Alternative Implementations for Pipelined Cepstral Mean Normalization

Alex S. Park

May 16, 2001

1 Background

1.1 Cepstral Mean Normalization

Cepstral mean normalization (CMN) is a signal processing techique that is used to remove the consistent channel effects of a speech recording. These channel effects can be thought of as elements of the signal that are non-essential to the actual speech content, such as telephone channel noise or vocal tract characteristics. CMN assumes that the channel of the recording can be modeled as a linear, time-invariant filter, and works by transforming the incoming signal to the cepstral domain. Because convolution in the time domain corresponds to addition in the cepstral domain, computing and subtracting the mean of the cepstrum of the signal should theoretically leave the remaining signal as the cepstrum of the original signal.

1.2 Current Implementation

As the incoming speech signal is received, the cepstrum of the available time signal is computed and passed on to the normalization sub-component of the system. After the signal has been normalized, it is then available as the input for the forward search component.

Currently, the system works by using a block accumulation method of normalization. Prior to recognition or modelling, some block size is specified. Within the normalization component, the cepstral mean for each block is computed by accumulating frame samples until the end of that block, and then dividing by the number of frames at the end of the block. As such, the frames for the first block are normalized by the mean for cepstral mean for that first block, while the frames from the last block of the utterance are normalized by the mean for the entire utterance.

An advantage of this method is that it eliminates a great deal of potential real-time recognition delay by pipelining the normalization component. Rather than having to wait until the entire input utterance is received and normalized before beginning the forward search, the blocks can be received, normalized, and sent to the search component even before the utterance is completely finished recording.

One of the disadvantages of this approach is that it does not represent a true cepstral mean calculation. Because the mean is updated as frames come in, it is only an approximation of the cepstral mean of the entire utterance. Another disadvantage is that the manner of pipelining introduces the possibility for a significant, and potentially avoidable, delay in between the normalization and forward search components. This is explained more fully in the following section.

2 Motivation

In real-time speech recognition, it is desirable to have low latency between the time when the speaker finishes speaking and when the recognition task is complete. In the SUMMIT speech recognizer, the latency between the end of the utterance and the beginning of the response is typically on the order of 1-2 seconds. Part of this delay is due to the end-point detector of the speech recorder, which doesn't mark the end of the utterance until some point after the speaker has finished speaking. In addition, the processing time required for recognition and latencies between system components also contribute to the overall time delay.

The current CMN implementation is pipelined so that the next step in the recognition task can make use of the cepstral data as it is normalized. However, since each frame depends on the mean calculated up to the end of the current block, the output from normalization is delayed by an amount of time equal to the block size. Depending upon the speed of the forward search, which uses the normalized cepstral data, this delay can translate to noticeable real-time latency.

Let the CMN block size be t seconds. Due to the calculation of the first mean, the forward search can not begin until after the first t seconds of input have been received and normalized. If the forward search proceeds slower than, or equal to, the rate of input, this delay will ultimately translate to a t second increase in the overall recognition task when compared to a system without normalization.

If the forward search proceeds faster than the rate of input, then at the end of each output block, the search will be able to catch up and wait for the next output block. In this case, most of the latency comes at the end of the utterance, and is due to the extra time that the forward search spends waiting for the last block to be normalized before it can be processed.

In both of the cases listed above, it is clear that an unnormalized system will have lower overall latency than the current implementation. The goal of this project is to examine alternative implementations for the normalization component which have the potential to avoid the same pipelined latencies of the current system.

3 Methodology

For this project we examined five alternative implementations of the CMN component, which are described below. The first four methods retain the problem of latency for the first block, but eliminate the need to wait until the end of the utterance to **begin** searching through the last block. The last method normalizes

and outputs frames at the same rate they are received, and therefore avoids both initial and final delays in proceeding with the forward search.

- First Block Mean - The mean for the first block is computed and is then used to normalize all subsequent blocks.
- Prior Block Mean - The first block is normalized with its own mean, and each subsequent block is normalized with the mean for the preceding block.
- Block Accumulation With Lag - The first block is normalized with its own mean, and each subsequent block is normalized with the accumulated mean for all preceding blocks.
- Block Mean with Frame Catchup - The first block is normalized with its own mean. Thereafter, the mean is updated with each subsequent frame as it is received.
- Frame by Frame - The mean is recalculated on every frame starting with the first input frame (no block division).

During testing, we also used three additional implementations for comparative purposes. These were:

- No Normalization - The output of the normalization componenent is the same as the input.
- Full Utterance Normalization - The utterance is normalized by the cepstral mean computed over the entire utterance.
- Block Accumulation - The current method. Each block is normalized by the mean for the entire utterance up to and including that block.

All of the above methods were implemented by modifying the Spectrum Mean Normalization code in the sls tree and rebuilding. The location of the modified files is included at the end of this document.

