Optimizing an Accent Classification System

Jon Bunting
Bradley Department of Electrical & Computer Engineering
Virginia Tech

***Abstract*—This paper seeks to use knowledge of a series of gradient descent variants to infer knowledge about the feature space of natural language processing problems like accent classification with the goal of choosing the best optimizer. Popular optimization techniques are compared based on their experimental performance in training an efficacious, automatic, deep-learning based accent classification system. Although it is quite unanimously thought that Adam and Nadam outperform SGD during training, researchers are skeptical about Adam and Nadam's superiority at finding optimum model parameters that generalize well (C. Wilson et al. 2017). This paper compares mini-batch stochastic gradient descent (SGD), RMSprop, Adaptive moment estimation (Adam), and Nesterov-accelerated adaptive moment estimation (Nadam) as optimization methods. The various techniques were compared to find the optimum method for training and validating an accent classification system based on a one-dimensional convolutional neural network (1-D CNN). To compare results, validation-accuracy learning curves were generated for each network optimized with a different gradient-based optimization technique.**

I. Introduction

Speech occurs when breath vibrates the vocal tract to produce a noise in conjunction with the opening and closing of the larynx and mouth. The human lungs, in combination with the larynx and mouth, are capable of producing highly intricate arrays of sound. As homo sapiens developed language, subsets of these noises were merged to form combinatorial segments called syllables. Various languages, that developed concurrently all over the Earth, isolated different subsets of noises to create phonemes and syllables to use for their verbal communication. Today we refer to these predispositions towards different articulation habits and phonemic pronunciations as accents.

Accents suggest a speaker's national or ethnic background, regardless of context or language being spoken (P. Watanaprakornkul C. E. 2011). Accent detection machines have numerous applications from improving computer-based language teaching programs to helping cognitive speech systems interpret nuances in speech by inferring the speaker's cultural background (Tepperman and Narayanan 2005). Today's Automatic Speech Recognition (ASR) systems are adept for native speakers of a given language, but their performance depreciates significantly for non-native speakers (Bahari et al. 2013). This performance reduction occurs because non-native speakers preserve their speaking style and substitute phonemes from their native language when they encounter a new, similar phoneme in a second language (Bahari et al. 2013).

Many researchers have published work trying to solve this critical preprocessing step towards the development of a robust, truly speaker-independent ASR (Zissman et al. 1996). This paper aims to contribute to this goal by comparing optimization techniques for an accent classification system based on Mel-Frequency Cepstral Coefficients (MFCCs) extracted from raw speech audio data from the George Mason Speech Accent Archive dataset. In comparing optimization techniques, this paper provides intuitions on the nature of MFCC-based feature spaces, the most popular features for both accent classification and speech recognition systems. Further this paper uses those intuitions to justify an appropriate gradient-based optimization technique that generalizes well.

This paper looks at four different variants of gradient descent: Mini-batch stochastic gradient descent, RMSprop, Adam, and Nadam. Each has differences and trade-offs that are briefly summarized. Subsequently, formulas are provided for their parameter update rules for comparison. Lastly, the inferences that can be drawn about the feature space and the optimizer's habits when a particular optimizer excels are described.

II. Background

A. Mel-Frequency Cepstral Coefficients

Popular research in the field of speech and accent recognition breaks speech up by phonemes, single distinct pronunciations, and identifies where the stress and length of time spent making those sounds is most emphasized (Davis and Mermelstein 1980). The stressed part of each phoneme or syllable is referred to as its nucleus (Stuttle 2003). Mel-Frequency Cepstral Coefficients (MFCC) are a popular feature extraction method for prosodic features because they scale an audio signal's power spectrum to approximate the human auditory system's perception of frequency bands. Introduced by Davis and Mermelstein in the 1980's, MFCC's are thought to be a superior feature extraction method for ASR and accent classification because their spectral estimates translate linearly spaced coefficients on a Mel-Scale to logarithmically spaced frequencies to create a filterbank that more accurately filters sound at frequency intervals which humans are more adept at perceiving (Dave 2013) (Davis and Mermelstein 1980).

