Transferring Insights on Biological Sleep to Robot Motor Skill Learning

Übertragung von Erkenntnissen über biologischen Schlaf auf Konzepte für das Erlernen von motorischen Fähigkeiten von Robotern

Bachelor-Thesis von David Rother aus Aschaffenburg

Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: M.Sc. Dorothea Koert

TECHNISCHE UNIVERSITÄT DARMSTADT

Transferring Insights on Biological Sleep to Robot Motor Skill Learning
Übertragung von Erkenntnissen über biologischen Schlaf auf Konzepte für das Erlernen von motorischen Fähigkeiten von Robotern

Vorgelegte Bachelor-Thesis von David Rother aus Aschaffenburg

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: M.Sc. Dorothea Koert

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den November 2, 2017

(D. Rother)

Abstract

This thesis presents a modular learning architecture based on insights from neurobiological sleep and the mixture of experts algorithm. In particular this thesis focuses on learning scenarios, where a complete dataset is not present from the start and can not be stored between training sessions. The proposed framework organizes learning in day-night cycles. Data is only recorded during the day by interaction with the environment and optimization and restructuring of internal models takes place during night, where no interaction with the environment is possible and only usage of the data of the previous day is allowed. An online learning algorithm is employed to learn local models in interaction with the environment. Pruning and optimization methods are used to refine a model in isolation and an extension to expectation maximization using trust regions is introduced. Experimental evaluations show that the proposed learning architecture performs significantly better if not all of the data can be stored.

Zusammenfassung

Diese Thesis präsentiert eine modulare Lernarchitektur basierend auf Erkentnissen aus dem neurobiologischem Schlaf und dem Mixture of Experts Algorithmus. Ein besonderer Fokus in der Thesis liegt auf Lernszenarios, bei denen nicht von Anfang an alle Datenpunkte bekannt sind und auch nicht abgespeichert werden können zwischen einzelnen Trainingsiterationen. In dem vorgeschlagenem System wird der Lernprozess in Tag und Nacht Zyklen strukturiert. Nur durch Interaktion mit der Umwelt können neue Datenpunkte aufgenommen werden, während nachts der Optimisierungsprozess stattfindet und interne Modelle restrukturiert werden können. Für die Dauer der Nachtphasen ist es nicht möglich mit der Umwelt zu interagieren, sodass für diese Prozesse nur das Nutzen der Daten des vorangegangenen Tages möglich ist. Um bereits einen Lernprozess im Wachzustand zu ermöglichen wird ein on-line Algorithmus verwendet, der lokale Modelle durch die Interaktion mit der Umwelt erlernen kann. Interne Modelle werden Nachts durch Pruning- und Optimisierungsmethoden verfeinert und es wird eine Erweiterung des Expectation Maximization Algorithmus mit Vertrauensregionen für die Optimisierungsphase eingeführt. Eine Auswertung von Experimenten zeigt, dass das vorgeschlagene System zum Lernen bessere Ergebnisse vorweist, wenn nicht alle Datenpunkte abgespeichert werden können.

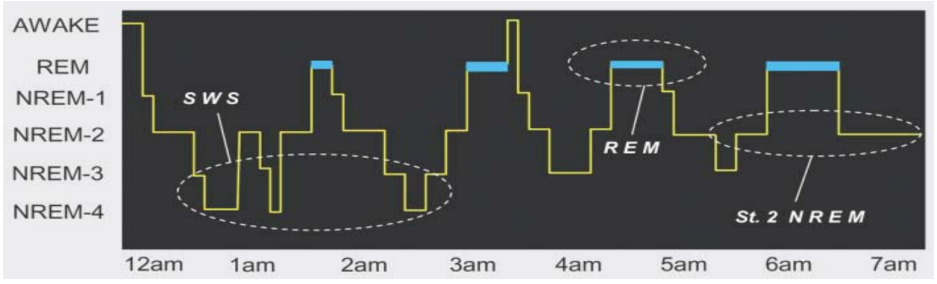
Inhaltsverzeichnis

1 Introduction

1.1 Motivation

Future robots are expected to face an increasing number of scenarios that require the ability of autonomous acting and decision making as teleoperation of a robot is unfeasible for many tasks. In particular, to master such scenarios autonomously, a robot needs to train various abilities beforehand and should additionally be able to adjust its inner models to the situation at hand. Learning multiple skills in one session is exceptionally hard in dynamic environments and the need to be able to weave previous knowledge together with new one arises. Research in neurobiology suggests that humans integrate newly acquired knowledge during sleep and show better performance after sleep [1]. Many approaches inspired by neurobiology for neurobiologic computing have been made. These approaches focused on either real time integration through online learning [2, 3] or two step optimization with learning of forward and backward models [4], but do not explicitly take concepts from biological sleep and its influences onto learning into account. While advanced algorithms for learning a single task in one sweep exist, only recently researchers started to investigate methods for long-term learning concepts [5], where models can integrate new knowledge acquired in a multi session learning scenario without either forgetting parts of previous sessions or having to store the data of previous sessions. Even though neural networks have been used before in wake sleep algorithms [4], in this thesis we explore mixture models as an alternative to a complex monolithic structure. A modular approach with mixture models could allow for dynamic model sizes or an easier recovery of knowledge from local models.

(a)



(b)

Figure 1.1: (a) During motor skill learning humans profit from multiple neurobiological processes that occur during different sleep phases [6]. (b) Transferring insights about these neurobiological processes to robots [7] could possibly enable more efficient robot motor skill learning during multiple sessions.

In particular, this thesis aims to integrate newly acquired knowledge into a mixture of experts model while minimizing interference of new information. Furthermore a day night cycle could possibly be beneficial, when the model has to be able to adapt to new situations throughout the day and can use only the data of the last day to maximize performance in a sleep stage. At last the model should be able to adapt it's size to the complexity of the task such that an overparametrization is not needed from the start.

1.2 Content

This thesis is structured in seven chapters. Chapter 2 provides an overview on mixture of experts (MoE) that has a structure already used in neuroscience [2] and the EM algorithm for MoE. Then current findings on sleep and its influences on the performance on tasks are explained with some background in neurobiology. A popular approach for motor skill learning is shown and Chapter 3 closes with learning algorithms to minimize interference and find online updates for MoE. In Chapter 4 the new framework is presented and how a wake-sleep could work. Further it will be expanded on some statistical background for interference minimization. Chapter 5 presents the developed framework for wake sleep algorithms. Chapter 6 will show the evaluation of a toy task with usage of the new framework and contains a discussion of the results. Chapter 7 concludes the thesis by showing possible future work and how the presented framework could be extended.

2 Foundations

In this chapter we start with an introduction of mixture of experts and explain why it is used. We show the necessary equations in use and show the derivations for the basic assumptions of the model. Afterwards we show a possible optimization technique for the MoE structure with linear experts. We conclude the chapter by showing how to optimize mixture of experts via Expectation Maximization (EM).

2.1 Mixture of Experts

At first we present the mathematical framework at use with mixture of experts. Mixture of Experts has been used on a wide variety of tasks since they were introduced roughly 25 years ago. Mixture of Experts can handle many classification and regression problems and has found applications in many areas such as healthcare, finance, surveillance, and recognition. The system itself is very flexible and allows for an easy incorporation of newer methods. This can be seen as for example in the development of new gating algorithms like sparse gating for large neural networks [8]. Because the framework is statistically sound many techinques were already combined with mixture of experts, such as Gaussian Processes, Support Vector Machines, Neural Networks or other Hidden Markov models. These model can be trained by well studied techniques such as the EM algorithm. By combination of these methods the mixture of experts architecture is still competetive for regression with nonstationary and piecewise continous data. It also able to learn non-linear data for classification problems as long as distinct patterns exist [9].

2.1.1 Mathematical Framework

The main idea of mixture of experts is to have a two layer setup, where the first stage is a gating network and the second stage consists of the experts [10]. This model can be viewed as a tree structure as is depicted in Figure 2.1 . The gating divides the complete input space and does a soft split of the responsibilities for each section for the experts. This allows for learning local models which are independent of each other.

We only describe the original regression model as we have to deal with continuous regression data. Using gaussians instead of a softmax function, the gate makes a soft split of the input space and assigns each region to an expert, which in our case will be linear models, so we have in total a Gaussian Mixture Model (GMM). The experts can then learn the much simpler parameters in those subspaces weighted by the assigned responsibility from the gating network for each data point. For some input $\boldsymbol{x}$ each expert calculates a prediction $\hat{\boldsymbol{y}}_j$. The gating then calculates a responsibility of each expert given $\boldsymbol{x}$ and computes the final prediction by combining the predictions of the experts. The following equations and explanations are given by [9]. The output of our network is computed by building the sum over the expert predictions weighted by the gating divided by the sum of the weights of the gating. As general model for optimization we are then given this conditional mixture of gaussians

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta}) = \sum_{j=1}^{K} g(\boldsymbol{x},\boldsymbol{v}_j) P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_j), \tag{2.1}$$

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_j) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left\{-\frac{1}{2}[\boldsymbol{y}-f_j(\boldsymbol{x},\boldsymbol{w}_j)]^T \boldsymbol{\Sigma}_j^{-1}[\boldsymbol{y}-f_j(\boldsymbol{x},\boldsymbol{w}_j)]\right\}, \tag{2.2}$$

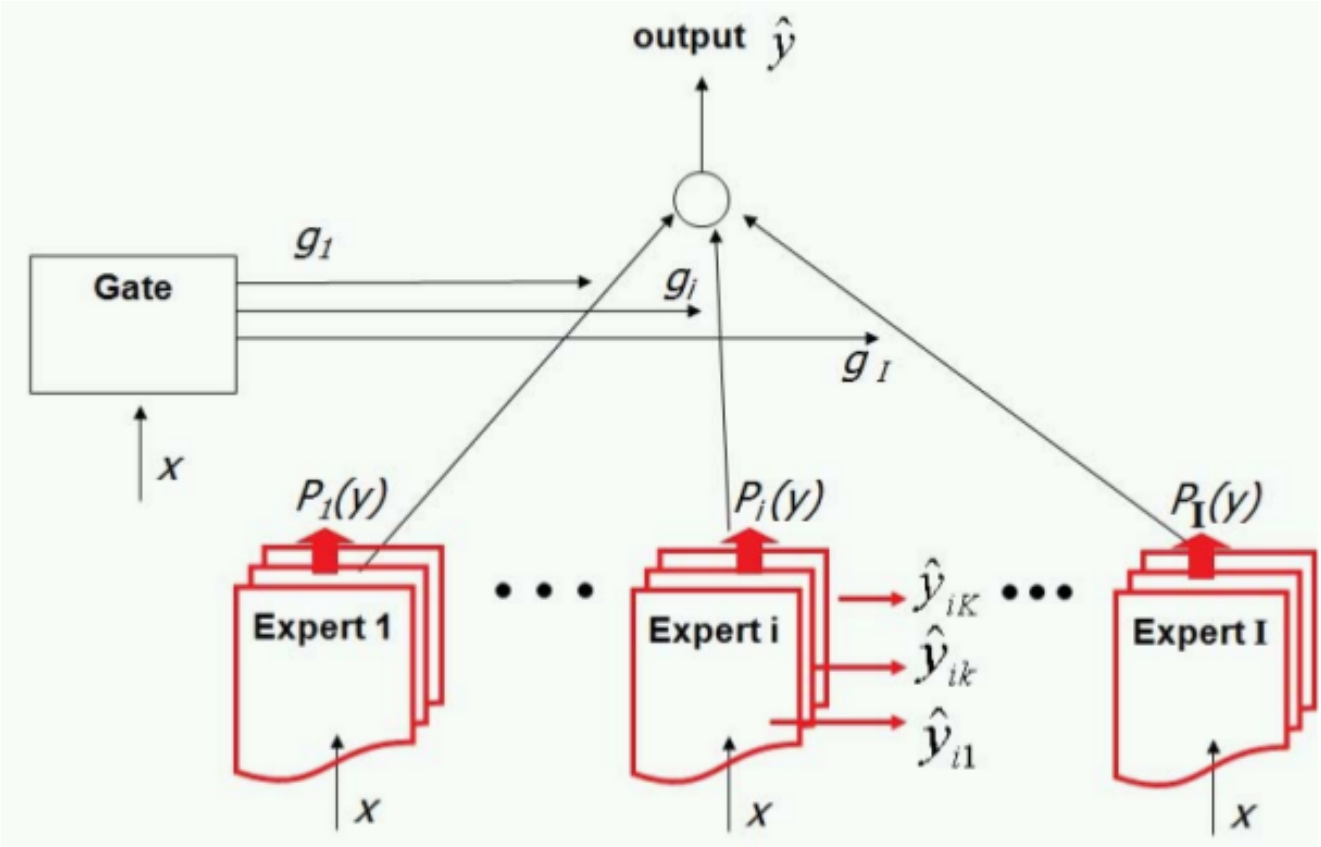


Figure 2.1: General architecture for MoE Systems. A number of I local experts (red) produces seperate predictions y_k for a given input x. A centralized gating network (left) assigns some factor g to each prediction y_k and applies weights to each expert to receive an optimal result by adding the weighted predictions together [9].

where g is the gating function, $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta})$ is the probability that a point $\boldsymbol{y}$ was produced from some input $\boldsymbol{x}$ and our model given by the parameters $\boldsymbol{\Theta}$. We use then this probabilistic gating network equation and add a maximization coefficient α.

$$g_j(\boldsymbol{x},\boldsymbol{v}) = \alpha_j \frac{P(\boldsymbol{x}|\boldsymbol{v}_j)}{\sum_i \alpha_i P(\boldsymbol{x}|\boldsymbol{v}_i)}, \quad \sum_j \alpha_j = 1, \alpha_j \geq 0, \tag{2.3}$$

$$P(\boldsymbol{x},\boldsymbol{v}_j) = \alpha_j(\boldsymbol{v}_j)^{-1} b_j(\boldsymbol{x}) exp\{c_j(v_j)^T t_j(\boldsymbol{x})\}, \tag{2.4}$$

where $\boldsymbol{v} = \{\alpha_j, \boldsymbol{v}_j, j = 1,...,K\}, t(\boldsymbol{x})$ is a vector of sufficient statistics and the $P(\boldsymbol{x}|\boldsymbol{v}_j)$'s are density functions from the exponential family. Since we use a gaussian probability distribution, our pdf is

$$P(\boldsymbol{x},\boldsymbol{v}_j) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{m}_j)\right\} \tag{2.5}$$

From a probabilistic point of view we can see that the gating function g is the posterior probability that an input $\boldsymbol{x}$ is assigned to the j-th expert. In our case we use the Gaussian function and thus we get through Bayes' Rule

$$g_j(\boldsymbol{x},\boldsymbol{v}) = P(j|\boldsymbol{x}) = \frac{\alpha_j P(\boldsymbol{x},\boldsymbol{v}_j)}{P(\boldsymbol{x},\boldsymbol{v})}, \quad P(\boldsymbol{x},\boldsymbol{v}) = \sum_i \alpha_i P(\boldsymbol{x}|\boldsymbol{v}_i). \tag{2.6}$$

By inserting the modified gating equation into our model Equation 2.1 we obtain

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta}) = \sum_j \frac{\alpha_j P(\boldsymbol{x}|\boldsymbol{v}_j)}{P(\boldsymbol{x},\boldsymbol{v})} P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_j). \tag{2.7}$$

2.1.2 EM-algorithm

One can do an ML estimation on the model directly and derive the EM algorithm for $\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta})$. But to be able to solve the maximization analytically one needs to rewrite Equation 2.7 as

$$P(\boldsymbol{y},z) = P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta})P(\boldsymbol{x},\boldsymbol{v}) = \sum_j \alpha_j P(\boldsymbol{x}|\boldsymbol{v}_j)P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_j). \tag{2.8}$$

To get the parameters $\alpha_j, \boldsymbol{v}_j, \boldsymbol{\theta}_j$ of the model we do a maximum likelihood estimate on our model based on $L = \sum_t \ln(P(\boldsymbol{y}^{(t)},\boldsymbol{x}^{(t)}))$, where t denotes the t-th data point in the dataset. From there we can use the following E- and M-Step to perform our EM-algorithm.

