Deep Learning for Automatic Speech Recognition

Ryan Hartsfield Garrett Lewellen

May 4, 2015

Introduction

In this paper we examine two deep learning method for automatic speech recognition: deep neural networks and convolutional neural networks based on the works of [DYDA12] and [AMJ+14] respectively.

TODO: Ryan - Talk about speech recognition in general, linguistics variations,

Traditional Approach

TODO: Ryan GMM-HMMs

Deep Neural Networks Approach

In this section we discuss the work on Large-Vocabulary Speech Recognition (LVSR) of [DYDA12] consisting of a context dependent hidden markov model and deep neural network hybrid architecture (CD-HMM-DNN) for the acoustic model. We will begin with an overview of the general architecture, then explain the algorithms used for pre-training, and conclude a discussion on the general procedure for training. Discussion of experimental results for this approach are deferred to the results section so that they can be compared to the convolutional neural network approach.

Architecture

TODO: Include figure of architecture?

To motivate their architecture, [DYDA12] rely on the standard noisy channel model for speech recognition presented in [JM08] where we wish maximize the likelihood of a decoded word sequence given our input audio observations:

$$\hat{w} = \underset{w \in \mathscr{L}}{\operatorname{argmax}} \, \mathbb{P}(w|x) = \underset{w \in \mathscr{L}}{\operatorname{argmax}} \, \mathbb{P}(x|w) \, \mathbb{P}(w) \tag{1}$$

Where $\mathbb{P}(w)$ and $\mathbb{P}(x|w)$ represent the language and acoustic models respectively. [JM08] state that the language model can be computed via an N-gram approach, but [DYDA12] do not state their method, instead the authors put their efforts into explaining their acoustic model:

$$\mathbb{P}(x|w) = \sum_{q} \mathbb{P}(x, q|w) \, \mathbb{P}(q|w) \cong \max \pi(q_0) \prod_{t=1}^{T} a_{q_{t-1} q_t} \prod_{t=0}^{T} \mathbb{P}(x_t|q_t) \tag{2}$$

Here the acoustic model is viewed as a sequence of transitions between states of tied-state triphones which [DYDA12] refer to as senones TODO: EXPLAIN WHAT SENONES ARE AND WHERE THEY COME FROM AND WHY THEY ARE USEFUL. §10.3 of [JM08] discusses Context-dependent Acoustic models: triphones and [LMM+14] talks about senone (tied-state triphones) which gives us the context dependent aspect of the architecture. The model assumes that there is a probability $\pi(q_0)$ for the starting state, probabilities $a_{q_{t-1} q_t}$ of transitioning to the state observed at step $t-1$ to step t, and finally, the probability of the acoustics given the current state q_t. [DYDA12] expand this last term further into:

$$\mathbb{P}(x_t|q_t) = \frac{\mathbb{P}(q_t|x_t) \, \mathbb{P}(x_t)}{\mathbb{P}(q_t)}$$

Where $\mathbb{P}(x_t|q_t)$ models the tied triphone senone posterior given mel-frequency cepstral coefficients (MFCCs) based on 11 sampled frames of audio TODO: EXPLAIN MFCCs AND WHY THEY ARE USEFUL; Jurasky seems to have a good explanation, may need to consult other resources, $\mathbb{P}(q_t)$ is the prior probability of the senone, and $\mathbb{P}(x_t)$ can be ignored since it does not vary based on the decoded word sequence we are trying to find.

Based on this formalism, [DYDA12] chose to use pre-trained deep neural networks to estimate $\mathbb{P}(q_t|x_t)$ using MFCCs as DNN inputs and taking the senone posterior probabilities as DNN outputs. The transitioning between events is bested modeled by hidden markov models whose notation, π, a, and q appears in Eqn. (2). Now that we have an overview of the general CD-DNN-HMM architecture, we can look at how [DYDA12] train their model.

