ECE 6123
Advanced Signal Processing
Introduction

Peter Willett

Fall 2017

1 Filters

1.1 Standard Filters

These are the standard filter types from DSP class.

IIR (infinite-length impulse response) filter

$$y[n] = \sum_{k=0}^{M-1} b_k^* x[n-k] - \sum_{k=1}^{N-1} a_k^* y[n-k] \tag{1}$$

$$H(z) = \frac{\sum_{k=0}^{M-1} b_k^* z^{-k}}{1+\sum_{k=1}^{N-1} a_k^* z^{-k}} \tag{2}$$

FIR (finite-length impulse response) filter.

$$y[n] = \sum_{k=0}^{M-1} b_k^* x[n-k] \tag{3}$$

$$H(z) = \sum_{k=0}^{M-1} b_k^* z^{-k} \tag{4}$$

Several things are worth mentioning.

- Please note the convention of the complex conjugate on the filter coefficients. We will (usually) work with complex arithmetic, and this turns out to be a convenient representation mostly in terms of the Hermitian transpose.
- A primary concern with IIR filters is stability. Adaptive filters change their coefficients, so one needs to be assured that no change will move the filter to an unstable configuration. In some constrained cases (such

as the adaptive notch filter[1]) this can be done, but in general it is too difficult. Hence almost all adaptive filters are FIR, and we will deal with them exclusively.

- These filters are for temporal signal processing, where causality is a concern. Noncausal signal processing ("smoothing") is of course a possibility, as is multidimensional signal processing.

An FIR filter can he written as

$$y[n] \;=\; \mathbf{w}^H \mathbf{x}_n \tag{5}$$

where

$$\mathbf{x}_n \;\equiv\; (x[n]\; x[n-1]\; x[n-2]\; \ldots\; x[n-M+1])^T \tag{6}$$

represents the input in "shift register" format as a column vector and

$$\mathbf{w} \;\equiv\; (w_0\; w_1\; w_2\; \ldots\; w_{M-1})^T \tag{7}$$

is a column vector containing the filter coefficients. It is common to use $\mathbf{w}$ for these in the optimal signal processing context as opposed to $\mathbf{b}$ as would be expected from standard DSP (4).

1.2 Adaptation

Filters adapt by small movements that we will investigate soon. That is, we have

$$y[n] \;=\; \mathbf{w}_n^H \mathbf{x}_n \tag{8}$$

where $\mathbf{w}_n$ is the filter coefficient vector at time n and

$$\mathbf{w}_n \;=\; \mathbf{w}_{n-1} \;+\; \mu(\Delta\mathbf{w})_n \tag{9}$$

The step size (presumably small) is μ and the direction $(\Delta\mathbf{w})_n$ is a vector that is a function of input, previous output and some "desired" signal $d[n]$ that $y[n]$ is being adaptive to match.

Some canonical structures are

System Identification. The adaptive filter tries to match the structure of some unknown plant. it is assumed the input to the plant is available and $d[n]$ here is the plant's output.

[1]An ANF has transfer function $H(z) = \frac{1-2bz^{-1}+z^{-2}}{1-2\alpha bz^{-1}+\alpha^2 z^{-2}}$ where the b is adapted to control the notch frequency and α is slightly less than unity.

System Inversion. The adaptive filter is placed in series with an unknown plant, and tries to match the input of that plant. The desired signal $d[n]$ here is the input delayed by some suitable number of samples to make the inversion feasible.

Prediction. The desired signal $d[n]$ here is the input signal delayed by some samples, and the goal is to represent the structure of the random process $x[n]$.

Interference Cancelation. The system tries to match whatever is "matchable" in a signal, for example in adaptive noise cancelation.

The last is rather vague, so consider the example

$$d[n] = s[n] + v_1[n] \tag{10}$$

$$x[n] = s[n] + v_2[n] \tag{11}$$

It is clear that based on $\{x[n]\}$ at least some part (i.e. $s[n]$) of $d[n]$ can be matched. The noises $v_i[n]$ remain.

2 Correlation

2.1 Definitions and Properties

This will be very important. We'll assume wide-sense stationarity (wss) for analysis and design and that unless otherwise stated means of zero. We define

$$r[m] \equiv \mathcal{E}\{x[n]x^*[n-m]\} \tag{12}$$

where the convention is important, and we might refer to this as $r_{xx}[m]$ if there is confusion. It is easy to see that

$$r[-m] = r^*[m] \tag{13}$$

As for cross-correlation we define

$$r_{xy}[m] \equiv \mathcal{E}\{x[n]y^*[n-m]\} \tag{14}$$

for two random signals $x[n]$ & $y[n]$. In matrix form we have

$$\mathbf{R} \equiv \mathcal{E}\{\mathbf{x}_n\mathbf{x}_{n-m}^H\} \tag{15}$$

and it is important to stress that $\mathbf{R}$ can be so defined whether $\mathbf{x}_n$ represents a "vectorized" scalar time process as in (6) or whether it is a vector time process[2]. Cross-correlation matrices are be defined similarly; we will define cross-correlation vectors shortly. Note that the $(i,j)^{th}$ element of $\mathbf{R}$ is

$$\mathbf{R}(i,j) \;=\; \mathcal{E}\{\mathbf{x}_n(i)\mathbf{x}_n^*(j)\} \tag{16}$$

which is probably obvious, but in the case of a vectorized *wss* we have $\mathbf{R}(i,j) \;=\; \mathcal{E}\{x[n+1-i]x^*[n+1-j]\}$.

Here are some properties of the correlation matrix. When the proof is obvious it is suppressed.

- It is Hermitian: $\mathbf{R}^H = \mathbf{R}$.
- If $x[n]$ represents a "vectorized" scalar time process as in (6), then the correlation matrix has a special form

$$\mathbf{R} \;=\; \begin{pmatrix} r[0] & r[1] & r[2] & r[3] & \dots & r[M-1] \\ r[-1] & r[0] & r[1] & r[2] & \dots & r[M-2] \\ r[-2] & r[-1] & r[0] & r[1] & \dots & r[M-3] \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ r[-(M-1)] & r[-(M-2)] & r[-(M-3)] & r[-(M-4)] & \dots & r[0] \end{pmatrix} \tag{17}$$

  which is called "Toeplitz." A Toeplitz matrix has constant elements along all super- and sub-diagonals.

- It is non-negative definite.

$$\mathbf{w}^H\mathbf{R}\mathbf{w} = \mathcal{E}\{\mathbf{w}^H\mathbf{x}_n\mathbf{x}_n^H\mathbf{w}\} = \mathcal{E}\{|y[n]|^2\} \geq 0 \tag{18}$$

- Define the "backwards" vector

$$\mathbf{x}_n^B \;\equiv\; (x[n-M+1]\;x[n-M+2]\;x[n-M+3]\;\dots\;x[n])^T \tag{19}$$

  Then

$$\mathbf{R}_B \;\equiv\; \mathcal{E}\{\mathbf{x}_n^B(\mathbf{x}_{n-m}^B)^*\} \;=\; \mathbf{R}^* \;=\; \mathbf{R}^T \tag{20}$$

Please note that the text is for some reason fond of referring to the random process under study as $u[n]$, which I think is a little confusing in light of more typical the unit step nomenclature.

[2]An example of a vector time process is that the i^{th} element of $\mathbf{x}_n$ is the measurement from the i^{th} microphone in an array at time n.

2.2 Autoregressive Models

Although we will soon see them again in the context of optimization, we have enough ammunition now to understand them in terms of models. An autoregressive (AR) model is a special case of (2) with unity numerator; that is,

$$\begin{aligned} y[n] &= x[n] - \sum_{k=1}^{N-1} a_k^* y[n-k] && (21) \\ y[n] &= x[n] - \mathbf{a}^H \mathbf{y}_{n-1} && (22) \\ H(z) &= \frac{\sigma_x^2}{1 + \sum_{k=1}^{N-1} a_k^* z^{-k}} && (23) \end{aligned}$$

where the input $x[n]$ is assumed to be white (and usually but not necessarily Gaussian) with power σ_x^2. Define

$$\begin{aligned} \mathbf{r} &\equiv \mathcal{E}\{\mathbf{y}_{n-1} y[n]^*\} && (24) \\ &= (r[-1]\, r[-2]\ \ldots\ r[-M])^T && (25) \\ &= (r[1]\, r[2]\ \ldots\ r[M])^H && (26) \end{aligned}$$

Then from (23) we can write

$$\begin{aligned} \mathbf{r} &= \mathcal{E}\{\mathbf{y}_{n-1} y[n]^*\} && (27) \\ &= \mathcal{E}\{\mathbf{y}_{n-1}(x[n] - \mathbf{a}^H \mathbf{y}_{n-1}])^*\} && (28) \\ &= -\mathbf{R}\mathbf{a} && (29) \end{aligned}$$

in which the only subtlety is that $\mathbf{y}_{n-1}$ and $x[n]$ be independent – the latter is a "future" input to the AR filter. Repeating the last, we have

$$\mathbf{R}\mathbf{a} = -\mathbf{r} \qquad (30)$$

in which (30) represents the celebrated "Yule-Walker" equations. Note that since all quantities can be estimated from the data $\{y[n]\}$ (30) provides a way to estimate an AR process from its realization[3].

3 Eigenstuff

3.1 Basic Material

For a general $M \times M$ (square) matrix $\mathbf{A}$ the equation

$$\mathbf{A}\mathbf{q} = \lambda\mathbf{q} \qquad (31)$$

[3]The power σ_x^2 needs to be computed separately. We will address this later.

has M solutions in λ, although these may be complex. This is easy to see as (31) implies that the determinant of $(\mathbf{A} - \lambda\mathbf{I})$, which is an M^{th}-order polynomial in λ and hence has M roots, is zero. It is also easy to re-write all solutions of (31) as

$$\begin{aligned} \mathbf{AQ} &= \mathbf{Q\Lambda} && (32) \\ \mathbf{A} &= \mathbf{Q\Lambda Q}^{-1} && (33) \end{aligned}$$

in which

$$\mathbf{Q} \equiv \begin{pmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \mathbf{q}_1 & \mathbf{q}_2 & \ddots & \mathbf{q}_M \\ \downarrow & \downarrow & \dots & \downarrow \end{pmatrix} \qquad \mathbf{\Lambda} \equiv \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{pmatrix} \qquad (34)$$

are the eigenvectors arranged into a column and a diagonal matrix of eigenvalues. We have not shown that $\mathbf{Q}^{-1}$ in general exists for (33), but that is not in scope for this course. By convention eigenvectors are scaled to have unit length.

3.2 Hermitian Matrices

Our matrices will usually be correlation matrices, and these are Hermitian. We have the following:

Eigenvalues are non-negative and real.

$$0 \leq \mathbf{q}_i^H \mathbf{R} \mathbf{q}_i = \mathbf{q}_i^H (\lambda_i \mathbf{q}_i) = \lambda_i |\mathbf{q}_i|^2 \qquad (35)$$

Eigenvectors are orthonormal.

$$\begin{aligned} \mathbf{q}_i^H \mathbf{R} \mathbf{q}_j &= \mathbf{q}_i^H \mathbf{R} \mathbf{q}_j && (36) \\ \mathbf{q}_i^H (\lambda_j \mathbf{q}_j) &= (\mathbf{q}_i \lambda_i)^H \mathbf{q}_j && (37) \\ \lambda_i \mathbf{q}_i^H \mathbf{q}_j &= \lambda_j \mathbf{q}_i^H \mathbf{q}_j && (38) \end{aligned}$$

For this to be true either $\lambda_i = \lambda_j$ or $\mathbf{q}_i^H \mathbf{q}_j = 0$. For distinct eigenvalues the latter must be true. For $N \leq M$ repeated eigenvalues there is a subspace of dimension N (orthogonal to all the eigenvectors with different eigenvalues) that is an eigen-space: any vector within it has the eigen-property (31). By convention we take an orthonormal basis of that eigen-space as the eigenvectors; it doesn't matter much which such basis.

Diagonalization. The analog to (33) is

$$\mathbf{R} \;=\; \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^H \tag{39}$$

since $\mathbf{Q}^{-1} = \mathbf{Q}^H$ – see previous property of orthonormality. Actually

$$\mathbf{R} \;=\; \sum_{i=1}^{M} \lambda_i \mathbf{q}_i^H \mathbf{q}_i \tag{40}$$

is a rather nice way of expressing the same thing.

Matrix trace is sum of eigenvalues and determinant is product. This comes from (39), but actually applies to any matrix $\mathbf{A}$.

3.3 Relation to Power Spectrum

It's perhaps not obvious, but there is only one non-trivial situation where the eigenstuff and DFT have a strong relationship. This is when the correlation is "circulant" for a Toeplitz matrix, meaning

$$r[m] \;=\; r[M+m] \tag{41}$$

In the case that the process is real, this means $r[m] = r[M-m]$ as well: the top row of the Toeplitz matrix is symmetric around its midpoint. Consider

$$\mathbf{q}_p \;=\; \left(1\; e^{j2p\pi/M}\; e^{j4p\pi/M}\; e^{j6p\pi/M}\; \ldots\; e^{j(M-1)p2\pi/M}\right)^H \tag{42}$$

Then the $(m+1)^{st}$ element of the product $\mathbf{R}\mathbf{q}_p$ is

$$\begin{aligned}
(\mathbf{R}\mathbf{q}_p)(m+1) \;&=\; \sum_{k=0}^{M-1} r(k-m)e^{-jkp2\pi/M} && (43)\\
&=\; \sum_{k=0}^{m-1} r(k-m)e^{-jkp2\pi/M} \\
&\qquad\qquad + \sum_{k=m}^{M-1} r(k-m)e^{-jkp2\pi/M} && (44)\\
&=\; \sum_{k=0}^{m-1} r(M+k-m)e^{-jkp2\pi/M} \\
&\qquad\qquad + \sum_{k=m}^{M-1} r(k-m)e^{-jkp2\pi/M} && (45)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{k=M-m}^{M-1} r(k)e^{-j(k+m-M)p2\pi/M} \\
&\qquad\qquad + \sum_{k=0}^{M-m-1} r(k)e^{-j(k+m)p2\pi/M} \quad (46) \\
&= e^{-jmp2\pi/M} \sum_{k=M-m}^{M-1} r(k)e^{-jkp2\pi/M} \\
&\qquad\qquad + \sum_{k=0}^{M-m-1} r(k)e^{-jkp2\pi/M} \quad (47) \\
&= e^{-jmp2\pi/M} \left(\sum_{k=0}^{M-1} r(k)e^{-jkp2\pi/M} \right) \quad (48) \\
&= S(p)e^{-jmp2\pi/M} \quad (49)
\end{aligned}$$

which implies that the $\mathbf{q}_p$, which is the p^{th} DFT vector, is an eigenvector with eigenvalue the p^{th} element of the power spectrum. One could go backwards from (39) and show that the circulant condition must be true if the DFT relationship holds.

But the DFT and frequency analysis have a fairly strong relationship to Toeplitz covariance matrices, as we shall see. One example is this:

$$\begin{aligned}
\lambda_i &= \mathbf{q}_i^H \mathbf{R} \mathbf{q}_i \quad (50) \\
&= \sum_{k=1}^{M} \sum_{l=1}^{M} \mathbf{q}_i(k)^* \mathbf{q}_i(l) r[k-l] \quad (51) \\
&= \sum_{k=1}^{M} \sum_{l=1}^{M} \mathbf{q}_i(k)^* \mathbf{q}_i(l) \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) e^{j\omega(k-l)} d\omega \quad (52) \\
&= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) |Q_i(\omega)|^2 d\omega \quad (53)
\end{aligned}$$

where

$$Q_i(\omega) \equiv \sum_{k=0}^{M-1} \mathbf{q}_i(k+1) e^{-j\omega k} \quad (54)$$

is the DFT of the eigenvector. Now since

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |Q_i(\omega)|^2 d\omega = \sum_{k=1}^{M} |\mathbf{q}_i(k)|^2 \quad (55)$$

by Parseval and since this is unity, (53) tells us that

$$\min_{\omega} \{S(\omega)\} \le \lambda_i \le \max_{\omega} \{S(\omega)\} \quad (56)$$

which is nicer looking than it is useful, unfortunately.

At this point it is probably worth looking at a particular case, that of a sinusoid in noise. Suppose

$$x[n] \;=\; ae^{j\omega n} \;+\; \nu[n] \tag{57}$$

where a and $\{\nu[n]\}$ are complex Gaussian, respectively a random variable with variance σ_a^2 and a white noise process with power σ_ν^2. Then with

$$\gamma_\omega \;\equiv\; (1\, e^{j\omega}\, e^{j2\omega}\; \ldots\; e^{j(M-1)\omega})^T \tag{58}$$

we have

$$\mathbf{R} \;=\; \sigma_a^2\gamma(\omega)\gamma(\omega)^H \;+\; \sigma_\nu^2\mathbf{I} \tag{59}$$

The eigenstuff is dominated by one eigenvalue equal to $M\sigma_a^2 + \sigma_\nu^2$ with eigenvector proportional to $\gamma(\omega)$. The other eigenvectors are orthogonal to $\gamma(\omega)$ (of course!) and have common eigenvalue σ_ν^2.

ECE 6123
Advanced Signal Processing
Optimal Filtering

Peter Willett

Fall 2017

1 Principle of Orthogonality

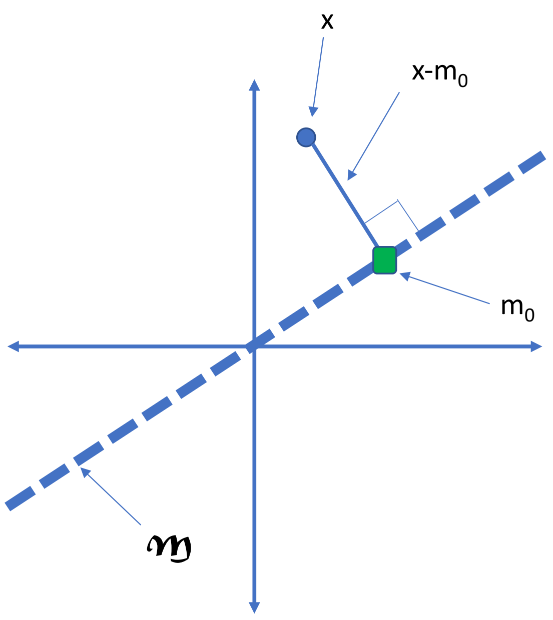
1.1 The Principle

Let $\mathcal{X}$ be a Hilbert space (a complete vector space with an inner-product defined) and $\mathcal{M} \subseteq \mathcal{X}$ is a subspace. Then for any $x \in \mathcal{X}$

$$(x - m_0, m) \;=\; 0 \qquad\qquad \forall m \in \mathcal{M} \tag{1}$$

is a necessary and sufficient condition that $m_0 \in \mathcal{M}$ minimize $||x - m_0||^2$.

The *poo* idea is as sketched below.



For us in this course the inner product (x, y) is $\mathcal{E}\{x, y^*\}$, and hence the norm $||x||^2$ is $\mathcal{E}\{x^2\}$.

1.2 Sufficiency

Let's try some other $m_1 \in \mathcal{M}$. Then

$$\begin{aligned}
||x - m_1||^2 &= ||x - m_0 + m_0 - m_1||^2 && (2)\\
&= ||x - m_0||^2 \;+\; (x - m_0, m_0 - m_1) \\
&\qquad +\; (m_0 - m_1, x - m_0) \;+\; ||m_0 - m_1||^2 && (3)\\
&= ||x - m_0||^2 \;+\; ||m_0 - m_1||^2 && (4)\\
&\geq ||x - m_0||^2 && (5)
\end{aligned}$$

with equality if and only if $m_1 = m_0$. The second line follows due to (1) and the fact that $(m_0 - m_1) \in \mathcal{M}$.

