02471 Machine Learning for Signal Processing

Week 5: Adaptive Linear Filtering with RLS

This exercise is based on S. Theodoridis: Machine Learning, A Bayesian and Optimization Perspective 2nd edition, section(s) 5.12, 6.6–6.8.

The objective of this exercise is to expand on the linear filtering exercises from week 3 and 4, and get better acquainted with adaptive linear filtering and the RLS algorithm.

Overview

Adaptive filters are used in a wide range of applications where the characteristics of the signals or the system is not known a priori, or is known to be time varying.

In this exercise we will consider the adaptive finite impulse response (FIR) filter. The adaptation strategies will be Least-Mean-Square (LMS), Normalized Least-Mean-Square (NLMS) and Recursive Least Squares (RLS), with the primary focus on RLS.

The exercise have the following structure:

5.1 will derive the RLS algorithm and relate the equations to implementation.

5.2 will study the convergence behavior of the derived RLS algorithm in a stationary environment, and compare to NLMS.

5.3 will study the behavior of NLMS and RLS in a non-stationary environment, and get hands-on experience with tuning, and see under certain circumstances the NLMS performs better than RLS.

5.4 will continue to work on audio signals in the same manner as the previous two weeks. The audio is contaminated with noise, both stationary and non–stationary, and a FIR Wiener filter is then used to suppress the noise. Adaptive algorithms are used to train the Wiener filter.

Notation

- $J(\boldsymbol{\theta})$ is a cost function that we are seeking to minimize with respect to $\boldsymbol{\theta}$.
- In this exercise, the vector $\boldsymbol{p}$ is used, which is NOT related to the cross-correlation vector (that was in previous exercises also denoted $\boldsymbol{p}$).
- n indicates the time step as we collect data $(y_n, \boldsymbol{x}_n)$, where y_n is our target, or desired signal, and $\boldsymbol{x}_n$ is our filter input.

Code

The code can be found in the .m and .py files named in the same way as exercises, ie. the code for exercise 5.x.y is in the file 5_x_y.m (or .py).

For coding exercises that requires implementation we will usually write `complete this line` where the implementing should be done.

Solutions

The solution is provided for all derivation exercises, and often hints are provided at the end of the document. If you get stuck, take a look at the hints, and if you are still stuck, take a look in the solution to see the approach being taken. Then try to do it on your own.

Solutions are also provided for some coding exercises. If you get stuck, take a look at the solution, and then try to implement it on your own.

5.1 Derivation of the RLS algorithm

In this exercise we will derive the RLS algorithm step–by step. Be sure to read section 6.6 in the book before carrying out this exercise.

The starting point is the cost function defined as

$$J(\boldsymbol{\theta}, \beta, \lambda) = \sum_{i=0}^{n} \beta^{n-i}(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2$$

which is related to Ridge regression, hence the RLS algorithm minimizes a Ridge regression–like cost in a sequential manner as data points are collected.

Exercise 5.1.1

For which value of β will this correspond to Ridge regression?

Exercise 5.1.2

What happens with the regularization term (the last term) as we observe more data ?

Exercise 5.1.3

Find an expression for the derivative of $J(\boldsymbol{\theta}, \beta, \lambda)$ with respect to $\boldsymbol{\theta}$ and set the derivative to zero. This will readily give the expressions:

$$\begin{aligned}
\Phi_n &= \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^T + \lambda \beta^{n+1} I \\
\boldsymbol{p}_n &= \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i y_i \\
\boldsymbol{\theta}_n &= \Phi_n^{-1} \boldsymbol{p}_n
\end{aligned}$$

Exercise 5.1.4

The goal is to find an iterative scheme for $\boldsymbol{\theta}_n$. Next step to achieve this goal is to create the time–iterative updates for Φ_n and $\boldsymbol{p}_n$. First write out the expressions for Φ_{n-1} and $\boldsymbol{p}_{n-1}$ by replacing n with $n-1$ in the above formulas. Next, manipulate the original expressions for Φ_n and $\boldsymbol{p}_n$ to get

$$\begin{aligned}
\Phi_n &= \beta \Phi_{n-1} + \boldsymbol{x}_n \boldsymbol{x}_n^T \\
\boldsymbol{p}_n &= \beta \boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n
\end{aligned}$$

Exercise 5.1.5

To arrive at the primary result (eq 6.44 and 6.45 in the book), we use Woodburry's inversion formula

