Practical 2021: Classifying Sounds

Due 7:59 PM on Friday, April 9th, 2021

This assignment must be completed in groups of 2 to 3 students. You can seek partners via Ed, and course staff will release a team-finding spreadsheet. Section 2 of this document describes the classification problem, and Section 1 includes submission logistics and FAQs. In this same directory, you will find a file called `practical-template.tex`. Use it as a LATEX template for your writeup, and be sure to read it for grading details.

1 Logistics

1.1 Implementation Details

You can download the train and test datasets here, and clone a starter code repository here. You will be required to submit all of your implementation code to Gradescope.

This assignment can be completed on either your own computer, or on Google Colab. We recommend that you complete the assignment using Google Colab.

- If you choose to complete the assignment on your own computer, you can begin by working from the the `starter_code.ipynb` file, which contains starter code to load the dataset. You can watch a short video walkthrough of downloading the data and running the starter code notebook here.

- If you choose to complete the assignment on Google Colab, begin by copying this Colab notebook, which contains starter code to load the dataset. You can watch a short video walkthrough of the Colab notebook here.

1.2 Submission Details

The main deliverable of this practical is a 3-4 page typewritten document in PDF format to Gradescope. The document must follow the `practical-template.tex` file in this directory *and* follow all instructions outlined in Section 3. All relevant text—including your discussions of why you tried various things and why the results came out the way they did—must be included in the maximum of 4 pages. If you need more space than 4 pages for tables, plots, or figures, that is okay.

1.3 Grading

Our grading focus will be on your ability to clearly and concisely describe what you did, present the results, and most importantly *discuss* how and why different choices affected

performance. You will *not* be graded on your model performance.

Submissions that successfully and thoughtfully complete the "Your Task" sections of Parts A-C reporting results as described in Section 3.1 will receive up to 18 points (considered full credit). Submissions that also thoughtfully explore extensions in Part D will receive up to 2 additional points, up to a total of 20 points.

See `practical-template.tex` for our desired submission format and more tips for what a full-credit submission looks like. All team members will receive the same practical grade.

1.4 Google Cloud

The Google Cloud Platform (GCP) offers a suite of cloud computing services. **You do not need to set up or use GCP to complete this practical.** Some students prefer to run code remotely on the Cloud instead of locally on their own computers for better job management. For more resources on getting started with GCP, see the Practical Addendum on Ed.

2 Problem Background

For this practical, you will classify sounds recorded using microphones around New York City into 10 classes. In making your predictions, you will primarily have at your disposal a series of amplitudes sampled for each sound. The classes of sounds under consideration in this practical are: `air_conditioner`, `car_horn`, `children_playing`, `dog_bark`, `drilling`, `engine_idling`, `gun_shot`, `jackhammer`, `siren`, and `street_music`.

The data source is the UrbanSound8k dataset, which contains labeled sound excerpts. These excerpts were created by splicing source audio recordings into shorter two-second non-overlapping clips. While multiple excerpts from the same source may be present within the train or test set, no source will have excerpts in both the train and test set. The dataset was created by NYU's Sounds of New York City (SONYC) initiative. For more information on how the dataset was created, you can read Sections 1 and 3 of this paper.

2.1 Data Files

There are 8 files of interest, which can be downloaded by clicking the download icon next to each file name on this Google Cloud bucket:

- `Xtrain_amp.npy`, `ytrain_amp.npy` – These files contain information about the 5779 sounds in the training set. The 44,100 columns of `Xtrain_amp.npy` are the

sampled amplitudes (2 seconds at 22050 samples/second). Integer labels `ytrain_amp.npy` denote the class of each sound.

- `Xtest_amp.npy, ytest_ampy.npy` – These files contain information about the 1546 sounds in the test set. The 44,100 columns of `Xtest_amp.npy` are the sampled amplitudes. Integer labels `ytest_amp.npy` denote the class of each sound. **Do NOT use these data for any training or parameter validation!**

- `Xtrain_mel.npy, ytrain_mel.npy` – These files contain the Mel spectrogram representation of the 5779 sounds in the training set. Each sound has a 2D spectrogram of shape (128×87). In short, the original amplitudes are partitioned into 87 time windows, and there are 128 audio-related features computed for each such window. Integer labels `ytrain_mel.npy` denote the class of each sound.

