Class Comments

In order to maintain continuity in the development of adaptive filters, we will skip (for now) adaptive beamforming (Linearly Constrained Minimum Variance filter and the Generalized Sidelobe Canceler) and the Kalman filter. We will cover both of these topics later in the course.

Method of Steepest Descent

Introduction

The requirement that an adaptive filter has to satisfy is to find a solution for $\mathbf{w}$ (length M) that satisfies the Wiener-Hopf equations. One way of doing this is to solve the system of equations. While this is straightforward it does represent serious computational difficulties.

The taxonomy is as follows:

1) If we are given $\mathbf{R}$ ($M \times M$) and $\mathbf{p}$ ($M \times 1$) and know these will be valid for all n, a minor (given cheap computational resources) source of complexity arises in the computation of $\mathbf{R}^{-1}$ [$O(M^3)$] required for the calculation of the Wiener filter $\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}$. However, we note the while the matrix inversion is difficult, in this case it needs only be computed once and the cost (for the inversion) may be acceptable. ***Solution: Wiener Filter***. (DONE)

2) If the signals are not stationary but we are periodically given $\mathbf{R}$ and $\mathbf{p}$, the complexity rises since we must periodically compute the Wiener filter–which would be expensive. What we would really like is a method to search for the Wiener filter. ***Solution: Method of Steepest Descent***

3) If we are not given $\mathbf{R}$ and $\mathbf{p}$, we must make short-time estimates of these matrices and either compute or search for an estimate of the Wiener filter. ***Solution: Adaptive Filter***

As we move toward the adaptive filter, we first examine the method of steepest descent in order to avoid the matrix inverse required with the Wiener filter.

The method of steepest descent (SD) is an old optimization technique. In adaptive filtering, we apply this technique to "search" the error performance surface, J for its minimum point. The corresponding coefficient vector will be the Wiener solution, $\mathbf{w}_o$. Again we wish to minimize our cost function defined to be the MSE.

Features of steepest descent

- recursive in the sense that starting with some initial (arbitrary) value for $\mathbf{w}(0)$, $J[\mathbf{w}(n+1)] \leq J[\mathbf{w}(n)]$, i.e. we improve (or adapt) with each iteration (sample). Note we now have a filter whose coefficients are time-varying.
- $\lim_{n\to\infty} \mathbf{w}(n) = \mathbf{w}_o$ and $\lim_{n\to\infty} J[\mathbf{w}(n)] = J(\mathbf{w}_o) = J_{\min}$ i.e. we end up with the Wiener filter and the minimum point on the error surface.
- finds the minimum point without any knowledge of the error surface itself.
- we arrive at our solution after *many steps* or iterations. Of course we're only willing to accept this iterative solution if the computational complexity is significantly reduced.

Some Preliminaries

Figure 8.1

We assume the following. The *estimation error* is given by

$$e(n) = \delta(\nu) - \boldsymbol{\omega}^H(\nu)\boldsymbol{\upsilon}(\nu)$$

where

$$\mathbf{w}(n) = [\,\omega_0(\nu) \quad \Lambda \quad \omega_{M-1}(\nu)]^T$$

and

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

The cost function is defined as the mean-squared error and yields the quadratic in **w**

$$J(n) = \sigma_\delta^2 - \boldsymbol{\omega}^H\boldsymbol{\pi} - \boldsymbol{\pi}^H\boldsymbol{\omega} - \boldsymbol{\omega}^H\mathbf{P}\boldsymbol{\omega}$$

where

σ_d^2 is the variance of the desired signal, $d(n)$

$\mathbf{p} = E[\boldsymbol{\upsilon}(\nu)\delta^*(\nu)]$ is the cross correlation vector between the input vector and desired response

$\mathbf{R} = E[\boldsymbol{\upsilon}(\nu)\boldsymbol{\upsilon}^H(\nu)]$ is the correlation matrix of the input vector.

Steepest-Descent Algorithm

To find $\mathbf{w}_o$ corresponding to J_{min} we do the following

1) Initialize $\mathbf{w}(0)$ with some guess for $\mathbf{w}_o$. If we haven't a clue we set $\mathbf{w}(0) = \mathbf{0}$.
2) Compute $\nabla J(n)$ (gradient vector) whose real and imaginary parts are defined as real and imaginary parts of the first derivative of $J(n)$ w.r.t. real and imaginary parts of $\mathbf{w}(n)$ (current guess).
3) Compute $\mathbf{w}(n+1)$ (next guess) by making a change or correction to $\mathbf{w}(n)$ (current guess) in a direction opposite to that of the gradient vector.
4) Go to 2)

Mathematically step 3) is given by the following recursive relation

$$\mathbf{w}(n+1) = \boldsymbol{\omega}(\nu) + \mu'[-\nabla\vartheta(\nu)]$$

where μ' scales the correction.

Figure. Sketch of SD for length 1 **w**.

One may initially think that step 2 may be computationally expensive. Let us solve for $\nabla J(n)$.

From our derivation of the principle of orthogonality we have

$$\nabla J = -2E[\mathbf{u}(n)e^*(n)]$$

which leads to the recursion

$$\mathbf{w}(n+1) = \boldsymbol{\omega}(\nu) + 2\mu' E[\boldsymbol{\upsilon}(\nu)\varepsilon^*(\nu)]. \tag{1}$$

We can simplify our expression for the gradient in terms of **R** and **p** as follows

$$\begin{aligned}\nabla J &= -2E[\mathbf{u}(n)e^*(n)] \\ &= -2E[\mathbf{u}(n)(d(n) - \mathbf{w}^H(n)\mathbf{u}(n))^*] \\ &= -2E[\mathbf{u}(n)(d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n))] \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)\end{aligned}$$

Substitution into the recursion yields

$$\begin{aligned}\mathbf{w}(n+1) &= \boldsymbol{\omega}(\nu) + \mu'[2\boldsymbol{\pi} - 2\mathbf{P}\boldsymbol{\omega}(\nu)] \\ &= \boldsymbol{\omega}(\nu) + \mu[\boldsymbol{\pi} - \mathbf{P}\boldsymbol{\omega}(\nu)]\end{aligned} \tag{2}$$

where for simplicity

$$\mu = 2\mu' .$$

μ again scales the size of the correction and is often called the step-size. We note in our update equation (2) the absence of a matrix inverse and a large computational complexity reduction.

We can view the SD algorithm (1), as a feedback model. With this view, we note that any adjustment we make to $\mathbf{w}(n)$ requires the measurement and use of $e(n)$.