Gait sequence modelling and estimation

using Hidden Markov Models

Presented by:
Kouame Hermann Kouassi

Prepared for:
Fred Nicolls
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering

November 9, 2017

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:..........................
Kouame H. Kouassi

Date:..............................

Acknowledgments

Abstract

- Open the **Project Report Template.tex** file and carefully follow the comments (starting with %).
- Process the file with **pdflatex**, using other processors may need you to change some features such as graphics types.
- Note the files included in the **Project Report Template.tex** (with the .tex extension excluded). You can open these files separately and modify their contents or create new ones.
- Contact the latex namual for more features in your document such as equations, subfigures, footnotes, subscripts & superscripts, special characters etc.
- I recommend using the **kile** latex IDE, as it is simple to use.

Contents

Chapter 1

Introduction

1.1 Background to the study

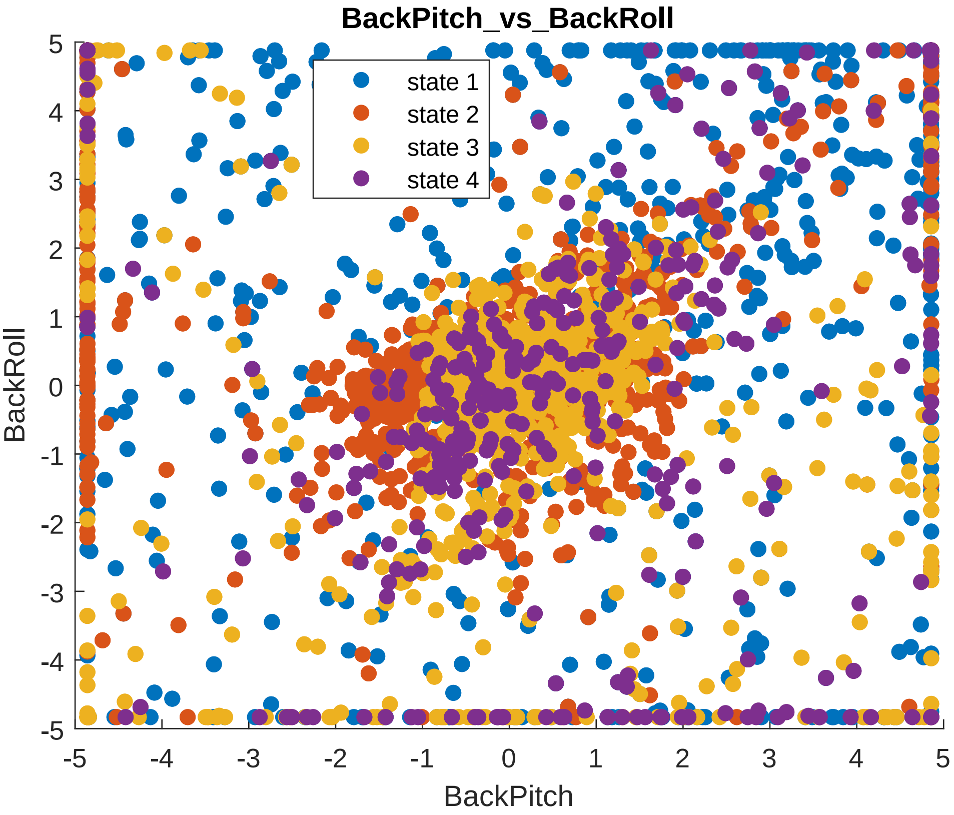
A very brief background to your area of research. Start off with a general introduction to the area and then narrow it down to your focus area. Used to set the scene [1].

Bio-inspired robotics uses nature to inform real-world engineering systems. Research has been conducted at UCT to investigate the manner in which a cheetah uses its tail for stability during high acceleration, quick turns and sudden braking, with an aim to incorporating identified mechanisms into sophisticated robot designs. One way to acquire useful data is to strap an inertial measurement unit (IMU) to an animal, and log the sensor data while certain actions are being performed. We currently have such a dataset of a dog moving, along with corresponding video data.

1.2 Objectives of this study

The objective of this project is to design, implement, and test Hidden Markov Models (HMM) for estimating gait sequence from Inertia Measurement Unit (IMU) data.

so that specific models can be formulated and their parameters estimated and interrogated. The project can be extended to include any other useful analysis of gait patterns from similar sensor measurements



1 - formulate model 2 - estimate its parameters 3 - Interrogate its parameters 4 - Useful analysis of gait patterns from IMU measurements

1.2.1 Problems to be investigated

Description of the main questions to be investigated in this study.

The main questions to be answered are the following:

1. How well can HMM model gait sequence dynamics using IMU data, in the abscence of enough training samples?
2. Can dimensionality reduction cause an increase in performance of HMM models when there is not enough training data?

1.2.2 Purpose of the study

Give the significance of investigating these problems. It must be obvious why you are doing this study and why it is relevant.

1.3 Scope and Limitations

Scope indicates to the reader what has and has not been included in the study. Limitations tell the reader what factors influenced the study such as sample size, time etc. It is not a section for excuses as to why your project may or may not have worked.

1 - Does not include data collection 2 - Focus on design of HMM only 3 - Focus on analysis of the model 4 - Focus on impact of dimensionality reduction

1.4 Plan of development

Here you tell the reader how your report has been organised and what is included in each chapter.

I recommend that you write this section last. You can then tailor it to your report.

Chapter 2

Literature Review

2.1 Gait sequence modelling and estimation

2.1.1 Quadrupede gait modelling

Periodicity

2.1.2 Quadrupede gait estimation

2.2 Case study: Inertia Measurement Unit

2.3 Hidden Markov Models

Hidden Markov Models (HMMs) are doubly embedded stochastic processes with a rich underlying statistical structure. Introduced at the end of the 1960s by Baum and colleagues, they have become one of the prefered techniques in speech recognition after the implementation of Baker and Jelinek in 1970s. HMMs have been successfully applied to various other engineering problems in pattern recognition for classification and fraud detection purposes, amongst others.

The type of HMM depends on the possible connections between the states. Thus, an HMM in which a state can transition to any other state is an ergodic. Other types such as the Left-Right model or Bakis do not allow all possible transitions between the states.

2.3.1 HMM parameters specification

An HMM is fully specified by the following parameters

1. N, the number of distinct states of the model. Together they form the set of individual states $S = \{S_1, S_2, ..., S_N\}$.
2. T, the number of observations. A sample observation sequence is denoted as $O = \{O_1, O_2, ..., O_T\}$.
3. $Q = q_t$, the set of states with q_t denoting the current state at time instance, t such that $q_t \epsilon S$ and $t = 1, 2, ..., T$.
4. K, the number of distinct observation symbols per state.
5. $V = \{v_1, v_2, ..., v_K\}$, the feature set of K dimensions.
6. $A = \{a_{ij}\}$, the state transition probabilities. a_{ij} denotes probability of transitioning from state S_i to state S_j.
7. $\Phi = \{\phi_j(k\}$, the probability distribution of observation symbols in state j.
8. The initial state distribution, $\pi = \pi_i$

For continuous HMM (CHMM), i.e, HMM with continuous-valued observations, Φ consists in a probability distribution function. Many applications have succefully modelled such distributions with mixtures of Gaussian distributions. As such, ϕ is approximated by a weighted sum of M multivariate Gaussian distributions η. For a given, observation sequence, ϕ and η are therefore given by equations 2.1 and 2.2,

$$\phi(O_t) = \sum_{m=1}^{M} \beta_{jm} \eta(\mu_{jm}, \Sigma_{jm}, O_t), \tag{2.1}$$

$$\eta(\mu, \Sigma, O) = \frac{1}{\sqrt{(2\pi)^K |\Sigma|}} exp(-\frac{1}{2}(O - \mu)' \Sigma^{-1} (O - \mu) \tag{2.2}$$

$$1 \leq j \leq N; 1 \leq m \leq M; \beta_{jm} \geq 0; \sum_{m=1}^{M} \beta_{jm} = 1$$

where β_{jm} is the mixture composition coefficient; μ_{jm}, Σ_{jm}, respectively the mean vector and covariance matrix of state j; M is the number of mixture components and K is the dimensionality of O. In practice, the log-likelihood of $\phi(O_t)$ is rather computed to avoid overflow when implemented on a machine.

As a summary, the compact specification of a continous valued observation HMM is defined by 2.3 and that of a discrete HMM in 2.4.

$$CHMM = \lambda_C = (A, \beta_{jm}, \mu_{jm}, \Sigma_{jm}, \pi) \tag{2.3}$$

$$DHMM = \lambda_D = (A, b_j(k), \pi) \tag{2.4}$$

Basic assumptions of HMMs theory

HMM theory is built on three basic assumptions listed below.

1. *The Markov assumption*: HMM assumes that the probability of being in the current at any instance of time t, is uniquely dependent on the previous state, at time, t + 1. More specifically, $a_{ij} = P[q_t = S_j | q_{t+1} = S_i]$. This assumption makes it unsuitable for long-range correlation capturing applications.

2. *The stationary assumption*: Furthermore, HMM state transition probabilities are assumed to be time-independent. Thus, the transition probabilities of two distinct time, t_1 and t_2 are identical, $P[q_{t_1} = S_j | q_{t_1 - 1} = S_i] = P[qt_2 = S_i | q_{t_2 - 1} = S_i]$. HMMs can therefore effectively model mechanisms with stationary observations.

3. *The output/observation independence assumption*: The current observation also known as emission symbol is statistically independent of the previous observations. It is "emitted" only by the current state, $P[O|q_1, q_2, ..., q_T, \lambda] = \prod_{t=1}^{T} P[O_t|q_t, \lambda]$.

The three assumptions make an HMM model a relatively simple graphic modelling to be implemented. This simplicity naturally comes with some limitations in modelling more complex problems, which however, may be modelled with higher order HMMs. Futhermore, the three assumptions are very similar to those of a Markov chain. This is because the stochastic process of an HMM pertaining to the hidden states can be reduced to a Markov chain. In fact, an HMM is an extension of a Markov Chain. The essential difference between the two is that, with the former, there is no a one-to-one mapping between the states and the observation symbols.

2.3.2 The three basics problems for HMM design

In , Lawrence argued that an HMM design needs to answer three fondamental problems. They are the *training problem*, the *evaluation problem*, and the *decoding problem*. Each problem and its solution is discussed in greater details next.

The evaluation problem

The evaluation problem is about answering this question: *Given the observation sequence $O = O_1O_2O_T$, and a model λ, how do we efficiently compute $P(O|\lambda$, the probability of the observation sequence?* The naive answer to this question is simply computing the $P(O|\lambda)$ according to equation 2.5:

$$P(O|\lambda) = \sum_{q_1}^{q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) ... a_{q_{T-1} q_T}(O)) \tag{2.5}$$

This approach has two issues, it is not only, computationally too expensive because of the exponential complexity with respect to T, but also, intractable for very long sequence. In pactice, $P(O|\lambda)$ is computed by an algorithm called *forward-backward* procedure, which is a more efficient method.

The decoding problem

The decoding problem can be reduced to this interrogation: *Given the observation sequence $O = O_1O_2O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1q_2...Q_T$ which is optimal in some meaningful sense i.e, best "explains" the observations?* Simply put, this problem is about deciphering the most likely hidden states that emitted the visible observation sequences. This is done dynamically using the Viterbi algorithm.

