02471 Machine Learning for Signal Processing

Adaptive Linear Filtering with LMS

Tommy Sonne Alstrøm

Cognitive Systems section

DTU Compute
Department of Applied Mathematics and Computer Science

Outline

- Course admin
- Last week review
- LMS and NLMS
- Justification of LMS
- Affine Projection Algorithm and normalized LMS
- Steady-state convergence analysis of LMS
- Next Week

Material: ML 2.6, 5.1–5.5.1, (skip 5.3.1), 5.6, 5.9.

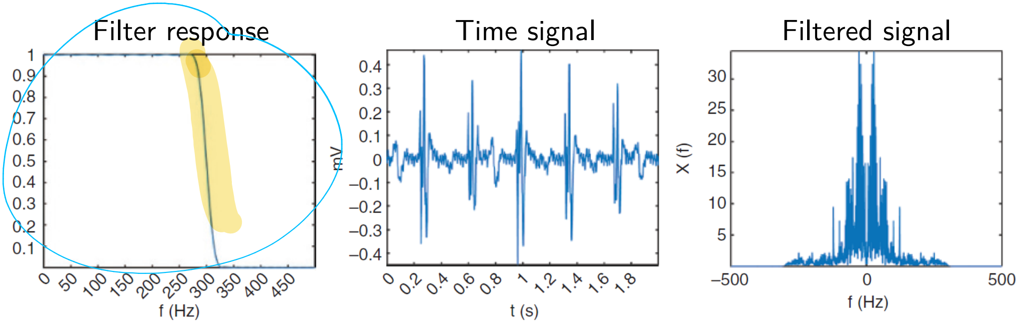
Administrative notes

- Problem set 1 second round, Sunday 6/10, pass/no pass.
- Problem set 2 first deadline, Sunday 27/10, pass/no pass.
  - Problem 2.1, 2.2, 2.4, and 2.5 should be solvable now.
- Feedback from last week
  - If lecturing lasts for 2 hours, it is better to have 2 times 10 min breaks.
  - Create a list of symbols used throughout the course (will create curated overleaf).
- Feedback from you is a critical component for improving both the course and my teaching.
- Type of feedback
  - Mention one thing that worked?
  - Mention one thing should be improved (both in current lecture and last weeks exercise)?
  - Mention one thing you would change if you gave the lecture.

Last week review

Two approaches to filter design

Design filter with specific frequency response:



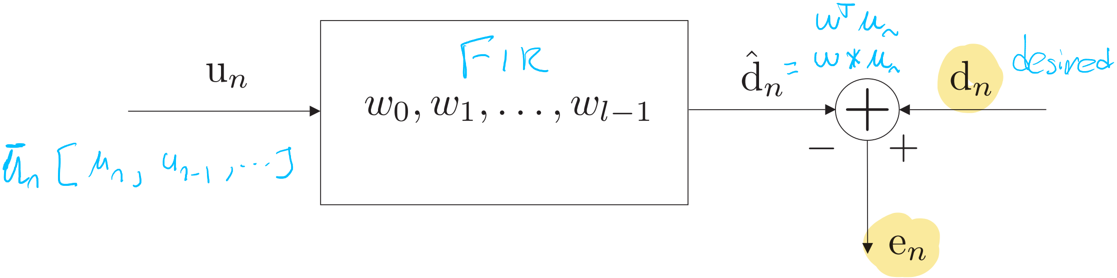
Idea:

Create a filter as a learning problem, ie. one that minimized a cost function.

Last week review

Linear filtering as learning problem

Filtering as a learning problem



The filter weights $\boldsymbol{\theta}$ (denoted as $\boldsymbol{w} = [w_0, w_1, \cdots, w_{l-1}]^T$ to indicate a linear filter) are learned such that the error is minimized

$$e_n = d_n - \hat{d}_n$$

If we choose to minimize the mean squared error, the filter is called a Wiener filter.

To use such filter, we need to specify a desired signal, d_n.

