Improvement of an Automatic Speech Recognition Toolkit

Christopher Edmonds, Shi Hu, David Mandle

December 14, 2012

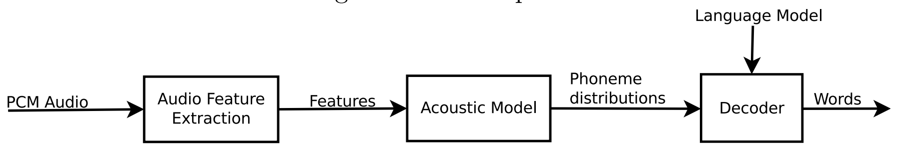
Abstract

The Kaldi toolkit provides a library of modules designed to expedite the creation of automatic speech recognition systems for research purposes. For purposes of acoustic modelling, the toolkit provides a framework for training Neural networks by stochastic gradient descent. Recent research, however, indicates that other standard nonlinear optimization techniques offer speed improvements and simplification of the training process. In this paper we build and test modules compatible with the toolkit to implement alternative training methods.

1 The Automatic Speech Recognition Pipeline

The entire automatic speech recognition pipeline serves to convert a segment of spoken word audio (known as an utterance) to a corresponding accurate transcript. For our purposes we can break the automatic speech-to-text transcription process into several key functional blocks: Feature extraction, acoustic modelling, and decoding.

Figure 1: ASR Pipeline



1.1 Feature extraction

During feature extraction, windows of raw PCM audio samples are transformed into features that better represent the speech content of the signal within that window. This is most often accomplished by use of Mel-frequency cepstral coefficients (MFCCs). These coefficients are derived from the spectrum of the frequency domain, representing the change in frequency content of the speech signal. The frequency domain is mapped nonlinearly so as to approximate human sensitivity to differences in pitch.

Consequently, each window of n audio samples comprises a "frame" that can be described by a feature vector of its n coefficients. Given an utterance m frames in duration, calculation of MFCC features will result in a matrix of features of dimension $m \times n$.

Other methods, such as PCA whitening, may then be used to transform and reduce this feature space.

1.2 Acoustic modelling

An acoustic model serves to map the extracted features to a sequence of likely spoken sounds, known as phonemes. In particular, this is accomplished by using a sliding window over the frames of feature data, and then employing a statistical model to predict the probability that the central frame of the window is a result of any given phoneme being spoken. As such, the acoustic model ultimately associates each frame of audio with a distribution over the phonemes.

1.3 Decoding

Given a distribution over all possible phonemes for each individual frame, the decoding process constructs a probable sequence of phonemes. This process must account for the probability of that spoken sequence generating the given audio signal, but also the prior probability of that sequence of phonemes being sensical.

This first probability is of course determined from the output of the acoustic model. This second probability, however, is determined by use of a language model which represents language usage via a hidden Markov model. In this language model, the states are phonemes (for a monophone model) or sequences of phonemes. The transition probabilities then represent the probability that one phoneme follows another in standard speech.

As the decoder passes over the data, the possible sequences of states are stored within a graph known as a lattice, which can be pruned to reject the least likely sequences. Once the most likely sequence is determined, the phonemes can be mapped to complete words, creating a transcription of the original audio.

2 The Kaldi Toolkit

Kaldi is a research-oriented open-source toolkit providing a number of utilities which serve to simplify the creation of an automatic speech recognition system [4]. The individual utilities are written in C++ as stand-alone executables. The toolkit specifies a number of file formats for use in exchanging data between these executables. This separation of utilities and the specification of file formats allows these utilities be easily pieced together within shell scripts to constitute end-to-end speech recognition systems.

For the purposes of feature extraction, Kaldi includes utilities to compute MFCCs, among other common speech audio features. Utilities are also included for computing feature-space transformations.

For use in acoustic modelling, Kaldi includes utilities for training and applying Gaussian mixture models and neural networks. These models may be used individually or employed serially together in a tandem fashion [1].

Additionally, Kaldi offers several utilities for decoding the output of the acoustic model, with utilities optimized for decoding with large language models, utilities which discard unlikely outputs, and utilities which make use of lattices to maintain the most likely outputs.

