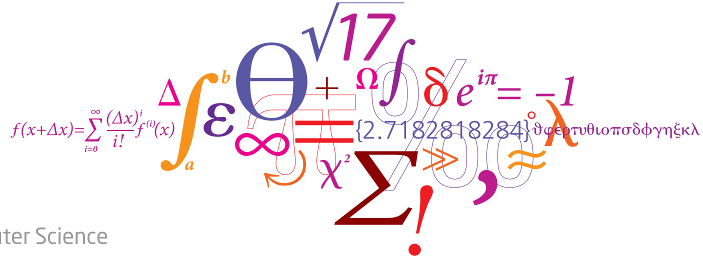
02471 Machine Learning for Signal Processing

Adaptive Linear Filtering with RLS

Tommy Sonne Alstrøm

Cognitive Systems section



DTU Compute
Department of Applied Mathematics and Computer Science

Outline

- Course admin
- Last week review
- Recursive Least Squares (RLS)
- Stationary simulation model and steady-state convergence analysis of LMS
- Non-stationary simulation model and convergence analysis
- Derivation of the RLS algorithm
- Relation between RLS and Newton's method
- Next Week

Material: ML 5.12, 6.1–6.3 (until "The LS estimator is BLUE"), 6.5 (only page 265), 6.6–6.8, 6.12. Most important sections are 6.6, 6.8, 6.12 and Algorithm 6.1.

Feedback

- Feedback from last week:
    - Solutions to exam sets will be uploaded on a running bases with a one week delay.
- 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.

Tentative course outline by lecture module

Week	Topic	Material (ML)
35	Digital signal processing, probability theory, machine learning	1.1–2.3
36	Matrix derivatives, constrained optimization, parameter estimation	3.1–3.3, 3.5, 3.8–3.11, A.1–A.2, C.1–C.2
37	Linear filtering	2.4, 4.1–4.3, 4.5–4.7
38	Adaptive filtering, LMS	2.6, 5.1–5.5.1, 5.9, 5.12
39	Adaptive filtering, RLS	6.1–6.3, 6.5–6.8, 6.12
40	Sparsity aware learning	8.2, 8.10.1–8.10.2, 9.1–9.5, 9.9
41	Shrinkage algorithms, Time-frequency analysis	10.1–10.2, 10.5–10.6
43	Dictionary learning, ICA, k-svd	2.5, 19.1–19.3, 19.5–19.7
44	Bayesian Modeling and EM	11.2, 12.1–12.2, 12.4–12.5, 12.10
45	State-space models, Hidden Markov models	15.1–15.3.1, 15.7, 16.4–16.5
46	State-space models, Kalman filter	4.9–4.9.1, 4.10, 17.3
47	Kernel methods, Kernel ridge regression	11.1–11.5, 11.7
48	Kernel methods, Support vector regression	11.8

Problem set 1 and 2 hand–ins

All problem sets are compulsory hand–ins. They will not be graded, but must be approved in order to attend the exam. The problem sets works as formative feedback and is your opportunity to get written feedback on your work.

Problem set 1, late submissions due 6/10

Material from week 1 as well as crucial prerequisites.

Problem set 2, due 27/10

Material from week 2–7: Parameter estimation, Linear filtering (Wiener), Time frequency analysis, Adaptive filtering, Compressed sensing. The problem set is meant as a half way test.

Last week review

Last week review

The Least-Mean-Squares Algorithm

The LMS algorithm – algorithm 5.1 in the book

```
• Initialize
    • $\boldsymbol{\theta}_{-1} = \mathbf{0} \in \mathbb{R}^l$
    • Select the value of $\mu$
• For $n = 0, 1, \cdots,$ Do
    • $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$
    • $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu e_n \boldsymbol{x}_n$
• End For
```

Parameters:

μ is the step size.

l is a filter length.