E-Step

In the E-step we compute the belief of the gating how likely each expert is responsible of each data point. This results in the matrix h, which will be used in the M-step. We define our objective function for the experts and the gating in which we try optimize on the given data. For the experts we get

$$Q_j^e(\boldsymbol{\theta}_j) = \sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})\ln(P(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)},\boldsymbol{\theta}_j)), j = 1,...,K; \tag{2.9}$$

And the objective function for the gating is:

$$Q^g(\boldsymbol{v}) = \sum_t \sum_j h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})\ln(P(\boldsymbol{x}^{(t)}|\boldsymbol{v}_j)), j = 1,...,K; \tag{2.10}$$

Which can be decomposed further into:

$$Q^g(\boldsymbol{v}) = \sum_t \sum_j h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})\ln(\alpha_j), \text{ with } \alpha = \alpha_1,...,\alpha_K \tag{2.11}$$

The h-matrix of the objective functions is then computed as follows

$$h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) = \frac{\alpha_j^{(k)} P(\boldsymbol{x}^{(t)}|\boldsymbol{v}_j^{(k)})P(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)},\boldsymbol{\theta}_j^{(k)})}{\sum_i \alpha_i^{(k)} P(\boldsymbol{x}^{(t)}|\boldsymbol{v}_i^{(k)})P(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)},\boldsymbol{\theta}_i^{(k)})}. \tag{2.12}$$

M-Step

In the M-step we use the computed beliefs stored in the matrix h to maximize the expert nets and the gating. Because our gating function is a gaussion pdf, which is a function from the exponential family, we can solve analytically the maximization of the net. We find a new estimate for $j = 1, ..., K$ by applying

$$\begin{aligned} \boldsymbol{\theta}_j^{(k+1)} = \arg\max_{\boldsymbol{\theta}_j} Q_j(\boldsymbol{\theta}_j), \quad \boldsymbol{v}_j^{(k+1)} = \arg\max_{\boldsymbol{v}_j} Q_j^g(v_j), \\ \alpha^{(k+1)} = \arg\max_{\alpha} Q^{\alpha}, \quad s.t. \sum_j \alpha_j = 1. \end{aligned} \tag{2.13}$$

Because we use a gaussian distribution the update of the parameters becomes for the maximization coefficients or priors in a bayesian view

$$\alpha_j^{(k+1)} = \frac{1}{N} \sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}). \tag{2.14}$$

The general update equation for the other gating parameters becomes

$$\boldsymbol{v}_j^{k+1} = \frac{\sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) t_j(\boldsymbol{x})(\boldsymbol{x}^{(t)})}{\sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})}. \tag{2.15}$$

Derived for gaussians the means of the gating then become

$$\boldsymbol{m}_j^{(k+1)} = \frac{1}{\sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})} \sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \boldsymbol{x}^{(t)}. \tag{2.16}$$

And the variances of the gating become:

$$\boldsymbol{\Sigma}_j^{(k+1)} = \frac{1}{\sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})} \sum_t h_j^{(k)}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})(\boldsymbol{x}^{(t)} - \boldsymbol{m}_j^{(k+1)})(\boldsymbol{x}^{(t)} - \boldsymbol{m}_j^{(k+1)})^T. \tag{2.17}$$

3 Related Work

This chapter presents the biologic fundamentals of sleep and memory and how mammals can learn through sleep. We expand on mechanisms on a neurobiologic scope that are responsible for learning and the basic idea how memory forms and develops especially in the context of aquiring motor skills. After the biological insight we then turn to a computer scientific view of motor skills in the context of robotics. We explore past approaches drawing biologic analogies for learning and look for similarities to our previously found neurobiologic learning processes to mesh biology with computer science approaches.

3.1 Memory and Brain Plasticity

This section cover the biologic view on neurobiologic learning and the effects of sleep on learning. We focus on the mechanisms involved to develop new motor skills and refine them during sleep. Since there is a need to understand what happens during sleep and what kind of mechanisms are active, we need to define what memory is on an abstract level and how it can be altered. After showcasing this we then explain how sleeping utilizes the different mechanisms to consolidate memory. Consolidation of memory usually refers to how much it can be altered in a sense that older memories are not as easily changed as new ones. The main concept we use to explain changes made to the brain alongside memory will be the one of brain plasticity. Brain plasticity is an umbrella term to describe lasting changes to the brain. This generally means that the brain may be subject to changes during every stage of life and is not losing the ability to restructure itself. To manipulate brain plasticity and thus memories the brain can either potentiate or depress synapsis. Potentiation refers to an increase in the likelihood of a synapsis firing, while depression refers to a decrease of the likelihood of a synapsis firing [6].

3.1.1 Memory Categories

While many different classification schemes exist for human memory, the most popular model makes a distinction between declarative memory and non declarative memory as depicted in Figure 3.1. Declarative memory is the explicit memory that is consciously available for us as in one can actively recall these memories. One can further subdivide it into episodic memory and semantic memory. Episodic memory represents our knowledge about events that happened in the past and experiences. Semantic memory on the other hand represents the factual and concept based knowledge. Going away from the conscious memory we move forward to the non-conscious memory. This part is largely described as non-declarative memory. Although the knowledge and memories are implicit they are used constantly, for example while breathing or doing slight movements. One can not recall the exact muscle stimulation needed for a certain task but can nevertheless perform said task by subconsciously accessing that knowledge. It can be said that non-declarative memory is the part of memory where humans store their knowledge of how to perform a certain skill or achieve a task. [6] The previous memory categories are also known as long term memory and usually mammals also posses working memory, where information is stored for a short amount of time. This information is maintained for a short amount of time to build blocks in working memory for further information processing and thus helping to achieve more complex goals. [11] So while we can model those distinct categories of memory it is important to keep in mind that every task requires a combination of the memory groups as they work tightly together and basically never work in isolation. [6]

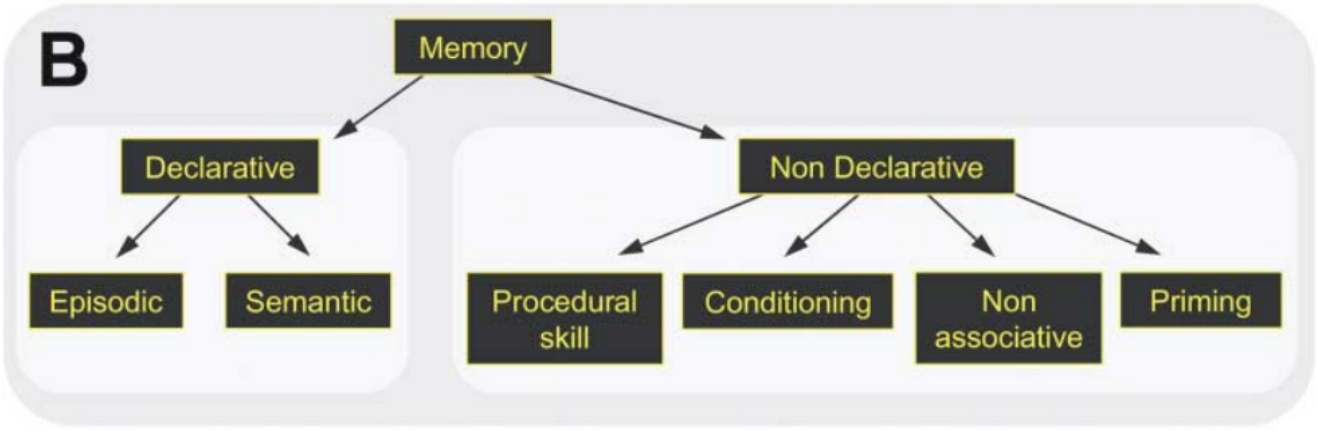


Figure 3.1: System to categorize memories in the brain. On the left side are declarative memories formed by factual knowledge of events or other facts, which can be remembered actively. On the right side is the Non Declarative memory consisting of implicit knowledge, where a human can not recall a skill directly but use it [6].

3.1.2 Memory Stages

Memory is not obtained in a single step so we present a theory how memory manifests over time as can be seen schematically in Figure 3.2. New Information is acquired during wakefulness and it is stored temporarily in the working memory. This happens as the human interacts with the environment and gets active feedback. [11] Afterwards memory is transferred from the working memory to long term memory. This happens through translocation and integration of memory. Coming alongside then is consolidation of memory. Memory consolidation is a process where memory becomes more and more resistant to interference or new memory that competes with old memory. This leads to two main influences on memory by consolidation. Stabilization and enhancement happen over time but stabilization primarily happens during wakefulness, while enhancement mainly occurs during sleep. Enhancement of memories covers a wide array of things that can happen. It integrates newly acquired memory, translocates in the brain and takes a more efficient structure for storing using less energy and can even erase some parts of our memory. Erasure or loss of memory are important to forget for example useless details of the past, which are not crucial for future tasks and another benefit is that the brain needs less energy as it does have to stabilize the potentiation level of less synapsis [6].

3.2 Sleep

In this section we first define sleep. We present the different phases of sleep and when and how often they occur. We expand on the different characteristics of the sleep phases and explain some mechanisms that happen during sleep that could influence the overall learning process. In the end more evidence is presented how sleep is related to learning and especially motor skill. Further we explore which manipulations of memory and brain plasticity in sleep are important to achieve a lasting change in brain plasticity with a focus on motor skills for mammals.

3.2.1 Definition of Sleep

Before we can talk about sleep we provide a basis definition of sleep to have common ground while discussing it. Sleep in general is a phase of external rest during which the power of the body is restored.

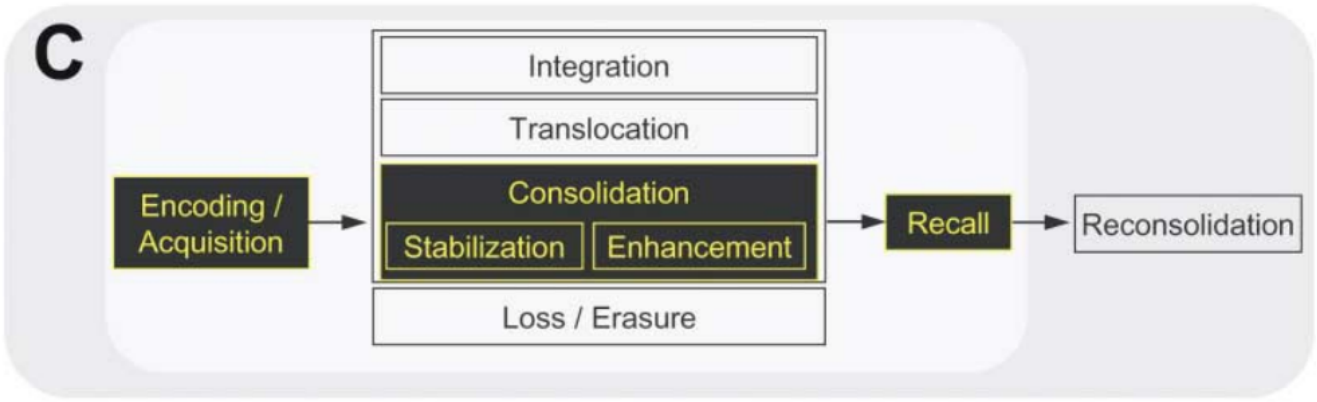


Figure 3.2: Model for acquisition of new memory. At first memory is obtained and encoded during wakefulness. Then it is integrated into the rest of the memory and the memory manifests itself through consolidation. Memory becomes more consolidated and gains longetivity through repeated recall and the following reconsolidation of the same [6].

It can be roughly subdivided into two major subphases which are NREM sleep and REM sleep. NREM stands for Non Rapid Eye Movement and is officially divided into three stages but is still seperated into 4 stages in many scientific researches. REM stands following from this for Rapid Eye Movement [12].

