# Machine Learning Techniques to Predict Grades

Candidate Number: 1018098

April 22, 2017

# 1 Introduction

Within university education, it is important to ensure that every student gets the attention and the materials that he or she requires for successfully finishing their degree. In order to be able to make early interventions in case a given student risks failing a class in a university degree, it is important to have a prediction of the students grades, as accurately and as early as possible. Over the course of the past years, multiple authors have explored possibilities to make grade predictions for students using techniques from machine learning (ML). This work will present some of those approaches and develop a new approach that uses *Hidden Markov Models* (HMMs) to predict grades in a university degree. Geographically, this work will focus on predicting grades for college students in the United States. The terms *college* and *university* will be used interchangeably and shall simply refer to the educational institution where students can receive undergraduate degrees in the USA.

**Predicting University Grades** Every student is unique, and there is no one model that is able to capture every student's performance. However, it is possible to build models that can make useful predictions for a large proportion of the student body. The first question in building grade prediction models is what characteristics, or *features*, of students one should use for the prediction. There are two kinds of features that are relevant in this context: *static features* are student characteristics that remain unchanged in time, such as the high school GPA or results in standardized tests such as the SAT. The SAT is a test that students in the USA take before entering college. It partly determines their chances of admission. In comparison to those unchanged characteristics, there are *dynamic features* of students, which in this context are given by the grades they receive over the course of their degree. The grades themselves remain unchanged once received, however, since the amount of this kind of data is expanding as time passes by, they are referred to as dynamic. Both kinds of features can be used to predict university grades, see the literature review in Section 2. In building grade prediction algorithms, it is important to incorporate a student's performance over time to take account of the development over the course of their degree.

**Hidden Markov Models (HMMs)** HMMs are *generative models* that describe stochastic processes, see the introduction to machine learning in Subsection A.1 in the appendix or [10]. Such models describe the evolution of a process in time $t$ as a sequence of discrete *states* $S_i$. The states themselves cannot be observed, but they send out *observation symbols* $v_k$ according to a probability distribution which can be used to infer the hidden sequence of states. In other words, there are two stochastic phenomena that are an integral part of HMMs: the process evolves from state to state according to a set of transition probabilities, and it sends out observation symbols according to a state specific probability distribution. HMMs are a generalised version of the simpler *Markov Models*, in which one assumes that the sequence of states itself is observable [12]. HMMs are widely used in practice, application areas include finance, see [9], speech recognition, see [7] and gene finding, see [11], to name just a few. For an excellent introduction to the theory of HMMs, see [12].

**Idea of this Work** The idea of this work is to use an HMM for grade prediction. The underlying assumption is that the grades a student receives do not, in a deterministic sense, specify the educational "state" that he or she is in, but should rather be taken as "grade symbols" that stem from this state and are sent out according to a probability distribution. In other words, if a given student receives a bad grade, it does not necessarily mean that he did not understand the course material and that he will do equally bad in the following exams, it might just be that he or she had a bad day. An HMM takes account of the statistical nature that university assessments have by treating the underlying educational states as *latent variables* that cannot be observed directly.

## 2 Overview of Previous Work

Grade prediction is a classical application of machine learning techniques and there are numerous authors who have published on the topic. This section will start out by investigating the simplest predictors of university performance: SAT scores (by subject) and the GPA achieved in high school. From here, it will move on to explore techniques of predicting grades in university based on previous university performance.

**Predictive Power of High School GPA and SAT Scores** A basic approach towards predicting a given student's future performance in university is to consider *static features* such as the GPA from high school, the overall SAT score and the SAT scores by subject. The data set that will be used in Section 5 to train and test an HMM includes information about students' high school GPA and SAT scores by subject, therefore, it is interesting to explore whether those features possess any predictive power. This question has been investigated for instance by [2]. In their paper, the authors explore to what extent they can predict a student's college performance, based on static features only. Their data was gathered at the University of South Carolina in a course on the Principles of Economics and is based on 521 students. Besides SAT and GPA scores from high school, they consider the features race and sex. Most important for this work is their finding that the SAT score can be an indicator for success in college [2]. The features race and sex are irrelevant for our case because they are not included in the given data set.

**Predicting Grades in Higher Education using Machine Learning** Using only static features to predict college performance is unsatisfactory because it neglects a student's development at college. Various authors have therefore applied different techniques from machine learning in order to predict grades in college, based on both static and dynamic features. A setting in which the availability of data is especially high is online learning, which is becoming more and more popular. In particular Massive Open Online Courses (MOOCs) are a rich source of educational data and a have therefore been investigated by multiple authors. In [8], the authors use data from an online course at the Hellenic Open University together with students' static features in order to predict the final outcome in one single class. The dynamic features are assignments that the students hand in before they sit the final exam. The authors compare various ML techniques for doing so, including naive Bayes, neural networks and support vector machines (SVPs). For an introduction to those models, refer to [10]. They find that for their setting, naive Bayes achieved the best overall results. Another interesting approach has been investigated by Xu, Moon and van der Schaar (2017) in [14], where they use an ensemble learning approach to generate predictions for student grades in a university degree program, rather than in an MOOC setting as before. They predict using static features as well as grades obtained up to the point of prediction. In contrast to [8], they predict across courses, so they do not try to estimate the final grade for a course given early assessments in the same course, but they rather explore the correlation between different courses and exploit this structure when making predictions for performance in future assessments. This situation is more similar to the set up in this work where the focus will lie on predicting grades across courses in a university degree program.

**HMMs in Grade Prediction** Despite the widespread usage of HMMs in many different areas of science and technology, they have not been used before to predict grades, to the best of the author's knowledge. This work will therefore establish to what extent this class of models is suitable for a grade prediction setting.

# 3 Theory of HMMs

This brief introduction to the theory of HMMs will follow in notation the work by Rabiner in [12]. The focus will lie on the application and the implementation of HMMs. A basic understanding of the concepts and terms central to ML is necessary to understand this section. For a brief introduction, see Subsection A.1 in the appendix.

**Hidden Markov Models** HMMs are generative, supervised, *latent variable* models that can be used in either regression or classification. Latent variable models describe phenomena in which some of the variables remain unobserved. In the case of HMMs, the latent variables are the hidden states $S_i$. Before actually explaining the theory of HMMs, it is enlightening to start out with the easier to understand discrete Markov Models. We will focus on the discrete case for both Markov Models as well as Hidden Markov Models because the model we will employ later to make grade predictions will be discrete in the state space as well as the time domain and the output symbols.

## 3.1 Discrete Markov Models

Consider a system that, at each instance of time, can find itself in one of the $N_{\mathcal{S}}$ states of the state space $\mathcal{S} := \{S_1, S_2, ..., S_{N_{\mathcal{S}}}\}$, where $S_i \in \mathbb{Z}\ \forall i \in \{1, ..., N_{\mathcal{S}}\}$ since we are assuming a discrete state space. We will assume that each change of state occurs at a discrete time and we will denote those time points with $t = 1, 2, ..., T$, where $T$ is the final time point in the sequence. Further denote by $q_t \in \mathcal{S}$ the state $S_i$ that the system was in at time $t$. The case where both the time points as well as the states are discrete is referred to as a *discrete Markov chain* [12].

**Markov Property** The probability of being in a certain state at a certain time will in general depend on the current state as well as the full history of all previous states, i.e. a probabilistic description of a state sequence would be

$$\mathbb{P}(q_t = S_j | q_{t-1} = S_i, ..., q_1 = S_k)\,. \tag{1}$$

Note that this does not depend on the time of the transition but only on the states that are involved. This assumption is referred to as a *time-homogeneous*, or *stationary* Markov Model. All of the models discussed in the following fulfill this property. In the case of a Markov Model, an additional assumption is made that the current state $q_t$ only depends on a certain number of previous states. This number determines the *order* of the Markov model. Most common are first order models, in which the current state $q_t$ only depends on the previous state $q_{t-1}$. This property is referred to as the *Markov property* and may be written as

$$\mathbb{P}(q_t = S_j | q_{t-1} = S_i, ..., q_1 = S_k) = \mathbb{P}(q_t = S_j | q_{t-1} = S_i) \, . \tag{2}$$

In the following discussion, we will assume a first order Markov Model.

**Transition Probabilities and Initial State Probabilities** We may now denote the discrete transition probabilities by the variables

$$a_{i,j} := \mathbb{P}(q_t = S_j | q_{t-1} = S_i) \, , \tag{3}$$

where both $i, j \in \{1, ..., N_{\mathcal{S}}\}$, $a_{i,j} \geq 0 \; \forall i, j \in \{1, ..., N_{\mathcal{S}}\}$ and $\sum_{j=1}^{N_{\mathcal{S}}} a_{i,j} = 1 \; \forall i \in \{1, ..., N_{\mathcal{S}}\}$, to make the quantities $a_{i,j}$ obey standard probabilistic requirements. We will define a *transition matrix* $(A)_{i,j} := a_{i,j}, \forall i, j \in \{1, ..., N_{\mathcal{S}}\}$, where $A \in \mathbb{R}^{N_{\mathcal{S}} \times N_{\mathcal{S}}}$. To fully describe the model, we also need initial state probabilities which we will define as follows:

$$\pi_i := \mathbb{P}(q_1 = S_i) \, . \tag{4}$$

Again, it holds $\pi_i \geq 0 \; \forall i \in \{1, ..., N_{\mathcal{S}}\}$ and $\sum_{i=1}^{N_{\mathcal{S}}} \pi_i = 1$. The quantity $\pi_i$ therefore determines the probability of initially being in state $S_i$. We will bundle the initial state probabilities in the vector $\pi := (\pi_1, ..., \pi_{N_{\mathcal{S}}})^\intercal \in \mathbb{R}^{N_{\mathcal{S}}}$. For an example of a three state discrete Markov chain, see Figure 2 in the appendix.

