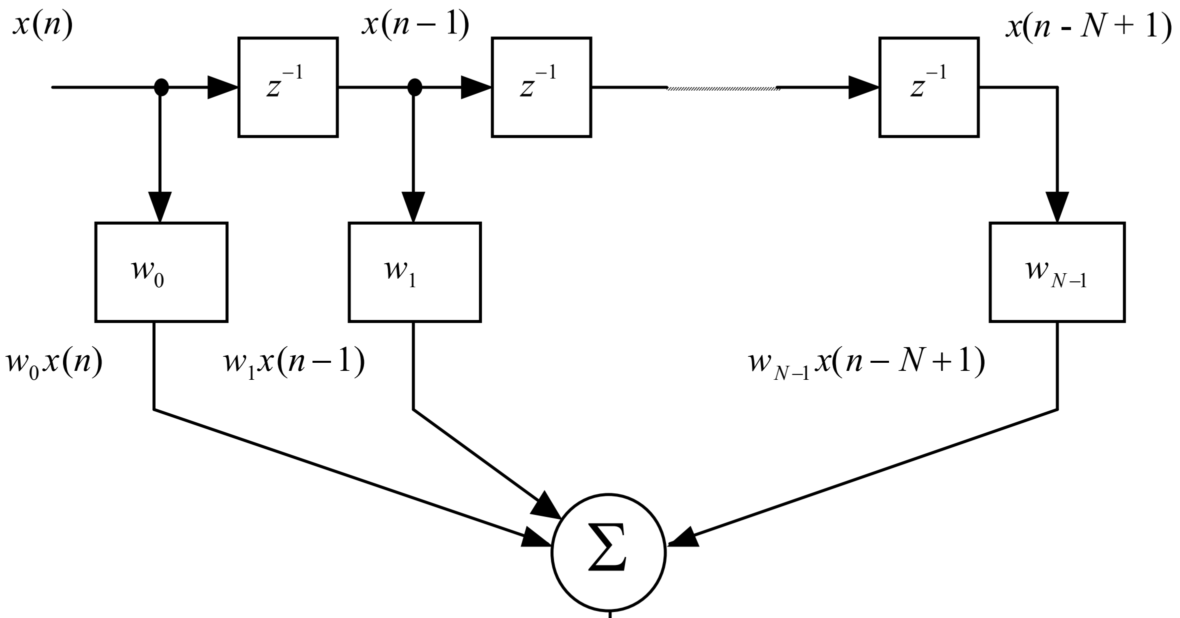
Adaptive Filters

Adaptive FIR Filter and the LMS Algorithm

Finite Impulse Response Filter

$$\underline{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^{\mathrm{T}}$$

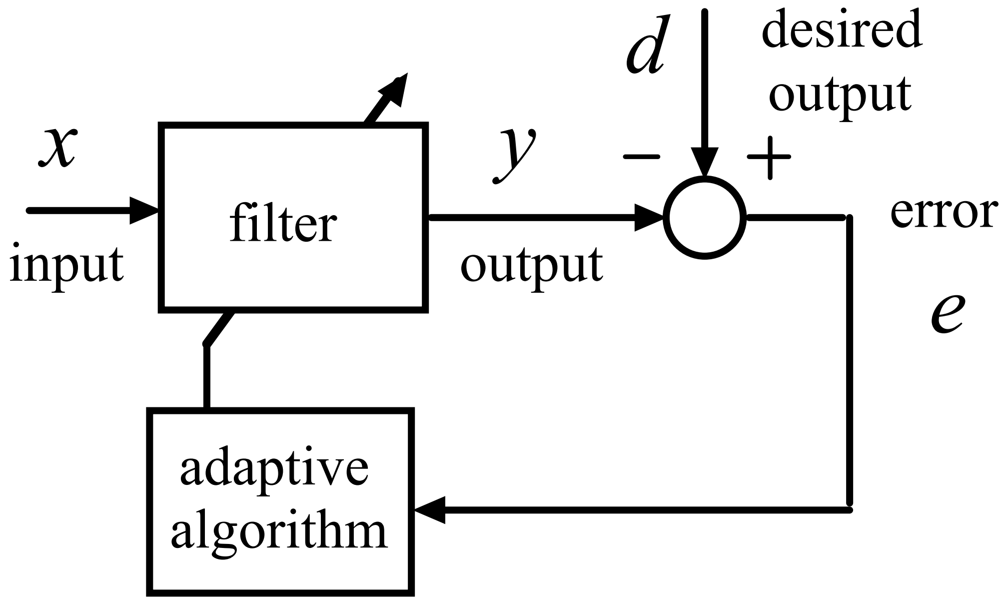


$$\underline{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^{\mathrm{T}}$$

$$y(n) = \sum_{k=0}^{N-1} w_k x(n-k)$$

$$y(n) = \underline{w}^{\mathrm{T}}(n)\underline{x}(n) = \underline{x}^{\mathrm{T}}(n)\underline{w}(n)$$

