EXAMINING COMMON ASSUMPTIONS ABOUT THE CONVERGENCE OF THE BAUM-WELCH TRAINING ALGORITHM FOR HIDDEN MARKOV MODELS

Abstract

Hidden Markov models (HMMs) model time series and have many applications. However, reliably training an HMM has proven to be non-trivial. One of the challenges is that the Baum-Welch algorithm finds a local, instead of a global optimum. In this paper, we study the conditions in which the local optimisation algorithm finds an optimum using global information from artificially generated models and data. Using a known global optimum, we can look at the distance and performance, in terms of log-likelihood, of a model at any point during or after training. We can then test a number of common assumptions, such as whether a model that is close to the global optimum actually has a good performance. We find that most common assumptions do not hold and question whether the optimisation criterion used by Baum-Welch is the most effective. Finally, we offer a number of considerations for future work that aim to help find a better optimum than using log-likelihood as an optimisation criterion alone.

1 INTRODUCTION

Hidden Markov models (HMMs) have been used extensively to model time series in applications of speech recognition (Rabiner, 1989), handwriting recognition (Bunke et al., 1995; Plötz and
Fink, 2009) and gene sequence segmentation (Eddy, 1998). However, non-Markov, neural methods for sequence classification are gaining considerable momentum in the handwriting recognition and speech recognition fields. The deep learning method Bidirectional Long Short Term Memory networks (BLSTM) (Graves et al., 2009; Frinken et al., 2012) delivers very promising results, at the cost of studies involving HMMs. HMMs still have an active following, and it is interesting to look at some of the assumptions that are commonly associated with the training algorithms for HMMs, and why this technique may be losing popularity.

In a previous study (van Oosten and Schomaker, 2014a), we investigated the role of the transition probabilities in hidden Markov modeling and showed that it is hard to learn the correct, known properties, such as the topology of the transition matrix, of a Markov process, from artificially-generated sequences of observations. In the current study we will apply similar methods to look at convergence of HMMs. We are particularly interested in the conditions where models seem to converge to a local optimum, as opposed to the global optimum. This is an interesting topic because generally, it is assumed that the closer the trained models are to the global optimum, the better the performance (usually measured by log-likelihood or classification accuracy).

The canonical training methods for HMMs are Baum-Welch (Rabiner, 1989) and Viterbi or Segmented k-means training (Rabiner et al., 1986). Both methods are expectation maximization (EM) techniques and the final solution is highly dependent of the initialisation. That means that the starting point, i.e., the set of initial model parameters — often chosen at random or by using a clustering method for parts of the model (Bhowmik et al., 2011) — determines whether or not a model will end up in a local or a global optimum. The only guarantee that EM gives is that the likelihood of a model does not decrease during training, i.e., the likelihood of modelling the training samples increases or stays the same: $P(O|\lambda_{i+1}) \geq P(O|\lambda_i)$, where $\lambda_i$ is the model at iteration $i$ and $O$ is the training set of observation sequences.

When the solution landscape is not smooth and having a single optimum, which is almost always the case for non-trivial HMMs,
Figure 2.1: The final optimum that a model will reach, when trained with Baum-Welch, depends on the initialisation. Model a will converge to the local optimum at the left, while model b will converge to the global optimum on the right. The optimum is in this case defined as the maximum. Conversely, in gradient descent one would minimise the loss function, but the problem of the presence of several local and a single best solution is similar, there.

as opposed to, e.g., the convex SVM loss function, and if the initial model is not located on a direct path to the global optimum, a local optimum will be reached. See Figure 2.1 for a schematic representation of the difference between local and global optima: Trajectory a leads to a local optimum, whereas trajectory b leads to the global maximum.

There are a number of studies that investigate achieving a better, or even a global optimum. These studies can be divided into two categories: I) Meta search algorithms, that employ global search over a local search method; and II) Modifications to the EM algorithm itself.

The first category concerns the meta search algorithms. A straightforward method is the ‘restart’ method: train a number of models and select the best one. However, these restarts can be costly and do not guarantee convergence to the global optimum. Zhang et al. (2008) also employ restarts, but devised a method that quickly prunes unsuccessful attempts. A related method is described by Lee and Park (2006) which uses simulated annealing, together with Baum-Welch, to optimize the models. This method also restarts the EM-algorithm, but instead of starting from a new
randomly initialised model, selects the new initial parameters using the simulated annealing method. Another method that uses restarts is described by Gil and Williams (2009). This method clusters models by their parameters and selects a new model to optimise by finding a point outside a cluster.

