JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

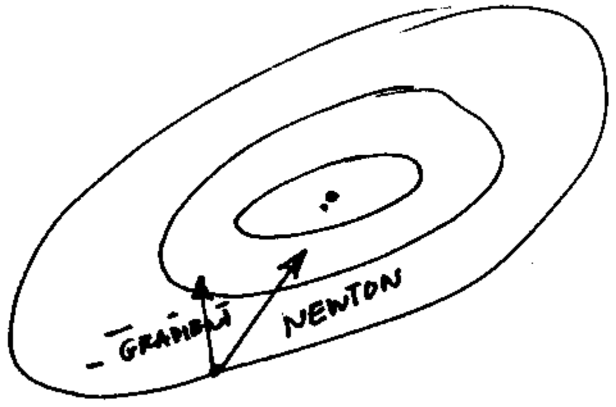
LMS/NEWTON ALGORITHM

Newton method does not follow the gradient. It goes directly in the direction of the minimum.

<u>NEWTON</u> $W_{k+1} = W_k - \mu R^{-1} \hat{\nabla}_k$

<u>LMS</u> $W_{k+1} = W_k + 2\mu \varepsilon_k X_k$

STEEPEST DESCENT $\left(W_{k+1} = W_k - \mu \hat{\nabla}_k \right)$



With m=1/2 we reach the minimum in one step. If m <1/2 we take more steps, but follow always the direction of the minimum.

The requirements are that we need to know R^{-1} and gradient at each step.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

What if we use the crude LMS estimate for the gradient?

We get what is called the LMS/Newton.

$$\hat{\nabla}_k = -2\varepsilon_k \bar{X}_k$$

$$\bar{W}_{k+1} = \bar{W}_k + 2\mu R^{-1} \varepsilon_k \bar{X}_k$$

Let us normalize μ to better compare the properties of the LMS/N with the LMS.

When R is diagonal with equal weights

$$\lambda_{av} R^{-1} = I$$

So if we want to have comparable μs, we must substitute

$$\mu \hookrightarrow \mu \lambda_{av}$$

and the algorithm becomes.

$$\bar{W}_{k+1} = \bar{W}_k + 2\mu \lambda_{av} R^{-1} \varepsilon_k \bar{X}_k$$

For convergence

$$0 < \mu < \frac{1}{\lambda_{MAX}}$$

For one step iteration

$$\mu = \frac{1}{2\lambda_{ave}}$$

It is the use of the information contained in R^{-1} that makes the algorithm much faster. We still do not know how to calculate it, but assuming we know R, the LMS/N goes in a straight line to the minimum, so it displays the fastest convergence.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

PROPERTIES OF LMS/NEWTON

Convergence

Convergence of the steepest descent is given by the slowest mode

$$\tau_n = \frac{1}{2\mu\lambda_n} \Longrightarrow T_{mse} = \frac{1}{4\mu\lambda_{min}}$$

In the LMS/N it is given by (just one time constant)

$$\tau_n = \frac{1}{2\mu\lambda_{ave}} \Longrightarrow T_{mse} = \frac{1}{4\mu\lambda_{ave}}$$

So the point is clear. LMS/N converges controlled by the $l_{avg.}$

So when eigenvalue spread is high it is much faster. If eigenvalue spread is one, same convergence.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

Misadjustment

For the LMS/N, from the expression of the cov[V$_k'$],

$$\mathrm{cov}[V_k'] = \frac{\mu \lambda_{ave} (\Lambda^{-1})^2}{4(1-\mu\lambda_{ave})} \mathrm{cov}[N_k']$$

we get

The gradient estimate has a cov[N$_k'$]

$$\mathrm{cov}[N_k'] = 4\xi_{MIN}\Lambda$$

so,

$$\mathrm{cov}[V_k'] = \frac{\mu\lambda_{ave}\xi_{MIN}}{1-\mu\lambda_{ave}}\Lambda^{-1}$$

The excess MSE becomes

$$\text{excess MSE} = E[V_T'^{T}\Lambda V'] = \sum_{n=0}^{L} \lambda_n E[v_{nk}'^{2}]$$

$$= \frac{(L+1)\mu\lambda_{ave}\xi_{MIN}}{1-\mu\lambda_{ave}}$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

and for small m denominator is close to 1, so $\left(\mu << \frac{1}{2\lambda_{av}}\right)$

$$\text{excess MSE} = \mu \, tr[R] \, \xi_{min}$$

and finally

$$M = \mu \, tr[R]$$

Therefore, for the same misadjustment the LMS/N is faster than the LMS by the ratio

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

SEQUENTIAL REGRESSION ALGORITHM (SER)

The problem is to estimate R^{-1} without making a lot of computations.