The normal equations

4.1

Minimizing the mean squared error $J(\boldsymbol{\theta}) = \mathbb{E}\left[(\mathrm{d}_n - \hat{\mathrm{d}}_n)^2\right]$ results in the normal equations

Normal Equations

$$\Sigma_u \boldsymbol{\theta}_* = \boldsymbol{r}_{du} \quad (\boldsymbol{r}_{du} \text{ is denoted } \boldsymbol{p} \text{ in the book})$$

Cross-cor

$$\boldsymbol{p} = [r_{du}(0)\ r_{du}(1)\ \cdots\ r_{du}(l-1)]^T$$

Covar./ Auto-corr

$$\Sigma_u = \begin{bmatrix} r_u(0) & r_u(1) & \cdots & r_u(l-1) \\ r_u(1) & r_u(0) & \cdots & r_u(l-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_u(l-1) & r_u(l-2) & \cdots & r_u(0) \end{bmatrix}$$

Wiener filter solution

$$\boldsymbol{\theta}_* = \Sigma_u^{-1} \boldsymbol{r}_{du}$$

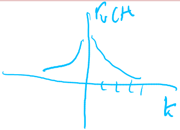
$(X^T X)^{-1} X^T y$

Stochastic processes

Auto-correlation and cross-correlation expressions for WSS processes (unnormalized)

$$\mu_n = \mu \quad \text{(constant mean)}$$

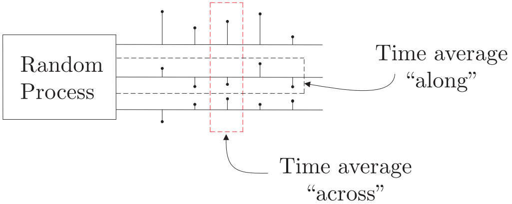
$$r_u(k) = \mathbb{E}[\mathrm{u}_n \mathrm{u}_{n-k}]$$ → Func. of k



$$r_{uv}(k) = \mathbb{E}[\mathrm{u}_n \mathrm{v}_{n-k}]$$

These are unnormalized, see `https://en.wikipedia.org/wiki/Autocorrelation`

Additionally, for mean-ergodic and covariance-ergodic processes



$$\mu_u = \frac{1}{N}\sum_{n=1}^{N} u_n$$

$$r_{uv}(k) = \frac{1}{N}\sum_{n=0}^{N-1} u(n)v(n-k), \quad k = 0, 1, \cdots, N-1$$

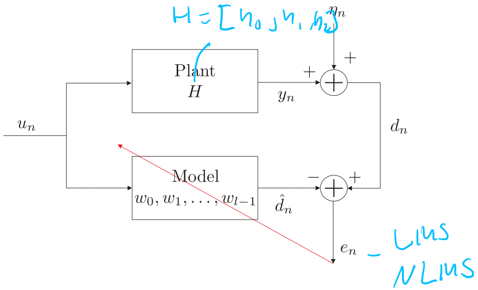
$$r'_{uv}(k) = \frac{1}{N-k}\sum_{n=k}^{N-1} u(n)v(n-k), \quad k = 0, 1, \cdots, N-1$$

e x u, 2

Typical applications

System identification

Goal: Model the impulse response of H



We have access to:

- The input signal u_n, and the noisy output d_n

Other applications

- Interference cancellation
- Echo cancellation
- Channel Equalization
- EEG signal analysis
- ...

Common for all: usually the signals are not wide-sense stationary but may be locally wide-sense stationary.

Last week summary

- Instead of designing filters by hand, we can train the filters from data. If we use a LTI filter and minimize the MSE the filter is called Wiener filter (or Linear filtering).
- We apply theory from stochastic processes to analyze and design wiener filters.
- We will in general be working with wide-sense stationary and ergodic processes. That means we can estimate the mean and correlation functions using only “one” realization (example). This makes the system applicable to real data.
- Many of the applications require adaptive filtering / online learning to be useful in practice.

LMS and NLMS

Adaptive filtering problem statement

- The signal statistics are slowly changing, so we need to change our filter weights as well.
- We need to have access to second–order statistics.
- In practice we rarely have access to second–order statistics, so we will focus on algorithm that can learn the statistics iteratively using data.

The Least-Mean-Squares Algorithm

The LMS algorithm – algorithm 5.1 in the book

- **Initialize**
  - $\boldsymbol{\theta}_{-1} = \mathbf{0} \in \mathbb{R}^l$ filter length
  - Select the value of μ – step size
- **For** $n = 0, 1, \cdots$, **Do**
  - $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$ target – output , $\bar{x}_n = [x_n, x_{n-w} \cdots x_{n-l+1}]$
  - $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu e_n \boldsymbol{x}_n$ – vector ??

