ML Final Report

- As a whole, we were able to learn an excellent amount of details on the problem of speech recognition from neural networks.
    - Throughout the semester we read literature that all provided different ideas on how to approach this problem, from recognizing spectrograms of wav files to performing FFT on a wav file for the necessary data to train on.
- We decided to choose the method of performing a FFT on the wav files, and taking the data at the 23 most critical bands at frequencies of 75, 100, 200, 300, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 2800, 3000, 3200, 3500, 4000, 4500, 5000.
    - We reached this idea from the paper of "Analysis of the Energy Distribution in Speech" from I.B Crandall and D. Mackenzie.
- The approach that we took was to recognize the 43 different phonemes. We took hundreds of samples of the phonemes being presented, performed FFT, and saved the real results.
- This gave us a dimensionality of 46 for each of the classes.
- For our neural network, we decided to do on hidden layer.
- The input to our network was a N X D matrix, where n is the number of training samples, this depended on how large we decided to make our batch.
- Then for our hidden layer, we make the dimensions D x H, where H is the number of nodes in the hidden layer.
- For our output layer, we have dimensions of H X C, where C is the number of classes. Therefore in our final output, we have an N X C matrix, and we can find the max of each row to determine which class is the predicted output.
- For the hidden layer, we used a ReLU activation, basically just taken the max of 0 and the result from the linear classifier, dot product of hidden layer weights and input x + the bias of hidden layer.
- Then to get the scores, we simply used the dot product of the weights on the output layer and the result from the hidden layer plus the bias of the output layer.
- We used a softmax regression for calculating the loss, which used the cross-entropy loss and regularization.
- We first computed the probabilities of each class from exponentiating the scores, dividing the result with the sum of the exponentiated scores along the columns, then we calculated the correct log probabilities by taking the the -log of the correct classes in the probability matrix just calculated.
- Then, the final loss is just sum of correct logs / num of training examples + the regularization which is regularization constant times the sum of weights of hidden layer squared + the regularization constant time the sum of weights of output layer squared.

- The next thing we calculated was the gradients which are to be used to change the weights and biases of the network.
- The gradient was calculated from the following steps:
    - First we took the matrix of probabilities, then we subtracted one from each score that is supposed to be the correct class, then divided that matrix but the number of inputs or training examples, we will denote this ds
- Then we proceeded by backpropagating through the network
    - To get partial with respect to the weights of output layer, we dotted the transpose of the hidden layer results (from ReLU activation) and the matrix ds
    - Then for the partial with respect to the bias of output layer, we summed the ds matrix along the rows
    - Then we back propagated through the hidden layer, first we took the dot product of ds with the weights of the output layer, we denote this matrix dh.
    - Then we backprop again because of the ReLU activation, we do this simply by making things zero in dh, where during the forward pass, the input to the hidden layer was less than or equal 0.
    - With this dh matrix, we are able to now calculate partial with respect to weights of the hidden layer. To do this we calculate the dot product of the transpose of the input matrix and the dh matrix.
    - To calculate the partial with respect to the bias of the hidden layer, we sum the dh matrix along the rows.
    - Then we add the regularization to the partial with respect to the weights. We calculate the regularization from simply 2 times the corresponding weight matrix.
    - Then to make changes in the weights and biases we add -learning rate * the corresponding partial to the weights and biases.
    - Finally, we calculated the optimal learning rate and regularization by testing various strengths and looping through until we find the optimal.
- For the input of a wav file, we decided to break up the file into ~.1 seconds and concatenated the results to generate what the file approximately says.

<u>Previous ML Checkpoints are attached after this page</u>

Franky Liang

- http://mathworld.wolfram.com/SigmoidFunction.html
  - Read about what the sigmoid function is
- http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/
  - Convolutions used to recognize images
  - CNN work well with images because images are 2D
  - I plan to do the exercises to better understand each step
- https://arxiv.org/pdf/1610.00087v1.pdf - VERY DEEP CONVOLUTIONAL NEURAL NETWORKS FOR RAW WAVEFORMS
  - Results indicate that more layers in a CNN leads to better results
  - Perhaps try adding more layers when we implement our CNN
- https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN_ASLPTrans2-14.pdf - CONVOLUTIONAL NEURAL NETWORKS FOR SPEECH RECOGNITION
  - Speech feature vectors into feature maps
  - Read again when implementing, to truly understand details
- https://www.sri.com/sites/default/files/publications/paper_odyssey14_y.lei_final.pdf
  - Using posteriors with CNN
- https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/
  - Friendly introduction on CNNs
  - Will review again when implementing CNN

Franky Liang
CAP6610/CIS4930

Our project will focus on speech recognition using convolutional neural networks. We will turn a user inputted WAV file into a spectrogram, and convert that spectrogram to pixels. With those pixels, we will analyze what is spoken. We will train our CNN based on data we find on the web such as that from Voxeo.