There are methods of estimating R^{-1} at each step without inverting R.

One of the most widely used is to use the matrix inversion lemma

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(DA^{-1}B + C^{-1}\right)^{-1} DA^{-1}$$

If we make

$$A = R_k \quad ; \quad B = X_k \quad ; \quad C = 1 \quad ; \quad D = X_k^T$$

$$R_{k+1} = R_k + X_k X_k^T \qquad (A+BCD)$$

Therefore,

$$R_{k+1}^{-1} = R_k^{-1} - \frac{R_k^{-1} X_k \cdot X_k^T R_k^{-1}}{1 + X_k^T R_k^{-1} X_k}$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

The bottom line is that we do not need to compute R^{-1}. We can compute it recursively from previous estimate.

It is obvious that the method requires a "good" starting value, because otherwise the errors can propagate (recursive algorithm).

Second we would like to "track" changes in the signal statistics. Therefore we would like to give more weight to the recent estimates of R^{-1} than old estimates. A rectangualr window will not the the job, however an exponential decay window will. Moreover, exponential windows can be recursively computed.

This is how the book derives the SER algorithm:

It starts with a ML estimate of the autocorrelation function

$$\hat{R}_k = \frac{1}{k+1} \sum_{l=0}^{k} X_l X_l^T$$

Then defines Q to give the short time memory feature

$$\hat{Q}_k = \sum_{l=0}^{k} \alpha^{k-l} X_l X_l^T$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

where alpha is $0 < \alpha < 1$

$$\alpha \simeq 2^{-\frac{1}{\text{LENGHT OF STAT. X}}}$$

For small values of α we can see that the estimate is basically independent of the initial condition. So just make it easy

$$\hat{R}_0 = \gamma I$$

where γ is a large positive constant (~100 times the power).

Notice also that there is an embedded recursive formula for Q

$$\hat{Q}_k = \alpha^0 X_k X_k^T + \sum_{l=0}^{k-1} \alpha^{k-l} X_l X_l^T = \alpha \hat{Q}_{k-1} + X_k X_k^T$$

If we premultiply by Q_k^{-1} and postmultiply by Q_{k-1}^{-1} and X_k

$$Q_{k-1}^{-1} = \alpha Q_k^{-1} + Q_k^{-1} X_k X_k^T Q_{k-1}^{-1}$$

$$Q_{k-1}^{-1} X_k = Q_k^{-1} X_k \left(\alpha + X_k^T Q_{k-1}^{-1} X_k\right)$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

Dividing by the scalar factor in parenteses and multiply on the right by $X_k^T Q_{k-1}^{-1}$

$$Q_k^{-1} = \frac{1}{\alpha}\left[Q_{k-1}^{-1} - \frac{(Q_{k-1}^{-1} X_k)(Q_{k-1}^{-1} X_k)^T}{\alpha + X_k^T (Q_{k-1}^{-1} X_k)} \right]$$

This the iterative procedure to compute Q^{-1}

So let us see how we apply all this.