In order to compare the performances of each method, two sets of experiments were conducted. For the first experiment, eight independent model sets were built and trained on the same data, each using a different normalization methods. Each recognizer was then tested against an independent data set using its own models. For the second experiment, the models trained using full utterance normalization were used to test the recognition accuracy of each of the eight recognizers. For both experiments, the independent test set consisted of 176 utterances collected in the LCSinfo domain, which is a corpus of LCS directory inquiries.

4 Results

Initially, the experiments were not run deterministically, and therefore, each time the models were trained and tested, different word error rates resulted for the same tests. These variations in results were due to the random seeding of mixture models that occurs during the training of the acoustic models. Although the results shown below were taken from deterministically trained models, using the average results of many non-deterministically trained models would have given better indications of comparative performance. Due

to time constraints, however, this was not feasible.

The results of running the tests on the self-trained models are shown in Table 1. For this first set of experiments, the recognizer using the full utterance normalizer had the best recognition performance with a 17.0% WER. However, the low variance of the results indicates that the error rates do not give a strong indication of how the different normalizers rank against each other. For example, the worst performing normalizer (Prior Block) had a word error rate of 20.9%, which is only a 23% relative degradation from the best performing normalizer (Full Utterance). This figure may seem large, but previous non-deterministic training trials on the Full Utterance models alone yielded word error rates ranging from 15.8% to 17.5%, which corresponds to a 10.7% relative degradation from best to worst.

In short, although this experiment was useful in showing how each method performed in general, the results were too close to make any conclusion about the relative merits of the different normalization methods. Without more extensive testing, we can only conclude that the different normalization schemes have similar performance, and that normalization does not necessarily yield significant performance gains over an unnormalized system.

Normalization Method	Self-Trained Models	
Method	WER (Errors)	SER (Errors)
Full Utterance	17.0% (185)	38.6% (68)
No Normalization	17.9% (195)	39.2% (69)
Block Accumulation (Current)	18.8% (205)	41.5% (73)
First Block Mean	18.6% (203)	42.0% (74)
Prior Block Mean	20.9% (228)	46.0% (81)
Block Accumulation With Lag	18.5% (202)	42.0% (74)
Block Mean with Frame Catchup	18.9% (206)	40.9% (72)
Frame by Frame	17.3% (189)	38.1% (67)

Table 1: Word and Sentence Error Rates for Experiment 1

The purpose of running tests against the Full Utterance normalized models for the second experiment was to see how well the different mean approximations compared to the "true" mean over the whole utterance. As expected, the unnormalized recognizer performed very poorly using the normalized models because the absence of an approximated mean resulted in poor matches between the testing data and the trained models.

Again, due to the lack of aggregate data, it is difficult to make conclusions about the results of close competitors. However, the four cumulative methods (indicated on the table with a **C**) all yielded results that were better than the two non-cumulative methods (indicated on the table with a **N**). This is also expected, as the cumulative methods evolve the mean to eventually approximate the full utterance mean, while the other two methods normalize only by approximate the mean over single blocks.

Normalization Method Method	Full Utterance Normalized Models WER (Errors)	SER (Errors)
Full Utterance	17.0% (185)	38.6% (68)
No Normalization	124.4% (1357)	100.0% (176)
Block Accumulation (**C**)	17.7% (193)	40.3% (85)
First Block Mean (**N**)	25.8% (281)	48.9% (86)
Prior Block Mean (**N**)	23.5% (256)	48.3% (85)
Block Accumulation With Lag (**C**)	21.0% (229)	45.5% (80)
Block Mean with Frame Catchup (**C**)	19.1% (208)	40.9% (72)
Frame by Frame (**C**)	19.4% (212)	40.9% (72)

Table 2: Word and Sentence Error Rates for Experiment 2

5 Conclusion and Future Work

Any of the new implementations has the potential to reduce latency, depending on the speed of the forward search. Since the search speed can be tuned by modifying the pruning threshold of the search, a common practice is to set this threshold to a value where the search runs at a speed close to realtime. With this in mind, the frame by frame accumulation method is favourable, as it eliminates the delay introduced by the first block mean computation used by the other approaches.

Two further experiments should be conducted before replacing the current normalization method. First, an aggregate version of Experiment 1 should be performed in order to get accurate recognition results for the different implementations. This will indicate if there will be a performance penalty associated with switching methods, and also whether CMN improves recognition accuracy at all. Second, real-time recognition experiments should be performed in order to determine if theoretical latency reductions actually carry over to lower overall response times.

6 Appendix

- Source code location - /server/s/asr-course/users/6345g18/codechange/
- Self normalized test results - /server/s/asr-course/users/6345g18/results-self/
- Full normalized test results - /server/s/asr-course/users/6345g18/results-full/