UNIVERSITY OF CAPE TOWN

Masters Thesis

Discriminative Training of Hidden Markov Models for Gesture Recognition

Author:
Jan Hendrik Combrink

Supervisor:
Dr. Fred Nicolls

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

Digital Image Processing
Department of Electrical Engineering

October 17, 2018

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Abstract

As homes and workplaces become increasingly automated, an efficient, inclusive and language-independent human-computer interaction mechanism will become more necessary. Isolated gesture recognition can be used to this end.

Gesture recognition is a problem of modelling temporal data. Non-temporal models can be used for gesture recognition, but require that the signals be adapted to the models. For example, the requirement of fixed-length inputs for support-vector machine classification. Hidden Markov models are probabilistic graphical models that were designed to operate on time-series data, and are sequence length invariant. However, in traditional hidden Markov modelling, models are trained via the maximum likelihood criterion and cannot perform as well as a discriminative classifier. This study employs minimum classification error training to produce a discriminative HMM classifier. The classifier is then applied to an isolated gesture recognition problem, using skeletal features.

The Montalbano gesture dataset is used to evaluate the system on the skeletal modality alone. This positions the problem as one of fine-grained dynamic gesture recognition, as the hand pose information contained in other modalities are ignored. The method achieves a highest accuracy of 87.3%, comparable to other results reported on the Montalbano dataset using discriminative non-temporal methods. The research will show that discriminative hidden Markov models can be used successfully as a solution to the problem of isolated gesture recognition.

Application for Approval of Ethics in Research (EiR) Projects
Faculty of Engineering and the Built Environment, University of Cape Town

APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data.The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook**(available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/usr/ebe/research/ethics.pdf

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Jan Hendrik Combrink
Department		Electrical Engineering
Preferred email address of applicant:		jhzcombrink@gmail.com
If a Student	Your Degree: e.g., MSc, PhD, etc.,	MSc
	Name of Supervisor (if supervised):	Dr Fred Nicolls
If this is a researchcontract, indicate the source of funding/sponsorship		~~Click here to enter text.~~
Project Title		Discriminative training of Hidden Markov Models for gesture recognition

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

SIGNED BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	JH Combrink		6 March 2017

APPLICATION APPROVED BY	Full name	Signature	Date
Supervisor (where applicable)	Dr Fred Nicolls		**06 Mar 2017**
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section1; and for all Undergraduateresearch (Including Honours).	E BOSE Click here to enter text.		6/3/17 Click here to enter a date.
Chair : Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the above questions.	Click here to enter text.		Click here to enter a date.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the world becomes more connected, homes and workplaces become more automated. An efficient, inclusive and language-independent human computer interaction (HCI) will prove very desirable.

1.1 Gesture recognition

Gestures are present in daily human activity. They participate in or facilitate communication by complementing speech, or are the primary medium such as for people with hearing disabilities. Gestures also substitute themselves to spoken language in situations requiring silent communication like noisy environments, under water or in secret communication (Escalera et al., 2017). The process of detecting and classifying gestures using computer hardware is termed gesture recognition. Applications for it are countless: HCI, communication, entertainment, security, commerce and sports. It can also have an important social impact in assistive-technologies for the handicapped or the elderly (Escalera et al., 2017).

Effective gesture detection and classification is challenging due to several factors: differences in the tempos and styles of articulation of individuals, infinitely many kinds of out-of-vocabulary motion, and real-time performance constraints (Neverova et al., 2014). Gesture recognition can be isolated or continuous. Isolated gesture recognition involves one at a time gestures that are easier to segment and can be processed on the knowledge that only one input will be required. Continuous gesture recognition is the process of recognising words or sentences from constant streams of movement and presents a much more challenging scenario. Such is the case for sign language recognition. A practical sign language recognition system should preside over a large vocabulary of words, and employ sub-unit cheremic (analogous to phonemic, but for gestures) models. Sub-unit models for sign language are still an unsolved problem (Koller et al., 2015). Isolated gesture recognition can also be a challenging problem owing to the large intra-class variability and inter-class similarity of gestures (Liang and Zheng, 2014).

The field of gesture recognition will keep benefiting from the availability and improvements of low cost depth sensors.

1.2 Generative and discriminative classifiers

The generative-discriminative dichotomy is discussed. Generative classifiers learn a model of the joint probability $P(X, Y)$, of the inputs X and the label Y. They make their predictions by using Bayes' rule to calculate $P(Y|X)$, and then picking the most likely label Y. Discriminative classifiers model the posterior $P(Y|X)$ directly, or learn a direct mapping from the inputs X to the class labels Y (Ng and Jordan, 2002).

There are benefits to the use of generative models. Since generative classifiers operate on a set of modelled distributions, adding a class to the classifier just means adding another model for the new distribution. The entire system must be retrained when a class is added to a discriminative classifier. A generative model, as the name implies, can synthesise new data by sampling from the distribution it has modelled. Ng and Jordan (2002) report that their findings support that generative classifiers need less training data to reach their asymptotic error than discriminative classifiers do.

There are several compelling reasons for using discriminative rather than generative classifiers, one of which is that the classification problem should be solved directly and not by solving a sub-problem like modelling $P(X|Y)$ (Ng and Jordan, 2002). Intuitively, discriminative classifiers should have lower asymptotic errors than generative classifiers.

1.3 Discriminative hidden Markov models

A hidden Markov model (HMM) is natively a generative model. HMMs are probabilistic graphical models that were designed to model time-series data. They have been successfully employed in a wide range of fields, most notably in speech recognition. As generative models, HMMs model the probability $P(X|Y)$ for sequences X, and HMM classifiers use Bayes' rule to pick the best corresponding label Y. The current state-of-the-art in speech recognition is still based on HMMs as phonemic models where deep learning methods have become the new paradigm for language and acoustic modelling (Saon et al., 2015), (Fohr et al., 2017).

HMMs are well suited to the problem of isolated gesture recognition as they are temporally invariant. The first-order Markov property allows for a simple temporal structure in which only the previous state and current observation is relevant for determining the system's current state. HMMs are not good at modelling state durations naturally. State duration is modelled implicitly by the self transitions of the states as an exponential distribution. The models can however be augmented to explicitly model state duration (Rabiner, 1989). This makes an HMM in its usual form unsuitable for modelling say a melody, since it will be better at modelling the series of notes than the timing. However, this idiosyncrasy makes complete sense for modelling gestures, where durations present problematic variance which most modelling techniques are not immune to. Non-temporal model based solutions to gesture recognition (such as convolutional neural networks or SVMs) require fixed length inputs. This necessitates interpolating original sequences to fixed length vectors or sampling at multiple temporal scales to represent the signal. Using HMMs to model sequences is a natural way of dealing with temporally variant data since it negates data preprocessing steps to account for varying sequence lengths.

Discriminative training of HMMs comes from the speech recognition literature. The maximum mutual information (MMI) criterion surfaced as early as in Bahl et al. (1986). There are very limited examples in the literature of discriminative HMMs being used for gesture recognition. According to Minka (2005), discriminative trained generative models are not generative models but are in fact a type of discriminative model since the parameters no longer embody the original underlying process. The discriminative training of HMMs via the minimum classification error objective turns the distribution estimation problem into one of modelling the classifier decision boundary directly. Especially in the M-class isolated recognition case, it turns an HMM from a distribution model to a discriminant function. However, the first-order Markov property still applies, and so does the new discriminant function's robustness toward state duration. Discriminative HMMs still possess the rich mathematical structures for dealing with temporally varying signals, even after they are no longer generative models. The simplest type of HMM allows for sequence length invariant function evaluation using the forward or Viterbi algorithms. It is argued here that for the problem of gesture recognition, discriminative HMMs offer the performance benefit of a discriminative classifier while preserving the benefit of sequence length invariance.

1.4 Dynamic gesture recognition on skeletal data

Dynamic gesture recognition is a sub-field that considers dynamic movements as opposed to poses. The recognition of finger-spelling alphabet letters is an example of a static gesture recognition problem where the information is contained in hand poses. Dynamic gesture recognition is generally concerned with larger-scale body movements. The Montalbano data set is a large commercial depth-sensor based gesture recognition data set, and contains gestures with mixtures of dynamic and pose components. The data set was designed and used for the Chalearn looking at people (LAP) 2013 and 2014 challenges (Escalera et al., 2013) (Escalera et al., 2014). The competitions pose different problems of multi-modal gesture recognition on twenty Italian anthropological signs. The gestures contain fine hand articulation so as to require RGB, depth, user-segmented depth and skeleton model frames. However, even the gestures with fine hand articulations have some dynamic component represented in the skeletal modality. The dataset thus offers a unique test case for fine-grained dynamic gesture recognition. This study uses the Montalbano dataset to evaluate an HMM classifier for fine-grained dynamic gesture recognition on the skeletal modality. Findings suggest that maximum likelihood (ML) HMMs are troubled with the similarity of some classes on this problem, and that discriminative training offers a good solution. Using non-temporal discriminative methods, Liang and Zheng (2014) present the following classification results on the segmented Montalbano test set: 92.80% accuracy using all modalities and 83.02% as the highest accuracy using only the skeletal modality. Evangelidis et al. (2014a) report a 90% classification accuracy on the segmented test set using the skeleton modality alone. Using only the skeletal modality this study achieves as a highest accuracy 87.3% as seen in chapter 5.

1.5 Objectives of the study

This study investigates discriminative training of HMMs via the minimum classification error (MCE) criterion as a solution to isolated gesture recognition problems such as a simple HCI system that uses skeletal features. Particularly, it is concerned with the case where the skeletal features are at their carrying capacity in terms of representing the gestures. The study however assumes that the gestures are already segmented. For the sequence modelling part of the system, the study advocates the use of discriminative HMMs. The study aims to:

- Show that MCE-HMMs can be used successfully for isolated gesture recognition,
- Provide useful findings for the implementation of a MCE-HMM gesture recognition system.

1.6 Contributions

There are various preliminary problems to solve in the design of an HMM-based discriminative gesture classifier. Problems such as finding good initialisations and model selection limit the performance that can be harnessed from an HMM classifier. The study highlights important underlying problems in both the gesture modelling and optimisation parts of the design. Modelling suggestions tend to be domain specific whereas optimisation suggestions tend to be general. This study provides optimisation recommendations that stem from the modelling of particularly long sequences found in gestures sampled at twenty frames per second (FPS). It is the intention of the study to offer a thorough treatment of MCE-HMM gesture classifier design.

1.7 Overview of the work

Chapter 2 lays out the relevant literature for the study but is intended to be read as an argument for discriminative instead of ML HMMs as a solution to isolated recognition problems. The chapter starts off with a treatment of the history and current status of HMMs in the gesture recognition literature, and then details the technical backdrop to discriminative HMMs. In chapter 3 the implementation details for the discriminative classifier design are elucidated. Common problems that cannot be overlooked are discussed along with their suggested solutions. Thereafter particular problems with the discriminative optimisation are discussed, along with the solutions employed in the study. Chapter 4 discusses the set up for deploying the implementation onto the Montalbano gesture dataset. First the specifications of the dataset are discussed and choices relating thereto. Then, other work on the gesture dataset is discussed along with the results that can be directly compared to the results from this study. Then more general gesture modelling topics are discussed such as the feature extraction, filtering and decorrelation. The chapter presents the benchmark performance results for the ML manifestation of the gesture classifier. In chapter 5 the experiments conducted on MCE training and the resulting discriminative HMMs are laid out. Finally the conclusions are presented in chapter 6. The appendix provides detailed information on the analytical gradients as well as derivations for some of the underlying formulae.

Chapter 2

Literature

This chapter details the literature on HMMs and discriminative training. It is intended to serve as an argument for the use of discriminative HMMs over ML HMMs. First, the history and current status of HMMs in gesture recognition are discussed. Then the technical backdrop to discriminative HMMs is defined from the relevant literature.

2.1 History and current status of hidden Markov models in gesture recognition

According to Juang and Rabiner (2005), the idea of hidden Markov models seems to have originated in the late 1960's at the Institute for Defence Analysis (IDA) in Princeton New Jersey. Baum (1972) referred to an HMM as a set of probabilistic functions of a Markov process. This implies two nested distributions, one pertaining to the Markov chain, and the other to the set of distributions associated with the states of the Markov chain (Juang and Rabiner, 2005). This doubly stochastic process was found to be useful in stock-market prediction, crypto-analysis of a rotary cipher (which was widely used during the second World War), and in the late 1970's, speaker identification systems (Juang and Rabiner, 2005). In the 1980's the field of speech recognition had undergone a change in methodology from a template-based pattern recognition paradigm, to a more rigorous statistical framework based on HMMs. Although HMMs were known and understood early on in laboratories such as IBM and the IDA, the methodologies were not complete until the mid 1980's, and only after widespread publication of the theory did they become the preferred method for speech recognition (Juang and Rabiner, 2005).

According to Mitra and Acharya (2007), probably the first publication to address the problem of hand gesture recognition using HMMs is the celebrated paper by Yamato et al. (1992). The authors used vector quantisation and discrete HMMs to recognise six classes of tennis strokes from image sequences. The image sequences underwent low-pass filtering to remove noise and background subtracted to extract the moving objects. Blobs that roughly represent the human poses were formed by transforming the extracted moving object images to binary images. A feature vector was then formed by applying a grid to the blob images and finding the black pixel ratio in each cell. The ratios of the cells made up the elements of the vector. These feature vectors were then quantised into symbols to be processed by discrete HMMs. In the isolated recognition of three individuals with three-hundred test sequences per individual, their system achieved an accuracy of 96%.

In Starner (1995) an HMM-system is described that recognises sentence level American Sign Language (ASL) based on a forty-word lexicon. Signs were modelled by HMMs on the word level where strong rules of context and grammar were used to make the recognition tractable. Viterbi decoding was used with and without a strong grammar, based on the known forms of the sentences (Mitra and Acharya, 2007). The user sits on a chair in front of a camera wearing distinctly coloured gloves to facilitate stable hand tracking. Instead of attempting to extract fine hand shape features, the tracking process produces only a coarse description of the hand shape, orientation, and trajectory. The pixel coordinates, angle of axis of least inertia, and the eccentricity of the bounding ellipse of each hand formed the feature vector fed to the HMMs (Starner, 1995). The system achieves a word accuracy of 97% using a strong grammar, and a word accuracy of 91% using no grammar. These results were later improved upon by Starner and Pentland (1997) to a word accuracy of 99.2% using the strong grammar and 91.3% using no grammar.

Campbell et al. (1996) conducted an investigation into invariant features for gesture recognition. The authors compared the recognition performance of different feature sets on sequences of continuous T'ai Chi using HMMs. The dataset comprised 108 six-gesture "sentences" based on eighteen T'ai Chi moves. The study utilized a stereo vision system which produces three-dimensional world coordinates of the head and hand positions to a $2cm$ approximate accuracy at up to 30 FPS. Several feature vectors derived from this data were input to an HMM recognition system (Campbell et al., 1996). All "sentences" were performed by one individual. The performer, seated in a swivel chair, performed the movements with the upper body and hands. Each sequence begins with the hands in a rest position. From there the six gestures are performed in a flowing manner characteristic of T'ai Chi. The hands then return to the rest position (Campbell et al., 1996). Two thirds of the dataset had the performer sitting in the same position and orientation; this was called the "original" set. A "shifted" set (constituting a sixth of the dataset) was performed with a translational shift of 14 inches from from the original position. A further "rotated" set (the remaining sixth) was performed with both an 8 inch shift and 45 degree rotation. Training and testing sets were divided half-half. The training set sequences were collected randomly from the three groups, where the testing set kept them separated. This allowed performance to be measured individually for shifted and rotated data (Campbell et al., 1996). Using velocities of the points in a cylindrical-coordinate system as features, a highest recognition rate of 95%, averaged over original, shift, and rotate test sets was achieved.

In Liang and Ouhyoung (1997) and Liang and Ouhyoung (1998) a real-time user-dependent continuous sign language recognition system is described that uses a sensor-glove and presides over a vocabulary of 250 words. The system uses HMMs to model fifty-one fundamental hand postures, six orientations, and eight motion primitives that are based on semantic analysis of Taiwanese Sign Language (Liang and Ouhyoung, 1998). The hand postures are found from flex sensors corresponding to ten finger joints on the glove. Azimuth angle, elevation and roll form the orientation features that are calculated from a three-dimensional tracker embedded in the glove. Lastly, motion trajectories are also extracted from the tracker. The critical problem of end point detection is first solved using time-varying parameter discontinuity detection. Several gesture-level hypotheses are then formed using the different modalities along with information from the language model. After that the most likely sentence is selected from sentence-level hypotheses. The average recognition rate for continuous sign language was 80.4%.

Starner et al. (1998) presented two real-time HMM-based continuous ASL recognition systems that use only a colour camera to track unadorned hands. Both systems preside over a forty word lexicon. The first system observes the user from a desk mounting, and achieves a 92% word accuracy. The second system has the camera mounted in a hat that is worn by the user and achieves 98% word accuracy. As in Starner (1995), the feature extraction does not attempt a fine description of hand shape, instead it focuses on the larger spatio-temporal evolution of the gesture. Vogler and Metaxas (1998) presented a methodology for recognizing isolated and continuous ASL based on HMMs. Using three orthogonally placed cameras and sophisticated physics-based vision algorithms to extract three-dimensional features, the system achieves a (continuous) word accuracy of 87.71% over a fifty-three sign lexicon. Furthermore, using a context-dependent HMM modelling to learn from the co-articulation of signs, the system achieves a 89.91% word accuracy.

From the early 2000's onwards, an abundance of publications dealing with work done on HMM gesture recognition can be found in the literature. Assessing the current status of the HMM as a gesture recognition tool is a complicated endeavour. By looking at competitive gesture recognition events and their associated publications, one can get a sense of what methods researchers deem most competitive. Work done on the ChaLearn Looking at People 2013, 2014 and 2017 challenges reflect many of the currently popular techniques employed by the gesture recognition community. Section 4.2 provides a discussion of some publications and general trends on the 2013 and 2014 competitions. The 2013 competition saw the HMM as the most widely applied classifier among contestants, whereas for the 2014 competition it made up less than a third of preferred methods. The 2014 competition had SVMs and different ensemble methods widely applied. These choices of classifiers also say something interesting about the rest of the methodologies put towards a solution for gesture spotting and recognition. It means that fewer contestants put their faith in Viterbi decoding over an HMM network (or using the level-building algorithm) for simultaneous segmentation and recognition. Instead, strategies are used such as interval sub-sampling to capture temporal evolution in a sliding window(s), and then passing it as input to a non-sequence classifier for per frame classifications. The latter method seems much more of a lower-level approach to the problem. This of course has the benefit of being simpler and, possibly, leaves room to focus on a discriminative feature representation. However if the number of classes were to increase, the temporal distribution of the sequence itself will be, to a larger degree than present, a discriminative feature. This might give sequence classifiers the upper hand. Wan et al. (2017) provides the results and analyses of the 2017 ChaLearn Looking at people challenge. Track one and two represent large-scale isolated and continuous gesture recognition challenges respectively. The methodologies employed by the contestants for the isolated and continuous tracks were predominantly deep learning strategies, some of which used recurrent neural networks for sequence learning (Wan et al., 2017).

2.2 The hidden Markov model as a Bayesian network

Bayesian networks are graphical models for representing the conditional independences between sets of random variables. They offer a graphical way of representing a particular factorisation of a joint distribution (Ghahramani, 2001). For HMMs, the Bayesian network framework serves as a compact unified way of listing the assumptions that make the basic problems, evaluation, decoding and learning, tractable. Figure 2.1 shows that the observations are governed by an underlying sequence of discrete random variables (denoted as squares). These random variables are called the states of the system and capture the transient information, while the observations are sampled from distributions that belong to these respective states. The figure also illustrates the first-order Markov property, by which the next state of the system is only determined by the current state and, given the current state, is conditionally independent of all previous states. This allows the factorisation of the joint probability of the state sequence $\boldsymbol{q}$ that generated the observation sequence X:

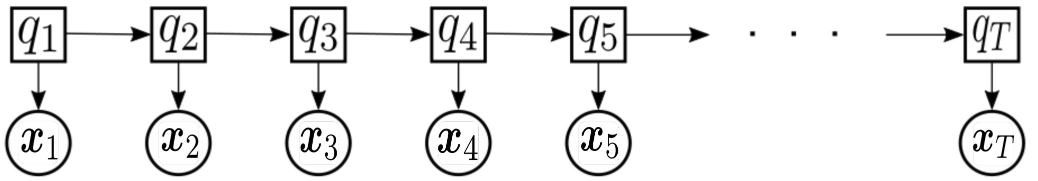


Figure 2.1: *an HMM represented as a Bayesian network. It illustrates the forward temporal probabilistic structure and conditional independence of the observations.*

$$P(\boldsymbol{q}|\lambda) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}, \lambda). \tag{2.1}$$

Also shown in figure 2.1 is that the observations themselves are independent, given the state sequence from which they were generated:

$$P(X|\boldsymbol{q}, \lambda) = \prod_{t=1}^{T} P(\boldsymbol{x}_t|q_t, \lambda). \tag{2.2}$$

Thus, the joint distribution of the observations and the corresponding state sequence, given the model, is factorised as (Ghahramani, 2001)

$$P(X, \boldsymbol{q}|\lambda) = P(X|\boldsymbol{q}, \lambda)P(\boldsymbol{q}|\lambda) = P(q_1|\lambda)P(\boldsymbol{x}_1|q_1, \lambda)\prod_{t=2}^{T} P(q_t|q_{t-1}, \lambda)P(\boldsymbol{x}_t|q_t, \lambda). \tag{2.3}$$

Given the complete HMM parameter set $\lambda = \{\pi, A, \{b(\boldsymbol{x})\}_{i=1}^{N}\}$, it can be restated as

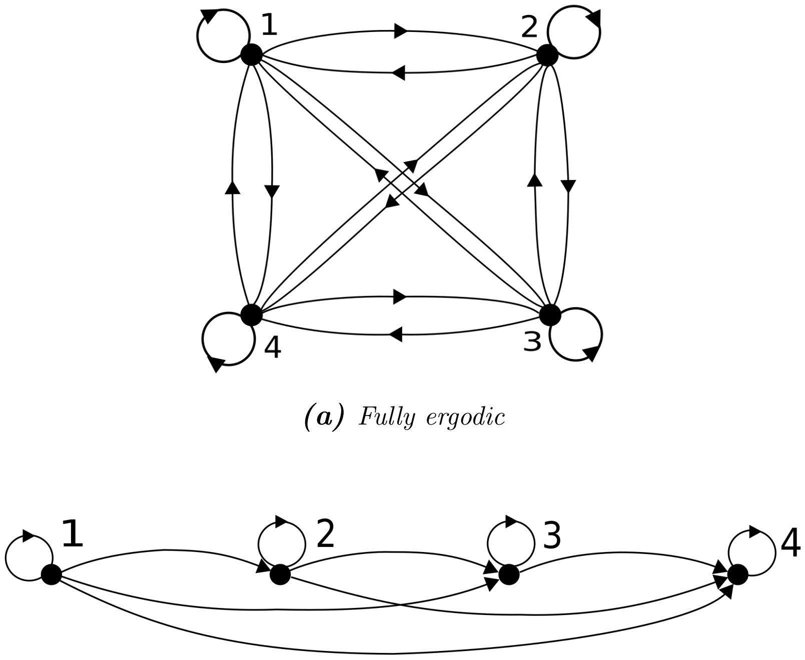
$$P(X, \boldsymbol{q}|\lambda) = \pi_{q_1} b_{q_1}(\boldsymbol{x}_1)\prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(\boldsymbol{x}_t). \tag{2.4}$$

Finally to find the probability of the observation sequence given the model, the joint probability is marginalised over all possible state sequences $\boldsymbol{q}$ of length T:

$$P(X|\lambda) = \sum_{\boldsymbol{q}} P(X|\boldsymbol{q}, \lambda)P(\boldsymbol{q}|\lambda) = \sum_{\boldsymbol{q}} \pi_{q_1} b_{q_1}(\boldsymbol{x}_1)\prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(\boldsymbol{x}_t). \tag{2.5}$$

2.3 Structure

A N-state HMM consists of an initial state distribution vector π, a state transition probability matrix A and the parameter set of its output distributions $\{b_i(\boldsymbol{x})\}_{i=1}^{N}$. The output distribution can be multinomial (discrete), Gaussian, Gaussian mixtures or a neural network (Ghahramani, 2001). Figure 2.2 shows the transition diagrams of some well-known HMM topologies. Figure 2.2(a) shows the fully ergodic model in which any state can be transitioned to from any other state in a single step. This is appropriate for modelling signals known to be cyclic. Figure 2.2(b) shows the transitional structure of a left-to-right HMM. In a LR-HMM, as time increases the state either transitions to a higher state index or remains in its current state. This makes the transition matrix upper triangular with $a_{ij} = 0$ if $j < i$ (Rabiner, 1989). This configuration is more appropriate for modelling acyclic sequences that have some trajectory from beginning to end, like speech or gesture data. It is important to note that the transition diagrams are graphical representations of the transition matrix and should not be confused with the graph in figure 2.1, which depicts the model's statistical independence assumptions. The transitional structure can further be constrained by limiting jumps to higher state indices by some predefined value Δ, in other words, the model cannot skip more than $\Delta - 1$ states in any transition. This makes for an upper triangular transition matrix with Δ diagonals as $a_{ij} > 0$ if $i \leq j \leq i + \Delta$. Models with this kind of transitional structure are called left-right-banded (LRB) models. The LRB structure acts as a regularisation since, for appropriate acyclic data, a banded-looking transition matrix will form naturally by the zeroing out of infrequent transitions during training. By forcing a banded structure a similar distribution is formed with fewer parameters, which generalises better. This study uses LRB models with $N/2$ bands.



