introduced for nonlinear SVR without bias term b by [11] via the ε-insensitive Huber loss as described in this section. The next section will describe the extension of this work to primal SVR with bias term and give details on a practical implementation of the resulting algorithm.

3.2 Primal SVR with bias

By including the bias term b in equation (3.21) and using the definition of the ε-insensitive Huber loss function in equation (3.23) it is straightforward to formulate the primal optimization problem for SVR with bias:

$$\min_{\beta,b} L_{\varepsilon,\Delta}(\beta, b) = \sum_{i=1}^{m} l_{\varepsilon,\Delta}(K_i\beta + b - y_i) + \lambda\beta^T K\beta \ . \tag{3.24}$$

In order to simplify the subsequent derivations it is necessary to establish some notation. First $r(\beta, b)$, the residual vector, is used as shorthand to denote the difference between the function value and targets:

$$r(\beta, b) = K\beta + b - y \ , \tag{3.25}$$

and $r_i(\beta, b)$ is the i-th component of this vector. Second it is convenient to define special sign functions for the quadratic and linear parts of the ε-insensitive Huber loss function (figure 3.5B). For the quadratic part the sign function is defined as:

$$s_i(\beta, b) = \begin{cases} +1, & \varepsilon < r_i(\beta, b) < \Delta \\ -1, & -\Delta < r_i(\beta, b) < -\varepsilon \\ 0, & \text{otherwise} \end{cases} , \tag{3.26}$$

and for the linear part as:

$$\bar{s}_i(\beta, b) = \begin{cases} +1, & \Delta \geq r_i(\beta, b) \\ -1, & r_i(\beta, b) \leq -\Delta \\ 0, & \text{otherwise} \ . \end{cases} \tag{3.27}$$

Finally, by defining an indicator function for the quadratic part of the ε-insensitive Huber loss via $w_i(\beta, b) = s_i(\beta, b)^2$, the loss function $l_{\varepsilon,\Delta}$ in equation (3.24) can be expanded as follows:

$$\begin{aligned} L_{\varepsilon,\Delta}(\beta, b) =& \sum_{i=1}^{m} w_i(\beta, b)(r_i(\beta, b) - \varepsilon)^2 + \sum_{i=1}^{m} \bar{s}_i(\beta, b)(\Delta - \varepsilon)(2|r_i(\beta, b)| - \Delta - \varepsilon) + \lambda \beta^T K \beta \\ =& r(\beta, b)^T W(\beta, b) r(\beta, b) - 2\varepsilon s(\beta, b)^T r(\beta, b) + \varepsilon^T W(\beta, b)\varepsilon \\ &+ 2(\Delta - \varepsilon)\bar{s}(\beta, b)^T r(\beta, b) - (\Delta^2 - \varepsilon^2)\bar{s}(\beta, b)^T \bar{s}(\beta, b) + \lambda \beta^T K \beta \ . \end{aligned} \tag{3.28}$$

Here the first term in the sum represents the quadratic, and the second term the linear part of the ε-insensitive Huber loss function. Further, the expression $W(\beta, b)$ is a diagonal matrix defined as $W(\beta, b) = \text{diag}\{w_1(\beta, b), \ldots, w_m(\beta, b)\}$, where the i-th entry on the diagonal is $w_i(\beta, b)$.

Since the ε-insensitive Huber loss function is differentiable, the objective function in equation (3.28) can be minimized by a descent algorithm. In general a descent algorithm starts with an initial guess β_0 and b_0 for the optimization variables and then determines the solution at the k-th iteration by setting:

$$\begin{pmatrix} \beta_{k+1} \\ b_{k+1} \end{pmatrix} = \begin{pmatrix} \beta_k \\ b_k \end{pmatrix} + \rho_k d_k \ , \tag{3.29}$$

where $d_k \in \mathbb{R}^m$ is the descent direction and $\rho_k \in \mathbb{R}^+$ is a positive step size. Different choices for the descent direction d_k lead to different unconstrained optimization algorithms for solving the primal SVR problem. For example, the steepest descent, diagonally scaled steepest descent, and the Newton algorithms result for particular choices of d_k [6].

In the subsequent description the objective function (3.28) is minimized as proposed in [74] by Newton's algorithm with:

$$d_k = -\nabla^2 L_{\varepsilon,\Delta}(\beta, b)^{-1} \nabla L_{\varepsilon,\Delta}(\beta, b) \ , \tag{3.30}$$

where $L_{\varepsilon,\Delta}(\beta, b)$ is the gradient and $\nabla^2 L_{\varepsilon,\Delta}(\beta, b)$ the hessian of the objective function. Section 3.2.1 describes how to calculate the gradient and hessian for the primal objective in equation (3.28). Usually these calculations are straightforward but for the primal SVR problem it is mandatory to find suitable factorizations of the resulting matrices to achieve an efficient optimization algorithm. Issues concerning the numerically stable inversion of the resulting system of linear equations are discussed in section 3.2.2. Further, an exact line search, described in section 3.2.3, will be used to determine the step size ρ_k in each iteration. This will guarantee the convergence of the resulting optimization algorithm [6].

3.2.1 Newton step

The Newton step for computing the solution at the next iteration is given by substituting equation (3.29) into equation (3.30). The partial derivatives with respect to β and b of the

first term in (3.28) are given by:

$$\begin{aligned} \frac{\partial r(\beta,b)}{\partial \beta} = K,\ \frac{\partial W(\beta,b)}{\partial \beta} = 0 \Rightarrow \frac{\partial r(\beta,b)^T W(\beta,b) r(\beta,b)}{\partial \beta} = 2K^T W(\beta,b) r(\beta,b) \\ \frac{\partial r(\beta,b)}{\partial b} = \mathbf{1},\ \frac{\partial W(\beta,b)}{\partial b} = 0 \Rightarrow \frac{\partial r(\beta,b)^T W(\beta,b) r(\beta,b)}{\partial b} = 2\mathbf{1}^T W(\beta,b) r(\beta,b)\ . \end{aligned} \tag{3.31}$$

The symbol $\mathbf{1}$ represents a vector of appropriate length that has all entries equal to one. By temporarily ignoring the multiplicative constants $(\Delta - \varepsilon)$ and ε the derivatives for the second and the fourth term in (3.28) can be deduced as:

$$\frac{\partial s(\beta,b)}{\partial \beta} = \frac{\partial s(\beta,b)}{\partial b} = 0 \Rightarrow \frac{\partial s(\beta,b) r(\beta,b)}{\partial \beta} = K^T s(\beta,b),\ \frac{\partial s(\beta,b) r(\beta,b)}{\partial b} = \mathbf{1}^T s(\beta,b)\ , \tag{3.32}$$

where $s(\beta,b)$ and $\bar{s}(\beta,b)$ are interchangable. Terms three and five in (3.28) are independent of β and b leading to:

$$\frac{\partial \varepsilon^T W(\beta,b)\varepsilon}{\partial \beta} = \frac{\partial \varepsilon^T W(\beta,b)\varepsilon}{\partial b} = \frac{\partial \bar{s}(\beta,b)^T \bar{s}(\beta,b)}{\partial \beta} = \frac{\partial \bar{s}(\beta,b)^T \bar{s}(\beta,b)}{\partial b} = 0\ . \tag{3.33}$$

Finally the derivatives for the last term in (3.28) are given by:

$$\frac{\partial \beta^T K \beta}{\partial \beta} = 2K\beta,\ \frac{\partial \beta^T K \beta}{\partial b} = 0\ . \tag{3.34}$$

Combining equations (3.31), (3.32), (3.33), and (3.34) the full gradient $\nabla L_{\varepsilon,\Delta}(\beta,b)$ can be written as follows:

$$\begin{aligned} \nabla L_{\varepsilon,\Delta}(\beta,b) = 2\begin{pmatrix} K^T(W(\beta,b)r(\beta,b) - \varepsilon s(\beta,b) + (\Delta-\varepsilon)\bar{s}(\beta,b) + \lambda\beta \\ \mathbf{1}^T(W(\beta,b)r(\beta,b) - \varepsilon s(\beta,b) + (\Delta-\varepsilon)\bar{s}(\beta,b)) \end{pmatrix} = 2\begin{bmatrix} K^T & 0 \\ \mathbf{1}^T & -\lambda \end{bmatrix} \cdot \\ \left(\begin{bmatrix} (W(\beta,b)K + \lambda I) & W(\beta,b)\mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{pmatrix} \beta \\ b \end{pmatrix} - \begin{pmatrix} W(\beta,b)y + \varepsilon s(\beta,b) - (\Delta-\varepsilon)\bar{s}(\beta,b) \\ 0 \end{pmatrix} \right) \end{aligned} \tag{3.35}$$

At first sight the expansion of the gradient vector given by the second equality in (3.35) seems to complicate the expression, but it is crucial to factorize $\nabla L_{\varepsilon,\Delta}(\beta,b)$ in this way to later simplify the Newton. For computation of the hessian let $\nabla L_{\varepsilon,\Delta}(\beta,b)_1$ be the first component, and $\nabla L_{\varepsilon,\Delta}(\beta,b)_2$ the second component of the gradient. Then the partial derivatives with respect to β for the first and second component are given by:

$$\frac{\partial \nabla L_{\varepsilon,\Delta}(\beta,b)_1}{\partial \beta} = 2(K^T W(\beta,b)K + \lambda K),\quad \frac{\partial \nabla L_{\varepsilon,\Delta}(\beta,b)_2}{\partial \beta} = 2\mathbf{1}^T W(\beta,b)K\ , \tag{3.36}$$

and with respect to the bias term b by:

$$\frac{\partial \nabla L_{\varepsilon,\Delta}(\beta,b)_1}{\partial b} = 2K^T W(\beta,b)\mathbf{1},\quad \frac{\partial \nabla L_{\varepsilon,\Delta}(\beta,b)_2}{\partial b} = 2\mathbf{1}^T W(\beta,b)\mathbf{1}\ . \tag{3.37}$$

Again, by combining equations (3.36) and (3.37) the full hessian $\nabla^2 L_{\varepsilon,\Delta}(\beta, b)$ can be written as:

$$\begin{aligned}\nabla^2 L_{\varepsilon,\Delta}(\beta,b) &= 2\begin{bmatrix} K^T(W(\beta,b)K+\lambda I) & K^T W(\beta,b)\mathbf{1} \\ \mathbf{1}^T W(\beta,b)K & \mathbf{1}^T W(\beta,b)\mathbf{1}\end{bmatrix} \\ &= 2\begin{bmatrix} K^T & 0 \\ \mathbf{1}^T & -\lambda\end{bmatrix}\begin{bmatrix} W(\beta,b)K+\lambda I & W(\beta,b)\mathbf{1} \\ \mathbf{1}^T & 0\end{bmatrix}.\end{aligned} \tag{3.38}$$

In the Newton algorithm the descent direction is given by multiplying the negative inverse of the hessian with the gradient. Explicitly computing the hessian inverse changes the order of factors in equation (3.38) and multiplication with the gradient in equation (3.35) hence completely cancels out the first factors. In addition the second factor of the hessian inverse cancels out the first factor inside the brackets of equation (3.35) and it now becomes clear why the gradient expression was expanded into that particular matrix vector product. Combining equations (3.38) and (3.35) the resulting Newton step is:

$$\begin{aligned}\begin{pmatrix}\bar\beta \\ \bar b\end{pmatrix} &= -(\nabla^2 L_{\varepsilon,\Delta}(\beta,b))^{-1}\nabla L_{\varepsilon,\Delta}(\beta,b) \\ &= \begin{bmatrix} W(\beta,b)K+\lambda I & W(\beta,b)\mathbf{1} \\ \mathbf{1}^T & 0\end{bmatrix}^{-1}\begin{pmatrix} W(\beta,b)y+\varepsilon s(\beta,b)-(\Delta-\varepsilon)\bar s(\beta,b) \\ 0\end{pmatrix} - \begin{pmatrix}\beta \\ b\end{pmatrix} \\ &= \begin{bmatrix} \overbrace{\begin{pmatrix} K_{sv_1,sv_1}+\lambda I & \mathbf{1}_{sv_1} \\ 1\dots1 & 0\end{pmatrix}}^{A_1} & \overbrace{\begin{pmatrix} K_{sv_1,sv_2} \\ 1\dots1\end{pmatrix}}^{A_2} & \overbrace{\begin{pmatrix} K_{sv_1,nsv} \\ 1\dots1\end{pmatrix}}^{A_3} \\ 0 & \lambda I & 0 \\ & 0 & \lambda I\end{bmatrix}^{-1}\begin{pmatrix} y_{sv_1}+\varepsilon s(\beta,b)_{sv_1} \\ 0 \\ (\Delta-\varepsilon)\bar s(\beta,b)_{sv_2} \\ 0\end{pmatrix} - \begin{pmatrix}\beta_{sv_1} \\ b \\ \beta_{sv_2} \\ \beta_{nsv}\end{pmatrix} \\ &= \begin{bmatrix} A_1^{-1}\begin{pmatrix} y_{sv_1}+\varepsilon s(\beta,b)_{sv_1} \\ 0\end{pmatrix} - \frac{1}{\lambda}A_1^{-1}A_2(\Delta-\varepsilon)\bar s(\beta,b)_{sv_2} \\ \frac{1}{\lambda}(\Delta-\varepsilon)\bar s(\beta,b)_{sv_2} \\ 0\end{bmatrix} - \begin{pmatrix}\begin{pmatrix}\beta_{sv_1} \\ b\end{pmatrix} \\ \beta_{sv_2} \\ \beta_{nsv}\end{pmatrix}.\end{aligned} \tag{3.39}$$

In equation (3.39) the sets $sv_1 = \{i, |r_i(\beta,b)| \geq \Delta\}$, $sv_2 = \{i, \varepsilon < |r_i(\beta,b)| < \Delta\}$, and $nsv = \{i, |r_i(\beta,b)| < \varepsilon\}$ are the index sets of training points with residual in the quadratic, linear, and zero part of the loss function. The second equality in (3.39) is obtained by expanding the definition of $W(\beta,b)$ followed by reordering the optimization variables such that the bias term comes to lie below the optimization variables β_i with $i \in sv_1$. With definition of the submatrices A_1, A_2 and A_3, the last equality in (3.39) can be deduced from the matrix inversion lemma 3.2.1.

Lemma 3.2.1. *Let $A_1 \in \mathbb{R}^{n\times n}$ be a non singular square matrix, $A_2 \in \mathbb{R}^{n\times m}$, $A_3 \in \mathbb{R}^{n\times k}$ and $\lambda \in \mathbb{R}\setminus\{0\}$. Then the inverse of the block matrix parameterized by A_1, A_2, A_3 and λ is given by:*

$$\begin{bmatrix} A_1 & A_2 & A_3 \\ 0 & \lambda I & 0 \\ 0 & 0 & \lambda I\end{bmatrix}^{-1} = \begin{bmatrix} A_1^{-1} & -\frac{1}{\lambda}A_1^{-1}A_2 & -\frac{1}{\lambda}A_1^{-1}A_3 \\ 0 & \frac{1}{\lambda}I & 0 \\ 0 & 0 & \frac{1}{\lambda}I\end{bmatrix}.$$

Lemma 3.2.1 can be verified by multiplying both sides of the equation with the original block matrix.

Now it becomes apparent that the factorizations introduced for the gradient in equation (3.35) and the hessian in equation (3.38) not only simplify the expression for the Newton step, but also avoid inversion of the full kernel matrix K for all training patterns. The only inversion needed to compute the Newton step involves the square matrix A_1 with size equal to the number of support vectors $|sv_1|$ in the quadratic part of the loss function. Unfortunately the matrix A_1 can be singular when none of the training patterns incurs a loss in the quadratic part, since then $sv_1 = \emptyset$ and $A_1 = 0$. It is important to note that this pathological situation cannot arise for the primal SVR problem without bias term b described in section 3.1.3.

There are two possibilities to circumvent this problem. The first possibility involves additional regularization of bias b by including the term λb^2 in the objective function (3.28). An empty set of support vectors $sv_1 = \emptyset$ then leads to $A_1 = -1$ which can always be inverted. Although feasible, this approach will not be pursued any further, as it complicates the comparison with the classical dual solution of the SVR problem. The second possibility uses the ε-insensitive quadratic loss (figure 3.5D) instead of the ε-insensitive Huber loss function by setting $\Delta = +\infty$. This does not avoid singularity of A_1 since the set sv_1 can still be empty, but now the optimization algorithm can be safely terminated when this condition arises since there is no set sv_2. In particular, with the ε-insensitive quadratic loss, sv_1 can be empty only when ε was chosen too large a priori by the user of the algorithm. All the derivations given in the following sections will therefore concentrate on using the ε-insensitive quadratic loss.

3.2.2 Cholesky factorization

For solving the system of linear equations involving matrix A_1 in (3.39) the following matrix inversion lemma 3.2.2 is useful.

Lemma 3.2.2. *Let $B \in \mathbb{R}^{n \times n}$ be a non singular square matrix, $v \in \mathbb{R}^n$ a vector, and $\mu \in \mathbb{R}$ a scalar. Then the inverse of the block matrix parameterized by B, v and μ is given by:*

$$\begin{bmatrix} B & v \\ v^T & \mu \end{bmatrix}^{-1} = \frac{1}{\psi} \begin{bmatrix} \psi B^{-1} - B^{-1} v v^T B^{-1} & B^{-1} v \\ v^T B^{-1} & -1 \end{bmatrix}, \text{ with } \psi = v^T B^{-1} v - \mu \; .$$

As before, this lemma can be verified by multiplying both sides of the equation with the original block matrix. The inverse matrix A_1^{-1} can be identified with the left side of the equation in lemma 3.2.2 by setting $B = K + \lambda I$, $\mu = 0$, and $v = \mathbf{1}$. The system of linear equations to be solved for the ε-insensitive quadratic loss is:

$$A_1^{-1} \begin{pmatrix} y_{sv_1} + \varepsilon s(\beta, b)_{sv_1} \\ 0 \end{pmatrix} = A_1^{-1} \begin{pmatrix} t \\ 0 \end{pmatrix} = \begin{pmatrix} B^{-1} t - \frac{B^{-1} \mathbf{1} \mathbf{1}^T B^{-1} t}{\mathbf{1}^T B^{-1} \mathbf{1}} \\ \frac{\mathbf{1}^T B^{-1} t}{\mathbf{1}^T B^{-1} \mathbf{1}} \end{pmatrix} . \tag{3.40}$$

The second equality in (3.40) follows by applying lemma 3.2.2. Thus the inversion of matrix A_1 is reduced to the inversion of matrix $B = K + \lambda I$. The matrix $K + \lambda I$ is positive definite for sufficiently large $\lambda > 0$ and hence it is possible to compute a Cholesky decomposition [54] of this matrix. With the Cholesky factors available, it is possible to efficiently solve linear equations involving matrix B for different right hand sides by back-substitution. In the example given by equation (3.40) one solves two systems: $Bu = t$ and $Bw = \mathbf{1}$. Consequently one has the relationships $B^{-1}t = u$ and $B^{-1}\mathbf{1} = w$ which can be substituted into equation (3.40) to yield:

$$A_1^{-1}\begin{pmatrix} t \\ 0 \end{pmatrix} = \begin{pmatrix} u - \frac{w\mathbf{1}^T u}{\mathbf{1}^T w} \\ \frac{\mathbf{1}^T u}{\mathbf{1}^T w} \end{pmatrix} . \tag{3.41}$$

In summary, to compute the next iterate $(\bar{\beta}, \bar{b})$ in the Newton algorithm by equation (3.39), it is necessary to compute the Cholesky decomposition of matrix $K + \lambda I$ with a cost of $\mathcal{O}(n^3), n = |sv_1|$, once and then solve two linear systems by back-substitution with a cost of $\mathcal{O}(n^2)$. Since the cost to evaluate the dot products in (3.41) is just $\mathcal{O}(n)$ the total run time complexity for the Newton step is $\mathcal{O}(n^3)$. A detailed description on how the Newton step is computed is given by algorithm 4 in appendix A.

3.2.3 Line search

The line search finds a point on the line segment between the solution (β_k, b_k) at the k-th iteration and the solution $(\bar{\beta}, \bar{b})$ determined by the Newton step that minimizes the value of the objective function. Stated otherwise, it searches for a step size $\rho \in [0, 1]$ such that the objective function is minimized in dependence of $\beta(\rho) = \beta_k + \rho(\bar{\beta} - \beta_k)$ and $b(\rho) = b_k + \rho(\bar{b} - b_k)$. The objective function (3.28) in dependence of the step size ρ, denoted by $\phi(\rho)$, is a piecewise continuous, quadratic function as shown in figure 3.6. Its derivative $\phi'(\rho)$ is a piecewise linear function that crosses zero at the point where the objective function attains its minimum. Jumps in the second derivative $\phi''(\rho)$ occur at the points ρ_1 to ρ_5 when training patterns enter or leave the set of support vectors.

To determine the optimal step size ρ the exact line search proceeds as follows: In the first step it determines the points $\rho_i \in [0, 1]$, where training patterns enter or leave the set of support vectors, or, equivalently, the corresponding residual enters or leaves the quadratic part of the loss function; In the second step the points ρ_i are sorted in non-decreasing order and the derivative $\phi'(\rho)$ between the pair ρ_i and ρ_{i+1} is considered. As the derivative is a linear function, a potential zero crossing of $\phi'(\rho)$ between ρ_i and ρ_{i+1} can be computed analytically. If no zero crossing is present the objective function is updated, since the support vector set changes after overstepping ρ_{i+1}, and the next pair ρ_{i+1} and ρ_{i+2} is examined.

It is important to note, that it is not sufficient to just determine the critical points ρ_i since it is also necessary to find out whether a training pattern leaves or enters the support

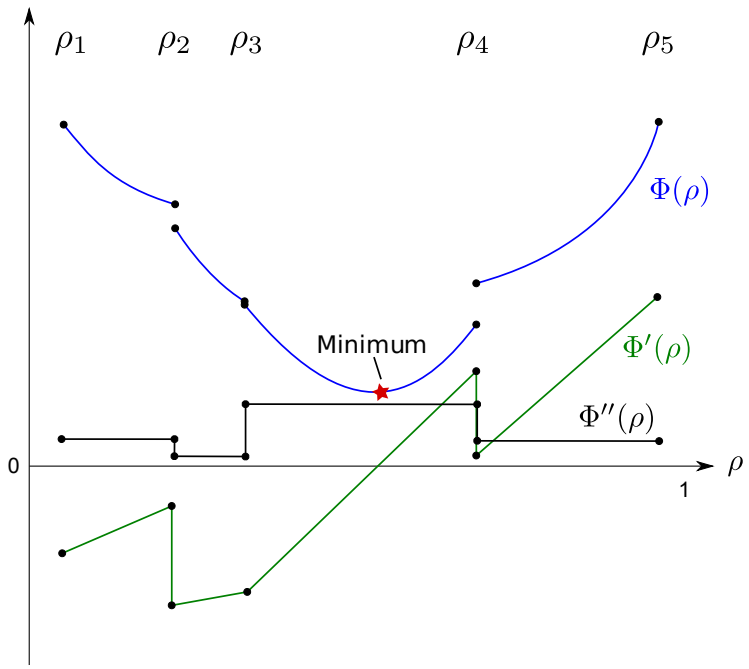


Figure 3.6: *The objective function $\phi(\rho)$ for the primal SVR problem with bias term b in dependence of the step size ρ is a piecewise continuous, quadratic function. The derivative $\phi'(\rho)$ is a piecewise linear, and the second derivative $\phi''(\rho)$ a piecewise constant function. At the points $\rho_1, \ldots, \rho_5$ the residual of a training pattern enters or leaves the quadratic part of the loss function, which causes the jumps in the second derivative $\phi''(\rho)$.*

vector set. Therefore one needs to distinguish the three cases shown in figure 3.7. The residual of the i-th training pattern in dependence of the step size ρ is given by:

$$r_i(\beta(\rho), b(\rho)) = [K\beta_k + b_k - y + \rho(K(\bar{\beta} - \beta_k) + \mathbf{1}(\bar{b} - b_k))]_i = r_i + \rho u_i \ , \tag{3.42}$$

where $r_i = r_i(\beta_k, b_k)$ and $u_i = (K(\bar{\beta} - \beta_k) + \mathbf{1}(\bar{b} - b_k))_i$ are introduced as abbreviations to keep the notation uncluttered. Equation (3.42) shows that the residual is a linear function in dependence of ρ and thus monotonically in- or decreases depending on the sign of u_i. From this one can conclude that the residual can cross the points $\pm\varepsilon$ only once when the step size is gradually increased from zero to one and that the cases shown in figure 3.7 cover all possibilities. The critical points ρ_i for each case can be computed as follows:

Case I: Residuals enter the quadratic part of the loss function from the ε-insensitive zone:

$$\rho_i = \frac{\operatorname{sgn}(u_i)\varepsilon - r_i}{u_i}$$

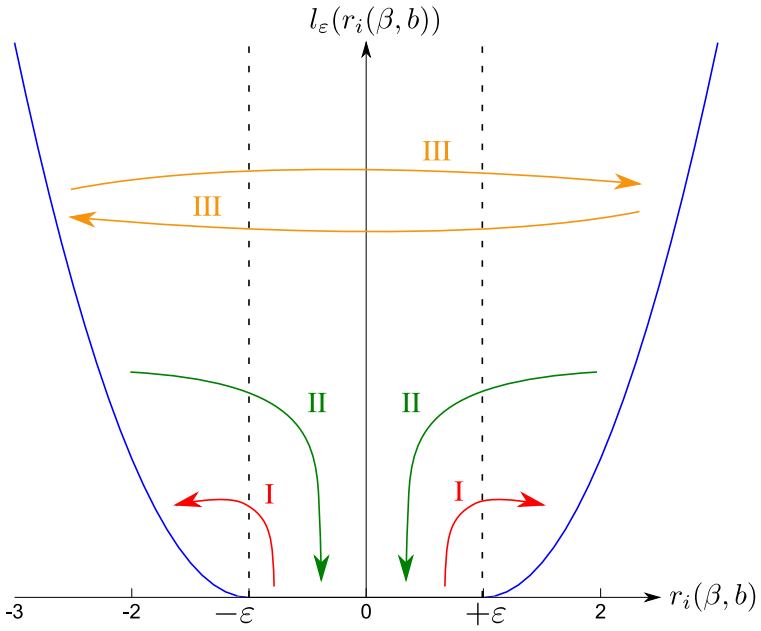


Figure 3.7: *Three cases need to be distinguished for the correct determination of residuals that enter or leave the quadratic part of the loss function. Case I comprises residuals in the ε-insensitive zone that enter the quadratic part. Case II considers residuals leaving the quadratic part and case III deals with residuals that reenter the quadratic part after leaving it.*

Case II: Residuals leave the quadratic part of the loss function:

$$\left.\begin{aligned} r_i > \varepsilon \Rightarrow \rho_i &= \frac{\varepsilon - r_i}{u_i} \\ r_i < -\varepsilon \Rightarrow \rho_i &= \frac{-\varepsilon - r_i}{u_i} \end{aligned}\right\} \quad \Rightarrow \rho_i = \frac{\operatorname{sgn}(r_i)\varepsilon - r_i}{u_i}$$

Case III: Residuals reenter the quadratic part of the loss function:

$$\left.\begin{aligned} r_i > \varepsilon \Rightarrow \rho_i &= \frac{-\varepsilon - r_i}{u_i} \\ r_i < -\varepsilon \Rightarrow \rho_i &= \frac{\varepsilon - r_i}{u_i} \end{aligned}\right\} \quad \Rightarrow \rho_i = \frac{-\operatorname{sgn}(r_i)\varepsilon - r_i}{u_i} \; .$$

When the critical points ρ_i are computed, all points that lie outside the admissible step size interval $(0, 1]$ are discarded. To find the minimizer of the objective function in dependence of ρ an analytical expression for $\phi'(\rho)$ is needed. By substituting $\beta(\rho)$ and $b(\rho)$ into the definition of the objective function (3.28) one obtains:

$$\begin{aligned} \phi(\rho) = {} & r(\beta(\rho), b(\rho))^T W(\beta(\rho), b(\rho)) r(\beta(\rho), b(\rho)) - 2\varepsilon s(\beta(\rho), b(\rho))^T r(\beta(\rho), b(\rho)) \\ & + \varepsilon^T W(\beta(\rho), b(\rho))\varepsilon + \lambda \beta(\rho)^T K \beta(\rho) \; . \end{aligned} \tag{3.43}$$

Differentiating equation (3.43) with respect to ρ then leads to the desired expression:

$$\phi'(\rho) = 2\left[u^T W(\beta(\rho), b(\rho)) r(\beta(\rho), b(\rho)) - \varepsilon s(\beta(\rho), b(\rho))^T u + \lambda(\bar{\beta} - \beta_k)^T K \beta_k\right] . \quad (3.44)$$

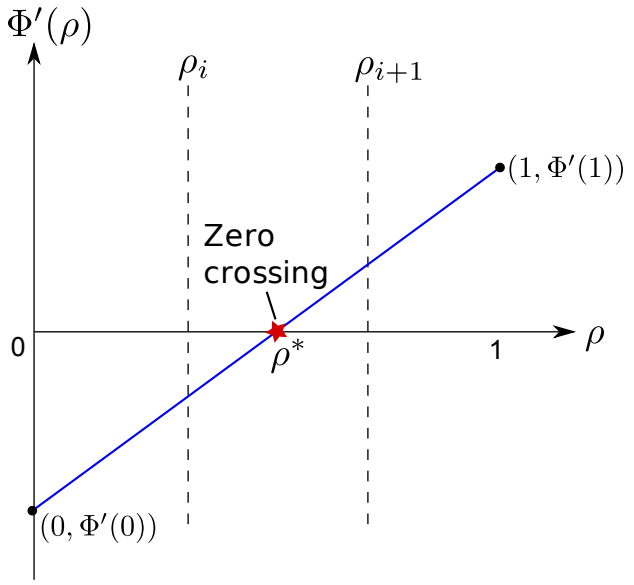


Figure 3.8: *Given a pair of points ρ_i and ρ_{i+1} the zero crossing ρ^* of function $\phi'(\rho)$ is calculated by extending the line segment between ρ_i and ρ_{i+1} to the points $(0, \phi'(0)$ and $(1, \phi'(1))$.*

To find a potential zero crossing between a pair of critical points ρ_i and ρ_{i+1} it is best to extend the line in the $(\phi'(\rho), \rho)$ plane to the points $(0, \phi'(0))$ and $(1, \phi'(1))$, as shown in figure 3.8. The benefit of doing this will become clear shortly. Now the zero crossing ρ^* can be calculated analytically and is given by:

$$l(\rho) = \phi'(0) + (\phi'(1) - \phi'(0))\rho \stackrel{!}{=} 0 \Rightarrow \rho^* = -\frac{\phi'(0)}{(\phi'(1) - \phi'(0))} . \quad (3.45)$$

Consequently only the quantities $\phi'(0)$ and $(\phi'(1) - \phi'(0))$ are needed during the exact line search to identify the point ρ^* that minimizes the objective function. Extending the line segment to zero and one has the advantage that $\phi'(0)$ and $(\phi'(1) - \phi'(0))$ can be updated easily when residuals enter or leave the quadratic part of the loss function. To make this clear $(\phi'(1) - \phi'(0))$ can be deduced from equation (3.44):

$$\begin{aligned}\phi'(1) - \phi'(0) = 2[&u^T(W(\bar{\beta}, \bar{b}) r(\bar{\beta}, \bar{b}) - W(\beta_k, b_k) r(\beta_k, b_k) - \varepsilon s(\bar{\beta}, \bar{b}) + \varepsilon s(\beta_k, b_k)) \\ &+ \lambda(\bar{\beta} - \beta_k)^T K(\bar{\beta} - \beta_k)] .\end{aligned} \quad (3.46)$$

This expression can be further simplified by using the relations $s(\bar{\beta}, \bar{b}) = W(\bar{\beta}, \bar{b})\mathrm{sgn}(r(\bar{\beta}, \bar{b}))$ and $s(\beta_k, b_k) = W(\beta_k, b_k)\mathrm{sgn}(r(\beta_k, b_k))$. In addition one can exploit that the set of support vectors does not change between the pair ρ_i and ρ_{i+1}, that is $W(\beta_k, b_k) = W(\bar{\beta}, \bar{b})$, and (3.46) can be rewritten as:

$$\begin{aligned}\phi'(1) - \phi'(0) = 2[&u^T W(\beta_k, b_k)(u - \varepsilon(\mathrm{sgn}(r(\bar{\beta}, \bar{b})) - \mathrm{sgn}(r(\beta_k, b_k)))) \\ &+ \lambda(\bar{\beta} - \beta_k)^T K(\bar{\beta} - \beta_k)] .\end{aligned} \quad (3.47)$$

From equation (3.47) it can be seen that after entry of the i-th residual into the quadratic part of the loss function the quantity $(\phi'(1) - \phi'(0))$ can be updated by adding the term $2u_i(u_i - \varepsilon(\operatorname{sgn}(\bar{r}_i) - \operatorname{sgn}(r_i)))$. Analogous this term has to be subtracted when the residual leaves the quadratic part. Similar considerations are valid for updating $\phi'(0)$.

Up to this point the line search follows the description given in [74] for SVC and was only extended to the regression case. Unfortunately this description of the line search fails to find the optimal step size ρ^* in some cases. Figure 3.9 shows one case where it is impossible to analytically determine the minimum of the objective function $\phi(\rho)$ by locating the zero crossing of the first derivative $\phi'(\rho)$. Of course one could devise a heuristic to choose an arbitrary point in the interval $[0, 1]$ in this case. But this only works when the descent direction is computed by the Newton step. If the negative gradient, or scaled gradient are used as alternative descent directions, as described in section 4.3.2 for the online algorithm, the heuristic choice of a step size can incur large prediction errors or can even lead to divergence of the algorithm. Clearly, when the minimum cannot be found

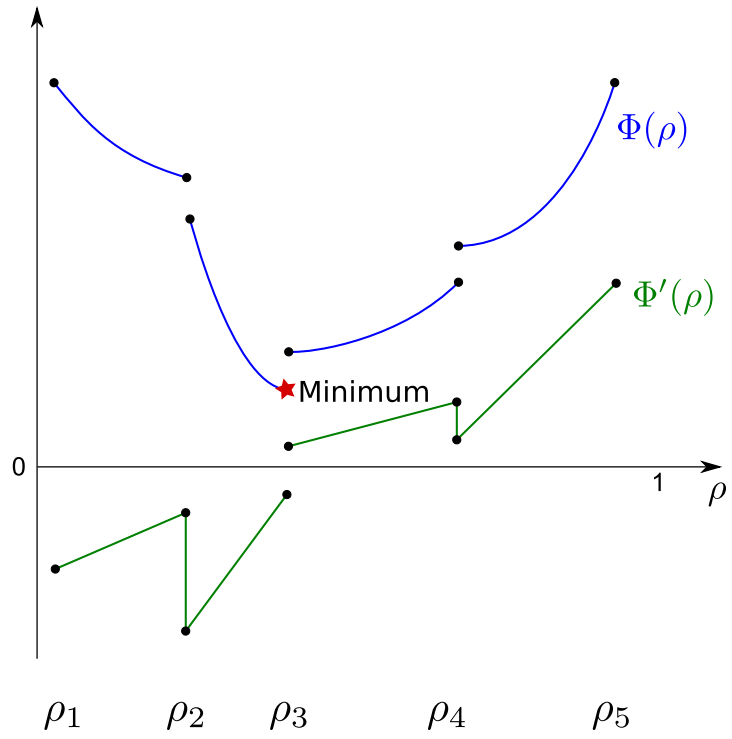


Figure 3.9: *Since the objective function $\phi(\rho)$ is a piecewise continuous, quadratic function there are cases when it is not possible to determine the minimum ρ^* analytically. In these cases the minimum is located at one of the points $\rho_1, \ldots, \rho_5$ and can be found by evaluating the objective function at each of these points.*

analytically the optimal step size ρ^* is located at one of the points ρ_i that are considered during the line search. To reliably find the minimizer of $\phi(\rho)$ it is therefore necessary to evaluate the objective function at all of these points and subsequently choose the point with the minimal function value, if the line search cannot find an analytical solution. Without this modification, sorting the points ρ_i is the most expensive step of the line search with a run time complexity of $\mathcal{O}(n \log n)$. One evaluation of the objective function costs $\mathcal{O}(n^2)$ operations which leads to an overall run time complexity of $\mathcal{O}(n^3)$ for the modified line

search. Since the cost for the Cholesky decomposition is also $\mathcal{O}(n^3)$ the robust version of the line search proposed here does not increase the overall run time complexity of the primal SVR algorithm. In appendix A the steps of the modified line search are summarized in algorithm 3.

3.2.4 Primal algorithm

The Newton step described in section 3.2.1 and the line search described in section 3.2.3 allow the optimization algorithm to make progress towards an optimal solution. The only missing parts to assemble a full optimization algorithm for the SVR problem with bias term are the initialization step and the termination criteria. To get a feasible initial guess of the solution one can set the coefficients β to zero and the bias term b to the mean of the target values that are in the training data set. After this, it is possible to determine the residuals and an initial guess for the set of support vectors. The optimization loop can be terminated when the set of support vectors does not change after one full Newton step with step size $\rho = 1$. Due to the problem discussed in section 3.2.1, the optimization loop is also terminated when the new set of support vectors is empty. From a numerical viewpoint it is practical to finish the optimization when the decrease in the objective function value is smaller than the machine precision. A practical implementation will therefore exit the optimization loop when one of these three termination criteria is met.

At the start of the optimization algorithm the initial guess of the set of support vectors can be far from the final solution. In particular there could be many training patterns in the support vector set that will not contribute to the final solution of the SVR problem. From this perspective the matrix A_1 that needs to be inverted in the Newton step can be huge which is detrimental to the run time of the algorithm due to the $\mathcal{O}(n^3), n = |sv_1|$ complexity of the matrix inversion step. To circumvent this problem [29] proposes to solve the optimization problem for primal SVC recursively, by first computing the solution on a subset of the training patterns to get a better initial estimate for the support vector set. This approach can also be used for the primal SVR problem.

Algorithm 2 in appendix A summarizes all the steps that are needed for the solution of the primal SVR problem with bias term. It uses the results from section 3.2.1 and 3.2.3 together with the practical considerations discussed in this section.

3.3 Results

How does the different behavior of the l_1- and l_2-loss function with respect to outliers (cf. section 3.1.3) influence the performance of the SVR algorithm on real data sets? Are there any differences between the primal and dual solution? These questions are investigated in subsection 3.3.1 and 3.3.2 by comparing the prediction accuracy of the SVR algorithm on data sets from different application areas.

Especially important are the four data sets **fb081008-r1**, **fb141008-r2**, **fb151008-r2**, and **180708-r1** which originate from the adaptive microstimulation experiments described in chapter 6. Each pattern in these data sets consists of recorded pre- and post-stimulus local field potentials and the regression target corresponds to the applied stimulus intensity. Detailed descriptions of all data sets used in this study are given in appendix B.

3.3.1 Comparison of l_1- and l_2-loss functions

For regression problems the performance is usually measured by the mean squared error (MSE) between predicted target values $f(x_i)$ and the true targets y_i on an independent test data set:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 \ . \tag{3.48}$$

The performance measure defined by equation (3.48) is a summary statistic that depends on the partitioning of the original data into training and test subsets, and can thus be regarded as a random variable with unknown distribution. The subsequent results therefore include 95% confidence intervals calculated by a bootstrap procedure [41, 47] with $B = 1000$ bootstrap samples alongside the point estimate of the MSE.

Data set	Dual l_1	Dual l_2	Primal l_2	Primal l_2 w/o b	Train size
abalone	(3, -2, -7)	(2, -2, -3)	(3, -2, -7)	(3, -2, -7)	500
cadata	(5, -1, -8)	(6, -2, -4)	(4, -1, -8)	(4, -1, -8)	500
cpusmall	(6, -2, -8)	(1, -1, -8)	(5, -2, -8)	(5, -2, -8)	500
fb081008-r1	(5, 0, -5)	(7, -2, -3)	(5, 0, -5)	(5, 0, -5)	300
fb141008-r2	(7, 0, -6)	(5, 0, -3)	(6, 0, -7)	(6, 0, -6)	300
fb151008-r2	(3, 2, -8)	(0, 2, -8)	(2, 2, -8)	(5, 1, -8)	300
fb180708-r1	(7, 0, -4)	(3, 1, -3)	(6, 0, -4)	(6, 0, -4)	300
housing	(4, -2, -8)	(2, -2, -3)	(4, -2, -8)	(4, -2, -8)	400
mpg	(2, -1, -8)	(0, -1, -3)	(2, -1, -7)	(3, -2, -8)	300
pyrim	(8, -2, -5)	(2, -2, -5)	(8, -2, -5)	(2, -3, -4)	55
space-ga	(8, -3, -8)	(8, -4, -4)	(7, -3, -8)	(7, -3, -8)	500
triazines	(5, -4, -3)	(2, -3, -3)	(4, -4, -3)	(4, -4, -3)	140

Table 3.1: *Values* $(\log(C), \log(\gamma), \log(\varepsilon))$ *for the regularization parameter* C*, RBF kernel parameter* γ*, and loss function parameter* ε *selected by 10-fold cross validation on the training data set. Here the* $\log(.)$*-function has base* e *and the last column shows the size of the training data set.*

The regularization parameter C, the width of the radial basis function (RBF) kernel γ [124], and the size of the insensitive zone in the loss function ε were chosen by 10-fold cross validation on the training data set. This selection of hyper-parameters was done once before calculating the bootstrap estimate of the MSE to avoid impractical run times. The

values of the hyper-parameters employed for the different data sets and algorithms, as well as the size of the training data set, are listed in table 3.1.