Started working on turning WAV file into a spectrogram, and then reading the pixels from that spectrogram.

https://github.com/franktank/practice-python/blob/master/wav_test.py

Code that we have written so far to convert WAV file into data that we can use to analyze.

Continued Reading:

https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

- Filter multiplied with receptive field (element wise multiplication), then multiplications are summed up (i*j*k multiplications where i,j,k size of filter). Used to represent this area.

Scrape https://evolution.voxeo.com/library/audio/prompts/airports/NorthWestAirlines.wav
To obtain WAV files with name correlated to what is spoken

Other resources:
http://karpathy.github.io/2015/10/25/selfie/
http://stp.lingfil.uu.se/~santinim/ml/2014/JurafskyMartinSpeechAndLanguageProcessing2ed_draft%202007.pdf
http://www.deepsky.com/~merovech/voynich/voynich_manchu_reference_materials/PDFs/jurafsky_martin.pdf
http://cs231n.github.io/neural-networks-1/
https://papers.nips.cc/paper/3674-unsupervised-feature-learning-for-audio-classification-using-convolutional-deep-belief-networks.pdf
https://www.youtube.com/watch?v=AgkfIQ4IGaM - Visualization
http://yosinski.com/deepvis
http://www.matthewzeiler.com/pubs/arxive2013/arxive2013.pdf

Franky Liang

- We decided to take a Fast Fourier Transform of our WAV files instead of simply converting to a spectrogram.
- The FFT can be done efficiently in O(n lg n) time which is particularly useful in examining large data sets.

- We wrote code for the FFT from reading about the Cooley-Tukey FFT algorithm.

- http://cs231n.stanford.edu/reports/tema8_final.pdf
- https://arxiv.org/abs/1601.06815
- https://arxiv.org/pdf/1506.03767.pdf
- These papers above suggest that using a FFT helps reduce the computational cost of convolutions.

- Also, other papers such as http://www.kdd.org/kdd2016/papers/files/rpp0534-chenA.pdf indicate that it might be better to perform a DCT instead of a FFT on the WAV file.
    - The reason for this DCT provide real number results.
    - However, DCT is less efficient, which might be okay in our situation since we will most likely not have a dataset so large that computations will take too long.

- Our code for the FFT can be seen at:

- Our next step is to start developing our CNN to handle this data. We will need to determine how to setup our model and train it.

- Interestingly. We also noted this article: https://arxiv.org/abs/1411.1792.
    - We can train our neural net with vast amounts of images that could be unrelated to speech recognition, simply because they lower layers are usually similar throughout CNNs, it is towards the later layers that need to be specialized.

http://www.openslr.org/12/
http://www.speech.cs.cmu.edu/databases/an4/
http://archive.ics.uci.edu/ml/

Currently we are working on determining our data to train on. We will have to figure out the best data set to use and possibly how to normalize throughout all the data.

On top of the determining what data to train on, we have started planning out the infrastructure of the CNN that we will build.

We have been discussing how many layers, how to set the initial weights and biases, how many neurons per layer, what each neuron should analyze, what function we will use to help classify, what the cost function will be, what the inputs will be, how we should set the output, and how we will determine / experiment to find the best learning rate.

"Analysis of the Energy Distribution in Speech" (http://journals.aps.org/pr/abstract/10.1103/PhysRev.19.221) , we found that 23 bands is all that is needed to analyze the energy distribution in speech

With this information, we plan to use 23 nodes for our input.

Furthermore, we plan to have 45 output nodes, 1 for a pause and 44 for each phonemes.
We will play with different number of nodes and layers.

My partner and I began building our convolutional neural network.

We primarily resort to the book Neural Networks and Deep Learning for reference.
http://neuralnetworksanddeeplearning.com/chap6.html

We began with an overview of the layers and the stochastic gradient descent method.

Worked on implementing the stochastic gradient descent part of our network.

Looked more into Convolutional Layer, Pooling Layer, and Fully connected layer. Will implement this week and decide on layer patterns after.

Contemplating whether to use MFCC on the dataset or to simply turn the WAV into spectorgrams.

Might just do spectrograms to focus more so on implementation of the CNN.