The training problem

Given the model, λ, the training problem raises the following question: *how do we adjust the model parameters λ to maximise the $P(O|\lambda)$, the probability of the probability of the observation sequence?* This problem is usually solved by iterative learning algorithms called expectation-maximisation. Examples of this algorithms are Baum-Welch method or any gradient based method.
When using Baum-Welch algorithm, the parameters are initialised by guesses then re-estimated iteratively to find the parameters with maximum likelihood. This method is vulnerable to local maxima issues. To avoid such cases, it is advice to run it multiple times with different initial values in order to keep the estimation with the highest likelihood value.

Overfitting, order of markov, robustness: bias-var

2.4 Expection maximasiation algorithm

2.5 The Viterbi algorithm

2.6 k-Nearest Neighbour

2.7 Dimensionality reduction

Dimensionality or dimension reduction is used pattern recognition, machine learning and statistics to find the most compact representation of the dataset by removing redundant and irrelevant information. It is achieve by extracting principal features, i.e, feature extracting or by selecting the most relevant subset of the initial feature vector, i.e, feature selection, using a supervised or an unsupervised approach.

2.7.1 Motivations for dimensionality reduction

When building a model, the need for dimensionality is supported by several reasons. Some of the important ones are presented in three points. Firstly, by reducing the feature space's dimension, we can build model with higher quality. In most classification problems, the feature domains contain variables with very little to no information for the purpose at hand. Thus, removing these features reduces the complexity of the problem which can in return, increases the model's accuracy.

Secondly, working with hundreds to thousands of features can be diffult to conceptualise and visualise. By using dimensionality reduction, we can better understand the model and present it to others by comprehensive visualisation.

The third reason is about efficiency in terms of computational time and storage. In general, pattern recognition and machine learning algorithms computionally intense. Besides, storage capacity is limited in some engineering applications such as embedded systems. So, solving the problem only with the relevant features can alleviate these two problems. Consequently, the computional speed of the model can increased by using dimensionality reduction.

Various dimensionality reduction have been developed in literature, the next section will present a handful of the ones used in the present work.

2.7.2 Feature selection: filters and wrappers

Filters and wrappers are two major categories of feature selection methods. The structure of both methods are illustrated by figure 2.1 and figure 2.2 , respectively. They both require a mechanism for generating a subset of the original feature set and a stoppin

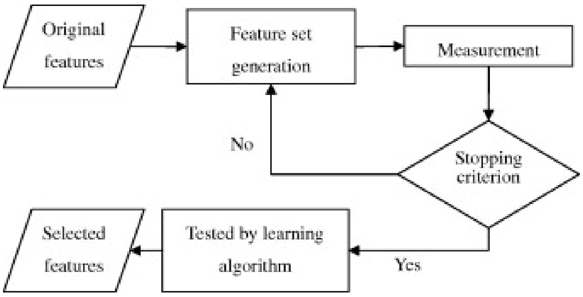


Figure 2.1: The procedure of filters in dimensionality reduction

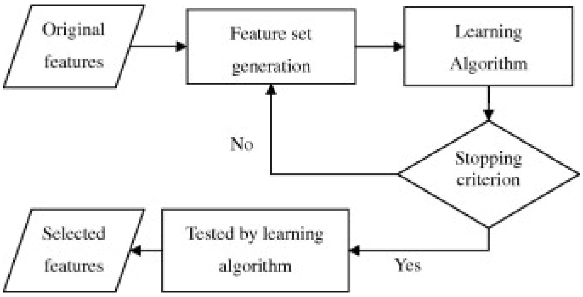


Figure 2.2: The procedure of wrappers in dimensionality reduction

criterion, which can be a distance measure, a measure of similarity between the features. However, wrappers identify the best feature subset using a learning algorithm usually by fully searching the feature space, whereas, filters use a simple measurement metric based on mutual information, correlation and other distance criteria. As a result of the two different approaches, filters are fast and do not guarantee optimal classification accuracy. On the other hand, wrappers give accurate prediction results but are very slow. Both approaches are often combined to build effective and efficient feature selection methods. One such approach is shown in figure 2.3. In this approach, the filters are used as a

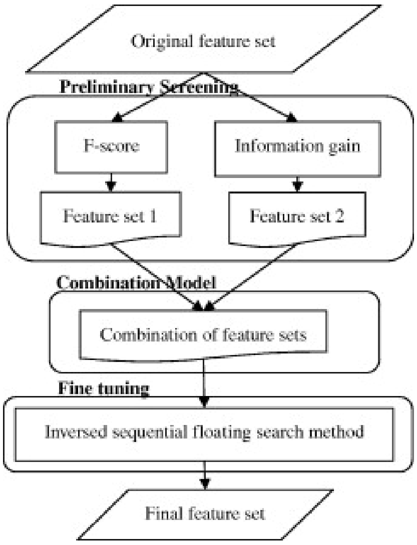


Figure 2.3: The procedure of hybrid filter-wrapper in dimensionality reductiony

preliminary stage to discard irrelevant features. A feature is deemed irrevant if it cannot discrinate between the different classes or if it contains redundant information. The output of the filter stage is a smaller set of relevant features which is fed into the wrapper to find the optimal final subset of features. Thus, the filter stage effectively reduces the search time of the wrapper. In classification problems, using the seperability index matrix to determine the classification content or degree of the feature generally results in very good results. The next section gives a particular attention to this approach.

Feature ranking using separability index matrix

In this section, we dives into a systematic approach to determine the 'classifiability of a feature' as present in In their paper, Jeong-Su, Sang Wan Lee and Zeungnam Bien, proposed a new criterion called separability index matrix to identify features that can discriminate between the different classes of a classification problem. The important concepts of this method are here defined together with their significance.

1. **Separability Degree Matrix (SDM)**

   SDM_k is a CxC symmetric matrix of the separability measures between the different classes of classification problem of C distinct classes given a particular feature x_k. It is defined in 1

   $$SDM_k = [J(w_i, w_j; x_k)]$$

   a

   where $J(w_i, w_j; x_k)$ denotes the separability or the distance value between class w_i and class w_j when the criterion function $J(.)$ such as mahalanobis or eucludian, is applied to the feature x_k. SDM_k is symmetric matrix with zero diagonal values, because for a given feature x_k, $J(w_i, w_j; x_k) = J(w_j, w_i; x_k)$ and $J(w_i, w_i; x_k) = 0$. This observation can be exploited to half the computation required to calculate $SDM_v = SDM_1, SDM_2, ..., SDM_N$, the set of all SDM for the feature set of size N.

2. **Separability Index Matrix (SIM)**

   $SIM_k = c_{ij}$ is a CxC matrix of binary values 0, 1. if $c_ij = 0$, then the classes w_i and w_j are not separable by the feature x_k.
   SIM_k is obtained by applying an threshold function to SDM_k. For instance,

   $$SIM_k(i,j) = 0 \quad if \quad SDM_k(i,j) < SDM_{avg}(i,j)$$
   $$SIM_k(i,j) = 1 \quad if \quad SDM_k(i,j) \geq SDM_{avg}$$

   where SDM_{avg} is the element wise average of SDM_v. SIM_k is significant because it can be used to systematically determine the irrelevant features, i.e, features whose $SIM_k = 0$, and/or redundant ones.

3. **Classifiability:** $G(x_k)$

Although, conventional distance criterion reveal the separability of the feature distribution, we are often more interested in, how effectively can a particular feature distinguish one class from another. This information is denoted $G(x_k)$, the ' classifiability of a the feature x_k '. $G(x_k)$ is computed by 2.6.

$$G(x_k) = \sum_{i=1}^{C}\sum_{j=1}^{C}(SIM_k * WM_k) \tag{2.6}$$

$$with \quad WM_k = SIM_k / \sum_{i=1}^{N} SIM_i \tag{2.7}$$

where * and / denote respectively element wise matrix multiplication and division.

This method can be used to effectively and efficiently rank the features in a classification for subset selection or an a preliminary step in a hybrid filter-wrapper feature selection solution.

2.7.3 Feature extraction

In this work, two distinct feature extraction tools were explored namely, the linear discriminant analysis and the principal component analysis. Each technique is further explained in the next two subsections.

Linear discriminant analysis

Linear discrimant analysis maps a K-dimensional dataset, $O \in \mathbb{R}^K$ into a subspace that maximizes the class between scatter with regards to the class within scatter. In order words, PCA finds the linear transformation - of the dataset into a lower dimensional space - that maximises the between class discrimination. It is a well received dimensionality reduction tool for classification problems by the computer vision and pattern recognition community.

LDA is a supervised dimensionality reduction method, it does require the dataset to be labeled. On the other hand, PCA, the next feature extraction method is unsupervised method.

Principal component analysis

Principle component analysis (PCA) is an orthogonal linear transformation of a dataset of T K-dimensional observations, $O \in \mathbb{R}^{TxK}$, into a so-called PC-space defined by the principal components (PC). Naturally, the dimension of the new space is smaller or equal to K, the original set's dimensionality. The components are ordered according to the variance of the features of the observation set. The principle components represent the eigenvectors of the covariance matrices of the features and the eigenvalues are measures of the features' variances. Thus, the first component corresponds to the feature with highest variance under the orthogonality constraint. In effect, PCA is not transforms the original observation set to a different space but also, ranks the features according to their variability. It is there be used as am unsupervised dimensionality reduction technique by simply discarding the features with low variance after finding the principal components.

2.7.4 Hybrid filter-wrapper methods

2.8 Sufficiency of Training Data

2.9 Techniques to increase Training Data

2.9.1 Mirroring

Chapter 3

Design methodology

3.1 Engineering design adopted

3.2 Understanding the problem

3.3 Understanding the dataset

3.4 Relevant literature

3.5 Testing the ability of the dataset to classify the states

3.5.1 Simple 16-states classifier using KNN

3.5.2 Predicting the successor using KNN

3.6 Design approaches

3.6.1 Turning the problem into a discrete problem

3.6.2 Splitting the problem into four smaller modules

Chapter 4

Hidden Markov Model and dimensionality reduction design

This section focuses on the design of the HMM used to test the hyphotheses postulated above.

4.1 Design overview

4.2 Description of available dataset

The available dataset was acquired from a moving dog using Inertia Measurement Units. Two inertial measurements units (IMU) were straped to the front and back of a dog. Each unit has an accelarometer, a gyroscope and a magnetometer. The dataset contains calibrated measurements of a dog running, walking, and trotting then walking; together with the footfalls. The footfall is represented by a binary value that indicates the state of the dog's leg: if it is on or above the ground, at a particular instant in its gait sequence. More specifically, the value 0 means leg up and the value 1 means leg down. The four variables representing the footfalls effectively constitute the ground truth, informing us about the state in which the dog is, at a given time in its movement.

The dataset can be retrieved from nine different matlab files. Each file contains twenty four matlab variables. The variables of interest are listed in the table 4.1.