We would like to compute the optimal weights by

$$\hat{R}_k \dot{W}_k = \hat{P}_k$$

P can also be estimated by

$$\hat{P}_k = \frac{1-\alpha}{1-\alpha^{k+1}} \sum_{l=0}^{k} \alpha^{k-l} d_l X_l$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

Therefore,

$$Q_k W_k = \sum_{l=0}^{k} \alpha^{k-l} d_l X_l$$

Let us assume that we want to compute W_{k+1} from $\hat{R}_k$ and $\hat{P}_k$ (rather than W_k).

$$Q_k W_{k+1} = \underbrace{\alpha \sum_{l=0}^{k-1} \alpha^{(k-1)-l} d_l X_l}_{\alpha Q_{k-1} W_k} + d_k X_k$$

$\alpha Q_{k-1} W_k$ but $Q_k = \alpha Q_{k-1} + X_k X_k^T$

$$= \left(Q_k - X_k X_k^T \right) W_k + d_k X_k$$

Since $d_k = \varepsilon_k + X_k^T W_k$

$$Q_k W_{k+1} = Q_k W_k + \varepsilon_k X_k$$

we get

$$W_{k+1} = W_k + Q_k^{-1} \varepsilon_k X_k$$

which is equivalent to LMS/N

$$\left(Q_k^{-1} = \frac{1-\alpha}{1-\alpha^{k+1}} R_k^{-1} \right)$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EFL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

RECURSIVE LEAST SQUARES

OVERVIEW OF BLOCK COMPUTATION

We saw that the MMSE solution gave the Wiener Hopf equation

$$W^* = R^{-1}P$$

$$R = E[x_k x_k^T] \qquad P = E[x_k d_k]$$

This solution is algebraic, and assumes the computation done in blocks.

For ergodic processes can substitute espected values by time averages

$$E \rightarrow \sum_{-\infty}^{\infty}$$

and we further estimate the true time autocorrealtion function with finite blocks of data.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

Typically we divide the data in N samples blocks, and hop N points. This procedure is sometimes called <u>block least squares</u>.

The most common of the LS techniques use the autocorreal-tion and covariance methods:

1. Find the time interval where data is approximately stationary.
2. Define a rect. window of that length (M samples)
3. Compute either the auto or covariance.

3a. Autocorrelation

$$\mathcal{E} = \sum_{l=0}^{M+L-1} e_n^2(l) = \sum_{l=0}^{M+L-1} \left[a_l d_l - \sum_{m=1}^{L} W_m a_{l-m+1} x_l \right]^2$$

Window is applied to the data prior to the calculation of the er-ror:

Notice that data exists in 0->M-1, but due to the filtering, error exists in 0-> M+L-1. This creates problems (error transients). Can use windows to minimize this problem.

3b. Covariance method

Here the error is windowed 0->M-1.

$$\varepsilon = \sum_{l=0}^{M-1} e^2(l) = \sum_{l=0}^{M-1} \left[a_l d l - \sum_{m=-L}^{M-L-1} w_m x_{l-m+1} \right]^2$$

However, data outside the block is used (- L) is used. Same data near the ends gets used twice. Now we do not have a true autocorrelation because data segments are of different length.

3c. Pre-window method

The covariance method is used, but the samples with indeces less than zero are set to zero value. (most widely used procedure in recursive computations).

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

All of these methods compute

$$w^{*} = R^{-1}P$$

The covariance method is much more time consuming (must use the Cholesky decomposition to compute the equation instead of the faster Durbin-Levinson algorithm).

Also, if the data statistics change in the middle of the interval will get a bad estimation.

These methods require large precision because the solution is computed algebraicaly.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

ESTIMATION OF THE AUTOCORRELATION FUNCTION

- Window estimates
- Recursive estimates

Window estimates are moving average (MA).

Recursive estimates are autoregressive (AR).

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

MOVING AVERAGE

Estimation is

$$R(p) = \frac{1}{N_0 - p} \sum_{n=1}^{N_0 - p} x(n)\, x(n-p)$$

Notice that we can think of the estimation as the output of a FIR filter (impulse response is a rectangle of length N).

It is unbiased

Its variance is

$$\mathrm{Var}\{R(p)\} \simeq \frac{\left[1 - R^2(p)\right]^2}{N_0 - p}$$

so the estimator is CONSISTENT.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

RECURSIVE ESTIMATION

Now let us substitute the FIR filter by an IIR filter, i.e. the estimation will be computed using the previous value of the quantity (autocorrelation iin this case).