Adaptive FIR Filter

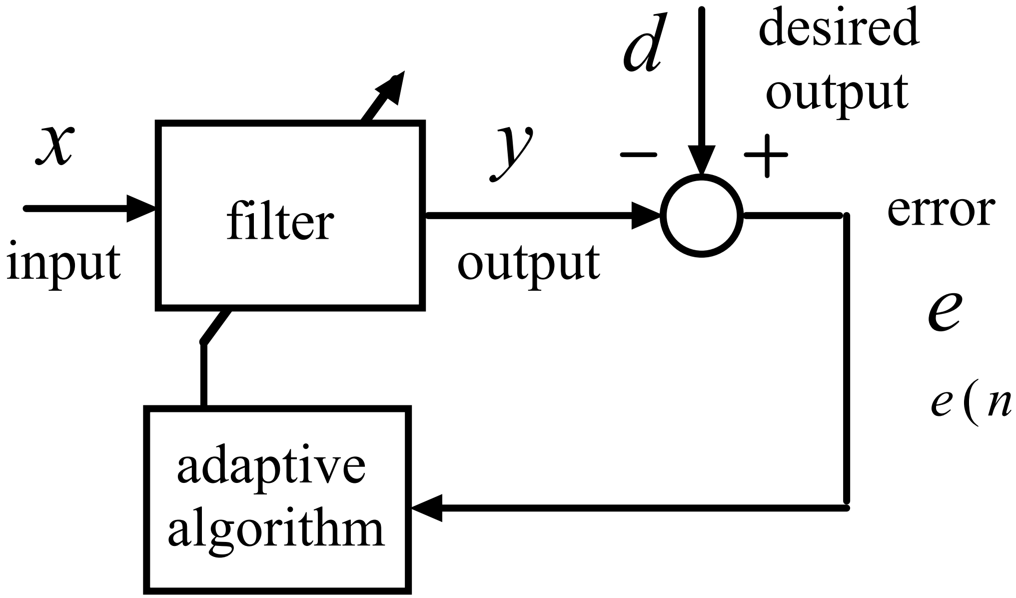


$$\underline{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^{T}$$

$$\underline{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^{T}$$

$$y(n) = \underline{w}^{T}(n)\underline{x}(n) = \underline{x}^{T}(n)\underline{w}(n)$$

Adaptive FIR Filter



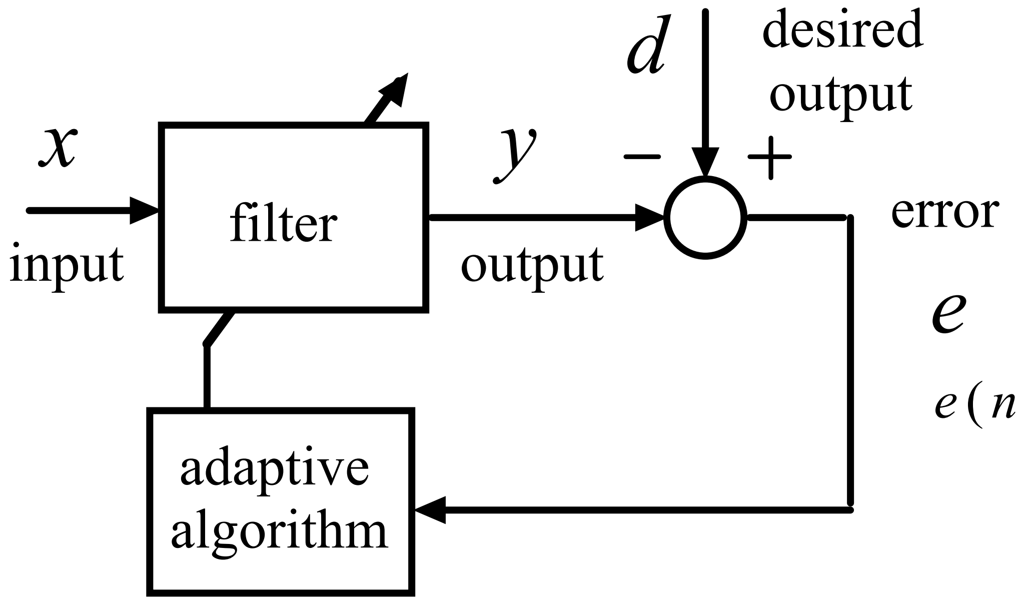
$$e(n) = d(n) - y(n)$$
$$= d(n) - \underline{x}^{T}(n)\underline{w}(n)$$

