Report file - Problem Set #8

Matteo Dell'Acqua
GitHub: MatteoDellAcqua6121

November 5, 2024

Abstract

This is the report for the problem set #8. Since the problem set is composed of three exercises, we divide the report into three sections, one for each problem. The scripts (labelled as ps_8.'problem number') and the raw file of the images are in this directory.

1 Problem 1

1.1 Formulation of the problem

We are asked to make a maximum likelihood estimation.

More explicitly, we are given a sample of a survey of yes/no answers depending on the interlocutor's age (in years). We model the probability of a positive answer at a fixed age as a binomial of probability $p = p(x)$, which follows the logistic function:

$$p(x; \beta_0, \beta_1) = \frac{1}{1 + \exp[-(\beta_0 + \beta_1 x)]}. \tag{1}$$

We then look for parameters (β_0, β_1) that maximize the natural logarithm of the likelyhood function:

$$\mathcal{L}(\beta_0, \beta_1; \vec{y}) = P(\vec{y}; \beta_0, \beta_1) = \prod_i p(x_i; \beta_0, \beta_1)^{y_i} (1 - p(x_i; \beta_0, \beta_1))^{1-y_i} \tag{2}$$

where $\vec{y}$ is the sampled results and is a *parameter* of the likelihood function, whose *imput* are (β_0, β_1); and we denoted with $P(\vec{y})$ the joint probability of sampling the whole vector $\vec{y}$.

1.2 Computational methods

After importing the data using `np.genfromtxt` and defining the logarithm of the likelihood 2 as:

```
def likely(variables):
    b0,b1=variables
    P=p(ages, b0,b1)
    eps=1e-5
    return -np.sum((answers*np.log(P+eps)+(1-answers)*np.log(1-P+eps)))
```

where we added a small amount ϵ to the argument of each logarithm to avoid `NaN` errors.

Finally, we find the maximum of the log-likelihood by minimizing its opposite via `sp.optimize.minimize`.

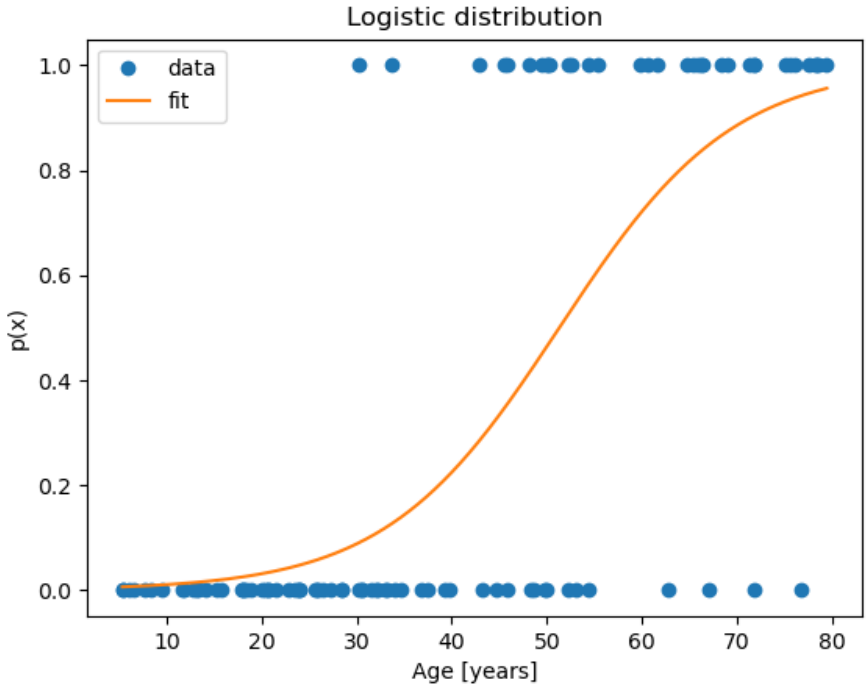

Figure 1: Plot of the sample survey (blue dots) and its modelling via a logistic function. On the horizontal axes the age (in years) of the participants, on the vertical axes yes/no answer are respectively encoded in $\{1, 0\}$.