3.2.2 Sleep Phases

Human sleep does not run through each sleep phase once but rather happens in multiple cycles of about 90 minutes each. In the earlier night stages three and four of NREM sleep dominate, which indicates a very deep sleep early in the night. At later times stage two NREM sleep and REM sleep seem to dominate. So while there are 90 minute cycles throughout the night, the phases do not equal each other and have different ratios of occurence for NREM and REM sleep.[13] The usual distribution of sleep phases for an adolescent can be seen in Figure 3.3. It is worth highlighting that the ratio of NREM sleep to REM

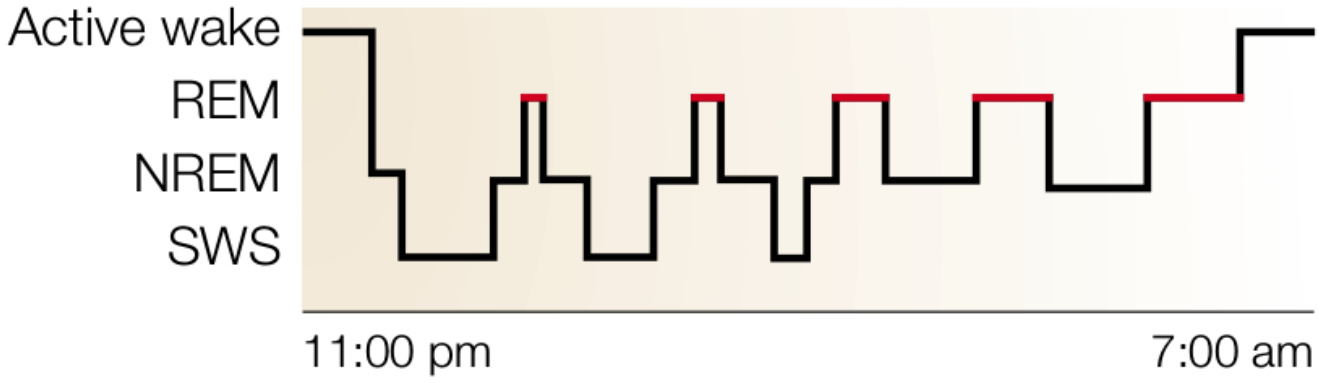


Figure 3.3: The common cycle of the sleep phases during the night for adolescent humans. Humans iterate between SWS, NREM and REM sleep stages, where at earlier stages of the night slow wave sleep dominates (NREM 3) and shifts continuously to more REM sleep at later sleep stages. [14]

sleep is not the same throughout the life of an individual. At the earliest stages of life a baby only shows

brain activity patterns of REM sleep because the slow wave sleep system seems not to be active yet. [15] For about the last two decades the idea became prominent that sleep can help with critical learning and memory building. This idea was further pushed by several studies that showed that after practicing a visual discrimination or motor skill task, a significant increase in performance could be measured on the same task after a certain amount of sleep, while no increased performance was found in an equally long wake time. The main areas in the brain associated with neuromodulation of the brain state in sleep are schematically shown in Figure 3.5 and brain states during each phase are described in Figure 3.6.

3.2.3 NREM sleep

As mentionend above NREM sleep can be divided into three and sometimes still into four categories. The first stage is NREM 1 and marks the transition between wakefulness and sleep. In the Electroencephalography (EEG) it can be seen that the brain waves transition from relatively unsynchronized beta and gamma brain waves, which are common for wakefulness, to the more synchronized alpha waves and then to theta waves.
Alpha waves usually occur during deep relaxation and symbol the transition to sleep in the EEG. Then once a subject falls asleep we can measure the theta waves [12]. In addition all sensory input from the sensory cortex gets blocked and the brain is in a state of isolation from the outside world during sleep as shown in Figure 3.4. For these changes in brain state to happen many neuromodulators are at work. At

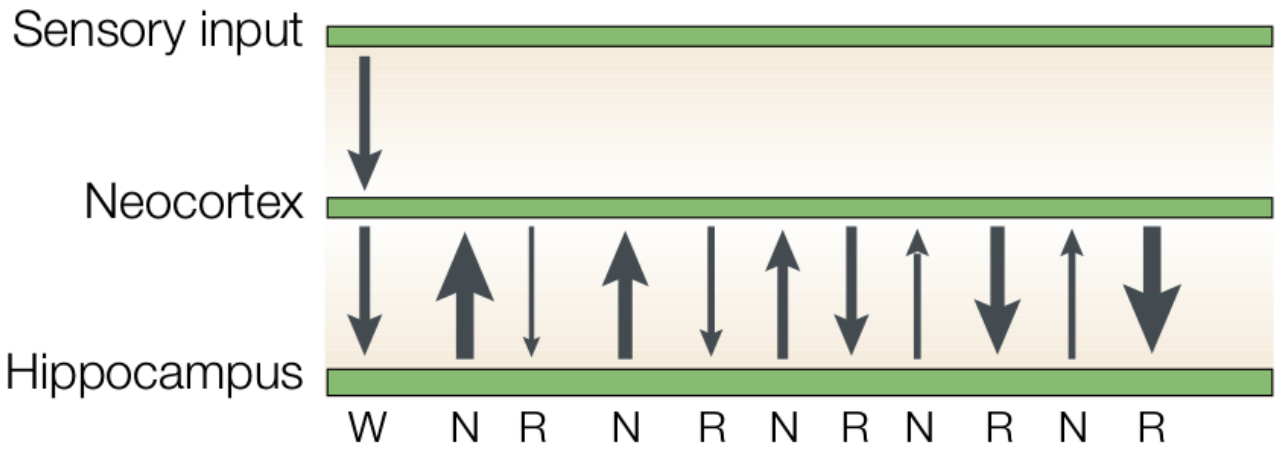


Figure 3.4: Sensory input gets blocked from the neocortex as soon as the brain enters a sleep state. Additionally the Hippocampus sends weaker signals as sleep goes on throughout the night and the neocortex simulates more sensory input [14].

the start of thalamortical oscillations we begin to observe activating inputs from different neurons, such as noradrenergic, histaminergic, aminergic and orexinergic neurons. After the brain entered the NREM sleep stage those thalamortical inputs gate and diminish the transmission of information from thalamic sensory relay areas [14]. Those inputs seem to diminish as circadian and homeostatic signals from the hypothalamus take over and can be seen as a sleep-wake switch [16].
The next sleep phase following NREM 1 sleep is NREM 2 sleep. The characteristic brain waves during this sleep phase are the theta waves. Additionally K-complexes and sleep spindles can be measured, which are a short burst of high frequency waves in the brain [12]. These special waveforms result from prolonged hyperpolarization of cortical neurons and are followed by a depolarized phase. This intense bursts of firing in the neurons play an important role in the plasticity process of the brain, learning and memory [14]. The synaptic homeostasis hypothesis (SHY) suggests that sleep scales down synaptic

strength to a certain baseline, while plastic processes occur during wakefulness that increase net synaptic strength, and the brain can thus maintain its plasticity [17]. So our synaptic renormalization should happen through synaptic depression during sleep and in specific NREM sleep. The hyperpolarization of certain neurons has then the effect to renormalize the firing rate of those neurons to eliminate higher firing rates by chance [18].
As the last sleep stage comes NREM 3 sleep which is also referred to as slow wave sleep (SWS). During NREM 3 sleep we measure delta waves in the brain. In earlier research NREM 3 was divided into NREM 3 and 4 stages but because of the brevity of NREM 3 stage alone the two became merged into one phase. During NREM 3 sleep cholinergic neurons from the basal forebrain remain inactive and the inactivity is linked to the increase in delta waves [19]. The cholinergic laterodosal tegmental nucleus and pedunculopontine tegmental nucleus neurons also have low discharge rates in comparison to discharge rates of REM sleep or wake phases. This change to decreased activity of neurons in NREM also stems from the inhibition of thalamortical neurons by sustained GABA input from the thalamic reticular nucleus. [14, 20] In general it can be found that during NREM 3 dreams occur more frequently than in NREM 2 but are less vivid in comparison to dreams during the REM phase. It can also be seen that in the brain areas that were active during a learning task have an increased slow wave activity. The firing rate patterns of those neurons are similar to those in the wake phase but differ completely on a time scale as the firing is much more rapid there. [21] This could resemble the consolidation of memory acquired during the day and is likely linked to an increase in performance of tasks learned during the last wake phase. After NREM 3 has occured the slow wave activity decreases in areas with intense neuronal firing and normalizes throughout the night, which indicates a local learning process instead of a global one [21].

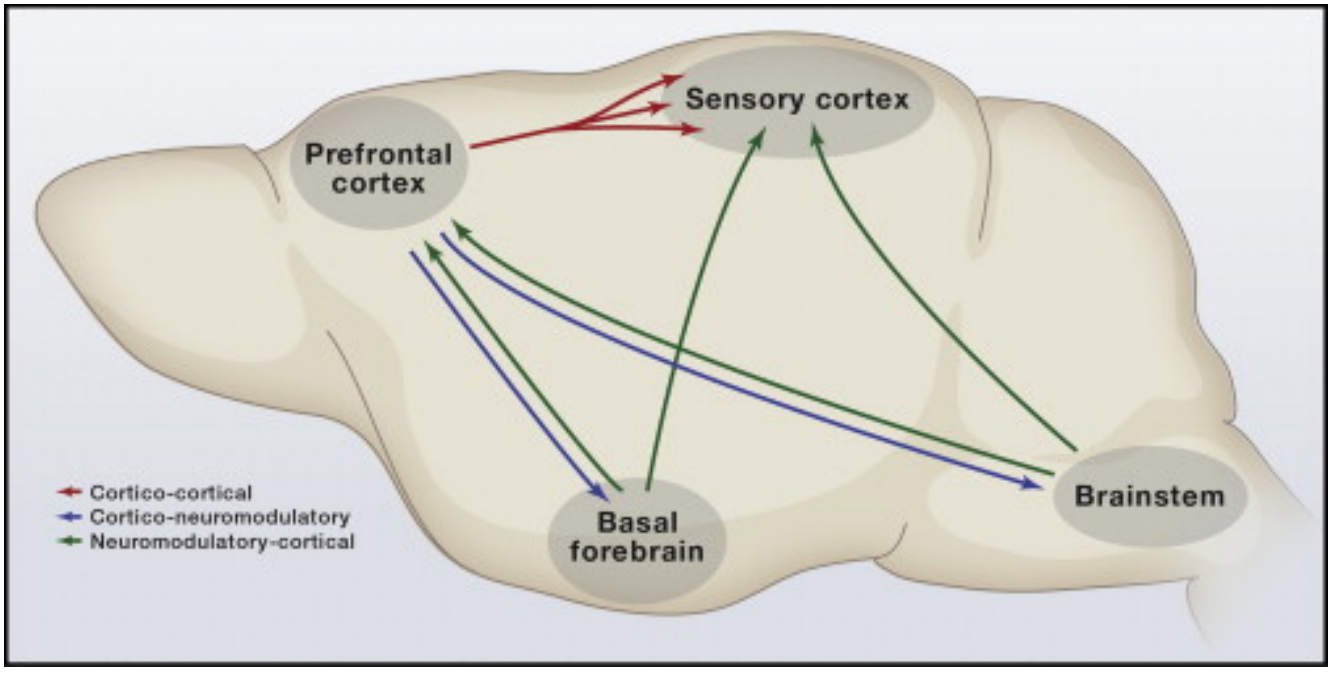


Figure 3.5: Schematic overview of the most important brain areas for neuromodulatory changes in the brain, where arrows show the major paths that connect the areas. Red lines indicate a cortico-cortical connection for information transmission. Green and blue lines indicate a pathway for neuromodulators to influence another region [20].

3.2.4 REM sleep

During the REM sleep phase humans have the most fluent and most vivid dreams. REM sleep is characterized by desynchronized waves, much like they occur during wakefulness, however while there is no difference in EEG patterns for REM and wakefulness we can distinguish the both by looking at the electromyogram (EMG), which measures the muscle tone. During wakefulness we can observe high muscle tone and we can move freely, while during wakefulness we observe a low muscle tone through the EMG indicating the inability to move freely [20]. In REM sleep neurons have similar firing patterns to activation patterns that were observable while learning a new motor skill task. Another feature is that during REM sleep not all connections do fire at the same time but there are regions which seemingly randomly replicate the firing patterns of the last wake phase. Because of the replay of memories in the REM phase it is largely considered to be responsible for integration of memory [21, 22, 20].

Behavioural state		**Wake**	**NREM**	**REM**
Cognitive consequences		Acquisition of information	Iteration of informatiion	Integration of information
Conscious experience	Sensation and perception	Vivid, externally generated	Dull or absent	Vivid, internally generated
	Thought	Logical progressive	Logical perseverative	Illogical bizarre
	Movement	Continuous voluntary	Episodic involuntary	Commanded but inhibited

Figure 3.6: Overview of brain states during the sleep phases compared to the wake phase. During sleep movement is largely inhibited and involuntary. Information gathered in a wake state is processed and integrated during sleep [14].

3.3 Motor Skills

This section handles the topic of motor skills. Motor skills and motor skill learning are fundamental skills of humans. For humans learning motor skills is part of the non-declarative memory as we can only memorize the movements but not the signals send to the muscles. For humans motor skills a largely about learning a complex sequence of movements and being able to adjust to external perturbations. It can also mean to improve ones reaction time or to train reflexes [1]. A concise definition for motor skills could for example look like this: The learned ability to bring about predetermined results with maximum certainty with a minimum outlay of energy and time [23]. With this definition we can start to minimize for certain aspects such as amount of energy used or the speed of task solving. Also consistency can be measured and maximized according to this definition. So motor skills are any physical activity to achieve a goal and the main difference between a basic motor skill and skilled performance in this context is the complexity of the movement. Furthermore a skilled performance usually consists of a sequence of basic motor skills to achieve a goal.

3.3.1 Categories of Motor Skills

Motor skills can be differentiated into two subcategories. The first category consists of the gross motor skills. Those are for humans a combination of large muscle actions resulting in coordinated movements.

This can be for example a human throwing a ball where the movement of the upper body, legs, and the arms only in the correct combination result in a correct, in the sense of how the throwing person intended the ball to fly, trajectory for the ball. Fine motor skill on the other side makes use of cooperation of small muscle groups and uses sensor feedback. For example while writing we use small muscle groups in the hand and fingers. We combine our knowledge of how to execute the task of writing with the visual and haptic feedback we get. This adjustment to feedback allows for precise movement.

3.3.2 Motor Skill Classification

Another way is to classify motor skills according to the type of movement. There are three different types of movement we seperate. The first group of movements build the discrete motor skills. Discrete motor skills can be identified by their brief duration and a distinct beginning and end. In this category one would for example put the skill to raise the arm. Obviously it is not possible to classify every motor skill to exactly only one class, hence the definition of the different classifications are unprecise to account for uncertainty. The second group of motor skills consists of serial motor skills. These are defined as a series or group of discrete skills strung together and they usually have a distinct beginning and end. To go on with the last example this would be raising the arm, grasping an object, lowering the arm and then release grasp of the object. The last group consists of continuous motor skills. Cotiuous motor skill are loosely defined by having no distinct end or beginning. For example this could be a rythmic type of movement. This classification scheme focuses largely on the act of the motor skills itself but one can also define them by focusing on the predictability of the environment. If we have a predictable environment or high control over it we call a motor skill a closed motor skill. A closed motor skill could be for example bowling where the athlete has a very predictable environment and can repeat each trial with the same circumstances. On the other hand there are open motor skills. Their key attribute is that there is little to no control over the environment or there is high uncertainty. For example driving in a rally could be considered as an open skill as the slope of the car can change a lot and the driver has to account for the different slopes which he does not know beforehand. It is also important that there can not be a discrete line drawn between open and closed motor skill as a closed motor skill could change into an open one depending on the situation at hand and vice versa.