To calculate the MFCC's, an audio clip is divided into short frames of 25ms when the audio samples are thought to be statistically constant, but still contain enough sample points to generate a reliable spectral estimate. The power spectrum is then calculated for each of the frames and is then passed through a Mel filterbank. A Mel-spaced filterbank is a set of triangular filters that isolate parts of the spectrum and scale them accordingly. Each vector is mostly zeros, except for in the part of the spectrum being isolated and scaled. These scaled values are then summed to create a set of filterbank energies, one for each vector (Fayek 2016) (Lyons 2012). An example of the filtering process can be seen below in Figure 1. After taking the Discrete Cosine Transform of the energies, we have generated our MFCCs. These coefficients are generated for each of the 25ms frames and are then normalized to be fed into a classifier (Fayek 2016) (Lyons 2012).

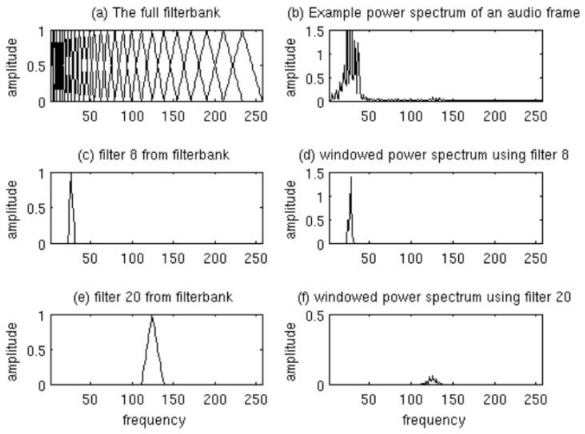

Fig. 1. Diagram demonstrating the Mel-Scale filtering process (Lyons 2012).

For this experiment, MFCC features were extracted from the raw audio using forty Mel-Scale filters. The sample window used was 25ms wide. The audio quality for all of the speech clips was 44.1 kHz and the FFT of the raw audio data was calculated using 512 fast fourier transform (FFT) bins. Upon calculating the MFCCs, cepstral mean and variance normalization was applied before feeding the feature data to the classifier. A generalized diagram of the steps taken to calculate the MFCCs can be seen below in Figure 2.

III. Problem Formulation

There are numerous challenges when choosing a good optimization technique and too often a generic optimizer is chosen without understanding how or if it will be efficient for the feature space for a given deep learning

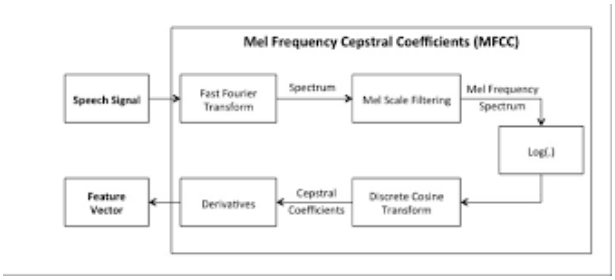

Fig. 2. Diagram demonstrating the steps for calculating MFCCs (Lutter 2014).

problem. For example, an optimum learning rate is necessary. Too small of a step size will result in an excessive number of iterations to converge to the minimum of a given feature space. Too large a step size will result in overstepping and possibly compromising convergence guarantees (Ruder 2016).

Learning rate schedules, i.e. annealing the learning rate during training at a predefined rate, are employed to adjust the learning rate during training to take incrementally smaller steps as the minimum value is approached to guarantee convergence; however, these schedules are unable to adapt to a feature space's characteristics and will apply the same changes in parameter updates regardless of sparsity (Ruder 2016). Further, when feature spaces are non-convex, as is likely in high-dimensional feature spaces, optimizers can get trapped in suboptimal local minima, like saddle points (Dauphin et al. 2014).

This paper introduces a series of gradient descent variants that have adaptive learning schedules: learning schedules that infer knowledge about the feature space of natural language processing problems like accent classification, and take optimal steps to converge in the fewest number of iterations. It has become default to employ these techniques because of their proven improvements on convergence to optimum training parameters. Recently, researchers are beginning to scrutinize these techniques for their ability to tune model parameters to the noise in a given dataset and therefore not generalize well to data dissimilar from the training dataset (C. Wilson et al. 2017). To combat these issues, this paper looks to compare some of the most popular optimization techniques to find the best optimizer that generalizes well for natural language processing problems like the accent classification problem.

