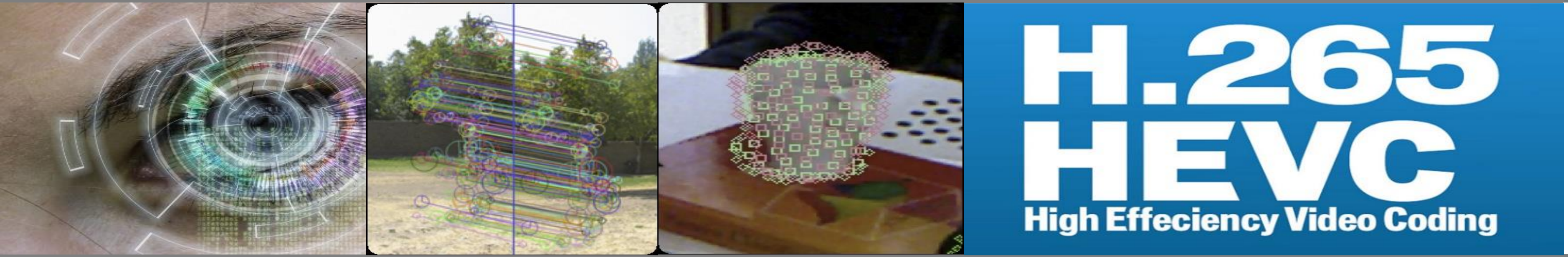
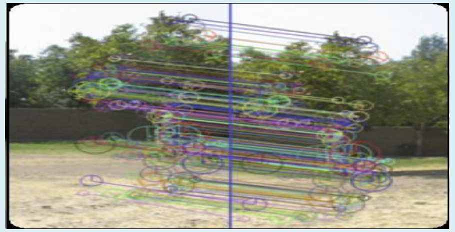
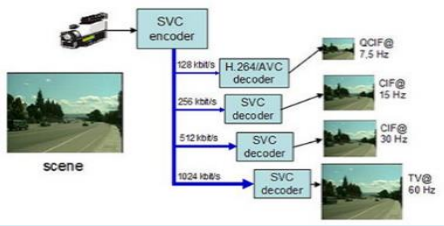
Adaptive System and Signal Processing Theory
(#9: Method of Steepest Descent & Least Mean Square)



2023. Spring

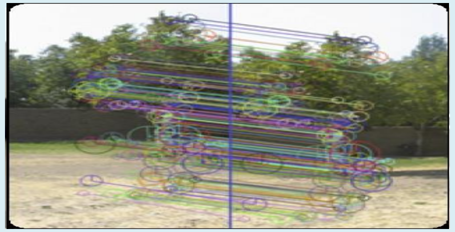
Prof. Byung-Gyu Kim
Intelligent Vision Processing Lab. (IVPL)
http://ivpl.sookmyung.ac.kr
Dept. of IT Engineering, Sookmyung Women's University
E-mail: bg.kim@ivpl.sookmyung.ac.kr

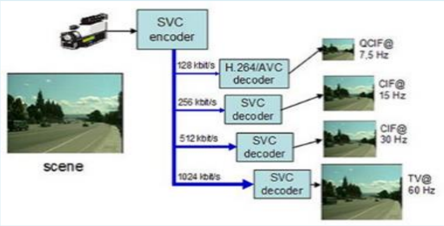




Contents

- Method of Steepest Descent
- Least-Mean-Square (LMS) Algorithm





Contents

- **Method of Steepest Descent**
- Least-Mean-Square (LMS) Algorithm

- Steepest Descent Method
  - Gradient-based adaptation of tap-weight vector of Linear FIR filter.
  - Using a deterministic feedback system that finds the minimum point of the error-performance surface without any knowledge of the surface itself.

