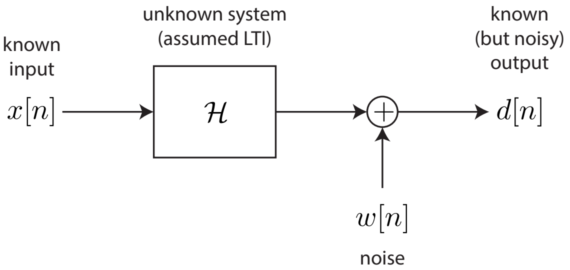
MMSE System Identification, Gradient Descent, and the Least Mean Squares Algorithm

D.R. Brown III

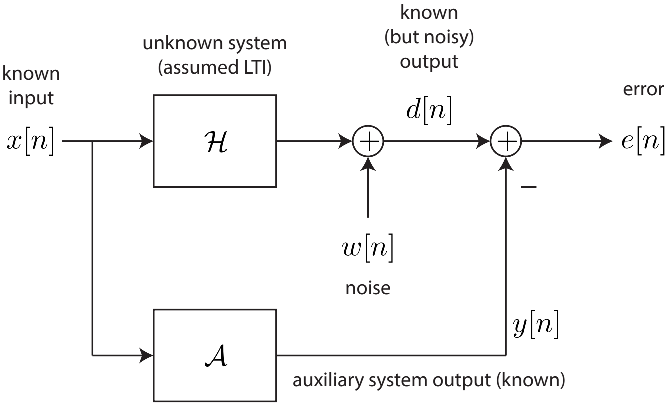
WPI

Problem Statement and Assumptions



- We want to estimate the impulse response of the unknown system.
- Just sending $x[n] = \delta[n]$ is not a good idea because we don't get any averaging.
- Our approach: build an "auxiliary system" and minimize the mean squared error.

Auxiliary System



The mean squared error (MSE) is defined as

$$\mathsf{MSE} = \mathsf{E}\left\{e^2[n]\right\} = \mathsf{E}\left\{(d[n] - y[n])^2\right\}.$$

We want to design the auxiliary system to minimize the MSE.

Warmup Problem: Unknown System is a Gain

Suppose $\mathcal{H}$ is simply a gain g and we wish to estimate this gain.

The auxiliary system is also a gain denoted as $\hat{g}$.

The MSE is then

$$\begin{aligned}\mathsf{MSE} &= \mathsf{E}\left\{(d[n]-y[n])^2\right\} \\ &= \mathsf{E}\left\{d^2[n]-2d[n]\hat{g}x[n]+\hat{g}^2x^2[n]\right\} \\ &= \mathsf{E}\left\{d^2[n]\right\}-2\hat{g}\mathsf{E}\left\{d[n]x[n]\right\}+\hat{g}^2\mathsf{E}\left\{x^2[n]\right\}\end{aligned}$$

To minimize the MSE, we take a derivative of MSE with respect to $\hat{g}$, set it equal to zero, and solve for $\hat{g}$. This results in

$$\hat{g} = \frac{\mathsf{E}\left\{d[n]x[n]\right\}}{\mathsf{E}\left\{x^2[n]\right\}} \approx \frac{\frac{1}{N}\sum_{n=0}^{N-1} d[n]x[n]}{\frac{1}{N}\sum_{n=0}^{N-1} x^2[n]}$$

Remarks

The minimum MSE (MMSE) solution is:

$$\hat{g} = \frac{\mathsf{E}\{d[n]x[n]\}}{\mathsf{E}\{x^2[n]\}}$$

Recall the output of the unknown system is $d[n] = gx[n] + w[n]$. We can substitute for $d[n]$ and use the linearity of the expectation to write

$$\hat{g} = \frac{\mathsf{E}\{(gx[n] + w[n])x[n]\}}{\mathsf{E}\{x^2[n]\}} = g + \frac{\mathsf{E}\{w[n]x[n]\}}{\mathsf{E}\{x^2[n]\}}.$$

If $x[n]$ is statistically independent of $w[n]$ (which is usually is) and one or both are zero mean then

$$\mathsf{E}\{w[n]x[n]\} = \mathsf{E}\{w[n]\}\,\mathsf{E}\{x[n]\} = 0.$$

Hence, if you have enough samples to accurately compute the expectations, this estimator converges to the correct value: $\hat{g} \to g$.

