COM4511 Speech Technology - Practical Exercise - Voice Activity Detection

Anton Ragni

1 Background

The aim of this task is to build and investigate the simplest form of a voice activity detection (VAD) system allowing to detect the presence of voice in audio. VAD can be thought of as a binary task of classifying audio into speech and non-speech segments. What makes VAD complicated is that audio segments are not identically and independently distributed but instead have a temporal structure. Given two consecutive voice segments the probability that the third segment contains voice is high. As the length of voice-only segments increases the probability that the next segment contains voice decreases.

A range of signal processing, statistical and machine learning approaches have been used for VAD. A simple machine learning approach would involve a feed-forward network

$$\begin{aligned} \boldsymbol{h}_t^{(1)} &= \boldsymbol{\phi}^{(1)}(\boldsymbol{W}^{(1)}\boldsymbol{x}_t + \boldsymbol{b}^{(1)}) && (1) \\ &\vdots && (2) \\ \boldsymbol{h}_t^{(L)} &= \boldsymbol{\phi}^{(L)}(\boldsymbol{W}^{(L)}\boldsymbol{h}_t^{(L-1)} + \boldsymbol{b}^{(L)}) && (3) \end{aligned}$$

where L is the number of layers, $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are layer l weight matrix and bias vector, $\boldsymbol{\phi}^{(l)}$ are layer l element-wise non-linearities (e.g. sigmoid, $\tanh$), $\boldsymbol{x}_t$ is a vector of features to be classified either as voice or not. There are several options how the neural network above can make a prediction. One option is to use sigmoid non-linearity

$$h_t^{(L)} = \frac{1}{1 + \exp(-(\boldsymbol{w}^{(L)\mathsf{T}}\boldsymbol{h}_t^{(L-1)} + b^{(L-1)}))} \tag{4}$$

which yields a value between zero and one. Another option is to use soft-max non-linearity

$$P(\boldsymbol{x}_t \text{ is speech}) = \frac{\exp(\boldsymbol{w}_1^{(L)^{\mathsf{T}}}\boldsymbol{h}_t^{(L-1)} + b_1^{(L-1)})}{\exp(\boldsymbol{w}_1^{(L)^{\mathsf{T}}}\boldsymbol{h}_t^{(L-1)} + b_1^{(L-1)}) + \exp(\boldsymbol{w}_2^{(L)^{\mathsf{T}}}\boldsymbol{h}_t^{(L-1)} + b_2^{(L-1)})} \tag{5}$$

The neural network parameters then can be optimised by either minimising the mean squared error or maximising cross entropy. Feed-forward neural network may not be the most optimal type of neural networks to use for sequential data. Consider, for example, a recurrent neural network

$$\boldsymbol{h}_t = \boldsymbol{\phi}(\boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{V}\boldsymbol{x}_t + \boldsymbol{b}) \tag{6}$$

Unlike feed-forward networks, the recurrent model enables long-term dependencies in $\boldsymbol{X}_{1:T} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ to be modelled. The increased modelling power comes at the expense of more complicated training and inference. In particular, the recursion above is known to exhibit troublesome training behaviour. Many alternative variants have been proposed including long short-term memory (LSTM) and gated recurrent unit (GRU) to address those issues.

There are multiple ways how VAD approaches can be assessed. These depend on the target application such as speech transmission and coding, keyword spotting, automatic speech recognition. In the simplest case VAD performance can be assessed based on accuracy of predicted labels. Other simple approaches include equal error rate (EER) on detection error trade-off (DET) curve. The DET curve is a plot of probability of miss against probability of false alarm. A point where these probabilities are the same is called EER.

2 Objective

Given a collection of parameterised audio files, where each parameter vector is labelled with either voice or silence, build a VAD system based on feed-forward neural network architecture and measure its performance using EER criterion. In order to relax strict independence assumptions consider appending a range of previous and following feature vectors such that the modified form of feature vector at time t is given by $\hat{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{x}_{t-N}^{\mathsf{T}} & \dots & \boldsymbol{x}_{t-1}^{\mathsf{T}} & \boldsymbol{x}_{t}^{\mathsf{T}} & \boldsymbol{x}_{t+1}^{\mathsf{T}} & \dots & \boldsymbol{x}_{t+N}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. As a complement, or alternative to, feed-forward neural network build a recurrent form of neural networks. Note that neural network toolkits, such as TensorFlow and PyTorch, typically provide a range of tutorials on how feed-forward and recurrent neural networks can be created, trained and evaluated in binary classification tasks. To save time and avoid commonly occurring mistakes it is advantageous to base implementation as much as possible on such publicly available code. It is also important to note that it is not necessary to use all of the available training data. Given constraints imposed by your workstation (memory size, number of CPUs, number of GPUs) and relatively large amounts of training data, it is important to make a rational decision about training data selection given available compute resources. The same however does not apply to the test data. To ensure interpretability of your results any metric must be evaluated on all available test data.

Your report is expected to include:

1. Discuss your implementation and provide argumentation for your design decisions including model topology, objective function, optimisation, termination criterion.
2. Measure performance on development and test data using EER criterion.
3. Discuss whether EER is a suitable criterion and if not comment on performance using alternative metrics or points on DET curve.
4. Compare and comment on possible performance difference on training, validation and test sets.