A. Vanilla Gradient Descent

Gradient descent is the most popular first-order iterative optimization algorithm for finding the minimum of an objective function, $J(\theta)$.

$$\text{Objective Function: } minJ(\theta),\ \theta \in \mathbb{R}^d$$
$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t)$$

In the case of deep learning applications, network weights that connect layers of a deep neural network are updated according to a minimized cross-entropy loss function. Using backpropagation, the gradient of the loss function for the network can be calculated. Gradient descent, or the method of steepest descent, updates the network parameters by taking a "step" in the negative direction of the calculated gradient for all training samples, $-\nabla_\theta$, towards the minimum of the feature space. This step factor, η, is referred to as the learning rate. Vanilla gradient descent can often be slow to converge and intractable for large datasets because the gradient of the loss function with respect to the entire dataset must be calculated for each iterate of parameter updates (Ruder 2016).

The gradient descent variants being considered in this paper are mini-batch stochastic gradient descent, RMSprop, Adam, and Nadam. They improve upon the vanilla gradient descent architecture in different ways and incur additional costs such as memory usage and additional computations. This paper judges these variants solely based on their generalization error, i.e. validation-accuracy. Validation-accuracy is the only metric being considered in this paper because it is well-established that the improvements that Nadam offers over SGD like Nesterov-accelerated momentum estimates will improve the training accuracy of the model: at

times to a fault. Researchers are finding that SGD and other optimizers can actually reach optimum points that generalize better because they don't tune their models based on the noise specific to the training dataset. Instead SGD can tune a model that focuses solely on the important aspects of the features that generalize better to outside data.

B. Mini-Batch Stochastic Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Mini-batch stochastic gradient descent (SGD) improves upon vanilla gradient descent by performing an update every mini-batch of n training samples. Additionally, instead of calculating the gradient with respect to the entire dataset, the gradient is only calculated with respect to the mini-batch of n samples. This reduces the number of parameter updates as compared to pure stochastic gradient descent which leads to more stable convergence. Further, mini-batch SGD retains all of the benefits of pure SGD by still updating incrementally and not relying on calculating the gradient for the entire dataset to reduce the overhead of redundant gradient computations on large batch sizes [CITE].

C. RMSprop

$$g_t = \nabla_{\theta_t} J(\theta_t)$$
$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

RMSprop is an adaptive learning method that divides the learning rate by an exponentially decaying average of squared gradients. Instead of storing a window of, w, previous squared gradients like Adagrad, RMSprop stores the sum of all previous gradients recursively defined as a decaying average of all previous gradients squared. This improves upon the memory usage of Adagrad by only storing one gradient estimate (Ruder 2016).

D. Adam

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t},\ \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Adam, or Adaptive Moment Estimation, is also an adaptive learning method. Similar to RMSprop, Adam keeps an exponentially decaying average of past gradients, m_t. v_t and m_t are estimates of the first moment (i.e. mean) and the second moment (i.e. the uncentered variance) of the gradients respectively. Typically, the first and second moment vectors are initialized to zero, however this creates a bias towards zero, especially during initial time steps and when the decay rates are small (i.e. β_1 and β_2 are close to 1). To counteract this issue, the bias-corrected moment estimations, $\hat{v}_t$ and $\hat{m}_t$, are calculated and used to update the network parameters (Ruder 2016) (Kingma and Ba 2014).

E. Nadam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$$

Nadam, or Nesterov-accelerated Adaptive Moment Estimation, combines Nesterov-accelerated gradient (NAG) descent and Adam. NAG improves upon vanilla momentum by giving it the prescience to not blindly follow the curvature of the feature space but instead slow down as it approaches the minima. NAG still uses the momentum term to update the parameters, but instead of calculating the gradient with respect to the current iterates parameters, the previous step's parameters are used. Therefore, instead of updating the parameters based on a current approximation of the future parameters, the parameters are updated on a past prediction of the current parameters. After taking a step in the direction of the current approximated parameters, the gradient with respect to the current parameters is used to perform a correction step. The anticipatory update results in increased responsiveness and prevents the parameters from overstepping in the case of

minima located at the basin of steep valleys in the feature space. Nadam incorporates these improvements to vanilla momentum by updating the parameters with the momentum step before computing the gradient. Instead of using the previous update's momentum vector, Nadam uses the current momentum vector to look ahead and directly update the parameters (Ruder 2016) (Goh 2017).

