



Course material

Course:

Digital Signal Processing II

Video:

2.3.2.c The LMS Filter.c

Concepts (extracted from automatically generated subtitles):

Name of stochastic gradient descent. Filter coefficients. Mean square error. Adaptive signal processing. Local gradient of the function. So called learning factor. Optimal least squares. Initial estimate. Wide sense. Approximation error. Location of the minimum. Function minimization problems. Desired signal. Stationary signals. Basic problem.



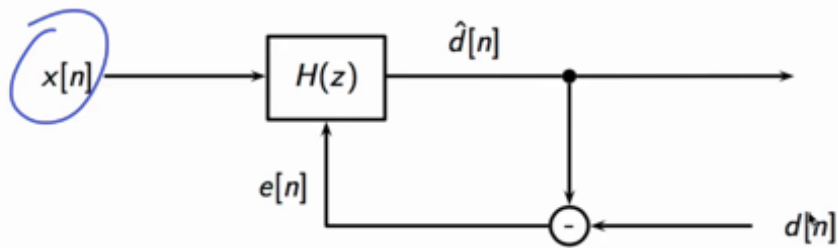
[to video sequence search](#)
(within Digital Signal Processing II.)



[to video](#)

Center for Digital Education. More educational support material here:

<https://exoset.epfl.ch/booc/>



- ▶ $d[n]$: desired signal
- ▶ $\hat{d}[n]$: adaptive approximation
- ▶ $e[n]$: error signal

73

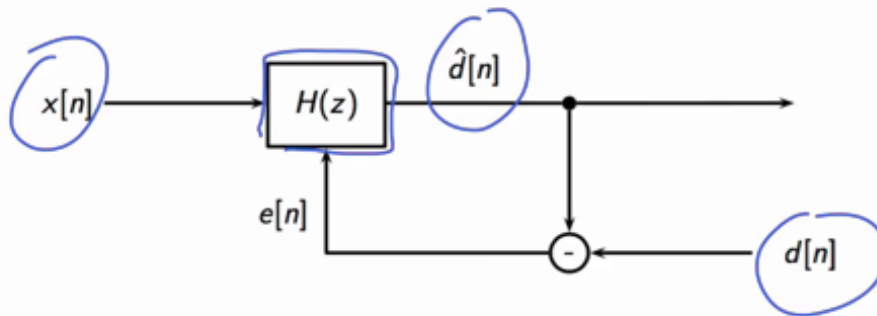
Okay, so let's go back to the basic problem in adaptive signal processing. We have an input, we have a desired signal and we want to adapt the filter

notes

summary

0m 0s





- ▶ $d[n]$: desired signal
- ▶ $\hat{d}[n]$: adaptive approximation
- ▶ $e[n]$: error signal

73

coefficients so that the result of filtering the input with a filter will give us a very good approximation to our desired signal. We choose to minimize the mean square error for the approximation error and we can do that

notes

summary

0m 8s



Two cases:

- ▶ for WSS signals, one-shot solution: Optimal Least Squares
- ▶ for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

97

in two ways. For wide sense stationary signals we have the one-shot solution which is given by the optimal least squares which we saw two lectures ago and for almost wide sense stationary signals we can use iterative solutions. These are signals that are almost stationary or locally stationary but that exhibit variation over time so we will need to constantly adapt the filter coefficients rather than trying to derive them in one shot. This requires an

notes

summary

0m 23s



Two cases:

- ▶ for WSS signals, one-shot solution: Optimal Least Squares
- ▶ for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

97

iterative solution which goes under name of stochastic gradient descent or LMS and before we analyze LMS in detail let's try to see if we can derive an iterative solution for the optimal least squares. Most function minimization

notes

summary

0m 46s



Two cases:

- ▶ for WSS signals, one-shot solution: Optimal Least Squares
- ▶ for “almost” WSS signals, iterative solutions: stochastic gradient descent or LMS

problems can be approached iteratively using a technique called steepest descent

notes

summary

1m 1s



Steepest descent:

start with a guess $\mathbf{x}_0$ and then, iteratively,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n)$$

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_0} \quad \frac{\partial f(\mathbf{x})}{\partial x_1} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_{N-1}} \right]^T$$