Problem: Unknown System is an FIR Filter

Suppose $\mathcal{H}$ is now a FIR filter with impulse response $\{h[0], \dots, h[L-1]\}$ and we wish to estimate this impulse response.

The auxiliary system is also a FIR filter with impulse response denoted as $\{\hat{h}[0], \dots, \hat{h}[L-1]\}$.

Note that the output of the auxiliary system can be written as

$$y[n] = \sum_{k=0}^{L-1} \hat{h}[k]x[n-k] = (\hat{\boldsymbol{h}})^\top \boldsymbol{x}[n]$$

where

$$\hat{\boldsymbol{h}} = \begin{bmatrix} \hat{h}[0] \\ \vdots \\ \hat{h}[L-1] \end{bmatrix} \quad \text{and} \quad \boldsymbol{x}[n] = \begin{bmatrix} x[n] \\ \vdots \\ x[n-(L-1)] \end{bmatrix}$$

This is just a representation of convolution as an inner/dot product.

Mean Squared Error

Recall that

$$(\boldsymbol{a}^\top \boldsymbol{b})^2 = \boldsymbol{a}^\top \boldsymbol{b}\boldsymbol{b}^\top \boldsymbol{a} = \boldsymbol{b}^\top \boldsymbol{a}\boldsymbol{a}^\top \boldsymbol{b}.$$

The MSE is then

$$\begin{aligned}
\mathsf{MSE} &= \mathsf{E}\left\{(d[n] - y[n])^2\right\} \\
&= \mathsf{E}\left\{\left(d[n] - (\hat{\boldsymbol{h}})^\top \boldsymbol{x}[n]\right)^2\right\} \\
&= \mathsf{E}\left\{d^2[n] - 2d[n](\hat{\boldsymbol{h}})^\top \boldsymbol{x}[n] + (\hat{\boldsymbol{h}})^\top \boldsymbol{x}[n]\boldsymbol{x}^\top[n]\hat{\boldsymbol{h}}\right\} \\
&= \mathsf{E}\left\{d^2[n]\right\} - 2(\hat{\boldsymbol{h}})^\top \mathsf{E}\left\{d[n]\boldsymbol{x}[n]\right\} + (\hat{\boldsymbol{h}})^\top \mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}\hat{\boldsymbol{h}}
\end{aligned}$$

To minimize the MSE, we take a gradient of the MSE with respect to $\hat{\boldsymbol{h}}$, set it equal to zero, and solve for $\hat{\boldsymbol{h}}$. This results in L equations...

Gradient Review

For $f : \mathbb{R}^L \mapsto \mathbb{R}$, recall the gradient is defined as

$$\nabla_{\boldsymbol{a}} f(\boldsymbol{a}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{a})}{\partial a_0} \\ \vdots \\ \frac{\partial f(\boldsymbol{a})}{\partial a_{L-1}} \end{bmatrix}$$

For example, suppose $\boldsymbol{a} = [a_0, a_1]^\top$ and

$$f(\boldsymbol{a}) = \boldsymbol{a}^\top \boldsymbol{a} = a_0^2 + a_1^2.$$

Then

$$\nabla_{\boldsymbol{a}} f(\boldsymbol{a}) = \begin{bmatrix} 2a_0 \\ 2a_1 \end{bmatrix} = 2\boldsymbol{a}$$