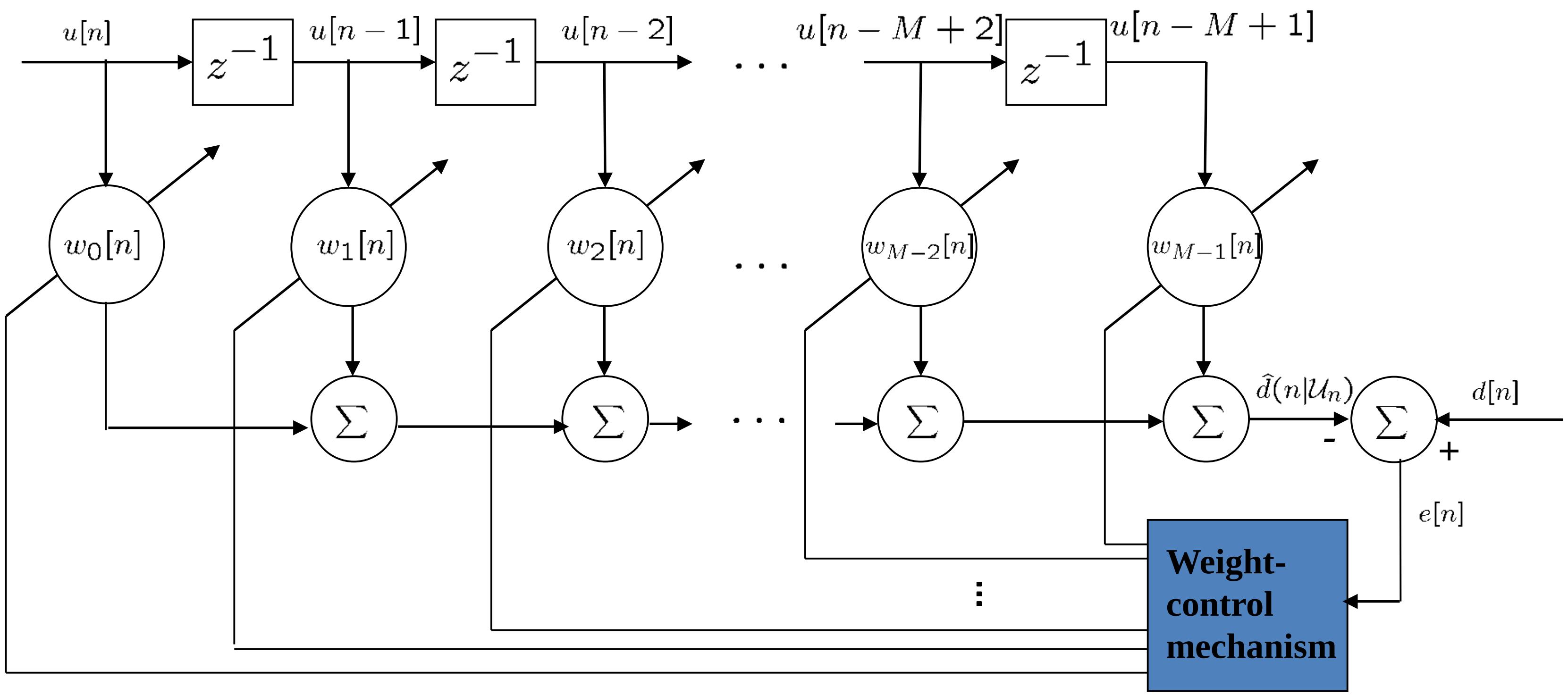
Wide-sense stochastic process of zero mean and the correlation matrix $\mathbf{R}$

- Some Preliminaries
  - Tap inputs : $u[n],\ u[n-1],\cdots,\ u[n-(M-1)]$
  - Tap weights: $w_0(n), w_1(n), \cdots, w_{M-1}(n)$
  - Desired response: $d[n]$
  - $\mathcal{U}_n$ is the space spanned by the tap inputs $u[n],\ u[n-1],\cdots,\ u[n-(M-1)]$.
  - The estimate of the desired response: $\hat{d}(n|\mathcal{U}_n)$
  - The estimation error

$$\begin{aligned} e[n] &= d[n] - \hat{d}[n|\mathcal{U}_n] \\ &= d[n] - \mathbf{w}^H[n]\mathbf{u}[n], \qquad (1) \end{aligned}$$

where $\mathbf{w}[n] = [w_0(n), w_1(n), \cdots, w_{M-1}(n)]^T,$

$\mathbf{u}[n] = [u[n],\ u[n-1],\cdots,\ u[n-(M-1)]]^T.$



Structure of adaptive transversal (FIR) filter.

Method of Steepest Descent(3)

- Cost function $J[n]$ (in MSE sense)

$$J[n] = \sigma_d^2 - \sum w_k^* p(-k) - \sum w_k^* p^*(-k) + \sum_k \sum_i w_k^* w_i r(i-k),$$
$$= \sigma_d^2 - \mathbf{w}^H[n]\mathbf{p} - \mathbf{p}\mathbf{w}[n] + \mathbf{w}^H[n]\mathbf{R}\mathbf{w}[n]. \quad (4)$$

Varies with time n.

  - The adaptive process has a task of seeking the bottom or minimum point of the error-performance surface as iteration time *n*.
  - Definitely, at the minimum point, the tap-weight vector is $\mathbf{w}_o$ which is from Wiener-Hopf equation.

- Steepest-Descent Algorithm
  - When input data rate is high and filter contains a large number of tap weights, there exist serious computational difficulties.

  ⇒ An alternative procedure: method of steepest descent.

Method of Steepest Descent(4)

- Procedure of the steepest descent method
  1. With an initial value $\mathbf{w}(0)$ for the tap-weight vector, which provide an initial guess where the minimum point of the error-performance surface may be located. Unless $\mathbf{w}(0)$ is available, $\mathbf{w}(0)$ is usually the null.
  2. With the initial or present guess, we compute the gradient vector which is defined as the derivative of the mean-squared error $J[n]$(on the real and imaginary part).
  3. We compute the next guess at the tap weight vector by making change in the initial or present guess in direction opposite to that of the gradient vector.
  4. We go back to (Step2) and repeat the process.

It is intuitively reasonable that successive corrections to the tap-weight vector in the dircetion of the negative of the gradient vector (i.e. in the direction of the steepest descent of the error-surface) should lead to the minimum mean-squared error $J_{min}(\mathbf{w}_o)$.

Method of Steepest Descent(5)

Let $\nabla J(n)$ denote the value of the gradient vector at time *n* and $\mathbf{w}(n)$ denote the tap-weight vector at time n.

