UNIVERSITY OF CALIFORNIA
RIVERSIDE

User Performance Predictions for Cognitive Training

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Sanjana Sandeep

June 2018

Thesis Committee:

Dr. Christian Shelton, Chairperson
Dr. Vagelis Papalexakis
Dr. Ahmed Eldawy

The Thesis of Sanjana Sandeep is approved:

Committee Chairperson

University of California, Riverside

ABSTRACT OF THE THESIS

User Performance Predictions for Cognitive Training

by

Sanjana Sandeep

Master of Science, Graduate Program in Computer Science
University of California, Riverside, June 2018
Dr. Christian Shelton, Chairperson

An important strategy in cognitive training of working memory is to fine-tune task difficulty based on the subject's performance. In this project, we analyze data gathered from a cognitive-learning experiment via video games. Our contribution is an algorithm that predicts a subject's accuracy, with the aim of optimizing game-play progression. To model an individual's cognitive ability growth trajectory over time and predict performance for future challenges, we integrate probabilistic graphical models with a psychometric control equation. These models are trained and tested on study datasets generated from n-back cognitive training video games: *recollect*, *tapback* and *recall*. The expectation maximization (EM) technique is used to train these models. Results confirm that the models predicting user performance in n-levels do well.

Acknowledgments

I would like to thank my advisor, Dr. Christian Shelton, for providing me with continuous support and encouragement throughout my studies at UCR. I am very grateful to Dr. Aaron Seitz for trusting me and for being extremely patient in waiting for results. Finally, I thank my internship mentors, Boris Kalchev, Amlan Chatterjee and Kaushik Macherla for their contributions to my professional development.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document has two goals: (1) for statisticians and the machine learning community, to indicate how their mathematical literature is applied to a field such as neuroscience and (2) for neuroscientists and psychologists, to show how results from mathematical systems can directly relate to their area of work. In cognitive psychology, working memory is defined as a subject's cognitive system workspace that holds information for short periods of time, that is mainly required for processing [23]. It is analogous to a computer's RAM. As an example, working memory is used to hold numbers that need to be carried across while performing long multiplications. A distinguishing feature is it's aspect of enhancing cognitive ability by training. The idea of working memory training is based on the premise that learning is dynamic. Working memory training aims to improve working memory capacity [3].

In recent years, video game software has been designed with the explicit goal of enhancing cognitive abilities. The Brain Game Center (BGC) at UCR is a research unit that

is dedicated to mental fitness. It aims to make scientifically optimized brain games that translate to performance in real-life activities [11]. One of the major components of video game design is *game-play progression*, which is defined as the set of rules to be followed by the player, in order to advance in the game. These rules are usually created by the video game designer. Game-play progression is a central topic in brain-training video games.

1.1 Problem Statement

One of the challenges in working memory training is deciding the most suitable challenge that would result in the subject learning by being engaged the most. The most suitable next challenge avoids overwhelming or underwhelming the user. In order to achieve this, a step in the right direction is to develop a system that could evaluate and predict a subject's performance provided the next challenge. The behavior of a subject at a certain point of time not only depends on the momentary stimulus, but also on the history of the stimuli. A statistical model that can temporally encode a subject's overall skill and predict his or her performance for future challenges is required.

The cognitive training video games developed at the BGC incorporate the game-play progression design principle called the *n-back*, which is the continuous updating working memory task. The attribute that we are attempting to measure is the subject's memory load represented by the n-level. The load factor (n) is directly proportional to the difficulty of the task. In the n-back task, a subject is presented with a sequence of stimuli, and he or she must identify if the current stimulus matches the one that was presented n steps ago. Figure 1.1 illustrates the color-matching n-back task [1]. The goal of this project is to come

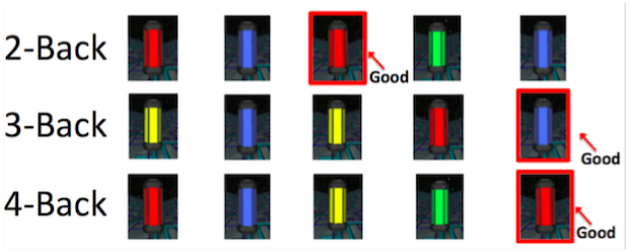


Figure 1.1: Examples of the color-matching task

up with an algorithm that estimates the difficulty range of a subject for cognitive training via these n-back video games.

1.2 Description of the Games

The n-back video games developed at the BGC, that are specifically used in the experiments of this project come in three flavors: *recall*, *tapback* and *recollect*.

1.2.1 The Recall Game

Recall is the n-back task in a space themed setting and is runner-styled. As a sequence of fuel cells with varying patterns are presented, the player must zap target fuel cells (match the n as from n-back) and dodge detractors. Figure 1.2 is a screen-shot of the game.

1.2.2 The Tapback and Recollect Games

The two versions of recollect are it's gamified and non-gamefied versions. Tapback is the non-gamified, most basic rendition of the n-back cognitive task. A subject is presented

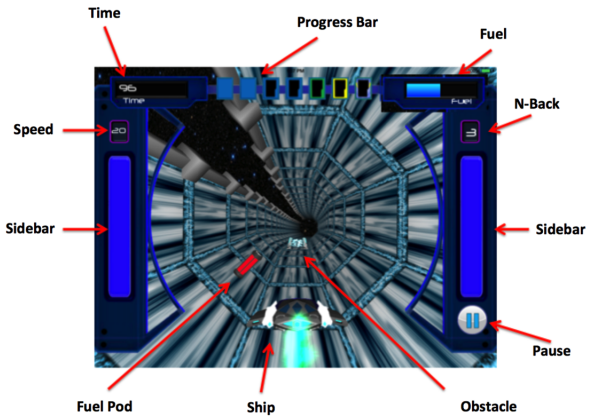


Figure 1.2: Screen of recall

with a sequence of circles with varying colors. The task is to match circles based on n-back. Figure 1.3 is a screen-shot of tapback.

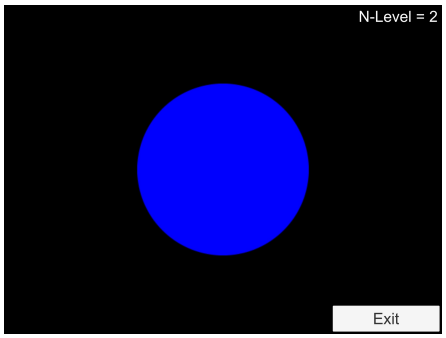


Figure 1.3: Screen of the tapback game.

Figure 1.4: Screen of the recollect game.

The gamified version is recollect, a space-themed game just like recall. The player is an astronaut who must collect resources in a sequence (based on matching n-back). The targets are plants and minerals, while the detractors are obstacles and hazardous materials. Figure 1.4 is a screen-shot of recall.

1.3 Time Series Analysis

A subject goes through a series of computerized training sessions for multiple days at the BGC. A training day consists of one or more training sessions. A training session is broken into a sequence of training *blocks*, wherein a block is defined as a single uninterrupted player-game interaction. Data gathered from this experiment has a dynamic and temporal structure. Thus, we are able to pose this as a time-series problem. The subjects' accuracies are recorded values, forming the observation time series $y_{1:T} = (y_1, ..., y_T)$. To draw inference from such a series, we build a latent variable model to understand the underlying learning mechanism of the subject that generates this series. Figure 1.5 shows how *true positives* (TP) or *hits*, *false positives* (FP) or *false alarms* and *true negatives* (TN) or *misses* are calculated for a *block*. For all experiments in this piece of work, accuracy per block is calculated as follows:

$$accuracy = \frac{TP}{TP + TN + FP}$$

Remark: The true equation for accuracy is $\frac{TP+TN}{TP+TN+FP+FN}$

1.4 Psychometric Modeling

In order to model the psychometrical latent variable, we generate a curve to characterize learning as the probability of a correct response. Since each stimulus/event in a block can either be a correct or an incorrect response, it is described by a *Bernoulli Probability Model*. Keeping this in mind, we compute a learning curve, which estimates the probability of a correct response as a function of the ability of the subject and the effec-

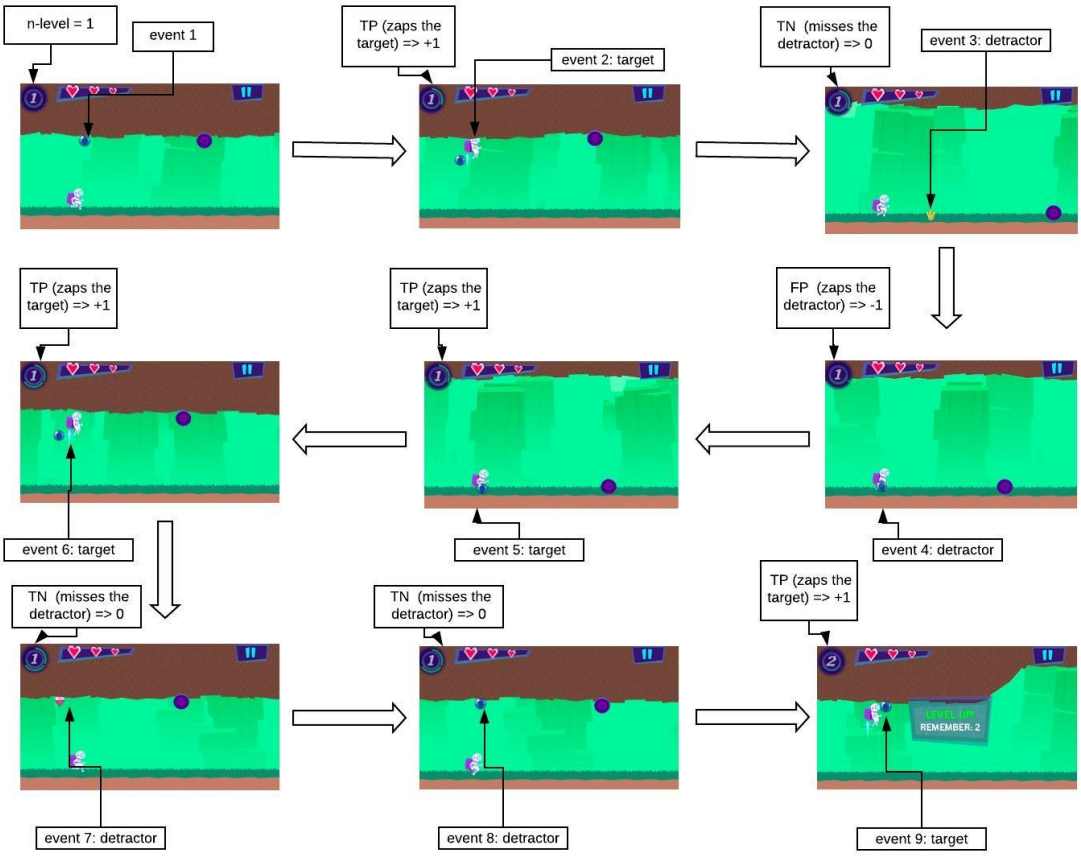


Figure 1.5: Sequence of events in a block of Recollect.

tiveness of the item in question. This dynamic Bernoulli response pattern is captured by the *item characteristic curve* (ICC) given by the following equation [22]:

$$q_{i,t} = c_i + \frac{(d_i - c_i)}{1 + e^{-\rho_i(x_{i,t} - z_{i,t})}} \tag{1.1}$$

where at t, $x_{i,t}$ is the ability of the subject i; the difficulty of a test item is indicated by $z_{i,t}$ which corresponds to the n-back level of the test; ρ_i , c_i and d_i are additional static item parameters accounting for the subject's discrimination, guess-ability and carelessness (accounting for noise) respectively; and $q_{i,t}$ is the Bernoulli item response variable which

corresponds to the correctness of the response to a stimulus in an block. This model assumes that the probability of a subject's correct response to a test item is a logistic function of the difference between the subject's ability and the difficulty of the item. It is assumed that the items are dichotomous, the function is strictly monotonic on the latent trait scale and all items are unidimensional.