Last week review

The Affine Projection Algorithm (APA)

The APA algorithm minimizes the following problem

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

Define the following matrix

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

Then, by using Lagrange multipliers, we got the following update

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

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

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

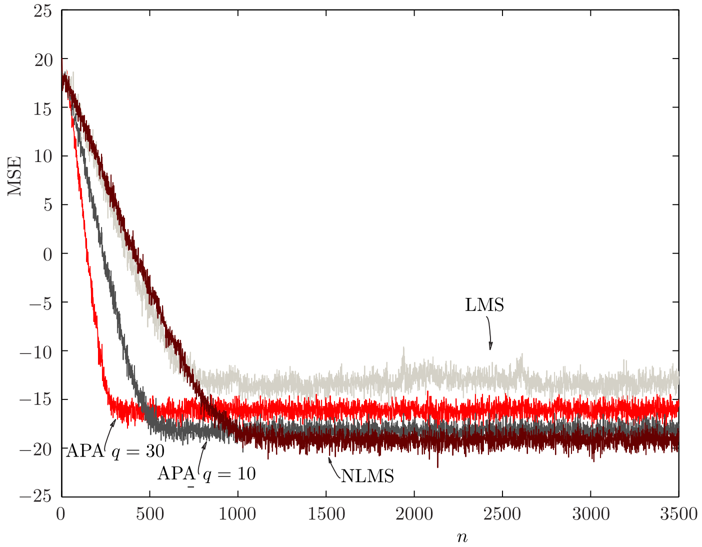
Parameters:

μ is the step size.

l is a filter length.

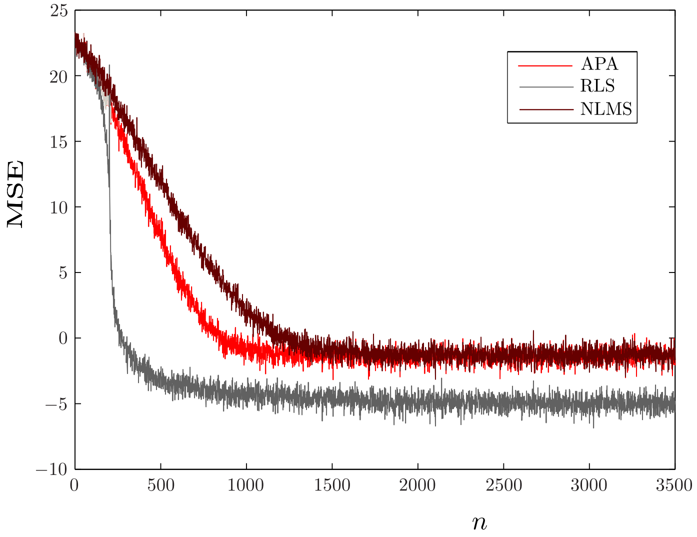
Last week review

Convergence rates



Recursive Least Squares (RLS)

RLS teaser – stationary environment convergence rate



The Recursive Least-Squares – algorithm 6.1

RLS

```
• Initialize
    • $\boldsymbol{\theta}_{-1} = \mathbf{0} \in \mathbb{R}^l$;   any other value is also possible.
    • $P_{-1} = \lambda^{-1} I$;   $\lambda > 0$ a user-defined variable
    • Select $\beta$ close to 1.
• For $n = 0, 1, \cdots,$ Do
    • $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$
    • $\boldsymbol{z}_n = P_{n-1} \boldsymbol{x}_n$
    • $\boldsymbol{k}_n = \frac{\boldsymbol{z}_n}{\beta + \boldsymbol{x}_n^T \boldsymbol{z}_n}$
    • $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \boldsymbol{k}_n e_n$
    • $P_n = \beta^{-1} P_{n-1} - \beta^{-1} \boldsymbol{k}_n \boldsymbol{z}_n^T$
• End For
```

Parameters:

β forget factor.

λ is regularization.