The second type of solutions covers modifications to the training algorithm itself. A study by Farago and Lugosi (1989) explains how to use an alternative to Viterbi training (i.e., only updating the most likely path) to reach a global optimum in the restricted case of left-right models. Siddiqi et al. (2007) show a method that not only avoids local optima, but also searches for the optimal number of states by ‘splitting’ states with very similar observation probabilities. There are also spectral techniques, such as the technique described by Hsu et al. (2012), for finding the true parameters under certain conditions. These techniques do not necessarily find the traditional transition and observation probabilities, but rather a representation that is closely related to them.

Since the prevailing method of learning the parameters of hidden Markov models is still the Baum-Welch method, we are interested in the conditions in which this method converges to local optima. This interest grew from the observation in a previous study (van Oosten and Schomaker, 2014a) that it is hard to find the topology of the underlying transition matrix of a known model. We found this by comparing a trained model to the model that generated the observation sequences.

In this paper we will take the method of generating both model and data a step further and introduce a framework to investigate the inner workings of the Baum-Welch algorithm from a convergence perspective. We will do this by artificially generating sequences of discrete tokens\(^1\) and comparing the trained model with the original model that was used to generate the sequences. As mentioned earlier, the general assumption is that finding the global optimum is good for performance. We will test this in Section 3 by observing whether there is a relation between the

---

1 We use discrete tokens for simplicity. In principle, the methods can also be applied to continuous-density HMMs.
distance to the global optimum and performance, in this case measured by log-likelihood. We expect to see an increase in log-likelihood when the distance to the global optimum is smaller. We then also expect to see that distance decreases during the training of a model, because if the likelihood increases, which is what Baum-Welch does, the distance should go down.

Another assumption that will be tested is whether the global optimum is easy to find when the distance is already small. This means that there is a relation between the distances of the initial model and the trained model to the global optimum (see Section 4). Related to this question is whether the global optimum can be reliably found when parts of the model are already known. For instance, in gene segmentation (Azad and Borodovsky, 2004), the transition matrix can usually be estimated directly from labelled transitions. We expect the distances to the global optimum to be very low if we do not need to estimate certain parts of the model (Section 5).

Finally, in Section 6 we touch on some ideas that still need to be developed further. We will investigate how much global information is actually needed to guide the local process to a global optimum. We believe that the methods of inspecting the inner workings of the Baum-Welch algorithm, a local search algorithm, by using global information are very promising.

2 METHOD

To investigate the convergence towards a global optimum, we need to know what the global optimum is. As we mentioned in the introduction, we will generate artificial data by running the algorithm by Rabiner (1989) using randomly chosen models. When we train a model on the data generated from a model, we call the generating model, $\lambda_O$, the original model and consider it to be the global optimum for the Baum-Welch training algorithm. We can compare the newly trained model, $\lambda_T$, to the original model. Figure 2.2 shows this procedure schematically.
Figure 2.2: Method to train and compare models: By generating data from a model with known parameters, and training on that data, we can compare the model with known parameters with the newly trained model. Model initialisation is usually done by randomly generating model parameters, but some experiments have a different method of initialisation.

Since we now define the original, generating model to be the global optimum, we need to know whether another model is ‘close’ to it. Rabiner (1989) also describes a distance measure $D(\lambda_p, \lambda_q)$, that is “a measure of how well model $\lambda_p$ matches observations generated by model $\lambda_q$, relative to how well model $\lambda_q$ matches observations generated by itself.”