3.3.3 Approaches to Motor Skill Learning

So how do we go about motor skill learning for robots? A common approach to learn motor skills is to extract motor primitives and learn those. A supervisor can decide which motor primitive or combination of motor primitives fit the situation best and then compute a motor command using the primitive. A generic system is shown in Figure 3.7. Motor primitives can be represented by nonlinear dynamic systems and as a result we can map primitive tasks to policies of the type

$$\dot{\boldsymbol{x}}^d = \pi_i(\boldsymbol{x}^d, \boldsymbol{x}, t, p_i) \tag{3.1}$$

where $\boldsymbol{x}^d$ is the internal state of the system, t denotes the time, $i \in 1, 2, ..., n$ is the index of the motor primitive in a library of movements, and task parameters p such as the duration or the goal or some parameters θ_i determine the shape of the primitive. [24] Other approaches to movement primitives are dynamic movement primitives (DMP's) and probabilistic movement primitives (ProMP's).

3.4 Current Algorithms

We now explore algorithms and learning architectures that have already been done for wake sleep algorithms, aswell as covering algorithms that are inspired by manipulating learning in a similar fashion as

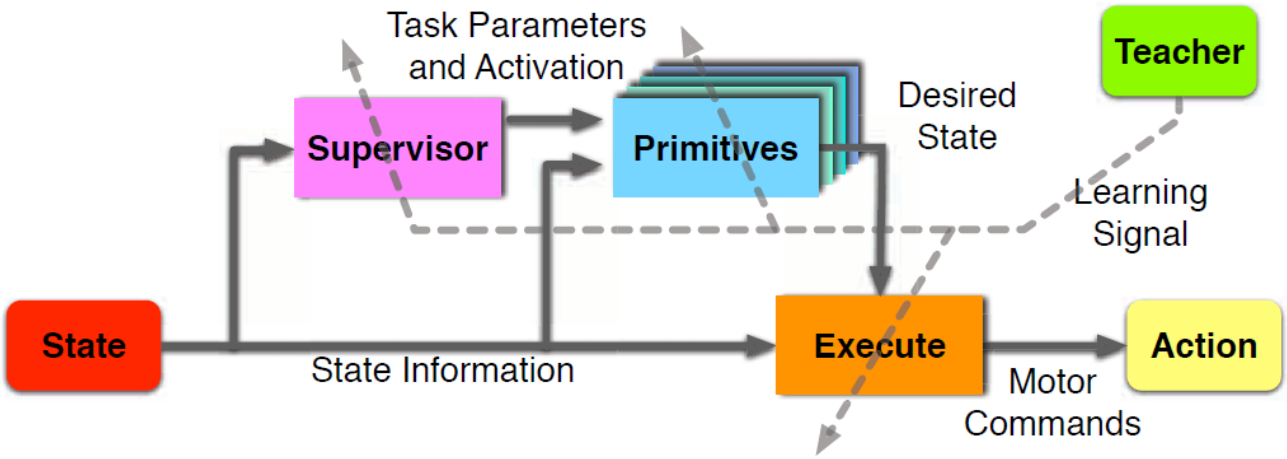


Figure 3.7: This figure shows a system for learning of motor skills. The supervisor chooses fitting motor primitives for the current state and the primitives compute the desired state, which then be executed. A teacher can propagate a learning signal during training to imrpove performance [24].

the brain obtains memory or help understanding the later proposed framework. We will mostly explore algorithms that deal with the fact that we do not have all data available at the same time and can not store infinite data.

3.4.1 Memory Building

A big factor in building memory for humans is the fact that we do not learn a new skill in one session but rather build a skill by training over several days. In the end we manage to integrate new memory into old memory with little interference of previous tasks. An important restriction here is that we do not have the data available of previous sessions but inly the learned model and somehow have to constraint changes made to the model so interference of the new learned data with the current model is as small as possible. For this problem previous work has been done for deep neural networks. Because of the overparametrization of the network for a single problem one should be able to learn multiple tasks if he is able to avoid changing parameters that are essential to the previous learned tasks. This can be achieved with a learning technique called elastic weight consolidation (EWC). EWC assumes that the log likelihood of the parameters to represent some data for task A is captured as $\log p(\theta|D)$. If we are now to learn a new task B we want to maximize the likelihood of our parameters over dataset B while not losing information on task A. For that they minimize the cost function L as follows:

$$L(\theta) = L(\theta_B) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{3.2}$$

where $L(\theta_B)$ is the loss for task B only, λ is a hyperparamtere how important the old task parameters are and F is the Fisher information matrix for the parameters of the previous task. As can be seen in Figure 3.8 this method works quite well on multiple tasks. [25]

3.4.2 Online Learning

In the following we discuss methods to handle online learning in a mixture of experts setting. Online learning plays an important role in being able to adapt fast to changes in the environment. Another key

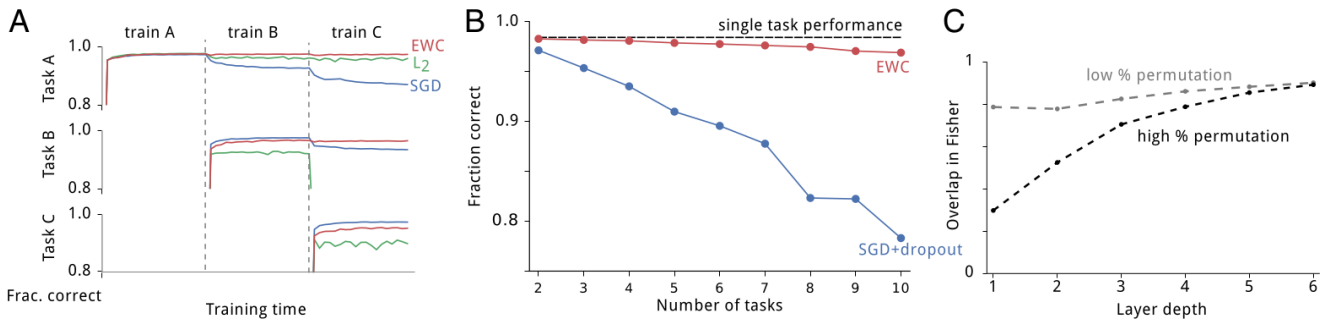


Figure 3.8: This figure shows results on the permuted MNIST task. (A) shows training curves for three random permutations on three tasks, where EWC (red) performs best. (B) shows the average performance of EWC (red) learning multiple tasks in comparison to SGD + dropout (blue). (C) shows that more similar tasks ,indicated by a low % permutation, result in a higher overlap in the Fisher information matrices in early layers [25].

consideration is the fact that we have to handle a continuous stream of input data. This is a common demand on a robot task and because we get so much data we can not store all of it not only because of hardware limitations to physically store the data but to be able to do efficient computation. Under these conditions it is in our interest to have a non-memory online learning algorithm where every new datapoint is incrementally incorporated in the learning system. In an ideal world we want to use an algorithm which uses the data structure provided by the mixture of experts model as previously established. Also by being able to grow the network of experts on the fly we can handle unseen taskspaces much better in a realtime setting and are able to adjust for slight errors which might occur. Our learning problem is given by

$$y = \beta^T \tilde{x} + \epsilon \tag{3.3}$$

3.4.3 Receptive Field Weighted Regression

Receptive Field Weighted Regression (RFWR) is a robust incremental learning approach as proposed by Schaal and Atkeson [2]. The basic structure of RFWR can be interpreted as a mixture of experts [26] where every expert is trained independent and locally from all others on the same data, while the gating can achieve accurate predictions over the input space by combining the trained experts. To give a complete overview we explain the complete model of RFWR and explain the similarities to the proposed mixture of experts system. The experts are given by:

$$y = x^T b_k + b_{0,k} = \tilde{x}^T \beta_k, \tilde{x} = (x^T, 1)^T \tag{3.4}$$

Where y is the target data, β denotes the parameters of the expert, $\tilde{x}$ is the input transformed by a feature function. The gating itself varies a bit from the mixture of experts. The RFWR structure does not incorporate a prior to compute the final output but only computes some weight for each expert multiplied by the prediction of the expert itself. The mixed prediction is then divided by the sum of the weights to get a normalized result and the final output equation thus looks like this:

$$\tilde{y} = \frac{\sum_{k=1}^{K} w_k \tilde{y}_k}{\sum_{k=1}^{K} w_k} \tag{3.5}$$

In the mixture of experts these weights correspond to the belief that an expert is responsible for a certain input and similarily for RFWR the weights w_k are the activation strength of the k'th expert. $\tilde{\boldsymbol{y}}_k$ denotes the prediction of the k'th expert accordingly. The weights are computed as follows:

$$w_k = \exp(-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{D}_k x), \text{ where } \boldsymbol{D}_k = \boldsymbol{M}_k^T \boldsymbol{M}_k \tag{3.6}$$

Using a gaussian kernel we have a similar structure to our mixture of experts with gaussian gating. Following this we can see that in fact the Distance matrix $\boldsymbol{D}$ equals the variance $\boldsymbol{\Sigma}$ in Equation 2.1. The matrix $\boldsymbol{D}$ is calculated from an upper triangular matrix $\boldsymbol{M}$ to ensure that $\boldsymbol{D}$ is positive definite and thus ensures algorithmic stability against numerical errors. The activation weight of the kernel in RFWR differs only in the normalization term of the multivariate normal distribution from the responsibility in the gaussian mixture of experts model. It is important to note that in the original equations the input is centered for each expert around some center c_k, which we left out in above equations as they are equal to the means $\boldsymbol{\mu}$ of the gating of the mixture of experts model and will be updated according to RFWR there. This does not change the behaviour or the computations of the algorithm but is rather a shift in the underlying structure to comply with the MoE approach to make the connection between the two easier to understand. In comparison to Figure 2.1 where the mixture of experts is depicted we can view the RFWR gate seperataly as given in Figure 3.9.

Learning is done for the parameters $\boldsymbol{M}$, c, and $\boldsymbol{\beta}$ for every expert and it's corresponding part in the gating independently or in terms of RFWR for every receptive field but for consistency we will refer to them as experts always. RFWR is also able to add new experts on the fly and is thus able to adapt to the complexity of the learning problem at hand. The update rules for the experts is derived from standard weighted regression and is employeed as recursive least squares. This yields

$$\boldsymbol{\beta}^{n+1} = \boldsymbol{\beta}^n + w\boldsymbol{P}^{n+1}\tilde{\boldsymbol{x}}\boldsymbol{e}_{cv}^T, \tag{3.7}$$

$$\text{where } \boldsymbol{P}^{n+1} = \frac{1}{\lambda}(\boldsymbol{P}^n - \frac{\boldsymbol{P}^n\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T\boldsymbol{P}^n}{\frac{\lambda}{w} + \tilde{\boldsymbol{x}}^T\boldsymbol{P}^n\tilde{\boldsymbol{x}}}) \text{ and } \boldsymbol{e}_{cv} = (\boldsymbol{y} - \boldsymbol{\beta}^{n^T}\tilde{\boldsymbol{x}}) \tag{3.8}$$

$\boldsymbol{P}$ determines how much the expert may be changed and is in a statistical sense the confidence in the parameters β. The update for $\boldsymbol{P}$ also contains a forgetting factor λ which will gradually diminish the influence of previous data points as the variance of the gating becomes more accurate. The gating is optimized against this cost function, where the index k is omitted because the update is the same for every expert in the gating

$$J = \frac{1}{\sum_{i=1}^P w_i}\sum_{i=1}^P \frac{w_i \,||\boldsymbol{y}_i - \tilde{\boldsymbol{y}}_i\,||^2}{(1 - w_i\tilde{\boldsymbol{x}}_i^T\boldsymbol{P}\tilde{\boldsymbol{x}}_i)^2} + \gamma\sum_{i,j=1}^n D_{ij}^2. \tag{3.9}$$

It is important to note that by penalizing the sum of the squared coefficients of the matrix $\boldsymbol{D}$ a reduction of the variance of the function approximation can be accomplished. The resulting recursive update rules for the gating are then

$$\begin{aligned}
&\boldsymbol{M}^{n+1} = \boldsymbol{M}^n - \alpha\frac{\partial J}{\partial \boldsymbol{M}} \text{ with } \mathbf{D} = \mathbf{M}^T\,\mathbf{M}, \text{ where } \mathbf{M} \text{ is upper triangular}\\
&W^{n+1} = \lambda W^n + w\\
&E^{n+1} = \lambda E^n + we_{cv}^2\\
&\boldsymbol{H}^{n+1} = \lambda\boldsymbol{H}^n + \frac{w\tilde{\boldsymbol{x}}e_{cv}}{1-h}, \text{, where } h = w\tilde{\boldsymbol{x}}^T\boldsymbol{P}^{n+1}\tilde{\boldsymbol{x}}\\
&\boldsymbol{R}^{n+1} = \lambda\boldsymbol{R}^n + \frac{w^2e_{cv}^2\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^T}{1-h}\\
&\frac{\partial J}{\partial \boldsymbol{M}} \approx \frac{\partial w}{\partial \boldsymbol{M}}\sum_i^p \frac{\partial J_{1,i}}{\partial w} + \frac{w}{W^{n+1}}\frac{\partial J_2}{\partial \boldsymbol{M}},
\end{aligned} \tag{3.10}$$

