MSE ESTIMATORS

Cont. (2)
RECURSIVE METHODS

Lezioni di Telecomunicazioni
Prof FRANCESCO A. N. PALMIERI

Corso di "Signal Processing and Data Fusion"

RECURSIVE METHODS (GRADIENT SEARCH)

The solution to the minimum squared error could be computed iteratively without matrix inversion.
The idea is to define the cost function and use a gradient search to find the solution.
Even if the solution, also in the singular case, could be obtained easily by solving the normal equation, the gradient search is very instructive to understand the nature of the stochastic searches that will be introduced later. In fact when the parameters to be found belong to a non linear function (i.e. a neural network), we cannot write the solution in closed form, as in the linear (or affine) case, and the only way to reach a solution is via a gradient search (iterative learning).

Recall that the cost function is

$$\mathcal{E}(\underline{c}) = E\left[\left(d[n] - \underline{c}^T \underline{x}[n]\right)^2\right] = E[d^2[n]] + \underline{c}^T E[\underline{x}[n]\underline{x}[n]^T]\underline{c}$$
$$- 2E[d[n]\underline{x}^T[n]]\underline{c} = \mathcal{E}_d + \underline{c}^T R_x \underline{c} - 2 r_{dx}^T \underline{c}$$

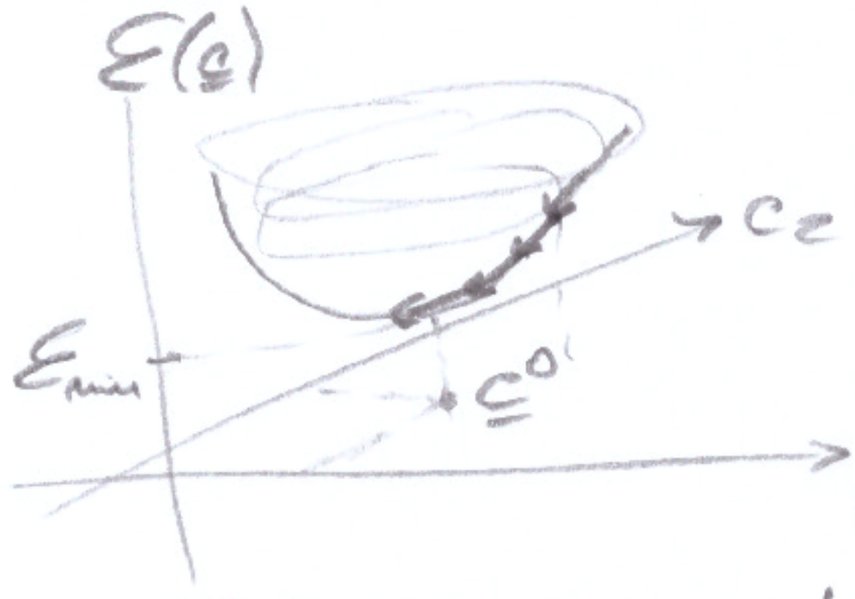
and the gradient

$$\nabla_{\underline{c}} \mathcal{E} = 2R_x \underline{c} - 2\underline{r}_{dx} \quad (*)$$

A gradient search (Newton algorithm) starts from an initial guess $\underline{c}(0)$ and proceeds on small steps in the direction opposite to the current gradient, i.e.

$$\underline{c}(k) = \underline{c}(k-1) - \mu\left(\underline{\underline{R}}_x \underline{c}(k-1) - \underline{r}_{dx}\right), \quad (**)$$

where μ is a step size parameter that controls the adaptation speed. The search is depicted in the following figure for $N=2$



The function $E(\underline{c})$ is a quadratic form and looks like a bowl. The minimum is unique because the function is convex. To verify (we did not do it before) let us compute the Hessian

$$H = \left(\nabla_{\underline{c}}\, \nabla_{\underline{c}}^T\right) E(\underline{c}) = \begin{bmatrix} \frac{\partial}{\partial c_1} \\ \frac{\partial}{\partial c_2} \\ \vdots \\ \frac{\partial}{\partial c_N} \end{bmatrix} \left[\frac{\partial}{\partial c_1}\ \frac{\partial}{\partial c_2}\ \cdots\ \frac{\partial}{\partial c_N}\right] E(\underline{c})$$

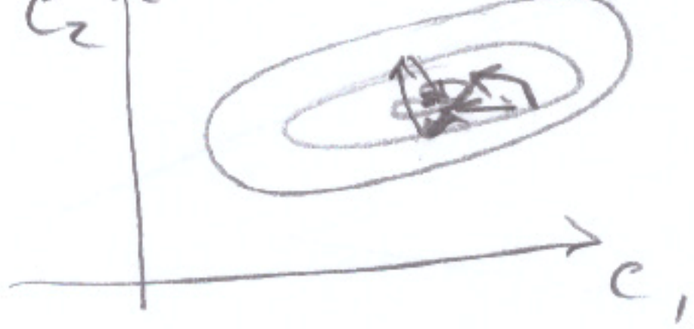
$$= \begin{bmatrix} \frac{\partial}{\partial c_1} \\ \vdots \\ \frac{\partial}{\partial c_N} \end{bmatrix} \left[2\,\underline{c}^T \underline{\underline{R}}_x - 2\,\underline{r}_{dx}^T\right]$$

$$= 2\,\underline{\underline{R}}_x$$

But $R_x \geq 0$ (def. positive), Therefore the function is convex anywhere.