$$\underline{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^{T}$$

$$\underline{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^{T}$$

$$y(n) = \underline{w}^{T}(n)\underline{x}(n) = \underline{x}^{T}(n)\underline{w}(n)$$

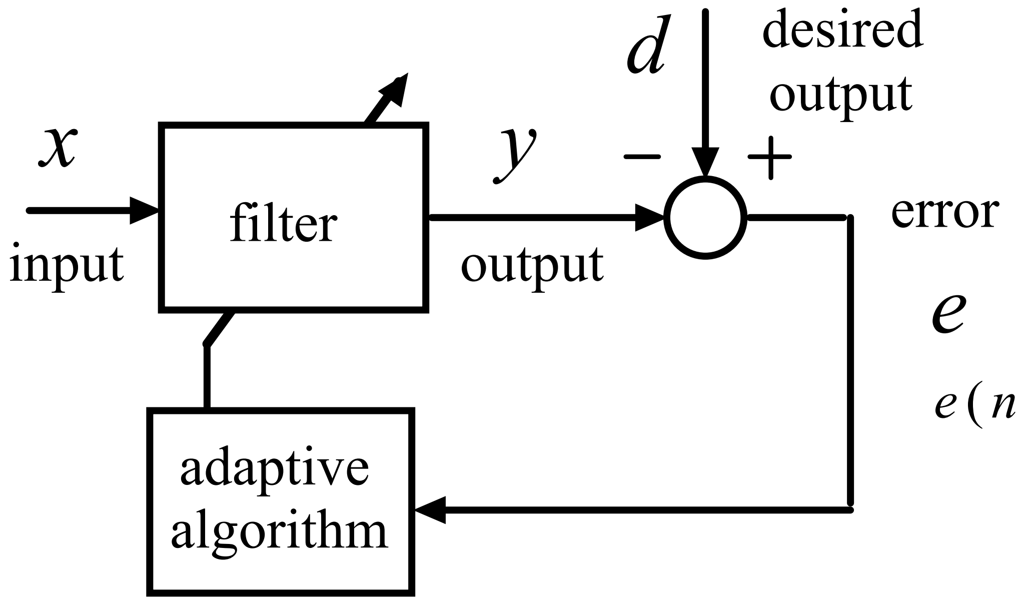
Defining a Cost Function



$$e(n) = d(n) - y(n)$$
$$= d(n) - \underline{x}^{T}(n)\underline{w}(n)$$

$$e^{2}(n) = d^{2}(n) - 2d(n)\underline{x}^{T}(n)\underline{w}(n) + \underline{w}^{T}(n)\underline{x}(n)\underline{x}^{T}(n)\underline{w}(n)$$

Defining a Cost Function



$$e(n) = d(n) - y(n)$$
$$= d(n) - \underline{x}^{T}(n)\underline{w}(n)$$

$$e^2(n) = d^2(n) - 2d(n)\underline{x}^{T}(n)\underline{w}(n) + \underline{w}^{T}(n)\underline{x}(n)\underline{x}^{T}(n)\underline{w}(n)$$

$$\xi(n) = E[e^2(n)]$$
$$= E[d^2(n) - 2d(n)\underline{x}^{T}(n)\underline{w}(n) + \underline{w}^{T}(n)\underline{x}(n)\underline{x}^{T}(n)\underline{w}(n)]$$