Pre-Training

Given the DNN model we wish to fit the parameters of the model to a training set. This is usually accomplished by minimizing a likelihood function and deploying a gradient descent procedure to update the weights. One complication to this approach is that the likelihood can be computationally expensive for multilayer networks with many nodes rendering the approach unusable. As an alternative, one can attempt to optimize a computationally tractable surrogate to the likelihood. In this case the surrogate is the contrastive divergence developed by [Hin02]. This sidestep enabled [HOT06] to develop an efficient pre-training process whose results can then be refined using a few iterations of the traditional result. In this portion of the paper we discuss the work of [Hin02] and explain the greedy algorithm of [HOT06] before going on to discuss the high-level training procedure of [DYDA12].

To understand the pre-training process, it is necessary to discuss Restricted Boltzmann Machines (RBM) and Deep Belief Networks (DBN). RBMs are an undirected bipartite graphical model with Gaussian distributed input nodes in a visible layer connecting to binary nodes in a hidden layer. Every possible arrangement of hidden, h, a visible, v, nodes is given a energy under the RBM model:

$$E(v,h) = -b^T v - c^T h - v^T W h \tag{3}$$

Where W is the weight of connections between nodes and vectors b and c correspond to the visible and hidden biases respectively. The resulting probability is then given by:

$$\mathbb{P}(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{4}$$

Where Z is a normalization factor. Based on the assumptions of the RBM, [DYDA12] derive expressions for $\mathbb{P}(h=1|v)$ and $\mathbb{P}(v=1|h)$ given by:

$$\mathbb{P}(h=1|v) = \sigma(c + v^T W) \qquad \mathbb{P}(v=1|h) = \sigma(b + h^T W^T) \tag{5}$$

Where σ is an element wise logistic function. [DYDA12] argue that Eq. (5) allows one to repurpose the RBM parameters to initialize a neural network. Training of the RBM is done by stochastic gradient descent against the negative log likelihood:

$$-\frac{\partial \ell(\theta)}{\partial w_{ij}} = \left\langle \frac{\partial E}{\partial \theta} \right\rangle_{\text{data}} - \left\langle \frac{\partial E}{\partial \theta} \right\rangle_{\text{model}} \tag{6}$$

however [DYDA12] point out that the gradient of the negative log likelihood cannot be computed exactly since the $\langle \cdot \rangle_{\text{model}}$ term takes exponential time. As a result, the contrastive divergence method is used to approximate the derivative:

$$-\frac{\partial \ell(\theta)}{\partial w_{ij}} \approx \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_1 \tag{7}$$

where $\langle \cdot \rangle_1$ is a single step Gibb Sampled expectation. TODO: What is the underscore data term?; How will you explain in terms of what you read in the product of experts papers?; where's the kullback leiber divergence...?

Now that we have an understanding of RBMs, we can shift our focus to DBNs. A Deep belief network are multilayer models with undirected connections between the top two layers and directed between other layers. To train these models, [HOT06] had the insight to treat adjacent layers of nodes as RBMs. One starts with the bottom two layers and trains them as though they were a single RBMS. Once those two layers are trained, then the top layer is treated as the input layer of a new RBM with the layer above that layer acting is the hidden layer of the new RBMs. The sliding window over the layers continues until the full DBN is trained. TODO: Ok... so that's good and all, but where's the connection between DBN and DNN?

Training

Training of the CD-DNN-HMM model consists of roughly a dozen involved steps. We won't elaborate here on the full details of each step, but will instead provide a high-level sketch of the procedure to convey its general mechanics.

The first high-level step of the procedure is to initialize the CD-DNN-HMM model. This is done by first training a decision tree to find the best tying of triphone states which are then used to train a CD-GMM-HMM system. Next, the unique tied state triphones are each assigned a unique senone identifier. This mapping will then be used to label each of the tied state triphones. (These identifiers will be used later to refine the DNN.) Finally, the trained CD-GMM-HMM is converted into a CD-DNN-HMM by retaining the triphone and senone structure and HMM parameters. This resulting DNN goes through the pre-training procedure discussed in depth earlier.