It is not difficult to show for general $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{C}$ of proper dimensions that

$$\nabla_{\boldsymbol{a}}(\boldsymbol{a}^\top \boldsymbol{b}) = \boldsymbol{b}$$
$$\nabla_{\boldsymbol{a}}(\boldsymbol{a}^\top \boldsymbol{C} \boldsymbol{a}) = 2\boldsymbol{C}\boldsymbol{a}.$$

Minimum Mean Squared Error

We have

$$\mathsf{MSE} = \mathsf{E}\left\{d^2[n]\right\} - 2(\hat{\boldsymbol{h}})^\top \mathsf{E}\left\{d[n]\boldsymbol{x}[n]\right\} + (\hat{\boldsymbol{h}})^\top \mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}\hat{\boldsymbol{h}}$$

The gradient can be computed as

$$\nabla_{\hat{\boldsymbol{h}}}\mathsf{MSE} = \boldsymbol{0} - 2\mathsf{E}\left\{d[n]\boldsymbol{x}[n]\right\} + 2\mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}\hat{\boldsymbol{h}}$$

This can be rearranged and solved for $\hat{\boldsymbol{h}}$ to write

$$\begin{aligned}\hat{\boldsymbol{h}} &= \left(\mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}\right)^{-1}\mathsf{E}\left\{d[n]\boldsymbol{x}[n]\right\}\\ &= \boldsymbol{R}^{-1}\boldsymbol{p}\end{aligned}$$

where $\boldsymbol{R} \in \mathbb{R}^{L\times L}$ is the autocorrelation matrix of the input and $\boldsymbol{p} \in \mathbb{R}^L$ is the cross correlation vector of the input with the output of unknown system.

Remarks

MMSE solution:

$$\hat{\boldsymbol{h}} = \boldsymbol{R}^{-1}\boldsymbol{p}$$

1. This is a generalization of our previous result for when the unknown system was a gain. In that case we had

$$R = \mathsf{E}\{x^2[n]\}$$
$$p = \mathsf{E}\{d[n]x[n]\}$$

   and $\hat{g} = p/R = R^{-1}p$.
2. We assume we have control of $x[n]$, so we can always make $\boldsymbol{R} = \mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}$ invertible.

Computing Minimum Mean Squared Error

We have the MMSE solution

$$\hat{\boldsymbol{h}} = \boldsymbol{R}^{-1}\boldsymbol{p}$$

with $\boldsymbol{R} = \mathsf{E}\left\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\right\}$ and $\boldsymbol{p} = \mathsf{E}\left\{d[n]\boldsymbol{x}[n]\right\}$. In practice, we can approximate the expectations by computing the averages

$$\boldsymbol{R} \approx \frac{1}{N}\sum_{n=0}^{N-1}\boldsymbol{x}[n]\boldsymbol{x}^\top[n]$$

$$\boldsymbol{p} \approx \frac{1}{N}\sum_{n=0}^{N-1}d[n]\boldsymbol{x}[n]$$

Then we have to compute the matrix inverse $\boldsymbol{R}^{-1}$ (with complexity $\mathcal{O}(L^3)$) and the matrix vector product $\boldsymbol{R}^{-1}\boldsymbol{p}$ (with complexity $\mathcal{O}(L^2)$). This is easy enough in Matlab, but more difficult on the DSK.

See the Matlab code `sysid.m` on the course website.

Computing Minimum Mean Squared Error: A Trick

If the input signal is "white" so that $x[n]$ is statistically independent of $x[m]$ for all $n \neq m$, then

$$\boldsymbol{R} = \rho \boldsymbol{I} = \begin{bmatrix} \rho & & \\ & \ddots & \\ & & \rho \end{bmatrix}$$