For the purpose of better understanding, we provide an example that compares two subjects' ICC in Figure 1.6. The difficulty location index is the point on the x-axis at which the curve

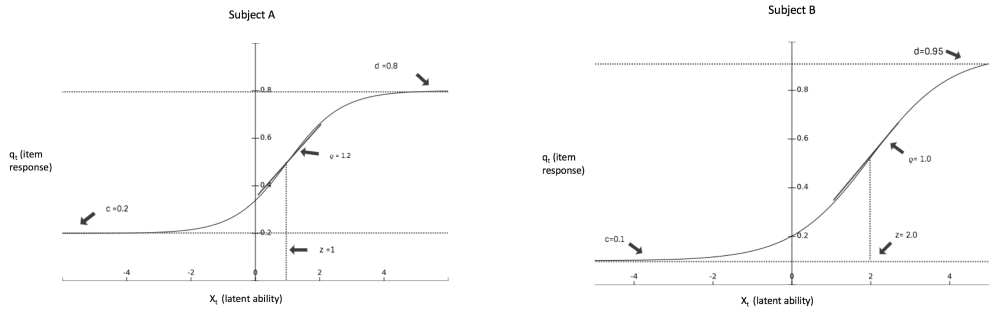


Figure 1.6: Item Characteristic Curve (ICC) for *subject A* and *subject B* at a particular event t.

crosses the midpoint of the two asymptotes on the y-axis. *Subject A* has an easier item function when compared to that of *subject B*, since $z_t(A) < z_t(B)$. *Subject A* has a steeper curve when compared to *subject B*, and hence has a higher discrimination (indicated by ρ). The lower asymptote parameter c describes how easy it is for a low ability subject to get a correct response and the upper asymptote d tells us how easy it is for a high ability subject to get an incorrect response. *Subject A* is more sensitive to noise effects, since $c_A > c_B$ and $d_A > d_B$.

Chapter 2

Modeling with a State Space Framework

A state space model (SSM) model is a probabilistic graphical model that describes the *observation time series* by the sequential evolution of the underlying *hidden state*. A first-order *Markov chain* model exhibits a property where the probability distribution of the next state is independent of the preceding sequence, conditioned only on the current state. The SSM is Markovian. Each training instance corresponds to a subject, with a *block* (see section 1.3) corresponding to the notion of time. In this section we consider multiple instances of the SSM. In Figure 2.1 the hidden state x_t encodes the hidden cognitive ability at time t. The observation y_t is the accuracy achieved at time t. The control input z_t corresponds to the difficulty (fixed n-back-level presented corresponding to the task), and the arrows denote conditional dependencies.

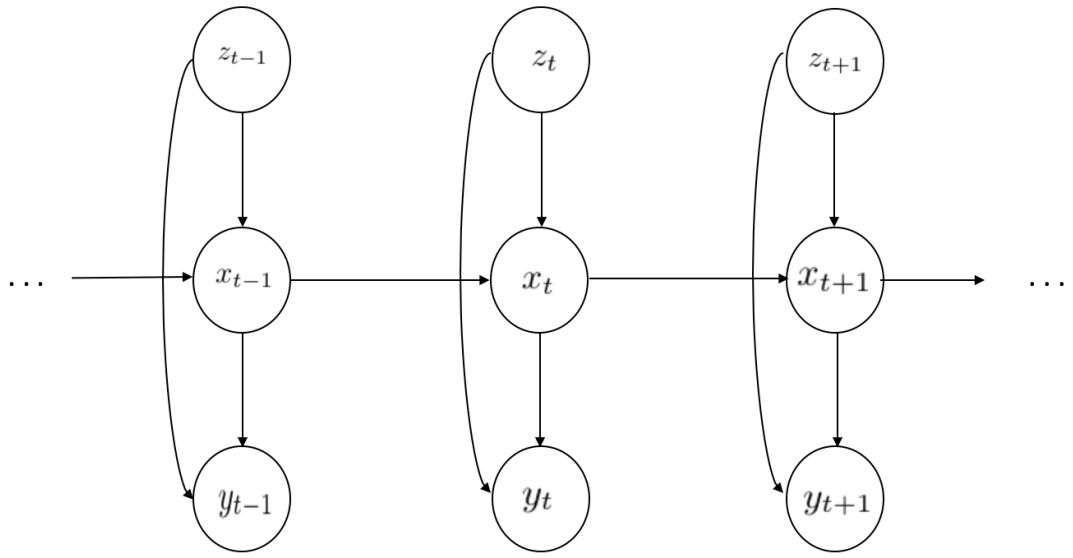


Figure 2.1: Graphical Representation of the SSM

2.1 Modeling with a HMM

The Hidden Markov Model (HMM) is an instance of a SSM with a discrete hidden state space. The behavior of the HMM is fully described by a three sets of probabilities :

- *Transition probabilities* : The transition matrix A is a $K \times K$ matrix, where K is the number of states. Since the hidden Markov chain is time-homogeneous, A is a time-independent stochastic matrix. Here, an element in the matrix $a_{ij} = P(x_{t+1} = j \mid x_t = i)$

- *Emission probabilities* : The emission matrix B is also represented as a $K \times T$ matrix, where T is the number of states. The emission probability $b_j(y_t) = P(y_t \mid x_t = j)$

- *Initial state probabilities*: Π is a vector of K elements. $\Pi_i = P(x_1 = i)$

Since the state space $\mathbf{X}$ is discrete and finite, we fix the range of $\mathbf{X}$ to be in the range of the n-backs presented. Revisiting psychometric modeling, we get the Bernoulli item response for an event denoted by q_t as described by equation 1.1. The sequence of Bernoulli response variables which are independent and identically distributed result in a Binomial observation density model for a block, described by the equation below:

$$P(y_t \mid x_t, z_t) = \binom{m_t}{r_t} q_t^{r_t} (1 - q_t)^{m_t - r_t} \tag{2.1}$$

$P(y_t \mid x_t, z_t)$ is the underlying observation (emission) probability at t of the HMM, which is also dependent on the incoming n-back level. However, the transition probabilities and the initial probabilities are still unknown, and are estimated using the *expectation maximization (EM)* algorithm: an algorithm that estimates model parameters by iteratively optimizing the likelihood function.

2.1.1 EM

Maximum likelihood (ML) estimation of generative statistical models with latent variables is often carried out via EM [7]. We denote the unknown parameters of the model by θ. The size of the state space $\mathbf{X}$ is m. The observation sequence corresponding to subject i is $Y^{(i)} = (y_1 ... y_T)$. The observation space is $\mathbf{D} = (Y_1 ... Y_N)$ where N is the number of subjects. The likelihood of the dataset given by:

$$P(D \mid \theta) = \prod_{i=1}^{N} P(Y^{(i)} \mid \theta).$$

The ML estimates for the parameters can be computed by maximizing $L(\theta \mid D)$ which is the *log-likelihood* (equivalent to likelihood, taken for it's computational simplicity and the

concave nature of the curve). The log-likelihood for a single observation sequence is given by:

$$L(\theta \mid Y) = \log(Y \mid \theta) = \log \sum_X P(Y, X \mid \theta)$$

Summing over all possible paths to maximize the sum is a difficult and inefficient solution. Instead, EM iteratively constructs a lower bound on $L(\theta \mid Y)$ and maximizes this lower bound.

$$\begin{aligned} \log \sum_X P(Y, X \mid \theta) &= \log \sum_X Q(X) \frac{P(Y, X \mid \theta)}{Q(X)} \\ &= \log \mathbf{E_q}[\frac{P(Y, X \mid \theta)}{Q(X)}] \\ &\geq \mathbf{E_q}[\log \frac{P(Y, X \mid \theta)}{Q(X)}] \end{aligned}$$

An arbitrary averaging distribution over X, is introduced as $Q(X)$. The last step is arrived at by generalizing *Jensen's inequality* [7] to expectations : $\mathbf{E}[f(X)] \geq f(\mathbf{E}[X])$. The final EM objective function is given as

$$L(\theta, Q) = \mathbf{E_q}[\log P(X \mid \theta)] + \mathbf{E_q}[\log P(Y \mid X, \theta)] - \mathbf{E_q}[\log Q(X)].$$

The lower bound for the log-likelihood is provided by the expected complete log likelihood and the entropy of the averaging distribution. Given this objective, the *E step* maximizes the $L(\theta, Q)$ over Q. The *M step* maximizes this lower bound over θ holding Q fixed. The algorithm iterates alternatively between these two steps, starting with some random initialization of the initial parameters. The EM can be viewed as the co-ordinate ascent on $L(\theta, Q)$, given by the following steps

$$Estep : Q_{t+1} = argmax_Q L(\theta, Q)$$

$$Mstep : \theta_{t+1} = argmax_\theta L(\theta, Q)$$

The EM algorithm is best suited in settings that have closed form expressions for the conditional expectation and maximization. It always monotonically increases the log-likelihood, therefore guaranteeing convergence. A drawback is that we do not automatically get the estimates of parameter variance.

2.1.2 EM for the HMM

The complete log-likelihood (latent variables and observations) is given by the following equation:

$$\log P(X, Y) = \log(\sum_X P(x_{t=1}) \prod_{t=2}^{T} P(x_t \mid x_{t-1}, \theta) \prod_{t=2}^{T} P(y_t \mid x_t, \theta)). \tag{2.2}$$

From the above equation, we notice that we cannot decompose the problem into independent ML subproblems; hence we apply EM. EM applied to an HMM is specifically called the *Baum Welch* algorithm [17]. The parameters that need to be estimated are $\theta = (A, \Pi)$. The emission formula is fixed and it's parameters are estimated separately (see section 2.1). The inference algorithm for the calculation of the state posteriors is called the *forward-backward algorithm.*

To efficiently compute the posterior marginals of all latent states $P(x_t \mid y_{1:t}), t \in (1...T))$, a dynamic programming algorithm is used, that can be completed in linear time, rather than an inefficient brute force technique that searches over all possible state sequences.