1.3 Necessity

Assume that $m_0 \in \mathcal{M}$ minimizes $||x - m||^2$ but that

$$(x - m_0, m) \;=\; \delta \qquad (6)$$

for some $m \in \mathcal{M}$ with $||m||^2 = 1$. Then let's try $m_0 + \delta m$ instead of m_0. We have

$$\begin{aligned}
||x - (m_0 + \delta m)||^2 &= ||x - m_0||^2 \;-\; (x - m_0, \delta m) \\
&\qquad -\; (\delta m, x - m_0) \;+\; ||\delta m||^2 && (7)\\
&= ||x - m_0||^2 \;-\; \delta^*(x - m_0, m) \\
&\qquad -\; \delta(m, x - m_0) \;+\; |\delta|^2 && (8)\\
&= ||x - m_0||^2 \;-\; |\delta|^2 && (9)\\
&< ||x - m_0||^2 && (10)
\end{aligned}$$

This contradicts that $m_0 + \delta_m$ minimizes $||x - m||^2$, so (6) cannot be true.

1.4 An Application

Suppose we have an observation vector $\mathbf{x}$ and we wish to approximate the vector $\mathbf{y}$ by $\hat{\mathbf{y}} = \mathbf{A}\mathbf{x}$ so as to minimize $||\mathbf{y} - \hat{\mathbf{y}}||^2$. The the *poo* tells us

$$\begin{aligned}
(\mathbf{y} - \hat{\mathbf{y}}, \mathbf{x}) &= 0 && (11)\\
\mathcal{E}\{(\mathbf{y} - \mathbf{A}\mathbf{x}\mathbf{x}^H) &= 0 && (12)\\
\mathbf{R}_{yx} \;-\; \mathbf{A}\mathbf{R}_{xx} &= 0 && (13)\\
\mathbf{A} &= \mathbf{R}_{yx}\mathbf{R}_{xx}^{-1} && (14)
\end{aligned}$$

Admittedly there are other ways to solve for this, but the *poo* is certainly quick and elegant.

2 FIR Weiner Filtering

With the usual desired signal $d[n]$ and data[1] vector $\mathbf{u}_n$ we minimize

$$J(\mathbf{w}) = \mathcal{E}\{||d[n] - \mathbf{w}^H\mathbf{u}_n||^2\} \tag{15}$$

via the *poo* as

$$\begin{aligned} \mathcal{E}\{((d[n] - \mathbf{w}_o^H\mathbf{u}_n)\mathbf{u}_n)^H\} &= 0 & (16)\\ \mathbf{w}_o^H &= \mathbf{R}_u^{-1}\mathbf{p} & (17) \end{aligned}$$

where

$$\mathbf{p} = \begin{pmatrix} p[0] \\ p[-1] \\ \vdots \\ p[-(M+1)] \end{pmatrix} \equiv \begin{pmatrix} \mathcal{E}\{(d[n]u[n]^*\} \\ \mathcal{E}\{(d[n]u[n-1]^*\} \\ \vdots \\ \mathcal{E}\{(d[n]u[n-(M-1)]^*\} \end{pmatrix} \tag{18}$$

and $\mathbf{R}_u$ has the usual correlation matrix definition. Substituting (17) to (15) it is easy to see that

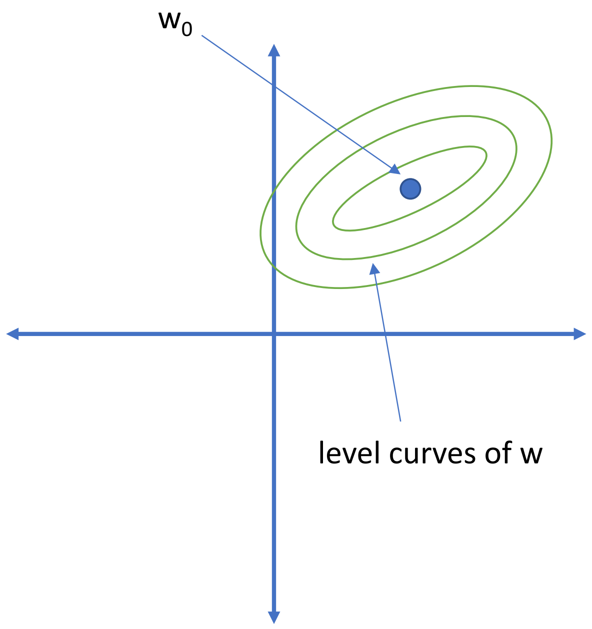
$$\begin{aligned} J(\mathbf{w}) &= \mathcal{E}\{(d[n] - \mathbf{w}^H\mathbf{u}_n)(d[n] - \mathbf{w}^H\mathbf{u}_n)^*\} & (19)\\ &= \sigma_d^2 - \mathbf{p}^H\mathbf{w} - \mathbf{w}^H\mathbf{p} + \mathbf{w}^H\mathbf{R}_u\mathbf{w} & (20)\\ J(\mathbf{w}_o) &= \sigma_d^2 - \mathbf{p}^H\mathbf{R}_u^{-1}\mathbf{p} & (21)\\ &= \sigma_d^2 - \mathbf{p}^H\mathbf{w}_o & (22)\\ &= \sigma_d^2 - \mathbf{w}_o^H\mathbf{p} & (23) \end{aligned}$$

From (20) we can write

$$\begin{aligned} J(\mathbf{w}) &= \sigma_d^2 - \mathbf{p}^H\mathbf{R}_u\mathbf{w} - \mathbf{w}^H\mathbf{R}_u\mathbf{p} + \mathbf{w}^H\mathbf{R}_u\mathbf{w} & (24)\\ &= \sigma_d^2 - \mathbf{p}^H\mathbf{w} - \mathbf{w}^H\mathbf{p} + \mathbf{w}^H\mathbf{R}_u\mathbf{w} & (25)\\ &= \sigma_d^2 - \mathbf{w}_o^H\mathbf{R}_u\mathbf{w} - \mathbf{w}^H\mathbf{R}_u\mathbf{w} + \mathbf{w}^H\mathbf{R}_u\mathbf{w} & \\ &\quad + \mathbf{w}_o^H\mathbf{R}_u\mathbf{w}_o - \mathbf{w}_o^H\mathbf{R}_u\mathbf{w}_o & (26)\\ &= J(\mathbf{w}_o) + (\mathbf{w} - \mathbf{w}_o)^H\mathbf{R}_u(\mathbf{w} - \mathbf{w}_o) & (27) \end{aligned}$$

As indicated in the sketch that follows (for $M = 2$) this means that the $J(\mathbf{w})$ is concave – actually quadratic – in $\mathbf{w}$, and has a unique minimum at the Wiener solution $\mathbf{w}_o$. This has implications for the adaptive filtering that will follow.

[1] Sorry: unlike me, the author loves $\mathbf{u}$.



3 IIR Wiener Filtering

3.1 The Complete-Data Case

Suppose we have (more generality is possible!)

$$u[n] = s[n] + w_1[n] \tag{28}$$
$$s[n] = d[n-n_0] + w_2[n] \tag{29}$$

where $\{d[n]\}$ is *wss* with correlation $r_{dd}[m]$ and $w_i[n]$ are noises with correlations $r_{w_iw_i}[m]$. For simplicity in this section let's assume everything is real. We wish to "filter" using

$$\hat{d}[n] = \sum_k w[k]u[n-k] \tag{30}$$

Then the filtering problem is easily solved by the *poo*:

$$\mathcal{E}\{(d[n] - \sum_k w[k]u[n-k])u(n-m)\} = 0 \qquad \forall m \tag{31}$$

or

$$r_{dd}[m-n_0] = \sum_k w[k](r_{dd}[m-k] + r_{w_1w_1}[m-k] + r_{w_2w_2}[m-k] \qquad \forall m \tag{32}$$

Using z-transforms we have

$$z^{-n_0} R_{dd}[z] \;=\; W(z)(R_{dd}[z] \;+\; R_{w_1 w_1}[z] \;+\; R_{w_2 w_2}[z]) \tag{33}$$

or

$$\begin{aligned} W(z) \;&=\; \frac{z^{-n_0} R_{dd}[z]}{R_{dd}[z] \;+\; R_{w_1 w_1}[z] \;+\; R_{w_2 w_2}[z]} & (34) \\ &=\; \frac{R_{du}[z]}{R_{uu}[z]} & (35) \end{aligned}$$

with the obvious definition of these two z-transforms. This is nice, but in general $W(z)$ will not be realizable, as it will be non-causal: it will have poles outside the unit circle.

3.2 Causal Wiener Filtering

With the constraint that $W(z)$ be causal we have (30) as

$$\hat{d}[n] \;=\; \sum_{k=0}^{\infty} w[k]u[n-k] \tag{36}$$

and (31) now

$$\mathcal{E}\{(d[n] - \sum_k w[k]u[n-k])u(n-m)\} \;=\; 0 \qquad \forall m \geq 0 \tag{37}$$

or

$$r_{dd}[m-n_0] \;=\; \sum_k w[k](r_{dd}[m-k] + r_{w_1 w_1}[m-k] + r_{w_2 w_2}[m-k] \qquad \forall m \geq 0 \tag{38}$$

Everything is fine, but since (38) is not true for all m we cannot take a z-transform and equate the two sides. Fortunately we can write

$$g[m] \;=\; r_{dd}[m-n_0] - \sum_k w[k](r_{dd}[m-k] - r_{w_1 w_1}[m-k] + r_{w_2 w_2}[m-k] \qquad \forall m \tag{39}$$

We don't especially care what $g[m]$ is.

But we *do* know two things. First, we can write it as

$$G(z) \;=\; z^{-n_0} R_{dd}[z] \;-\; W(z) R_{uu}[z] \tag{40}$$

And second: we know that it $g[m] = 0$ for $m \geq 0$ while $w[k] = 0$ for $k < 0$. Now factor

$$R_{uu}(z) \;=\; [R_{uu}(z)]_+ \, [R_{uu}(z)]_- \tag{41}$$

where the two parts refer to the positive- and negative-time portions of $r_{uu}[m]$. In the case that $R_{uu}(z)$ is rational this may be accomplished by grouping all poles and zeros inside the unit circle into $[R_{uu}(z)]_+$ and all those outside the unit circle into $[R_{uu}(z)]_-$; but more generally it requires taking an inverse z-transform of $R_{uu}(z)$, separating the left- and right-sided behavior, then taking z-transforms of the two, separately.

Then we have

$$\underbrace{\frac{G(z)}{[R_{uu}(z)]_-}}_{m<0} = \frac{z^{-n_0}R_{dd}[z]}{[R_{uu}(z)]_-} - \underbrace{W(z)\,[R_{uu}(z)]_+}_{m\geq 0} \tag{42}$$

after division. Now since the LHS is the convolution of two left-sided sequences it is left-sided – that is, it is zero[2] for $m \geq 0$. Likewise the second term on the RHS is the convolution of two right-sided sequences, hence it is right-sided – that is, it is zero for $m < 0$. Altogether this gives us a nice equality:

$$W(z) = \frac{1}{[R_{uu}(z)]_+}\left[\frac{z^{-n_0}R_{dd}[z]}{[R_{uu}(z)]_-}\right]_+ \tag{43}$$

or

$$W(z) = \frac{1}{[R_{uu}(z)]_+}\left[\frac{R_{du}[z]}{[R_{uu}(z)]_-}\right]_+ \tag{44}$$

with the more general notation.

Here is a simple example, rather simpler than what was done in class. Suppose we have

$$u[n] = d[n-n_0] \tag{45}$$

and $r_{dd}[m] = \rho^{|m|}$ and $\rho \in \Re$. Then

$$R_{dd}(z) = \frac{1-\rho^2}{(1-\rho z)(1-\rho z^{-1})} \tag{46}$$

$$R_{du}(z) = z^{-n_0}R_{dd}(z) \tag{47}$$

Then

$$[R_{uu}(z)]_+ = \frac{1-\rho^2}{1-\rho z^{-1}} \tag{48}$$

$$[R_{uu}(z)]_- = \frac{1}{1-\rho z} \tag{49}$$

[2]This is indicated by the under-braces.

where the apportionment of the constant doesn't much matter – one could take the square root and make the terms symmetric. Then

$$\begin{aligned} W(z) &= \frac{1}{[R_{uu}(z)]_+}\left[\frac{R_{du}[z]}{[R_{uu}(z)]_-}\right]_+ && (50)\\ &= \left(\frac{1-\rho z^{-1}}{1-\rho^2}\right)\left[\left(\frac{z^{-n_0}(1-\rho^2)}{(1\ -\ \rho z)(1\ -\ \rho z^{-1})}\right)\left(\frac{1\ -\ \rho z}{1}\right)\right]_+ && (51)\\ &= \left(\frac{1-\rho z^{-1}}{1-\rho^2}\right)\left(\frac{z^{-n_0}(1-\rho^2)}{1\ -\ \rho z^{-1}}\right) && (52)\\ &= z^{-n_0} && (53)\end{aligned}$$

for the case that $n_0 \geq 0$. Not surprisingly, then, the optimal Wiener filter chooses $\hat{d}[n] = u[n - n_0]$ for non-negative n_0.

The case of $n_0 < 0$ is more interesting, since now a prediction is being made. In this case we follow from (51) in a different way:

$$\begin{aligned} &\left[\left(\frac{z^{-n_0}(1-\rho^2)}{(1\ -\ \rho z)(1\ -\ \rho z^{-1})}\right)\left(\frac{1\ -\ \rho z}{1}\right)\right]_+ \\ &= \left[\left(\frac{z^{-n_0}(1-\rho^2)}{1\ -\ \rho z^{-1}}\right)\right]_+ && (54)\\ &= (1-\rho^2)\left(\sum_{n=0}^{\infty}\rho^{-(n+n_0)}z^{-n}\right) && (55)\\ &= \frac{\rho^{-n_0}(1-\rho^2)}{1\ -\ \rho z^{-1}} && (56)\end{aligned}$$

Now we have

$$W(z) \ = \ \rho^{-n_0} \qquad (57)$$

from (52); or in fact $\hat{d}[n] = \rho^{-n_0}u[n]$ for $n_0 < 0$. This too, makes sense: you are predicting, and the best two-step (say) prediction is ρ^2 times the most recent value.

ECE 6123
Advanced Signal Processing
Linear Prediction

Peter Willett

Fall 2017

1 AR Models and Wiener Prediction

1.1 Relationship

Suppose we wish to "predict" $u[n]$ linearly, based on $\{u[n-1], u[n-2], \ldots, u[n-M]\}$ via

$$\hat{u}[n] \;=\; \mathbf{w}^H \mathbf{u}_{n-1} \tag{1}$$

Then the Wiener filter is straightforward:

$$\mathbf{R}\mathbf{w} \;=\; \mathbf{r} \tag{2}$$

where

$$\mathbf{r} \;\equiv\; \mathcal{E}\left\{ \begin{pmatrix} u[n-1] \\ u[n-2] \\ \vdots \\ u[n-M] \end{pmatrix} u[n]^* \right\} \;=\; \begin{pmatrix} r[-1] \\ r[-2] \\ \vdots \\ r[-M] \end{pmatrix} \tag{3}$$

Why should we want to make such a prediction when all we need is to wait a sample to see the real value $u[n]$? The answer is that if $u[n]$ were what we wanted we should do just that: wait for it. The reason to pose this as a prediction problem is that the structure is important and useful.

Equation (2) should look familiar. In fact, to repeat from the first lecture, we have the autoregressive (AR) model:

$$u[n] \;=\; \nu[n] - \sum_{k=1}^{M-1} a_k^* u[n-k] \tag{4}$$

$$u[n] \;=\; x[n] - \mathbf{a}^H \mathbf{u}_{n-1} \tag{5}$$

where the input $\nu[n]$ is assumed to be white (and usually but not necessarily Gaussian) with power σ_ν^2. In the introduction we found that we could recover the AR model (i.e., the a_k's) from knowledge of the autocorrelation lags (the $r[m]$'s) via

$$\mathbf{R}\mathbf{a} \;=\; -\mathbf{r} \tag{6}$$

which are the "Yule-Walker" equations. That is: $\mathbf{w} = -\mathbf{a}$. And that does make a good amount of sense: according to (4), using $\hat{u}[n] = \mathbf{w}^H \mathbf{u}_{n-1}$ eliminates all the "randomness" in $u[n]$ except that from $\nu[n]$; and $\nu[n]$ can't be removed.

1.2 Augmented Yule-Walker Equations

Some helpful insight provided by the Wiener approach to AR modeling is from the generic Wiener equation

$$J_{min} = \sigma_d^2 - \mathbf{w}_o^H \mathbf{p} \tag{7}$$

In the AR case (2) we write this as

$$\sigma_\nu^2 = r[0] - \mathbf{w}^H \mathbf{r} \tag{8}$$

We can concatenate (8) and (2) to get

$$\begin{pmatrix} r[0] & \mathbf{r}_M^H \\ \mathbf{r}_M & \mathbf{R}_M \end{pmatrix} \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} = \begin{pmatrix} P_M \\ 0 \end{pmatrix} \tag{9}$$

where we have stressed the dimension of the matrix and vector in the subscripts. It is probably useful to note the full matrix in long form

$$\begin{pmatrix} r[0] & r[1] & r[2] & \dots & r[M] \\ r[-1] & r[0] & r[1] & \dots & r[M-1] \\ \vdots & \vdots & \ddots & \vdots & \\ r[-M] & r[-(M-1)] & r[-(M-2)] & \dots & r[0] \end{pmatrix} \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} = \begin{pmatrix} P_M \\ 0 \end{pmatrix} \tag{10}$$

or

$$\mathbf{R}_{M+1} \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} = \begin{pmatrix} P_M \\ 0 \end{pmatrix} \tag{11}$$

to see the naturalness of this concatenation.

Let us re-define[1] the AR vector $\mathbf{a}$ as

$$\mathbf{a} \equiv \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{a}_{old} \end{pmatrix} = \begin{pmatrix} 1 \\ a_1 \\ a_2 \\ \vdots \\ a_M \end{pmatrix} \tag{12}$$

[1]Sorry, but this has to be done.

where $\mathbf{a}_{old}$ is as in (2) and (4). Then

$$\mathbf{R}_{M+1}\mathbf{a}_M = \begin{pmatrix} P_M \\ 0 \end{pmatrix} \tag{13}$$

is a way to re-write (11). Here the subscript on $\mathbf{a}$ denotes the order of the predictor – it is a vector of length $M+1$.

1.3 Backwards Prediction

Suppose we want to "predict" $u[n-M]$ based on $\{u[n-M+1], , u[n-m+2], \ldots, u[n-1, u[n]\}$ (i.e., $\mathbf{u}_n$). The Wiener solution $\hat{u}[n-M] = \mathbf{g}^H\mathbf{u}_n$ is pretty simple:

$$\mathbf{R}\mathbf{g} = \mathbf{p} = \mathcal{E}\{u[n-M]^*\mathbf{u}_n\} = \begin{pmatrix} r[M] \\ r[M-1] \\ \vdots \\ r[1] \end{pmatrix} = \mathbf{r}^{B*} \tag{14}$$

and that does indeed mean "backwards and conjugated". We can concatenate this and the Wiener error equation as

$$\begin{pmatrix} \mathbf{R}_M & \mathbf{r}^{B*} \\ (\mathbf{r}^B)^T & r[0] \end{pmatrix} \begin{pmatrix} -\mathbf{g} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ P_M^{backwards} \end{pmatrix} \tag{15}$$

where $P_M^{backwards}$ is the Wiener error for the backward prediction. Now let's write the Wiener solution another way, using $\hat{u}[n-M] = (\mathbf{g}^B)^H\mathbf{u}_n^B$, which is identical in effect to the "normal" ordering. Now we have

$$\mathcal{E}\{\mathbf{u}^B(\mathbf{u}^B)^H\}\mathbf{g}^B = \mathcal{E}\{\mathbf{u}^B u[n-M]^*\} \tag{16}$$

$$\mathbf{R}^*\mathbf{g}^B = \mathbf{r}^* \tag{17}$$

$$\mathbf{R}\mathbf{g}^{B*} = \mathbf{r} \tag{18}$$

Compare (18) to (2) and it clear that we can write

$$\mathbf{g}^{B*} = \mathbf{w} \tag{19}$$

$$\begin{pmatrix} \mathbf{g}^{B*} \\ 1 \end{pmatrix} = \mathbf{a} \tag{20}$$

$$P_M^{backwards} = P_M \tag{21}$$

That is: the AR process "looks the same" whether viewed in forward time or reverse time. That's a cute point, but the main by-product of this analysis is that we can write

$$\begin{pmatrix} \mathbf{R}_M & \mathbf{r}^{B*} \\ \mathbf{r} & r[0] \end{pmatrix} \begin{pmatrix} -\mathbf{w}^{B*} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ P_M \end{pmatrix} \tag{22}$$

or

$$\mathbf{R}_{M+1}\mathbf{a}_M^{B*} = \begin{pmatrix} 0 \\ P_M \end{pmatrix} \tag{23}$$

as an alternate way to write the augmented Yule-Walker equations.