IV. Implementation

A. GMU Speech Accent Archive Dataset

This paper compares the experimental performance in training an efficacious, automatic, machine-learning based accent classification system for the Arabic, Mandarin Chinese, English, and Spanish languages. To develop this accent classification system, speech data was pulled from the Speech Accent Archive dataset from George Mason University. The archive is composed of MP3 audio clips labeled with phonetic transcriptions from speakers with over one-hundred different national and ethnic accents. All of the recordings are of the speakers saying one common English paragraph that includes a number of phonetically ambiguous words for people with different accents. Because the goal of an accent identification model is to detect their pronunciation tendencies, regardless of the words being spoken, the language does not influence the results in a bias towards English pronunciations.

B. One-dimensional Convolutional Neural Network

The model being optimized in this paper is a one-dimensional convolutional neural network (1-D CNN). For the accent classification problem, the rate of speech is not an issue. Instead it is the sharp transitions in human-audible frequencies that must be learned for accent classificationm making a 1-D CNN an appropriate choice. The model used was comprised of nine convolutional layers, each with batch normalization and ten percent drop-out to combat over-fitting. Max pooling was applied to all but the output of the first layer. Lastly, an affine layer was appended to the end of the network.

This model was built using the Keras python libraries with a Tensorflow backend. All coding for this project was done in interactive python (IPython or Jupyter Notebooks). After downloading the audio data from the GMU archive servers, the archive data was split into two distinct datasets, a training/testing dataset and a validation set. Each dataset contained an equal proportion of audio clips from each language and were shuffled to create a random distribution for the classifier. The audio clips were padded with trailing zeros to fit each clip into a standard size array and the datasets were pickled for easier access. Due to the relatively small available dataset, ninety percent of the audio clips were used for the training/testing dataset while the remaining ten percent were used for the validation set.

Finally, MFCC accent features were extracted from the raw audio files and fed into the 1-D CNN for training or for validation. As the classifier was incrementally trained, a learning curve was generated by predicting values the model hadn't seen during training from the validation set. These validation scores were recorded and used to generate learning curve graphs to evaluate the performance of the model at different training epochs for each of the accent feature vectors.

V. Results

The results generated from running the experiments ranged greatly from promising to lackluster. Although all of the optimization techniques compared in this paper are popular for tuning a deep neural network used for ASR and accent classification, the experiments were unable to unanimously confirm these results. Unfortunately, inconclusive results were collected due to computing constraints. Preliminary results were collected and are displayed below. The following results in Figure 3 show a comparison of validation accuracies for the same 1-D CNN model with different optimizers

being used. The results are sporadic despite a general trend towards improving generalization when training the model using more epochs.

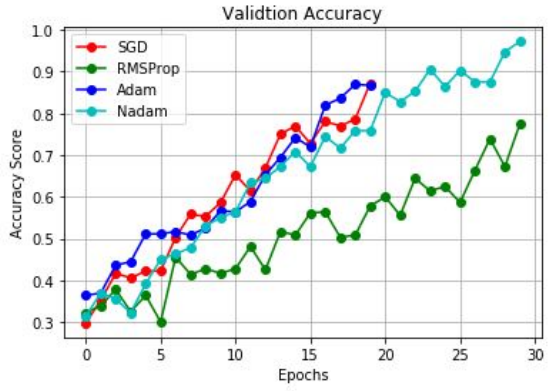

Fig. 3. Graph comparing optimizer's effect on validation accuracy.

The most successful optimization method was mini-batch SGD, with a max validation score after twenty epochs of 87.22%. Given the upward trend of the data, it is suspected that the model would have continued to increase its cross-validation scores if more computing time were available to perform more training epochs. The second most successful optimization method was Adam, with a max validation score of 86.66%. The third most successful optimization method was Nadam, with a validation score of 75.83% after twenty epochs. Nadam was able to run for ten extra epochs as compared to SGD and Adam and was able to achieve a max validation score of 97.22%. The least successful optimization method after running for twenty epochs was RMSprop, with validation score of 57.77%. RMSprop also had extra computation time and was allowed to run for ten additional epochs, eventually achieving a max validation score of 77.49%.