(a) Fully ergodic

(b) Left-to-right

Figure 2.2: *Transition diagrams for well known model topologies. Adapted from (Rabiner, 1989).*

2.4 The MAP classifier decision rule

Given a set of M competing models $\Lambda = \{\lambda_i\}_{i=1}^{M}$, and an observation sequence X to be classified, the classifier decision is defined by

$$C(X) = C_i \quad \text{if} \quad P(\lambda_i|X) = \arg\max_j P(\lambda_j|X), \tag{2.6}$$

where the posterior probability is found from Bayes' rule:

$$P(\lambda_i|X) = \frac{P(X|\lambda_i)P(\lambda_i)}{P(X)}. \tag{2.7}$$

Given a uniform prior distribution over the model classes and noting that $P(X)$ is constant, the decision rule reduces to

$$C(X) = C_i \quad \text{if} \quad P(X|\lambda_i) = \max_j P(X|\lambda_j). \tag{2.8}$$

2.5 Model selection

The basic philosophy of HMMs is that an observation sequence can be well modelled if the parameters of the HMM are carefully and correctly chosen. The problem with this philosophy is that it is sometimes inaccurate. There is no guarantee that the signal, sampled from the real underlying distribution, will obey the constraints imposed by the model parameters (Rabiner, 1989). The models have to be selected a priori and then trained on finite data. Overfitting is prevalent when the training set is small relative to the complexity of the model. Overfitting refers to the scenario where a model fits the training set very well but generalises poorly to unseen data from the same distribution (Ghahramani, 2001). There exists however some optimal model in structure and parameter set size. This model will relate closest to the actual data distribution given the modelling assumptions stated in section 2.2. Model selection or learning model structure is the problem of selecting a particular model structure from several alternatives. For an HMM, the 'structure' would include everything from the number of states to the topology and the choices regarding output distributions.

A principled way of addressing overfitting is by means of a hyper-parameter selection procedure. The traditional way of model selection is known as grid search or parameter sweep. It is done by an exhaustive search through a manually specified subset of the hyper-parameter space, guided by a performance metric (Hsu et al., 2003). Performance metrics such as cross-validation (or evaluation on a held-out validation set if enough data is available) provides an estimate of the true generalisation error, however it is only computationally practical for determining a few hyper-parameters (Ghahramani, 2001). This study employs a grid search strategy for model selection as discussed in section 3.1.3.

2.6 Discriminative training

So far HMM classifier design was discussed in terms of the maximum likelihood methodology. The goal is to capture the true distribution for each modelled class, and accordingly, parameter estimation happens on a per-model basis. However, as previously mentioned, the models aren't perfect and make assumptions on the distributions they model for tractability. More importantly, a classifier based on maximum likelihood is oblivious to distributional overlap. If it so happens that for any two classes the features that are to be modelled are very similar, the particular differences won't necessarily be learned via the model parameters and will result in misclassification. That said, a validated or cross-validated likelihood score is a good metric for how well the particulars of a true distribution are modelled. Model selection can optimise an estimate of the true distribution but is in general either computationally expensive or hard to implement.

A practical design would thus be one based on discriminative training. The goal is to optimise the performance of the classifier directly over the entire parameter set. It uses the training data to learn HMM parameters that optimally separate the modelled classes by their likelihoods. The two main discriminative criteria for HMM training are maximum mutual information (Bahl et al., 1986) and minimum classification error (MCE) (Chou et al., 1992) (Juang et al., 1997). The following sections discusses the MCE criterion, first from the function definition and then from the perspective of optimisation.

2.6.1 Objective functions and optimisation

Discriminative training is guided and controlled by an objective function. It is a scalar function $f(\Lambda)$ of the HMM classifier parameter set Λ evaluated on the training data. Objective functions are useful because they express in a compact form the goal of the particular training technique and are suitable for optimisation. For an M-class problem with a training set $\mathcal{D} = \{\{X_i^l\}_{l=1}^{L_i}\}_{i=1}^{M}$, with L_i samples per class, the maximum likelihood objective as discussed in the preceding sections is expressed in functional form as

$$f_{ML}(\lambda_i) = \sum_{l=1}^{L_i} logP(X_i^l|\lambda_i). \tag{2.9}$$

The function aims to maximise the posterior $P(\lambda|X)$ by approximating $P(X|\lambda)$ as closely as possible. Given a model, the function is traditionally optimised using the Baum-Welch (BW) algorithm, but a very efficient alternative based on the most likely path is the segmental k-means (SKM) algorithm (Juang and Rabiner, 1990).

Discriminative training of HMMs is a harder optimisation problem than training them for maximum likelihood. There is also no update rule as convenient and provably correct as the Baum-Welch algorithm for discriminative criteria, and accordingly early work on discriminative training used gradient descent for optimisation (Povey, 2005). The two most well-known discriminative criteria for HMM estimation, MMI and MCE, brought about the evolution of their own corresponding, well-established optimisation algorithms.

2.6.2 The minimum classification error objective function

The minimum classification error objective aims to change classifier design from a distribution estimation problem into one of estimating parameters for minimal error directly. The difficulty associated with such a training approach lies in the derivation of an objective function that has to be consistent with the performance measure and also suitable for optimization (Juang et al., 1997). MCE estimation is a discriminant function based approach to pattern recognition, where the classifier decision is treated as comparisons among a set of discriminant functions and where parameters are estimated to minimise the expected error over the training data when these decisions are applied (He et al., 2008). The MCE objective function was originally defined for isolated-word speech recognition, where each utterance can belong to one of a fixed number M classes (Povey, 2005). The loss function is constructed in such a way that the recognition error rate of the classifier is represented as a smooth, differentiable function. Minimising the MCE function has a direct relation to classifier error rate reduction (He et al., 2008). The MCE objective function is constructed from likelihood-based generative models as described below.

The MCE criterion defines a class conditional likelihood function on the entire classifier parameter set Λ:

$$g_i(X, \Lambda) = \log P(X|\lambda_i). \tag{2.10}$$

The classifier decision in (2.8) can then be restated as

$$C(X) = C_i \quad \text{if} \quad g_i(X, \Lambda) = \max_j g_j(X, \Lambda). \tag{2.11}$$

The key to the error criterion is to express the operational decision rule above in a smooth functional form (Juang et al., 1997). If an error count function is defined for the class i such that

$$e_i(X, \Lambda) = \begin{cases} 1 & X \in C_i \quad \text{and} \quad g_i(X, \Lambda) \neq \max_j g_j(X, \Lambda) \\ 0 & \text{otherwise,} \end{cases} \tag{2.12}$$

and this function is averaged over a data set to find an empirical estimate of the classifier error, the estimate would be a piecewise constant function of the classifier parameters and a poor candidate for optimization (Juang et al., 1997). Thus the function cannot merely count the incorrect classifications over the data, it must contain a measure of correctness so as to have the error be represented over a continuous interval of the parameter space. Furthermore, besides optimisation, the inability of such an empirical error rate function to distinguish near miss and barely correct cases will most likely impair the performance of the recognizer on independent data (Chou et al., 1992). Using the definition of $g_i(X, \Lambda)$ as a probabilistic distance, a misclassification measure is defined as

$$d_i(X, \Lambda) = -g_i(X, \Lambda) + log \left[\frac{1}{M-1} \sum_{j \neq i} e^{g_j(X, \Lambda)\eta} \right]^{\frac{1}{\eta}}, \tag{2.13}$$

which not only indicates correct or incorrect classification but assigns a measure of distance for how correct or incorrect a classification is. This misclassification measure is a continuous function of the classifier parameters that attempts to emulate the classifier decision rule (Juang et al., 1997). For an observation sequence $X \in C_i$, a value of $d_i(X, \Lambda) > 0$ indicates misclassification and $d_i(X, \Lambda) \leq 0$ a correct decision. The bracketed term in the misclassification function is a L_η-norm approximation to the highest competing class likelihood $\max_{j \neq i} P(X|\lambda_j) = \|P(X|\lambda_j)\|_\infty$ as $\eta \to \infty$. To complete the definition, the misclassification measure with its large unbounded dynamic range is embedded in the logistic function

$$\ell_i(X, \Lambda) = \frac{1}{1 + e^{-\gamma d_i(X, \Lambda)}}. \tag{2.14}$$

Any member of the sigmoid family will be a good candidate. However, the logistic function squeezes the aforementioned dynamic range into a zero-one interval that neatly approximates the classifier error count in (2.12). The parameter γ controls the centre slope of the logistic function. On a random training sample X, the performance of the classifier Λ is thus measured by

$$\ell(X, \Lambda) = \sum_{i=1}^{M} \ell_i(X) \delta(X \in C_i), \tag{2.15}$$

where δ is an indicator function. For a classification problem with M different classes, the expected loss of the classifier Λ is defined as

$$L(\Lambda) = \sum_{i=1}^{M} \int_{X \in C_i} \ell_i(X, \Lambda) P(X) dX \tag{2.16}$$

$$= E_X[\ell(X, \Lambda)]. \tag{2.17}$$

Minimising $L(\Lambda)$ is the basis for using the generalised probabilistic descent method described in section 2.6.4. To evaluate the performance of the classifier on a data set of P samples, (2.15) is averaged over the data,

$$L_P(\Lambda) = \frac{1}{P} \sum_{i=1}^{M} \sum_{l=1}^{L_i} \ell_i(X_i^l, \Lambda). \tag{2.18}$$

For simplicity the data set is divided per class. The empirical loss of the classifier $L_P(\Lambda)$ can be directly optimised with gradient descent or a second order method. However, doing so will be computationally expensive for complex models or large data sets since in each iteration the gradients have to be accumulated over all of the data for each parameter. The empirical loss, defined on the P independent samples, will converge to the expected loss as the data set size $P \to \infty$ (Juang et al., 1997).

2.6.3 The relation of MCE to other discriminative criteria

Besides the obvious conclusion that both MMI and MCE objectives will improve the performance of a classifier Λ, it can be shown that they are in fact mathematically very similar. The uniform prior distribution isolated recognition case is considered. Given the settings of the MCE function parameters $M = 2$, $\eta = 1$ and $\gamma = 1$, (2.13) is restated as

$$d_i(X, \Lambda) = -\log P(X|\lambda_i) + \log \sum_{j \neq i} P(X|\lambda_j). \tag{2.19}$$

With this distance measure substituted in (2.14), the loss for the training sequence X is rewritten as

$$\ell_i(d_i(X, \Lambda)) = \frac{1}{1 + \frac{P(X|\lambda_i)}{\sum_{j \neq i} P(X|\lambda_j)}} \tag{2.20}$$

$$= \frac{\sum_{j \neq i} P(X|\lambda_j)}{\sum_{j \neq i} P(X|\lambda_j) + P(X|\lambda_i)} \tag{2.21}$$

$$= \frac{\sum_{j \neq i} P(X|\lambda_j)}{\sum_j P(X|\lambda_j)}. \tag{2.22}$$

From the loss function, a corresponding utility function can be defined as

$$u_i(d_i(X, \Lambda)) = 1 - \ell_i(d_i(X, \Lambda)) \tag{2.23}$$

$$= 1 - \frac{\sum_{j \neq i} P(X|\lambda_j)}{\sum_j P(X|\lambda_j)} \tag{2.24}$$

$$= \frac{P(X|\lambda_i)}{\sum_j P(X|\lambda_j)}. \tag{2.25}$$

The MCE utility evaluates to the same posterior probability as the MMI function for a single training sequence X. The important difference is that the individual MMI functions of X are multiplied to form the aggregate over the dataset, whereas for MCE loss, or utility, they are summed (He et al., 2008).

2.6.4 Generalised probabilistic descent optimisation

The generalised probabilistic descent (GPD) algorithm minimises the expected loss of the classifier Λ, defined in (2.16). The update is essentially a preconditioned stochastic gradient descent (SGD) operation

$$\tilde{\Lambda}_{n+1} = \tilde{\Lambda}_n - \epsilon_n U_n \nabla \ell(X_n, \Lambda)|_{\Lambda=\Lambda_n} \tag{2.26}$$

in a transformed parameter space, where ϵ_n is a sequence of positive numbers known as the learning rate and U_n a positive definite preconditioning matrix. The gradient vector $\nabla \ell(X, \Lambda_n)|_{\Lambda=\Lambda_n}$ is calculated w.r.t the transformed parameters of the loss function (2.15) evaluated on a random sample X at time n. The set of parameter transformations $\mathcal{T}(\Lambda) = \tilde{\Lambda}$ maintains the original constraints on the HMM parameters. Thus for minimising a function of HMMs, SGD as an unconstrained optimisation technique can be used. The following transformations apply to HMMs with Gaussian mixture output distributions:

$$a_{ij} \to \tilde{a}_{ij} \qquad a_{ij} = \frac{e^{\tilde{a}_{ij}}}{\sum_{j=1}^{N} e^{\tilde{a}_{ij}}} \tag{2.27}$$

$$c_{jk} \to \tilde{c}_{jk} \qquad c_{jk} = \frac{e^{\tilde{c}_{jk}}}{\sum_{k=1}^{K} e^{\tilde{c}_{jk}}} \tag{2.28}$$

$$\mu_{jkd} \to \tilde{\mu}_{jkd} \qquad \tilde{\mu}_{jkd} = \frac{\mu_{jkd}}{\sigma_{jkd}} \tag{2.29}$$

$$\sigma_{jkd} \to \tilde{\sigma}_{jkd} \qquad \tilde{\sigma}_{jkd} = \log \sigma_{jkd}. \tag{2.30}$$

Notice that for the sum-to-one constraints, only transformations out of the new parameter space can be specified, whereas the last two transformations are reversible. The inverse transforms are thus used to update the parameters using gradients calculated with respect to the transformed parameters, in terms of the original parameters. For clarity, the transition probability update of the i'th model $a_{ij}^{(i)} \in \lambda_i$ of the classifier is shown below. First the step in the separate parameter space is found,

$$\Delta\tilde{a}_{ij}^{(i)} = \tilde{a}_{ij\,\{n+1\}}^{(i)} - \tilde{a}_{ij\,\{n\}}^{(i)} = -\epsilon_n \frac{\partial \ell_i(X_n, \Lambda)}{\partial \tilde{a}_{ij}^{(i)}}\bigg|_{\tilde{a}_{ij}^{(i)}=\tilde{a}_{ij\,\{n\}}^{(i)}}, \tag{2.31}$$

and then substituted into the inverse transform

$$a_{ij\,\{n+1\}}^{(i)} = \frac{a_{ij\,\{n\}}^{(i)} e^{\Delta\tilde{a}_{ij}^{(i)}}}{\sum_{j=1}^{N} a_{ij\,\{n\}}^{(i)} e^{\Delta\tilde{a}_{ij}^{(i)}}}. \tag{2.32}$$

This produces an update for $a_{ij}^{(i)}$ that can be calculated entirely from the current non-transformed parameters Λ.

Chapter 3

Implementation

This chapter discusses the implementation of the HMM classifier used in the study. First, general problems encountered with HMM classifiers are elaborated upon, along with the solutions employed in the study. Thereafter useful recommendations for the discriminative optimisation procedure presented in section 2.6.4 are detailed, mostly from the neural networks literature. For illustration, results are presented below on real or toy problems all of which are based on the Montalbano gesture dataset. Details on the dataset are discussed in chapter 4.

3.1 Solutions to general problems

This section details general problems to modelling with HMMs and solutions employed in the study. The results from this section are the actual solutions used for the classifier design for the twenty-class Italian cultural gesture classifier. Every gesture is modelled with a single HMM.

3.1.1 Initialization

According to Rabiner (1989), initialisation is crucial for finding a good solution during ML training of HMMs. In this study, it was found that the most important parameter subset to initialise was the mean vectors of the output distributions. Two initialisation heuristics are discussed, flat-initialisation and averages along sequences. In flat initialisation the global mean and variance are first found from the training data set, and then used to initialise every Gaussian component of every output distribution in the model. This study uses an improvised method of binning all of the training sequences into N (the number of model states) bins. The average of each bin is then used to initialise the mean vectors of the corresponding output distribution. The study employs both the segmental k-means and Baum-Welch algorithms for ML training. It was found that segmental k-means is more robust to initialisation. Figure 3.1 shows the relative magnitudes of the converged SKM pre-training likelihoods for flat and binned initialisation. Twelve out of twenty bar-pairs indicate higher convergence likelihoods for binned initialisation.

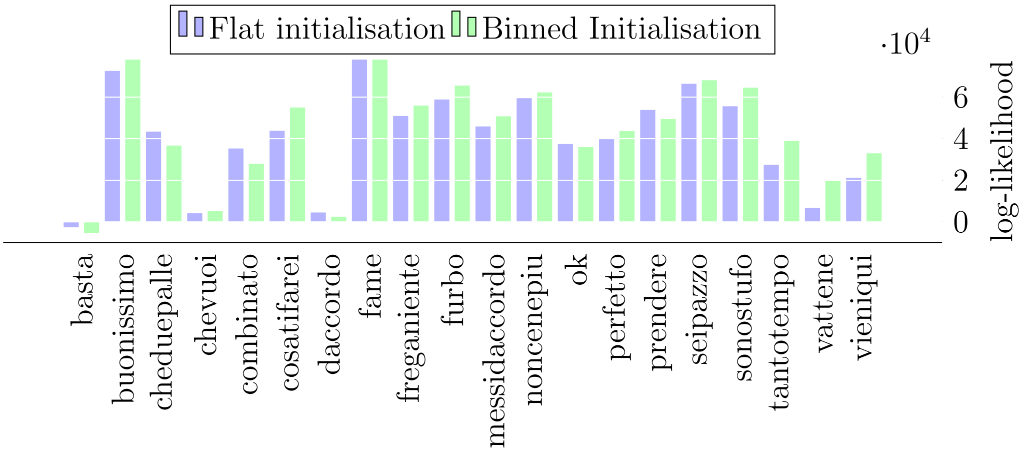


Figure 3.1: *A bar plot showing the relative convergence likelihoods for a SKM pre-training of the classifier, for flat- and binned initialisation respectively.*

3.1.2 Local optima

To curb the effects of local optima in the ML parameter space, a combination of the segmental k-means and Baum-Welch algorithms was used. SKM was found to be more robust to initialisation than the BW algorithm, although the BW algorithm on average reaches higher asymptotic maxima. Figure 3.2 shows the training curves of a BW, SKM and SKM-BW optimisation. The BW optimisation (displayed in red) seems to get stuck in a local optimum while the SKM likelihood (in green) increases. The blue curve represents a SKM-BW pair which is identical to SKM up to a point and then uses BW to arrive at a final optimum. This study uses the SKM algorithm for pre-training and the BW algorithm for further training as it ensures that all parameters are changed. In order to lessen the gravity of local optima, data can

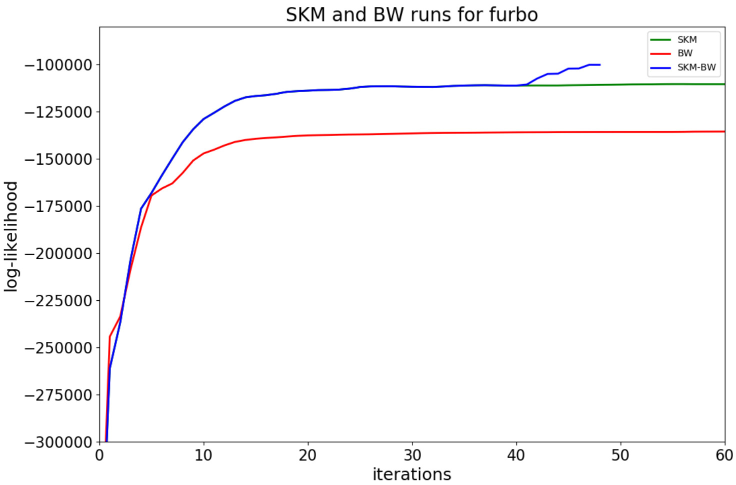


Figure 3.2: *Baum-Welch and segmental k-means training on the same data and initialisation for the gesture "furbo".*

be centred so as to have a zero mean, and scaled so as to have unit variance in all dimensions. This is called data standardisation and will improve the optimisation surface in the parameter space. Section 3.2.1 details how standardisation was applied in MCE training. If the data is also decorrelated, all of the data points will be distributed in the shape of a hypersphere centred on the origin. Principal component analysis (PCA) whitening can be used to this end, and is evaluated in terms of MCE training in chapter 5.

3.1.3 Model selection

For finding an optimal set of models for the ML classifier, a grid search strategy is implemented. Here a two-dimensional space of hyper-parameters is exhaustively searched for the optimal pair. The whole number interval $[6, 20]$ for N, the number of states, and the whole number interval $[5, 20]$ for K, the number of mixtures in the output distributions, form the space to be searched. For every combination of hyper-parameters, a model is trained using the procedure outlined in section 3.1.2, and evaluated on a held-out validation set as explained in section 3.1.4. The model with the highest validation likelihood is then stored. This is done on a per-class basis and can be run separately on multiple machines to save time. All grid searches were performed using the University of Cape Town ICTS HPC Cluster (`http://hpc.uct.ac.za`). Table 3.1 compares a grid searched model set with a high-complexity model set and shows that for each class the grid searched model generalises better. The high-complexity model set is likely over-fit on the training data. Higher validation set performance translates to higher test set performance.