Figure 3.10 compares the prediction accuracy in terms of the MSE for the dual SVR algorithm with l_1- and l_2-loss functions on 12 different data sets. The dots symbolize the point estimates of the MSE and associated 95% confidence intervals are represented by colored bars. It can be seen that on most data sets the l_1- and l_2-loss perform equally well. The l_1-loss is slightly better on the **mpg** data set while the l_2-loss is slightly better on the **cadata** and **fb-081008-r1** data set. But these differences are not statistically significant at the .05 level as indicated by the overlapping 95% confidence intervals. The confidence intervals are largest for the **pyrim** and **triazines** data sets due to the small total number of patterns.

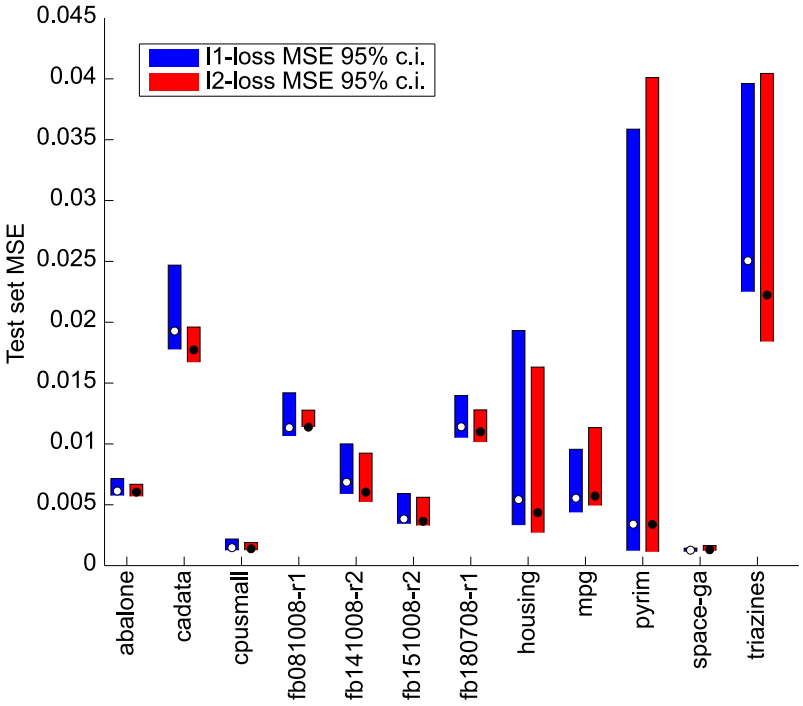


Figure 3.10: *Comparison of the dual SVR algorithm with l_1- and l_2-loss functions. The performance is evaluated by point estimates of the MSE (dots) and associated 95% confidence intervals (bars). Although the l_1-loss performs slightly better on the **mpg** data set, while the l_2-loss is better on the **cadata** and **fb-081008-r1** data set, these differences are not significant at the .05 level.*

The results shown in figure 3.10 are surprising at first sight, since one would expect the l_1-loss function to perform better than the l_2-loss function. But it is important to recall that the example in section 3.1.3 was deliberately chosen to show the superiority of the l_1-loss in the presence of outliers. In most practical applications such extreme outliers occur only rarely or are easily removed by appropriate preprocessing of the input patterns. Furthermore the target values in many application areas are measured by devices that add noise to the targets which is governed by a Gaussian distribution. Under these circumstances the l_2-loss function is suitable for measuring the loss between predicted and true target values [29]. It should be noted that in case of adaptive microstimulation the choice of the l_2-loss function is not at all detrimental to prediction accuracy, as indicated by the results on the data sets **fb081008-r1**, **fb141008-r2**, **fb151008-r2**, and **180708-r1**.

3.3.2 Comparison of primal and dual algorithms

The performance of dual SVR with bias from section 3.1.1, primal SVR with bias from section 3.1.3, and primal SVR without bias from section 3.2 is compared by the MSE and 95% confidence intervals on 12 different data sets as described in the previous section. A summary of the results obtained with the different SVR algorithms is shown in figure 3.11. There are only small differences in the performance across all data sets. Remarkably the primal SVR algorithm without bias term seems to perform better on the **pyrim** data set. But, as pointed out before, the difference is not statistically significant at the .05 level and the confidence intervals are large due to the low number of patterns in this data set.

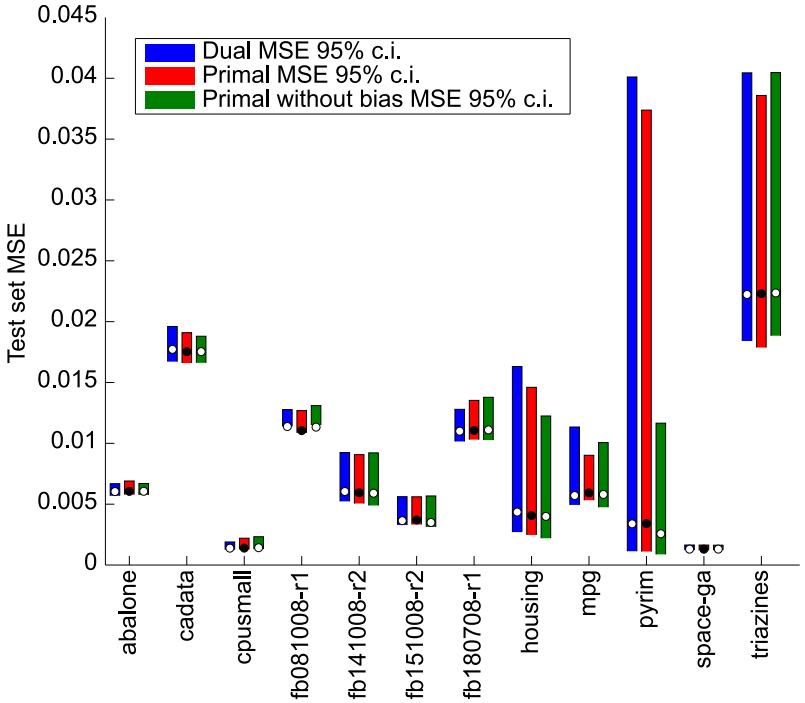


Figure 3.11: *Comparison of the primal and dual SVR with bias term and the primal SVR without bias term. Point estimates and associated 95% confidence intervals for the MSE are shown for 12 data sets from different application areas. Although there are small differences in the performance of the three algorithms, none is statistically significant at the .05 level.*

Section 3.1.2 described the difference between primal and dual optimization in the context of the RLS algorithm. Why does one not observe any difference in the performance of primal and dual SVR in figure 3.11? The reason is, that the results shown in figure 3.11 represent the exact solutions of the regression problem and according to primal and dual optimization theory these solutions have to be equal. In other words there are no differences since one does not consider approximate solutions to the regression problem. Nevertheless one can observe very small dissimilarities in the performance of the primal and dual SVR algorithm, but these can be attributed to the different optimization algorithms that are employed: SMO for solution of the dual [27, 107], and Newton's algorithm for the primal.

It is not knowledge, but the act of learning,
not possession, but the act of getting there,
which grants the greatest enjoyment.

Carl Friedrich Gauss (1777 - 1855)

4 Online SVR

The problem of inferring optimal stimulation parameters based on the background brain activity is associated with many uncertainties that are caused by the inherent complexity of the nervous system. Chapter 3 discussed how the problem of adaptive stimulation is reducible to a regression problem that can be solved by SVR. The discussion implicitly assumed that a full set of training patterns originating from a stationary distribution is provided for training the SVR algorithm. This assumption is highly questionable in the context of adaptive stimulation. For a prosthetic device, like a visual cortical prosthesis as described in section 2.3.2, it is unrealistic to acquire a complete set of training patterns, since it would drastically limit its practical use. After turning on a cortical prosthesis there will be a rather short calibration phase that allows collecting a small set of training patterns. More important, the function estimated by SVR on the training patterns might already be outdated when it is used to adapt the stimulation parameters. Such a situation could arise if there is a temporal drift or switch in the distribution of the training patterns or a temporally changing functional relationship between training patterns and targets. At the moment these considerations seem to be quite theoretical, but the results in section 7.3 of chapter 7 indicate, that there are temporal differences in the performance of adaptive stimulation. Although the origin of these differences could not be clarified yet, they could be explained by either a temporally changing distribution of training patterns or a temporally changing functional relationship.

4.1 Online versus Offline SVR training

In the 'offline' or 'batch' setting that is described in chapter 3 all data is available for training the SVR algorithm and the estimated function is held fixed for computing predictions on unseen data. This chapter will be concerned with the 'online' setting, where the current estimate of the function is continuously updated upon arrival of a new pair (x_t, y_t) of training pattern and target at time point t. At first glance online algorithms seem to be inferior to offline algorithms. On the one hand they do not have access to the full data set and cannot anticipate the properties of the next training pattern, on the other hand online algorithms are usually restricted to store only a small subset of recently

seen training patterns and have only a fixed time window to optimize the current solution. Surprisingly it is possible to construct examples where online algorithms actually perform better than their offline counterparts. In the first example the training patterns $x_t \sim \mathcal{N}(0.5, 0.1)$ are normally distributed with mean 0.5 and standard deviation 0.1. The functional relationship between the patterns x_t and the targets y_t is defined as follows:

$$y_t = \begin{cases} x_t^2, & kb < t < (k+1)b, \text{ even } k \\ x_t, & \text{otherwise} , \end{cases} \tag{4.1}$$

where b denotes the block size. According to this definition the targets y_t are either a linear or quadratic function of the training patterns x_t and functions are switched temporally in block wise fashion. Figure 4.1A shows the results obtained on this example data set by offline training with the primal SVR algorithm described in section 3.2 and by online training with the primal online algorithm introduced in section 4.3. The performance measure used here is the current average error which at iteration t corresponds to the mean squared error on the training patterns seen so far:

$$\text{Current average error} = 1/t \sum_{i=0}^{t} (y_i - f(x_i))^2 . \tag{4.2}$$

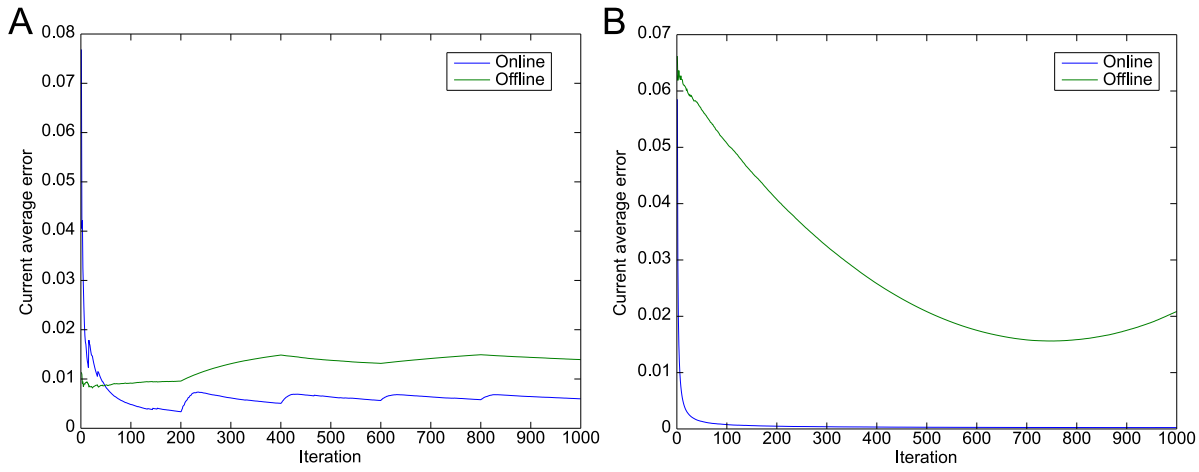


Figure 4.1: *Online versus offline SVR training.* ***A:*** *The function relating training patterns and targets switches block wise every 200 iterations between a linear and quadratic function. Online training outperforms offline training in this case, since the online algorithm is able to track the switching functional relationship by rapidly adjusting its current solution.* ***B:*** *The function relating training patterns and targets continuously drifts with time. In this setting online training incurs smaller prediction errors compared to offline training because this kind of drift is hard to capture by the single function computed by the offline algorithm.*

For this example the regularization and loss function parameters were set to $\lambda = 1/2$ and $\varepsilon = 0.01$. It can be seen that during the initial 50 iterations the current average error of the online algorithm is higher than the error of the offline algorithm. This is the expected behavior, since the online algorithm could not yet identify the functional relationship. After this initial phase the predictions made by the online algorithm are more accurate than those of the offline algorithm. The block size in equation (4.1) was chosen to be $b = 200$ which can also be seen in figure 4.1A as the error of the online algorithm slightly increases after each switch of the underlying functional relationship.

For the second example shown in figure 4.1B the training patterns $x_t \sim \mathcal{N}(0.5, 0.01)$ again follow a normal distribution with mean 0.5 and standard deviation 0.01 but the relationship to the targets, given by $y_t = x_t^2 + t/(2m)$, where m is the total number of patterns, drifts with time. In this scenario the difference between the prediction error of the online and offline algorithm is even more pronounced. Intuitively this can be expected since a drifting functional relationship is hard to explain by a single fixed function that is computed by the offline algorithm.

Of course these examples were contrived to show the superiority of online training and one cannot use these examples to draw general conclusions about the relative merits of online training. But it is nevertheless interesting to analyze why the offline algorithm failed to capture the relevant relationship in these examples. The explanation for this failure is simple: In both examples the functional relationship was deliberately chosen to have a temporal dependence that is implicitly encoded by the order of the training patterns. For the offline algorithm the order of the training patterns does not matter since the solution it produces is invariant to reordering the training patterns. Consequently the information about the temporal dependence of the function to be learnt gets lost in the offline setting.

The perceptron learning algorithm is one of the first online training algorithms [114]. Its successor, the back propagation algorithm for training perceptrons with multiple layers of neurons, was also first formulated for the online setting and then extended to the batch setting [117]. At that time personal computers or workstations still had severe limitations in terms of memory capacity and CPU speed, and online training algorithms were even employed in presence of a complete data set. Compared to this historic development of training algorithms in the field of neural networks, the development of SVM training algorithms developed in the opposite direction. From the beginning [142] SVMs were formulated in the batch setting and subsequently proposed solution approaches all concentrated on this setting [107, 27].

4.2 Online training state of the art

This chapter will be concerned with online training algorithms for SVMs, or, more precisely, online algorithms to solve the SVR problem. Current state of the art algorithms for online SVM training can be roughly divided into three groups based on their solution approach.

In the first group algorithms use stochastic gradient descent to minimize the objective function [78, 31, 143] and thus maintain an approximate solution to the learning problem in every iteration step. The stochastic gradient descent framework for online training was first introduced by the naive online risk minimization algorithm (NORMA) [78], which was later combined with a sophisticated step size adaptation method [143]. NORMA will be described in more detail in section 4.2.1, while the step size adaptation will not be considered any further in this chapter because it is hard to implement and a less complicated implicit update rule was reported to perform better in practice [31] . The sparse implicit learning with kernels (SILK) will be described in section 4.2.2.

In contrast to the first group, algorithms from the second group maintain an exact solution to the learning problem in every iteration step [23, 88]. This means that after arrival of the next training pattern the current solution is incrementally updated, if necessary, until it is optimal for all training patterns seen so far. To avoid the problems associated with a fixed function in the offline setting it is possible to remove old training patterns from the current data set and incrementally update the solution. These incremental learning algorithms have been proposed for both, classification [23] and regression [88]. Although the notion of maintaining an exact solution in every iteration step is attractive it requires an unbounded number of operations to be executed upon arrival of the next training pattern. In other words it is impossible to restrict the time needed per iteration for incremental learning, which prevents the use of this approach in a real-time environment. As explained later in section 7, the real-time system used for adaptive stimulation puts hard constraints on the available iteration time.

The drawback of an unbounded run time per iteration is overcome by algorithms in the third group. The LaSVM [13] algorithm employs two steps per iteration which both need a fixed number of operations. The 'process' step looks at the next training pattern and adds it to the support vectors if necessary. In the 'reprocess' step the coefficients of two patterns already residing in the support vector set are optimized and can lead to their removal. Together these steps can be seen as a reorganization of the steps performed by the SMO algorithm [107] for batch training [13]. This means that LaSVM maintains an approximate dual solution to the SVM learning problem. In light of the discussion given in section 3.1.2, concerning the comparison of approximate primal and dual solutions, it is problematic to compute an approximate dual solution and the LaSVM algorithm will not be considered any further in this chapter.

4.2.1 Naive online risk minimization

Online training can be formally seen as the problem of estimating a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ based on a sequence $\mathcal{S} = ((x_1, y_1), \ldots, (x_m, y_m))$ of examples $(x_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$ that arrive at time point t. In the naive online risk minimization algorithm (NORMA) [78] the unknown

function f is found by minimizing the instantaneous regularized risk:

$$R_{inst,\lambda}(f, x_t, y_t) = \frac{\lambda}{2}\|f\|^2_{\mathcal{H}} + l(f(x_t), y_t) \ , \tag{4.3}$$

by using stochastic gradient descent. The definition of the risk functional in (4.3) is almost identical to the definition of the primal objective function in equation (3.18). But here the loss function is evaluated only on the next training pattern (x_t, y_t), while equation (3.18) contains the average loss across all training patterns in the data set. Similar to the offline setting the solution is regularized by the first term in equation (4.3) and parameter $\lambda \in \mathbb{R}^+$ is used to trade off between the complexity of function f and the incurred loss on pattern (x_t, y_t). By computing the partial derivative of equation (4.3) with respect to f one obtains the following general update rule

$$f_{t+1} = f_t - \eta_t \partial_f R_{inst,\lambda}(f, x_t, y_t)|_{f=f_t} \ , \tag{4.4}$$

that relates the estimate f_t of the function at the last iteration to the new estimate f_{t+1}. Equation (4.4) just expresses a gradient descent step with learning rate $\eta_t \in \mathbb{R}^+$ in the space of admissible functions f. Now, it is assumed that the space of functions is a kernel Hilbert space $\mathcal{H}$ with the reproducing property $f_t(x_t) = \langle f, k(., x_t)\rangle_{\mathcal{H}}$, introduced in chapter 3. Using the reproducing property one can calculate the partial derivative of the loss function with respect to f:

$$\partial_f l(f_t(x_t), y_t) = \partial_f l(\langle f_t, k(., x_t)\rangle_{\mathcal{H}}, y_t) = l'(f_t(x_t), y_t)k(., x_t) \ , \tag{4.5}$$

with $l'(z, y) = \partial_z l(z, y)$. Thus the loss function has to be differentiable in its first argument. Since $\partial_f \lambda/2\|f\|^2_{\mathcal{H}} = \lambda f$, the update rule (4.4) can be rewritten using equation (4.5) :

$$f_{t+1} = f_t - \eta_t \lambda f_t - \eta_t l'(f_t(x_t), y_t)k(x_t, .) = (1 - \eta_t\lambda)f_t - \eta_t l'(f_t(x_t), y_t)k(x_t, .) \ . \tag{4.6}$$

It is clear that the sequence of functions $f_1, f_2, \ldots$ will converge if in the limit of an infinite number of iterations the distance between consecutive elements of the sequence approaches zero. In mathematical terms this means that $\lim_{t\to\infty} \|f_{t+1} - f_t\| \to 0$. Consequently given fixed value of $\lambda > 0$ the learning rate has to fulfill the condition $\eta_t < 1/\lambda \, \forall_t$, in order to ensure the convergence of NORMA. This directly follows from the right hand side of equation (4.6).

The representer theorem given by equation (3.19) states that every function $f \in \mathcal{H}$ can be expanded as a linear combination of kernel functions that are centered on the training patterns. Using the representer theorem the functions f_{t+1} and f_t in equation (4.6) can be expanded as follows:

$$\sum_{i=1}^{t-1} \alpha_i k(x_i, x) + \alpha_t k(x_t, x) = \sum_{i=1}^{t-1} (1 - \eta_t\lambda)\alpha_i k(x_i, x) - \eta_t l'(f_t(x_t), y_t)k(x_t, x) \ . \tag{4.7}$$

By comparing the left and right hand sides of equation (4.7) it is possible to read off the general update rules for the expansion coefficients given by:

$$\alpha_t \leftarrow -\eta_t l'(f_t(x_t), y_t), \qquad \alpha_i \leftarrow (1 - \eta_t\lambda)\alpha_i \ \forall_{i<t} \ . \tag{4.8}$$

Until now the derivation of NORMA was independent of the specific choice of a loss function. With the general update rules for the coefficients in equation (4.8) it is possible to solve both, classification and regression problems online by choosing an appropriate loss function. Since this chapter is concerned with online training algorithms for SVR, the following description will focus on the ε-insensitive quadratic, or l_2 loss, that is shown in figure 3.5D of chapter 3. To repeat, the l_2 loss is defined by $l(f(x), y) = \max\{0, 1/2(|y_i - f(x_i)| - \varepsilon)\}^2$ and its derivative with respect to function f can be written as:

$$\begin{aligned} l'(f(x), y) &= \begin{cases} f(x) - y - \varepsilon, & f(x) - y > \varepsilon \\ f(x) - y + \varepsilon, & f(x) - y < -\varepsilon \\ 0, & \text{otherwise} \end{cases} \\ &= \begin{cases} f(x) - y - \operatorname{sgn}(f(x) - y)\varepsilon, & |f(x) - y| > \varepsilon \\ 0, & \text{otherwise} . \end{cases} \end{aligned} \tag{4.9}$$

Inserting equation (4.9) into equation (4.8) yields the update rules for the expansion coefficients in the case of online SVR:

$$\begin{aligned} \alpha_t &\leftarrow \begin{cases} -\eta_t[f_t(x_t) - y_t - \operatorname{sgn}(f_t(x_t) - y_t)\varepsilon], & |f_t(x_t) - y_t| > \varepsilon \\ 0, & \text{otherwise} \end{cases} \\ \alpha_i &\leftarrow (1 - \eta_t\lambda)\alpha_i \; \forall_{i<t} . \end{aligned} \tag{4.10}$$

The estimated function f did not yet involve a bias term $b \in \mathbb{R}$. It is straightforward to extend NORMA and let it optimize the function $g(x) = f(x) + b, \; f \in \mathcal{H}$ with an additional bias term. Optimization of $g(x)$ can be decomposed into separately optimizing over f and b. The minimization of the instantaneous regularized risk in equation (4.3) with respect to f can be carried out as already described. One just has to replace $f_t(x_t)$ by $g_t(x_t) = f_t(x_t) + b_t$ in the update equation (4.10) for the coefficients α_i. The gradient descent step to minimize the risk functional with respect to the bias term is given by:

$$b_{t+1} = b_t - \eta_t \partial_b R_{inst,\lambda}(g, x_t, y_t)|_{g=f_t+b_t} . \tag{4.11}$$

Since the bias term is not regularized, the partial derivative of the risk functional in equation (4.11) reduces to the partial derivative of the l_2 loss. This partial derivative is given by equation (4.9) when $f_t(x_t)$ is replaced by $g_t(x_t) = f_t(x_t) + b_t$. The update rule for the bias term b can then be deduced from equation (4.11):

$$b_{t+1} \leftarrow \begin{cases} b_t - \eta_t[g_t(x_t) - y_t - \operatorname{sgn}(g_t(x_t) - y_t)\varepsilon], & |g_t(x_t) - y_t| > \varepsilon \\ b_t, & \text{otherwise} \end{cases} . \tag{4.12}$$

4.2.2 Implicit online learning with kernels

When an online algorithm updates its current solution to a learning problem it has to balance two needs to perform well. On the one side it has to retain the information it

extracted about the learning problem from previously seen input patterns. This means that the algorithm should be conservative and preserve the current solution. On the other side it has to make more accurate predictions if the same pattern is observed again. This means that the algorithm should be corrective and update the current solution. These ideas are encapsulated in the general framework for online algorithms proposed in [79]. The sparse implicit learning with kernels (SILK) algorithm [31] uses this framework to derive implicit update rules for online SVM training. The estimate of function f_{t+1} at the next iteration is obtained from:

$$f_{t+1} = \arg\min_f \frac{1}{2}\|f - f_t\|_{\mathcal{H}}^2 + \eta_t \left(\frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + l(f(x_t), y_t) \right) . \tag{4.13}$$

Minimizing the first term in equation (4.13) prevents large deviations of the new estimate f_{t+1} from the current solution f_t and helps the algorithm to be conservative. The second term in equation (4.13) corresponds to the regularized risk functional introduced in equation (4.3) and helps the algorithm to be corrective. The simultaneous minimization of both terms balances between the needs to be conservative and corrective. By increasing the learning rate $\eta_t \in \mathbb{R}^+$ it is possible to emphasize the corrective aspect of the algorithm. Differentiation of equation (4.13) with respect to f yields:

$$\begin{aligned} & f - f_t + \eta_t \lambda f + \eta_t \partial_f l(f(x_t), y_t) = (1 + \eta_t \lambda) f - f_t + \eta_t \partial_f l(f(x_t), y_t) \stackrel{!}{=} 0 \\ \Rightarrow & f = \frac{1}{(1 + \eta_t \lambda)} f_t - \frac{\eta_t}{(1 + \eta_t \lambda)} \partial_f l(f(x_t), y_t) . \end{aligned} \tag{4.14}$$

With $\tau_t = \frac{\eta_t \lambda}{(1+\eta_t \lambda)}$ the right hand side of equation (4.14) can be written in a more compact way:

$$f_{t+1} = (1 - \tau_t) f_t - (1 - \tau_t) \eta_t \partial_f l(f_{t+1}(x_t), y_t) . \tag{4.15}$$

At this point it is assumed that the function f resides in a reproducing kernel Hilbert space (RKHS). This is similar to the assumption made in the description of NORMA in section 4.2.1. Now it is further assumed that the gradient of the loss function is an element of the RKHS. According to the representer theorem (3.19) the gradient can be expanded as:

$$\partial_f l(f_{t+1}(x_t), y_t) = l'(f_{t+1}(x_t), y_t) k(x_t, .) = \beta_t k(x_t, .) , \tag{4.16}$$

and hence $\beta_t = l'(f_{t+1}(x_t), y_t)$. Using this relationship and expanding the functions f_{t+1} and f_t as linear combinations of kernel functions equation (4.15) can be reformulated:

$$\sum_{i=1}^{t-1} \alpha_i k(x_i, .) + \alpha_t k(x_t, .) = \sum_{i=1}^{t-1} (1 - \tau_t) \alpha_i k(x_i, .) - (1 - \tau_t) \eta_t \beta_t k(x_t, .) . \tag{4.17}$$

Here it is again possible to directly read off the general update rules for the expansion coefficients by comparing the left and right hand sides of equation (4.17):

$$\alpha_t \leftarrow -(1 - \tau_t) \eta_t \beta_t, \qquad \alpha_i \leftarrow (1 - \tau_t) \alpha_i \; \forall_{i<t} . \tag{4.18}$$

It is important to point out that the update rules for NORMA (4.8) and SILK (4.18) are almost identical. The difference in the update rule for coefficient α_t lies in the derivative of the loss function, which is $l'(f_t(x_t), y_t)$ for NORMA and $l'(f_{t+1}(x_t), y_t)$ for SILK.

The description of SILK is so far valid for any differentiable loss function. In the following the l_2 loss function will be used to derive an online training algorithm for the SVR problem. To replace the expression f_{t+1} in the derivative of the loss function $l'(f_{t+1}(x_t), y_t)$ one can use the following relationship that can be gleaned from the left hand side of equation (4.17):

$$f_{t+1}(x_t) = (1 - \tau_t) f_t(x_t) + \alpha_t k(x_t, x_t). \tag{4.19}$$

For the l_2 loss the derivative is given in equation (4.9) and it is thus necessary to distinguish three different cases. In the first case $f_{t+1}(x_t, y_t) > \varepsilon$ and by using (4.19) the derivative of the l_2 loss can be expanded as:

$$\beta_t = f_{t+1}(x_t) - y_t - \varepsilon = (1 - \tau_t) f_t(x_t) + \alpha_t k(x_t, x_t) - (y_t + \varepsilon) \tag{4.20}$$

Using the update rule (4.18) for coefficient α_t and substituting β_t by the right hand side of (4.20) it is possible to derive a closed form expression for α_t:

$$\begin{aligned} \alpha_t &= -(1 - \tau_t)\eta_t \beta_t = -(1 - \tau_t)^2 \eta_t f_t(x_t) - (1 - \tau_t)\eta_t \alpha_t k(x_t, x_t) + (1 - \tau_t)\eta_t (y_t + \varepsilon) \\ \Rightarrow \alpha_t &= \frac{(1 - \tau_t)\eta_t \left(y_t + \varepsilon - (1 - \tau_t) f_t(x_t)\right)}{\left(1 + (1 - \tau_t)\eta_t k(x_t, x_t)\right)} \, . \end{aligned} \tag{4.21}$$

Although α_t is now computable from expressions known in the previous iteration at time point t it is still necessary to decide when the update rule is applicable. In other words it is necessary to answer the question when $f_{t+1}(x_t) - y_t$ is larger than ε. With the help of equation (4.19) this can be reduced to a condition on α_t:

$$\alpha_t > \frac{\varepsilon - (1 - \tau_t) f_t(x_t) + y_t}{k(x_t, x_t)} \, . \tag{4.22}$$

Finally, by combining equations (4.21) and (4.22), a simplified condition not involving α_t is deducible:

$$(1 - \tau_t)\eta_t f_t(x_t) - y_t > \varepsilon \, . \tag{4.23}$$

All derivations given for the first case also hold in the second case of the l_2 loss, when $f_{t+1}(x_t, y_t) < -\varepsilon$. It is only necessary to change the sign of ε and reverse the inequalities. In the third case one has $\beta_t = 0$ and equation (4.18) implies that $\alpha_t = 0$. To sum up the update equations for the coefficients α_i for the l_2 loss are:

$$\begin{aligned} &\alpha_i \leftarrow (1 - \tau_t)\alpha_i \; \forall_{i<t} \\ &\alpha_t \leftarrow \begin{cases} \frac{(1-\tau_t)\eta_t(\operatorname{sgn}(\nu)\varepsilon - \nu)}{(1+(1-\tau_t)\eta_t k(x_t, x_t))}, & |\nu| > \varepsilon \\ 0, & \text{otherwise} \, , \end{cases} \end{aligned} \tag{4.24}$$

where $\nu = (1 - \tau_t)\eta_t f_t(x_t) - y_t$.

There still remains the open question on how to choose an appropriate learning rate η_t for the NORMA and SILK algorithm. The convergence analysis of NORMA indicates [78] that the learning rate should decay with an increasing number of iterations according to the schedule $\eta_t = \eta_0/\sqrt{t}$. Unfortunately this does not give any hint on how to select the initial learning rate η_0 in practice. It is therefore mandatory to tune the initial learning rate η_0 for each application separately in order to achieve good prediction performance with both NORMA and SILK [78, 31]. This limitation of NORMA and SILK will be overcome by the new online algorithm proposed in section 4.3.

4.3 Primal online algorithm

The algorithm developed in section 3 solves the primal SVR problem in the offline setting. It uses Newton's method with an exact line search to minimize the objective function of the primal SVR problem in equation (3.24) iteratively. If the algorithm is terminated prematurely, for example after one Newton step, the resulting solution, namely the coefficients β and the bias term b, is an approximation to the primal SVR problem. Since the value of the objective function is decreased by each Newton step the approximate solution is closer to the optimum than the initial guess for β and b. In this way the algorithm produces a series of functions $f_1, f_2, \ldots, f_n$ that converges to the optimal function f_n after n iterations.

In the online setting algorithms receive an infinite sequence $\mathcal{S} = ((x_1, y_1), (x_2, y_2), \ldots)$ of examples. The next pattern in the sequence (x_t, y_t) arriving at time point t provides additional information about the learning problem and is used by the online algorithm to update its current solution. Under the assumption that the patterns are drawn form a stationary distribution there will be a time point, say t^*, where the next pattern in the sequence does not contribute any additional information about the learning problem. Seen from a different point of view a finite sample of training patterns, say $\mathcal{S}^* \subset \mathcal{S} = ((x_1, y_1), \ldots, (x_{t^*}, y_{t^*}))$, will be sufficient to describe the underlying distribution, at least to a level of precision that is required for the accurate prediction on the next pattern in the sequence.

The primal online algorithm (PRIONA) proposed here uses the Newton step from section 3.2.1 and the exact line search from section 3.2.3 in each iteration to minimize the primal SVR problem defined by the subset of patterns it already received[19]. Thus, PRIONA generates a series of functions $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_t$, which of course is different from the series of functions produced in the offline setting. But if the patterns are drawn from a stationary distribution the offline primal algorithm operating on the subset $\mathcal{S}^*$ and PRIONA receiving the sequence $\mathcal{S}$ will converge to the same optimal solution, that is $\hat{f}_t \to f_n$ for $t \to \infty$. There are two reasons for this: First, the sequence $\mathcal{S}$ can be adequately described by the subset $\mathcal{S}^*$, as argued in the last paragraph; Second, PRIONA executes an optimization step at each time point that is equivalent to one iteration of the offline primal algorithm.

Actually the informal argument given above holds for any offline algorithm that is applied in the online setting. This line of thought is also followed by the LaSVM algorithm [13], for example. But the offline counterpart in the LaSVM algorithm finds an optimal solution to the dual SVM problem. As shown in section 3.1.2 approximate solutions for the dual optimization problem can be slow to converge to the optimum of the primal problem. Therefore it can be expected that an online algorithm that uses approximations to the primal optimization problem, like PRIONA, yields a higher speed of convergence.

To recapitulate, the essential idea of PRIONA lies in applying the optimization steps of the offline primal algorithm described in section 3.2 in an online setting. Although this idea is straightforward, there are several choices that have to be made for a practical implementation of PRIONA [19]. In many problems where online training is applicable strict time constraints are imposed on one iteration step of the algorithm. When the time per iteration is limited, only a subset of the recently seen training patterns can be buffered and processed to update the current solution. Strategies to remove patterns from the buffer will be discussed in section 4.3.1 and evaluated on several data sets in section 4.4.2. While restricting the number of stored training patterns is one way to diminish the iteration time, reducing the computational complexity of each iteration step is another possibility. Replacements for the Newton step will be described in section 4.3.2 and empirically analyzed in section 4.4.3. Most of the time in the Newton step is consumed by inverting the hessian matrix. This can be avoided if the inverse is updated incrementally after arrival of the next training pattern, a possibility discussed in section 4.3.3. Since this approach proves to be difficult for the SVR loss function, section 4.3.4 introduces an algorithm for online kernel ridge regression that can fully exploit the advantages of incrementally updating the hessian inverse.

4.3.1 Buffering strategies

Most online training algorithms for SVMs [78, 143] use the "first in first out" (FIFO) principle to manage the patterns in the buffer (figure 4.2A). This strategy is easy to implement, but is suboptimal from a theoretical point of view. In online classification problems for example, it makes more sense to remove patterns that are far away from the decision boundary [35], since their removal has the least impact on the current solution. Later this strategy was refined to remove those patterns that have the lowest classification error on a subset of the training data and it was shown that this approach performs better on noisy data sets [146].

For regression problems there is no decision boundary but the concept is transferable by removing patterns with the smallest absolute value of the residual $r_i = K_i\beta_i + b - y_i$. If $|r_i|$ is smaller than the loss function parameter ε the pattern does not contribute to the current solution and can be safely removed. Alternatively one can base the decision for removing a pattern on the absolute value of the corresponding coefficient β_i. The different buffering strategies that are applicable to online SVR training are illustrated in figure 4.2.

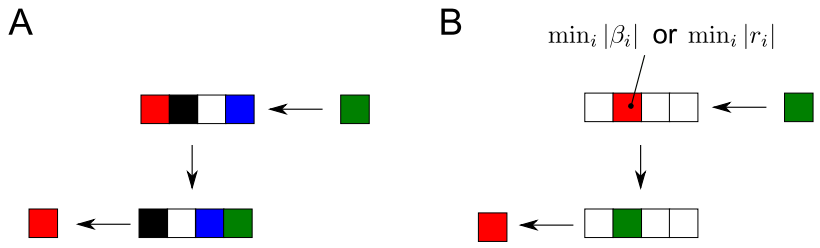


Figure 4.2: ***A:*** *FIFO. The oldest pattern is removed.* ***B:*** *The pattern with minimum absolute value of coefficient β_i or residual r_i is removed.*

4.3.2 Descent directions

Due to the inversion of the hessian matrix, the Newton step has a run time complexity of $\mathcal{O}(n^3)$, where n denotes the number of support vectors. But there are other possibilities to determine the descent direction and the values of the optimization variables $(\bar{\beta}, \bar{b})$ at the next optimization step.

It is well known that the gradient of a function points in the direction of its steepest increase. The negative of the gradient therefore is a valid descent direction as illustrated in figure 4.3. For the primal SVR problem the gradient is given by equation (3.35) and the new values of the optimization variables in case of the l_2 loss can be determined via:

$$\begin{pmatrix} \bar{\beta} \\ \bar{b} \end{pmatrix} = \begin{pmatrix} \beta \\ b \end{pmatrix} - \nabla L_\varepsilon(\beta, b) = \begin{pmatrix} \beta \\ b \end{pmatrix} - 2 \begin{pmatrix} K^T W(\beta, b) r(\beta, b) - \varepsilon s(\beta, b)^T K + \lambda K \beta \\ \mathbf{1}^T W(\beta, b) r(\beta, b) - \varepsilon s(\beta, b)^T \mathbf{1} \end{pmatrix} . \quad (4.25)$$

Assuming that the matrix vector product $K\beta$ is cached and just updated every iteration the run time complexity to compute the gradient step in equation (4.25) is only $\mathcal{O}(m \cdot n)$, where m denotes the number of patterns in the input buffer. In most applications the support vectors completely fill the input buffer and $n = m$. For these cases the gradient step has a run time complexity of $\mathcal{O}(n^2)$. Despite the fact that the negative gradient points in the direction of steepest descent this does not necessarily mean that it points towards the minimum of the function, as can be seen in figure 4.3B.

Let's suppose for the moment that the function f to be minimized is a quadratic function. Then the hessian of f is a diagonal matrix that can be inverted in linear time. Around a certain neighborhood of a point many functions are approximated well by a quadratic function. Thus, it makes sense to pre-multiply the negative gradient by the inverse of the diagonal portion of the hessian. This scaled gradient descent direction for the primal SVR problem is given by:

$$\begin{pmatrix} \bar{\beta} \\ \bar{b} \end{pmatrix} = \begin{pmatrix} \beta \\ b \end{pmatrix} - D \nabla L_\varepsilon(\beta, b) \ , \quad (4.26)$$

In equation 4.26 D is the diagonal scaling matrix with the i-th entry given by:

$$D_{ii} = \begin{bmatrix} (K^T W(\beta, b) + \lambda I) K & K W(\beta, b) \mathbf{1} \\ \mathbf{1}^T W(\beta, b) K & \mathbf{1} W(\beta, b) \mathbf{1} \end{bmatrix}^{-1}_{ii} \Rightarrow D_{ii} = \begin{cases} (K_{i,sv}^T K_{i,sv} + \lambda K_{ii})^{-1}, & 1 \leq i \leq m \\ 1/n, & i = m+1 \end{cases} . \quad (4.27)$$

Equation (4.27) uses $K_{i,sv}$ to denote the entries support vectors in the i-th column of the kernel matrix K. The complexity to determine matrix D and its inverse is $\mathcal{O}(n \cdot m)$ and, under the same assumptions made for the gradient direction, the overall run time complexity is $\mathcal{O}(n^2)$. For non-quadratic functions the scaled gradient and Newton directions are different (figure 4.3A), while for quadratic functions they coincide (figure 4.3B).

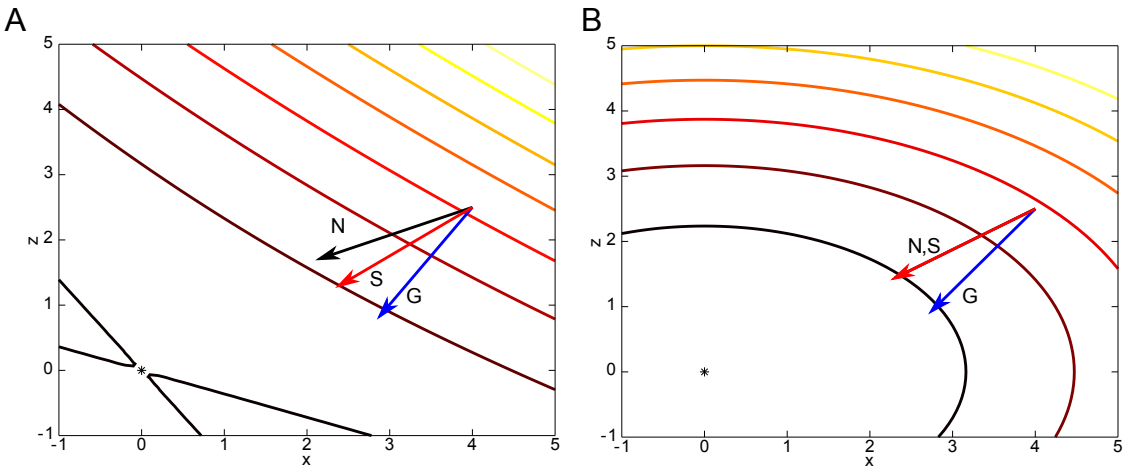


Figure 4.3: *Newton (N), gradient (G), and scaled gradient (S) descent directions for function* $f(x, z) = x^2 + 2z^2 + cxz$ *at point* $(4, 2.5)$. *For illustration purposes the descent directions have been normalized to length two.* ***A:*** *For* $c = 3.5$ *the function has a saddle point at* $(0, 0)$ *marked by the asterisk (*). All descent directions are different in this case. The gradient (G) is perpendicular to the contour lines of the function.* ***B:*** *The hessian of* f *is diagonal for* $c = 0$ *and the Newton and scaled gradient descent directions are identical. The asterisk (*) marks the minimum of* f.

