Speaker adaptive speech recognition with HMMs and DNNs

Elon Såndberg
elons@kth.se

Helen From
hfrom@kth.se

Abstract

Speaker independent speech recognition systems are often desirable however speaker adaptive systems aims to adapt the recognition to a specific speaker to further increase classification accuracy. In this report an HMM and a DNN are used to recognise spoken digits. The HMM was adapted to speaker specific data to increase the accuracy in classification. It was possible to see a slight improvement after one iteration of the adaptation.

1 Introduction

1.1 Speaker adaptive speech recognition

Speaker adaptive speech recognition is the idea to use a speaker independent speech recognition system and adapt the system to a specific speaker with possibly only a small amount of speaker specific data. The idea has been implemented for different types of models and has been shown to successfully decrease the error rate in speech recognition. [3]

A common way to handle different speakers in speech recognition is to normalise the speaker specific data to better fit the model. [5] Instead we want to adapt the model to the specific speaker. We believe this will better handle a larger variation in the speech, such as differences in dialects or pronunciations at a speaker level.

We will implement a general hidden Markov model and a deep neural network for recognition of spoken digits. Given the predictions from the two models we use a subset with high accuracy to adapt the speaker specific HMM. We use two different classifiers to compare their predictions and increase the total accuracy since we adapt the HMM with unlabelled speaker specific data.

1.2 Hidden Markov models

A hidden Markov model, in short HMM, is a generative sequence model which can be used for classification by mapping a sequence of observations to labels. [4] The model is probabilistic and can choose the most probable label sequence. The HMM consists of transition values, representing the probabilities of jumping between states, and emission distributions, representing the probability of observing a specific emission for each state. A Gaussian Mixture HMM uses multiple emission distributions, represented by means and co-variances of the observation distributions. This is done when the emission is a normally distributed value instead of a discrete observation. All future mentions of HMM's will refer to Gaussian Mixture HMM's.

In speech recognition it is common to use a left-to-right HMM to model the order of speech sounds and prevent going backwards in time. [1] However in this report we will implement a more general HMM that will enable jumping between phonemes in different orders to be able to model different series of words with the same HMM.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

1.3 Deep neural networks

A deep neural network (DNN) is a network of nodes, called neurons, connected by weights. The network is trained by updating the weights so that for a given input one receives a desirable output.

Similarly to HMM, a DNN can be used in speech recognition to map speech input to labels. [2]

2 Method

2.1 Data

The data used are spoken series of digits. The recordings were pre-processed by cutting it into time frames and extracting logarithmized MFCC features. Each time step in every recording was labelled using a specific HMM for each recording which was built using the known digit sequence for that recording. For the DNN a stacked version of the data was used where 7 time steps in sequence were concatenated, thus each stacked time step consisted of an array of the 6 previous time steps and the current time step. This is done since this is a densely connected feed forward DNN and previous observations does not affect the next observation, only updating the weights can affect the outcome of an observation. A recurrent network could also have been used to achieve the same effect.

2.2 Speaker adaptive speech recognition

The adaptive speech recognition system uses one speaker independent HMM and DNN to classify speech, and one adaptive HMM which is updated to fit to the current speaker.

The speaker independent HMM consists of groups of states representing every phoneme. Each phoneme is represented by three states which are connected in a left to right order, with one exception which is a state representing short pauses which is only one state. Each phoneme group is then connected to all other groups with a probability based on the phoneme distribution of all digits, and under the assumption that all digits were represented equally, i.e every phoneme which was the end of a digit had an equal probability to go to the phonemes which was the beginning of a digit. An illustration of an HMM following this structure is shown in figure 1. The example model in the illustration consists of 3 different phonemes, represented by a colour each and the arrows represent the possible transitions. Note, every phoneme can move to every other phoneme in this picture, however in this limiting case of only investigating spoken digits some phonemes will never go to some other and thus those connections will have 0 probability. Our actual model consists of 21 phonemes, meaning 61 states, hence does not cover the full spectrum of phonemes but is adapted specifically for digit recognition.