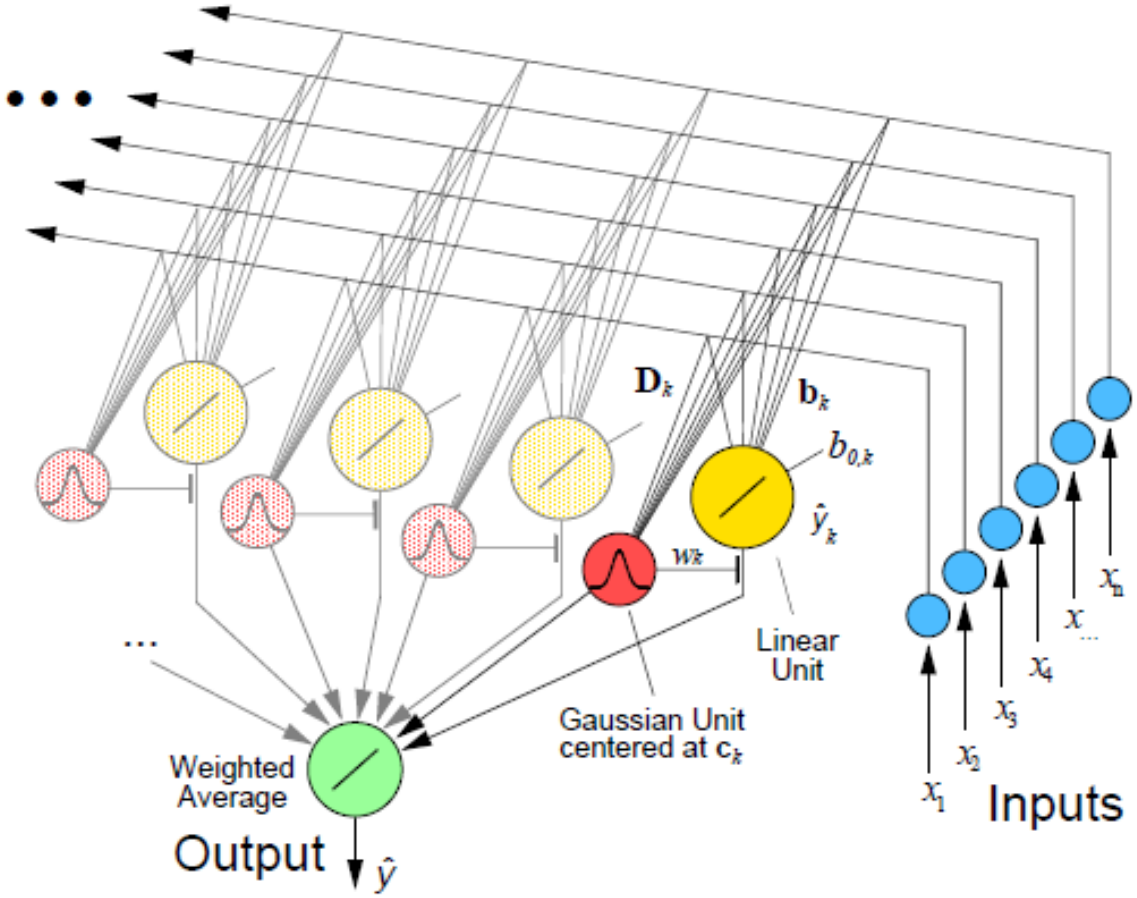


Figure 3.9: Network illustration of Receptive Field Weighted Regression for n dimensional input and one dimensional output. Every receptive field computes a prediction from a linear activation unit and a corresponding activation weight with a gaussian unit. By taking the weighted average of the predictions of the single fields we obtain the final prediction [2].

where

$$\frac{\partial w}{\partial M_{rl}} = -\frac{1}{2} w \boldsymbol{x}^T \frac{\partial \boldsymbol{D}}{\partial M_{rl}} \boldsymbol{x}, \quad \frac{\partial J_2}{\partial M_{rl}} = 2\gamma \sum_{i,j=1}^{n} \boldsymbol{D}_{ij} \frac{\partial D_{ij}}{\partial M_{rl}}$$

$$\frac{\partial D_{ij}}{\partial M_{rl}} = \delta_l j M_{ri} + \delta_i l M_{rj} \ (\delta \text{ is the Kronecker operator})$$

$$\sum_{i=1}^{p} \frac{\partial J_{1,i}}{\partial w} \approx -\frac{E^{n+1}}{(W^{n+1})} + \frac{1}{W^{n+1}} (e_{cv}^2 - (2\boldsymbol{P}^{n+1} \tilde{\boldsymbol{x}} (y - \tilde{\boldsymbol{x}}^T \boldsymbol{\beta}^{n+1})) \otimes \boldsymbol{H}^n - (2\boldsymbol{P}^{n+1} \tilde{\boldsymbol{x}} \tilde{\boldsymbol{x}}^T \boldsymbol{P}^{n+1}) \otimes \boldsymbol{R}^n),$$

where the operator $\otimes$ denotes an element-wise multiplication of two homomorphic matrices or vectors with a subsequent summation of all coefficients, $\boldsymbol{Q} \otimes \boldsymbol{V} = \sum \boldsymbol{Q}_{ij} \boldsymbol{V}_{ij}$. All recursive variables are initialized with zeros, except for the initial distance metric that is set to a manually chosen default value $\boldsymbol{D} = \boldsymbol{D}_{def}$ [2].

This learning system can then be used in the following algorithm to allocate new experts as needed in an incremental learning environment.

Algorithm 1 RFWR recursive update algorithm [2]

Initialize RFWR with no receptive field (RF)
for every new training sample $(\boldsymbol{x},\boldsymbol{y})$ **do**
  for k=1 to K **do**
    calulate the activation from Equation 3.6
    update experts and gating
  if no linear model was activated by more than w_{gen} **then**
    create a new RF with $\boldsymbol{\mu}=\boldsymbol{x}, \quad \boldsymbol{D}=\boldsymbol{D}_{def}$

4 Concepts

This chapter presents a concept that integrates insights on biological sleep in a wake sleep algorithm, which takes advantage of the offline processing capabilities of the computer and is flexible to new situations during online training. Furthermore we add time scaling to the learning process to achieve a basic learning architecture, which is capable of learning in multiple sessions without having to save the data of previous learning sessions while not affecting the performance of previous learned tasks.

4.1 Algorithm - Framework

First we define a broader structure in which we are able to change parts, to achieve the best possible results. For this we started by transferring the neuronal learning mechanisms of the brain to a machine learning perspective and work from there on. As we saw in reference to related work neuronal stuff we have multiple distinct sections of learning, during which the brain achieves different goals to obtain a general well performing structure. This results in a first high level algorithm from a neuronal perspective as can be seen in Algorithm 2.

Algorithm 2 Wake Sleep Mixture of Experts

```
train by interaction with outside world
Enter Sleepstage
while modelsize != old modelsize do
    NREM 1 and 2.
    NREM 3.
    REM
```

There are four parts that are vital to our algorithm. The first part is the interaction with the environment where we are learning incrementally through interaction and gather information about the environment. It is important that the resulting MoE architecture has the ability to adapt to changing environments and tasks while not losing any performance on previously learned data. The desired behavior would be to complement the given structure to learn more new tasks. The second main stage is the sleep phase in our algorithm. During this phase there will occur three different subphases each with a different goal in mind. The first subphase is the NREM 1 and 2 sleep and is largely responsible for a better generalization. We want to achieve better generalization capabilities by pruning experts that may have been added prematurely to the network system or simply the expert density is higher than needed in certain areas. Afterwards we continue with the second subphase which equals to the NREM 3 sleep. We can recall that during this phase in the mammal brain the brain restructures it's own synapsis and optimizes the performance on the tasks encountered during the last day. We choose the EM algorithm as our optimization technique and introduce additional extensions similar to the principle of time consolidation in the learning process of the mammal brain. The last phase in sleep is the REM phase. Since it is not completely clear from a biological view of point what the purpose is of the REM sleep phase in a learning setting we employ some validation methods on the previously learned model to ensure it's robustness.

With the new goals for each phase of our learning algorithm we can reformulate above algorithm with the abstract learning mechanisms we want to employ.

Algorithm 3 Wake Sleep Mixture of Experts

Do online learning on incoming data D_i
Begin offline training until model convergence
while model not converged **do**:
    prune experts
    optimize experts according to dataset D_i
    validate learned experts | increase robustness of model

4.1.1 Online Training

In Chapter 3 we showed the RFWR algorithm. RFWR has proven to work but ultimately an update rule for the maximization coefficients α is missing. While we could extend the derivations to include the maximization coefficient, a simpler approach is chosen with less customization done to the algorithm. That is feasible in this case since we want to present a minimal working example and the framework is designed to be able to change the learning algorithms as one sees fit. As basic algorithm we chose to implement Recursive Least Squares (RLS). RLS is a filter based on the least squares method and is used to solve linear problems or to estimate model parameters and is thus fitted to learn our system of linear experts. Because it employs an recursive update we can use it for online training. In the following we show the update rules for every parameter in our MoE system. For the gating we get the update [27]

$$\alpha_j^{t+1} = \alpha_j^t + \left(\frac{1}{L}\right)[h_j^t - \alpha_j^t] \tag{4.1}$$

$$\boldsymbol{\mu}_j^{t+1} = \boldsymbol{\mu}_j^t + \lambda_j^{t+1}[\boldsymbol{x}^t - \boldsymbol{\mu}_j^t] \tag{4.2}$$

$$\boldsymbol{\Sigma}_j^{t+1} = \boldsymbol{\Sigma}_j^t + \lambda_j^{t+1}\left[(\boldsymbol{x}^t - \boldsymbol{\mu}_j^t)(\boldsymbol{x}^t - \boldsymbol{\mu}_j^t)^T - \boldsymbol{\Sigma}_j^t\right], \tag{4.3}$$

where

$$\lambda_j^{t+1} = \frac{h^{t+1}}{\sum_{k=1}^{t+1} h^k},$$

$\boldsymbol{\Sigma}$ is the covariance of the gating, $\boldsymbol{\mu}$ is the mean of the gating and α is a maximization coefficient. The denominator D to calculate λ can be approximated by considering only the most recent values and the approximation becomes recursively

$$D^{t+1} = (1 - \frac{1}{L})D^t + h_{t+1}, \tag{4.4}$$

where L is the number of datapoints to be considered approximately.

For the linear experts the update rule is the same as the update in Equation 3.7 as RFWR already employed RLS for them as update method.

4.1.2 Pruning for Mixture of Experts

After establishing a method to update the algorithm, we want to restructure and optimize the model for the last sleep phase. To do so we use pruning to minimize the needed experts and parameters and afterwards the EM algorithm with a KL constraint to optimize the remaining experts. This resembles the NREM sleep phase as discussed previously. For pruning we use the fact that the maximization coefficient α directly correalates to the amount of hidden belief the gating has an expert on the trained dataset [28]. We can use this information to prune all experts that are not protected through time consolidation and have low α values. The benefits of pruning are that we can reduce model complexity, avoid overfitting, and reduce computation time.

4.1.3 EM with Trust Regions with Slow Wave Sleep and Time consolidation

The next step is the optimization and reallocation of our experts. We can achieve this by using expectation maximization. But the problem with the usual expectation maximization is that, if one trains a network on two different datasets previous knowledge is not considered by the EM algorithm when training on the second dataset. To counteract this problem we constraint the change one can make in the EM algorithm to experts and the gating. We show the derivations for the new update rule of mixture of experts as follows. In the M-step for learning our Gaussian MoE we try to maximize

$$\begin{aligned}
&\sum_k \sum_n h_{nk}(\log(\alpha_k) + \log(\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})) = \\
&\sum_k \sum_n (h_{nk}\log(\alpha_k) + h_{nk}\log(\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})) = \\
&\sum_k \sum_n h_{nk}\log(\alpha_k) + \sum_n h_{nk}\log(\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})).
\end{aligned} \tag{4.5}$$

Our multivariat gaussian is given as

$$\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) = \frac{1}{(2\alpha)^{n/2}\det(\boldsymbol{\Sigma})^{1/2}} \exp\{-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\}. \tag{4.6}$$

We compute the log of our multivariate gaussian to be able to maximize the log likelihood as

$$\begin{aligned}
&\log\left(\frac{1}{(2\alpha)^{n/2}\det(\boldsymbol{\Sigma_k})^{1/2}} \exp\{-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\}\right) = \\
&\log\left(\frac{1}{(2\alpha)^{n/2}\det(\boldsymbol{\Sigma_k})^{1/2}}\right) + \log\left(\exp\{-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\}\right) = \\
&\log\left(\frac{1}{(2\alpha)^{n/2}}\right) + \log\left(\det(\boldsymbol{\Sigma_k})^{-1/2}\right) - \frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = \\
&\log\left(\frac{1}{(2\alpha)^{n/2}}\right) - \frac{1}{2}\log\left(\det(\boldsymbol{\Sigma_k})\right) - \frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k).
\end{aligned} \tag{4.7}$$

Now we are able to rewrite the objective function with the derived log of the gaussian as

$$
\begin{aligned}
L_1 &= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log(\mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \left[\log\left(\frac{1}{(2\alpha)^{n/2}} \right) - \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \frac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - h_{nk} \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - h_{nk} \frac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - h_{nk} \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \frac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} h_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - \sum_n h_{nk} \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \right. \\
&\quad \left. \sum_n \frac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} h_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - \sum_n h_{nk} \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \right. \\
&\quad \left. \sum_n \frac{1}{2} h_{nk} \mathrm{tr}\left((\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k})^\mathrm{T} \boldsymbol{\Sigma}_\mathrm{k}^{-1} (\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k}) \right) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - \sum_n h_{nk} \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \right. \\
&\quad \left. \frac{1}{2} \sum_n h_{nk} \mathrm{tr}\left((\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k}) (\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k})^\mathrm{T} \boldsymbol{\Sigma}_\mathrm{k}^{-1} \right) \right] \\
&= \sum_k \left[\sum_n h_{nk} \log(\alpha_k) + \sum_n h_{nk} \log\left(\frac{1}{(2\alpha)^{n/2}} \right) - \frac{1}{2} \log\left(\det(\boldsymbol{\Sigma})\right) - \right. \\
&\quad \left. \sum_n h_{nk} - \frac{1}{2} \sum_n h_{nk} \mathrm{tr}\left((\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k}) (\mathrm{x_n} - \boldsymbol{\mu}_\mathrm{k})^\mathrm{T} \boldsymbol{\Sigma}_\mathrm{k}^{-1} \right) \right] \qquad (4.8)
\end{aligned}
$$

We permutate the centered input around the mean and the covariance matrix because it makes it easier in the following steps to differentiate the function. This is done using a property of traces, where

$$\mathrm{tr}(\boldsymbol{ABC}) = \mathrm{tr}(\boldsymbol{CAB}) = \mathrm{tr}(\boldsymbol{BCA}).$$

Next we use the fact that the trace of a scalar is the scalar itself and since our term always boils down to a scalar one may rewrite it as

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \mathrm{tr}(\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}) = \mathrm{tr}(\boldsymbol{x}\boldsymbol{x}^T \boldsymbol{A}). \qquad (4.9)$$

But to get not only the optimal result for the current dataset, but impose a cap on the change the gating and the experts can undergo to prevent forgetting of previously learned information, we introduce a constraint using the Kullback Leibler Divergence such that

$$\text{s.t.} \quad \mathrm{KL}(p_{\text{old}} || p) \leq \varepsilon \qquad (4.10)$$

$$\log(p_{\text{old}}) \log\left(\frac{p_{\text{old}}}{p} \right) \qquad (4.11)$$