According to the definition of the steepest descent,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)]. \qquad (5)$$

- μ : positive real value (step size) that controls the size of incremental correction as time goes, and

$$\nabla J[n] = \begin{bmatrix} \frac{\partial J[n]}{\partial a_0[n]} + j\frac{\partial J[n]}{\partial b_0[n]} \\ \frac{\partial J[n]}{\partial a_1[n]} + j\frac{\partial J[n]}{\partial b_1[n]} \\ \vdots \\ \frac{\partial J[n]}{\partial a_{M-1}[n]} + j\frac{\partial J[n]}{\partial b_{M-1}[n]} \end{bmatrix} \qquad (6)$$

$$= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n).$$

$$\boxed{\nabla_k J = -2p(-k) + 2\sum_{i=0}^{M-1} w_i r(i-k)}$$

$$\therefore\ \mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)],\ n = 0, 1, 2, \cdots. \qquad (7)$$

Mathematical formulation of the steepest descent algorithm
Feedback model

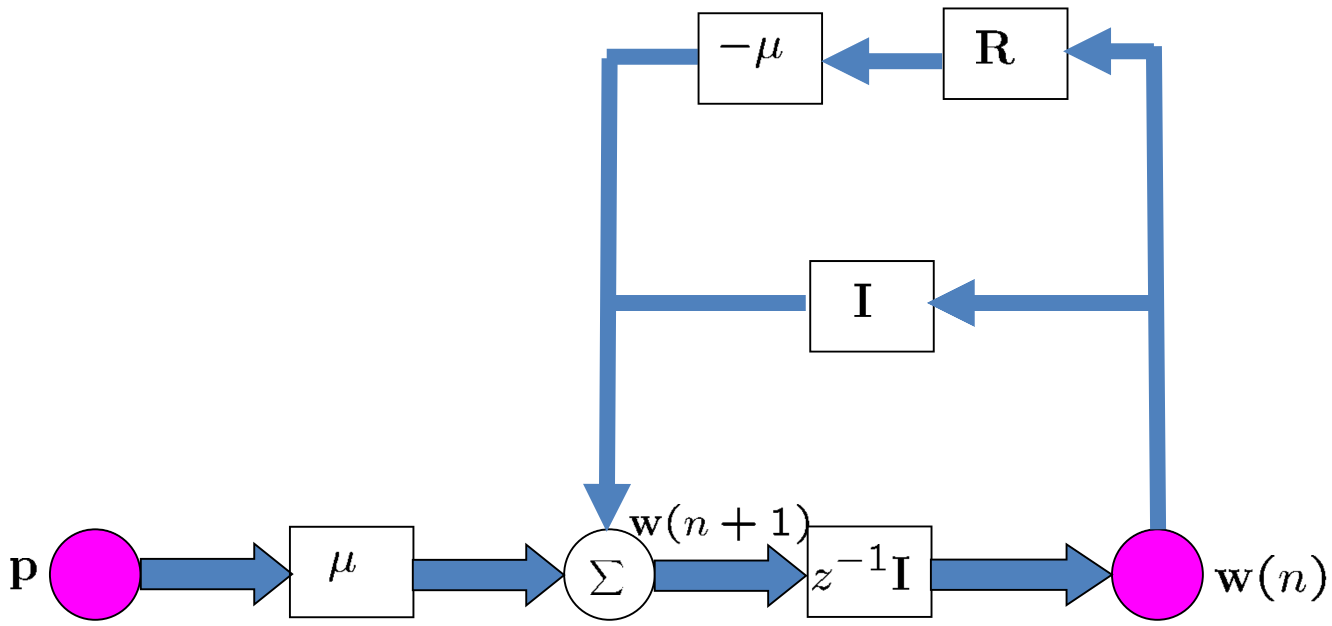
IVPL
Intelligent Vision Processing Lab

- Stability of The Steepest Descent Algorithm

  Since the steepest descent method has a feedback loop,

  - μ (step size)
  - Correlation matrix $\mathbf{R}$ of the tap-weight vector $\mathbf{u}[n]$.

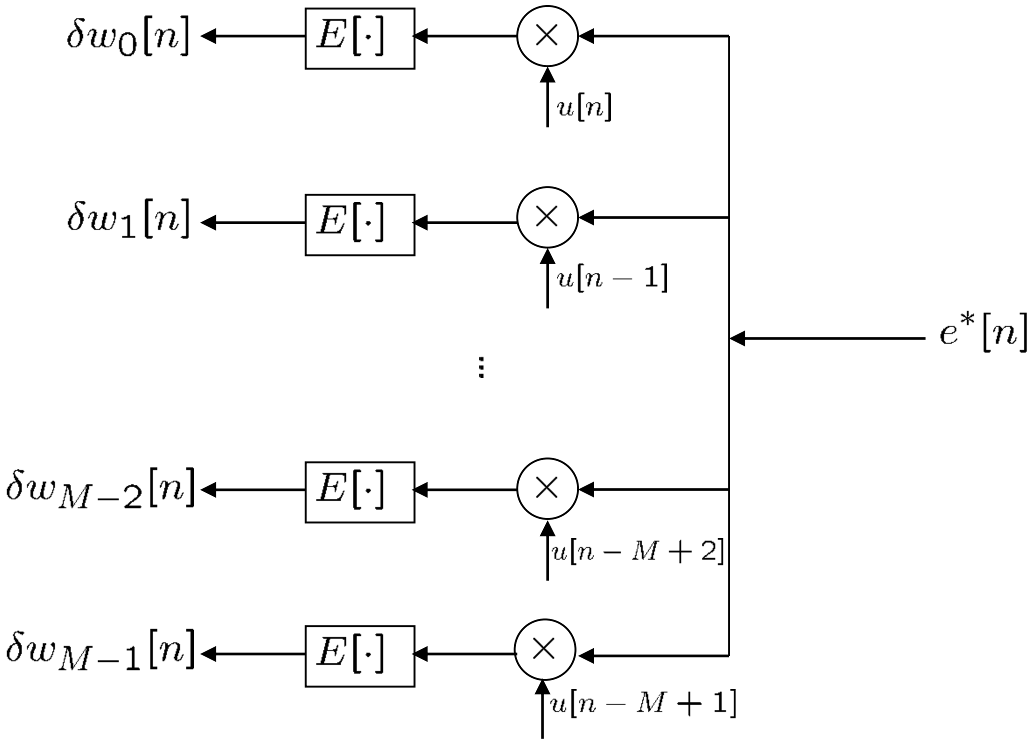
- Condition for stability of ~: $0 < \mu < \dfrac{2}{\lambda_{max}}$ λ_{max} = ***the max. eigen value of*** $\mathbf{R}$.