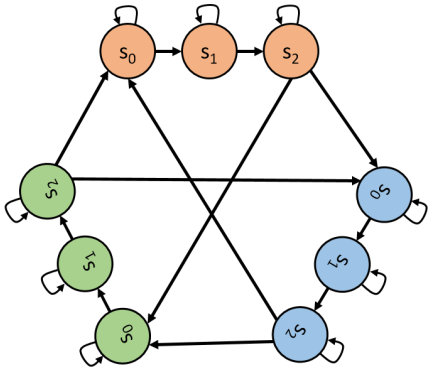


Figure 1: An HMM with 3 phonemes with 3 states each. Each colour represent one phoneme.

The DNN used was densely connected feed forward network, 4 hidden layers deep with 256 nodes per layer. It was trained for 10 epochs with a batch size of 8192 on the labelled stacked data.

The speaker independent HMM and DNN models were combined to decide when to update the HMM to adapt it to a speaker. For each time step that the HMM and DNN classified the same and the

probability of correctness for the DNN were above a cutoff value, this data was used as the speaker dependent training data.

2.2.1 Adapting the HMM

A standard EM algorithm for the HMM was used to update the means and co-variances, with the speaker dependant training data from the combined HMM and DNN model used as correct values. The means and co-variances was then updated by taking a weighted mean between the old and updated values according to equation 1 and 2.

$$\mu_{new} = \frac{n_1\mu_1 + n_2\mu_2}{n_1 n_2} \tag{1}$$

$$\sigma^2_{new} = \frac{n_1(\sigma_1^2 + (\mu_1 - \mu_{new})^2) + n_2(\sigma_2^2 + (\mu_2 - \mu_{new})^2)}{n_1 + n_2} \tag{2}$$

Here μ and σ^2 represents the mean and co-variance respectively of the speaker independent(1), speaker dependent(2) and new HMM's. n_2 is the frequency of states in the speaker dependent training data. n_1 is determined as a weight times 1 + the mean of n_1. This weight then determines how much of the old data to keep.

We only update the mean and co-variance for frames with a high probability of being correctly classified by the DNN and HMM according to the cutoff value as described above. Since our goal was to be able to work on unlabelled data we thought this was a good approach.

3 Experiments

We first trained and tested the accuracy as the percentage of correctly classified frames of our speaker independent HMM and DNN. The data used for training were 8600 recordings of random sequences of digits for different speakers. For testing, 77 recordings on the same form as the training data, for one specific speaker were used.

For the experiments some different metrics were used, these were:

- Accuracy. Percentage of correctly classified frames.
- Log Likelihood. The log probability of the most probable path through the HMM.
- Subset Accuracy. Percentage of correctly classified frames of the initially correctly classified frames after adapting the model.
- Cutoff Accuracy. Percentage of correctly classified frames above the cutoff value.

First we trained a speaker dependent model on speaker specific data and replaced the mean and co-variance values with the updated values from the data which were correctly classified. Then we instead combined the speaker independent HMM with the speaker dependent HMM using a weighed mean as described in section 2.2.1. We varied the weight to see if there was any weight that gave an increase in accuracy.

With a fixed weight we then wanted to test the performance of our model without knowledge about which data were correctly classified. We added a cutoff value as described in section 2 to update our adaptive HMM using only parts of the data.

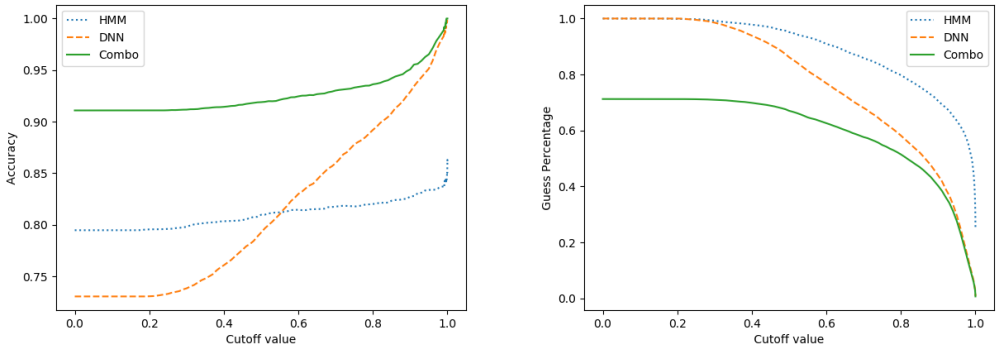
We also looked at the accuracy when varying both the weight and the cutoff value.