However, using a distance measure based on generated data by either model, we can not measure the distance of any part of a model $\lambda = (A, B, \pi)$ separately. More importantly, Rabiner’s distance measure uses log-likelihood: a measure that is used to optimize the models in the first place. We therefore propose to use a different distance measure that is based on the $\chi^2$ distance. The distance metric is the sum of all the $\chi^2$ distances over all rows of the three matrices of a model $\lambda$, and we can selectively choose what distances to use in our comparison: $D_A(\lambda_p, \lambda_q)$ for comparing only transition matrices, $D_B(\lambda_p, \lambda_q)$ for only comparing observation matrices, $D_\pi(\lambda_p, \lambda_q)$ for the initial state distributions,
and any sum of these three distances. We define $D_A(\lambda_p, \lambda_q)$ as follows:

$$D_A(\lambda_p, \lambda_q) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{(A^p_{ij} - A^q_{ij})^2}{A^p_{ij} + A^q_{ij}}$$

where $A^p_{ij}$ denotes the transition probability from state $i$ to state $j$ in model $\lambda_p$. $D_B(\lambda_p, \lambda_q)$ and $D_\pi(\lambda_p, \lambda_q)$ are defined similarly. We also define the sum of the distances of $A$, $B$ and $\pi$ as $D_{AB\pi}(\lambda_p, \lambda_q) = D_A(\lambda_p, \lambda_q) + D_B(\lambda_p, \lambda_q) + D_\pi(\lambda_p, \lambda_q)$.

The log-likelihood measure is what the Baum-Welch training algorithm optimises, and is therefore usually seen as the performance metric for a single model. When doing a simple classification task to decide which model is the most likely model to have generated a certain sequence, the common method is to return the model with the highest log-likelihood:

$$\text{class}(\vec{o}) = \arg\max_i [\log L(\vec{o} | \lambda_i)]$$

In a Bayesian sense, the actual log-likelihood is now less interesting, as long as the ‘correct’ model has the highest log-likelihood. It can be argued, however, that the goal of the training should be to reach the global optimum, provided that the assumption holds that a small distance to the global optimum leads to a high log-likelihood. In most experiments in this study, we use the average log-likelihood on a dataset as a measure of performance. That is: we compute the log-likelihood for all observation sequences in that dataset and use their average value as a measure of performance.

The models in this study have $N = 20$ states and a set of $M = 20$ discrete observable tokens. We generated 1500 sequences per model with a length of $|\vec{o}| = 50$ tokens. The sequences were of fixed length to reduce complexity and prevent potential problems with variable sequence length. The length of these sequences was chosen to be roughly the average sequence length of our previous experiments with sliding windows over words in our handwriting recognition system.

A dataset of 5000 random model initialisations will be used frequently in this study. Training new models starting at these
initialisations takes roughly 8 minutes per model. Distributing the training of 5000 models over 8 cores takes about 3.5 days to complete.

3 RELATION BETWEEN DISTANCE TO THE GLOBAL OPTIMUM AND PERFORMANCE IN TERMS OF MAXIMUM LIKELIHOOD

The first assumption tested in this section is that models that are close to the global optimum also perform better. We use the average log-likelihood on the training set as a measure of performance. When measuring the performance of a single model, we cannot use accuracy since there are no models to compare to.

We randomly initialised 5000 models and trained them on a single dataset. These models then all represent different starting points in the ‘fitness’ landscape. The dataset of 1500 instances was generated by using the Rabiner algorithm on model $\lambda_O$ and used to train all models. $\lambda_O$ has a transition matrix with a Bakis topology. To prevent also having to estimate the topology from scratch, the random models were limited to a Bakis model as well. Note that this restriction should make the model estimation easier than is the case for arbitrary topologies.

After training the 5000 randomly initialised models on the training set, we plot $D_{AB\pi}(\lambda_T, \lambda_O)$, the distance of the trained model to the original, against the average log-likelihood. Figure 2.3 shows the results of this experiment. For comparison, the average log-likelihood of the generating model attained the value $-143.27$.

We can see that there is no clear relationship between distance and performance. However, we can see that the few models that do end up at a very low distance from the original model $\lambda_O$, tend to have a better performance. That does not mean that only models with a low distance show a high performance (Figure 2.3, upper left data points). On the contrary: the models with an average log-likelihood of approximately $-143.2$ show a wide range of distances of 1.8 to 6.5.
Figure 2.3: Scatter plot of obtained log-likelihood values on a training set with respect to the measured $\chi^2$ distance between the true (original) model and many obtained models ($N=5000$) that were trained from random initial conditions. Log-likelihood for the generating model is $-143.27$. The scatter plot appears to indicate that there is a wide dispersion of solutions, both in terms of log-likelihood and model distance. Also, in log-likelihood space there appear to be ‘plateaus’ of solutions, with a wide variation in the distance between trained model and original.