2 The Levinson-Durbin Algorithm

Solution of $M+1$ linear equations requires $\mathcal{O}((M+1)^3)$ operations. But the YW equations are special: they actually contain only $M+1$ unique "inputs" $\{r[0],\ r[1], \ldots,\ r[M]\}$ whereas the general complexity applies to $(M+1)^2 + (M+1)$. Can we exploit this structure? Note that the YW equations are rather unusual, in that there are M unknowns on the LHS and one (P_M) on the RHS.

Of course the answer is yes: the LD algorithm. We begin by proposing a structure.

$$\mathbf{a}_m = \begin{pmatrix} \mathbf{a}_{m-1} \\ 0 \end{pmatrix} + \Gamma_m \begin{pmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{pmatrix} \tag{24}$$

where we are noting the order as m rather than the *true* (or at least we assume it's true) model order M, since we will start with $m = 0$ and work up to $m = M$. This will be an inductive development, so we need to show that the structure replicates. The structure might be suggested by (13) and (23); but we need to show that it works.

We multiply (24) by $\mathbf{R}_{M+1}$; we want this product to be consistent with (13). We have

$$\begin{aligned} \mathbf{R}_{m+1}\mathbf{a}_m &= \begin{pmatrix} \mathbf{R}_m & \mathbf{r}_m^{B*} \\ (\mathbf{r}^B)^T & r[0] \end{pmatrix} \begin{pmatrix} \mathbf{a}_{m-1} \\ 0 \end{pmatrix} \\ &\quad + \Gamma_m \begin{pmatrix} r[0] & \mathbf{r}_m^H \\ \mathbf{r}_m & \mathbf{R}_m \end{pmatrix} \begin{pmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{pmatrix} && (25) \\ &= \begin{pmatrix} \mathbf{R}_m \mathbf{a}_{m-1} \\ (\mathbf{r}^B)^T \mathbf{a}_{m-1} \end{pmatrix} + \Gamma_m \begin{pmatrix} \mathbf{r}_m^T \mathbf{a}_{m-1}^{B*} \\ \mathbf{R}_m \mathbf{a}_{m-1}^{B*} \end{pmatrix} && (26) \\ &= \begin{pmatrix} P_{m-1} \\ 0 \\ \Delta_{m-1} \end{pmatrix} + \Gamma_m \begin{pmatrix} \Delta_{m-1}^* \\ 0 \\ P_{m-1} \end{pmatrix} && (27) \end{aligned}$$

where

$$\Delta_{m-1} \equiv (\mathbf{r}^B)^T \mathbf{a}_{m-1} \tag{28}$$

So, how do we make (27) into (13)? Easy: choose

$$\Gamma = \frac{-\Delta_{m-1}}{P_{m-1}} \tag{29}$$

Once we do that, we have

$$\mathbf{a}_m = \begin{pmatrix} \mathbf{a}_{m-1} \\ 0 \end{pmatrix} + \Gamma_m \begin{pmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{pmatrix} \tag{30}$$

as desired, and

$$P_m = P_{m-1}(1 - |\Gamma|^2) \tag{31}$$

The LD algorithm consists of starting with $P_0 = r[0]$ & $\mathbf{a}_0 = 1$, and iterating on m: (28), (29), (30) then (31). Notice that the missing RHS of (13) – that is, P_m – is created as is needed.

3 Other Neat Things

3.1 The Lattice Structure

Consider (4). We can turn this into a "prediction error filter" (actually a *forward* PEF) as

$$u[n] + \sum_{k=1}^{m} a_k^* u[n-k] = \nu[n] \tag{32}$$

$$f_m[n] = u[n] - \left(\sum_{k=1}^{m} w_k^* u[n-k] \right) \tag{33}$$

$$f_m[n] = \sum_{k=0}^{m} a_{m,k}^* u[n-k]) \tag{34}$$

$$F_m(z) = H_{f,m}(z) U(z) \tag{35}$$

where (32) is a restatement of the AR model, the second is (33) is the same in Wiener filtering notation, where $f_m[n]$ denotes the prediction error for the m^{th}-order predictor, (34) uses the new formulation for $\mathbf{a}_m$ and (35) is the z-transform of (33). This is basically presented to suggest what is meant by $f_m[n]$ and $H_{f,m}(z)$. We do the same thing for "backward" prediction errors

$b_m[n]$ and $H_{b,m}(z)$. We'll use

$$b_m[n] = u[n-m] - \left(\sum_{k=1}^{m} g_k^* u[n+1-k]\right) \tag{36}$$

$$b_m[n] = u[n-m] - \left(\sum_{k=1}^{m} w_k u[n-m+k]\right) \tag{37}$$

$$b_m[n] = \sum_{k=0}^{m} a_{m,k} u[n-m+k]) \tag{38}$$

$$B_m(z) = H_{b,m}(z)U(z) \tag{39}$$

and it should be noted closely that $b_m[n]$ refers to the "error" in predicting $u[n-m]$.

We write

$$\begin{aligned} H_{f,m}(z) &= \sum_{k=0}^{m} a_{m,k}^* z^{-k} &(40)\\ &= \sum_{k=0}^{m-1} a_{m-1,k}^* z^{-k} + \Gamma_m^* \sum_{k=0}^{m-1} a_{m-1,m-k-1}^* z^{-k-1} &(41)\\ &= H_{f,m-1}(z) + \Gamma_m^* z^{-1} H_{b,m-1}(z) &(42)\end{aligned}$$

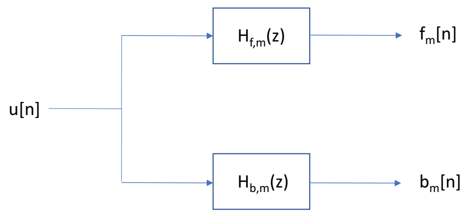
where (40) leads to (41) via (30). We also have

$$\begin{aligned} H_{b,m}(z) &= \sum_{k=0}^{m} a_{m,m-k}^* z^{-k} &(43)\\ &= \sum_{k=0}^{m} a_{m,k}^* z^{-(m-k)} &(44)\\ &= z^{-m}\left(H_{f,m}(1/z^*)\right)^* &(45)\end{aligned}$$

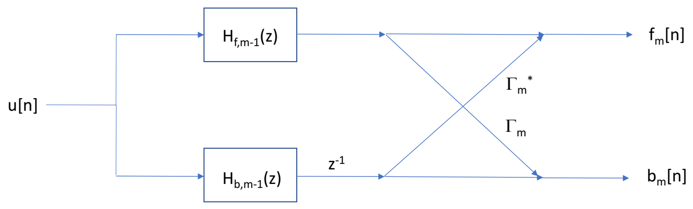
Substituting (42) into (45) we have

$$\begin{aligned} H_{b,m}(z) &= z^{-m}\left(H_{f,m-1}(1/z^*)\right)^* + \Gamma_m^* z z^{-m}\left(H_{b,m-1}(1/z^*)\right)^* &(46)\\ &= z^{-1} H_{b,m-1}(z) + \Gamma_m^* z z^{-m}\left(z^{m-1} H_{b,m-1}(1/z^*)\right) &(47)\\ &= \Gamma_m^* H_{f,m-1}(z) z^{-1} + z^{-1} H_{b,m-1}(z) &(48)\end{aligned}$$

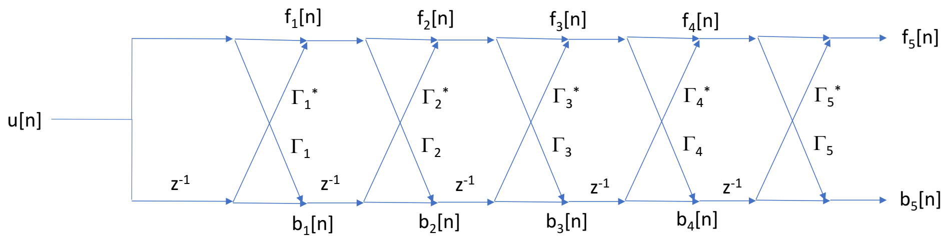
Then we have



being equivalent to



and overall we have the nice structure



exemplified for a fifth-order PEF.

3.2 Orthogonality

This is pretty simple once you remember that the *p.o.o.* governs all this optimal filtering. Let's assume that $i < j$ and remember that $b_i[n]$ is a linear function of $\{u[n-i],\ u[n-i+1],\ \dots,\ u[n]\}$. By the *p.o.o.*, $b_j[n]$ is orthogonal to $\{u[n-j+1],\ u[n-j+2], \dots,\ u[n]\}$, and hence it is orthogonal to its subset $\{u[n-i+1],\ u[n-i+2], \dots,\ u[n]\}$. Case closed: $b_i[n]$ and $b_j[n]$ are orthogonal

$$\mathcal{E}\{b_i[n]b_j[n]^*\} \ = \ 0 \tag{49}$$

for all $i \neq j$ – and this expectation is by definition P_i for the case $i = j$.

Now let's write this out in full:

$$b_0[n] \ = \ u[n] \tag{50}$$

$$\begin{aligned}
b_1[n] &= u[n-1] + a_{1,1}u[n] && (51)\\
b_2[n] &= u[n-2] + a_{2,1}u[n-1] + a_{2,2}u[n] && (52)\\
\vdots &= \vdots \\
b_m[n] &= u[n-m] + a_{m,1}u[n-m+1] + \ldots a_{m,m}u[n] && (53)
\end{aligned}$$

This can be written as

$$\mathbf{b}_n = \mathbf{L}\mathbf{u}_n \tag{54}$$

where $\mathbf{L}$ is a lower triangular matrix containing in the i^{th} row the backwards PEF. Since orthogonality tells us that the b's are uncorrelated, we have

$$\mathbf{D} = \mathbf{L}\mathbf{R}\mathbf{L}^H \tag{55}$$

where $\mathbf{D}$ is a diagonal matrix with $(i,i)^{th}$ element P_i. We can also write (55) as

$$\mathbf{R} = \mathbf{L}^{-1}\mathbf{D}\mathbf{L}^{-H} \tag{56}$$

indicating that the PEF's and the correspond error powers are actually the LDU decomposition of the correlation matrix of the data.

3.3 Stability

It's obvious that the PEFs are stable – they're FIR. But one reason to create a PEF structure is to be able to recreate the corresponding AR model. Since that involves the reciprocal of the PEF, we need to know if the zeros of the PEF are inside the unit circle. If not the AR model is unstable and trying to use one would be hopeless.

We repeat (42)

$$H_{f,m}(z) = H_{f,m-1}(z) + \Gamma_m^* z^{-1} H_{b,m-1}(z) \tag{57}$$

and plug in (45)

$$H_{b,m-1}(z) = z^{-(m-1)} \left(H_{f,m-1}(1/z^*)\right)^* \tag{58}$$

to get

$$H_{f,m}(z) = H_{f,m-1}(z) + \Gamma_m^* z^{-m} \left(H_{f,m-1}(1/z^*)\right)^* \tag{59}$$

We convert this to the DTFT (discrete-time Fourier transform) as

$$H_{f,m}(e^{j\omega}) = H_{f,m-1}(e^{j\omega}) + \Gamma_m^* e^{-jm\omega} H_{f,m-1}(e^{j\omega})^* \tag{60}$$

Since (60) comprises the sum of two complex vectors, the first one of magnitude $|H_{f,m-1}(e^{j\omega})|$ and the second one, since $|\Gamma_m| < 1$, of magnitude less than $|H_{f,m-1}(e^{j\omega})|$, we can see that as ω travels from zero to 2π the total phase change of $H_{f,m}(e^{j\omega})$ must be the same as of $H_{f,m-1}(e^{j\omega})$ – the second term in the sum can have no effect. As such $H_{f,m}(e^{j\omega})$ begins and ends its phase at the same point $\forall m$.

We turn now to a generic FIR model

$$\begin{aligned} H(z) &= \prod_{i=1}^{m}(1 - z_i z^{-1}) && (61) \\ &= z^{-m}\prod_{i=1}^{m}(z - z_i) && (62) \\ H(e^{j\omega}) &= e^{-jm\omega}\prod_{i=1}^{m}(e^{j\omega} - z_i) && (63) \end{aligned}$$

It is easy to see that a NASC for all zeros to be inside the unit circle is that the total phase change as ω travels from zero to 2π must be zero. That is what we have, hence the FPEF is indeed stable – the FPEF is *minimum-phase*. It can be shown that all zero are *outside* the unit circle for the BPEF (it is maximum-phase).

ECE 6123
Advanced Signal Processing
Adaptive Filtering with LMS

Peter Willett

Fall 2017

1 The Gradient Descent Idea

1.1 Finding the Wiener Filter

Suppose we begin with the Weiner filtering cost function

$$\begin{aligned} J(\mathbf{w}) &= \mathcal{E}\{|d[n] - \mathbf{w}^H\mathbf{u}_n\} & (1)\\ &= \sigma_d^2 - 2\Re\{\mathbf{w}^H\mathbf{p}\} + \mathbf{w}^H\mathbf{R}\mathbf{w} & (2)\end{aligned}$$

where the terms are as usual. Suppose further that we wish to achieve this iteratively, as opposed to in one step as before. One might reasonably: Why do this when you manifestly *can* get to the solution in one step? The answer is that we will attempt to see how this can work when the correlations $\mathbf{p}$ and $\mathbf{R}$ are not knows or are changing. But that will come.

The basic idea is to observed that one can reduce the error by moving in a direction of "steepest-descent" which is

$$\mathbf{w}_n = \mathbf{w}_{n-1} - \mu\nabla J_{\mathbf{w}}(\mathbf{w}_{n-1}) \quad (3)$$

where μ is some small step size and of course $\nabla_{\mathbf{w}}$ represents the gradient with respect to $\mathbf{w}$.

1.2 Interlude about Complex Gradients

This subsection is eminently skippable. However, it is not perhaps obvious how to take a gradient of a linear of quadratic form when complex vectors are involved. The answer.turns out to be: it's exactly what you think it is. We begin with (2), and write

$$\begin{aligned} \mathbf{w} &\equiv \mathbf{w}_r + j\mathbf{w}_i & (4)\\ \mathbf{p} &\equiv \mathbf{p}_r + j\mathbf{p}_i & (5)\\ \mathbf{R} &\equiv \mathbf{R}_r + j\mathbf{R}_i & (6)\end{aligned}$$

where all RHS vectors and matrices are real and clearly the subscript refer to real and imaginary parts. We have (2) as

$$\begin{aligned} J(\mathbf{w}) &= \sigma_d^2 - 2\mathbf{w}_r^T\mathbf{p}_r - 2\mathbf{w}_i^T\mathbf{p}_i + \mathbf{w}_r^T\mathbf{R}_r\mathbf{w}_r \\ &\quad - \mathbf{w}_r^T\mathbf{R}_i\mathbf{w}_i + \mathbf{w}_i^T\mathbf{R}_i\mathbf{w}_r + \mathbf{w}_i^T\mathbf{R}_r\mathbf{w}_i & (7) \\ &= \sigma_d^2 - 2\mathbf{w}_r^T\mathbf{p}_r - 2\mathbf{w}_i^T\mathbf{p}_i + \mathbf{w}_r^T\mathbf{R}_r\mathbf{w}_r \\ &\quad - 2\mathbf{w}_r^T\mathbf{R}_i\mathbf{w}_i + \mathbf{w}_i^T\mathbf{R}_r\mathbf{w}_i & (8) \\ &= \sigma_d^2 - 2\mathbf{w}_r^T\mathbf{p}_r - 2\mathbf{w}_i^T\mathbf{p}_i + \mathbf{w}_r^T\mathbf{R}_r\mathbf{w}_r \\ &\quad + 2\mathbf{w}_i^T\mathbf{R}_i\mathbf{w}_r + \mathbf{w}_i^T\mathbf{R}_r\mathbf{w}_i & (9) \end{aligned}$$

Our aim in going from (7) to (8) & (9) is to isolate the real or imaginary parts as row-vectors in the the inner products, and we have used the fact that since $\mathbf{R}^H = \mathbf{R}$ we have $\mathbf{R}_r^T = \mathbf{R}_r$ and $\mathbf{R}_i^T = -\mathbf{R}_i$. We have

$$\begin{aligned} \nabla\mathbf{w}_r J(\mathbf{w}) &= -2\mathbf{p}_r + 2\mathbf{R}_r\mathbf{w}_r - 2\mathbf{R}_i\mathbf{w}_r & (10) \\ \nabla\mathbf{w}_i J(\mathbf{w}) &= -2\mathbf{p}_i + 2\mathbf{R}_i\mathbf{w}_r + 2\mathbf{R}_r\mathbf{w}_i & (11) \end{aligned}$$

via (8) and (9) respectively. We therefore write

$$\begin{aligned} \nabla\mathbf{w} J(\mathbf{w}) &= \nabla\mathbf{w}_r J(\mathbf{w}) + j\nabla\mathbf{w}_i J(\mathbf{w}) & (12) \\ &= -2\mathbf{p}_r + 2\mathbf{R}_r\mathbf{w}_r - 2\mathbf{R}_i\mathbf{w}_r \\ &\quad j(-2\mathbf{p}_i + 2\mathbf{R}_i\mathbf{w}_r + 2\mathbf{R}_r\mathbf{w}_i) & (13) \\ &= -2(\mathbf{p}_r + j\mathbf{p}_i) + 2\mathbf{R}_r(\mathbf{w}_r + j\mathbf{w}_i) \\ &\quad -2\mathbf{R}_i(j\mathbf{w}_r - \mathbf{w}_i) & (14) \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w} & (15) \end{aligned}$$

This is the answer you might expect and might even know.

2 The LMS Algorithm

Equation (15) might seem to give us the way to update (3). One approach might be to estimate – say, by a block average – the required correlations **p** and **R** and perform exactly that[1]. However, This does, however, require a certain amount of computation overhead in terms of the solution to a set of linear equations; and the block-average idea is not especially reactive to changes. A better idea in terms of the latter would be a "forgetting factor" sort of average. On the surface one is left with the $\mathcal{O}(M^3)$ computational

[1]Of course this would beg the question as to why not simply go directly to the Wiener solution directly via the linear equations.

load, but in fact the exponential average is exactly what we shall see when we discuss the RLS algorithm, and its update will be shown cleverly to be $\mathcal{O}(M^2)$. The LMS update is $\mathcal{O}(M)$, meaning that quite lengthy filters are easily in reach.