The next high-level step iteratively refines the CD-DNN-HMM. To do this, first the originally trained CD-GMM-HMM model is used to generate a raw alignment of states TODO: figure out what it's aligning to which is then mapped to its corresponding senone identifier. This resulting alignment is then used to refine the DBN by backpropagation. Next, the prior senone probability is estimated based on the number of frames TODO: elaborate more on these frames somewhere paired with the senone and the total number of frames. These estimates are then used to refine the HMM transition probabilities to maximize the features. Finally, if this newly estimated parameters do not improve accuracy against a development set, then the training procedure terminates; otherwise, the procedure repeats this high-level step.

TODO: Discuss computational time to train

Convolutional Neural Networks Approach

TODO: Ryan TODO

In this section we explore the work done in [AMJ+14] on the use of convolutional neural network-hidden markov model hybrid (CNN-HMM) for phone recognition performed on the TIMIT dataset and a large-vocabulary voice search task. In this model, temporal variability is handled by the HMM, while convolutions occur along the frequency axis of the acoustic signal. We will begin by exploring the overall structure of the network, addressing the shape of the input and looking closely at the convolution and pooling plys. Pre-training is similar to that used in [DYDA12], so we will instead focus on the fine-tuning of parameters using the back-propagation algorithm. We will then transition into the results section where we will compare the performance of the CD-DNN-HMM with that of the CNN-HMM.

Architecture

TODO: Go over structure of input, why not MFCCs, why frequency axis

Over the past few years, convolutional neural networks have proven to be highly successful in image recognition tasks. Not only have they achieved the highest results on the MNIST character recognition datasetTODO: cite, but they've even proven to outstrip hand-crafted object recognition systems on the much more complex ImageNet dataset.

Convolution Ply

Without padding the input with dummy values, the convolution operation would produce a lower-dimensional feature map where each dimension decreases by the filter size F minus one. If this is done, there will be the same number of units in the input feature map as in that of the convolution ply.

Pooling Ply

The pooling ply in the network acts as a next step from the convolution ply where the resolution of the feature maps is reduced. TODO: Why is this important

The pooling function used in a CNN is normally either one of *averaging* or *maximization*. In this work, a max-pooling function was used throughout the network:

$$p_{i,m} = \max_{n=1}^{G} q_{i,(m-1)xs+n}$$

where G is the pooling size and s is the shift size. If $G = s$, there is no overlap between pooling windows. TODO: cite for maxpool vs average + 1 sent [AMJ+14] decided to vary these independently to discover whether or not this held true for acoustic processing as well.

Limited Weight Sharing

TODO: Explain why limited compared to image processing method

Training

TODO: back prop with weight matrices

Pre-training of the CNN is also done with RBMs, with some slight modifications to account for both convolution and pooling plies. This results in a CRBM, which will not be covered here.

Experimental Results

System Configurations

[DYDA12] report that their system relies on nationwide language model consisting of 1.5 million trigrams. For their acoustic model, then use a five hidden layer DNN with each layer containing 2,000

hidden units.

TODO: Add in CNN details

Datasets

TODO: Garrett

Results

		Bing Mobile		TIMIT	
	Architecture	Dev.	Test	Dev.	Test
Sentence Accuracy	CD-GMM-HMMM	70.3%	68.4%		
	CD-DNN-HMM	71.8%	69.6%		
Phone Error	CNN-HMM				20.07%

Table 1: Accuracy and error rates reported by [DYDA12] and [AMJ+14].

Direct comparison of the two systems is complicated by the fact that both papers report different metrics against different datasets. [DYDA12] reports a sentence level accuracy rate, while [AMJ+14] reports the phone error rate.

Conclusions

TODO: Garrett TODO

References

[AMJ+14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, 22(10):1533–1545, 2014.

[DYDA12] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech & Language Processing*, 20(1):30–42, 2012.

[Hin02] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[JM08] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, 2nd Edition*. Prentice Hall, 2008.

[LMM+14] Haizhou Li, Helen M. Meng, Bin Ma, Engsiong Chng, and Lei Xie, editors. *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*. ISCA, 2014.