Table 3.1: *A table showing the validation likelihoods of a large uniform model set and that of a grid searched set.*

Name	N	K	Validation likelihood	N	K	Validation likelihood
basta	20	30	33306.6	19	29	34076.9
buonissimo	20	30	50752.3	9	16	52629.6
cheduepalle	20	30	48853.7	20	29	49429.3
chevuoi	20	30	47276.3	15	27	53453.3
combinato	20	30	62224.9	14	21	69154.7
cosatifarei	20	30	39225.9	11	22	43492.7
daccordo	20	30	48067	13	28	48768.5
fame	20	30	61049.2	16	25	65329.7
freganiente	20	30	47576.1	18	29	49778.5
furbo	20	30	49096.2	20	22	51186.7
messidaccordo	20	30	55487.6	20	27	56281.6
noncenepiu	20	30	40521.7	18	21	44892.1
ok	20	30	43382.4	18	17	43667.2
perfetto	20	30	41641.3	16	17	44665.9
prendere	20	30	42636.9	11	28	47243.3
seipazzo	20	30	59236.7	16	28	60977.6
sonostufo	20	30	37163.2	13	29	46738.3
tantotempo	20	30	38538.9	20	27	39166
vattene	20	30	38635.5	20	26	39403.2
vieniqui	20	30	35636.5	8	29	38390.3

3.1.4 Over-training

The Left-Right Banded Structure

This study only uses left-right banded HMMs (see section 2.3) with the number of bands fixed at $N/2$. Figure 3.3 shows the relative magnitudes of the validation likelihoods for each class given its LR, and LRB modelling. Sixteen out of twenty bar-pairs indicate that the LRB structure generalises better than LR.

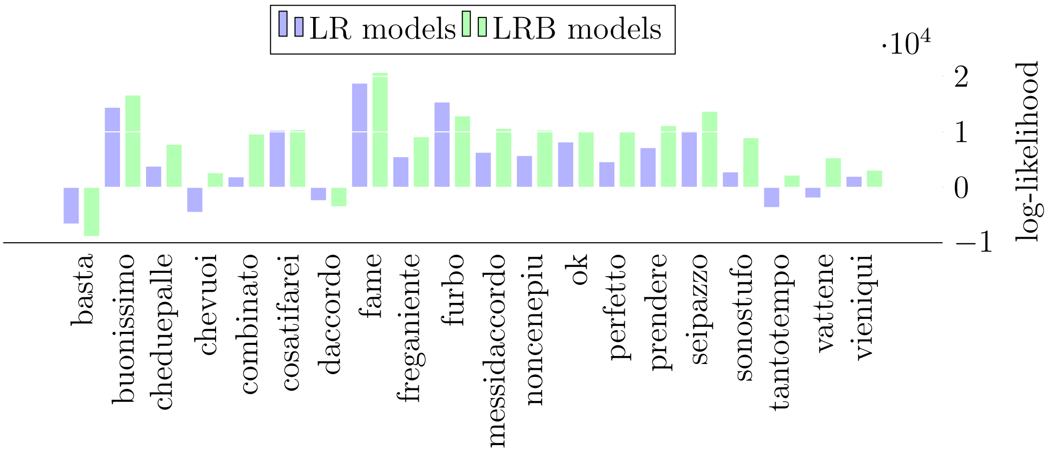


Figure 3.3: *A bar plot showing the relative final validation likelihoods for a classifier based on LR and LRB models.*

Held-out validation set

As discussed in section 2.5, given enough data, a held-out validation set can serve as a generalisation metric as opposed to doing expensive cross-validation. Furthermore, cross-validation is not feasible for discriminative training of HMMs, and the use of a validation set is crucial to be able to do early stopping. With enough free parameters the system will learn the data perfectly along with its noise, and will generalise poorly to unseen samples. Figure 3.4 shows a plot of the well-known early stopping scenario.

3.2 Stochastic gradient descent

The discriminative optimisation is now discussed. Some of the results from this section are produced from toy problems based on the data used in the study. In implementing stochastic gradient descent (the GPD algorithm), some recommendations in Bottou (2012) and LeCun et al. (2012) proved very useful. In summary they are:

- Randomising the training samples.
- Transforming the training samples to have zero mean and unit variance.
- Keeping track of both training and validation error.

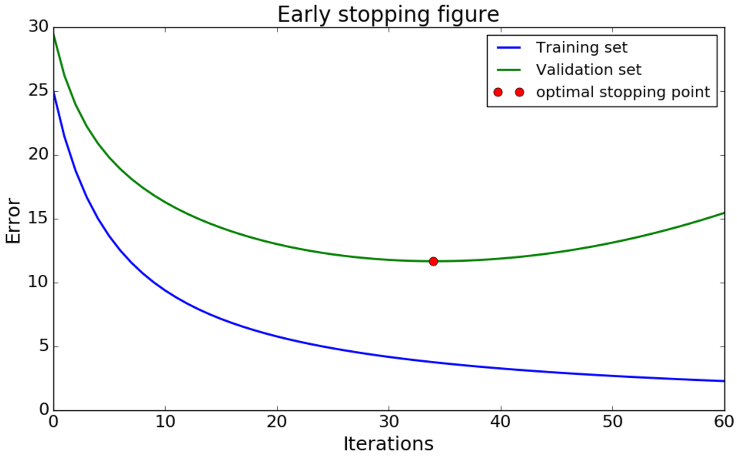


Figure 3.4: *A figure showing a typical example of over-training. The red dot indicates the optimal point to stop training.*

- Choosing a learning schedule using a subset of the training data.

The rest of this section details the particular choices used for the discriminative optimisation along with the reasoning behind them.

3.2.1 Preconditioning

Preconditioning a gradient descent operation either means pre-multiplying the gradient with some rotation matrix or transforming the data. Discussed here are the MCE preconditioning matrix and the data standardisation technique used in this study.

Pre-conditioner matrix for MCE HMM gradients

The preconditioning matrix U_n is used in GPD training to address the sensitivity of the mean parameter update μ_{jkd}, which is affected by the size of the corresponding variance value. The matrix U_n is an identity matrix with entries of σ^2_{jkd} along all of the diagonal elements that correspond to the μ_{jkd} entries of the gradient vector. Since the derivative with respect to μ_{jkd} is a weighted sum of ratios with σ^2_{jkd} as the denominator, the preconditioning matrix decouples the update from the variance values (Chou et al., 1992). The U_n pre-multiplication is entirely embodied by the μ parameter transform, as finding a derivative w.r.t $\tilde{\mu}$ and transforming it back is equal to multiplication of the original gradient with σ^2_{jkd}. A complete list of derivatives w.r.t transformed parameters can be found in the appendices.

Data transformation scheme

The data standardisation procedure outlined in LeCun et al. (2012) is used in this study to improve GPD training, however there are a few important hurdles to doing so. For MCE training, models are required to be initialised with a ML training procedure. The models were also to operate on test data from the original distribution.

This means that the models, trained on the original data, had to be transformed along with the training data to the zero mean, unit variance region of the feature space, and transformed back to be evaluated on validation data from the original distribution. The procedure is as follows: two sets of models are kept in memory during the optimisation, one for the standardised data distribution and one for the original. The original data mean and variance vectors are stored, and used to translate the first set of ML trained models to the centred and scaled region. This is done by translating and scaling all output distributions from all models by the data mean and variance respectively. After every MCE update, the second set of models is updated from the first set by adding the original mean and scaling up with the original variance. Figure 3.5 shows how the standardisation procedure improves the conditioning of an optimisation for a five-class toy problem based on the gesture dataset. In the figure, the standardised error can be seen to decrease in a smoother fashion than that of the unstandardised error.

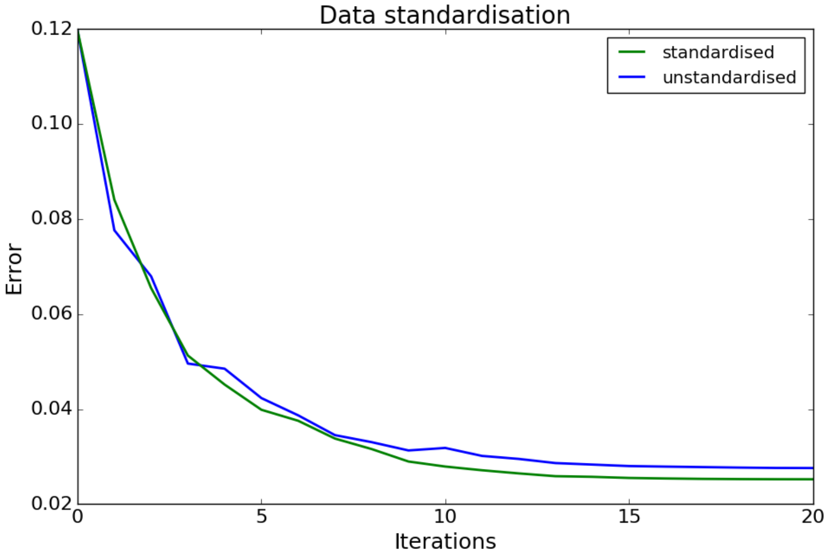


Figure 3.5: *Training curves before and after applying standardisation. The curve shows the MCE functions for a five-class classifier on which the sparsest optimisation of section 5.1 was applied.*

3.2.2 Sigmoid central slope

The parameter γ in (2.14) controls the central slope of the sigmoid loss-function. This value is normally set to $\gamma \geq 1$ (Juang et al., 1997). However, by means of experimentation it was found that the value $\gamma = 0.1$ led to better conditioning of the optimisation. Figure 3.6 shows how the value $\gamma = 0.1$ achieves a steeper descent and a lower final minimum than the value $\gamma = 1$. The optimisations were run on identical data and models for the PCA version of the feature set in (4.12). Setting the γ value to a value less than one does unfortunately cause the empirical loss function to no longer directly correspond to the error rate on the training set. However, the validation error function is evaluated with $\gamma = 1$ in all MCE experiments, and can be seen in chapter 5 to follow the training error for lower γ values. For all of the MCE experiments up to section 5.4, a slope value of $\gamma = 0.1$ is used. Section 5.4 investigates the effect of further decreasing the slope value, and concludes that lower values allow better conditioning but that at some point the loss function no longer corresponds to the problem. The experiment shows that given lower slope values, the validation error still follows the loss function and that for the current problem, over-training is only seen when the slope value is set to $\gamma = 0.01$.

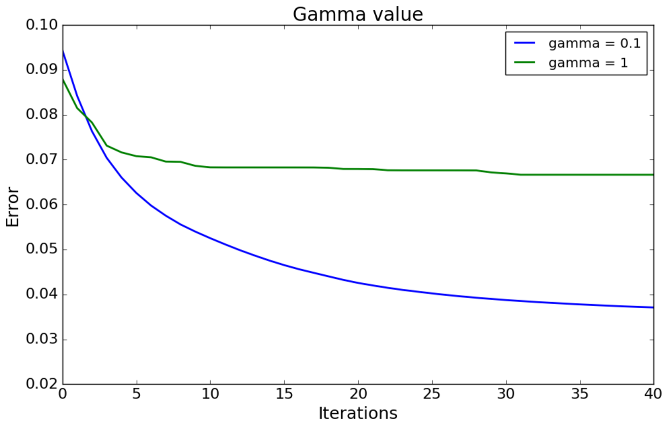


Figure 3.6: *A figure showing two five-class MCE runs from the data set, with a gamma value of 0.1 showing improved conditioning of the optimisation.*

3.2.3 Norm value

The parameter η controls the norm in the second term of the misclassification measure (2.13). In a sense it controls the relative contribution of the competing models to the distance measure, and correspondingly the relative influence an example sequence has on the model parameters. When it is set to one, all models contribute equally. When it is set to infinity, the highest-scoring competitor makes the sole contribution. At high η values, the optimisation will become slightly more robust as it approximates the infinity norm case. This is because fewer parameters are allowed to change drastically at the presentation of a new sample. This study only considers the $\eta = 1$ and $\eta \longrightarrow \infty$ cases.

3.2.4 Learning Schedule

Stochastic gradient descent is guaranteed to converge for the following conditions on the learning rate (Chou et al., 1992):

$$\sum_{n=1}^{\infty} \epsilon_n \longrightarrow \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \epsilon_n^2 < \infty. \tag{3.1}$$

In order to converge, the learning rate is annealed so as to take smaller steps as the optimisation approaches the minimum. Usually the learning rate is empirically chosen, and a good method of doing so is finding it on a subset of the training data (Bottou, 2012). A linearly decreasing learning schedule

$$\epsilon_n = \epsilon_0 \left(1 - \frac{n}{n_{max}}\right) \tag{3.2}$$

was chosen for the GPD optimisation. Empirically it was found to be quite robust and worked well with momentum. Although SGD is independent of dataset size (Bottou, 2012), it was decided as a matter of convenience to specify the maximum number of iterations n_{max} and cost function evaluations in epochs over the training set.

3.2.5 Momentum

To accelerate learning, a classical momentum operation (Sutskever et al., 2013)

$$v_{n+1} = \theta v_n - \epsilon_n U_n \nabla \ell(X_n, \Lambda)|_{\Lambda=\Lambda_n} \tag{3.3}$$

$$\tilde{\Lambda}_{n+1} = \tilde{\Lambda}_n + v_{n+1} \tag{3.4}$$

is embedded in the update. The same momentum constant of $\theta = 0.2$ is used in all experiments and was also empirically chosen. The figure below shows how an optimisation is accelerated using the momentum operation.

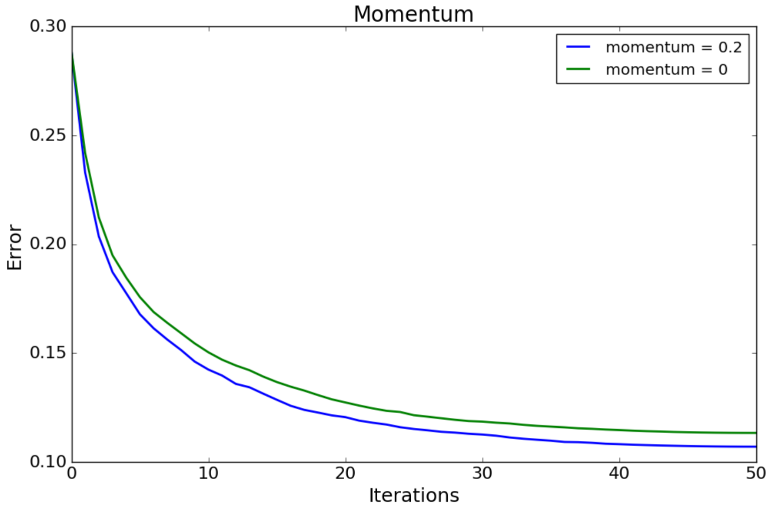


Figure 3.7: *Training curves for the exact same scenario before and after applying momentum. The curve shows the MCE functions for a five-class classifier on which the densest optimisation of section 5.1 was applied.*

3.3 Overview

The entire process for arriving at the discriminative HMM classifier is described. From the data set, labelled gesture intervals are used to segment, from the skeleton stream, joint coordinate trajectories of interest. The joint coordinate trajectories are fed into a per-frame feature extraction function to produce feature sequences. Section 4.3 describes the feature extraction processes. As certain joints are noisy (in particular the hands), the sequences are first smoothed using a low-pass filter as described in section 4.3.3. Feature optimisation forms a further dimension of investigation in the study and is described in section 4.3.4. With feature sequences extracted and smoothed, a maximum likelihood classifier is constructed using the methodologies described in section 3.1. The maximum likelihood classifier is either a uniform set of models or one produced from individual grid searches (see section 2.5). In chapter 4, three maximum likelihood performance benchmarks are presented for different feature sets. The performance of the classifier is then improved with discriminative training using the methodology described in section 3.2. Chapter 5 presents results from multiple experiments with minimum classification error training. All algorithms were implemented in C++ using the Eigen library for handling matrices (`http://eigen.tuxfamily.org`). With methods borrowed from the HMM toolbox by Kevin Murphy and the highly optimised linear algebra routines from Eigen, the MCE training implementations can run within hours what would otherwise take days for a naive implementation. Training on the highest density updates (see section 5.1) for all classes and data would otherwise be infeasible. Because the MCE optimisations are stochastic, multiple runs were conducted for peace of mind. The University of Cape Town ICTS HPC Cluster (`http://hpc.uct.ac.za`) was used for the multiple runs of all major experiments.

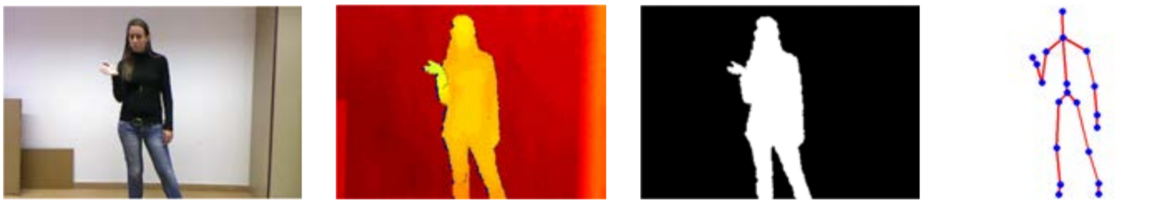
Chapter 4

Dataset and set-up

This chapter discusses the gesture recognition part of the study. First the specifications of the data set are detailed, along with its particular use in the study. Thereafter the similarity problem is explained, which forms an important reference. The chapter then discusses feature sets, filtering, decorrelation and the ML benchmark results.

4.1 The Chalearn LAP 2014 gesture dataset

According to Escalera et al. (2014), the Montalbano dataset was the largest commercial depth-camera based gesture dataset in the literature in 2014. Table 4.1 summarises the layout of the dataset. It is an extension of the 2013 ChaLearn multi-modal gesture recognition challenge data, and was used in the third track of the 2014 competition. The aim of this competition was to do simultaneous spotting and recognition on multi-modal data, with the Jaccard index as the evaluation metric. The dataset consists of 20 Italian cultural or anthropological gesture classes. For each labelled gesture in the set there are four visual modalities: RGB, depth, user-segmented depth, and joint coordinates from the Kinect skeletal model (see figure 4.1).



(a) *RGB* ***(b)*** *Depth* ***(c)*** *User mask* ***(d)*** *Skeleton*

Figure 4.1: *The different modalities for the Montalbano dataset (Escalera et al., 2014).*

4.1.1 Departure from dataset original intent

For this study, the use of the dataset differs from the intent of the competition in two important ways. Firstly, this study only considers the classification problem and as

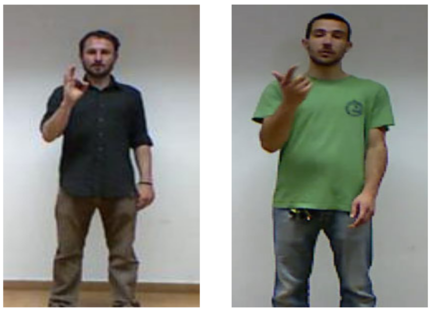
Table 4.1: *Characteristics of the Montalbano dataset (Escalera et al., 2014).*

Training sequences	Validation sequences	Testing sequences	Sequence Duration	FPS
393 (7,754 gestures)	287 (3,362 gestures)	276 (2,742 gestures)	1-2 min	20
Modalities	**Number of users**	**Gesture categories**	**Labelled sequences**	**Labelled frames**
RGB, Depth, User mask, Skeleton	27	20	13,858	1,720,800

such employs the truth spotting labels to segment a test set, just as with the training and validation sets. Secondly, the study only considers the skeleton modality and is therefore geared toward a dynamic gesture recognition problem. The gestures contain varying degrees of hand pose articulation, where some gestures are almost entirely static with the hand pose containing all of the information. This is visible in the confusion matrices as some gestures are classified correctly with substantially higher frequencies than others. This is illustrated next.

4.1.2 Similarity of gestures given single modality

Since skeletal joint trajectories are the only features considered, some classes in the Montalbano dataset have high false positive rates. Those gestures rely on hand shapes that live mainly in the RGB modality and only leave traces of discernible patterns in the skeletal modality. An example is shown in figure 4.2 where the gestures ”ok” and ”noncenepiu” occupy the same upper body skeletal pose. The dynamic difference between the two is that for ”ok”, the hand translates forward and backward slightly, where for ”noncenepiu” the wrist rotates the hand pose from left to right. In a stream of skeletal points these gestures look very similar and cause overlap in classification. This especially affects the ML classifier as distributional overlap will be prevalent, it is argued, in such cases. Figure 4.3 shows how a uniform ML model set performs on the test data. It can be seen that certain gestures tend to have false positives, especially the portion from ”freganiente” to ”prendere” along the diagonal. This illustrates the problem of similarity and provides a reference for evaluating confusion matrices presented later.



(a) *ok* ***(b)*** *noncenepiu*

Figure 4.2: *Two gestures from the Montalbano dataset that look very similar when represented only in skeletal features. Images taken from Escalera et al. (2013).*

4.1.3 Training to validation ratio

As can be seen from table 4.1, the training to validation ratio is roughly 3 : 1. Since the validation data is used only to track generalisation error while training, the datasets are re-divided into a 4 : 1 ratio for training and validation.

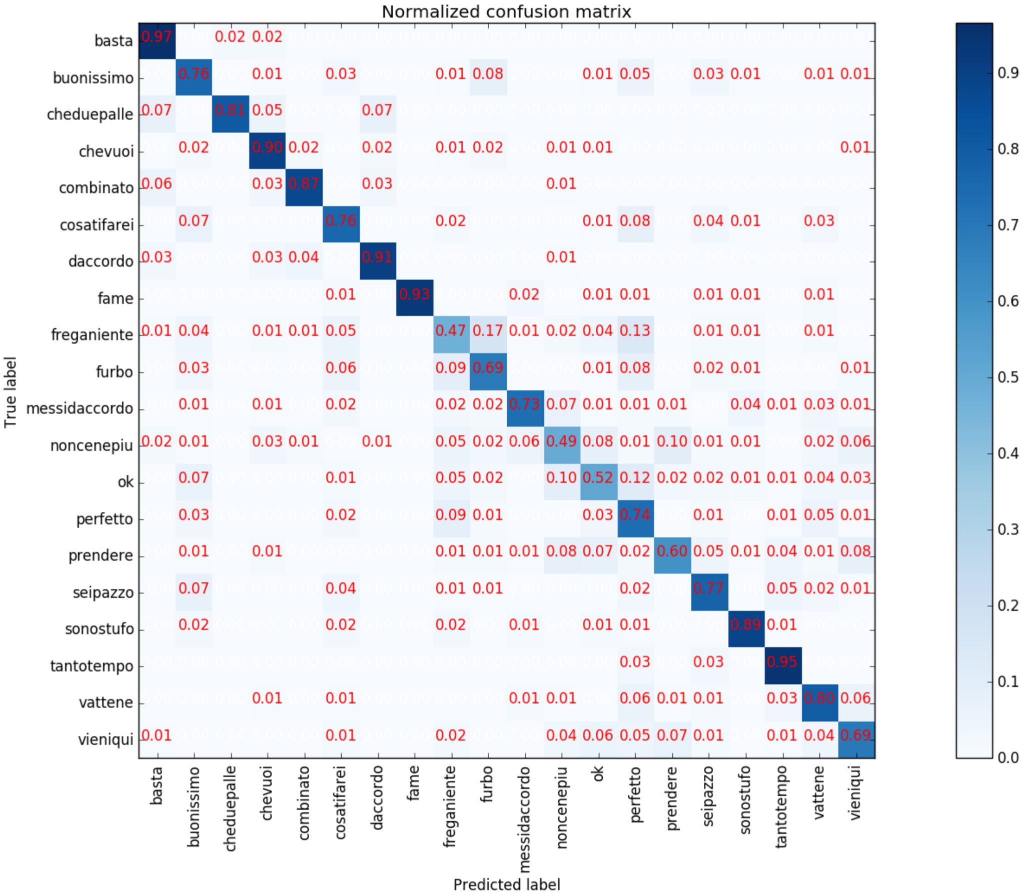


Figure 4.3: *The test set confusion matrix for a uniform six-state, twenty-mixture model set on the feature set* (4.12)*. A high number of false positives can be seen in the rows and columns of some of the gestures. Accuracy: 76.42%.*

4.2 Methods and results of the contestants

After the conclusion of the challenge a rich set of academic papers citing it came about as the majority of the contestants published their work. This section is concerned with that work done on the 2014 challenge. First a general overview of methods used and results brought forward are discussed. Then the results that can be directly compared with the work in this study are discussed.