We need estimators for $\mathbf{p}$ and $\mathbf{R}$, and LMS espouses the very simplest:

$$\begin{aligned} \hat{\mathbf{p}} &= \mathbf{u}_n d[n]^* && (16) \\ \hat{\mathbf{R}} &= \mathbf{u}_n \mathbf{u}_n^H && (17) \end{aligned}$$

Note that both are (by definition) unbiased estimators. We thus have the LMS update

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}_{n-1} - \mu(-\mathbf{u}_n d[n]^* + \mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}) && (18) \\ &= \mathbf{w}_{n-1} + \mu \mathbf{u}_n (d[n]^* - \mathbf{u}_n^H \mathbf{w}_{n-1}) && (19) \\ &= \mathbf{w}_{n-1} + \mu \mathbf{u}_n e[n]^* && (20) \end{aligned}$$

where we have absorbed the constant 2 into the unspecified μ, and of course we have

$$\hat{d}[n] = \mathbf{w}_{n-1}^H \mathbf{u}_n \qquad (21)$$

as the filter output at time n.

3 LMS Analysis

3.1 Discussion

There are many ways to analyze the LMS algorithm. The way shown in the text is excellent but complicated. I used to teach it, but in later years I've come to the conclusion that it provides inexact but good answers for very restrictive assumptions. In short, it is a lot of effort that provides a very sharp answer that is not especially intuitive, isn't exact nor applies when things are not Gaussian[2]. What we seek is guidance about μ: how large should it be? Clearly a large μ is a concern in that it may "go unstable" and throw $\mathbf{w}_n$ wildly in various directions. A small μ avoids this, and has the additional benefit that the added "gradient noise" in steady state (due to continual vacillations in $\mathbf{w}_n$) can be reduced. But a filter with a small μ may take a long time to converge.

[2]I have seen conference presentations and journal papers that purport to give exact answers but are quite indigestible, both in development and solution.

3.2 Convergence

We begin with

$$\mathbf{w}_n \;=\; \mathbf{w}_{n-1} \;-\; \mu(-\mathbf{u}_n d[n]^* + \mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}) \tag{22}$$

and define $\varepsilon_n \equiv \mathbf{w}_n - \mathbf{R}^{-1}\mathbf{p}$ to be the filter error. We have

$$\begin{aligned}
\varepsilon_n &= \varepsilon_{n-1} \;-\; \mu(-\mathbf{u}_n d[n]^* + \mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}) && (23)\\
\mathcal{E}\{\varepsilon_n\} &= \mathcal{E}\{\varepsilon_{n-1}\} \;-\; \mu(-\mathcal{E}\{\mathbf{u}_n d[n]^*\} + \mathcal{E}\{\mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}\}) && (24)\\
\mathcal{E}\{\varepsilon_n\} &= \mathcal{E}\{\varepsilon_{n-1}\} \;-\; \mu(-\mathbf{p} + \mathcal{E}\{\mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}\}) && (25)
\end{aligned}$$

We claim that we can write

$$\mathcal{E}\{\mathbf{u}_n \mathbf{u}_n^H \mathbf{w}_{n-1}\} \;=\; \mathcal{E}\{\mathbf{u}_n \mathbf{u}_n^H\}\mathcal{E}\{\mathbf{w}_{n-1}\} \tag{26}$$

and offer the justification that whatever dependence there may be between $\mathbf{u}_n$ and $\mathbf{w}_{n-1}$ – and in the case that $\{\mathbf{u}_n\}$ forms an independent sequence there would be none – it is second-order. That is, the changes in $\mathbf{w}_{n-1}$ arising from recent $\mathbf{u}_n$'s are small perturbations around the expected value. As such we claim

$$\begin{aligned}
\mathcal{E}\{\varepsilon_n\} &\approx \mathcal{E}\{\varepsilon_{n-1}\} \;-\; \mu(-\mathbf{p} + \mathbf{R}\mathcal{E}\{\mathbf{w}_{n-1}\}) && (27)\\
&= \mathcal{E}\{\varepsilon_{n-1}\} \;-\; \mu(-\mathbf{p} + \mathbf{R}[\mathcal{E}\{\varepsilon_{n-1}\} - \mathbf{R}^{-1}\mathbf{p}]) && (28)\\
&= \mathcal{E}\{\varepsilon_{n-1}\} \;-\; \mu\mathbf{R}\mathcal{E}\{\varepsilon_{n-1}\} && (29)\\
&= [\mathbf{I} \;-\; \mu\mathbf{R}]\; \mathcal{E}\{\varepsilon_{n-1}\} && (30)
\end{aligned}$$

Hence a necessary condition for convergence (in the mean, and subject to our approximation) is that all eigenvalues of $\mathbf{I} \;-\; \mu\mathbf{R}$ be less than unity in magnitude; and since all eigenvalues of $\mathbf{R}$ are non-negative that means

$$\mu \;<\; \frac{2}{\lambda_{max}} \tag{31}$$

Now, the whole point of LMS is to avoid explicit knowledge of $\mathbf{R}$, much less of its eigenstructure. So a nice way to assure convergence is to note

$$Tr(\mathbf{R}) \;=\; \sum_{i=1}^{M} \lambda_i \tag{32}$$

which means

$$Tr(\mathbf{R}) \;\geq\; \lambda_{max} \tag{33}$$

As such, a reasonable way to assure convergence is to select

$$\mu \;<\; \frac{2}{Tr(\mathbf{R})} \tag{34}$$

We can use

$$\mu \;<\; \frac{2}{Mr[0]} \tag{35}$$

if the process $\{\mathbf{u}_n\}$ is a sliding window on a scalar time series. Please note that this is not the general case at all.

Suggestions about the rate of convergence are also available from (30). Specifically, the slowest eigenmode of $\mathbf{w}_n$ to converge will clearly have rate

$$\rho \;=\; \max\{|1-2\mu\lambda_{min}|, |1-2\mu\lambda_{max}|\} \tag{36}$$

meaning that the error in this mode will converge to zero at rate ρ^n. Fastest (min-max) convergence is obtained when μ is selected such that the two are equal:

$$1-2\mu\lambda_{min} = -(1-2\mu\lambda_{max}) \tag{37}$$

or

$$\mu \;=\; \frac{1}{\lambda_{max}+\lambda_{min}} \tag{38}$$

whereat we would find the slowest rate to be

$$\rho \;=\; \frac{\lambda_{max}-\lambda_{min}}{\lambda_{max}+\lambda_{min}} \tag{39}$$

Note that we want ρ to be as close to zero as possible for quick convergence. If $\lambda_{min} = 0$ this means that we have no convergence at all; but it is hard to interpret that fact since for this zero eigenvalue $\mathbf{q}_{min}$ we have necessarily

$$\mathbf{q}_{min}^H\mathbf{u_n} \;=\; 0 \tag{40}$$

In general errors in $\mathbf{w}_n$ in the subspace coined by the smaller eigenvalues of $\mathbf{R}$ may be large; but they may also have little effect on the filter's performance. See the next subsection.

3.3 Steady-State Error

The aim is to approximate the effect of a "jumpy" $\mathbf{w}_n$ on the error. We need second moments, and we have to do things indirectly. We have of course

$$\mathbf{w}_n \;=\; \mathbf{w}_{n-1} \;+\; \mu\mathbf{u}_n e[n]^* \tag{41}$$

We take the variance:

$$\begin{aligned}\mathcal{E}\{|\mathbf{w}_n|^2\} &= \mathcal{E}\{|\mathbf{w}_{n-1} + \mu\mathbf{u}_n e[n]^*|^2\} && (42)\\ &= \mathcal{E}\{|\mathbf{w}_{n-1}|^2]\} + \mu\mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{u}_n e[n]^*\}\\ &\quad + \mu\mathcal{E}\{\mathbf{u}_n^H\mathbf{w}_{n-1}e[n]\} + \mu^2\mathcal{E}\{\mathbf{u}_n^H\mathbf{u}_n|e[n]|^2\} && (43)\end{aligned}$$

Now, we make the assumption that the filter is converged (no longer transient) so we can assume that

$$\mathcal{E}\{|\mathbf{w}_n|^2\} = \mathcal{E}\{|\mathbf{w}_{n-1}|^2\} \quad (44)$$

and both using this in (43) and expanding for $e[n]$ we have

$$\begin{aligned}0 &= \mu\mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{u}_n(d[n]^* - \mathbf{u}_n^H\mathbf{w}_{n-1})\}\\ &\quad + \mu\mathcal{E}\{(d[n] - \mathbf{w}_{n-1}^H\mathbf{u}_n)\mathbf{u}_n^H\mathbf{w}_{n-1}\} + \mu^2\mathcal{E}\{\mathbf{u}_n^H\mathbf{u}_n|e[n]|^2\} && (45)\end{aligned}$$

where we've also taken advantage of the fact that we can re-order products of things that are scalar. Now, using our previous assumption that $\mathbf{w}_{n-1}$ and $\mathbf{u}_n$ are independent, we have

$$\begin{aligned}0 &\approx \mu\mathcal{E}\{\mathbf{w}_{n-1}\}^H\mathbf{p} - \mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{R}\mathbf{w}_{n-1}\}\\ &\quad + \mu\mathbf{p}^H\mathcal{E}\{\mathbf{w}_{n-1}\} - \mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{R}\mathbf{w}_{n-1}\} + \mu^2\mathcal{E}\{\mathbf{u}_n^H\mathbf{u}_n|e[n]|^2\} && (46)\end{aligned}$$

At convergence it is easy to see that

$$\mathcal{E}\{\mathbf{w}_{n-1}\} = \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} \quad (47)$$

which rewrites (46) as

$$0 = 2\mu\mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} - 2\mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{R}\mathbf{w}_{n-1}\} + \mu^2\mathcal{E}\{\mathbf{u}_n^H\mathbf{u}_n|e[n]|^2\} \quad (48)$$

The tricky step here is to remember the *p.o.o.*: we assume that at convergence $\mathcal{E}\{\mathbf{u}_n e[n]\} = 0$, and hence claim that this implies $\mathbf{u}_n$ and $e[n]$ are independent. As such (48) becomes

$$0 \approx 2\mu\mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} - 2\mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{R}\mathbf{w}_{n-1}\} + \mu^2 Tr(\mathbf{R})\mathcal{E}\{J(\mathbf{w}_{n-1})\} \quad (49)$$

or

$$\mathcal{E}\{\mathbf{w}_{n-1}^H\mathbf{R}\mathbf{w}_{n-1}\} = \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} + \frac{1}{2}\mu Tr(\mathbf{R})\mathcal{E}\{J(\mathbf{w}_{n-1})\} \quad (50)$$

where we have used $J(\mathbf{w}_{n-1}) \equiv |e[n]|^2$.

Now (50) isn't especially illuminating, but it is useful – remember that we said this was indirect. So let us examine the steady-state error directly:

$$
\begin{aligned}
\mathcal{E}\{J(\mathbf{w}_{n-1})\} &= \mathcal{E}\{|e[n]|^2\} && (51)\\
&= \mathcal{E}\{|d[n] - \mathbf{w}_{n-1}^H \mathbf{u}_n|^2\} && (52)\\
&\approx \sigma_d^2 - 2\mathbf{p}^H \mathbf{R}^{-1}\mathbf{p} + \mathcal{E}\{\mathbf{w}_{n-1}^H \mathbf{R}\mathbf{w}_{n-1}\} && (53)\\
&= \sigma_d^2 - \mathbf{p}^H \mathbf{R}^{-1}\mathbf{p} + \frac{1}{2}\mu Tr(\mathbf{R})\mathcal{E}\{J(\mathbf{w}_{n-1})\} && (54)\\
&= \frac{J_{min}}{1 - \frac{1}{2}\mu Tr(\mathbf{R})} && (55)
\end{aligned}
$$

where $J_{min} \equiv J(\mathbf{R}^{-1}\mathbf{p}) = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p}$ is the optimal Wiener filter error. Note that (53) requires the usual approximation that that $\mathbf{w}_{n-1}$ and $\mathbf{u}_n$ are independent and (54) results from insertion of (50). It is interesting to compare (55) to (32): apparently the upper bound on μ causes divergent steady-state error. It is perhaps not surprising to find that the smaller μ the better then steady-state performance.

4 Subspace Tracking

The use of the LMS algorithm in a problem that has easily expressible Wiener terms ($d[n]$, $\mathbf{u}_n$, etc.) is straightforward. This section discusses an especially famous LMS application that is both non-standard and confusingly-named. I find the textbook obscurantist about subspace tracking, hence I'll call out my own understanding of it.

Suppose we wish to design an LMS algorithm to minimize

$$J(\mathbf{w}) \equiv \frac{1}{2}\mathcal{E}\{|\mathbf{w}^H\mathbf{u}_n|^2\} \qquad (56)$$

subject to a constraint

$$\mathbf{w}^H\mathbf{q} = 1 \qquad (57)$$

and the factor $\frac{1}{2}$ in (56) is irrelevant but will make our lives simpler. There is no $d[n]$ here and the presence of a constraint is new; let us see what happens. We will later see this is the MVDR problem in beamforming or spectral estimation; but here we wish to solve it adaptively, whereas later we will use block averages. The idea is that we seek a filter $\mathbf{w}$ that has minimum output power subject to the stricture that it "listens" to a frequency (or direction) $\mathbf{q}$. Put another way, we wish to place nulls (zeros) of the filter where they can do the most good, but not suppress any desired signal at all. It's worth mentioning we could replace (57) by

$$\mathbf{w}^H\mathbf{A} \;=\; \mathbf{b} \tag{58}$$

for some $\mathbf{b}$ (might be all 1's) if that we want to "listen" to several directions or frequencies at the same time. More on that later, for now we'll stay with (57).

We adopt a Wiener approach, and pose this as a Lagrange multiplier optimization:

$$\begin{aligned} J(\mathbf{w}) &= \frac{1}{2}\mathbf{w}^H\mathbf{R}\mathbf{w} \;-\; \lambda(\mathbf{w}^H\mathbf{q} - 1) && (59)\\ \nabla J(\mathbf{w}) &= \mathbf{R}\mathbf{w} \;-\; \lambda\mathbf{q} && (60)\end{aligned}$$

Using the LMS idea we have the update

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}_{n-1} \;-\; \mu\nabla J_{\mathbf{w}}(\mathbf{w}_{n-1}) && (61)\\ &= \mathbf{w}_{n-1} \;-\; \mu\nabla(\mathbf{R}\mathbf{w} \;-\; \lambda\mathbf{q}) && (62)\end{aligned}$$

The LMS idea is to estimate

$$\hat{\mathbf{R}} \;=\; \mathbf{u}_n\mathbf{u}_n^H \tag{63}$$

so we get

$$\mathbf{w}_n \;=\; \mathbf{w}_{n-1} \;-\; \mu(\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1} \;-\; \lambda\mathbf{q}) \tag{64}$$

The *subspace-tracking* idea is to select λ to satisfy (57) at all times n. We get

$$\begin{aligned} (\mathbf{w}_{n-1} \;-\; \mu(\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1} \;-\; \lambda\mathbf{q}))^H\mathbf{q} &= 1 && (65)\\ (\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1})^H\mathbf{q} &= \lambda^*\mathbf{q}^H\mathbf{q} && (66)\\ (\mathbf{w}_{n-1}^H\mathbf{u}_n)(\mathbf{u}_n^H\mathbf{q}) &= \lambda^*\mathbf{q}^H\mathbf{q} && (67)\\ \lambda &= \frac{\mathbf{q}^H\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1}}{\mathbf{q}^H\mathbf{q}} && (68)\end{aligned}$$

where (66) follows because the constraint is assumed satisfied for time $n-1$ and (67) because we can re-order products of scalars.

So now (64) becomes

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}_{n-1} \;-\; \mu(\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1} \;-\; \mathbf{q}\lambda) && (69)\\ &= \mathbf{w}_{n-1} \;-\; \mu(\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1} \;-\; \mathbf{q}\frac{\mathbf{q}^H\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1}}{\mathbf{q}^H\mathbf{q}}) && (70)\\ &= \mathbf{w}_{n-1} \;-\; \mu\left(\mathbf{I} \;-\; \frac{\mathbf{q}\mathbf{q}^H}{\mathbf{q}^H\mathbf{q}}\right)\mathbf{u}_n\mathbf{u}_n^H\mathbf{w}_{n-1} && (71)\\ &= \mathbf{w}_{n-1} \;-\; \mu\mathbf{P}\mathbf{u}_n e[n]^* && (72)\end{aligned}$$

where the error is all the remaining "noise"

$$e[n] \;\equiv\; \mathbf{w}_{n-1}^H \mathbf{u}_n \tag{73}$$

and we have

$$\mathbf{P} \;\equiv\; \mathbf{I} \;-\; \frac{\mathbf{q}\mathbf{q}^H}{\mathbf{q}^H\mathbf{q}} \tag{74}$$

It's easy to see that $\mathbf{P}$ is a *projection* matrix: $\mathbf{P}\mathbf{u}_n$ removes all the $\mathbf{u}_n$ that is parallel to $\mathbf{q}$ (where according to the constraint you want there to be no updating) but leaves leaves all the remaining subspace unchanged. The projection (74) becomes

$$\mathbf{P} \;\equiv\; \mathbf{I} \;-\; \mathbf{A}(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H \tag{75}$$

if (58) is used in place of (57).

ECE 6123
Advanced Signal Processing
Adaptive Filtering with RLS

Peter Willett

Fall 2017

We have seen the LMS algorithm, whose update rule is

$$y[n] \;=\; \hat{d}[n] \;=\; \mathbf{w}_{n-1}^H \mathbf{u}_n \tag{1}$$

The LMS approach is clearly fortuitous in that its computational load is $\mathcal{O}(M)$ per sample. It suffers in that it can converge – and adapt – slowly to changing input statistics. At another extreme is the least-squares approach (perhaps via the SVD) whose update is

$$\mathbf{w} \;=\; (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d} \;=\; (N\hat{\mathbf{R}})^{-1}(N\hat{\mathbf{p}}) \;=\; \hat{\mathbf{R}}^{-1}\hat{\mathbf{p}} \tag{2}$$

where the estimates are formed by *block-averages* of the last input data (both $\{\mathbf{u}_i\}_{i=n-M+1}^n$ and $\{d[i]\}_{i=n-M+1}^n$). Obviously the LS approach wrings as much information as available out of a possibly-abbreviated block of data. It will thus adapt quickly; but in general its computational load is $\mathcal{O}(M^3)$ per sample.