Signal flow of the steepest descent algorithm.

Method of Steepest Descent(5): Stability of The Steepest Descent Algorithm (2)

- How to select the size of μ ?
  - Not too large, not too small.
  - Because of no exact mathematical analysis, we should evaluate it by experimentation.



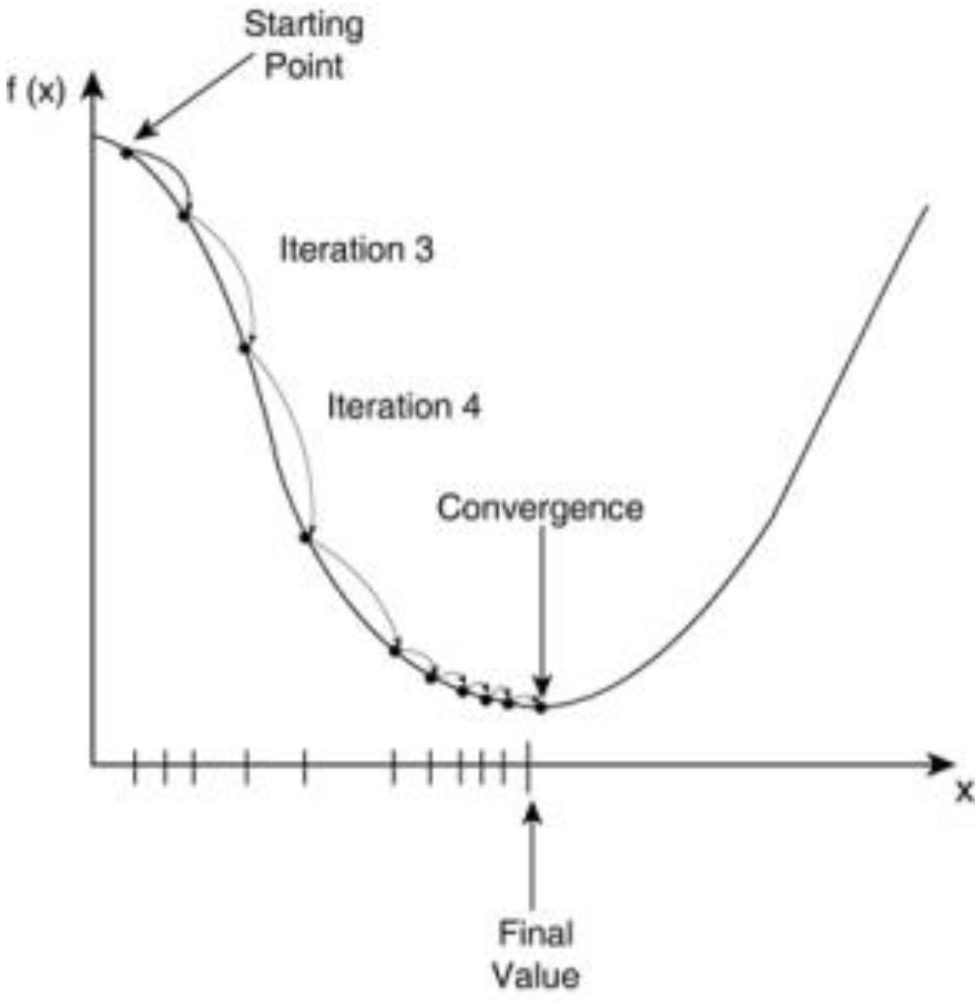
Bank of computing the corrections to the element of tap-weights.

IVPL Intelligent Vision Processing Lab

Method of Steepest Descent(5)