The data used for the difference experiments were: 10 digit sequences from the same speaker corresponding to 1708 time frames, 20 sequences of digits from the same speaker corresponding to 3047 time frames and 40 sequences of digits from the same speaker corresponding to 5767 time frames. Each time frame was 20 ms.

4 Results

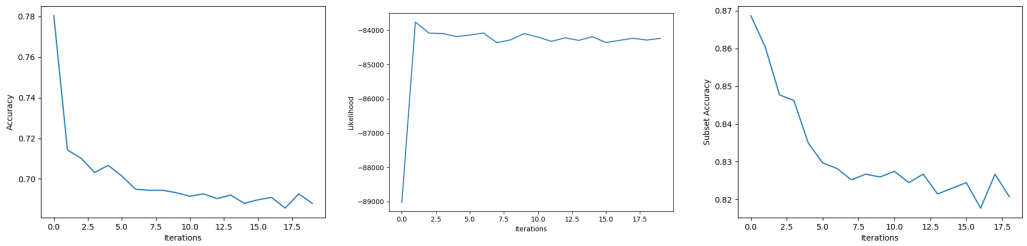
Figure 2 (a) shows the accuracy of classification for a certain cutoff value of the speaker independent HMM, the DNN and the combination of them both. The models were only doing classification when

they had a probability of correctness above the cutoff value. Figure 2 (b) shows the percentages of data that was classified.



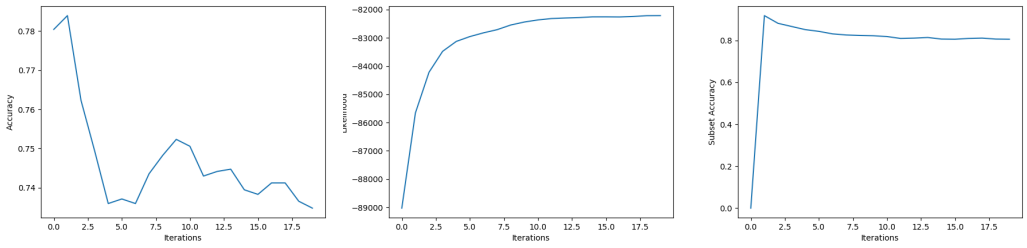
(a) Accuracy vs cutoff value (b) Percentage of classified frames vs cutoff value

Figure 2: Classification accuracy and percentage of data used while only classifying when the model has a probability of correctness above the cutoff value.



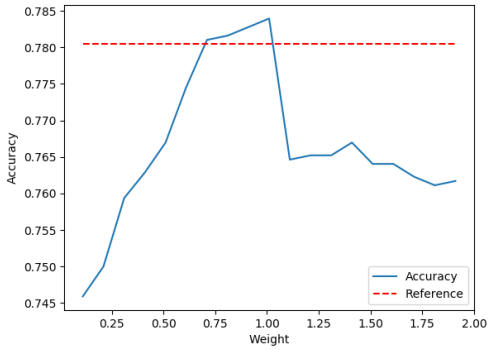
(a) Accuracy vs Iterations (b) Log likelihood vs Iterations (c) Subset Accuracy vs Iterations

Figure 3: Accuracy, log likelihood and subset accuracy when updating based on the subset of frames where the classifier was correct. The mean and co-variance values where replaced with the updated values. 10 digit sequences from the same speaker was used as the data.

