Practical 1 - Speech commands

Yonatan DELORO

February 15, 2019

1 I. Classification of single voice commands

No question. CF. code

Speech features First of all, as we aim at classifying single English words, we are not interested in features caraterzing the pitch but only in the features defining. Hence the MFCC features appears a priori as a better option than the Mel-filterbanks.
The parameters of the speech features such as the min/max frequency or window size are poorly chosen, look inside the resources mentioned in the class to find the best parameters for mel-filterbanks and MFCC.
- Mel-filterbanks
- MDP

Classification models Two different models were tried :
(i) Logistic Regression (LR) A first proposal for this multi-label classification problem is to perform a logistic regression for each class against all others. Each data is therefore labelled with the class obtaining the largest probability. I added a l2 regularization term to the cross-entropy loss of the LRs to allow for better generalization. In other words, the loss to be optimized for each of the 30 LR is : $\mathcal{L}(x_i, y_i) = -\sum_{i=1}^{N} y_i \log \sigma(wx_i) + (1-y_i)\log(1-\sigma(wx_i)) + \alpha||w||^2$. I chose the α parameter maximizing the accuracy score obtained on the validation set.
(ii) Neural network
The neural net proposed is a shallow neural net, far from the best you can train. You should try bigger, deeper architectures, different types of regularization, activation functions, learning rate and so on. You can change the** Runtime of your colab instance and use a GPU**.
- MDP
- Convolution Network

Data normalization and augmentation For better generalization, I also observed : - the impact of normalizing (centering and scaling) the speech features. - the impact of augmenting the original train dataset of 300 instances per class with noisy instances. More precisely, the process of augmentation is the following : I selected a random subset of the $300 * 30$ waveforms of the train dataset of a given size, and to each of these I added a random sequence extracted from the waveform of one of the 6 given background noises (artificial or real) lasting the same amount of time as the original sample waveform.
A standard way of improving generalization is to do mean-variance normalization on your data set. This is done by computing the mean and variance of each feature dimension on the entire training set, and then use it to normalize train, valid and test set
The dataset provides noises samples, either artificial (pink, white noise) or real (dishes, bike) in the folder backgroundnoise. You can try augmenting your dataset by adding noise to the waveforms before computing the features
The model is only trained on 300 examples per class, if your hardware allows it, try training on more examples

1.1 Results

Model	Train	Validation	Test
Mel-filterbanks + LR			
MFCC + LR			
MFCC + MDP			
MFCC + CNN			
MFCC + Normalization + CNN			
MFCC + Augmentation + CNN			

Table 1: Results for sentence classification using Logistic Regression on top of bag-of-words embeddings, with l2-regularization (regularization strength of 2 for simple average of word vectors, and of 90 for Idf-weighted average, values found using grid-search to minimize the cross-entropy loss on the dev set).

Add time considerations
You should find the best model by comparing validation accuracies. After you find your best model, finally test it on the test set and print the result. Comment on the results (best model, best features, classes that are the most difficult to recognize).

2 II. Classification of segmented voice commands

2.1 q1.

$S + D + I$ is positive by definition, and zero when the sequences are perfectly aligned. Therefore, we cannot have WER¡0.
However $S + D + I$ can be larger than N. For instance the hypothesis sequence can be of length $2N$ and contain no word of the reference sequences, in which case $S + D + I = 2N$ (shortest path to align the two sequences is composed of N substitutions and N insertions). Therefore, we can have WER¿100.

2.2 q2.

With such line

```
elif train_labels.count(label) < nb_ex_per_class:
            fs, waveform = wav.read(full_name)
            train_wavs.append(waveform)
            train_labels.append(label)
```

we ensured that the discriminator of speech commands has been trained with balanced dataset $P(W_i) = constant.$
Therefore, we can rewrite the acoustic model using Bayes rule as :

$$\begin{aligned} P(X_i|W_i) &= \frac{P(W_i|X_i)P(X_i)}{P(W_i)} \\ &= \frac{1}{|W|}P(W_i|X_i)P(X_i) \\ &\propto P(W_i|X_i)P(X_i) \\ &\propto P_{\text{discriminator single word}}(W_i|X_i) \end{aligned}$$

2.3 q3.

The greedy decoder assumes full independence between consecutive spoken words : we label each waveform of the sentence with the word label of highest probability for the chosen discriminator model, ie. the convolutional neural network of part I.

Word error rate between "go marvin one right stop" and " go marvin one left stop" is 0.2 as the shortest path to go from one sentence to the other using the three operations(insertions, substitutions, deletions) is 1 : we substitute "right" by "left". Hence WER = 1/5 = 0.2 = 20

2.4 q4.

	Train (classification error, cross-entropy loss)	Dev (classification error, cross-entropy loss)
Average of word vectors	(52.0%,1.20)	(57.3%, 1.29)
Idf-Weighted Average	(53.0%, 1.22)	(57.3%, 1.32)

Table 2: Results for sentence classification using Logistic Regression on top of bag-of-words embeddings, with l2-regularization (regularization strength of 2 for simple average of word vectors, and of 90 for Idf-weighted average, values found using grid-search to minimize the cross-entropy loss on the dev set).

Using Idf-weighted average did not really improve the results, maybe would we see an improvement for larger set of sentences (better estimations of words scarcities). However, performing a Multi-Class SVM on the top of the embeddings instead of a Logistic Regression improved a bit the classification score on the train and on the dev set.

	Train (classification error)	Dev (classification error)
Idf-Weighted Average	45.5%	56.5%

Table 3: Results for sentence classification using Multi-Class SVM on top of bag-of-words embeddings (penalty for outlier/misclassified point equal to 6 minimized the dev error in our grid search)

3 Deep Learning models for classification

Question 1

I used a categorical cross-entropy loss, which is adequate for the task (network outputs vectors encoding probabilities for each sentence to belong to each class, and labels to predict being hot-encoded).

Let us denote N the number of sentences, x_i the hot encoding of the i-th sentence, y_i its label, and $\hat{p_i}$ the network output prediction.
With these notations, the loss expresses as follows :

$$\mathcal{L}(\mathbf{x_i}, \mathbf{y_i}) = -\sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{k=1}}^{\mathbf{5}} \mathbf{1}_{\mathbf{y_i=k}} \log(\hat{\mathbf{p}}_{\mathbf{i}})$$