α_n : learning factor

98

and the idea is to start with an initial estimate for the location of the minimum and then progressively refine the estimate by computing the local gradient of the function that we want to minimize and moving in the opposite direction. The gradient tells you how you go up so you go in the opposite direction. So at least in three dimension the intuition is that gradient descent is like rolling a ball down a slope and letting the ball roll to the bottom of whatever surface the function represents. The good thing with quadratic error functions is that there is a global minimum and so you will be guaranteed to reach the minimum even with an iterative procedure. How does this work? We start with a guess and then at each step we update our guess by computing the gradient and moving in the

notes

summary

1m 4s



Steepest descent:

start with a guess $\mathbf{x}_0$ and then, iteratively,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \nabla f(\mathbf{x}_n)$$

↪ ↪

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_0} \quad \frac{\partial f(\mathbf{x})}{\partial x_1} \quad \cdots \quad \frac{\partial f(\mathbf{x})}{\partial x_{N-1}} \right]^T$$

α_n : learning factor

98

opposite direction. That's why there's a minus here and we also introduce a so called learning factor here whose function is to slow down the motion of this metaphorical ball that's rolling down the slope. We don't want to move too fast we prefer to converge to the minimum a bit slowly and so we modulate the descent via this scalar learning factor. The gradient is simply a vector that contains all the partial derivatives of the function that you're trying to minimize. One of the problems with steepest descent is that it can get stuck in some local minima of the function but if we apply steepest descent to our mean square error minimization we're in luck because the

notes

summary

2m 1s



- ▶ for a quadratic error surface, minimum is always global
- ▶ gradient is easy to compute:

$$\begin{aligned}\nabla J(\mathbf{h}) &= \left[\frac{\partial E[e^2[n]]}{\partial h[0]} \quad \frac{\partial E[e^2[n]]}{\partial h[1]} \quad \dots \quad \frac{\partial E[e^2[n]]}{\partial h[N-1]} \right]^T \\ &= 2(\mathbf{R}\mathbf{h} - \mathbf{g})\end{aligned}$$

error surface is quadratic and so the minimum will exist and will be a global minimum. The gradient is also very easy to compute with a quadratic cost function. Again it's a partial derivative of the expected value of the square error with respect to all of the filter coefficients that are trying to compute

notes

summary

2m 42s



- ▶ for a quadratic error surface, minimum is always global
- ▶ gradient is easy to compute:

$$\nabla J(\mathbf{h}) = \left[\frac{\partial E[e^2[n]]}{\partial h[0]} \quad \frac{\partial E[e^2[n]]}{\partial h[1]} \quad \dots \quad \frac{\partial E[e^2[n]]}{\partial h[N-1]} \right]^T$$

$$= 2(\mathbf{R}\mathbf{h} - \mathbf{g})$$

and if we do the math we find out that the gradient is simply two times the

notes

summary

3m 1s



- ▶ for a quadratic error surface, minimum is always global
- ▶ gradient is easy to compute:

$$\nabla J(\mathbf{h}) = \left[\frac{\partial E[e^2[n]]}{\partial h[0]} \quad \frac{\partial E[e^2[n]]}{\partial h[1]} \quad \dots \quad \frac{\partial E[e^2[n]]}{\partial h[N-1]} \right]^T$$

$$= 2(\mathbf{R}\mathbf{h} - \mathbf{g})$$

$\mathbf{R} \mathbf{g}$

$$h_0$$

$$h_1 \leftarrow h_0 - \alpha 2(\mathbf{R}h_0 - \mathbf{g})$$

$$h_2 \leftarrow h_1 - \alpha 2(\mathbf{R}h_1 - \mathbf{g})$$

99

autocorrelation matrix times the vector of filter coefficients minus $\mathbf{g}$ which is a vector of cross correlation coefficients that we found in the least square solutions before. So in the iterations $\mathbf{r}$ and $\mathbf{g}$ are known and fixed we start with an estimate for the filter coefficients and the next step the next estimate will be the previous estimate minus the learning factor times two times $\mathbf{r}^T \mathbf{h}_0$ minus $\mathbf{g}$. Okay and then of course $\mathbf{h}_2$ will be $\mathbf{h}_1$ minus alpha two times $\mathbf{r}^T \mathbf{h}_1$ minus $\mathbf{g}$ and so on so forth. We can look at the process graphically for

