Lab 1 – Feed forward networks

In this lab, you will use the MIT/TIDIGITS corpus (Leonard and Doddington, 1993) to learn about model architecture and regularization. Skeleton code has been provided on Blackboard that solves your previous problem set and provides some additional functionality. Highlights of the functionality includes:

- automatic caching of the PCA analysis and feature generation, saving time after the first run. These files are not compatible across machine architectures.
- a data structure that will allow you to specify keras network architectures. This will allow you to use the same training and code, passing in a model architecture. See feedforward.generate_model for documentation on the data structure. An example of using this can be seen in the provided skeleton for driver.py. Furthermore, one can automate the generation of different architectures by constructing the data structure algorithmically.
- The DFTStream class has new features to support Mel filterbank processing. Using a specfmt of "Mel" will generate the output of Mel filters. The constructor allows one to specify how many Mel filters are used. The librosa package is required for the Mel filterbank construction, see DFTStream for instructions on how to install it.
- utils.get_class(file_list) uses regular expressions to identify the digit class from filenames. Use this to pull out labels from a list of files.

Several portions of the code need to be completed in addition to the driver which ties everything together:

- Generation of fixed length spectrograms from endpointed speech, write this in mydsp.utils.fixed_len_spectrogram. The class Endpointer will run speech detection from an RMS stream and allow you to find the frames associated with speech. If you have problems getting this to work, start off truncating to a fixed time somewhere near the middle of the file. You'll end up cutting off more of the speech, but it will let you proceed and you can come back and fix it if you have time.
- Generation of features: mydsp.features.extract_features_from_corpus(). Given a set of files, generates fixed length features from a spectrogram (numpy's flatten() is very good for this). Returns a matrix of feature vectors for a corpus.
- myclassifier.feedforward.CrossValidator – Class for performing cross validation. Constructor should set up the test using scipy's StratifiedKFold class and iterate across the folds. For each fold invoke the train_and_evaluate__model which trains a Keras model with K-1 folds and tests on the last fold. Note that Keras returns accuracy, compute the **error rate** (1 – accuracy).

In this lab, you will explore:

- wide vs. deep architectures (e.g. wide vs. deep)

- regularization: L1, L2, and dropout.
- Optional: feature extraction: Mel filtered cepstral coefficients (MFCC) vs. PCA. MFCC is a truncated discrete Fourier transform of the Mel filterbank outputs. For details, read Oppenheim and Schafer (2004) or Picone (1993).

Unlike problem sets, there is less direction and you are expected to explore.

Penalty regularizers such as L1 and L2 are implemented as classes in keras, and can be instantiated from keras.regularizers.l1 or keras.regularizers.l2 with argument λ (try 0.01 for starters). When creating Dense layers, use the kernel_regularizer keyword argument to indicate that the weights should be subjected to the regularization penalty. As an example, a 20-unit Dense layer with 20 inputs could be created as follows:

```
Dense(20, kernel_regularizer=keras.regularizer.l2(0.01)
```

Dropout is handled as a separate layer type by keras. It is inserted *prior* to the layer from which random nodes are to be dropped, and only requires a single parameter, the dropout rate. On very large networks, it is not uncommon to use dropout rates of .5, but your dataset is small by deep net standards and it is suggested to try lower rates (e.g. .2 or .3).

In this set of experiments, you will use the TIDIGITS train directory as your development set. Use k-fold cross validation on the development data to explore the various architectures that you propose. Once you have identified your best architecture, take the k different models that were trained and evaluate them on independent validation data. Be sure to use the same PCA transform as you used on your development set for the evaluation data. Estimating a separate PCA axis set from the test data will result in incompatible features. As the test directory was held out from the development set, the files contained therein can serve as an independent validation set. By testing all the models from your best architecture, you can estimate the variance of your results on the validation data.

Write a lab report to document your findings. I strongly recommend reading carefully the Hesselbach et al. (2012) article on writing scientific papers as they outline the structure that is expected for this report. Additional points to remember:

- Reports are not code documentation. It is fine to mention the tools that you used, e.g. "I implemented feed-forward neural networks in Python 3.5 using TensorFlow 1.9 and the Keras library 2.2.2. The first experiment addressed L1 regularizers, and allowed L1 penalties to vary from..." Note that we did not reference the details of how we did this in keras. We did not indicate what function this was in nor the keyword parameter that was provided to the Dense network.
- Representing your data. A picture is indeed worth a thousand words, but thousands of pictures will numb the mind. Think about the story you want to tell, then design plots that are representative of what you want to show. Suppose that you designed an experiment to show what happens as a net becomes deeper. You could plot the mean error rates on your development set against the network depth and then add dotted lines at +/- the standard deviation of your error. Of

course, you would have to hold the other parameters of your network constant, so this would not automatically be the right representation to tell a story of width vs depth although you could certainly reframe it do so (e.g. hold the number of nodes constant and play with the architecture). If you plan on using a black and white printer, make sure that your plots are distinguishable in black and white and avoid referring to the "blue" line. Even if you do have beautiful plots, it is nice to show actual numbers in tables as values are difficult to read off plots. If you are interested in making better plots, I would suggest browsing Edward Tufte's *Viusal Display of Quantative Information* (1983). A confusion matrix would be useful in your output. Confusion matrices are tables showing counts of the actual class vs. the predicted class. See scikit's sklearn.metrics.confusion_matrix for details.

- Avoid contractions in formal writing.
- The one area that your manuscript will differ from papers that are submitted for publication is that your methods should describe in detail things that are already published. As an example, when describing L2 regularization, you should explain what L2 regularization does in your own words (citing appropriately) and include formula. You do not need to describe in detail things that were covered in previous assignments (e.g. Fourier transforms).
- Equations: Do not cut and paste equations from the web. Use Word/LaTeX/Libre Office's equation typesetting tools. Cut and pasted image rarely look professional when printed.
- Do not wait until the last minute to write your report. Start on it early, you can be writing the introduction and methods even before you finish your experiments. The report is worth 50% of the points. If you are not comfortable with your writing skills, consider going to SDSU's writing center. You can schedule an appointment here.

This lab is worth 120 points, half from your code, half from your lab report. Hard- and soft-copies of both your code and lab report should be provided. Please remember to save trees and only print code files that you modified (I know what the code that I wrote looks like). The electronic copies will be submitted as if they were two different assignments as the report will be evaluated by the TurnItIn service for plagiarism and style. Note that you have the opportunity to fix any problems that are found.

REFERENCES

**Hesselbach, R. A., Petering, D. H., Berg, C. A., Tomasiewicz, H. and Weber, D.** (2012). A Guide to Writing a Scientific Paper: A Focus on High School Through Graduate Level Student Research. *Zebrafish* **9**, 246-249, doi:10.1089/zeb.2012.0743.

**Leonard, R. G. and Doddington, G.** (1993). TIDGITS, vol. 2007: Linguistic Data Consortium, Philadelphia.

**Oppenheim, A. V. and Schafer, R. W.** (2004). From frequency to quefrency: a history of the cepstrum. *IEEE Signal Proc. Mag.* **21**, 95-99,106.

**Picone, J. W.** (1993). Signal modeling techniques in speech recognition. *Proc. IEEE* **81**, 1215-1247.

**Tufte, E. R.** (1983). The visual display of quantitative information. Cheshire, Conn. (Box 430, Cheshire 06410): Graphics Press.