In the remainder of the paper we will address a series of concrete Findings, that are based on our empirical work with the simulations described in this study. These simulations are inspired by our work in handwriting recognition.

Finding 1: The relation between distance to the original model and the final performance measure for HMM training, i.e., the maximum likelihood, is not clear. This means that distance is a poor predictor of (best) ML estimate and vice versa.

There is another assumption that is related to the question whether distance is a good predictor of performance: if the performance during training goes up, the distance to the global optimum should go down. It is not unreasonable to assume this if there is a clear relation between distance and performance, because it would mean that a better performance would lead to models closer to the global optimum.
This can be tested by plotting the distance and performance against the training iteration. We randomly selected four models from the 5000 models from the previous experiment, and plotted the distances and average likelihood in Figure 2.4.

We can see that the model distance to the original model $\lambda_O$, after a small dip, actually goes up again. Depending on how long we let the models train, the distance might even end up larger than at the very first training iteration. At the same time, we see that the average log-likelihood shows a common pattern of quick
improvements and then levelling off to an asymptote, reaching more or less the same log-likelihood for all four models.

**Finding 2:** The distance of the parameters of a model to the global optimum does not necessarily go down during training, in contrast to the log-likelihood, that increases to an asymptote as a property of the Baum-Welch algorithm.

It should be noted that the implementation of the Baum-Welch algorithm in this study was evaluated against other implementations, without revealing significant differences in an earlier study (van Oosten and Schomaker, 2014a).

## 4 Relation Between Initial and Trained Distance to the Global Optimum

The final optimum that an EM algorithm reaches is very dependent of the initial start condition. In this section, we test the assumption that the closer we are to the global optimum (i.e., the smaller the distance from a model to $\lambda_O$), the easier it is to find the global optimum. In other words, if the assumption is valid, we expect there to be a relation between $D_{AB}(\lambda_I, \lambda_O)$ and $D_{AB}(\lambda_T, \lambda_O)$.

In this section we are looking at the same 5000 models that were trained in the previous experiment, but now from the perspective of the assumption that the closer we are to the global optimum initially, the closer the final trained model will be. However, there is a model that is even closer to the global optimum than any of the random initialisations: the original model $\lambda_O$, which obviously has a distance of 0.

We initialise the Baum-Welch training with the original model (i.e., $\lambda_I \leftarrow \lambda_O$) and train on the data generated using $\lambda_O$ until we reach a convergence-criterion\textsuperscript{2}. It is interesting to note that it took the Baum-Welch algorithm around 1200 iterations to reach

\textsuperscript{2} In this case, a very strict criterion of 50 iterations with a change in average log-likelihood of smaller than $10^{-12}$. 
convergence. The final distance $D_{AB\pi}(\lambda_T, \lambda_O)$, averaged over ten different models, is 0.43.

Note that this means that the training makes the model drift away from the original model. This can be explained by the fact that we do not have an infinitely large dataset. The experiments are performed with 1500 sequences, but this is still only a limited view on all 310 parameters for this model (10 × 10 state transitions, 10 × 20 observation probabilities and 10 initial state probabilities). The view is limited because the observations only give partial evidence for the underlying states. Getting a perfectly trained model $\lambda_T$ would require an infinitely large number of generated sequences, and thus an infinite amount of training time.

**Finding 3:** When initialised perfectly at the global optimum, the final model will have drifted away a bit, but still ends up very close to the global optimum.

Returning to the 5000 models that we trained, we expect that when running Baum-Welch algorithm on a model with a small initial distance to the original model will lead to a final trained model with a smaller distance to the original model.

Figure 2.5 shows the results of plotting these distances against each other, using densities. Using the diagonal line, we can see that all the models that end up above the line increased in distance, while everything below it lowered the distance to the original model. We can see here that even though the strict majority of the models ended up (78%) below the line, there is still a large number of models that moved away from the global optimum. We can also see that there is not a very clear relation between the two distance measures. On average, the Baum-Welch training will have a slight beneficial effect, by improving the distance by $-1.41$.