We add this constraint to our lagrangian and introduce a lagrangian multiplier so we get

$$L = L_1 + \eta L_2,$$

, where

$$\begin{aligned} L_2 =& \varepsilon - \mathrm{KL}(\mathcal{N}(\mathrm{x_n}|\boldsymbol{\mu}_{\mathrm{k,old}}, \boldsymbol{\Sigma}_{\mathrm{k,old}})||\mathcal{N}(\mathrm{x_n}|\boldsymbol{\mu}_{\mathrm{k}}, \boldsymbol{\Sigma}_{\mathrm{k}})) \\ =& \varepsilon - \left(\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_{\mathrm{k}}^{-1}\boldsymbol{\Sigma}_{\mathrm{k,old}}\right) + \frac{1}{2}(\boldsymbol{\mu}_{\mathrm{k}} - \boldsymbol{\mu}_{\mathrm{k,old}})^{\mathrm{T}})\boldsymbol{\Sigma}_{\mathrm{k}}^{-1}(\boldsymbol{\mu}_{\mathrm{k}} - \boldsymbol{\mu}_{\mathrm{k,old}}) + \frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}_{\mathrm{k}})}{\det(\boldsymbol{\Sigma}_{\mathrm{k,old}})}\right) - \frac{\mathrm{D}}{2}\right). \end{aligned} \tag{4.12}$$

Our lagrangian expression then becomes

$$\begin{aligned} &L(\eta, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \\ &\sum_k \left[\sum_n h_{nk}\log(\alpha_k) + \sum_n h_{nk}\log\left(\frac{1}{(2\alpha)^{n/2}}\right) - \frac{1}{2}\log\left(\det(\boldsymbol{\Sigma})\right)\sum_n h_{nk} - \frac{1}{2}\sum_n h_{nk}\mathrm{tr}\left((\mathrm{x_n} - \boldsymbol{\mu}_{\mathrm{k}})(\mathrm{x_n} - \boldsymbol{\mu}_{\mathrm{k}})^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathrm{k}}^{-1}\right)\right] \\ &+ \eta\left(\varepsilon - \left(\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_{\mathrm{k}}^{-1}\boldsymbol{\Sigma}_{\mathrm{k,old}}\right) + \frac{1}{2}(\boldsymbol{\mu}_{\mathrm{k}} - \boldsymbol{\mu}_{\mathrm{k,old}})^{\mathrm{T}})\boldsymbol{\Sigma}_{\mathrm{k}}^{-1}(\boldsymbol{\mu}_{\mathrm{k}} - \boldsymbol{\mu}_{\mathrm{k,old}}) + \frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}_{\mathrm{k}})}{\det(\boldsymbol{\Sigma}_{\mathrm{k,old}})}\right) - \frac{\mathrm{D}}{2}\right)\right). \end{aligned} \tag{4.13}$$

Dual Functions

After obtaining our lagrangian expression we can now differentiate the lagrangian with respect to the mean and the covariance. With the derivation we can then formulate a dual function which we can optimize. Firstly, we take the derivative of L_1 with respect to $\boldsymbol{\Sigma}_{\boldsymbol{k}}^{-1}$

$$\frac{\partial L_1}{\partial \boldsymbol{\Sigma}_k^{-1}} = \frac{1}{2}\boldsymbol{\Sigma}_k\sum_n h_{nk} - \frac{1}{2}\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T, \tag{4.14}$$

and we do not update the mean function yet i.e. $\mu_k = \mu_{k,\mathrm{old}}$ so that L2 changes to

$$L2 = \varepsilon - \left(\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\Sigma}_{k,\mathrm{old}}\right) + \frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}_{\boldsymbol{k}})}{\det(\boldsymbol{\Sigma}_{k,\mathrm{old}})}\right) - \frac{D}{2}\right). \tag{4.15}$$

We take the derivative of the modified L_2 w.r.t Σ_k^{-1} and obtain

$$\frac{\partial L_2}{\partial \boldsymbol{\Sigma}_{\boldsymbol{k}}^{-1}} = -\frac{1}{2}\boldsymbol{\Sigma}_{k,\mathrm{old}} + \frac{1}{2}\boldsymbol{\Sigma}_{\boldsymbol{k}}. \tag{4.16}$$

Thus our full derivate of the lagrangian is

$$\begin{aligned} \frac{\partial L}{\partial \Sigma^{-1}} =& \frac{\partial L_1}{\partial \Sigma^{-1}} + \eta\frac{\partial L_2}{\partial \Sigma^{-1}} \\ =& \frac{1}{2}\Sigma_k\sum_n h_{nk} - \frac{1}{2}\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \\ & \eta(-\frac{1}{2}\boldsymbol{\Sigma}_{k,\mathrm{old}} + \frac{1}{2}\boldsymbol{\Sigma}_{\boldsymbol{k}}). \end{aligned} \tag{4.17}$$

To obtain a minimum we set this to zero and solve for $\mathbf{\Sigma}_k$

$$
\begin{aligned}
0 =& \frac{1}{2}\mathbf{\Sigma}_k \sum_n h_{nk} - \frac{1}{2}\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \\
& \eta(-\frac{1}{2}\mathbf{\Sigma}_{k,\text{old}} + \frac{1}{2}\mathbf{\Sigma}_k) \\
& \frac{1}{2}\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \eta\frac{1}{2}\mathbf{\Sigma}_{k,\text{old}} = \frac{1}{2}\mathbf{\Sigma}_k \sum_n h_{nk} + \eta\frac{1}{2}\mathbf{\Sigma}_k \\
& \frac{\frac{1}{2}\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \eta\frac{1}{2}\mathbf{\Sigma}_{k,\text{old}}}{\frac{1}{2}\sum_n h_{nk} + \eta\frac{1}{2}} = \mathbf{\Sigma}_k \\
& \frac{\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T + \eta\mathbf{\Sigma}_{k,\text{old}}}{\sum_n h_{nk} + \eta} = \mathbf{\Sigma}_k
\end{aligned}
\tag{4.18}
$$

This can be rewritten as

$$
\frac{\sum_n h_{nk}}{\sum_n h_{nk} + \eta}\frac{\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T}{\sum_n h_{nk}} + \frac{\eta}{\sum_n h_{nk} + \eta}\mathbf{\Sigma}_{k,\text{old}} = \mathbf{\Sigma}_k, \tag{4.19}
$$

and we obtain the final form of the derivation as

$$
(1-\alpha)\frac{\sum_n h_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T}{\sum_n h_{nk}} + \alpha\mathbf{\Sigma}_{k,\text{old}} = \mathbf{\Sigma}_k^*. \tag{4.20}
$$

with α being

$$
\alpha = \frac{\eta_\Sigma}{(\sum_n h_{nk}) + \eta_\Sigma} \tag{4.21}
$$

If we decouple the covariance from the mean our lagrangian becomes

$$
\begin{aligned}
& L(\eta_\Sigma, \Sigma_k) = \\
& -\frac{1}{2}\log(\det(\mathbf{\Sigma}))\sum_n h_{nk} - \frac{1}{2}\sum_n h_{nk}\text{tr}\left((\text{x}_\text{n} - \boldsymbol{\mu}_\text{k})(\text{x}_\text{n} - \boldsymbol{\mu}_\text{k})^\text{T}\mathbf{\Sigma}_\text{k}^{-1}\right) \\
& + \eta_\Sigma\varepsilon - \eta_\Sigma\left(\left(\frac{1}{2}\text{tr}\left(\mathbf{\Sigma}_\text{k}^{-1}\mathbf{\Sigma}_{\text{k,old}}\right) + \frac{1}{2}\log\left(\frac{\det(\mathbf{\Sigma}_\text{k})}{\det(\mathbf{\Sigma}_{\text{k,old}})}\right) - \frac{\text{D}}{2}\right)\right).
\end{aligned}
\tag{4.22}
$$

And we can finally obtain the dual by inserting the solution for the optimal covariance. We get the optimal covariance by calculating η as we solve the optimization problem given by our dual function.

$$
D(\eta_\Sigma) = L(\eta_\Sigma, \Sigma^*) \tag{4.23}
$$

After updating the covariance we need to update the mean. The Lagrangian for the mean matrix disregarding the covariance is given by

$$
L_\mu(\eta, \mathbf{\Sigma}) = \left(\sum_n -\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T\mathbf{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)\right) + \eta\varepsilon_\mu - \eta\frac{1}{2}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_{\text{old}})^T\mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_{\text{old}}) \tag{4.24}
$$

The full derivative for the mean is then

$$\frac{\partial L_\mu}{\partial \boldsymbol{\mu}_k} = \left(\sum_n h_{nk} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right) - \eta \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k,\text{old}}). \tag{4.25}$$

Set the derivative to zero and solve for $\boldsymbol{\mu}_k$ to obtain a minimum

$$\begin{aligned}
0 &= \sum_n h_{nk} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) - \eta \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k,\text{old}}) \\
0 &= \sum_n h_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) - \eta (\boldsymbol{\mu}_k - \boldsymbol{\mu}_{k,\text{old}}) \\
-\eta \boldsymbol{\mu}_{k,\text{old}} - \sum_n h_{nk} \boldsymbol{x}_n &= -\eta \boldsymbol{\mu}_k - \sum_n h_{nk} \boldsymbol{\mu}_k \\
-\eta \boldsymbol{\mu}_{k,\text{old}} - \sum_n h_{nk} \boldsymbol{x}_n &= \boldsymbol{\mu}_k (-\eta - \sum_n h_{nk}) \\
\frac{-\eta}{-\eta - \sum_n h_{nk}} \boldsymbol{\mu}_{k,\text{old}} - \frac{\sum_n h_{nk} \boldsymbol{x}_n}{-\eta - \sum_n h_{nk}} &= \boldsymbol{\mu}_k \\
\frac{-\eta}{-\eta - \sum_n h_{nk}} \boldsymbol{\mu}_{k,\text{old}} - \frac{\sum_n h_{nk}}{-\eta - \sum_n h_{nk}} \frac{\sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk}} &= \boldsymbol{\mu}_k \\
\frac{\eta}{\eta + \sum_n h_{nk}} \boldsymbol{\mu}_{k,\text{old}} + \frac{\sum_n h_{nk}}{\eta + \sum_n h_{nk}} \frac{\sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk}} &= \boldsymbol{\mu}_k
\end{aligned} \tag{4.26}$$

This can also be rewritten as

$$(1-\alpha) \frac{\sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk}} + \alpha \boldsymbol{\mu}_{k,\text{old}} = \boldsymbol{\mu}_k^*. \tag{4.27}$$

with α being

$$\alpha = \frac{\eta}{(\sum_n h_{nk}) + \eta}. \tag{4.28}$$

We can now write our Dual function for the mean and optimize it to get the optimal η

$$\begin{aligned}
D(\eta) =& L(\eta, \boldsymbol{\mu}_k^*) = \left(\sum_n -h_{n,k} \frac{1}{2} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \Sigma^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right) + \eta \varepsilon_\mu - \eta \frac{1}{2} (\mu_k - \boldsymbol{\mu}_{k,\text{old}})^T \Sigma^{-1} (\mu_k - \boldsymbol{\mu}_{k,\text{old}}) \\
=& \left(\sum_n -h_{n,k} \frac{1}{2} (\boldsymbol{x}_n - \frac{\eta \boldsymbol{\mu}_{k,\text{old}} + \sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk} + \eta})^T \Sigma^{-1} (\boldsymbol{x}_n - \frac{\eta \boldsymbol{\mu}_{k,\text{old}} + \sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk} + \eta}) \right) + \\
& \eta \varepsilon_\mu - \eta \frac{1}{2} (\frac{\eta \boldsymbol{\mu}_{k,\text{old}} + \sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk} + \eta} - \boldsymbol{\mu}_{k,\text{old}})^T \Sigma^{-1} (\frac{\eta \boldsymbol{\mu}_{k,\text{old}} + \sum_n h_{nk} \boldsymbol{x}_n}{\sum_n h_{nk} + \eta} - \boldsymbol{\mu}_{k,\text{old}}) \\
=& \left(\sum_n -h_{n,k} \frac{1}{2} (\boldsymbol{x}_n - \frac{\eta \boldsymbol{\mu}_{k,\text{old}} + c_1}{c_2 + \eta})^T \Sigma^{-1} (\boldsymbol{x}_n - \frac{\eta \boldsymbol{\mu}_{k,\text{old}} + c_2 \boldsymbol{x}_n}{c_2 + \eta}) \right) + \\
& \eta \varepsilon_\mu - \eta \frac{1}{2} (\frac{\eta \boldsymbol{\mu}_{k,\text{old}} + c_1}{c_2 + \eta} - \boldsymbol{\mu}_{k,\text{old}})^T \Sigma^{-1} (\frac{\eta \boldsymbol{\mu}_{k,\text{old}} + c_1}{c_2 + \eta} - \boldsymbol{\mu}_{k,\text{old}})
\end{aligned} \tag{4.29}$$

with

$$c_1 = \sum_n h_{nk} \boldsymbol{x}_n \qquad c_2 = \sum_n h_{nk}.$$

As this describes the update rules for the gating parameters, we can reformulate the update for the linear experts aswell. Since given our assumption each expert builds a multivariate gaussian distribution aswell, the update rules are similar with the difference that $\boldsymbol{x}_n$ is substituted with $\boldsymbol{y}_n$ and the mean $\boldsymbol{\mu}_n k$ is given as $\boldsymbol{x}_n \boldsymbol{\beta}$. This yields as new optimal mean for an expert

$$\boldsymbol{\beta}^* = (1-\alpha)(\boldsymbol{X}^T \boldsymbol{h} \boldsymbol{X})^{-1}(\boldsymbol{X}^T \boldsymbol{h} \boldsymbol{Y}) + \alpha \boldsymbol{\beta}_{\text{old}}, \tag{4.30}$$

with α being

$$\alpha = (\boldsymbol{X}^T \boldsymbol{h} \boldsymbol{X} + \frac{\eta}{N} \boldsymbol{X}^T \boldsymbol{X})^{-1}(\frac{\eta}{N} \boldsymbol{X}^T \boldsymbol{X}). \tag{4.31}$$

Slow Wave Activity Bounds

After obtaining new update rules for the EM algorithm we need to set the bound ε to which we constraint the change of the gating and the experts. An important property which we want to cover accordingly to the biologic model is that the amount of slow wave activity in a brain area is higher in areas which were under use during training sessions during the last day. This means that we want a looser bound on experts, which were used alot during the last training session and a bound close to zero or set to zero as the expert was not or close to not used. To recreate this behavior we decided to add the activation values of each expert for each data point together and then normalize the values to be between zero and one. The activation values are the belief of the gating that an expert is responsible for a datapoint. This results in

$$\zeta_k = \sum_i h_i^{(k)}(\boldsymbol{y}_i | \boldsymbol{x}_i), \tag{4.32}$$

where $h_i^{(k)}$ is computed as in Equation 2.12 and ζ_k is the slow wave activity of the k-th expert. To obtain the final bound we normalize such that $\sum_k \zeta_k = 1$. Our bound ε for the k-th expert is then ζ_k.