Observations			
Body part	Accelerometer	Gyroscope	Magnetometer
Front	accFrontX	FrontPitch	magFront_cal
	accFrontY	FrontRoll	magFront_cal2
	accFrontZ	FrontYaw	magFront_cal3
— Back	accBackX	BackPitch	magBack_cal
	accBackY	BackRoll	magBack_cal2
	accBackZ	BackYaw	magBack_cal3

Table 4.1: IMU measurements and footfall variables in dataset

4.3 Quadrupede Gait sequence modelling with HMM

One of the objectives of this project is to effectively model the gait sequence dynamic of the dog from IMU measurements using HMM. Quadrupedes gait can be modelled as a succession of latent states observed through the 'visible' IMU measurements. The states representing the footfalls and the observations, the outputs of the accelerometer, gyroscope and the magnetometer. Similar to human gait mechanism, it is sound to assume that the current state of a quadrupede is conditionally dependent on its previous state. This inference combined with the statistical robustness of HMM makes it the best model candidate when the available dataset is not too large.

4.3.1 HMM model elements: states and observations properties

The problem at hand requires 16 distinct states that make up the state vector S, shown in equation 4.1

$$S = S_i = \{(LF, RF, LB, RB)\} = \{0000, 0001, 0010, ..., 1111\}. \tag{4.1}$$

$$|S| = N = 2^4 = 16$$

$$i = 1, 2, ..., 16$$

The 16 distinct states are derived from the combination of the four binary footfalls. In practice, the dataset may not reveal all the 16 states.

The stream of IMU measurement form the observation sequence. An observation instance is a row vector of K dimensions. The initial K value before any dimensionality reduction is 18, from the 18 IMU measurements. Thus, an observation sequence O is a TxK matrix of continuous-valued voltages as presented in 4.2. T is the total number of the successive measurements.

$$O_t = \{O_t^k\} = O_t^1, O_t^2, ..., O^18_t. \tag{4.2}$$

$$k = 1, 2, ..., 18. \tag{4.3}$$

$$t = 1, 2, ..., T. \tag{4.4}$$

4.3.2 Splitting the 16-states HMM in two 4-states HHMs

In order to simplify the problem, it was decided to split the four legs in two sub-parts: two front legs and two back legs. This decision exploits the fact that *"fore and hind quarters of dogs behaved like two independent bipeds"* as demonstrated in.
As a result, the initial 16-states HMM becomes two distinct 4-states HMMs. These two models may be combined to reconstruct the holistic 16-states model. With the two separate HHMs, we are faced a simpler task of dsicriminating between 4 distinct states instead of 16. A further benefit of this decision is that this work may well be applied to human gait estimation using IMU measurements. From here onward, we will focus on the 4-states HMM model.

4.3.3 State transitions

It was assumed that in its movement, a dog may transition from one state S_i to any other state S_j where $i, j = 1, 2, 3, 4.$. For instance, if a dog has its left leg above ground and its right leg on ground, at time instance t, it may move to any of the 3 other possible positions or remain in the same state, in the next time instance, t + 1. Thus, we are dealing with an ergodic type HMM, where all the states transitions are allowed. The graphical model of the simplified HMM is illustrated by figure 4.1

4.4 Pre-processing and increasing the dataset

Very little data pre-processing was required given that a relatively clean data was already available in Matlab files. So, this stage of the design consisted in three simple steps.

1. **Extracting and labelling the feature vector**: In this step, the TxK observation matrix was formed using the 18 different variables listed in 4.1, for a given gait

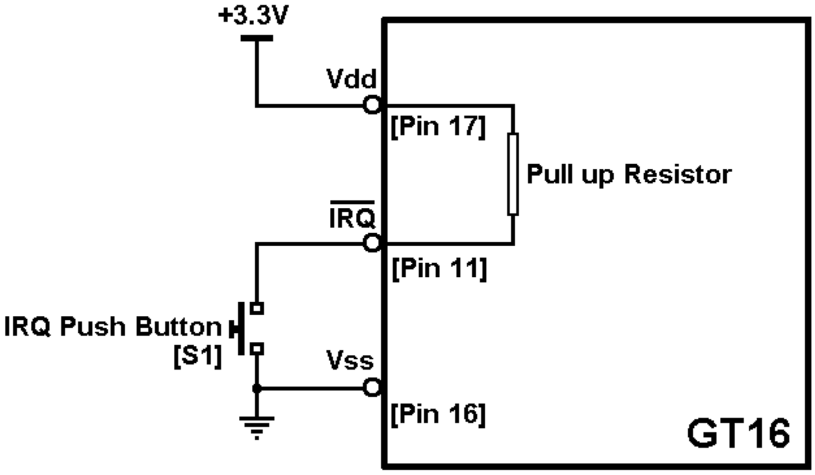


Figure 4.1: Ergodic HMM graphical model showing the hidden states, observation, and transitions between states

sequence. Using, the footfalls as ground-truth, i.e, (LF, RF) and (LB, RB), each observation instance was correctly labelled. The labelling is useful to stratify the different into the four states whenever required in the training process.

2. **Aggregating the gait sequences:** The individual gait sequence samples - for a walking, running and trotting dog - are very small in size, it was therefore decided to aggregate them in a single longer sequence. As such, the observation sequence becomes that of a dog walking, trot, then running or a similar combination of the three actions. The impact of such a decision on the model's parameter is that, instead of fitting a single action, it fits all the three types of actions. In theory, this loss of specificity can negatively affects the model's precision. Netherveless, the design a approach can be applied to a specific action if there is enough training data.

3. **Increasing the data by mirroring**: The aggregated observation sequence amounts to 2695 different samples. In order, to further increase the dataset, the mirrored sequence was appended to the initial sequence. This decision works on the assumption that, given a gait sequence, the reverse sequence is also a valid sequence from the subject matter. Practically speaking, each mirrored observation sample is a duplicate, no new sample is added. Moreover, the state distribution remains unchanged. However, as demonstrated in the results section 5.3, this increase in data size caused an increase in the model's performance.

As a summary, the output of the pre-processing stage is an 18x5390 matrix where each row represents an observation sample and each column a particular feature. This dataset was fed directly into the HMM model or in the dimensionality reduction module to remove irrelevant or redundant features. The design of the dimensionality reduction stage is now explained.

4.5 Dimensionality reduction

Five different dimensionality reduction methods were considered in this work: three feature selection methods and two are of feature extraction type. The feature selection methods are: a feature ranking based on separability of Index, a distance based forward feature selection, and a full-search feature selection with pre-defined output feature vector dimension. Principle component analysis (PCA) and linear discrimant analysis (LDA) were considered regarding the feature extraction. The two different types were considered to determine the most suitable one in gait sequence analysis with IMU measurements. For the feature selection methods, a preliminary experiment was performed to determine the optimal number of features. Before presenting this experiement, it is necessary to explain the feature ranking method based on seperability of index.

4.5.1 Feature ranking with separability Index (SI)

The ability of a feature X_k to classify the different states was defined as the 'between-class classifiability', $G(X_k)$. Refer back 3 for greater details. This quantity can be used a metric for ordering the different features. The greater the $G(X_k)$, the better feature X_k is at classifying the different states. Applying this concept to our problem at hand, the different $G(X_k) = G_(X_1), G(X_2), ..., G(X_18)$ were computed using the mahalanobis distance . Then, the 18 features (the IMU measurements) were ranked accordingly, in a descending order.

The ranking is advantageous because it can be used as feature selection method 4.5.3. This was the case when determining the minimum feature size discussed in the next subsection.

4.5.2 Determining optimal number of features using SI and MCE

The objective of this experiment was to determine the minimum number of features that best represent the dataset in a classification problem.

The experiment was performed with a K-Nearest Neighbour classifier together with feature ranking using separability index, by following the steps below:

1. **Step 1**: ***Features ranking***: Firstly, the features were ranked in a descending order according to the 'classifiability of each feature' as defined in 3.
2. **Step 2**: ***Classifier design***: Then, a 1-Nearest Neighbour (1-NN) classifier was designed and trained using half of the available dataset with the first K different features (K ¡ 18, the initial feature vector size). The first K features are assumed to be the best possible candidate features as explained in 4.5.1.
3. **Step 3**: ***Classifier's performance***: Furthermore, the classifier was tested with the remaining dataset and the misclassification error (MCE) was recorded.
4. **Step 4**: ***Iteration over the feature subset size***: Finally, Step 2 and 3 were repeated while varying the subset size from 1 to 18.

Figure 4.2 shows the percentage misclassification error (MCE) as a function of the feature subset size.

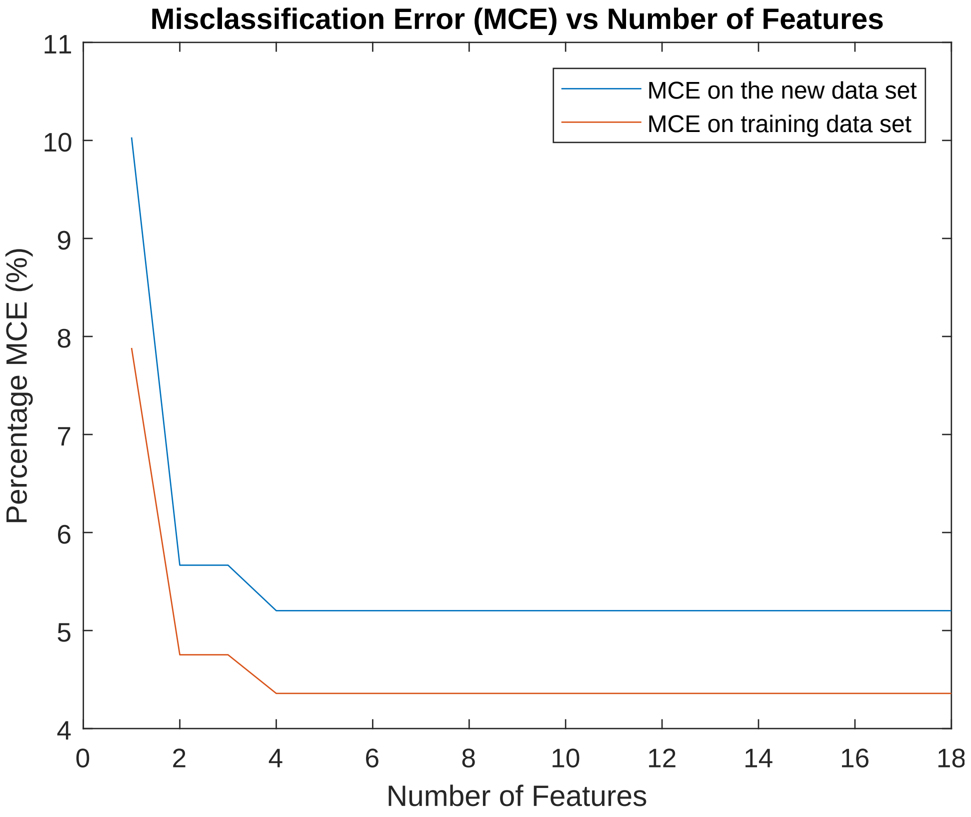


Figure 4.2: Misclassification error vs feature number using separability index and KNN

The MCE decresed from 1 feature to 3 and stays constant from 4 features to 18. The minimum number of features that best distinguishes the four states is therefore four. The following feature subset selection methods will consequently focus on determining the best combination of the four features.

