MLPR Assignment 1 – Predicting Audio

Due: 12 noon Monday 16 October, 2023

This assignment does not form any part of your final grade. Most of your time spent on this class should be spent on independent study, which won't have marks attached either.

You are encouraged to work in pairs on this assignment. You can let us know if you would like to work by yourself, in a pair of your choice, or would like to be assigned to a random pair on this MS Form. We will assign the first set of pairs for forms submitted by 2pm Tuesday 4 Oct. You'll be able to choose again whether to work alone or with a partner for assignment 2, and can work with the same partner or a different one.

You should be able to do each assignment by yourself. However, we encourage you to work in pairs if you wish to do so. Pair coding is often useful, to stop you getting stuck for long periods, and to pick up random tips. If in a pair, you *must* work together though, rather than attempting to divide up the work, and doing it separately.

Because this assignment isn't assessed, you may discuss it freely outside of your pair, and ask clarifying questions on the class forum. However, please practice moving towards doing assessed work. If something is ambiguous, can you choose a reasonable path, write down what you've done, and keep going? If so, do that! If you do need to ask a question, put effort in to describe what you've tried so far. Put *spoiler* on the first line of any forum post that includes part of an answer.

Make sure to submit your answers as soon as you write them. Late assignments are likely to get little or no personalized feedback. Unlike the in-note questions, you can edit answers again after submitting (until the deadline).

Background: Raw audio data is represented as a sequence of amplitudes. Lossless audio compression systems (like `flac`) use a model to predict each amplitude in turn, given the sequence so far. The residuals of the predictions are compressed and stored, from which the audio file can be reconstructed. We'll do some initial exploratory analysis of an audio file. Really we're just using the file as a convenient source of a lot of data points for an exercise.

Download the data here (65 MB).

Programming: You must use Python+NumPy+Matplotlib, and may not use any other libraries (e.g., not SciPy, pandas, or sklearn), or code written by other people. When we suggest functions you could use (e.g., `np.load`), you can get quick help at the Python prompt with the help function, e.g., `help(np.load)`

1. **Getting started:** Load a *long* array into Python:
   ```
   amp_data = np.load('amp_data.npz')['amp_data']
   ```

   a) Plot a line graph showing the sequence in `amp_data`, and a histogram of the amplitudes in this sequence. Include the code for your plots, with one to three sentences about anything you notice that might be important for modelling these data.

   [Visit question on website]

   We will create a dataset that will be convenient for trying different regression models, without some of the complications of responsible time series modelling. Take the vector in `amp_data` and wrap it into a $R \times 21$ matrix, where each of the R rows contains 21 amplitudes that were adjacent in the original sequence. As the vector's length isn't a multiple of 21, you'll have to discard some elements before reshaping the data.

   It should be clear from your plot that the distribution over amplitudes changes over time. Randomly shuffle the rows of the matrix. Then split the data into parts for

training (70%), validation (15%), and testing (15%). Each dataset should take the first $D = 20$ columns as a matrix of inputs `X`, and take the final column to create a vector of targets `y`. Name the resulting six arrays: `X_shuf_train`, `y_shuf_train`, `X_shuf_val`, `y_shuf_val`, `X_shuf_test` and `y_shuf_test`. The shuffling means that our training, validation and testing datasets all come from the same distribution. Creating this ideal setting can be useful when first learning about some different methods. Although we should remember our models might not generalize well to new files with different distributions.

Useful NumPy functions: `np.reshape`, `np.random.permutation`

In future questions you will need repeated access to your shuffled datasets. You could set the "random seed" for the shuffling operation, so that your code creates the same datasets each time. You may also wish to save temporary copies of the shuffled datasets, but save the random seed regardless.[1]

Be careful: if code for pre-processing data is incorrect, then all further experiments will be broken. Do some checking to ensure your code does what you think it does.

b) Include your code for creating the six arrays above from the original `amp_data` array. Your answers to future questions should assume these arrays exist, rather than relisting the code from this part.

[Visit question on website]

2. **Curve fitting on a snippet of audio:**

Given just one row of inputs, we could fit a curve of amplitude against time through the 20 points, and extrapolate it one step into the future.

Plot the points in one row of your `X_shuf_train` data against the numbers $t = \frac{0}{20}, \frac{1}{20}, \frac{2}{20}, \ldots, \frac{19}{20}$, representing times. We can fit this sequence with various linear regression models, and extrapolate them to predict the 21st time step at time $\frac{20}{20} = 1$. Indicate the point you're predicting from `y_shuf_train` on the plot at $t = 1$.

First fit and plot a straight line to the 20 training points. Then fit a quartic polynomial, by expanding each time t to a feature vector $\boldsymbol{\phi}(t) = [1 \;\; t \;\; t^2 \;\; t^3 \;\; t^4]^\top$, and fitting a linear model by least squares. Plot both fits between $t = 0$ and $t = 1$.

a) Include the code for a plot that shows 20 training points, a test point, a straight line fit, and a quartic fit.

[Visit question on website]

b) Explain why the linear fit might be better if we use only the most recent two points, at times $t = \frac{18}{20}$ and $t = \frac{19}{20}$, rather than all 20 points. Also explain why the quartic fit might be better with a longer context than is best for the straight line model.

[Visit question on website]

c) Based on your manual visualization of this snippet of audio data, and maybe one or two other rows of your training dataset, roughly what order of polynomial and context length do you guess might be best for prediction and why?