**Finding 4:** $D_{AB\pi}(\lambda_i, \lambda_O)$, the distance of the initial model to the original model is a bad predictor for $D_{AB\pi}(\lambda_T, \lambda_O)$, the distance of the trained model.
Another way to look at the relation between initialisation and the found optimum is by plotting the distance during training. Figure 2.6 shows the trajectories of four models over time, represented by the distances of transition and observation probabilities. The initial model is indicated in the figure by a circle, while the final model is indicated by a triangle. The figure clearly shows the sub-optimal path through the $D_A$ versus $D_B$ landscape.
Trajectory ‘a’ starts promising, but suddenly deteriorates in transition probability distance. The trajectory migrates to the strange attractor indicated by ‘*’. Trajectory ‘b’ is reasonable: the distance of the transition matrix improves a lot, and the distance of the observation probabilities improve slightly as well. Trajectory ‘c’ is more chaotic and, like ‘a’, also ends up in the attractor indicated by ‘*’. Trajectory ‘d’ has, at the end of the training, not changed much when looking at the transition probabilities, but did improve considerably in observation probability distance.
It is interesting that the distance of the observation probabilities seems to be optimised first, as indicated by the initial downward trajectory. Even though the distances seem to be minimized in the first couple of training iterations, the trajectory moves away from the desired \((0, 0)\) point later in the training, just like we saw in Figure 2.4. This leads us to question whether the Baum-Welch optimization criterion, i.e., maximizing the log-likelihood, is the most optimal criterion.

5 TRAINING WITH PARTIALLY KNOWN MODELS

In certain applications some information can be easily inferred from the data. As mentioned in the introduction, the transition probabilities for models in gene segmentation can be estimated very reliably by using maximum likelihood estimation (Azad and Borodovsky, 2004). This usually means that the actual transitions are labelled in the training dataset. In other applications such as handwriting and speech recognition this information is not readily available.

In (van Oosten and Schomaker, 2014a), we examined the importance of a correctly estimated transition matrix. In this section, we will examine a related question: How closely can we get to the global optimum if certain properties are fixed to the known properties from the original model.

To answer this question we will copy the probabilities that we want to fix from the original model. This is also called ‘clamping’, from the idea of a voltage clamp in electrophysiology. We will look at two experiments: in the first experiment the transition probabilities are fixed while the observation probabilities will be trained. The second experiment will clamp the observation probabilities, and while keeping these fixed train the transition probabilities. We will then look at the final distances to the original model.

Performing these experiments, we found that all models converge to a model that has a very small distance to the original model. Table 2.1 shows the distances for the experiments described in
Table 2.1: Results of having certain properties of the model already known. The numbers are averages over 10 different models. ‘Fixed’ means that the model took either the transition or observation probabilities and these were not updated during training. ‘Trained’ means that the probabilities were randomly initialised and updated during training. The low distance to the original model when keeping the observation probabilities fixed seems to indicate that the observation probabilities are more important than the transition probabilities.

<table>
<thead>
<tr>
<th>Transition prob.</th>
<th>Observation prob.</th>
<th>$D_{AB\pi}(\lambda_O)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>trained</td>
<td>0.26</td>
</tr>
<tr>
<td>trained</td>
<td>fixed</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Finding 5: Keeping either the transition or observation probabilities fixed makes the Baum-Welch algorithm converge very close to the global optimum.

6 Implications

In the previous sections of this report, we have looked at the distance from a trained model to the original model with the purpose of ‘debugging’ the Baum-Welch process – to see when it converges to a global optimum. All the experiments seem to indicate that there are configurations of an initial model that do converge to a model close to the data-generating model. However, the distance does not seem to be a very good predictor for either
average log-likelihood or the distance of the final model to the global optimum, the original model $\lambda_0$.

It is difficult to gauge from distance alone what the optimal initial position is. It would be useful to have the information on whether a model is moving in the right direction at training time, but – alas – ‘original’ model for all practical purposes of HMMs is not available. If we could push a model in the right direction, this might give the Baum-Welch algorithm enough information to find an attractor close to the global optimum. In this section we will look at how much of a push the training algorithm actually needs and what we can do with that.

6.1 How much do we need to ‘push’ a model in the right direction?

For the purpose of pushing the Baum-Welch algorithm in the right direction, we will modify the updates to the parameters by using the known, original model. This means of course that this method cannot be used in practice, but it will give us some insight into how much the algorithm needs to be shown the correct direction.