Defining a Cost Function

$$\xi(n) = E[e^2(n)]$$
$$= E[d^2(n) - 2d(n)\underline{x}^{\mathrm{T}}(n)\underline{w}(n) + \underline{w}^{\mathrm{T}}(n)\underline{x}(n)\underline{x}^{\mathrm{T}}(n)\underline{w}(n)]$$

$$\xi(n) = E[d^2(n)] + \underline{w}^{\mathrm{T}}(n)E[\underline{x}(n)\underline{x}^{\mathrm{T}}(n)]\underline{w}(n) - 2E[d(n)\underline{x}^{\mathrm{T}}(n)]\underline{w}(n)$$
$$= E[d^2(n)] + \underline{w}^{\mathrm{T}}(n)R\underline{w}(n) - 2\underline{p}^{\mathrm{T}}\underline{w}(n)$$

where $$\underline{p} = E[d(n)\underline{x}(n)]$$
$$R = E[\underline{x}(n)\underline{x}^{\mathrm{T}}(n)]$$

From now on, consider mean squared error to be a (quadratic) function of $\underline{w}$.

$$\xi(\underline{w}) = E[d^2(n)] + \underline{w}^{\mathrm{T}}(n)R\underline{w}(n) - 2\underline{p}^{\mathrm{T}}\underline{w}(n)$$

Minimum of the Cost Function

Differentiate mean squared error with respect to $\underline{w}$.

$$\frac{\partial \xi(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}}\left(E[d^2(n)] + \underline{w}^{\mathrm{T}} R \underline{w} - 2\underline{p}^{\mathrm{T}}\underline{w}\right)$$

$$= 2R\underline{w} - 2\underline{p}$$

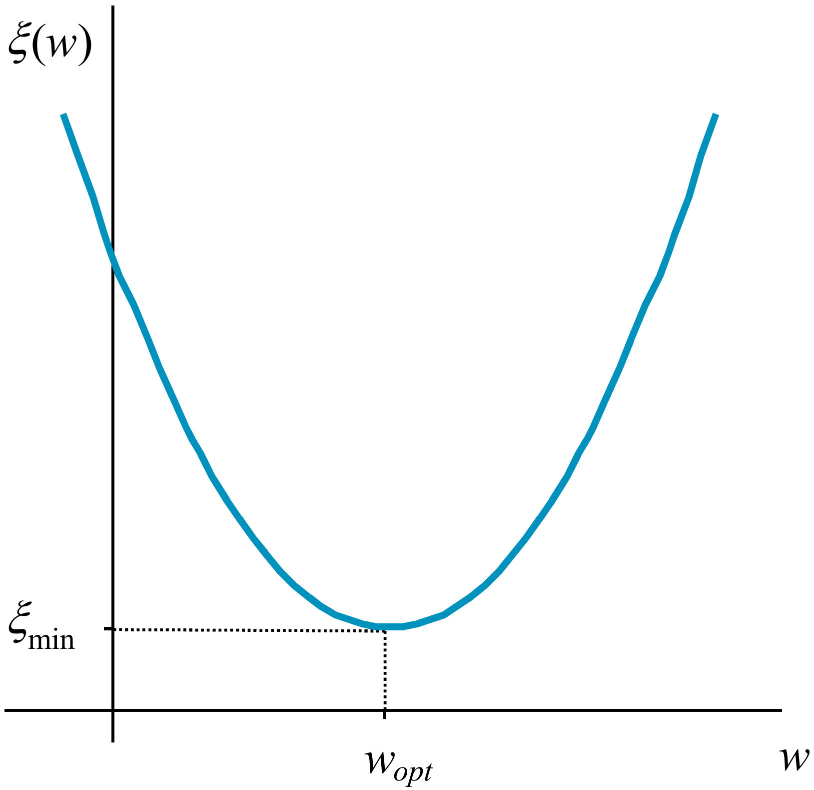
Derivative will equal zero at minimum corresponding to optimum value of $\underline{w}$.

$$2R\underline{w}_{opt} - 2\underline{p} = 0$$

Hence, the optimum value of $\underline{w}$ is a function of constant statistical properties of $\underline{x}$ and d.