3 Marking Scheme

Two critical elements are assessed: training (65 marks) and evaluation (35 marks). *Note:* Even if you cannot complete this task as a whole you can certainly provide a description of what you were planning to accomplish.

1. **Training.** Write a code (**not advised**) or adopt (**advised**) an existing implementation for training either feed-forward or recurrent neural network models to classify sequences of speech parameter vectors into either voice or silence classes (binary classification). For example, if Python language is used then the execution of your code may look like

   ```
   python train.py audio/train labels/train model
   ```

   where `audio/train` is a directory containing training speech parameter vector sequences, `labels/train` is a directory containing training label sequences, `model` is an output directory for storing the neural network model. Other arguments may be also included. For instance, if validation set is used to track model generalisation, the corresponding audio and label files must also be provided.

   Provide a description of the following key elements.

   - **Models (5 marks).** Describe each model examined by you in this task. This can include a feed-forward model, a recurrent model or both. For each model specify the exact architecture used including the number of layers, types of layers, dimensions of layer parameters, types of non-linearities used.
   - **Objective function (5 marks).** Describe objective function used to estimate model parameters. Write it down using standard mathematical notation.
   - **Optimisation (5 marks).** Describe optimisation of model parameters. Include a discussion of the following aspects (a) model parameter initialisation (b) stochastic optimisation, (c) first-order optimisation, (d) second-order optimisation, (e) regularisation, (f) generalisation approaches (e.g. validation). If some of these aspect were not used please comment why.
   - **Termination criterion (5 marks).** Specify the criterion used to decide when to stop optimisation. Specify one alternative approach. Specify at least one disadvantage of each termination criterion.
   - **Efficient data use (5 marks).** Describe the rationale for using either all or a part of available training data. If all training data was used then describe what steps did you take to ensure that you can complete training in a reasonable amount of time. If a part of training data was used then describe what criterion was used to decide how much and what data to use. If both stochastic optimisation and a part of training data were used then describe the difference between drawing equal numbers of minibatches from either the whole of training data or only a part of it.

- **Performance (40 marks).** Report the objective function values on training, validation and testing data. Report the number of data points these values were computed over. Plot the change in objective function during training on both training and validation data. Comment on any similarity or dissimilarity between these curves.

2. **Evaluation.** Write a code (**not advised**) or adopt (**advised**) an existing implementation for evaluating binary classification performance of either feed-forward or recurrent neural network models in binary classification of speech parameter sequences. For example, if Python language is used then the execution of your code may look like

   ```
   python evaluate.py model audio/test labels/test
   ```

   where `model` is a directory containing the model to be evaluated, `audio/test` is a directory containing testing speech parameter vector sequences, `labels/test` is a directory containing testing label sequences.

   - **Accuracy (15 marks).** Compute and report on accuracy of your VAD system on both validation and testing data. Provide a mathematical description of accuracy computed. Compute accuracy of a random classifier (assuming each point is always classified as one of the available classes). Report the number of data points used to compute accuracy. Discuss one issue using accuracy in this task.
   - **DET (15 marks).** Plot DET curve on both validation and testing data. Compare both curves. Discuss one issue of using DET curves in this task.
   - **EER (5 marks).** Estimate EERs on both validation and testing data from DET curves produced in the previous step. Discuss one issue of using EER in this task.

4 Hand-in procedure

All outcomes, however complete, are to be submitted jointly in a form of a package file (`zip/tar/gzip`) that includes directories for each task which contain the associated required files. Submission will be performed via MOLE.

5 Resources

Three sets of resources are provided for this task: training, validation and testing. Each set comprises

- audio files in a parameterised form; each audio file is a matrix $\boldsymbol{X}_{T\times D}$ where T is a length and D is the number of features.
- corresponding label files; each label file is a vector $\boldsymbol{y}_{T\times 1}$ such that each y_t is either 0 (silence) or 1 (voice).

The names of training, validation and testing files are as follows:

- training:

```
NIS-2E07004-NI00x-R7  VIT-3E07006-VI00x-R7
NIS-2E07004-NI01x-R7  VIT-3E07006-VI01x-R7
NIS-2E07004-NI02x-R7  VIT-3E07006-VI02x-R7
NIS-2E07004-NI03x-R7  VIT-3E07006-VI03x-R7
NIS-2E07005-NI00x-R7  VIT-3E07006-VI04x-R7
NIS-2E07005-NI01x-R7  VIT-3E07007-VI00x-R7
NIS-2E07005-NI02x-R7  VIT-3E07007-VI01x-R7
NIS-2E07005-NI03x-R7  VIT-3E07007-VI02x-R7
NIS-2E07005-NI04x-R7  VIT-3E07007-VI03x-R7
NIS-2E07005-NI05x-R7
```

- validation:

```
EDI-1E07002-ED00x-R7
EDI-1E07002-ED01x-R7
```

- testing:

```
CMU-0E07000-CM00x-R7
CMU-0E07000-CM01x-R7
```

Each audio and label file is stored as NumPy `ndarray` structure. The minimum code needed to load one of those files is:

```
#import numpy

with open(FILENAME, 'rb') as f:
        A=numpy.load(f)
```

where `FILENAME` is the name of audio or label file and `A` is NumPy `ndarray` object.