The recursive equation (**) obviously has its only stationary point at the solution

$\underline{\underline{R}}_x \underline{c} - \underline{r}_{dx} = \underline{0}$, but we need to choose μ to control the speed, because large μ may make the recursion wander around the solution. Small μ may cause the algorithm to be too slow.



SEARCH DYNAMICS : (COEFFICIENT EVOLUTION)

To understand the evolution of $\{c(k)\}$, consider the difference vector

$$\underline{v}(k) = \underline{c}(k) - \underline{c}^o$$

Now we have to verify that $v(k) \to 0$. Substituting in (**), we obtain

$$\underline{v}(k) + \underline{c}^o = \underline{v}(k-1) + \underline{c}^o - \mu\left(\underline{\underline{R}}_x(\underline{v}(k-1) + \underline{c}^o) - \underline{r}_{dx}\right)$$

$$\underline{v}(k) = \underline{v}(k-1) - \mu\left(\underline{\underline{R}}_x \underline{v}(k-1) + \underline{\underline{R}}_x \underline{c}^o - \underline{r}_{dx}\right)$$

$$\underline{v}(k) = \left(\underline{\underline{I}} - \mu \underline{\underline{R}}_x\right)\underline{v}(k-1)$$

The convergence of this recursion is determined by the eigenvalues of $(\underline{\underline{I}} - \mu \underline{\underline{R}}_x)$

Use eigenvalue decomposition for $\underline{\underline{R}}_x$

$$\underline{\underline{R}}_x = \underline{\underline{Q}}\, \underline{\underline{\Lambda}}_x \underline{\underline{Q}}^T \qquad \underline{\underline{\Lambda}}_x = \mathrm{diag}(\lambda_1 \lambda_2 \ldots \lambda_N)$$

eigenvalues

eigenvectors → $[q_1 q_2 \ldots q_N]$

to get

$$\underline{v}(k) = (\underline{\underline{I}} - \mu \underline{\underline{Q}} \underline{\underline{\Lambda}}_x \underline{\underline{Q}}^T) \underline{v}(k-1)$$

Define $\underline{V}(k) = \underline{\underline{Q}}^T \underline{v}(k)$ difference vector in the eigenspace of $\underline{\underline{x}}$

$$\underline{\underline{Q}}^T \underline{v}(k) = \underline{\underline{Q}}^T (\underline{\underline{I}} - \mu \underline{\underline{Q}} \underline{\underline{\Lambda}}_x \underline{\underline{Q}}^T) \underline{v}(k-1)$$

$$\underline{V}(k) = (\underline{\underline{Q}}^T - \mu \underline{\underline{Q}}^T \underline{\underline{Q}} \underline{\underline{\Lambda}}_x \underline{\underline{Q}}^T) \underline{v}(k-1)$$

$$\underline{V}(k) = (\underline{\underline{I}} - \mu \underline{\underline{\Lambda}}_x) \underline{V}(k-1)$$

The components of $\underline{V}(k)$ are decoupled

$$V_i(k) = (1 - \mu \lambda_i) V_i(k-1) \quad i = 1, \dots, N$$

Convergence to zero is obtained if

$$|1 - \mu \lambda_i| < 1 \quad \forall i$$

To make sure that all components converge, we need

$$|1 - \mu \lambda_{max}| < 1$$

$$-1 < 1 - \mu \lambda_{max} < 1$$

$$-2 < -\mu \lambda_{max} < 0$$

$$0 < \mu \lambda_{max} < 2$$

$$\boxed{0 < \mu < \frac{2}{\lambda_{max}}}$$

Each component converges exponentially with factor $(1 - \mu \lambda_i)$

If want to associate a time constant to each R.S mode we set

$$1-\mu\lambda_i = e^{-\frac{1}{\tau_i}} \qquad \forall i = 1, \dots, N$$

$$\tau_i = -\frac{1}{\ln(1-\mu\lambda_i)}$$

τ_i is the time it takes to reach $\frac{1}{e}$ of $V_i(0)$.

If $\mu\lambda_i$ is small $1-\mu\lambda_i$ is around 1 and negative

using $\ln x \simeq x-1$, $\ln(1-\mu\lambda_i) \simeq 1-\mu\lambda_i - 1$

$$\boxed{\tau_i \simeq \frac{1}{\mu\lambda_i}}$$

More detailed analysis of the evolution of $c(k)$, can be obtained observing that

$$v(k) = Q\, v(k) \qquad (Q \text{ is unitary})$$

Therefore

$$\underline{c}(k) = \underline{c}^0 + \underline{\underline{Q}}\,\underline{v}(k)$$

$$c(k) = \underline{c}^0 + \sum_{i=1}^{N} q_i\, v_i(k)$$

Each element can be written as

$$c_j(k) = c_j^0 + \sum_{i=1}^{N} q_{ij}\, v_i(0)\,(1-\mu\lambda_i)^k \qquad j = 1, \dots N$$

The overall behaviour is the composition of N modes. The slowest mode is governed by the largest eigenvalue and the fastest by the smallest one.

