APPENDIX Q:

MATRIX INVERSION LEMMA AND

WOODBURY'S IDENTITY

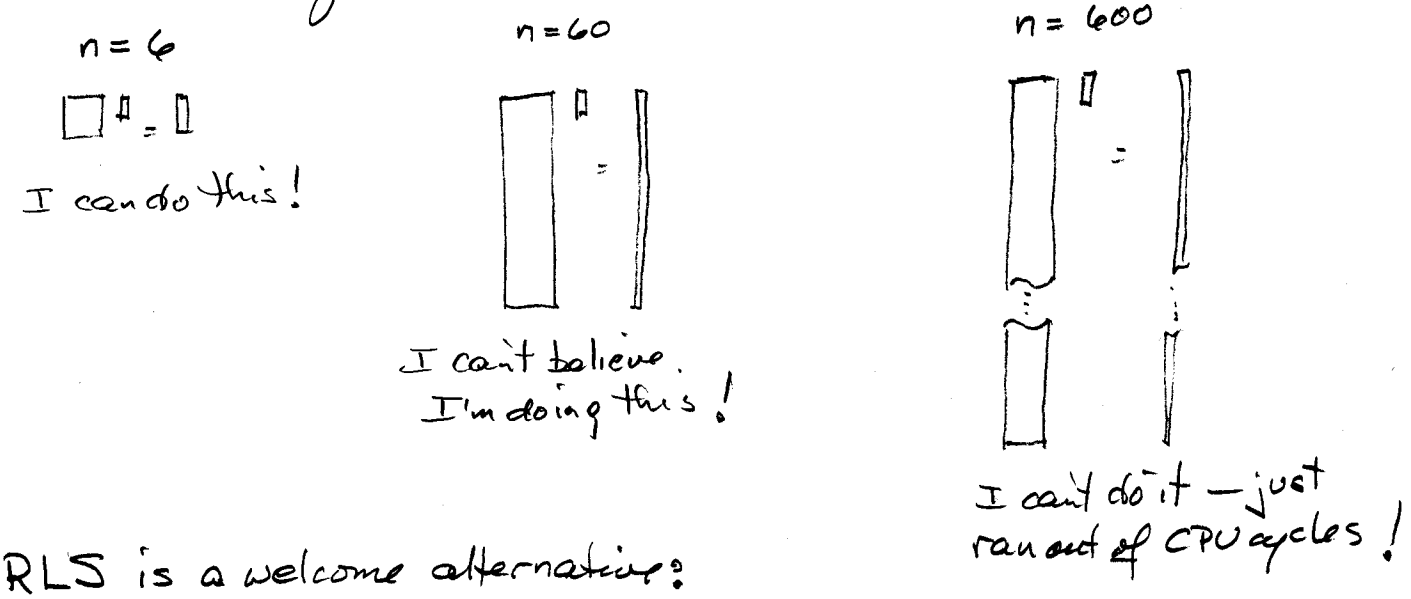
(from course ENSC 810 Statistical Signal Processing)

8 RECURSIVE LEAST SQUARES

Haykin Chapt 13

8.1 Why We Are Interested in RLS

- Our block oriented solution of LS takes care of some situations. Often though, samples continue to come in, and the LS solution evolves. We want to filter with the best solution, so we need a way to solve LS at every time
- One way (not recommended): let A grow, solve SVD or normal equations at each step.



- RLS is a welcome alternative:
  - Its estimate is always optimum; i.e., $\hat{\underline{w}}(n)$ is the LS solution with $u(1), \ldots u(n)$ and $d(1), \ldots, d(n)$
  - Constant and reasonable computation per time step
  - When environment is stationary, statistical, $\hat{\underline{w}}(n)$ converges in the norm (i.e. mean square) to the mmse solution
  - In stationary, statistical environment, convergence speed is independent of eigenvalues of R
  - Converges in about 2M time steps — much faster than LMS.
  - But computation is higher than LMS

8.2 A Basic Recursive Algorithm

- Denote data array at time n as $A(n)$. Then

$$A^{\dagger}(n)A(n) = \Phi(n) = \sum_{i=1}^{n} \underline{u}(i)\underline{u}^{\dagger}(i)$$

$$A^{\dagger}(n)\underline{d}(n) = \underline{z}(n) = \sum_{i=1}^{n} d^{*}(i)\underline{u}(i)$$