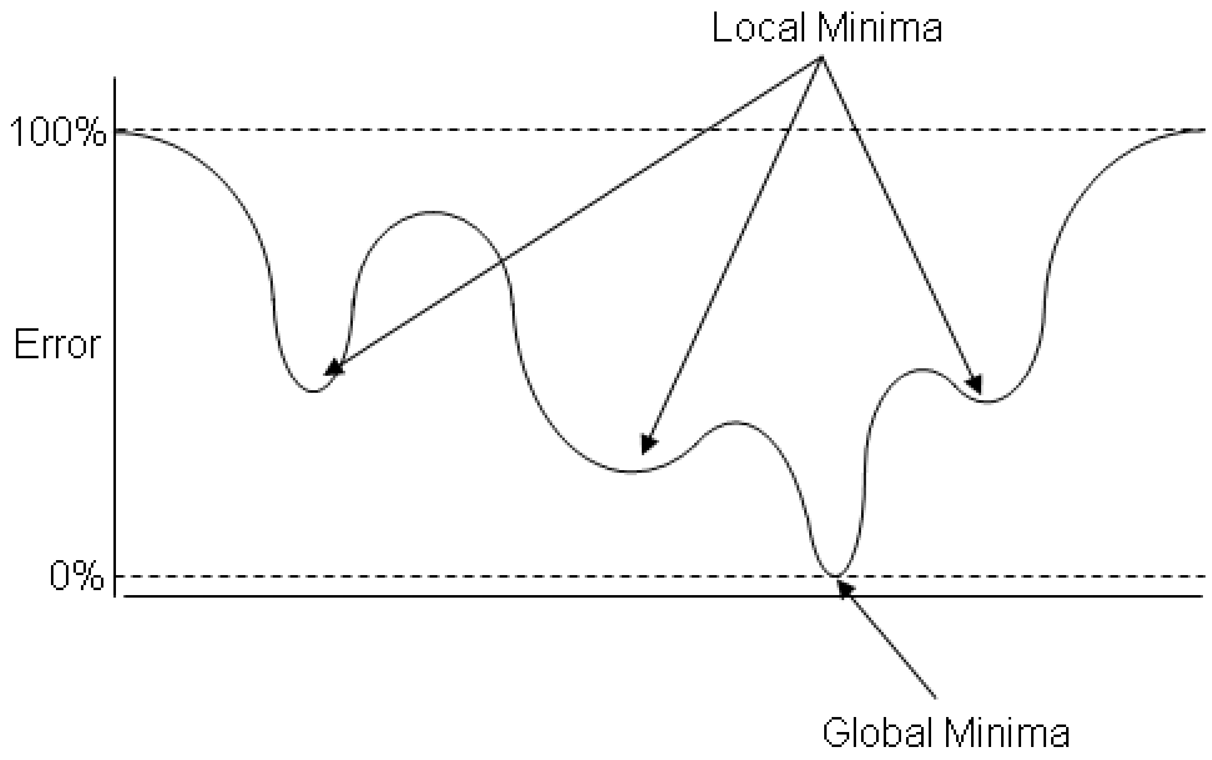
❖ Error Surface: Monotonically Decreasing or Increasing

- One global minimum



Method of Steepest Descent(6)

❖ Error Surface: Non-monotonically Decreasing or Increasing
- Many local minima (minimums)
- One global minimum