It goes like this. Consider

$$\begin{aligned} \hat{\mathbf{R}}_n &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_n\mathbf{u}_n^H & (3)\\ &= \lambda\hat{\mathbf{R}}_{n-1} + \mathbf{u}_n\mathbf{u}_n^H & (4)\\ \hat{\mathbf{p}}_n &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_n d[n]^* & (5)\\ &= \lambda\hat{\mathbf{p}}_{n-1} + \mathbf{u}_n d[n]^* & (6) \end{aligned}$$

where λ is slightly less than unity. This estimator might be called an exponential average or one with a "forgetting-factor" (that is: λ). It's easy to see

$$\begin{aligned} \mathcal{E}\{\hat{\mathbf{R}}_n\} &= \frac{1}{1-\lambda}\mathbf{R} & (7)\\ \mathcal{E}\{\hat{\mathbf{p}}_n\} &= \frac{1}{1-\lambda}\mathbf{p} & (8) \end{aligned}$$

so

$$\hat{\mathbf{R}}_n^{-1}\hat{\mathbf{p}}_n \;\rightarrow\; \mathbf{R}^{-1}\mathbf{p} \tag{9}$$

Now define

$$\begin{aligned}
\mathbf{P}_n &\equiv \hat{\mathbf{R}}_n^{-1} && (10)\\
&= \left(\lambda\hat{\mathbf{R}}_{n-1} \;+\; \mathbf{u}_n\mathbf{u}_n^H\right)^{-1} && (11)\\
&= \lambda^{-1}\hat{\mathbf{R}}_{n-1}^{-1} \;-\; \frac{\lambda^{-2}\hat{\mathbf{R}}_{n-1}^{-1}\mathbf{u}_n\mathbf{u}_n^H\hat{\mathbf{R}}_{n-1}^{-1}}{1+\lambda^{-1}\mathbf{u}_n^H\hat{\mathbf{R}}_{n-1}^{-1}\mathbf{u}_n} && (12)\\
&= \lambda^{-1}\mathbf{P}_{n-1} \;-\; \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^H\mathbf{P}_{n-1} && (13)
\end{aligned}$$

where

$$\mathbf{k}_n \;\equiv\; \frac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n}{1+\lambda^{-1}\mathbf{u}_n^H\mathbf{P}_{n-1}\mathbf{u}_n} \tag{14}$$

To go from (11) to (12) we use the matrix-inversion lemma[1] which is

$$(\mathbf{A} \;+\; \mathbf{BCD})^{-1} \;=\; \mathbf{A}^{-1} \;-\; \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right)\mathbf{DA}^{-1} \tag{15}$$

Continuing from (14) we have

$$\begin{aligned}
\mathbf{k}_n\left(1+\lambda^{-1}\mathbf{u}_n^H\mathbf{P}_{n-1}\mathbf{u}_n\right) &= \lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n && (16)\\
\mathbf{k}_n &= \lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n \;-\; \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^H\mathbf{P}_{n-1}\mathbf{u}_n && (17)\\
&= \left(\lambda^{-1}\mathbf{P}_{n-1} \;-\; \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^H\mathbf{P}_{n-1}\right)\mathbf{u}_n && (18)\\
\mathbf{k}_n &= \mathbf{P}_n\mathbf{u}_n && (19)
\end{aligned}$$

where (19) follows from (18) by insertion of (13). Now we have

$$\begin{aligned}
\mathbf{w}_n &= \hat{\mathbf{R}}_n^{-1}\hat{\mathbf{p}}_n && (20)\\
&= \mathbf{P}_n\left(\lambda\hat{\mathbf{p}}_{n-1} \;+\; \mathbf{u}_n d[n]^*\right) && (21)\\
&= \lambda\left(\lambda^{-1}\mathbf{P}_{n-1} \;-\; \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^H\mathbf{P}_{n-1}\right)\hat{\mathbf{p}}_{n-1} \;+\; \mathbf{P}_n\mathbf{u}_n d[n]^* && (22)\\
&= \mathbf{w}_{n-1} \;-\; \mathbf{k}_n\mathbf{u}_n^H\mathbf{w}_{n-1} \;+\; \mathbf{k}_n d[n]^* && (23)\\
&= \mathbf{w}_{n-1} \;+\; \mathbf{k}_n\alpha[n]^* && (24)
\end{aligned}$$

where

$$\alpha[n] \;\equiv\; d[n] \;-\; \mathbf{w}_{n-1}^H\mathbf{u}_n \tag{25}$$

is a twist on the "error" $e[n] = d[n] - \mathbf{w}_n^H\mathbf{u}_n$.

[1]This is also known as the Woodbury formula.

The RLS, which is rather clever, iterates according to

1. According to (24): $\alpha[n] = d[n] - \mathbf{w}_{n-1}^H \mathbf{u}_n$.
2. According to (14): $\mathbf{k}_n = \frac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n}{1+\lambda^{-1}\mathbf{u}_n^H \mathbf{P}_{n-1}\mathbf{u}_n}$.
3. According to (13): $\mathbf{P}_n = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^H\mathbf{P}_{n-1}$.
4. According to (25): $\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n\alpha[n]^*$.

If one is interested in the output, one can also compute $y[n] = \mathbf{w}_n^H \mathbf{u}_n$. This may seem like a silly statement, but in some applications only the filter's form may be of interest; and unlike the LMS that requires $e[n]$ for its update, RLS does not explicitly need $e[n]$, only $\alpha[n]$. Note that steps #2 and #3 each are $\mathcal{O}(M^2)$ while steps #1 and #4 need only $\mathcal{O}(M)$.

ECE 6123
Advanced Signal Processing
Spectral Estimation

Peter Willett

Fall 2017

1 Basics of Spectral Estimation

1.1 Introduction

We are all familiar with the discrete-time Fourier transform (DTFT) and the discrete Fourier transform (DFT) – the latter being implemented efficiently via the fast Fourier transform (FFT). The former useful for analyzing deterministic signals; the latter is more practical, and gives a way to understand the frequency behavior of a signal that is given to you as a time series, one that may not have an explicit expression that nicely sums to something compact or conversely whose DTF is amenable to integration. But what does it *mean* when we take the FFT of a random signal? Here we will explore this; we will when necessary assume the signal $\{x[n]\}_{n=0}^{N-1}$ is *wss*, zero mean and Guassian[1].

We begin this section by discussing the periodogram, which is the most obvious approach to spectral estimation: it has a big problem, which we will solve later. We continue with a discussion of the meaning of resolution. We then establish the relationship between spectral estimation and beamforming – it turns out that much of what we do can be used for array signal processing provided the source is monochromatic (or can be made to be so by filtering) and the array is a uniformly-spaced linear array (ULA).

The following sections deal with nonparametric and parametric spectral estimation. As the name implies, non-parametric spectral estimation makes no assumptions about the nature of the spectrum, and we look at the Bartlett, Welch and Capon approaches. Parametric methods do make such an assumption, and the ones we explore here are based on AR models and on modeling as sinusoids-plus-noise.

[1]This is only important when we are discussing the periodogram, so explore its consistency. We will assume in that section that $x[n] \in \Re$ for ease of explanation; the complex case is the same but notationally more difficult

1.2 The Periodogram

Recall that the power spectrum of a random process $\{x[n]\}$ is defined as

$$S(\omega) \equiv \sum_{k=-\infty}^{\infty} r[k]e^{-j\omega k} \tag{1}$$

where $\{r[k]\}$ is the (usual) autocorrelation $r[k] = \mathcal{E}\{x[n]x[n-k]^*\}$. How about we estimate it from our data $\{x[n]\}_{n=0}^{N-1}$ as

$$\hat{S}(\omega) \equiv \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j\omega n}\right|^2 \tag{2}$$

that is, as the DTFT magnitude square and suitably[2] scaled? Note that the periodogram is efficiently computed as

$$\hat{S}(\omega)\big|_{\omega=\frac{2\pi k}{N}} = \frac{1}{N}|X(k)|^2 \tag{3}$$

where $X(k)$ is the k^{th} DFT (or FFT) output.

We need some statistical analysis of the periodogram. We begin with the mean:

$$\begin{aligned}
\mathcal{E}\{\hat{S}(\omega)\} &= \frac{1}{N}\mathcal{E}\left\{\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x[n]x[m]^* e^{-j\omega(n-m)}\right\} && (4)\\
&= \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r[n-m]e^{-j\omega(n-m)} && (5)\\
&= \frac{1}{N}\sum_{k=-(N-1)}^{N-1} (N-|k|)r[k]e^{-j\omega k} && (6)\\
&= S(\omega)\star\mathcal{F}\left[1-\frac{|k|}{N}\right] && (7)\\
&= S(\omega)\star W_B(\omega) && (8)
\end{aligned}$$

where $W_B(\omega)$ is the DTFT of the (triangular) Bartlett window $w_B[k]$:

$$\begin{aligned}
W_B(\omega) &= \mathcal{F}\left[1-\frac{|k|}{N}\right] && (9)\\
&= \mathcal{F}\left[w_B[k]\right] && (10)\\
&= \left(\frac{1}{N}\frac{\sin(\omega N/2)}{\sin(\omega/2)}\right)^2 && (11)
\end{aligned}$$

[2]We will soon see why the scaling.

That is, the expected value of the periodogram is a smoothed version of the true power spectrum: it gets convolved with the sinc-squared.

Turning now to the variance, we compute the second moment. We need here to – briefly – assume[3] that $\{x[n]\}$ is real and Gaussian. We use the fact that for jointly-Gaussian zero-mean random variables we have

$$\mathcal{E}\{ABCD\} \;=\; \mathcal{E}\{AB\}\mathcal{E}\{CD\} \;+\; \mathcal{E}\{AC\}\mathcal{E}\{BD\} \;+\; \mathcal{E}\{AD\}\mathcal{E}\{BC\} \tag{12}$$

We get

$$\begin{aligned}
&\mathcal{E}\{(\hat{S}(\omega))^2\} \\
&\quad= \frac{1}{N^2}\mathcal{E}\left\{\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} x[n]x[m]x[p]x[q]e^{-j\omega(n-m+p-q)}\right\} && (13)\\
&\quad= \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} r[n-m]r[p-q]e^{-j\omega(n-m+p-q)} \\
&\qquad + \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} r[n-q]r[m-p]e^{-j\omega(n-m+p-q)} \\
&\qquad + \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\sum_{p=0}^{N-1}\sum_{q=0}^{N-1} r[n-p]r[m-q]e^{-j\omega(n-m+p-q)} && (14)\\
&\quad= 2|S_1(\omega)|^2 \;+\; |S_2(\omega)|^2 && (15)
\end{aligned}$$

where

$$S_1(\omega) \;\equiv\; \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r[n-m]e^{-j\omega(n-m)} \tag{16}$$

$$S_2(\omega) \;\equiv\; \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r[n-m]e^{-j\omega(n+m)} \tag{17}$$

Comparing (16) to (6) we see from (8) that

$$S_1(\omega) \;=\; S(\omega) \star W_B(\omega) \tag{18}$$

On the other hand, we have

$$S_2(\omega) \;=\; \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r[n-m]e^{-j\omega(n+m)} \tag{19}$$

[3]The Gaussian assumption is important. That of being real simplifies the notation.

$$= \frac{1}{N} \sum_{k=-(N-1)}^{N-1} \sum_{m=|k|}^{N-1-|k|} r[k] e^{-j\omega(k+2m)} \tag{20}$$

$$\begin{aligned} &= \left(\frac{1}{N} \sum_{k=-(N-1)}^{-1} r[k] e^{-j\omega k} \left(\sum_{m=-k}^{N-1} e^{-j\omega 2m} \right) \right) \\ &\quad + \left(\frac{1}{N} \sum_{k=0}^{N-1} r[k] e^{-j\omega k} \left(\sum_{m=0}^{N-1-k} e^{-j\omega 2m} \right) \right) \end{aligned} \tag{21}$$

$$\tag{22}$$

As $N \to \infty$ the inner sums do not converge but are bounded – the bound does not grow with N – say, bounded in magnitude by C. We could therefore[4] write

$$|S_2(\omega)| < C \left| \frac{1}{N} \sum_{k=-(N-1)}^{N-1} r[k] e^{-j\omega k} \right| \tag{23}$$

and since the sum converges to the power spectrum, the term $S_2(\omega)$ is asymptotically zero. As such

$$\mathcal{E}\{(\hat{S}(\omega))^2\} = 2\,(S(\omega) \star W_B(\omega))^2 \tag{24}$$

$$\text{Var}\{\hat{S}(\omega)\} = \mathcal{E}\{(\hat{S}(\omega))^2\} - \left(\mathcal{E}\{\hat{S}(\omega)\}\right)^2 \tag{25}$$

$$= (S(\omega) \star W_B(\omega))^2 \tag{26}$$

which leaves us the important message that *the periodogram is not consistent* – its variance does not decrease to zero as $N \to \infty$.

1.3 Rayleigh Resolution Limit

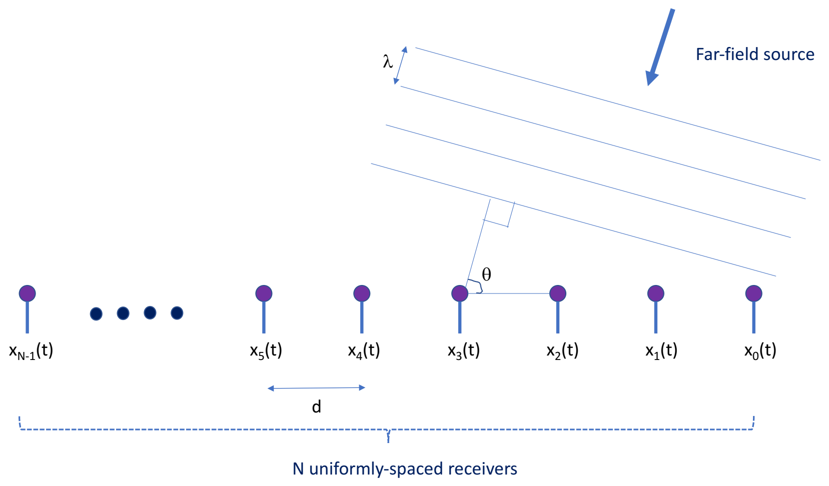
With reference to (8) and (11), two frequencies may appear, after convolution with $W_B(\omega)$, as a single spectral "bump". When this happens we say that the frequencies are not resolvable in the classical (periodogram-based) sense. Normally it is assumed such a merging happens when the two frequencies are closer together than the frequency spacing between the peak and first zero of $W_B(\omega)$, or $\frac{2\pi}{M}$. We call this the *Rayleigh resolution limit.*

1.4 Array Signal Processing

Consider a uniform linear array of sensors: microphones, hydrophones, radar receivers, etc. The uniform spacing is important for what follows here; but

[4]The argument could be made more precise, since for finite k both the inner sums converge to $\delta(\omega)$.

planar uniform arrays apply as well with greater complexity of notation. We require a far-field and monochromatic (single-frequency) source. Far-field means the wavefronts when they arrive at the sensor are planar (as opposed to curved). The monochromatic nature is important for the mathematics, but in fact one could assume that an FFT operation is occurring at the sensors, and the operations about to be described can be performed separately at each frequency and (possibly) combined. The notional setup is as pictured below.



The source is oriented at angle θ with respect to "horizontal" of the array – some people prefer to have θ with respect to broadside, the difference is will be that cos gets replaced by sin. Now suppose the source emits frequency f – the wavelength and speed of propagation are related to it as $f\lambda = c$. The signal received at the n^{th} sensor is

$$x_n(t) \;=\; A' e^{j2\pi f(t - nd\cos(\theta)/c)} \tag{27}$$

where A' is a complex amplitude and d is the inter-sensor spacing. If all sensors sample at the same time, we could write

$$\begin{aligned} x[n] &= Ae^{-j2\pi f nd\cos(\theta)/c} && (28)\\ &= Ae^{-j2\pi n\left(\frac{d}{\lambda}\right)\cos(\theta)} && (29)\end{aligned}$$

where we no longer need the time index and we've absorbed the phase caused by the sampling time into A. What is remarkable is that the signal now appears as a (spatial) sinusoid indexed by sensor number as opposed to time sample, and

$$\kappa \;=\; 2\pi\left(\frac{d}{\lambda}\right)\cos(\theta) \tag{30}$$

is the (spatial) frequency. An immediate consequence of this is that to avoid aliasing we need to have

$$2\pi\left(\frac{d}{\lambda}\right)\cos(\theta) \;<\; \pi \tag{31}$$

$$d\cos(\theta) \;<\; \frac{\lambda}{2} \tag{32}$$

and since $\cos(\theta) \leq 1$ this means that we must have

$$d \;<\; \frac{\lambda}{2} \tag{33}$$

in order to be sure there be no spatial aliasing at all.

Perhaps most interesting is that we see that we can apply our spectral estimation methods to the array processing problem: once we have the spatial frequency of the "sinusoid" we invert (30) to get the direction of arrival (DOA). The wrinkle is Rayleigh resolution, for which the limit is

$$2\pi\left(\frac{d}{\lambda}\right)\cos(\theta+\Delta) - 2\pi\left(\frac{d}{\lambda}\right)\cos(\theta) \;>\; \frac{2\pi}{M} \tag{34}$$

or with Δ small (and M sufficiently large),

$$\Delta \;>\; \frac{\lambda}{Md\sin(\theta)} \tag{35}$$

approximately. This gives an upper limit on Rayleigh resolution of two DOA's (we hope to do better!). One thing that is very noticeable is the deterioration of resolvability near "endfire" – when θ is close to 0 or π.

2 Nonparametric Spectral Estimation: The Bartlett and Welch Procedures

Inconsistency would seem to be a "deal-killer" for any estimator. But there is an easy fix. For data record $\{x[n]\}_{n=0}^{N-1}$, and assuming that $N = LM$, write

$$\hat{S}_i(\omega) \;\equiv\; \frac{1}{M}\left|\sum_{n=0}^{M-1} x[n+iM]e^{-j\omega n}\right|^2 \tag{36}$$

for $i = 0,\ ,1,\ \ldots,\ L-1$ and form the Bartlett spectral estimator as

$$\hat{S}_B(\omega) \;=\; \frac{1}{L}\sum_{i=0}^{L-1} S_i(\omega) \tag{37}$$

It is easy to see that

$$\mathrm{Var}\{\hat{S}_B(\omega)\} \;\approx\; \frac{1}{L}\left(S(\omega)\star W_B(\omega)\right)^2 \tag{38}$$

which indicates[5] that the Bartlett periodogram is indeed consistent. The price paid is that in this case

$$W_B(\omega) \;=\; \left(\frac{1}{M}\frac{\sin(\omega M/2)}{\sin(\omega/2)}\right)^2 \tag{39}$$

where (8) describes the mean of the Bartlett periodogram. Note that (39) does not change as N increases: the Bartlett periodogram converges, but converges to a smeared version of the power spectrum,

The Welch method somewhat improves on Bartlett in two ways: by allowing overlap (and hence better resolution due to a larger M in (39)) and by introducing windowing that can potentially reduce sidelobes (and hence eliminate interference of distant "loud" tones on quieter ones that the periodogram may be trying hard to discern). In the Welch approach we no longer require that $LM = N$, but continue with M as the length of the sections and L as the number of sections; call K the number of samples to jump between sections. We replace (36) by

$$\hat{S}_i(\omega) \;\equiv\; \frac{1}{M}\left|\sum_{n=0}^{M-1} w[n]x[n+iK]e^{-j\omega n}\right|^2 \tag{40}$$

where $w[n]$ is the window used, and the Welch periodogram $S_W(\omega)$ is formed from these exactly as $S_B(\omega)$ is in (37). Now we have

$$\mathcal{E}\{\hat{S}(\omega)\} \;=\; S(\omega)\star W(\omega) \tag{41}$$

where

$$W(\omega) \;\equiv\; \frac{1}{M}\left|\sum_{n=0}^{M-1} w[n]e^{-j\omega n}\right|^2 \tag{42}$$

Some careful analysis has shown that some degree of overlap is not too harmful: with 50% overlap (38) is increased by a factor $\frac{9}{8}$, approximately.

[5]The approximation is that the limited dependency between M-blocks of data has been ignored. Statisticians would invoke a "mixing" condition.