The forward-backward algorithm:

- The forward pass calculates $\alpha_t(i) = P(y_{1:t}, x_t = i \mid \theta)$

$$\alpha_1(i) = \Pi_i b_i(y_1) \quad 1 \leq i \leq N$$
$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(y_{t+1}) \quad 1 \leq t \leq T-1$$

- The backward pass goes backward in time to compute the backward message $\beta_t(i) = P(y_{t+1:T} \mid x_t = i, \theta)$

$$\beta_T(i) = 1 \quad 1 \leq i \leq N$$
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_{t+1}(j) \quad 1 \leq t \leq T-1$$

The EM algorithm is given below:

- E step: The degree to which each state explains each data point is the state posterior, and the degree to which the combination of the current and the previous states contribute to the observations is defined by the transition posterior. Summation of these quantities over t can be interpreted as expectations.

$$\gamma_t(i) = P(x_t = i \mid y_{1:T}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$
$$\xi_t(i,j) = P(x_t = i, x_{t+1} = j \mid y_{1:T}) = \frac{\alpha_t(i) a_{ij} b_j(y_{t+1)\beta_{t+1}(j)}}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(y_{t+1})\beta_{t+1}(j)}$$

- M step: The re-estimation guarantees that $\hat{\theta} = (\hat{A}, \hat{\Pi})$, from which the observation sequence is more likely to be generated than $\theta = (A, \Pi)$.

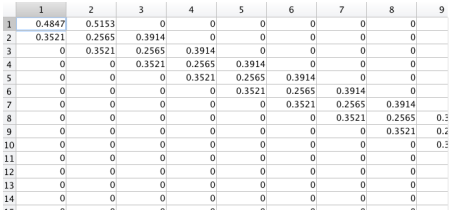
$$\hat{a_{ij}} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{n=1}^{N}\sum_{t=1}^{T-1} \gamma_t(i)}$$
$$\hat{\Pi}_i = \frac{\Sigma_{n=1}^{N} \gamma_{t=1}(i)}{N}$$

The EM algorithm iteratively alternates between the E step and the M step, until a convergence criteria is met. As the length of the subsequence grows, the values of α and β become extremely small and are *renormalized* at every step. The renormalization procedure consists of multiplying $\alpha_t(i)$ by a scaling coefficient that is independent of i. The scaled $\alpha_t(i)$ is maintained within the dynamic range $1 \leq t \leq T$. The coefficients of $\beta_t(i)$ are scaled similarly. At the end of the computation, the scaling coefficients are canceled [17]. Initialization of parameters must be random (to break the symmetry of the model). Prior information can also be set to these parameters, in order to speed up computation in the direction of local maximum.

The hyper-parameters of the HMM are the item-parameters (ρ, c, d), required in equation 1.1. They are picked via *grid search*, which is a technique of exhaustively searching through a manually specified subset of the hyper-parameter space. Log-likelihood score is the performance metric on the training set. The parameter space is discretized with manually set bounds.

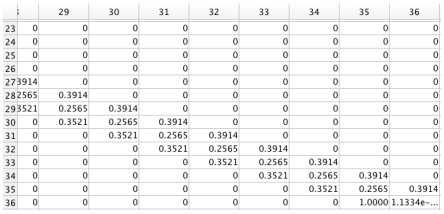
Making the transition matrix A sparse makes our model less prone to overfitting. The underlying state sequence of the model has the property that between consecutive time steps, the state index either stays the same or increases or decreases by 1. A subject is either upgraded or downgraded by at-most 1 level. Attributing to this fact, we propose a tri-diagonal state space model. We also share transition parameters across each diagonal, implying that the probability of adjacent transitions are identical across all-time-steps. From the perspective of the game, a subject's probability of transitioning from a 1-level to

a 2-level is the same as the probability of him transitioning from level 6 to 7. Although, this assumption does not seem practical in an ideal training setting, with this constraint, we can significantly reduce the number of parameters. Parameter-tying across each diagonal is done at every iteration of the re-estimation procedure. Figures 2.2 and 2.3 show the top and bottom portions of a re-estimated transition matrix.



	1	2	3	4	5	6	7	8	9
1	0.4847	0.5153	0	0	0	0	0	0	
2	0.3521	0.2565	0.3914	0	0	0	0	0	
3	0	0.3521	0.2565	0.3914	0	0	0	0	
4	0	0	0.3521	0.2565	0.3914	0	0	0	
5	0	0	0	0.3521	0.2565	0.3914	0	0	
6	0	0	0	0	0.3521	0.2565	0.3914	0	
7	0	0	0	0	0	0.3521	0.2565	0.3914	
8	0	0	0	0	0	0	0.3521	0.2565	0.3
9	0	0	0	0	0	0	0	0.3521	0.2
10	0	0	0	0	0	0	0	0	0.3
11	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	

Figure 2.2: Top portion of the transition matrix.



	29	30	31	32	33	34	35	36
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0
27	3914	0	0	0	0	0	0	0
28	2565	0.3914	0	0	0	0	0	0
29	3521	0.2565	0.3914	0	0	0	0	0
30	0	0.3521	0.2565	0.3914	0	0	0	0
31	0	0	0.3521	0.2565	0.3914	0	0	0
32	0	0	0	0.3521	0.2565	0.3914	0	0
33	0	0	0	0	0.3521	0.2565	0.3914	0
34	0	0	0	0	0	0.3521	0.2565	0.3914
35	0	0	0	0	0	0	0.3521	0.2565
36	0	0	0	0	0	0	0	1.0000

Figure 2.3: Bottom portion of the transition matrix.

2.1.3 Posterior Inference

Once we have the best estimates for the parameters of the model, our model should be able to predict the observation time-series. Predicting one-step ahead is a problem of computing the conditional expectation of y_t given $y_{1:t-1}$ at t denoted by $\hat{y}_t$.

$$\begin{aligned} \hat{y}_t &= \mathbf{E}(y_t \mid y_{1:t-1}) \\ &= \sum_x \alpha_t(x)\mathbf{E}(y_t \mid x) \\ &= \sum_x \alpha_t(x) q_t \end{aligned} \tag{2.3}$$

In equation (2.3), we arrive to step 2 from step 1 by marginalizing the expectation over all possible state paths. The mean of the Binomial observation model, given by $q_t m_t$ gives us

the value for the conditional expectation $\mathbf{E}(y_t \mid x)$. Since, we require accuracy, $\hat{y}_t$ becomes $\frac{(q_t m_t)}{m_t}$, giving rise to the result in step 3. To overcome limitations of having a discrete state-space, we now move onto exploring SSM with a continuous state space.

2.2 Modeling with a UKF

A Kalman filter model (KF) is a Bayes' filter, where the states and the observations are Gaussian distributions.TODO:cite It is a special case of a HMM, with a continuous and infinite state space. The well known Kalman filter is only suitable for linear systems. We consider our system to be of non-linear nature. The state space framework consists of:

$$x_t = \mathbf{A}x_{t-1} + \mathbf{B}z_t + \mathbf{v}_t, \quad v_t \sim \mathcal{N}(0, Q_t) \tag{2.4}$$

$$y_t = \mathbf{g}(x_t, u_t, z_t) + \mathbf{w}_t, \quad w_t \sim \mathcal{N}(0, R_t) \tag{2.5}$$

In equation (4.1), $\mathbf{A}$ is the state transition matrix applying the first-order Markov effect; $\mathbf{B}$ is the control input matrix that applies the effect of the control input parameters on the states, the process noise for the parameters in the state space framework is $\mathbf{v}_t$, where it is assumed to be drawn from a zero mean multivariate normal distribution with covariance matrix $\mathbf{Q}$. Equation (4.2) describes the measurements of the system. Function $\mathbf{g}$ maps the states and the control inputs to the observations. Similar to the process noise is the observation noise, $\mathbf{w}_t$, that is assumed to be zero mean Gaussian white noise, with covariance matrix $\mathbf{R}$. Only the measurement sequence is observed while the state and the noise variables are latent. The observation model is chosen to be the control IRT equation (2.1).

$$g(x_t, u_t) = c + \frac{(d-c)}{1 + e^{-\rho_t(x_t - u_t)}} \tag{2.6}$$

In the HMM, the item-parameters were picked via cross-validation. We hold this value to be consistent across all training subjects. This assumes all subjects to possess equal discrimination power. However, in a practical scenario, this might not be true. A subject's discrimination is dynamic, hence we incorporate this behavior as another dimension in the state space. A multi-dimensional state variable is proposed: $x_t = \begin{bmatrix} \hat{x_t} \\ \\ \rho_t \end{bmatrix}$ where the slope; ρ of (1.1), is the second dimension in the latent state space.

When there is a need to solve problems of non-linear nature, two versions of the non-linear Kalman filter: the *extended Kalman filter* (EKF) and the *unscented Kalman filter* (UKF) [15] can be used. Both these filters approximate non-linear estimation. The EKF linearizes the nonlinear models so that the traditional linear KF equations can be applied on it. The state distribution that is represented by a Gaussian random variable is propagated through the first-order linearization of the non-linear system. However, forcing linearization causes the EKF to be unstable (when assumptions of local linearity is violated) and causes it to converge slowly to the right solution. The UKF addresses this problem, by approximating the Gaussian distribution instead of approximating the nonlinear transformation function itself. We use the UKF to estimate our underlying accuracy time-series that is corrupted by Gaussian white noise.

2.2.1 Unscented Transform

The *unscented transformation* is a deterministic sampling approach, which picks a minimal set of sample points, that match the first, second and third order moments of the underlying Gaussian state distribution, thereby capturing it's true mean and covariance.