**Determining the Model Parameters for a Discrete Markov Chain** In a Markov chain, the states themselves correspond to physical quantities that can be measured, they are *observable*. One can imagine for example constructing a Markov chain to describe the weather where each state corresponds to either sun, rain or clouds [12]. In practice, given a model and some data, one would often like to estimate the corresponding model parameters that best describe the observed data. A technique to do so shall established here following [10]. In the case of a discrete Markov chain, *maximum likelihood estimators* (MLE) for the model parameters

$\theta := (A, \pi)$ can be calculated analytically. This is shown in detail in Subsection A.3 in the appendix. The task of estimating model parameters becomes much harder in the case of HMMs, as will be explained in following paragraphs.

## 3.2 Discrete Hidden Markov Models

The previous exploration of discrete Markov chains allows us to extend this concept in a straightforward manner to Hidden Markov Models. We will again focus on the case where both the state space $\mathcal{S}$ as well as time $t$ are discrete, thereby focusing on discrete time and state HMMs. The main distinction to a Markov chain is that in the case of an HMM, the states $S_i \in \mathcal{S}$ cannot be observed directly. However, the states send out *observation symbols* $v_k \in \mathcal{V}$, where $\mathcal{V}$ is the *observation space*. We will focus on the case in which $\mathcal{V}$ is finite and discrete and there are $N_\mathcal{V}$ distinct elements in $\mathcal{V}$. Each state can now send out observation symbols from $\mathcal{V}$ according to a state dependent *observation symbol probability distribution*. We define the probability of observing symbol $v_k$ in state $S_j$ as

$$b_j(k) = \mathbb{P}(O_t = v_k | q_t = S_j)\,, \tag{5}$$

where it holds again $\sum_{k=1}^{N_\mathcal{V}} b_j(k) = 1\ \forall j \in \{1, ..., N_\mathcal{S}\}$ and $b_j(k) \geq 0\ \forall j \in \{1, ..., N_\mathcal{S}\}, k \in \{1, ..., N_\mathcal{V}\}$. Here, $O_t$ defines the observation made at time $t$. For ease of notation, define the *observation matrix* $(B)_{j,k} = b_j(k)$, where $B \in \mathbb{R}^{N_\mathcal{S} \times N_\mathcal{V}}$. One can now summarise the model parameters of a discrete states, discrete time and discrete observation symbols HMM (short: discrete HMM) in $\theta = (A, B, \pi)$.

**The Three Basic Problems Associated with HMMs** The following exploration of HMMs is based on the work of Rabiner, see [12]. The aim of this introduction to the theory behind HMMs is to be able to understand how an HMM was used in later parts of this work to predict student's future performance. Basically, the tasks that had to be carried out were to estimate the model parameters and to predict future observations. However, it is helpful to think about three basic problems. Problem 1 lies in calculating the probability of observing a sequence of observations (Subsection 3.3), problem 2 consists of inferring the most likely sequence of hidden states given a sequence of observations (Subsection 3.4) and problem 3 is given by estimating model parameters given observations (Subsection 3.5). Possible ways to solve those problems are presented in the following. In the first two problems, we will assume that the model parameters $\theta$ are known.

## 3.3 Problem 1: Calculating the Observation Sequence Probability

First, we have to adapt the definition of the observation sequence given in the appendix for Markov chains, where the states themselves were the observables. Define $Q$ to be a sequence of length $T$ of hidden states, $Q = (q_1, q_2, ..., q_T)$. Define $O$ to be a sequence of length $T$ of observations, $O = (O_1, O_2, ..., O_T)$, where the observations are given by the discrete symbols, e.g. $O_t = v_k$ in case symbol $v_k \in \mathcal{V}$ was observed at time $t$. The challenge of problem (1) is to compute the probability of an observation sequence $O$ given model parameters $\theta$, $\mathbb{P}(O|\theta)$. If we additionally condition this probability on one particular hidden state sequence and assume that observations are independent, we can write

$$\mathbb{P}(O|Q,\theta) = \prod_{t=1}^{T} \mathbb{P}(O_t|q_t,\theta) = \prod_{t=1}^{T} b_{q_t}(O_t)\,. \tag{6}$$

The joint probability distribution of $O$ and $Q$ is given by

$$\mathbb{P}(O,Q|\theta) = \mathbb{P}(O|Q,\theta)\,\mathbb{P}(Q|\theta)\,, \tag{7}$$

where the first term in this equation is given by Equation (6). The second term may be written as

$$\mathbb{P}(Q|\theta) = \pi_{q_1}\, A(q_1,q_2) \cdot ... \cdot A(q_{T-1},q_T)\,. \tag{8}$$

We can now compute the probability of an observation sequence $O$ conditioned on the model parameters $\theta$ by summing the joint distribution in Equation (7) over all possible sequences of hidden states $Q$

$$\mathbb{P}(O|\theta) = \sum_Q \mathbb{P}(O|Q,\theta)\,\mathbb{P}(Q|\theta) \tag{9}$$

$$= \sum_Q \pi_{q_1}\, b_{q_1}\, A(q_1,q_2) \cdot ... \cdot A(q_{T-1},q_T)\, b_{q_T}(Q_T)\,. \tag{10}$$

**Complexity of this Computation** In principle, Equation (10) would allow for the solution of problem 1. However, in practice this is often impossible to do because evaluating $\mathbb{P}(O|\theta)$ according to Equation (10) requires $\mathcal{O}\left(T \cdot N_{\mathcal{S}}^T\right)$ calculations, e.g. for a model with $N_{\mathcal{S}} = 5$ states and $T = 100$ observations, this requires $\mathcal{O}\left(10^{72}\right)$ calculations [12]. In order to avoid this, we need to introduce the concept of the *forward algorithm.*

**The Forward Algorithm** We define the *forward messages* $\alpha_t(i)$ as

$$\alpha_t(i) := \mathbb{P}(O_1, ..., O_t, q_t = S_i|\theta) \,. \tag{11}$$

The forward message $\alpha_t(i)$ is the probability of observing the partial sequence $O_1, ..., O_t$ and being in state $S_i$ at time $t$, conditioned on the model parameters $\theta$. The forward algorithm is a procedure that allows for the recursive calculation of the forward messages. The initialisation is given by

$$\alpha_1(i) = \pi_i b_i(O_1) \,, \tag{12}$$

which is the probability of starting in state $S_i$ and sending out symbol $O_1$. Based on this, the following forward messages for all states $S_j$ can be calculated recursively as

$$\alpha_{t+1}(j) = \left( \sum_{i=1}^{N_\mathcal{S}} \alpha_t(i) a_{i,j} \right) b_j(O_{t+1}) \,, \tag{13}$$

which intuitively states that in order to find the probability of being in state $S_j$ at time $t+1$ with the partial observation sequence $O_1, ..., O_{t+1}$, we need to sum over the probabilities for observing $O_1, ..., O_t$ and being in state $S_i$ at time $t$ times the transition probability of going from state $S_i$ to state $S_j$, see Figure 3(a) in the appendix for an illustration. Once this is established, we just need to multiply by the probability of observing $O_{t+1}$ in state $S_j$. Once the forwards messages are calculated for all time points, we can calculate our desired observations sequence probability as

$$\mathbb{P}(O|\theta) = \sum_{i=1}^{N_\mathcal{S}} \alpha_T(i) \,. \tag{14}$$

**Complexity of the Forward Algorithm** Recall that the brute force approach of Equation (10) required $\mathcal{O}\left(T \cdot N_\mathcal{S}^T\right)$ calculations to compute the observation sequence probability. In contrast, the forward algorithm requires $\mathcal{O}\left(T \cdot N_\mathcal{S}^2\right)$ calculations to compute the observation sequence probability. For the example presented above with $N_\mathcal{S} = 5$ and a sequence length of $T = 100$, the forwards algorithm requires about 69 orders of magnitude less computational effort, see also [12], than using the direct definition in Equation (10).

**Backward Messages** For Subsection 3.5, we will also require quantities referred to as the *backward messages*. These are defined to be

$$\beta_t(i) = \mathbb{P}(O_{t+1}\, O_{t+2} \ldots O_T | q_t = S_i, \theta) \,. \tag{15}$$

They are the reversed time analogues of the forwards messages with the difference that we condition on being in state $S_i$ at time $t$. There is again a recursion formula for the backwards messages which reads

$$\beta_t(i) = \sum_{j=1}^{N_{\mathcal{S}}} a_{i,j} b_j(O_{t+1}) \beta_{t+1}(j) \,, \tag{16}$$

see Figure 3(b) in the appendix for an illustration.

## 3.4 Problem 2: Inferring the Most Likely Sequence of Hidden States

This problem will be central to predicting student's grades in later parts of this work. The first step towards predicting someone's future performance is to estimate what *educational state* he or she is currently in. This can be done by looking at the sequence of observation symbols (grades) and calculating what the most likely sequence of hidden states was that the student encountered. This will then allow us to estimate the current educational state that the student is in and hence to make a prediction about what state he or she will be in next. There are multiple definitions for the most likely sequence of hidden states. One possible definition is to construct a sequence from the the hidden states $S_i$ which are individually most likely. However, we will use an alternative definition that aims to find the single best state sequence $Q$, given the sequence of observations $O$, i.e. it aims at finding the sequence which *as a whole* is most likely. The corresponding algorithm is known as the *Viterbi algorithm* and shall be explained in the following paragraph. For more information on the algorithm, see [13].