4.5.3 Feature ranking

This ranking filter method is based on ranking the features using separability of index as presented in 4.5.1. It is simple and efficient dimensionality reduction method. It can be implemented in two very simple steps. Firstly, the features need to ranked in a descending order using a ranking criterion such as the 'feature classifiability'. Secondly, the first K features are selected where K is the desired number of output features. It is built on the assumption that the combination of features with higher 'classifiability content' is the best candidate. The drawback of this assumption is the possibility of redundant features. For this reason, this method can be used as a prelimirary step to a more sophisticated feature selection approach. In this present work, it was used as a separate filter method and its performance was compared to other methods in 5.3. The next section presents a more sophisticated but computationally expensive method: the forward selection.

4.5.4 Forward feature selection

Forward feature selection is a sequential feature subset selection, "in which features are sequentially added to an empty candidate set until the addition of further features does not decrease" a stopping condition is met. The stopping condition can be a pre-defined number of features as a design constraint or until the addition of further features leaves the evaluation criterion unchanged.

In this project, 1-Nearest Neighbour (1-NN) was used as evaluation criterion. Furthermore, since the purpose of the feature selection was to reduce the 18 initial features down to a much smaller number, the output feature number was limited to four, the optimal number of features according to finding of the previous experiement performed in 4.5.2. Because of this method is based on a greedy search, it is very computationally expensive. In order to make the algorithm faster, a preliminary step was introduced to select the relevant features. Thus, the features with zero classification ability were discarded. These features were identified by simply comparing their 'classifiability content' to zero. This step can significantly decrease the computationally time of the overall algorithm.

Although, the forward feature selection is more sophisticated than the feature ranking, it does not guarantee the best possible subset of features. This is because it does not do a full search of all possible combinations. The next filter method takes advantage of the relatively small initial feature vectore size, i.e, 18 to perform a full-search feature selection of pre-defined number of output features.

4.5.5 Full-search feature selection

Given the pre-defined output feature vector size (4) and the relatively small initial feature vector size (18), a full-search feature selection can be explored. Thus, by elimating the irrelevant features as explained in the above method 4.5.4, the number of combinations to be searched can be drastically reduced. In fact, nine IMU measurements had no 'classification content', reducing the number of combination to just $\binom{9}{4} = 126$. Because a full-search will ensure the best possible combination of features, this feature selection method was implemented. Here, the evaluation criterion was the prediction accuracy of the CHMM model. Effectively, making this method a pseudo hybrid filter-wrapper with pre-defined output feature vector dimension.

4.5.6 Principal Component Analysis (PCA)

4.5.7 Linear Discriminant Analysis (LDA)

4.5.8 Optimal PCA component number

4.6 Solving the three basics HMM problems

As a reminder, a continuous HMM model is completely specified by its initial state distribution: π transition matrix: A; the mean covariance matrices: μ, Σ which can be combined into Φ. If the observations are modelled with gaussian mixture distributions, one addition parameter is required for the initial mixture distribution: β. The next sub-sections discuss how each parameters was estimated in this project.

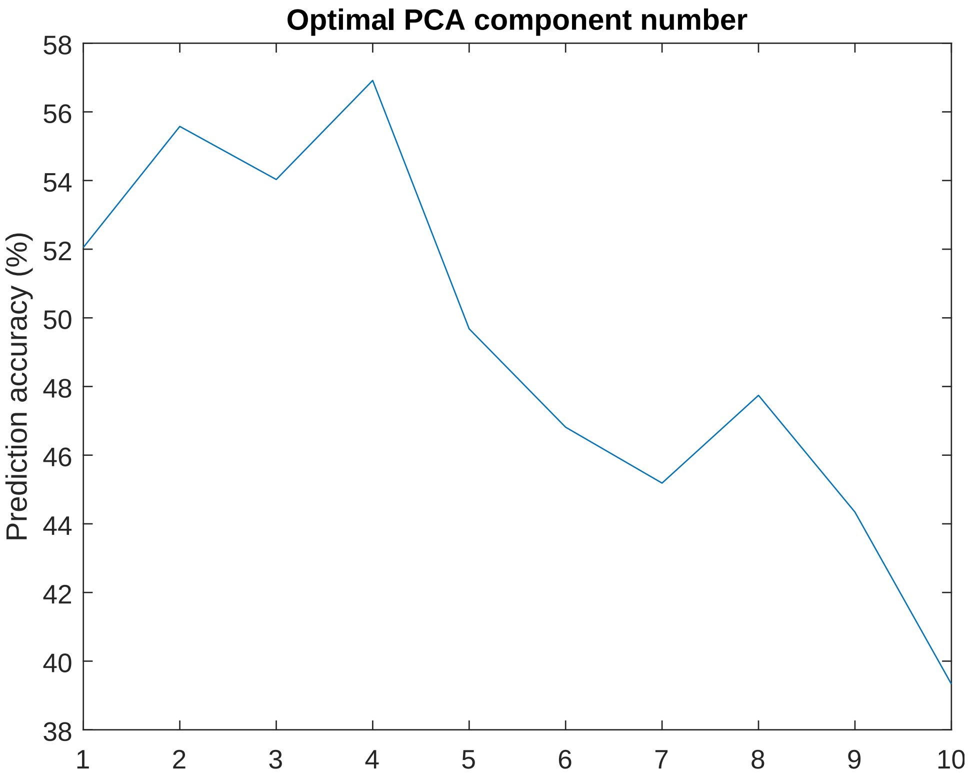


Figure 4.3: Optimal PCA component number

4.6.1 Solution to the training problem

As a reminder, the training consists in estimating the parameters the parameters of the HMM model. The subsections below explain how each parameter was estimated.

Estimation of the transition matrix: A

For each of the front and back 4-state HMM, the state transition matrix A, is a 4-by-4 matrix. Two different approaches were considered in the estimation of A. The two methods make use of the expectation maximisation algorithm but differ in the input arguments considered.

1. ***Approach 1: EM algorithm using Gaussian mixture model***: This method is the standard approach found in literature using expectation algorithm such as Welch-Baum algorithm any gradient based method. This algorithm was briefly explain in 2.5. So, prior to maximising the transition probabilities -in the M-step-, each observation is first labeled as having been emitted by a particular state S_i - in the E-step - using the Gaussian mixture model used to model the observations. This

process may introduce some degree of error, depending on how well the Gaussian mixture models the observation sequence. The resulting transition matrix may not be very reprentative of the real underlying state transition mechanism, especially when the dataset is not large enough. For this reason, the second approach, which eliminates the labeling step by exploiting available ground-truth, was developed in this work.

2. ***Approach 1: EM algorithm using ground-truth***: This technique takes advantage of the ground truth provided by the labeled dataset to reduce the HMM model to its underlying Markov process.
   This was achieved by setting the observation sequence to the state sequence, i.e, $O = O_1 O_2 ... O_T = S_1 S_2 ... S_T = S$. In effect, each continuous observation is replaced by its label. Effectively, the continuous-valued observations are transformed into discrete values, for the purpose of accurately determining the transition matrix. This transformation is sound because, the transition matrix is concerned with the probability of transitioning from one state to the other, it does not care about the actual value of the observation.
   Up to this point, we have a "discrete observation sequence O with known states, S". The next and final step is simply about computing the maximum likelihood of the transition using EM algorithm, as one would do for a discrete HMM with known state sequence.
   To make the transition matrix more representation of the state transition machanism, prior knowledge, also known as, pseudocount was added to the estimated transition probability as follows: $A = \{a_{ij}\} = \{a_{ij_{estimated}} + prior_j\}$. In this work,the pseudocount, $PseudoA_j$ for a given state S_j was set to the number of occurences of S_j, in the training data plus a constant value C: $PseudoA = |S_j| + C$. The additional constant C is to cater for the states and transitions not reflected in the dataset but that are theoretically possible. It can be chosen empirically until the transition matrix does not contain any zero. Although, this method is very simple and more effective, it has two limitations. It not only assumes that the number of states is known but also requires the training data to be labelled. Since, these two constraints are met in this design, it was chosen as a better approach.

Estimation of the mean matrix, covariance matrices and mixture distribution: μ, Σ and β

"Gaussian mixture models (GMMs) are powerful in modeling any desired continuous distribution and are used for example in speech processing successfully", and IMU based

HMM for gait human classification. Thus, in this project, the HMM's observations were also modelled with GMMs density functions, characterised by a KxMxN mean matrix: μ an KxKxMxN co-variance matrices: Σ and a MxN, mixture prior distribution β, where: K is *the number of features*; M, *mixture number*; and N, *number of distinct states.*
More practically, the observations were grouped into N classes based on the four distinct states. Then, the three GMM parameters were estimated for each state using a expectation maximisation (EM) as described by the points below.

- ***Parameters initialisation by k-means++***: The initial GMM parameters were estimated using a variant of k-means called k-means++ which uses a heuristic to find clusters in a dataset.
- ***EM-algorithm step***: The iterative EM algorithm is thereafter applied to optimise the initial parameters. The algorithm can be optimised by turning some parameters for both speed and accuracy. The main parameters considered in this projects are explained in the next points. In general, model accuracy was given preference over speed.
- ***Replication***: Given that the EM-algorithm may suffer from local maxima problem, it was repeated multiple times, i.e, 10 times, with different initial values at each iteration. The model with the highest log-likelihood value is therefore chosen.
- ***Maximum number of iteration***: Because the EM-algorithm does not always, converge, a maximum number of iteration needs to be provided. In the present work, it was experimentally set to 1000 iterations by favouring accuracy over speed.
- ***Termination tolerance***: The EM-algorithm terminates upon the log-likelihood equating to a certain value with a tolerance called termination tolerance. This value was set to 10^{-10}
- ***Regularization value***: This is a positive value added to the diagonal of each covariance matrix to ensure that the estimates are positive-definite.

In literature, the number of mixture of IMU data is set empirically to 2 or 3 mixture components, in this project, it was estimated using akaike information criterion (AIC), which is a measure of information loss. So, gaussian mixture models were built using EM algorithm while varying the M from 1 to 18, randomly stop at the maximum feature number. Since AIC is a measure of information loss, the model with the minimum AIC best represents the dataset. The number of mixture M, is therefore set to the mixture number of this model. This algorithm is outlined in 4.1 and a typical result is shown in 4.4.

```
function [numComponents, AIC] = optimal_mixture_component_by_AIC(data)
  X = data.observ;
  max_mixt_num = 18;
  AIC = zeros(1, max_mixt_num);
  BIC = zeros(1, max_mixt_num);
  GMModels = cell(1, max_mixt_num);
  options = statset('MaxIter', 2000);
  for k = 1:max_mixt_num
    GMModels{k} = estimate_model_parameter(X, k);
    AIC(k)= GMModels{k}.AIC;
    BIC(k)= GMModels{k}.BIC;
  end

  [minAIC, numComponents] = min(AIC);
  numComponents
  minAIC
end
```