1.3 Results

We find the values of (β_0, β_1) maximizing the likelihood to be:

$$\overline{\beta_0} = -5.620 \pm 1.066, \quad \overline{\beta_1} = 0.1096 \pm 0.0209, \tag{3}$$

(with $-\log\big(\mathcal{L}(\overline{\beta_0}, \overline{\beta_1})\big) = 34.7238$) and their covariance matrix to be:

$$\begin{pmatrix} 1.137 & -2.134 \cdot 10^{-2} \\ -2.134 \cdot 10^{-2} & 4.366 \cdot 10^{-4} \end{pmatrix} \tag{4}$$

where the covariance is estimated by the inverse of the hessian of the log-likelihood computed at $(\overline{\beta_0}, \overline{\beta_1})$ and the error by the square root of its diagonal entries.

Finally, we report the plot of the fit vs the original data in fig. 1. While both estimated parameters suffer from a $\sim 20\%$ uncertainty, the plot seems to reasonably model the data.

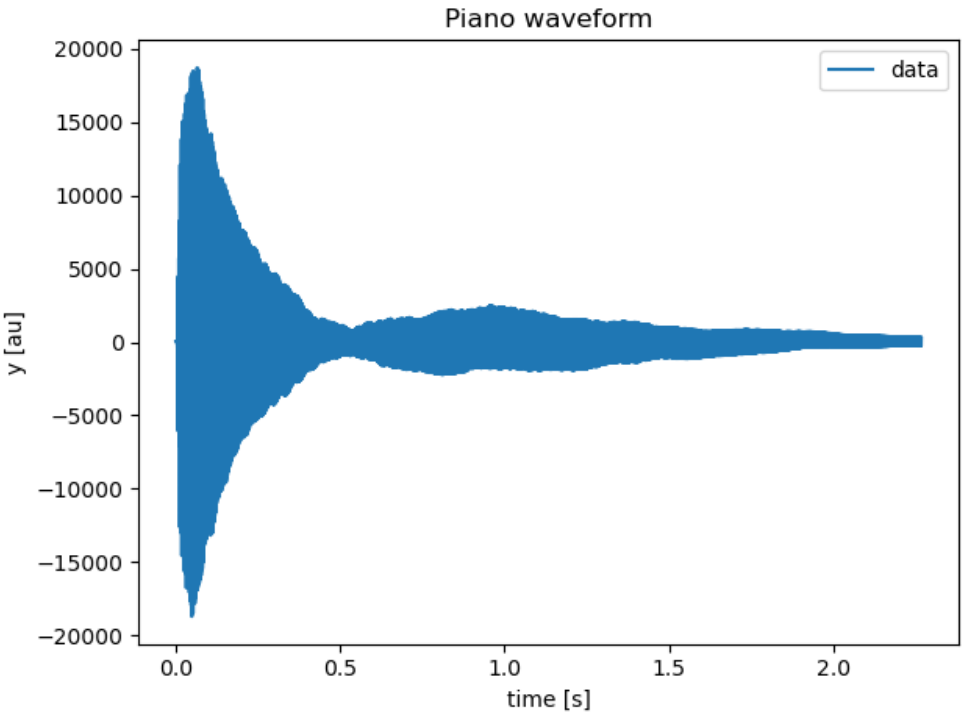

Figure 2: Raw signal of the piano.

2 Problem 2

2.1 Formulation of the problem

In this problem we are going to apply a discrete Fourier transform to signals h coming from either a piano or a trumpet:

$$H_n = \sum_{k=0}^{N-1} h(t_k) \exp(2\pi i k n / N) \tag{5}$$

in order to discover the note played.

2.2 Computational methods

After importing the data using `np.genfromtxt` we apply the `sp.fft.fft` funtion to it.

```
fft=np.absolute(sp.fft.rfft(data))
```

2.3 Results

We plot the original signals and their Fourier transform in figs. 2 to 5.