**The Viterbi Algorithm** The Viterbi algorithm is based upon the quantity

$$\delta_t(i) := \max_{q_1, q_1, \ldots, q_{T-1}} \mathbb{P}(q_1\, q_2 \ldots q_t = S_i, O_1 O_2 \ldots O_t | \theta) \,, \tag{17}$$

which intuitively is the largest possible probability of arriving in state $S_i$ at time $t$ (most likely sequence of states that ends up in $S_i$ at time $t$) and observing the observation sequence $O_1, O_2, \ldots, O_t$. The trick is again that we can calculate those quantities recursively by the relation

$$\delta_{t+1}(j) = \left( \max_i \delta_t(i) a_{i,j} \right) b_j(O_{t+1}) \,. \tag{18}$$

In order to be able to find the most likely sequence of hidden states later on, we have to keep track of the index where the maximum was attained. We will store this index in the variable $\psi_t(j)$. The first step of the algorithm is to initialise the two variables $\delta$ and $\psi$ by $\delta_1(i) = \pi_i b_i(O_1)$ and $\psi_1(i) = 0$ for all $i \in \{1, ..., N_{\mathcal{S}}\}$. The recursion step for $\delta$ is then given by Equation (18) and the recursion step for $\psi$ is given by

$$\psi_t(j) = \text{argmax}_{1 \leq i \leq N_{\mathcal{S}}} \left(\delta_{t-1}(i)\, a_{i,j}\right) . \tag{19}$$

The most likely last hidden state in the sequence is then directly given via

$$\hat{q_T} = \text{argmax}_{1 \leq i \leq N_{\mathcal{S}}}\, \delta_T(i) \, . \tag{20}$$

For all time points $t < T$, the state on the most likely path can now be calculated via a technique called *backtracking*:

$$\hat{q_t} = \psi_{t+1}(\hat{q_{t+1}}) \, , \tag{21}$$

which means that we search through our array $\psi$ backwards in time to infer the most likely sequence of states that the system went through.

## 3.5 Problem 3: Estimating the Model Parameters

This problem is essential in any situation where one wishes to build a HMM to describe real data. In Subsections 3.3 and 3.4 we have assumed that the model parameters $\theta = (A, B, \pi)$ are known. In this subsection, we will discuss how $\theta$ can be estimated given a training set $\mathcal{D}$. We wish to find $\theta$ such that $\mathbb{P}(O|\theta)$ is maximised, where we focus on the case of just one sequence $O$, although the concept can easily be generalised to multiple sequence $O_i \in \mathcal{D}$. We will present a method known as the *Baum-Welch method*, which iteratively finds a local maximum of the probability $\mathbb{P}(O|\theta)$. The method is equivalent to the EM (expectation-maximisation) algorithm which was originally developed to maximise the likelihood function in the case of missing data points, see [3]. For this reason, the method is also referred to as the *Baum-Welch EM algorithm.*

**The Baum-Welch EM Algorithm** The basic quantity that the algorithm deals with is the *two-slice marginal* $\xi_t(i,j)$, defined as

$$\xi_t(i,j) := \mathbb{P}(q_t = S_i, q_{t+1} = S_j | O, \theta) \, , \tag{22}$$

which is the probability of being in state $S_i$ at time $t$ and transitioning to state $S_j$ at time $t+1$, conditioned on the observation sequence $O$ and the model parameters $\theta$. We can write the two slice marginal in terms of forwards and backwards messages as

$$\xi_t(i,j) \propto \alpha_t(i)\, a_{i,j}\, b_j(O_{t+1})\, \beta_{t+1}(j) \, , \tag{23}$$

which is illustrated in Figure 4 in the appendix. Correctly normalised, the two slice marginal takes the form

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{i,j}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N_\mathcal{S}} \sum_{j=1}^{N_\mathcal{S}} \alpha_t(i)\, a_{i,j}\, b_j(O_{t+1})\, \beta_{t+1}(j)}\,. \tag{24}$$

We will now define $\gamma_t(i)$ to be the probability of being in state $S_i$ at time $t$, conditioned on the observation sequence $O$ and the model parameters $\theta$. Note that we may write $\gamma_t(i)$ as a sum of the two slice marginals over the states that the systems transits into,

$$\gamma_t(i) = \sum_{j=1}^{N_\mathcal{S}} \xi_t(i,j)\,. \tag{25}$$

Further note that we may write the expected number of transitions from state $S_i$ into any other state as $\sum_{t=1}^{T-1} \gamma_t(i)$ and the expected number of transitions from state $S_i$ to state $S_j$ as $\sum_{t=1}^{T-1} \xi_t(i,j)$. We can use those expressions to iteratively estimate the model parameters $\theta$.

**Iterative Procedure to Estimate the Model Parameters** Before stating the formulae which we will use to iteratively estimate the model parameters, it is helpful to state their definitions in an intuitive sense. The following interpretation is taken from [12]:

$$\hat{\pi}_i = \text{expected number of times that the system starts out in state } S_i\,, \tag{26}$$

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from } S_i \text{ to any other state}}\,, \tag{27}$$

$$\hat{b}_j(k) = \frac{\text{expected number of times that symbol } v_k \text{ is send out from state } S_j}{\text{expected number of times the system is in state } S_j}\,. \tag{28}$$

With those interpretations in mind, and making use of the definitions stated above, one can conclude

$$\hat{\pi}_i = \gamma_1(i)\,, \tag{29}$$

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}\,, \tag{30}$$

$$\hat{b}_j(k) = \frac{\sum_{t=1,\,\text{s. t.}\, O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(i)}\,. \tag{31}$$

Equations (29) to (31) now enable an iterative estimation of the model parameters in the following manner:

1. initialise, $\theta^{(0)} = \theta_0$,
2. update to $\theta^{(n+1)}$ by substituting $\theta^{(n)}$ into the right hand side of Equations (29) to (31),
3. repeat step 2 until a convergence criterion is met.

In this procedure, $\theta_0$ is an initial guess for the model parameters and $n$ is a loop counter.

**Properties of the Baum-Welch EM Algorithm** The above procedure does not look like a typical EM algorithm, however, it can be shown that it is equivalent to maximising a quantity called *Baum's auxilary function* and that the procedure can in fact be written as an EM algorithm. See for example [10] for an equivalent formulation as an EM-algorithm. As mentioned before, the procedure will in general not converge to global optima but only to local ones. Using multiple, randomly chosen starting points for the algorithm can be a strategy to overcome this shortfall. The Viterbi algorithm and the Baum-Welch EM algorithm are the two algorithms which will be used to train the HMM model for grade prediction and to issue predictions. The following Section 4 will explain how the algorithms were employed.

# 4 Methods

This section will explain how the theoretical results established in the previous section were used in order to predict students future performance.

## 4.1 The Dataset

The dataset used to train and test the model was collected at the Mechanical and Aerospace Engineering department at UCLA and contained 1295 students that graduated between 2013 and 2015. The data received was anonymised. Available information included student's high school GPA, their SAT scores by subject, their length of stay and all of the grades they received during their stay at UCLA. The first three sets of information will be denoted the static features, whereas the grades will form the dynamic features.

**Static Features** The assumption was adapted that the students' static features will have an influence on their performance at university, as reported for example in [2]. This work focused on the influence of the two features high school GPA ($GPA_H$) and SAT score in mathematics ($SAT_M$). The range for $GPA_H$ was from 3.71 to 5 whereas the range for $SAT_M$ was from 560 to 800. In Figure 5 in the appendix, a scatter plot of the two variables is shown. Subsection 4.3 will explain how the static features were incorporated into the HMM.

**Dynamic Features** For each grade, a course code specified the corresponding course. The academic year at UCLA is split into four quarters, however, we only consider grades received during the first three of those since the course structure in the last quarter is different from the structure in the other quarters, which makes it harder to incorporate the information. For each grade, the quarter it was received in is known. Let $O_i$ denote the observation sequence of all grades for student $i$.

**Pre-Processing** In a first step, the students were removed whose length of stay at UCLA was shorter than three quarters, because it is very difficult to make predictions based on an HMM with so few data points. Additionally, students were removed from the data set if either their value for $GPA_H$ or $SAT_M$ was missing. The remaining number of students in the data set was 960.

**Training Set $\mathcal{D}$ and Testing Set $\mathcal{T}$** This dataset was now split into a training set $\mathcal{D}$, which contained 60% of the students (576 students) and a testing set $\mathcal{T}$, which contained 40% of the students (384 students). No validation set $\mathcal{W}$ was used, mostly because of time constraints. It would, however, be very interesting to include a validation set $\mathcal{W}$ in order to be able to learn the number of hidden states $N_{\mathcal{S}}$ and the number of observation symbols $N_{\mathcal{V}}$.

**Missing Data** In a further step, the student data was checked for missing grades and the sequences of grades $O_i$, were cut off at the point where a grade was missing, such that the remaining, shorter observation sequences contained at least one grade per term. Some of the new, shortened sequences were now shorter than three observations, so they were removed again. This yielded final student numbers of $N_{\mathcal{D}} = 571$ students in the training set $\mathcal{D}$ and $N_{\mathcal{T}} = 381$ students in the testing set $\mathcal{T}$. In Figure 6 in the appendix, histograms are shown of the length of stay of the students in quarters before and after this step of processing.

**Averaging Over Grades** The grade sequences $O_i$ in both sets were then modified such that there was exactly one grade available per quarter per student, i.e. it was averaged over all grades that student $i$ in $\mathcal{D}$ or $\mathcal{T}$ obtained in quarter $t$, where $t$ shall measure time in quarters since the respective student enrolled at UCLA, so $t$ is discrete. This is a very strong simplification since it treats all grades as being equally informative, moreover, it does not take care of the fact that a quarter in which ten grades were obtained is more informative then a quarter in which only one grade was obtained. This simplification should be removed in further works and the course correlation should be taken into account as well as the varying level of informativeness that different quarters possess, see Subsection 6.1 for a further discussion of the model simplifications.