Suppose x_t is the Gaussian random variable state at time t that undergoes a non-linear transformation with a mean $\hat{x}$ and covariance V_x.The unscented transformation method of capturing the statistics of this variable, is as follows [15]:

- Step 1: Calculate $2L+1$ *sigma vectors* $\mathcal{X}_i$, where L=2 is the number of dimensions and $\lambda = \alpha^2(L+\kappa) - L$ is the scaling parameter. κ is another scaling parameter set to 0. $\alpha = 10^{-4}$ is the spread around the mean.

$$\mathcal{X}_0 = \hat{x}$$

$$\mathcal{X}_i = \hat{x} + (\sqrt{(L+\lambda)V_x})_i \quad i = 1, ..., L$$

$$\mathcal{X}_i = \hat{x} - (\sqrt{(L+\lambda)V_x})_{i-L} \quad i = L+1, ..., 2L$$

- Step 2: Propagate the sigma vectors through the non-linear function g.

$$\mathcal{Y}_i = g(\mathcal{X}_i)$$

- Step 3: Compute the weights associated with the sigma points.

$$\mathcal{W}_0^{(m)} = \frac{\lambda}{(L+\lambda)}$$

$$\mathcal{W}_0^{(c)} = \frac{\lambda}{(L+\lambda)} + (1 - \alpha^2 + \beta)$$

$$\mathcal{W}_i^{(m)} = \mathcal{W}_i^{(c)} = \frac{1}{2(L+\lambda)}$$

- Step 4: Calculate the weighted mean and covariance.

$$\hat{y}_t \approx \sum_{i=0}^{2L} (\mathcal{W}_i^{(m)} \mathcal{Y}_i) \tag{2.7}$$

$$V_{yyt} \approx \sum_{i=0}^{2L} (\mathcal{W}_i^{(c)} (\mathcal{Y}_i - \hat{y})(\mathcal{Y}_i - \hat{y})^T) \tag{2.8}$$

2.2.2 Dual Estimation

The simultaneous estimation of the augmented latent state x_t and the model parameters $\Theta = (A, B, Q, R, \mu_0, \Sigma_0)$ via the UKF, is the dual estimation problem. In this section, we bring together the Kalman filter and the EM algorithms.

Bayesian filtering and smoothing distributions are the essentials of estimation.

- $Pr(x_t \mid y_1....y_{t-1}, \Theta)$ are filtering distributions.
- $Pr(x_t \mid y_1....y_T, \Theta)$, for $t \leq T$, are smoothing distributions.

The UKF Algorithm equations are given below [15]:

- Initialization:

$$x_{1|0} = \mu_0$$

$$V_{1|0} = \Sigma_0$$

- Predict Step: (computing a-priori estimate)

$$x_{t|t-1} = \mathbf{A}x_{t-1|t-1} + \mathbf{B}u_t$$

$$V_{t|t-1} = \mathbf{A}\hat{V}_{t-1|t-1}\mathbf{A}^{tr} + \mathbf{Q}$$

- Apply steps 1-4 from previous section.

- Compute Kalman gain:

$$\hat{V}_{xyt} = \sum_{i=0}^{2L} \mathcal{W}_i^{(c)} (\mathcal{X}_i - \hat{x}_t^-)(\mathcal{Y}_i - \hat{y})^T$$

$$\mathcal{K}_t = \hat{V}_{yyt}^{-1} \hat{V}_{xyt}$$

- Update step: (computing a-posteriori estimate)

$$x_{t|t} = x_{t|t-1} + \mathcal{K}_t(y_t - \hat{y}_t)$$

$$V_{t|t} = V_{t|t-1} - \mathcal{K}_t \hat{V}_{yyt} \mathcal{K}^T$$

$$V_{t-1,t|t} = \hat{V}_{xyt}$$

The trajectory estimates obtained via smoothing due to additional information available, tend to be smoother than the filtering estimates. Since we have a linear-dynamics model, we stick to a Rauch-Tung-Striebel (RTS) smoother, which is a closed form smoother for linear-Gaussian models.

The backward recursion equations for the smoothened means and covariances are as follows:

- Computing the smoother gain matrix :

$$J_t = V_{t|t} \mathbf{A}^T V_{t+1|t}^{-1}$$

- Computing the smoothened mean and variance:

$$x_{t|T} = x_{t|t} + J_t(x_{t+1|T} - x_{t+1|t})$$

$$V_{t|T} = V_{t|t} + J_t(V_{t+1|T} - V_{t+1|t}) {J_t}^T$$

- Computing lag-one covariance:

$$V_{t-1,t|T} = V_{t-1,t|t} + V_{t-1,t|t}(V_{t|T} - V_{t|t})V_{t|t}^{-1}$$

The smoothened state is an optimal estimate at an instant of time. Using the smoothened estimates and lag-one covariance, we can compute the *sufficient statistics*, a summary of the data, useful for learning purposes. In our case, the *expected sufficient statistics* are computed in the *E step* of the EM algorithm. The EM algorithm for our framework is given below:

- Initialize the parameters Θ and evaluate the initial value of log-likelihood.
- Iterate until convergence:
  - *Expectation step*: Perform Kalman filter and RTS smoother. Compute:

$$\langle y_t y_t^T \rangle = \sum_t y_t y_t^T$$

$$\langle y_t x_t^T \rangle = \sum_t y_t x_{t|T}^T$$

$$\langle y_t u_t^T \rangle = \sum_t y_t u_t^T$$

$$\langle x_t u_t^T \rangle = \sum_t x_{t|T} u_t^T$$

$$\langle u_t u_t^T \rangle = \sum_t u_t u_t^T$$

$$\langle x_t x_t^T \rangle = \sum_t (V_{t|T} + x_{t|T} x_{t|T}^T)$$

$$\langle y_t \hat{y}_t^T \rangle = \sum_t y_t \hat{y}_t^T$$

$$\langle \hat{y}_t \hat{y}_t^T \rangle = \sum_t (\hat{y}_t \hat{y}_t^T + V_{yyt})$$

$$\langle x_{t-1} x_t^T \rangle = \sum_t (V_{t-1,t|T} + x_{t-1|T} x_{t|T}^T)$$

$$\langle x_{t-1} x_{t-1}^T \rangle = \sum_t V_{t-1|T} + x_{t-1|T} x_{t-1|T}^T$$

$$\langle x_t u_{t-1}^T \rangle = \sum_t x_{t|T} u_{t-1}^T$$

$$\langle x_{t-1} u_{t-1}^T \rangle = \sum_t x_{t-1|T} u_{t-1}^T$$

$$\langle u_{t-1} u_{t-1}^T \rangle = \sum_t u_{t-1} u_{t-1}^T$$

- *Maximization step*: Update the parameters as follows:

$$M = \begin{bmatrix} \sum_{n=1}^{N}\langle x_{t-1}x_{t-1}^{tr}\rangle & \sum_{n=1}^{N}\langle x_{t-1}u_{t-1}^{tr}\rangle \\ \sum_{n=1}^{N}\langle x_{t-1}^{tr}u_{t-1}\rangle & \sum_{n=1}^{N}\langle u_{t-1}u_{t-1}^{tr}\rangle \end{bmatrix}^{-1} \begin{bmatrix} \sum_{n=1}^{N}\langle x_{t-1}x_{t}^{tr}\rangle & \sum_{n=1}^{N}\langle x_{t}u_{t-1}^{tr}\rangle \end{bmatrix}$$

$$\mathbf{A} = M[:, 1:2]$$

$$\mathbf{B} = M[:, 3:5]$$

$$\mathbf{Q} = \frac{\begin{bmatrix} \sum_{n=1}^{N}\langle x_t x_t^T\rangle - \sum_{n=1}^{N}\langle x_{t-1}x_t^T\rangle\mathbf{A}^T - \mathbf{A}\sum_{n=1}^{N}\langle x_{t-1}^T x_t\rangle - \\ \sum_{n=1}^{N}\langle x_t u_{t-1}^T\rangle\mathbf{B}^T - \mathbf{B}\sum_{n=1}^{N}\langle x_t^T u_{t-1}\rangle + \mathbf{A}\sum_{n=1}^{N}\langle x_{t-1}u_{t-1}^T\rangle\mathbf{B}^T + \\ \mathbf{B}\sum_{n=1}^{N}\langle x_{t-1}^T u_{t-1}\rangle\mathbf{A}^T + \mathbf{A}\sum_{n=1}^{N}\langle x_{t-1}x_{t-1}^T\rangle\mathbf{A}^T + \mathbf{B}\sum_{n=1}^{N}\langle u_{t-1}u_{t-1}^T\rangle\mathbf{B}^T \end{bmatrix}}{\sum_{n=1}^{N} T_n - N}$$

$$\mathbf{R} = \frac{\left[\sum_{n=1}^{N}\langle y_t y_t^T\rangle - \sum_{n=1}^{N}\langle y_t\hat{y}_t^T\rangle - \sum_{n=1}^{N}\langle y_t^T\hat{y}_t\rangle + \sum_{n=1}^{N}\langle \hat{y}_t\hat{y}_t^T\rangle\right]}{\sum_{n=1}^{N} T_n}$$

$$\mu_0 = \frac{\sum_{n=1}^{N} x_{1|T}}{N}$$

$$\Sigma_0 = \frac{\sum_{n=1}^{N} V_{1|T}}{N} - \mu_0\mu_0^T$$

where N = *number of sequences* and T_n = *length of sequence n*.

2.2.3 Limitations

Earlier in the chapter, we mentioned that the strength of the UKF is that it avoids derivatives and is not a local approximation, instead is based on values covering a larger area. However, it should be kept in mind that this is not a truly global approximation. The fundamental assumption in our framework is that the state, observation and noise

terms are modeled as Gaussian random variables, which computationally simplifies the Bayesian recursion. This assumption maybe unrealistic; thereby the predictions are never fully-accurate. The UKF is computationally more expensive than our HMM model, e.g: Cholskey factorizations need to be applied at every time-step during filtering, for calculating the sigma vectors. If we intend to increase the dimensionality of the state space, we will require more data for convergence, since there are more parameters to estimate. Finally, both these models do not differentiate between users based on performance. We can hope to get better results if we cluster subjects, based on their level of performance.

Chapter 3

Extending to Mixture Models

In this section, we extend our framework to mixture models. Besides forming a framework for building complex distributions, mixture models can also be used to cluster data. Our motivation comes from the goal of building a descriptive model for the data, rather than prediction per se.

3.1 Clustering with HMMs

The aim is to cluster subject sequences into K clusters, such that all subject sequences belonging to a specific group have similar performance. In a more colloquial sense, our goal is to be able to differentiate the fast learners from the slow learners.
There are two types of clustering mechanisms, based on method:

- *Hard clustering* : In this method, an observation is assigned to a single cluster and therefore, the partition of observations is predicted.

- *Soft clustering* : In this method, an observation is assigned the likelihood of belonging

to each cluster, instead of being assigned to the cluster itself.

Our first assumption is that K is known, and the second is that the distributions do not carry equal weight with respect to the contributions to the observed data. This fact is represented by associating a prior probability: π_k with each cluster distribution. A subject is characterized by the following mixture probability density function:

$$Pr(Y_i \mid \Theta) = \Sigma_{k=1}^{K} \pi_k Pr(Y_i \mid \theta_k),$$

where $\Theta = (\theta_1 ... \theta_K, \pi_1 ... \pi_K)$ and θ_k corresponds to the parameters for just the k^{th} component HMM.