3 Nonparametric Spectral Estimation: MVDR

This approach, sometimes known as the MVDR (for "minimum-variance distortionless response" which is nicely descriptive) and sometimes as the Capon method (which is less so) is an excellent way to "listen" to weak frequencies (or directions) without fear of interference from other stronger ones. In fact, the stronger these interferers are, the less problems they cause. The idea is to form a "filter"

$$y_\omega[n] \;=\; \mathbf{w}(\omega)^H \mathbf{x}_n \tag{43}$$

whose output[6] represents what is present at frequency ω: the expected output $y[n]$ should contain what is at frequency ω and as little else as possible, and the expected output power is the power at that frequency. The MVDR idea is to select $\mathbf{w}(\omega)$ such that

$$\mathbf{w}(\omega) \;=\; \arg\min_{\mathbf{w}} \left\{ \mathcal{E}\{|y_\omega[n]|^2\} \right\} \text{ subject to } \mathbf{w}(\omega)^H \mathbf{q}(\omega) = 1 \tag{44}$$

where

$$\mathbf{q}(\omega) \;\equiv\; \begin{pmatrix} 1 \\ e^{-j\omega} \\ e^{-j2\omega} \\ \vdots \\ e^{-j(M-1)\omega} \end{pmatrix} \tag{45}$$

is a sinusoid (vector) at frequency ω. Notionally, then we want to listen faithfully to frequency ω (the constraint); but we want to minimize all interference (the minimization). If there is a strong frequency component at frequency ω' it is reasonable to expect the minimization to place a zero accordingly:

$$W_\omega(z)|_{z=e^{j\omega'}} \;\equiv\; \sum_{k=0}^{M-1} w_\omega[k]^* z^{-k}|_{z=e^{j\omega'}} \;=\; \mathbf{w}(\omega)^H \mathbf{q}(\omega') \;\approx\; 0 \tag{46}$$

At any rate, we have

$$\mathcal{E}\{|y_\omega[n]|^2\} \;=\; \mathbf{w}(\omega)^H \mathbf{R} \mathbf{w}(\omega) \tag{47}$$

[6]It is interesting to consider this in light of the interpretation of spectral estimation applied to array processing: one can actually *listen* in a particular direction (spatial frequency) by forming these $y[n]$'s at all (temporal) frequencies and then constructing the time-series coming from that by in the inverse DFT. The filter to be used to do this will appear shortly as (50).

and we solve the minimization via Lagrange multipliers as

$$\mathbf{R}\mathbf{w}(\omega) = \lambda \mathbf{q}(\omega) \tag{48}$$

Substituting back we have

$$\lambda = \frac{1}{\mathbf{q}(\omega)^H \mathbf{R}^{-1} \mathbf{q}(\omega)} \tag{49}$$

which gives us

$$\mathbf{w}(\omega) = \frac{\mathbf{R}^{-1}\mathbf{q}(\omega)}{\mathbf{q}(\omega)^H \mathbf{R}^{-1} \mathbf{q}(\omega)} \tag{50}$$

and hence

$$\begin{aligned}
\hat{S}_{mvdr}(\omega) &= \mathcal{E}\{|y[n]|^2\} && (51)\\
&= \frac{\mathbf{q}(\omega)^H \mathbf{R}^{-1}\mathbf{R}\mathbf{R}^{-1}\mathbf{q}(\omega)}{(\mathbf{q}(\omega)^H \mathbf{R}^{-1}\mathbf{q}(\omega))^2} && (52)\\
&= \frac{1}{\mathbf{q}(\omega)^H \mathbf{R}^{-1}\mathbf{q}(\omega)} && (53)
\end{aligned}$$

4 Parametric Spectral Estimation: AR Modeling

4.1 The Yule-Walker Approach

This is rather obvious, given what we have seen before. Assume that we have estimated autocorrelations $\{r[k]\}_{k=0}^{M}$. We solve the augmented Yule-Walker equations and have thence

$$\hat{S}_{yw}(\omega) = \frac{P_M}{|1 + \sum_{k=1}^{M} a_k^* e^{-jk\omega}|^2} \tag{54}$$

where P_M is the same as σ_ν^2 as seen before. Levinson-Durbin will simplify the solution to the YW equations.

4.2 Maximum Entropy Spectral Estimation

It is an interesting fact that the YW spectral estimate is the *maximum-entropy* spectral estimator of the spectrum given knowledge of the M autocorrelations $\{r[k]\}_{k=0}^{M}$. That is, amongst all the (wss) random processes that have $\{r[k]\}_{k=0}^{M}$ as their first $M+1$ values, the M^{th}-order AR model is the "most random" in the sense of Shannon's entropy – it is "better" than any other AR order, or ARMA or MA or sinusoid-plus-noise (etc.) model in this sense. This course does not pre-suppose any familiarity with information theory, so we won't prove this.

The intuition is that the entropy (disorder) of a *wss* random process is related to the variance of the prediction error. Suppose we knew $\{r[k]\}_{k=0}^{M}$, and our prediction error power was σ_M^2. Now instead suppose we know more: we know $\{r[k]\}_{k=0}^{N}$, where $N > M$. It is tautologically true that we have $\sigma_N^2 \leq \sigma_M^2$, meaning knowing more autocorrelations must help in reducing entropy. The only situation in which it does not help (i.e., $\sigma_N^2 = \sigma_M^2$ for $N > M$) is when the process is AR of order M, since in that case the coefficients used to predict $u[n]$ and multiply $\{u[n-M-1], u[n-M-2], \ldots\}$ are all zero. Hence the AR process is maximally unpredictable amongst all *wss* random processes for which $\{r[k]\}_{k=0}^{M}$ are known.

4.3 Relationship to MVDR

Note that we have

$$\hat{S}_{mvdr}(\omega) = \frac{1}{\mathbf{q}(\omega)^H \mathbf{R}^{-1} \mathbf{q}(\omega)} \tag{55}$$

Now from our earlier work we know that

$$\mathbf{R}^{-1} = \mathbf{L}^H \mathbf{D}^{-1} \mathbf{L} \tag{56}$$

where

$$\mathbf{D} = \begin{pmatrix} P_0 & 0 & 0 & \ldots & 0 \\ 0 & P_1 & 0 & \ldots & 0 \\ 0 & 0 & P_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & P_M \end{pmatrix} \tag{57}$$

in which P_i is the i^{th}-order prediction error and

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ a_{1,1} & 1 & 0 & \ldots & 0 \\ a_{2,2} & a_{2,1} & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{M,M} & a_{M,M-1} & a_{M,M-2} & \ldots & 1 \end{pmatrix} \tag{58}$$

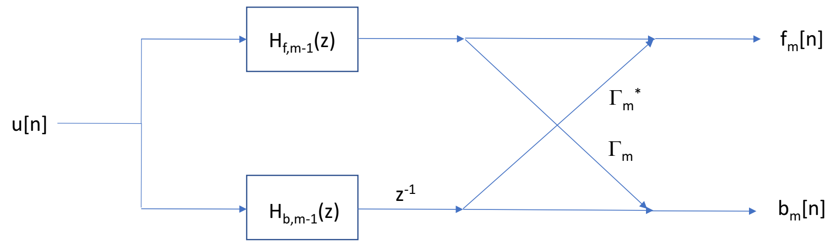
is a matrix of AR predictors. So we are able to write

$$\hat{S}_{mvdr}(\omega) = \left(\sum_{m=0}^{M} \frac{1}{P_m \hat{S}_{yw,m}(\omega)} \right)^{-1} \tag{59}$$

where $\hat{S}_{yw,m}(\omega)$ is the m^{th}-order YW spectral estimate. The MVDR spectral estimate is consequently the "parallel resistors"-weighted sum of YW spectra.

4.4 The Burg Algorithm

The YW spectral estimation approach has two steps: first estimate the correlations, then insert these to YW, presumably efficiently solved via Levinson-Durbin. The Burg approach begins from an earlier place: it assumes only that a record of data is available. There is no need to estimate correlations, Burg estimates the spectrum directly. Now, below is repeated the *lattice* interpretation of the m^{th}-order forward- and backward-error prediction filters (PEFs) from the section on linear prediction that we enjoyed earlier.



Let us suppose, as in the figure, that we have $\{f_{m-1}[n]\}$ & $\{b_{m-1}[n]\}$; that is, we are trying to find the m^{th}-order model and have worked from model order 1, then 2, all the way up to $m-1$. The notion is that we choose Γ_m to minimize the prediction error.

Let us recall from an earlier section of the course that if we posed

$$J(\mathbf{w}) = \sigma_d^2 - 2\Re\{\mathbf{w}^H\mathbf{p}\} + \mathbf{w}^H\mathbf{R}\mathbf{w} \tag{60}$$

then we could write

$$\nabla\mathbf{w}J(\mathbf{w}) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w} \tag{61}$$

which is a nice reference – complex derivatives / gradients are sometimes hard to remember. We also have

$$J(w) = \sigma_d^2 - 2\Re\{w^*p\} + |w|^2R \tag{62}$$

$$\frac{dJ(w)}{dw} = -2p + 2Rw \tag{63}$$

when these are particularized to scalars – apologies that this is belabored.

Suppose we want to minimize the m^{th}-order forward prediction error

$$\mathcal{E}\{|f_m[n]|^2\} = \mathcal{E}\{|f_{m-1}[n] + \Gamma^* b_{m-1}[n-1]|^2\} \tag{64}$$

We take the gradient and get

$$\begin{aligned} 0 &= \nabla_\Gamma \left(\mathcal{E}\{|f_{m-1}[n] + \Gamma_m^* b_{m-1}[n-1]|^2\} \right) && (65) \\ &= 2\mathcal{E}\{b_{m-1}[n-1] f_{m-1}[n]^*\} \;+\; 2\Gamma_m \mathcal{E}\{|b_{m-1}[n-1]|^2\} && (66) \end{aligned}$$

or we get the minimizing reflection coefficient

$$\Gamma_m \;=\; \frac{-\mathcal{E}\{b_{m-1}[n-1] f_{m-1}[n]^*\}}{\mathcal{E}\{|b_{m-1}[n-1]|^2\}} \tag{67}$$

Now it is also interesting to minimize

$$\mathcal{E}\{|b_m[n]|^2\} \;=\; \mathcal{E}\{|b_{m-1}[n-1] + \Gamma f_{m-1}[n]|^2\} \tag{68}$$

We take the gradient and get

$$\begin{aligned} 0 &= \nabla_\Gamma \left(\mathcal{E}\{|b_{m-1}[n-1] + \Gamma_m f_{m-1}[n]|^2\} \right) && (69) \\ &= \nabla_\Gamma \left(\mathcal{E}\{|b_{m-1}^*[n-1] + \Gamma_m^* f_{m-1}[n]^*|^2\} \right) && (70) \\ &= 2\mathcal{E}\{b_{m-1}[n-1] f_{m-1}[n]^*\} \;+\; 2\Gamma_m^* \mathcal{E}\{|f_{m-1}[n]|^2\} && (71) \end{aligned}$$

and we now get the minimizing reflection coefficient

$$\Gamma_m \;=\; \frac{-\mathcal{E}\{b_{m-1}[n-1] f_{m-1}[n]^*\}}{\mathcal{E}\{|f_{m-1}[n]|^2\}} \tag{72}$$

The symmetry is pleasing between the two; but it is perhaps strange to have b and f treated differently. So the Burg approach is actually to minimize

$$\mathcal{E}\{|f_m[n]|^2\} \;+\; \mathcal{E}\{|b_m[n]|^2\} \tag{73}$$

and the solution is easily seen to be

$$\Gamma_m \;=\; \frac{-2\mathcal{E}\{b_{m-1}[n-1] f_{m-1}[n]^*\}}{\mathcal{E}\{|b_{m-1}[n-1]|^2\} + \mathcal{E}\{|f_{m-1}[n]|^2\}} \tag{74}$$

The Burg spectral estimate $\hat{S}_{burg}(\omega)$ is the AR spectrum (like (54)) that uses the Γ_m's as its reflection coefficients. Levinson-Durbin offers an easy way to transform these into AR parameters (the a's), and P_M is directly estimable from $\mathcal{E}\{|f_m[n]|^2$, and the expectations necessary to calculate Γ_m are estimated from $f_{m-1}[n]$ and $b_{m-1}[n]$. Burg offers a slick way to build up the AR model step by step directly from the data. There is some evidence that the Burg spectrum is more "peaky" than the YW spectrum (i.e., sinusoids stand out more clearly). This may have to do with the fact that its zeros have to be inside the unit circle (since mathematically $|\Gamma_m| \leq 1$) whereas with estimated $r[k]$'s this may not be true for the YW estimator[7].

[7]The idea is that zeros that "want" to get arbitrarily close to the unit circle but can't "escape" it can do so with Burg; whereas with YW they can escape and become less close to the unit circle.

5 Parametric Spectral Estimation: Sinusoids in White Noise

5.1 Justification of the Sinusoid Model

Let us begin with an arbitrary (Toeplitz) correlation matrix $\mathbf{R}$, and define

$$\tilde{\mathbf{R}} \equiv \mathbf{R} - \lambda_{min}\mathbf{I} \tag{75}$$

It is clear that $\tilde{\mathbf{R}}$ shares the same eigenvectors as $\mathbf{R}$, while each of its eigenvalues is reduced by λ_{min}. There is at least one zero eigenvalue, and let us call the associated eigenvector $\mathbf{g}$. We have

$$\begin{aligned}
0 &= \mathbf{g}^H\tilde{\mathbf{R}}\mathbf{g} && (76)\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} g[m]^* g[n] r[m-n] && (77)\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} g[m]^* g[n] \frac{1}{2\pi}\int_{-\pi}^{\pi}\tilde{S}(\omega)e^{j\omega(m-n)}d\omega && (78)\\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi}\tilde{S}(\omega)\sum_{m=0}^{M-1}\sum_{n=0}^{M-1} g[m]^* g[n] e^{j\omega(m-n)}d\omega && (79)\\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi}\tilde{S}(\omega)|G(\omega)|^2 d\omega && (80)
\end{aligned}$$

We have blithely and obviously defined

$$\mathbf{g} \equiv \begin{pmatrix} g[0] \\ g[1] \\ g[2] \\ \vdots \\ g[M-1] \end{pmatrix} \tag{81}$$

$$G(z) \equiv \sum_{m=0}^{M-1} g[m] z^{-k} \tag{82}$$

$$G(\omega) \equiv \sum_{m=0}^{M-1} g[m] e^{-j\omega m} \tag{83}$$

$$S(\omega) \equiv \sum_{k=-\infty}^{\infty} r[k] e^{-j\omega k} \tag{84}$$

$$\tilde{S}(\omega) \equiv \sum_{k=-\infty}^{\infty} [r[k] - \lambda_{min}\delta[k]] e^{-j\omega k} \tag{85}$$

where $\delta[k]$ is the unit impulse in the DSP sense. It is important to note that no claim is made that $S(\omega)$ be the actual power spectral density; in fact, it is only *one of* the power spectra whose first M autocorrelations match those of the true random process.

Now from (82) it is seen that $G(z)$ is a polynomial of order $M-1$, and hence it has $M-1$ roots (zeros). Some (or all) of these may be on the unit circle, so (83) can hence be zero for at most $M-1$ values. And since $|G(\omega)|^2 \geq 0$ (80) makes it clear that we have

$$\tilde{S}(\omega)|G(\omega)|^2 \;=\; 0 \quad \forall\omega \tag{86}$$

which tells us that $\tilde{S}(\omega)$ can be non-zero at only those ω's for which $G(\omega) = 0$. There are only at most $M-1$ such ω's and hence we know that we can write

$$\tilde{S}(\omega) \;=\; \sum_{n=1}^{M-1} p_n\delta(\omega-\omega_n) \tag{87}$$

$$S(\omega) \;=\; \sigma^2 + \sum_{n=1}^{M-1} p_n\delta(\omega-\omega_n) \tag{88}$$

where the p_k's are nonnegative real numbers (some can be zero), and hence

$$r[k] \;=\; \sigma^2\delta[k] + \sum_{n=1}^{M-1} p_n e^{j\omega_n k} \tag{89}$$

This (89) tells us a remarkable thing: the first M correlations of any *wss* random process can be written as the sum of a δ-function and $M-1$ complex sinusoids. Put another way – and a bit more notionally – any random process can be thought of as arising from sinusoids plus white noise. This is a backdoor proof of the Caratheodory Theorem. Note that none of this is meant to imply that *all* power spectra have the form (88); what is shown is that for any *wss* random process for which we know the fist M autocorrelations $\{r[k]\}_{k=0}^{M-1}$ there *exists* a random process consistent with those autocorrelations that has form (88). This is perhaps a statement that is parallel to that relating to AR processes: there are many *wss* random processes that have $\{r[k]\}_{k=0}^{M-1}$, but amongst them the one with maximum entropy is the AR process of order $M-1$.

We end by proffering

$$\mathbf{R} \;=\; \sigma^2\mathbf{I} + \sum_{n=1}^{M-1} p_n\mathbf{q}(\omega_n)\mathbf{q}(\omega_n)^H \tag{90}$$

in which $\mathbf{q}(\omega)$ is as in (45), as a general model the correlation matrix of a *wss* random process. Note that there is no reason to expect that the ω_n's are related either to each other or to the "DFT frequencies" – actually, their values are what need to be sought; and to be general we should allow some (or all) p_n's to be zero.

5.2 Pisarenko Harmonic Decomposition

The discussion in the previous section tells us that the eigendecomposition of $\mathbf{R}$ is key, and suggests the following prescription.

1. Estimate $\mathbf{R}$.
2. Find the minimum eigenvalue of $\mathbf{R}$: λ_{min}. We know that σ^2 in (90) is λ_{min}.
3. Find the eigenvector $\mathbf{g}$ that corresponds to λ_{min}.
4. Find the roots of $G(z)$ (see (90)).
5. Keep those roots that on the unit circle[8] and label them $z_m = e^{j\omega_m}$.
6. Solve the Vandermonde system

$$\begin{pmatrix} r[1] \\ r[2] \\ \vdots \\ r[M-1] \end{pmatrix} = \begin{pmatrix} e^{j\omega_1} & e^{j\omega_2} & \cdots & e^{j\omega_{M-1}} \\ e^{j2\omega_1} & e^{j2\omega_2} & \cdots & e^{j2\omega_{M-1}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j(M-1)\omega_1} & e^{j(M-1)\omega_2} & \cdots & e^{j(M-1)\omega_{M-1}} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{M-1} \end{pmatrix} \tag{91}$$

This looks great. And unfortunately it doesn't work very well. The problem is in steps (1) & (3): when a correlation matrix is *estimated* rather than analytically given, the eigenvector polynomial's roots are not especially inclined to be on the unit circle. Notionally, the concern is that essentially all the estimation hard work is performed by the eigenvector corresponding to the *minimum* eigenvalue; and exactly this eigenvalue is by its nature the least well estimated.

[8]In any sort of "practice" roots that are close to the unit circle will do.

5.3 MUSIC

First, this has nothing to do with horns and violins. It stands for **<u>mu</u>**ltiple **<u>si</u>**gnal **<u>c</u>**lassification. Let us work with the ideas from the Pisarenko analysis. First, let us assume that we have

$$\mathbf{R} \;=\; \sigma^2\mathbf{I} + \sum_{n=1}^{L} p_n \mathbf{q}(\omega_n)\mathbf{q}(\omega_n)^H \tag{92}$$

where the only difference from (90) is that in (92) the signal is assumed to contain $L < M-1$ sinusoids. That implies that the multiplicity of the minimum eigenvalue (i.e., σ^2) is $M-L > 1$. This is useful, since with a larger "noise-subspace" suggests more accurate estimation of it: Pisarenko works perfectly well in theory, it's the practice with estimated $\mathbf{R}$ where it can fail.

Now, note that due to the orthogonality property of the eigenvectors of a Hermitian matrix we for have all of these "minimal" eigenvectors $\{\mathbf{g}_m\}_{m=L+1}^{M}$ that

$$\mathbf{g}_m^H\mathbf{q}(\omega_n) \qquad n = 1, 2, \ldots, L \tag{93}$$

This means that the MUSIC spectral estimator

$$\hat{S}music(\omega) \;=\; \frac{1}{\sum_{m=L+1}^{M} |\mathbf{g}_m^H\mathbf{q}(\omega)|^2} \tag{94}$$

should have strong peaks at $\omega = \omega_n$, $n = 1, 2, \ldots, L$. Note that MUSIC is not really a spectral estimator, in the sense that it does not provide complete information about the true spectrum $S(\omega)$. All it tries to do – and it succeeds quite nicely – is to show the sinusoidal frequencies as peaks. In the array processing application these peaks would be DOA's.