Simplist IIR is the first order lowpass filter

$$y(n) = x(n) + b\,y(n-1) \qquad 0 < b < 1$$

This estimator is biased. Just try to estimate the mean value of x(n) when it is a constant signal with added zero mean gaussian noise with power σ^2.

$$x(n) = m + e(n)$$

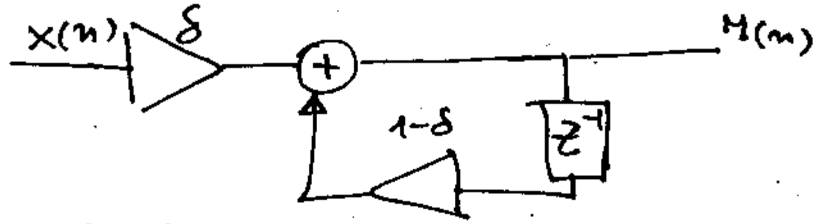
$$y(n) = m\,\frac{1-b^{n+1}}{1-b} + \sum_{i=0}^{n} b^i e(n-i) \qquad (y(n)=0 \quad n<0)$$

$$E[y(n)] = m\,\frac{1-b^{n+1}}{1-b}$$

$$\sigma_o^2 = \sigma_e^2\,\frac{1-b}{1+b}$$

TAKING $b = 1-\delta$ $\quad \boxed{\sigma_o^2 \simeq \sigma_e^2\,\frac{\delta}{2}}$

The block diagram of the estimator is $M(n)=(1-\delta)M(n-1)+\delta x(n)$



If we define the time constant τ

$$e^{-1/\tau}=b$$

which for b close to 1 leads to

$$\tau \simeq \frac{1}{1-b}=\frac{1}{\delta}$$

Now the power of the FIR estimator for the same signal gives

$$y(n)=\frac{1}{N_0}\sum_{i=0}^{N_0-1}x(n-i) \Rightarrow (\sigma_0')^2=\frac{\sigma_e^2}{N_0}$$

$$\boxed{2\tau \simeq N_0}$$

So comparing with the IIR, we can say that the recursive etimator is equivalent to the FIR whose width is twice the time constant.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

RECURSIVE LEAST SQUARES

What if one uses a window that is infinite, but recursively generated, such as a decaying exponential. All data is considered, but the recent past is weighted more.

In principle we could think that this would increase the computations.

However, if we find a <u>recursive solution</u> this may change.Recursive solution means that a new set of coefficients and new estimates are computed <u>every data point</u>.

Advantages:

1. This procedure will guarantee optimality at each step.
2. We are using data more effectively.

At the end of M points we will have the same W^* as the block methods, which give better estimates of W^* than the iterative procedures (no rattling).

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

ESL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

We want to find a solution of the error

$$\varepsilon_{k+1} = \sum_{l=0}^{k} a_l (d_l - y_l)^2$$

and arrive at it by calculating it from the previous estiamate.

Assuming we know W^*_k, how can we calculate W^*_{k+1}?

$$\varepsilon_{k+1} = \varepsilon_k + | d_k - y_k |^2$$

One possible solution is

1. Update R

$$\bar{R}_{k+1} = \bar{R}_k + X_k X_k^T$$

2. Update P

$$\bar{P}_{k+1} = \bar{P}_k + d_k \bar{X}_k$$

3. Invert R_{k+1}

4. Compute $W_{k+1} = R_{k+1}^{-1} P_{k+1}$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

This is very time consuming ~ $N^3 + 2N^2 + N$ multiplications at each step for a N length filter.

KALMAN/GODARD ALGORITHM

Assume an exponential decaying window $a_\ell = \alpha^\ell$

and the prewindow definition of the error

$$\mathcal{E}_{k+1} = \sum_{l=0}^{k} \alpha^{k-\ell} (d_\ell - W_\ell^T x_\ell)^2$$

The time averaged autocorrelation function is

$$(k+1)\,\hat{R}_k = \sum_{l=0}^{k} \alpha^{k-\ell} x_\ell x_\ell^T$$

and the crosscorrelation

$$(k+1)\,\hat{P}_k = \sum_{l=0}^{k} \alpha^{k-\ell} x_\ell d_\ell$$

Therefore,

$$W_k^* = \hat{R}_k^{-1} \hat{P}_k$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EFL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