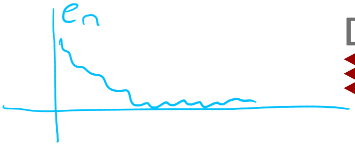
    scalar
- **End For**

Parameters:

μ is the step size.

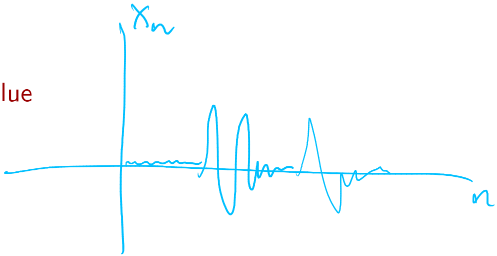
l is a filter length.

The Normalized Least-Mean-Square Algorithm 5.2



NLMS

- **Initialize**
  - $\boldsymbol{\theta}_{-1} = \mathbf{0} \in \mathbb{R}^l$
  - Select the value of $0 < \mu < 2$, and a small δ value
- **For** $n = 0, 1, \cdots,$ **Do**
  - $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$
  - $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \frac{\mu}{\boldsymbol{x}_n^T \boldsymbol{x}_n + \delta} e_n \boldsymbol{x}_n$
- **End For**



Q what is the role

power of the signal

Parameters:

μ is the step size.

l is a filter length.

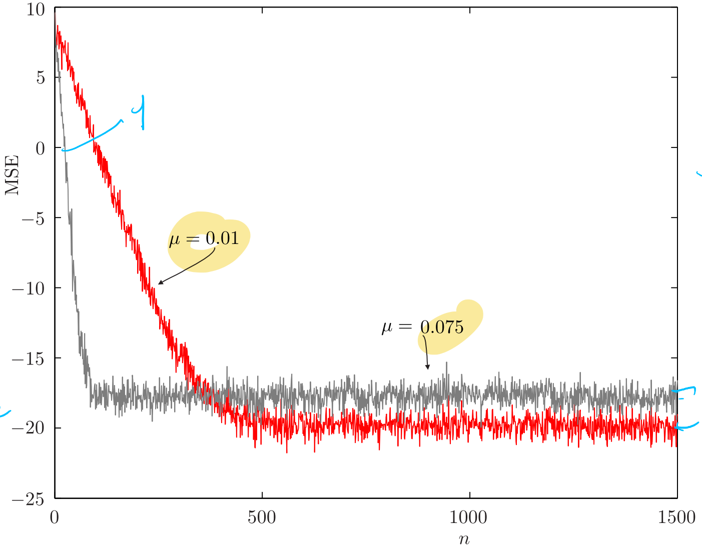
Convergence rate LMS

dB

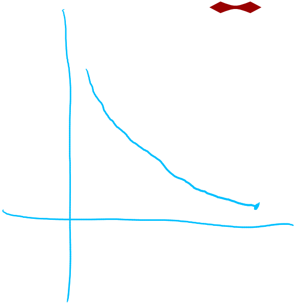
$10 \cdot \log_{10}(e)$

10^{-1}

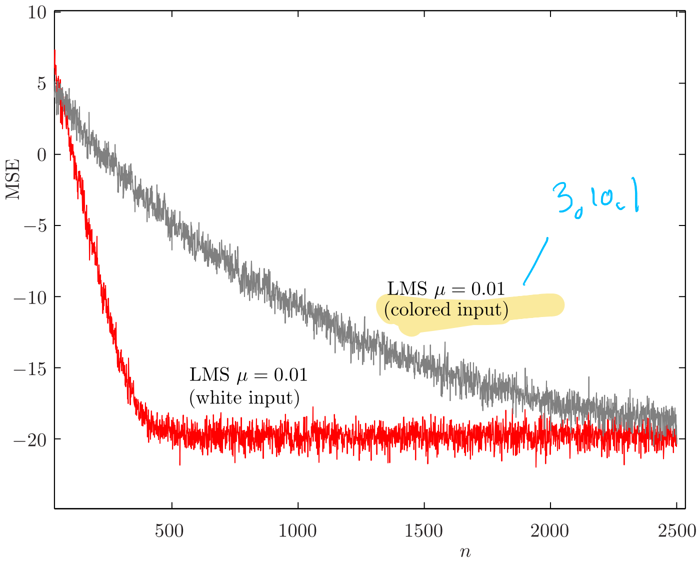
10^{-2}



why?