In summary the gradient and scaled gradient step are feasible replacements for the Newton step in the PRIONA algorithm and incur a lower run time overhead per iteration. It is worthwhile to point out that the line search introduced in section 3.2.3 does not need to be modified to work with these alternative descent directions.

4.3.3 Incremental updates

By incrementally updating the inverse of the hessian matrix it is possible to shorten the run time of an online iteration without sacrificing the properties of the Newton step. The idea of applying incremental updates to an inverted matrix upon arrival of the next training patterns has been previously employed for incremental learning of SVC [23] and SVR [88] as well as for online learning with Gaussian processes [36]. In all cases the incremental matrix updates are derived from the Sherman-Morrison-Woodbury formula [54]:

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1},\ A \in \mathbb{R}^{m \times m},\ U, V \in \mathbb{R}^{m \times k}\ . \quad (4.28)$$

Equation (4.28) essentially states that the inverse of an $m \times m$ matrix $(A + UV^T)$ after a low rank update can be determined by computing the inverse of a smaller $k \times k$ matrix $(I+V^T A^{-1}U)$. This obviously results in substantial savings of computation time, if $k \ll m$. How can this be exploited in an online iteration of PRIONA? For the Newton step it is necessary to invert the matrix $K_{sv,sv} + \lambda I$. When the next pattern in the sequence replaces a support vector one has to replace the corresponding row and column in matrix $K_{sv,sv}$. The same is true when the input buffer is not yet full and the matrix has to be extended. In general it is therefore interesting to study how the inverse of a matrix can be updated with formula (4.28) after replacing one row and column.

So lets consider an exchange of the i-th row and column given by block entries $(a_1\, a_2\, a_3)$ in a symmetric matrix $A \in \mathbb{R}^{m\times m}$ with the new row and column vector $(b_1\, b_2\, b_3)$. The updated matrix A' can then be expressed as a low rank update of A as follows:

$$A' = A + \begin{bmatrix} 0 & b_1 - a_1 & 0 \\ b_1 - a_1 & b_2 - a_2 & b_3 - a_3 \\ 0 & b_3 - a_3 & 0 \end{bmatrix} = A + \begin{pmatrix} 0 & c_1 \\ 1 & 0 \\ 0 & c_3 \end{pmatrix} \begin{pmatrix} c_1 & c_2 & c_3 \\ 0 & 1 & 0 \end{pmatrix} = A + uv^T\,, \quad (4.29)$$

with $c_j = b_j - a_j$. In the next step the right hand side of (4.29) can be expanded with the help of the Sherman-Morrison-Woodbury formula:

$$(A')^{-1} = (A + uv^T)^{-1} = A^{-1} - A^{-1}u(I + v^T A^{-1}u)^{-1}v^T A^{-1}\,. \quad (4.30)$$

The expression $(I+v^T A^{-1}u)^{-1}$ on the right hand side of (4.30) is the inverse of a 2×2 matrix and directly computable in closed form. Together with the other matrix multiplications the evaluation of equation (4.30) thus requires $\mathcal{O}(m^2)$ operations.

Initially it seems that incremental updates applied to the inverse of the hessian matrix can lower the complexity of the Newton step, but this notion ignores the fact that in each online iteration more than one pattern can enter or leave the set of support vectors. In the worst case scenario all of the patterns in the input buffer leave the support vector set upon arrival of the next pattern and the incremental update of the inverse requires m exchanges of a row and column vector. Although it is unlikely that this will frequently happen in practice, the run time to update the inverted hessian is $\mathcal{O}(m^3)$ in the worst case. This is even worse than the $\mathcal{O}(n^2)$ operations needed for inverting the hessian afresh in each iteration step.

However, thinking about incremental updates for online training is not in vain. The next question that comes to mind is how changes in the support vector set can be avoided. The set of support vectors is naturally not under direct control of the algorithm and the easiest way to avoid changes is by having no support vectors at all. From the discussion given in chapter 3 it is clear that support vectors emerge by choosing a loss function with ε-insensitive zone. By replacing the l_2 loss with a quadratic loss function one can therefore get rid of the support vectors. The resulting algorithm obviously does not solve the SVR problem anymore, but instead can be regarded as an online algorithm for kernel ridge regression.

4.3.4 Online kernel ridge regression

After replacing the l_2 loss function with a quadratic loss the objective function for the primal SVR problem in equation (3.24) is:

$$\min_{\beta,b} L(\beta, b) = \lambda \beta^T K \beta + \sum_{i=1}^{m} (y_i - K_i \beta + b) \,. \tag{4.31}$$

Again, the first term in equation (4.31) is responsible for regularizing the solution and the second term minimizes the loss over all training patterns. Differentiation of equation (4.31) with respect to the optimization variables (β, b) yields the gradient given by:

$$\nabla L(\beta, b) = 2 \begin{bmatrix} K & 0 \\ \mathbf{1}^T & -\lambda \end{bmatrix} \begin{pmatrix} r(\beta, b) + \lambda \beta \\ \mathbf{1}^T \beta \end{pmatrix} = 2 \begin{bmatrix} K & 0 \\ \mathbf{1}^T & -\lambda \end{bmatrix} \left(\begin{bmatrix} K + \lambda I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \begin{pmatrix} \beta \\ b \end{pmatrix} - \begin{pmatrix} y \\ 0 \end{pmatrix} \right) . \tag{4.32}$$

The second partial derivatives lead to the hessian matrix that can be factorized as follows:

$$\nabla^2 L(\beta, b) = 2 \begin{bmatrix} K & 0 \\ \mathbf{1}^T & -\lambda \end{bmatrix} \begin{bmatrix} K + \lambda I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} . \tag{4.33}$$

The Newton step, which is the outcome of multiplying the gradient in equation (4.32) by the inverse of equation (4.33), is then given by:

$$\begin{pmatrix} \bar{\beta} \\ \bar{b} \end{pmatrix} = -(\nabla^2 L(\beta, b))^{-1} \nabla L(\beta, b) = \begin{bmatrix} K + \lambda I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}^{-1} \begin{pmatrix} y \\ 0 \end{pmatrix} - \begin{pmatrix} \beta \\ b \end{pmatrix} . \tag{4.34}$$

At this point it is important to note that due to the choice of the loss function the inverse of the hessian in equation (4.34) always depends on all training patterns and not just a subset of patterns as in SVR. This implies that during online training the hessian inverse can be updated efficiently in $\mathcal{O}(m^2)$ time by employing the technique of incremental updates described in section 4.3.3.

Once more a line search is used to minimize the objective function by finding a point on the line segment between (β_k, b_k) and $(\bar{\beta}, \bar{b})$. While the line search for primal SVR training is intricate and requires tracking of the support vector set, the line search for online kernel ridge regression has a closed form solution. With the definition of $\beta(\rho) = \beta_k + \rho(\bar{\beta} - \beta_k)$ and $b(\rho) = b_k + \rho(\bar{b} - b_k)$ the objective function (4.31) in dependence of the step size ρ is:

$$\phi(\rho) = \lambda \beta(\rho)^T K \beta(\rho) + r(\beta(\rho), b(\rho))^T r(\beta(\rho), b(\rho)) \,, \tag{4.35}$$

and its derivative:

$$\phi'(\rho) = 2[\lambda(\bar{\beta} - \beta_k)^T K \beta(\rho) + (K(\bar{\beta} - \beta_k) + \mathbf{1}(\bar{b} - b_k))^T r(\beta(\rho), b(\rho))] \,. \tag{4.36}$$

By setting derivative $\phi'(\rho)$ in equation (4.36) to zero one can explicitly solve for step size

$$\rho = -\frac{\lambda(\bar{\beta} - \beta_k)^T K \beta_k + u^T r}{\lambda(\bar{\beta} - \beta_k)^T K (\bar{\beta} - \beta_k) + u^T u} \,. \tag{4.37}$$

Here $u = K(\bar{\beta} - \beta_k) + \mathbf{1}(\bar{b} - b_k)$ and $r = K\beta_k + \mathbf{1}b_k - y$ were introduced as abbreviations to simplify the formula. It can be verified that the ρ given in equation (4.37) is a minimizer of the function $\phi(\rho)$ by computing the second derivative:

$$\phi''(\rho) = 2[\lambda(\bar{\beta} - \beta_k)^T K(\bar{\beta} - \beta_k) + u^T u] > 0\,. \tag{4.38}$$

The positivity in equation (4.38) follows from the positive definiteness of the kernel matrix K. This section showed how a simple exchange of the loss function in the primal SVR optimization problem leads to an online algorithm for kernel ridge regression. Originally kernel ridge regression was derived via the dual optimization problem in [120].

4.4 Results

This section presents results to answer the various questions raised in the preceding subsections. In particular, the benefits of using a bias term in online training algorithms is discussed in section 4.4.1. Further, sections 4.4.2 and 4.4.3 explore the best buffering strategy and descent direction to be used in conjunction with the PRIONA algorithm. And finally section 4.4.4 compares the performance of all online algorithms described in this chapter on several benchmark data sets. The data sets are identical to those used in chapter 3 for the comparison of primal and dual SVR training and are described in detail in appendix B. Performance differences between the different online algorithms are especially important on the four data sets **fb081008-r1**, **fb141008-r2**, **fb151008-r2**, and **180708-r1** originating from the adaptive microstimulation experiments described in chapter 6.

In all of the presented results the median of the squared prediction errors is used as a performance measure. It is important to note that in online training the median is a more adequate measure than the mean, since the prediction errors do not follow a normal distribution. The reason for this are the presence of very large errors during the initial iterations of an online algorithm. At that point in time training patterns are scarce and little information is available about the regression problem. The point estimates for the median squared errors are complemented by 95% confidence intervals determined via the bootstrap method [41] with $B = 1000$ bootstrap samples. The RBF kernel is used for SVR training on all data sets with regularization, kernel, and loss function parameters set to the values given in table 3.1.

4.4.1 Online training with and without bias

In the online setting one can imagine situations where training with a bias term can be advantageous. One example is data, where the target values undergo large shifts frequently. When training with bias term, such shifts can be compensated for in the next iteration by a single change in the bias parameter, while training without offset might require several iterations to cope with this situation. But how often does this situation occur in practice?

Figure 4.4A shows the results of the NORMA algorithm with and without bias term on different data sets. The bars represent the 95% confidence intervals for the median of the squared prediction errors and the asterisks (*) indicate significant results, as assessed by a Wilcoxon rank sum test at the $\alpha = 0.05$ level. On 50% of the data sets NORMA training with bias term leads to significantly better results than training without bias term. But there are two data sets, namely **cadata** and **mpg**, where the opposite statement is true, and training without bias term is the better choice.

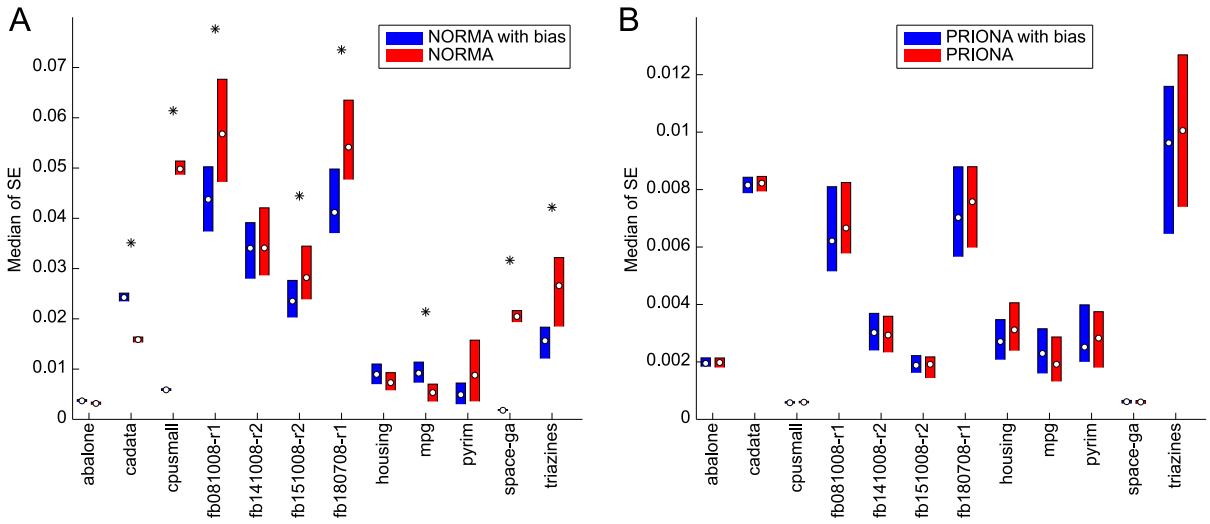


Figure 4.4: *Online training with and without bias term.* ***A:*** *The* NORMA *algorithm performs significantly better with bias term on half of the benchmark data sets. On the* ***cadata*** *and* ***mpg*** *data sets* NORMA *training without bias term yields better results.* ***B:*** *There is no significant performance difference for the* PRIONA *algorithm with and without bias term on all the data sets.*

Although training NORMA with bias yields better results on half of the data sets this observation does not hold for the PRIONA algorithm as shown in figure 4.4B. The inclusion of the bias term does not show any significant performance improvement on all of the data sets. This result is astonishing first, but one might argue that PRIONA can compensate more easily for possible shifts in the target values since it can update several coefficients β_i in each iteration step. In contrast, NORMA changes only a single coefficient α_t upon arrival of the next pattern and lets all other coefficients decay according to equation (4.10). As a consequence adjustment of an additional bias term is more beneficial for NORMA on data sets with shifting target values.

4.4.2 Comparison of buffering strategies

The time needed by an online algorithm in each iteration step can be controlled by restricting the number of buffered training patterns. After the buffer is completely filled one element has to be removed when the next training pattern arrives. In section 4.3.1 it was proposed to either remove the oldest pattern (FIFO), the pattern with minimum absolute value for the coefficient β_i, or the pattern with the minimum absolute value of the residual. Figure 4.5 shows the results of PRIONA in conjunction with different strategies for pattern removal.

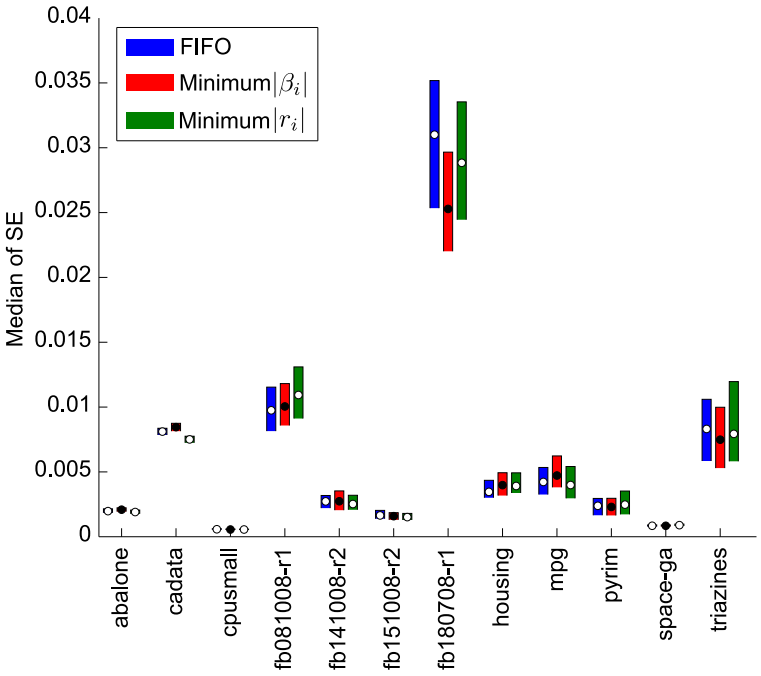


Figure 4.5: *The performance of* PRIONA *with different buffering strategies. On most data sets the buffering strategy does not have a significant influence on the performance of the algorithm, as judged by the overlapping 95% confidence intervals for the estimated median of the residuals. Only for* ***cadata*** *removing the point with the minimum residual from the buffer works slightly better than the other strategies.*

The results in figure 4.5 suggest that selection of a particular buffering strategy is not critical for obtaining good prediction performance with the PRIONA algorithm. On all data sets, except **cadata**, there are no significant performance differences between the buffering strategies, and even for **cadata** removing the pattern with minimum absolute value of the residual works only slightly better than the alternative removal strategies.

Similar to the results for the PRIONA algorithm the buffering strategy has only a small influence on the prediction accuracy of the NORMA and SILK algorithm (figure 4.6). The most notable difference is on the **mpg** data set for the NORMA algorithm where removal of the pattern with minimum absolute coefficient works best. This strategy also performs significantly better for NORMA on the **cadata** and **cpusmall** data sets although the performance difference is very small.

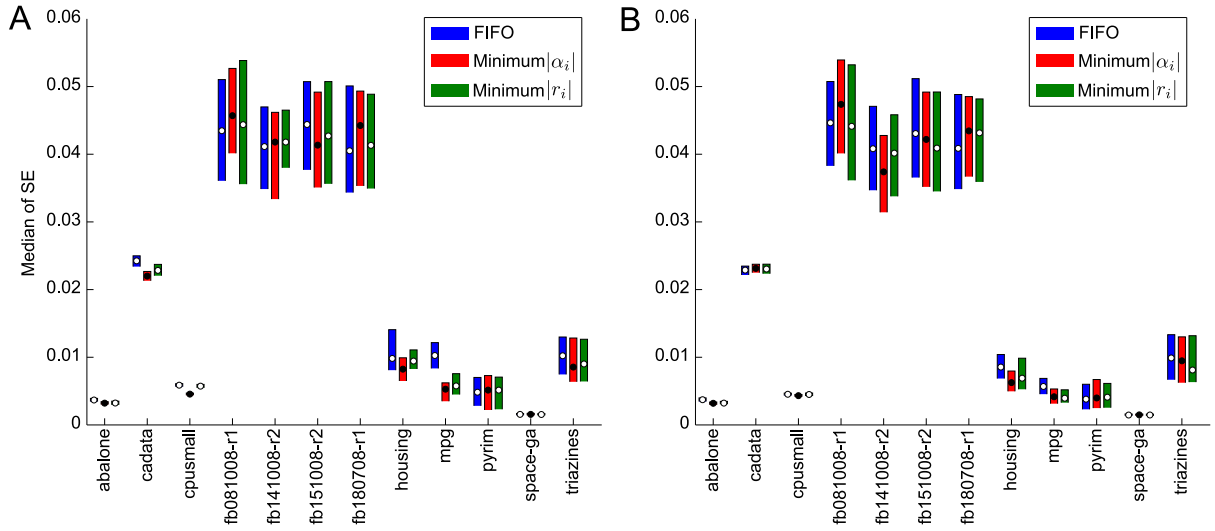


Figure 4.6: *The performance of* Norma *and* Silk *with different buffering strategies.* ***A:*** *For* Norma *significant performance differences are only present on the* ***cadata****,* ***cpusmall*** *and* ***mpg*** *data sets. On these three data sets removal of the pattern with minimum absolute coefficient value yields the best results. Yet, with exception of the* ***mpg*** *data set, the differences in prediction accuracy is very small.* ***B:*** *For* Silk *all buffering strategies perform equally well.*

4.4.3 Comparison of descent directions

The next point on the path towards the optimal solution is found in Priona by first computing the descent direction and then conducting a line search along this direction. As described in section 4.3.2 feasible descent directions are given by the Newton step, the steepest descent step along the negative gradient, or the diagonally scaled steepest descent step. The theoretical run time complexity for the Newton step is $\mathcal{O}(n^3)$, while computation of the gradient, and diagonally scaled gradient has a run time complexity of $\mathcal{O}(n^2)$.

Figure 4.7 shows the empirical run times[1)] for computing the different descent directions in dependence of the input buffer size. All measurements were made on the **abalone** data set. If there are at most 100 elements in the buffer the empirical run time to compute all descent directions is virtually the same. As expected, computation of the Newton step requires more time than computation of the other descent directions at larger buffer sizes. But with increasing buffer size the time needed by the Newton step grows more slowly than predicted by the worst case cubic bound, which is represented by the dotted line in figure 4.7. For the diagonally scaled gradient direction the theoretical $\mathcal{O}(n^2)$ scaling coincides well with the empirical run time measurements.

[1)] Measurements were made on a dual core AMD® Opteron™ 275 processor with 2.2GHz and 1GB of main memory under Matlab®.

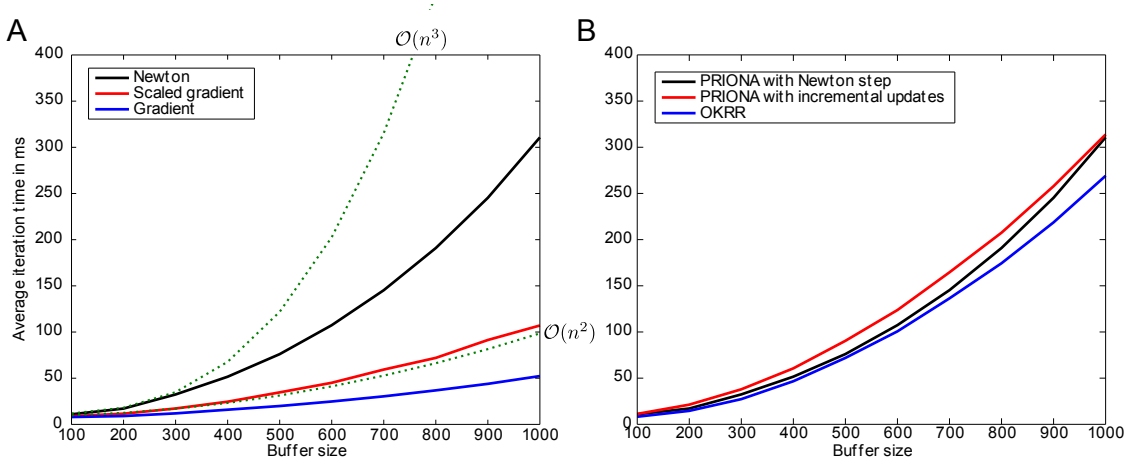


Figure 4.7: *Dependence of average iteration time in milliseconds on the size of the input buffer for the* **abalone** *data set.* ***A:*** *For different descent directions. The dotted curves illustrate cubic and quadratic run time complexities and were fitted to the first two iteration times measured for the Newton and scaled gradient directions respectively.* ***B:*** *Newton step compared to incremental updates and online kernel ridge regression.*

In section 4.3.3 incremental updates to the inverse hessian matrix were introduced and are expected to scale better than the full inversion of the hessian matrix when not all support vectors change after arrival of the next training pattern. Unfortunately this expectation is not fulfilled in practice, at least on the **abalone** data set, as shown in figure 4.7B. Here incremental updates have larger empirical run times than the Newton step, if the buffer contains between 100 and 900 elements. For this reason PRIONA with incremental updates will not be considered any further in the ensuing discussion. Nevertheless incremental updates are beneficial to use in conjunction with the online kernel ridge regression (OKRR) algorithm introduced in section 4.3.4. It can be seen in figure 4.7B that on average OKRR requires less time per iteration than PRIONA.

How do the different descent directions influence the precision of the PRIONA algorithm? Figure 4.8 shows the performance of PRIONA in combination with the Newton step, diagonally scaled gradient, and gradient descent directions. It is of course unfair to directly compare the descent directions since the Newton step requires more time per iteration. Therefore the buffer was restricted to contain only 64 elements for the Newton step. The markers in the upper part of figure 4.8 represent estimates of the average iteration time in milliseconds[2)] and the black bars[3)] indicate the associated 95% confidence intervals.

In the upper part of figure 4.8 it can be seen that the average iteration time for the Newton

[2)] Measurements were made on an AMD® ATHLON™ 64 processor with 2.2GHz and 2GB of main memory under MATLAB®.

[3)] Barely visible, since confidence intervals are smaller than symbols representing the average iteration time.

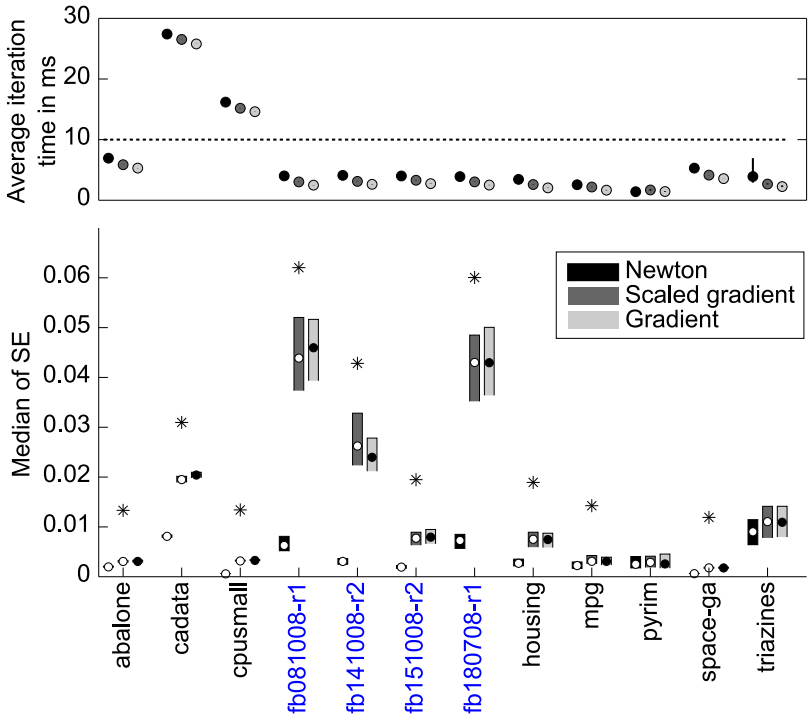


Figure 4.8: *The performance of* Priona *using different descent directions. On ten of the data sets, computing the Newton step leads to significantly better results in comparison to the gradient and diagonally scaled gradient descent directions. Here the buffer was restricted to contain only 64 elements for* Priona *with the Newton step.*

step is in the range of the iteration times measured for the alternative descent directions. The data sets **fb081008-r1**, **fb141008-r2**, **fb151008-r2**, and **180708-r1**, highlighted in blue, stem from the adaptive stimulation experiments, where new patterns are acquired at a rate of 100Hz. This forces the iteration times of online algorithms to be shorter than 10ms. As the results in figure 4.8 show, this temporal constraint is fulfilled by Priona with the Newton step, if the input buffer is restricted to contain less than 64 patterns.

Besides considerations concerning the iteration time, Priona with the Newton step also produces significantly better results on ten out of twelve data sets, as shown in the lower part of figure 4.8. Again asterisks (*) indicate significant results, as assessed by an independent Wilcoxon rank sum test at the $\alpha = 0.05$ level. This suggest that the descent direction in Priona should be determined via the Newton step, as it results in more precise predictions even under strict constraints on the iteration time.

4.4.4 Comparison of online training algorithms

The results presented so far examine the relevance of the bias term and to justify the choice of buffering strategy and descent direction for the Priona algorithm. Now, the online training algorithms Norma, Silk, Priona, and Okrr will be compared in terms of both, iteration time and precision. The initial learning rate of Norma and Silk was tuned individually for each data set across the set $\eta_0 \in \{0.1, 0.2, \ldots, 10\}$. This additional work is superfluous for Priona and Okrr, since the best step size is selected in each iteration by an exact line search.

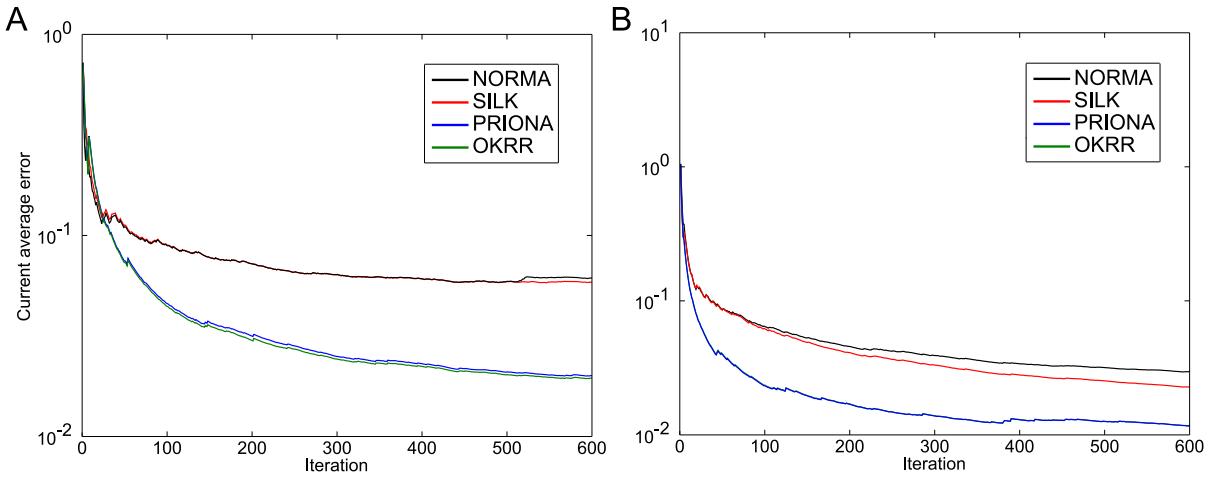


Figure 4.9: *Convergence of online SVR algorithms on two data sets from the adaptive stimulation experiments. The current average error (equation* (4.2)*) is the mean squared error between predicted and target values on the patterns seen so far.* ***A:*** *On the* ***fb180708-r1*** *data set* PRIONA *and* OKRR *have a lower current average error than* NORMA *and* SILK *after iteration 25 and* OKRR *performs slightly better than* PRIONA. ***B:*** *On the* ***fb141008-r1*** *data set* SILK *performs better than* NORMA*, but it does not match the precision of* PRIONA *and* OKRR.

Figure 4.9 shows the convergence of the online algorithms on two data sets from the adaptive stimulation experiments. Here the buffer size of PRIONA and OKRR was restricted to 64 while NORMA and SILK were allowed to store all the patterns. In figure 4.9A NORMA and SILK have the highest convergence speed during the first 25 iterations, but converge to a larger asymptotic error than PRIONA and OKRR subsequently. On the first data set OKRR has a slightly smaller current average error than PRIONA while the results of both algorithms are indistinguishable on the second data set (figure 4.9B). Overall PRIONA and OKRR converge to a lower asymptotic error than NORMA and SILK.

The observations made for the two examples from the adaptive stimulation experiments carry over to the other benchmark data sets, as shown in figure 4.10. Table 4.1 lists the optimal buffer size used for each data set and algorithm and the best initial learning rates η_0 for NORMA and SILK. With these settings PRIONA and OKRR perform better than NORMA and SILK in terms of precision on all but the **pyrim** data set. The asterisks (*) in figure 4.10 indicate where the median squared error of PRIONA is significantly lower than the median squared error of NORMA and SILK. This was assessed by a combination of two independent Wilcoxon rank sum test at the $\alpha = 0.05$ level. Since the precision of OKRR and PRIONA match on all data sets the same assertion also holds for OKRR of course.

At first sight these results suggest that PRIONA and OKRR outperform NORMA and SILK on a broad range of different regression problems, but this analysis neglects the fact that PRIONA and OKRR require more time per iteration on average than NORMA and SILK

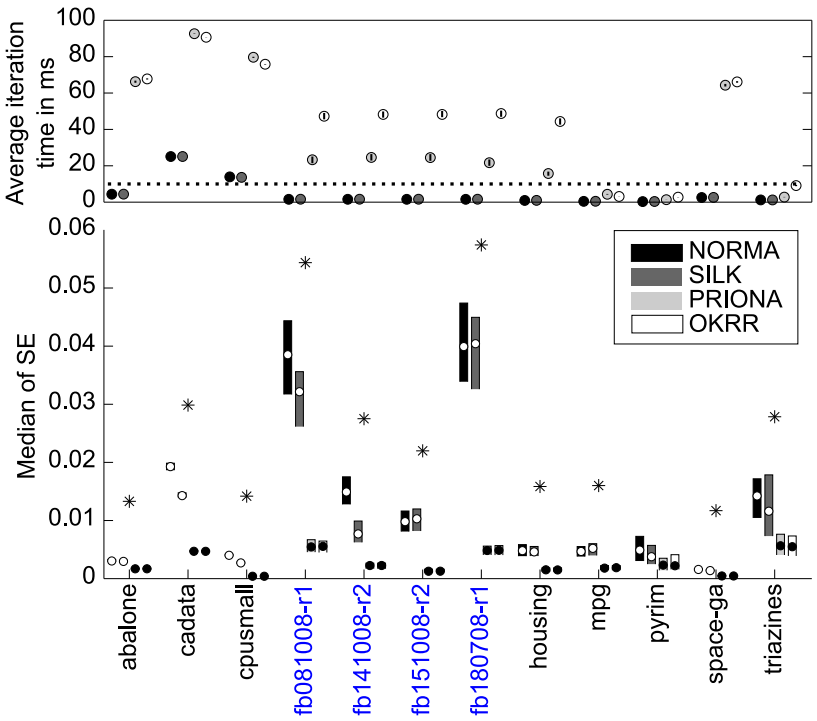


Figure 4.10: *The performance of online training algorithms with optimal buffer size. Asterisks (*) indicate results where the median squared error of* Priona *is significantly lower than both* Norma *and* Silk *as assessed by a combination of two independent Wilcoxon rank sum test at the $\alpha = 0.05$ level. Markers in the upper part of the plot indicate estimates of the average iteration times and black bars represent the associated 95% confidence intervals.*

as shown in the upper part of figure 4.10. Especially for the data sets from the adaptive stimulation experiment the iteration time of Priona and Okrr exceeds the 10ms constraint. Another issue with the current analysis concerns the best initial learning rates for Norma and Silk which are at the upper bound of the tuning interval for the **cadata** and **cpusmall** data sets.

To address these issues, the comparison of online algorithms was repeated under two additional conditions. First, the buffer of Priona and Okrr was limited to contain at most 64 patterns. Second, Norma and Silk were allowed to retain all of the training patterns in the buffer, if this strategy yielded better results. The results obtained under these new conditions are shown in figure 4.11.

With restricted buffer size the iteration time of Priona and Okrr does not exceed the 10ms constraint and is in the same range as the iteration times of Norma and Silk on most data sets (figure 4.11A). In spite of this restriction Priona produces results with higher precision than Norma and Silk on ten of the twelve data sets. Allowing Norma and Silk to buffer all patterns improves the precision on some of the data sets, most notably for **cadata**, but it still does not reach the level of precision achieved by Priona and Okrr (figure 4.11B). Apparently Norma and Silk are unable to exploit the unlimited iteration time under this condition to produce more accurate predictions, which can be seen in figure 4.11B by the time used per iteration of both algorithms. In addition it is important to note that the optimal values for the initial learning rate η_0, given in table 4.2, now lie inside the tuning interval, which excludes the possibility that inappropriate selection of η_0

caused the observed performance difference on the **cadata** and **cpusmall** data sets.

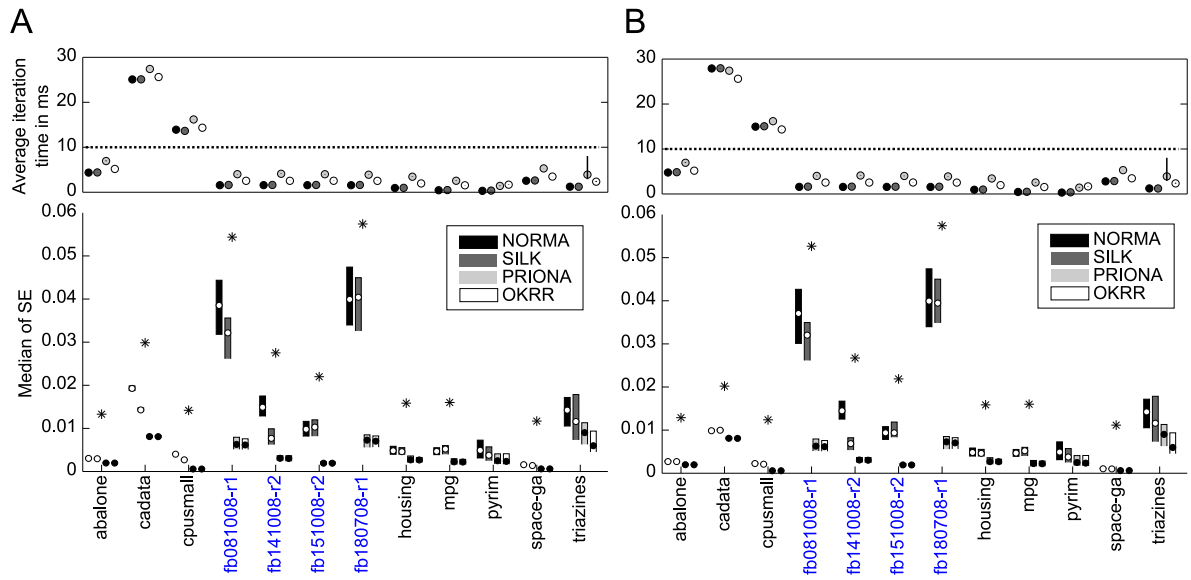


Figure 4.11: ***A:*** *When the buffer for* Priona *and* Okrr *is restricted to contain only* 64 *patterns the precision of the predicted values changes only slightly. The largest change occurs on the* ***triazines*** *data set. But now the estimated average iteration time for* Priona *and* Okrr *on the feedback data sets (highlighted in blue) is below the 10ms barrier indicated by the dotted line.* ***B:*** *The precision of the predicted values increases only moderately for* Norma *and* Silk*, if the buffer size equals the number of patterns in the data set.*

In conclusion the analysis in this section encourages the use of the new online algorithms Priona and Okrr in practice, especially in situations where the available time per iteration is limited. Although the computational complexity of one update step in Priona and Okrr is higher compared to Norma and Silk one can regulate the time per iteration by restricting the buffer size. Buffering a larger number of patterns results in convergence to a lower asymptotic error which allows to directly trade off between computation time and quality of prediction. This trade off is particularly valuable in practice for applications that put hard upper bounds on the iteration time like adaptive microstimulation, where the SVR solution is updated at a rate of 100Hz. The relationship between buffer size and prediction error also holds for Norma and Silk, but, as shown in figure 4.11B, the quality of prediction is worse compared to Priona and Okrr even for unlimited buffer sizes. Finally, Norma and Silk need tedious tuning of the initial learning rate η_0 for each data set – a task that is not necessary for Priona and Okrr which choose the optimal step size automatically via the exact line search.

	Optimal buffer size				Optimal η_0	
Data set	NORMA	SILK	PRIONA	OKRR	NORMA	SILK
abalone	512	512	512	512	1.00	2.30
cadata	512	512	512	512	2.10	10.00
cpusmall	512	512	512	512	1.40	10.00
fb081008-r1	512	512	512	512	1.00	3.30
fb141008-r2	512	512	256	512	1.10	3.80
fb151008-r2	512	512	512	512	0.70	1.10
fb180708-r1	512	512	512	512	0.90	1.30
housing	512	512	512	512	1.30	2.30
mpg	512	512	128	128	0.80	1.00
pyrim	64	64	128	128	0.80	3.10
space-ga	512	512	512	512	0.90	3.30
triazines	256	128	256	256	0.50	1.60

Table 4.1: *Optimal buffer sizes were selected for all online training algorithms from the set* $\{64, 128, 256, 512\}$. *The optimal initial learning rate* η_0 *for* NORMA *and* SILK *was tuned across the range* $0.1, 0.2, \ldots, 10$.

	Optimal buffer size		Optimal η_0	
Data set	NORMA	SILK	NORMA	SILK
abalone	4177	4177	0.60	0.70
cadata	20640	20640	1.30	1.50
cpusmall	8192	8192	1.30	3.60
fb081008-r1	600	600	1.10	3.30
fb141008-r2	600	600	1.00	3.70
fb151008-r2	600	600	0.60	0.90
fb180708-r1	512	600	0.90	0.90
housing	512	512	1.30	2.30
mpg	512	512	0.80	1.00
pyrim	64	64	0.80	3.10
space-ga	3107	3107	0.80	1.70
triazines	256	128	0.50	1.60

Table 4.2: *Optimal buffer sizes for* NORMA *and* SILK *when sizes are selected from the set* $\{64, 128, 256, 512, m\}$, *where* m *is the number of patterns in the data set. The optimal initial learning rate* η_0 *for* NORMA *and* SILK *was tuned across the range* $0.1, 0.2, \ldots, 10$.

