MLPR Assignment 1 – Predicting Audio

Due: 4pm Thursday 19 October, 2017

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. I have emailed the class list about how you should get and register your assignment partner. If you can't get a pair, working in a three is also acceptable. Just make sure all three of you are really actively involved, and I'll expect more for the final part. You may also work by yourself, but I don't recommend it.

You may also discuss this assignment freely outside of your pair, and ask clarifying questions on the class forum. However, please don't just ask for the answers, which won't help you. Put effort in to describe what you've done so far, and where you are stuck. Put *spoiler* on the first line of any forum post that includes part of an answer.

You should not write a lengthy report. Please only provide the short explanations, plots, and code requested, with the question part numbers (Q1a, Q1b, Q2a, . . . , Q3bii, . . .) clearly labelled. Don't write more than a reasonable paragraph where it asks for a few sentences. Do not list all of your code in one block at the end and expect your marker to identify all of the relevant parts. Split it up as requested. If you want feedback on other things, place this material in an appendix, or ask questions on the forum. If you like, you can also include a complete code listing in an appendix, as long as this listing is in addition to the snippets requested. Your answers *must* be in a PDF file called `report.pdf`. Documents in other formats (e.g., `.docx`, `.ipynb`) will not be marked.

Submission instructions: Only one of you in each pair (or three) should submit your answers. Create a plain text file called `students.txt` containing your student numbers (format: `s1234567`, *not* exam numbers) separated by spaces. Then one of you should submit the document from a DICE machine using *exactly* the following `submit` command:

```
submit mlpr cw1 report.pdf students.txt
```

Late assignments are likely to get little or no personalized feedback.

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.

The data: The data file for this question can be obtained on a DICE machine at:
`/afs/inf.ed.ac.uk/group/teaching/mlprdata/audio/amp_data.mat`
This file isn't tiny, 65MB. If working on DICE, please don't copy it into your home directory: the class would create >10GB of copies in backed up space (not huge, but unnecessary). Instead follow the practice that would be necessary if the dataset were actually huge: load the file directly from the `/afs` location, or make a symbolic link into your working directory using "`ln -s`". If you want to create a derived version of this file, you should do so with a script, so your version can easily be reproduced. Your script should save its data file to a fast local location that isn't backed up, such as a directory inside `/disk/scratch` or `/tmp`. (Warning: the contents of `/tmp` is usually deleted when a machine reboots.)

Programming: Consult the background material on programming in the notes. You are expected to work in Matlab/Octave or Python+Numpy+Matplotlib. As well as online tutorials and web searches, you can use the built-in documentation: "`help functionname`" in Matlab/Octave or "`help(functionname)`" in Python tells you how to use a function. The "`lookfor`" or "`np.lookfor`" function can help you find which function to use.

Questions:

1. **Getting started:** Load the data into Matlab/Octave (using `load`) or Python (using `scipy.io.loadmat`). You will obtain a *long* array `amp_data`.

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

   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 $C \times 21$ matrix, where each row 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 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 Matlab/Octave functions: `reshape`, `randperm`
   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 the shuffled datasets to scratch space, although save the random seed regardless, in case the scratch space fails.

   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.

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 $t = 1$.

   The interval is now [0, 1] and so our x_grid should be a number of intervals in this space

   a) Include your plot and the code that made it. The plot should show 20 training points, a test point, a straight line fit, and a quartic fit.

   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. The answer to this part should only be a few sentences.

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? Give a few sentences of explanation. Attempt to answer this question honestly, if only to yourself, before continuing.

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

When we use K basis functions (polynomial or otherwise) with C datapoints we construct a $C \times K$ matrix of input features Φ. If the least squares weights are unique, they can be written:

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

where $\mathbf{x}$ are the previous amplitudes, as stored in a row of `X_shuf_train`. We normally see $\mathbf{y}$ in the equation above. In this curve-fitting context, the amplitudes in $\mathbf{x}$ are on the "y-axis" plotted against time t, and the Φ matrix contains transformed times. I have used C for the length of the context that we are predicting from (e.g., 20 time-steps), because later I 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:

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

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 $\mathbf{v}$. Include your working, which should be at most a few lines of linear algebra.

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 $f(t=1) = \mathbf{v}^\top \mathbf{x}$.

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 $\boldsymbol{\phi}^\top = [1 \;\; t \;\; t^2 \;\; \ldots \;\; t^{K-1}]$.

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.

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.

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.[1]

1. 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!

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

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 least squares we can find the vector $\mathbf{v}$ that minimizes

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

   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, which use C previous amplitudes to predict the next amplitude.

   a) What context length C has 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.

   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?

   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 the histogram in your report, with a one or two sentence comment on what you observe.

5. **What next?** Think about how you might improve the predictions of the best model considered so far. What would be an incremental improvement that you could plausibly try yourself, without writing a lot more code or using machine learning libraries[2]? Answers can be anything from a few sentences to half a page (maximum) depending on how keen you are. Try to include a plot that shows some aspect of the data that motivates your suggestion.

Things I 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 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.

2. "Implement WaveNet" is not an answer! https://deepmind.com/blog/wavenet-generative-model-raw-audio/