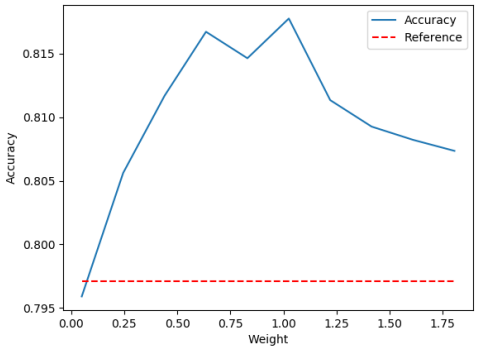


(a) Accuracy vs Iterations (b) Log likelihood vs Iterations (c) Subset Accuracy vs Iterations

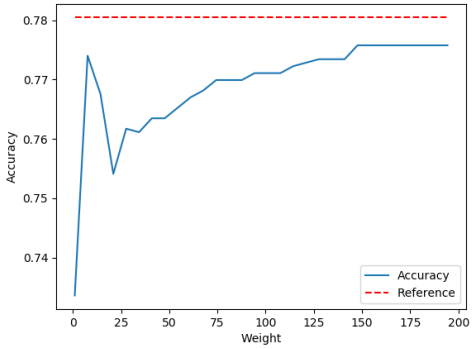
Figure 4: Accuracy, log likelihood and subset accuracy when updating based on the subset of frames where the classifier was correct. The mean and co-variance values where combined with the old values. 10 sequences of digits from the same speaker was used as the data.



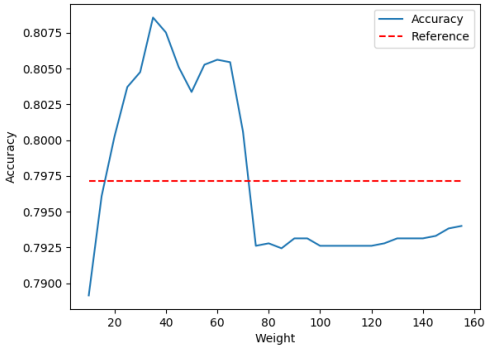
(a) Accuracy vs Weight
10 digit sequences using correct classifications



(b) Accuracy vs Weight
40 digit sequences using correct classifications

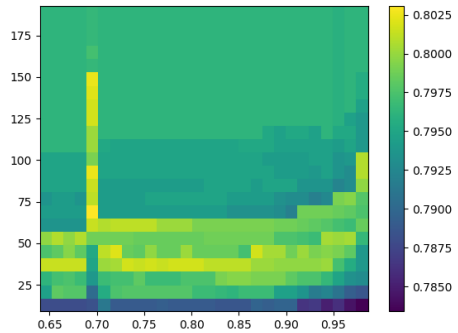


(c) Accuracy vs Weight
10 digit sequences using Cutoff value classification

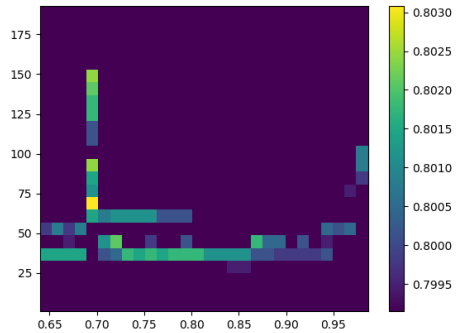


(d) Accuracy vs Weight
40 digit sequences using Cutoff value classification

Figure 5: Accuracy after one iteration vs the value of the weight used when combining the old and updated models. The reference line is the accuracy of the speaker independent classifier. 10 sequences of digits from the same speaker was used as the data for figure (a),(c) and 40 for figure (b),(d). (a) and (b) used only correct classification while (c) and (d) used classifications with a cutoff value of 0.85.

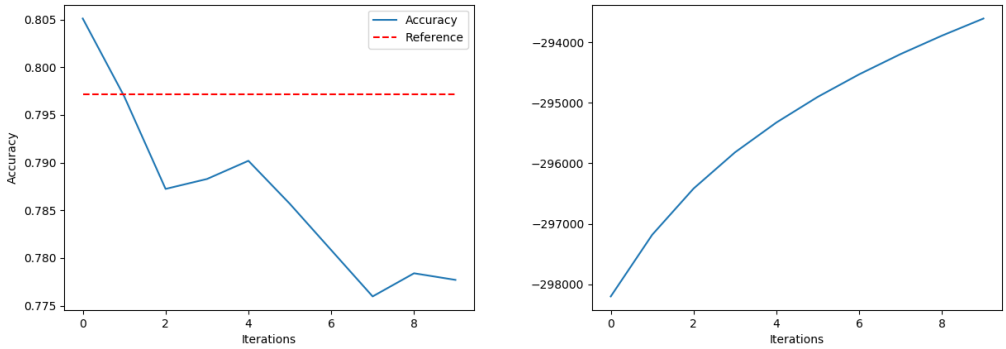


(a) Accuracy vs weight and cutoff value.