## 4.2 Model Specifications

An HMM with $N_{\mathcal{S}} = 3$ discrete hidden states was constructed, where the hidden states were interpreted as *educational states* that represent a student's educational standing, where state $S_3$ was the most desirable state to be in while state $S_1$ was the least desirable to be in, $\mathcal{S} = \{S_1, S_2, S_3\}$. Time $t$ in the model was discretised and represented quarters since the enrolment of a given student. The output symbol $v_k$ was interpreted as being the average grade that a given student obtained in a given quarter. The grade spectrum at UCLA ranges from 0 to 4.0, where 0 is the worst and 4.0 is the best possible grade. The grade spectrum was discretised into $N_{\mathcal{V}}$ bins, where bin $v_1$ represented the lowest grades and bin $v_{N_{\mathcal{V}}}$ represented the highest grades. This was done because it would have been difficult to describe the finite range $[0, 4]$ with a Gaussian emission. It is not impossible to do so, but it requires some of the theory introduced in the previous section to be modified. Further work on this model should attempt to use a continuous output spectrum instead of a discrete, binned spectrum. In this work, the discrete grade output spectrum will be described by a *Multinoulli distribution*.

**The Hidden State Chain** The HMM constructed for the grade prediction problem is shown in Figure 2 in the appendix. It was assumed that self transitions (remaining in the current state) are allowed. However, it is not allowed to jump from the lowest state $S_1$ directly to the highest state $S_3$ and vice versa. For this transition to happen, state $S_2$ has to be visited in between. The unknowns of the model are the transition probabilities $a_{i,j}$ for both $i, j \in \{1, ..., N_{\mathcal{S}}\}$, the initial state probabilities $\pi_i$ for $i \in$

$\{1, ..., N_{\mathcal{S}}\}$ and the observation symbol probabilities, $b_j(k)$ for $j \in \{1, ..., N_{\mathcal{S}}\}$ and $k \in \{1, ..., N_{\mathcal{V}}\}$.

**Model Analogy** The basic model can be thought of as containing $N_{\mathcal{S}} = 3$ biased dice that have $N_{\mathcal{V}}$ sides each, where the probability of rolling side $v_k$ of dice $S_j$ is given by $b_j(k)$. The sides of the dice are equivalent of the grades and the different dice are equivalent to the educational states. There is a probabilistic process that decides which dice is going to be rolled next. The probability of using dice $S_j$, given that the last dice rolled was dice $S_i$, is given by $a_{i,j}$. The probability of dice $S_i$ being the first to be rolled in the experiment is given by $\pi_i$. This simple dice rolling experiment describes the procedure of obtaining a certain grade class $v_k$ in a given quarter $t$, where $t$ simply counts how often any dice has been rolled so far.

## 4.3 Training the HMM

For all computations, Matlab was used. In addition, the toolbox PMTK3 developed by Kevin Murphy and others was employed which provides a probabilistic modelling toolkit for Matlab [4]. The toolbox contains an efficient implementation of HMMs, including the Viterbi Algorithm, the Forwards-Backwards Algorithm, the Baum-Welch EM Algorithm and many more. It supports discrete output HMMs as well as HMMs with Gaussian or Student distributed emissions.

**Incorporating the Static Features** The training set $\mathcal{D}$ was split up into two groups $A$ and $B$: in group $A$ were the students with very high $GPA_H$ and $SAT_M$ while in group $B$ were the students with lower $GPA_H$ and $SAT_M$. The aim of this split was to be able to take into account the static features of the students by training two separate HMMs, one for group $A$ ($HMM_A$) and one for group $B$ ($HMM_B$). The split was carried out in the following manner: first, the two quantities $GPA_H$ and $SAT_M$ were made comparable by the following transformation:

$$\widetilde{GPA}_H = \frac{GPA_H - \mathbb{E}(GPA_H)}{\mathrm{SD}(GPA_H)}\,, \tag{32}$$

$$\widetilde{SAT}_M = \frac{SAT_M - \mathbb{E}(SAT_M)}{\mathrm{SD}(SAT_M)}\,, \tag{33}$$

where the average was used as an estimator for the expectation and SD stands for the *standard deviation* of the sample. In a further step, a score $S$ was defined as

$$S := \widetilde{GPA}_H + \widetilde{SAT}_M\,. \tag{34}$$

The median of $S$ over the set $\mathcal{D}$ was computed and students with a score $S$ greater than the median were assigned to group $A$ while students with a score $S$ smaller or equal to the median were assigned to group $B$. This yielded $|A| = 285$ students in group $A$ and $|B| = 286$ students in group $B$. In Figure 5 in the Appendix, the quantities $\widetilde{GPA}_H$, $\widetilde{SAT}_M$ and the score $S$ are plotted.

**Learning the Model Parameters** The HMM for group $A$ and the HMM for group $B$ were trained separately using the Baum-Welch EM algorithm, see Subsection 3.5. A prior was used for the transition matrix $A$ that ensured that no direct transitions from state $S_1$ to state $S_3$ were allowed. A prior was also used for the observation symbol emission distribution that set the probabilities of obtaining very high grades when being in the lowest state $S_1$ to zero. Equivalently, the probability of obtaining a very low grade when being in the best state $S_3$ was also set to zero. This was done to reduce the number of model parameters and to avoid *overfitting*.

## 4.4 Testing the HMM

The testing set $\mathcal{T}$ was split into groups $A$ and $B$ using the same procedure that was employed to split the training set $\mathcal{D}$. For students in group $A$, the model $HMM_A$ was used whereas for students in group $B$, the model $HMM_B$ was used. The two models differed only in their model parameters $\theta = (A, B, \pi)$. For each grade sequence $O_i \in \mathcal{T}$, grade prediction started at time $t_{min} > 1$. At least the first (average) grade needed to be known in order to be able to predict the second (average) grade. Grade prediction was carried out online, meaning that only grades up to the current time $t$ were used.

**Predicting the Grade Class** Given the first $t \geq t_{min}$ (average) grades of sequence $O_i$, the most likely sequence of hidden states $\hat{Q}$ was estimated using the Viterbi algorithm, see Subsection 3.4. This yielded an estimate of the current educational state $q_t \in \mathcal{S}$. Given this current state, a prediction was calculated for the next most likely state to be encountered, $q_{t+1} \in \mathcal{S}$. This can be performed easily by checking the estimated transition matrix $A$ for the biggest element in the row corresponding to state $q_t \in \mathcal{S}$. Given the estimate for the next most likely state $q_{t+1}$, a prediction was calculated for the most likely next observation symbol $O_{t+1} \in \mathcal{V}$. This can be done by searching the estimated emission matrix $B$ for the greatest element in the row corresponding to state $q_{t+1}$. This observation symbol was then used as a prediction for the next grade symbol to be observed. The predicted grade symbol was compared with the actual next grade symbol and both were saved in an array in order to establish the performance of the algorithm later. This procedure was carried out for every time point $t_{min} \leq t \leq T_i - 1$. In this manner, it was possible to evaluate how the performance of the algorithm changes as more and more grades become available.

## 5 Results

The model used had $N_{\mathcal{S}} = 3$ discrete hidden states and $N_{\mathcal{V}} = 6$ discrete observation symbols per state. As described in the previous sections, the model was quite a basic approach to implementing an HMM for grade prediction that does not take into account the differences between different courses etc. Therefore, the benchmarks that the model was tested against were chosen to be quite basic as well. The first benchmark $\mathcal{B}_1$ was random guessing based on a uniform probability for each grade class and the second benchmark $\mathcal{B}_2$ was to use always the last obtained grade as a prediction for the next grade. The results obtained by the HMM will be compared to the results obtained by using those two benchmark predictors $\mathcal{B}_1$ and $\mathcal{B}_2$.

**Estimated Model Parameters** Before presenting the actual prediction results, it is enlightening to take a look at the parameters that the model learned from the data. Recall that two separate models were trained, one for group $A$ and one for group $B$. The two separate models trained yielded the estimated model parameters shown in Tables 1, 2 and 3 in the appendix.

**Exemplary Model Trajectories** In order to get a feeling for what the model does, an exemplary model trajectory is shown in Figure 7 in the appendix. Shown are the first $T-1$ grades, the estimated sequence of hidden states, the next predicted hidden state, the next predicted grade, and the actual, realised next grade. Four more exemplary model trajectories are shown in Figure 8 in the appendix in order to illustrate where some of the weaknesses and strengths of the model lie.

**Accuracy and Average Misclassification Error** As a main result, the HMM was compared with the two benchmarks $\mathcal{B}_1$ and $\mathcal{B}_2$ in terms of their predictive accuracy and average misclassification error over the course of several predictions. The accuracy was defined as the number of correct predictions over the total number of predictions issued. The average classification error was defined as

$$e_t = \frac{1}{N_{sample}} \sum_{i=1}^{N_{sample}} (O_{t+1} - \hat{O}_{t+1}) \,, \tag{35}$$

where $N_{sample}$ was the number of students for which a grade symbol was available in quarter $t+1$, so that the predictions could be checked against the real outcome. As always, 0 is the observation (actual, realised) and $\hat{O}$ is the predicted observation (the grade symbol). The results are shown in Figure 1. The HMM behaves similarly to the method $\mathcal{B}_2$ in terms of both accuracy and average misclassification error. Both methods achieve an accuracy of around 60% as soon as more than five grade symbols become available and an average misclassification error of around half a class.