Listing 4.1: Pseudo-code for finding the optimal number of mixture using AIC

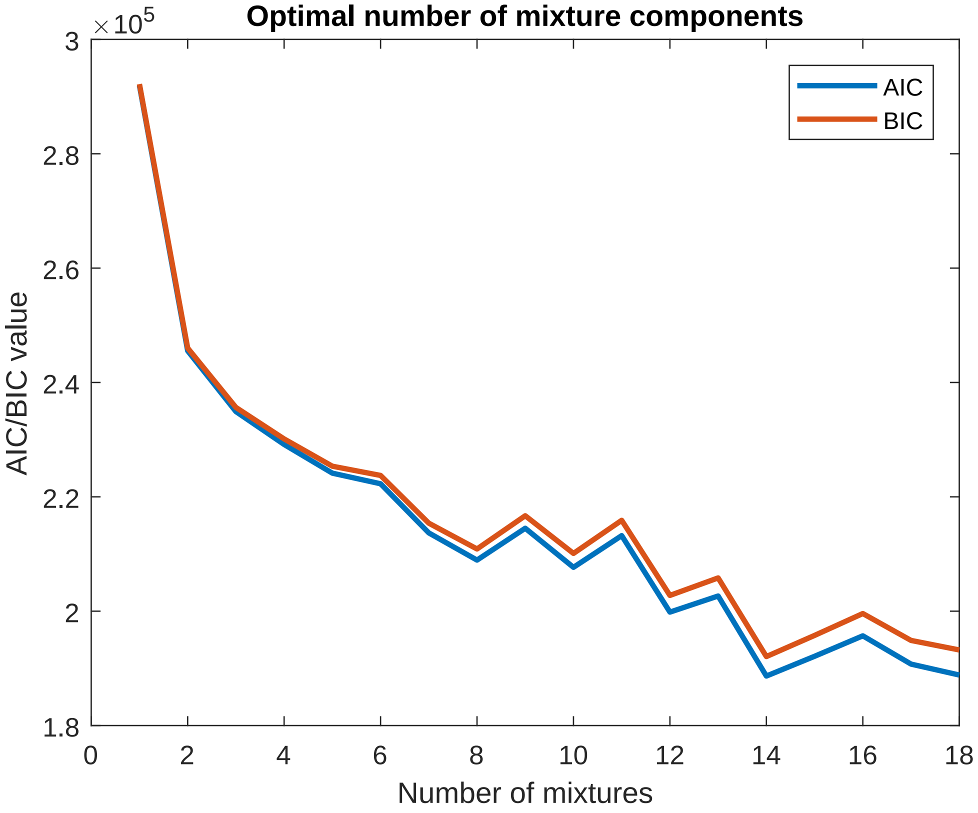


Figure 4.4: Finding best number of mixture component using AIC

As shown in 4.4 the same experiment can be performed using Baysian Information Criterion (BIC), an equivalent but stricter information loss measure. The more the number of components, the less information is loss. In this example, 14 is the optimal number of component for the range 0 to 18. As expected, the trend in BIC confirms this fact.

Inital state distribution: π estimation

In this project, the initial state distribution was estimated using the probability of occurence of each state in the training data sample. Thus, for a given set of length T, $\pi = \{\pi_i\} = \frac{|S_i|}{T}$. Similar to the transition matrix estimation, this could have been determined by an iterative process if the state sequence was unkown. However, given that there is an available state sequence, it was deemed unnecessary.

4.6.2 Solution to evaluation problem

The evaluation problem is about efficiently computing the probability of a given an observation sequence $O = O_1 O_2 ... O_T$. In other words, the likelihood of the sequence having been generated by the HMM model. This problem was solved in two parts described below. Note that the log-likelihood or log-posterior probabilities were computed to avoid overflow or underflow problems.

1. ***Part 1: Computing the log-probabilities***: In this step, the estimated Gaussian model is used to calculate the log- probabilities of the dog being in all four states according to 2.1 and 2.2.
2. ***Part 2: Determining the log-likelihood with Log-forward-backward***: The log-probabilities together with the state transition matrix, A and the initial state distribution *pi* are fed into the log-forward-backward algorithm to compute the log-likelihood.

4.6.3 Solution to decoding problem

The decoding problem consists in predicting the hidden states, i.e, the positions of the dog's limbs that "generated" the sequence of IMU measurements. After successfully decoding the initial state, $Q_0 = q_0$ the problem is reduced to predicting the next state $Q_{t+1} = q_{t+1}$ when in state $Q_t = q_t$. This simplification follows from the *Markov assumption* 2.3.1. Fortunately, the state transition matrix A gives insight into determining the most likely next state Q_{t+1} fron the current Q_t with the initial state distribution *pi* as bias. This fact can be represented by the following equation $Q_{t+1} = AQ_t + \pi$. Practically speaking, the state sequence was decoded with the Log-Viterbi algorithm using A, *pi* and the log-probabilities obatined from the Gaussian model 2.3.1. This algorithm dynamically computes the posterior log-probabilities of the observations for all four states. The state with the highest probability is most likely to have generated the observation in sequence.

4.7 Model's classification accuracy evaluation

This brief section deals with the evaluation of the HMM model's classification accuracy. The problem being about successfully identifying the hidden states, the accuracy of the model was evaluated as a percentage of the number of correctly predicted states over the total number of states. Differently put, the classification error is the percentage of misclassified states. However, misclassifications that occur at the transition from a give state S_i to the next state S_{i+1} were not considered a misclassification. These errors are simply due to the HMM model predicting the transition at one step early or one step late in the dog's sequence. Listing 4.2 shows the pseudo-code of the accuracy evaluation.

```
function accuracy = evaluate(ground_truth, estimation)
  accurate_estimation = sum(ground_truth == estimation) +
    count_wrong_transitions(ground_truth, estimation);
  accuracy = 100 * accurate_estimation/size(ground_truth, 1);
end

function count = count_wrong_transitions(ground_truth, estimation)
  count = 0;
  for i = 1:size(ground_truth, 1) - 2
    if (ground_truth(i) ~= ground_truth(i + 1) || estimation(i) ~=
    estimation(i + 1)) ... % transition in ground_truth or estimation
      && ((ground_truth(i + 1) ~= estimation(i + 1))... % estimation lagged
     or led the transition
      && (ground_truth(i + 2) == estimation(i + 2)) ... % by just one step
    , i.e, it got it correct in the next step
      && (ground_truth(i + 1) == ground_truth(i + 2) || estimation(i + 1)
   == estimation(i + 2)) %make sure it was just a transition in ground
    truth or in estimation
      count = count + 1;
    end
  end
end
```

Listing 4.2: Pseudo-code for computing the model's accuracy

4.7.1 Cross-validation

Talk about partitioning

4.8 Practical implementation

The prototype design was implemented with three main MATLAB toolboxes: *MATLAB Statistics and Machine Learning Toolbox*, pattern recognition tool called *prtoools* and an open-source HMM implementation by QIUQIANG KONG called *matlab-hmm*. Table 4.2 summarises the particular design component for which they were used along with the specific called.

Design component	**Toolbox**	**Function**
Transition matrix	Statistics and Machine Learning	hmmestimate
Gaussian mixture	Statistics and Machine Learning	fitgmdist
State posterior probability	matlab-hmm	Gmm_logp_xn_given_zn
Log-likelihood	matlab-hmm	LogForwardBackward
State sequence secoding	matlab-hmm	LogViterbiDecode
Forward feature selection	prtools	featself
Separability degree matrix	prtools	distmaha
PCA	prtools	pcam
LDA	prtools	fisherm
K-NN classifier	prtools	knnc
Data partitioning	MATLAB and prtools	cvpartition and gendat
Visualisation	MATLAB and prtools	scatter, plot and scatterd

Table 4.2: Main design components together with toolboxes and functions used for their implemenention

