Part IIA Project - SF1: Data Analysis - Final Report

Bailey Brookes — Corpus Christi — bdb31

June 4, 2018

Abstract

Test

Contents

1 Introduction

2 Summary of previous work

2.1 Week 1 - Spectrum Analysis

Week 1 focused on spectrum analysis and basic digital signal processing techniques, which where then applied to real audio files. The following was learnt from week 1:

- The Fast Fourier transform (FFT) is much faster than the Discrete Fourier transform (DFT) as it makes use of redundancy and recursion to more efficiently compute the transform.
- The FFT is affected my many parameters of the input data sequence:
  - The resolution of the FFT increase with the length of the data sequence. This means zero padding (adding zeros to the end of the data) can be used to increase resolution and efficiency if the padding makes the data length a power of 2.
  - Windowing removes and sharp cut-offs when a proportion of the data is taken for the FFT, reducing the effect of spectral leakage.
  - Amplitude modulation has different effects depending on the type of modulation used, with random noise having the same effect as additive noise, a linear increase having spectral leakage and periodic modulation having peaks at the signal and modulation frequencies.
- Spectrum analysis techniques where then used on music and speech data to compare and constst signle notes and transients in music, and vowels and consonants in speech.
- All these techniques where then applied to a organ waveform to try and identify the notes present.

For more detailed explanation of week 1, see 'Part IIA Project - SF1: Data Analysis - First Interim Report'

2.2 Week 2 - Noise Reduction

Week 2 used the techniques of week 1 and the Weiner filter theory and applied it to noise reduction. The follwing was learnt from week 2:

- The filter `overlap_add.m` was analysed, and technique of 'overlap add' was look as specification for a tool to be used later in developing a noise reduction filter.
- A noise reduction filter using the Weiner filter with overlap add was developed and tested on audio files. To filter however, the noise added to the speech had to be estimated. Two methods of noise estimation where used:

  - Using a silent part of the speech to estimate to noise floor.
  - Subtracting a smoothed version of the signal from the original in each frame and using that in each frame as the noise. This method results in more digital distortion (caused by filter coefficients being zero resulting in sharp jumps) but is more robust as doesn't require a silent portion length hard coded into the filter and for some files no silent portions exist.

- One big problem with the Weiner filter is the digital distortion caused by filter coefficients being 0 and sharp jumps ensuing. This was decreased by zero padding each frame to increase the resolution o the frequency-domain filter but did add more processing time as more padding was used. Hence there was a trade off of time vs sound quality.
- Mean Squared Error (MSE) serves as a good figure of merit for a filter but doesn't account well for the introduction of digital distortion.

For more detailed explanation of week 1, see 'Part IIA Project - SF1: Data Analysis - Second Interim Report'

3 Parameter Estimation

3.1 Simple linear model

For the simple linear model, a constant with additive Gaussian white noise $y = \theta + e_n$ where $e_n = N(e_n; 0, 1)$ meaning $y = N(y; \theta, 1)$. The Maximum likelihood (ML) and maximum a posteriori (MAP) can either be computed by putting the following eqation into matrix form with generator matrix G a column matrix of 1s and then using the forms of the estimates given in the handout, or by directly forming the likelihood and posterior functions and maximization, which is done here.

3.1.1 ML estimate

The likelihood function, $p(\mathbf{y}|\theta)$ is given by the product of each data point as the data is independent identically distributed.

$$p(\mathbf{y}|\theta) = \prod_{i=1}^{N} p(y_i|\theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{\frac{1}{2\sigma^2}(y-\theta)^2}$$

Maximising the likelihood function is the same as maximising the log likelihood so by taking logs and differentiating with respect to θ and setting the result equal to zero, the ML estimate of θ is obtained:

$$
\begin{aligned}
L(\mathbf{y}|\theta) = \log\left(\prod_{i=1}^{N} p(y_i|\theta)\right) &= \log\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{\frac{1}{2\sigma_e^2}(y_i-\theta)^2}\right) \\
\therefore L(\mathbf{y}|\theta) &= -\frac{N}{2}\log(2\pi\sigma_e^2) - \frac{1}{2\sigma_e^2}\sum_{i=1}^{N}(y_i - \theta^2) \\
\frac{dL(\mathbf{y}|\theta)}{d\theta} &= \sum_{i=1}^{N}(y_i - \theta) = \sum_{i=1}^{N} y_i - N\theta = 0 \\
\therefore \theta^{ML} &= \frac{\sum_{i=1}^{N} y_i}{N}
\end{aligned}
$$

That is, the ML estimate for θ is the mean of the dataset y.