- Steepest-Descents Path
  - *Perpendicular to tangent slope at each point.*

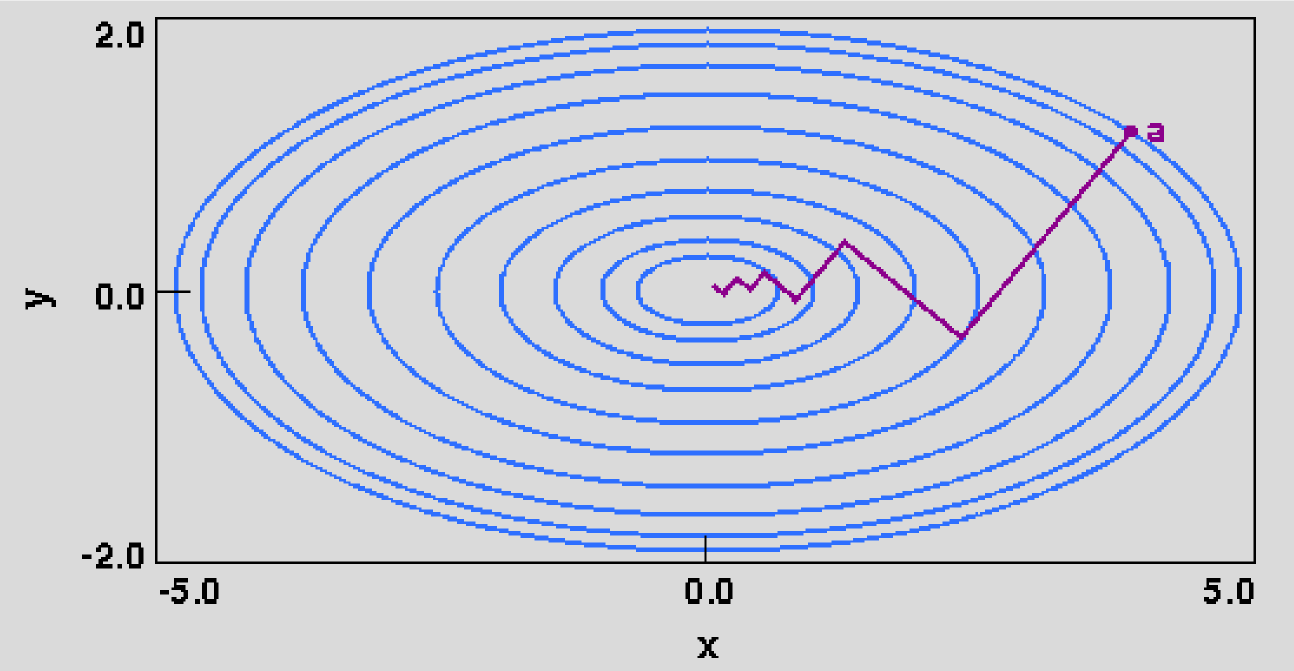
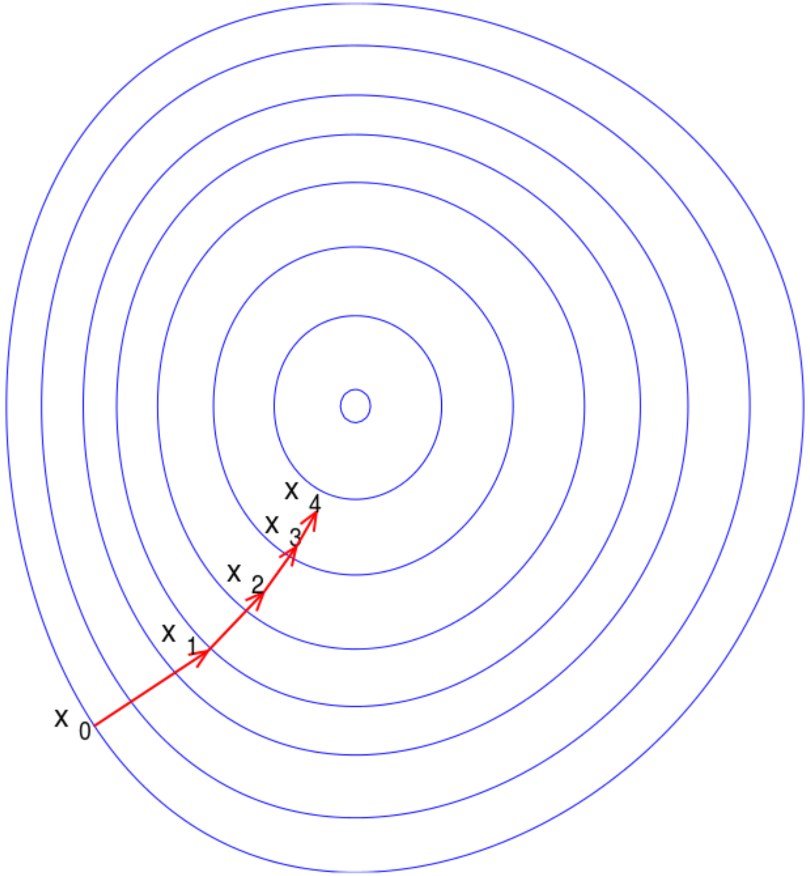
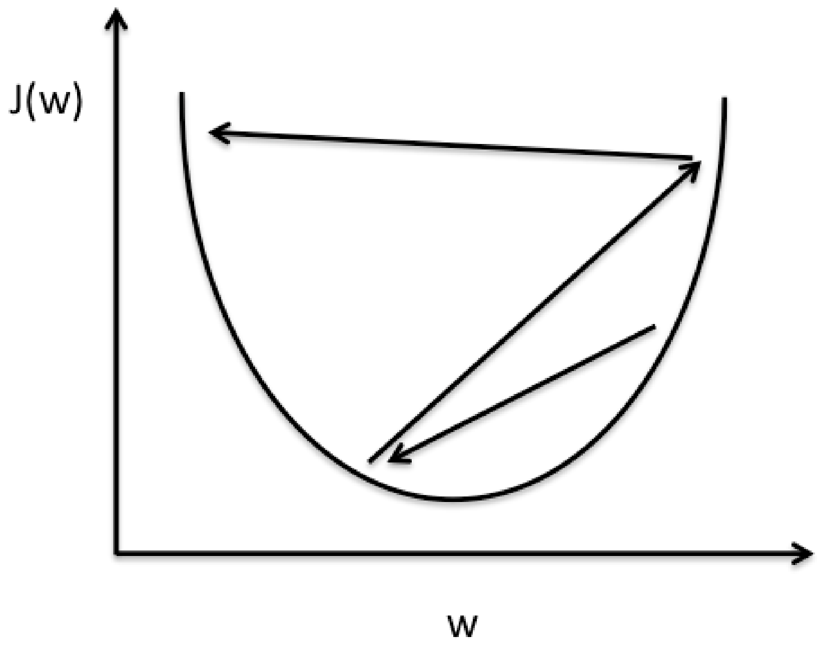


Figure 4–4. Minimization Path following a Steepest-Descents Path

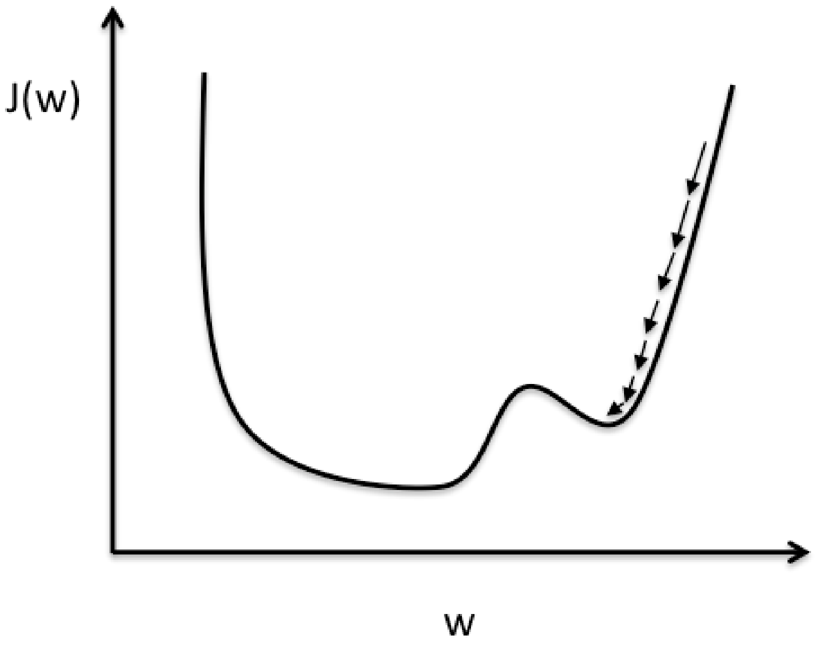


Method of Steepest Descent(8)