4.9 Implementation testing

4.9.1 Prove that the reduced dataset represents the original well enough

4.9.2 Prove that information loss is not much: BIC and AIC

4.9.3 Proving learning ability by increasing loglik for many iteration: check one of papers

4.9.4 Prove that prior affects the prediction accuracy

Design

Chapter 5

Results

5.1 Experiement 1: The effect of aggregating both front and back sonsor measurements

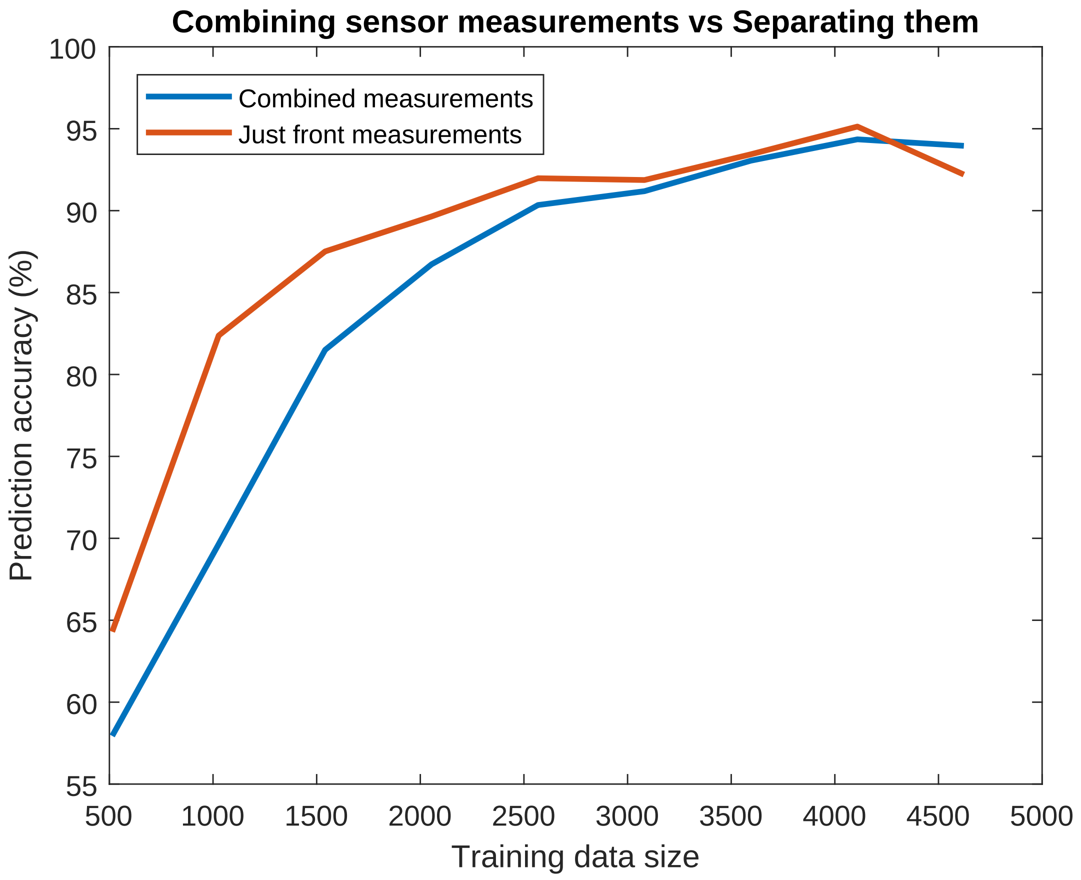


Figure 5.1: The effect of aggregating both front and back sensor on accuracy

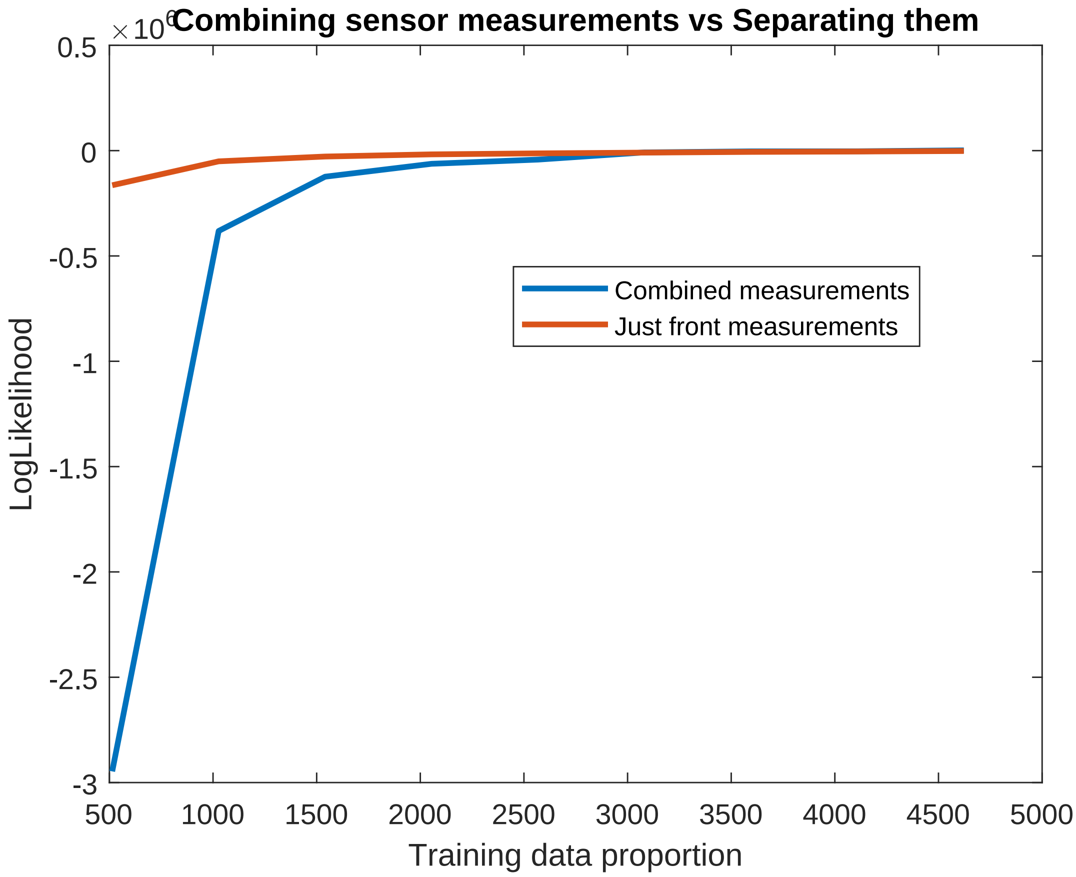


Figure 5.2: The effect of aggregating both front and back sensor on log-likelihood

5.2 Experiement 2: The effect of a CHMM' observation dimensionality on its performance

5.2.1 Aim of the experiement

The aim of this experiement is to investigate how the number of features impacts the accuracy of a Hidden Markov Model with continuous emission symbols (CHMM), in the abscence of enough training data. Thus, the hypothesis under investigation is:
In the absence of enough training data, a CHMM with observations of high dimensionality performs poorly.

5.2.2 Experiment apparatus

To perform this expereiment, the following materials are required:

- λ, a continuous Hidden Markov Model specified by $\lambda = (A, \beta_{jm}, \mu_{jm}, \Sigma_{jm}, \pi)$.
- At least two sample data sets training the model and testing it.
- A criterion to rank and select subsets of features.
- A measure to evaluate the performance of the CHMM model.
- Finally, a way to visualise the results of the experiments

5.2.3 Experiment procedure

The expreriment was performed with the steps listed below:

1. Step 0 - Preliminary data pre-processing: This step consisted in the data pre-processing as described in
2. Step 1 - Partitioning data into training and test sets: Here, the dataset was randomly sampled into training and test sets. The training set was relatively small, it was a sequence 539 observations.
3. Step 2 - Feature ranking: The features were sorted in a descending order based on their ability to discriminate the different states of the CHMM. The separability index method described in was used for this purpose.
4. Step 3 - Data subset selection: Select the optimal feature subset, starting with 1 dimension.
5. Step 4 - Model building and training: The CHMM model, λ was built and trained using with training dataset using the optimal feature subset.
6. Step 5 - Model testing and evaluation: The model was tested with the test dataset. The test consisted in decoding the most likely state sequence given a previously unseen sequence of observations. This path prediction was evaluated based on the evaluation criterion presented in
7. Step 6 - Iteration: Step 3 through step 5 were repeated while varying the feature subset size until the maximum size, which is 18 in this case. In each iteration, the prediction accuracy was stored in an array for visualisation.
8. The different accuracies were finally ploted as a function of the observation dimensionality. Moreover, the observations were grouped based on the corresponding hidden state

sequence and scattered in a 2-dimensional principal component space. This is to compare the decoded states against the ground-truth.

5.2.4 Experiment results

The results of the experiments are presented in figure 5.3, 5.4, 5.5, 5.6, 5.7, 5.8.
5.3 and 5.4 show how the hidden state decoding performance and the log-likelihood estimated by the CHMM model, train with just 539 observations, varies as the observation dimensionality increases.
5.5 and 5.6, 5.7 and 5.8 illustrate how the estimated state sequences compare to reality for observation sequences of 5-dimensions and 18-dimensions. 5 and 18 dimensions were presented because they are the two extremes in terms of accuracy. Other dimensions may be found in the appendices,

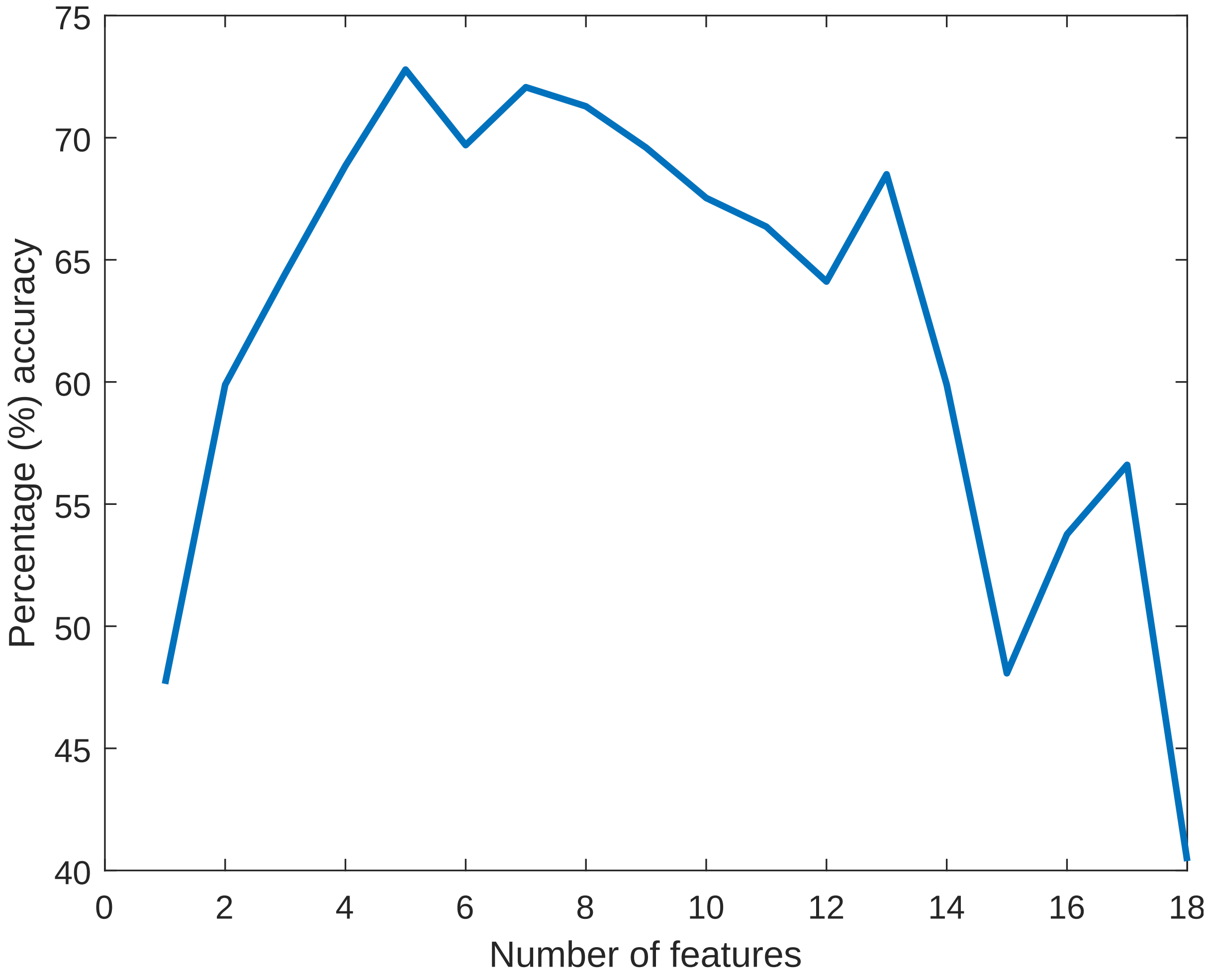


Figure 5.3: The effect of CHMM's observation dimensionality the state sequence decoding accuracy