3.1.2 MAP estimate

To compute the map estimate, the likelihood function must be multiplied by a prior of θ. Throughout this report, we will assume a Gaussian prior on θ for simplicity:

$$\theta = N(\theta; m_\theta, \sigma_\theta^2)$$

Where m_θ, σ_θ^2 are the prior mean and variance on the θ parameter respectively. The MAP estimate is compute my forming the posterior density and then maximising it in the same way as the ML estimate: differentiating and setting to zero.

$$
\begin{aligned}
p(\theta|y) \propto p(y|\theta)p(\theta) &= \prod_{i=1}^{N} \frac{1}{2\pi\sigma_e^2}\frac{1}{2\pi\sigma_\theta^2} e^{-\frac{1}{2\sigma_e^2}(y_i-\theta)^2} e^{\frac{1}{2\sigma_\theta^2}(\theta-m_\theta)^2} \\
L(\theta|y) &= \sum_{i=1}^{N} -\frac{1}{2\sigma_e^2}(y_i-\theta)^2 - \frac{1}{2\sigma_\theta^2}(\theta - m_\theta)^2 + Constant \\
\frac{dL(\theta|y)}{d\theta} &= \sum_{i=1}^{N} y - N\theta + \frac{\sigma_e^2}{\sigma_\theta^2}\left(N\theta + Nm_\theta\right) = 0 \\
\therefore \theta^{MAP} &= \frac{\frac{\sum_{i=1}^{N} y}{N} + m_\theta\frac{\sigma_e^2}{\sigma_\theta^2}}{1 + \frac{\sigma_e^2}{\sigma_\theta^2}}
\end{aligned}
$$

As the variance tends infinite, meaning the prior becomes spread out and each value of θ is equally likely, the MAP estimate becomes the ML estimate.

The true posterior probability is normalised by $p(x)$ which is found by using the sun=m rule of probability. However it is easier to calculate by using Matlab to normalised the posterior for us which gives the following result:

$$p(\theta|y) = N\Big(\theta; \theta^{MAP}, \frac{\sigma_e^2 \sigma_\theta^2}{N\sigma_\theta^2 + \sigma_e^2}\Big)$$

Compared to the result for the likelihood:

$$p(y|\theta) = N(y; \theta^{ML}, \sigma_e^2)$$

3.1.3 Investigating these estimates

The Matlab script `Linear_model.m` investigates both of these estimates for different data length, noise variances and prior distributions on θ.

For small, medium and large N, the likelihood of θ remains unchanged but or larger N, the posterior get sharper as the variance of the estimate gets smaller. This is shown if Figure BLANK. The decrease invariance is obvious from the equation for the posterior distribution as as N grows large, the denominator of the variance grows as well shrinking the variance.

For increases in the error variance, both the ML and MAP estimates variance increase and the pdfs spread out as there is now more uncertainty in the measurement. For MAP, this can be combated by using more data samples are discussed above.

3.2 Linear Model with trend

Now with two parameters a multivariate model must be used, meaning that first the linear model of the model must be formed, and then the estimates of these parameter can be found using the results in the handout. The generator matrix G is:

$$G = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & N-1 \\ 1 & N \end{bmatrix}$$

And for simplicity we will assume that the θ parameters are independently normally distributed with the following parameters:

$$\theta = \begin{bmatrix} a \\ b \end{bmatrix} \text{ with mean } \mathbf{m}_\theta = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \text{ and covariance matrix } \mathbf{\Sigma}_\theta = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

The ML and MAP estimates are then computed as they are in the handout. The Matlab script `Linear_trend.m` investigates this. Some examples are tabulated below.

3.3 Model Choice

Here 3 models are considered: Pure noise, a constant plus noise and a linear trend plus noise. The script `Model_choice.m` can generate data from either model, work out the MAP estimates for the parameters and also the marginal likelihoods of each of the models. This assumes the modes,a re all equally likely, a useful assumption for later use of this script. Some example results are tabulated below.

One interesting thing to note is that the results for the marginal likelihoods $p(x|M_i)$, which can be sued interchangeably with the posterior $p(M_i|x)$ as long as the models are equally likely and the priors on θ are Gaussian, produce very small values for even moderate values of N (for example 10). WHY IS THIS THE CASE.

3.4 Bayesian Needle in a digital haystack

3.5 The Autoregressive (AR) Model

3.5.1 Generating AR data

3.5.2 Parameter Estimation for AR

The same results for the ML nd MAP estimates can be used for the AR model, meaning the same code can be used. All there is to do is form the linear model, specifically the generator matrix for an AR model where x is zero before the first sample is:

$$G = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ x_1 & 0 & \cdots & 0 & 0 \\ x_2 & x_1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-P} & x_{N-P-1} \end{bmatrix}$$

When the estimator reaches the models true order, there is a sharp decrease in the mean squared error (MSE) as shown in Figure BLANK. It then stays roughly constant. This can be used to indicate the true model order as although a higher order results in a slightly smaller MSE, it can result in over fitting.

3.5.3 Model Selection for AR

Now the AR model is in linear form, model selection can be done again. If ML estimation is used, model selection is only choosing the appropriate order of AR model. For MAP, this can be further tuned by selecting both an appropriate order and set of parameters.

3.5.4 Predicting Missing data