It must be hard to be a model
because you'd want to be like the photograph of you
and you can't ever look that way.
Andy Warhol (1928-1987)

5 Model selection

The SVR algorithm uses a set of training patterns to estimate a function that is fully specified in terms of the coefficients β and offset b, as previously described in chapter 3. While the coefficients and offset are automatically determined as the solution of an optimization problem, there are so called hyper-parameters that need to be selected before the optimization starts. These hyper-parameters include the regularization parameter λ, the loss function parameter ε, and the various parameters of the kernel function, like γ in case of the RBF kernel. Model selection refers to the problem of finding suitable hyper-parameters based on available training data.

Often the specific choice of a kernel function is guided by the application domain for the SVR problem and is described for adaptive microstimulation in section 6.1.4. In addition it is sometimes possible to make an educated guess for the hyper-parameters based on experience and prior knowledge about the application. For example, the loss function parameter ε can be chosen to match the level of noise present in the regression targets. But in absence of such clues one has to resort to model selection in most practical situations.

5.1 State of the art

One of the simplest ways to solve the model selection problem independent of the learning algorithm is an exhaustive search over the full space of hyper-parameters, where each parameter combination is ranked according to an estimate of the error on the test set, and the best combination is chosen in the end. The most popular estimate of the test set error is obtained by cross validation [45]. For n-fold cross validation (CV) the available patterns are evenly split into n blocks. The algorithm is trained on one of the blocks and tested on the remaining $n-1$ blocks. After repeating this process n times the test set error can be estimated by averaging across the individual test errors of each fold.

Although the exhaustive search is appealing due to its simplicity, it scales exponentially with the number of hyper-parameters to be tuned. It is therefore infeasible to select more than 3-4 hyper-parameters simultaneously in practice. Further, since the CV by itself is computational intensive, this approach can only be applied in non time critical

situations, like the selection of hyper-parameters C, γ, and ε in the offline analysis of section 3.3. As opposed to this, there should be just a short time delay of about 5 minutes between the collection of training data and closed loop feedback in the adaptive stimulation experiments, which precludes an exhaustive search for the SVR hyper-parameters.

Fortunately there are computationally lighter approaches to solve the model selection problem for SVMs [124]. Two of the most popular approaches are either based on Bayesian reasoning or bounds on the test set error. For Bayesian model selection the SVR problem is restated in a probabilistic formulation [127, 32], that allows the hyper-parameters to be found by optimizing the evidence function [10]. This optimization is either tackled by the Laplace approximation [32] or a variational approach [127]. Yet, this chapter will focus on the derivation of test error bounds that can be minimized with respect to the hyper-parameters.

Initially bounds on the test error were introduced for classification and used information about the number of support vectors, the classification margin, the radius of the smallest sphere containing the training patterns [142], and the expansion coefficients [70, 124]. While some of these bounds turned out to be overly conservative in practice, like the bound derived from expansion coefficients [70], other bounds, like the radius margin bound, turned out to be too loose when applied in the regression setting [28]. Later a tighter bound based on the span of the support vectors was proposed [141] and it has been shown that this span bound can be optimized to select the hyper-parameters [30].

The two most important bounds in the context of SVR, the radius margin bound, and the span bound, are described in the following sections 5.1.1 and 5.1.1. The minimization of the span bound and CV error with respect to the hyper-parameters by the Quasi-Newton method is outlined in section 5.2 and section 5.3 compares model selection by minimization of the span bound and CV error on several benchmark data sets.

5.1.1 Leave-one-out bounds

The leave-one-out (LOO) error is equivalent to the m-fold CV error if there are a total of m patterns in the training set. Stated otherwise, it is computed by repeatedly training the algorithm on $m-1$ patterns an averaging across the error on the left out pattern. Formally the LOO error for SVR is defined by:

$$LOO = \sum_{t=1}^{m} |f^t(x_t) - y_t| \,, \tag{5.1}$$

where f^t denotes the function estimated by SVR when the t-th pattern is removed from the training set. It can be shown that the LOO error is an almost unbiased estimate of the test error, where unbiased refers to the fact that LOO provides an estimate for training sets of size $m-1$ instead of m [124]. The direct evaluation of equation (5.1) is computationally more costly than CV of course, and one is therefore interested in bounding this error with

a quantity that can be directly derived from the SVR solution. The starting point for the subsequently discussed LOO bounds is the solution of the SVR problem with l_2 loss function, where the associated optimization problem is given by:

$$\begin{aligned} &\min_{\alpha,\alpha^*} \frac{1}{2}(\alpha-\alpha^*)^T(K+I/C)(\alpha-\alpha^*)+\varepsilon\sum_{i=1}^{m}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{m}y_i(\alpha_i-\alpha_i^*) \\ \text{subject to} \quad &\sum_{i=1}^{m}(\alpha_i-\alpha_i^*)=0 \\ &0\le\alpha_i,\alpha_i^*\ \forall i=1,\ldots,m\ . \end{aligned} \tag{5.2}$$

The optimization problem in equation (5.2) is similar to the dual SVR formulation given in equation (3.7) of chapter 3. Only the box constraints for the coefficients α_i are replaced by positivity constraints. For the l_2 loss, the regularization parameter C is incorporated into the modified kernel matrix $K+I/C$. Note that each entry in the modified kernel matrix $(K+I/C)_{ij}=\left\langle\tilde{\phi}(x_j),\tilde{\phi}(x_j)\right\rangle$ may still be written as the dot product between patterns in feature space, if the mapping is modified according to:

$$\tilde{\phi}(x_i)=\begin{pmatrix}\phi(x_i)\\ e_i/\sqrt{C}\end{pmatrix}\ . \tag{5.3}$$

In equation (5.3) e_i denotes the i-th unit vector.

Radius margin bound

The radius margin bound for SVR is derived under the mild assumption that there are always free support vectors. This implies that there are always training patterns with associated coefficients $|(\alpha_i^*-\alpha_i)|>0$. Under this assumption it has been proven [28] that for $\alpha_t>0$ one has $f^t(x_t)\ge y_t$ and conversely for $\alpha_t^*>0$ one has $f^t(x_t)\le y_t$. This means, that information about the relative position of function f^t with respect to the regression targets is extractable from the value of the coefficients. Consequently, the following relationships for the LOO error can be derived [28]:

$$\begin{aligned} &|f^t(x_t)-y_t|\le\varepsilon,\ \text{for}\ \alpha_t=\alpha_t^*=0 \\ &f^t(x_t)-y_t\le 4R^2\alpha_t+\varepsilon,\ \text{for}\ \alpha_t>0 \\ &y_t-f^t(x_t)\le 4R^2\alpha_t^*+\varepsilon,\ \text{for}\ \alpha_t^*>0\ , \end{aligned} \tag{5.4}$$

where R is the radius of the smallest sphere that encloses all the points $\tilde{\phi}(x_i)$. In combination the equalities in equation (5.4) lead to the radius margin bound for the SVR LOO error:

$$LOO\le 4R^2\sum_{t=1}^{m}(\alpha_t+\alpha_t^*)+m\varepsilon\ . \tag{5.5}$$

Independent of the proof technique used to derive equation (5.5) one can gain an intuitive understanding of the radius margin bound. To begin with, the LOO error on pattern x_t is expected to grow if the function f^t deviates from f. Deviations of the function are naturally related to changes of the weight vector $w = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)x_i$. Clearly, the removal of x_t will lead to a large change of the weight vector if the distance of x_t to the remaining patterns in the data set is large and if the absolute value of the associated coefficient $|(\alpha_i^* - \alpha_i)|$ is large. In the primal SVR problem, given in equation (3.2), only one of the constraints can be satisfied at a time and hence the dual variables either satisfy $\alpha_i^* > 0, \alpha_i = 0$ or $\alpha_i^* = 0, \alpha_i > 0$. Consequently, the absolute value of the coefficients in the weight vector fulfills the relationship $|(\alpha_i^* - \alpha_i)| = \alpha_i + \alpha_i^*$. With these considerations in mind the radius margin bound just states that the LOO error will be large if both, the distance to the other support vectors and the absolute value of the coefficients is large. Figure 5.1 shows how the distance between patterns can be bounded by the sphere with minimum radius that encloses all the support vectors.

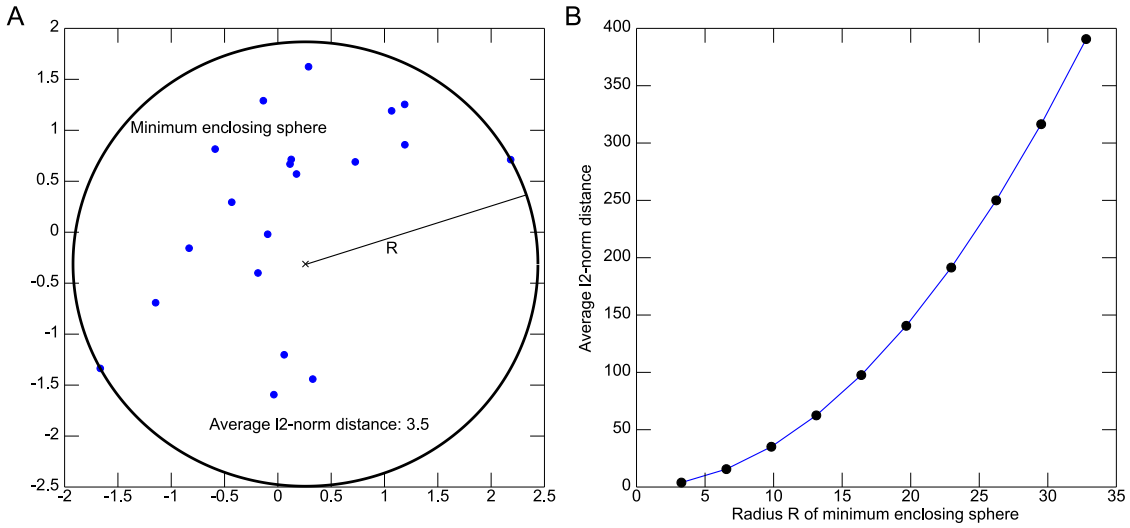


Figure 5.1: *Relationship between average l_2 norm distance between a point set and the radius R of the minimum enclosing sphere.* ***A:*** *The minimum enclosing sphere and its radius for $n = 20$ points drawn from a normal distribution $\mathcal{N}(0, 1)$.* ***B:*** *The average l_2 norm distance between $n = 1000$ points from $\mathcal{N}(0, k)$ with $k = 1, \ldots, 10$, and the radius of the minimum enclosing sphere.*

The preceding discussion gives an intuitive understanding of the first term in equation (5.5). Without the second term in the radius margin bound the loss function parameter ε can be arbitrarily increased, which will in turn lead to an empty set of support vectors and a zero radius margin bound. The experimental results with the radius margin bound in [28] indicate that it is not the best choice for model selection based on LOO bounds. The radius margin bound is therefore not used for the comparison in section 5.3. Nevertheless the radius margin bound paves the way for the minimum span bound described next.

Minimum span bound

The minimum span (MSP) bound [30, 28] for SVR is similar to the radius margin bound and defined by:

$$LOO \leq \sum_{t=1}^{m} (\alpha_t + \alpha_t^*) S_t^2 + m\varepsilon. \tag{5.6}$$

The arguments given in the preceding section for the intuitive interpretation of the radius margin bound also hold for the span bound. But instead of R^2, the quantity S_t^2 bounds the distance between the left out pattern $\phi(x_t)$ and the other support vectors. Further, S_t^2 is computed for each support vector separately while the radius margin bound uses the radius of the smallest enclosing sphere for all patterns. The quantity S_t^2 is called the span of the support vectors and corresponds to the minimum of the following optimization problem:

$$\begin{aligned} &\min_{\lambda} \|\phi(x_t) - \sum_{i \in sv \setminus t} \lambda_i \phi(x_i)\|^2 \\ &\text{subject to} \quad \sum_{i \in sv \setminus t} \lambda_i = 1 \ . \end{aligned} \tag{5.7}$$

In words the value of the span S_t^2 is equivalent to the minimal distance between the left out pattern and the convex hull of the remaining support vectors. This fact is illustrated in figure 5.2 for a set of two dimensional patterns. Unfortunately the span S_t^2 is

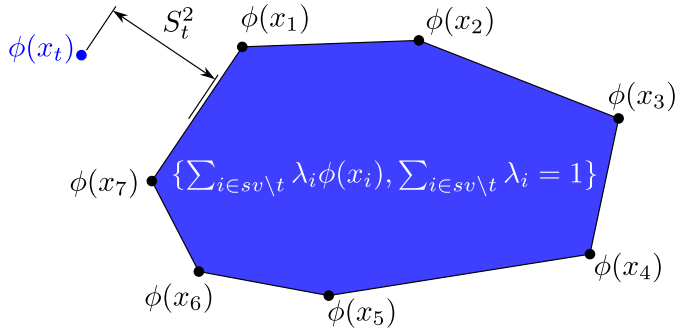


Figure 5.2: *The value of the span S_t^2 corresponds to the minimal distance between training pattern $\phi(x_t)$ and the convex hull of all other training patterns $\phi(x_1), \ldots, \phi(x_7)$ in the feature space. Each point in the convex hull can be written as $\sum_{i \in sv \setminus t} \lambda_i \phi(x_i)$ with $\sum_{i \in sv \setminus t} \lambda_i = 1$.*

not a continuous function [30], which would prevent its minimization with respect to the SVR hyper-parameters. To circumvent this problem and make S_t^2 continuous the term $\eta \sum_{i \in sv \setminus t} \lambda_i^2 \frac{1}{\alpha_i + \alpha_i^*}$ is added to the objective function in equation (5.7), where $\eta = 0.1$ is a small constant [28]. For the following discussion S_t^2 will refer to the minimum of the modified optimization problem to avoid notational clutter. In practice it is not necessary to solve the optimization problem directly, since there exists an analytical solution for the span:

$$S_t^2 = \frac{1}{(M^{-1})_{tt}} - \frac{\eta}{\alpha_t + \alpha_t^*}, \text{ with } M = \begin{bmatrix} (K + I/C + D)_{sv,sv} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}. \tag{5.8}$$

In equation (5.8) D is a diagonal matrix with entries $D_{ii} = \eta/(\alpha_i + \alpha_i^*)$. Obviously the value of the span can be directly determined from the optimal solution (α, α^*) of the SVR problem.

Now, for model selection the MSP bound is minimized with respect to the hyper-parameters that are subsequently represented by the vector θ. The necessary gradients are:

$$\frac{\partial S_t^2}{\partial \theta} = -\frac{1}{(M^{-1})_{tt}^2}\frac{\partial (M^{-1})_{tt}}{\partial \theta} + \frac{\eta}{(\alpha_t + \alpha_t^*)^2}\frac{\partial(\alpha_t + \alpha_t^*)}{\partial \theta} \tag{5.9}$$

$$\frac{\partial (M^{-1})_{tt}}{\partial \theta} = (\frac{\partial M^{-1}}{\partial \theta})_{tt} = \left(-M^{-1}\frac{\partial M}{\partial \theta}M^{-1}\right)_{tt} \tag{5.10}$$

$$\frac{\partial M}{\partial \theta} = \begin{bmatrix} \frac{\partial (K+I/C)_{sv}}{\partial \theta} + \frac{\partial D_{sv}}{\partial \theta} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (\frac{\partial D}{\partial \theta})_{tt} = -\frac{\eta}{(\alpha_t + \alpha_t^*)^2}\frac{\partial(\alpha_t + \alpha_t^*)}{\partial \theta}\ . \tag{5.11}$$

Equality (5.10) follows from a well known result about matrix derivatives [54]. For equation (5.9) and (5.11) it remains to be clarified, how to determine the partial derivative $\frac{\partial(\alpha_t + \alpha_t^*)}{\partial \theta}$.

The KKT conditions state, that the product of primal constraints and dual variables has to be zero at the optimum. For the SVR problem in equation (5.2) this leads to the following relationships:

$$\begin{aligned} ((K + I/C)(\alpha^* - \alpha))_i + b &= y_i + \varepsilon, && \text{if } \alpha_i > 0 \\ ((K + I/C)(\alpha^* - \alpha))_i + b &= y_i - \varepsilon, && \text{if } \alpha_i^* > 0 \\ |((K + I/C)(\alpha^* - \alpha))_i + b - y_i| &\leq \varepsilon, && \text{if } \alpha_i = \alpha_i^* = 0 \end{aligned} \tag{5.12}$$

With the shorthand $\hat{\alpha} = \alpha^* - \alpha$ the KKT conditions (5.12) for support vectors, with $\alpha_i, \alpha_i^* > 0$ and $\sum_{i=1}^m (\alpha_i^* - \alpha_i) = 0$, can be reformulated in compact form as:

$$\begin{matrix} (K + I/C)\hat{\alpha} + b & = & y - \mathrm{sgn}(\hat{\alpha})\varepsilon \\ \mathbf{1}^T\hat{\alpha} & = & 0 \end{matrix} \Leftrightarrow M\begin{pmatrix}\hat{\alpha}\\ b\end{pmatrix} = \begin{pmatrix} y - \mathrm{sgn}(\hat{\alpha})\varepsilon \\ 0 \end{pmatrix} \tag{5.13}$$

In the discussion of the radius margin bound it was argued that $(\alpha_t + \alpha_t^*) = |(\alpha^* - \alpha)|$ and it follows that $\frac{\partial(\alpha_t + \alpha_t^*)}{\partial \theta} = \mathrm{sgn}(\hat{\alpha}_t)\frac{\partial \hat{\alpha}_t}{\partial \theta}$. Only the right hand side of equation (5.13) depends on ε while the remaining hyper-parameters in vector θ depend on the matrix M. By using the product rule to differentiate the KKT conditions (5.13) one can distinguish two cases.

In the first case $\theta \neq \varepsilon$ and differentiation of the KKT conditions yields:

$$M\begin{pmatrix}\partial\hat{\alpha}/\partial\theta \\ \partial b/\partial\theta\end{pmatrix} + \frac{\partial M}{\partial \theta}\begin{pmatrix}\hat{\alpha}\\ b\end{pmatrix} = 0 \Rightarrow \begin{pmatrix}\partial\hat{\alpha}/\partial\theta \\ \partial b/\partial\theta\end{pmatrix} = -M^{-1}\begin{pmatrix}\frac{\partial(K+I/C)}{\partial\theta}\hat{\alpha} \\ 0\end{pmatrix}\ , \tag{5.14}$$

from which $\frac{\partial \hat{\alpha}_t}{\partial \theta}$ can be determined.

In the second case $\theta = \varepsilon$ and differentiation of the KKT conditions yields:

$$\begin{pmatrix}\partial\hat{\alpha}/\partial\varepsilon \\ \partial b/\partial\varepsilon\end{pmatrix} = M^{-1}\begin{pmatrix}-\mathrm{sgn}(\hat{\alpha}) \\ 0\end{pmatrix}\ , \tag{5.15}$$

from which $\frac{\partial \hat{\alpha}_t}{\partial \varepsilon}$ can be determined. In summary the MSP bound is minimized with respect to a general vector of hyper-parameters θ by combining equations (5.9)-(5.11), (5.14), and (5.15).

How can this framework be used for a particular kernel function? Let's consider the example of the RBF kernel that is discussed in detail in section 6.1.4. The RBF kernel function for patterns x_i, x_j is defined by:

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \ , \tag{5.16}$$

and the resulting kernel matrix is $K_{ij} = k(x_i, x_j)$. Since there is only one kernel parameter γ the vector of hyper-parameters is $\theta = (C, \gamma, \varepsilon)$. The gradients of the span bound depend on the kernel function only through equation (5.11). It is therefore sufficient to compute the partial derivative of $\frac{\partial (K+I/C)_{sv}}{\partial \theta}$. Evidently the partial derivative with respect to the loss function parameter ε is zero. For the kernel parameter γ the partial derivative is:

$$\frac{\partial (K + I/C)_{ij}}{\partial \gamma} = -\|x_i - x_j\|^2 \exp(-\gamma \|x_i - x_j\|^2) \ , \tag{5.17}$$

and the derivative with respect to the regularization parameter C is:

$$\frac{\partial (K + I/C)_{ij}}{\partial C} = \begin{cases} -\frac{1}{C^2}, & i = j \\ 0, & \text{otherwise} \ . \end{cases} \tag{5.18}$$

The only missing part to turn this framework into a model selection method is an optimization algorithm that uses solely gradient information to minimize the MSP bound.

5.2 Minimizing the MSP bound and CV error

The Quasi-Newton optimization algorithm is a common choice when no information is available about the second order derivative of the objective function to be minimized [6]. As indicated by its name, the algorithm works similar to the Newton optimization used in chapter 3 for solving the primal SVR problem with l_2 loss. In each iteration step the descent direction is computed as the product of a positive definite matrix and the negative gradient of the objective function. While this matrix corresponds to the inverse Hessian in the standard Newton optimization, the Quasi-Newton algorithm uses an approximation of the inverse Hessian calculated from the gradients $\nabla f(\theta_{k+1}), \nabla f(\theta_k)$, the iterates θ_{k+1}, θ_k, and the approximation H_k at the previous step k. During the first iteration the approximation is usually initialized by the identity matrix and is guaranteed to remain positive definite throughout the whole optimization process, if updated according to the Broyden-Fletcher-Goldfarb-Shanno method [6]. Algorithm 1 in appendix A contains a formal description of the Quasi-Newton algorithm. The chain of iterates produced by the Quasi-Newton optimization of a simple example function is shown in figure 5.3.

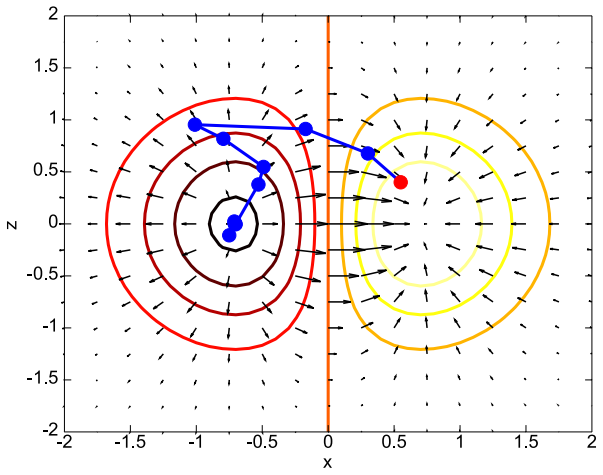


Figure 5.3: *Contour plot of example function* $f(x, z) = x \exp^{-x^2-z^2}$ *and the optimization path followed by the Quasi-Newton method. Black arrows represent the gradient of the function. The red dot at* $(0.55, 0.4)$ *indicates the starting point of the optimization and the blue dots intermediate solutions.*

The SVR algorithm is known to be quite robust with respect to small changes in the hyper-parameters [28]. It is therefore practical to optimize the hyper-parameters on a logarithmic scale. In this case the gradient calculations of section 5.1.1 are still applicable by exploiting the following relationship between the partial derivatives of the objective function:

$$\frac{\partial \ln \theta}{\partial \theta} = \frac{1}{\theta} \Rightarrow \frac{\partial f(\theta)}{\partial \ln \theta} = \theta \frac{\partial f(\theta)}{\partial \theta} \,. \tag{5.19}$$

While the gradients of the MSP bound are computable analytically as described in section 5.1.1, the gradients of the CV error have to be approximated numerically by central differences:

$$\frac{\partial f(\theta)}{\partial \theta} = \frac{f(\theta + h) - f(\theta - h)}{2h} \,. \tag{5.20}$$

If k denotes the number of selected hyper-parameters one gradient evaluation according to equation (5.20) costs $2k$ function evaluations and hence scales linearly with k. One can empirically study the scaling behaviour by considering the selection of an increasing number of parameters γ_i for the extended RBF kernel:

$$k(x, z) = \exp(-\sum_{i=1}^{k} \gamma_i \sum_{j \in \mathcal{I}_i} (x_j - z_j)^2) \,, \tag{5.21}$$

where $x, z \in \mathbb{R}^n$ and $\{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$ denotes an arbitrary partition of the index set $\{1, \ldots, n\}$. With this definition of the RBF kernel it is possible to implement a feature selection by individually choosing the scaling factors γ_i. Figure 5.4 confirms the expected scaling behaviour for the **fb081008-r1** data set.

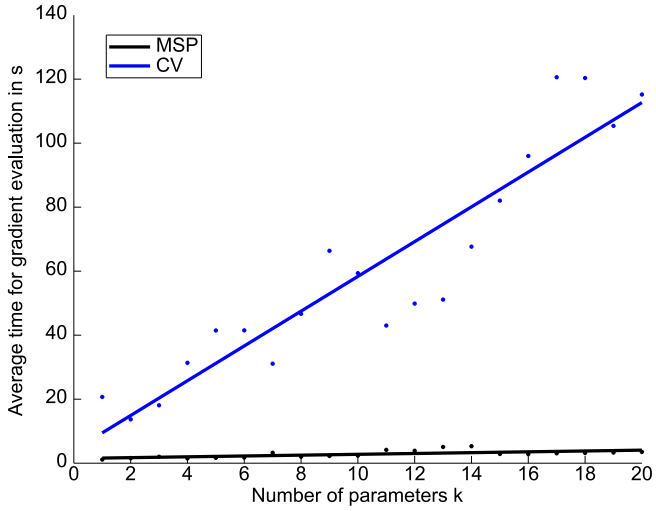


Figure 5.4: *Average run time required to evaluate the MSP bound and CV error gradients for an increasing number k of factors γ_i in the extended RBF kernel function. Both the MSP bound and CV error gradient evaluations scale linearly in dependence of k. But the constants for the linear scaling function are better for MSP bound gradient evaluations.*

5.3 Results

The potential to select SVR hyper-parameters by means of the MSP bound or tenfold CV error is evaluated here with respect to the run time of the selection method and the precision of the resulting SVR model on test data. In order to asses the benefits of the MSP bound and CV error independent of the optimization method both error measures are combined once with an exhaustive search over the whole parameter space and once with the Quasi-Newton algorithm described in section 5.2. This essentially leads to a comparison of four different model selection methods:

1. Minimization of the MSP bound with the Quasi-Newton algorithm (QN-MSP)
2. Exhaustive search using the MSP bound (ES-MSP)
3. Minimization of the CV error with the Quasi-Newton algorithm (QN-CV)
4. Exhaustive search using the CV error (ES-CV).

The performance of the different model selection methods for choosing parameters C, γ, and ε for SVR with RBF kernel is shown in figure 5.5 in terms of the average run time required by each method and the average MSE achieved on independent test data. The results are obtained by first running each selection method ten times on random partitions of the data into 50% training and test subsets and subsequently computing 95% confidence intervals by the bootstrap with B=1000 bootstrap samples [41].

On all data sets except **cpusmall**, **pyrim**, and **triazines** the selection by ES-MSP incurs significantly larger test set errors than the other methods. Qualitatively similar results are obtained for selection by QN-MSP, although it performs better than ES-MSP on **abalone**,

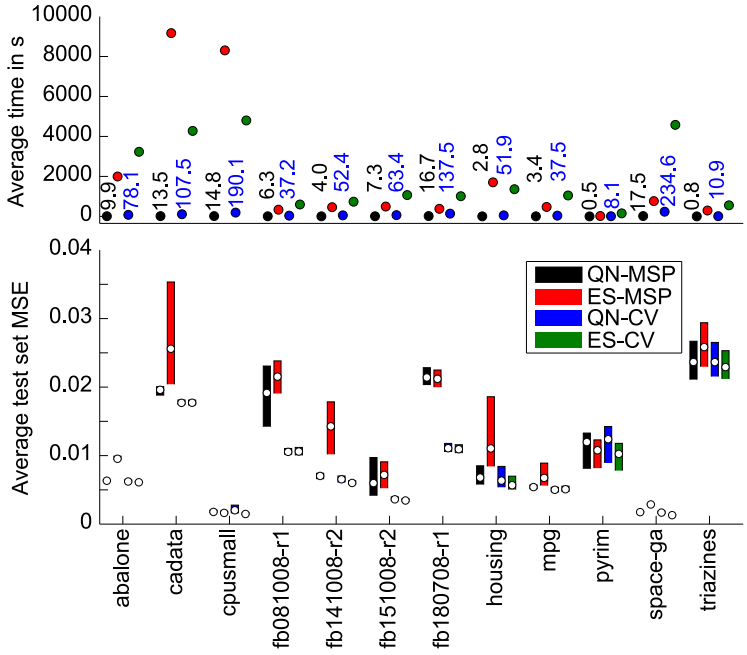


Figure 5.5: *Comparison of the model selection methods QN-MSP, ES-MSP, QN-CV, and ES-CV on twelve benchmark data sets. The lower part of the figure shows 95% confidence intervals for the average MSE on the test set. The upper part of the figure shows the average run time in seconds required by each selection method. To ease the comparison the run times for two fastest methods QN-MSP and QN-CV are given numerically.*

cadata, **housing**, and **mpg**. In contrast the methods based on the CV error measure consistently yield the best results across all data sets.

The upper part of figure 5.5 shows the average selection time consumed by the different model selection methods. As expected, minimization of the error measure with the Quasi-Newton algorithm requires less run time than an exhaustive search over parameter space. Further, minimization of the MSP bound is more efficient than minimization of the CV error. The latter observation can be attributed to the fact that gradient evaluations for the CV error, which use the central difference formula, are more costly than evaluations of the analytical MSP bound gradient. But QN-CV is still a viable approach with respect to run time in practice, especially for the feedback data sets **fb081008-r1**, **fb141008-r2**, **fb151008-r2**, and **fb180708-r1**, where the method requires a maximum of about two minutes to select the hyper-parameters.

The thorough examination of the $\ln C$ - $\ln \gamma$ plane in parameter space indicate spurious minima of the MSP bound in the region of high C/γ values. As opposed to QN-MSP which might get stuck before reaching the global minimum, these spurious extremal points are always found by ES-MSP. In part this difference can be explained by numerical round off errors that make the evaluation of the analytical gradient of the MSP bound inaccurate. But this failure can be overcome by computing the gradients by automatic differentiation [56], although at the expense of substantially higher run times. If the MSP bound gradient is evaluated by automatic differentiation the results of QN-MSP and ES-MSP are almost identical (results not shown).

The underestimation of the LOO error by the MSP bound can be observed for the **cadata** data set, where QN-MSP produces a lower test set error in contrast to ES-MSP.

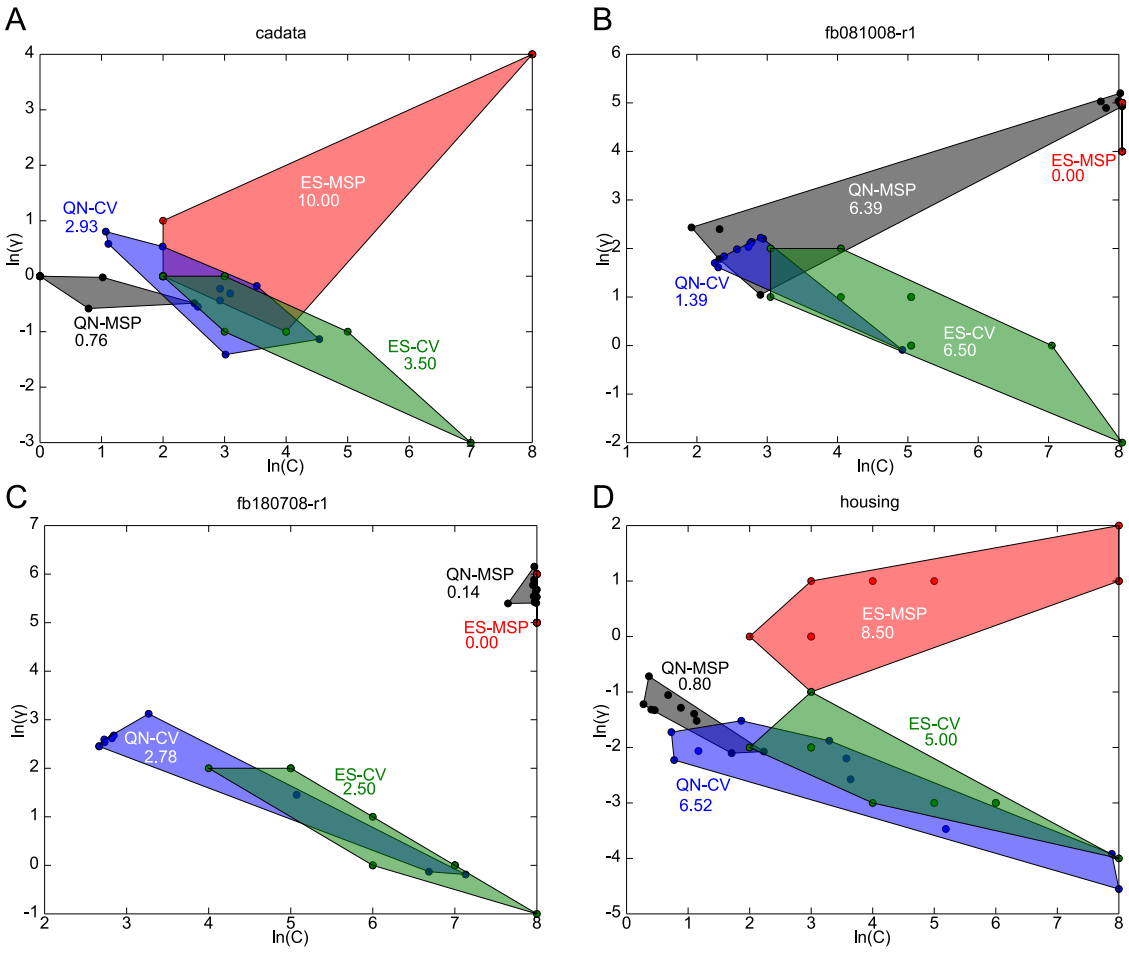


Figure 5.6: *Examples of parameter regions in the* $\ln C$ - $\ln \gamma$ *plane found by different model selection methods. Each point corresponds to one hyper-parameter combination chosen for one of the ten random splits into training and test subsets. The parameter regions are represented by the convex hull of these points. Numbers below the method name are the area of the parameter region* ***A: cadata****.* ***B: fb081008-r1****.* ***C: fb180708-r1****.* ***D: housing****.*

Figure 5.6 shows the $\ln C$ - $\ln \gamma$ parameter space and the optimal combination of hyper-parameters identified by the different model selection methods. Obviously QN-MSP finds parameter pairs that are located in a region of low C and γ values while the selection region of ES-MSP extends up to the point $(\ln C, \ln \gamma) = (8, 4)$. For the housing data in figure 5.6D one can observe the same behavior of QN-MSP and ES-MSP, but for the adaptive stimulation data sets **fb081008-r1** and **fb180708-r1**, shown in figures 5.6B and C, both methods end up in an inadequate region of parameter space of high C and γ values.

The area of the parameter region displayed in figure 5.6 is a measure of spread for the optimal hyper-parameter combinations and indicates how stable a particular model selection method is with respect to changes in the training data. Although there are notable differences between the areas of different model selection methods for individual data sets, there is no consistent trend across all data sets.

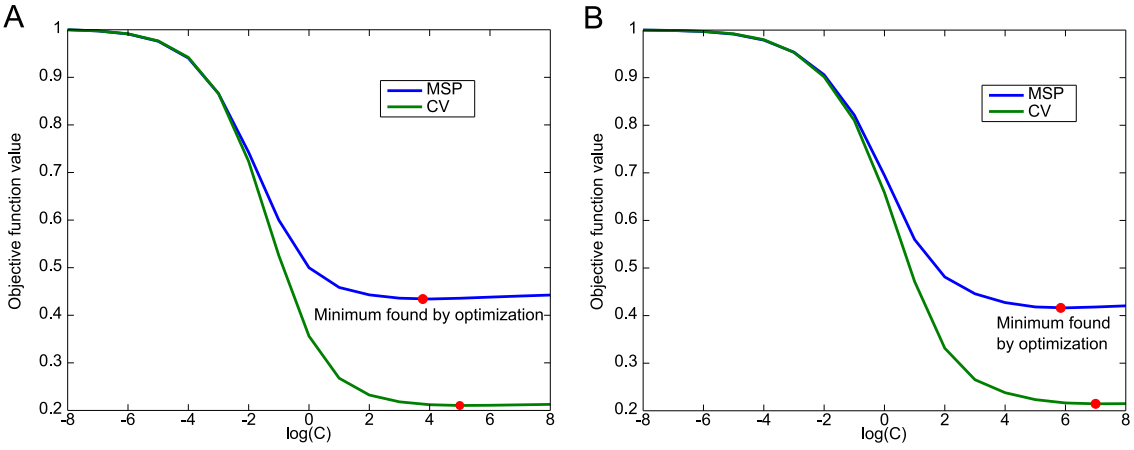


Figure 5.7: *Selection of regularization parameter C by minimizing the MSP bound or CV error. The MSP and CV values are scaled to the interval $[0, 1]$ to ease the comparison. For both data sets (**A: fb081008-r1** and **B: fb180708-r1**) the optimal values for $\ln C$ are close, and selection via the MSP bound results in a lower value for $\ln C$.*

The presented results suggest that QN-CV is the method of choice for the model selection problem, but this picture changes when one considers the tuning of a single hyper-parameter only, for instance the regularization parameter C. This scenario arises in the adaptive stimulation experiments described in section 4, where the SVR algorithm is essentially used in conjunction with the linear kernel function and the loss function parameter is fixed at the noise level of the stimulus generator ($\varepsilon = 0.002$). Under these conditions the MSP bound does not display any spurious minima and the minima of the MSP bound and the CV error are located close together, as shown in figure 5.7 for the two data sets **fb081008-r1** and **fb180708-r1** from the adaptive stimulation experiments.

Figure 5.8 shows the average test set MSE, the selected regularization parameter C, and the average time needed for the selection process for the four feedback data sets. The

value of the MSP bound and the CV error are minimized by a golden section search[1)] in this case. Surprisingly the minimization of the MSP bound is better suited for this kind of model selection problem since it incurs a lower test set MSE, selects the regularization parameter in a stable manner, and consumes the least run time for the selection.

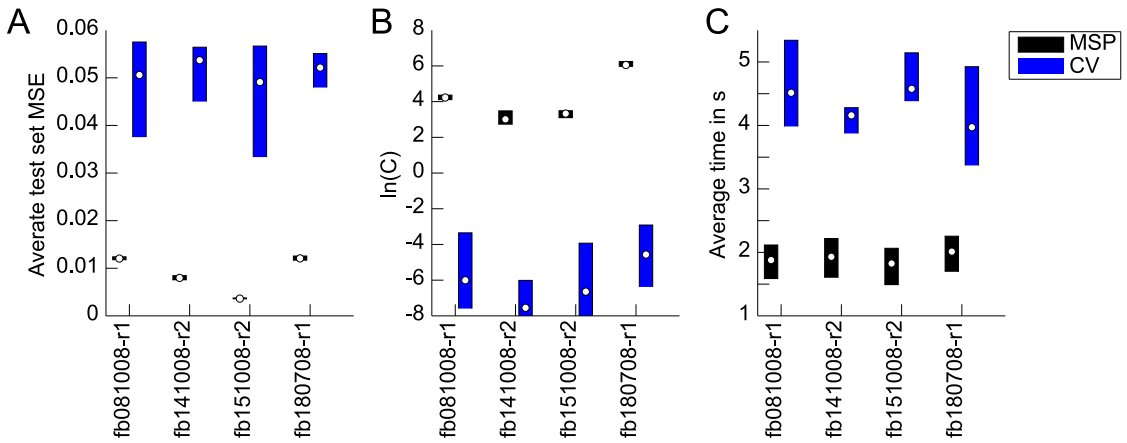


Figure 5.8: *Comparison of MSP bound and CV error minimization for selecting the regularization parameter C of the linear SVR algorithm on four feedback data sets. Bars in all plots represent 95% confidence intervals.* ***A:*** *The average MSE on the test set.* ***B:*** *The value of the selected regularization parameter C.* ***C:*** *The average time required by the model selection method in seconds.*

In conclusion there is no silver bullet for solving the model selection problem. For selection of the hyper-parameters of SVR with RBF kernel the MSP bound underestimates the true LOO error for high values of C and γ, and minimizing the CV error with the Quasi-Newton algorithm is the better approach. On the other hand the MSP bound instead of the CV error should be optimized, if only the regularization parameter C is selected. The situation where only C needs to be optimized arises frequently in practice, for example in the adaptive microstimulation experiments described in chapter 7, where suitable kernel parameters are determined separately.

[1)] Using the **fminbnd** function in MATLAB®.

He who seeks for methods
without having a definite problem in mind
seeks in the most part in vain.

David Hilbert (1862 - 1943)

6 Decoding the cortical state

This chapter describes how the primal SVR algorithm developed in chapter 3 and the model selection method of chapter 5 are applied to the problem of adaptive microstimulation in the barrel cortex of anesthetized rats. The application of the online training algorithm developed in chapter 4 is discussed in chapter 7. Before it is possible to dynamically change the stimulus parameters it is crucial to find a relationship between the brain state, the stimulus evoked potential, and the stimulus parameters. It is clear that the brain state is somehow reflected by the ongoing brain activity, but it remains to be clarified how the state is encoded by the ongoing activity. So far there has been no prior work that investigated whether the brain's state can be used to adjust stimulus parameters in order to stabilize evoked cortical potentials.

