Week 3 exercises

This is the first page of *assessed* questions, as described in the background notes. These questions form 70% of your mark for Week 3. The introductory questions in the notes and the Week 3 discussion group task form the remaining 30% of your mark for Week 3.

To answer these questions, you will need to build on the programming parts of the Week 2 exercises.

Unlike the questions in the notes, you'll not immediately see any example answers on this page. However, you can edit and resubmit your answers as many times as you like until the deadline (Friday 9 October 4pm UK time). This is a *hard deadline*: This course does not permit extensions and any work submitted after the deadline will receive a mark of zero. See the late work policy.

Please only answer what's asked. Markers will reward succinct to-the-point answers. You can put any other observations in the "Add any extra notes" button (but this is for your record, or to point out things that seemed strange, not to get extra credit).

Feedback: We'll return feedback on your submission via email by Wednesday 14 October.

Good Scholarly Practice: Please remember the University requirements for all assessed work for credit. Furthermore, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (permitting access only to yourself). You may not publish your solutions after the deadline either.

1 Modelling audio continuation

This exercise continues the "Modelling audio" exercise from the Week 2 exercises. Complete that exercise first before doing this one. Your code should use the dataset splits created in that part, with the names you were asked to use.

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.

1. **Choosing a polynomial predictor based on performance [25 marks]:**

   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, there is a closed-form solution (which we will derive later in the course):

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

   where $\mathbf{x}$ are the previous 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:

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

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

   [The website version of this note has a question here.]

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

[The website version of this note has a question here.]

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.

[The website version of this note has a question here.]

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 on last week's sheet.

[The website version of this note has a question here.]

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]

[The website version of this note has a question here.]

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

[The website version of this note has a question here.]

2. **Fitting linear predictors across many snippets [15 marks]:**

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

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

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, 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.

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!

[The website version of this note has a question here.]

b) Take the predictor with the best validation error and compare it to the best polynomial model from Q1 on the test set. Include your code, and the two test set numbers. Which approach is better?

[The website version of this note has a question here.]

3. **What next? [30 marks]** 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[2].

   You could answer this question by formulating a few sentences to half a page with a maximum word count of 300. However, if you do not write or run any code, you can get a maximum of 12 marks for this question.

   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.

   The bulk of the marks available for this part are for succinct answers focussed on a single good idea. Most of the marking difference will come from explanations and code being excellent and concise. Only truly excellent answers will get more than half marks on this part.

   [The website version of this note has a question here.]

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