4.2.1 General overview

There were many sophisticated methodologies for the gesture segmentation and recognition problem to be found in the solutions of the participants. However as a matter of practicality, only the winning method is detailed here and other methodologies are treated in overview.

The evaluation metric

The data for the competition is organised as a massive set of RGB-D video sequences with the accompanying skeleton streams. Each sequence documents a single user performing various gestures (about ten per sequence), and comes with a file providing the segmentations and ground truth labels. As previously mentioned the competition employs the Jaccard index as the evaluation metric. For a gesture category n in sequence s, the Jaccard index is defined as:

$$J_{s,n} = \frac{A_{s,n} \cap B_{s,n}}{A_{s,n} \cup B_{s,n}}, \tag{4.1}$$

where $A_{s,n}$ is the ground truth of gesture n at sequence s, and $B_{s,n}$ is the prediction for gesture n at sequence s. Both $A_{s,n}$ and $B_{s,n}$ are binary vectors where unity values correspond to the frames in which the n-th gesture is being performed (Escalera et al., 2014). The Jaccard index is unity for a gesture n only at the intersecting frames of the ground truth and predictions. For false positives, where a gesture is predicted but does not show up in the ground truth, the index evaluates to zero (Escalera et al., 2014). The test set evaluation metric by which contestants were scored is the mean Jaccard index (averaged over all classes n) over all sequences.

First place

Neverova et al. (2014) used a multi-scale deep learning architecture to land first place on the gesture recognition track of the competition. The architecture operates on multiple spatial and temporal scales to capture information out of all modalities and to accommodate varying length gestures respectively. The modalities used are the skeleton stream, the depth video stream and the RGB video stream which they convert to grey scale. The skeleton stream is used to extract a feature descriptor they call the articulated pose. It is made up of normalised skeletal joints, their velocities and accelerations, various angle features and pairwise joint distances to form a 183-dimensional feature vector for each frame. This articulated pose descriptor serves as an efficient feature representation for the upper body dynamic component of the gestures. Based on it alone, the system achieves a Jaccard index of 0.808. Unfortunately no test set classification results are available to compare with. The depth and grey scale modalities are used for convolutional representation learning of the finer hand articulations. The left and right hands are first cropped out. Their positions are then stabilised within the new frames and then the frames are normalised to zero mean and unit variance. Finally the left hand frames are flipped vertically and combined with the right hand frames in a single training set. For each channel, a two-stage convolutional learning of the representation is done independently. The two streams are then fused with a set of fully connected hidden layers to form the left- and right hand paths. The left- and right hand paths are then fused along with the output of the articulated pose layer using a fully connected shared hidden layer. This allows the learning of correlations across modalities. This architecture is applied at three temporal scales by uniformly sampling an input data sliding window using different between-frame intervals. For each temporal scale (or interval), a prediction is formed by the deep architecture described earlier. The predictions are then fused with a simple voting strategy using a single weight per model. Using frame aggregation and temporal filtering, the network outputs a prediction for every frame. An additional binary classifier based on the articulated pose feature descriptor is used to distinguish periods of activity from periods of rest. This improves localisation and accordingly, the Jaccard index as pre- and post stroke frames tend to produce erroneous classifier output. Neverova et al. (2014) achieve a Jaccard index of 0.85 on the test set, placing them first in the competition rankings. The strength in their methodology lies in powerful per frame classifications that are then better localised using the binary classifier.

Table 4.2: *A table showing modalities, temporal segmentation strategies and classifiers used by the participants of the LAP 2014 challenge, along with their rankings. SK - skeleton, DNN - deep learning neural network, RF - random forrest, RT - regression tree, MRF - Markov rondom field, kNN - k-nearest neigbors. Adapted from Escalera et al. (2014).*

Team	**Score**	**Rank**	**Modalities**	**Segmentation**	**Classifier**
LIRIS	0.849987	1	SK, Depth, RGB	Joints motion	DNN
CraSPN	0.833904	2	SK, Depth, RGB	Sliding windows	Adaboost
JY	0.826799	3	SK, RGB	MRF	MRF, kNN
CUHK-SWJTU	0.791933	4	RGB	Joints motion	SVM
Lpigou	0.788804	5	Depth, RGB	Sliding windows	DNN
stevenwudi	0.787310	6	SK, Depth	Sliding windows	HMM, DNN
Ismar	0.746632	7	SK	Sliding windows	RF
Quads	0.745449	8	SK	Sliding windows	SVM
Telepoints	0.688778	9	SK, Depth, RGB	Joints motion	SVM
TUM-fortiss	0.648979	10	SK, Depth, RGB	Joints motion	RF, SVM
CSU-SCM	0.597177	11	SK, Depth	Sliding windows	SVM, HMM
iva.mm	0.556251	12	SK, Depth, RGB	Sliding windows	SVM, HMM
Terrier	0.539025	13	SK	Sliding windows	RF
Team Netherlands	0.430709	14	SK, Depth, RGB	DTW	SVM, RT
VecsRel	0.408012	15	SK, Depth, RGB	DTW	DNN
Samgest	0.391613	16	SK, Depth, RGB	Sliding windows	HMM
YNL	0.270600	17	SK	Sliding windows	HMM, SVM

Overview of remaining methodologies

In the original 2013 competition, the three most popular classifiers (in descending order) were HMMs, random forests and neural networks. Dynamic programming techniques such as dynamic time warping along with HMMs constituted the sequence learning methods used in the competition and represents a fair portion of proposed solutions. Figure 4.4 shows a bar graph with the relative proportions of various classifiers used in the solutions of the contestants. Random forest and neural network variants were the second and third most popular techniques respectively (Escalera et al., 2013).

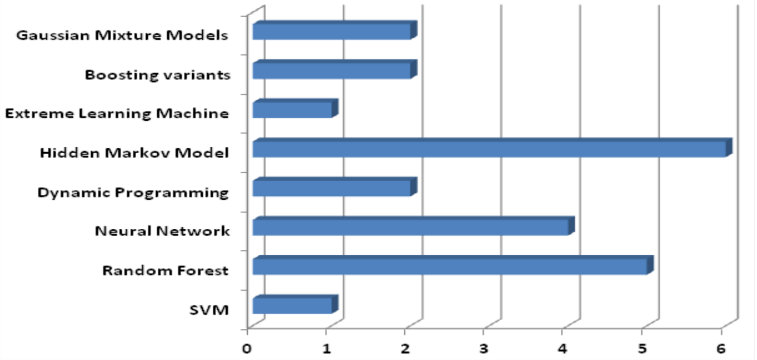


Figure 4.4: *The relative proportions for the choice of preferred classifiers among the participents of the 2013 LAP challenge. Taken from Escalera et al. (2013).*

The majority of participants of the 2014 challenge preferred non-temporal classifiers in their solutions to the simultaneous gesture segmentation and recognition problem. Table 4.2 summarises the modalities, segmentation strategies and classifiers used by the contestants of the 2014 challenge. The table is a shortened version of the one presented in Escalera et al. (2014). From the table it can be seen that only five participants used HMMs in their solutions, whereas eight participants used SVMs, four participants used deep convolutional neural networks and at least four used ensemble methods such as random forests or boosting. It can be seen that the majority of participants make use of the skeleton modality as it facilitates the recognition of large scale upper body gestures, but that the winning methods make use of more than just one modality. For a successful solution to the detection and recognition task, especially in light of the Jaccard index as evaluation metric, it is necessary to integrate the temporal segmentation and classification strategies so as to optimally localise gestures. The dataset contains cases where one gesture follows directly on another without a discernible pause in between. For this reason, using a binary classifier to detect gestures of variable length and simply feeding them to a sequence classifier is not a good solution. In such cases, however, per-frame classification automatically handles the segmentation. Wu and Shao (2014) use an interesting strategy for segmentation and recognition. They use deep belief networks and convolutional neural networks on the skeleton and depth modalities respectively, to estimate the emission probabilities of a set of HMMs. An ergodic state is added to the looped network of parallel-connected HMMs so that Viterbi decoding can be used for per-frame segmentation and recognition. Their software submission landed in sixth place with a Jaccard index of 0.787310, which in their publication is cited as the result when only the skeleton modality is used. The publication reports a Jaccard index of 0.8162 using all modalities. The top-performing teams implement non-temporal per-frame classifiers. However it is suspected that such solutions will eventually struggle if more gesture categories were to be added, since partial gestures will then be harder to discriminate. With the exception of the convolutional neural network strategies, the per-frame classifiers mentioned used very sophisticated feature extractions to produce sufficiently discriminative features, a task that will become harder as more classes are introduced.

4.2.2 Results concerning the single skeleton modality and segmented test data

From table 4.2 it can be seen that almost all contestants use the skeletal modality. However, most publications report their results in terms of simultanious segmentation and classification. For comparison with this work, publications that report classification results on the segmented test set are cited. The paper submissions of the contestants in eighth and eleventh places present classification results on the test set using the skeleton modality alone. These results are the only results that are directly applicable in the context of this work, and comparable to the results presented here.

Eighth place

Evangelidis et al. (2014a) place particular emphasis on how their system outperforms other methods when only skeletal data is used. Their software submission to the judges of the competition placed them eighth out of seventeen contestants. In the journal paper accompanying their work, they state that a fixed software bug re-ranks the solution up to fourth place. The original software submission achieved a Jaccard index of 0.745, and the revised solution an index of 0.816. They also present a ranking of the contestants who published results using only skeletal features, in which their fixed bug places them first with and index of 0.768. For their multi-modal result they use a skeletal feature descriptor along with colour features from histograms of flows. Forming the base of their skeletal descriptor are skeletal features they call quads. Quads encode the geometric relation of joint quadruples leading to a low-dimensional feature vector. A quad is formed from an ordered set of four skeletal joints in $\mathbb{R}^3$ $[\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \mathbf{x}_4]$, where $(\mathbf{x}_1, \mathbf{x}_2)$ is the most widely separated joint pair (Evangelidis et al., 2014b). A new local coordinate system is formed where $\mathbf{x}_1$ is the origin and $\mathbf{x}_2$ is mapped to $[1\ 1\ 1]^T$. The two in-between joints are then translated, rotated and scaled into the new coordinate system. These transformed in-between joints are then vertically concatenated to form a $\mathbb{R}^6$ quad feature vector. Fourteen such ordered sets of joints are used to describe the full upper-body pose. To complete the definition of the descriptor, a temporal sequence of quads are represented by a Fisher vector with a GMM as the underlying generative model. The GMM is first trained via EM on the quads of the training set, thereafter it can be used to generate a Fisher vector for any sequence of quads. Using three sliding windows, per-frame Fisher vectors (belonging to the window's center frame) are fed to a multi-class linear SVM to produce per-frame labels and their associated costs. Continuous gesture recognition is then formulated as the problem of finding a per-frame gesture labelling for a long sequence as a piece-wise constant function.

The problem of finding the appropriate piece-wise constant function is cast into an energy minimisation framework, whereby the per-frame labelling cost along with a penalty term for smoothness are jointly minimised. To solve this global optimisation problem, Evangelidis et al. (2014a) use a dynamic programming algorithm that recursively populates a cost table and then backtracks to find the minimum energy labelling. The no-gesture class labelling is provided by an additional binary classifier. For the articulated pose 1536-dimensional Fisher vectors are formed from a 128-component GMM, and for colour features 6144-dimensional Fisher vectors are formed from a 64-component GMM. An early fusion of the modalities is performed by simply concatenating the vectors into a single input to be classified. Using their sophisticated skeletal descriptor on its own, Evangelidis et al. (2014a) was able to achieve a 90% accuracy in the isolated classification of the testing set.

Eleventh place

Liang and Zheng (2014) placed eleventh out of seventeen contestants with a Jaccard index of 0.597. They employ a late fusion of three classifiers based on different modalities. Spatio-temporal skeleton features are classified using a concatenated HMM-SVM classifier pair. The per-category probability scores of the HMM gesture evaluations are thus passed on to the SVM classifier as features. The authors claim that the cascade classifier performs better on the validation set than the HMM classifier alone. Additionally they use a SVM with a radial basis function (RBF) kernel to classify higher level, so called holistic features, of the skeleton modality. The variance, mean, minimum and maximum of the skeleton descriptor are aggregated over a gesture interval to form the holistic features fed to the RBF-SVM. Lastly the depth modality is used with the two-dimensional motion trail model (2DMTM) (Liang and Zheng, 2013) to capture motion region information. The 2DMTM is a clever and very efficient representation of a gesture. It reduces a series of depth maps into four gray-scale images that capture the distributions of the motion- and static regions of the gesture. From the 2DMTM gesture representation, pyramid histograms of oriented gradient (PHOG) features are extracted to be classified by a linear SVM. Late fusion of the three classifiers is done with a weighted sum of their output scores. The weights were empirically defined on the validation set. For the spatio-temporal skeletal features, the authors use pairwise joint distances and bone orientations of twelve upper body joints. The joints of the head, shoulder-center, spine, hip-center, shoulders, elbows, wrists and hands form the basis of the descriptor. Before extracting features, the world coordinates of each joint are made person-centric for invariance to a user's position in the scene. Pairwise distance features were also normalised by the sum of all pairwise distances of the subject. From the Kinect skeleton stream, bone rotation features are available as quaternions. The authors normalise the quaternion of each joint by its L2-norm to produce the bone-orientation features that are concatenated to the pairwise distances to form the final descriptor. Liang and Zheng (2014) use a hand-location based gesture detector for segmentation of continuous gestures. The gesture boundaries are detected by a rule-based filtering of the vertical coordinates of the hands. To limit the effects of noise on the hand joints, they use the average of each hand's y-coordinates over the frames of a short sliding window as that hand's location. Assuming that all gestures are correctly classified, the authors evaluate the spotting module on the test set and find that it scores 0.8196 as a Jaccard index. The authors also present results on the isolated classification of the test set using their separate and combined modalities. They report an accuracy of 77.47% for the HMM-SVM on the spatio-temporal skeleton features, 83.02% for the RBF-SVM on the holistic skeleton-statistics features, 76.99% for the linear-SVM on the depth image features, and 92.8% for the combination of the classifiers. It is important to understand that the Jaccard score of the spotting module cannot simply be multiplied by the multi-modal classifier accuracy to arrive at the authors' competition ranking. This is due to the presence of deliberately distracting gestures and gestures following on one another so as to be spotted as one gesture in the test data.

4.3 Feature extraction

Feature extraction is the most important part of an HMM classifier design. The most suitable feature extractions for certain tasks is the topic of countless publications in the field of pattern recognition (Fink, 2014). This section discusses the features used and the optimisations that were found useful.

4.3.1 Normalised skeletal points

Raw skeleton joint coordinates carry much information about the gesture being performed, but need to be approximately centred and normalised to account for multiple users. By using joint coordinates relative to some central skeleton joint, the hip-center in this case, the features are invariant to a user's position in the scene. Furthermore, scaling all coordinates by some characteristic body dimension, in this case the distance from hip- to shoulder-center, the features become less dependant on body size. The normalised skeletal feature set used was the twelve-dimensional concatenated vector

$$\boldsymbol{x}_t = \left[\boldsymbol{j}_{le}, \boldsymbol{j}_{lw}, \boldsymbol{j}_{re}, \boldsymbol{j}_{rw}\right], \tag{4.2}$$

where

$$\boldsymbol{j}_{le} = \frac{\boldsymbol{j}_{left-elbow} - \boldsymbol{j}_{hip-center}}{s} \tag{4.3}$$

$$\boldsymbol{j}_{lw} = \frac{\boldsymbol{j}_{left-wrist} - \boldsymbol{j}_{hip-center}}{s} \tag{4.4}$$

$$\boldsymbol{j}_{re} = \frac{\boldsymbol{j}_{right-elbow} - \boldsymbol{j}_{hip-center}}{s} \tag{4.5}$$

$$\boldsymbol{j}_{rw} = \frac{\boldsymbol{j}_{right-wrist} - \boldsymbol{j}_{hip-center}}{s} \tag{4.6}$$

$$s = \|\boldsymbol{j}_{shoulder-center} - \boldsymbol{j}_{hip-center}\|. \tag{4.7}$$

Figure 4.5 shows the test set result of a model set formed by grid search on the feature set (4.2).

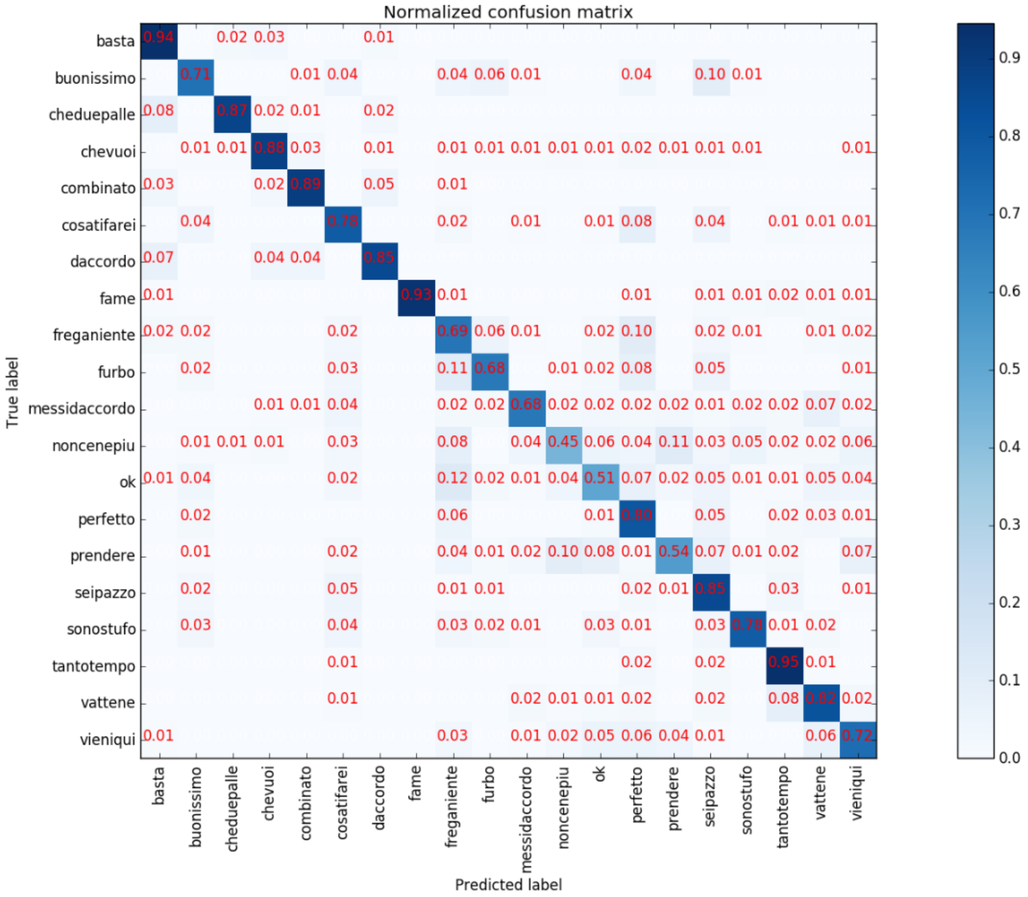


Figure 4.5: *The test set confusion matrix using grid searched models for the position features of* (4.2)*. Accuracy: 76.8%.*

4.3.2 Spherical angles of limbs

Kim and Kim (2015) used spherical angles of limbs for dynamic gesture recognition. The same feature set was employed in this study. Spherical angles offer a denser representation than joint coordinates since only two dimensions are required to represent a bone orientation if the radial distance is ignored. This feature set considers only directional information and is thus invariant to bone length. Figure 4.6 shows the pose vectors from which the angles are calculated. First the pose vectors are found from the shoulder, elbow and wrist coordinates of both arms. The four pose vectors are then transformed into their polar and azimuth angles respectively using

$$\theta = \arccos\left(\frac{v_z}{\|\boldsymbol{v}\|}\right) \tag{4.8}$$

$$\varphi = \arctan\left(\frac{v_y}{v_x}\right), \tag{4.9}$$

where $r = \|\boldsymbol{v}\|$ is omitted from the representation. Using information from six joint coordinates, an eight-dimensional feature vector is formed as

$$\boldsymbol{x}_t = \left[\theta_{rs-re}, \varphi_{rs-re}, \theta_{re-rw}, \varphi_{re-rw}, \theta_{ls-le}, \varphi_{ls-le}, \theta_{le-lw}, \varphi_{le-lw}\right]. \tag{4.10}$$

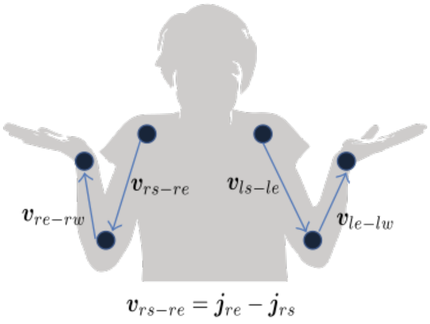


Figure 4.6: *Pose vectors from which spherical angles are calculated. Figure adapted from Kim and Kim (2015).*

Appending additional features

Using the spherical angles as base features, two feature sets were formed by adding hand joint information to the vector (4.10). The first is formed with further angle vectors calculated from the wrist-hand pose vectors

$$[\theta_{rw-rh}, \varphi_{rw-rh}] \qquad \text{and} \qquad [\theta_{lw-lh}, \varphi_{lw-lh}]\,, \tag{4.11}$$

and the second with the hand vectors relative to the head

$$[(\boldsymbol{j}_h - \boldsymbol{j}_{lh})] \qquad \text{and} \qquad [(\boldsymbol{j}_h - \boldsymbol{j}_{rh})]\,. \tag{4.12}$$

This forms a twelve and fourteen-dimensional feature vector respectively. The feature set (4.12) performs best given the grid search test set results from figures 4.5, 4.7 and 4.8.