We introduce a mix-factor $\alpha$ here. During the update to the parameters of the model, the mix-factor modifies the standard Baum-Welch update to include some knowledge of the original model. The new update rule is formulated as follows:

$$\lambda_{i+1} = \lambda_i + \Delta P$$

where

$$\Delta P = (1 - \alpha)\Delta P_{BW} + \alpha\Delta P_T$$

with $\Delta P_{BW}$ being the update that would be performed by using Baum-Welch alone, and $\Delta P_T$ is the difference with the true, original model, and thus the delta required to get to the true model in a single step. With the $\alpha$ parameter, we can therefore tweak how much of the true model will be added to the current model, and thus see how much we need to force the update in the right direction to find the global optimum. Setting $\alpha$ to 0, the update rule would be identical to the standard Baum-Welch update rule.
Figure 2.7: Results of the experiments with a mix-factor $\alpha$ between 0 and 0.1 with 0.001 increments, each data point is the average distance to the global optimum of 100 different initialisations. The 100 different initialisations are shared between the different $\alpha$ values, so the initialisation 0 for $\alpha = 0.01$ is the same as initialisation 0 for $\alpha = 0.02$. The graph shows that a mix of 99% Baum-Welch and $\sim 1\%$ additional information leads to a desired low distance.

It is important to note that for $\alpha > 0$, the Baum-Welch guarantee $P(O|\lambda_{i+1}) \geq P(O|\lambda_i)$ no longer holds. This is a necessity of our goal: to push Baum-Welch out of the path towards a local optimum. This does not mean that a push out of that path will always result in a worse log-likelihood, but the fundamental guarantee no longer applies.

Starting at $\alpha = 0$, with increments of 0.001, we trained 100 different models per value of $\alpha$ and took the average distance to the global optimum. The results are shown in Figure 2.7.

We can see that at $\alpha \approx 0.015$, the distance has reached a distance of around 0.25, a drop of almost 5 from the default Baum-Welch algorithm. So, with a relatively small push, we can already push the model in the right direction to reach an optimum very close to the original model. This is good news, because we believe that this push can be trained using a meta-learning algorithm.
6.2 Is meta-learning necessary?

We want to use the fact that just a small push in the correct direction has a big impact on the final distance to the global optimum. However, in a realistic setting, the known model is not available, so the correct direction is unknown. This section describes an idea to apply the information that can be obtained when the global knowledge is available, in a setting where only local knowledge can be used.

The basic idea is to train, an external classification system to predict the push that is needed to move in the right direction given only the information available at the time. The data that is necessary for training such a neural network can be obtained by the following procedure: I. Generate data from a random model, II. Train multiple, randomly initialised models on the data, and III. Store the Baum-Welch update, $\Delta P_{\text{BW}}$ as well as the difference with the true model, $\Delta P_{\text{T}}$.

While this approach needs more time to be fleshed out, a possible direction that is being investigated right now is to use a multilayer perceptron (MLP), such as sketched in Figure 2.8. It uses both the current model and the Baum-Welch updates as inputs and $\Delta P_{\text{T}}$ as the output. The model can be used to transform current
updates (when there is no global knowledge available) to a push in possibly the right direction. We consider this to be the next phase in the research towards finding the global optimum for HMMs. The novelty is that we can now use global knowledge to guide a local search algorithm.

7 Discussion and Conclusion

In this paper, we have examined the inner process of training HMMs. We have looked at the convergence of the Baum-Welch algorithm using tools such as generating models and sequences, comparing models using a $\chi^2$-based metric, and changes in distance over time. We employed these methods to study common assumptions about HMMs and look at the conditions where HMMs converge to a local optimum, rather than a global one. Using a model, $\lambda_O$, we can generate sequences to train new, randomly initialised models. We then call $\lambda_O$ the global optimum for the models trained on the generated sequences.

The common assumption that is tested in this study is that when the model is close to the global optimum (measured by distance), the performance (in terms of likelihood) will be good. In Section 3 we have shown that the assumption does not hold by showing that distance is a poor predictor of likelihood. In the same section, we have also shown that the distance of a model to the global optimum does not necessarily go down during training with the Baum-Welch algorithm.