6.1 State of the art

Although the best way to characterize the brain state for the purpose of adaptive microstimulation is unknown, there is an abundance of work about decoding the voluntary modulations of brain activity to drive an effector, like a cursor on a computer screen or a prosthetic hand [97, 44, 125]. The preprocessing and decoding strategies described in this work will serve as a starting point for finding an appropriate description of the brain state for adaptive microstimulation.

Historically the first reliable way to decode reaching directions of a monkey in three dimensional space from brain activity is the population vector method [53]. In these early experiments a monkey had to do a center-out reaching task, where it started with its hand at the center of a three dimensional cube and had to move the hand to one of the cube's corners upon receiving a cue. Simultaneously the activity of single neurons in the primary motor cortex was recorded. It was discovered that single neurons respond to a broad range of movement directions, but also had one preferred direction where the firing rate is highest. When the preferred direction is represented by a vector in three dimensional space the actual moving direction of the monkey's hand could be decoded from the population activity by computing the weighted vector sum over the whole population of recorded neurons.

For each cell the weight of the direction vector was a function of the movement direction and proportional to the firing rate of the cell [53].

The population vector method only allows predicting movement directions, but it was later shown that even trajectories of three dimensional hand movements can be predicted by using back-propagation neural networks to decode the brain activity [145]. Furthermore this work demonstrated that signals recorded from cortical areas outside the primary motor cortex could also be used for predicting the hand trajectory. Recently this and other pioneering work on monkeys [136] has been applied in a clinical trial to restore rudimentary motor actions in a tetraplegic patient [64]. In this study the action potentials of single neurons were recorded via an electrode array [99] implanted in the primary motor cortex of the patient. The patient modulated the recorded neural activity by imagined hand movements and these modulations could be decoded to either move a cursor on a computer screen, or open and close a prosthetic hand [64].

The work on decoding motor actions described so far relies on the activity of single neurons as a control signal. This type of signal is also called single unit activity (SUA) and requires the recording of action potentials from a single neuron. Unfortunately the quality of the SUA is susceptible to changes on the boundary between electrode surface and the neural tissue, which is a major drawback for long-term clinical applications [133]. In contrast to SUA the local field potential (LFP) is easier to record and is more stable with respect to changing electrode properties. It has been shown, that the LFP is a viable alternative to SUA when it comes to decoding the movement direction [91, 113], and it has also been used as a control signal in human subjects [75]. Besides the LFP and SUA the so called multi-unit activity (MUA) has been employed to predict movement direction, grasp type and, two dimensional hand trajectories in monkeys [133]. As the name already points out the MUA can be regarded as the superimposed SUA of multiple cells, but, like the LFP, the MUA does not need sophisticated recording techniques to acquire it.

To summarize, the SUA, LFP, and MUA are signals reflecting the ongoing brain activity and have been successfully used to read out intended movement directions and trajectories of the hand. In this chapter it will be explored whether these signals are suited for the purpose of adaptive microstimulation.

Before describing the extraction of LFP, MUA, and the signal's phase in sections 6.1.1, 6.1.2, and 6.1.3, it is instructive to see what kind of neural activity can be captured by a microelectrode recording. Figure 6.1 shows the spatial resolution of common recording techniques [126]. The electroencephalogram (EEG) is recorded via electrodes on the scalp and is mostly used for clinical diagnostics, for example to monitor the level of anesthesia. It has a coarse spatial resolution and samples the activity from a large portion of the cortical surface. More focal measurements are possible by placing the electrodes directly on the cortical surface, which is done to record the electrocorticogram (ECoG). An even higher spatial resolution is achieved by inserting microelectrodes into the neural tissue of the cortex. From the raw data recorded with microelectrodes it is possible to obtain the LFP which samples the activity of a local population of neurons. When the microelectrode is

carefully placed in the vicinity of a cell body it is feasible record the action potentials of a single nerve cell.

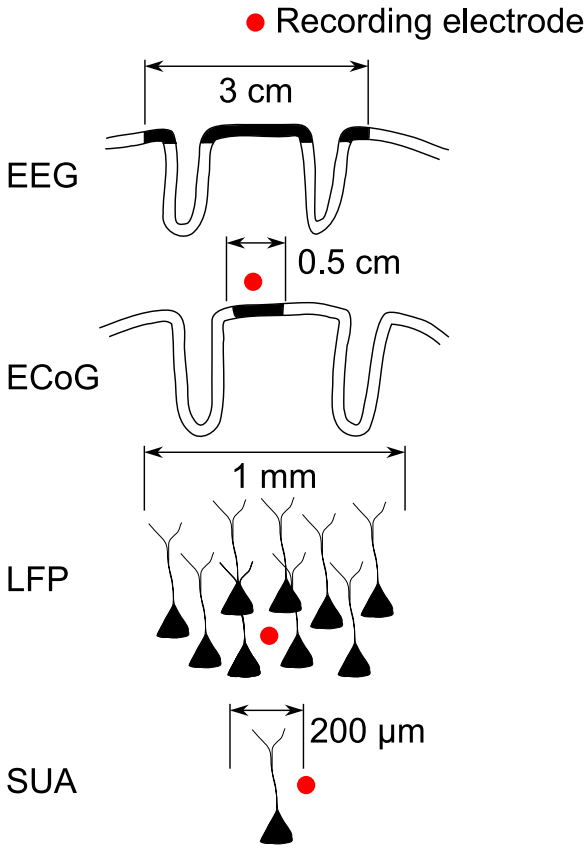


Figure 6.1: *Spatial resolution of different recording techniques. For noninvasive EEG recordings the electrode is located at a distance of about 1.5cm from the cortical surface. As a consequence the EEG has a low spatial resolution and neural activity is recorded from a large portion of the cortical surface that spans up to 3cm. The electrocorticogram (ECoG) samples a population of neurons that cover about 0.5cm of cortical surface and provides a more focused measurement in contrast to the EEG. To record the ECoG, the electrode is directly placed on the cortical surface but it does not penetrate the neural tissue. In contrast, microelectrodes are inserted into the neural tissue to record LFP and SUA. The LFP reflects the synaptic activities of a local population of neurons that may span up to 1mm. The highest spatial resolution is obtained by recording the action potentials or SUA of a single nerve cell.*

6.1.1 Local field potentials

The LFP is extracted from the raw signal recorded via a microelectrode by bandpass filtering in the range of 1-200Hz. Figure 6.2A shows the raw signal recorded from the barrel cortex of an anesthetized rat, and figure 6.2B the LFP obtained after filtering. As opposed to action potentials, that indicate the activity of a single neuron after receiving synaptic input, the LFP directly reflects the synaptic activity of a more or less local population of neurons [92]. This makes the LFP harder to interpret, but it has been shown that the LFP and its spectral decomposition encode the movement direction in a center-out reaching task with a monkey [113]. It is even possible to discriminate between different types of movement intentions, like hand or eye movements, by analyzing the LFP [121]. Further it has been reported, that the LFP conveys approximately twice as much information about arm movement directions in comparison to the ECoG [91].

When multiple LFPs are recorded via a laminar electrode array, with electrodes oriented perpendicular to the cortical surface, it is possible to calculate the actual current

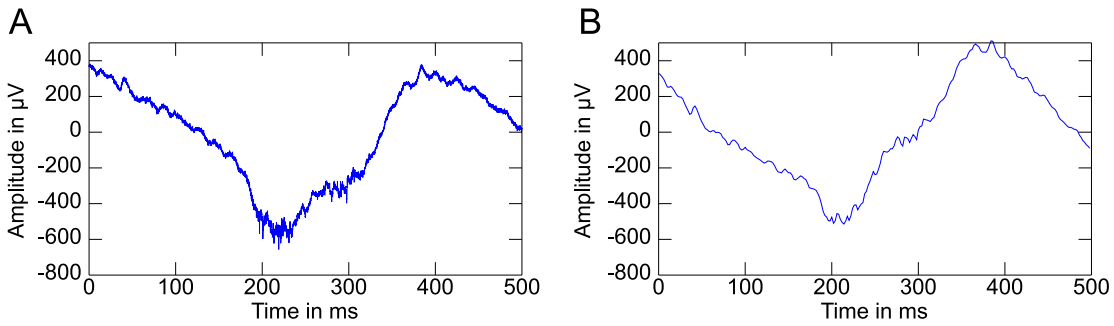


Figure 6.2: ***A:*** *Example of raw signal recorded from barrel cortex with a sampling frequency of 20KHz.* ***B:*** *Local field potential extracted from the raw signal by bandpass filtering in the range of 1-200Hz (Butterworth order 2).*

sources that generate the recorded field potential via current source density (CSD) analysis [96, 105, 106]. This type of analysis provides detailed information about the neural activity in a cortical column and has been applied to analyze the sensory response after whisker stimulation in anesthetized rats for example [40]. Unfortunately the CSD analysis currently cannot be applied to extract information from recorded LFPs in the adaptive microstimulation experiments since only one electrode is available to register the neural activity (cf. figure 6.6).

6.1.2 Multi-unit activity

The processing to extract the MUA is more elaborate compared to the extraction of the LFP. First the recorded raw signal is bandpass filtered in the range of 300-6000Hz to remove the slow potentials. Then all values deviating more than two standard deviations from the mean signal are clipped and the resulting values are squared to rectify the signal. Finally, the signal is averaged by lowpass filtering with a cutoff frequency of 100Hz, down sampled to 500Hz, and each value is raised to the power 0.5 to complete the rectification [133]. Figure 6.3 shows the result of this processing for a short cutout of the raw signal.

As mentioned previously, the MUA can be regarded as the superposition of the SUA of a population of nerve cells. The spikes, or action potentials, that constitute the SUA have been proven to be a reliable source of information in past studies of the nervous system [7]. With the recent introduction of large scale recordings it is possible to detect spike patterns of whole ensembles of neurons [22]. However, large scale recordings of spikes are faced with some serious limitations in practice. On the one hand it is hard to maintain stable recordings with good signal quality for long time periods in experimental setups. On the other hand the SUA cannot be obtained directly from the raw signal in most cases but instead requires sophisticated spike sorting algorithms to assign detected spikes to a particular neuron [85, 63, 62, 16]. Besides these practical considerations, the relationship

between SUA and the overall level of neural activity in a brain region is often unclear, while the MUA provides a measure to quantify this relationship [84].

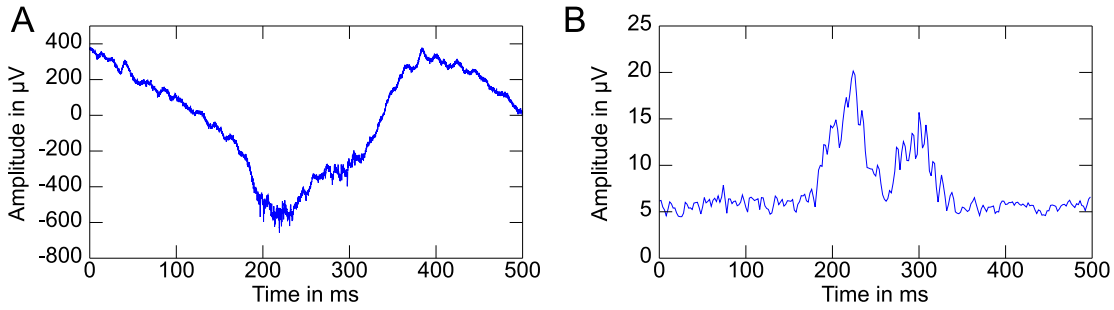


Figure 6.3: ***A:*** *Example of raw signal recorded from barrel cortex with a sampling frequency of 20KHz.* ***B:*** *Multi-unit activity extracted from the raw signal by bandpass filtering in the range of 300-6000Hz, rectification, lowpass filtering with cutoff frequency 100Hz, and down sampling to 500Hz.*

In contrast to LFPs and SUA, the MUA is not routinely used in experimental studies, but it has been shown to encode the reaching direction and grasp type in monkeys [133]. In these experiments the monkey had to perform either a center-out reaching task or trace a given path in two dimensions with its hand. The highest accuracy for the classification of reaching direction and grasp type was achieved with MUA, while the performance of a classifier based on SUA or multiple spikes was worse. Surprisingly the combination of LFP and MUA worked best for the prediction of two dimensional hand trajectories. This result is unexpected since it has been recently reported that spikes can be inferred from the LFP to a certain extent [111], which implies some redundancy in the information carried by LFP and MUA. But in some applications the LFP and MUA seem to complement each other, which can be seen by comparing figure 6.2B and figure 6.3B. This ad hoc observation is further supported by the results presented in section 6.5.1, where the best results for the direct solution (section 6.3.1) are achieved with a combination of LFP and MUA.

6.1.3 Phase synchronization

The basic phenomenon of synchronization is present in different scientific areas and was initially discovered by Huygens [115]. Classically synchronization is defined as the adjustment of frequencies of periodic self-sustained oscillators and emerges due to weak interactions. More recently phase synchronization of chaotic systems has been defined as the appearance of a certain relation between the phases of interacting systems, or alternatively between the phase of the system and an external force. Aside from the emergence of phase relationships the amplitudes of the interacting systems can remain chaotic and are in general uncorrelated [115].

In the field of neuroscience synchronization is a potential mechanism for information processing within a brain area and it might also play an important role for the communication between different brain areas [115]. One fundamental question in the visual system, that so far remained unanswered, concerns the binding of different but related visual features to enable the perception of a visual pattern or object. Results from animal experiments suggest that this binding problem could be partially solved by synchronization of the neural activity in visual cortex [130]. Under these assumptions a visual pattern is perceivable if the neurons representing the visual features were synchronously active. On a more local scale, analysis of the LFP phase has been reported to predict the sensory response evoked by whisker deflections in the barrel cortex of anesthetized rats [60].

The phase synchrony between neural signals can be determined by wavelet methods or the Hilbert transform, although both approaches were reported to yield equivalent results for the analysis of EEG and ECoG signals [82]. Given a scalar signal $s(t)$ the corresponding analytical signal $\zeta(t)$ is a complex function defined by:

$$\zeta(t) = s(t) + i(\mathcal{H}s)(t) = A(t)e^{i\phi(t)} \,, \tag{6.1}$$

where $(\mathcal{H}s)(t)$ is the Hilbert transform of the signal, $A(t)$ is the amplitude, and $\phi(t)$ the so called instantaneous phase. The Hilbert transform of $s(t)$ is defined as:

$$(\mathcal{H}s)(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s(t)}{\tau - t} dt \,. \tag{6.2}$$

Due to the singularity of the integrand in equation (6.2) the integral is taken in the sense of the Cauchy principal value, e.g. $\lim_{\epsilon \to 0}(\int_{-\infty}^{\tau-\epsilon} + \int_{\tau+\epsilon}^{\infty})$. From a computational point of view the discrete time analytic signal and the corresponding Hilbert transform can be computed by means of the FFT algorithm [90].

The phase synchrony between two scalar signals is usually quantified by first finding the instantaneous phase of both signals via the Hilbert transform followed by the computation of a phase locking value that measures the similarity of the instantaneous phases. Although one at least needs two signals to define synchrony it is also possible to directly search for connections between the instantaneous phase and other neural events like evoked sensory responses [60]. The latter approach is followed here, since brain activity is recorded solely over one channel in the experimental setup for adaptive stimulation.

6.1.4 Kernel functions

Every linear algorithm that just computes dot products $\langle x, z \rangle$ between input patterns $x, z \in \mathbb{R}^n$, and hence is invariant to rotations of the input space, can be extended to the nonlinear case by replacing the computation of the dot product by the evaluation of a kernel function. The kernel function $k(x, z)$ is equivalent to the dot product $\langle \phi(x), \phi(z) \rangle$ between input patterns that have been mapped by the nonlinear function ϕ to a so called feature space. This feature space is usually higher dimensional than the input space, and

for some kernel functions it can even have an infinite number of dimensions. It is therefore not feasible in most cases to explicitly compute the mapping ϕ and then evaluate the dot product. But fortunately it is possible to derive closed form expressions for many feature mappings that allow efficient evaluation of $\langle \phi(x), \phi(z) \rangle$.

From an alternative point of view the kernel function can be considered as a similarity measure between input patterns. Of course the notion of similarity strongly depends on the application and the type of objects that the input patterns represent. The choice of a suitable kernel functions then allows incorporating prior knowledge about the problem at hand without changing the algorithm. Examples are kernel functions that were developed for discrete structures like strings [86], graphs [51, 128], and spike trains [129]. Before describing the kernel for spike trains in section 6.1.4 and the kernel function for adaptive stimulation in section 6.4 it is instructive to first study the properties of standard kernel functions like the polynomial kernel and RBF kernel [124].

Polynomial kernel

For the polynomial kernel an input pattern $x \in \mathbb{R}^n$ is mapped to the space of all monomials that have maximal degree d. Here it is possible to explicitly write down the feature mapping by indexing the feature vector with the n-dimensional vector $\mathbf{j}$ that represents one feasible combination of exponents in the monomial:

$$\phi_{\mathbf{j}}(x) = \prod_{i=1}^{n} x_i^{j_i}, \ \mathbf{j} \in \mathbb{N}^n, \ \sum_{i=1}^{n} j_i = d \tag{6.3}$$

Using this explicit form of the feature mapping one can derive the following closed form expression for evaluating the polynomial kernel:

$$\begin{aligned} \langle \phi_{\mathbf{j}}(x), \phi_{\mathbf{j}}(z) \rangle = \sum_{1 \le i_1 \le \ldots \le i_d \le n} x_{i_1} \cdot \ldots \cdot x_{i_d} \, z_{i_1} \cdot \ldots \cdot z_{i_d} &= \sum_{i_1=1}^{n} x_{i_1} z_{i_1} \cdot \ldots \sum_{i_d=1}^{n} x_{i_d} z_{i_d} \\ &= (\sum_{i=1}^{n} x_i z_i)^d = \langle x, z \rangle^d \ . \end{aligned} \tag{6.4}$$

The first equality in equation (6.4) holds, due to the sum constraint in equation (6.3) for the vector of exponents. Obviously the special case of $d = 1$ is equivalent to the linear kernel. What is the dimensionality of the feature space that is induced by the polynomial kernel? Each component of the feature vector consists of one monomial which is the product of d single features x_i that were chosen from the n components of the input pattern with replacement. Since there are $\binom{d+n-1}{d}$ different choices the feature space induced by the polynomial kernel has dimensionality $\binom{d+n-1}{d}$. For example, if $d = 5$ and $n = 20$ then the feature space has 42504 dimensions.

Besides the polynomial kernel $k(x, z) = \langle x, z\rangle^d$ there exists the inhomogeneous polynomial kernel that uses an additional parameter c and is defined as:

$$k(x, z) = (\langle x, z\rangle + c)^d = \sum_{k=0}^{d} \binom{d}{k} c^{d-k} \langle x, z\rangle^k \,, \; c \in \mathbb{R}, \; d \in \mathbb{N} \,. \tag{6.5}$$

The second equality in equation (6.5) follows from the binomial formula. Although the change in definition seems to be subtle at first sight it can be seen that the inhomogeneous polynomial kernel actually computes a weighted sum over polynomial kernels of all degrees between zero and d. By increasing parameter c the relative weighting of higher order polynomials is decreased [128].

RBF kernel

There exists no explicit expression for the feature vector that is associated with the RBF kernel. On two input patterns $x, z \in \mathbb{R}^n$ the RBF kernel function is evaluated according to:

$$k(x, z) = \exp(-\gamma \|x - z\|^2) \,, \tag{6.6}$$

where $\gamma \in \mathbb{R}^+$ is the kernel parameter that controls the shape of the RBF kernel. When γ approaches zero the RBF kernel function is almost linear in dependence of the distance $\|x - z\|$, as shown in figure 6.4. In the limit of large values for γ the RBF kernel function approximates the nonlinear Dirac δ-function.

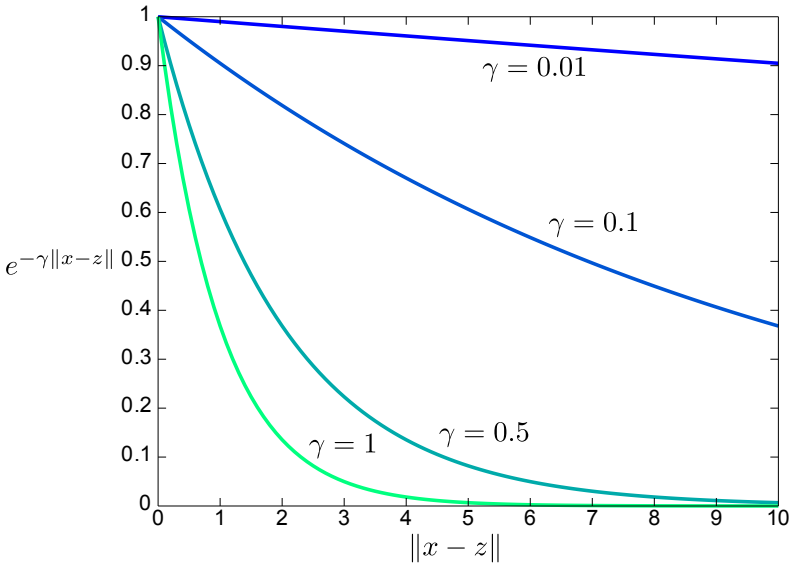


Figure 6.4: *Shape of the RBF kernel function for different values of the kernel parameter γ. When γ tends to infinity the kernel function approaches the nonlinear Dirac δ-function in the limit while for small values of γ it approaches a linear function.*

The specific choice of the RBF kernel parameter γ[1)] influences the shape of the function estimated by SVR as well. This influence is illustrated in figure 6.5 for a two dimensional

[1)] Often the parameterization $\sigma = \sqrt{\frac{1}{2\gamma}}$ is used alternatively.

regression problem. For large values of γ the predicted points lie close to the underlying nonlinear **peaks** function. On the other hand the **peaks** function is not approximated well by the estimated function for a small value of γ since the entire predicted points lie close to plane in $\mathbb{R}^2$. It is important to note that in both cases the regularization and loss function parameters were fixed at $C = 100$ and $\varepsilon = 1e-03$ during training which guarantees that the observed differences are actually caused by changes of γ.

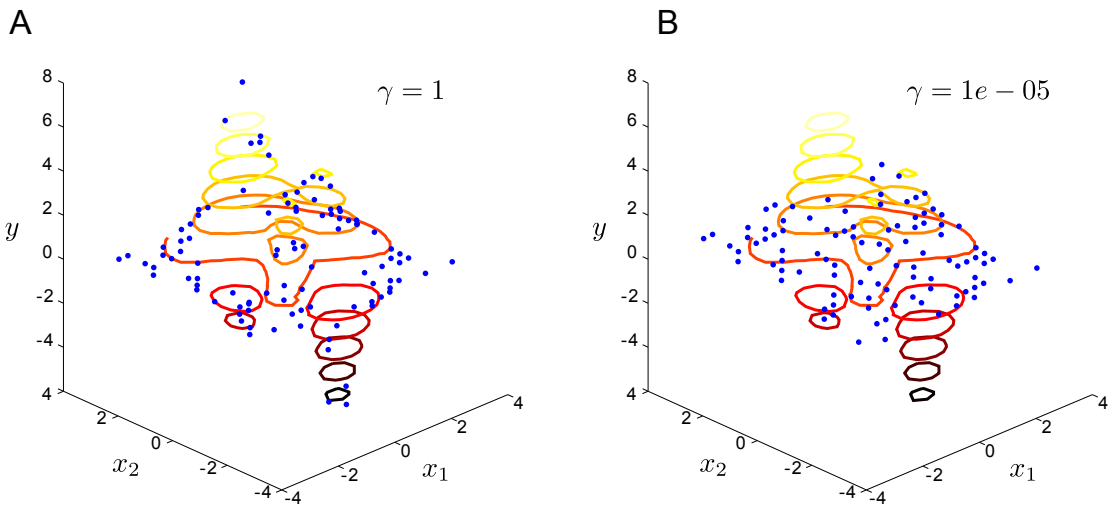


Figure 6.5: *Two dimensional toy regression problem with 200 data points generated by the* MATLAB® ***peaks*** *function, where 75% of the points were randomly selected for SVR training.* ***A:*** *By setting $\gamma = 1$, the estimated function is nonlinear and a good approximation to the underlying* ***peaks*** *function. The blue dots represent the predicted values on the test data set.* ***B:*** *The test set predictions lie close to a plane when $\gamma = 1e-05$, which means that the estimated function is almost linear.*

Contrary to the polynomial kernel the dimensionality of the feature space induced by the RBF kernel cannot be derived directly. Rather it is known that the kernel matrix K, having entries $K_{ij} = \exp(-\gamma\|x_i - x_j\|)$, has full rank as long as the input patterns are distinct [124]. Stated otherwise, the feature vectors $\phi(x_1), \ldots, \phi(x_m)$ are linearly independent and the RBF kernel evaluated on an infinite number of distinct training patterns hence induces a infinite dimensional feature space.

Another interesting property of the RBF kernel is its invariance to both, rotations of the input space and permutations of the input space coordinates. Let $R \in \mathbb{R}^n$ be an arbitrary rotation or permutation matrix and $x, z \in \mathbb{R}^n$ two input patterns. Since by definition $R^T R = I$ one has:

$$\begin{aligned} \|Rx - Rz\|^2 &= x^T R^T Rx - 2x^T R^T Rz + z^T R^T Rz = x^T x - 2x^T z + z^T z = \|x - z\|^2 \\ \langle Rx, Rz\rangle^d &= (x^T R^T Rz)^d = (x^T z)^d = \langle x, z\rangle^d \ . \end{aligned} \tag{6.7}$$

This means that, irrespective of rotations in the input space or changes in the order of input pattern components, the RBF as well as the polynomial kernel always yield the same value. On the one hand there are applications, like the recognition of letters, where the rotation invariance is desirable. On the other hand the rotation and permutation invariance can also discard valuable information about the learning problem. If, for example, the input patterns represent time series then the RBF and polynomial kernel ignore any temporal interdependency due to the permutation invariance.

The input patterns x, z in the adaptive stimulation experiments are time series that represent the pre- and post-stimulus brain signal (cf. figure 6.6). From the preceding discussion it is clear, that the RBF and polynomial kernel are not the best choice in this case since they discard any temporal information. To overcome this limitation the following section describes a kernel function for spike trains that takes the temporal structure of the input patterns into account.

Spikernel

The Spikernel is a kernel function specifically developed to provide a similarity measure between sequences of firing rates [129], and it builds on previous work on string kernels [86]. Before describing the structure of the Spikernel's feature vector it is necessary to establish some notation. First let $S, T \in \mathbb{R}^{q \times m}$ denote sequences of firing rates from q neurons, where $|S|$ is the length of sequence S. The firing rate of all neurons at time point i is denoted by S_i and the subsequence of firing rates from time point t_1 to t_2 is symbolized by $S_{t_1:t_2}$. Further, Ss is the concatenation of sequence S with the additional firing rate s and the distance between two sequences is measured by quadratic loss function $l_2(S_{\mathbf{i}}, U) = \sum_{k=1}^{n} (S_{i_k} - U_k)^2$. Similar to the polynomial kernel it is possible to explicitly specify the mapping to feature space:

$$\phi_U^n(S) = \sum_{\mathbf{i} \in I_{n,|S|}} \mu^{l_2(S_{\mathbf{i}}, U)} \nu^{|S| - i_1} \,, \tag{6.8}$$

where the summation occurs over all indices $\mathbf{i} = (i_1, \ldots, i_n) \in I_{n,l} = \{\mathbf{i} \in \mathbb{N}^n,\ 1 \leq i_1 < \ldots < i_n \leq l\}$ and the kernel parameters are $\mu, \nu \in [0, 1]$. In other words one coordinate of the feature vector in equation (6.8) corresponds to a sum over all n-long subsequences of S. When the sequence U is bin-wise similar to a subsequence of S the feature in equation (6.8) has a big value and the overall impact of the similarity measure is controlled by the parameter μ. In contrast to the RBF and polynomial kernel the Spikernel takes into account temporal information by considering subsequences. The second factor in equation (6.8) ensures that subsequences being far from the time point of interest i_1 are weighted less. The relative importance of this time point is regulated by ν. A simple example helps to derive a computable expression for the feature vector in (6.8).

For $n = 3$ and a subsequence length of $|S| = 5$ the index set of the sum is given by:

$$\begin{aligned} I_{3,5} = \{&(1\,2\,3),\\ &(1\,2\,4),(1\,3\,4),(2\,3\,4),\\ &(1\,2\,5),(1\,3\,5),(1\,4\,5),(2\,3\,5),(2\,4\,5),(3\,4\,5)\}\,. \end{aligned} \tag{6.9}$$

Evidently the sum over all subsequences of S can be split based on the last index i_3 and the feature vector is:

$$\begin{aligned} \phi_U^3(S) = \sum_{\mathbf{i}\in I_{3,5}} \mu^{l_2(S_\mathbf{i},U)}\nu^{5-i_1} =& \sum_{i_3=5} \nu\mu^{l_2(S_5,U_3)} \sum_{\mathbf{i}\in I_{2,4}} \mu^{d(s_i,U)}\nu^{4-i_1} + \\ & \sum_{i_3=4} \nu^2\mu^{l_2(S_4,U_3)} \sum_{\mathbf{i}\in I_{2,3}} \mu^{d(s_i,U)}\nu^{3-i_1} + \\ & \sum_{i_3=3} \nu^3\mu^{l_2(S_3,U_3)} \sum_{\mathbf{i}\in I_{2,2}} \mu^{d(s_i,U)}\nu^{2-i_1} \\ =& \sum_{i=1}^{|S|} \nu^{|S|-i+1}\mu^{l_2(S_i,U_n)} \sum_{\mathbf{i}'\in I_{2,|S_{1:i-1}|}} \mu^{l_2(S_{\mathbf{i}'},U_{1:n-1})}\nu^{|S_{1:i-1}|-i_1}\,. \end{aligned} \tag{6.10}$$

This splitting reveals the recursive nature of the feature vector as the last sum in equation (6.10) is equivalent to a feature mapping for S and U, with the last firing rate chopped off, and subsequences of length $n-1$. In general the feature mapping is given by the following recursive formula:

$$\phi_U^n(S) = \begin{cases} \sum_{i=1}^{|S|} \nu^{|S|-i+1}\mu^{l_2(S_i,U_n)}\phi_{U_{1:n-1}}^{n-1}(S_{1:i-1}), & 0 < n \leq |S| \\ 0, & |S| < n \wedge n > 0 \\ 1, & n = 0\,. \end{cases} \tag{6.11}$$

The kernel function for a fixed subsequence length n corresponds to the dot product between feature vectors and is given by:

$$\begin{aligned} k^n(S,T) = \phi^n(S)\cdot\phi^n(T) &= \int_{\mathbb{R}^{q\times n}} \phi_U^n(S)\phi_U^n(T)dU \\ &= \sum_{i=1}^{|S|}\sum_{j=1}^{|T|} \nu^{|S|+|T|-i-j+2} \int_{\mathbb{R}^q} \mu^{l_2(S_i,U_n)}\mu^{l_2(T_j,U_n)}dU_n \int_{\mathbb{R}^{q\times n-1}} \phi_{U_{1:n-1}}^{n-1}(S_{1:i-1})\phi_{U_{1:n-1}}^{n-1}(T_{1:j-1})dU_{1:n-1} \\ &= \sum_{i=1}^{|S|}\sum_{j=1}^{|T|} \nu^{|S|+|T|-i-j+2} k^*(S_i,T_j)k^{n-1}(S_{1:i-1},T_{1:j-1})\,. \end{aligned} \tag{6.12}$$

In equation (6.12) the recursive structure of equation (6.11) was exploited to simplify the expression. So far only subsequences of length n were considered. To take into account

all possible subsequence lengths the Spikernel is defined by $k(S,T) = \sum_{i=1}^{n} k^i(S,T)$. The Spikernel can be evaluated efficiently by dynamic programming [34], which avoids unnecessary computations of common sub-expressions:

$$\begin{aligned} k^n(Ss,Tt) =& \nu^2 k^*(s,t)k^{n-1}(S,T) \\ &+ \sum_{i=1}^{|S|}\sum_{j=1}^{|Tt|} \nu \cdot \nu^{|S|+|Tt|-i-j+2} k^*(S_i,Tt_j)k^{n-1}(S_{1:i-1},Tt_{1:j-1}) \\ &+ \sum_{i=1}^{|Ss|}\sum_{j=1}^{|T|} \nu \cdot \nu^{|Ss|+|T|-i-j+2} k^*(Ss_i,T_j)k^{n-1}(Ss_{1:i-1},T_{1:j-1}) \\ &- \sum_{i=1}^{|S|}\sum_{j=1}^{|T|} \nu^2 \cdot \nu^{|S|+|T|-i-j+2} k^*(S_i,T_j)k^{n-1}(S_{1:i-1},T_{1:j-1}) \\ =& \nu^2 k^*(s,t)k^{n-1}(S,T) + \nu(k^n(S,Tt) + k^n(Ss,T)) - \nu^2 k^n(S,T)\ . \end{aligned} \tag{6.13}$$

Apparently the evaluation of $k^n(Ss,Tt)$ is reducible to computation of $k^n(S,Tt)$, $k^n(Ss,T)$, $k^n(S,T)$, and $k^{n-1}(S,T)$. These expressions are stored in a dynamic programming table that is initialized with $k^0(S,T) = 1$ and $k^i(S,T) = 0,\ \forall i > \min\{|S|,|T|\}$, which follows from recursive definition of the feature vector in equation (6.11).

In conclusion the Spikernel allows exploiting the temporal information by considering features computed from subsequences of the input patterns. The extent of the time warps is controlled by the maximal subsequence length n, while the similarity between subsequences and the importance of the time point of interest are steered by parameters μ and ν respectively. Although the Spikernel was originally devised to operate on sequences of binned firing rates, it is equally applicable to sequences that represent the binned activity of LFPs.

6.2 Recording setup

The setup used to record data during the adaptive microstimulation experiments is outlined in figure 6.6. Here only the setup for recording and stimulation is described, while the discussion of the setup necessary for closed-loop stimulation is deferred to chapter 7. The recording of brain activity and delivery of electrical stimuli occurs simultaneously in the barrel cortex (cf. section 2.5) of anesthetized rats (figure 6.6A). One example of a stimulation trial is shown in figure 6.6B and consists of the recorded pre-stimulus signal denoted by x and the post-stimulus signal denoted by y. The particular shape of the post-stimulus signal y, also termed evoked potential, is known to be highly variable across stimulation trials [77] and is influenced by both, the ongoing brain activity [1] before the stimulus x, and the stimulus intensity itself. In the experiment the stimulus intensity, denoted by s, is adjusted by changing the amplitude of a bipolar rectangular current pulse as shown in figure 6.6D.

With the notation just introduced each stimulation trial is completely described the triple (x, y, s). To accomplish the closed-loop control of stimulus intensity s, it will be necessary to learn the relationship between x, y and s by SVR. Therefore each experiment starts with ten minutes of recording and stimulation to collect a set of training patterns (x, y, s). Stimulation pulses are delivered with a frequency of 1Hz and varying intensity, which amounts to a training data set comprising 600 patterns. During the first experiments the range of intensities suitable for adaptive microstimulation was still unknown. Previous studies of microstimulation in rat barrel cortex covered the whole range between 0.8 and 4.8nC [20].

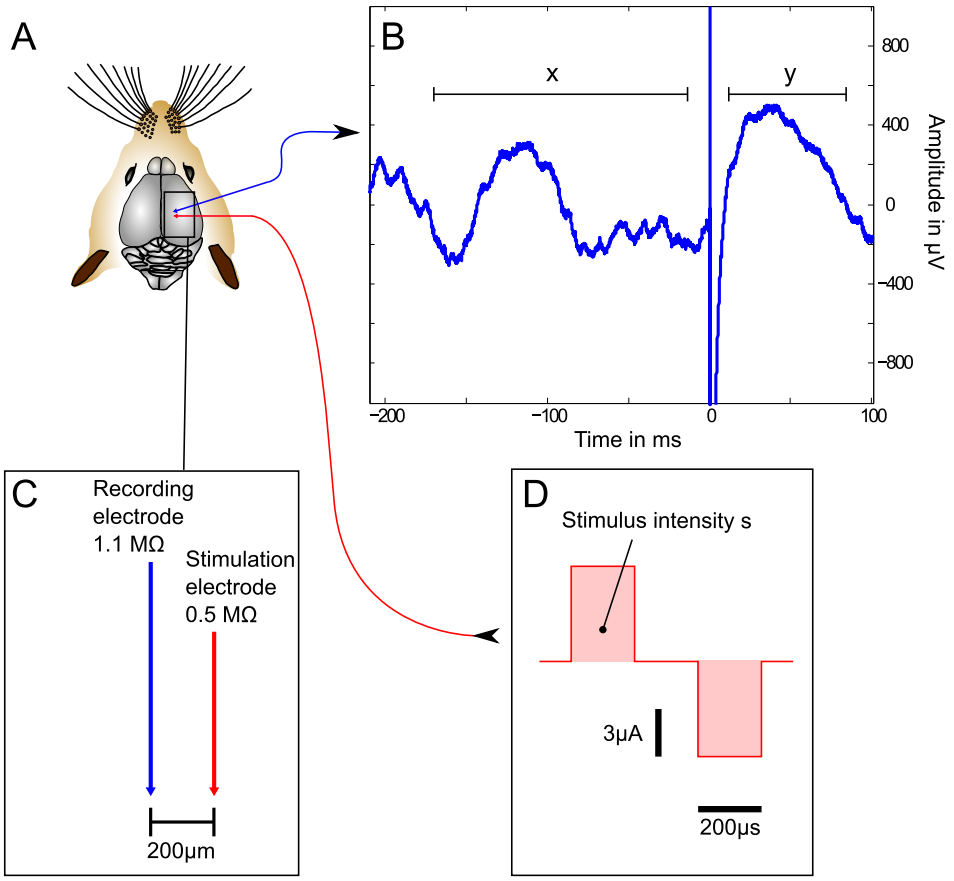


Figure 6.6: ***A:*** *Recording of the brain activity and stimulation occurs simultaneously in the barrel cortex of anesthetized rats.* ***B:*** *Raw data recorded during one stimulation trial. The pre-stimulus signal is denoted by x and the post-stimulus signal by y.* ***C:*** *The tips of recording and stimulation electrodes are separated by 200µm and the recording electrode has higher impedance than the stimulation electrode.* ***D:*** *Bipolar rectangular current pulses are used for stimulation. The stimulus intensity s is varied by changing the amplitude of the pulse and the duration of each pulse is fixed.*

Here three ranges for the stimulus intensity were examined separately: 0.8-1.6nC, 2.4-3.2nC, and 4.0-4.8nC. For the collection of a training data set the intensity s is sampled, uniform and randomly, from one of the ranges with a resolution of 0.025nC.

6.3 Formal problem definition

Given a target evoked potential y^* and the recorded brain activity x what stimulus intensity s^* is required to obtain an evoked response that is close to y^*? This question represents the core problem to be solved repeatedly during adaptive microstimulation. Without doubt it is necessary to learn a functional relationship between x, y and s by SVR training before it is possible to answer the question above. There are two different approaches to tackle this problem [17], termed direct and inverse solution.

6.3.1 Direct solution

For the direct solution a single scalar function $f : (x, y) \mapsto s$ is estimated by SVR. After fixing the second argument at the target evoked potential y^* the function can be used to predict the optimal intensity s^* in dependence of the ongoing brain activity x. This approach has the advantage that only a single function has to be estimated. In addition it is easy to extend this approach via the use of different kernel functions to estimate a nonlinear mappings. On the downside it is hard to evaluate the quality of function f, since it is only possible to get precision estimates for the predicted stimulus intensities, but not for the evoked potential. It is therefore difficult to make precise statements about the stabilization capability of the estimated function.

6.3.2 Inverse solution

The inverse solution models the relationship between pre- and post-stimulus signals by estimating a set of functions $\mathbf{f} : (x, s) \mapsto y$, an approach which seems to be more natural with respect to the goal of stabilizing the evoked potential. A function $g : (x, y^*) \mapsto s$, which allows to determine the optimal intensity s^* given the ongoing brain activity and the target potential, is found by partially inverting the set of estimated functions. Let $f_j(x') = \langle w_j, x' \rangle + b_j$ be the j-th function in the set of functions $\mathbf{f} : (x, s) \mapsto y$, where the vector $x' = (s\, x)$ is assumed to have the stimulus intensity as its first component and the weight vector $w_j = (w_j^s\, w_j)$ is partitioned accordingly. Now the function g is found by minimizing the l2-loss between $\mathbf{f}(x')$ and the target evoked potential y^* which leads to the following optimization problem:

$$\begin{aligned} \min_s \; & \frac{1}{2}\|\mathbf{f}(x') - y^*\|^2 \\ \text{subject to} \quad & l \leq s \leq u \;, \end{aligned} \tag{6.14}$$

where $l, u \in \mathbb{R}$ are the upper and lower bound for the stimulus intensities. Since the constraint is enforceable by simple projection to the interval $[l, u]$ the unconstrained optimization problem can be directly solved for s. Let

$$h(s) = \frac{1}{2} \sum_j (s w_j^s + \langle w_j, x \rangle + b_j - y_j^*)^2 \ . \tag{6.15}$$

Computing the derivative of equation (6.15) and setting to zero yields:

$$h'(s) = \sum_j (s w_j^s + \langle w_j, x \rangle + b_j - y_j^*) w_j^s \stackrel{!}{=} 0 \tag{6.16}$$

$$\Leftrightarrow s = \frac{\sum_j (y_j^* - \langle w_j, x \rangle - b_j) w_j^s}{\sum_j (w_j^s)^2} \ . \tag{6.17}$$

The value of s given in equation (6.16) is a local minimum since $h''(s) = \sum_j (w_j^s)^2 \geq 0$ and the desired function $g : (x, y^*) \mapsto s$ is given by evaluating the right hand side of equation (6.17).