As long as the window is recursively computed

$$\begin{cases} \hat{R}_{k+1} = \alpha \hat{R}_k + X_k X_k^T \\ \hat{P}_{k+1} = \alpha \hat{P}_k + d_k X_k \end{cases}$$

So $\hat{P}_{k+1} = R_{k+1} W^*_{k+1}$

Substituting

$$R_{k+1} W^*_{k+1} = \alpha R_k W_k^* + d_k X_k$$

$$= (R_{k+1} - X_k X_k^T) W_k^* + d_k X_k$$

$$= R_{k+1} W_k^* + X_k (d_k - X_k^T W_k)$$

Defining $e_k = d_k - X_k^T W_k$

(notice that the error is calculated with old coefficients) LEFT -MULTIPLYING by R^{-1}_{k+1}

$$W_{k+1} = W_k^* + R^{-1}_{k+1} e_k X_k$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EFL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

This is formally equivalent to the LMS/Newton algorithm.

1. It is different from the block LS methods because here estimates and coefficients are used <u>every new sample</u>.
2. Also the update of W_k is by the right amount.

We still need to invert efficiently R_k^{-1}. Using the inversion lemma this is done easily.

We saw that
$$R_{k+1}^{-1} = R_k^{-1} - \frac{R_k^{-1} X_k X_k^T R_k^{-1}}{1 + X_k^T R_k^{-1} X_k}$$

Now, if we subsitute this expression in the weight update equation, $W_{k+1}^* = R_{k+1}^{-1} P_{k+1}$ WITH $\alpha = 1$

$$W_{k+1}^* = \left\{ R_k^{-1} - \frac{\overbrace{R_k^{-1} X_k}^{Z_k} X_k^T R_k^{-1}}{1 + \underbrace{X_k^T R_k^{-1} X_k}_{q}} \right\} \left\{ P_k + d_k X_k \right\}$$

$Z_k \rightarrow$ FILTER INFORMATION VECTOR (KALMAN GAIN)

$q \rightarrow$ NORMALIZED POWER

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EFL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

If R and P are given as

$$R_k = \sum_0^k \alpha^{k-l} x_l x_l^T$$

$$P_k = \sum_0^k \alpha^{k-l} d_l x_l$$

Alpha is given by

$$\alpha = 2^{-1/\text{LENGTH OF STATIONARITY}}$$

with an effective averaging period of $\simeq \frac{1}{1-\alpha}$

R, P become

$$\left\{\begin{array}{l} R_{k+1} = \alpha\left[R_k + \frac{1}{\alpha} X_k X_k^T\right] \\ P_{k+1} = \alpha\left[P_k + \frac{1}{\alpha} d_k X_k\right] \end{array}\right.$$

and the optimum weigth

$$W_{k+1}^* = W_k^* + \left\{\frac{d_k - y_k^o}{\alpha + q}\right\} Z_k$$

and the updating of R^{-1}

$$R_{k+1}^{-1} = \frac{1}{\alpha}\left[R_k^{-1} - \frac{Z_k Z_k^T}{\alpha + q}\right]$$

JOSE C. PRINCIPE
UNIVERSITY OF FLORIDA
EEL 6935- SPRING 90
904-392-2662
principe@brain.ee.ufl.edu

Let us define:

Optimal weight at iteration k

$$W_k^* = R_k^{-1} P_k$$

Filtered information vector (Kalman gain)

$$Z_k = R_k^{-1} X_k$$

Apriori output

$$y_o(k) = X_k^T W_k^*$$

Normalized power

$$q = X_k^T Z_k$$

Then the equation becomes

$$W_{k+1}^* = W_k^* - \frac{z_k X_k^T W_k^*}{1+q} + d_k Z_k - \frac{d_k q Z_k}{1+q}$$

$$= W_k^* + \frac{[d_k - y_k^o]\, Z_k}{1+q}$$

This equation embodies the RLS algorithm. Let us interpret the result.

