Normalized LMS Algorithm

In the LMS algorithm

$$\hat{\mathbf{w}}(n+1) = \hat{\boldsymbol{\omega}}(\nu) + \mu\boldsymbol{\upsilon}(\nu)\varepsilon^*(\nu)$$

the correction $\mu\mathbf{u}(n)e^*(n)$ applied to $\hat{\mathbf{w}}(n)$ is directly proportional to $\mathbf{u}(n)$. Thus when $\mathbf{u}(n)$ contains a significant number of large sample values, the LMS algorithm experiences a gradient noise amplification problem. The gradient noise arising from our estimation of **R** and **p**. This problem can of course effect the convergence.

In order to make the convergence rate independent of signal power, we normalize the correction by the input signal power

$$\|\mathbf{u}(n)\|^2 = \boldsymbol{\upsilon}^H(\nu)\boldsymbol{\upsilon}(\nu) .$$

This normalization applied to the LMS algorithm yields the normalized least-mean-square algorithm

$$\hat{\mathbf{w}}(n+1) = \hat{\boldsymbol{\omega}}(\nu) + \frac{\tilde{\mu}\boldsymbol{\upsilon}(\nu)\varepsilon^*(\nu)}{\|\boldsymbol{\upsilon}(\nu)\|^2} .$$

Comments

1) We can think of the NLMS algorithm as an LMS algorithm with a time-varying step-size

$$\mu(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2} .$$

2) It can be shown that the NLMS algorithm is convergent in the mean square if

$$0 < \tilde{\mu} < 2 .$$

Typically we'll choose $\tilde{\mu} = 1$.

3) NLMS exhibits a rate of convergence potentially faster than LMS for both correlated (large χ) and uncorrelated data (small χ) for the same level of steady-state misadjustment. More specifically it can be shown that for NLMS the square root of the eigenvalue spread influences the convergence speed. See

M. Rupp, "The behavior of LMS and NLMS algorithms in the presence of spherically invariant processes," *IEEE Trans. on Signal Processing*, vol. 41, no. 3, Mar. 1993.

4) If $\|\mathbf{u}(n)\|^2$ is small there may be numerical difficulties when performing the division. Therefore we add the small constant *a*

$$\boxed{\hat{\mathbf{w}}(n+1) = \hat{\boldsymbol{\omega}}(\nu) + \frac{\tilde{\mu}\boldsymbol{\upsilon}(\nu)\varepsilon^*(\nu)}{\alpha + \|\boldsymbol{\upsilon}(\nu)\|^2}} .$$

Typically we'll choose $a = 0.01$.

Due to is simplicity and robustness, NLMS is the adaptive algorithm used in practice virtually all the time...although more power signal processors are providing the engine for the application of more complex, faster converging algorithms.

Affine Projection Algorithms (APA) or Generalized NLMS

This section summarizes the results from

- D. Morgan and S. Kratzer, "On a Class of Computationally Efficient, Rapidly Converging Generalized NLMS Algorithms," *IEEE Signal Processing Letters*, vol. 3, no. 8, Aug. 1996.
- S. Sankara and A. Beex, "Convergence Behavior of Affine Projection Algorithms," *IEEE Trans. Signal Processing*, vol. 48, no. 4, Apr. 2000.
- S. Haykin, *Adaptive Filter Theory, 4th Ed.*, Prentice Hall, 2002

We may formulate the criterion for designing an affine projection filter as one of optimization subject to multiple constraints:

Minimize the squared Euclidean norm of the change in the weight vector

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)$$

subject to the set of N constraints

$$d(n-k) = \hat{\mathbf{w}}^H(n+1)\mathbf{u}(n-k) \text{ for } k = 0, 1, ..., N-1$$

where N is smaller than the dimensionality M of the input date space or, equivalently, the weight space.

In English, we are requiring the updated adaptive weight vector to be as close as possible to the previous weight vector (minimal disturbance) while minimizing the error vector over a block of N input vectors.

Using the method of Lagrange multipliers with multiple constraints (Appendix C), we formulate our cost function using the minimization and constraint criteria

$$J(n) = \left\|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\right\|^2 + \sum_{k=0}^{N-1} \mathrm{Re}[\lambda_k^*(d(n-k) - \hat{\mathbf{w}}^H(n+1)\mathbf{u}(n-k))]$$

where λ_k are the Lagrange multipliers. We can rewrite the cost function in matrix form as

$$\boxed{J(n) = \left\|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\right\|^2 + \mathrm{Re}[(\mathbf{d}(n) - \hat{\mathbf{w}}^H(n+1)\mathbf{U}(n))\lambda]}$$

where

$$\mathbf{d}(n) = [d(n) \quad d(n-1) \quad \mathrm{L} \quad d(n-N+1)]$$

is the $1 \times N$ desired response vector,

$$\mathbf{U}(n) = [\mathbf{u}(n) \quad \mathbf{u}(n-1) \quad \mathrm{L} \quad \mathbf{u}(n-N+1)]$$

is the $M \times N$ matrix whose columns are the N most recent input signal vectors

$$\mathbf{u}(n) = [u(n) \quad u(n-1) \quad \mathrm{L} \quad u(n-M+1)]^T,$$

and

$$\lambda = [\lambda_0 \quad \lambda_1 \quad \mathrm{L} \quad \lambda_{N-1}]^H.$$

is the $N \times 1$ Lagrange vector.