$$\underline{w}_{opt} = R^{-1}\underline{p}$$

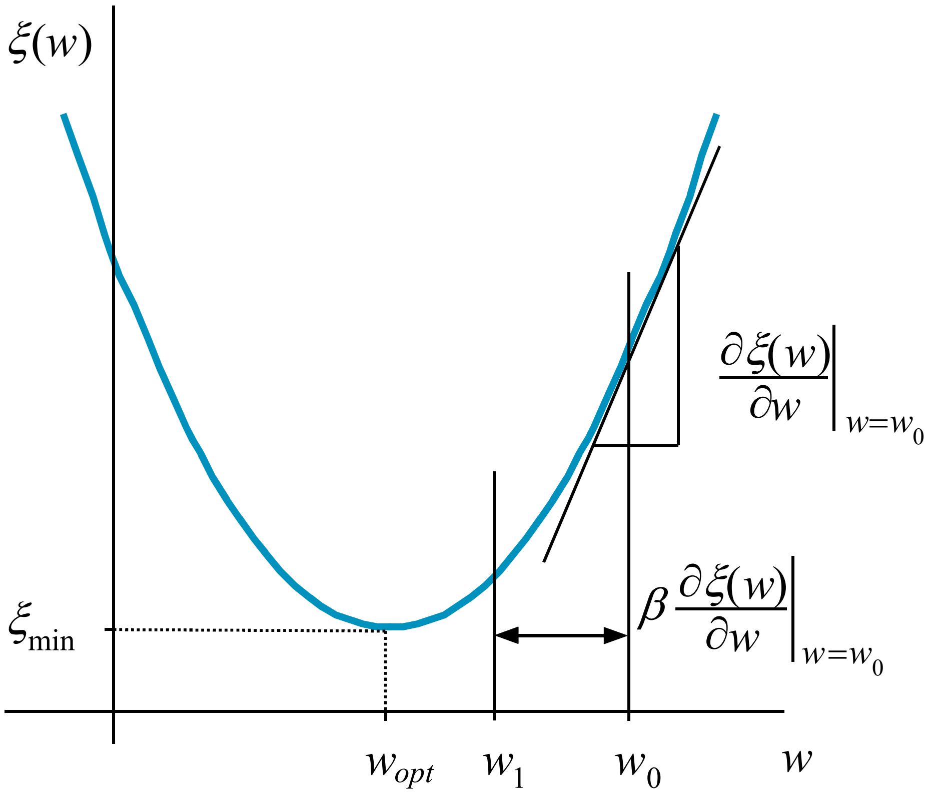
Visualizing the Cost Function



$$\xi(\underline{w}) = E[d^2(n)] + \underline{w}^T(n)R\,\underline{w}(n) - 2\,\underline{p}^T\,\underline{w}(n)$$

$$\underline{w}_{opt} = R^{-1}\,\underline{p}$$

Steepest Descent



$$\frac{\partial \xi(\underline{w})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}}\left(E[d^2(n)] + \underline{w}^T R \underline{w} - 2\underline{p}^T \underline{w}\right)$$

$$= 2R\underline{w} - 2\underline{p}$$

The LMS Algorithm

- The steepest descent method requires an estimate of the gradient of the cost function at each step.
- There are various ways of estimating that gradient.
- A general method might be to alter the value of $\underline{w}$ slightly, and over a suitable period of time in each case, assess the value of the cost function.

The LMS Algorithm

- However, we will look at a method that requires only *instantaneous* measurements in order to estimate the gradient of the cost function.
- The Least Mean Squares (LMS) algorithm uses instantaneous error squared e_k^2 as an estimate of mean squared error $E[e_k^2]$.

The LMS Algorithm

This yields the following gradient estimate

$$\underline{\hat{\nabla}}(n) = \begin{bmatrix} \dfrac{\partial \hat{\xi}(n)}{\partial w_0(n)} \\ \dfrac{\partial \hat{\xi}(n)}{\partial w_1(n)} \\ \vdots \\ \dfrac{\partial \hat{\xi}(n)}{\partial w_{N-1}(n)} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial e^2(n)}{\partial w_0(n)} \\ \dfrac{\partial e^2(n)}{\partial w_1(n)} \\ \vdots \\ \dfrac{\partial e^2(n)}{\partial w_{N-1}(n)} \end{bmatrix}$$

Using vector notation

$$\underline{\hat{\nabla}}(n) = \frac{\partial \hat{\xi}(n)}{\partial \underline{w}(n)} = \frac{\partial e^2(n)}{\partial \underline{w}(n)}$$

The LMS Algorithm

Differentiating the expression for instantaneous squared error with respect to $\underline{w}$

$$e_k^2 = d_k^2 + \underline{w}^{\mathrm{T}} \underline{x}_k \underline{x}_k^{\mathrm{T}} \underline{w} - 2 d_k \underline{x}_k^{\mathrm{T}} \underline{w}$$

$$\frac{\partial e_k^2}{\partial \underline{w}} = 2 \underline{x}_k \underline{x}_k^{\mathrm{T}} \underline{w} - 2 d_k \underline{x}_k$$

$$= 2 \left(\underline{x}_k^{\mathrm{T}} \underline{w} - d_k \right) \underline{x}_k$$

$$= -2 e_k \underline{x}_k$$

The LMS Algorithm

The steepest descent algorithm using this gradient estimate is:

$$\underline{w}_{k+1} = \underline{w}_k - \beta \hat{\underline{\nabla}}_k$$
$$= \underline{w}_k + 2\beta e_k \underline{x}_k$$