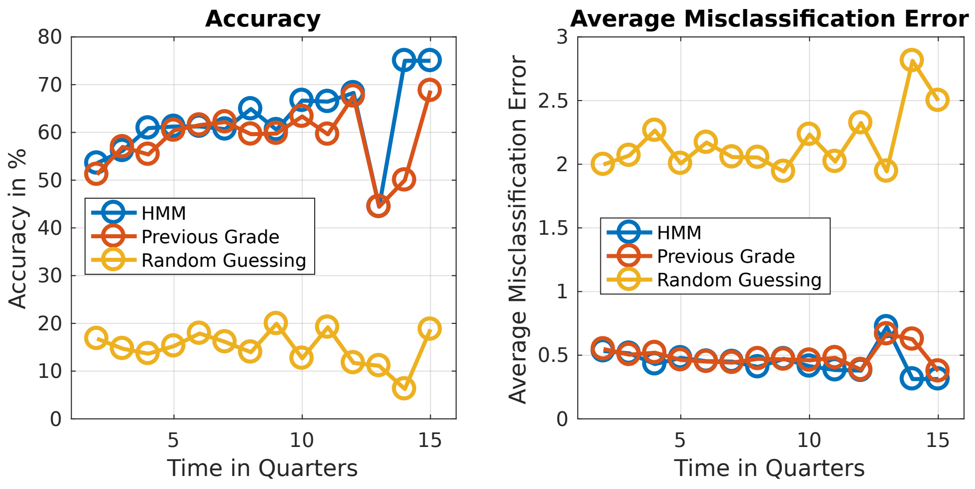


Figure 1: Main result of this work. For a detailed description the graphs, see Figure 9 in the appendix.

## 6 Summary and Discussion

This work demonstrated how one can use Hidden Markov Models to predict students future performance at university. It was shown that in its most basic form, the HMM is able to predict the discrete grade class that the next average grade will fall into with about 60% accuracy and with an average misclassification error of about half a class.

## 6.1 Simplifications

Multiple simplifications were introduced along the way. By removing some of those simplifications and thereby making the model more realistic, better results are likely to be obtained. Some of the simplifications made include averaging over all course grades per quarter, irrespective of the course code, treating all averages as equally informative irrespective of the corresponding sample size, neglecting the correlation between different courses, incorporating the static features in a very simple way, no adequate control of model complexity via a validation set $\mathcal{W}$ or similar, discretisation of the average grades instead of using a continuous emission distribution and a very rudimentary treatment of missing data values. By removing sequences and shortening sequences, around one quarter of the original data was lost. If the model could be modified to deal with missing data values, the additional data could be used in a validation set $\mathcal{W}$.

## 6.2 Comparison with the Benchmarks

It was observed that the HMM and the benchmark $\mathcal{B}_2$ yielded significantly better performance than random guessing, $\mathcal{B}_1$. Interestingly, both the accuracy of the HMM as well as the accuracy of $\mathcal{B}_2$ improve over the first three quarters. In case of the HMM this is expected as the HMM takes into account a students history of grades. In case of the method $\mathcal{B}_2$, this might be related to the fact that initially, when students first arrive at university, the variance in their grades is likely to be bigger than after the first couple of quarters, where they have settled in to university life and adapted to the academic requirements. To summarise, the HMM is a promising model to predict students grades that requires further research to be able to really judge its predictive power.

# References

[1] L. E. Baum, J. A. Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

[2] E. Cohn, S. Cohn, D. C. Balch, and J. Bradley. Determinants of undergraduate GPAs: SAT scores, high-school GPA and high-school rank. *Economics of Education Review*, 23(6):577–586, 2004.

[3] 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 (methodological)*, pages 1–38, 1977.

[4] M. Dunham and K. Murphy. PMTK3: Probabilistic modeling toolkit for Matlab/Octave, version 3. *https://github.com/probml/pmtk3*, 2012.

[5] G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[6] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning.* Springer series in statistics Springer, Berlin, 2009.

[7] F. Jelinek. *Statistical methods for speech recognition.* MIT press, 1997.

[8] S. Kotsiantis, C. Pierrakeas, and P. Pintelas. Predicting students' performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, 18(5):411–426, 2004.

[9] R. S. Mamon and R. J. Elliott. *Hidden Markov models in finance*, volume 4. Springer, 2007.

[10] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[11] L. Pachter, M. Alexandersson, and S. Cawley. Applications of generalized pair hidden Markov models to alignment and gene finding problems. *Journal of Computational Biology*, 9(2):389–399, 2002.

[12] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[13] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[14] J. Xu, K. H. Moon, and M. van der Schaar. A machine learning approach for tracking and predicting student performance in degree programs. *Accepted and to appear in IEEE Journal of Selected Topics in Signal Processing.*

## 7 Acknowledgements

I would like to thank Prof. Mihaela van der Schaar and her team for sharing their data with me which enabled me to train and test my model.

# A Appendix

## A.1 A Brief Introduction to Machine Learning

This work will not focus on explaining the principles of ML in general but rather give an introduction of the theory that is specific to HMMs, see Section 3. For a general introduction to ML, see [10] or [6]. However, some basic terms need to briefly be explained here because they are necessary for the understanding of the basis theory of HMMs.

**Training Set, Testing Set and Validation Set** The general aim of ML is to build data-driven algorithms that can make predictions about unseen data. Those algorithms learn from previous data in a *training set* $\mathcal{D}$ and can make decisions on data in a *testing set* $\mathcal{T}$ based on rules that were not explicitly given by the programmer but rather extracted from the training set $\mathcal{D}$. Depending on the situation and the type of data, a model is chosen. The model will depend on unknown *model parameters* which will have to be learned from the training set. If the model contains *hyper parameters* additionally to the model parameters, then one can use a third data set, a so called *validation set* $\mathcal{W}$ to learn the hyper parameters. Hyper parameters can for example be the number $K$ of nearest neighbours in a $K$-nearest neighbour model or the number $N_{\mathcal{S}}$ of hidden states in an HMM, see Section 3. If the original data set is too small to be split into three parts, one can employ a technique called *cross-validation*, which iteratively uses parts of the training set to learn the hyper parameters.

**Supervised Learning and Unsupervised Learning** There are two basic settings in ML: *supervised learning* and *unsupervised learning*. In supervised learning, the data is labelled, i.e. there are data points $x_i \in \mathbb{R}^{n_x}, i \in \{1, ..., N\}$ and corresponding labels $y_i$, where $N$ shall be the total number of data points. The task is to predict the class labels $\hat{y}_i$ of unseen data points $x_i \in \mathcal{T}$, given the pairs $(x_i, y_i) \in \mathcal{D}$. Throughout this work, a hat over a quantity shall refer to an *estimator* of a quantity. The other setting, unsupervised learning, refers to the situation where there are no class labels $y_i$, but we are interested in extracting very general, in some sense useful, information about the data. A common task is to find *clusters* in the data or to perform *dimensionality reduction* if the data is high dimensional.

**Classification and Regression** The class labels $y_i$ can either be discrete, in which case the set up is known as *classification*, or continuous, in which case the situation is referred to as *regression*. Some basic regression models include linear and logistic regression, whereas some basic classification models are given by support vector machines (SVPs) and classification trees. Many models can however be adapted to be used in one or the other situation.

**Generative Models and Discriminative Models** In order to make predictions about unseen data, a model is required. The model will usually be probabilistic, although this does not always hold as in the case of SVPs, which are non-probabilistic (see [10]). A model used in ML can either be generative or it can be discriminative. Discriminative models describe dependencies between data points by a conditional probability distribution. They do not allow one to sample from the joint distribution of the variables because they only specify the conditional distribution. Examples for discriminative models include logistic regression, SVPs and neural networks. In contrast to discriminative models, generative models aim at describing the underlying joint probability distribution, they are full probabilistic models that allow for sampling of each of the involved variables. Examples of generative models include HMMs, naive Bayes and Gaussian mixture models.

**Online and Offline** In the case of models describing time series data such as HMMs, there are two possible scenarios one can be in: in the *offline* scenario, all data points in the sequence are available to work with. In contrast, in the *online* scenario, only the data points up to the current time $t$ are available. One can imagine, for example, predicting student grades as they come in. Only the grades that lie in the past are available to base the prediction on, while grades that lie ahead of the current time are unknown.

**Overfitting and Underfitting** A model should be designed such that it generalises the concepts learned in the training set $\mathcal{D}$. The opposite would be be a model that simply memorises all of the data points in the training set $\mathcal{D}$. This model would show very good performance on the training set $\mathcal{D}$, but very poor performance on the testing set $\mathcal{T}$. This is a classical example of *overfitting*. The more complex the model, e.g. the more free parameters it contains, the better it can describe the training data. However, the level of complexity in the model has to be controlled in order to avoid overfitting and also *underfitting*, which describes the phenomena in which the model is not complex enough for the given situation. There are several ways to control the model complexity, two prominent methods are *regularisation* and the usage of a *validation set* $\mathcal{W}$. A regularisation term in the model penalises model complexity, it can be designed such that *sparse solutions* in parameter space are favourable. A validation set $\mathcal{W}$ allows the model complexity to be chosen empirically to optimise the performance.

## A.2 HMM Hidden State Chain

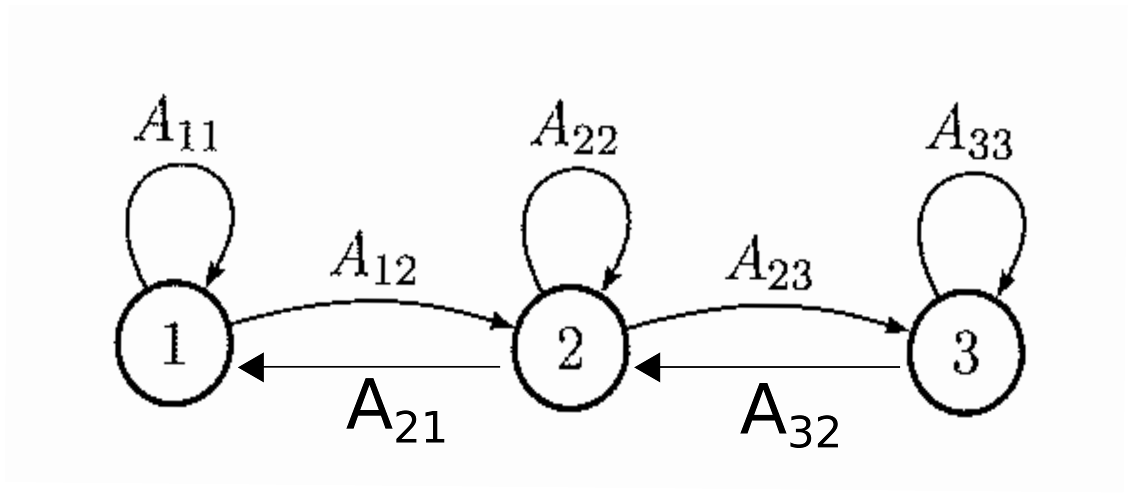


Figure 2: Illustration of the hidden state chain of the HMM constructed for the grade prediction problem. Figure taken from [10]. Two transitions were added to the original figure.

