Machine learning for signal processing

[5LSL0]

Rik Vullings - TU/e (Flux 7.076)
Ruud van Sloun - TU/e (Flux 7.078)

Assignments Optimum and linear filters

April 4, 2019

Optimum linear and adaptive filters

In this assignment optimum and adaptive filtering algorithms have to be implemented and evaluated. The objective is to evaluate and compare different facets of the performance of these adaptive filtering algorithms.
Fill in your answers on the predefined reports Assignment1.docx

1.1 Preview

1.1.1 Adaptive filtering

The adaptive filtering techniques covered in this course attempt to find a set of weights for a Finite Impulse Response (FIR) filter $\underline{\mathbf{w}} = (w_0, w_1)^t$ that minimizes an objective function J. The objective function is chosen such that it represents the squared difference between a filtered version of an input signal $x[k]$ and a given reference signal $y[k]$, resulting in $J = E(e[k]^2)$ with $e[k] = y[k] - \underline{\mathbf{w}}^t[k]\underline{\mathbf{x}}[k]$. The final FIR filter parameters will therefore give the best possible estimate of the reference signal based on $x[k]$.

1.1.2 Known statistics — Wiener/GD/Newton

When the signal statistics are known, i.e., we collect the auto correlation of $x[k]$ in matrix $\mathbf{R}_x$ and the cross correlation between $x[k]$ and $y[k]$ in the vector $\underline{\mathbf{r}}_{yx}$, we can easily solve the optimization problem. As deducted in the course, we can compute these optimal weights by $\underline{\mathbf{w}}_0 = \mathbf{R}_x^{-1}\underline{\mathbf{r}}_{yx}$. This solution is also referred to as the **Wiener filter**.

Finding the inverse of $\mathbf{R}_x$ can be computationally expensive and we therefore require a less complex way to find $\underline{\mathbf{w}}_0$. The **Steepest Gradient Descent (GD)** algorithm approaches $\underline{\mathbf{w}}_0$ by updating in the negative gradient direction: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k])$, with α determining the rate of convergence. A drawback of the GD algorithm is that the filter weights do not converge uniformly. The eigenvalue spread of auto correlation matrix $\mathbf{R}_x$ influences the rate of convergence of individual weights.

The Newton algorithm improves upon the GD algorithm by undoing the coloration of $\mathbf{R}_x$. Newton's method applied to our case results in the following update rule: $\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\mathbf{R}_x^{-1}(\underline{\mathbf{r}}_{yx} - \mathbf{R}_x\underline{\mathbf{w}}[k])$. Obviously, both GD and Newton's method are not always directly applicable in practice, because we may not know the signal statistics.

1.1.3 Unknown statistics — LS/LMS/RLS

To overcome the practical problems of GD and Newton's algorithm, we need a way to estimate the signal statistics from our input observations. Ideally, we would compute $\bar{\mathbf{R}}_x$ and $\bar{\underline{\mathbf{r}}}_{yx}$ of stationary signals as follows:

$$\bar{\mathbf{R}}_x = \lim_{L\to\infty} \frac{1}{L} \sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]\underline{\mathbf{x}}^t[k-i] \tag{1.1}$$

$$\bar{\underline{\mathbf{r}}}_{yx} = \lim_{L\to\infty} \frac{1}{L} \sum_{i=0}^{L-1} \underline{\mathbf{x}}[k-i]y[k-i] \tag{1.2}$$

The Least Squares (LS) solution computes the Wiener filter based on the above estimates of the signal statistics. To avoid summing a large number of elements, several algorithms exist that estimate the statistics using less complex methods.