We compute W_{k+1}^* by using the previous value plus a corrective term, which depends on x_k and d_k in three ways:

1. The apriori difference (or apriori prediction error)

$$e_k^0 = d_k - y_k^0$$

which is the difference between the desired sample and the filter output using the present sample but the old filter coefficients.

y^0_k is in fact the output estimate before x_k is used to update W.

2. Z_k is the filter imformation vector because R^{-1} acts to influence the direction and the length of the data vector. Because it magnifies e^0_k it is called the Kalman gain.
3. q is just a measure of the input signal power, normalized by R^{-1}_k.

JOSE C. PRINCIPE
UNIVERSITY OF FLORIDA
EEL 6935- SPRING 90
904-392-2662
principe@brain.ee.ufl.edu

What is the similiarity with LMS/Newton?

LMS/N

$$W_{k+1}^{*} = W_k^{*} + 2\mu R^{-1} \varepsilon_k X_k$$

RLS

$$W_{k+1}^{*} = W_k^{*} + \frac{1}{1+q} e_k^{\circ} R_k^{-1} X_k$$

So when

$$2\mu = \frac{1}{1+q} = \frac{1}{1+X_k^T R^{-1} X_k}$$

the LMS/N becomes RLS.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EFL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

What is the similiarity with the normalized LMS?

$$w_{k+1} = w_k + 2\mu_k \varepsilon_k x_k$$

$$\mu(k) = \frac{\alpha}{\gamma + X_k^T X_k} \qquad 0 < \alpha < 2$$

When γ=1, at least we see that the step size of the normalized LMS and the RLS are equivalent, and the heuristic is theoretically justified.

JOSE C. PRINCIPE
UNIVERSITY OF FLORIDA
EFL 6935- SPRING 90
904-392-2662
principe@brain.ee.ufl.edu

RLS algorithm

1. Get x_k and d_k.
2. Form X_k by bringing the new value x_k.
3. Compute the apriori output y^0_k.
4. Compute the apriori error e^0_k.
5. Compute the filtered information vector Z_k.
6. Compute the normalized power q.
7. Compute the gain $v=1/(1+q)$.
8. Compute the normalized $\tilde{Z}_k = v.Z_k$.
9. Update the optimal weight vector.

$$W_{k+1} = W_k + e^0_k \tilde{Z}_k$$

10. Update $R^{-1}_{k+1} = R^{-1}_k - \tilde{Z}_k Z^T_k$.

Start the algorithm with $R^{-1}_0 = 100 \times \sigma \times I$.

An exponential window is recommended. With the window the algorithm is the same except:

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

COMPUTATION COMPLEXITY OF RLS

Straight computation of $W^* = R^{-1}P$

using Gaussian elimination for R^{-1} requires L^3 multiplications.

RLS

a) Step 4,7 simple O(1)

b) Step 3,6,8,9 (vector dot product,scalar/vector) O(L).

c) Step 5,10 (matrix vector , vector outer product) $O(L^2)$.

Therefore, for each input sample

$2L^2+4L$ multiplications (and equal amount of additions and one division).

For a block of N points (effectively N-L+1 iterations)

$$C_{RLS} = (N-L+1)2L^2 + (N-L+1)4L$$

JOSE C. PRINCIPE
UNIVERSITY OF FLORIDA
EEL 6935- SPRING 90
904-392-2662
principe@brain.ee.ufl.edu

For the straight block LS with same filter length and block size

$$C_{LS} = (N-L+1)L^2 + (N-L+1)\cdot L + L^3 + L$$

We see that RLS is more expensive in terms of $O(L^2)$ and $O(L)$ terms. So when filter order is small compared with the segment length, the RLS becomes worse than BLS!!!

So why use RLS?

1. Numerically better behaved.
2. RLS provides w* at every step, so better in nonstationary environments.
3. It leads to lower cost computational techniques such as the fast RLS.

JOSE C. PRINCIPE
UNIVERSITY OF FLORIDA
EEL 6935- SPRING 90
904-392-2662
principe@brain.ee.ufl.edu

FAST RLS

RLS used the structure of the auto/cross correlation function to decrease computation. But did nothing to take advantage of the structure of X ($X_{k=1}$ is obtained from X_k through the addition of one sample) and Z (Z_{k+1} could be calculated from the previous, avoiding the updating of R_k^{-1} and its multiplication by X(k)). This is very important because these steps are $O(L^2)$, and without them the complexity would drop to O(L).