4.3.3 Exponential filtering

The hand joint coordinates of the skeletal stream tend to be noisy on the Monalbano dataset. Tang and Dannenberg (2013) use exponential filtering to smooth oversampled skeletal data as a preprocessing stage in a gesture recognition problem. The exponential filter is a discrete version of an analogue first-order low-pass filter

$$\hat{\boldsymbol{x}}_t = \alpha \boldsymbol{x}_t + (1-\alpha)\hat{\boldsymbol{x}}_{t-1}, \tag{4.13}$$

where $0 \leq \alpha \leq 1$. A filter value of $\alpha = 1$ results in no filtering. The vector $\boldsymbol{x}_t$ represents the observations of a gesture sequence X, and $\hat{\boldsymbol{x}}_t$ the observations of a filtered gesture sequence. In this study a minimal filtering, with $\alpha = 0.8$, is used across all experiments.

4.3.4 Principal component analysis

A PCA transform was used to decorrelate and whiten data to investigate its effect on discriminative training. The transform

$$\boldsymbol{z}_t = \boldsymbol{\Lambda}^{-1}\boldsymbol{\Phi}^T(\boldsymbol{x}_t - \bar{\boldsymbol{x}}) \tag{4.14}$$

was first calculated on the training data, and then applied to all sets. The matrices $\boldsymbol{\Lambda}$ and $\boldsymbol{\Phi}$ are the eigenvalues and eigenvectors of the training scatter matrix respectively. The vector $\bar{\boldsymbol{x}}$ is the training data mean and $\boldsymbol{x}_t$ the observation at time t for any sequence X of the training, validation or testing sets. The decorrelation and whitening transform brought about a great improvement in ML performance. It was found however that dimension reduction via the transform had little effect. Dropping the two least informative dimensions resulted in a very similar accuracy, and further reduction caused a notably worse result. Figure 4.9 shows the test set results for the grid search performed on the PCA version of feature set (4.12).

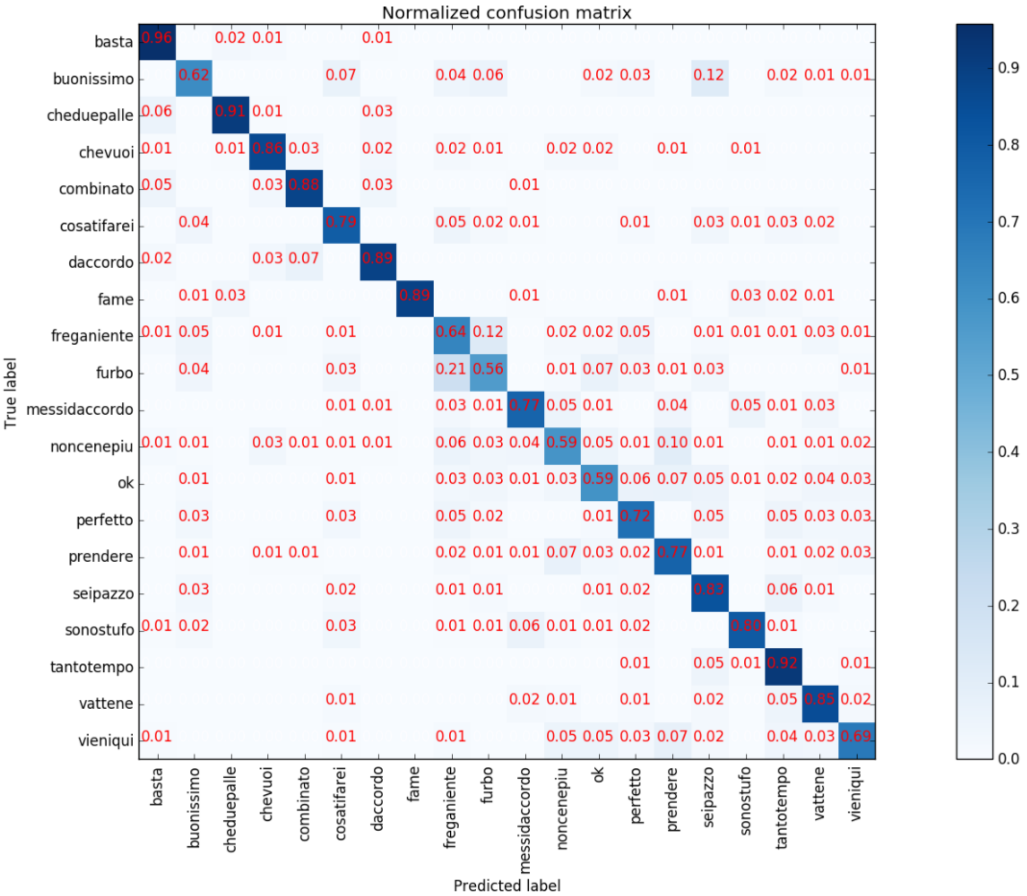


Figure 4.7: *The test set confusion matrix using grid searched models for the angular features in* (4.11). *Accuracy: 77.8%.*

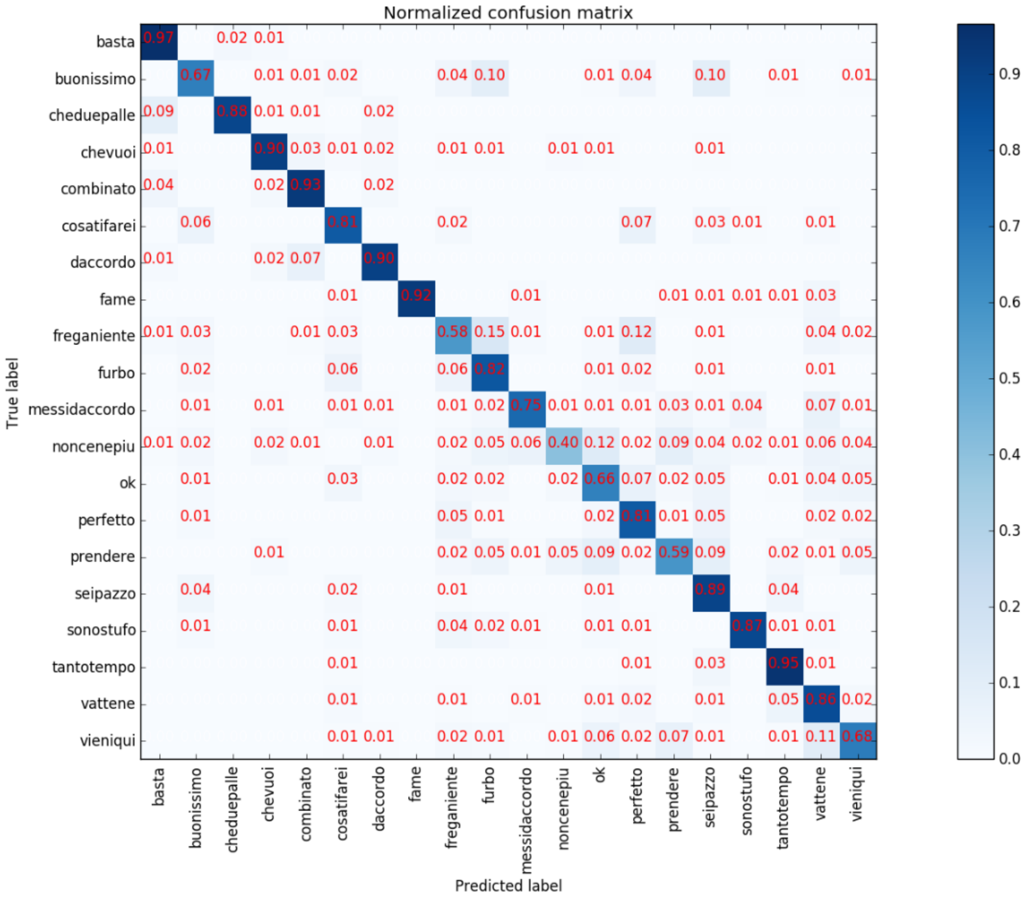


Figure 4.8: *The test set confusion matrix using grid searched models for the combined features in* (4.12). *Accuracy: 79.4%.*

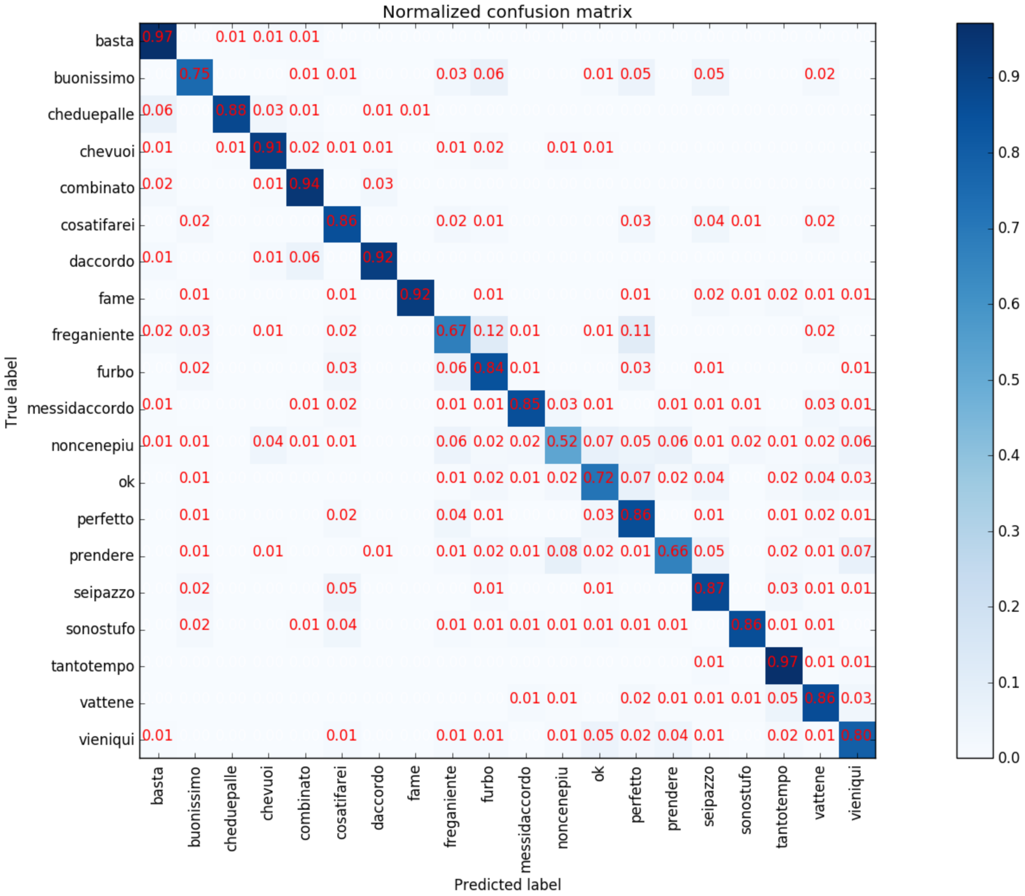


Figure 4.9: *The test set confusion matrix using grid searched models for the PCA version of the features in* (4.12). *Accuracy: 83.41%.*

Chapter 5

Experiments

This chapter details the experiments on MCE training. All experiments are conducted on the feature set of (4.12) and its PCA-whitened version. The following dimensions are investigated: the forms of the objective function, post-optimisation distributional changes, decorrelated features, and the slope of the loss function.

5.1 Objective functions

There are design choices to make when training HMMs with the MCE criterion. These choices manifest in how an HMM is used as a discriminant function and the particular form of the objective function. The second term in the misclassification measure (2.13) can be substituted for its infinity norm case, so that

$$d_i(X, \Lambda) = -g_i(X, \Lambda) + G(X, \Lambda), \tag{5.1}$$

where

$$G(X, \Lambda) = \max_{j \neq i} g_j(X, \Lambda). \tag{5.2}$$

A loss function with this misclassification measure embedded in it is discontinuous. However, as will be shown later, such a loss function can still be optimised stochastically with the GPD algorithm. As for an HMM discriminant function, this study considers two cases: first the standard data sequence likelihood

$$g_i(X, \Lambda) = \log P(X|\lambda_i) = \log \sum_{\boldsymbol{q}} P(X|\boldsymbol{q}, \lambda_i) P(\boldsymbol{q}|\lambda_i), \tag{5.3}$$

which involves all of the parameters in the model, and then the likelihood along the most probable path in the summation

$$g_i(X, \Lambda) = \log P(X, \bar{\boldsymbol{q}}|\lambda_i) = \log P(X|\bar{\boldsymbol{q}}, \lambda_i) + \log P(\bar{\boldsymbol{q}}|\lambda_i). \tag{5.4}$$

Using (5.4) as the discriminant function considers only the parameters along the Viterbi path. The optimisation of the MCE scheme based on Viterbi decoding is called the segmental GPD algorithm as originally intended in (Chou et al., 1992). From (5.1) and (5.2) it can be seen that only two models will have non-zero gradients in the gradient vector. These choices thus have the effect of controlling the sparsity of the overall parameter update. Table 5.1 ranks the four possible objective function combinations in terms of the density of their parameter update.

Table 5.1: *A layout of the different versions of the MCE objective function.*

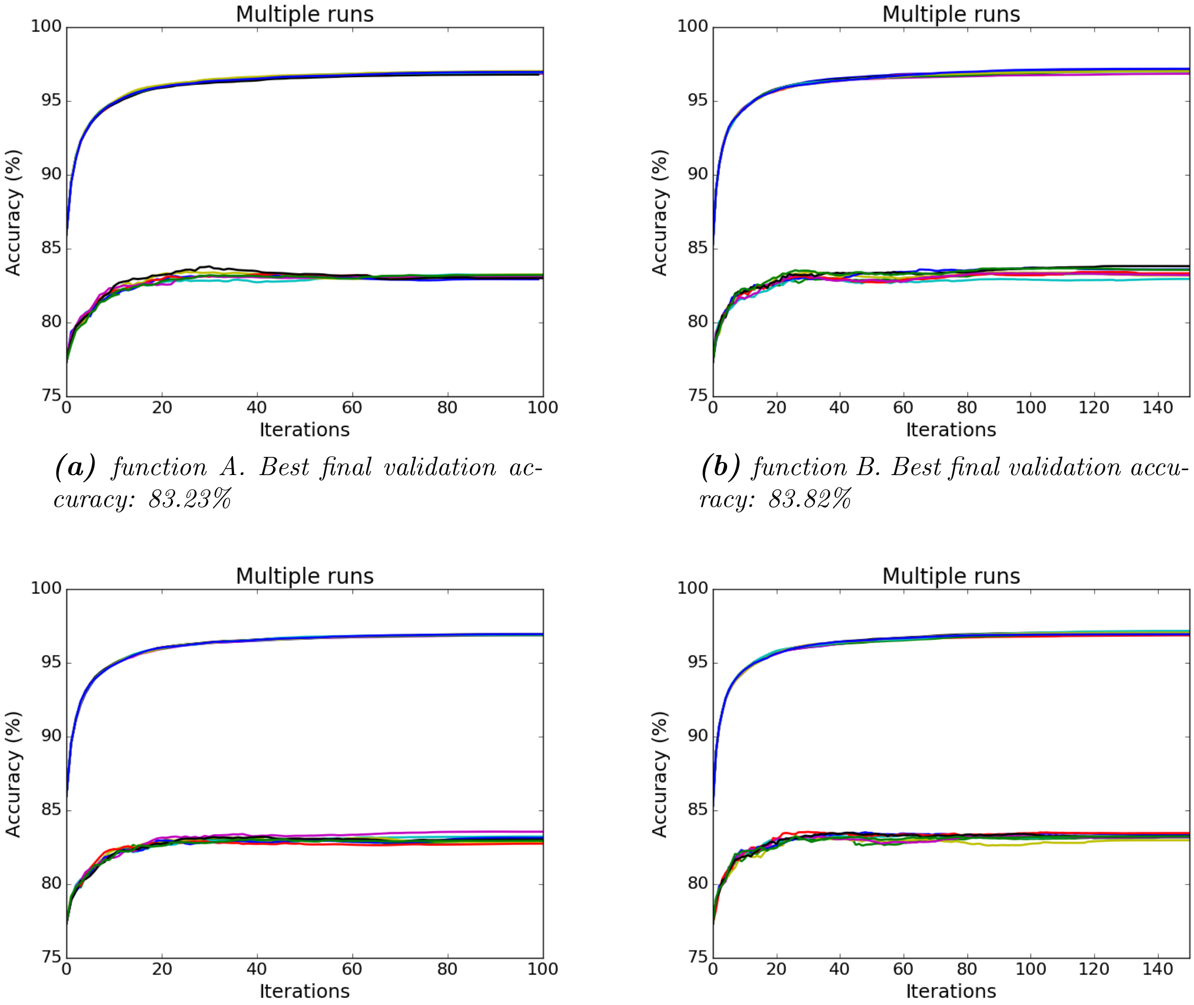
Name	update along	discriminant f.	Competing term
A	all paths of all models	$\log P(X\|\lambda_i)$	$log\left[\frac{1}{M-1}\sum_{j\neq i} e^{g_j(X,\Lambda)\eta}\right]^{\frac{1}{\eta}}$
B	all paths of 2 models	$\log P(X\|\lambda_i)$	$\max_{j\neq i} g_j(X,\Lambda)$
C	Viterbi path of all models	$\log P(X,\bar{\boldsymbol{q}}\|\lambda_i)$	$log\left[\frac{1}{M-1}\sum_{j\neq i} e^{g_j(X,\Lambda)\eta}\right]^{\frac{1}{\eta}}$
D	Viterbi path of 2 models	$\log P(X,\bar{\boldsymbol{q}}\|\lambda_i)$	$\max_{j\neq i} g_j(X,\Lambda)$

5.1.1 Hypothesis

Using the sparsest update scheme lightens the computational load tremendously since gradients need only be calculated for a small subset of the total parameter vector Λ. The gradients of all of the parameters that are not considered in the most likely path of the two competing models will evaluate to zero. Using a denser update means that more parameters are influenced by the presentation of each example. Intuitively, given a good choice of optimisation parameters, a dense optimisation will start converging in fewer iterations even if the iterations take longer. However, by the same reasoning, a sparser optimisation is less sensitive to step size. It is hypothesised that given a complete optimisation, the choice of objective function is of little consequence to the final test set performance improvement. This justifies the use of the sparsest form for large-scale problems.

5.1.2 Experiment

To investigate the effect of the forms of the objective function, each function must be evaluated on its optimisation performance. To this end the performance on the validation set and the test set need to be considered. The validation set performance is tracked during the optimisation and the test set performance is found from a confusion matrix. A standardised validation performance function is employed in order to directly compare between runs. Regardless of the particular objective function being optimised, the validation set performance is tracked using the first row function of table 5.1. Here the parameter settings $\gamma = 1$ and $\eta = 1$ are used so that the function would correspond directly to the error rate on the validation set. The optimisations based on the four versions of the objective function are shown in figure 5.1. The optimisations were all conducted on a uniform $N = 10$, $K = 20$ model set and non-PCA data. The use of non-PCA data meant that the standardisation procedure from section 3.2.1 had to be used.



(a) *function A. Best final validation accuracy: 83.23%*

(b) *function B. Best final validation accuracy: 83.82%*

(c) *function C. Best final validation accuracy: 83.55%*

(d) *function D. Best final validation accuracy: 83.46%*

Figure 5.1: *Eight runs for each objective function on non-PCA data. In all figures the top curves represent the performance on the training set and the bottom curves the performance on the validation set. Functions B and D run for more iterations as their optimisations take longer to plateau.*

5.1.3 Findings

Figure 5.1 shows the eight runs for each function's optimization. Overall there is a larger spread in final values for the validation set than for the training set. This is to be expected since the training data is used directly in the optimisations. Table 5.2 summarises the training results in terms of training time per iteration (expressed as a factor relative to the sparsest optimisation), the validation and test accuracies. The final validation recognition rates can be seen to be very close to the test set accuracies, although the validation accuracies are systematically slightly higher. This is likely due to the model set in the experiment, as it was chosen initially for its good performance on the validation set. It is concluded that the particular form of the objective function does not significantly affect the final test set performance. As an additional finding, the densest optimisation reaches its asymptotic error in the least number of iterations.

Table 5.2: *A summary of the objective function experiment.*

	Training time factor	Final Validation recognition rate	Test set accuracy
A	5.597	83.23%	83.05%
B	1.658	83.82%	83.35%
C	1.609	83.55%	83.14%
D	1	83.46%	83.11%

5.2 Distributional changes

The following section examines the changes to the distributions in individual model states after minimum classification error training. From the previous experiment it is easy to verify that the new optimisation results in increased classifier accuracy, but from the objective function equation it is not clear what happens to the distributions when training for maximum discrimination. It could be argued that the new training regime simply changes an HMM from a distribution model to a function that helps describe a decision boundary, and thus that the distributions are no longer really distributions. However, the way in which the distributions change could be enlightening when tied to an example of distributional overlap and how it is mitigated on a gesture recognition problem.

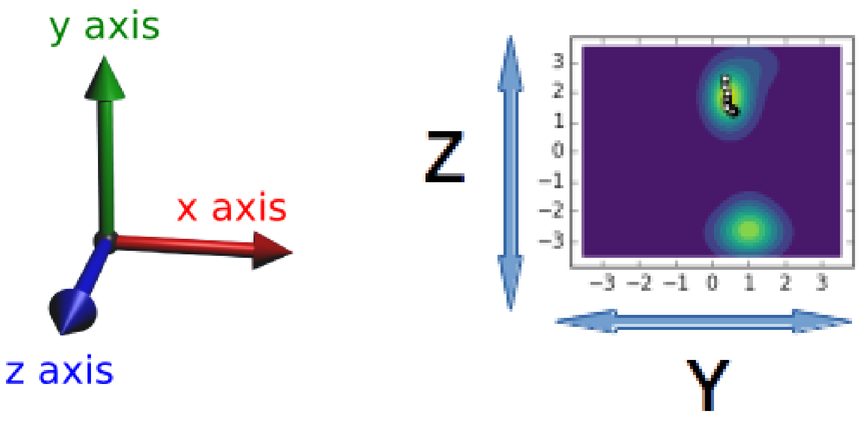
5.2.1 Hypothesis

Up to now it is understood that an MCE training takes ML pre-trained models, and that the distance function still relies on the likelihood-score of the correct gesture. Thus the score of the distribution of the correct model on its own data sequence is still of high importance since the MAP decision rule requires the correct model to have the highest score. In the context of gesture recognition, it is interesting to relate the changes in the distributions of generative and discriminative HMMs' per state distributions to the data-points of the actual sequence. It is hypothesised that discriminative training worsens the modelling of the data for both the correct and incorrect models, while worsening the wrong-class models a bit more than the correct-class models. This has the effect of forcing the MAP decision rule to pick the correct class more often, since competitor class probabilities will tend to score low.

5.2.2 Experiment

To see the effect of MCE training on the state distributions of the classifier, it is necessary to find representative cases in the data where these distributional changes can be tied to a correct classification of a previously misclassified sample. The distributional overlap problem will be illustrated as well as how discriminative training mitigates this effect. Toward this goal generative- and discriminative classifications on the validation set were mined for samples where the generative set misclassified and the discriminative set classified correctly. A six-state, five-mixture model set was used. Three such cases were found and are shown in figures 5.3, 5.4 and 5.5.