Unlike the direct solution this approach allows to measure the precision with respect to the evoked potentials, but it requires estimation of a set of functions and additional computation time. Although it is feasible to use kernel functions for estimating nonlinear mappings with the inverse solution, it is necessary to compute pre-images [81] of the weight vector as an intermediate step. Thus the extension to nonlinear functions is not as straightforward as for the direct solution.

Until now the formal description used the variables x and y to refer to the pre- and post-stimulus signals without specifying how these signals are extracted from the recorded raw data. The direct and inverse solution will be evaluated in combination with different preprocessing methods for extracting the LFP (section 6.1.1), MUA (section 6.1.2), and the signal's phase (section 6.1.3) in section 6.5.1.

6.4 ANOVA kernel

The modified analysis of variance (ANOVA) kernel proposed in this section allows to incorporate prior knowledge about the temporal structure of pre- and post-stimulus signals and the time point of stimulation onset. Similar to the Spikernel (section 6.1.4) it overcomes the drawback of rotation invariance inherent to the RBF and linear kernel function, but additionally decreases the number of adjustable kernel parameters. While the Spikernel has three parameters (n, μ, ν) the ANOVA kernel has two: the monomial degree d, and μ for the weighting function. This reduces the tuning efforts during the training phase in the adaptive microstimulation experiments. The simple structure of the ANOVA kernel further permits explicit computation of the feature vector for low monomial degrees d which speeds up the kernel evaluations throughout the online prediction of stimulus intensities. Results

that compare the performance of RBF, polynomial, spike, and ANOVA kernel functions on data from the adaptive stimulation experiment will be presented in section 6.5.3.

The feature space induced by the ANOVA kernel is spanned by monomials analogous to the polynomial kernel. But unlike the polynomial kernel, the monomials in the ANOVA kernel exclude single features raised to a power greater than one [128]. Hence the feature vector can be expressed as:

$$\phi_{\mathbf{j}}(x) = \prod_{i=1}^{n} x_i^{j_i},\ \mathbf{j} \in \{0,1\}^n,\ \sum_{i=1}^{n} j_i = d\ , \tag{6.18}$$

where d is the maximal degree of the monomials. This definition is similar to equation (6.3) except for the restriction imposed on the index vector, which ensures that its entries are either zero or one. Using the explicit definition in (6.18) the dot product between mapped input patterns is given by:

$$\langle \phi_{\mathbf{j}}(x), \phi_{\mathbf{j}}(z) \rangle = \sum_{1 \le i_1 < \ldots < i_d \le n} x_{i_1} \cdot \ldots \cdot x_{i_d}\, z_{i_1} \cdot \ldots \cdot z_{i_d} = \sum_{1 \le i_1 < \ldots < i_d \le n} \prod_{k=1}^{d} x_{i_k} z_{i_k}\ . \tag{6.19}$$

Although the differences between the ANOVA and polynomial kernel are subtle, equation (6.19) shows that there is no closed form expression for the ANOVA kernel. Compared to the polynomial kernel evaluation of the ANOVA kernel consequently requires more computation time. This is somehow counter intuitive since the polynomial kernel implicitly considers all monomials with degree d and not just a subset like the ANOVA kernel. Each monomial in the ANOVA kernel is the product of d single features that are chosen from a set of n features. Accordingly the dimensionality of the feature space is $\binom{n}{d}$. For example, if $d = 5$ and $n = 20$ then the feature space has 15504 dimensions. This is about one third of the number of dimensions in the feature space induced by the polynomial kernel.

How can the ANOVA kernel be used to encapsulate prior knowledge about the adaptive stimulation problem? As opposed to the polynomial kernel, the ANOVA kernel allows individual weighting of single features. By giving monomials with nonconsecutive indices in the index vector $\mathbf{j}$ a lower weight one can therefore encode the knowledge that the inputs are time series. Further, by giving higher weight to single features that are close to a certain point in time, it is possible to incorporate knowledge about a time point of interest. A simple example that uses the proposed weighting scheme is shown in figure 6.7A.

In summary the weight $\omega_{ij} = \mu^{|i-\tau|+|j-\tau|+|i-j|}$ for the product term $x_i x_j$ captures both, the fact that the input pattern $x \in \mathbb{R}^n$ is a time series, and the importance of the time point τ. For the data from the adaptive stimulation experiments the time point of interest τ is chosen to coincide with the onset of the stimulation pulse. Intuitively this choice makes sense since the recorded brain signal and the evoked potential in the vicinity of the stimulus are expected to carry most of the information about the stimulus intensity. By analyzing the weight vector of the direct solution with linear kernel this intuition can be corroborated as features close to the stimulation pulse receive the largest absolute weight.

Decreasing the parameter μ in the ANOVA kernel has the effect, that the temporal structure of the input patterns and the time point of interest are emphasized, as shown in figure 6.7B. With the proposed weighting scheme the equation of the modified ANOVA kernel is given by:

$$\langle \phi_{\mathbf{j}}(x), \phi_{\mathbf{j}}(z) \rangle = \sum_{1 \leq i_1 < \ldots < i_d \leq n} \prod_{k=1}^{d-1} \mu^{i_{k+1} - i_k} \prod_{k=1}^{d} \mu^{|i_k - \tau|} \prod_{k=1}^{d} x_{i_k} z_{i_k} \ . \tag{6.20}$$

After introducing individual weights for the product terms the efficient evaluation of the ANOVA kernel is still possible by using dynamic programming [34]. The structure of the dynamic programming table is shown in figure 6.7C for the recursive evaluation of the kernel function up to third degree for four-dimensional input patterns $x, z \in \mathbb{R}^4$. The rows of the table are ordered by the increasing cumulative temporal distance of the product terms, e.g. the third row of the third table contains all possible products with three factors, $a_i a_j a_k$, where the temporal distances, $|i - j|$ and $|j - k|$, sum up to two, while the zeroth row of the second table does not contain any products, since all products of second degree are at least one time step apart. Along the columns the product terms are ordered according to the largest index of the constituent factors, e.g. the third column always contains product terms where the factor with the largest index is given by factor a_3.

The recursive evaluation of the modified ANOVA kernel initially starts with the table containing the products of degree one, that are ordered in the manner just described (figure 6.7C, table $d = 1$). In the subsequent steps of the recursive evaluation the table containing products with d factors is filled with the help of the table containing products with $d - 1$ factors and so on, e.g. the table with products of degree two (figure 6.7C, table $d = 2$) is constructed based on information stored in the initial table (figure 6.7C, table $d = 1$).

When creating a new table entry one only considers the entries in the previous table which lie on the "upper diagonal", or more precisely, entries which are p columns to the left and p rows up ($p = 1, 2, \ldots$). If the new table entry has column index j, it is constructed by multiplying the p-th entry on the diagonal in the previous table by $\mu^p a_j$ and summing up the results. The reason for multiplying by μ^p is that the p-the entry on the diagonal lies p columns to the left of the new entry and, as a consequence of the column ordering by largest index, has a temporal distance of p to the new factor a_j. Figure 6.7C exemplifies the construction process for the entry in the third row and fourth column of the last table ($d = 3$), which is calculated based on the entries in the second table ($d = 2$), highlighted in blue. A formal description for evaluating the modified ANOVA kernel is provided by algorithm 5 in appendix A.

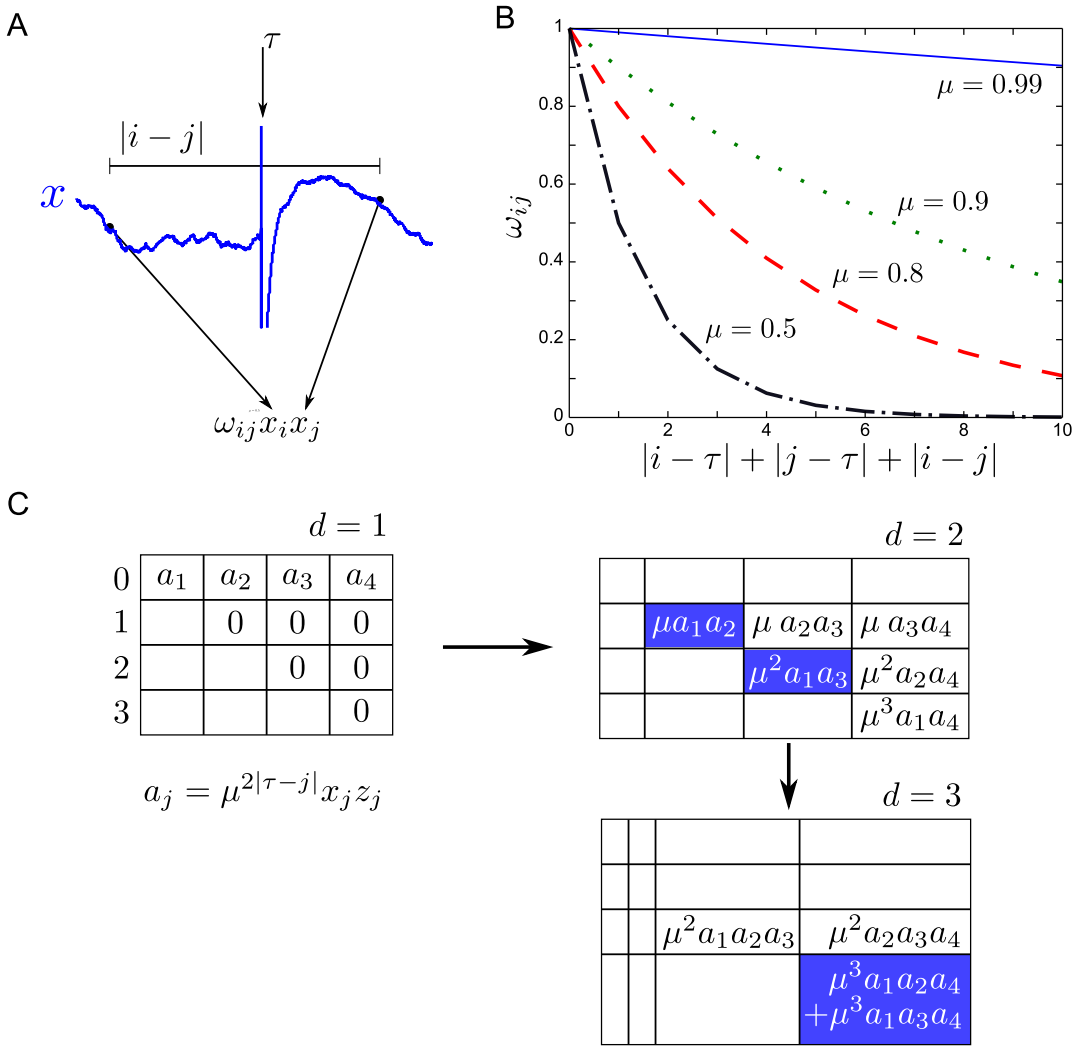


Figure 6.7: *Example application of the ANOVA kernel.* ***A:*** *The input pattern x corresponds to a single stimulation trial and the elements x_i and x_j to the amplitudes of the signal at time points i and j. The weight ω_{ij} for the monomial feature $x_i x_j$ is determined by the power function $\omega_{ij} = \mu^{|i-\tau|+|j-\tau|+|i-j|}$, which encodes the temporal distance between the amplitudes x_i and x_j and their temporal distance from the time point of interest τ.* ***B:*** *The power function for different base weights μ.* ***C:*** *Dynamic programming tables for evaluating the ANOVA kernel up to third degree for input patterns $x, z \in \mathbb{R}^4$.*

6.5 Results

The data collected in the adaptive stimulation experiments consists of 15 data sets with stimulus intensities in the range of 0.8-1.6nC and 5 data sets for each of the other ranges, namely 2.4-3.2nC and 4.0-4.8nC. This section presents results for the different techniques previously described in this chapter on all of these data sets. In particular, section 6.5.1 compares the prediction accuracy of the direct and inverse solution and section 6.5.2 presents optimal time windows for pre- and post-stimulus LFPs. Finally, results of linear, RBF, polynomial, and ANOVA kernels, as well as the Spikernel, are given in section 6.5.3.

6.5.1 Comparison of direct and inverse solutions

Each of the data sets from the adaptive stimulation experiments contains 600 patterns. For the SVR training with linear kernel, required by both direct and inverse solution, the available patterns are equally divided into training and test data sets. The performance of both approaches is measured on unseen test data by the root mean squared error (RMSE) between predicted and actual stimulus intensities. The RMSE is computed from the MSE in equation (3.48) by taking the square root and makes the results easier to interpret since the errors are then in units of nC. Analogous to the results presented in chapter 4, 95% confidence intervals are determined for the RMSE by using the bootstrap method [41] with $B = 1000$ bootstrap samples.

In order to study the information extraction methods of section 6.1 in conjunction with the direct and inverse solution the duration of the pre-stimulus signals is fixed at 500ms. The duration of the post-stimulus signal is 100ms for the direct solution and either 20ms or 100ms for the inverse solution. Although signals are down sampled to 500Hz following the preprocessing of the raw signal, the dimensionality of the input patterns is still very high. The combined pre- and post-stimulus signal for the direct solution for instance has 300 dimensions. To reduce the dimensionality and additionally remove redundant information the signals are optionally averaged over bins of 100ms length or projected to a PCA [72] subspace. For the PCA projection 50 components are retained for the pre-stimulus signal and 10 components for the post-stimulus signals. Keeping the components with the largest eigenvalue ensures that most of the variance in the pre- and post-stimulus signals is explained by the subspace (table 6.1). After extracting the LFP spectral decompositions are computed in the α- (1-12Hz), β- (13-30Hz), γ- (31-60Hz), and γ_h-band (61-100Hz). The definition of these frequency bands is taken from [133].

The regularization parameter λ and the loss function parameter ε for the SVR training are selected by minimizing the MSP bound as described in chapter 5. Since the stimulus intensities are directly predicted in the direct solution it is admissible to fix parameter ε at the measured noise level (0.002nC) of the stimulator, while the regularization parameter λ is again optimized via the MSP bound.

Figure 6.8 shows the results obtained with the direct and inverse solution in combination

	Direct	Inverse
pca-lfp	$99.2 \pm 0.4\%$	$99.4 \pm 0.4\%$
pca-mua	$85.0 \pm 4.3\%$	$81.4 \pm 6.2\%$
pca20	-	100%

Table 6.1: *Variance, in per cent that is retained after projecting LFPs and MUA to the PCA subspace. Each table entry shows the mean and standard deviation computed across all data sets.*

with different information extraction methods on the data set **fb131208-r5**. Overall the predicted stimulus intensities of the direct solution are more accurate than those predicted by the inverse solution. With respect to the different extraction methods the direct solution works best on the LFP and its projections, or a combination of LFP and MUA. The inverse solution achieves the best results with MUA or binned LFPs. Obviously not suited for the adaptive stimulation problem are the various spectral decompositions of the LFP and the instantaneous phase of the signal.

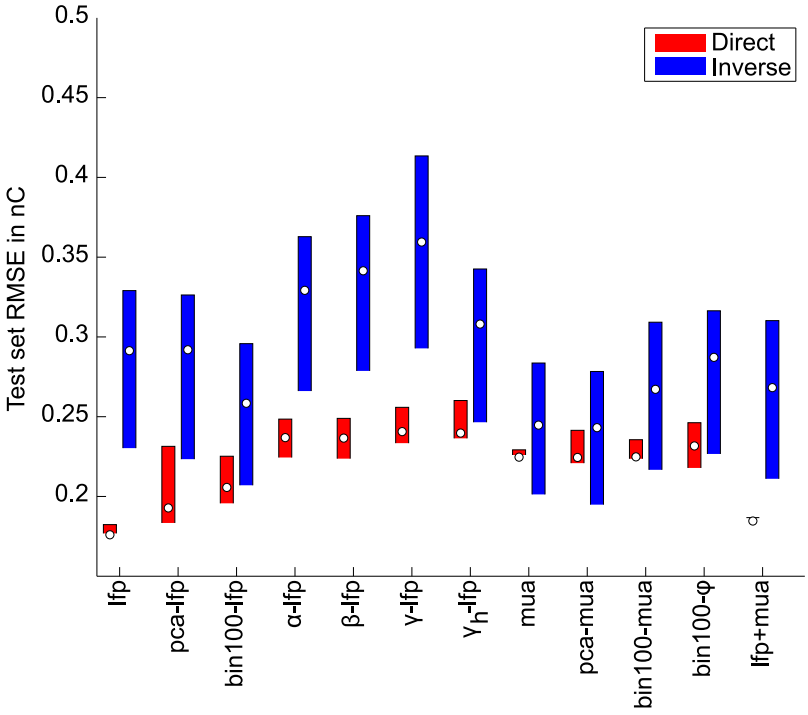


Figure 6.8: *Results of the direct and inverse solution for the **fb131208-r5** data set in combination with different preprocessing methods. Overall the direct solution achieves more accurate predictions than the inverse solution. The LFP and the MUA contain sufficient information for the prediction task. The phase and spectral decomposition of the LFP are less suitable to describe the current brain state in the context of adaptive stimulation.*

The assertions made for the **fb131208-r5** data set carry over to the other data sets as well. Table 6.2 lists the best extraction method and test set RMSE for the direct and inverse solution on the remaining data sets. It can be seen that across all data sets the direct solution provides more accurate predictions than the inverse solution. With exception of the **fb-031208-r8** data set, the direct solution can glean most of the information from the LFP, its PCA projection, or a combination of LFP and MUA.

		Direct		Inverse		
Intensity range	Data set	x,y	RMSE	x	y	RMSE
0.8-1.6nC	fb-180708-r1	lfp	0.14	bin100-lfp	pca20	0.22
	fb-081008-r1	lfp+mua	0.12	lfp+mua	pca20	0.14
	fb-081008-r6	lfp	0.15	bin100-lfp	pca20	0.22
	fb-081008-r8	lfp	0.12	bin100-lfp	pca20	0.12
	fb-141008-r2	lfp+mua	0.10	lfp+mua	pca20	0.11
	fb-151008-r2	lfp	0.06	lfp+mua	bin100	0.11
	fb-151008-r4	lfp+mua	0.08	lfp+mua	pca20	0.10
	fb-151008-r6	lfp	0.07	bin100-mua	pca20	0.12
	fb-151008-r8	lfp	0.12	bin100-lfp	pca20	0.15
	fb-031208-r2	pca-lfp	0.08	lfp	pca20	0.09
	fb-031208-r6	lfp	0.14	lfp+mua	pca20	0.17
	fb-041208-r4	lfp	0.11	lfp+mua	pca20	0.13
	fb-131208-r2	pca-lfp	0.12	bin100-lfp	pca20	0.14
	fb-131208-r5	lfp	0.18	pca-mua	pca20	0.24
	fb-131208-r7	lfp	0.13	lfp+mua	pca20	0.20
2.4-3.2nC	fb-031208-r3	lfp	0.16	bin100-lfp	pca20	0.21
	fb-031208-r7	lfp	0.17	bin100-mua	pca20	0.19
	fb-041208-r5	lfp	0.14	lfp+mua	pca20	0.15
	fb-131208-r3	lfp+mua	0.19	pca-mua	pca20	0.25
	fb-131208-r8	pca-lfp	0.17	lfp+mua	pca20	0.21
4.0-4.8nC	fb-031208-r4	pca-lfp	0.22	bin100-lfp	pca20	0.44
	fb-031208-r8	pca-mua	0.22	bin100-lfp	pca20	0.48
	fb-041208-r6	pca-lfp	0.21	bin100-lfp	pca20	0.35
	fb-131208-r4	pca-lfp	0.23	mua	pca20	0.43
	fb-131208-r9	pca-lfp	0.21	bin100-mua	pca20	0.35

Table 6.2: *Best preprocessing methods and test set RMSE in nC for the direct and inverse solution on all data sets recorded in the adaptive stimulation experiments. The pre- and post-stimulus signals are denoted by x and y respectively.*

The overall picture is similar for the inverse solution, although it relies more often on information from the MUA. The PCA projection of the first 20ms of the evoked potential consistently works best across all data sets for the inverse solution, the only exception being the **fb-151008-r2** data set.

By comparing the results in table 6.2 with respect to the intensity range it becomes apparent that predictions are most accurate for the lowest intensity range of 0.8-1.6nC and decrease in precision for the higher intensity ranges. Since the direct solution performs best, the inverse solution will not be considered any further in the following subsections.

6.5.2 Optimal time windows

Up to this point the direct solution used fixed time windows for the pre- and post-stimulus LFPs. This section deals with the question whether there are optimal time windows that

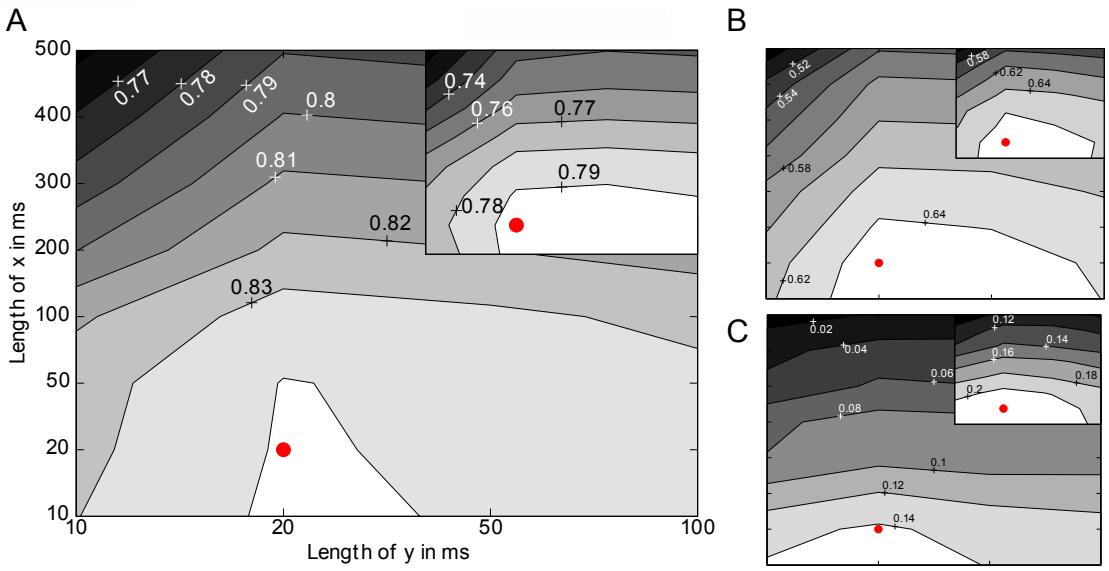


Figure 6.9: *Optimal time windows for the direct solution with linear SVR.* ***A:*** *The mean squared correlation coefficient R^2 in dependence of different window sizes for the pre- and post-stimulus signal (x and y) of the* ***fb081008-r1*** *data set. The maximum standard deviation (SD) of the R^2 estimates is* 0.02 *and the red dot marks the optimal time window – 20ms of pre- and post-stimulus data. This result is consistent across all (n=15) data sets, as shown by the population average in the inset. Stimulus intensities are in the range of 0.8-1.6nC.* ***B:*** *Same as in* ***A*** *but for the range of 2.4-3.2nC and data set* ***fb-031208-r7*** *(maximum SD=0.05).* ***C:*** *Same as in* ***A*** *but for the range of 4.0-4.8nC and data set* ***fb-031208-r8*** *(maximum SD=0.03).*

can be consistently used across all data sets. Here the squared correlation coefficient R^2

between predicted and true intensities is used to measure the performance[2]. To get an estimate of the R^2 value on the test set it is averaged across ten random splits of the available patterns into 50% training and test data. The time windows for the pre-stimulus signal x are varied between 10 and 500ms, and for the post-stimulus signal y between 10 and 100ms.

Figure 6.9A shows the average test set R^2 in dependence of the window length for the range of 0.8-1.6nC on the **fb081008-r1** data set. For this data set the optimal time window has a length of 20ms for both pre- and post-stimulus LFPs. This result also holds across all data sets for this intensity range, as shown by the population average in the inset of figure 6.9A. It is surprising that exactly the same time windows are optimal even for the other intensity ranges as shown by the results in figure 6.9B and C. In conclusion there is a definite answer for the question posed at the beginning of this section: 20ms of pre- and post-stimulus LFPs yield the best results for the direct solution with linear kernel. These optimal time windows form the basis for the comparison of different kernel functions described in the following section.

6.5.3 Comparison of kernel functions

The direct solution so far employed the linear kernel to estimate the function $f : (x, y) \mapsto s$. But what kernel functions are suitable to model a possibly nonlinear relationship between x, y and s? To answer this question and compare the different kernel functions described in section 6.1.4 and section 6.4, the RMSE on the test set is used as a performance measure. Like in section 6.5.2 the RMSE is averaged across ten random splits of the available patterns into 50% training and test data. For the SVR training the loss function parameter is fixed at $\varepsilon = 0.002$ and the regularization parameter is set to $\lambda = 0.005$ during the search for optimal kernel parameters. The kernel parameters and search ranges are given in table 6.3.

Kernel function	Parameters	No. of combinations
Linear	-	-
RBF	$\gamma \in \ln\{-12, -11, \ldots, 12\}$	25
Polynomial	$d \in \{2, 3, \ldots, 14\}$	13
Spikernel	$n \in \{5, 10, 15\}$ $\mu \in \{0.99, 0.9, 0.8, 0.7\}$ $\nu = \{0.99, 0.9, 0.7, 0.5\}$	48
ANOVA	$d \in \{2, 4, \ldots, \min\{14, \lvert x\rvert + 1\}\}$ $\mu \in \{0.99, 0.9, 0.8, 0.7, 0.6, 0.5\}$	≤ 42

Table 6.3: *Ranges for the kernel function parameters searched by tenfold cross-validation. The last column shows the total number of combinations to be tested during the search.*

[2] $R^2 \in [0, 1]$ and corresponds to the fraction of variance in the intensities explained by the predictions.

After optimal kernel parameters have been found the regularization parameter is again optimized via the MSP bound as described in chapter 5. It is important to note that the MSP bound cannot be used to optimize all parameters of the ANOVA kernel function and the Spikernel, since this would involve solving a combinatorial optimization problem with respect to the parameters d and n.

Figure 6.10 shows the performance of the ANOVA kernel relative to the other kernel functions for all data sets with stimulus intensities in the range of 0.8-1.6nC. Points below the diagonal are data sets where the ANOVA kernel achieves a lower prediction error on the test set. Clearly, the ANOVA kernel outperforms the polynomial kernel with exception of one data set. The same statement can be made with respect to the linear kernel, although the performance difference is smaller. Further, there are only slight differences between the ANOVA, RBF, and Spikernel across all data sets.

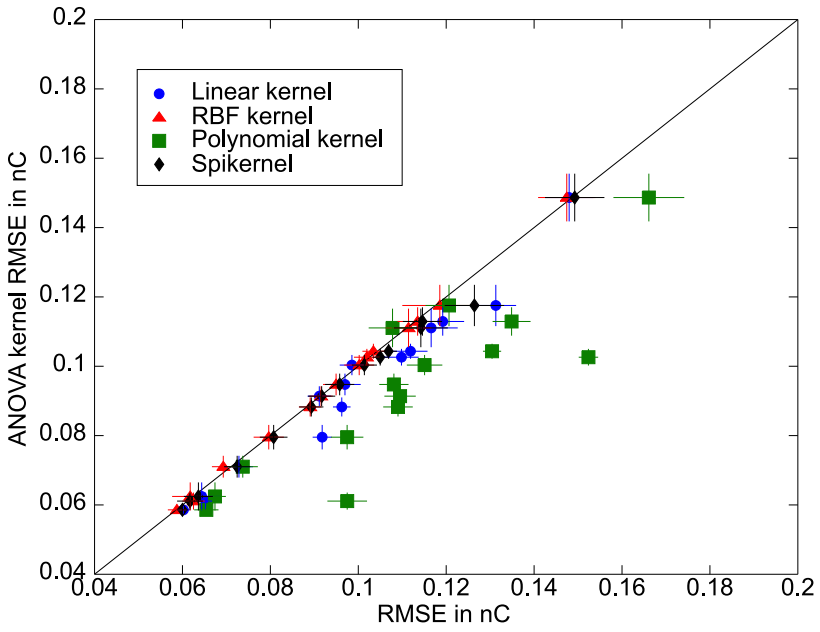


Figure 6.10: *Performance of the ANOVA kernel relative to the other kernel functions for the range of 0.8-1.6nC. Symbols represent the average RMSE on the test set, and lines indicate the standard deviation. The performance for linear, RBF, and polynomial kernel, as well as the Spikernel are plotted against the performance of the ANOVA kernel. Points below the diagonal are data sets where the ANOVA kernel performs better than the corresponding alternative kernel function.*

The observations just made for the range of 0.8-1.6nC are equally valid for the other ranges, as shown in figure 6.11A and B. Comparing the results across all intensity ranges shows, that predictions are less accurate for the higher ranges, which is consistent with the findings

of section 6.5.1.

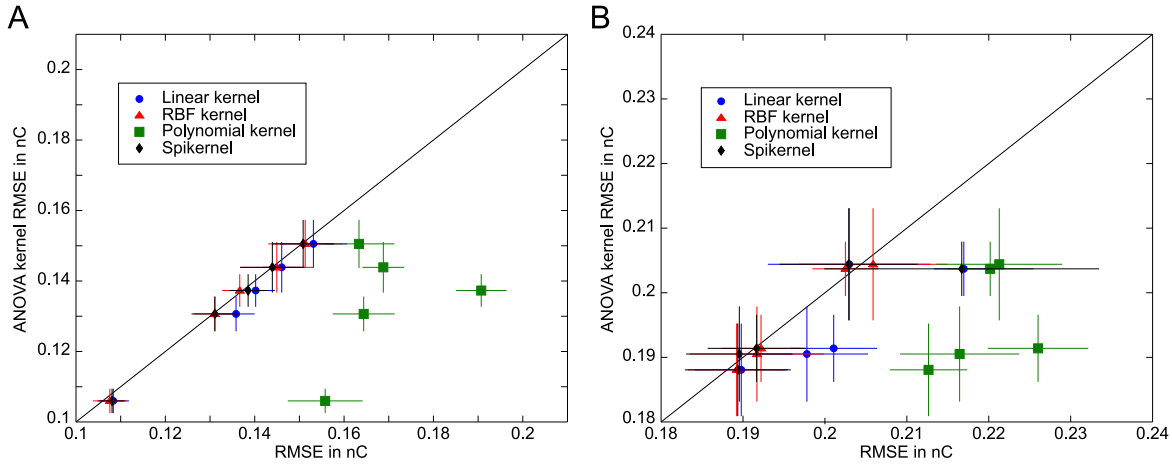


Figure 6.11: *Performance of the ANOVA kernel relative to other kernel functions. Symbols represent the average RMSE on the test set, and lines indicate the standard deviation.* ***A:*** *For the range of 2.4-3.2nC.* ***B:*** *For the range of 4.0-4.8nC.*

In conclusion the analysis presented in this section implies that the direct solution approach yields the most accurate prediction of optimal stimulus intensities. For this approach the cortical state is best represented by the LFP or a combination of both LFP and MUA, although the performance gain is small when MUA is used in addition to the LFP. On all analyzed data sets the optimal time windows for pre- and post-stimulus signals are short (20ms). This indicates a rapidly changing cortical state, at least with respect to the given objective of stabilizing cortical potentials. Further, the prediction errors of both direct and inverse solution are smallest in the lowest intensity range of 0.8-1.6nC – a discovery corroborated by the results for closed loop stimulation given in chapter 7.

Combined with the ANOVA kernel function, proposed in section 6.4, the prediction accuracy of the direct solution is better in comparison to the linear and polynomial kernel. The ANOVA kernel does not outperform the RBF and Spikernel, but it still is favorable when stimulus intensities are predicted in real time during adaptive microstimulation. Across all data sets the optimal monomial degree of the ANOVA kernel is low $d = 2$, or $d = 3$. This gives the opportunity to explicitly compute the feature vector in advance and use SVR with the linear kernel instead. With this strategy it is not necessary to evaluate the ANOVA kernel by dynamic programming for the prediction of optimal stimulus intensities. It is important to note that this approach is not feasible in conjunction with the RBF kernel since there exists no explicit expression for the feature vector. For the Spikernel equation (6.11) describes the feature vector, but its evaluation is elaborate and requires dynamic programming even for small values of the maximal subsequence length n. Therefore the direct solution combined with the ANOVA kernel is used in chapter 7 for predicting optimal stimulus intensities.

Ce qui embellit le désert
dit le Petit Prince
c'est qu'il cache un puits quelque part...
Antoine de Saint-Exupéry (1900-1944)

7 Adaptive microstimulation

While the last chapter investigated the best way to describe the relationship between pre-/post-stimulus activity and the stimulus intensity by using SVR on data collected offline, this chapter is concerned with the question whether the SVR model can be employed to control the stimulus intensity in an online feedback system in real time. To recapitulate the aim of adaptive microstimulation, figure 7.1 illustrates the stimulus trains and evoked cortical potentials under open and closed loop conditions.

In the open loop condition the stimulus intensity is held fixed irrespective of the ongoing brain activity and the evoked potentials are highly variable. When the control loop is closed, by using SVR to predict the optimal stimulus intensities for a given target response, it is expected that the evoked potentials can be stabilized and are less variable. Until now

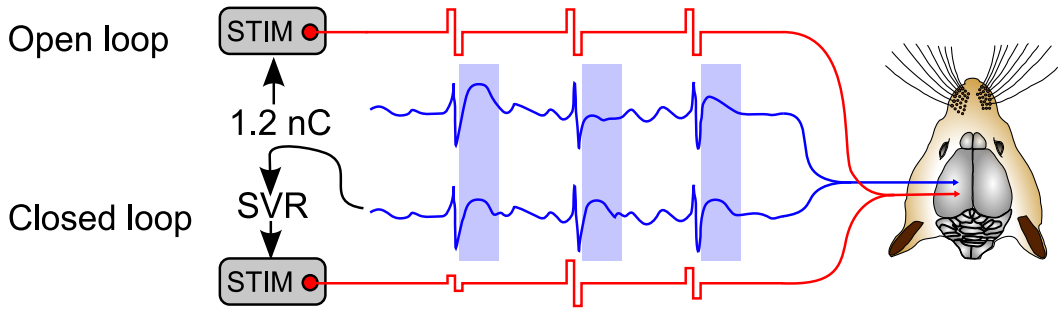


Figure 7.1: *Open versus closed loop stimulation. The stimulus intensity is fixed under the open loop condition and evoked potentials are highly variable. By closing the feedback loop optimal stimulus intensities with respect to a target potential are predicted by SVR and the evoked potentials are expected to be less variable.*

there have been no previous attempts to establish the envisaged online feedback system. Although the underlying idea is simple, the feedback system is challenging from a technical point of view. More details on technical aspects will be given in section 7.2. For now it is important to keep in mind that predicted stimulus intensities can change rapidly within

10-50ms and that it is necessary to update the intensities at a rate of 100Hz.

Closed loop control has been previously established in the context of functional electrical stimulation [69] and the stimulation of cell cultures [147, 38]. The system proposed for functional electrical stimulation could potentially fulfill the stringent timing requirements of adaptive stimulation, but it uses an explicit model for the knee joint in its control algorithm – an approach that is not feasible for adaptive stimulation since one would have to explicitly model whole parts of the brain. The activity in whole cortical columns in rat barrel cortex can of course be simulated, as shown in [89], but this is only possible by using huge amounts of computational resources, not to mention the additional resources consumed by a control algorithm that is build on top of the simulation.

For the closed loop stimulation of cardiac cell cultures described in [147], stimulus parameters are changed based on neural responses recorded over time periods of several seconds. This time scale is too coarse for adaptive microstimulation, since the evoked potentials to be controlled only last up to 100ms. In the work of [38] cultured neurons from rats are used to steer the behavior of a computer generated animal that lives in a virtual world. Sensory information about the virtual environment of the animal is routed back to the cell culture by stimulating the neurons. The parameters of the stimulus are fixed but the spatial location of the stimulation electrode is updated every 100ms. Later work used the same 10Hz update rate to maintain a certain level of spike bursts in cultured neurons by using a fixed rule to adjust the stimulation voltage [144].

All of these studies either utilize explicit models or simple fixed rules to control stimulus parameters. For adaptive microstimulation SVR is applied to estimate an appropriate function for controlling the stimulus intensity from recorded data, as described in chapter 6. A similar data driven approach has proven to be viable for controlling the width of stimulation pulses in the peripheral nervous system by using a neural network [12]. This work showed that modulations of the pulse width could be used to steer muscle contractions and thus achieve a predefined rotational angle of a pig's limb.

The experimental setup and control conditions used for studying closed loop stimulation are described in section 7.1. An overview of technical problems that had to be solved to establish the online feedback system is given in section 7.2. Finally section 7.3 presents results that quantify the extent to which adaptive microstimulation can stabilize evoked cortical potentials.

7.1 Experimental setup

An overview of the experimental setup is shown in figure 7.2. Raw signals are recorded from rat barrel cortex over the PCI interface by a data acquisition board (Multichannel Systems, ME128) on a computer running a real time operating system (Interval Zero, ETS). Optimal stimulus intensities predicted by SVR are delivered via a stimulus generator (Multichannel Systems, STG2008) that is programmed over the serial RS-232 interface. Communication

with a host computer that controls and monitors the real time operating system occurs over the Ethernet interface.

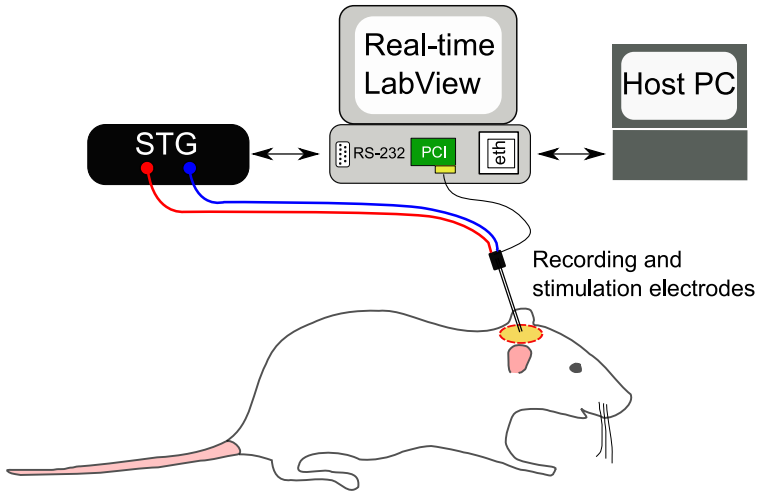


Figure 7.2: *Experimental setup for closed loop stimulation. Data acquisition and programming of the stimulus generator (STG) are handled by the real time operating system provided by the* LabView® *real time module. The setup is monitored and controlled by a host computer connected via the Ethernet interface.*

Each experimental session consists of two parts. In the first part data for the SVR training are collected by delivering stimuli with a frequency of 1Hz and randomly chosen intensity from on of three ranges (0.8-1.6nC, 2.4-3.2nC, and 4.0-4.8nC). This part is identical to the data collection described in chapter 6. The stimulation with random intensity lasts for ten minutes and yields a data set with 600 training patterns. For the second part of the experimental session the trained SVR model is transferred to the real time computer to predict optimal stimulus intensities for a predefined target evoked potential. The target evoked potential is extracted from the data collected in the first part of the experiment and corresponds to the average evoked potential for one stimulus intensity. In particular this intensity is 1.2nC for the 0.8-1.6nC range, 2.8nC for the 2.4-3.2nC range, and 4.4nC for the 4.0-4.8nC range.

The SVR model is constructed by using the ANOVA kernel described in section 6.4. The most accurate predictions with the ANOVA kernel are obtained with a monomial degree of ($d = 2$, or $d = 3$) and a base weight of $\mu = 0.9$ as revealed by a previously conducted analysis of offline data. The time point of interest, represented by the remaining kernel parameter τ, is chosen to coincide with the onset of the stimulation pulse. Furthermore the loss function parameter ε is fixed at the measured jitter of the STG and the optimal regularization parameter λ is found by minimizing the MSP bound as described in chapter 5. With these settings it takes approximately two minutes to train the SVR model.

When the ANOVA kernel is evaluated by the dynamic programming approach of section 6.4 it is not possible to predict optimal stimulus intensities in less than 10ms. Fortunately the optimal degree of the monomials is small ($d = 2$, or $d = 3$). It is therefore feasible to explicitly compute the mapping to the feature space induced by the ANOVA kernel and use the linear SVR algorithm instead. With this strategy optimal stimulus intensities can be updated efficiently at a rate of 100Hz.