From the largest coefficient in the transformed signal, we can compute the note played to be

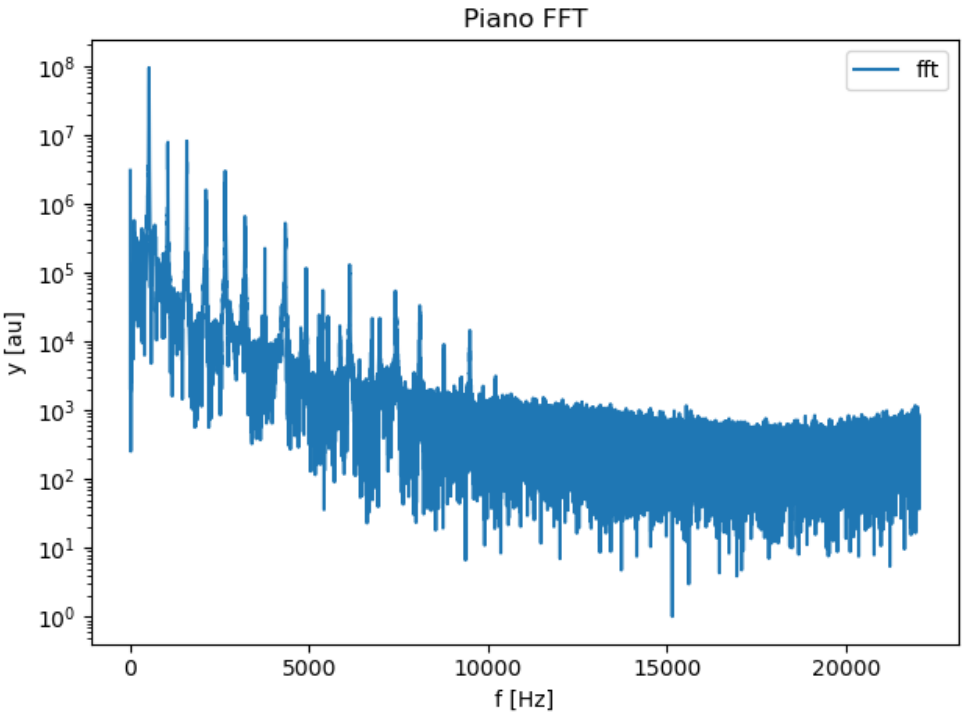

Figure 3: Discrete Fourier transform of the signal of the piano in logarithmic scale.