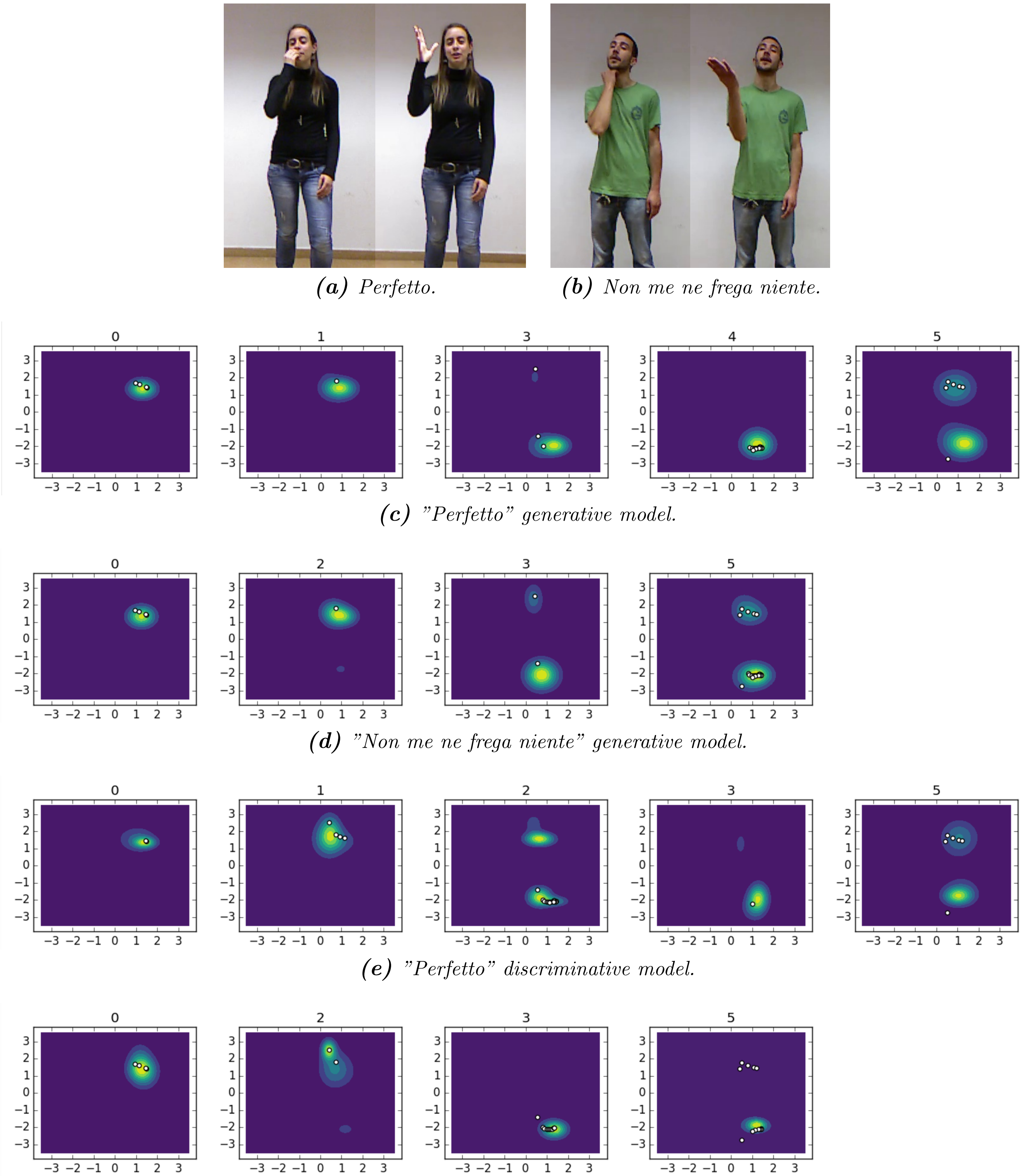
The characteristic motions of the gesture classes in question are briefly described. The gesture "Perfetto" ("Perfect" in Italian) starts with a closed hand at the mouth as if to kiss it, and then translates away from the face while the hand opens to its ending pose. The gesture "Non me ne frega niente" ("I don't give a damn") starts with the back of the fingers placed under the chin, and moved outward in a scraping motion from under the chin to a flat upright hand with the palm facing the signer. The gesture "Le vuoi prendere" ("Do you want to take these?") starts with a flat hand placed next to the body with the palm facing up. The hand then moves slightly in front of the body with a chopping motion generated from the wrist. The gesture "Che vuoi" ("What do you want?") is the quintessential Italian gesture often seen in films, where both hands are closed at the fingertips in front of the body and shaken slightly back and forth. Finally the gesture "Viene qui", which must be the global gesture for "Come here", starts with a flat hand in front of the body, and then moves in a waving motion inward to the signer. The figures also show the Viterbi clusterings of the data points into the correct and incorrect model states on a confused gesture given the generative and discriminative model sets.



(a) Axes for gesture RGB images.

(b) Axes for distribution plots.

Figure 5.2: *Accompanying figure to explain the systems of axes used in the plots to follow. The axes used for the gesture images are the absolute X-Y-Z axes. The axes used for the distribution plots are the components of the relative vector between the right hand and the head.*



(a) *Perfetto.*

(b) *Non me ne frega niente.*

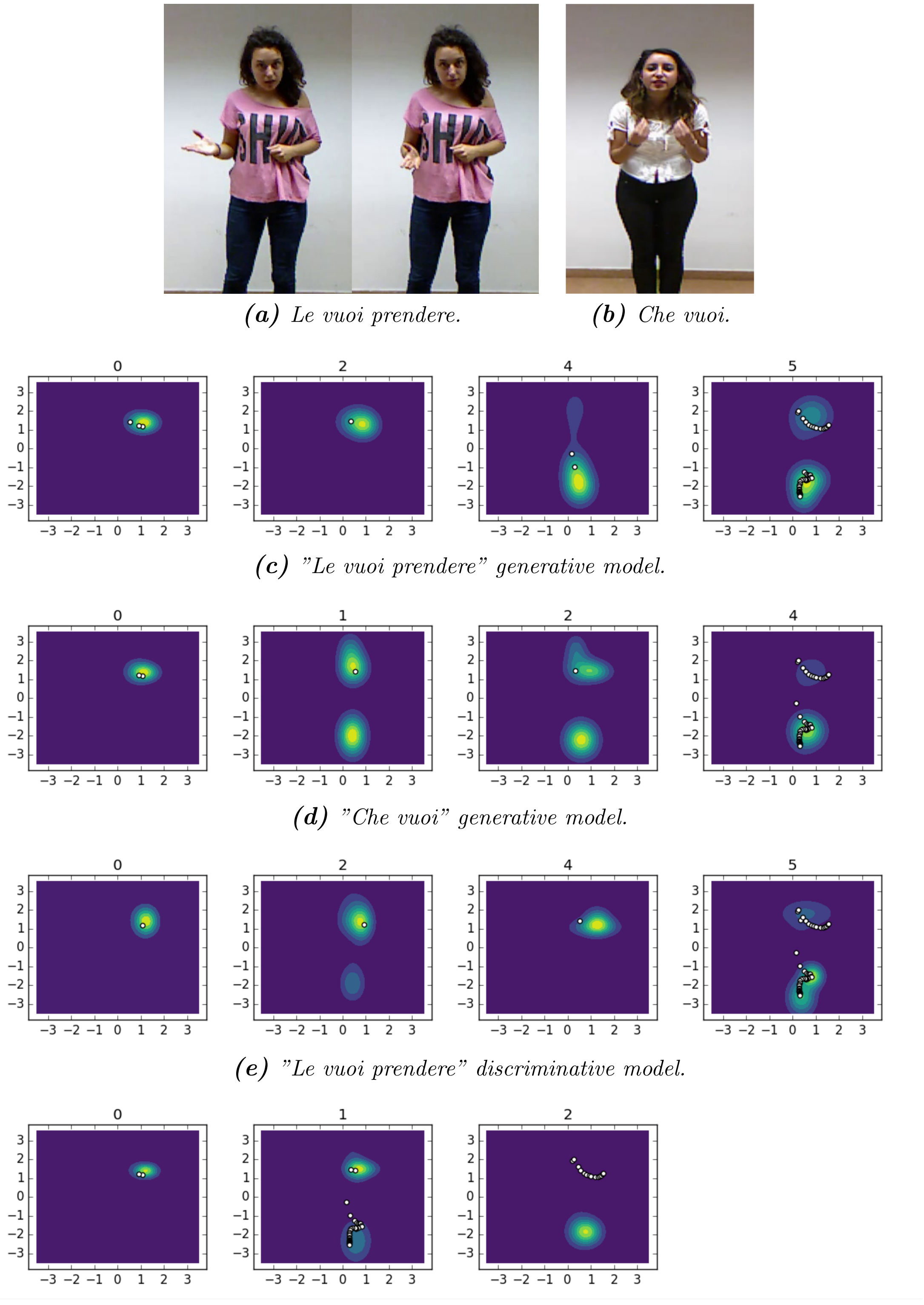
(c) *"Perfetto" generative model.*

(d) *"Non me ne frega niente" generative model.*

(e) *"Perfetto" discriminative model.*

(f) *"Non me ne frega niente" discriminative model.*

Figure 5.3: *The figures show, from the top: the characteristic poses, the Viterbi-clustering and accompanying distribution plots for the classes "Perfetto" and "Non me ne frega niente". The data points come from a sample in the validation set where "Perfetto" is misclassified as "Non me ne frega niente" in the generative case, and classified correctly in the discriminative case. The plots show the Y-Z dimensions of the head to right-hand distance features, which are the last two elements of the feature vector* (4.12).



(a) *Le vuoi prendere.*

(b) *Che vuoi.*

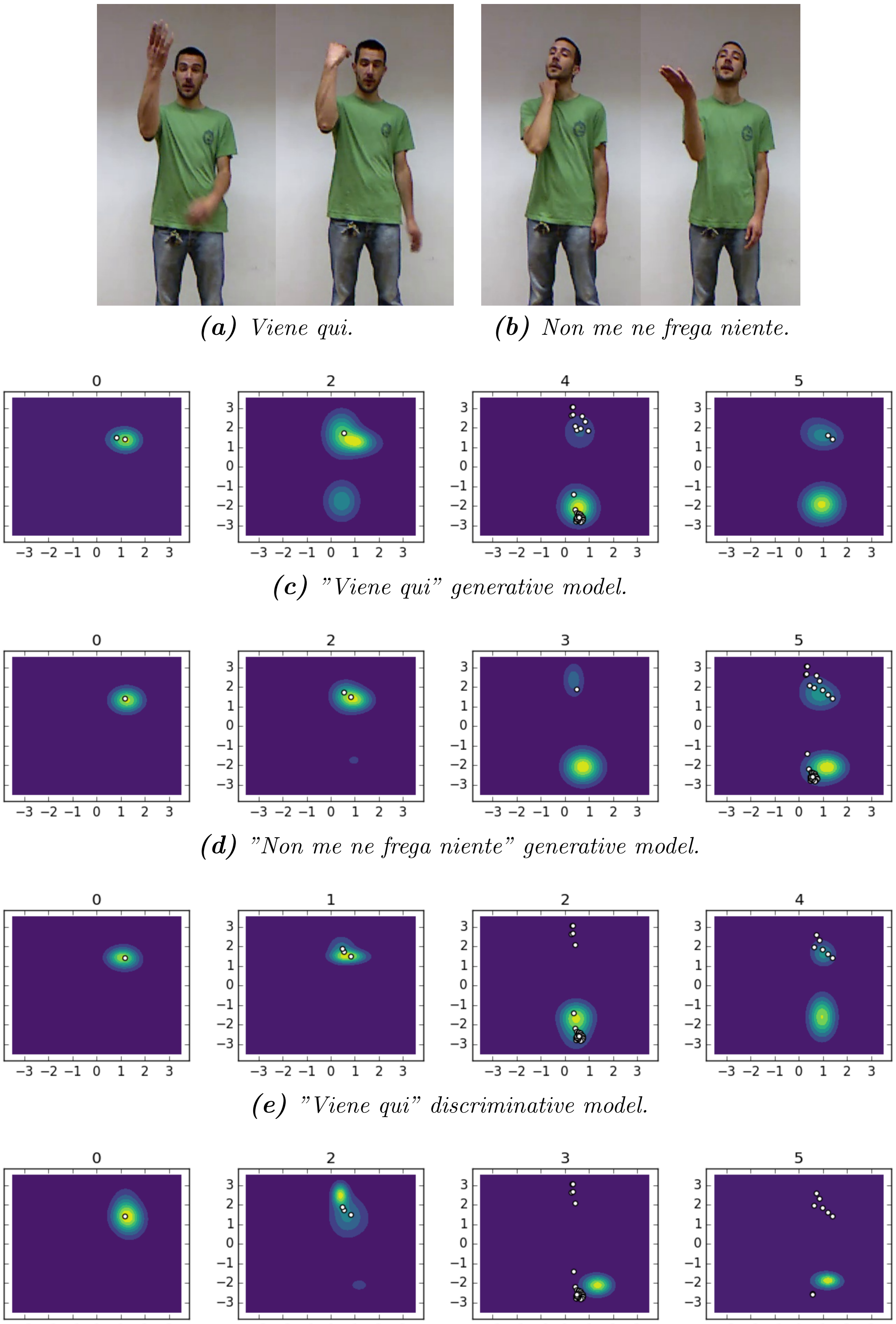
(c) *"Le vuoi prendere" generative model.*

(d) *"Che vuoi" generative model.*

(e) *"Le vuoi prendere" discriminative model.*

(f) *"Che vuoi" discriminative model.*

Figure 5.4: *The figures show, from the top: the characteristic poses, the Viterbi-clustering and accompanying distribution plots for the classes "Le vuoi prendere" and "Che vuoi". The data points come from a sample in the validation set where "Le vuoi prendere" is misclassified as "Che vuoi" in the generative case, and classified correctly in the discriminative case. The plots show the Y-Z dimensions of the head to right-hand distance features, which are the last two elements of the feature vector* (4.12).



(a) *Viene qui.*

(b) *Non me ne frega niente.*

(c) *"Viene qui" generative model.*

(d) *"Non me ne frega niente" generative model.*

(e) *"Viene qui" discriminative model.*

(f) *"Non me ne frega niente" discriminative model.*

Figure 5.5: *The figures show, from the top: the characteristic poses, the Viterbi-clustering and accompanying distribution plots for the classes "Viene qui" and "Non me ne frega niente". The data points come from a sample in the validation set where "Viene qui" is misclassified as "Non me ne frega niente" in the generative case, and classified correctly in the discriminative case. The plots show the Y-Z dimensions of the head to right-hand distance features, which are the last two elements of the feature vector* (4.12).

5.2.3 Findings

The plots in figures 5.3, 5.4 and 5.5 show the Y-Z dimensions of the head-to-hand relavite vector, the last two elements of the feature vector (4.12). The rest of the features are angles and result in very compact distribution plots. The distributions from the last two elements were plotted bacause they were deemed the most interesting visually. The distributions contain information obout the translation of the right hand in the Y-Z plane. All the gestures in question contain translations away from the body along the Z-axis. Figure 5.3 shows two distinct blobs in the Z direction. Notice that for "Perfetto" and "Non me ne frega niente" there is a translation of the right hand away from the body into the direction of the negative Z-axis. This can be seen as the distributions in the positive Z-axis region are stronger for the beginning states and the distributions in the negative Z-axis region are stronger at the ending states for both gestures. This is also the case for gestures "Viene qui" and "Non me ne frega niente" in figure 5.5. In figure 5.4, the effect of the sideward diagonal chopping motion of the gesture "Le vuoi prendere" can be seen as distributions are elongated slightly along the Y-axis.

The distribution plots in figures 5.3, 5.4 and 5.5 show that the majority of the points get clustered into the final states. This is a result of using LRB models, since the transitional structure forces the observations to move up on the list of ascending states. They also show that most of the distributional change seems to be found on the competitor-class model states, relative to the data points belonging to the appropriate class. Further, it can be seen that distributions tend to get smaller and more concentrated as well as translate slightly, especially at the ending states. Some of the final state distributions have translated in such a way that data points lie outside (or rather weakly inside) of distributions for both correct and incorrect classes. Accordingly, the likelihood score of the entire sequence decreases on account of the unrepresented data points. In these cases the data points are represented slightly better for the correct class.

In light of these findings it is concluded that the distributional changes brought on by MCE training, at least to the emission models, generally cause lower likelihoods on the data points, even for those belonging to the "correct" class. These distributional changes ensure that for any class, the training data points are better represented than for competitors.

5.3 Decorrelated features

The cepstral coefficients used as features in the majority of speech recognition systems can be assumed to be approximately decorrelated (Fink, 2014). The results presented on MCE training in Juang et al. (1997) employ such features. Decorrelation improves the conditioning of an optimisation as it removes dependencies between feature dimensions, which causes more efficient steps in the parameter space (LeCun et al., 2012). A PCA transform is used to decorrelate features in this study.

5.3.1 Hypothesis

PCA decorrelation is considered a feature optimisation, especially when combined with a whitening operation. It can be seen from comparing the confusion matrices in figures 4.8 and 4.9, that PCA-whitening improved the performance of the ML classifier using feature set (4.12). It is hypothesised that decorrelation improves the robustness of the optimisation and allows reducing the error function down to a lower minimum given sufficient model capacity.

5.3.2 Experiment

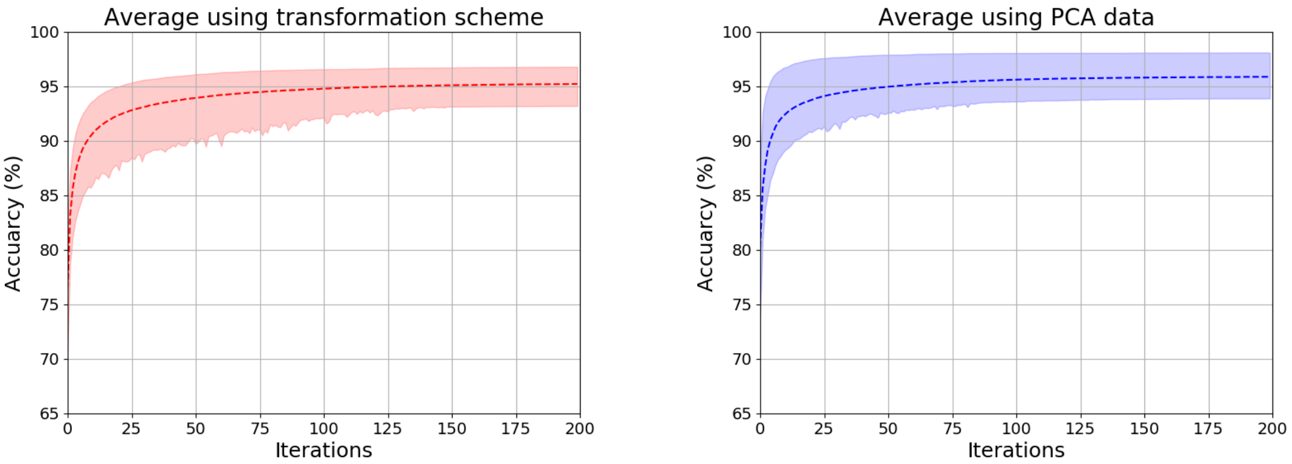
The data transformation scheme described in section 3.2.1 is a way of normalising the training data without applying a PCA-whitening transform to the training, validation and testing sets. It translates and scales to produce zero-mean and unit-variance training data while preserving the correlation among the feature dimensions, and leaves the validation and testing sets unchanged. To accomplish this two model sets are kept in memory during the optimisation, one transformed to fit the new training data and one for the original feature space. For every MCE-GPD iteration, the transformed model set is first updated, and then the original feature space model set is updated from it using the reverse scaling and translation. The reverse transformation of the models for every update should incur some error. However the optimisation, being stochastic, uses noisy updates anyway and is seen in section 5.1 to also optimise validation set performance. Given PCA-transformed data the model set transformations are not required, and the data already has a zero-mean and unit-variance. Regarding the comparison of the data transformation scheme to PCA-transformed data, the study makes two assumptions: that the error due to the model set transformations is negligible, and that the two cases have equivalent normalisation of the training data. It is therefore reasoned that the difference in optimisation friendliness and performance between MCE runs using feature set (4.12) and its PCA counterpart is purely due to correlation among feature dimensions.

To test the hypothesis that decorrelation allows higher asymptotic maxima for sufficient complexity, MCE optimisations are run for the transformation scheme and for PCA data across a series of model structures. The experiment is done in the same multiple run fashion as in section 5.1. Structures with fewer parameters will have higher bias error than that of larger structures. The bias error will eventually close in on some irreducible error value as the parameter set size increases. If the hypothesis is correct, the PCA runs will land on a lower irreducible error for the largest parameterisations. At the same time this experiment tests the effect of decorrelation on the robustness of the optimisation, as a common set of optimisation parameters are used. A constant set of optimisation parameters will cause step sizes to be a bit too large for some structures, and a bit too small for others. Generally, the training curves will be notably more noisy for the lower complexity models and smoother for the higher complexity models. The higher complexity models have been found in the study to be more robust in their optimisation. If the hypothesis is correct the PCA-optimisation will reach its asymptotic minimum in fewer iterations and its lower parameterised runs will generally be less noisy.

Average curves are calculated from all runs and stuctures and are plotted in figure 5.6 for the transformation scheme and PCA-data respectively. The shaded area is enveloped by the lowest and highest number-of-parameter structures, and shows the degree of variation.

5.3.3 Findings

Figure 5.6 shows that indeed the asymptotic maxima (which is just unity minus the minima multiplied by one hundred) are higher for the high number of parameter sets of the PCA-optimisations. This can be seen from the upper surfaces of the shaded areas. An interesting thing to note from the curves is that there seems to be a closer grouping of final maxima for the transformation scheme. It indicates that some of the lower parameterised models have a lower final training error than those using PCA data. This is likely due to the inter-dimensional correlations being learned by the emission distributions of the models. From the curves it can also be seen that for PCA-transformed data the asymptotic maxima are reached earlier. This can be seen from the average curves (the dashed lines) on both figures. Looking at the bottom ends of the shaded areas, the optimisations are generally less noisy in the PCA case. It is concluded that decorrelation improves the optimisation robustness and the post optimisation performance of the classifier.



(a) *MCE runs for multiple structures using the data transformation on feature set* (4.12).

(b) *MCE runs for multiple structures using the PCA-whitened version of feature set* (4.12).

Figure 5.6: *The figures show the MCE training curves for multiple model structures. The structures are formed from all combinations of the following hyperparameters:* $N = \{4, 5, 6, 7, 8\}$ *and* $K = \{3, 4, 5, 6, 7, 8\}$. *The runs use the sparsest update and a common set of optimisation parameters based on the linear annealing schedule described in section 3.2.4. The parameters are* $\epsilon_0 = 0.05$, $n_{max} = 200$ *epochs and* $\theta = 0.2$.