3 Some Experimentation

To familiarize ourselves with the features of the toolkit, we ran a number of speech transcription tests with a variety of configurations. Training and testing data was acquired from the Wall Street Journal Continuous Speech Recognition (WSJ-CSR) Corpus [3], which consists of a variety of sets of English natural language speech utterances with corresponding ground-truth transcriptions. Our experiments used data from the SI-84 (7240 utterances) and Dev'93 (504 utterances) subsets of the WSJ corpus. We experimented with different sized training sets, simple MFCCs, delta coefficients (the time derivative of MFCCs), and Kaldi's selection of feature-space transforms.

The results of these experiments are documented in Table 1.

4 A Proposed Modification

Among its utilities, Kaldi provides tools for implementing deep neural networks as part of the acoustic model. These tools include a framework for training neural networks by the method of stochastic gradient descent (SGD). However, recent research by Le et. al [2] suggests that the deep network training process can be simplified and accelerated by employing other standard nonlinear optimization techniques, namely limited memory BGFS (L-BFGS) or the conjugate gradient method with line search.

While stochastic gradient descent is valuable for optimizing over a large training set, the SGD method is sensitive to the choice of learning rate and the convergence criteria. Consequently, stochastic gradient descent must often be wrapped within a cross validation framework to obtain an optimal result on unfamiliar data.

Table 1: Kaldi transcription results

System Configuration	Word Error Rate (WER)
Features: MFCC coefficients ***Acoustic Model:*** Monophone, GMM ***Training set:*** 2000 shortest utterances of SI-84 ***Test set:*** Dev'93	35.24%
Features: MFCC delta coefficients ***Acoustic Model:*** Triphone, GMM ***Training set:*** 2000 shortest utterances of SI-84 ***Test set:*** Dev'93	20.15%
Features: MFCC delta and delta-delta coefficients ***Acoustic Model:*** Triphone, GMM ***Training set:*** 2000 shortest utterances of SI-84 ***Test set:*** Dev'93	18.03%
Features: MFCC with Linear Discriminant analysis (LDA) and Maximum Likelihood Linear Transform (MLLT) feature-space transformations for feature-space reduction ***Acoustic Model:*** Triphone, GMM ***Training set:*** SI-84 ***Test set:*** Dev'93	16.81%

By comparison, L-BFGS is a stable algorithm whose convergence can be simply checked by the norm of the gradient. However, unlike stochastic gradient descent, L-BFGS necessitates the calculation of the gradient over the entire training set. As a result, L-BFGS does not directly scale well to training sets with large numbers of examples. However, by training over minibatches of the data, this weakness is mitigated.

Given the potential improvement to the training process that alternative optimization methods can deliver, as documented by [2], we develop additional utilities compatible with the Kaldi pipeline that integrate off-the-shelf implementations of these alternative methods.

5 Our Approach

We worked off of an existing codebase provided by Andrew Maas that implements deep neural network training in MATLAB using minibatching and integration with the minFunc optimization method library [5]. We aimed to incorporate this code with the Kaldi pipeline to construct a tandem deep neural network and GMM acoustic model.

Carrying out this integration required the creation of new recipes for both the training and decoding steps in Kaldi to support a tandem architecture acoustic model that incorporates the given L-BFGS neural network training codebase.

5.1 Training

At a high level, the training procedure is similar to other tandem ASR systems. First, posterior probability distributions are generated over the phonemes via Kaldi's monophone GMM training procedure on the SI-84 dataset. These distributions are then used as labels to train the network on the windows of raw MFCC feature vectors from the SI-84 training set. We used a 17 sample wide window of features as the input features to the neural network. Training in this way is very similar to hybrid ASR systems.

In order to integrate the MATLAB neural network implementation into the Kaldi toolkit we had to perform several steps. First the features and labels needed to be transformed from Kaldi's table format into a MATLAB matrices. Similarly we had to create file format transformations to take MATLAB output and create Kaldi tables. We wrapped this work into a set of shell script commands that allow us to run MATLAB as a part of the Kaldi workflow.

The output of the neural network is a probability distribution label indicating the probability that the central frame of the window corresponds to a given phoneme. Once the network has been trained, features from all of the datasets were fed through the network to create a set of transformed features. These transformed features are then used to train a GMM using labels from a previously trained triphone GMM system.