- `Xtest_mel.npy, ytest_mel.npy` – These files contain the Mel spectrogram representation of the 1546 sounds in the test set. Each sound has a 2D spectrogram of shape (128×87). Integer labels `ytest_mel.npy` denote the class of each sound. **Do NOT use these data for any training or parameter validation!**

2.2   Class Distribution

The distribution of sound classes in the training data is approximately as follows. It may be worthwhile to keep in mind that some classes are very infrequent.

0	air_conditioner	12.6%
1	car_horn	3.54%
2	children_playing	12.53%
3	dog_bark	9.42%
4	drilling	10.93%
5	engine_idling	12.98%
6	gun_shot	1.49%
7	jackhammer	11.85%
8	siren	12.03%
9	street_music	12.61%

3   Your Task and Deliverables

Below in Parts A-C we list three concrete explorations to complete in the context of this task. Through this process of guided exploration, you will be expected to think critically about how you execute and iterate your approach and describe your solution.

You are welcome to use whatever libraries, online resources, tools, and implementations that help you get the job done. Note, however, that you will be expected to *understand*

everything you do, even if you do not implement the low-level code yourself. It is your responsibility to make it clear in your writeup that you did not simply download and run code.

3.1 Evaluation Metrics

In this practical, we would like you to primarily focus on optimizing for accuracy:

$$\text{Classification Accuracy} = \frac{\text{Number Correctly Classified Examples}}{\text{Total Number of Examples}}.$$

In Parts A - D, you will be asked to train several different classification models. **For each model you train, calculate the model's train and test set overall, and per-class classification accuracy** and include these results in your write-up.

Even better, try to run several seeds for each model in a given experiment and report mean and standard deviation of the evaluation metric. This helps you see which variability is due to chance vs. which is attributable to your experimental choices.

3.2 Part A: Feature Engineering

We have provided you with two sets of data files: `amp`, which contain raw amplitude data for each of the sound recordings, and `mel`, which contain derived Mel spectrogram features for each of the sound recordings. Here we first provide more context for each of these feature representations and then provide instructions for your deliverables.

In this practical, our data source is recorded audio clips. Audio data is sampled at discrete time steps. These samples are taken at a specified rate, named the sampling frequency given in Hz. The raw data in the `Xtrain_amp.npy` and `Xtest_amp.npy` files contain the sampled signal amplitudes at each timestep. For example, the Figure 1 shows the signal amplitude versus timestep for an example file with label `children_playing`. A large body of machine learning and signal processing research has explored the predictive value of various audio feature engineering techniques that take as input raw amplitude data. Here we provide some basic intuition behind spectrogram features, which can be implemented using python package `librosa`[1].

A spectrogram is a 2D visual representation of the spectrum of frequencies of a signal as it varies with time[2]. There are several different kinds of spectrograms, most of which are generated by applying a Fourier transform to the sampled amplitudes[3]. Our spectrograms partition the ampltiude arrays into 87 sub-sequences and compute the presence of

---

[1]https://librosa.org/doc/latest/index.html

[2]https://en.wikipedia.org/wiki/Spectrogram

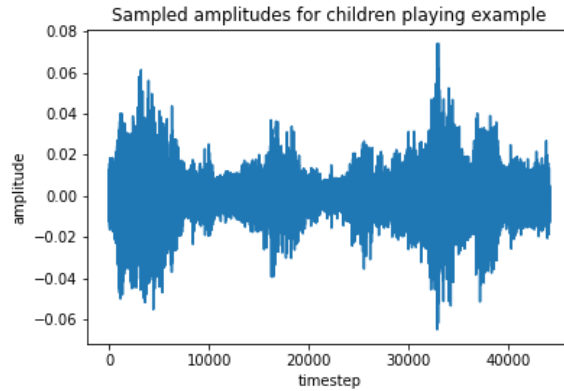[3]See the following optional educational resources about spectrograms if you're interested: [1, 2].

Figure 1: Plot of sound amplitudes for example file with label `children_playing`.

128 different frequencies in each of these sub-sequences.

In the practical repository, you will find the file `generate_spectrograms.ipynb`. This notebook contains starter code that was used to transform raw sampled amplitdues to generate a Mel spectrogram, which is a type of short-time fourier transform with frequencies on the Mel (log) scale. To use Mel spectrogram features, you do not need to re-run this code file, and can instead just load the saved `Xtrain_mel.npy` and `Xtest_mel.npy` files.