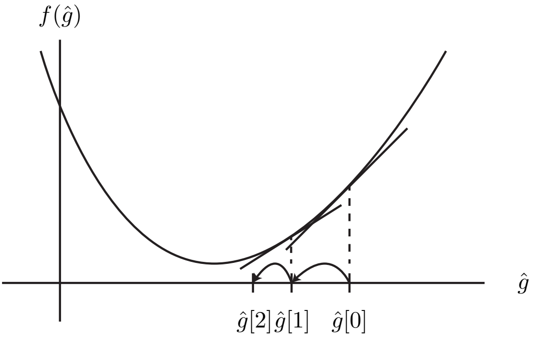
This is easy to invert and the resulting MMSE estimate of the unknown system's impulse response is simply

$$\hat{\boldsymbol{h}} = \boldsymbol{R}^{-1}\boldsymbol{p} = \frac{1}{\rho}\boldsymbol{p}.$$

Even with this trick, this approach is not desirable for a real-time system because of its batch nature. We still have to collect lots of samples to approximate the expectations.

We would like a way of automatically adapting $\hat{\boldsymbol{h}}$ as new samples arrive so that $\hat{\boldsymbol{h}} \to \boldsymbol{h}$ and the mean squared error is minimized.

Exact Derivative Descent



Idea: Starting from an initial guess $\hat{g}[0]$, take small steps proportional to the negative of the derivative of the objective function $f(\hat{g})$.

$$\hat{g}[n+1] = \hat{g}[n] - \mu \left[\frac{\partial}{\partial a} f(a) \right]_{a=\hat{g}[n]}$$

Exact Derivative Descent for System ID

For the case when our unknown system is a gain, we have

$$\begin{aligned}\frac{\partial}{\partial \hat{g}}\mathsf{MSE} &= -2\mathsf{E}\{d[n]x[n]\} + 2\hat{g}\mathsf{E}\{x^2[n]\} \\ &= -2p + \hat{g}2R\end{aligned}$$

So (absorbing the factor of 2 into the stepsize μ), the exact derivative descent algorithm would be implemented as

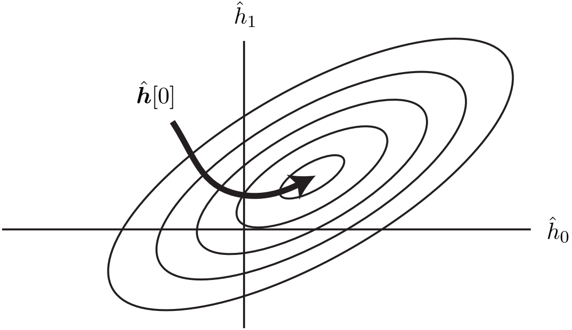
$$\hat{g}[n+1] = \hat{g}[n] - \mu(\hat{g}[n]R - p)$$

Remarks:

- As long as μ is small enough, this is guaranteed to converge since the MSE objective function is quadratic and has a unique minimum.
- Note that this iteration avoids the division required to compute the MMSE solution directly, i.e., $\hat{g} = p/R$.
- More "adaptive" than the direct (batch) estimator, but we still need to collect samples and estimate R and p.

Exact Gradient Descent

The same idea works with multidimensional objective functions $f : \mathbb{R}^L \mapsto \mathbb{R}$ except we use a gradient rather than a derivative.



$$\hat{\boldsymbol{h}}[n+1] = \hat{\boldsymbol{h}}[n] - \mu \left[\nabla_{\boldsymbol{a}} f(\boldsymbol{a})\right]_{\boldsymbol{a}=\hat{\boldsymbol{h}}[n]}$$

Exact Gradient Descent for System ID

For a FIR unknown system, we have

$$\begin{aligned}\frac{\partial}{\partial \hat{\boldsymbol{h}}}\text{MSE} &= -2\text{E}\{d[n]\boldsymbol{x}[n]\} + 2\text{E}\{\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\}\hat{\boldsymbol{h}} \\ &= -2\boldsymbol{p} + 2\boldsymbol{R}\hat{\boldsymbol{h}}\end{aligned}$$

Like before, the exact gradient descent algorithm would be implemented as

$$\hat{\boldsymbol{h}}[n+1] = \hat{\boldsymbol{h}}[n] - \mu(\boldsymbol{R}\hat{\boldsymbol{h}}[n] - \boldsymbol{p})$$