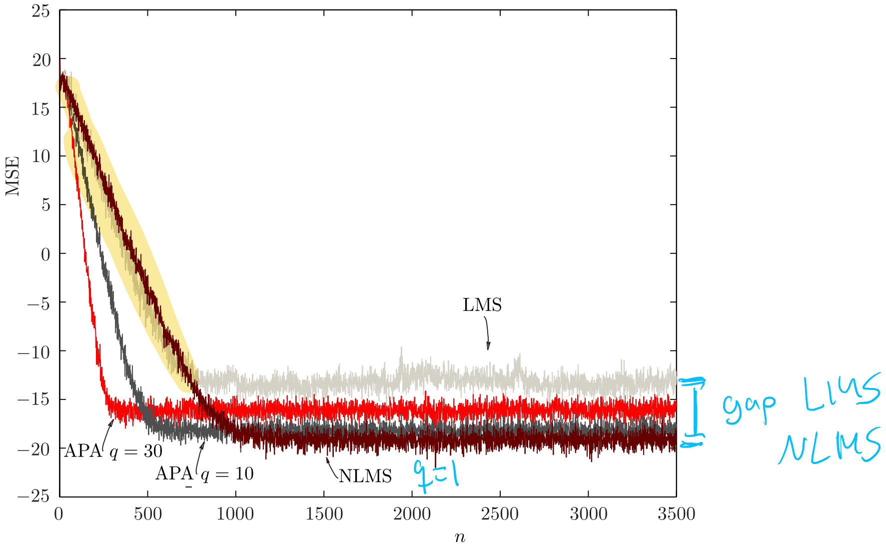
Convergence rate LMS



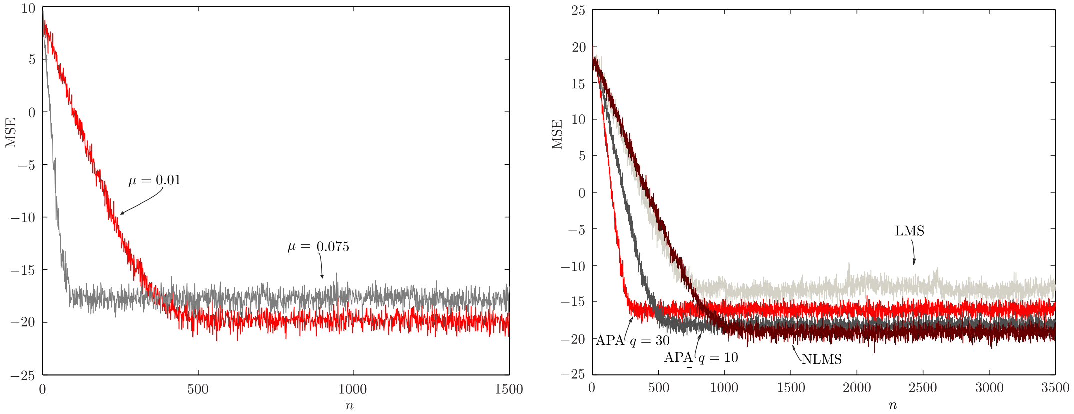
3, 10, 1

$\Sigma = $

Convergence rate NLMS



Convergences



Discuss pros and cons of learning curves - can you think if applications appropriate of the difference curves?

Summary

- Many applications require adaptive algorithms (also called sequential learning, or online learning in machine learning).
- The value of μ affects the performance greatly (need theory to guide us).
- NLMS is generally preferred over LMS.

Justification of LMS

Recipe

Steps involved in developing LMS

1. Develop iterative scheme for when statistics are known ($r_x(k)$ and $r_{dx}(k)$ are known).
2. Convert iterative scheme to data-driven approach ($r_x(k)$ and $r_{dx}(k)$ are estimated).
3. Ensure algorithm has agility (can adapt to changing environment).
4. Perform convergence analysis for steady state.

How did we arrive at LMS?

We want to develop an iterative scheme

adjustment

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \boxed{\mu_i \Delta} \boldsymbol{\theta}^{(i)}, \quad \mu_i > 0, \quad i > 0$$

such that

$$J(\boldsymbol{\theta}^{(i)}) < J(\boldsymbol{\theta}^{(i-1)})$$

where $J(\cdot)$ is the differentiable cost function.

