Arben Cela
ESIEE Paris
Department of Computer Science and Telecommunication
Deep-Learning course : Tutorial 1

7 mars 2021

Adaptive Noise Cancellation

1 Introduction

Let us suppose that a doctor, in trying to review the electroencephalogram (EEG) of a distracted graduate student, finds that the signal he would like to see has been contaminated by a 60-Hz noise source. He is examining the patient on-line and wants to view the best signal that can be obtained. Figure 10.6 shows how an adaptive filter can be used to remove the contaminating signal.

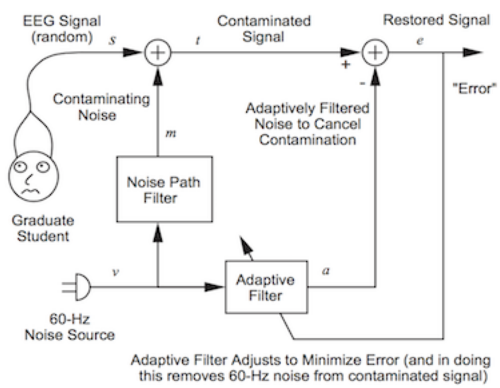

Figure 10.6 Noise Cancellation System

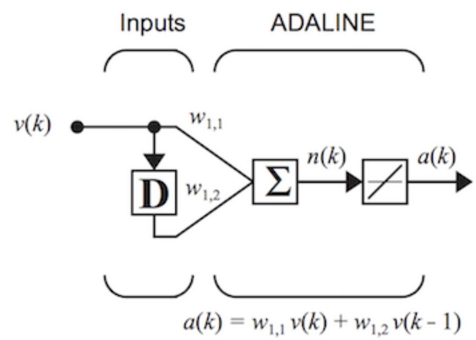

Figure 10.7 Adaptive Filter for Noise Cancellation

As shown, a sample of the original 60-Hz signal is fed to an adaptive filter, whose elements are adjusted so as to minimize the ***error*** $\mathbf{e(t)}$. The output of the noise path filter is the perturbating/contaminating signal $\mathbf{m(t)}$. The **adaptive filter** will do its best to reproduce this signal, but it only knows about the original noise source $\mathbf{v(t)}$ and contaminated *EEG signal* $\mathbf{t(t)}$. Thus, it can only reproduce the part of $\mathbf{t(t)}$ that is linearly correlated with $\mathbf{v(t)}$, which is $\mathbf{m(t)}$. In effect, the adaptive filter will attempt to mimic the noise path filter, so that the output of the filter $\mathbf{a(t)}$ will be close to the contaminating noise $\mathbf{m(t)}$. In this way the error $\mathbf{e(t)}$ will be close to the original uncontaminated *EEG signal* $\mathbf{s(t)}$. In this simple case of a single sine wave noise/perturbation source, a neuron with two weights and no bias is sufficient to implement the filter. The inputs to the filter are the current and previous values of the noise source. Such a two-input filter can attenuate and phase-shift the noise $\mathbf{v(t)}$ in the desired way. The filter is shown in **Figure 10.7**.

1.1 Mathematical solution

We can apply the mathematical relationships developed in course slides [1] to analyze this system. In order to do so, we will first need to find the input correlation matrix $R = [zz^T]$ and the input/target cross-correlation vector $h = E[tz]$. In our case the input vector is given by the current and previous values of the noise source : $Z(k) = [v(k), v(k-1)]^T$ while the target is the sum of the current signal and filtered noise : $t(k) = s(k) + m(k)$. So we obtain :

$$R = \begin{pmatrix} E[v^2(k)] & E[v(k)v(k-1] \\ E[v(k-1)v(k)] & E[v^2(k-1)] \end{pmatrix}, \qquad h = \begin{pmatrix} E[(s(k)+m(k))v(k)] \\ E[(s(k)+m(k))v(k-1)] \end{pmatrix}$$

1. **please see slide nr. 19**

In order obtain specific values for these two quantities we must define the noise signal **v(t)**, the *EEG signal* **s(t)** and the filtered noise **m(t)** . For this exercise we will assume : the *EEG signal* is a white (uncorrelated from one time step to the next) random signal uniformly distributed between the values *-0.2 and +0.2*, the noise source (60-Hz sine wave sampled at 180 Hz) is given by : $v(k) = 1.2sin(\frac{2\pi k}{3})$ and the filtered noise that contaminates the *EEG signal* is the noise source attenuated by a factor of 10 and shifted in phase by $\pi/2$: $m(k) = 0.12sin(\frac{2\pi k}{3} + \frac{\pi}{2})$,

The elements of the input correlation matrix are : $R = \begin{pmatrix} 0.72 & -0.36 \\ -0.36 & 0.72 \end{pmatrix}$ whereas the input/target cross-correlation vector are : $h = \begin{pmatrix} 0 \\ -0.0624 \end{pmatrix}$

1.1.1 Involved Signals Specifications :

- EEG signal free of perturbation is $s(t)$ and saved in data file **Data_EEG.txt**
- The contaminating/perturbating signal $m(t)$ is a 60-Hz modulated signal from original signal $v(t)$ and sampled at a frequency of 180-Hz.
- The noisy EEG signal $t(t)$ is given by : $t(t) = s(t) + m(t)$
- As input of **ADALINE** neural network we use original noise signal $v(k)$ and $v(k-1)$ (delayed value of one sampling period).

1.1.2 Objectives :

- Find the weights of **ADALINE neural network** based on theoretical/mathematical solution.
- Find the weights of **ADALINE neural network** [2] (given in **figure 10-7** playing the role of adaptive filter of **figure 10-6**) composed of only **one neuron** with two inputs, $v(k)$ and $v(k-1)$, and one output $a(k)$ (second order **Adaline Network filter**) applying a gradient based learning/minimisation algorithm.
- Calculate the weights for third order **ADALINE neural network** and compare with the previous cases.
- Implement the **ADALINE neural network filter** using the following functions of **Neural Network Matlab Toolbox** (**linearlayer() ; num2cell ; preparets() ; train() ; view(net) ; net()**) or the **Python Deep Learning library** **Kehras**
- Find the analytical solution of the weights calculation and simulate your the **ADALINE neural network** obtained.
- **Analyse the results and conclude**.

1.1.3 Delivrables :

A detailed report must be deposited at the place reserved under BlackBoard before the 18 of April 2021 specifying :

- **object :** *TPx_Cela_Gry_nom1_nom2* where **x** is the number of tutorial (1,2 or 3) and **y** is the number of groupe (1 or 2)
- **the full name of two authors of the report**

2. **please see slide nr. 23**