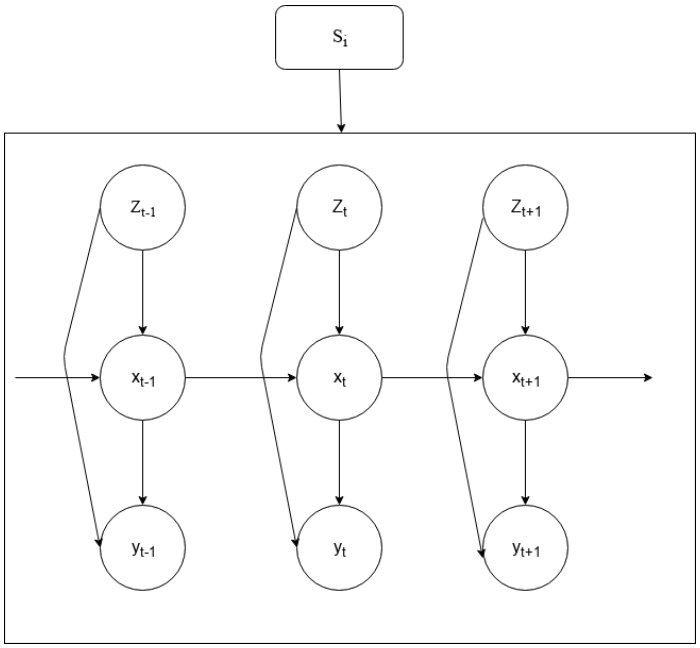


Figure 3.1: Graphical representation of the Mixture HMM for a subject i.

The purpose of EM clustering is to propose a soft clustering method. The goal is to estimate the parameters of each cluster component, so as to maximize the likelihood

of the observed data. We first assume that K is known. These mixture models can be interpreted in terms of discrete latent variables, that define assignments of data points to specific components of the mixture. We introduce unobserved random variables S_i that are the indicator variables from whom the observation sequence Y_i is sampled. The EM algorithm for clustering the hidden Markov mixture model with K components is given below:

- Choose prior π^0 in the probability simplex and randomly initialize the parameters, $\theta_k^0 = (A_k, \Pi_k)$, where $k = 1...K$
- Iterate until convergence: l=0
  - For each sequence i, where i = 1 ... N:
  - *Expectation step*:
    - Compute the *responsibilities* using current parameter values

$$w_k^l = Pr(S_i = k \mid y_{1:T}, \Theta^l) = \frac{\pi_k^l Pr(y_{1:T} \mid \theta_k^l)}{\Sigma_{k=1}^K \pi_k^l Pr(y_{1:T} \mid \theta_k^l)}$$

      $Pr(y_{1:T} \mid \theta_k^l)$ is computed via the *forward* pass (see section 2.12).
    - *Maximization step*:
      - Re-estimate the parameters using the current responsibilities.

        Revisiting section 2.12 yields $\xi_t^l(i,j)$ and $\gamma_t^l(i)$

$$\pi_k^{l+1} = \frac{\Sigma_{k=1}^{K} w_k^l}{N}$$
$$\hat{a_{ij}^{l+1}} = \frac{\sum_{n=1}^{N} w_k^l \sum_{t=1}^{T-1} \xi_t^l(i,j)}{\sum_{n=1}^{N} w_k^l \sum_{t=1}^{T-1} \gamma_t^l(i)}$$
$$\hat{\Pi}_i^{l+1} = \frac{\Sigma_{n=1}^{N} \gamma_{t=1}^l(i)}{N}$$

During post processing, inference of the underlying cluster for a sample sequence is determined by evaluating the responsibilities for each of the models, and assigning it to the model with the maximum value.

3.2 Clustering with UKFs

When the discrete latent state variable specifies assignments at each time t, we call it a *switching Kalman filter* model. A mixture of UKFs is formed by *soft switching*, wherein we take a weighted combination of all the component models. Here, the weight is $Pr(S_t = i \mid y_{1:T})$. The assumption is that S_t has Markovian dynamics. At time t, the belief state $Pr(X_t \mid y_{1:t})$ is a mixture of K^t Gaussian distributions. In order to deal with the problem of exponential growth of belief states with time, the algorithm collapses states, thereby approximating the mixture with a single Gaussian distribution[16]. Now we have $\Theta = (\theta_1 ... \theta_K, \pi_1 ... \pi_K, Z)$, where Z is the transition matrix and the component model parameters are $\theta_k = (A, B, Q, R, \mu_0, \Sigma_0)$. In order to obtain the maximum likelihood estimates of the parameters, we perform EM. The equations for inference are as below[16]:

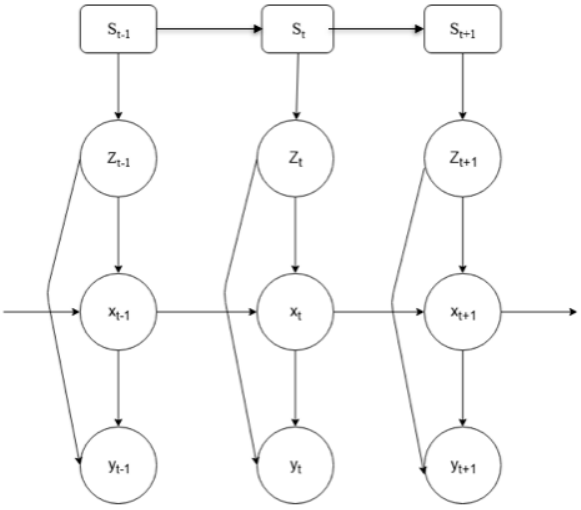


Figure 3.2: Graphical representation of the Mixture UKF model.

- Forward pass:

$$(x_{t|t}^{i(j)}, V_{t|t}^{i(j)}, V_{t,t-1|t}^{i(j)}, L_t^{i(j)}) = UKF(x_{t-1|t-1}^{i}, V_{t-1|t-1}^{i}, y_t, A_j, B_j, Q_j, R_j)$$

$$M_{t-1,t|t}(i,j) = Pr(S_{t-1} = i, S_t = j \mid y_{1:t}) = \frac{L_t(i,j)Z(i,j)M_{t-1|t-1}(i)}{\Sigma_i \Sigma_j L_t(i,j)Z(i,j)M_{t-1|t-1}(i)}$$

$$M_{t|t}(j) = \Sigma_i M_{t-1,t|t}(i,j)$$

$$W^{i|j} = Pr(S_{t-1} = i \mid S_t = j, y_{1:t}) = \frac{M_{t-1,t|t}(i,j)}{M_{t|t}(j)}$$

$$(x_{t|t}^{j}, V_{t|t}^{j}) = Collapse(x_{t|t}^{i(j)}, V_{t|t}^{i(j)}, W_t^{i|j})$$

- Backward pass:

$$(x_{t|T}^{j(k)}, V_{t|T}^{j(k)}, V_{t+1,t|T}^{j(k)}) = RTS(x_{t+1|T}^{k}, V_{t+1|T}^{k}, x_{t|t}^{j}, V_{t|t}^{j}, V_{t+1|t+1}^{k}, V_{t+1|t+1}^{j(k)}, A_k, Q_k)$$

$$U_t^{j|k} = Pr(S_t = j \mid S_{t+1} = k, y_{1:T}) \approx \frac{M_{t|t}(j)Z(j,k)}{\Sigma_{j'} M_{t|t}(j')Z(j',k)}$$

$$M_{t,t+1|T}(j,k) = U_t^{j|k} M_{t+1|T}(k)$$

$$M_{t|T}(j) = \Sigma_k M_{t,t+1|T}(j,k)$$

$$W_t^{k|j} = Pr(S_{t+1} = k \mid S_t = j, y_{1:T}) = \frac{M_{t,t+1|T}(j,k)}{M_{t|T}(j)}$$

$$(x_{t|T}^j, V_{t|T}^j) = Collapse(x_{t|T}^{j(k)}, V_{t|T}^{j(k)}, W_t^{k|j})$$

$$(x_{t|T}, V_{t|T}) = Collapse(x_{t|T}^j, V_{t|T}^j, M_{t|T}(j))$$

$$x_{t+1|T}^{j(k)} \approx x_{t+1|T}^k$$

$$V_{t+1,t|T}^k = CrossCollapse(x_{t+1|T}^{j(k)}, x_{t|T}^{j(k)}, V_{t+1|T}^{j(k)}, U_t^{j|k})$$

$$x_{t|T}^{()k} = \Sigma_j x_{t|T}^{(j)k} U_t^{j|k}$$

$$V_{t+1,t|T} = CrossCollapse(x_{t+1|T}^k, x_{t|T}^{()k}, V_{t+1,t|T}^k, M_{t+1|T}(k))$$

In the above equations, Collapse and CrossCollapse functions refer to the moment matching technique. The collapsing technique is proven to be effective, since the moments of the collapsed Gaussian are closest to the original mixture distribution. The EM algorithm for estimation is given below:

- Choose prior π^0 in the probability simplex and randomly initialize Z and the parameters, $\theta_k^0 = (A_k, \Pi_k)$, where $k = 1...K$

- Iterate until convergence:

  - For each sequence i, where i = 1 ... N:

  - *Expectation step*:

    * Compute the *responsibilities/ weights* using current parameter values from the inference equations.

$$W_t^j = Pr(S_t = j \mid y_{1:T}, \Theta) = M_{t|T}(j)$$

    * Revisiting the Expectation step in section 2.12, multiply the current responsibilities with each the expected sufficient statistic.

  - *Maximization step*:

    * In addition to the equations below, the other parameters are obtained via the Maximization step (see section 2.12)

$$\pi_k = \frac{\Sigma_{k=1}^K W_1^k}{N}$$
$$\hat{Z_{ij}} = \frac{\sum_{n=1}^N \sum_{t=2}^T Pr(S_{t-1} = i, S_t = j \mid y_{1:T})}{\sum_{n=1}^N \sum_{t=1}^{T-1} W_t^k}$$

In both techniques mentioned above, initialization of cluster centers is important. The centers are initialized with parameters that are estimated with random sub-samples.

Chapter 4

Results

4.0.1 Datasets

As described in Chapter 1, the datasets are generated from the games tapback, recall and recollect. These games already adapt themselves to the subject's performance. The recall and tapback datasets use the *classic staircase algorithm*, wherein a subject levels up by 3 hits (see section 1.3) and levels down by 2 false alarms. The recollect dataset is more liberal in placing the user in unnatural challenges by including multiple training algorithms.

The other algorithms are as follows:

- *staircase moderate* (a1) is the algorithm wherein a level-up corresponds to 3 hits and a level-down corresponds to 3 false alarms.
- *staircase difficult* (a2) is the algorithm wherein a level-up corresponds to 6 hits and a level-down corresponds to 2 false alarms.

- *mini-block moderate* (a3) is the algorithm wherein a level-up corresponds to more than 5 hits and a level-down corresponds to more than 3 false alarms.

- *mini-block difficult* (a4) is defined as the algorithm wherein a level-up corresponds to more than 3 hits and a level-down is 0 false alarms.