Where does RLS come from

Exponentially weighted least-squares

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

RLS solves the minimization problem at observation n:

$$\boldsymbol{\theta}_n = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \beta, \lambda)$$

Have you seen such cost function before? Discuss the roles of β, λ, and n

Summary

- The RLS algorithm is obtained by a modified Ridge regression cost function.
- RLS obtains superior convergence rate compared to LMS (or so it seems, details to follow).
- The algorithm is far more complex and computationally demanding so it is not all good.

Stationary simulation model and steady-state convergence analysis of LMS

How do we simulate stationary environment?

Steady state environment

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

- $\boldsymbol{\theta}_o \in \mathbb{R}^l$ Optimal filter weights.
- η_n is zero mean Gaussian i.i.d (white noise).

LMS summary

Defining the coefficient error vector as $c_n := \boldsymbol{\theta}_n - \boldsymbol{\theta}_*$ and require convergence on the diagonal of the covariance matrix $\Sigma_{c,n} := \mathbb{E}\left[\mathbf{c}_n \mathbf{c}_n^T\right]$ we got

LMS Step size bounds

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

Misalignment and excess MSE

$$J_n = J_{\min} + J_{\text{exc}}$$

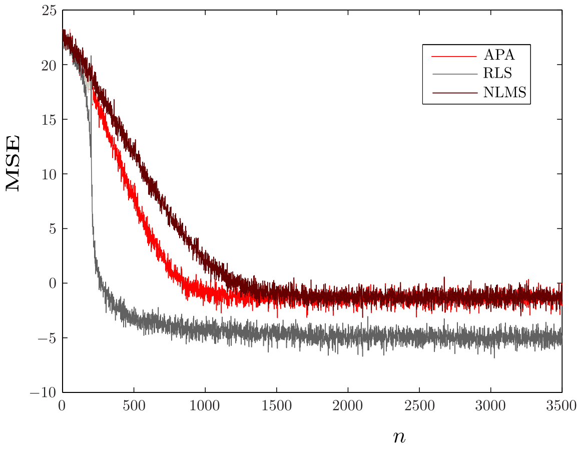
$$\mathcal{M} := \frac{J_{\text{exc}}}{J_{\min}}$$

Excess MSE and misadjustment for LMS at time instant n

$$J_{\text{exc,n}} = \operatorname{trace}\{\Sigma_x \Sigma_{c,n-1}\}$$

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

Convergence rates of RLS – stationary environment (example 6.1)



Data generation model:

$$y_n = \boldsymbol{\theta}_o^T \boldsymbol{x}_n + \eta_n \,, \; \boldsymbol{\theta} \in \mathbb{R}^{200} \; \eta_n \sim \mathcal{N}(0, 0.01I)$$

For APA: $\mu = 0.2$, $\delta = 0.001$, $q = 30$.

For RLS: $\beta = 1$, $\lambda = 0.1$.

For NLMS: $\mu = 1.2$, $\delta = 0.001$.

APA and NLMS parameters chosen so algorithms converge to same noise floor.

RLS parameters chosen for optimal noise floor.

You will create these simulations in the exercise today.

Non-stationary simulation model and convergence analysis

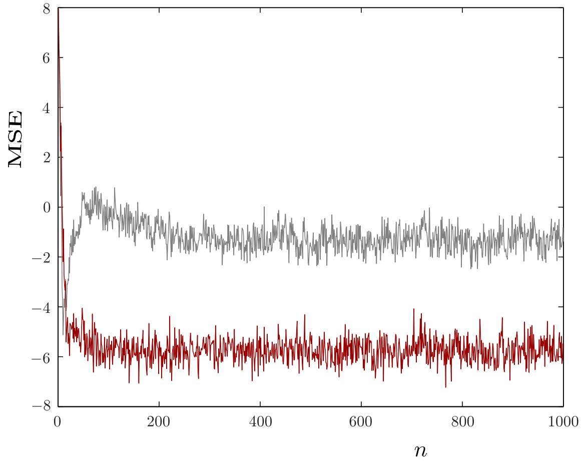
How do we simulate non-stationary environment?