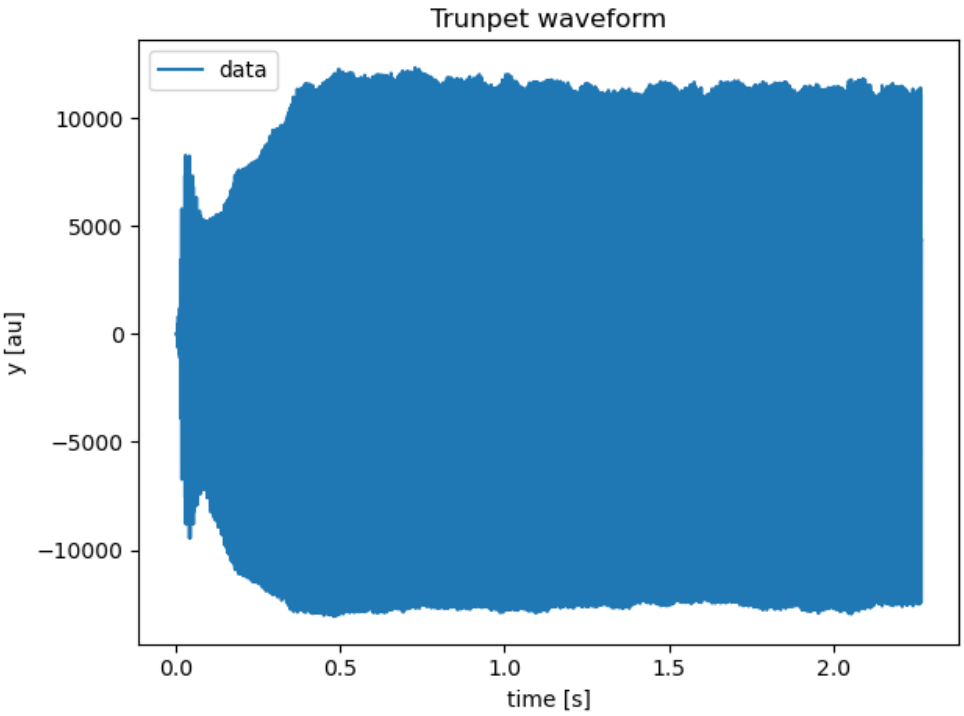

Figure 4: Raw signal of the trumpet.

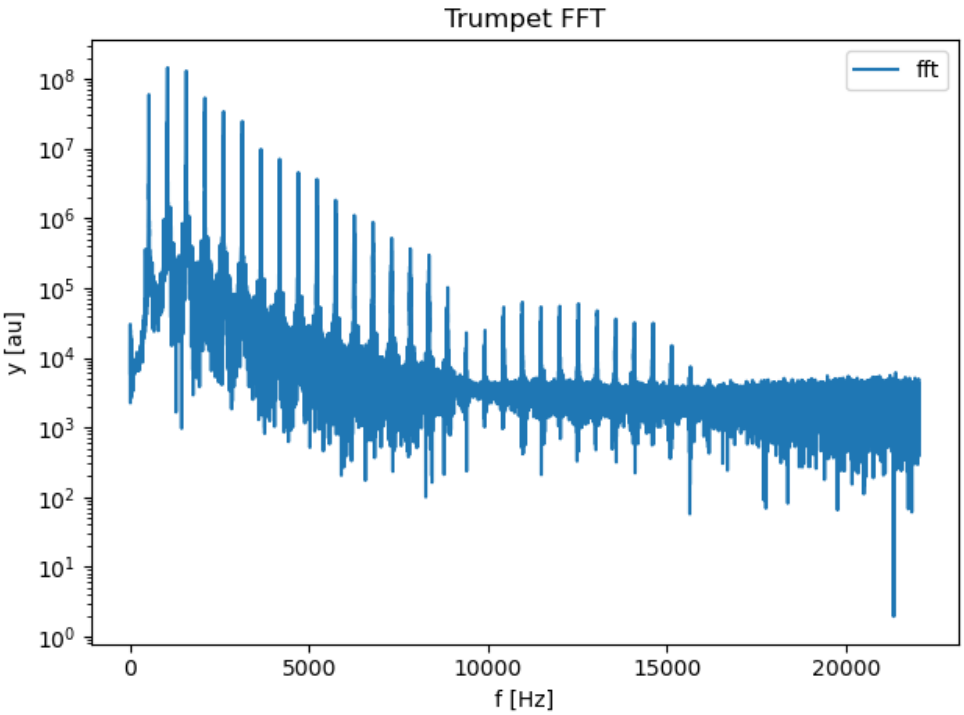

Figure 5: Discrete Fourier transform of the signal of the trumpet in logarithmic scale.

(remembering that $f_n = n/N\Delta$ and using the sampling frequency is 44100 Hx):

$$f_{\text{piano}} = 524.79\,\text{Hx} \rightsquigarrow \text{C5 plus 6 cents}, \quad f_{\text{trumpet}} = 1043.847\,\text{Hx} \rightsquigarrow \text{C6 minus 4 cents} \tag{6}$$

3 Problem 3

3.1 Formulation of the problem

In this problem we are going to apply a discrete Fourier transform to the data of the Dow Jones Industrial Average and analyse how much we could compress the data without meaningful loss of information.