First of all, the Least Mean Squares (LMS) algorithm uses instantaneous estimates of the statistics given by: $\bar{\mathbf{R}}_x = \underline{\mathbf{x}}[k]\underline{\mathbf{x}}^t[k]$ and $\bar{\underline{\mathbf{r}}}_{yx} = \underline{\mathbf{x}}[k]y[k]$. To obtain $\underline{\mathbf{w}}_0$, the GD update is applied at each iteration. This results in:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\underline{\mathbf{x}}[k]e[k] \tag{1.3}$$

Normalized Least Mean Squares (NLMS) extends LMS by using a normalized input. An estimate of the input signal power is maintained in $\hat{\sigma}_x^2$. Two example algorithms for updating the estimated signal power are given in the course. The NLMS update rule becomes:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + \frac{2\alpha}{\hat{\sigma}_x^2}\underline{\mathbf{x}}[k]e[k] \tag{1.4}$$

Finally, the Recursive Least Squares (RLS) algorithm is a practical version of the LS algorithm. As opposed to an infinite window, RLS uses an exponentially decreasing sample window characterized by the 'forget factor' γ. Instead of inverting $\bar{\mathbf{R}}_x$, RLS maintains a recursive estimate of its inverse $\bar{\mathbf{R}}_x^{-1}[k]$. Combined with the estimated $\bar{\underline{\mathbf{r}}}_{yx}[k]$, the final filter weights are given by $\underline{\mathbf{w}}[k] = \bar{\mathbf{R}}_x^{-1}[k]\bar{\underline{\mathbf{r}}}_{yx}[k]$. The recursive update rule for the estimates of the signal statistics is given by:

$$\underline{\mathbf{g}}[k+1] = \frac{\bar{\mathbf{R}}_x^{-1}[k]x[k+1]}{\gamma^2 + \underline{\mathbf{x}}^t[k+1]\bar{\mathbf{R}}_x^{-1}[k]\underline{\mathbf{x}}[k+1]} \tag{1.5}$$

$$\bar{\mathbf{R}}_x^{-1}[k+1] = \gamma^{-2}(\bar{\mathbf{R}}_x^{-1}[k] - \underline{\mathbf{g}}[k+1]\underline{\mathbf{x}}^t[k+1]\bar{\mathbf{R}}_x^{-1}[k]) \tag{1.6}$$

$$\bar{\underline{\mathbf{r}}}_{yx}[k+1] = \gamma^2\bar{\underline{\mathbf{r}}}_{yx}[k] + \underline{\mathbf{x}}[k+1]y[k+1] \tag{1.7}$$

1.2 Assignment 1a: Known statistics

Consider the general filter optimization scheme as depicted in Fig. 1.1. In this part of the

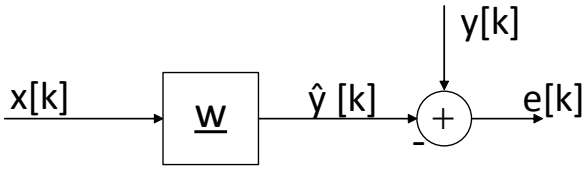

Figure 1.1: Assignment 1: MMSE model

assignment you have to calculate and design a three-tap FIR filter with $\underline{\mathbf{w}} = (w_0, w_1, w_2)^t$, by minimizing the MSE $J = E\{e^2[k]\}$. The following statistics are given:

$$\mathbf{R}_x = \begin{pmatrix} 5 & -1 & -2 \\ -1 & 5 & -1 \\ -2 & -1 & 5 \end{pmatrix} \; ; \; \underline{\mathbf{r}}_{yx} = \begin{pmatrix} 1 \\ 5.3 \\ -3.9 \end{pmatrix}.$$

1.2.1 Wiener Filter

Fill in your answers on the predefined report Assignment1.docx and do not forget to include relevant calculations and figures in this predefined report!

a) Calculate by hand an expression for the MSE $J = E\{e^2[k]\}$ as a function of the three weights w_0, w_1, and w_2. Next, find the filter weights w_0, w_1, and w_2 that minimize the MSE J by setting $\partial J/\partial w_0 = 0$, $\partial J/\partial w_1 = 0$, and $\partial J/\partial w_2 = 0$. Show your calculations and give the numerical values of $\underline{\mathbf{w}}$.

b) Find an expression for the correlation $\underline{\mathbf{r}}_{xe}$ between $\underline{\mathbf{x}}[k]$ and $e[k]$. What should $\underline{\mathbf{r}}_{xe}$ be when $\underline{\mathbf{w}}$ equals the optimal Wiener filter $\underline{\mathbf{w}}_o$? What does this mean?

c) In practical cases, $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$ may be unknown. How can you estimate these statistics from a set of observations $\underline{\mathbf{x}}[k]$ and $y[k]$? Which tradeoff do you have to make?