(b) Accuracy above reference vs weight and cutoff value.

Figure 6: Accuracy after one iteration as a function of weight (y-axis) and cutoff value (x-axis). 20 sequences of digits from the same speaker was used as the data.



(a) Accuracy vs Iterations

(b) Log Likelihood vs Iterations

Figure 7: Accuracy and log likelihood when updating based on the subset of frames with a cutoff value of 0.85. The mean and co-variance values where combined with with a weight of 34. 40 sequences of digits from the same speaker was used as the data.

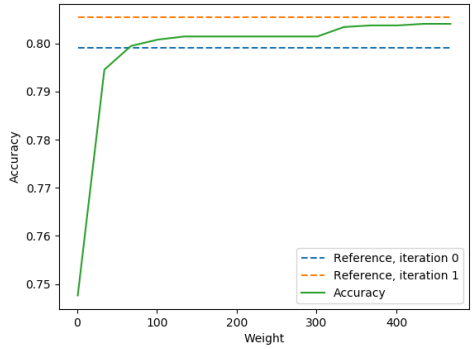


Figure 8: Accuracy after two iterations vs weight. The first iteration used it's optimal weight (34). 40 sequences of digits from the same speaker was used as the data.

5 Discussion and conclusions

From figure 2 we can see that the HMM is a better classifier at low cutoff values, and thus a better classifier over all but it remains roughly constant when the cutoff value increase. The DNN however increases drastically in accuracy with the cutoff value. The combined model uses the best of both worlds by getting both the better accuracy from the HMM at low cutoff values while also gaining the boost from the DNN at higher cutoff values and is thus superior to the DNN and HMM alone. A cutoff value of around 0.85 seemed to be a well rounded value since the classification rate is still high, at around 50% and the classification accuracy is around 95%. Hence this value was chosen when testing the model.

To see if the adaptive HMM could get better at all, tests were done with where the updates where preformed on frames which were known to be correct. In figure 3 we see no improvements when the updated mean and co-variance values were simply replaced with the updated ones, which is to be expected since information from the speaker independent HMM is lost during the update. Phonemes for which the HMM have a lower accuracy will not be represented as much as phonemes which have a good accuracy, thus fewer data points will be collected for these values and skewing the mean and co-variance values.

This problem is addressed in the next test. In figure 4 the same test was preformed but the updated mean and co-variances where combined with the old values thus retaining information for the

phonemes which lacked data points. It is to be noted that the subset accuracy appears to increase sharply in the beginning, this is however not the case but an artefact from how the subset accuracy was calculated since there is no subset data for the zeroth iteration. A small improvement can be seen in the first iteration in this test. Although very small it does show that it is possible to improve the classifier using this method. A more detailed look into how the weight, when combining the old and new values, affects the accuracy can be seen in figure 5, which shows two important things. First that the accuracy is strongly dependant upon the weight and it is not just a fluke that the accuracy improved. Secondly that more data gives a much bigger improvement in accuracy, which is to be expected.

The next question one might have is if this also works when we do not know the correct answers to each time frame. As seen in figure 5 this is investigated over many different weights and it is clear that an increase in accuracy can be achieved, but with a limitation. In (c) we never cross the reference line, but given some more data we can see in (d) that the accuracy improves a lot. It improves in the same manner as previously with a strong correlation to the weight and with a peak. This peak behaviour is well explained by the theory behind the weight, as we have a small weight the updated model is influenced a lot by the new data, thus retaining very little information and becoming worse. As the weight increases we retain more information and can actually learn from the well classified points while still retaining enough information on the worse classified points, thus increasing the overall accuracy. As the weight continues to increase we put too much emphasis on retaining information and the new information does not help much but rather just "confuses" the model. As the weight continues to increase it will become more and more like the model before it was updated and the accuracy will converge with the reference line. Another theoretical effect which can be hinted at when comparing (b) and (d) is that more data moves the optimal weight further back. Theoretically this should be the case since more data means you need to retain less information. Even though both (b) and (d) have 40 data sequences, (b) uses more data for training since it uses all correct classifications, which is around 78% of the data, while (d) only uses the cutoff value estimate at 0.85 which from figure 2 can be seen to be around 50%. The fact that this estimate is not completely correct also plays a big role. The exact correlation between the optimal weight value and the data amount is not investigated further in this paper.