Non-stationary environment

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

- $\alpha < 1$ is the autoregressive coefficient (memory).
- $\boldsymbol{\omega}_n$ is zero mean Gaussian i.i.d with diagonal covariance (white noise).

Convergence rates of RLS, time–tracking (example 6.2)



Gray: RLS, Red: NLMS

Data generation model:

$$y_n = \theta_{o,n}^T \mathbf{x}_n + \eta_n,\ \boldsymbol{\theta} \in \mathbb{R}^5,\ \eta_n \sim \mathcal{N}(0, 0.01I)$$

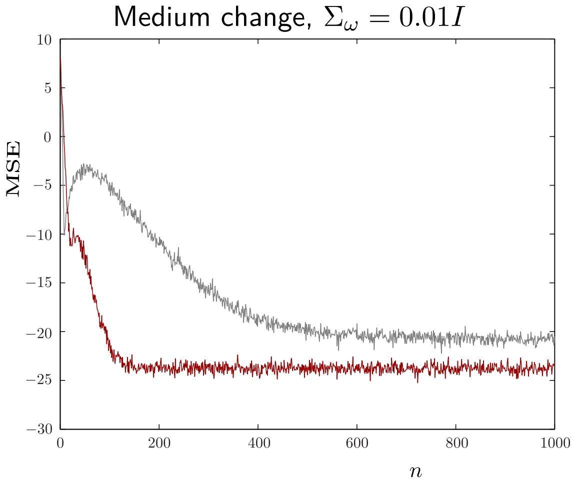
$$\theta_{o,n} = \alpha\theta_{o,n-1} + \boldsymbol{\omega}_n,\ \alpha = 0.97,\ \omega_n \sim \mathcal{N}(0, 0.1I)$$

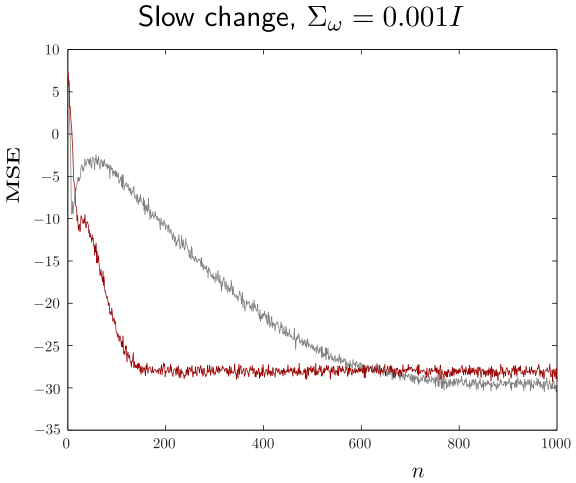
For RLS: $\beta = 0.995,\ \lambda = 0.001.$

For NLMS: $\mu = 0.5,\ \delta = 0.001.$

Parameters were selected for best performance.

Convergence rates of RLS, time–tracking





Gray: RLS, Red: NLMS

Convergence analysis of LMS under non-stationary environment

Stationary environment excess MSE

$$J_{\text{exc}} \simeq \frac{1}{2}\mu\sigma_\eta^2 \operatorname{trace}\{\Sigma_x\}$$

Non-stationary environment excess MSE

$$J_{\text{exc}} \simeq \frac{1}{2}\left(\underbrace{\mu\sigma_\eta^2 \operatorname{trace}\{\Sigma_x\}}_{\text{excess}} + \underbrace{\frac{1}{\mu}\operatorname{trace}\{\Sigma_\omega\}}_{\text{lag}}\right)$$

Is there anything in particular we should consider?

Excess MSE for the adaptive algorithms (table 6.1)

If we analyze the error rates under the presented non-stationary model, we obtain the following at steady-state (assumptions and limitations apply!!)