## A.3 Determining the Model Parameters for a Discrete Markov chain

This exploration follows an example in [10]. Let $O_i$ define a sequence of observations of length $T_i$, e.g. $O_i := (S_1, S_3, S_5, S_1)$ at discrete time points $t = 1, 2, 3, 4$ for the case $T_i = 4$. Let further $\mathcal{D}$ be a training set containing $N_\mathcal{D}$ sequences $O_i$ of corresponding length $T_i$, $\mathcal{D} := (O_1, ..., O_{N_\mathcal{D}})$. We may then write the probability of observing a particular sequence $O_i$ of observations as

$$\mathbb{P}(O_i|\theta) = \pi(q_1)\, A(q_1, q_2) \dots A(q_{T-1}, q_T)\,, \tag{36}$$

where $A(q_{t-1}, q_t)$ shall denote the element of matrix $A$ corresponding to the probability of transitioning from state $q_{t-1}$ to state $q_t$ and $\theta$ shall denote the combined vector of model parameters. The term $\pi(q_1)$ represents the probability of initially being in state $q_1$. We can rewrite this probability using the indicator function $\mathbb{1}$ as follows:

$$\mathbb{P}(O_i|\theta) = \prod_{j=1}^{N_{\mathcal{S}}} (\pi_j)^{\mathbb{1}(q_{i,1}=S_j)} \prod_{t=2}^{T_i} \prod_{j=1}^{N_{\mathcal{S}}} \prod_{k=1}^{N_{\mathcal{S}}} (a_{j,k})^{\mathbb{1}(q_{i,t}=S_k, q_{i,t-1}=S_j)} \,, \tag{37}$$

where $q_{i,t}$ shall refer to the observed state in sequence $O_i$ at time $t$. As before, let $N_{\mathcal{S}}$ denote the number of discrete states in the state space $\mathcal{S}$. Define the counting variables

$$N_j^1 := \sum_{i=1}^{N_{\mathcal{D}}} \mathbb{1}(q_{i,1} = S_j) \,, \quad N_{j,k} := \sum_{i=1}^{N_{\mathcal{D}}} \sum_{t=1}^{T_i - 1} \mathbb{1}(q_{i,t} = S_j, q_{i,t+1} = S_k) \,, \tag{38}$$

where $N_j^1$ represents the number of times that the initial state was given by state $S_j$ and $N_{j,k}$ represents the number of times a transition from state $S_j$ to state $S_k$ was observed in the data. Those definitions allow for the *log-likelihood* to be rewritten in the following way:

$$\log \mathbb{P}(\mathcal{D}|\theta) = \sum_{i=1}^{N_{\mathcal{D}}} \log \mathbb{P}(O_i|\theta) = \sum_{j=1}^{N_{\mathcal{S}}} N_j^1 \log \pi_j + \sum_{j=1}^{N_{\mathcal{S}}} \sum_{k=1}^{N_{\mathcal{S}}} N_{j,k} \log a_{j,k} \,. \tag{39}$$

One can now define an *objective function* $f(\theta) = \log \mathbb{P}(\mathcal{D}|\theta)$ and maximise this objective with respect to $\theta = (A, \pi)$ subject to the constraints

$$\sum_{j=1}^{N_{\mathcal{S}}} \pi_j = 1 \,, \tag{40}$$

$$\sum_{k=1}^{N_{\mathcal{S}}} a_{j,k} = 1 \,, \quad \forall j \in \{1, ..., N_{\mathcal{S}}\} \,. \tag{41}$$

Thinking of the problems as an equality constrained optimisation problem, we can write down the *Lagrangian function*

$$\mathcal{L}(\theta, \lambda_0, \lambda_1, ..., \lambda_{N_{\mathcal{S}}}) = f(\theta) + \lambda_0 \left( \sum_{j=1}^{N_{\mathcal{S}}} \pi_j - 1 \right) + \sum_{j=1}^{N_{\mathcal{S}}} \lambda_j \left( \sum_{k=1}^{N_{\mathcal{S}}} a_{j,k} - 1 \right) \,, \tag{42}$$

where the *Lagrange multipliers* $\lambda_0, \lambda_1, ..., \lambda_{N_{\mathcal{S}}}$ have been introduced. We can now take derivatives

$$\frac{\partial \mathcal{L}}{\partial \pi_j} = \frac{N_j^1}{\pi_j} + \lambda_0 \tag{43}$$

$$\frac{\partial \mathcal{L}}{\partial a_{j,k}} = \frac{N_{j,k}}{a_{j,k}} + \lambda_j \,. \tag{44}$$

For optimality, we set those equal to zero, solve for $\pi_j$ and $a_{j,k}$ and sum over $j$ and $k$, respectively

$$\sum_{j=1}^{N_{\mathcal{S}}} \pi_j = -\frac{1}{\lambda_0} \sum_{j=1}^{N_{\mathcal{S}}} N_j^1 \,, \tag{45}$$

$$\sum_{k=1}^{N_{\mathcal{S}}} a_{j,k} = -\frac{1}{\lambda_j} \sum_{k=1}^{N_{\mathcal{S}}} N_{j,k} \,. \tag{46}$$

However, note that the right hand side of Equations (45) and (46) is one by the constraints in Equations (40) and (41). It follows for the Lagrange multipliers:

$$\lambda_0 = -\sum_{j=1}^{N_{\mathcal{S}}} N_j^1 \,, \tag{47}$$

$$\lambda_j = -\sum_{k=1}^{N_{\mathcal{S}}} N_{j,k} \,. \tag{48}$$

Therefore, the MLEs for the model parameters in a discrete Markov chain may be written as *normalised counts*

$$\pi_j = \frac{N_j^1}{\sum_{j=1}^{N_{\mathcal{S}}} N_j^1} \,, \tag{49}$$

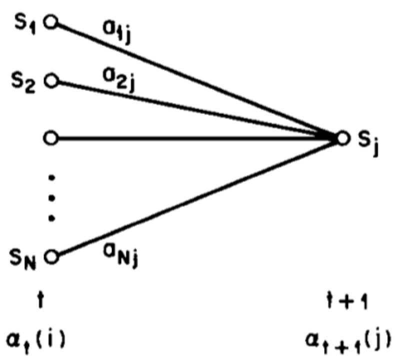
$$a_{j,k} = \frac{N_{j,k}}{\sum_{k=1}^{N_{\mathcal{S}}} N_{j,k}} \,. \tag{50}$$

This results is intuitive: The most likely estimator for $\pi_j$ is the number of times that state $S_j$ was found to be the initial state divided by the total number of sequences in $\mathcal{D}$ since it holds

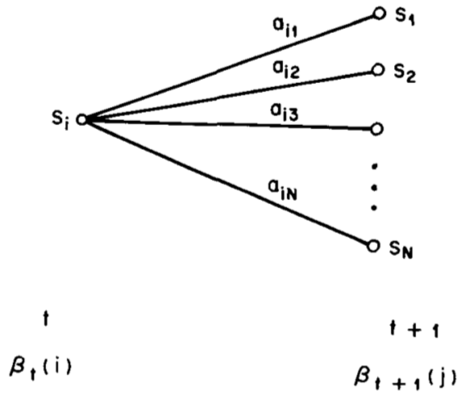
$$\sum_{j=1}^{N_{\mathcal{S}}} N_j^1 = N_{\mathcal{D}} \,. \tag{51}$$

The result for $a_{j,k}$ also has an intuitive explanation: it is the number of times that there was a transition from state $S_j$ to state $S_k$ observed in $\mathcal{D}$, divided by the total number of times that state $S_j$ was visited by any sequence in $\mathcal{D}$.

## A.4 HMM Algorithms



(a) Forward messages.



(b) Backwards messages.

Figure 3: On the left: illustration of the forwards messages which are the essential part of the forward algorithm, see [1]. Moreover, the forwards messages are needed in the Viterbi algorithm which we will be using to find the most likely sequence of hidden states, see [13] and [5]. The recursive calculation is based on the fact that there are $N_{\mathcal{S}}$ states that the system can have been in prior to state $S_j$, and the previous forwards messages can be used to infer the probabilities associated with the partial observation sequences leading to those previous states. On the right: illustration of the backwards messages which, together with the forwards messages, form the essential part of the forwards-backwards algorithm. We are not actually interested in the forwards-backwards algorithm, however, the backwards messages are also required in the Baum-Welch EM Algorithm which we will be using later on when estimating model parameters for the grade prediction problem. The principle of the backwards messages is analogous of the forwards messages, backwards in time. Both figures were taken from [12].