We can show

$$\frac{\partial J(n)}{\partial \hat{\mathbf{w}}^*(n+1)} = 2[\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)] - \mathbf{U}(n)\lambda$$

Setting the above to zero, we get

$$\delta\hat{\mathbf{w}}(n+1) = \frac{1}{2}\mathbf{U}(n)\lambda \ .$$

To eliminate the Lagrange vector, we first rewrite our constraint equation as

$$\begin{aligned} \mathbf{d}^H(n) &= \mathbf{U}^H(n)\hat{\mathbf{w}}(n+1) \\ &= \mathbf{U}^H(n)[\hat{\mathbf{w}}(n) + \delta\hat{\mathbf{w}}(n+1)] \\ &= \mathbf{U}^H(n)\hat{\mathbf{w}}(n) + \frac{1}{2}\mathbf{U}^H(n)\mathbf{U}(n)\lambda \end{aligned} .$$

Solving for the Lagrange vector we have

$$\begin{aligned} \lambda &= 2[\mathbf{U}^H(n)\mathbf{U}(n)]^{-1}[\mathbf{d}^H(n) - \mathbf{U}^H(n)\hat{\mathbf{w}}(n)] \\ &= 2[\mathbf{U}^H(n)\mathbf{U}(n)]^{-1}\mathbf{e}^H(n) \end{aligned}$$

where the $1 \times N$ error vector is defined as

$$\mathbf{e}(n) = \mathbf{d}(n) - \hat{\mathbf{w}}^H(n)\mathbf{U}(n).$$

Finally, we substitute the Lagrange vector and arrive at the optimum weight vector change

$$\delta\hat{\mathbf{w}}(n+1) = \mathbf{U}(n)[\mathbf{U}^H(n)\mathbf{U}(n)]^{-1}\mathbf{e}^H(n)$$

Thus our iterative update equation for the filter coefficients (including a scale factor, $\tilde{\mu}$ to scale the correction) is

$$\boxed{\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tilde{\mu}\mathbf{U}(n)[\mathbf{U}^H(n)\mathbf{U}(n)]^{-1}\mathbf{e}^H(n)}.$$

APA can also be written as

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tilde{\mu}\mathbf{U}^+(n)\mathbf{e}^H(n)$$

where

$$\mathbf{U}^+(n) = \mathbf{U}(n)[\mathbf{U}^H(n)\mathbf{U}(n)]^{-1}$$

is the Moore-Penrose pseudo inverse of $\mathbf{U}(n)$ for $N < M$ (pinv in MATLAB).

The above is referred to as the Affine Projection Algorithm (APA). This algorithm was also recognized as a Generalized NLMS (GNLMS) for the following reason. For the NLMS, we have (ignoring a for numerical stability)

$$\begin{aligned} \hat{\mathbf{w}}(n+1) &= \hat{\boldsymbol{\omega}}(\nu) + \frac{\tilde{\mu}\boldsymbol{\upsilon}(\nu)\varepsilon^*(\nu)}{\|\boldsymbol{\upsilon}(\nu)\|^2} \\ &= \hat{\boldsymbol{\omega}}(\nu) + \tilde{\mu}\boldsymbol{\upsilon}(\nu)[\boldsymbol{\upsilon}^H(\nu)\boldsymbol{\upsilon}(\nu)]^{-1}\varepsilon^*(\nu) \end{aligned} .$$

This (NLMS) can be generalized to

$$\hat{\mathbf{w}}(n+1) = \hat{\boldsymbol{\omega}}(\nu) + \tilde{\mu}\mathbf{Y}(\nu)[\mathbf{Y}^H(\nu)\mathbf{Y}(\nu)]^{-1}\boldsymbol{\varepsilon}^H(\nu)$$

where

$$\mathbf{U}(n) = [\boldsymbol{\upsilon}(\nu) \quad \boldsymbol{\upsilon}(\nu-1) \quad \Lambda \quad \boldsymbol{\upsilon}(\nu-N+1)]$$

is the M x N matrix whose columns are the N most recent input signal vectors

$$\mathbf{u}(n) = [\upsilon(\nu) \quad \upsilon(\nu-1) \quad \Lambda \quad \upsilon(\nu-M+1)]$$

and

$$\mathbf{e}(n) = [\varepsilon(\nu) \quad \varepsilon(\nu-1) \quad \Lambda \quad \varepsilon(\nu-M+1)]$$

is the associated error vector. Clearly if $N = 1$, we have the NLMS algorithm.