Time Consolidation

The last modification we want to modify our bound ε is by decreasing the bound as an expert grows older. This should have a consolidation effect on parts of the model such that even on a long term basis with a lot of training sessions previous knowledge is not lost. We modify our previous calculated bound by dividing ε through the squared age and get finally

$$\varepsilon_\zeta = \frac{\varepsilon}{t_k^2},$$

where t_k denotes the age of the k-th expert and is set to the number of day night cycles an expert exists.

5 Code Framework

In this chapter we will describe the framework programmed for this work. We start by showing the interfaces it does provide and will then specify which parts a user has to provide to be able to build his own solution using the framework. At last we present some basic functionality provided by the framework and give a starting point for the ease of use. We use Python 3.5 as programming language because python allows for a fast development speed and is generally easy to read even for newer users. We use the modules numpy, scipy and matplotlib as they are scientific standard use for python and provide very easy accessible functions, which use C-libraries for faster computations. For further utility for saving and loading data structures pickle is used.

5.1 Interface

To be able to use the main functionality one has to import the wakesleep.py file and create a WakeSleepLearer object. To create an object there are numerous options which will be listed with a short description here.

- gate: Mandatory argument. Provide the gating class which holds the experts and does the gating calculations.
- online_learning_function: Mandatory argument. The function definition to use for on-line learning.
- pruning_function: Mandatory argument. The function definition to use for pruning.
- optimization_function: Mandatory argument. The function definition to use for optimization.
- validation_function: Mandatory argument. The function definition to use for validating.
- test_input_data: Mandatory argument. Data to test the model against. Must have the same number of data points as test_target_data.
- test_target_data: Mandatory argument. Data to test the model against. Must have the same number of data points as test_input_data.
- data_noise: Mandatory argument. Noise underlying the input data.
- base_path: Optional argument, defaults to an empty string. Stores the model at the specified location.
- pruning_activated: Optional argument, defaults to True. Sets pruning activated if True and off if False.
- optimization_activated: Optional argument, defaults to True. Sets optimization activated if True and off if False.
- validation_activated: Optional argument, defaults to True. Sets validation activated if True and off if False.
- mode: Optional argument, defaults to Mode.WAKE. Sets the algorithm either in wake or sleep state.

- optimization_options: Optional argument, defaults to None. Additional parameters passed to the optimization function.
- online_learning_options: Optional argument, defaults to None. Additional parameters passed to the online_learning function.
- pruning_options: Optional argument, defaults to None. Additional parameters passed to the pruning function.
- validation_options: Optional argument, defaults to None. Additional parameters passed to the validation function.
- archiving: Optional argument, defaults to False. If True archives all input and target data provided during online learning.

Note that if you do not want to provide functions like i.e. the pruning function you can simply pass None and set the use of pruning to False.

5.2 Classes and Methods

In this section we will provide info which functions the gating class must provide and the arguments and return values for different mandatory functions. The gating class needs to provide methods presented in Table 5.1.

Table 5.1: Gating

Methodname	Input parameters	Return values	Expected behavior
compute_output	Numpy array with the same dimensions as one input data point	Numpy array with the same as one target data point	Computes the prediction of the current gating model
age	No input	No output	increases the age of all experts internally by one
clear_swa	No input	No output	removes the accumulated slow wave activity of the gating

To complete the frameworks functionality we need to provide some core functions that implement the previously established concepts in Chapter 4. The function definitions can be seen in Table 5.2.

Table 5.2: Functions provided to the framework

Method	Input parameters	Return values	Expected behavior
online learning	gating object, input signal, teaching signal	prediction	compute prediction using current gating model and incremental update of gating
pruning	gating object, slow wave activity array	None	prune all experts deemed unnecessary by some inner heuristic
optimization	gating object, input data, target data, test input data, test target data	log likelihood array, RMSE array	restructuring of inner model and optimization of model with latest data
validation	gating object	Number	Measurement of robustness of the gating, where a higher value is better

6 Evaluation

In this chapter we evaluate the previously introduced framework using a MoE architecture on different Datasets. First, we look at the performance of online learning and expectation maximization in isolation for different datasets and compare those with each other. Afterwards we present results for online learning with pruning and compare the different model sizes with respect to the performance. For online learning we try different thresholds for splitting of experts and explore different starting values for various parameters such as the variance. We try different modelsizes for mixture of experts and set a maximum of 40 iterations for each run and compare the usual EM-algorithm to our EM-algorithm with trust regions as described in Chapter 4. At last we evaluate all components together in the framework and make use of the accumulated slow wave activity during online training and use time consolidation. We present results for different thresholds to add and prune experts and try different bounds in our optimization phase. We evaluate the frameworks performance by splitting up the used datasets to simulate multiple day night cycles and show different areas of the task space during different training sessions and compare it to normal online learning. We omit evaluating mixture of experts directly on a multi day learning setup since the algorithm is unable to learn distinct task spaces over multiple sessions.
As datasets we use three different functions for comparison in lower dimensional space. At the beginning we sample the sine function $y = \sin(x)$ in the input area $(0, 14)$ 560 times and use as validation set a sample of 200 values in the same input area. Our second evaluation is on the NIST Nelson dataset where we use 128 data points for training and the same amount for testing.

6.1 Learning a Sine Curve

In our first experiment we let each model learn a sine curve with different configurations. We apply some normal distributed random noise on the dataset where the mean is the actual value of the sine and the variance is 0.1.

6.1.1 Online learning

Our first evaluation is for online learning and the results can be seen in Table 6.1 for 1 training epoch and in Table 6.2 for 10 training epochs. We use three different thresholds to achieve different model complexities and measure the log likelihhod and the root mean squared error (RMSE) on a test set. The foregetting factor λ was set to 0.95 as is usually done for the RMS filter and L to 25. We set the default value of P to 200 for the experts as we have little trust in the expert parameters at the start and set the variance for the experts to 0.1 as we currently do not employ updates for the expert variance.

Table 6.1: Results of online learning on a sine curve after 1 training epoch

Splting Threshold	Modelsize	avg. log likelihood	Best log likelihood	avg RMSE	Best RMSE
1e-50	27	72.430185	72.77131	0.078455	0.076897
1e-60	25	70.622091	71.19415	0.086291	0.083903
1e-100	20	64.657671	65.557115	0.108188	0.105196

For the gating we use a default variance of 0.001 as we want an expert to be responsible only for a small amount of data at the beginning and then expand further through the gating algorithm which gives more responisbility to single experts. We initialized with 1 expert set around the first input and can see

Table 6.2: Results of online learning on a sine curve after 10 training epoch

Splting Threshold	Modelsize	avg. log likelihood	Best log likelihood	avg RMSE	Best RMSE
1e-50	29	80.513387	80.640829	0.019527	0.017092
1e-60	27	80.398130	80.538530	0.021515	0.019111
1e-100	21	79.574718	79.724714	0.032443	0.030765

that for the highest threshold we get the smallest model but also remain below the performance of the more complex models.

We see that for more epochs of training the difference in performance gets smaller and the final difference can be seen in Figure 6.1. We can address the large difference in performance after the first training epoch to the fact that a single expert has to cover more data points and is likely to fit better to the last trained data points. Another interesting feature is that with more training epochs the uncertainty in the average performance of online learning gets close to zero and it indicates robust learning which is not dependable on the initialization. Over more training epochs a more ideal solution for each expert is found and every tested model approximates a similar solution. Given that low of a performance difference it is safe to assume that some overfitting is happening and further tests for the optimal threshold would have to be made.

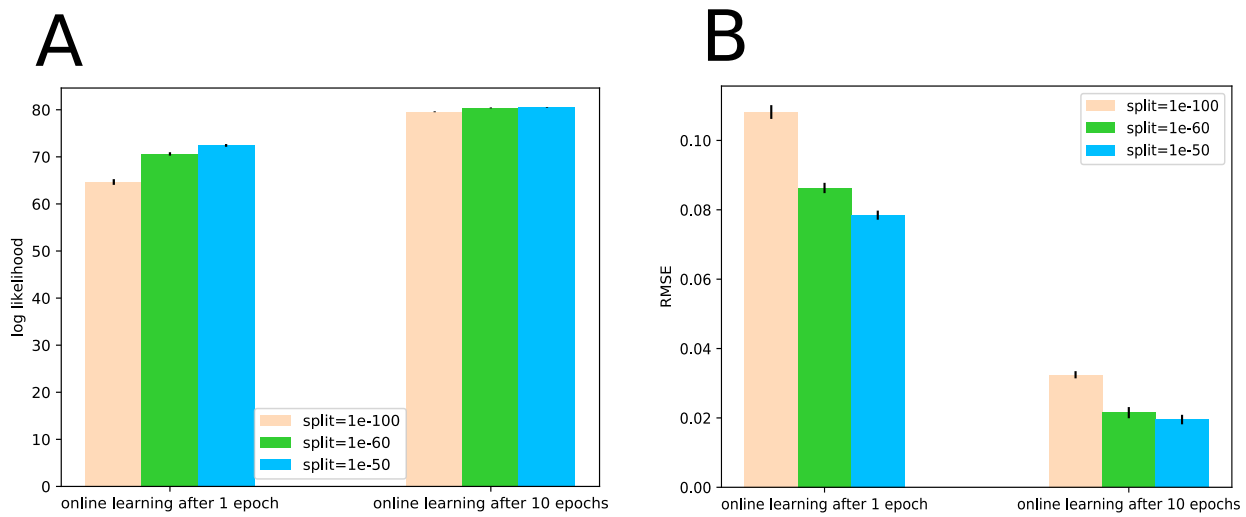


Figure 6.1: Average performance results over ten trials on learning a sine curve using online learning.(A) On the left side are the log likelihoods after performing online learning on the dataset for one epoch and on the right side are log likelihoods after ten training epochs. The black lines indicate the recorded standard deviation observed during the different trials. The colors group the thresholds used when to add a new expert. We can observe an increase in log likelihood with more training, as well as a decrease in the standard deviation. (B) Plot of the RMSE using the same thresholds as in (A). The RMSE generally decreases with more training episodes but the standard deviation indicated by the lack lines does not change much.

6.1.2 Expectation Maximization

For the expectation maximization we use a maximum of 40 iterations. We tested the EM algorithm with 20, 25 and 30 experts, where we distributed the experts evenly over the input space for the initialization of the gating mean. The parameters of the experts were chosen randomly with exception of the variance,

which we set to 0.1 as we do not update the variance of the experts. We can see that EM converges on the dataset in Figure 6.2 over the course of 40 iterations. In Table 6.3 is the performance of the EM algorithm. We can see that the average performance is largely below the performance of online learning, but the best result for each model size is actually a little it better than the best result of online learning with 10 training epochs. In general we get a large variance in the log likelihood and the RMSE as can be seen in Figure 6.3, because the EM algorithm relies on a good initialization to not get stuck in bad local optima as it only converges to local optima. At last we can see that the results for the model sizes do vary only by a small amount and overfitting is likely to occur with more than 20 models.

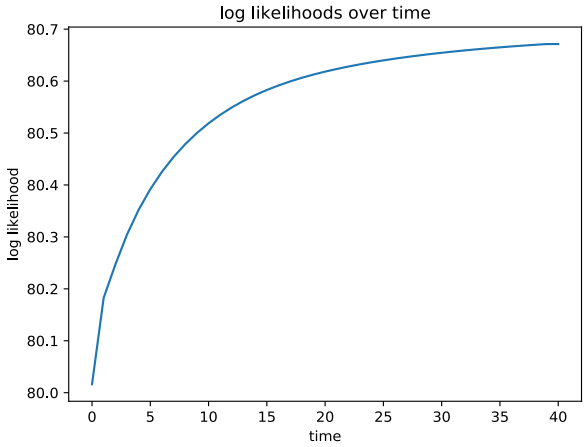


Figure 6.2: Evolution of the log likelihood over 40 iterations using the EM algorithm. The model used 20 experts.

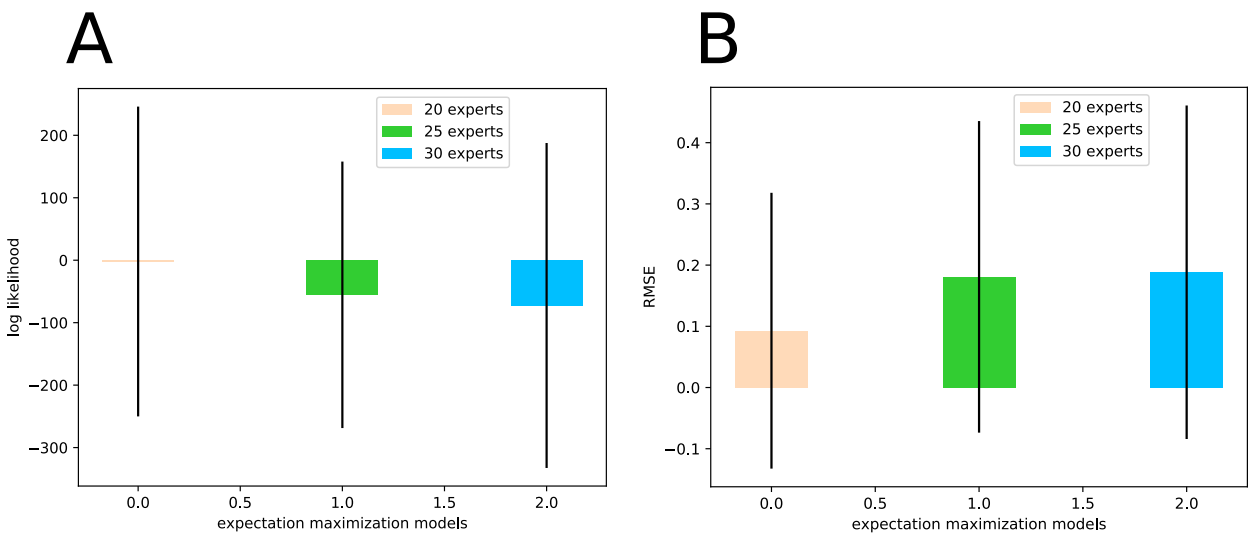


Figure 6.3: Average performance results over ten trials on learning a sine curve using expectation maximization. (A) Average log likelihoods for different numbers of experts. For every amount of expert there is a huge standard deviation shown by the lack lines. The best average result is obtained from the smallest model on the leftmost. (B) Average RMSE for different numbers of experts. The average RMSE is the best for the smallest model size on the left but the standard deviation is in general very high for every model size.