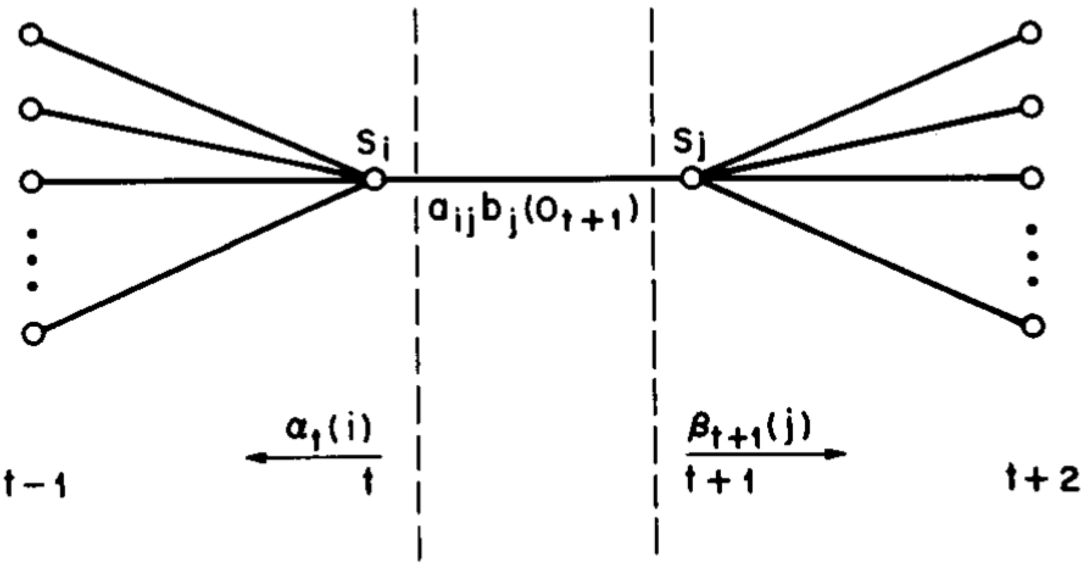


Figure 4: Illustration of the calculation of the two-slice marginals $\xi_t(i,j)$. Figure taken from [12].

## A.5 The Static Features and the two Groups of Students

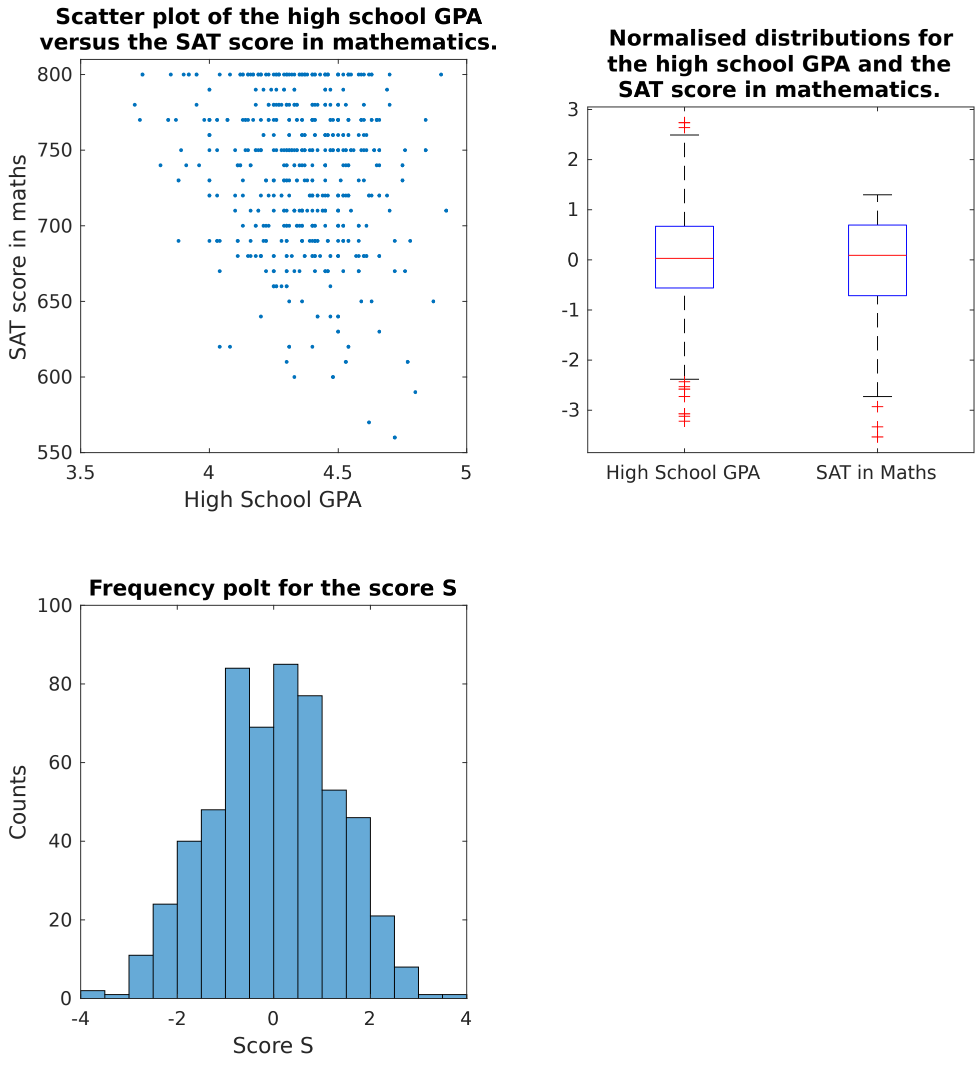


Figure 5: Illustration of the two static features high school GPA and SAT score in mathematics in the training set $\mathcal{D}$. In the top left corner, a scatter plot of the SAT score in mathematics versus the high school GPA is shown. In the top right corner, the two quantities $\widetilde{GPA}_H$ and $\widetilde{SAT}_M$ are shown in a boxplot. One can see that their distributions have very similar first and second moments, a result of the transformation that was carried out. In the lower left corner, a histogram of the score $S$ is shown. Students were assigned to either one of the two groups $A$ or $B$ depending on the value of their score $S$. Students with the top 50% of the scores were assigned to group $A$ while students with the lower 50% of the scores were assigned to group $B$.

## A.6 The Length of Stay of Students in the Dataset

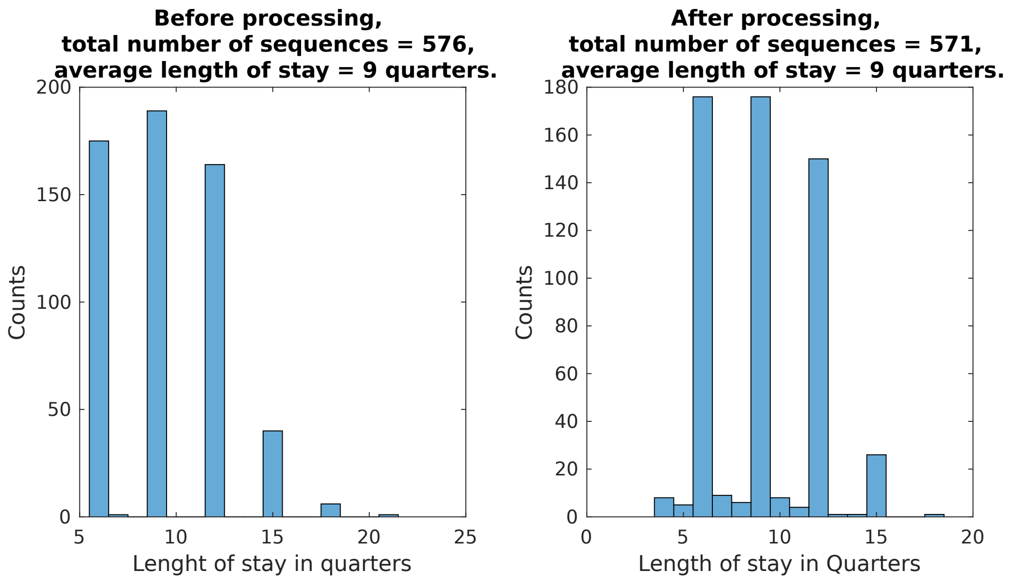


Figure 6: Comparison of the distribution of the quarters spent at university, before (left) and after (right) cutting the sequences at the point where there is no single grade per quarter. This reduced the sample size by five since some of the shortened sequences had a length shorter than the minimum length $T_{min}$. There are very few students staying longer than $T = 12$ quarters, which will make it difficult in the results section to obtain useful statistics in this region.

## A.7 Estimated Model Parameters

| | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| $S_1$ | 0.76 | 0.24 | 0.00 |
| $S_2$ | 0.08 | 0.73 | 0.19 |
| $S_3$ | 0.00 | 0.12 | 0.88 |

Estimated transition matrix for group $A$.

| | $S_1$ | $S_2$ | $S_3$ |
|---|---|---|---|
| $S_1$ | 0.64 | 0.36 | 0.00 |
| $S_2$ | 0.10 | 0.78 | 0.13 |
| $S_3$ | 0.00 | 0.09 | 0.91 |

Estimated transition matrix for group $B$.

Table 1: Transition matrices for both groups of students. It is interesting that the students with less competitive static features from group $B$ have higher probabilities to move to better educational states from state $S_1$ but not from state $S_2$, and that they have higher probability to remain in the highest educational state $S_3$. However, this does not necessarily mean that they receive better grade symbols as this is also determined by the discrete observation spectrum, see the tables below.

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ |
|---|---|---|---|---|---|---|
| $S_1$ | 0.02 | 0.12 | 0.28 | 0.58 | 0.00 | 0.00 |
| $S_2$ | 0.00 | 0.01 | 0.01 | 0.21 | 0.77 | 0.00 |
| $S_3$ | 0.00 | 0.00 | 0.00 | 0.01 | 0.18 | 0.81 |

Table 2: Discrete state dependent probabilities for the observations symbols for group $A$.

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $v_5$ | $v_6$ |
|---|---|---|---|---|---|---|
| $S_1$ | 0.03 | 0.13 | 0.31 | 0.53 | 0.00 | 0.00 |
| $S_2$ | 0.00 | 0.00 | 0.04 | 0.26 | 0.70 | 0.00 |
| $S_3$ | 0.00 | 0.00 | 0.00 | 0.01 | 0.24 | 0.75 |

Table 3: Discrete state dependent probabilities for the observations symbols for group $B$.