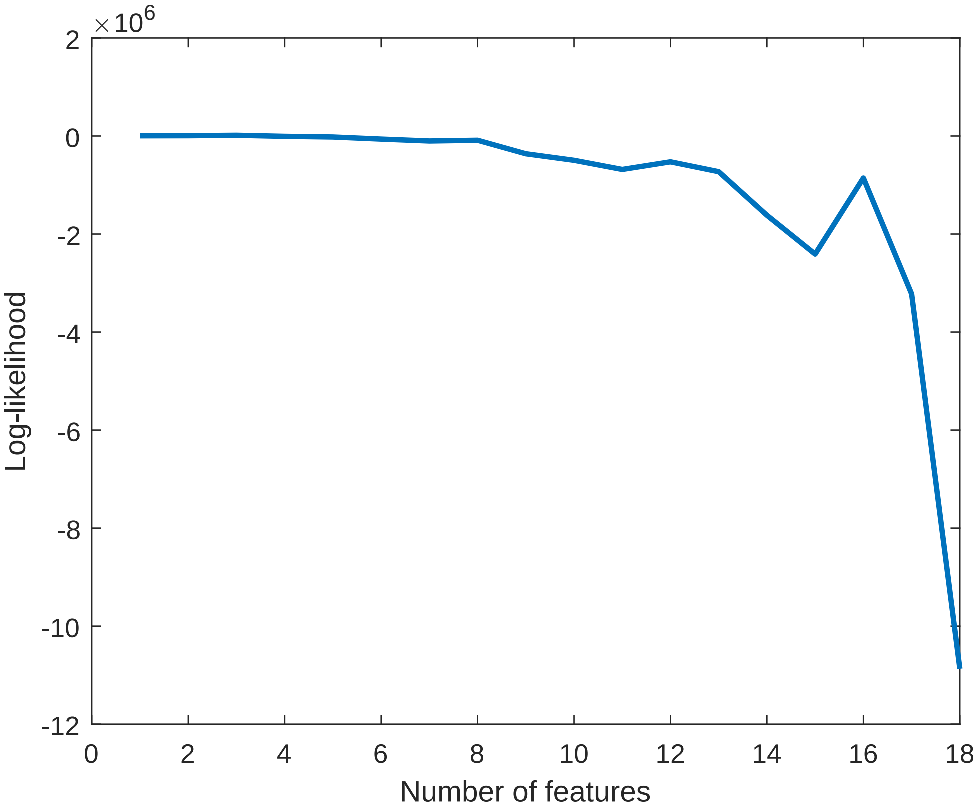


Figure 5.4: The effect of CHMM's observation dimensionality the log-likelihood

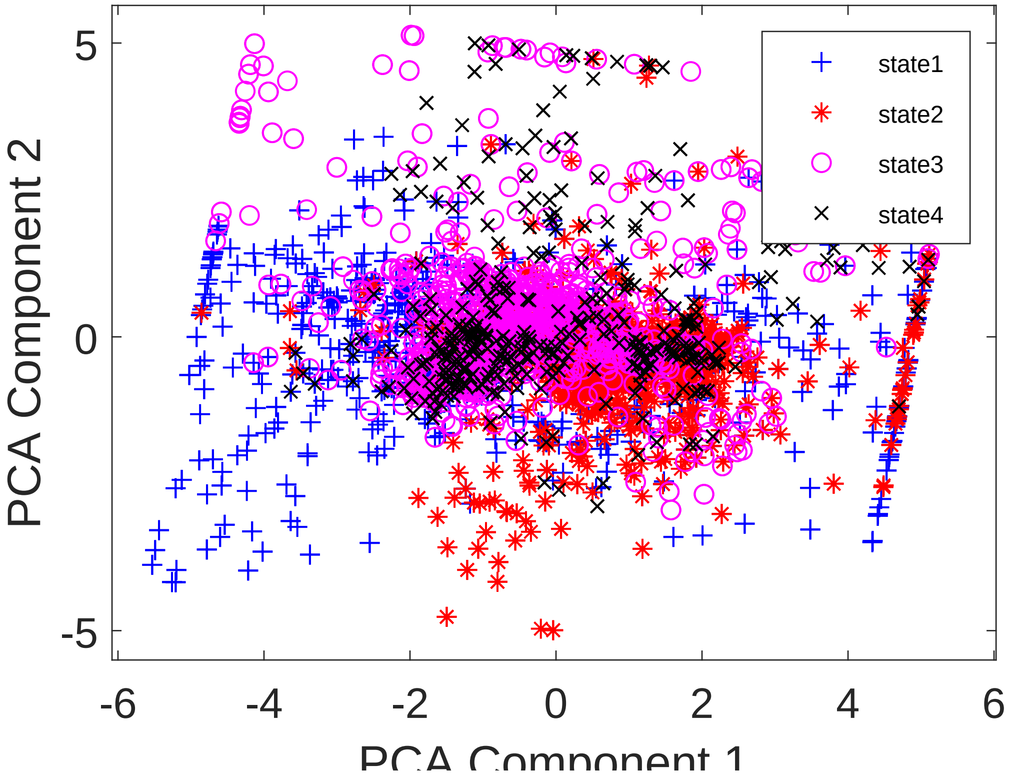


Figure 5.5: Scatter plot of 5-dimensional observations grouped per state based on ground-truth state sequence

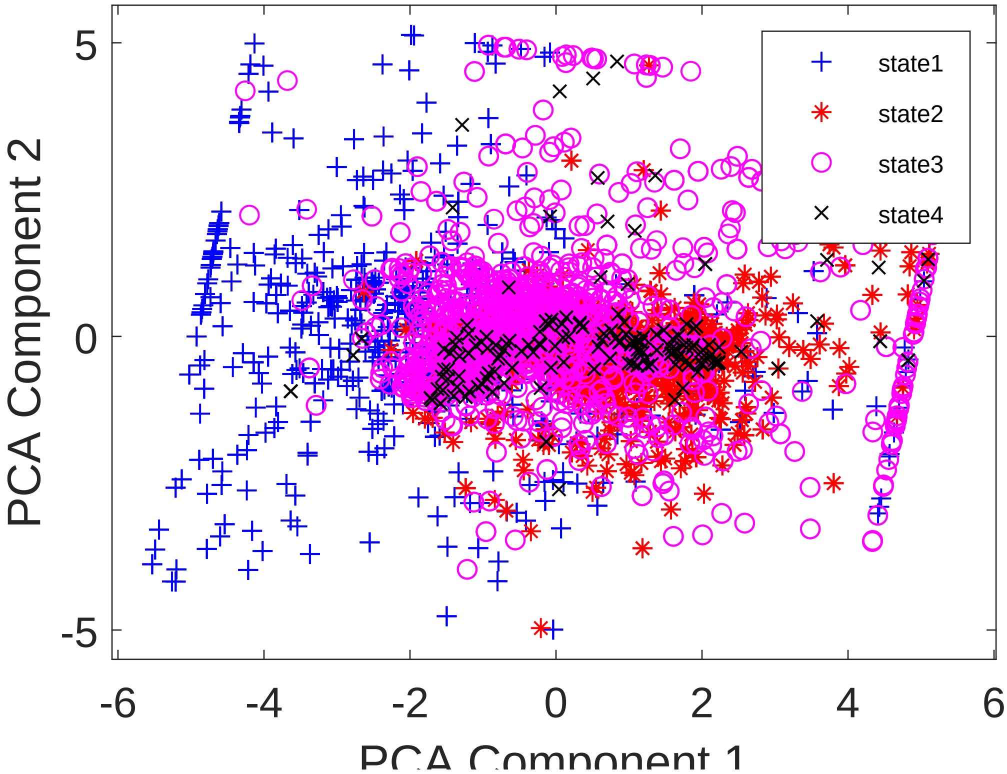


Figure 5.6: Scatter plot of 5-dimensional observations grouped per state based on estimated state sequence

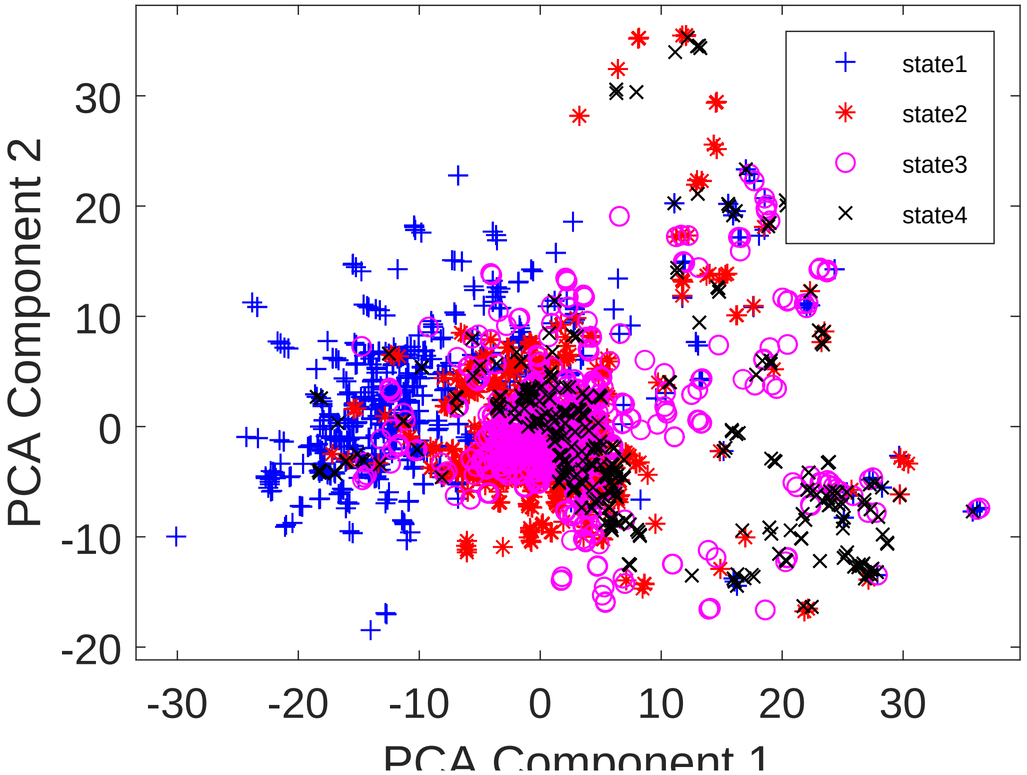


Figure 5.7: Scatter plot of 18-dimensional observations grouped per state based on ground-truth state sequence

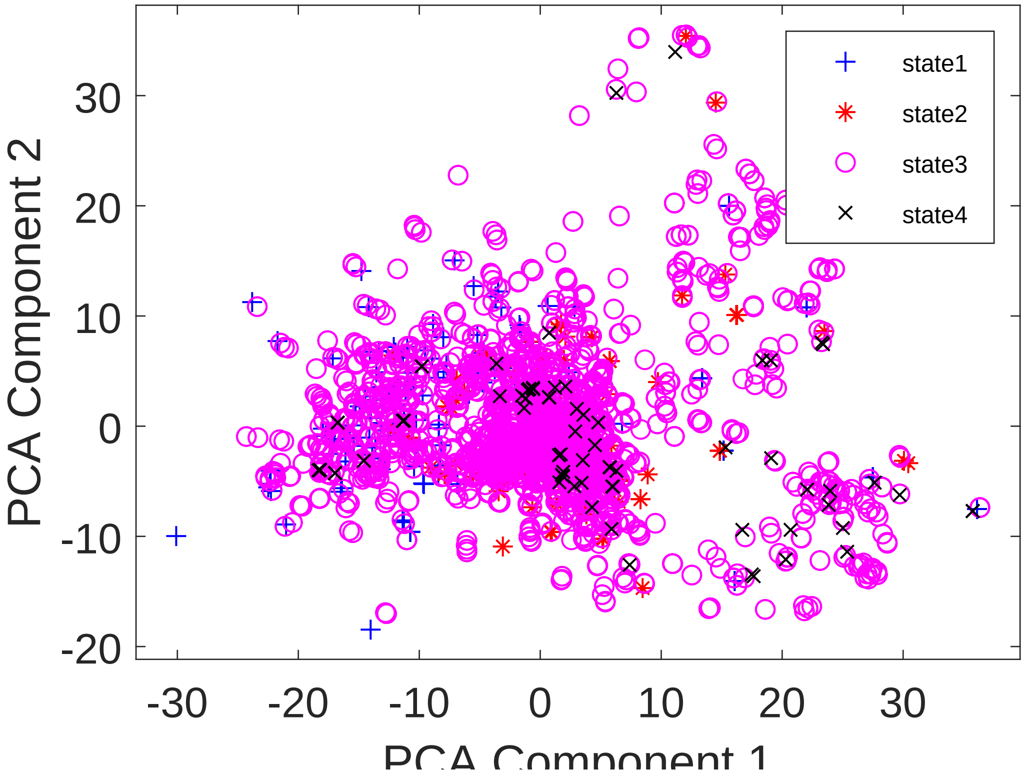


Figure 5.8: Scatter plot of 18-dimensional observations grouped per state based on estimated state sequence

5.2.5 Analysis of results

5.2.6 Conclusions and recommendations of the experiment

5.3 Experiement 3: The effect of dimensionality reduction on CHMM's performance

5.3.1 Aim of the experiement

The aim of this experiment is to investigate the effect of dimensionality reduction on the performance of a continuous Hidden Markov Model (CHMM). The hypothesis under investigation is therefore the following: ***Dimension reduction can cause an increase in a CHMM model performance when there is not enough training data.*** Thus, the performance of the CHMM without and with various dimensionality reduction are compared to test the hypothesis.