Algorithm	Excess MSE (l denotes filter length)
LMS	$\frac{1}{2}\mu\sigma_\eta^2 \operatorname{trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1} \operatorname{trace}\{\Sigma_\omega\}$
APA	$\frac{1}{2}\mu\sigma_\eta^2 \operatorname{trace}\{\Sigma_x\} \mathbb{E}\left[\frac{q}{\lVert x\rVert^2}\right] + \frac{1}{2}\mu^{-1} \operatorname{trace}\{\Sigma_x\} \operatorname{trace}\{\Sigma_\omega\}$
NLMS	$\frac{1}{2}\mu\sigma_\eta^2 \operatorname{trace}\{\Sigma_x\} \frac{1}{\sigma_x^2(l-2)} + \frac{1}{2}\mu^{-1} \operatorname{trace}\{\Sigma_x\} \operatorname{trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1} \operatorname{trace}\{\Sigma_\omega \Sigma_x\}$

With these results, we can find the optimal μ or β (how??), and e.g. obtain the following:

LMS/RLS expected error ratio

$$\frac{J_{\min}^{\mathrm{LMS}}}{J_{\min}^{\mathrm{RLS}}} = \sqrt{\frac{\operatorname{trace}\{\Sigma_x\} \operatorname{trace}\{\Sigma_\omega\}}{l \operatorname{trace}\{\Sigma_\omega \Sigma_x\}}}$$

Summary

- Many applications have time-changing environment that calls for adaptive algorithms
- RLS can have superior convergence speed compared to NLMS
- RLS is not always better though, it has higher computational complexity and can also diverge due to finite floating point precision.
- NLMS is very stable and the “safe” choice.

Derivation of the RLS algorithm

Minimizing the cost function

Exponentially weighted least-squares

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

1. Take the derivative wrt $\boldsymbol{\theta}$ and set to zero.
2. Create time-iterative updates, i.e on the form $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \Delta$

Minimizing the cost function

Exponentially weighted least-squares

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

The derivative is

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

Which leads to

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

Deriving the time-iterative updates

$$
\begin{aligned}
\boldsymbol{p}_n &= \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i y_i \\
&= \boldsymbol{x}_n y_n + \sum_{i=0}^{n-1} \beta^{n-i} \boldsymbol{x}_i y_i \\
\boldsymbol{p}_{n-1} &= \sum_{i=0}^{n-1} \beta^{n-1-i} \boldsymbol{x}_i y_i \\
\beta \boldsymbol{p}_{n-1} &= \sum_{i=0}^{n-1} \beta^{n-i} \boldsymbol{x}_i y_i \\
\boldsymbol{p}_n &= \beta \boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n
\end{aligned}
$$

Deriving the time-iterative updates

Using a similar approach for Φ_n, we arrive at

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

The goal was to find $\boldsymbol{\theta}_n = \Phi_n^{-1}\boldsymbol{p}_n$, thus we need to compute

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

We use Woodburry's inversion formula (in the exercise).

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

What is A, B, C and D?

Primary result

Inversion recursion, and Kalman gain k

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

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

For notational purposes we define $P_n = \Phi_n^{-1}$, and, for later, we rewrite $\boldsymbol{k}_n$, to get

$$\begin{aligned} P_n &= \beta^{-1}P_{n-1} - \beta^{-1}\boldsymbol{k}_n\boldsymbol{x}_n^T P_{n-1} \\ \boldsymbol{k}_n &= \left(\beta^{-1}P_{n-1} - \beta^{-1}\boldsymbol{k}_n\boldsymbol{x}_n^T P_{n-1}\right)\boldsymbol{x}_n \\ &= P_n\boldsymbol{x}_n \end{aligned}$$

RLS Summary

RLS minimized the exponentially weighted least-squares cost function

Exponentially weighted least-squares

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

Choose forget factor β and regularization λ

We have an iterative update formula for $\boldsymbol{\theta}_n$, meaning we can update $\boldsymbol{\theta}_n$ at each new observation $(y_n, \boldsymbol{x}_n)$, instead of solving the cost function anew.