Your task: Train the following two models.

1. Perform PCA on the raw amplitude features (`Xtrain_amp`, `Xtest_amp`). Train a logistic regression model on the 500 most significant PCA components.

2. Perform PCA on the Mel spectrogram features (`Xtrain_mel`, `Xtest_mel`). Train a logistic regression model on the 500 most significant PCA components.

Discuss which feature representation resulted in higher model performance, and why you hypothesize this feature representation performed better than the other. Also discuss why we might have asked you to perform PCA first and the impact of that choice.

3.3   Part B: Model Classes

Now, you will experiment with more expressive nonlinear model classes to maximize accuracy on the audio classification task. Examples of nonlinear models include random forests, KNN, and neural networks.

Your task: Train at least one nonlinear model on a feature representation of your choice. Compare your results to the logistic regression models in Part A and discuss what your results imply about the task.

3.4 Part C: Hyperparameter Tuning and Validation

In this section, you will explore hyperparameter tuning. Model hyperparameters such as network architecture or random forest maximum tree depth determine the expressivity of the model class. Training hyperparameters such as learning rate, weight decay, or regularization coefficients influence optimization and can encourage desirable properties (such as sparsity) in the final learned models.

Popular hyperparameter tuning techniques include random search, where you train a set of models with hyperparameters chosen uniformly at random from a set of possible values, [4] and grid search, where all possible parameter values are considered exhaustively [5].

Your task: Perform a hyperparameter search to maximize predictive accuracy for two model classes of your choice. You can choose which hyperparameters you search over (feel free to search over multiple simultaneously if you'd like!), but you must search over at least 5 possible values for at least 1 hyperparameter. Explore the changes in performance as you choose different hyperparameter values. In your writeup, discuss your validation strategy and your conclusions.

Note: Choose how to present your results of your hyperparameter search in a way that best reflects how to communicate your conclusions.

3.5 Bonus, Part D: Explore some more!

This section is not required to receive a full-credit grade of 18 points on the practical. See Section 1.3 for more details about practical grading.

Your task: Try any combination of the suggestions below, or come up with your own ideas to improve model training or expand your evaluation! In your write-up, discuss what you tried, what happened, and your conclusions from this exploration.

Some ideas:

- **Alternative feature representations:** Try out other popular audio feature representations like Mel-frequency cepstrum coefficients (MFCCs) [6] or others listed in librosa's documentation [7]. You could also explore alternative dimensionality reduction techniques to use instead of PCA.

---

[4]https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf
[5]https://scikit-learn.org/stable/modules/grid_search.html
[6]https://librosa.org/doc/main/generated/librosa.feature.mfcc.html
[7]https://librosa.org/doc/main/feature.html

- **Use a CNN on the spectrogram data**: CNN architectures have been shown to achieve high classification accuracy when trained on audio spectrogram data [8]!

- **Explore other evaluation metrics**: You may also consider optimizing for or reporting other metrics e.g. precision or recall across various classes.

- **Address class imbalance**: Some classes are very infrequent in the training dataset. Popular techniques to address class imbalance are using a class-weighted loss function [9] or by upsampling infrequent classes during training [10].

- **Use a generative classifier:** You could build a model for the class-conditional distribution associated with each type of sound and compute the posterior probability for prediction.

- **Use a support vector machine:** If you prefer your objectives convex.

- **Go totally Bayesian:** Worried that you're not accounting for uncertainty? You could take a fully Bayesian approach to classification and marginalize out your uncertainty in a generative or discriminative model.

3.6 Other FAQs

What language should I code in? You can code in whatever language you find most productive. We will provide some limited sample code in Python. You should not view the provided Python code as a required framework, but as hopefully-helpful examples.

Can I use {scikit-learn | pylearn | torch | shogun | other ML library}? You can use these tools, but not blindly. You are expected to show a deep understanding of the methods we study in the course, and your writeup will be where you demonstrate this.

What do I submit? You will submit both your write-up on Gradescope and all of your practical code to a supplemental assignment on Gradescope.

Can I have an extension? Yes, your writeup can be turned in late according to standard homework late day policy. You can use at most two late days on the practical.

---

[8]https://research.google/pubs/pub45611/
[9]https://www.kdnuggets.com/2018/12/handling-imbalanced-datasets-deep-learning.html
[10]https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28