VI. Future Work

In continuation of this project, future work will focus on improving the model and achieving increasingly better generalization. To do this, the optimization techniques employed in this paper to learn more about the feature space of accent MFCC data will be used to ensure that the most appropriate gradient descent variant is being utilized. After the model has been scrutinized and is confidently performing near its optimum potential with the George Mason Speech Accent Archive dataset, more data will be sought out. Data was limited for this experiment. In total, only 400 audio clips were available from the GMU Speech Accent Archive to ensure an equal distribution of clips from each of the four languages. This limitation could contribute to potential overfitting issues when trying to generalize the classifier to more languages or data outside of the GMU Accent Archive dataset. If the experiment were to be repeated, a different dataset would be used, or more data would be collected to contribute to the archive.

After improving results with a set of standard datasets, future work will be done to generalize the model to more raw data sources with a wider range of more pragmatic content. To do this, more focus will be placed on speech segmentation to isolate the nucleus of syllables and phonemes to more accurately capture accent-specific prosodic features. The length of these variable segments and the amount of pause between them are two features that were unable to be represented in the current model because a fixed length window was used, but are thought to be exceptional accent indicators (Tepperman and Narayanan 2005). At a minimum, these features should be able to suggest to a classifier if the speaker is native or non-native to the language being spoken (Tepperman and Narayanan 2005). Additionally, semi-supervised models might be relevant to this problem to better utilize the large amount of unlabeled speech data available on the internet.

VII. Conclusion

This paper used knowledge of a series of gradient descent variants to infer knowledge about the feature space of MFCC-based accent data with the goal of choosing the best optimizer. Inspired by researchers who are skeptical about the overall performance of superior optimizing

techniques like Adam and Nadam, this paper compared optimizers solely based on their ability to converge to model parameters that generalized well. After collecting preliminary results, mini-batch stochastic gradient descent outperformed RMSprop, Adam, and Nadam in validating an accent classification system based on a one-dimensional convolutional neural network (1-D CNN). Going forward, more conclusive results will be collected and a more robust model will be developed and optimized to solve the accent classification problem in working towards a truly speaker-independent automatic speech recognition system.

REFERENCES

Bahari, Mohamad et al. (2013). *Accent recognition using i-vector, Gaussian Mean Supervector and Gaussian posterior probability supervector for spontaneous telephone speech*.

C. Wilson, Ashia et al. (2017). "The Marginal Value of Adaptive Gradient Methods in Machine Learning". In:

Dauphin, Yann et al. (2014). "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *CoRR* abs/1406.2. URL: http://arxiv.org/abs/1406.2572.

Dave, Namrata (2013). "Feature extraction methods LPC, PLP and MFCC in speech recognition". In: *International Journal For Advance Research in Engineering And Technology(ISSN 2320-6802)* Volume 1.

Davis, S and P Mermelstein (1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4, pp. 357–366. ISSN: 0096-3518. DOI: 10.1109/TASSP.1980.1163420.

Fayek, H (2016). "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between." In: vol. 1. URL: http : / / haythamfayek . com / 2016 / 04 / 21 / speech - processing-for-machine-learning.html.

Goh, Gabriel (2017). "Why Momentum Really Works". In: *Distill*. DOI: 10.23915/distill.00006. URL: http : / / distill . pub / 2017 / momentum.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: {A} Method for Stochastic Optimization". In: *CoRR* abs/1412.6. URL: http://arxiv.org/abs/1412.6980.

Lutter, M (2014). "Mel-Frequency Cepstral Coefficients." In: vol. 1. URL: http://recognize-speech.com/feature-extraction/mfcc.

Lyons, J (2012). "Mel Frequency Cepstral Coefficient (MFCC) tutorial." In: vol. 1. URL: http : / / practicalcryptography . com / miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

P. Watanaprakornkul C. E., P Chien (2011). "Accent Classification." In: vol. 1, pp. 1–5.

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *CoRR* abs/1609.0. URL: http://arxiv.org/abs/1609.04747.

Stuttle, M N (2003). "A Gaussian Mixture Model Spectral Representation for Speech Recognition." In: vol. 1, pp. 1–163.

Tepperman, Joseph and Shrikanth Narayanan (2005). "Automatic Syllable Stress Detection Using Prosodic Features for Pronunciation Evaluation of Language Learners". In: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on* 1.

Zissman, Marc et al. (1996). *Automatic dialect identification of extemporaneous conversational,Latin American Spanish speech*.