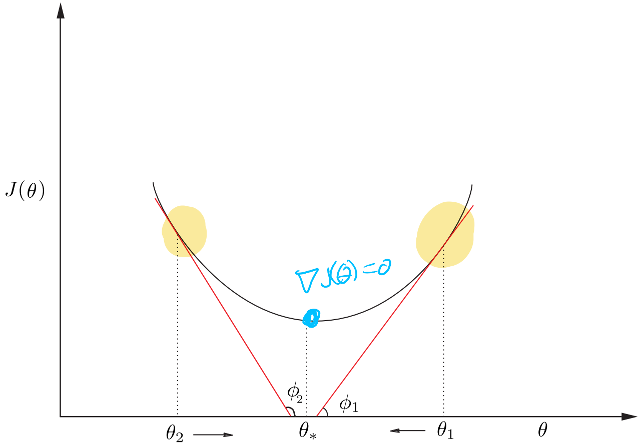
1–dimensional example

Iterative scheme:

$-\nabla J(\theta)$

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_i \Delta\boldsymbol{\theta}^{(i)}$$

If $\Delta\boldsymbol{\theta}^{(i)}$ is the negative gradient direction we will end up at the minima.



From gradient descent to LMS

Gradient Descent

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i \nabla J(\boldsymbol{\theta}^{(i-1)})$$

From last week, we found the gradient, $\nabla J(\boldsymbol{\theta}) = \Sigma_x \boldsymbol{\theta} - \boldsymbol{p}$. week 3

$r_x(k)$ $r_{dx}(k) - E[x_n d_{x-n}]$

$E[x_n x_{x-n}]$

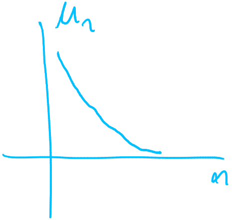
At the optimum, $\nabla J(\boldsymbol{\theta}) = \Sigma_x \boldsymbol{\theta} - \boldsymbol{p} = \mathbf{0}$.

Issue: this only works if we have access to second order statistics.

Use Robbins–Monro scheme – stochastic approximation

If we have a cost function defined w.r.t. an expectation, we can apply the Robbins-Monro algorithm

Robbins-Monro algoritm



$$J(\boldsymbol{\theta}) = \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}, \mathbf{y}, \mathbf{x})]$$

$$\nabla_{\theta} J(\boldsymbol{\theta}) = \cancel{\mathbb{E}}[\nabla \mathcal{L}(\boldsymbol{\theta}, \mathbf{y}, \mathbf{x})]$$

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} - \mu_n \nabla \mathcal{L}\left(\boldsymbol{\theta}_{n-1}, \boldsymbol{y}_n, \boldsymbol{x}_n\right)$$

$$\sum_n \mu_n^2 < \infty, \quad \sum_n \mu_n \longrightarrow \infty : \quad \text{convergence conditions.}$$

$\mu_{n+1} < \mu_n$ $\qquad$ $\mu_n = \frac{1}{n}$

Derivation of the update

MSE

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\mathrm{e}^2\right] \qquad \mathrm{e} = \mathrm{y} - \boldsymbol{\theta}^T\mathrm{x} \quad \Big| \quad d - \hat{d}$$

$\theta^T x$

$$\nabla_{\theta} J(\boldsymbol{\theta}) = \mathbb{E}\left[\nabla_{e}\mathrm{e}^2\right]$$

Chain rule

$$= \mathbb{E}[2\mathrm{e}\nabla_{\theta}\mathrm{e}]$$

$$= -2\mathbb{E}\left[\underbrace{(\mathrm{y} - \boldsymbol{\theta}^T\mathrm{x})}_{e}\mathrm{x}\right] = -2\mathbb{E}[e \cdot x]$$

$-2 \Rightarrow$ absorbed in μ

The Robbins–Monro scheme was

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \nabla \mathcal{L}\left(\boldsymbol{\theta}_{n-1}, \boldsymbol{y}_n, \boldsymbol{x}_n\right)$$

So we get

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \boxed{(y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n)\boldsymbol{x}_n}$$

Final result

The Robbins-Monro update

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_n(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta}^{(i-1)})\boldsymbol{x}_n$$

last slide

To gain agility to track non-stationary environments, we use a constant step size.

LMS update

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \boxed{\mu}(y_n - \boldsymbol{x}_n^T\boldsymbol{\theta}^{(i-1)})\boldsymbol{x}_n = \theta^{(i-1)} + \mu \cdot e \cdot x_n$$

fixed step size θ_{n-1}

✓ LMS

Summary

SGD

- The LMS algorithm is a stochastic gradient algorithm that estimates second-order statistics from data.