[Visit question on website]

3. **Choosing a polynomial predictor based on performance:**

When we use K basis functions (polynomial or otherwise) with C datapoints we

1. When doing research on larger datasets you won't want to commit them to version control, to have to distribute copies to anyone following your work, or to pay to back up redundant processed copies.

construct a $C \times K$ matrix of input features Φ. If the least squares weights are unique, there is a closed-form solution:

$$\mathbf{w} = (\Phi^\top \Phi)^{-1}(\Phi^\top \mathbf{x}), \tag{1}$$

where $\mathbf{x}$ are the previously observed amplitudes, as stored in a row of `X_shuf_train`. We usually see $\mathbf{y}$ in the equation above but, in this context, the amplitudes in $\mathbf{x}$ are on the "y-axis" plotted against time t, and the Φ matrix contains transformed times. We have used C for the length of the context that we are predicting from (e.g., 20 time-steps), because later we will want to use N for the number of sequences in the training set.

a) The model's prediction for the next time step is:

$$f(t=1) = \mathbf{w}^\top \boldsymbol{\phi}(t=1). \tag{2}$$

For the least squares weights $\mathbf{w}$ given above, this prediction can be written as a linear combination of the previous amplitudes, $f(t=1) = \mathbf{v}^\top \mathbf{x}$. Identify an expression for $\mathbf{v}$.

Hint: No complicated solving techniques are required here. Identify a collection of symbols in the first definition of $f(t=1)$ such that if we rename the collection as $\mathbf{v}$, the prediction can be rewritten as $f(t=1) = \mathbf{v}^\top \mathbf{x}$.

[Visit question on website]

b) i) Provide code for a function `Phi(C, K)` that constructs a $C \times K$ design matrix Φ, representing the C most recent time steps before the time we wish to predict ($t=1$). That is, the input times are $t^{(C)} = \frac{19}{20}, t^{(C-1)} = \frac{18}{20}, \ldots$ The row for time t should have K features $\boldsymbol{\phi}^\top = [1 \;\; t \;\; t^2 \;\; \ldots \;\; t^{K-1}]$.

[Visit question on website]

ii) Provide code for a function `make_vv(C, K)` that returns the vector $\mathbf{v}$ that you derived in part a) for a model with K features and a context of C previous amplitudes, using the function from the previous part.

[Visit question on website]

iii) Include a short demonstration that using two vectors from `make_vv(C, K)`, for appropriate C and K, you can make the same predictions at time $t=1$ as the linear and quartic curves you fitted in Q2.

[Visit question on website]

c) The advantage of identifying the prediction as a linear combination, $\mathbf{v}^\top \mathbf{x}$, is that we can compute $\mathbf{v}$ once and rapidly make next-step predictions for N different short sequences of C amplitudes. We don't need to fit N separate models!

i) Report code that evaluates predictors for a range of context lengths C and numbers of basis functions K on your shuffled training set.

Your code should identify which setting of K and C gives the smallest square error on the training set. Report this setting of K and C.[2]

[Visit question on website]

ii) Use your selected system to report the mean square error on the training, validation, and test sets. Include your code and resulting numbers.

2. Yes, it's weird we don't need the validation set to make model choices here. It's because fitting $\mathbf{v}$ for a given (K, C) didn't look at the audio data at all!

[Visit question on website]

4. **Fitting linear predictors across many snippets:**

   It's possible we could do better by picking different basis functions. However, no matter which basis functions we pick, a linear model fitted by least squares will predict the next amplitude using a linear combination of the previous amplitudes.

   Given a large dataset, we can try to fit a good linear combination directly, without needing to specify basis functions. Using standard linear least squares fitting code, we can find the vector $\mathbf{v}$ that minimizes

   $$\sum_{n=1}^{N} \left(y^{(n)} - \mathbf{v}^\top \mathbf{x}^{(n)}\right)^2, \tag{3}$$

   on the training set. Again, $y^{(n)}$ is the nth amplitude to predict, based on a history of previous amplitudes in $\mathbf{x}^{(n)}$.

   We can choose to make the prediction based on a shorter history than all of the previous 20 amplitudes available. For $C = 1 \ldots 20$, fit vectors $\mathbf{v}^{(C)}$, each of length C, that can combine C previous amplitudes to predict the next amplitude.

   a) What context length C has the lowest mean square error on the training set? Explain why in 1–3 sentences. What context length has lowest mean square error on the validation set? Include the code you used to answer this part.

      [Visit question on website]

   b) Take the predictor with the best validation error and compare it to the best polynomial model from Q3 on the test set. Include your code, and the two test set numbers. Which approach is better?

      [Visit question on website]

   c) Plot a histogram of the residuals on the validation data for your best model, and compare it to the histogram of amplitudes you plotted in Q1. Include your code, and a one or two sentence comment on what you observe.

      [Visit question on website]

5. **What next?** Think about how you might improve the predictions of the best model considered so far. Propose an incremental improvement without writing a lot more code or using machine learning libraries[3].

   You could answer this question by formulating a few sentences to half a page with a maximum word count of 300. It is much better to give enough detail on one idea, such that another student could understand the motivation and implement it, rather than to list multiple vague or incomplete ideas.

   Alternatively, you could make a specific proposal with Python code but again, you must use Python+NumPy+Matplotlib, and may not use any other libraries.

   [Visit question on website]

Things we haven't asked you to do.

If this assignment was easy for you, there's plenty more you could investigate if you're interested. Be careful not to evaluate too many more models on the test set. Perform the bulk of any further investigations on the training and validation data.

As in Q5, can you think of any creative ways to fit a better model? Do any of your conclusions

3. For example, "implement WaveNet" is not an answer!

change if you use regularization? You could think about how to predict the magnitude of your prediction errors (which is useful in a compression system).

Predicting amplitudes sequentially through a non-shuffled audio file is a more complicated challenge. The distribution over amplitudes will be changing, so ideally we would continually adapt our model to recent data. If keen, you could look at non-shuffled data. The first step is probably to see how badly you generalize on data far into the future, compared to audio shortly after a sequence used for training.