Now, as a practical matter we can form the $\mathbf{g}$'s directly from the estimated autocorrelation matrix $\hat{\mathbf{R}}$. But we could also use the techniques that we have learned about the SVD, and form

$$\mathbf{A}^H \;=\; \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_N \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \tag{95}$$

and write

$$\mathbf{A} \;=\; \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H \tag{96}$$

and recall that since

$$\hat{\mathbf{R}} \;=\; \frac{1}{N}\mathbf{A}^H\mathbf{A} \tag{97}$$

we consequently find the eigenvectors of $\hat{\mathbf{R}}$ in the unitary matrix $\mathbf{V}$. It is often useful to write

$$\mathbf{V} = (\mathbf{V}_s \;\; \mathbf{V}_n) \tag{98}$$

where these contain eigenvectors respectively from the "signal" and "noise" subspaces. So we could write

$$\hat{S}music(\omega) = \frac{1}{\mathbf{q}(\omega)^H \mathbf{V}_n \mathbf{V}_n^H \mathbf{q}(\omega)} \tag{99}$$

which is the noise-subspace version of MUSIC. The signal-subspace version is

$$\hat{S}music(\omega) = \frac{1}{M - \mathbf{q}(\omega)^H \mathbf{V}_s \mathbf{V}_s^H \mathbf{q}(\omega)} \tag{100}$$

Another variant of MUSIC is to write (99) as

$$\hat{S}music(z) = \frac{1}{\mathbf{z}^H \mathbf{V}_n \mathbf{V}_n^H \mathbf{z}} \tag{101}$$

where

$$\mathbf{z} \equiv \begin{pmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^{M-1} \end{pmatrix} \tag{102}$$

If we form

$$\begin{aligned} D(z) &\equiv |\mathbf{V}_n^H \mathbf{z}|^2 && (103) \\ &= H(z)H(1/z^*)^* && (104) \end{aligned}$$

then the angles of the roots of $H(z)$ should provide the peaks of $\hat{S}_{music}(\omega)$. This is, not surprisingly, referred to as root-MUSIC.

Finally let us recall

$$\hat{S}_{mvdr}(\omega) = \frac{1}{\mathbf{q}(\omega)^H \mathbf{R}^{-1} \mathbf{q}(\omega)} \tag{105}$$

Now we can write

$$\mathbf{R} = \sum_{m=1}^{L} \lambda_m \mathbf{g}_m \mathbf{g}_m^H + \lambda_{min} \sum_{m=L+1}^{M} \mathbf{g}_m \mathbf{g}_m^H \tag{106}$$

Suppose we "enhanced" the signal subspace by a factor κ:

$$\mathbf{R}_\kappa = \sum_{m=1}^{L} \kappa \lambda_m \mathbf{g}_m \mathbf{g}_m^H + \lambda_{min} \sum_{m=L+1}^{M} \mathbf{g}_m \mathbf{g}_m^H \tag{107}$$

and thence

$$\mathbf{R}_\kappa^{-1} \;=\; \sum_{m=1}^{L} \frac{1}{\kappa\lambda_m}\mathbf{g}_m\mathbf{g}_m^H + \frac{1}{\lambda_{min}}\sum_{m=L+1}^{M}\mathbf{g}_m\mathbf{g}_m^H \tag{108}$$

It is easy to see that

$$\hat{S}_{music}(\omega) \;=\; \lim_{\kappa\to\infty}\left\{\frac{\lambda_{min}^{-1}}{\mathbf{q}(\omega)^H\mathbf{R}_\kappa^{-1}\mathbf{q}(\omega)}\right) \tag{109}$$

meaning that MUSIC is the essentially same as MVDR with asymptotic enhancement of the signal subspace.

There is one more note about MUSIC – and it's an important one. Let us go right back to (92) and re-write as

$$\mathbf{R} \;=\; \sigma^2\mathbf{I} + \sum_{n=1}^{L} p_n\mathbf{q}(\theta_n)\mathbf{q}(\theta_n)^H \tag{110}$$

where the difference is that these **q**-vectors are parameterized not by frequency (ω) but in some other way (θ). An example would be that the observations $\mathbf{x}_n$ are from a general array of sensors and θ_n is a representation of the position (in three dimensions) of the n^{th} source. If we can write, via physics, the signal that we would *expect* (in a noise-free situation) to observe[9] at the array elements $\mathbf{x}_n$, then **R** according to (110) is a valid representation of the correlation matrix. The MUSIC idea works acceptably here too: when θ is "swept" along all its possible values[10], the MUSIC peaks should be observed at the θ_n's. This is why the "SI" in MUSIC is for "signal" not "sinusoid" – it's more general than just sinusoids.

5.4 The Minimum-Norm Method

In the signal-subspace version of MUSIC we recognized that $|\mathbf{V}_s\mathbf{q}(\omega_n)|^2 = M$ for any signal-space frequency ω_n; and we get a spectral peak by taking the reciprocal of $M - |\mathbf{V}_s\mathbf{q}(\omega_n)|^2$. Minimum-norm attempts to form that directly by seeking a "filter" **a** such that

$$\mathbf{V}_s^H\mathbf{a} \;=\; 0 \tag{111}$$

[9]This might, for example, be via electromagnetic modeling that accounts for all propagation paths and reflections that would be encountered by a source at θ.

[10]This may take some doing if θ is multi-dimensional. For example, if θ is two-dimensional, such as azimuth / range, then the MUSIC "spectrum" is a surface.

However, $\mathbf{V}_s^H$ is a “short / fat” matrix, so the solution is underdetermined. Naturally, then we seek the **a** with minimum norm – that is the SVD idea. Let us write (111) in linear-predictor format with

$$\mathbf{a} = \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} \tag{112}$$

and likewise partition

$$\begin{aligned} \mathbf{V}_s &= \begin{pmatrix} \mathbf{g}_s^T \\ \mathbf{G}_s \end{pmatrix} && (113) \\ \mathbf{V}_n &= \begin{pmatrix} \mathbf{g}_n^T \\ \mathbf{G}_n \end{pmatrix} && (114) \end{aligned}$$

which isolates the top rows of the two matrices. We have from (111)

$$\begin{aligned} 0 &= \mathbf{V}_s^H \mathbf{a} && (115) \\ &= \begin{pmatrix} \mathbf{g}_s^* & \mathbf{G}_s^H \end{pmatrix} \begin{pmatrix} 1 \\ -\mathbf{w} \end{pmatrix} && (116) \\ \mathbf{G}_s^H \mathbf{w} &= \mathbf{g}_s^* && (117) \\ \mathbf{G}_s^T \mathbf{w}^* &= \mathbf{g}_s && (118) \end{aligned}$$

We seek to minimize $\mathbf{w}^H\mathbf{w}$ subject to (118). We have

$$\begin{aligned} \nabla \left(\mathbf{w}^H \mathbf{w} \, - \, 2\lambda^T (\mathbf{G}_s^T \mathbf{w}^* - \mathbf{g}_s) \right) &= 0 && (119) \\ \mathbf{w} &= \mathbf{G}_s \lambda && (120) \end{aligned}$$

so reinstatement of the constraint gives us

$$\begin{aligned} (\mathbf{G}_s^T \mathbf{G}_s^*)\lambda^* &= \mathbf{g}_s && (121) \\ (\mathbf{G}_s^H \mathbf{G}_s)\lambda &= \mathbf{g}_s^* && (122) \\ \lambda &= \left(\mathbf{G}_s^H \mathbf{G}_s \right)^{-1} \mathbf{g}_s^* && (123) \\ \mathbf{w} &= \mathbf{G}_s \left(\mathbf{G}_s^H \mathbf{G}_s \right)^{-1} \mathbf{g}_s^* && (124) \end{aligned}$$

Let us simplify. We have

$$\begin{aligned} \mathbf{I} &= \mathbf{V}_s^H \mathbf{V}_s && (125) \\ &= \begin{pmatrix} \mathbf{g}_s^* & \mathbf{G}_s^H \end{pmatrix} \begin{pmatrix} \mathbf{g}_s^T \\ \mathbf{G}_s \end{pmatrix} && (126) \\ \mathbf{G}_s^H \mathbf{G}_s &= \mathbf{I} \, - \, \mathbf{g}_s^* \mathbf{g}_s^T && (127) \\ \left(\mathbf{G}_s^H \mathbf{G}_s \right)^{-1} &= \mathbf{I} \, + \, \frac{\mathbf{g}_s^* \mathbf{g}_s^T}{1 - \mathbf{g}_s^T \mathbf{g}_s^*} && (128) \end{aligned}$$

where (128) follows via the matrix-inversion lemma. We thus substitute back to (124) to get

$$\begin{aligned}
\mathbf{w} &= \mathbf{G}_s \left(\mathbf{I} + \frac{\mathbf{g}_s^* \mathbf{g}_s^T}{1 - \mathbf{g}_s^T \mathbf{g}_s^*} \right) \mathbf{g}_s^* && (129) \\
&= \mathbf{G}_s \frac{\mathbf{g}_s^* - (\mathbf{g}_s^T \mathbf{g}_s^*) \mathbf{g}_s^* + \mathbf{g}_s^* (\mathbf{g}_s^T \mathbf{g}_s^*)}{1 - \mathbf{g}_s^T \mathbf{g}_s^*} && (130) \\
&= (1 - \mathbf{g}_s^T \mathbf{g}_s^*)^{-1} \mathbf{G}_s \mathbf{g}_s^* && (131)
\end{aligned}$$

hence

$$\mathbf{a} = \begin{pmatrix} 1 \\ -(1 - \mathbf{g}_s^T \mathbf{g}_s^*)^{-1} \mathbf{G}_s \mathbf{g}_s^* \end{pmatrix} \tag{132}$$

An expression equivalent to (132) is also available in terms of the noise subspace. Write

$$\begin{aligned}
\mathbf{I} &= \mathbf{V}\mathbf{V}^H && (133) \\
&= (\mathbf{V}_s \;\; \mathbf{V}_n) \begin{pmatrix} \mathbf{V}_s^H \\ \mathbf{V}_n^H \end{pmatrix} && (134) \\
&= \begin{pmatrix} \mathbf{g}_s^T & \mathbf{g}_n^T \\ \mathbf{G}_s & \mathbf{G}_n \end{pmatrix} \begin{pmatrix} \mathbf{g}_s^* & \mathbf{G}_s^H \\ \mathbf{g}_n^* & \mathbf{G}_n^H \end{pmatrix} && (135)
\end{aligned}$$

hence we have (136)-(138)

$$\begin{aligned}
\mathbf{g}_s^T \mathbf{g}_s^* + \mathbf{g}_n^T \mathbf{g}_n^* &= 1 && (136) \\
\mathbf{g}_s^T \mathbf{G}_s^H + \mathbf{g}_n^T \mathbf{G}_n^H &= \mathbf{0} && (137) \\
\mathbf{G}_s \mathbf{G}_s^H + \mathbf{G}_n \mathbf{G}_n^H &= \mathbf{I} && (138)
\end{aligned}$$

We can therefore write

$$\mathbf{a} = \begin{pmatrix} 1 \\ (\mathbf{g}_n^T \mathbf{g}_n^*)^{-1} \mathbf{G}_n \mathbf{g}_n^* \end{pmatrix} \tag{139}$$

which is an alternative expression for (132). We can write

$$\hat{S}_{mn}(\omega) = \frac{1}{|\mathbf{a}^H \mathbf{q}(\omega)|^2} \tag{140}$$

and either (132) or (139) can be used.

An interpretation is as follows. We “enhance” the correlation matrix to

$$\begin{aligned}
\mathbf{R}' &\equiv \lim_{\kappa \to \infty} \left\{ \frac{1}{\kappa} \mathbf{R}_\kappa \right\} && (141) \\
&= \mathbf{V}_s \mathbf{V}_s^H && (142)
\end{aligned}$$

where $\mathbf{R}_\kappa$ is as in (107) – that is, $\mathbf{R}'$ contains only the signal subspace. We want to find a filter $\mathbf{a}$ of the form (112) such that the output power is zero – then the frequency response is zero at the frequencies contained in the signal subspace and (140) has (∞) peaks at those frequencies. But the output power of the minimum-norm filter $\mathbf{a}$ is

$$|\mathbf{R}'\mathbf{a}|^2 \;=\; 0 \tag{143}$$

or

$$|\mathbf{V}_s^H\mathbf{a}|^2 \;=\; 0 \tag{144}$$

or

$$\mathbf{G}_s^H\mathbf{w} \;=\; \mathbf{g}_s^* \tag{145}$$

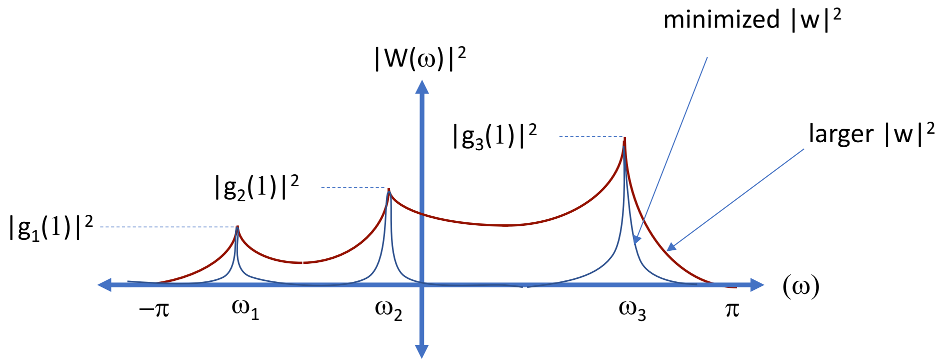
Writing the n^{th} row of (145) and conjugating, we have

$$\sum_{m=1}^{M=1} w[m]^* e^{j\omega_n m} \;=\; g_s[n] \tag{146}$$

which specifies that $|W(\omega_n)|^2 = |g_s[n]|^2$ Now, (145) is underdetermined for $\mathbf{w}$; hence the "minimum-norm" idea is to minimize $\mathbf{w}^H\mathbf{w}$. The reason this is interesting is that

$$|\mathbf{w}|^2 \;=\; \sum_{m=1}^{M-1} |w[m]|^2 \;=\; \frac{1}{2\pi}\int_{-\pi}^{\pi} |W(\omega)|^2 d\omega \tag{147}$$

by Parseval. So if we minimize $|\mathbf{a}|^2 = 1 + |\mathbf{w}|^2$ we are actually minimizing the area under the integral of the magnitude-squared prediction filter, which – notionally at least – forces the filter to sharpen its focus on sinusoids. This is as pictured below.



5.5 ESPRIT

Actually the same person invented both MUSIC and ESPRIT (Professor Thomas Kailath), hence they have cool names. ESPRIT stands for **e**stimation of **s**inusoid **p**arameters by **r**otational-**i**nvariant **t**echniques – whose relevance is perhaps a little murky, but which does sound quite uplifting. Suppose we write as usual when looking for sinusoids

$$x[n] = \sum_{l=1}^{L} b_l e^{j\omega_l n} + w[n] \tag{148}$$

where $w[n]$ is the usual AWGN. In matrix form we have

$$\begin{aligned}
\mathbf{x}_n &= \begin{pmatrix} x[n] \\ x[n-1] \\ x[n-2] \\ \vdots \\ x[n-M+1] \end{pmatrix} && (149)\\
&= \begin{pmatrix} 1 & 1 & \dots & 1 \\ e^{-j\omega_1} & e^{-j\omega_2} & \dots & e^{-j\omega_L} \\ e^{-j2\omega_1} & e^{-j2\omega_2} & \dots & e^{-j2\omega_L} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-j(M-1)\omega_1} & e^{-j(M-1)\omega_2} & \dots & e^{-j(M-1)\omega_L} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{pmatrix} \\
&\qquad + \begin{pmatrix} w[n] \\ w[n-1] \\ w[n-2] \\ \vdots \\ w[n-M+1] \end{pmatrix} && (150)\\
&= \mathbf{S}\mathbf{b} + \mathbf{w}_n && (151)
\end{aligned}$$

where $\mathbf{S}$ is $M \times L$. Now suppose we write $y[n] = x[n+1]$. Then we have

$$\mathbf{y}_n = \mathbf{S}\boldsymbol{\Omega}^*\mathbf{b} + \mathbf{w}_n \tag{152}$$

where

$$\boldsymbol{\Omega} = \begin{pmatrix} e^{-j\omega_1} & 0 & 0 & \dots & 0 \\ 0 & e^{-j\omega_2} & 0 & \dots & 0 \\ 0 & 0 & e^{-j\omega_3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & e^{-j\omega_L} \end{pmatrix} \tag{153}$$

Now we have

$$\mathbf{R}_{xx} = \mathbf{S}\mathbf{P}\mathbf{S}^H + \sigma^2\mathbf{I} \tag{154}$$

where[11] $\mathbf{R}_{xx} \equiv \mathcal{E}\{\mathbf{x}_n\mathbf{x}_n^H\}$ and

$$\mathbf{P} = \mathcal{E}\{\mathbf{b}\mathbf{b}^H\} = \begin{pmatrix} P_1 & 0 & 0 & \dots & 0 \\ 0 & P_2 & 0 & \dots & 0 \\ 0 & 0 & P_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & P_L \end{pmatrix} \tag{155}$$

and $P_i \equiv \mathcal{E}\{|b_i|^2\}$. We also have

$$\mathbf{R}_{xy} = \mathbf{S}\mathbf{P}\mathbf{\Omega}\mathbf{S}^H + \sigma^2\mathbf{\Gamma} \tag{156}$$

where $\mathbf{R}_{xy} \equiv \mathcal{E}\{\mathbf{x}_n\mathbf{y}_n^H\}$ and

$$\mathbf{\Gamma} \equiv \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix} \tag{157}$$

Define

$$\mathbf{C}_{xx} \equiv \mathbf{R}_{xx} - \sigma^2\mathbf{I} \tag{158}$$

$$\mathbf{C}_{xy} \equiv \mathbf{R}_{xy} - \sigma^2\mathbf{\Gamma} \tag{159}$$

Then solving the generalized eigenvalue equation

$$(\mathbf{C}_{xx} - \lambda\mathbf{C}_{xy})\mathbf{g} = 0 \tag{160}$$

is tantamount to looking for solutions λ to

$$\mathbf{S}\mathbf{P}\mathbf{S}^H - \lambda\mathbf{S}\mathbf{P}\mathbf{\Omega}\mathbf{S}^H = 0 \tag{161}$$

$$\mathbf{S}\mathbf{P}(\mathbf{I} - \lambda\mathbf{\Omega})\mathbf{S}^H = 0 \tag{162}$$

yields the sinusoids $\{e^{j\omega_l}\}$ directly as the solutions λ. The solution to (160) is sometimes known as a *matrix pencil.* We can write (160) as

$$(\mathbf{C}_{xx}\mathbf{C}_{xy}^{-1} - \lambda\mathbf{I})(\mathbf{C}_{xy}\mathbf{g}) = 0 \tag{163}$$

[11]The ponderous subscript notation is necessary here.

or

$$(\mathbf{C}_{xx}\mathbf{C}_{xy}^{-1})(\mathbf{C}_{xy}\mathbf{g}) \;=\; \lambda(\mathbf{C}_{xy}\mathbf{g}) \tag{164}$$

which is a standard equation for eigenstuff. So λ and $\mathbf{C}_{xy}\mathbf{g}$ from (164) solve (160). There are more-efficient solutions, however. And beyond our scope here is that ESPRIT is actually a total least-squares (TLS) solution.