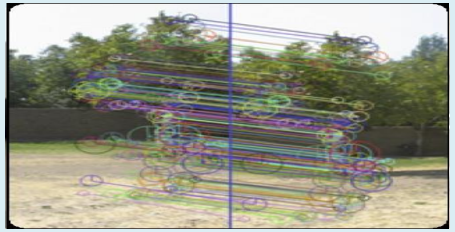
❖ Effect of the size of μ (step size or learning rate)

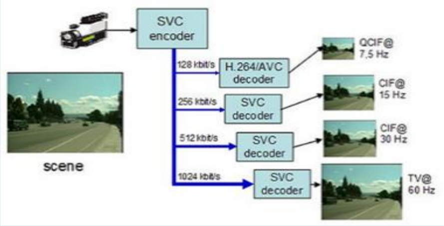


Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.





Contents

- Method of Steepest Descent
- **Least-Mean-Square (LMS) Algorithm**

Least-Mean-Square (LMS) Algorithm (1): Chap. 9 (p.365~)

- Least-Mean-Square (LMS) Algorithm
  - Feature
    - Its simplicity.
    - It does not measurements of the pertinent correlation nor does it require matrix inversion.
- Overview of the LMS Algorithm
  - Two basic operations
    - Filtering process
      - Compute the output of the transversal filter.
      - Generate an estimation error by comparing the output to a desired response.
    - Adaptive process
      - Involves the automatic adjustment of the tap-weights of the filter as the estimation error.

Least-Mean-Square (LMS) Algorithm (2): LMS Adaptation Algorithm (1)

- Least-Mean-Square (LMS) Adaptation Algorithm
  - Problem in reality
    - Exact measurements of the gradient vector ($\nabla J(n)$) are not possible since this would require prior knowledge of both the $\mathbf{R}$ and $\mathbf{p}$.
    - So the gradient vector must be estimated from the available data.
- Development of Estimate of $\nabla J(n)$

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n). \qquad (1)$$

  - Estimates of $\mathbf{R}$ and $\mathbf{p}$.
    - A Simple choice of estimators is to use instantaneous estimates based on sample values of the tap-input vector and desired response as given by

$$\hat{\mathbf{R}}(n) = \mathbf{u}[n]\mathbf{u}^H[n]. \qquad (2)$$

$$\hat{\mathbf{p}}(n) = \mathbf{u}[n]d^*[n]. \qquad (3)$$

Least-Mean-Square (LMS) Algorithm (2): LMS Adaptation Algorithm (2)

Then, the instantaneous gradient vector ($\hat{\nabla} J(n)$) is given as

$$\hat{\nabla} J(n) = -2\mathbf{u}[n]d^*[n] + 2\mathbf{u}[n]\mathbf{u}^H[n]\mathbf{w}(n). \quad (4)$$

- Estimates $\hat{\nabla} J(n)$ may be viewed as the gradient vector ∇ applied to the instantaneous squared error $|e[n]|^2$.

Using the adaptation equation of the steepest descent, we can write new recursion as: With initial guess $\mathbf{w}[0]$,

1. Filter output : $y[n] = \mathbf{w}^H[n]\mathbf{u}[n].$
2. Estimation error: $e[n] = d[n] - y[n].$
3. Tap-weight adaptation: $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}[n] + \mu\mathbf{u}[n]e^*[n].$

Adaptive Least-Mean-Square (LMS) Algorithm

- At each iteration or time update, it also requires knowledge of the most recent values: $\hat{\mathbf{w}}[n],\ d[n],\ \mathbf{u}[n]$
- The estimate of $\mathbf{R}$ and $\mathbf{p}$ have relatively large variance, So it cannot give good performance with a high precision.

IVPL Intelligent Vision Processing Lab

Least-Mean-Square (LMS) Algorithm (2): LMS Adaptation Algorithm (3)

- Steepest Descent Method vs. LMS Algorithm
  - Major difference in the gradient vector $\nabla J(n)$.
    - In LMS algorithm, $u[n-k]e^*[n]$ is used as an estimator of element k in $\nabla J(n)$.
    - Expectation operator is missed out from all the paths of the steepest descent.

  ⇨ This provides simplicity of implementation, but suffers from a gradient noise.