$\mu_n \to \mu$

- To obtain agility, we use a fixed step-size. This means convergence guaranties provided by Robbins-Monro are invalid. ½
  - Care must be taken for choosing the step-size (sec 5.3).
  - We can use the eigenvalues of Σ_x to guide our choice of step-size.
  - Large spread in eigenvalues means slow convergence (sec 5.3).

  LA

- LMS is a flexible algorithm that requires tuning of the step-size.

Affine Projection Algorithm and normalized LMS

The Affine Projection Algorithm (APA)

The APA algorithm minimizes the following problem

$$\boldsymbol{\theta}_n = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}\|^2$$

$$\text{s.t.} \quad \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} = y_{n-i}, \quad i = 0, 1, \ldots, q-1$$

DESIGN

LA : $x_n^T \theta = y_n$ $q=1$ | assume FIR $l \geq q$

Lagrange multiplier

$$\begin{array}{ll} \text{minimize} & J(\boldsymbol{\theta}) \\ \text{subject to} & f_i(\boldsymbol{\theta}) = 0, \quad i = 1, 2, \ldots, m \end{array}$$

This problem can be solved by optimizing a modified function, usually called the Lagrangian

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = J(\boldsymbol{\theta}) - \sum_{i=1}^{m} \boxed{\lambda_i} f_i(\boldsymbol{\theta})$$

Lag. Mult

The saddle point of this function is then found by solving $\nabla L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0}$

Deriving updates for APA I

$$\boldsymbol{\theta}_n = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}\|^2$$

$$\text{s.t.} \quad \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} = y_{n-i}, \quad i = 0, 1, \dots, q-1$$

$$L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \overbrace{(\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1})^T (\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1})}^{J(\theta)} + \sum_{i=0}^{q-1} \lambda_i \left(y_{n-i} - \overset{x^T_{n-i}\theta}{\boldsymbol{\theta}^T \boldsymbol{x}_{n-i}} \right)$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = 2\boldsymbol{\theta} - 2\boldsymbol{\theta}_{n-1} - \sum_{i=0}^{q-1} \lambda_i \boldsymbol{x}_{n-i}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} L(\boldsymbol{\theta}, \boldsymbol{\lambda}) = \mathbf{0} \Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \frac{1}{2} \sum_{i=0}^{q-1} \lambda_i \boldsymbol{x}_{n-i}$$

$$X_n := [\boldsymbol{x}_n, \dots, \boldsymbol{x}_{n-q+1}]^T, \qquad \boldsymbol{\lambda}^T := [\lambda_0, \dots \lambda_{q-1}], \qquad \boldsymbol{y}_n^T := [y_n, \dots, y_{n-q+1}]$$

Deriving updates for APA II

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \frac{1}{2} X_n^T \boldsymbol{\lambda}$$
$$X_n \boldsymbol{\theta} = \boldsymbol{y}_n$$

Solve the two equations for the two unknowns:

$$\begin{aligned} X_n \left(\boldsymbol{\theta}_{n-1} + \frac{1}{2} X_n^T \boldsymbol{\lambda} \right) = \boldsymbol{y}_n \quad &\Leftrightarrow \\ \frac{1}{2} X_n X_n^T \boldsymbol{\lambda} = \boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1} \quad &\Leftrightarrow \\ \frac{1}{2} \boldsymbol{\lambda} = \left(X_n X_n^T \right)^{-1} (\boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1}) &\Rightarrow \\ \boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} (\boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1}) & \end{aligned}$$

Deriving updates for APA III

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} (\boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1})$$

$$\boldsymbol{e}_n := \boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1}$$

prob

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + X_n^T \left(X_n X_n^T \right)^{-1} \boldsymbol{e}_n$$

APA sol. (no step size

The update algorithm then becomes

$$\boldsymbol{e}_n = \boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1}$$

sol to prob

$$\boldsymbol{\theta} = \boldsymbol{\theta}_{n-1} + \mu X_n^T \underline{\left(\delta I + X_n X_n^T \right)^{-1}} \boldsymbol{e}_n$$

introduce step size

For $q = 1$

NLMS

we hit Y_n iff $\mu = 1$

$$e_n = y_n - \boldsymbol{x}_n \boldsymbol{\theta}_{n-1}$$

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \boxed{\frac{\mu}{\boldsymbol{x}_n^T \boldsymbol{x}_n + \delta}} e_n \boldsymbol{x}_n$$