6 Results

We compared the performance of the neural network in Kaldi's Tandem recipe to our own neural network. Apart from differences in training algorithms there are a couple of notable differences between the Kaldi neural network and that which we configured. The Kaldi neural network contains one more layer of hidden nodes that is used to create a bottleneck configuration. Additionally, the Kaldi network operates on spliced features from 31 frames at a time while our network uses a window of 17 frames as the input features.

Our neural network training results are summarized in Table 2. The training error represents neural network labelling accuracy against the SI-84 training set with the monophone GMM labels while the testing error represents the networks accuracy against the Dev'93 test set with monophone GMM labels.

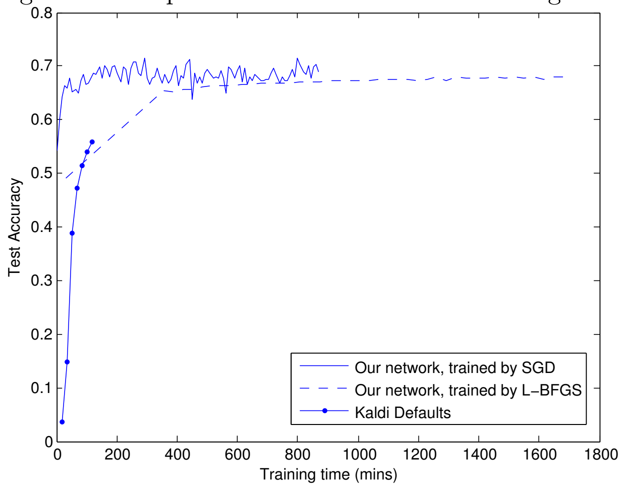
We also capture the amount of time necessary to train the neural networks. Making direct comparisons here is difficult since the Kaldi implementation uses a high degree of parallelization and is written in C++ versus the single threaded MATLAB code used in our implementation.

As can be seen from Table 2 our neural network achieved better accuracies than the Kaldi bottleneck neural network. Interestingly our SGD training achieved better results than L-BFGS. From the graphs though it is possible to see that L-BFGS improves more monotonically while SGD tends to have higher variation in accuracy rates.

Table 2: Neural network results

Neural Network	SI-84 (Training) Accuracy	Dev'93 (Testing) Accuracy	Training Time
Kaldi tandem defaults	61.16%	55.85%	2 hrs
Our network, (SGD) ($\alpha = 1 \times 10^{-4}$)	80%	71.40%	14.5 hrs
Our network, (L-BFGS)	76%	68.02%	28 hrs

Figure 2: Comparison of Test Error vs. Training Time



We took the network trained by SGD and used it to create transformed features to feed into the existing Kaldi GMM training scripts. Results from the entire tandem system compared to Kaldi's current tandem system are captured in Table 3. We were unable to achieve results equivalent to the Kaldi tandem system. We attempted PCA whitening to improve WER but did not achieve improved results. It is possible that removing some layers from our network when transforming features would result in better features for GMM training. This is supported by the fact that Kaldi tandem implementation removes 3 layers and uses the output of the bottleneck nodes as transformed features.

Table 3: End-to-end tandem results

Neural Network	WER, Dev'93	WER, Eval'92
Kaldi tandem defaults	18.82%	11.71%
Our network, (SGD)	34.83%	22.05%

7 Conclusions

Our alternative system achieves neural network output comparable to the results attained by Kaldi's default neural network utilities. However, our resulting word error rates are significantly worse. This suggests that the features being fed from our neural network to the GMM are not well approximated by the Gaussian mixture model. Although we attempted to integrate PCA to reduce the dimensionality of our neural network, we were unsuccessful in achieving a reasonable word error rate. While further research into the implementation of Kaldi's GMM is likely necessary to troubleshoot this discrepancy, we suspect that acquiring features from earlier in the neural network may improve GMM training.

8 Acknowledgements

We would like to thank Daniel Povey for his help in navigating the Kaldi toolset. We are also very thankful for all of the guidance, support and resources that Andrew Maas provided us through this process.

References

[1] H. Hermansky, D.P.W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE, 2000.

[2] Q.V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A.Y. Ng. On optimization methods for deep learning. In *Proc. of ICML*, 2011.

[3] D.B. Paul and J.M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

[4] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[5] M. Schmidt. minFunc - Unconstrained differentiable multibariate optimization in Matlab. `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`, 2012.