Let us start by defining a slightly different estimate of $\tilde{Z}_k$.

Define $\quad V_k = W_{k+1}^* - W_k^*$

as the difference between optimum weight vectors at time k+1 and k.

$$R_{k+1} W_{k+1}^* = R_{k+1}(W_k^* + V_k) = P_{k+1}$$
$$= R_{k+1} W_k^* + R_{k+1} V_k$$

Substituting the recursive defininitions of R and P

$$(R_k + X_k X_k^T) W_k^* + R_{k+1} V_k = P_k + d_k X_k$$
$$\left\{ \begin{array}{l} R_k W_k^* = P_k \\ y_0(k) = X_k^T W_k^* \end{array} \right.$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

Then we have

$$R_{k+1} V_k = \{ d_k - y_0(k) \} X_k$$

$$e_0(k) = d(k) - y_0(k)$$

$$V_k = e_0(k) \, R_{k+1}^{-1} X_k$$

and also

$$Z$$

V is the update to W that guarantees optimality. If we define Z as

$$\tilde{Z}_k = R_{k+1}^{-1} X_k = \left\{ \sum_{l=L-1}^{k} X_l X_l^T \right\}^{-1} X_k$$

then we see that it only depends on the input sequence x(k). In order to compute $\tilde{Z}_k$ efficiently we have to use the concept of backward B_k and foward prediction A_k.

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

1. Compute the apriori forward prediction error

$$\varepsilon_0(k+1) = x_{k+1} + A_k^T X_k$$

2. Update the forward prediction vector

$$A_{k+1} = A_k - \tilde{Z}_k \varepsilon_0(k+1)$$

3. Compute the posteriori prediction error

$$\varepsilon(k+1) = x(k+1) + A_{k+1}^T X_k$$

4. Compute prediction cross power

$$\Sigma_{k+1} = \Sigma_k + \varepsilon(k+1)\varepsilon_0(k+1)$$

5. Form the augmented vector

$$F = \begin{bmatrix} \varepsilon(k+1)/\Sigma_{k+1} \\ \tilde{Z}_k + A_{k+1}\varepsilon(k+1)/\Sigma_{k+1} \end{bmatrix} \begin{array}{l} 1 \text{ ELEMENT} \\ L \text{ ELEMENTS} \end{array}$$

6. Partition F

$$F = \begin{bmatrix} M(k+1) \\ \mu(k+1) \end{bmatrix} \begin{array}{l} L \text{ ELEMENTS} \\ 1 \text{ ELEMENT} \end{array}$$

7. Compute the apriori backward prediction error

$$\eta_0(k+1) = x(k-L+1) + B_k^T X_{k+1}$$

JOSE C. PRINCIPE

UNIVERSITY OF FLORIDA

EEL 6935- SPRING 90

904-392-2662

principe@brain.ee.ufl.edu

8. Update the backward prediction vector B

$$B_{k+1} = \left[B_k - M_{k+1}\, \eta_o(k+1)\right] / \left[1 - \mu(k+1)\, \eta_o(k+1)\right]$$

9. Update $\tilde{Z}_k$

$$\tilde{Z}_{k+1} = M_{k+1} - B_{k+1} \cdot \mu(k+1)$$

(For a proof see Messerschmitt)

Now knowing $\tilde{Z}_k$ we can state a fast algorithm for computing W.

For each K:

1. Compute the apriori output

$$y_o(k+1) = X_{k+1}^T W_k^*$$

2. Form the apriori output prediction error

$$e_o(k+1) = d(k+1) - y_o(k+1)$$

3. Update $\tilde{Z}_k$ to $\tilde{Z}_{k+1}$
4. Compute the impulse response vector

$$W_{k+1}^* = W_k^* + e_o(k+1)\, \tilde{Z}_{k+1}$$

Using this approach algorithms require about 7L multiply-adds per sample(O(L)).

However, these algorithms have numerical problems. The fastest is the algorithm the less accurate it becomes. This is an area of current research.