3.2 Computational methods

After importing the data using `np.genfromtxt` we apply the `sp.fft.rfft` function to it. We then delete part of the data and invert the resulting transform via `sp.fft.irfft`. Here a sample of our code for the 10% case:

```
fft=sp.fft.rfft(data)
r=len(fft)
for i in np.arange(np.uint(math.ceil(r*0.9))):
    fft[r-i-1]=0
```

3.3 Results

We report the original data, its reconstruction with only the first 10% of the Fourier coefficients and that with only the first 2% respectively in figs. 6 to 8.

We can see that at every step the reconstructed signal is getting more dissimilar from the original: while the 10% approximation can still be regarded as a faithful compression for some use cases, the 2% one differs significantly from the raw data.

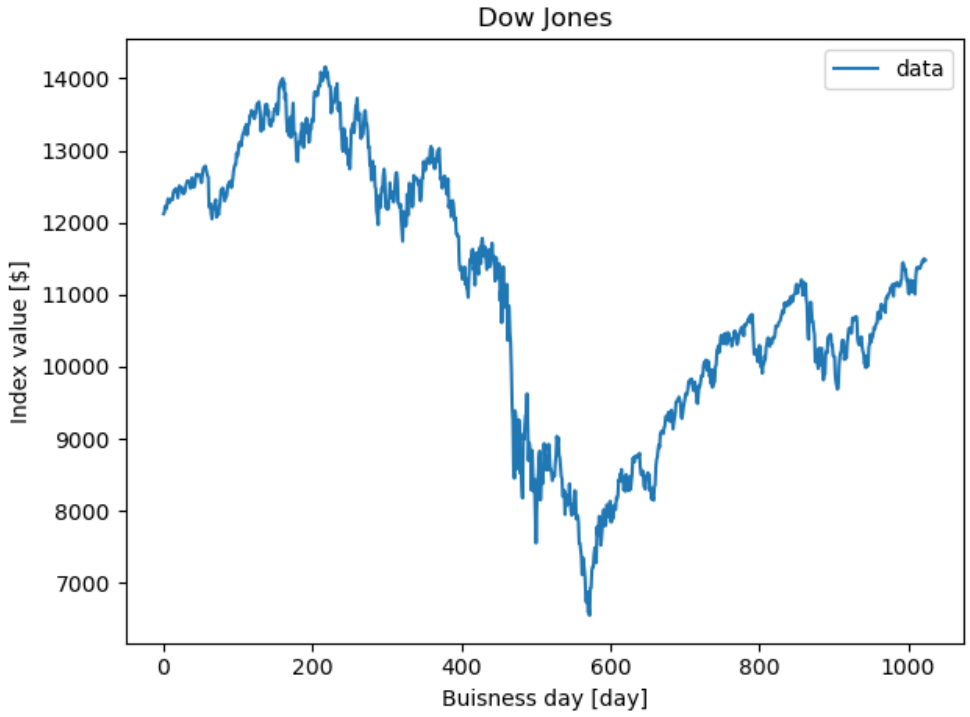

Figure 6: Daily closing values of each business day from late 2006 to the end of 2010 of the Dow Jones Industrial Average.