- *mini-block reset* (a5) is the algorithm that starts each session at n-back 2, and requires more than 5 hits to level-up.

	Tapback	Recall	Recollect
Total number of subjects	36	48	177
Maximum number of sessions per day	1	1	2
Average number of sessions across all subjects	20	20	11
Average number of blocks across all subjects	190	440	435
Highest n-back presented	9	7	12
Lowest n-back presented	1	1	1

Table 4.1: Datasets Statistics

4.0.2 Model Performance Comparison

Errors are reported in terms of root mean squared error (RMSE), capturing positional accuracy. The RMSE for a single sequence is expressed as follows:

$$RMSE = \sqrt[2]{\frac{1}{T}\Sigma_{t=1}^{T}(y_t - \hat{y}_t)^2}$$

The total RMSE computed across all test sequences is the metric for model performance. In our first experiment, we fit the models on the recall dataset, and create the test-pool with the tapback dataset. The hyper-parameters picked are $\rho = 1.6, c = 0.1$ and $d = 1$. In our

	HMM	UKF
Experiment 1	9%	17%
Experiment 2	16%	31%

Table 4.2: RMSE of the models

second experiment, we fit the models and evaluate performance on the recollect dataset, by creating a 80:20 split. Table 4.2 shows the performance of the models on both experiments.

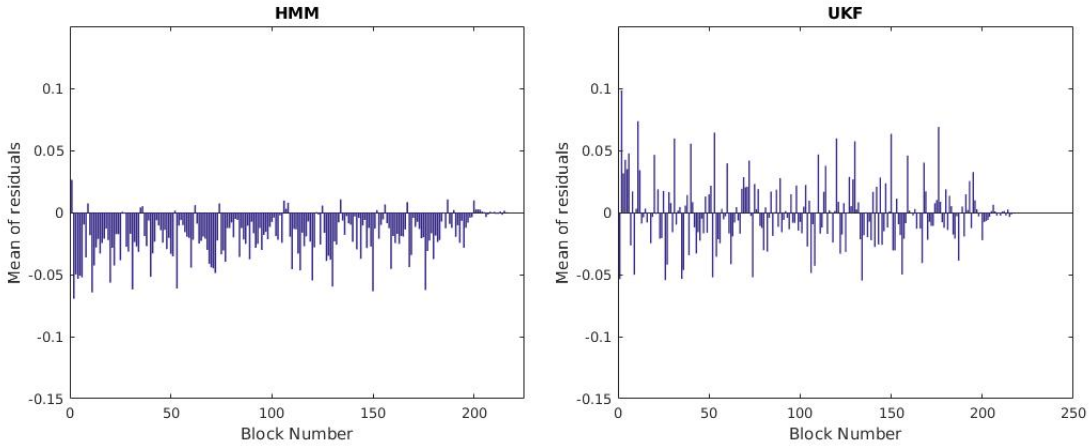


Figure 4.1: The mean of error residuals across all test-subjects in the tapback dataset obtained from the HMM (left) and the UKF model (right).

We pick specific subjects from the datasets for comparing the models' performances. Subject *SP436* from the tapback dataset plays lengthy trials of the *staircase algorithm* in a progressive format. Subject *SP451* from the tapback dataset plays the *staircase algorithm* for a session each day and plays in a progressive format until he or she reaches the n-level 8. The fluctuations between n-levels 8 and 9 are captured by both models. Subject

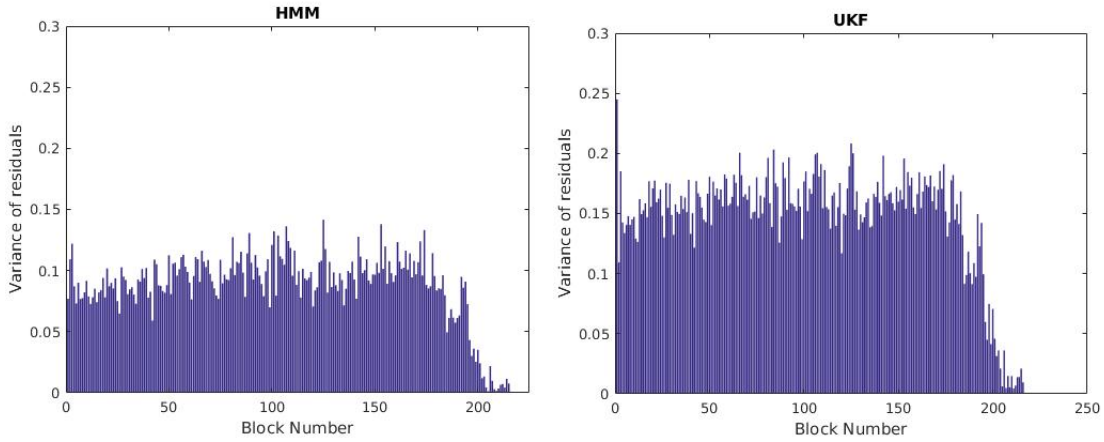


Figure 4.2: The variance of error residuals across all test-subjects in the Tapback dataset obtained from the HMM (left) and the UKF model (right).

RLB135 from the recollect dataset plays 2 sessions per day, and trains on the *mini-block reset*. Subject *RLB153* plays 2 sessions per day, and trains on the *mini-block moderate* algorithm.

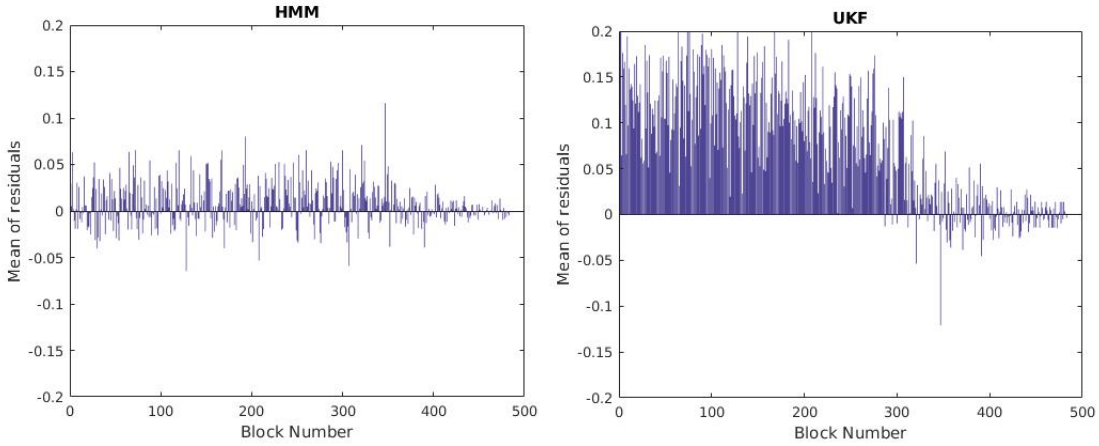


Figure 4.3: The mean of error residuals across all test-subjects in the Recollect dataset obtained from the HMM (left) and the UKF model (right).

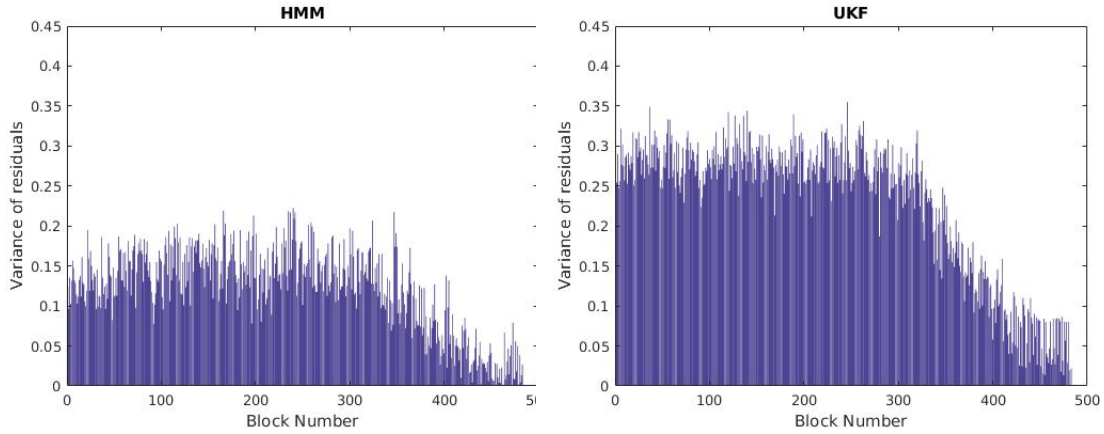


Figure 4.4: The variance of error residuals across all test-subjects in the Recollect dataset obtained from the HMM (left) and the UKF model (right).

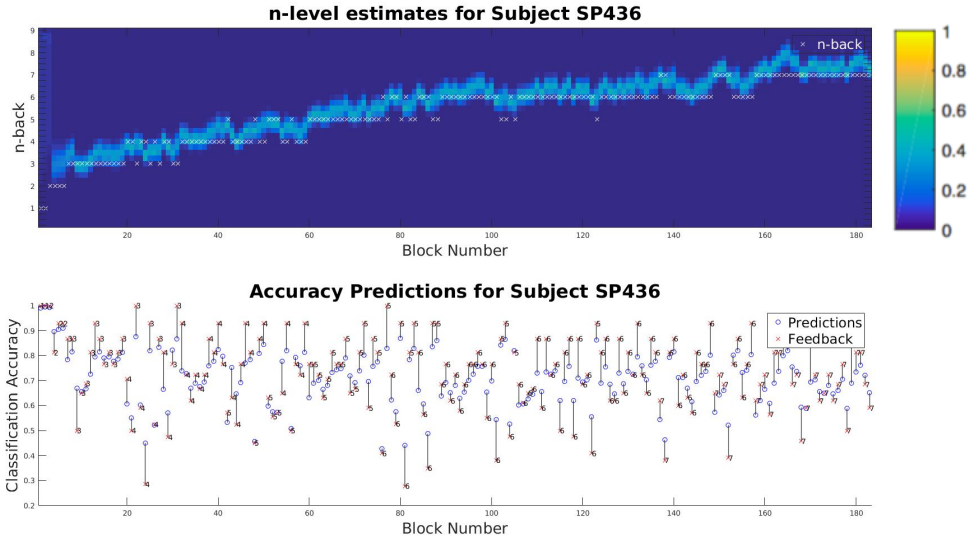


Figure 4.5: (Top): estimates of *SP436* n-level trajectory using the HMM. (Bottom): prediction vs observed classification accuracy at each time-step.