During the second part of one experimental session the stimuli are again delivered at a rate of 1Hz but the intensity is set according to one of three conditions. Under the first condition, termed constant, the intensity is set to a constant value of 1.2, 2.8, or 4.4nC for each of the ranges respectively. This condition is important in order to quantify the stabilization of evoked potentials that can be achieved by adaptive stimulation. The intensities under the second condition, termed adaptive, are set to the value predicted by the SVR algorithm.

In four animals stimulation under the third control condition, termed noise, was similar to the adaptive condition but with the input x to the SVR algorithm replaced with white Gaussian noise matching the amplitude of the ongoing activity. Unfortunately the distribution of stimulus intensities differs between noise and adaptive stimulation trials as shown in figure 7.3A for the example presented in figure 7.8. Therefore stimulation trials under the noise condition cannot rule out the possibility that stabilization effects observed under adaptive stimulation are solely caused by the peculiar distribution of stimulus intensities.

To additionally exclude distributional influences the noise control was replaced in the three remaining animals by a control condition, termed shifted, which sets the intensities to the value determined during the last adaptive trial. Figure 7.3B shows the distribution of stimulus intensities for the example presented in figure 7.7. In contrast to the noise control the distribution of stimulus intensities under the shifted and adaptive conditions are almost identical with small differences caused by the block-wise randomization of control conditions.

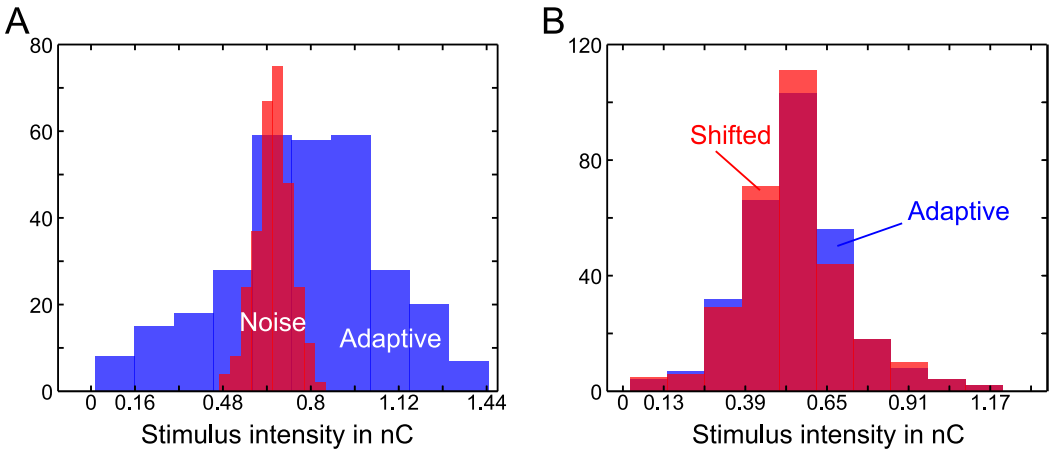


Figure 7.3: *Stimulus intensity histograms.* ***A:*** *Distribution of stimulus intensities under the adaptive and noise condition for the closed loop experiment shown in figure 7.8. The two distribution differ substantially and intensities emitted under the noise condition cover only a sub-range of intensities emitted by adaptive stimulation.* ***B:*** *Replacing the intensities of the noise control condition by shifted intensities guarantees almost identical distributions.*

Of course one could alternatively use a random permutation of adapted stimulus intensities to exclude distributional effects, but this is not practicable since the three stimulation conditions are applied in block-wise randomized order. The block-wise randomization is important to eliminate temporal influences on the stabilization effect. For example, it might be easier to stabilize the evoked potential at the beginning of the experimental session. It is important to note that the shifted intensities might be temporally correlated to the adapted intensities, which makes this kind of control condition in some sense stronger than a condition that would use a permutation of adapted intensities. The time delay between shifted and adapted intensity values can vary between one and four seconds due to the block-wise randomization.[1)]

The closed loop stimulation is investigated in an experimental group comprising seven Sprague-Dawley rats of both sexes with a body weight between 250 and 400 grams. The animals are anesthetized with an intraperitoneally administered mixture of ketamine (100mg/kg) and xylacine (20mg/kg), and the body temperature is maintained at 37°C with a heating pad. Additional doses of ketamine are used to keep the hind-paw reflex below threshold and preserve the level of anesthesia. For the craniotomy over somatosensory cortex the animal is placed in a stereotaxic apparatus and after removal of the dura mater the electrode array is inserted perpendicular to the cortical surface by a hydraulic micropositioner [18]. The cortical depth of recordings varied between 200 and 1000μm for different experimental sessions.

7.2 Technical considerations

As shown in figure 7.2 the data acquisition occurs over the PCI interface. For the PCI card (Multichannel Systems, ME-128) used in the experiments there were no drivers available for LabView®. In order to enable instant access to recorded data under the real time operating system a driver was implemented in user space with the infrastructure provided by the virtual instruments software architecture [108]. After configuring the data acquisition card, by setting the sampling frequency, number of recording channels, etc., the driver can initiate the recording and has to subsequently handle the interrupts of the device that indicate the arrival of the next data packet. Each of these interrupts, that occur every 10ms, is served through the driver by setting up a DMA transfer that copies the sampled raw data to a location in RAM where it is accessible to the user program. The end of the DMA transfer is signaled by another interrupt that is handled by the driver. During the DMA transfer the CPU is not locked and other threads can execute and use the available CPU time. By giving the driver thread the highest priority on an operating system with preemptive scheduling policy [134], like Interval Zero's ETS operating system, it can be ensured that hardware interrupts are always served on time.

[1)] This becomes clear by considering the following sequences of conditions: $\ldots \mid a\ s\ c \mid \ldots$, $\ldots \mid a\ c\ s \mid \ldots$, $\ldots \mid a\ s\ c \mid s \ldots$, and $\ldots \mid a\ s\ c \mid c\ s \ldots$, where the vertical bars symbolize the borders of one block.

In addition to the data acquisition it is necessary to compute the SVR prediction, program the stimulus generator, visualize the recorded data, and perform disk I/O in the real time environment. These tasks are executed asynchronously by separate threads that exchange data over queues, as shown in figure 7.4. With this architecture it is possible to compute optimal stimulus intensities instantly after arrival of the next data packet and thus sustain an update rate of 100Hz.

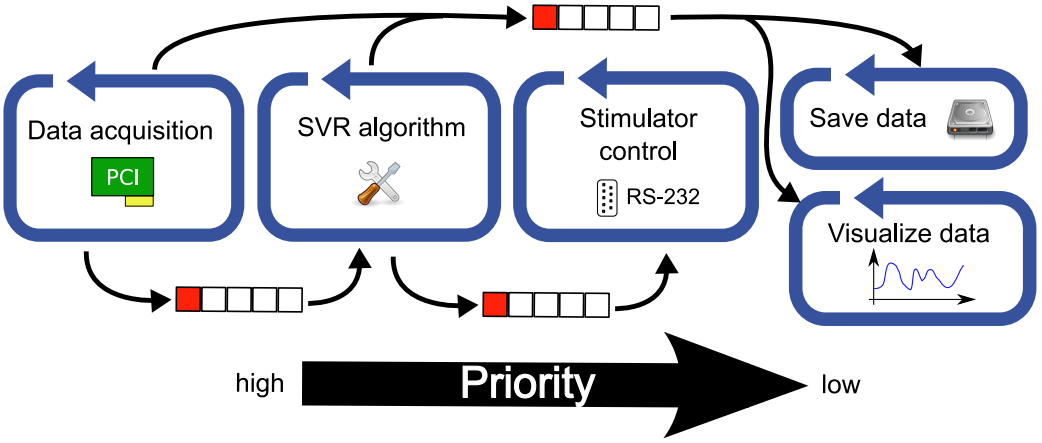


Figure 7.4: *Under the* LabView® *real time operating system data acquisition, SVR prediction, programming of the STG, data I/O, and visualization are handled each by a separate thread. To enable the asynchronous execution of these threads communication occurs over queues. The data acquisition thread has the highest priority to avoid delays in handling the interrupt that signals the arrival of new data.*

The stimulus intensities predicted by SVR during the closed loop stimulation in one of the experimental sessions is shown in figure 7.5. It can be seen that the intensity values change rapidly on a millisecond time scale. Consequently any delays incurred by programming the STG and the hardware itself are detrimental for adaptive stimulation. Initial experiments were conducted with the STG1008 (Multichannel Systems) that is real time capable due to its RS-232 interface. Unfortunately the measured delay between programming the device and the output of the stimulation pulse can last up to 300ms. To overcome this problem an RS-232 interface was developed for a single channel stimulator (A368, World Precision Instruments) that allowed programming the width and amplitude of rectangular pulses with short time delays of 300-700μs. But this custom solution had to be abandoned since the analog hardware of this stimulator could not produce clean rectangular pulses, especially in the low intensity range of 0.8-1.6nC.

At this point it is important to note that the particular shape of the pulse is crucial since one phase of the pulse just lasts 200μs and the transferred charge directly correlates with the evoked neural activity. Of course one could argue that the impact of the pulse's shape can be ameliorated by simply increasing the pulse width and diminishing the current am-

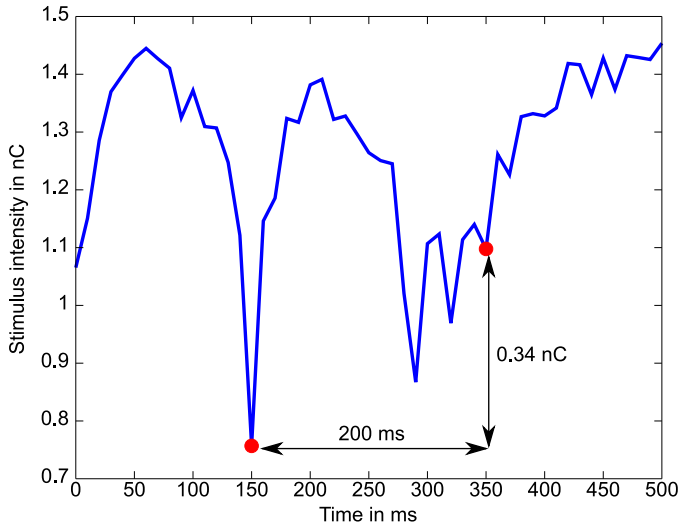


Figure 7.5: *Optimal stimulus intensities predicted by SVR on the* ***fb-081008-f6*** *data set. Intensity values change rapidly, for example by 0.34nC over a time period of 200ms.*

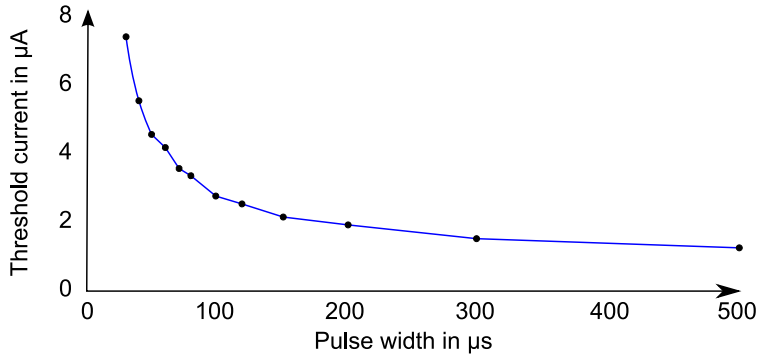


Figure 7.6: *The relationship between pulse width and threshold current for eliciting orthodromic action potentials [100].*

plitude, but this is not practicable as the threshold current to elicit an action potential is a nonlinear function of the pulse width (figure 7.6). Classically the relationship between threshold current and pulse duration is characterized by two measures: the rheobase corresponds to the asymptotic value of the threshold current for large pulse widths and the chronaxie is defined as the pulse width associated with twice the rheobase current [100]. The final solution adopted for the stimulator interface consists of the STG2008 (Multichannel Systems) with an additional RS-232 interface that allows programming of the device in real time. Further, this solution is able to produce clean rectangular pulses across all intensity ranges and has a short delay of 1-2ms between programming and pulse output.

7.3 Results

The potentials elicited by constant and adaptive stimulation together with the associated target response during one experimental session (**fb031208-f2**) are shown in figure 7.7A-C for the three different intensity ranges. For the range of 0.8-1.6nC it can be seen that the evoked potentials are less variable under the adaptive condition during a time period of six milliseconds that directly follows the stimulation pulse. This stabilization effect cannot be observed for the higher intensity ranges. The histograms in figure 7.7D-F show that the amplitudes of the evoked potentials are not normally distributed and it would be therefore erroneous to quantify the stabilization effect by a simple ratio of variances[2)].

Instead the stabilization effect is quantified by first computing the error between the evoked and target potential and subsequently calculating the probability that the error under the adaptive or shifted condition is smaller than the error during stimulation with constant intensity. In other words this directly answers the question: How large is the probability of hitting the target potential with higher precision by adaptive stimulation than by constant stimulation? Independent of the distribution of the errors this probability can be estimated in a robust way by the area under the receiver operating characteristic (ROC) curve [55]. For this purpose the value on the ordinate of the ROC graph is the probability $P(e_a < v)$ that the error under adaptive stimulation is smaller than a certain value v, while the value on the abscissa corresponds to the probability $P(e_c < v)$ for the constant condition. Then the area under the ROC curve (AUC) is equivalent to the probability $P(e_a < e_c)$, that the error under adaptive stimulation is smaller than the error under constant stimulation [5, 57]. Like all statistical measures the AUC is itself a random variable with associated uncertainties. Therefore the results presented in the following additionally give 95% confidence intervals for the AUC estimate. In principle there are several parametric and non-parametric ways to determine the AUC and its confidence intervals [109, 37]. Here the AUC is computed exactly by summing the area of rectangles under the discrete ROC graph, and the bootstrap method with B=1000 bootstrap samples is used to find the 95% confidence intervals [41].

The results of this quantitative analysis are shown in figure 7.7G-I for the adaptive and shifted stimulation conditions. Clearly, for the range of 0.8-1.6nC the probability of hitting the target potential is higher for adaptive stimulation, at least during the first six milliseconds that directly follow the stimulation pulse (figure 7.7G). For the higher intensity ranges of 2.4-3.2nC and 4.0-4.8nC (figures 7.7H,I) adaptive stimulation achieves no significant stabilization effect in this example. In general the stabilization effect declines gradually with increasing stimulation intensity range as shown by the summary of all results in figure 7.10.

The AUC value for the shifted control condition does not exceed the chance level, an AUC value of 0.5, in figure 7.7 and hence one can safely exclude the possibility that the observed stabilization effect under the adaptive condition is solely caused by the peculiar distribution of stimulus intensities.

[2)] The estimator for a ratio of variances is highly sensitive to deviations from the normal distribution [119].

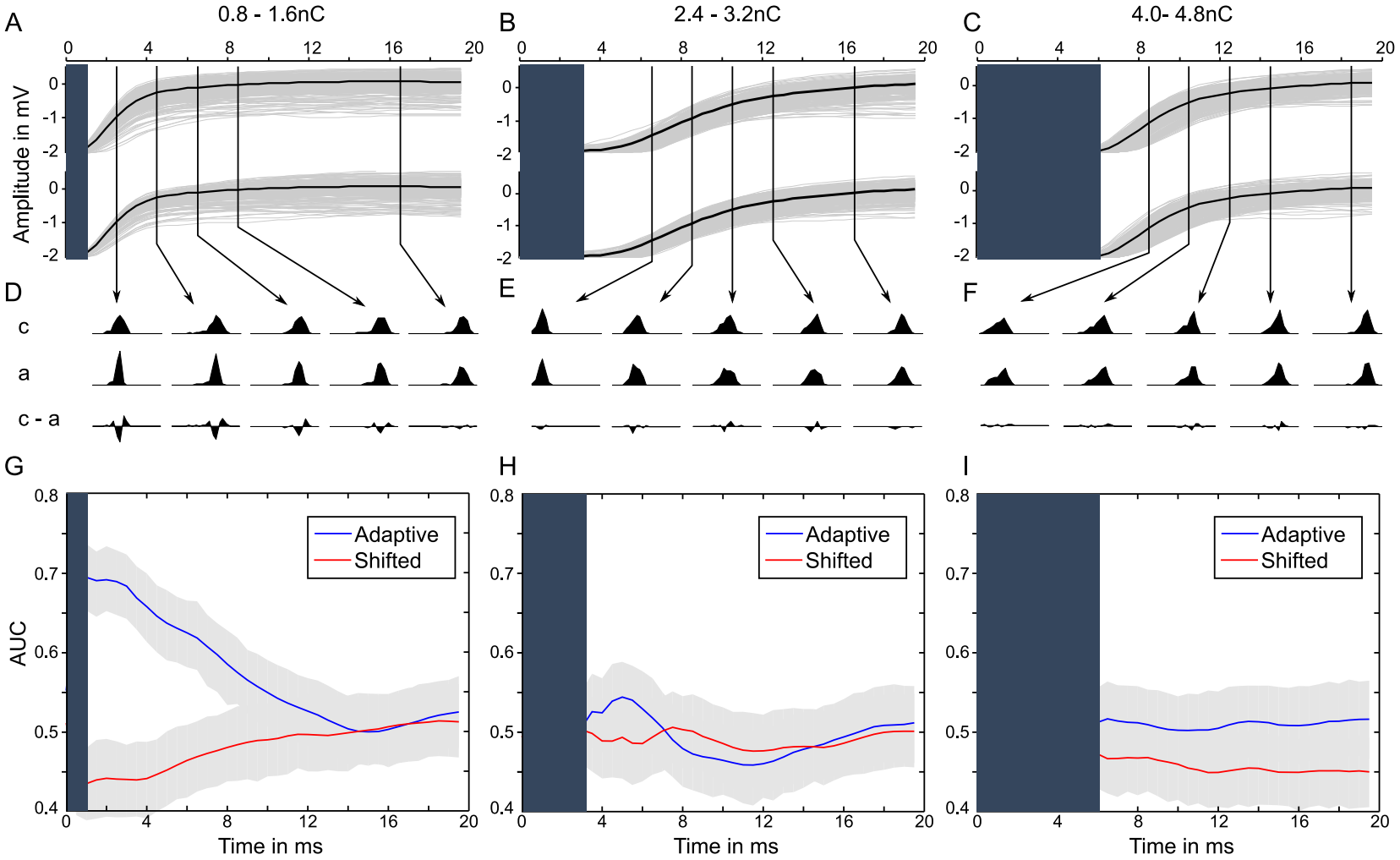


Figure 7.7: *Results of closed loop stimulation.* ***A-C:*** *Evoked potentials in (n=300) trials for stimulation with constant intensity (top) and adaptive stimulation (bottom) together with the target evoked potential (black).* ***D-F:*** *Amplitude histograms are shown at several post-stimulus time-points for constant stimulation (c), adaptive stimulation (a), and the difference between histograms (c-a).* ***G-I:*** *AUC for adaptive and shifted stimulation conditions with 95% confidence intervals shown in gray. The stimulus artifact was excluded from the analysis as indicated by the dark gray areas.*

The results of closed loop stimulation obtained with the shifted control condition are qualitatively similar to the results under the noise control condition, as shown for the range of 0.8-1.6nC in figure 7.8. But due to the difference between the stimulus intensity distributions for noise and adaptive conditions (figure 7.3) one cannot eliminate distributional influences on the stabilization effect in this case.

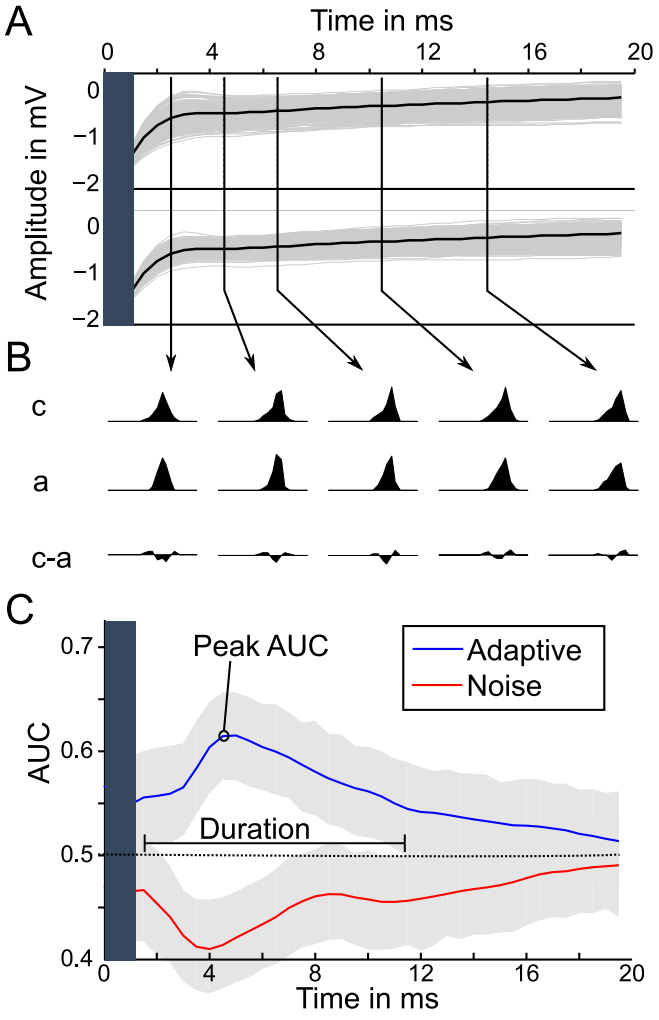


Figure 7.8: *Result of closed loop stimulation with noise control for the range of 0.8-1.6nC on the* ***fb180708-f1*** *data set.* ***A:*** *Evoked potentials in (n=300) trials for stimulation with constant intensity (top) and adaptive stimulation (bottom) together with the target evoked potential (black).* ***B:*** *Amplitude histograms are shown at several post-stimulus time-points for constant stimulation (c), adaptive stimulation (a), and the difference between histograms (c-a).* ***C:*** *Effect size of the difference between adaptive and constant trials (blue) and shifted and constant trials (red) across time expressed as AUC values. Gray shading signify 95% confidence intervals. The stimulus artifact was excluded from the analysis as indicated by the dark gray areas. To summarize the stabilization effect the peak AUC value and duration are determined.*

Irrespective of the particular control condition the AUC values gradually decay behind the peak that follows the stimulation pulse. This indicates that immediate succession of measurement and stimulation is crucial for achieving the observed stabilization effect.

The full data set comprises 28 closed loop stimulation sessions recorded from seven different animals. In four animals all three stimulation ranges were investigated resulting in a total of 18 data sets. The remaining 10 data sets were recorded in experiments with three animals where only the lowest intensity range of 0.8-1.6nC was applied in order to analyze the influence of the recording depth.

The time course of the stabilization effect is highly stereotypical across all experimental

sessions, with a peak of the AUC value immediately after the stimulation pulse followed by a gradual decay to chance level. The results for all sessions are therefore summarized by the peak AUC value and the duration where the stabilization effect is significant, e.g. the lower bound of the 95% confidence interval stays above chance level. Figure 7.8C shows how peak AUC value and duration are determined and figure 7.9 quantifies the stabilization effect for all 28 data sets by plotting peak AUC value against the duration.

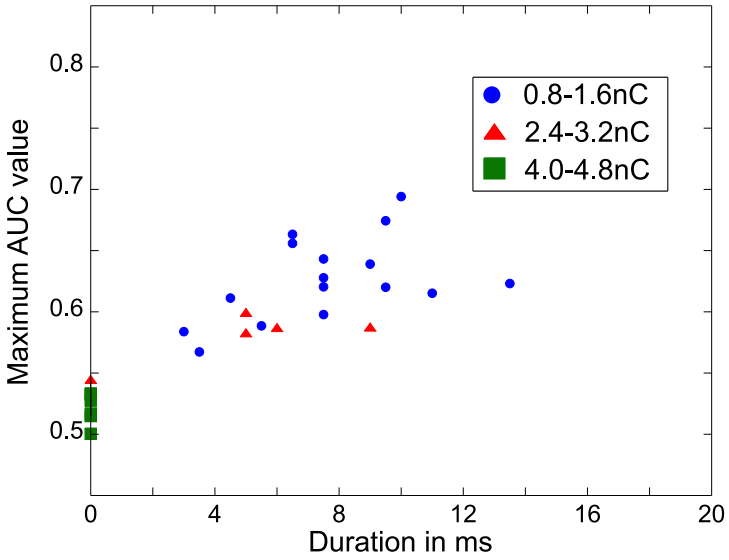


Figure 7.9: *Summary of the closed loop stimulation experiments for all 28 recording sessions. The stabilization effect is measured in terms of the peak AUC value and the duration where the lower bound of the 95% confidence interval stays above chance level. The different stimulation ranges are represented by symbols.*

For the four animals where all three intensity ranges were tested figure 7.10 shows the dependence of the stabilization effect, quantified by the product of peak AUC value and duration, in dependence of the stimulus intensity. It is clear that the biggest effect is usually obtained for the lowest intensity range.

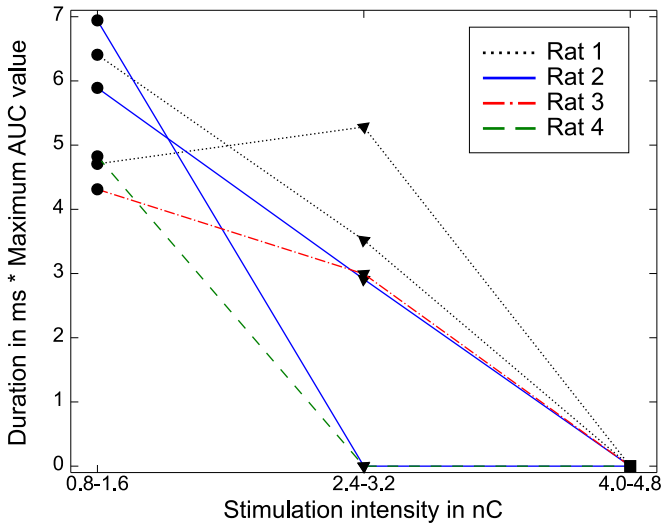


Figure 7.10: *Strength of stabilization effect, as measured by the product of peak AUC value and duration, in dependence of the stimulation intensity range. All intensity ranges were tested in four different animals leading to a total of 18 data sets shown in this figure.*

Only in one session of the first animal (rat 1 in figure 7.10) the effect for the intensity of 2.4-3.2nC is bigger compared to the 0.8-1.6nC range. This exception might be explained by the temporal dependence of the stabilization effect discussed later. Nevertheless these results contain the general tendency that more stabilization is achieved for low stimulation intensities, although the effect size can vary substantially between recording sessions and animals even for the range of 0.8-1.6nC (figure 7.9).

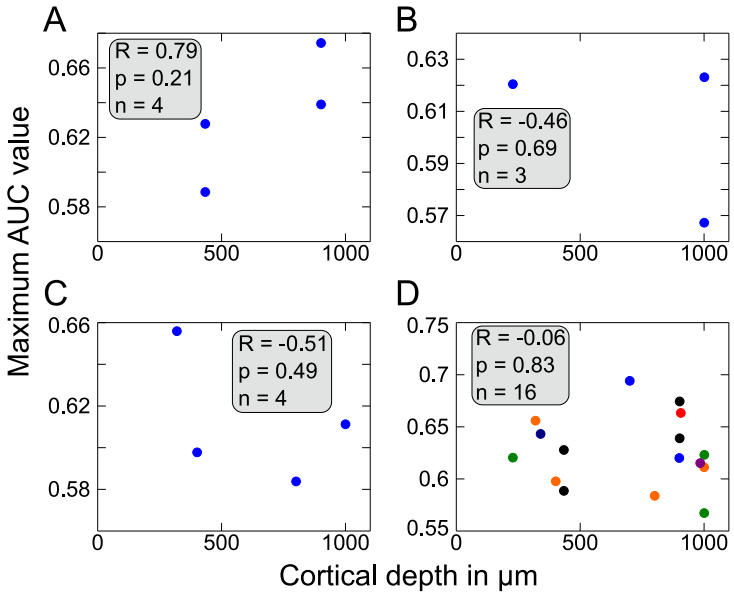


Figure 7.11: *Peak AUC value in dependence of cortical depth for the range of 0.8-1.6nC.* ***A-C:*** *Individually for three animals.* ***D:*** *Across all data sets recorded for the range of 0.8-1.6nC.*

One could speculate that specific cortical layers are more amenable to stabilization than others, or that the depth of the recording location plays a role. While the first conjecture cannot be investigated due to lack of suitable data the second possibility can be ruled out. Figures 7.11A-C show the peak AUC value in dependence of cortical depth for the three animals where several recordings were made in the 0.8-1.6nC range. Apparently there is no consistent relationship between effect size[3)] and cortical depth, and the correlation between Peak AUC value and cortical depth is not significant in each of the animals. This conclusion remains valid when the relationship is analyzed across all data sets recorded for the lowest range (figure 7.11D).

The trial wise analysis revleals a temporal dependence of the stabilization effect, as shown in figure 7.12 for several example data sets. The effect is strongest during the initial trials of the experimental session shown in figures 7.12A and C, while it is sustained almost throughout the entire session in figures 7.12B,D and E, or even increases during the final trials the session shown in figure 7.12F. Obviously there is no systematic evolution of the stabilization effect with time. To date the cause of this temporal dependence could not be clarified. A first speculation that it is caused by changing levels of anesthesia is not supported by the currently available experimental data.

[3)] Similar results are obtained if the product of peak AUC and duration or duration only are used to quantify the stabilization effect.

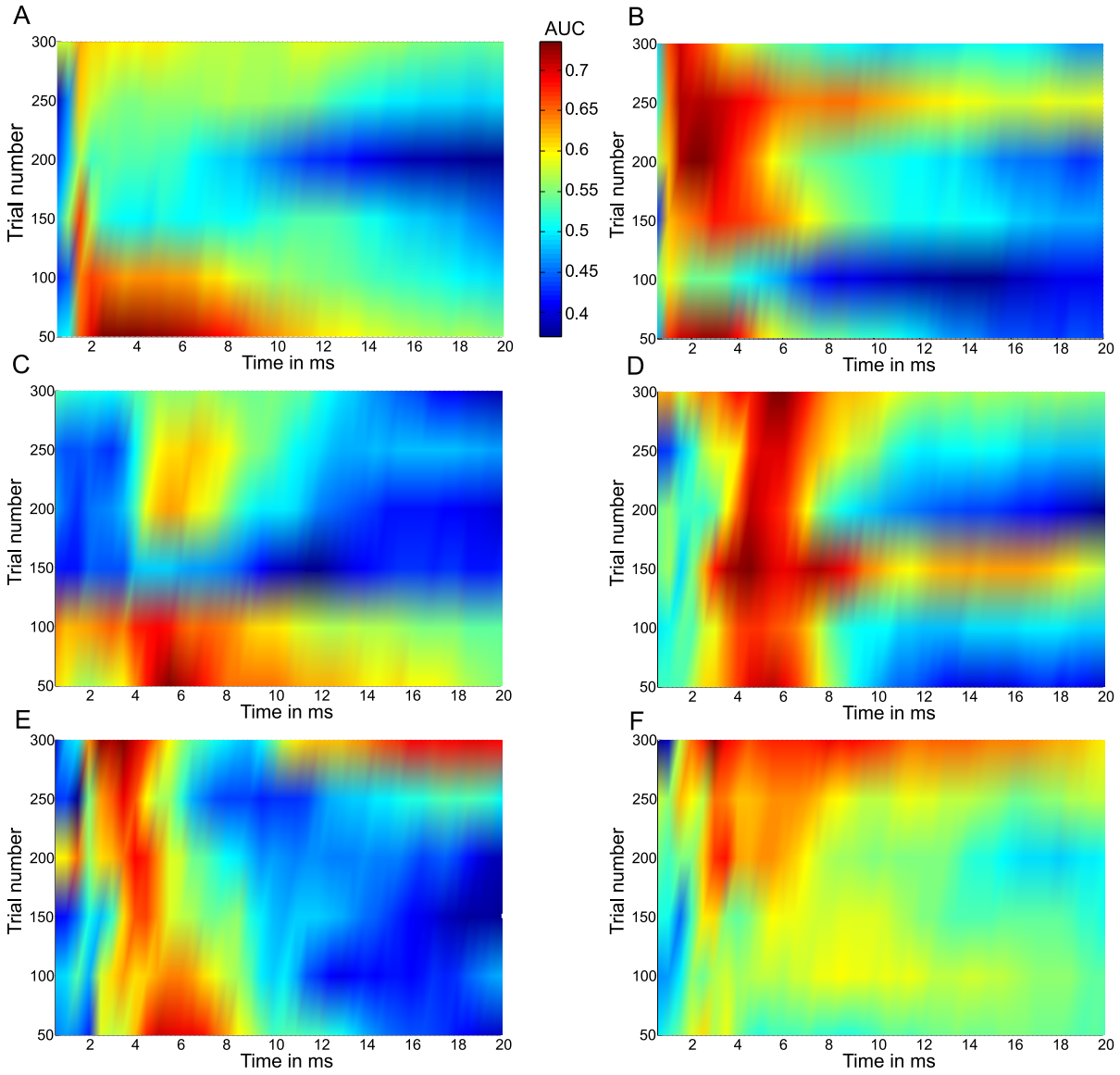


Figure 7.12: *AUC values in dependence of the stimulation trial.* ***A,C:*** *For data sets* ***fb031208-f2*** *and* ***fb180708-f1*** *the initial trials yield the highest AUC values.* ***B,D,E:*** *For data sets* ***fb041208-f4***, ***151008-f1***, *and* ***141008-f1*** *the stabilization effect is almost constant across the whole session.* ***F:*** *For data set* ***fb081008-f3*** *AUC values increase during the final trials of the experimental session.*

A definite answer in this direction will rather be given by future closed loop stimulation experiments in awake, behaving animals.

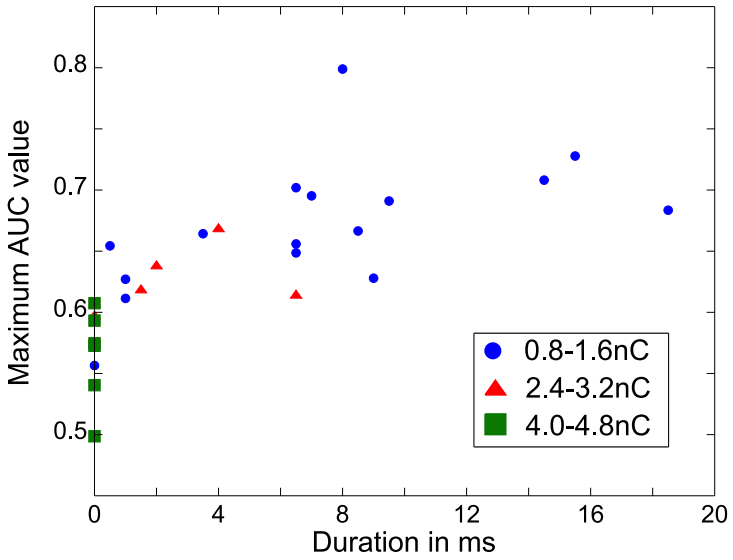


Figure 7.13: *Summary of the closed loop stimulation experiments for all 28 recording sessions. In contrast to figure 7.12 the analysis is conducted on a manually selected sub-range of trials where the stabilization effect is present.*

Of course this temporal influence could be alternatively caused by the SVR model becoming inadequate over time. Should this be the case, then PRIONA, the novel online algorithm for SVR training developed in chapter 4, could help to continuously update the current SVR solution and stop the temporal degradation of the stabilization effect. The special properties of PRIONA, the straightforward adjustment of iteration times and its fully automatic choice of optimal step sizes, have proven valuable during integration into the existing experimental setup. Although online training with PRIONA is already implemented in the current experimental setup this option could not be systematically tested yet, due to a lack of experimental time.

But how would the effect size change if it were possible to avoid the temporal degradation by online training? To answer this question the analysis done for all experimental sessions in figure 7.12 is repeated with manually selected ranges of trials where the stabilization effect is present. The results of this second analysis are shown in figure 7.13, implying that peak AUC values up to 0.8 and durations up to 18ms are achievable if the temporal degradation is avoided.

In conclusion these results show the feasibility to stabilize evoked potentials by monitoring the ongoing brain activity. This finding supports the hypothesis that cortical states influence the processing of sensory information [1, 77, 59, 104]. Further, the transitions between cortical states seem to be governed by small time constants since optimal stimulus intensities change rapidly on a millisecond time scale. The changing input-output relationship underlying the fluctuating cortical state is thus probably caused by fast synaptic inputs rather than slow modulatory transmitter systems.

The intensity interval where stabilization is possible, 0.8-1.6nC, is known to cover the

range just sub-threshold for detection of single pulses in awake, behaving animals [21]. It could therefore be argued, that stabilization is not possible for the stimulus intensity range that is relevant for detection. But future prosthetic devices will need to deliver repetitive stimulation for long time periods in order to create perceptions, and repetitive pulses are known to lower the detection threshold by more than a factor of two. For example, the 80% detection level is reached with a single pulse at 2.8nC or 15 pulses at 1.2nC delivered with a frequency of 320Hz [21].

The short duration of the stabilization effect is less problematic when stimulation is viewed in the context of a prosthetic device, since a visual prosthesis [42, 122, 33] for example will need to provide repetitive stimulation pulses with a frequency around 25Hz in order to enable the flicker-free perception of phosphenes. On the contrary, the short time duration rather seems to favor an approach where evoked activity in sensory cortical areas is stabilized in a pulse-by-pulse fashion. Certainly it first remains to be clarified by future experiments with awake, behaving animals, whether the observed effects are large enough to influence perception and behavior.

Prediction is very difficult
especially about the future
Niels Bohr (1885 - 1962)

8 Conclusion and Outlook

This dissertation shows for the first time that adaptive microstimulation can stabilize evoked cortical potentials in the barrel cortex of anesthetized rats. The results imply that cortical states, which are reflected in the ongoing brain activity, exhibit rapid temporal changes on a millisecond time scale. The concept of closed loop stimulation has been previously employed in a few experimental environments to control the activity of cultured nerve cells [147, 144]. Yet, none of these feedback systems had to cope with the timing constraints of adaptive microstimulation and did not require efficient learning algorithms since the control variable was adjusted by fixed rules. Thus the established experimental setup is exceptional from both a technical and algorithmic point of view.

The contributions of this dissertation on the algorithmic side include:

- A model selection method for support vector regression (SVR) that is based on minimizing the cross validation error with the Quasi-Newton algorithm as described in chapter 5. This approach yields better results across several benchmark data sets in contrast to minimizing the minimum span bound when the hyper-parameters are selected for SVR with the radial basis function kernel. On the other hand minimization of the minimum span bound is the method of choice when the regularization parameter C needs to be tuned.

- An extension of the primal formulation for the SVR algorithm to learning with bias term as described in chapter 3.

- The primal online algorithm [19], PRIONA, a novel online training algorithm for SVR which is based on the primal formulation (chapter 4). It outperforms current state of the art online training algorithms [31, 78] in terms of convergence speed and allows an easier trade off between iteration time and prediction accuracy. PRIONA additionally has the advantage that it does not require the setting of a learning rate – a task that is often cumbersome in practice.

- A direct and inverse modeling approach for solving the adaptive stimulation problem with SVR [17].

- An application specific kernel function which explicitly encodes prior knowledge about the temporal structure of the stimulation trials. As discussed in chapter 6 the custom kernel function yields better results on a majority of data sets in contrast to the linear and polynomial kernel, and it is more amenable to implementation in a real time environment.

The results presented in this thesis naturally spark off several future research topics. On the experimental side the first issue to be addressed is the temporal dependence of the stabilization effect described in chapter 7. It remains to be clarified what causes this temporal dependence and whether it is avoidable by using online training algorithms that continuously update the SVR model. Additional experiments are also required to investigate the transfer of adaptive microstimulation from anesthetized to awake and behaving animals. If this transfer is successful, future research has to clarify whether adaptive microstimulation can actually influence the animal's behavior and perception.

One direction of research on the algorithmic side should aim at enhancing the stabilization effect by more accurately extracting the cortical state from the recorded signals. Certainly the potential for improving algorithms in this direction is limited by the current recording technique that uses a single channel only. If recordings from multiple electrodes are available it will be possible to devise more sophisticated algorithms for analyzing the background brain activity. The potentials acquired by a microelectrode array are usually generated by underlying current sources that are located close in space and are tightly coupled by the local neural network. It will be therefore necessary for instance to investigate how methods for independent component analysis [68, 4] have to be modified when the independence assumption about the sources is partially weakened or violated.

The ultimate goal of adaptive microstimulation is the creation of stable sensory percepts that correspond to complex spatiotemporal activation patterns. Another direction of research should consequently focus on extending the single pulse stimulation in the temporal as well as the spatial domain. The natural next step in this direction is the development of an algorithm capable of stabilizing cortical potentials evoked by stimulation pulse trains instead of single pulses. Under these conditions it will be important to analyze how the evoked potentials of temporally close stimulation pulses interact with each other and incorporate this knowledge into the algorithm. The stimulation with pulse trains might even offer the opportunity to actively probe the cortical state instead of passively analyzing the recorded brain activity. In this scenario the first pulse in each train would be delivered with a fixed intensity under the assumption that the cortical state is represented with higher precision in the associated evoked potential. With the information gleaned from the first evoked potential the intensity of the subsequent pulses in the train could then be adjusted. Besides the cortical state the algorithm could also exploit the knowledge of stimulus intensities predicted at previous time points. Yet, it is still an open question how the algorithm should deal with the uncertainties of the predicted intensities in this situation.