The LMS Algorithm

- Gradient estimate is imperfect.
- Adaptive process will be noisy.
- Conservative choice of *ß* value advisable
- Algorithm is simple.
- Not computationally intensive
- Ideal for real-time implementation

The LMS Algorithm

- Variants of the basic LMS algorithm

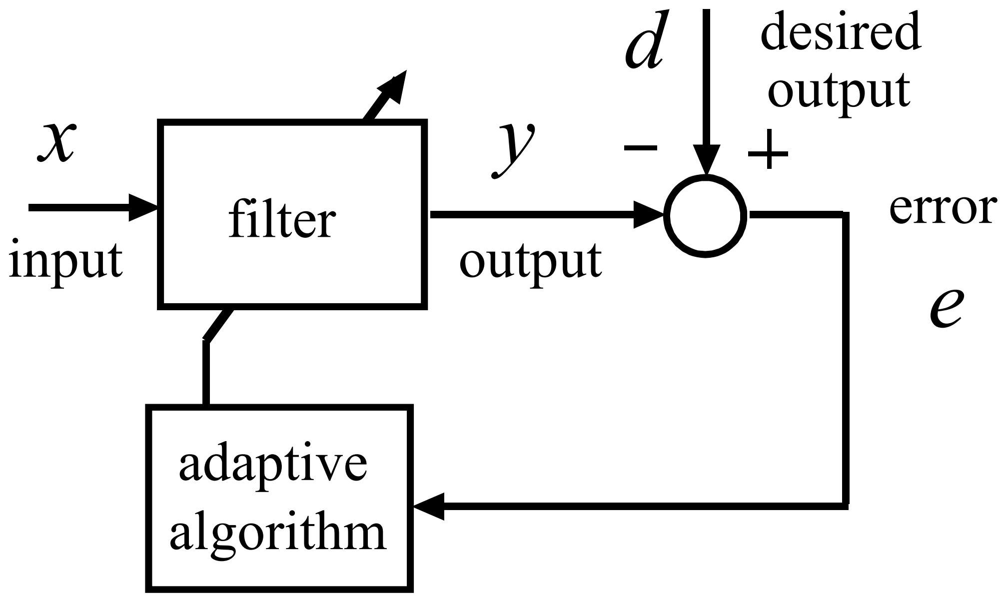
$$\underline{w}_{k+1} = \underline{w}_k + 2\beta \operatorname{sgn}(e_k)\underline{x}_k$$

$$\underline{w}_{k+1} = \underline{w}_k + 2\beta e_k \operatorname{sgn}(\underline{x}_k)$$

$$\underline{w}_{k+1} = \underline{w}_k + 2\beta \operatorname{sgn}(e_k)\operatorname{sgn}(\underline{x}_k)$$

Adaptive Filters - Key Points

- We've looked at adaptive filters that may be represented in the form



Adaptive Filters - Key Points

- The filter adjusts its characteristics to minimize the average power in e.
- Depending on how desired output d is derived, this behavior can be put to a number of different uses.
- For given statistical properties of x and d, average power in e is a function of $\underline{w}$.

Adaptive Filters - Key Points

- Adaptation is the search for filter parameter settings (weights, coefficients) that minimize the variance of e.
- The filter adjusts its characteristics to minimize the average power in e.
- Depending on how desired output d is derived, this behavior can be put to a number of different uses.
- For given statistical properties of x and d, average power in e is a function of $\underline{w}$.

Adaptive Filters - Key Points

- Adaptation is the search for filter parameter settings $\underline{w}$ that minimize average power in e.
- If the filter is a linear FIR, average power in e is a quadratic function of $\underline{w}$.
- Steepest descent is therefore feasible.
- But requires knowledge of the gradient of the cost function (average power)

Adaptive Filters - Key Points

- The LMS algorithm provides an *instantaneous estimate* of gradient for use in the steepest descent algorithm.
- Enabling us to search for $\underline{w}$ that minimizes $C(\underline{w})$ *on-line*, with minimal computational burden
- *LMS algorithm* and *adaptive FIR filter* are the basis of many other *learning* systems.