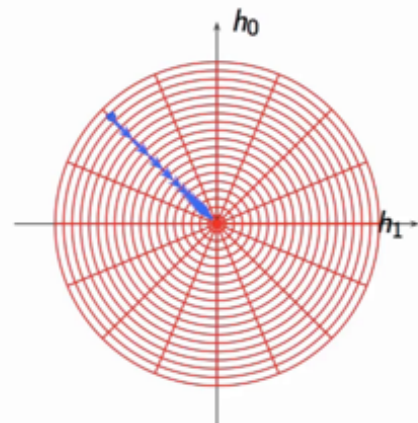
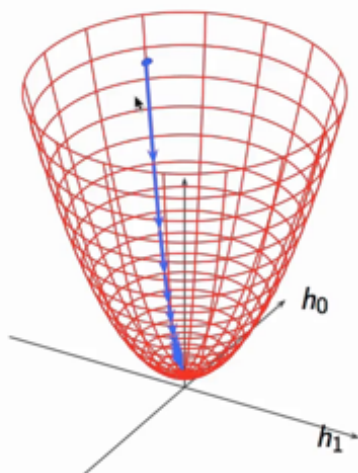
notes

summary

3m 4s



Steepest descent for white input



100

a wide input signal and you see we can start from a guess somewhere here in the view from the top and at each step we're moving in the direction of

notes

summary

3m 52s



- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic* gradient descent:

$$E[e^2[n]] \leftarrow e^2[n]$$

maximum slope towards the bottom of the paraboloid. For a wide input any point is a good guess because the gradient will always point towards the origin so I can start here and I will go just as fast towards the bottom of the paraboloid. For a non-wide input because of the elliptical cross-section we can have good guesses if we're aligned or almost aligned with the axis but if we start from a side point the initial gradient will send us a bit sideways and so the convergence will be a bit zigzagging like so. For non-wide input we should also be careful with the learning factor if it is too big the fact that the gradient is not aligned with the axis might send us off to a point of higher level with respect to the minimum than we started from at which point the algorithm will diverge. For wide sense stationary signals the one-shot solution, the least square solution and the iterative solutions are the same. For time

notes

summary

4m 1s



- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic* gradient descent:

$$E[e^2[n]] \leftarrow e^2[n]$$

variance signals on the other hand we need to follow the changes in statistical properties of the input and so the iterative solution is the way to go.

notes

summary

5m 1s



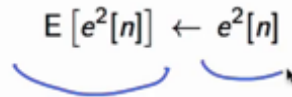
notes

5m 13s



- $$E[e^2[n]] \leftarrow e^2[n]$$

- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic gradient descent*:

$$E[e^2[n]] \leftarrow e^2[n]$$


genius that is just as radical as it is surprisingly effective. We will replace the expected value of the squared error with the squared error itself. So we will

notes

summary

5m 30s



- ▶ for WSS signals, one-shot and iterative are the same
- ▶ for time-varying signals, we need to follow the changes: iterative solution
- ▶ computation of time-varying correlations is costly
- ▶ *stochastic gradient descent*:

$$\underbrace{\hspace{10em}} \quad \underbrace{E[e^2[n]]} \leftarrow \underbrace{e^2[n]}$$

just get rid of the averages and use the unaveraged error signal. This becomes then a stochastic gradient descent because the gradient becomes a stochastic variable itself. We don't have expectations anymore in the formula. So

notes

summary

5m 42s



$$\nabla J = \left[\frac{\partial E[e^2[n]]}{\partial h[0]} \quad \frac{\partial E[e^2[n]]}{\partial h[1]} \quad \cdots \quad \frac{\partial E[e^2[n]]}{\partial h[N-1]} \right]^T$$

we go from this the gradient of the mean square error to this which is the

notes

summary

5m 58s



$$\underline{e[n]} = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2e[n] \mathbf{x}_n$$

$$\mathbf{x}_n = [x[n] \quad x[n-1] \quad x[n-2] \quad \dots \quad x[n-N+1]]^T.$$

gradient of the squared error. Let's compute then this new stochastic gradient. Remember the error is the difference between the desired signal and our