The cutoff value also plays a role in how well the accuracy improves, as can be seen in figure 6. In (a) it is clear that the same features as before can be seen when moving along the Y-axis, low weights have a low accuracy, then it increases as the weight increases, reaching a maximum and then sharply decreases. This behaviour can be seen across all cutoff values and the cutoff value only seams to shift where this peak occurs. One value which stands out is where the cutoff value is 0.7, the weight peak is moved up significantly and high accuracy values are achieved for a much wider range of weights. This might just be that the relation is not as smooth as it appears from the rest of the picture and that there are certain cutoff values which are particularly good. Since we have discrete steps we can have missed results that are better, for example we have previously encountered accuracies of 80.5% for the same data at cutoff values around 0.85 which does not show in this figure. In order to investigate this further a higher resolution image of this area would be needed, unfortunately generating this current figure is time intensive, thus a higher resolution would require a lot more time or more computing power.

Sadly, as can be seen in figure 7 (a), this improvement from the first iteration does not follow to the next iterations and the accuracy decreases again. As seen in figure 8 it was also investigated if changing the weight after the first iteration had any affect on accuracy, again without any good results.

One potential reason for this is that we only update the model on data we are already fairly confident at. This could cause our model to become better on the specific data at the expense of other. Since the likelihood for the data it was confident at goes up, as seen in 7 (b), the Viterbi algorithm puts more weight into those points and potentially make the overall path better. However, since we do not train on frames where it is not confident the likelihood for these frames might go significantly down more than the confident frames goes up, thus making the correct path less likely. This might be what happens after the first iteration.

The increase of the log likelihood in figure 3 (b),4 (b) and 7 (b) also indicates that the implementation of this method has been done correctly since the likelihood increases, indicating a successful training.

5.1 Future Work

The most important future work relating to this subject is to look at more speakers. This paper only investigates a single speaker, how the optimal weight, improved accuracy etc differs between speakers is not touched upon and is extremely important if this system were to be implemented in practice.

Investigating the exact relation between the optimal weight and amount of data would also be a good idea for future work, and to investigate ways to improve the accuracy after one iteration, such as looking at how using other weights than the optimal weight for the first iteration affects the second iterations accuracy.

The DNN is known to be overconfident sometimes and it could therefore be a good idea to use some nonlinear function to calibrate the DNN to get more comparable results to the HMM. If the probabilities of the DNN and the HMM were more comparable, the cutoff value could be used differently and include both the DNN and the HMM.

It could also be interesting to add a frequency limit that would make sure that we never update the HMM on states where we do not have enough data, however this is basically already implemented using the weighted mean and will probably not lead to any significant improvements.

Updating the transition values might also be useful to model speech patterns for different speakers. However this is not applicable in this paper since the speech patterns are predetermined and only contain digits spoken with a roughly uniform distribution.

Further down the line a good idea would be to compare the system using different dialects and other variation to other speaker dependant systems.

Acknowledgements

We would like to thank Gustav Eje Henter for taking his time to help us along the way.

References

[1] Mrs. Varsha Degaonkar, Dr. Anju V. Kulkarni, and Dr. Radhika Menon. Speech recognition using hidden markov model. 2013.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 2012.

[3] Xuedong Huang and Kai-Fu Lee. On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition. *IEEE Transacitons on speech and audio processing*, 1993.

[4] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, chapter 9. Stanford University, 2016.

[5] Koichi Shinoda. Speaker adaptation techniques for automatic speech recognition. *APSIPA*, 2011.