The Affine Projection Algorithm (APA) and NLMS

APA and NLMS

The APA algorithm minimizes the following problem

$$\begin{aligned}\boldsymbol{\theta}_n &= \arg\min_{\theta} \|\boldsymbol{\theta} - \boldsymbol{\theta}_{n-1}\|^2 \\ \text{s.t.} \quad & \boldsymbol{x}_{n-i}^T \boldsymbol{\theta} = y_{n-i}, \quad i = 0, 1, \ldots, q-1\end{aligned}$$

Define the following matrix

$$X_n^T := [\boldsymbol{x}_n, \ldots, \boldsymbol{x}_{n-q+1}]$$

Then, by using Lagrange multipliers, we get the following update, $(0 \leq \mu \leq 2)$

$$\begin{aligned}\boldsymbol{e}_n &= \boldsymbol{y}_n - X_n \boldsymbol{\theta}_{n-1} \\ \boldsymbol{\theta} &= \boldsymbol{\theta}_{n-1} + \mu X_n^T \left(\delta I + X_n X_n^T\right)^{-1} \boldsymbol{e}_n\end{aligned}$$

For $q = 1$, we call the algorithm nlms.

Steady-state convergence analysis of LMS

Convergence analysis of coefficients

5.5.1

Define the coefficient error vector as

$$c_n := \boldsymbol{\theta}_n - \boldsymbol{\theta}_*$$

$$\begin{bmatrix} r_x(0) & r_x(1) & \cdots \\ \vdots & & \end{bmatrix}$$

Analyzing convergence in the mean,

$$\mathbb{E}\left[\mathbf{c}_n\right] \to 0$$

as $n \to \infty$ we get the following results

eigval(Σ_x)

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

Σ_x eq 1

Y: 1% + 1%
N: 2%
??: 97

However analyzing the covariance matrix, $n \to \infty$

$$\Sigma_{c,n} := \mathbb{E}\left[\mathbf{c}_n \mathbf{c}_n^T\right]$$

$$0 < \mu < \frac{2}{\operatorname{trace}\{\Sigma_x\}} = \frac{2}{l \cdot \hat{r}_x(0)}$$

eq. 2

Y: 5%
N: 1%
??: 94%

Convergence analysis of error

ML 5.5.1

Define the misalignment as

Misalignment

$$\mathcal{M} := \frac{J_{\text{exc}}}{J_{\min}}$$

$$\frac{\text{excess sol MSE}}{\text{Best sol MSE}}$$

Analyzing the mean squared error behavior, we get the error at step n:

$$J_n := \mathbb{E}\left[\mathrm{e}_n^2\right] = \cdots (\text{ML eq } 5.46 - 5.47) \cdots = J_{\min} + \operatorname{trace}\left\{\Sigma_x \Sigma_{c,n-1}\right\}$$

Excess MSE at time instant n

$$J_{\text{exc},n} = \operatorname{trace}\left\{\Sigma_x \Sigma_{c,n-1}\right\} : \quad \text{excess MSE at time instant } n$$

Expected misalignment, for small values of μ

$$\mathcal{M} \simeq \frac{1}{2}\mu \operatorname{trace}\left\{\Sigma_x\right\} : \quad \text{misadjustment}$$

is $\mathrm{Tr}(\Sigma_x)$ constant
Y; so!
N:

1st + 2nd order stats,

Summary

- The convergence analysis involve a fair amount of assumptions and is lengthy to derive.
- You are expected to be familiar with the material but can skip any derivation that the book defers to "Problem X derives this".
- The convergence analysis provides theoretical foundation for choosing the step-size μ. We need agility in the algorithm, but also low misalignment. This is a trade-off.

Lecture summary

- Adaptive algorithms, such as LMS and NLMS can be used to estimate second-order statistics from data.

  ML

- The algorithms needs to be carefully tuned per application basis, as there is trade-offs between agility and total error.
- In general, NLMS is considered more stable and better than LMS.
- These algorithms are deployed in many real-world problems to enable filtering in a changing environment.

Next week

2022: prob 2.4

Material: ML 5.12, 6.1–6.3 (until "The LS estimator is BLUE"), 6.6–6.8, 6.12.

- Adaptive filtering continued
- Tracking performance in non-stationary environment
- Recursive Least-Squares (RLS) adaptive algorithm