  - Aspect of minimum mean-squared error,
    - Min. mean-squared error:

$$J(n) \rightarrow \hat{J}_{min} \text{ as } n \rightarrow \infty.$$

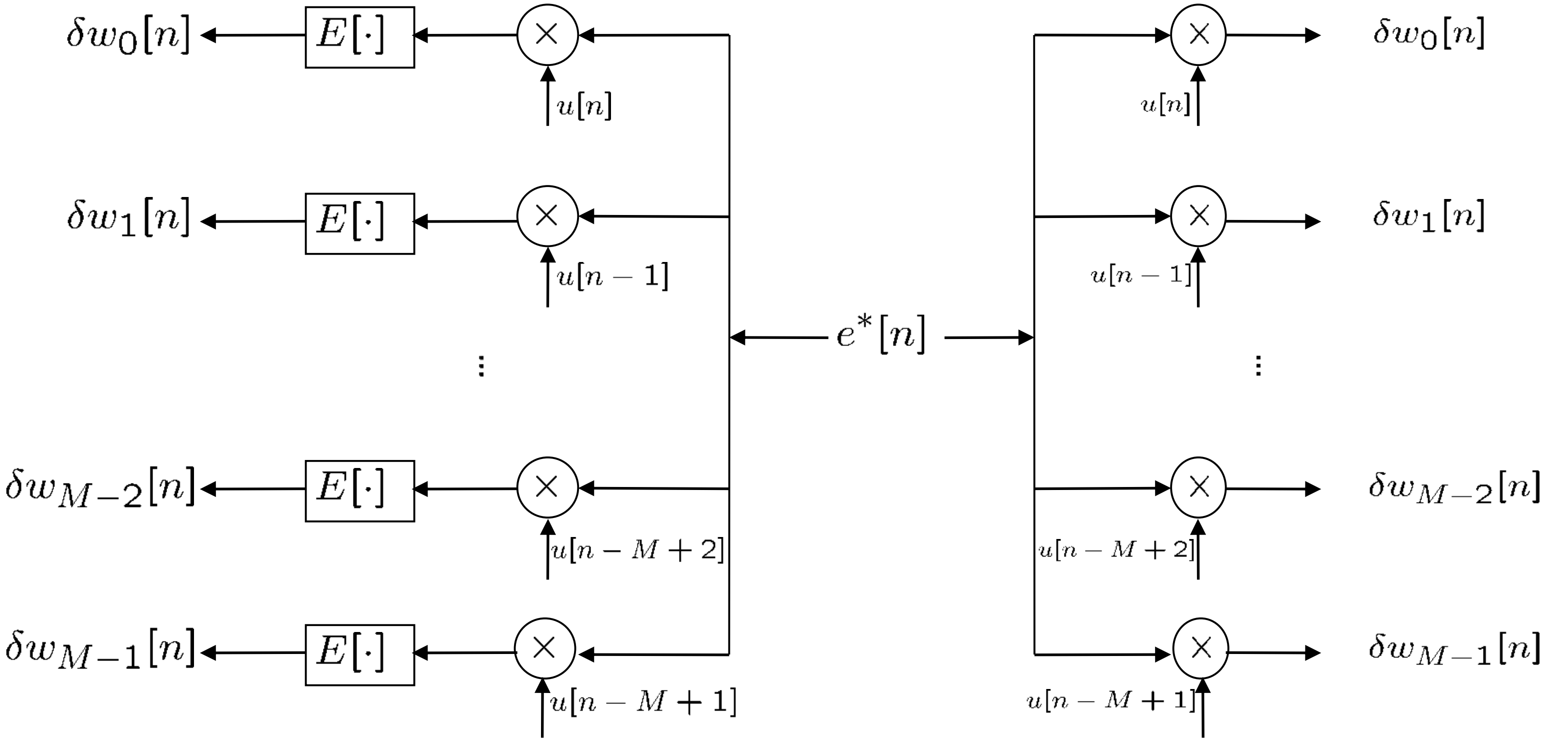
← ***Because of the gradient noise,***

With the optimal solution J_{min}, the difference between two minimum errors can be defined as "excess mean-squared error" by:

$$\begin{aligned} J_{ex}(\infty) &= J(\infty) - J_{min} \\ &= \hat{J}_{min} - J_{min}. \end{aligned} \quad (5)$$

A price paid for using adaptation mechanism to control the tap-weight in the LMS algorithm in place of the steepest descent method.

- Bank of computing correction terms of the Tap-weights vector



The steepest descent method. **The LMS algorithm.**

- Convergence Criteria
  - When can we stop the iterations or time update?
    - A stronger criterion: $E\left[||\epsilon(n)||\right] \rightarrow 0 \text{ as } n \rightarrow \infty.$ (in the mean.)
      where $\epsilon(n) = \mathbf{w}(n) - \mathbf{w}_o.$
    - Another way of describing the convergence of the LMS algorithm:
      $J[n] = E\left[|e(n)|^2\right] \rightarrow \text{constant as } n \rightarrow \infty.$ (in the mean-square)

      If amount of change between cost values at time n and n-1(previous) is less than the predefined threshold value, then we may say that the filter system has converged to the stable state.

- ***Summary of The LMS Algorithm***

1. Filter output : $y[n] = \mathbf{w}^H[n]\mathbf{u}[n].$
2. Estimation error: $e[n] = d[n] - y[n].$
3. Tap-weight adaptation: $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}[n] + \mu\mathbf{u}[n]e^*[n].$

HW#4

- P. 406: Computer Experiment on Adaptive Prediction
  - SGD vs LMS 기법의 수렴성 및 특성 비교
  - 결과는 Fig. 9.14에 도시되어 있음

- 기한: to next week.

Thank you for your attention.!!!
QnA

http://ivpl.sookmyung.ac.kr