67355 Introduction to Speech Processing

Exercise 2

In this exercise you will implement a Music-Genre linear classifier that relies on manually crafted features.
The exercise should be done **in pairs** and is to be submitted via moodle by the deadline appearing under the submission box.
See submission guidelines for further instructions

1 Assignment overview

1.1 Problem description

In this exercise you are required to implement a LINEAR classifier (logistic regression) to perform music genre classification.

- Your classifier should rely on MANUALLY CRAFTED features using various features seen in class or other ideas you may think of.
- You should implement a LINEAR classifier, i.e. have a single learned weights tensor and bias vector, and optimize it using SGD (you are not allowed to use external packages and autograd!). Your code should be compatible with the specified API below.
- You are encouraged to implement a logistic regression classifier (as seen in class).
- You are not allowed to gather and use additional data, other than the data provided, to train your model. You are allowed to use any data you would like to test your model. You are given a basic test set derived from the same data source (disjointed with the train segment).

1.2 Files overview

In Ex2.zip you will find three directories and a .py file

- jsons directory contains train/test split .json files to be used as your train/test partitions. You should read your train and test sets from the .json files, as you should NOT re-submit the data files.
  We will be using .json files to test your training pipeline during our evaluations, please make sure you read out your data files from these .json files!
  You could feel free to use different .json files, e.g. if you wish to perform cross validation, just make sure your training pipeline works with the provided train.json and test.json
- parsed_data_mp3 directory contains the actual .mp3 files to be used throughout the exercise.
- model_files directory is an empty dir, to which you will store your model weights or other files needed to load a pretrained model.
- genre_classifier .py defines a most basic API to be used throughout the ex that will be covered in the next subsection. You may define the train loop/ train manager as you see fit as long as you compile with the given API.
- dummy_test.py a dummy test you can use for sanity checks and develop further.

Important note: We will be using the API functions to evaluate your performance, failing to use these function will break the tests and grand you a grade of 0 for the auto-tests. Please make sure that your code matches the specified API!

1.3 API Calls

We will define our call API here, there are more functions to be implemented - see code documentation for further instructions.
NOTE: We will not be performing a function call on any method outside this API yet, the code includes additional functions we encourage you to use and implement.

1.3.1 genre_classifier .Genre(Enum)

An Enum class defining a label to integer mapping. You are not required to use it, but you are required to use the map it defines.

1.3.2 genre_classifier .TrainingParameters

A dataclass defining training configuration. We will be using this dataclass to initialize your training process. Please do not remove existing entries in this class. You may change the default values, and add new entries as you see fit - just make sure that all entries have default values that does NOT include any absolute path.

1.3.3 genre_classifier .ClassifierHandler

A factory-like class that has two static functions to be implemented:

1. train_new_model: should initialize a complete training (loading data, init model, start training/fitting) and to save weights/other to model_files directory. This function should recieve a TrainingParameters dataclass object and perform training accordingly, see code documentation for further details.

2. get_pretrained_model: should initialize a genre_classifier .MusicClassifier object, load its pre-trained weights and return the initialized object.

1.3.4 genre_classifier .MusicClassifier

Your main model class.

1. classify : this function performs inference, i.e. returns a integer label corresponding to a given waveform, see class Genre(Enum) inside the code for a label to integer mapping. This function should support batch inference, i.e. classify a torch.Tensor comprising of stacked waveforms of shape $[B, 1, T]$ denoting B mono (single channel) waveforms of length T. Assume the input is always of this form (shape $[B, 1, T]$) where $B = 1$ in the single example case.

2. get_weights_and_biases: This function returns the weights and biases associated with this model object, should return a tuple: (weights, biases)

2 Grading

We will be applying automatic and manual evaluations for your submitted assignment.
For the automatic part, we will be evaluating your classifier accuracy over the held-out test set, and over an additional "private" test-set, and will be comparing your performance to a simple baseline of our own.
Please make sure you write clear, documented code with reasonably-sized functions and meaningful variable and function names so it would be simple for us to review your submitted work.

3 Submission Guidelines

- All used code pieces should be submitted, alonside a README.txt file with a single line containing your IDs separated by comma.
- Make sure to include your model weights / other files needed to restore your pretrained model as we do not plan on re-training your model.
- Make sure that you comply with the held-out API.
- Please submit a single zip/tar file containing all relevant files.