In Section 4 we looked at another assumption: if the initial model is close to the global optimum, the final trained model will also be close. Combined with the findings of Section 3, we have found that the distance of the initial model to the global optimum is a bad predictor for the final distance of the trained model to the global optimum. Therefore, there is no guarantee that a small initial distance will yield a good model.

Finally, in Section 5, we have shown that when we have a partially known, and fixed, model, the Baum-Welch algorithm does not have trouble finding the global optimum. This fixing, also known
as clamping, is common in applications in bio-informatics, most notably in gene-segmentation (Azad and Borodovsky, 2004).

These findings mostly contradict most common assumptions about HMMs and the training method that is most frequently employed to train HMMs and lead us to question whether maximizing the log-likelihood is the best optimisation criterion for training an HMM. It is important to note that we have only tackled a small portion of the problem. For instance, we have not looked at classification accuracy as the measure of performance. We have chosen to look at log-likelihood first because this is the measure that is optimised by Baum-Welch. It is commonly assumed that being close to the global optimum leads to a good log-likelihood, which in turn leads to a good classification accuracy \( D_{ABR}(\lambda_i, \lambda_O) \rightarrow \log \mathcal{L} \rightarrow \text{accuracy} \). In our experiments, we have only looked at the first step in this chain, but it is still very interesting to look at the second step in future work as well: does a good log-likelihood on the training set lead to a good classification accuracy.

Another problem that is not addressed in this paper is whether handwriting can actually be modelled as a Markovian process. There are studies that use context to model parts of the handwriting outside the current field of view of the model (Bianne-Bernard et al., 2011; Dolfing and Haeb-Umbach, 1997). A different method is applied by Frinken et al. (2014): by using a graph-cut approach to decoding HMMs, non-Markovian constraints can be used to model long-term dependencies in handwriting.

Ultimately, we are interested in the context in which an HMM will be used. We identify two movements currently. The first movement is towards systems that use convolutional neural nets and/or recurrent neural nets such as (B)LSTMs (Graves et al., 2009; Frinken et al., 2012). There seems to be a growing community of researchers and engineers that move in this direction, given by the recent successes achieved by this system.

On the other hand, in systems that still use HMMs, we see a movement towards fairly complex systems, with a lot of human (engineering) effort being spent in designing and optimising
systems that employ HMMs, either in terms of system complexity (Artières et al., 2007; Bideault et al., 2015; Khemiri et al., 2015; Roy et al., 2014; Ahmad et al., 2014), or in the number of hyperparameters to tune (Britto et al., 2001; Rothacker and Fink, 2015; Puigcerver et al., 2015).

In both neural networks and HMMs research, we believe there is great need to understand the mechanism within the black box (see for other examples Elman, 1990 for multi-layer perceptrons and Schuster-Böckler et al., 2004 for HMMs). In this study, we have shown one particular method: looking at a local process using global information. We believe there is still much to be gained from this approach. For example, we could learn to recognise (un)successful trajectories quickly, or even use meta-learning to modify the inner-loop to ‘push’ the algorithm towards the global optimum. The experiments with the mix-factor are hopeful: The Baum-Welch algorithm only needs a small push in the right direction to find the global optimum.

To conclude, we have used artificially generated data to examine common assumptions about the convergence property of the Baum-Welch training algorithm. We found that the distance to the global optimum is a poor predictor of the final log-likelihood and that the distance does not necessarily go down during training. We also found that the distance of the initial, random model to the global optimum is a poor predictor for the distance of the final, trained model. Finally, we found that keeping either the transition or observation probabilities fixed makes the Baum-Welch algorithm converge very close to the global optimum.

**POSTSCRIPT**

In this chapter, we examined the machine learning aspect of the pipeline, as discussed in Chapter 1, by looking at the training process of HMMs. We challenged a number of assumptions on the convergence to the global optimum by creating a model, considering it the global optimum and generating data. This method can be used to study many different machine learning methods that have a tendency to end up in local optima.
Before moving on to other machine learning methods, we will use HMMs again to study another part of the handwriting recognition pipeline: Feature Extraction. This part of the pipeline builds a representation of the handwritten word images, in such a way that the machine learning can compute the class of a word image.

In the next chapter, we will move from the full-model comparisons from this chapter, to studying the relation between the learning algorithm and the feature representation. We will use the same method of generating data with known properties to study whether these properties can be learned and what happens if we remove temporal modelling from the model.