Update formula

$$\begin{aligned}
\boldsymbol{\theta}_n &= \boldsymbol{\theta}_{n-1} + \boldsymbol{k}_n e_n \\
\boldsymbol{k}_n &= \left(\beta^{-1} P_{n-1} - \beta^{-1} \boldsymbol{k}_n \boldsymbol{x}_n^T P_{n-1}\right) \boldsymbol{x}_n \\
P_n &= \beta^{-1} P_{n-1} - \beta^{-1} \boldsymbol{k}_n \boldsymbol{x}_n^T P_{n-1} \\
e_n &= y_n - \boldsymbol{x}_n^T \boldsymbol{\theta}_{n-1}
\end{aligned}$$

Summary

- The RLS can efficiently be implemented in a iterative update manner instead of solving the cost function completely at every step.
- It can diverge with finite precision arithmetic, and more robust implementations exists.
- You will derive this algorithm completely in the first exercise, and they replicate the convergence plots.

Relation between RLS and Newton's method

Newton's iterative minimization method

Newton's iterative scheme

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \underbrace{\mu_i \left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}}_{\text{Stepsize}} \nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)$$

Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration i the deviation from the optimum value follows the following pattern

$$\ln\ln\frac{1}{\left\|\theta^{(i)} - \theta_*\right\|^2} \propto i: \quad \text{quadratic convergence rate}$$

$$\ln\frac{1}{||\theta^{(i)} - \theta_*||^2} \propto i: \quad \text{linear convergence rate}$$

Applying Newton's method to the mean squared error loss

$$J(\theta) = \frac{1}{2}\mathbb{E}\left[\left(\mathrm{y} - \theta^T\mathbf{x}\right)^2\right] = \frac{1}{2}\sigma_y^2 + \frac{1}{2}\boldsymbol{\theta}^T\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbb{E}[\mathbf{x}\mathrm{y}]$$

$$-\nabla J(\boldsymbol{\theta}) = \mathbb{E}[\mathbf{x}\mathrm{y}] - \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\boldsymbol{\theta} = \mathbb{E}\left[\mathbf{x}\left(\mathrm{y} - \mathbf{x}^T\boldsymbol{\theta}\right)\right] = \mathbb{E}[\mathbf{x}\mathrm{e}]$$

$$\nabla^2 J(\boldsymbol{\theta}) = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = \Sigma_x$$

Now apply newtons iteration scheme on observations

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \Sigma_x^{-1}\boldsymbol{x}_n e_n$$

Newton's iterative scheme

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \underbrace{\mu_i\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}}_{\text{Stepsize}}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)$$

Approximating the covariance matrix

From last slide

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n \Sigma_x^{-1} \boldsymbol{x}_n e_n$$

If we apply the following approximation (why does this make sense?)

$$\Sigma_x \simeq \frac{1}{n+1}\Phi_n = \left(\frac{1}{n+1}\lambda\beta^{n+1} I + \frac{1}{n+1}\sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^T \right)$$

Then we can obtain the RLS update formula for $\boldsymbol{\theta}_n$ by reading off the following coefficients:

$$\mu_n = \frac{1}{n+1}$$

$$\boldsymbol{k}_n = P_n \boldsymbol{x}_n$$

$$P_n = \left(\lambda\beta^{n+1} I + \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^T \right)^{-1}$$

Summary

- Newton's method is a second order method that obtains quadratic convergence speed.
- RLS can be derived using Newton's method, and thus reveals RLS is a method that uses the second order derivatives.
- This explains the superior convergence rate of RLS.

Lecture summary

- The recursive least squares (RLS) achieves quadratic convergence speed.
- RLS is computationally more expensive compared to LMS and APA.
- RLS will achieve superior steady state performance.
- For tracking performance, there are still tradeoffs to consider, and sometimes NLMS performance is better (and more robust).

Next week

Material: ML 8.2, 8.10.1–8.10.2, 9.1–9.5, 9.9

- Sparsity-aware learning.
- Alternatives to the ℓ_2 norm.
- Compressed sensing.
- Leads to dictionary learning (later).