5.4 The slope value of the sigmoid function

So far no over-training has been seen in the MCE experiments. What was seen instead is a plateau of both the training and validation error. It is known that although SGD reaches an asymptotic minimum rather quickly, it takes long to converge relative to deterministic gradient descent (Bottou, 2012). As an explanation for the lack of over-training, it is worth considering that the SGD operations might hop around the minimum and not reach the point where it learns the training noise, despite the substantial improvement on the validation performance. One might think that by adding capacity to the models, the training noise will eventually be learned. This assumption makes sense, however it has not been encountered for the current problem. Figure 5.7 shows the training curves of a very high-complexity model set. The optimisation was not run to convergence, but stopped after 1000 epochs. The values of the training curve indicate that the optimisation is still minimising training error, however the curve also indicates a problem with step size. For the training a small constant step size was used in an attempt to run brute force until convergence. Unfortunately it seems that the small step size at some point remains too large to allow for further fine tuning. An important point to mention is that the optimal step size decay is difficult to find. There are sophisticated methods from the neural networks literature that find the principal eigenvalue of the Hessian using only gradient information. LeCun et al. (2012) cite a method that finds an approximation to the gradient-Hessian product using finite differences and the power-method, Pearlmutter (1994) presents a method to find an exact version of the product, analitically, using a differential operator that can be applied to the graidient equations. These methods, however, rely on stable gradients and are therefore mostly meant for deterministic descent since stochastic gradients are noisy. The optimisation considered here uses no second-order information, and this leads to an optimisation that is difficult to parameterise well when convergence is expected.

As explained earlier, lowering the slope of the sigmoid loss function means that the function no longer directly corresponds to training error. However, it was also found earlier that lowering the slope to a value of $\gamma = 0.1$ yielded an objective function that is easier to optimise while still decreasesing the validation error, which employs $\gamma = 1$, along a similar trajectory as the training error. Schluter and Macherey (1998) use a probability scaling constant ρ in the form P^{ρ} to improve the convergence of MMI training for speech recognition. It is used as a standard method and is known to lead to good test-set performance (Povey, 2005). If such a scaling constant is applied to MCE training using the sparsest update (function D in table 5.1), it is equivalent to lowering the central slope of the sigmoid. The argument is briefly developed below. If the scaled probability is substituted into the discriminant function, it becomes

$$g_i(X, \Lambda) = \log P(X|\lambda_i)^{\rho} = \rho g_i(X, \Lambda). \tag{5.5}$$

If this is substituted into the distance measure, it becomes

$$d_i(X, \Lambda) = -\rho g_i(X, \Lambda) + log \left[\frac{1}{M-1} \sum_{j \neq i} e^{g_j(X,\Lambda)\rho\eta} \right]^{\frac{1}{\eta}}, \tag{5.6}$$

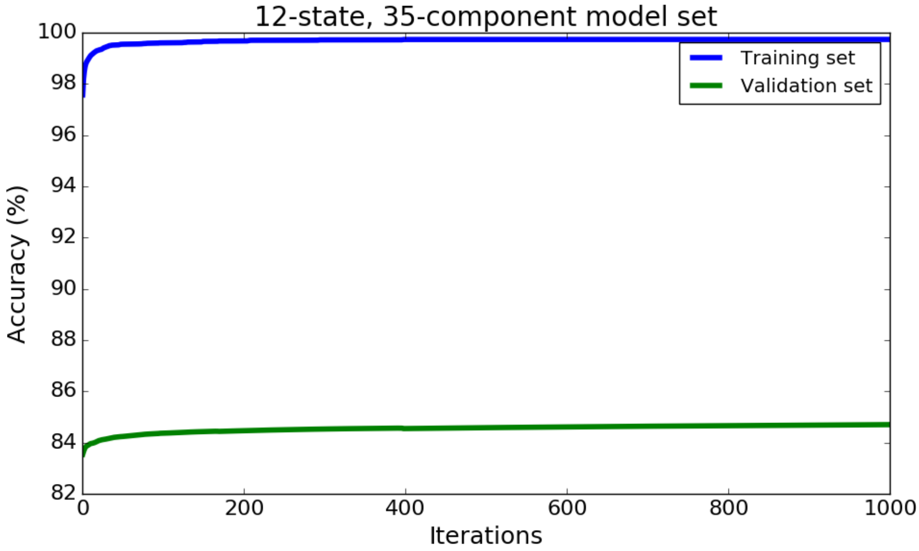


Figure 5.7: *A figure showing the training curves for a massive twelve state, thirty-five mixture-component model set. The training was run for 1000 epochs and stopped before convergence. The final training and validation accuracies are 99.74% and 84.7% respectively.*

where the infinity-norm case is

$$d_i(X,\Lambda) = -\rho g_i(X,\Lambda) + \rho \max_{j\neq i}(g_j(X,\Lambda)) = \rho d_i(X,\Lambda). \tag{5.7}$$

Embedded in the loss function the slope value is then directly multiplied by the scaling constant as

$$\ell_i(X,\Lambda) = \frac{1}{1+e^{-\gamma\rho d_i(X,\Lambda)}}. \tag{5.8}$$

The scaling constant does not behave as well in the η-norm case, as can be seen from the distance measure to model parameter derivative. In the presence of the scaling constant the distance measure gradient (see appendix A) becomes

$$\begin{aligned}
\frac{\partial d_i(X)}{\partial\theta} &= -\rho\frac{\partial g_i(X)}{\partial\theta} + \frac{1}{\eta}\frac{\partial}{\partial\theta}\left[\log\left(\frac{1}{M-1}\right) + \log\sum_{j\neq i} e^{g_j(X)\rho\eta}\right] \\
&= -\rho\frac{\partial g_i(X)}{\partial\theta} + \frac{1}{\eta}\left[\frac{\sum_{j\neq i} e^{g_j(X)\rho\eta}\frac{\partial g_j(X)}{\partial\theta}\rho\eta}{\sum_{j\neq i} e^{g_j(X)\rho\eta}}\right] \\
&= -\rho\frac{\partial g_i(X)}{\partial\theta} + \frac{\sum_{j\neq i} e^{g_j(X)\rho\eta}\frac{\partial g_j(X)}{\partial\theta}\rho}{\sum_{j\neq i} e^{g_j(X)\rho\eta}} \\
&= \begin{cases} -\rho\frac{\partial g_i(X)}{\partial\theta} & \theta\in\lambda_i \\ \rho\phi_{i,k}(X)\frac{\partial g_k(X)}{\partial\theta} & \theta\in\lambda_{k\neq i}, \end{cases}
\end{aligned} \tag{5.9}$$

where

$$\phi_{i,k}(X) = \frac{e^{g_k(X)\rho\eta}}{\sum_{j\neq i} e^{g_j(X)\rho\eta}}. \tag{5.10}$$

The value η weights the contribution of a competitor model by a function of its score on the input X. The value ρ now also influences this contribution function so that equal contributions require the product of η and ρ to be unity. However, for the the infinity norm case, such as the sparsest update, the scaling is equivalent to relaxing the sigmoid slope. The formulations above are intended to justify the lowering of the sigmoid function by relating it to a well known scaling term from the MMI literature. The lowered sigmoid slope is already used throughout all MCE experiments in the study.

5.4.1 Hypothesis

The HMM evaluation functions (5.3) and (5.4), when used on long observation sequences, tend to result in small probabilities since those sequences imply long multiplication chains. When these probabilities are represented in log-scale, the dynamic range of the misclassification distances will be large. For the gesture data considered, log-probabilities of around -1000 are not uncommon. The problem of gesture data might call for a much lower slope of the logistic function, so as to broaden the range of barely-correct and near-miss examples reflected in the gradients. Figure 5.8 shows the sigmoid functions for different values of the central slope γ. In the large range of values $d_i(X, \Lambda)$ can occupy, it can be seen that $\gamma = 1$ is a good approximation to a misclassification counter function, but that it does not acceptably reflect in the range of barely-correct or near-miss cases. Figure 5.9 on the other hand shows the gradients of those loss functions in terms of misclassification distance value $d_i(X, \Lambda)$. Essentially it shows that lowering the γ value broadens the range of distances that will reflect gradients. Here it can be seen why an optimiser for a slope value of $\gamma = 1$ is almost impossible to choose parameters for, given large fluctuations in $d_i(X, \Lambda)$.

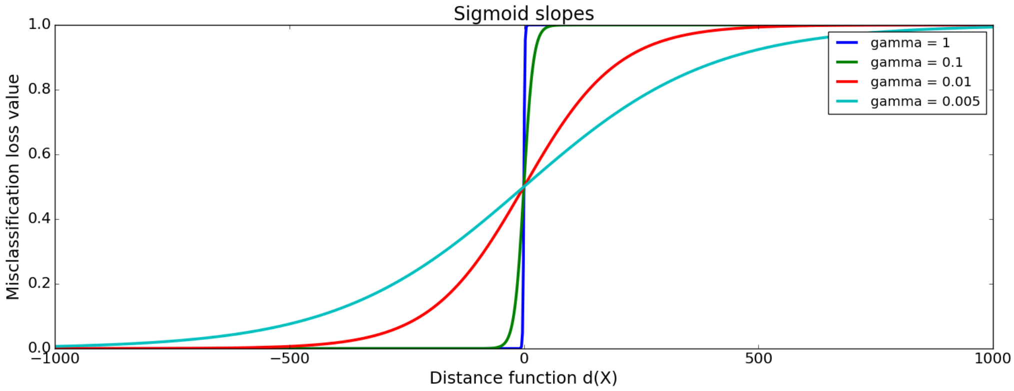


Figure 5.8: *A figure showing the sigmoid function for three values of gamma.*

It is anticipated that γ values lower than 0.1 will not only yield more optimisation-friendly objective functions, but will still lead the validation performance and further more, allow over-training to be seen on the validation set. It is hypothesised that as the value of the slope is decreased, better performance on the validation set will

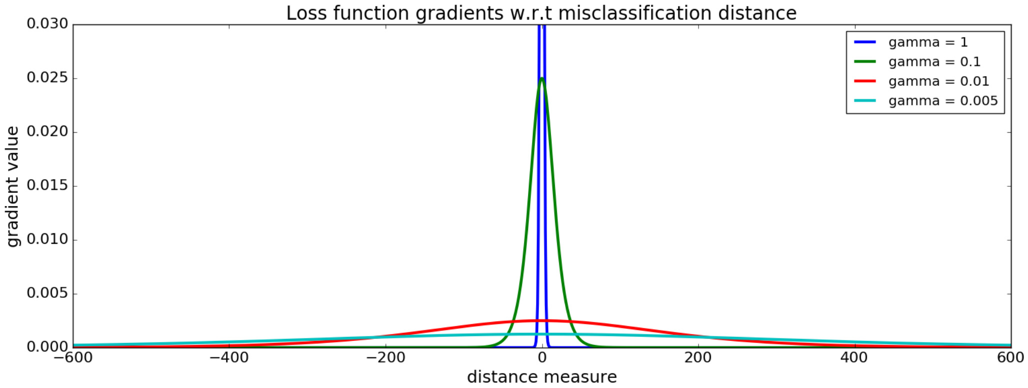


Figure 5.9: *A figure showing the gradients of the loss function w.r.t the misclassification distance measure for different values of γ.*

be seen up to some optimal point. Thereafter validation performance will again decrease.

5.4.2 Experiment

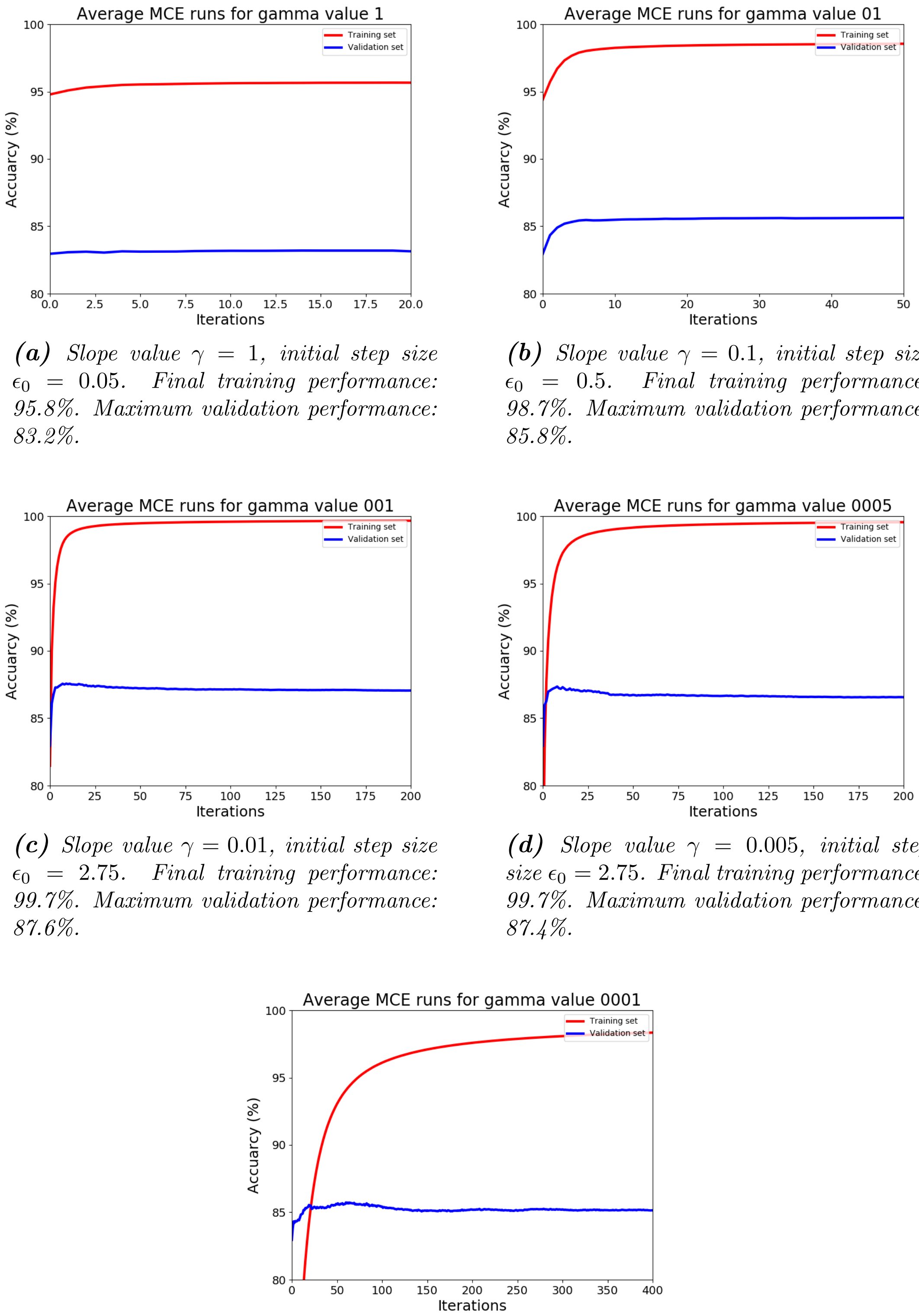
To investigate the effects of relaxing the sigmoid slope on optimisation performance, a series of MCE optimisations were set up using the same model set with varying slope values. The experiment is done in the same multiple-run fashion as before, and the averaged training curves accross the runs are plotted. The plots will show over-training behaviour and higher maxima for lowered slope values if the hypothesis is correct. When the slope is notably further decreased, over-training behaviopur is expencted but at lower maxima, indicating that the function being learned no longer corresponds with the training set error. There then clearly exists an optimal value somewhere in between. For the experiment in figure 5.10, a twelve-state, eleven-mixture component model set was used. This time a decay function of

$$\epsilon_n = \frac{\epsilon_0}{1 + \omega n} \tag{5.11}$$

was used for the step size annealing, where ω is a small positive constant. The parameters ϵ_0 and ω were tailored to each γ value.

5.4.3 Findings

Figure 5.10 shows the optimisation curves for five MCE runs with varying slope values. From the curves it is clear that the value $\gamma = 1$ does not optimise performance on the training or validation sets. The value $\gamma = 0.1$ (used throughout all previous MCE experiments) does optimise the training and validation performance, but does not over-train. The value $\gamma = 0.01$ shows over-training with the validation optimum around the tenth iteration. It also achieves the highest validation performance of 87.6%, indicating that the increase in conditioning helps to dig out a little more information out of the training data. The value $\gamma = 0.01$ is thus the empirical optimum slope value. The value $\gamma = 0.005$ shows very clear over-training and a



(a) *Slope value* $\gamma = 1$, *initial step size* $\epsilon_0 = 0.05$. *Final training performance: 95.8%. Maximum validation performance: 83.2%.*

(b) *Slope value* $\gamma = 0.1$, *initial step size* $\epsilon_0 = 0.5$. *Final training performance: 98.7%. Maximum validation performance: 85.8%.*

(c) *Slope value* $\gamma = 0.01$, *initial step size* $\epsilon_0 = 2.75$. *Final training performance: 99.7%. Maximum validation performance: 87.6%.*

(d) *Slope value* $\gamma = 0.005$, *initial step size* $\epsilon_0 = 2.75$. *Final training performance: 99.7%. Maximum validation performance: 87.4%.*

(e) *Slope value* $\gamma = 0.001$, *initial step size* $\epsilon_0 = 2.75$. *Final training performance: 98.8%. Maximum validation performance: 85.7%.*

Figure 5.10: *Five runs of MCE-GPD training using varying gamma values. The optimisations were performed on the PCA version of feature-set* (4.12) *using the sparsest update. Eight runs were made for every value. The average curves for all runs are represented above.*

higher validation performance than that of $\gamma = 0.1$, but slightly lower than that of $\gamma = 0.01$. When comparing the training curves of $\gamma = 0.01$ and $\gamma = 0.005$, the latter case looks slightly over-smoothed. This is obvious in the case of $\gamma = 0.001$, where the training performance function surpasses the validation performance function around the twentieth iteration. In this case the highest validation performance is 85.7%, just below that for $\gamma = 0.1$. It is concluded that a lower slope value is both easier to optimise and achieves better performance, where the empirical optimum is $\gamma = 0.01$.

5.5 Best result

The experiment in section 5.1 determined that the sparsest update can be used as a faster MCE optimisation for large problems, despite that the underlying function is actually discontinuous. Section 5.2 attempts to elucidate the underlying mechanism that allows minimum classification error of the training data. Section 5.3 shows how PCA-decorrelation can improve performance of MCE-training and the final classifier. Section 5.4 compares the value of the sigmoid slope to a known scaling constant in MMI-training for speech recognition and shows that it is related to the dynamic range of the misclassification measure. An empirical best slope value of $\gamma = 0.01$ was found.

Using the accumulated findings of previous experiments, a best result model set was trained and tested. The PCA-version of feature set (4.12), and a twelve-state, eleven-mixture component model set was used. Further, a slope value of $\gamma = 0.01$ and the decay function (5.11) for step size annealing was used. The decay function used the values $\epsilon_0 = 2.75$ and $\omega = 1/P$, where P is the number of training samples (see (2.18)). Using an early stop, the result in figure 5.11 was produced. The model set achieves a test set accuracy of 87.3%. This result is noteworthy because it competes well with two powerful methods from other studies. Liang and Zheng (2014) use RBF-SVMs on holistic features of a skeletal gesture sequence. The mean, variance, minimum and maximum of their skeletal feature vector is accumulated over the frames of a segmented gesture. It is a very dense feature set that, given good segmentation, should be quite discriminative. On the segmented test set this method achieves 83.02%. Evangelidis et al. (2014a) perform SVM classification on a 1536-dimensional Fisher vector representation calculated from a sliding window of skeletal features. This is a powerful discriminative representation that uses an underlying probabilistic model. The method achieves 90% on the isolated classification of the test set.

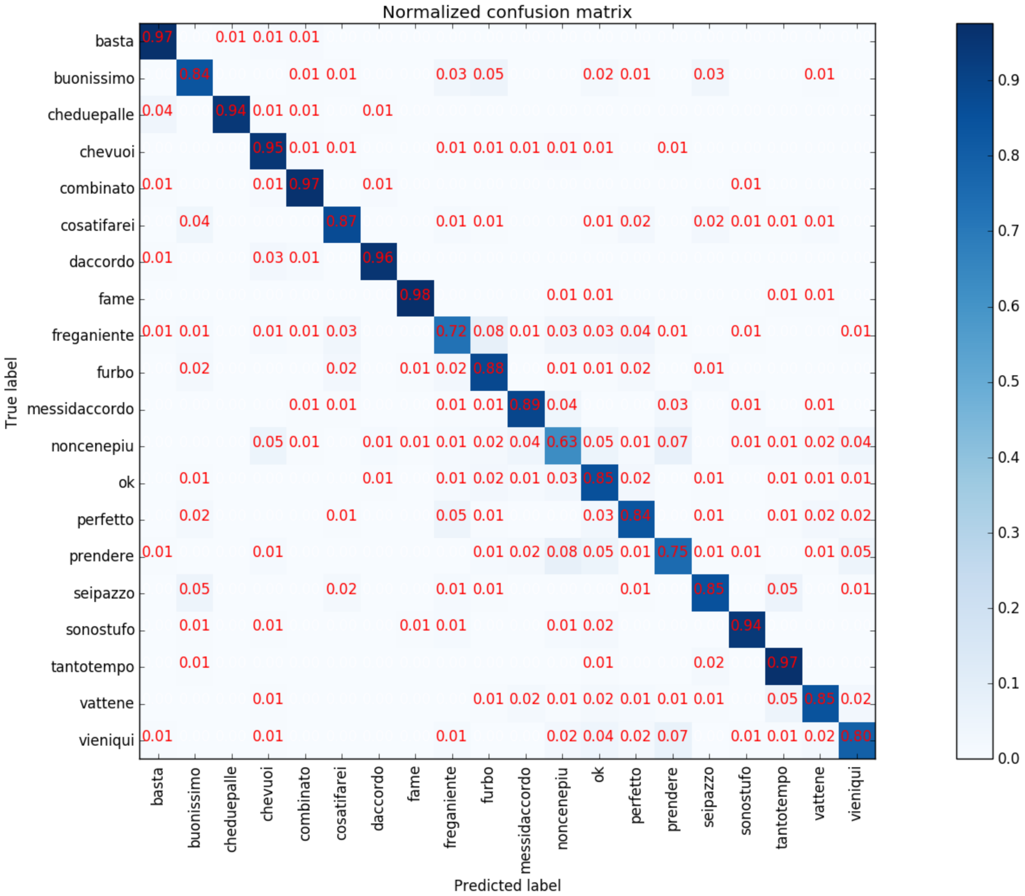


Figure 5.11: *A test set confusion matrix for the best performing model set. The model set uses uniform twelve-state, eleven-mixture component models. Accuracy:* 87.3%.

Chapter 6

Conclusions

There are very few examples of discriminative HMMs being used in the gesture recognition literature. This study successfully implements discriminative HMMs for the recognition of fine grained dynamic hand gestures. The dataset used is intended for gesture recognition from multiple modalities, and because only the skeleton modality is used, many gestures are very similar given their features. From the discussion on other work performed on the dataset in section 4.2, the top-performing teams seemed to have used quite high-dimensional feature descriptors for classification on the skeleton stream. Evangelidis et al. (2014a) use 1536-dimensional Fisher vectors calculated from windows of so called quads as features for SVM classification. These vectors do contain the temporal signature of the gestures, but are still very high-dimensional as descriptors. Relatively, this study employs a very small feature vector of only fourteen dimensions while performing competitively.