notes

summary

6m 1s



$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2e[n] \mathbf{x}_n$$

$$\mathbf{x}_n = [x[n] \quad x[n-1] \quad x[n-2] \quad \dots \quad x[n-N+1]]^T.$$

approximation which is obtained by a FIR filtering so a linear combination of past samples of the input. If we take the partial derivative of the error with

notes

summary

6m 13s



$$e[n] = d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]$$

$$\frac{\partial e^2[n]}{\partial h[i]} = -2e[n]x[n-i].$$

$$\nabla J_n = -2\underbrace{e[n]}_{\text{scalar}} \underbrace{\mathbf{x}_n}_{\text{vector}}$$

$$\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & x[n-2] & \dots & x[n-N+1] \end{bmatrix}^T.$$

↑

respect to the filter coefficient number i we get minus 2 times e of n times x of n minus i . As a vector the gradient is simply the multiplication between the current error value and a vector that contains the current sample and n minus 1 past samples. So let's now apply steepest descent using a stochastic

notes

summary

6m 22s



$$\mathbf{h}_0 = [h_0[0] \quad h_0[1] \quad \dots \quad h_0[N-1]]^T \text{ initial guess}$$

$$e[n] = d[n] - \mathbf{h}_n^T \mathbf{x}_n$$

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \alpha_n e[n] \mathbf{x}_n$$

gradient. We start with a guess for our filter coefficients and at each step we compute two things. First we compute the error as the difference between the desired signal and the approximation which now we can express simply as an inner product between the filter coefficients and this vector of current and past input samples and then we update the filter coefficients via a

notes

summary

6m 47s



$$\mathbf{h}_0 = [h_0[0] \quad h_0[1] \quad \dots \quad h_0[N-1]]^T \text{ initial guess}$$

$$\left\{ \begin{array}{l} e[n] = d[n] - \mathbf{h}_n^T \mathbf{x}_n \\ \mathbf{h}_{n+1} = \mathbf{h}_n + \alpha_n e[n] \mathbf{x}_n \end{array} \right.$$

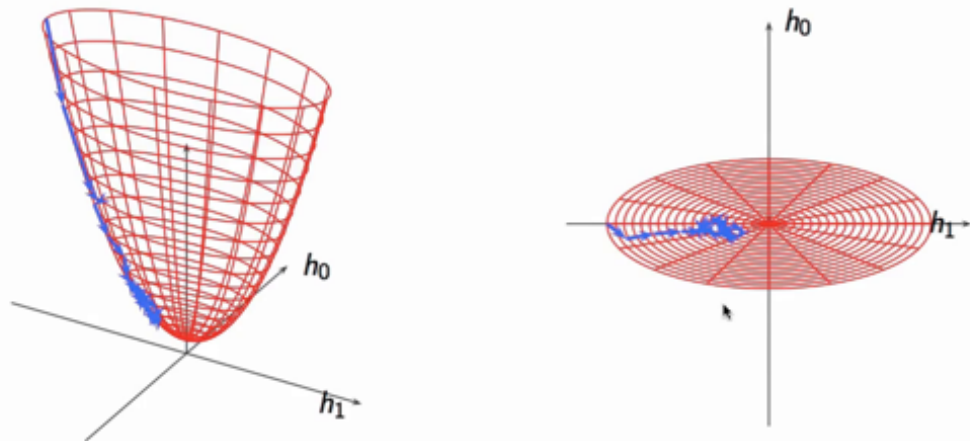
learning factor and via the stochastic gradient which is the multiplication between the error value that we just compute and the vector of input samples. So it's a two-step update of the filter coefficients which will hopefully converge to the minimum of the error surface. Does this work? Well for instance for white input we can see that it does although the convergence

notes

summary

7m 13s





111

is a bit erratic, it's not as fluid as in the gradient descent that uses the mean square error and instead of converging to the actual minimum of the surface the adaptation seems to hover in the vicinity of the minimum. For correlated inputs the system works as well although the erratic behavior is perhaps a little bit pronounced. What is really important in all cases is choosing the learning factor very wisely and trade speed of convergence for stability. So the learning factor usually is chosen rather small.

notes

summary

7m 38s