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}$$

Use $A = \beta\Phi_{n-1}$, $B = \boldsymbol{x}_n$, $C = \boldsymbol{x}_n^T$, $D = 1$, to get

$$\Phi_n^{-1} = \beta^{-1}\Phi_{n-1}^{-1} - \beta^{-1}\boldsymbol{k}_n\boldsymbol{x}_n^T\Phi_{n-1}^{-1}$$
$$\boldsymbol{k}_n = \frac{\beta^{-1}\Phi_{n-1}^{-1}\boldsymbol{x}_n}{1 + \beta^{-1}\boldsymbol{x}_n^T\Phi_{n-1}^{-1}\boldsymbol{x}_n}$$

Exercise 5.1.6 (This is a difficult exercise)

Now we have time–iterative expressions for Φ_n and $\boldsymbol{p}_n$. Show by substituting the expressions into $\boldsymbol{\theta}_n = \Phi_n^{-1}\boldsymbol{p}_n$, and defining $e_n = y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n$, that

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \boldsymbol{k}_n e_n$$

which concludes the derivation of the RLS algorithm.

Exercise 5.1.7

Inspect the code for RLS and relate the code and derived equations to algorithm 6.1 in the book. Try and implement the update formulas on your own (i.e. after inspection, erase the RLS code, and re-implement it).

5.2 Adaptive filtering simulations in stationary environment

The total error for a filter trained with an adaptive algorithm can be divided into three parts:

$$J = J_{\text{min}} + J_{\text{excess}} + J_{\text{lag}}$$

where J_{min} is the minimum achievable error (the error if the weights are optimal), J_{excess} is the error due to fluctuations around the true weights, and J_{lag} is the error induced by the filter lagging behind the correct solution. The expressions for $J_{\text{exc}} = J_{\text{excess}} + J_{\text{lag}}$ is summarized in table 6.1 in the book.

In this exercise we will simulate a stationary environment (hence $J_{\text{lag}} = 0$) and create convergence curves. This exercise corresponds to example 6.1, and problem 6.20 in the book, ignoring the APA algorithm. Make sure to read both the example 6.1 and problem 6.20 before completing this exercise.

Exercise 5.2.1

We consider the following model which result in a stationary environment:

$$y_n = \boldsymbol{\theta}_o^T\boldsymbol{x}_n + \eta_n$$

Explain why this results in a stationary environment.

Exercise 5.2.2

Run the code associated with this exercise. The code will reproduce figure 6.8 in the book. Inspect the code and identify the elements that generate the data and runs the algorithms.

Exercise 5.2.3

Investigate the different parameters of the algorithms (filter size, step size, forget factor, regularization), and study their effects on convergence speed and error floor. Discuss the results.

5.3 Adaptive filtering simulations in non–stationary environment

In this exercise we will simulate a non–stationary environment and create convergence curves. This exercise corresponds to example 6.2, and problem 6.21 in the book. Make sure to read both the example 6.2 and problem 6.21 before completing this exercise.

Exercise 5.3.1

We consider the following model which result in a non-stationary environment:

$$y_n = \boldsymbol{\theta}_{o,n}^T \boldsymbol{x}_n + \eta_n$$
$$\boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n$$

Explain why this results in a non–stationary environment.

Exercise 5.3.2

Run the code associated with this exercise. The code will reproduce figure 6.9 in the book. Inspect the code and identify the elements that generate the data and runs the algorithms.

Exercise 5.3.3

Investigate the different parameters of the algorithms (filter size, step size, forget factor, regularization), and study their effects on convergence speed and error floor. Make sure to reproduce figure 6.10 in the book. Discuss the results.

5.4 Adaptive filtering on audio

This exercise will demonstrate the application of an adaptive Wiener filter on real data, where there is piece of audio contaminated by noise, and then the adaptive Wiener filter is used to suppress the noise. It is a repetition and expansion of the audio exercise from last week, but now with RLS.

Exercise 5.4.1

Use and inspect the code for exercise 5.4.1. Determine the input signal, the output signal, and desired signal. Is the clean audio the filter output?

Try out RLS and experiment with the parameters of RLS. Listen to the sound, and experiment with the parameters for LMS, NLMS and RLS and different filter lengths. Is the results satisfactory (Listen to the sounds)? Discuss pros and cons of RLS compared to LMS and NLMS. Which algorithm performs best in terms of MSE? Which in terms of audible quality?

Exercise 5.4.2

Use and inspect the code for exercise 5.4.2. The filter will now handle a non-stationary noise source. Go through the same steps as the previous exercise and consider the same questions.

HINTS

For exercise 5.1.1

Ridge regression is treated in sec 3.8 and 6.6 in the book.

For exercise 5.1.6

You need to establish that $\Phi_n^{-1}\boldsymbol{x}_n = \boldsymbol{k}_n$, and use $e_n = y_n - \boldsymbol{x}^T\boldsymbol{\theta}_{n-1}$ in order to get the final update rule.