Given only skeletal features, the maximum likelihood classifier in this study suffers from distributional overlap. This can be seen from the first confusion matrix in chapter 4. The confusion matrices in the rest of the chapter show the results of a ML-based mitigation strategy, one of model selection for the maximum likelihood of each respective class's validation data. Chapter 5 details the results of discriminative training as a mitigation. It is clear that if distributional overlap is a threat, then discriminative training offers a reasonable solution. Section 5.2 attempts to visualise the differences in the emission distributions that account for the increases in classifier accuracy. In the plots, the distributional overlap problem is clearly illustrated. From the difference in distributions an important observation was made. That despite that MCE training lowers the overall likelihoods, the likelihood of the "correct" model will be higher than its competitors. This is because the embedded MAP rule still requires the "correct" model to have the highest likelihood. It is argued accordingly that other uses such as Viterbi decoding over sequences of gestures could also benefit from MCE training.

This study successfully detailed the design of a maximum likelihood and a discriminative HMM gesture classifier. The study has shown that a discriminative HMM classifier is a good solution to an isolated gesture problem in the light of feature similarity. It does so with a performance result of 87.3% accuracy being achieved on an over 4000-sequence test set using the skeletal modality alone. This result compares to the 83.02% reported in Liang and Zheng (2014) and the 90% reported in Evangelidis et al. (2014a). The study further serves as a roadmap for implementing a similar gesture classifier.

Given the evaluation of this study's methodology on a gesture dataset that is designed for simultaneous spotting and recognition, it is regrettable that automatic segmentation and recognition using some variant of the Viterbi algorithm could not be investigated. It is after all the main reason for the use of HMMs in speech recognition. The study thus recommends as a direction of further enquiry the incorporation of a Viterbi-based segmentation and recognition strategy. It would facilitate a much wider comparison to other work on the dataset, and if co-articulation needs to be modelled, there are sentence level forms of the MCE objective function in the literature.

As low-cost depth sensors become increasingly more available and the technology therein improves, more gesture datasets with more sophisticated skeletal articulation will surface. Better function approximation methods that would otherwise be too data-hungry will become more viable, such as the use of convolutional neural networks in conjunction with HMMs, as is the case in speech recognition.

Appendix A

Derivatives for the Minimum Classification Error function for Stochastic Gradient Descent

This appendix deals with the derivatives of the various forms of the MCE loss function and some underlying derivations.

A.1 The loss function

The expected loss per sample X is considered for SGD:

$$\begin{aligned} L(\Lambda) &= E[\ell(X;\Lambda)] \\ &= \ell_i(X;\Lambda) \qquad X \in C_i. \end{aligned} \tag{A.1}$$

The loss function gradient

$$\frac{\partial \ell_i(X)}{\partial \theta} = \frac{\partial \ell_i(X)}{\partial d_i(X)} \frac{\partial d_i(X)}{\partial \theta} \tag{A.2}$$

can belong to any model in the classifier Λ. The first term in the chain-rule expansion is the derivative of the loss function with respect to the misclassification measure

$$\frac{\partial \ell_i(X)}{\partial d_i(X)} = \gamma \ell_i(X) \left(1 - \ell_i(X)\right). \tag{A.3}$$

A.2 Misclassification measures

The second term in the expansion (A.2) is the misclassification measure derivative. The misclassification function has two cases considered in the study, the max and η-norm functions.

A.2.1 Max norm case

When the norm value $\eta \longrightarrow \infty$ is used, only the highest scoring competitor model is considered in the misclassification measure. The derivative is as follows:

$$\begin{aligned} d_i(X) &= -g_i(X) + \max_{j \neq i}(g_j(X)) \\ &= -g_i(X) + g_k(X) \\ \frac{\partial d_i(X)}{\partial \theta} &= \begin{cases} -\frac{\partial g_i(X)}{\partial \theta} & \theta \in \lambda_i \\ \frac{\partial g_k(X)}{\partial \theta} & \theta \in \lambda_k \\ 0 & \theta \ni \lambda_i, \lambda_k. \end{cases} \end{aligned} \tag{A.4}$$

A.2.2 General η-norm case

In the general case, the value η weights the contribution of a competitor model by a function of its score on the input X. The derivative is as follows:

$$\begin{aligned} \frac{\partial d_i(X)}{\partial \theta} &= -\frac{\partial g_i(X)}{\partial \theta} + \frac{1}{\eta}\frac{\partial}{\partial \theta}\left[\log\left(\frac{1}{M-1}\right) + \log\sum_{j \neq i} e^{g_j(X)\eta}\right] \\ &= -\frac{\partial g_i(X)}{\partial \theta} + \frac{1}{\eta}\left[\frac{\sum_{j \neq i} e^{g_j(X)\eta}\frac{\partial g_j(X)}{\partial \theta}\eta}{\sum_{j \neq i} e^{g_j(X)\eta}}\right] \\ &= -\frac{\partial g_i(X)}{\partial \theta} + \frac{\sum_{j \neq i} e^{g_j(X)\eta}\frac{\partial g_j(X)}{\partial \theta}}{\sum_{j \neq i} e^{g_j(X)\eta}} \\ &= \begin{cases} -\frac{\partial g_i(X)}{\partial \theta} & \theta \in \lambda_i \\ \phi_{i,k}(X)\frac{\partial g_k(X)}{\partial \theta} & \theta \in \lambda_{k \neq i}, \end{cases} \end{aligned} \tag{A.5}$$

where

$$\phi_{i,k}(X) = \frac{e^{g_k(X)\eta}}{\sum_{j \neq i} e^{g_j(X)\eta}}. \tag{A.6}$$

A.3 Discriminant functions

Regarding the discriminant function, the study considers two cases: the standard HMM evaluation function and the Viterbi probability.

A.3.1 Viterbi probability

In the standard HMM evaluation function, the dominant term in the summation over all possible state paths is the joint observation and path probability along the single most likely path:

$$\begin{aligned} g_i(X, \Lambda) = \log P(X, \bar{\boldsymbol{q}}|\lambda_i) &= \log P(X|\bar{\boldsymbol{q}}, \lambda_i) + \log P(\bar{\boldsymbol{q}}|\lambda_i) \\ &= \log \pi_{\bar{q}_1} + \sum_{t=1}^{T-1} \log a_{\bar{q}_t, \bar{q}_{t+1}} + \sum_{t=1}^{T} \log b_{\bar{q}_t}(\boldsymbol{x}_t). \end{aligned} \tag{A.7}$$

Transformed parameter gradients

Using the chain-rule expression

$$\frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \tilde{\theta}} = \frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \theta} \frac{\partial \theta}{\partial \tilde{\theta}}, \tag{A.8}$$

the transformed parameter derivatives are found as:

$$\frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \tilde{a}_{ij}} = \sum_{t=1}^{T-1} \delta(\bar{q}_t, i)\delta(\bar{q}_{t+1}, j)(1 - a_{ij}), \tag{A.9}$$

$$\frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \tilde{c}_{jk}} = \sum_{t=1}^{T} \delta(\bar{q}_t, j)\frac{\mathcal{N}(\boldsymbol{x}_t, \bar{\mu}_{jk}, \bar{\sigma}_{jk})}{b_j(\boldsymbol{x}_t)} c_{jk}(1 - c_{jk}), \tag{A.10}$$

$$\frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \tilde{\mu}_{jkd}} = \sum_{t=1}^{T} \delta(\bar{q}_t, j)\frac{\mathcal{N}(\boldsymbol{x}_t, \bar{\mu}_{jk}, \bar{\sigma}_{jk})}{b_j(\boldsymbol{x}_t)} c_{jk} \left(\frac{x_t^d - \mu_{jkd}}{\sigma_{jkd}}\right), \tag{A.11}$$

$$\frac{\partial \log P(X, \bar{\boldsymbol{q}}|\lambda)}{\partial \tilde{\sigma}_{jkd}} = \sum_{t=1}^{T} \delta(\bar{q}_t, j)\frac{\mathcal{N}(\boldsymbol{x}_t, \bar{\mu}_{jk}, \bar{\sigma}_{jk})}{b_j(\boldsymbol{x}_t)} c_{jk} \left[\left(\frac{x_t^d - \mu_{jkd}}{\sigma_{jkd}}\right)^2 - 1\right]. \tag{A.12}$$

A.3.2 General sequence probability

The general evaluation function case of the discriminant function is now considered. First the standard parameter derivatives are found and then from them the transformed parameter derivatives. The general evaluation function is as follows:

$$\begin{aligned} P(X|\lambda) &= \sum_{\boldsymbol{q}} P(X, \boldsymbol{q}|\lambda) \\ &= \sum_{\boldsymbol{q}} P(X|\boldsymbol{q}, \lambda)P(\boldsymbol{q}|\lambda) \\ &= \sum_{\boldsymbol{q}} \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1},q_t} b_{q_t}(\boldsymbol{x}_t). \end{aligned} \tag{A.13}$$

To efficiently calculate the product summation of $P(X|\lambda)$, the forward process is used:

$$\alpha_1(i) = \pi_i b_i(\boldsymbol{x}_1) \qquad 1 \leq i \leq N \tag{A.14}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\boldsymbol{x}_{t+1}) \qquad 1 \leq t \leq T-1 \tag{A.15}$$

$$1 \leq i \leq N$$

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{A.16}$$

To state this process in a single differentiable equation, we use a more compact matrix notation. Let $\bar{\alpha}$, $\bar{\beta}$ be column vectors of the forward and backward variables at time t. The $N \times N$ transition matrix is here denoted as $\mathbf{A}$. Let $\mathbf{B}$ be a $N \times N$ diagonal matrix of the observation distributions at time t (Qin et al., 2000):

$$\mathbf{B}_t = \begin{bmatrix} b_1(\boldsymbol{x}_t) & & \\ & \ddots & \\ & & b_N(\boldsymbol{x}_t) \end{bmatrix}. \tag{A.17}$$

In this case the observation at time t, $\boldsymbol{x}_t$, is in fact a vector of continuous values and dimension D, $\boldsymbol{x}_t = [x_t^1, x_t^2, x_t^3, ..., x_t^D,]$. The emissions are modelled by multivariate mixture of Gaussian distributions

$$b_j(\boldsymbol{x}_t) = \sum_{k=1}^{K} c_{jk} \mathcal{N}(\boldsymbol{x}_t, \bar{\mu}_{jk}, \bar{\sigma}_{jk}). \tag{A.18}$$

The forward and backward recursions can now be expressed as follows:

$$\bar{\alpha}_{t+1}^{\tau} = \bar{\alpha}_t^{\tau} \mathbf{A}\mathbf{B}_{t+1} \tag{A.19}$$

$$\bar{\beta}_t = \mathbf{A}\mathbf{B}_{t+1}\bar{\beta}_{t+1}, \tag{A.20}$$

and the likelihood function as

$$\begin{aligned} P(X|\lambda) &= \bar{\alpha}_T^{\tau}\bar{u} \\ &= \bar{\pi}^{\tau}\mathbf{B}_1.\mathbf{A}\mathbf{B}_2.\mathbf{A}\mathbf{B}_3.\mathbf{A}\mathbf{B}_4...\mathbf{A}\mathbf{B}_T\bar{u}. \end{aligned} \tag{A.21}$$

The parameter derivative employs the product rule and the definitions of (A.19) and (A.20):

$$\begin{aligned} \frac{\partial P(X|\lambda)}{\partial\theta} &= \frac{\partial\bar{\pi}^{\tau}\mathbf{B}_1}{\partial\theta}\bar{\beta}_1 + \bar{\alpha}_1^{\tau}\frac{\partial\mathbf{A}\mathbf{B}_2}{\partial\theta}\bar{\beta}_2 + \bar{\alpha}_2^{\tau}\frac{\partial\mathbf{A}\mathbf{B}_3}{\partial\theta}\bar{\beta}_3 + ... + \bar{\alpha}_{T-1}^{\tau}\frac{\partial\mathbf{A}\mathbf{B}_T}{\partial\theta}\bar{\beta}_T \\ &= \frac{\partial\bar{\pi}^{\tau}\mathbf{B}_1}{\partial\theta}\bar{\beta}_1 + \sum_{t=1}^{T-1}\bar{\alpha}_t^{\tau}\frac{\partial\mathbf{A}\mathbf{B}_{t+1}}{\partial\theta}\bar{\beta}_{t+1}. \end{aligned} \tag{A.22}$$

Since the derivative of a matrix is a matrix of its differentiated elements, and θ is a specific parameter, the resulting matrix expressions contain only one non-zero term. Therefore the summations run only across the affected transitions of states. For derivatives w.r.t $\theta \in \{\pi_i, a_{ij}\}$, $i, j \in [1, N]$:

$$\begin{aligned}\frac{\partial P(X|\lambda)}{\partial \pi_i} &= \frac{\partial \bar{\pi}^\tau \mathbf{B}_1}{\partial \pi_i}\bar{\beta}_1 \\ &= \beta_1(i) b_i(\boldsymbol{x}_1) \end{aligned} \tag{A.23}$$

$$\begin{aligned}\frac{\partial P(X|\lambda)}{\partial a_{ij}} &= \sum_{t=1}^{T-1} \bar{\alpha}_t \frac{\partial \mathbf{AB}_{t+1}}{\partial a_{ij}}\bar{\beta}_{t+1} \\ &= \sum_{t=1}^{T-1} \alpha_t(i) b_j(\boldsymbol{x}_{t+1})\beta_{t+1}(j). \end{aligned} \tag{A.24}$$

For derivatives w.r.t $\theta \in \{c_{jk}, \mu_{jkd}, \sigma_{jkd}\}$, $j \in [1, N]$, $k \in [1, K]$, $d \in [1, D]$:

$$\frac{\partial P(X|\lambda)}{\partial \theta} = \frac{\partial P(X|\lambda)}{\partial b_j(\boldsymbol{x}_t)} \frac{\partial b_j(\boldsymbol{x}_t)}{\partial \theta} \tag{A.25}$$

$$\begin{aligned}\frac{\partial P(X|\lambda)}{\partial b_j(\boldsymbol{x}_t)} &= \frac{\partial \bar{\pi}^\tau \mathbf{B}_1}{\partial b_j(\boldsymbol{x}_1)}\bar{\beta}_1 + \sum_{t=1}^{T-1} \bar{\alpha}_t^\tau \frac{\partial \mathbf{AB}_{t+1}}{\partial b_j(\boldsymbol{x}_{t+1})}\bar{\beta}_{t+1} \\ &= \frac{\partial \bar{\pi}^\tau \mathbf{B}_1}{\partial b_j(\boldsymbol{x}_1)}\bar{\beta}_1 + \sum_{t=2}^{T} \bar{\alpha}_{t-1}^\tau \frac{\partial \mathbf{AB}_{t}}{\partial b_j(\boldsymbol{x}_{t})}\bar{\beta}_{t} \\ &= \pi_j \beta_1(j) + \sum_{t=2}^{T}\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \beta_t(j) \\ &= \pi_j \left(\frac{b_j(\boldsymbol{x}_1)}{b_j(\boldsymbol{x}_1)}\right)\beta_1(j) + \sum_{t=2}^{T}\left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\boldsymbol{x}_t)\right]\left(\frac{1}{b_j(\boldsymbol{x}_t)}\right)\beta_t(j) \\ &= \frac{\alpha_1(j)\beta_1(j)}{b_j(\boldsymbol{x}_1)} + \sum_{t=2}^{T} \frac{\alpha_t(j)\beta_t(j)}{b_j(\boldsymbol{x}_t)} \\ &= \sum_{t=1}^{T} \frac{\alpha_t(j)\beta_t(j)}{b_j(\boldsymbol{x}_t)}. \end{aligned} \tag{A.26}$$

Transformed parameter gradients

$$\begin{aligned}
\frac{\partial \log P(X|\lambda)}{\partial \tilde{a}_{ij}} &= \frac{\partial \log P(X|\lambda)}{\partial P(X|\lambda)} \frac{\partial P(X|\lambda)}{\partial a_{ij}} \frac{\partial a_{ij}}{\partial \tilde{a}_{ij}} \\
&= \left(\frac{1}{P(X|\lambda)}\right) \left(\sum_{t=1}^{T-1} \alpha_t(i) b_j(\boldsymbol{x}_{t+1}) \beta_{t+1}(j)\right) a_{ij}(1 - a_{ij}) \\
&= \sum_{t=1}^{T-1} \xi_{ij}(1 - a_{ij}). \qquad \text{(A.27)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log P(X|\lambda)}{\partial \tilde{c}_{jk}} &= \frac{\partial \log P(X|\lambda)}{\partial P(X|\lambda)} \frac{P(X|\lambda)}{b_j(\boldsymbol{x}_t)} \frac{b_j(\boldsymbol{x}_t)}{c_{jk}} \frac{c_{jk}}{\tilde{c}_{jk}} \\
&= \left(\frac{1}{P(X|\lambda)}\right) \left(\sum_{t=1}^{T} \frac{\alpha_t(j)\beta_t(j)}{b_j(\boldsymbol{x}_t)}\right) \mathcal{N}(\boldsymbol{x}_t, \bar{\mu}_{jk}, \bar{\sigma}_{jk}) c_{jk}(1 - c_{jk}) \\
&= \sum_{t=1}^{T} \gamma_t(j,k) c_{jk}(1 - c_{jk}). \qquad \text{(A.28)}
\end{aligned}$$

$$\frac{\partial \log P(X|\lambda)}{\partial \tilde{\mu}_{jkd}} = \sum_{t=1}^{T} \gamma_t(j,k) c_{jk} \left(\frac{x_t^d - \mu_{jkd}}{\sigma_{jkd}}\right). \qquad \text{(A.29)}$$

$$\frac{\partial \log P(X|\lambda)}{\partial \tilde{\sigma}_{jkd}} = \sum_{t=1}^{T} \gamma_t(j,k) c_{jk} \left[\left(\frac{x_t^d - \mu_{jkd}}{\sigma_{jkd}}\right)^2 - 1\right]. \qquad \text{(A.30)}$$

Bibliography

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *proc. ICASSP*, volume 86, pages 49–52.

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer.

Campbell, L. W., Becker, D. A., Azarbayejani, A., Bobick, A. F., and Pentland, A. (1996). Invariant features for 3-d gesture recognition. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 157–162. IEEE.

Chou, W., Juang, B.-H., and Lee, C.-H. (1992). Segmental GPD training of HMM based speech recognizer. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 473–476. IEEE.

Escalera, S., Athitsos, V., and Guyon, I. (2017). Challenges in multi-modal gesture recognition. In *Gesture Recognition*, pages 1–60. Springer.

Escalera, S., Baró, X., Gonzalez, J., Bautista, M. A., Madadi, M., Reyes, M., Ponce-López, V., Escalante, H. J., Shotton, J., and Guyon, I. (2014). Chalearn looking at people challenge 2014: Dataset and results. In *Workshop at the European Conference on Computer Vision*, pages 459–473. Springer.

Escalera, S., Gonzàlez, J., Baró, X., Reyes, M., Lopes, O., Guyon, I., Athitsos, V., and Escalante, H. (2013). Multi-modal gesture recognition challenge 2013: Dataset and results. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 445–452. ACM.

Evangelidis, G., Singh, G., and Horaud, R. (2014a). Continuous gesture recognition from articulated poses. In *European Conference on Computer Vision Workshops*, pages 595–607.

Evangelidis, G., Singh, G., and Horaud, R. (2014b). Skeletal quads: Human action recognition using joint quadruples. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4513–4518. IEEE.

Fink, G. A. (2014). *Markov models for pattern recognition: from theory to applications*. Springer Science & Business Media.

Fohr, D., Mella, O., and Illina, I. (2017). New paradigm in speech recognition: Deep neural networks. In *IEEE International Conference on Information Systems and Economic Intelligence*.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42.

He, X., Deng, L., and Chou, W. (2008). Discriminative learning in sequential pattern recognition. *IEEE Signal Processing Magazine*, 25(5).

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Juang, B.-H., Chou, W., and Lee, C.-H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio processing*, 5(3):257–265.

Juang, B.-H. and Rabiner, L. R. (1990). The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(9):1639–1641.

Juang, B.-H. and Rabiner, L. R. (2005). Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67.

Kim, H. and Kim, I. (2015). Dynamic arm gesture recognition using spherical angle features and hidden Markov models. *Advances in Human-Computer Interaction*, 2015:7.

Koller, O., Forster, J., and Ney, H. (2015). Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding*, 141:108–125.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Liang, B. and Zheng, L. (2013). Three dimensional motion trail model for gesture recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 684–691.

Liang, B. and Zheng, L. (2014). Multi-modal gesture recognition using skeletal joints and motion trail model. In *ECCV Workshops (1)*, pages 623–638.

Liang, R. and Ouhyoung, M. (1997). A real-time continuous gesture interface for taiwanese sign language. *Submitted to UIST*.

Liang, R.-H. and Ouhyoung, M. (1998). A real-time continuous gesture recognition system for sign language. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 558–567. IEEE.

Minka, T. (2005). Discriminative models, not discriminative training. Technical report.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324.

Neverova, N., Wolf, C., Taylor, G. W., and Nebout, F. (2014). Multi-scale deep learning for gesture detection and localization. In *Workshop at the European conference on computer vision*, pages 474–490. Springer.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160.

Povey, D. (2005). *Discriminative training for large vocabulary speech recognition.* PhD thesis, University of Cambridge.

Qin, F., Auerbach, A., and Sachs, F. (2000). A direct optimization approach to hidden Markov modeling for single channel kinetics. *Biophysical Journal*, 79(4):1915–1927.

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

Saon, G., Kuo, H.-K. J., Rennie, S., and Picheny, M. (2015). The IBM 2015 english conversational telephone speech recognition system. *arXiv preprint arXiv:1505.05899*.

Schluter, R. and Macherey, W. (1998). Comparison of discriminative training criteria. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 493–496. IEEE.

Starner, T. and Pentland, A. (1997). Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer.

Starner, T., Weaver, J., and Pentland, A. (1998). Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375.

Starner, T. E. (1995). Visual recognition of american sign language using hidden markov models. Technical report, Massachusetts Inst Of Tech Cambridge Dept Of Brain And Cognitive Sciences.

Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147.

Tang, J. and Dannenberg, R. B. (2013). Extracting commands from gestures: Gesture spotting and recognition for real-time music performance. In *International Symposium on Computer Music Modeling and Retrieval*, pages 72–85. Springer.

Vogler, C. and Metaxas, D. (1998). Asl recognition based on a coupling between hmms and 3d motion analysis. In *Computer Vision, 1998. Sixth International Conference on*, pages 363–369. IEEE.

Wan, J., Escalera, S., Baro, X., Escalante, H. J., Guyon, I., Madadi, M., Allik, J., Gorbova, J., and Anbarjafari, G. (2017). Results and analysis of chalearn lap multi-modal isolated and continuous gesture recognition, and real versus fake expressed emotions challenges. In *ChaLearn LaP, Action, Gesture, and Emotion Recognition Workshop and Competitions: Large Scale Multimodal Gesture Recognition and Real versus Fake expressed emotions, ICCV*, volume 4.

Wu, D. and Shao, L. (2014). Deep dynamic neural networks for gesture segmentation and recognition. In *Workshop at the European Conference on Computer Vision*, pages 552–571. Springer.

Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE.