Lecture 13: Survey of adaptive filtering methods

Overview

- Basic problems
  - Specify the model to be used (linear, nonlinear, structure)
  - Specify the given data
  - Specify the optimality criterion to be satisfied
  - Specify the parameters to be found
- Fundamental tools
  - Handling the expectation operator
  - Computing the gradient of a given performance criterion
  - Optimal Wiener filter
  - Computing the optimal value of a performance criterion
- Basic algorithmic structures
  - Steepest descent algorithm
  - LMS algorithm
  - LMS variants
  - Frequency-domain LMS

  - Levinson – Durbin algorithm
  - Recursive Least Squares algorithm
  - Back-propagation algorithm

- Applications
  - Adaptive noise cancellation
  - Adaptive Channel equalization
  - Adaptive echo cancellation

Optimum linear Filtering: Problem statement *Lecture 2*

- Given the set of input samples $\{u(0), u(1), u(2), \ldots\}$ and the set of desired response $\{d(0), d(1), d(2), \ldots\}$
- In the family of filters computing their output according to

$$y(n) = \sum_{k=0}^{\infty} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{1}$$

- Find the parameters $\{w_0, w_1, w_2, \ldots\}$ such as to minimize the mean square error defined as

$$J = E[e(n)^2] \tag{2}$$

where the error signal is

$$e(n) = d(n) - y(n) = d(n) - \sum_{l=0}^{\infty} w_l u(n-l) \tag{3}$$

□

The family of filters (5) is the family of linear discrete time filters (IIR or FIR).

Adaptive filtering: Problem statement *Lecture 3*

- Consider the family of variable parameter FIR filters, computing their output according to

$$\begin{aligned} y(n) &= w_0(n)u(n) + w_1(n)u(n-1) + \dots w_{M-1}(n)u(n-M+1) \\ &= \sum_{k=0}^{M-1} w_k(n)u(n-k) = \underline{w}(n)^T \underline{u}(n), \quad n = 0, 1, 2, \dots, \infty \end{aligned}$$

  where parameters $\underline{w}(n)$ are allowed to change at every time step, $u(t)$ is the input signal, $d(t)$ is the desired signal and $\{u(t)$ and $d(t)\}$ are jointly stationary.

- Given the parameters $\underline{w}(n) = [w_0(n), w_1(n), w_2(n), \dots, w_{M-1}(n)]^T$, find an adaptation mechanism $\underline{w}(n+1) = \underline{w}(n) + \delta\underline{w}(n)$, or written componentwise

$$\begin{aligned} w_0(n+1) &= w_0(n) + \delta w_0(n) \\ w_1(n+1) &= w_1(n) + \delta w_1(n) \\ &\dots \\ w_{M-1}(n+1) &= w_{M-1}(n) + \delta w_{M-1}(n) \end{aligned}$$

  such as the adaptation process converges to the parameters of the optimal Wiener filter, $\underline{w}(n+1) \rightarrow \underline{w}_o$, no matter where the iterations are initialized (i.e. $\forall\ \underline{w}(0)$).

□

Linear LS estimation problem *Lecture 9*

Problem statement

- Given the set of input samples $\{u(1), u(2), \ldots, u(N)\}$ and the set of desired response $\{d(1), d(2), \ldots, d(N)\}$
- In the family of linear filters computing their output according to

$$y(n) = \sum_{k=0}^{M-1} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{4}$$

- Find the parameters $\{w_0, w_1, \ldots, w_{M-1}\}$ such as to minimize the sum of error squares

$$\mathcal{E}(w_0, w_1, \ldots, w_{M-1}) = \sum_{i=i_1}^{i_2} [e(i)^2] = \sum_{i=i_1}^{i_2} [d(i) - \sum_{k=0}^{M-1} w_k u(i-k)]^2$$

  where the error signal is

$$e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k u(i-k)$$

□

Recursive Least Squares Estimation *Lecture 10*

Problem statement

- Given the set of input samples $\{u(1), u(2), \ldots, u(N)\}$ and the set of desired response $\{d(1), d(2), \ldots, d(N)\}$
- In the family of linear filters computing their output according to

$$y(n) = \sum_{k=0}^{M} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{5}$$

- Find *recursively in time* the parameters $\{w_0(n), w_1(n), \ldots, w_{M-1}(n)\}$ such as to minimize the sum of error squares

$$\mathcal{E}(n) = \mathcal{E}(w_0(n), w_1(n), \ldots, w_{M-1}(n)) = \sum_{i=i_1}^{n} \beta(n,i)[e(i)^2] = \sum_{i=i_1}^{n} \beta(n,i)[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2$$

where the error signal is

$$e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)$$

and the forgetting factor or weighting factor reduces the influence of old data

$$0 < \beta(n,i) \leq 1, \qquad i = 1, 2, \ldots, n$$

usually taking the form $(0 < \lambda < 1)$

$$\beta(n,i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n$$

□

Fundamental tools

- Handling the expectation operator

  Lecture 2

Now we concentrate to find $r_x(0), r_x(1)$ for the AR process

$$x(n) + a_1 x(n-1) + a_2 x(n-2) = v(n)$$

First multiply in turn the equation with $x(n)$, $x(n-1)$ and $x(n-2)$ and then take the expectation

$$Ex(n)\times \rightarrow \qquad Ex(n)x(n) + Ex(n)a_1 x(n-1) + Ex(n)a_2 x(n-2) = Ex(n)v(n)$$
$$\text{resulting in} \qquad r_x(0) + a_1 r_x(1) + a_2 r_x(2) = Ex(n)v(n) = \sigma_v^2$$

The equality $Ex(n)v(n) = \sigma_v^2$ can be obtained multiplying the AR model difference equation with $v(n)$ and then taking expectations

$$Ev(n)\times \rightarrow \qquad Ev(n)x(n) + Ev(n)a_1 x(n-1) + Ev(n)a_2 x(n-2) = Ev(n)v(n)$$
$$\text{resulting in} \qquad Ev(n)x(n) = \sigma_v^2$$

- Computing the gradient of a given performance criterion

Lecture 2

Define the gradient operator ∇, having its k-th entry

$$\nabla_k = \frac{\partial}{\partial w_k} \tag{6}$$

and thus, the $k-$th entry of the gradient of criterion J is (remember, $e(n) = d(n) - \sum_{l=0}^{\infty} w_l u(n-l)$)

$$\nabla_k\ J \ = \ \frac{\partial J}{\partial w_k} = 2E\left[e(n)\frac{\partial e(n)}{\partial w_k}\right] = -E\left[e(n)u(n-k)\right]$$

For the criterion to attain its minimum, the gradient of the criterion must be identically zero, that is

$$\nabla_k\ J = 0, \quad k = 0, 1, 2, \ldots$$

resulting in the fundamental

Principle of ortogonality: $$E\left[e_o(n)u(n-k)\right] \ = \ 0, \quad k = 0, 1, 2, \ldots \tag{7}$$

Lecture 2

Mean square error surface

Then the cost function can be written as

$$J_{\underline{w}} = \sigma_d^2 - 2\sum_{i=0}^{M-1} p(-i)w_i + \sum_{l=0}^{M-1}\sum_{i=0}^{M-1} w_l w_i R_{i,l} \tag{8}$$

J attains the minimum, J_{min}, where the gradient is zero

$$\begin{aligned} \nabla_{\underline{w}} J &= 0 \\ \frac{\partial J}{\partial w_k} &= 0, \quad k = 0, 1, \ldots, M-1 \\ \frac{\partial J}{\partial w_k} &= -2p(-k) + 2\sum_{l=0}^{M-1} w_l r(k-l) = 0, \quad k = 0, 1, \ldots, M-1 \end{aligned}$$

which finally gives the same Wiener – Hopf equations

$$\sum_{l=0}^{M-1} w_l r(k-l) = p(-k) \tag{9}$$

Lecture 7

At stage m the optimality criterion is:

$$J_m = E[f_m^2(n)] + E[b_m^2(n)]$$

and using the stage m equations

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + \Gamma_m b_{m-1}(n-1) \\ b_m(n) &= b_{m-1}(n-1) + \Gamma_m f_{m-1}(n) \end{aligned}$$

$$\begin{aligned} J_m &= E[f_m^2(n)] + E[b_m^2(n)] = E[(f_{m-1}(n) + \Gamma_m b_{m-1}(n-1))^2] + E[(b_{m-1}(n-1) + \Gamma_m f_{m-1}(n))^2] \\ &= E[(f_{m-1}^2(n) + b_{m-1}^2(n-1)](1 + \Gamma_m^2) + 4\Gamma_m E[b_{m-1}(n-1) f_{m-1}(n)] \end{aligned}$$

Taking now the derivative with respect to Γ_m of the above criterion we obtain

$$\frac{d(J_m)}{d\Gamma_m} = 2E[(f_{m-1}^2(n) + b_{m-1}^2(n-1)]\Gamma_m + 4E[b_{m-1}(n-1) f_{m-1}(n)] = 0$$

and therefore

$$\Gamma_m^* = -\frac{2E[b_{m-1}(n-1) f_{m-1}(n)]}{E[(f_{m-1}^2(n)] + E[b_{m-1}^2(n-1)]}$$

Lecture 7

Imposing the same optimality criterion as in Burg method

$$J_m = E[f_m^2(n)] + E[b_m^2(n)]$$

the gradient method applied to the lattice filter parameter at stage m is

$$\frac{d(J_m)}{d\Gamma_m} = 2E[f_m(n)b_{m-1}(n-1) + f_{m-1}(n)b_m(n)]$$

and can be approximated (as usually in LMS algorithms) by

$$\hat{\nabla} J_m \approx 2[f_m(n)b_{m-1}(n-1) + f_{m-1}(n)b_m(n)]$$

We obtain the updating equation for the parameter Γ_m

$$\Gamma_m(n+1) = \Gamma_m(n) - \frac{1}{2}\mu_m(n)\hat{\nabla} J_m = \Gamma_m(n) - \mu_m(n)(f_m(n)b_{m-1}(n-1) + f_{m-1}(n)b_m(n))$$

- Optimal Wiener filter

  Lecture 2

Matrix formulation of Wiener – Hopf equations

Let us denote

$$\underline{u}(n) = \begin{bmatrix} u(n) & u(n-1) & u(n-2) & \ldots & u(n-M+1) \end{bmatrix}^T$$

$$R = E[\underline{u}(n)\underline{u}^T(n)] = E \begin{bmatrix} u(n) \\ u(n-1) \\ u(n-2) \\ u(n-M+1) \end{bmatrix} \begin{bmatrix} u(n) & u(n-1) & u(n-2) & \ldots & u(n-M+1) \end{bmatrix} \quad (10)$$

$$= \begin{bmatrix} r(0) & r(1) & \ldots & r(M-1) \\ r(1) & r(0) & \ldots & r(M-2) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r(M-1) & r(M-2) & \ldots & r(0) \end{bmatrix}$$

$$\underline{p} = E[\underline{u}(n)d(n)] = \begin{bmatrix} p(0) & p(-1) & p(-2) & \ldots & p(1-M) \end{bmatrix}^T \quad (11)$$

$$\underline{w}_0 = \begin{bmatrix} w_{o,0} & w_{o,1} & \ldots & w_{o,M-1} \end{bmatrix}^T \quad (12)$$

then Wiener – Hopf equations can be written in a compact form

$$R\underline{w}_0 = \underline{p} \quad \text{with solution} \quad \underline{w}_o = R^{-1}\underline{p} \quad (13)$$

- Computing the optimal value of a performance criterion *Lecture 2*

  Minimum Mean square error

  Using the form of the criterion

$$J_{\underline{w}} = \sigma_d^2 - 2\underline{p}^T \underline{w} + \underline{w}^T R \underline{w} \tag{14}$$

  one can find the value of the minimum criterion (remember, $R\underline{w}_0 = \underline{p}$ and $\underline{w}_o = R^{-1}\underline{p}$):

$$\begin{aligned} J_{\underline{w}_o} &= \sigma_d^2 - 2\underline{p}^T \underline{w}_o + \underline{w}_o^T R \underline{w}_o = \sigma_d^2 - 2\underline{w}_o^T R \underline{w}_o + \underline{w}_o^T R \underline{w}_o \\ &= \sigma_d^2 - \underline{w}_o^T R \underline{w}_o \\ &= \sigma_d^2 - \underline{w}_o^T \underline{p} \\ &= \sigma_d^2 - \underline{p}^T R^{-1} \underline{p} \end{aligned} \tag{15}$$

Basic algorithmic structures

- Steepest descent algorithm **SD ALGORITHM**

  Steepest descent search algorithm for finding the Wiener FIR optimal filter

  Given

  - the autocorrelation matrix $R = E\underline{u}(n)\underline{u}^T(n)$
  - the cross-correlation vector $\underline{p}(n) = E\underline{u}(n)d(n)$

  Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$.

  Iterate for $n = 0, 1, 2, 3, \ldots, n_{max}$

  $$\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - R\underline{w}(n)]$$

  Stop iterations if $\|\underline{p} - R\underline{w}(n)\| < \varepsilon$

  $\square$

  Designer degrees of freedom: $\mu, \varepsilon, n_{max}$

- LMS algorithm

LMS algorithm

Given
- the (correlated) input signal samples $\{u(1), u(2), u(3), \ldots\}$, generated randomly;
- the desired signal samples $\{d(1), d(2), d(3), \ldots\}$ correlated with $\{u(1), u(2), u(3), \ldots\}$

1 Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$, for example $\underline{w}(0) = 0$.

2 Iterate for $n = 0, 1, 2, 3, \ldots, n_{max}$

2.0 Read /generate a new data pair, $(\underline{u}(n), d(n))$

2.1 (Filter output) $y(n) = \underline{w}(n)^T \underline{u}(n) = \sum_{i=0}^{M-1} w_i(n) u(n-i)$

2.2 (Output error) $e(n) = d(n) - y(n)$

2.3 (Parameter adaptation) $\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n) e(n)$

or componentwise

$$\begin{bmatrix} w_0(n+1) \\ w_1(n+1) \\ \cdot \\ \cdot \\ \cdot \\ w_{M-1}(n+1) \end{bmatrix} = \begin{bmatrix} w_0(n) \\ w_1(n) \\ \cdot \\ \cdot \\ \cdot \\ w_{M-1}(n) \end{bmatrix} + \mu e(n) \begin{bmatrix} u(n) \\ u(n-1) \\ \cdot \\ \cdot \\ \cdot \\ u(n-M+1) \end{bmatrix}$$

□

The complexity of the algorithm is $2M + 1$ multiplications and $2M$ additions per iteration.

- LMS variants
  - Normalized LMS

$$\boxed{w_j(n+1) = w_j(n) + \frac{\tilde{\mu}}{a + \|\underline{u}(n)\|^2} e(n)u(n-j)}$$

  - Sign algorithms
    * The Sign algorithm (other names: pilot LMS, or Sign Error)

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n)\ sgn(e(n))$$

    * The Clipped LMS (or Signed Regressor)

$$\underline{w}(n+1) = \underline{w}(n) + \mu\ sgn(\underline{u}(n))e(n)$$

    * The Zero forcing LMS (or Sign Sign)

$$\underline{w}(n+1) = \underline{w}(n) + \mu\ sgn(\underline{u}(n))\ sgn(e(n))$$

  - Momentum LMS algorithm

$$\boxed{\underline{w}(n+1) - \underline{w}(n) = \gamma(\underline{w}(n) - \underline{w}(n-1)) + \tilde{\mu}(1-\gamma)e(n)\underline{u}(n)}$$

- Levinson – Durbin algorithm

Levinson – Durbin recursions

$$\underline{a}_m = \begin{bmatrix} \underline{a}_{m-1} \\ 0 \end{bmatrix} + \Gamma_m \begin{bmatrix} 0 \\ \underline{a}^B_{m-1} \end{bmatrix} \qquad \text{Vector form of L – D recursions}$$

$$a_{m,k} = a_{m-1,k} + \Gamma_m a_{m-1,m-k}, \qquad k = 0, 1, \ldots, m \qquad \text{Scalar form of L – D recursions}$$

$$\Delta_{m-1} = \underline{a}^T_{m-1} \underline{r}^B_m = r(m) + \sum_{k=1}^{m-1} a_{m-1,k} r(m-k)$$

$$\Gamma_m = -\frac{\Delta_{m-1}}{P_{m-1}}$$

$$P_m = P_{m-1}(1 - \Gamma^2_m)$$

- RLS algorithm (Lecture 10)

  All necessary equations to form the RLS algorithm:

$$\begin{aligned}
\underline{k}(n) &= \frac{\lambda^{-1}P(n-1)\underline{u}(n)}{1+\lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} \\
\alpha(n) &= d(n) - \underline{u}(n)^T\underline{w}(n-1) \\
\underline{w}(n) &= \underline{w}(n-1) + \underline{k}(n)\alpha(n) \\
P(n) &= \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)
\end{aligned}$$

Delta Rule Algorithm

Given
- the (correlated) input vector samples $\{\underline{u}(1), \underline{u}(2), \underline{u}(3), \ldots\}$, generated randomely;
- the desired signal samples $\{d(1), d(2), d(3), \ldots\}$

1 Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$, for example $\underline{w}(0) = 0$.

2 Iterate for $t = 0, 1, 2, 3, \ldots, n_{max}$

2.0	Read a new data pair,	$(\underline{u}(t), d(t))$
2.1	(Compute the output)	$y(t) = h[\underline{w}(t)^T \underline{u}(t)] = h[\sum_{i=1}^{N} w_i(t) u_i(t)]$
2.2	(Compute the error)	$e(t) = d(t) - y(t)$
2.3	(Parameter adaptation)	$\underline{w}(t+1) = \underline{w}(t) + \mu \underline{u}(t) e(t) h'[\underline{w}^T(t) \underline{u}(t)]$

□

Sample examination

1. State the problem of optimal filter design for the forward predictor (model, data available, criterion to be minimized).

2. Consider the predictor of order 1

$$\hat{u}(n) = au(n-1)$$

   - a) Compute the optimal value of the parameter a, as a function of autocorrelation values of the process $u(n)$
   - b) Draw the lattice filter structure of the predictor.
   - c) Compute the optimal parameters of the lattice predictor.

3. Consider the RLS algorithm:

$$\begin{aligned}
\underline{k}(n) &= \frac{\lambda^{-1}P(n-1)\underline{u}(n)}{1+\lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} \\
\alpha(n) &= d(n) - \underline{u}(n)^T\underline{w}(n-1) \\
\underline{w}(n) &= \underline{w}(n-1) + \underline{k}(n)\alpha(n) \\
P(n) &= \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)
\end{aligned}$$

   Explain what are the variables to be initialized at time $t = 0$.

4. Define a sigmoidal perceptron and derive the adaptation policy for it. Draw a block diagram representing the adaptation process for a sigmoidal perceptron.

5.
    - Write the steepest descent adaptive algorithm for the FIR filter of order 1.
    - Use an example to show explicitly the computations required for the first two iterations of the algorithm.

6. Application description: Draw the structure of an adaptive noise canceller. Discuss the significance of each signal.