5.3.2 Experiment apparatus

The assets needed to perform the experiment are listed below.

- λ, a continuous Hidden Markov Model specified by $\lambda = (A, \beta_{jm}, \mu_{jm}, \Sigma_{jm}, \pi)$.
- Dimensionality reduction methods. Two wrapper and two filter methods were considered. The wrapper methods were Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA). The two filters methods were feature ranking with similarity index and a combination of forward feature selection and similarity index.
- A Performance metric. The metric used were the hidden state decoding accuracy and the log-likelihood of the EM algorithm.

5.3.3 Experiment procedure

The experiment was performed as follows. First, the dataset was partition into two different set for training and testing using random sampling. Using the same traing dataset five different models were built and trained, $\Lambda = (\lambda_{NoReduction}, \lambda_{PCA}, \lambda_{LDA}, \lambda_{SI}, \lambda_{SI-forward})$. Then the different models were tested with the same test dataset and the prediction accuracy as well as the EM algorithm loglikehood were recorded. The training and the testing were repeated while varying the proportion of training set used from 10% to 90% of the total dataset. The prediction accuracies and the EM algorithm likelihoods were finally plotted as a function of the training data size for each model. These findings are presented in the figures 5.9 and 5.10.

5.3.4 Experiment results

Firstly, figure 5.9 how the performances of the five CHMMs compare against each other as the training data size increases. Secondly, the loglikelihoods presented in 5.10 show how effectively can each model recognise an observation sequence generated by the underlying mechanism.

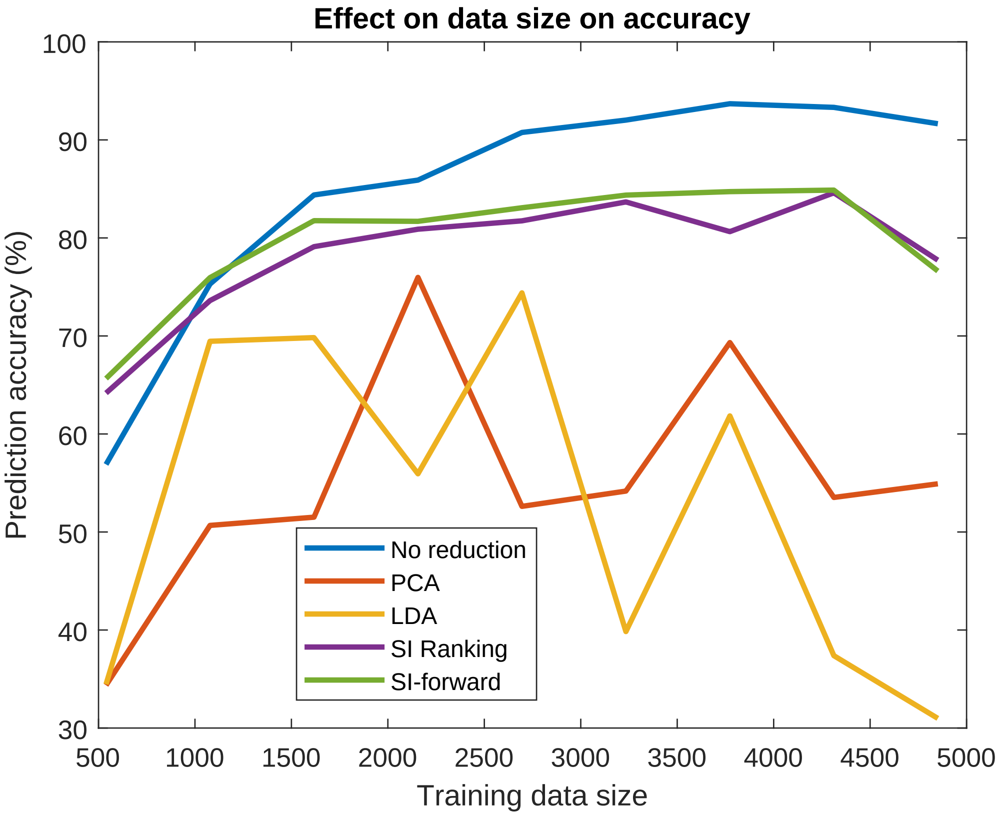


Figure 5.9: The effect of training datasize on the prediction accuracy

5.3.5 Analysis of results

5.3.6 Conclusions and recommendations of the experiment

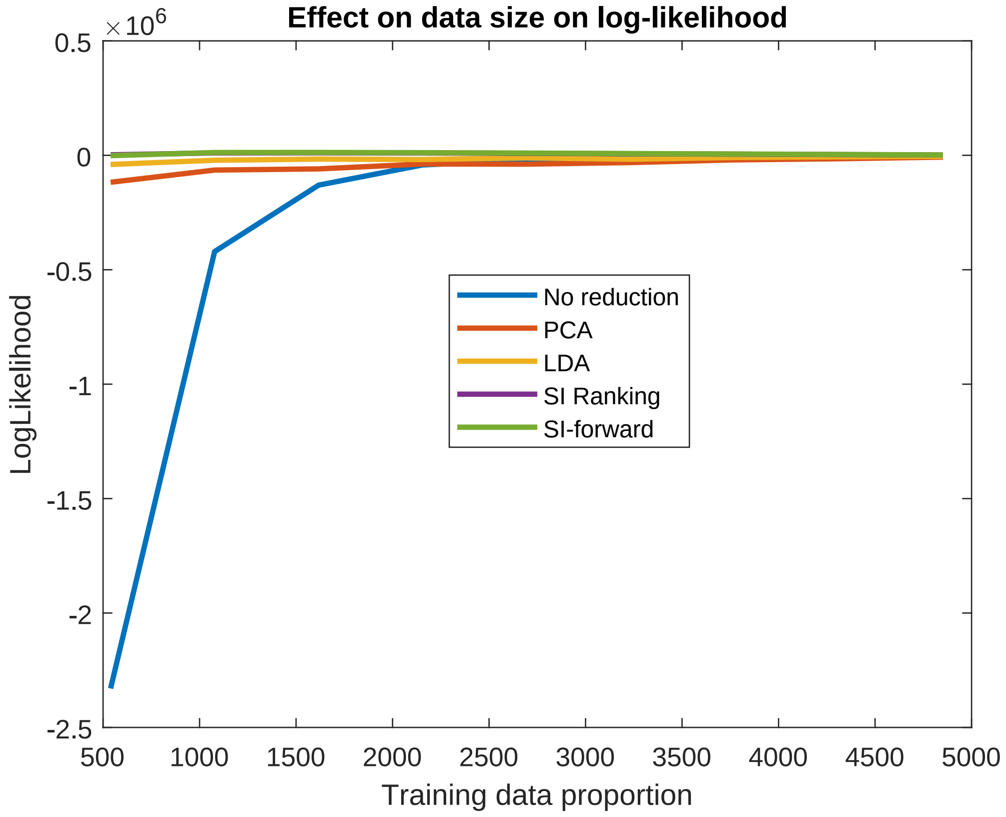


Figure 5.10: The effect of training datasize on the log likelihood

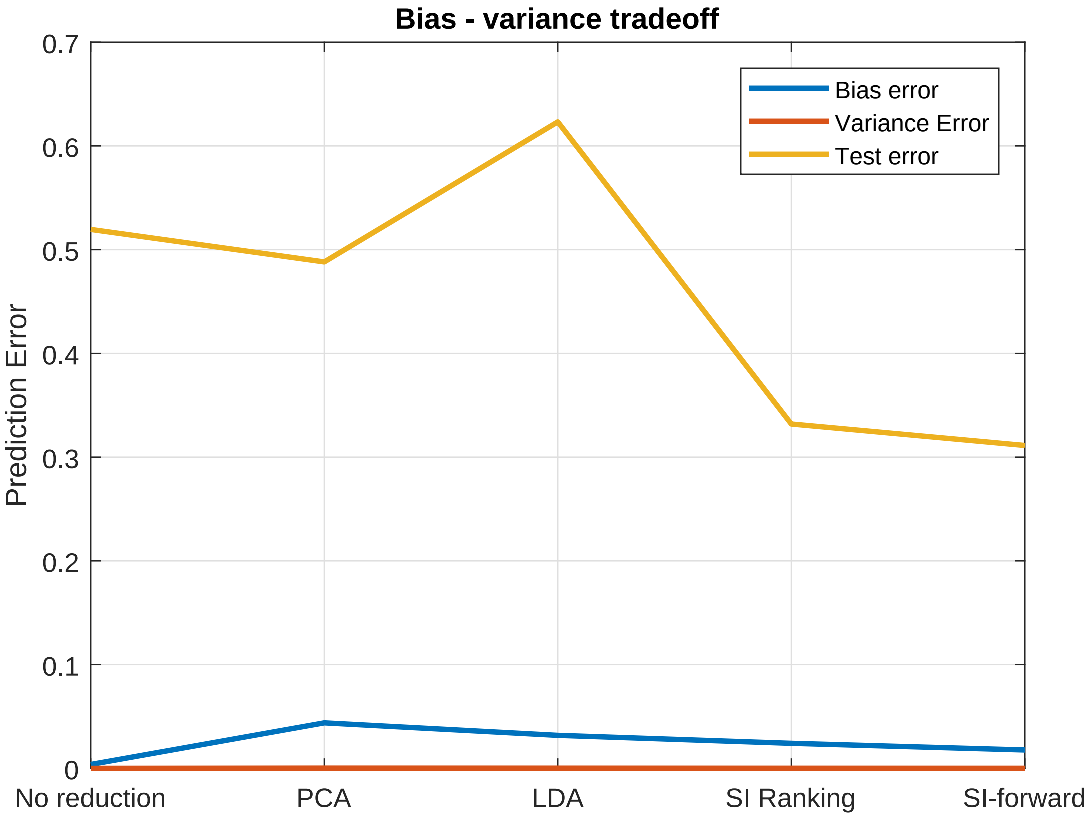


Figure 5.11: Bias-Variance tradeoff analysis

Chapter 6

Discussion

Here is what the results mean and how they tie to existing literature...

Discuss the relevance of your results and how they fit into the theoretical work you described in your literature review.

Chapter 7

Conclusions

These are the conclusions from the investivation and how the investigation changes things in this field or contributes to current knowledge...

Draw suitable and intelligent conclusions from your results and subsequent discussion.

Chapter 8

Recommendations

Make sensible recommendations for further work.

Use the IEEE numbered reference style for referencing your work as shown in your thesis guidelines. Please remember that the majority of your referenced work should be from journal articles, technical reports and books not online sources such as Wikipedia.

Bibliography

[1] M. S. Tsoeu and M. Braae, “Control Systems,” *IEEE*, **vol. 34(3)**, pp. 123-129, 2011.

[2] J. C. Tapson, *Instrumentation*, UCT Press, Cape Town, 2010.

Appendix A

Additional Files and Schematics

Add any information here that you would like to have in your project but is not necessary in the main text. Remember to refer to it in the main text. Separate your appendices based on what they are for example. Equation derivations in Appendix A and code in Appendix B etc.

Appendix B

Addenda

B.1 Ethics Forms