## A.8 Exemplary Model Trajectories

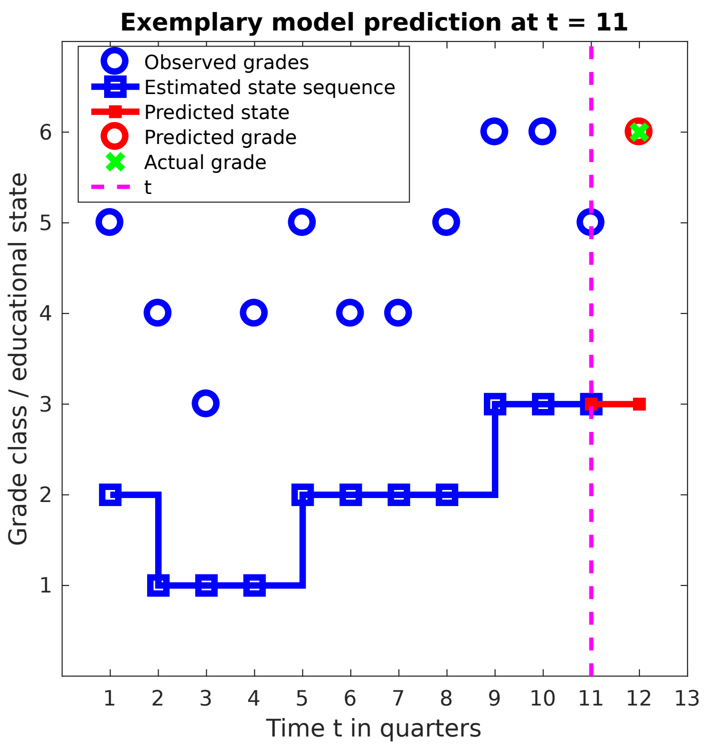
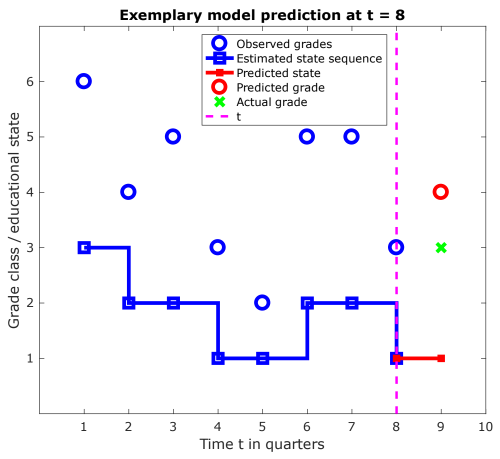
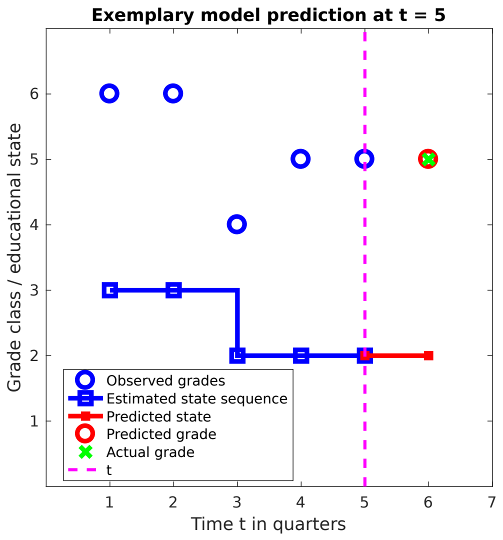


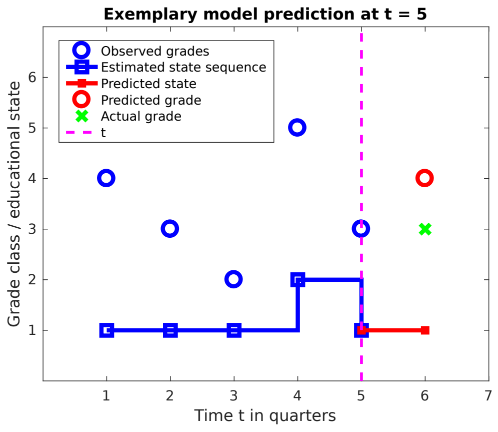
Figure 7: Exemplary model trajectory for a student from group $A$. This student stayed at UCLA for $T_i = 12$ quarters. The prediction is made after $t = 11$ quarters, so there is a relatively large amount of information available. The blue circles are the observed, discrete grade symbols up to time $t$. The blue line is the sequence of hidden states that the Viterbi algorithm estimated this student to have travelled through. The red circle is the predicted grade symbol for time $t = 12$. The green cross denotes the actual, observed grade symbol at time $t = 12$. The red line is the predicted state that the student will be in at time $t = 12$. The vertical, purple, dotted line specifies the current time (the time at which the prediction is issued). The model predicts the average grade for quarter twelve to be given by grade symbol $v_6 = 6$, which agrees with the actual, observed outcome. A model that simply uses the last obtained grade class as a prediction for the next grade class would have failed in this case.



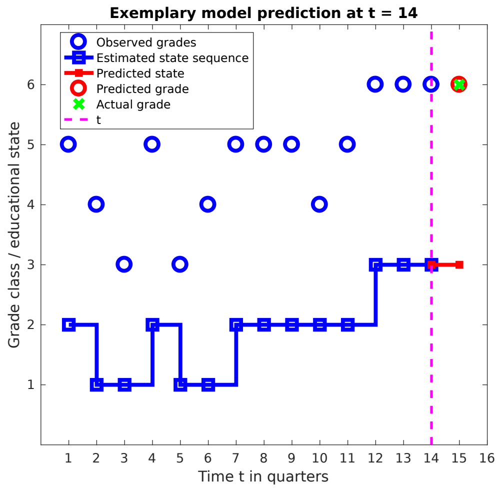
(a) Group $A$.



(b) Group $A$.



(c) Group $B$.



(d) Group $B$.

Figure 8: Four exemplary model trajectories illustrating different scenarios. In (a), the student from group $A$ stayed at UCLA for $T = 9$ quarters. The grade symbols he or she received varied between $v_2 = 2$ and $v_6 = 6$. The HMM predicts that the symbol in quarter nine will be one class better than the previous symbol. This differs from what the simple benchmark method $\mathcal{B}_2$ would have estimated. In this particular situation, the HMM is wrong and the simple benchmark $\mathcal{B}_2$ is right. In (b), a different situation is shown for another student from group $A$. In this case, both the HMM as well as $\mathcal{B}_2$ issue the correct prediction. In (c), there is a student from group $B$ whose grades vary between the symbols $v_2 = 2$ and $v_5 = 5$. The HMM is wrong in this case. Even though it predicts the student to only be in the lowest educational state, it still issues a grade prediction of $v_4 = 4$. This agrees with the parameters that the model learned from the data, see Subsection A.7 in the appendix. In (d), there is a student from group $B$ who stayed at UCLA for 15 quarters. The final predictions of HMM and $\mathcal{B}_2$ agree with each other and with the realised grade symbol.

## A.9 Accuracy and Average Misclassification Error

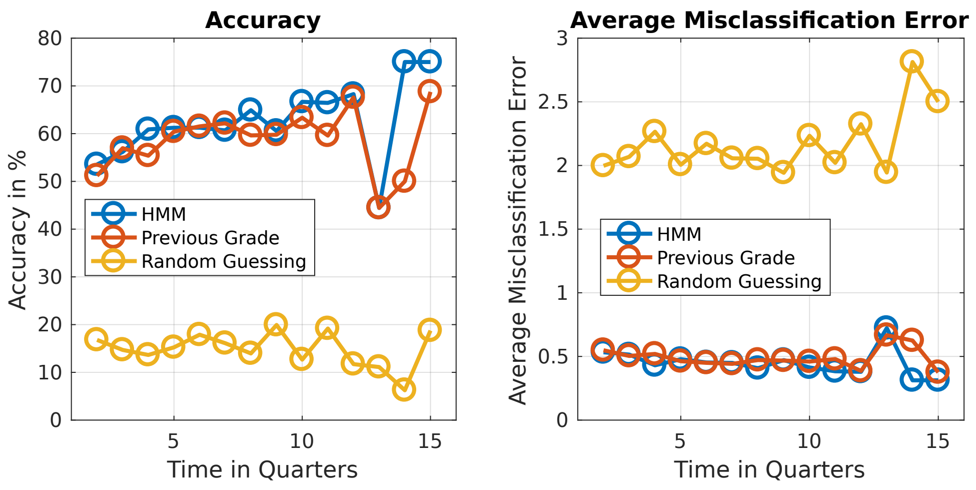


Figure 9: Main result of this work. On the left, the accuracy of the three methods is compared as a function of time $t$ which is measured in quarters. One would expect the HMM to become more accurate as more and more data becomes available, which can indeed be observed between $t = 2$ and $t = 5$, where the accuracy rises from approximately 50% to approximately 60%. From this point onwards, the accuracy of the HMM does not significantly increase, but rather oscillates around 60%. The data should only be taken seriously until $t = 12$, after which the amount of available sequences that are long enough becomes very rare (less than 20), and the statistics are not reliable any more. One can observe that both the HMM and the benchmark $\mathcal{B}_2$ behave considerably better than random guessing $\mathcal{B}_1$. However, it is difficult to argue that the HMM performs better than using the previous grade as a predictor for the next grade, given by $\mathcal{B}_2$. The two models behave very similar, the HMM is usually one or two percent more accurate than $\mathcal{B}_2$. On the right, the average misclassification error for all three models is shown. Again, $\mathcal{B}_2$ and the HMM produce significantly better results than random guessing, $\mathcal{B}_1$. Both the HMM and the method $\mathcal{B}_2$ have an average misclassification error of approximately half a class for most of the time, except for the very end, where the data becomes unreliable.