Table 6.3: Results of best EM and average EM learning after 40 iterations

Modelsize	Best log likelihood	best RMSE	avg. log likelihood	avg. RMSE
20	80.696516	0.015886	-2.068305	248.015046
25	80.750626	0.014619	-55.402912	213.331131
30	80.829316	0.012551	-72.503998	260.149786

Table 6.4: Results of the complete framework on the sine dataset

Training Days	Modelsize	Optimization with bounds	avg. log likelihood	avg. RMSE	Best log likelihood	Best RMSE
1	25	off	80.852443	0.011801	80.903688	0.010218
1	25	on	78.840527	0.039704	79.040146	0.037888
4	24.4	on	65.272665	0.080268	76.928935	0.054254
10	20.7	on	70.950707	0.084649	74.606028	0.067843

6.1.3 Wake sleep framework

For our framework we will look at the parameter configuration for every single phase first and then at the parameter configuration for the experts and the gating. We test the effect of splitting the data set to see how a day night cycle of learning with the proposed methods affects the rate of learning nad if we get comparable results on our dataset. Further we want to evaluate the proposed optimization with bounds and the effects it has if we have multiple day and night cycles. For our setup of the online learning we used the same parameter configuration as for online learning only and set the splitting threshold to 1e-60. Experts are pruned if their value calculated as $\frac{\zeta}{\iota_k^2}$ is below 1e-6. For optimization we set the maximum number of iterations to 40 and only use the data gathered in the last online learning phase. Validation is done by comparing the last log likelihood of the model before the sleep iteration and comparing it with the log likelihood of the new model after the updates and rejecting the update if the new log likelihood is below 95% of the old log likelihood. We can see the final results in Table 6.4.

The results for one day night cycle are the best and if we turn off the optimization with bounds we get the best results for one day night cycle. We can explain the difference in performance for one day night cycle with the small number of maximum iterations and the model cannot converge in the given time and is thus at the end at a worse result. If we have four day night cycles we can observe that the best result is better than the best result for 10 day and night cycles. However the trained model seems to be more volatile to randomness as can be seen in Figure 6.4. Further although we have the worst performance with ten day night cycles we also get the smallest modelsize. In comparison with pure online learning the framework performs better if the data is only seen once and can not be saved over the course of two days. However if we are able to save the data and reiterate multiple times over it with our online learning algorithm we are able to achieve better results.

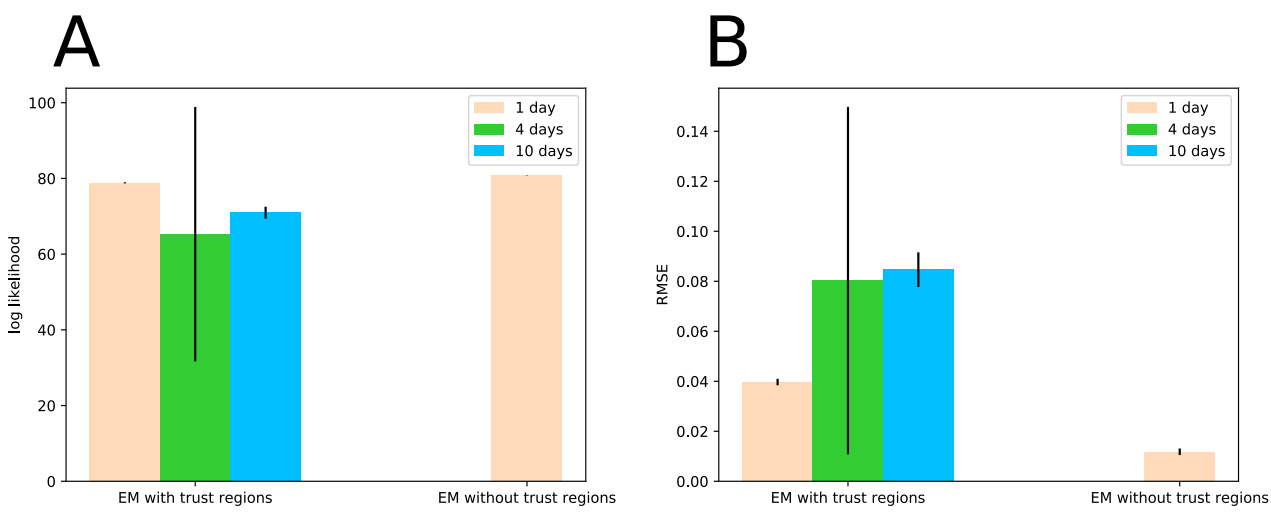


Figure 6.4: Average performance results over ten trials on learning a sine curve using the proposed learning framework. (A) On the left side expectation maximization is used with trust regions and tested training by splitting the dataset over multiple days. The best result is obtained when all data is observed when all data is learned in one day. For four days the standard deviation as the black lines was pretty high and thus the average performance remained below the log likelihood of training over ten days. The overall best result is obtained by using EM without trust regions on the right side for one day. The results for EM without trust regions for more days are omitted in this plot as they did not yield usable models and are in the end not designed to work this task. (B) Results for RMSE in the same order as in (A). Generally the results regarding performance are the same but training over four days had a slightly better RMSE on average than training over ten days. Again the lowest RMSE was obtained from training with expectation maximization without trust regions and the results for more training days are omitted for the same reasons as in (A).

6.2 Learning a Regression Dataset

As the second Dataset to evaluate our framework on we used the Nelson NIST dataset. The dataset provides two dimensional input and 1 dimensional output. We tested online learning in isolation, expectation maximization and our proposed framework with one setup each. For online learning we used a splitting threshold of 1e-45. The default variance of experts was tweaked to be one, the parameters are random in the range $[-1, 1]$ and the starting P values were initialized with 200. The gating uses a default variance of ten. To configure the online learning parameters we set L again to 25 and the forgetting factor λ to 0.95 as for the sine learning problem. We do online learning only for 1 epoch for better comparisons with the wake sleep framework in this case. As for expectation maximization we initialized ten experts evenly distributed over the input space and set the default values for the experts the same as for online learning. The variance of the gating was determined as the variance of the input data divided by the number of inputs. Lastly, we set the parameters of our framework to be the same for online learning and optimization. The pruning threshold was set again to 1e-6. We split the dataset in two equally large sets to perform training over two days and be able to compare the results with pure online learning. We use ten trials with different random initializations as discussion basis.

6.2.1 Results on the Nelson dataset

The results can be seen in Table 6.5. We can see that the EM algorithm delivers the best results by far and also has a very low standard deviation in comparison to the other methods. More interesting is the comparison between online learning and the wake sleep framework as they both do not require to save any data. Online learning reaches a better result as the wake sleep framework for its best model but is highly volatile in performance, which would be an especially bad in case a robot has to act autonomously for a longer period of time. In that case the wake sleep framework provides only slightly worse results at peak performance while also being consistent. It seems that splitting data and performing optimization on each split in combination with online learning lowers the risk of bad results due to randomness significantly.

Table 6.5: Results on the NIST Nelson dataset with 10 trials for each method

Training Method	Best log likelihood	avg. log likelihood	std. log likelihood	Best RMSE	avg. RMSE	std. RMSE
Online Learning	-4193.353799	-50608288.740323	99988086.490593	1.825910	96.038171	174.110462
Expectation Maximization	-2347.818751	-3290.682167	314.967808	1.375545	1.619293	0.0813980
Wake Sleep Framework	-4616.450426	-10525.2426784	2556.538985	1.914286	2.850482	0.394318

7 Conclusion & Outlook

This thesis introduced a concept for a modular learning algorithm framework based on insights from neurobiological sleep. Therefore, we investigated and summarized the effects of sleep on learning for humans and focused in particular on neuronal mechanisms. On the robotics side we looked into the foundations of motor skill learning and presented expectation maximization as optimization method and receptive field weighted regression for online learning. As model a mixture of experts architecture was chosen because it offers an already explored abstraction to the structure of the brain and is statistically sound. Experimental evaluations of the porposed concept showed increased performance for training in multiple sessions where we can not save training data in between sessions. Our online algorithm in isolation achieved sufficient results in approximating a desired function without having to save data.
The experiments revealed that expectation maximization in isolation achieved only good results on a complete dataset. By connecting online learning with optimization through the biological concept of learning through sleep we achieved better results as online learning alone, however once we can learn the complete data offline normal online learning or expectation maximization prove at least equally powerful. But in our scenario we are able to not only achieve more precise results then online learning but we could also reduce model complexity by implementing a day night cycle learning framework.
A hard problem remaining is the correct choice of hyperparameters as extensive search of a good set is still computationally infeasible. Correct use of pruning proved turned out to be difficult as a good heuristic is hard to find. However, the newly introduced concept achieved an increase of the average performance of the mixture of experts and created a modular basis that allows us to improve even more in the future by implementing other algorithms for each phase. In practice, this could help autonomous robots that have limited data storage and have to act autonomously for longer periods of time.
A lot of future work can still be done using the results of this thesis. Even tough we implemented a first simplistic pruning strategy we have yet to find a more sound strategy and future experiments should test different pruning strategies and their effects. Also while our approach works in lower dimensional space more testing not only in higher dimensional space, but in more realistic scenarios has to be done as the current version of online learning may not fit for higher dimensional learning. Online learning could also be significantly improved by finding a better way to automatically add new experts instead of one set threshold, where splitting occurs. This would also have the benefit of reducing the amount of hyperparameters one has to choose and optimize for.
A different possibly important feature of future online learning algorithms should be to include some form of time consolidation. Right now the maximum bound of our optimization is set to one and more tests should be made regarding it. An increase in the threshold should increase the convergence rate and probably yields better results locally, while decreasing it protects previously obtained knowledge better. In combination with the time consolidation used different setups may be tested and an optimal solution searched.
Another point to look at is the use of different learning architectures other then mixture of experts or using different models for the experts. Comparisons of linear experts to more sophisticated models such as neural networks are yet to be made and could yield interesting results and allow to learn more complex problems. Also the structure of mixture of experts can be changed in future experiments through the use of hierarchical mixture of experts.

Bibliography

[1] A. R. Luft and M. M. Buitrago, "Stages of motor skill learning," *Molecular neurobiology*, vol. 32, no. 3, pp. 205–216, 2005.

[2] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[3] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[4] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The wake-sleep algorithm for unsupervised neural networks," *Science*, vol. 268, pp. 1158–1161, 1995.

[5] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and autonomous systems*, vol. 15, no. 1-2, pp. 25–46, 1995.

[6] M. P. Walker and R. Stickgold, "Sleep-dependent learning and memory consolidation," *Neuron*, vol. 44, no. 1, pp. 121–133, 2004.

[7] IAS, "Our robots, http://www.ausy.tu-darmstadt.de/research/robots, accessed on 01.11.2017."

[8] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[9] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[10] L. Xu, M. I. Jordan, and G. E. Hinton, "An alternative model for mixtures of experts," in *Advances in neural information processing systems*, pp. 633–640, 1995.

[11] J. Eriksson, E. K. Vogel, A. Lansner, F. Bergström, and L. Nyberg, "Neurocognitive architecture of working memory," *Neuron*, vol. 88, no. 1, pp. 33–46, 2015.

[12] B. S. Lu and P. C. Zee, "Neurobiology of sleep," *Clinics in chest medicine*, vol. 31, no. 2, pp. 309–318, 2010.

[13] R. W. McCarley, "Neurobiology of rem and nrem sleep," *Sleep medicine*, vol. 8, no. 4, pp. 302–330, 2007.

[14] J. A. Hobson and E. F. Pace-Schott, "The cognitive neuroscience of sleep: neuronal systems, consciousness and learning," *Nature Reviews Neuroscience*, vol. 3, no. 9, pp. 679–693, 2002.

[15] C. F. Levinthal, *Introduction to physiological psychology*. Prentice-Hall, Inc, 1990.

[16] C. B. Saper, T. C. Chou, and T. E. Scammell, "The sleep switch: hypothalamic control of sleep and wakefulness," *Trends in neurosciences*, vol. 24, no. 12, pp. 726–731, 2001.

[17] G. Tononi and C. Cirelli, "Sleep function and synaptic homeostasis," *Sleep medicine reviews*, vol. 10, no. 1, pp. 49–62, 2006.

[18] G. Tononi and C. Cirelli, "Sleep and the price of plasticity: from synaptic and cellular homeostasis to memory consolidation and integration," *Neuron*, vol. 81, no. 1, pp. 12–34, 2014.

[19] M. G. Lee, O. K. Hassani, A. Alonso, and B. E. Jones, "Cholinergic basal forebrain neurons burst with theta during waking and paradoxical sleep," *Journal of Neuroscience*, vol. 25, no. 17, pp. 4365–4369, 2005.

[20] S.-H. Lee and Y. Dan, "Neuromodulation of brain states," *Neuron*, vol. 76, no. 1, pp. 209–222, 2012.

[21] R. Huber, M. F. Ghilardi, M. Massimini, and G. Tononi, "Local sleep and learning," *Nature*, vol. 430, no. 6995, pp. 78–81, 2004.

[22] F. Siclari, B. Baird, L. Perogamvros, G. Bernardi, J. J. LaRocque, B. Riedner, M. Boly, B. R. Postle, and G. Tononi, "The neural correlates of dreaming," *Nature neuroscience*, 2017.

[23] B. Knapp, *Skill in sport: the attainment of proficiency*. Routledge, 1963.

[24] J. Peters, K. Mülling, J. Kober, D. Nguyen-Tuong, and O. Krömer, "Towards motor skill learning for robotics," in *Robotics Research*, pp. 469–482, Springer, 2011.

[25] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, p. 201611835, 2017.

[26] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[27] V. Ramamurti and J. Ghosh, "Structurally adaptive modular networks for nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 10, no. 1, pp. 152–160, 1999.

[28] V. Ramamurti and J. Ghosh, "Structural adaptation in mixture of experts," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 4, pp. 704–708, IEEE, 1996.