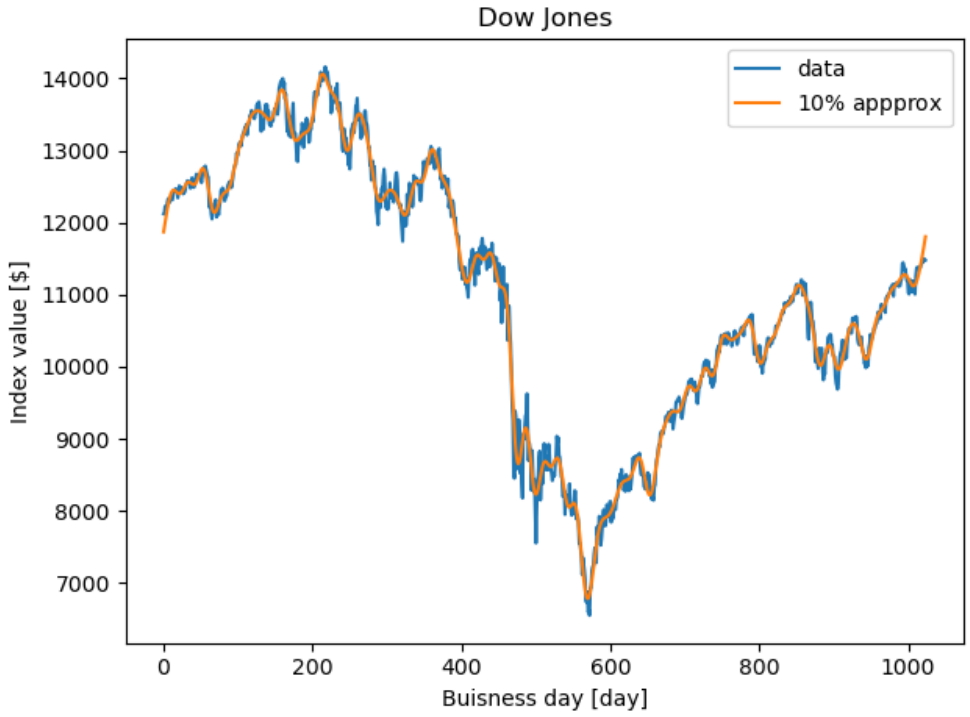

Figure 7: Daily closing values of each business day from late 2006 to the end of 2010 of the Dow Jones Industrial Average and its reconstruction using only the first 10% of the Fourier coefficients.

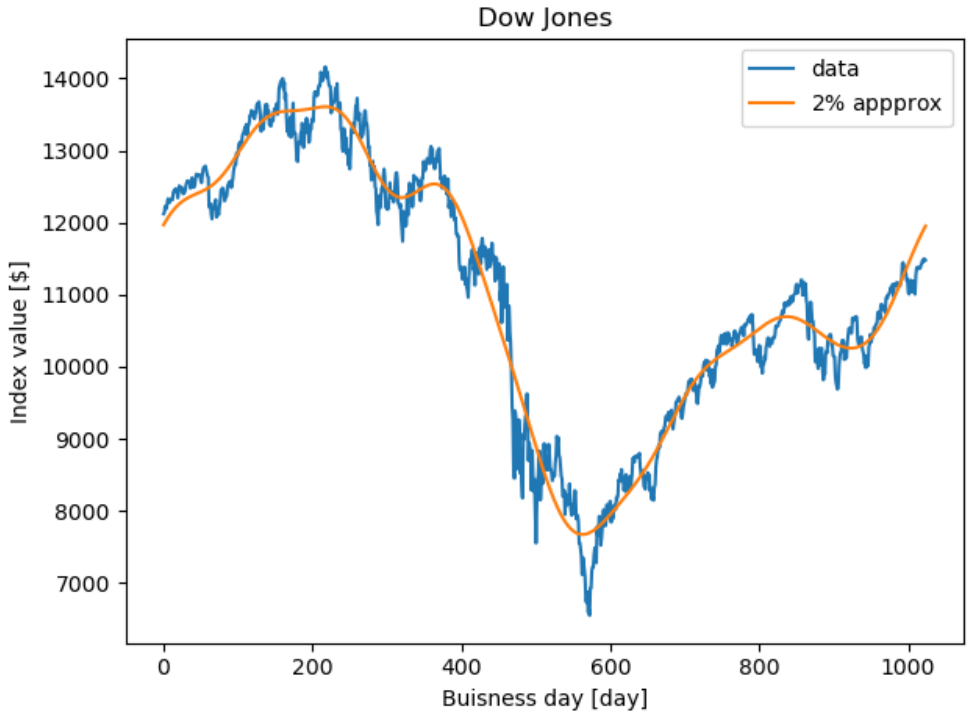

Figure 8: Daily closing values of each business day from late 2006 to the end of 2010 of the Dow Jones Industrial Average and its reconstruction using only the first 2% of the Fourier coefficients.