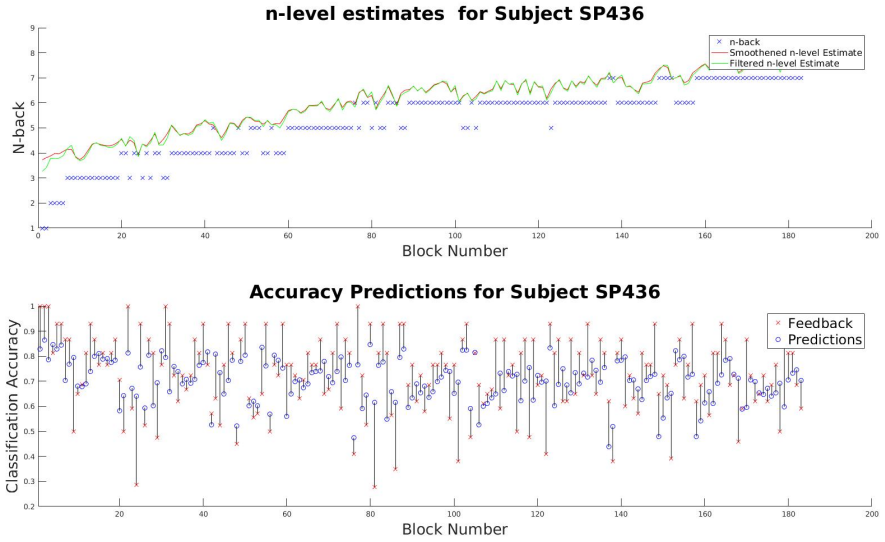


Figure 4.6: (Top): the noisy estimates of *SP436* n-level trajectory using the UKF. (Bottom): prediction vs observed classification accuracy at each time-step.

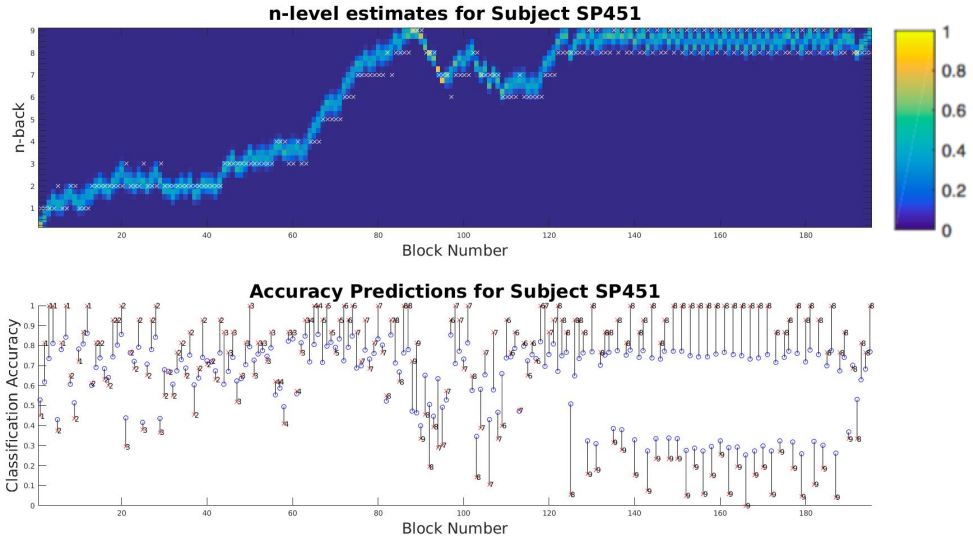


Figure 4.7: (Top): estimates of *SP451* n-level trajectory using the HMM. (Bottom): prediction vs observed classification accuracy at each time-step.

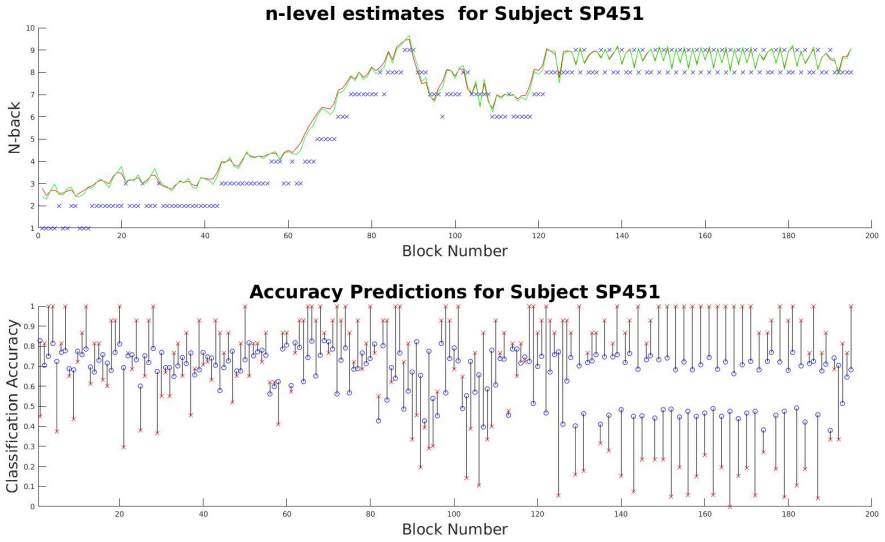


Figure 4.8: (Top): the noisy estimates of *SP451* n-level trajectory using the UKF. (Bottom): prediction vs observed classification accuracy at each time-step.

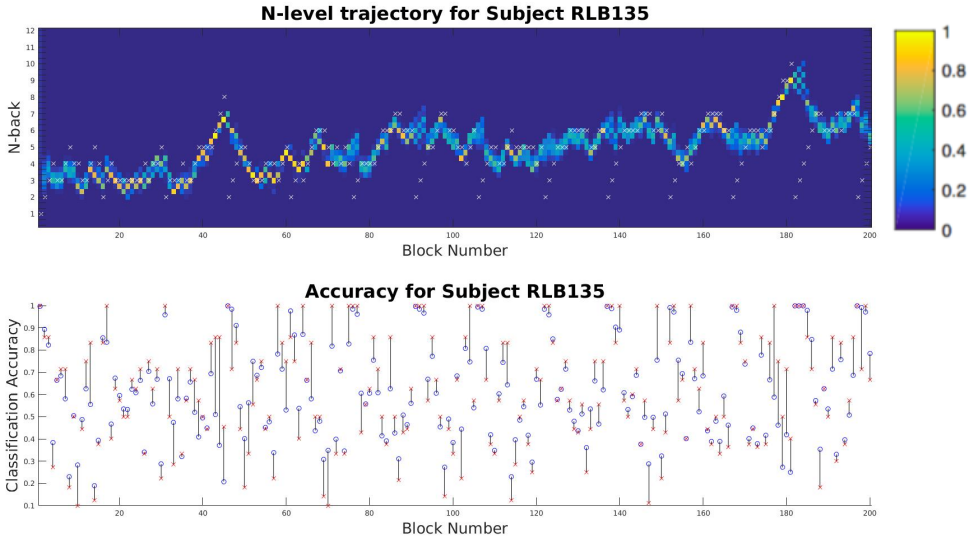


Figure 4.9: (Top): estimates of *RLB135* n-level trajectory using the HMM. (Bottom): prediction vs observed classification accuracy at each time-step.

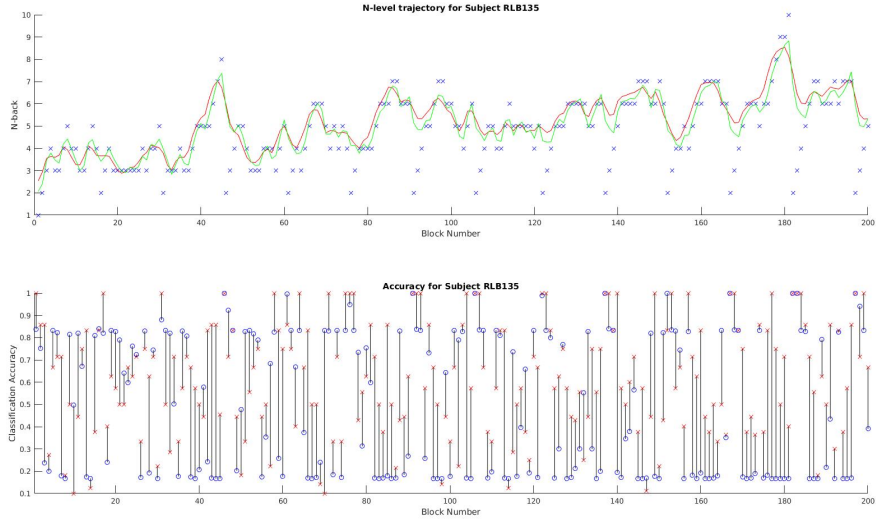


Figure 4.10: (Top): the noisy estimates of *RLB135* n-level trajectory using the UKF. (Bottom): prediction vs observed classification accuracy at each time-step.

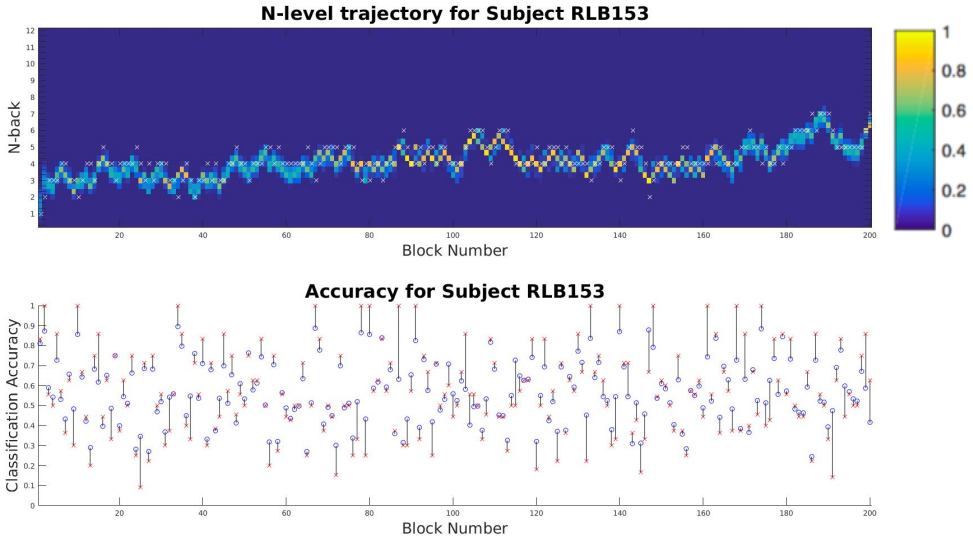


Figure 4.11: (Top): estimates of *RLB153* n-level trajectory using the HMM. (Bottom): prediction vs observed classification accuracy at each time-step.