Remarks:

- As long as μ is small enough, this will also guaranteed to converge since the MSE objective function is (multidimensional) quadratic and has a unique minimum.
- Note that this iteration avoids the matrix inverse required to compute the MMSE solution directly, i.e., $\hat{\boldsymbol{g}} = \boldsymbol{R}^{-1}\boldsymbol{p}$.
- More "adaptive" than the direct (batch) estimator, but we still need to collect samples and estimate $\boldsymbol{R}$ and $\boldsymbol{p}$.

Approximate Gradient Descent for System ID (1/2)

The main problem with the exact gradient descent algorithm is that we have to collect lots of samples to get accurate estimates of $\boldsymbol{R}$ and $\boldsymbol{p}$.

$$\boldsymbol{R} \approx \frac{1}{N} \sum_{n=0}^{N-1} \boldsymbol{x}[n]\boldsymbol{x}^\top[n]$$

$$\boldsymbol{p} \approx \frac{1}{N} \sum_{n=0}^{N-1} d[n]\boldsymbol{x}[n]$$

These approximations become more accurate as N becomes larger.

What if we did something dumb? What if we just set $N = 1$?

$$\tilde{\boldsymbol{R}}[n] = \boldsymbol{x}[n]\boldsymbol{x}^\top[n]$$

$$\tilde{\boldsymbol{p}}[n] = d[n]\boldsymbol{x}[n]$$

These are terrible estimates of $\boldsymbol{R}$ and $\boldsymbol{p}$!

Approximate Gradient Descent for System ID (2/2)

Bad estimates of $\boldsymbol{R}$ and $\boldsymbol{p}$:

$$\tilde{\boldsymbol{R}}[n] = \boldsymbol{x}[n]\boldsymbol{x}^\top[n]$$
$$\tilde{\boldsymbol{p}}[n] = d[n]\boldsymbol{x}[n]$$

Let's just plug these into our gradient descent algorithm and see what happens (recall that $y[n] = (\hat{\boldsymbol{h}}[n])^\top \boldsymbol{x}[n] = \boldsymbol{x}^\top[n]\hat{\boldsymbol{h}}[n]$):

$$\begin{aligned}
\hat{\boldsymbol{h}}[n+1] &= \hat{\boldsymbol{h}}[n] - \mu(\tilde{\boldsymbol{R}}\hat{\boldsymbol{h}}[n] - \tilde{\boldsymbol{p}}) \\
&= \hat{\boldsymbol{h}}[n] - \mu(\boldsymbol{x}[n]\boldsymbol{x}^\top[n]\hat{\boldsymbol{h}}[n] - d[n]\boldsymbol{x}[n]) \\
&= \hat{\boldsymbol{h}}[n] - \mu(\boldsymbol{x}[n]y[n] - d[n]\boldsymbol{x}[n]) \\
&= \hat{\boldsymbol{h}}[n] - \mu(y[n] - d[n])\boldsymbol{x}[n] \\
&= \hat{\boldsymbol{h}}[n] + \mu e[n]\boldsymbol{x}[n]
\end{aligned}$$

This is called the "Least Mean Squares" (LMS) algorithm. LMS is the "workhorse of adaptive filtering".

LMS Basics

Recursion:

$$\hat{\boldsymbol{h}}[n+1] = \hat{\boldsymbol{h}}[n] + \mu e[n]\boldsymbol{x}[n]$$

Remarks:

- Completely sample-by-sample operation.
- Start with any guess $\hat{\boldsymbol{h}}[0]$ you want (avoid infinities and NaNs). Remarkably, this is guaranteed to converge to the MMSE solution if μ is sufficiently small.
- Convergence is not monotonic like exact gradient descent, but the convenience of not having to estimate $\boldsymbol{R}$ and $\boldsymbol{p}$ is generally more desirable.