The LS solution must satisfy the normal equations at each time step

$$\Phi(n)\hat{\underline{w}}(n) = \underline{z}(n)$$

- Work with normal equations, despite the dynamic range (though see square root RLS). Here's an easy recursion to update Φ and $\underline{z}$:

$$\Phi(n) = \Phi(n-1) + \underline{u}(n)\underline{u}^{\dagger}(n)$$

$$\underline{z}(n) = \underline{z}(n-1) + d^{*}(n)\underline{u}(n)$$

This gives a rudimentary recursive solution to LS:

```
for n ← 1, 2, .. ∞ do
    begin
    Φ(n) ← Φ(n-1) + u(n)u†(n)
    z(n) ← z(n-1) + d*(n)u(n)
    ŵ(n) ← Φ⁻¹(n) z(n)
    use ŵ(n)
    end
```

We have a continuously-evolving solution to the deterministic normal equations — always solves the LS problem up to time n — with a constant computational load per time step.

- What's wrong with it?
  - starts at time 1, first M-1 iterations have singular $\underline{\underline{\Phi}}(n)$
  - $\underline{\underline{\Phi}}(n)$, $\underline{z}(n)$ grow with n
  - the bigone: we have to invert an M×M matrix every step
- We'll look at 1st & 3rd complaints later. For the moment, address the 2nd.

We could have defined

$$\underline{\underline{\Phi}}(n) = \frac{1}{n}\sum_{i=1}^{n} \underline{u}(i)\,\underline{u}^{+}(i) \qquad \underline{z}(n) = \frac{1}{n}\sum_{i=1}^{n} d^{*}(i)\,\underline{u}(i)$$

and used the update

$$\underline{\underline{\Phi}}(n) = \frac{n-1}{n}\,\underline{\underline{\Phi}}(n-1) + \frac{1}{n}\,\underline{u}(n)\,\underline{u}^{+}(n)$$

$$\underline{z}(n) = \frac{n-1}{n}\,\underline{z}(n-1) + \frac{1}{n}\,d^{*}(n)\,\underline{u}(n)$$

- But neither this nor the original formulation allow for drift and tracking. A common solution is exponential weighting

$$\underline{\underline{\Phi}}(n) = \lambda\,\underline{\underline{\Phi}}(n-1) + \underline{u}(n)\,\underline{u}^{T}(n) \qquad 0<\lambda\leq 1$$

$$\underline{z}(n) = \lambda\,\underline{z}(n-1) + d^{*}(n)\,\underline{u}(n)$$

It's easy to show that the resulting $\underline{\hat{w}}(n)$ minimizes the exponentially weighted sum of squares

$$\mathcal{E}(n,\underline{w}) = \sum_{i=1}^{n} \lambda^{n-i}\,|e(i)|^{2}$$

$$e(i) = d(i) - \underline{\hat{w}}^{+}(i)\,\underline{u}(i)$$

Variations are possible — moving window, other weightings using ΔE higher than 1st order.

8.3 Matrix Inversion Lemma

- To make this a practical algorithm, we should avoid explicit inversion of $\underline{\Phi}(n)$. The matrix inversion lemma and Woodbury's Identity are the key — they provide a way to do the recursion on $\underline{\Phi}^{-1}(n)$ instead of $\underline{\Phi}(n)$.

- Matrix inversion lemma.

  If $\underset{M\times M}{A} = \underset{M\times M}{B^{-1}} + \underset{M\times N}{C}\,\underset{N\times N}{D^{-1}}\,\underset{N\times M}{C^{+}}$ $\quad A, B, D$ pos def

  then $A^{-1} = B - BC(D + C^{+}BC)^{-1}C^{+}B$

  > see Scharf 2.9 for a set of matrix inversion formulas

- Woodbury's identity

  $A \longleftrightarrow R \qquad B \longleftrightarrow R_0^{-1} \qquad C = \underline{u} \qquad D^{-1} = \gamma^2$

  So if $R = R_0 + \gamma^2 \underline{u}\,\underline{u}^{+}$ $\quad R_0$ pos definite

  then $R^{-1} = R_0^{-1} - \dfrac{\gamma^2}{1 + \gamma^2 \underline{u}^{+} R_0^{-1} \underline{u}} R_0^{-1}\,\underline{u}\,\underline{u}^{+} R_0^{-1}$

  Very useful, since we have

  $$\underline{\Phi}(n) = \lambda \underline{\Phi}(n-1) + \underline{u}(n)\,\underline{u}^{+}(n)$$