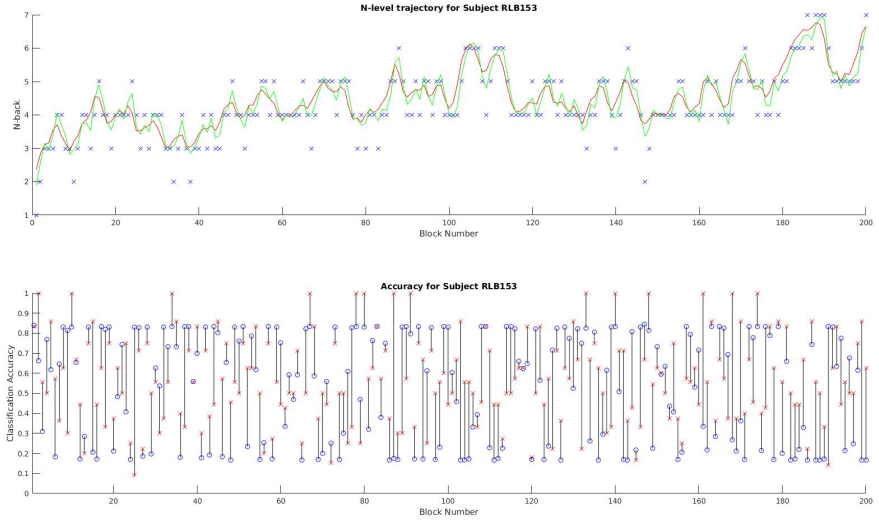


Figure 4.12: (Top): the noisy estimates of *RLB153* n-level trajectory using the UKF. (Bottom): prediction vs observed classification accuracy at each time-step.

4.0.3 Time Series Clustering Results

We notice that there is no performance improvement nor degradation with the mixture models. So, we make use of these models to cluster subjects. We choose K=3 in all clustering experiments. A potential extension would be to learn K simultaneously along with the model. We split the recollect dataset based on the year of training. The difference between the two splits is as follows:

	Recollect-2016	Recollect-2017
Number of sessions per day	2	1
Number of training algorithms	6	1

By making this distinction, we can hope to get an insight into the effect of training algorithms on the clusters. Table 4.3 shows the characteristics of the clusters formed. Cluster 2 of recollect-2016 are subjects that train on more difficult algorithms when compared to cluster 1. Majority of the subjects in cluster 2 tend to reach upto a high n-back of 10.

The recollect dataset fixes the number of training days to 11. In order to analyze subjects'

	Cluster 1	Cluster 2	Cluster 3
Recollect dataset-2017			
Total number of subjects	93	6	0
Frequent maximum n-backs reached	5-9	5	–
Recollect dataset -2016			
Total number of subjects	25	53	0
Frequent maximum n-backs reached	5-8	6, 10	–
Frequent algorithms played	a2, a5	a4, a3, a2, a5	–

Table 4.3: Clustering results

performance trends on each training day, we look at the maximum n-back reached on those days. This analysis is illustrated by the histogram plots in figures 4.13 and 4.15. Figures

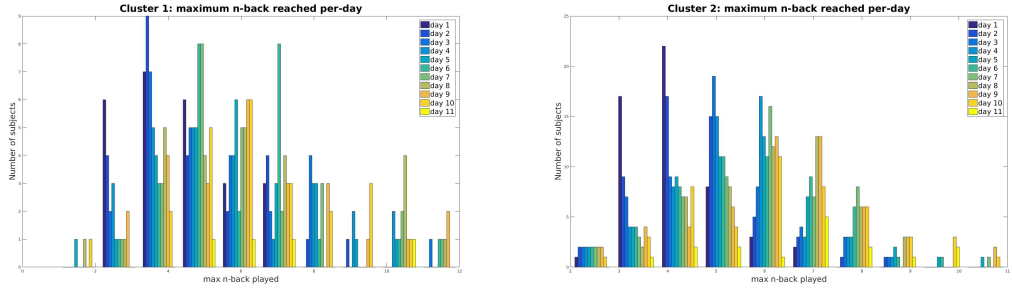


Figure 4.13: The number of subjects with maximum n-backs performed per day across cluster-1 (left) and cluster-2 subjects (right) for *Recollect 2016*.

4.14 and 4.16 compare the rate of change of n-back in terms of *block* (episode) length, measured in terms of the number of *events* or *trials*.

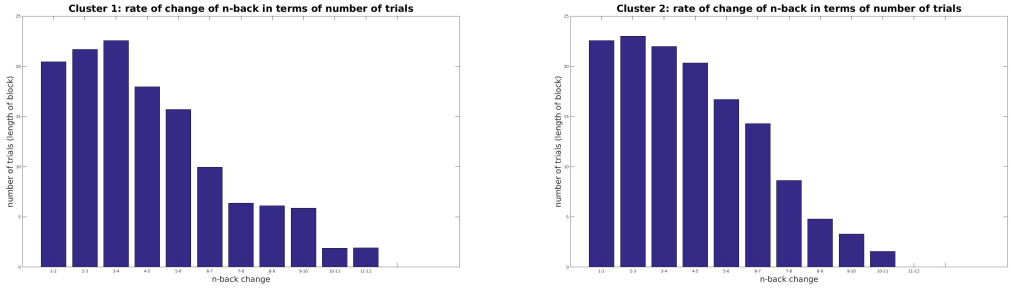


Figure 4.14: Rate of change of n-back in terms of the block length across cluster-1 (left) and cluster-2 subjects (right) for *Recollect 2016*.

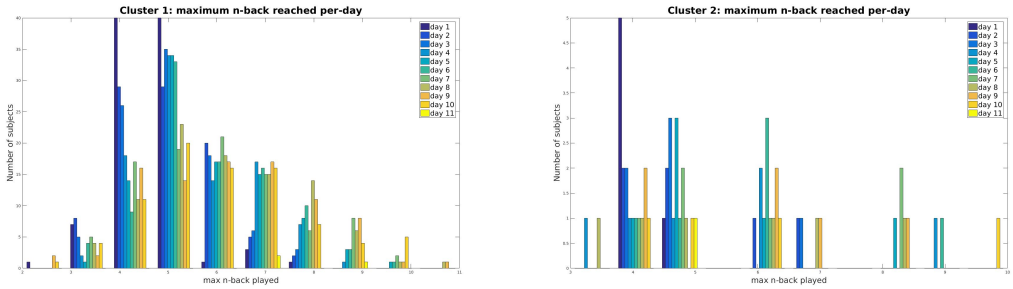


Figure 4.15: The number of subjects with maximum n-backs performed per day across cluster-1 (left) and cluster-2 subjects (right) for *Recollect 2017*.

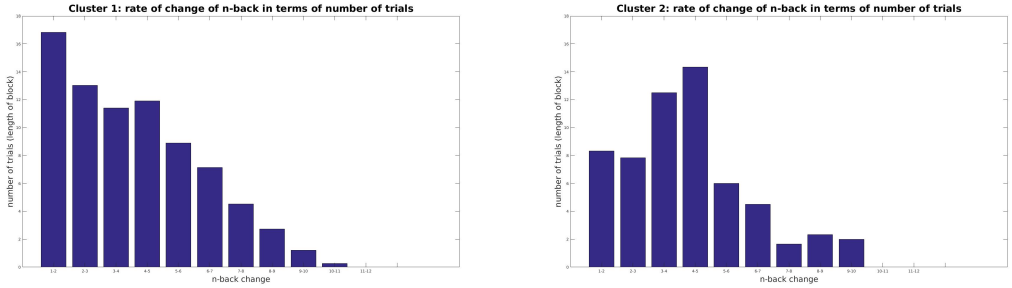


Figure 4.16: Rate of change of n-back in terms of the block length across cluster-1 (left) and cluster-2 subjects (right) for *Recollect 2017*.

Chapter 5

Conclusions

In this section, our focus is to analyze the results of our experiments and discuss potential extensions. Studies conducted at UCR and elsewhere have shown that the n-back increases working memory in trainees [11]. In our approach, we employ a state space framework with a control algorithm that helps us decide whether a subject is ready for a challenge. We develop a universal model for all subjects, that eventually adapts to a specific subject. We notice that the two models (HMM and UKF model) are quite powerful in the initial experiment which consists of only a single type of training algorithm. Performance degradation in the second experiment can be explained because of multiple training algorithms present in the dataset. By clustering subjects based on training algorithms', we can hope to get more accurate predictions. The HMM consistently performs better than the UKF model, due to it's reduced capacity and also due to the linear dynamics system of the UKF. A promising route to explore, would be to consider a non-linear dynamics system in the UKF model. The datasets that we currently use include only subjects performing the

color task type. In order to account for subjects who perform multiple task types, a possible improvement is to convert the uni-dimensional latent state space into a multi-dimensional state space.

Clustering the subjects according to performance does not improve performance. This could be due to the increased number of parameters. We were limited to using simplified models because of the concern of data. For future data-streams, deep-learning techniques such as a LSTM framework might prove to be effective. In our experiments, we even try to fit the LSTM model onto our current datasets. However, the model is not powerful enough in making accurate predictions because of limited amounts of data and too many parameters.

Bibliography

[1] Recall the game — https://itunes.apple.com/us/app/recall-the-game/id890271623?mt=8. iTunes Connect.

[2] Recollect the game — https://itunes.apple.com/us/app/recollect-the-game/id1028598975?mt=8. iTunes Connect.

[3] Working memory — simplypsychology.org, 2018. [Online; accessed 11-April-2018].

[4] Frank B. Baker. *The Basics of Item Response Theory*. Heinemann, 1985.

[5] Tamer Basar. *A New Approach to Linear Filtering and Prediction Problems*, pages 532–. Wiley-IEEE Press, 2001.

[6] A. BIRNBAUM. Some latent trait models and their use in inferring an examinee's ability. *Statistical Theories of Mental Test Scores*, 1968.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[8] Phil Blunsom. Hidden Markov models, 2004.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[10] Concept Derivation, , and Maria Isabel Ribeiro. Kalman and extended Kalman filters:.

[11] Jenni Deveau, Susanne Jaeggi, Victor Zordan, Calvin Phung, and Aaron Seitz. How to build better memory training games. 8:243, 01 2014.

[12] Jacob Goldberger. Unsupervised clustering with E.M. Report.

[13] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering: Theory and Practice with MATLAB*. Wiley-IEEE Press, 4th edition, 2014.

[14] S.S. Haykin. *Kalman Filtering And Neural Networks*. Wiley Online Library, 2001.

[15] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. pages 182–193, 1997.

[16] Kevin P. Murphy. Switching Kalman filters. Technical report, 1998.

[17] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989.

[18] Padhraic Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, pages 648–654. MIT Press, 1997.

[19] Eric A. Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. pages 153–158, 2000.

[20] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[21] Lloyd R. Welch. Hidden Markov models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4), December 2003.

[22] Wikipedia contributors. Item response theory — Wikipedia, the free encyclopedia, 2018. [Online].

[23] Wikipedia contributors. Working memory — Wikipedia, the free encyclopedia, 2018. [Online; accessed 11-April-2018].