Computer Science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra (1930-2002)

A Algorithms

Algorithm 1: *Minimizes function $f(\theta)$ by the Quasi-Newton method.*

Input: Objective function f, gradient ∇f, starting point θ_0, lower bounds l, and upper bounds u.

Output: Point θ^* where $f(\theta^*)$ attains its minimum.

QUASI-NEWTON(f, ∇f, θ_0, l, u)

(1) $k \leftarrow 1$, $H_1 \leftarrow I$, $\gamma \leftarrow 0.5$, $\epsilon \leftarrow 1e-05$
(2) $\theta_1 \leftarrow \max(\min(\theta_0, u), l)$
(3) **repeat**
    # Compute search direction p
(5) $p \leftarrow -H\nabla f(\theta_k)$
    # Perform line search with a maximum of 100 iterations
(7) $i \leftarrow 0$
(8) **repeat**
(9) $\lambda \leftarrow \gamma^i$
(10) $i \leftarrow i + 1$
(11) $\theta_{k+1} \leftarrow \max(\min(\theta_k + \lambda p, u), l)$
(12) **until** $f(\theta_{k+1}) > f(\theta_k) + 1e-04\lambda\nabla f(\theta_k)p \ \vee \ i > 100$
    # Update approximation of inverse Hessian matrix
(14) $t \leftarrow \nabla f(\theta_{k+1}) - \nabla f(\theta_k)$
(15) $s \leftarrow \theta_{k+1} - \theta_k$
(16) **if** $t^T s > 0$
(17) $H_{k+1} \leftarrow (I - st^T/t^T s)H_k(I - ts^T/t^T s) + ss^T/t^T s$
(18) $k \leftarrow k + 1$
(19) **until** $\|\nabla f(\theta_k)\| < (1 + f(\theta_k)) * \epsilon \ \vee \ |\frac{f(\theta_{k+1}) - f(\theta_k)}{f(\theta_k)}| < \epsilon$
(20) $\theta^* \leftarrow \theta_k$

Algorithm 2: *Primal SVR training by Newton Optimization*

Input: Kernel matrix K, regression targets y, regularization parameter λ, and loss function parameter ε.

Output: Coefficient vector β, bias term b and support vector index set sv.

PrimalSVR(K, y, λ, ε)

(1) $\beta \leftarrow 0$
(2) $m \leftarrow$ Length(y)
(3) **if** $m > 1000$
    # Use a subset of the training patterns to estimate the support vector index set.
(5) $m' \leftarrow m/2$
(6) $(\beta_{1:m'}, b, sv) \leftarrow$ PrimalSVR($K_{1..m',1..m'}$, $y_{1..m'}$, λ, ε)
(7) $\hat{y} \leftarrow K_{1..m,sv}\,\beta_{sv} + b$
(8) **else**
    # Use the mean of the regression targets as an initial guess for the bias term.
(10) $b \leftarrow 1/m \sum_{i=1}^{m} y_i$
(11) $\hat{y} \leftarrow b$
(12) **while true**
(13) $r \leftarrow \hat{y} - y$
(14) $sv \leftarrow \{i, |r_i| > \varepsilon\}$
(15) $nsv \leftarrow \{i, |r_i| \leq \varepsilon\}$
    # Compute the full Newton step using algorithm 4.
(17) $(\bar{\beta}, \bar{b}) \leftarrow$ Newton-step(K, y, λ, ε, r, sv)
(18) $\bar{r} \leftarrow K_{1..m,sv}\,\bar{\beta}_{sv} + \bar{b}$
    # Check for convergence of algorithm.
(20) **if** $(\bar{r}_i > \varepsilon\, \forall_{i \in sv} \wedge \bar{r}_i \leq \varepsilon\, \forall_{i \in nsv}) \vee |sv| = 0$
(21) **return** $(\bar{\beta}, \bar{b}, sv)$
(22) $u \leftarrow K_{1..m,sv}(\bar{\beta} - \beta)_{sv} + (\bar{b} - b)$
(23) $\phi'(0) \leftarrow 2[u_{sv}^T r_{sv} - \varepsilon\, \mathrm{sgn}(r_{sv})^T u_{sv} + \lambda(\bar{\beta} - \beta)^T K \beta]$
(24) $(\phi'(1) - \phi'(0)) \leftarrow 2[u_{sv}^T(u_{sv} - \varepsilon(\mathrm{sgn}(\bar{r}_{sv}) - \mathrm{sgn}(r_{sv}))) + \lambda(\bar{\beta} - \beta)^T K(\bar{\beta} - \beta)]$
    # Find step size ρ with algorithm 3.
(26) $\rho \leftarrow$ Line-search(u, r, $\bar{r}$, ε, $\phi'(0)$, $(\phi'(1) - \phi'(0))$)
    # Update the coefficients and bias term.
(28) $\beta \leftarrow \beta + \rho(\bar{\beta} - \beta)$
(29) $b \leftarrow b + \rho(\bar{b} - b)$
(30) $\hat{y} \leftarrow \hat{y} + \rho u$

Algorithm 3: *Exact line search for the primal SVR problem with bias.*

Input: u, r, $\bar{r}$, ε, $\phi'(0)$, and $(\phi'(1) - \phi'(0))$.
Output: Step size $\rho^* \in (0, 1]$.

LINE-SEARCH(u, r, $\bar{r}$, ε, $\phi'(0)$, $(\phi'(1) - \phi'(0))$)

(1) **foreach** $i \in nsv$
    # Case I
(3) $\rho \leftarrow (\text{sgn}(u_i)\varepsilon - r_i)/u_i$
(4) **if** $\rho \in (0, 1]$
(5) $\Omega \leftarrow \Omega \cup (\rho, nsv(i), \text{'entering'})$
(6) **foreach** $i \in sv$
    # Case II
(8) $\rho \leftarrow (\text{sgn}(r_i)\varepsilon - r_i)/u_i$
(9) **if** $\rho \in (0, 1]$
(10) $\Omega \leftarrow \Omega \cup (\rho, sv(i), \text{'leaving'})$
(11) **foreach** $i \in sv$
    # Case III
(13) $\rho \leftarrow (-\text{sgn}(r_i)\varepsilon - r_i)/u_i$
(14) **if** $\rho \in (0, 1]$
(15) $\Omega \leftarrow \Omega \cup (\rho, sv(i), \text{'entering'})$
(16) $\Omega \leftarrow$ QUICK-SORT(Ω) # The quicksort algorithm as described in [34]
(17) s $\leftarrow 0$
(18) $\phi_{min} \leftarrow \infty$
(19) **foreach** $(\rho, i, mark) \in \Omega$
(20) $\rho^* \leftarrow -\phi'(0)/(\phi'(1) - \phi'(0))$
(21) **if** $s < \rho^* \leq \rho$
(22) **return** ρ^*
(23) **if** $\rho^* > \rho$
    # Minimum of $\phi(\rho)$ is to the right of ρ
(25) $\rho^* \leftarrow \rho$
(26) **if** $\rho^* <= s$
    # Minimum of $\phi(\rho)$ is to the left of s
(28) $\rho^* \leftarrow$ s
(29) **if** $\phi(\rho^*) < \phi_{min}$
(30) $\rho_{min} \leftarrow \rho^*$
(31) **if** mark = 'entering'
(32) $\phi'(0) \leftarrow \phi'(0) + 2u_i(r_i - \varepsilon\,\text{sgn}(r_i))$
(33) $(\phi'(1) - \phi'(0)) \leftarrow (\phi'(1) - \phi'(0)) + 2u_i(u_i - \varepsilon(\text{sgn}(\bar{r}_i) - \text{sgn}(r_i)))$
(34) **if** mark = 'leaving'
(35) $\phi'(0) \leftarrow \phi'(0) - 2u_i(r_i - \varepsilon\,\text{sgn}(r_i))$
(36) $(\phi'(1) - \phi'(0)) \leftarrow (\phi'(1) - \phi'(0)) - 2u_i(u_i - \varepsilon(\text{sgn}(\bar{r}_i) - \text{sgn}(r_i)))$
(37) $s \leftarrow \rho$
    # This point is reached only if the minimum could not be found analytically.
(39) **return** ρ_{min}

Algorithm 4: *Computes the Newton step for the primal SVR problem with bias.*
Input: Kernel matrix K, regression targets y, regularization parameter λ, loss function parameter ε, residual vector r, and support vector index set sv.
Output: Vector $\bar{\beta}$ and scalar $\bar{b}$.
NEWTON-STEP(K, y, λ, ε, r, sv)
(1) $\bar{\beta} \leftarrow 0$
(2) $t \leftarrow y_{sv} + \varepsilon \operatorname{sgn}(r_{sv})$
(3) $R \leftarrow$ CHOLESKY-DECOMPOSITION($K_{sv,sv} + \lambda I$)
Using triangular Cholesky factor R, solve the linear systems
as described in section 3.2.2 by two back-substitutions.
(6) $u \leftarrow R^{-1}(R^{-T}t)$
(7) $w \leftarrow R^{-1}(R^{-T}\mathbf{1})$
(8) $\bar{\beta}_{sv} \leftarrow u - (w\mathbf{1}^T u)/(\mathbf{1}^T w)$
(9) $\bar{b} \leftarrow (\mathbf{1}^T u)/(\mathbf{1}^T w)$

Algorithm 5: *Evaluates the ANOVA kernel with the time point of interest weighting scheme by dynamic programming*
Input: Input patterns $x, z \in \mathbb{R}^n$, base weight $\mu \in [0, 1]$, maximal monomial degree d, and time point of interest $\tau \in [0, n]$.
Output: Value $k(x, z)$ of the ANOVA kernel
ANOVA-KERNEL(x, z, μ, d, τ)
Initialize dynamic programming table
(2) **for** $k = 2$ **to** n
(3) **for** $j = 1$ **to** n
(4) $T(1, k, j) \leftarrow 0$
(5) **for** $j = 1$ **to** n
(6) $T(1, 1, j) \leftarrow mu^{|\tau - j|} x_j z_j$
(7) $k(x, z) \leftarrow 0$
Recursively fill dynamic programming table
(9) $p_1 \leftarrow 2$
(10) $p_2 \leftarrow 1$
(11) **for** $i = 2$ **to** d
(12) **for** $k = i$ **to** n
(13) **for** $j = k$ **to** n
(14) $T(p_1, k, j) \leftarrow 0$
(15) **for** $l = 1$ **to** $k - i + 1$
(16) $T(p_1, k, j) \leftarrow T(p_1, k, j) + \mu^{|\tau - j| + l} x_j z_j T(p_2, k - l, j - l)$
(17) $k(x, z) \leftarrow k(x, z) + T(p_1, j, k)$
Swap p_1 and p_2
(19) $p_1 \leftrightarrow p_2$

Everything is vague to a degree
you do not realize
till you have tried to make it precise.

Bertrand Russell (1872-1970)

B Data sets

The following subsections give detailed descriptions of the data sets used in the evaluation of primal and dual SVR training in chapter 3, the comparison of online SVR algorithms in chapter 4, and the comparison of model selection methods in chapter 5. An overview of all data sets is given in table B.1. With exception of a linear scaling to the interval $[0, 1]$, applied to the prediction targets, the data sets obtained from Internet repositories were used without further modifications. The scaling operation facilitates the comparison of prediction errors.

Data set	Number of patterns	Dimension	Source
abalone	4177	8	UCI Repository
cadata	20640	8	StatLib
cpusmall	8192	12	Delve Repository
fb081008-r1	600	20	Feedback experiment
fb141008-r2	600	20	Feedback experiment
fb151008-r2	600	20	Feedback experiment
fb180708-r1	600	20	Feedback experiment
housing	506	13	UCI Repository
mpg	392	7	UCI Repository
pyrim	74	27	UCI Repository
space-ga	3107	6	StatLib
triazines	186	60	UCI Repository

Table B.1: *Overview of benchmark data sets used for the evaluation of primal/dual SVR training and the comparison of online SVR algorithms.*

B.1 Abalone

The age of abalone is usually determined by cutting the shell through the cone and counting the rings with the aid of a microscope after staining. This procedure is a very tedious task. This data set was collected to predict the age of abalone from easily obtainable physical

measurements only [95]. The dimensions of the data set include the sex (male, female, or infant), the length, diameter, and height of the shell in mm, and the whole, meat, gut, and dry weight in grams. Certainly, further information like weather patterns and location that indicate food availability and are not included in the data set would be required to achieve more accurate predictions. Patterns with missing values were already removed from the original examples in the data set available from the UCI repository [3].

B.2 Cadata

The dimensions of each training pattern represent the median income, the median age, total number of rooms per population, the number of bedrooms per population, the number of people in each household, and the number of households. Except for the median income all dimensions were transformed by computing the natural logarithm. This information was gleaned from the 1990 Census data for California, where estimates for each variable were determined across block groups comprising 1425 individuals on average. The six variables mentioned above are complemented by the median income raised to the second and third power. Based on this information the median house price can be predicted as shown in the original study [102].

B.3 Cpusmall

In order to optimize resource usage of a server in a multi-user environment it is important to know what portion of time the CPU runs in user mode in dependence of other computer system activity measurements. This data set was collected on a Sun Sparcstation 20/712 with 128 Mbytes of memory running in a multi-user university department, where users would do a large variety of tasks like accessing the Internet, editing files, or running CPU-intensive programs. To make prediction of the user mode CPU time more challenging all dimensions related to paging information were excluded from the full data set.

B.4 Feedback

All the data sets that follow the naming scheme **fbDDMMYY-rN** or **fbDDMMYY-fN** originate from the experiments on adaptive microstimulation described in more detail in chapters 6 and 7. The naming scheme contains the date of the experiment (DDMMYY) and the number (N) of the recording (r) or feedback stimulation (f). One training pattern of this data set includes 20ms of recorded pre- and post-stimulus local field potentials during one stimulation trial. The signal, recorded with a sampling frequency of 20KHz, was subsequently down-sampled to 500Hz after extraction of the local field potential. In the

direct solution for adaptive stimulation, described in chapter 6, the goal is the prediction of stimulus intensity based on the recorded pre- and post-stimulus brain signal.

B.5 Housing

In the original work [58] this data set was used to study the influence of air pollution, as measured by the nitric oxides concentration, on the median house price. The relevant information stems from census tract data collected in 1970 for the Boston metropolitan area. The dimensions represent crime rate, proportion of residential land zoned for lots over 25,000 square feet, proportion of non-retail business acres per town, an indicator whether a census tract bounds a river, nitric oxides concentration, average number of rooms per dwelling, proportion of owner-occupied units built prior to 1940, weighted distances to five Boston employment centers, an index for accessibility to radial highways, full-value property tax rate per 10,000$, pupil-teacher ratio, a measure for the proportion of blacks, and the percentage of lower status of the population.

B.6 Mpg

For the prediction of city-cycle fuel consumption, measured in miles per gallon, each pattern of this data set contains information for one car model. Each car model is described by the number of cylinders, the displacement, the horsepower of the engine, the car weight, the acceleration, the year of the car model, and its origin [110]. In the data set available from the UCI repository [3] eight patterns, having missing values for the miles per gallon measurement, were excluded from the original data.

B.7 Triazines and Pyrim

Both data sets were collected to establish qualitative structure activity relationships for the problem of E. Coli dehydrofolate reductase inhibition by the enzymes triazine and pyrimidine [132].

B.8 Space-ga

The data set stems from a geographical analysis of spatial data [101], collected during the U.S. presidential election in 1980. Each training pattern includes the number of votes cast in the election per county, the population in each county of 18 years of age or older, the population in each county with a 12th grade or higher education, the number of owner-occupied housing units, the aggregate income, and the spatial coordinates of the county.

Based on this information the logarithm of the proportion of votes cast for both candidates has to be predicted.

Abbreviations

ANOVA Analysis of variance. 3, 103–106, 110–112, 115

AUC Area under ROC curve. 120–126

CSD Current source density. 92

CV Cross validation. 75, 76, 82–84, 86, 87

DMA Direct memory access. 117

ECoG Electrocorticogram. 90, 91, 94

EEG Electroencephalogram. 90, 91, 94

FFT Fast fourier transform. 94

FIFO First in first out. 60, 61

GABA Gamma amino butyric acid. 21

KKT Karush Kuhn Tucker. 80

LFP Local field potential. 90, 91, 93, 94, 103, 106, 107, 112

LGN Lateral geniculate nucleus. 9–12

LOO Leave one out. 76–78, 84, 87

MSE Mean squared error. 47–49, 83, 84, 87, 106

MSP Minimum span. 79–84, 86, 87, 106, 111, 115

MUA Multi unit activity. 90, 92, 93, 103, 107, 112

NORMA Naive online risk minimization. 54, 56–59, 66–68, 70–74

OKRR Online kernel ridge regression. 69–74

PCA Principal component analysis. 106, 107, 109

PCI Peripheral component interconnect. 114, 117

POM Posterior medial. 21

PRIONA Primal online algorithm. 3, 59, 60, 62, 65–74, 129

RBF Radial basis function. 47, 65, 75, 81–83, 87, 95–98, 103, 110, 112

RKHS Reproducing kernel Hilbert space. 25

RLS Regularized least squares. 29, 30, 49

RMSE Root mean squared error. 106–108, 110–112

ROC Receiver operating characteristic. 120

SD Standard deviation. 109

SILK Sparse implicit learning with kernels. 58, 59, 67, 68, 70–74

SMO Sequential minimal optimization. 29, 49, 54

STG Stimulus generator. 115, 118

SUA Single unit activity. 90–93

SVC Support vector classification. 27, 36, 45, 46

SVM Support vector machine. 25, 26, 53, 54, 57, 60

SVR Support vector regression. 3, 4, 26–29, 31–34, 36, 37, 40, 42, 46, 48, 49, 51–53, 58–61, 63–65, 71, 73, 75–83, 86, 87, 89, 96, 97, 102, 106, 110, 112–116, 118, 119, 126, 129, 130, 135

V1 Primary visual cortex. 2, 9–13

VPM Ventral posterior medial. 21

Bibliography

[1] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen. Dynamics of Ongoing Activity: Explanation of the Large Variability in Evoked Cortical Responses. *Science*, 273(5283):1868–1871, 1996. 2, 3, 18, 100, 128

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950. 32

[3] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007. 138, 139

[4] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *JMLR*, 3:1–48, 2002. 132

[5] D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of mathematical psychology*, 12(4):387–415, 1975. 122

[6] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 2003. 30, 37, 81

[7] W. Bialek, F. Rieke, R. R. de Ruyter van Steveninck, and D. Warland. Reading a neural code. *Science*, 252(5014):1854–1857, 1991. 92

[8] N. Birbaumer and L. G. Cohen. Brain-computer interfaces: communication and restoration of movement in paralysis. *The Journal of Physiology*, 579(3):621, 2007. 1

[9] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kübler, J. Perelmouter, E. Taub, and H. Flor. A spelling device for the paralysed. *Nature*, 398:297–298, 1999. 1

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1st edition, 1996. 25, 76

[11] L. Bo, L. Wang, and L. Jiao. Recursive finite newton algorithm for support vector regression in the primal. *Neural Computation*, 19:1082–1096, 2007. 32, 34, 36

[12] M. Bogdan, M. Schröder, and W. Rosenstiel. Towards restoration of hand grasp function of quadriplegic patients based on an artificial neural net controller using peripheral nerve stimulation - an approach. In *ESANN proceedings*, pages 427–432, 2003. 116

[13] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005. 54, 60

[14] G. S. Brindley and W. S. Lewin. The sensations produced by electrical stimulation of the visual cortex. *Journal of Physiology*, 196:479–493, 1968. 2, 16

[15] D. Brugger. Parallel support vector machines. Technical Report WSI-2006-01, Eberhard-Karls Universität Tübingen, WSI für Informatik, 1 2006. 29

[16] D. Brugger, M. Bogdan, and W. Rosenstiel. Automatic cluster detection in kohonen's som. *IEEE Tran. Neural Networks*, 19(3):442–459, 2008. Automatic cluster detection in Kohonen's SOM. 92

[17] D. Brugger, S. Butovas, M. Bogdan, C. Schwarz, and W. Rosenstiel. Direct and inverse solution for a stimulus adaptation problem using SVR. In *ESANN proceedings*, pages 397–402, Bruges, 2008. 3, 102, 131

[18] D. Brugger, S. Butovas, M. Bogdan, C. Schwarz, and W. Rosenstiel. Real-time adaptive microstimulation increases reliability of electrically evoked cortical potentials. *Submitted to Journal of Neuroscience*, 2009. 3, 119

[19] D. Brugger, W. Rosenstiel, and M. Bogdan. Online SVR training by solving the primal optimization problem. In *IEEE International Workshop on Machine Learning for signal processing*, 2009. 3, 59, 60, 131

[20] S. Butovas and C. Schwarz. Spatiotemporal effects of microstimulation in rat neocortex: a parametric study using multielectrode recordings. *J Neurophysiol*, 90(5):3024–39, 2003. eng. 101

[21] S. Butovas and C. Schwarz. Detection psychophysics of intracortical microstimulation in rat primary somatosensory cortex. *Eur J Neurosci*, 25(7):2161–9, 2007. eng. 129

[22] G. Buzsáki. Large-scale recording of neuronal ensembles. *Nature Neuroscience*, 7:446–451, 2004. 92

[23] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000. 54, 62

[24] K. Cha, K. W. Horch, and R. A. Normann. Mobility performance with a pixelized vision system. *Vision Research*, 32:1367–1372, 1992. 17, 18

[25] K. Cha, K. W. Horch, and R. A. Normann. Reading speed with a pixelized vision system. *Journal of the Optical Society of America A*, 9(5):673–677, 1992. 17

[26] F. S. Chance, L. F. Abbott, and A. D. Reyes. Gain modulation from background synaptic input. *Neuron*, 35(4):773–82, 2002. eng. 3, 18

[27] C. C. Chang and C. J. Lin. LIBSVM: a Library for Support Vector Machines, 2006. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`. 29, 49, 53

[28] M. W. Chang and C. J. Lin. Leave-one-out bounds for support vector regression model selection. *Neural Computation*, 17:1188–1222, 2005. Leave-one-out bounds for support vector regression model selection. 76, 77, 78, 79, 82

[29] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1135–1178, 2007. 29, 31, 36, 46, 48

[30] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002. 76, 79

[31] L. Cheng, S.V.N. Vishwanathan, D. Schuurmans, S. Wang, and T. Caelli. Implicit online learning with kernels. In *NIPS*, pages 249–256. MIT Press, 2007. 54, 57, 59, 131

[32] W. Chu, S. S. Keerthi, and C. J. Ong. Bayesian support vector regression using a unified loss function. *IEEE Tran. Neural Networks*, 15(1):29–44, January 2004. 76

[33] E. D. Cohen. Prosthetic interfaces with the visual system: biological issues. *Journal of Neural Engineering*, 4:14–31, 2007. 1, 14, 15, 16, 129

[34] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001. 100, 105, 135

[35] K. Crammer, J. S. Kandola, and Y. Singer. Online classification on a budget. In *NIPS*, 2003. 60

[36] L. Csató and M. Opper. Sparse representation for gaussian process models. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS*, pages 444–450. MIT Press, 2000. 62

[37] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988. 122

[38] T. B. DeMarse, D. A. Wagenaar, A. W. Blau, and S. M. Potter. The Neurally Controlled Animat: Biological Brains Acting with Simulated Bodies. *Autonomous Robots*, 11(3):305–310, 2001. 116

[39] A. Destexhe, M. Rudolph, and D. Pare. The high-conductance state of neocortical neurons in vivo. *Nat Rev Neurosci*, 4(9):739–51, 2003. eng. 18

[40] I. M. Devonshire, J. E. W. Mayhew, and P. G. Overton. Cocaine preferentially enhances sensory processing in the upper layers of the primary sensory cortex. *Neuroscience*, 146(2):841–851, 2007. 92

[41] T. J. DiCiccio and B. Efron. Bootstrap confidence intervals. *Statistical Science*, 11(3):189–228, 1996. Bootstrap Confidence Intervals. 47, 65, 83, 107, 122

[42] W. H. Dobelle. Artificial vision for the blind by connecting a television camera to the visual cortex. *ASAIO J.*, 46:1–7, 2000. 2, 16, 17, 129

[43] W. H. Dobelle and M. G. Mladejovsky. Phosphenes produced by electrical stimulation of human occipital cortex, and their application to the development of a prosthesis for the blind. *Journal of Physiology*, 243:553–576, 1974. 2, 16

[44] J. P. Donoghue. Connecting cortex to machines: recent advances in brain interfaces. *Nature Neuroscience*, 5(supp):1085–1088, 2002. 89

[45] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001. 25, 75

[46] Dudel, Menzel, and Schmidt. *Neurowissenschaft – Vom Molekül zur Kognition*. Springer, 2. auflage edition, 2001. 6

[47] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Chapman & Hall, 1st edition, 1994. 47

[48] R. E. Fan, P. H. Chen, and C. J. Lin. Working Set Selection Using Second Order Information for Training Support Vector Machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005. 29

[49] M. Feucht, T. Laube, N. Bornfeld, P. Walter, and M. Velikay-Parel. Entwicklung einer epiretinalen prothese zur stimulation der humanen netzhaut. *Ophthalmologe*, 102(7):688–691, 2005. 1, 14

[50] K. Fox. *Barrel Cortex*. Cambridge University Press, first edition edition, 2008. 19, 21

[51] T. Gärtner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. *LNAI*, 2777:129–143, 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. 95

[52] F. Gekeler and E. Zrenner. Stand des subretinalen implantatprojekts. *Ophthalmologe*, 102(10):941–949, 2005. 14

[53] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986. 89, 90

[54] G. H. Golub and C. F. van Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996. 26, 30, 41, 62, 80

[55] D. M. Green and J. A. Swets. *Signal detection theory and psychophysics*. Wiley, New York,, 1966. 66021059 [by] David M. Green [and] John A. Swets. illus. 24 cm. Bibliography: p. 417-428. 122

[56] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics. Society for Industrial Mathematics, 1st edition, January 1987. 84

[57] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29, 1982. 122

[58] D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978. 139

[59] A. Hasenstaub, R. N. Sachdev, and D. A. McCormick. State changes rapidly modulate cortical neuronal responsiveness. *J Neurosci*, 27(36):9607–22, 2007. eng. 3, 128

[60] R. Haslinger, I. Ulbert, C. I. Moore, E. N. Brown, and A. Devor. Analysis of lfp phase predicts sensory response of barrel cortex. *Journal of Neurophysiology*, 96(3):1658, 2006. 94

[61] H. Hentsche, F. Haiss, and C. Schwarz. Central signals rapidly switch tactile processing in rat barrel cortex during whisker movements. *Cerebral Cortex*, 16(8):1142–1156, 2006. 3, 22, 23

[62] T. Hermle, M. Bogdan, C. Schwarz, and W. Rosenstiel. ANN-based system for sorting spike waveforms employing refractory periods. *Lecture notes in computer science*, 3696:121, 2005. 92

[63] T. Hermle, C. Schwarz, and M. Bogdan. Employing ICA and SOM for spike sorting of multielectrode recordings from CNS. *Journal of Physiology-Paris*, 98(4-6):349–356, 2004. 92

[64] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442:164–171, 2006. 90

[65] W. H. House. Cochlear implants. *Annals of Otology, Rhinology and Laryngology*, 85:3–91, 1976. 2

[66] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *Journal of physiology*, 148(3):574–592, 1959. 12

[67] M. S. Humayun, J. D. Weiland, G. Y. Fujii, R. Greenberg, R. Williamson, J. Little, B. Mech, V. Cimmarusti, G. Van Boemel, G. Dagnelie, and E. de Juan Jr. Visual perception in a blind subject with a chronic microelectronic retinal prosthesis. *Vision Research*, 43(24):2573–2581, 2003. 1, 14

[68] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000. Independent component analysis: algorithms and applications. 132

[69] S. Jezernik, R. G. V. Wassink, and T. Keller. Sliding mode closed-loop control of FES controlling the shank movement. *Biomedical Engineering, IEEE Transactions on*, 51(2):263–272, 2004. 116

[70] T. Joachims. Estimating the generalization performance of an svm efficiently. In Pat Langley, editor, *ICML*, pages 431–438. Morgan Kaufmann, 2000. 76

[71] K. O. Johnson, S. S. Hsiao, and T. Yoshioka. Neural coding and the basic law of psychophysics. *Neuroscientist*, 8(2):111–121, 2002. 8

[72] I.T. Jolliffe. *Principal component analysis*. Springer New York, 2nd edition, October 2002. 107

[73] E. R. Kandel, J. H. Schwarz, and T. M. Jessell. *Principles Of Neural Science*. McGraw-Hill, fourth edition edition, 2000. 6, 10, 11

[74] S. S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *JMLR*, 6:341–361, 2005. 36, 37, 45

[75] P. R. Kennedy, M. T. Kirby, M. M. Moore, B. King, A. Mallory, N. S. Inc, and G. A. Atlanta. Computer control using human intracortical local field potentials. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(3):339–344, 2004. 90

[76] G. S. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, April 1970. 32

[77] M. A. Kisley and G. L. Gerstein. Trial-to-trial variability and state-dependent modulation of auditory-evoked responses in cortex. *J Neurosci*, 19(23):10451–60, 1999. eng. 3, 100, 128

[78] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Trans. Sig. Proc.*, 52(8):2165–2175, 2004. 54, 59, 60, 131

[79] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, 1997. 57

[80] S. Kotler. A half century of artificial-sight research has succeeded. and not this blind man can see. *Wired Magazine*, 2002. 16

[81] J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. In *Proceeding of the 20th International Conference on Machine Learning*, 2003. 103

[82] M. Le Van Quyen, J. Foucher, J. P. Lachaux, E. Rodriguez, A. Lutz, J. Martinerie, and F. J. Varela. Comparison of hilbert transform and wavelet methods for the analysis of neuronal synchrony. *Journal of Neuroscience Methods*, 111(2):83–98, 2001. 94

[83] Y. LeCun, L. Bottou, B. Yoshua, and Patrick H. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 25

[84] A. D. Legatt, J. Arezzo, and H. G. Vaughan Jr. Averaged multiple unit activity as an estimate of phasic changes in local neuronal activity: effects of volume-conducted potentials. *J Neurosci Methods*, 2(2):203–17, 1980. 93

[85] M. S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Comput. Neural Syst.*, 9:53, 1998. A review of methods for spike sorting: the detection and classification of neural action potentials. 92

[86] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002. 95, 98

[87] N. H. Lovall and G. J. Suaning. Retinal neuroprosthesis: science fact of science fiction? *Expert Rev. Ophthalmol.*, 2(2):145–148, 2007. 14

[88] J. Ma, J. Theiler, and S. Perkins. Accurate on-line support vector regression. *Neural Computation*, 15:2683–2703, 2003. 54, 62

[89] H. Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006. 116

[90] L. Marple Jr. Computing the discrete-time "analytic" signal via fft. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 47(9):2600–2603, 1999. 94

[91] C. Mehring, M. P. Nawrot, S. C. de Oliveira, E. Vaadia, A. Schulze-Bonhage, A. Aertsen, and T. Ball. Comparing information about arm movement direction in single channels of local and epicortical field potentials from monkey and human motor cortex. *Journal of Physiology-Paris*, 98(4-6):498–506, 2004. 90, 91

[92] U. Mitzdorf. Current source-density method and application in cat cerebral cortex: investigation of evoked potentials and EEG phenomena. *Physiological Reviews*, 65(1):37–100, 1985. 91

[93] A. R. Moller. History of cochlear implants and auditory brainstem implants. *Adv. Otorhinolaryncol.*, 64:1–10, 2006. 1

[94] V. B. Mountcastle, W. H. Talbot, and H. H. Kornhuber. The neural transformation of mechanical stimuli delivered to the monkey's hand. In A. V. S. de Reuck and Julie Knight, editors, *Ciba Foundataion Symposium: Touch, Heat and Pain*, pages 325–351. London: Churchill, 1966. 8

[95] W. J. Nash. *The Population Biology of Abalone (Haliotis Species) in Tasmania. I. Blacklip Abalone (H Rubra) from the North Coast and the Islands of Bass Strait*. Sea Fisheries Division, Marine Research Laboratories-Taroona, Department of Primary Industry and Fisheries, Tasmania., 1994. 138

[96] C. Nicholson and J. A. Freeman. Theory of current source-density analysis and determination of conductivity tensor for anuran cerebellum. *Journal of Neurophysiology*, 38(2):356–368, 1975. 92

[97] M. A. L. Nicolelis. Actions from thoughts. *Nature*, 409:403–407, January 2001. 89

[98] M. A. L. Nicolelis. Brain–machine interfaces to restore motor function and probe neural circuits. *Nature Reviews Neuroscience*, 4(5):417–422, 2003. 1

[99] R. A. Normann, E. M. Maynard, P. J. Rousche, and D. J. Warren. A neural interface for a cortical vision prosthesis. *Vision Research*, 39:2577–2587, 1999. 17, 90

[100] L. G. Nowak and J. Bullier. Axons, but not cell bodies, are activated by electrical stimulation in cortical gray matter. *Experimental Brain Research*, 118(4):477–488, 1998. 121

[101] R. K. Pace and R. Barry. Quick computation of regressions with a spatially autoregressive dependent variable. *Geographical Analysis*, 29(3):232–247, 1997. 139

[102] R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33(3):291–297, 1997. 138

[103] C. C. Petersen. The functional organization of the barrel cortex. *Neuron*, 56:339–355, 2007. 21, 22

[104] C. C. Petersen, T. T. Hahn, M. Mehta, A. Grinvald, and B. Sakmann. Interaction of sensory responses with spontaneous depolarization in layer 2/3 barrel cortex. *Proc Natl Acad Sci U S A*, 100(23):13638–43, 2003. eng. 3, 128

[105] K. H. Pettersen, A. Devor, I. Ulbert, A. M. Dale, and G. T. Einevoll. Current-source density estimation based on inversion of electrostatic forward solution: effects of finite extent of neuronal activity and conductivity discontinuities. *J. Neurosci. Meth.*, 154:116–133, 2006. 92

[106] K. H. Pettersen, E. Hagen, and G. T Einevoll. Estimation of population firing rates and current source densities from laminar electrode recordings. *J Comput Neurosci*, 24(3):291–313, Jun 2008. 92

[107] J. Platt. Fast training of SVMs using sequential minimal optimization. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1999. 29, 49, 53, 54

[108] VXI plug & play System Alliance. Vpp-4.3: The visa library, October 2008. 119

[109] G. Qin and L. Hotilovac. Comparison of non-parametric confidence intervals for the area under the ROC curve of a continuous-scale diagnostic test. *Statistical Methods in Medical Research*, 17(2):207, 2008. 122

[110] J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243. Morgan Kaufmann, 1993. 139

[111] M. J. Rasch, A. Gretton, Y. Murayama, W. Maass, and N. K. Logothetis. Inferring spike trains from local field potentials. *J. Neurophysiol.*, 99:1461–1476, 2008. 93

[112] J. P. Rauschecker and R. V. Shannon. Sending sound to the brain. *Science*, 259(8):1025–1028, 2002. 2

[113] J. Rickert, S. C. de Oliveira, E. Vaadia, A. Aertsen, S. Rotter, and C. Mehring. Encoding of Movement Direction in Different Frequency Ranges of Motor Cortical Local Field Potentials. *Journal of Neuroscience*, 25(39):8815–8824, 2005. 90, 91

[114] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 53

[115] M. G. Rosenblum, A. S. Pikovsky, J. Kurths, C. Schäfer, and P. A. Tass. Phase synchronization: from theory to data analysis. *Handbook of Biological Physics*, 4:279–321, 2001. 93, 94

[116] M. Rudolph and A. Destexhe. The discharge variability of neocortical neurons during high-conductance states. *Neuroscience*, 119(3):855–73, 2003. eng. 18

[117] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 53

[118] R. N. S. Sachdev, F. D. Ebner, and C. J. Wilson. Effect of subthreshold up and down states on the whisker-evoked response in somatosensory cortex. *Journal of Neurophysiology*, 92:3511–3521, 2004. 3, 18, 22

[119] L. Sachs and J. Hedderich. *Angewandte Statistik: Methodensammlung mit R.* Springer, 2006. 122

[120] C. Saunders, A. Gammerman, and V. Vovk. Ride regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998. 36, 65

[121] H. Scherberger, M. R. Jarvis, and R. A. Andersen. Cortical local field potential encodes movement intentions in the posterior parietal cortex. *Neuron*, 46(2):347–354, 2005. 91

[122] E. M. Schmidt, M. J. Bak, F. T. Hambrecht, C. V. Kufta, D. K. O'Rourke, and P. Vallabhanath. Feasibility of a visual prosthesis for the blind based on intracortical microstimulation of the visual cortex. *Brain*, 119:507–522, 1996. 2, 17, 18, 129

[123] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, September 1999. 26

[124] B. Schölkopf and A. J. Smola. *Learning with kernels*. MIT Press, 2002. Learning with kernels. 25, 27, 28, 47, 76, 95, 97

[125] A. B. Schwartz. Cortical neural prosthetics. *Annual Review of Neuroscience*, 27(1):487–507, 2004. 89

[126] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran. Brain-controlled interfaces: Movement restoration with neural prosthetics. *Neuron*, 52(1):205–220, 2006. 1, 90

[127] M. Seeger. Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. In Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, editors, *NIPS*, pages 603–609. The MIT Press, 1999. 76

[128] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press New York, NY, USA, 2004. 95, 96, 104

[129] I. Shphigelman, Y. Singer, R. Paz, and E. Vaadia. Spikernels: predicting arm movements by embedding population spike rate patterns in inner-product spaces. *Neural Computation*, 17:671–690, 2005. 95, 98

[130] W. Singer and C. M. Gray. Visual Feature Integration and the Temporal Correlation Hypothesis. *Annual Reviews in Neuroscience*, 18(1):555–586, 1995. 94

[131] A. J. Smola. *Learning with Kernels*. PhD thesis, Technische Universität Berlin, 1998. 29

[132] A. Srinivasan and R. D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37–57, 1999. 139

[133] E. Stark and M. Abeles. Predicting movement from multiunit activity. *J. Neurosci.*, 27:8387–8394, 2007. Predicting movement from multiunit activity. 90, 92, 93, 107

[134] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2nd edition, 2001. 119

[135] G. E. Tassiker. US Patent 2760483, 1956. 14

[136] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz. Direct cortical control of 3d neuroprosthetic devices. *Science*, 296(5574):1829–1832, 2002. 90

[137] E. Timofeeva, C. Mérette, C. Émond, P. Lavallée, and M. Deschênes. A map of angular tuning preference in thalamic barreloids. *Journal of Neuroscience*, 23(33):10717–10723, 2003. 22

[138] I. W. Tsang, J. T. Kwok, and P. M. Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6:363–392, 2005. 31

[139] I. W. Tsang, J. T. Kwok, and K. T. Lai. Core Vector Regression for Very Large Regression Problems. In *Proceedings of the* 22^{nd} *International Conference on Machine Learning*, pages 913–920, 2005. 31

[140] R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. Technical Report SOR-97-21, Princeton University, 1997. 29

[141] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000. 76

[142] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, second edition, 1999. 25, 26, 53, 76

[143] S. V. N. Vishwanathan, N. N. Schraudolph, and A. J. Smola. Step size adaptation in reproducing kernel Hilbert space. *JMLR*, 7:1107–1133, 2006. 54, 60

[144] D. A. Wagenaar, R. Madhavan, J. Pine, and S. M. Potter. Controlling Bursting in Cortical Cultures with Closed-Loop Multi-Electrode Stimulation. *Journal of Neuroscience*, 25(3):680–688, 2005. 116, 131

[145] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. Nicolelis. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(6810):361–5, 2000. eng. 90

[146] J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In *Proc. of the 10th Int. Workshop on Artificial Intelligence and Statistics*, pages 413–420, 2005. 60

[147] R. H. Whittington, L. Giovangrandi, and G. T. A. Kovacs. A closed-loop electrical stimulation system for cardiac cell cultures. *Biomedical Engineering, IEEE Transactions on*, 52(7):1261–1270, 2005. 116, 131

[148] M. L. Winston. *The Biology of the Honey Bee*. Harvard University Press, first edition edition, 1991. 9

[149] T. A. Woolsey and H. van der Loos. The structural organization of layer iv in the somatosensory region (s1) of mouse cerebral cortex. *Brain Research*, 17(2):205–242, 1970. 19

[150] World Health Organization. *Fact Sheet No. 282: Magnitude and causes of visual impairment*, November 2004. 13

[151] G. Zanghirati and L. Zanni. A parallel solver for large quadratic programs in training support vector machines. *Parallel Computing*, 29:535–551, 2002. 29

[152] L. Zanni, T. Serafini, and G. Zanghirati. Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems. *Journal of Machine Learning Research*, 7:1467–1492, 2006. 29

[153] E. Zrenner. Will retinal implants restore vision? *Science*, 259(8):1022–1025, 2002. 1, 14

[154] E. Zrenner, D. Besch, K. U. Bartz-Schmidt, F. Gekeler, V. P. Gabel, and C. Kutt. Subretinal chronic multi-electrode arrays implanted in blind patients. *Investigative Ophthalmology and Visual Science*, 47(5):1538, 2006. 1, 14, 15