The time to reach the solution will be within the bounds

$$\frac{-1}{\log(1-\mu\lambda_{max})} < \tau_a < \frac{-1}{\log(1-\mu\lambda_{min})}$$

SEARCH DYNAMICS : (SQUARED ERROR EVOLUTION)

In one of the previous derivations, we have showed that the error can be expressed in the following canonical form

$$\mathcal{E}(\underline{c}(k)) = \underbrace{\mathcal{E}(\underline{c}^o)}_{\mathcal{E}^o} + (\underline{c}(k)-\underline{c}^o)^T \underline{\underline{R}}_x (\underline{c}(k)-\underline{c}^o) \quad (1)$$
$$\mathcal{E}(k)$$

Using the decomposition introduced above for the coefficient evolution, we have

$$\mathcal{E}(k) = \mathcal{E}_o + \underline{v}(k)\,\underline{\underline{Q}}\,\underline{\underline{\Lambda}}_x\,\underline{\underline{Q}}^T\,\underline{v}(k)$$

$$= \mathcal{E}_o + \underline{v}'(k)\,\underline{\underline{\Lambda}}_x\,\underline{v}(k)$$

$$= \mathcal{E}_o + \sum_{i=1}^{N} |v_i(k)|^2 \lambda_i$$

$$= \mathcal{E}_o + \sum_{i=1}^{N} \lambda_i (1-\mu\lambda_i)^{2k} v_i^2(0)$$

Under the same conditions found above $(|1-\mu\lambda_i| < 1)$ $\mathcal{E}(k) \to \mathcal{E}_o$

(1) Easy to verify expanding the quadratic forms.

Now since the error is the superposition of N modes, just as before, we have

$$\tau_{i\varepsilon} \simeq -\frac{1}{2\ln(1-\mu\lambda_i)} \simeq \frac{1}{2\mu\lambda_i}$$

The mean squared error $E(k)$ converges in general twice as fast as the coefficients.

STOCHASTIC METHODS
(THE LMS ALGORITHM)

The Newton gradient descent method, presented in the previous section, requires the preliminary computation of correlation matrix $\underline{R}_x$ and the cross-correlation vector $\underline{r}_{dx}$. It is possible to avoid this step and direct a search that is directly driven by the data. (DATA-BASED METHODS).

We start from considering the availability of a "TRAINING SET" of pairs

$$\{(\underline{x}[n], d[n]), n = 1, \dots L\}$$

A current version of the filter $\underline{c}(k)$ correspond to the instantaneous error

$$\mathcal{E}[n] = d[n] - \underline{c}^T(k)\underline{x}[n]$$

and its squared value

$$\mathcal{E}^2[n] = (d[n] - c^T(k)x[n])^2$$

The instantaneous gradient of $\mathcal{E}^2[n]$ with respect to $c(k)$ is

$$\nabla_{\underline{c}(k)} \mathcal{E}^2[n] = 2(d[n] - \underline{c}^T[k]\underline{x}[n])\underline{x}[n]$$

Now if we decide to make a small step of adaptation $\forall n$, in the direction opposite to the current gradient, we have

$$c(n) = \underline{\hat{c}}(n-1) - \mu \underline{x}[n] \left(d[n] - \underline{\hat{c}}^T(n-1) \underline{x}[n] \right)$$

this is the famous LMS algorithm.

Despite the very rough idea of making an adaptation step on every example, the algorithm converges if μ is small and if we present the examples many times.

To explain why these updates may work, consider the evolution of their means.

$$E[c[n]] = E[c[n-1]] - \mu \left(E[x[n] d[n]] - E[x[n] \underline{x^T[n] c(n-1)}] \right)$$

$$= E[c[n-1]] - \mu \left(\underline{r}_{dx} - \underline{E[x[n] y[n]]} \right)$$

$$= E[x[n] x^T[n]] \underline{\hat{c}}^o$$

if $y[n] = y^o[n]$

in optimal conditions

Therefore it looks like the mean evolution of the algorithm follows the dynamics studied for the Newton method.

A complete analysis of the LMS algorithm is beyond the scope of these lectures, and can be found in classical texts on adaptive signal processing [Haykin, Adaptive Signal Processing].

In practice the convergence will be very noisy and μ has to be carefully controlled to avoid excessive wandering around and excessively low speed. In fact μ can be

made time varying $\mu(n)$ and the data in the training set must usually be presented many times. Each time the training set is presented, we say that an "epoque" has passed.

Various ideas have been proposed in the literature to obtain convergence and accuracy. For example, the data could be shuffled at every epoque. The step size parameter $\mu(k)$ could be adapted according to the current convergence speed (ADAM algorithm)

The stochastic search of the LMS algorithm is the precursor of the famous backpropagation algorithm that will be presented for the neural network architectures in the following lectures where it is impossible to derive analytical solutions. In all cases the search follows a noisy gradient descend path.