Appendix

Since we didn't hand in a report draft in time we used the comments we got from Gustav Eje Henter as a "peer review". The report was sent to Gustav in a fairly unfinished state and hence some of the comments might just be that we have reformulated things, as we had planned, to make things more clear.

* "A hidden Markov model, in short HMM, is a sequence classifier" Nej. En HMM är en generativ sekvensmodell som också kan användas för olika former av klassificering. - We've reformulated the sentence.

* "The HMM consists of (...) emission matrix, representing the probability of observing a specific emission for each state" Det är bara vid diskreta outputsymboler som det är lätt att placera emissionssannolikheter i en matris. I andra fall föredrar jag perspektivet med en outputfördelning per tillstånd. Att vi placerar fördelningsparametrar såsom medelvärden och varianser i matriser är mest en implementationsdetalj. - We've reformulated the sentence.

* "A Gaussian Mixture HMM uses two emission matrices" För att vara en "mixture", och inte bara en "Gaussian", krävs flera Gaussfördelade komponenter och vikter som anger sannolikheten för varje komponent (samt medelvärden och varianser/kovarianser, förstås). - We've reformulated the sentence.

* "The model consists of 3 different phonemes" Jag hoppas detta bara gäller illustrationen, inte er faktiska HMM. :) - We've reformulated the sentence to make it more clear.

* Vad är motivationen för att anpassa HMM:ens outputfördelningar, och inget annat, i er HMM + DNN klassificerare? - We wanted to focus on only adapting the HMM, hopefully that's more clear how and why in the report now.

* "For each time step that the HMM and DNN classified the same and the probability of correctness were above a cutoff value for both of the models, the data of the time step was saved to be used to update the emission matrix for the speaker adaptive HMM." Ni uppdaterar alltså modellen (HMM:ens outputfördelningar) på en delmängd av datamaterialet från den nya talaren. Det betyder att HMM:en kommer att bli bättre på att efterlikna dessa data, och potentiellt bli sämre på att modellera andra data (se sista uppgiften på labb 2). I detta fall gör ni modellen mer lik de data den redan är bra på, potentiellt på bekostnad av de frames ("ramar"?) som den just nu inte kan modellera. Vad är motivationen för detta? - We've added a part in the discussion to clarify/comment on this.

* "The models were only guessing when they had a probability of correctness above the cutoff value." Betyder detta att ni inte utvärderade på alla ramar/datapunkter? Eller att modellerna "bara gissade" (slumpmässig gissning istället för prediktion) på ramar där prediktionen fungerade bra. Det verkar som att ni avser det förstnämnda. - We've tried to reformulate this to make it more clear.

* "The emission matrix was updated by taking a weighted mean between the old emission matrix and an emission matrix trained on the selected data, where the old emission matrix had a higher weight to retain most of it's information and only update observations with many data points and not update observations with few." Ni uppdaterar inte "observations" utan fördelningar för observationer, t.ex. "state-conditional emission distributions". När ni skriver "only observations with many data points", avser detta tillstånd som förkom många gånger i det ursprungliga , eller tillstånd som förkom många gånger i anpassningsdata, eller något annat? Kan ni skriva ned de exakta matematiska uttryck ni använder för att uppdatera de olika parametrarna? Det finns flera sätt att vikta ihop saker, och vissa är bättre än andra. Tar ni medelvärden av standaravvikelser eller varianser, t.ex.? - Added equations.

* "A cut off frequency was also tried where a certain number of observations for a state had to be selected in order to update the mean and variance in the emission matrix for that state." Terminologin "cutoff value" och "cut off frequency" är förvillande. Dessa tycks vara två olika saker. - We comment a potential cutoff frequency in future works but we decided not to include it in the report since the scope seemed large enough without it.

* Är accuracy per ram, tillstånd, state, eller vad? -Per frame, which is now commented when we start talking about accuracy.

* Rita grafer över likelihood (er loss function) också, inte bara accuracy! Sådana grafer visar ifall er träning fungerade som den skulle och ökade likelihood. Om likelihood går ned är det troligt att er träningsparadigm eller implementation är olämplig/fel; om likelihood går upp är det troligt att ni förbättrat "fel" del av modellen. - Added a lot of different plots.