1.2.2 Steepest Gradient Descent

Computing the optimal Wiener filter involves inverting $\mathbf{R}_x$. To avoid the matrix inversion, one can use an iterative approach. The Steepest Gradient Descent (GD) algorithm is such an algorithm.

d) When does the GD algorithm reach steady-state? In other words, when the GD reaches steady-state, what is the value of $\underline{\mathbf{w}}[k]$?

e) Give the range of α which makes the GD algorithm stable. Calculate the eigenvalues of $\mathbf{R}_x$ by hand (i.e. show the derivation). Hint: one of the eigenvalues $\lambda = 7$.

f) Implement the filter update rule of the GD algorithm in Python. Take the following steps to do so.

- Download the file `assignment1_data.csv` from Canvas. This file contains generated samples of both $\underline{\mathbf{x}}[k]$ (column 1) and $y[k]$ (column 2).
- Initialize the filter
- Implement the adaptive filter and perform N iterations of this adaptive filter, where N is the number of samples in the dataset. During each iteration, update the filter coefficients.
- Plot the trajectory of the filter coefficients as they evolve, together with a contour plot of the objective function J. Plot the convergence for two of the three filter coefficients $\underline{\mathbf{w}}$: w_0 and w_1 to avoid having to create a 3D plot. For generating the contours, choose $w_2 = -0.5$. **NOTE: for other exercises in this assignment, use the same value of w_2 and every time plot the convergence of w_0 and w_1 when being asked.**
- Use a value of α that ensures a smooth and stable convergence of $\underline{\mathbf{w}}$. Comment on the convergence of w_0 and w_1.

1.2.3 Newton's Method

Input coloration influences the performance of the GD algorithm. To solve this issue, the Newton algorithm performs pre-whitening of the input signals.

g) Show that Newton's method makes all filter weights converge at the same rate.

h) For which α is Newton's method stable?

i) Implement the Newton filter update rule in Python. Again use an α that gives smooth convergence. Use the code from the previous exercise as starting point and modify it such that the filter uses the Newton update rule. Run your code and comment on the convergence of the filter parameters.

1.3 Assignment 1b: Unknown Statistics

Both the GD and Newton method assume that $\mathbf{R}_x$ and $\underline{\mathbf{r}}_{yx}$ are known. However, in practical cases these are often unknown. The Least Mean Squares (LMS) and the Normalized LMS (NLMS) algorithms use an instantaneous estimate of the signal statistics.

1.3.1 LMS and NLMS

j) Implement the LMS filter update rule, where you use your code from Assignment 1a as starting point. Again, use a filter length of 3 and an appropriate α. What trade-off do you make when choosing α? Include the resulting plot and the most relevant line(s) of code in your answer form (**use Assignment1.docx**).

k) Repeat Assignment j) for Normalized LMS.

1.3.2 RLS

The RLS algorithm attempts to whiten the input using estimates of the signal statistics.

l) Explain in your own words how RLS is related to the previous adaptive algorithms (LS/GD/Newton/LMS/NLMS). How does RLS solve the practical issues of the corresponding algorithm?

m) Implement the RLS update rule in Python. Use $\gamma = 1 - 10^{-4}$ and include the plot of convergence of the filter coefficients in your answer form. How does γ influence the convergence behavior?

n) Compare the advantages and disadvantages of LMS, NLMS, and RLS in terms of computational complexity and accuracy. Try to rank them (#1 is the best, #3 the worst). Give a brief explanation.