ECE 6123
Advanced Signal Processing
Model Order Selection

Peter Willett

Fall 2017

1 Background on Hypothesis Testing

Let us begin with the principle of optimal decision-making. it is simple to show that given a set of simple hypotheses $\mathcal{H}_i$ ($i \in \{1, 2, \ldots, I\}$), the optimal – in the sense of a minimization of the probability of error – decision is to select

$$\mathcal{H}_j \;=\; \arg\max_{\mathcal{H}_i}\{p(\mathbf{u}|\mathcal{H}_i)Pr(\mathcal{H}_i)\} \tag{1}$$

where $\mathbf{u}$ is the observed data and $p(\cdot)$ represents a probability density. A *simple* hypothesis is one in which $p(\mathbf{u}|\mathcal{H}_i)$ has meaning or can be written. To see this, write

$$\Omega_i \;=\; \{\mathbf{u} \text{ such that } \mathbf{u} \in \Omega_i \text{ means decide } \mathcal{H}_i\} \tag{2}$$

Then

$$\begin{aligned} P(error) \;&=\; \sum_{i=1}^{I} Pr(\mathbf{u} \notin \Omega_i|\mathcal{H}_i)Pr(\mathcal{H}_i) && (3)\\ &=\; 1 - \sum_{i=1}^{I} Pr(\mathbf{u} \in \Omega_i|\mathcal{H}_i)Pr(\mathcal{H}_i) && (4)\\ &=\; 1 - \sum_{i=1}^{I} \int_{\Omega_i} p(\mathbf{u}|\mathcal{H}_i)Pr(\mathcal{H}_i)d\mathbf{u} && (5)\\ &=\; 1 - \int \sum_{i=1}^{I} \{\mathcal{I}(\text{decide } \mathcal{H}_i)p(\mathbf{u}|\mathcal{H}_i)Pr(\mathcal{H}_i)\}\, d\mathbf{u} && (6) \end{aligned}$$

which is clearly minimized by the rule (1). An example of a simple hypothesis is $\mathcal{H}_i$ that $\{u[n]\}$ is white and Gaussian with mean time series $\{\mu_i[n]\}$.

A *composite*-hypothesis situation, on the other hand, is one in which we have $p(\mathbf{u}|\theta)$ and

$$\mathcal{H}_i \;=\; \{\theta \in \Theta_i\} \tag{7}$$

for some exhaustive set of Θ_i's. Note that if there exists any *prior* probability measure on θ then this is actually a simple hypothesis test, since we can write

$$p(\mathbf{u}|\mathcal{H}_i) \;=\; \int p(\mathbf{u}|\theta)p(\theta|\mathcal{H}_i)d\theta \tag{8}$$

But otherwise the test is *composite*. The most common testing strategy for composite testing is to use the generalized likelihood (GL)

$$\max_{\theta\in\Theta_i}\{p(\mathbf{u}|\theta)\} \tag{9}$$

and in the case of only two hypotheses it would be simpler to express this as a ratio: the GLR.

To be concrete, suppose you have been given a section of time series $\{u[n]\}_{n=0}^{N-1}$. You are asked to fit an AR model to this. What order AR model? If we maximize (9) the answer is: as large as we can make it. This is because a second-order model is a special case of.a third-order model, and hence the maximized likelihood under a third-order assumption can be no smaller than that under a second-order assumption.

Notionally, there comes a point when increasing the order of the model amounts to "fitting the noise" – it is not providing better explanation of the data, it is just able to *wiggle* more to reduce the deviations. However, how to deal with unknown model order is not at all straightforward; the reason is that unless $p(\theta|\mathcal{H}_i)$ and $Pr(\mathcal{H}_i)$ are known, there is no solidly Bayesian means to test. At any rate, there are two ingredients that we must have – a maximized likelihood and an appropriate *penalty* for over-fitting – and we will attack both in subsequent sections.

2 Maximized Likelihood

2.1 The AR Case

According to the AR model

$$u[n] \;=\; \nu[n] - \sum_{k=1}^{M-1} a_k^* u[n-k] \tag{10}$$

the best predictor for $\{u[n]\}$ based on the past is

$$\hat{u}[n] \;=\; \sum_{k=1}^{M-1} a_k^* u[n-k] \tag{11}$$

which leaves prediction error $\{\nu[n]\}$ having power σ_ν^2 – which we usually call $\{f_m[n]\}$ and P_m for the m^{th}-order model – which according to (10) is a white time sequence. It's easy to see that we have

$$\begin{aligned}
\log(p(\mathbf{u})) &= \sum_{n=0}^{N-1} \log(p(u[n]|u[n-1],\ldots,u[0])) && (12)\\
&\longrightarrow \sum_{n=0}^{N-1} \log(p(u[n]|u[n-1],\ldots,u[n-M])) && (13)\\
&= \begin{cases} \left(-\frac{\sum_{n=0}^{N-1} f_m[n]^2}{2P_m} - \frac{N}{2}\log(2\pi P_m)\right) & \in \Re \\ \left(-\frac{\sum_{n=0}^{N-1} |f_m[n]|^2}{P_m} - N\log(\pi P_m)\right) & \notin \Re \end{cases} && (14)
\end{aligned}$$

Presumably this increases with model-order m and decreases with the number of data N.

2.2 The Eigen-method Case

Suppose we have $\{\mathbf{u}_n\}_{n=1}^N$ that are complex Gaussian based on covariance matrix $\mathbf{R}$. We have

$$\begin{aligned}
p(\{\mathbf{u}_n\}_{n=1}^N) &= \frac{1}{|\pi\mathbf{R}|^N} e^{-\sum_{n=1}^N \mathbf{u}_n^H\mathbf{R}^{-1}\mathbf{u}_n} && (15)\\
&= \frac{1}{|\pi\mathbf{R}|^N} e^{-Tr(\sum_{n=1}^N \mathbf{u}_n^H\mathbf{R}^{-1}\mathbf{u}_n)} && (16)\\
&= \frac{1}{|\pi\mathbf{R}|^N} e^{-Tr(\mathbf{R}^{-1}\sum_{n=1}^N \mathbf{u}_n\mathbf{u}_n^H)} && (17)\\
&= \frac{1}{|\pi\mathbf{R}|^N} e^{-NTr(\mathbf{R}^{-1}\hat{\mathbf{R}})} && (18)
\end{aligned}$$

where of course

$$\hat{\mathbf{R}} \equiv \frac{1}{N}\sum_{n=1}^N \mathbf{u}_n\mathbf{u}_n^H \qquad (19)$$

Our goal is to maximize (18) with respect to $\mathbf{R}$. But since this is an eigen-method, we constrain $\hat{\mathbf{R}}$ to be of reduced rank, say $p < M$.

Let us begin by assuming that the eigenvalues of $\mathbf{R}$ (i.e., $\{\lambda_i\}$) are fixed – this means that $|\mathbf{R}|$ is also fixed. We write

$$\hat{\mathbf{R}} = \sum_{i=1}^M \hat{\lambda}_i\hat{\mathbf{v}}_i\hat{\mathbf{v}}_i^H \qquad (20)$$

as the eigendecomposition of the empirical covariance matrix. We then have

$$\begin{aligned}
Tr(\mathbf{R}^{-1}\hat{\mathbf{R}}) &= \sum_{i=1}^{M} \hat{\lambda}_i Tr(\mathbf{R}^{-1}\hat{\mathbf{v}}_i\hat{\mathbf{v}}_i^H) & (21)\\
&= \sum_{i=1}^{M} \hat{\lambda}_i Tr(\hat{\mathbf{v}}_i^H\mathbf{R}^{-1}\hat{\mathbf{v}}_i) & (22)\\
&\geq \sum_{i=1}^{M} \hat{\lambda}_{(i)}/\lambda_{(i)} & (23)
\end{aligned}$$

where $\hat{\lambda}_{(1)} \geq \hat{\lambda}_{(2)} \geq \ldots \geq \hat{\lambda}_{(M)}$ and $\lambda_{(1)} \geq \lambda_{(2)} \geq \ldots \geq \lambda_{(M)}$. Equation (23) follows from the same logic that we applied to minimize the Frobenius norm of a low-rank approximation to a given matrix; the difference is that there we minimized the Frobenius norm and hence maximized the trace-term; here we are *minimizing* the trace term and hence we match the largest $\hat{\lambda}_i$ with the smallest λ_i^{-1} – which means the largest $\hat{\lambda}_i$ is paired to the largest λ_i, second-largest to second-largest, etc. We thus have

$$\log(p(\{\mathbf{u}_n\}_{n=1}^N)) = N\sum_{i=1}^{M} \hat{\lambda}_{(i)}/\lambda_{(i)} - \sum_{i=1}^{M} N\log(\pi\lambda_{(i)}) \qquad (24)$$

We take the gradient with respect to $\{\lambda_{(i)}\}_{i=1}^M$ under the constraint that $\lambda_{(i)} = \lambda_0$ for $p < i \leq M$. Setting it to zero we have

$$\begin{aligned}
0 &= -\frac{N\hat{\lambda}_{(i)}}{\lambda_{(i)}^2} + \frac{N}{\lambda_{(i)}} & (25)\\
\Longrightarrow \lambda_{(i)} &= \hat{\lambda}_{(i)} & (26)
\end{aligned}$$

for $i \in \{1, p\}$, and

$$\begin{aligned}
0 &= -N\sum_{i=p+1}^{M} \frac{\hat{\lambda}_{(i)}}{\lambda_0^2} + N\left(\frac{M-p}{\lambda_0}\right) & (27)\\
\Longrightarrow \lambda_0 &= \frac{1}{M-p}\sum_{i=p+1}^{M} \hat{\lambda}_{(i)} & (28)
\end{aligned}$$

for $i \in \{p+1, M\}$. Clearly the maximum

$$\begin{aligned}
&\log(p(\{\mathbf{u}_n\}_{n=1}^N))\\
&\leq N\left(\sum_{i=1}^{p} \frac{\hat{\lambda}_{(i)}}{\hat{\lambda}_{(i)}} + \frac{\sum_{i=p+1}^{M} \hat{\lambda}_{(i)}}{\frac{1}{M-p}\sum_{i=p+1}^{M} \hat{\lambda}_{(i)}}\right)
\end{aligned}$$

$$- N\left(\sum_{i=1}^{p}\log(\pi\hat{\lambda}_{(i)}) + (M-p)\log\left(\pi\frac{1}{M-p}\sum_{i=p+1}^{M}\hat{\lambda}_{(i)}\right)\right) \tag{29}$$

$$= \quad -N\left(p \,+\, (M-p) \,+\, \sum_{i=1}^{M}\log(\hat{\lambda}_{(i)}) \,-\, (M-p)\sum_{i=p+1}^{M}\log\left(\hat{\lambda}_{(i)}^{\frac{1}{M-p}}\right)\right)$$

$$- N\left(M\log(\pi) + (M-p)\log\left(\frac{1}{M-p}\sum_{i=p+1}^{M}\hat{\lambda}_{(i)}\right)\right) \tag{30}$$

$$= \quad N\left((M-p)\log\left(\frac{\left(\prod_{i=p+1}^{M}\hat{\lambda}_{(i)}\right)^{\frac{1}{M-p}}}{\frac{1}{M-p}\sum_{i=p+1}^{M}\hat{\lambda}_{(i)}}\right) - M\log(\pi e) - \log(|\hat{\mathbf{R}}|)\right) \tag{31}$$

So, in words: the hard work of the test statistic is done by the *ratio of the geometric to arithmetic means of the eigenvalues in the (empirical) noise subspace.*

2.3 A Little Bit of Random Matrix Theory

RMT is an emerging field for statisticians, with much activity. The results are not simple to prove, and no effort will be given here to offer proofs. There are applications in testing and especially in communications. Signal processors are interested, but are struggling to find applications.

First, please be aware that we are interested (here) in square Hermitian matrices. There are two such classes. The first is the **Wigner** class that involves an $M \times M$ matrix $\mathbf{A} = \mathbf{A}^H$ that is composed of zero-mean complex Gaussian random variables with $1/M$ as their variance[1]. The second is the **Wishart** class of random matrices where

$$\hat{\mathbf{R}} \;=\; \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_n\mathbf{u}_n^H \tag{32}$$

where $\mathcal{E}\{\mathbf{u}_n\mathbf{u}_n^H\} = \mathbf{S}$ which is of dimension $M \times M$. In the Wishart class we sometimes are interested in asymptotics where

$$\lim_{N\to\infty}\left\{\frac{M}{N}\right\} = \gamma \tag{33}$$

shows that there is a scaling between matrix size and estimation accuracy – it does not apply to a situation of near-convergence to a good estimate of the covariance matrix.

[1]Obviously this can be scaled; but all entries must be *iid*.

Wishart Density. It can be shown that the probability density function (pdf) of $\hat{\mathbf{R}}$ is

$$p(\hat{\mathbf{R}}) \;=\; \frac{\left|\hat{\mathbf{R}}\right|^{N-M-1} e^{-\frac{1}{2}Tr(\mathbf{S}^{-1}\hat{\mathbf{R}})}}{2^{\frac{MN}{2}}|\mathbf{S}|^{\frac{N}{2}}\Gamma_M(\frac{N}{2})} \tag{34}$$

where

$$\Gamma_M\left(\frac{N}{2}\right) \;\equiv\; \pi^{\frac{M(M-1)}{4}}\prod_{i=1}^{M}\Gamma\left(\frac{N}{2}-\frac{i-1}{2}\right) \tag{35}$$

is the "multi-variate Gamma function" and in which Γ denotes the usual Gamma function. The pdf (34) is usually written as $\hat{\mathbf{R}} \sim W_M(\mathbf{S}, N)$. It is not asymptotic, and applies for any N and M. The pdf (34) looks fascinating, but I'll admit that I've never seen an application of the Wishart pdf.

Semi-Circle Law. This applies to the Wigner case. It says that the *marginal* pdf of any eigenvalue has pdf

$$p(\lambda) \;=\; \frac{\sqrt{4-\lambda^2}}{2\pi} \tag{36}$$

This (36) is not precisely the pdf for any finite-size matrix, but can be shown to be the asymptotic pdf as $M \to \infty$.

Marcenko-Pastur Law. This is the analog of (36) for Wishart matrices, which is probably more useful for us. In this case the result is asymptotic: the scaled situation of (33). For the case $\gamma < 1$ we have

$$p(\lambda) \;=\; \begin{cases} \frac{\sqrt{(b_+-\lambda)(\lambda-b_-)}}{2\pi\gamma\lambda} & b_- \le \lambda \le b_+ \\ 0 & \text{else} \end{cases} \tag{37}$$

in which

$$b_- \;\equiv\; (1-\sqrt{\gamma})^2 \tag{38}$$
$$b_+ \;\equiv\; (1+\sqrt{\gamma})^2 \tag{39}$$

For $\gamma > 1$ we have

$$p(\lambda) \;=\; \frac{1}{1-\gamma}\delta(\lambda) \;+\; \frac{1}{\gamma}\begin{cases} \frac{\sqrt{(b_+-\lambda)(\lambda-b_-)}}{2\pi\gamma\lambda} & b_- \le \lambda \le b_+ \\ 0 & \text{else} \end{cases} \tag{40}$$

in which

$$b_- \;\equiv\; 0 \tag{41}$$
$$b_+ \;\equiv\; (1+\sqrt{\gamma})^2 \tag{42}$$

> This difference – that there are zero eigenvalues – is not so surprising, in that if $\gamma < 1$ it is necessarily that $\hat{\mathbf{R}}$ be singular, since there are fewer snapshots than dimensions.

There are (many) other interesting RMT results. One example is the Tracy-Widom theory for the pdf of the largest eigenvalue. Obviously this would be quite useful when testing for a nontrivial *signal subspace* from data. It is not presented since it is quite complex.

2.4 Asymptotic Distribution of the MLE

Under mild but non-trivial regularity conditions the MLE $\hat{\theta}$ can be converges, as the number of samples upon which is computed goes to infinity, to Gaussian, with mean θ (the true parameter) and covariance $\mathbf{J}_\theta^{-1}$; that is, we have

$$p(\hat{\theta}) \approx \sqrt{\left|\frac{\mathbf{J}_\theta}{2\pi}\right|} e^{-\frac{1}{2}(\hat{\theta}-\theta)^T \mathbf{J}_\theta (\hat{\theta}-\theta)} \tag{43}$$

The latter quantity $\mathbf{J}_\theta$ is the *Fisher information matrix* (FIM). Generally one does not know the true θ so one is content to use $\mathbf{J}_{\hat{\theta}}$ – this is called the *observed information* (OI), which has little theoretical backing but it often useful in situations where $\mathbf{J}_\theta$ is not independent[2] of θ. There is nothing to be embarrassed about in using the OI instead of the FIM; just be aware that it is an approximation.

3 Penalty Criteria

If we knew $Pr(\mathcal{H}_i)$ and $p(\theta|\mathcal{H}_i)$ then we would have (1) as

$$\mathcal{H}_j = \arg\max_{\mathcal{H}_i} \left\{ \int p(\mathbf{u}|\mathcal{H}_i, \theta) p(\theta|_i) d\theta Pr(\mathcal{H}_i) \right\} \tag{44}$$

and we would be done. We know neither. But we would like some means to *penalize* more-complex models, such that we could select

$$\mathcal{H}_j = \arg\max_{\mathcal{H}_i} \left\{ \int p(\mathbf{u}|\mathcal{H}_i, \theta) p(\theta|_i) d\theta Pr(\mathcal{H}_i) - \kappa_p \right\} \tag{45}$$

[2]An example of such lack of dependence is the estimation of the mean of Gaussian data; but such nice behavior is the exception rather than the rule.

as the penalty that applies to a model with p free parameters (such[3] as AR(p)). But in fact we need more than this, since some p^{th}-order models are more attractive than others. It is bests to let the data decide.

There are several "penalty terms" for model order that have some appeal: the Akaike information criterion (AIC), Rissanen's minimum descriptor length (MDL) and the Bayesian information criterion (BIC) come to mind. There are others, and it is a field of continual developments. No penalty term has a really rigorous development; but that is forgivable since the problem of model order selection (without prior information) is not well-posed.

3.1 AIC

First, please recall (or be introduced to) the Kullback-Leibler (KL) divergence between to probability measures (densities)

$$d_{kl}(p,q) \equiv \int p \log\left(\frac{p}{q}\right) \tag{46}$$

We have $d_{kl} = 0$ if and only if $p = q$; otherwise $d_{kl} > 0$. The KL divergence has a great deal of importance in information theory, and is of paramount importance in *large deviations* theory where it describes convergence exponents. And, indeed, if $p(x,y)$ is a joint distribution and $q(x,y)$ has the same marginals but is the special case that the two are independent, then $d_{kl}(p,q)$ is the same as Shannon's Information. But for our purposes, just be aware that d_{kl} is a measure of the difference between p and q.

Akaike assumed:

θ_0 is the true parameter for the true model, which has dimension (number of parameters to be estimated) p_0.

θ is the expected value of the parameter, of order p, for the model being tested.

$\hat{\theta}$ is the maximum-likelihood estimate (MLE) of the parameter, of order p, for the model being tested.

Akaike in 1975 wanted to choose the best model in the sense of minimizing

$$d_{kl}(p_{\theta_0}, p_\theta) \equiv \int p_{\theta_0} \log\left(\frac{p_{\theta_0}}{p_\theta}\right) \tag{47}$$

[3]In the eigenmethod case, the number of free parameters, in the notation just used, is pM, corresponding to the requisite eigenvalues and eigenvectors in the signal-subspace. It is noted that each eigenvector only requires $M-1$ parameters due to its unit-length requirement.

which amounts to maximizing

$$\int p_{\theta_0}(\mathbf{u}) \log\left(p_\theta(\mathbf{u})\right) d\mathbf{u} \tag{48}$$

where we have defined $\mathbf{u} \equiv \{\mathbf{u}_n\}_{n=1}^N$. Under the the assumption that $\hat{\theta}$ is sufficient for θ we have both

$$p_\theta(\mathbf{u}) \;=\; p_{\hat{\theta}}(\mathbf{u}) p_\theta(\hat{\theta}) \tag{49}$$

which follows from the *factorization theorem* for sufficient statistics; and the asymptotic MLE distribution expression (43). Substituting (49) and (43) into (48) we propose to maximize

$$\int p_{\theta_0}(\mathbf{u}) \left(\log\left(p_{\hat{\theta}}(\mathbf{u})\right) - \frac{1}{2}(\hat{\theta}-\theta)^T \mathbf{J}_\theta (\hat{\theta}-\theta) + \frac{1}{2}\log\left(\left|\frac{\mathbf{J}_\theta}{2\pi}\right|\right)\right) d\mathbf{u} \tag{50}$$

over the model type and order.

The AIC development says that the first term in (50) is the maximized likelihood. The second term assumes that the covariance is indeed $\mathbf{J}^{-1}$, so the expectation results in p, the dimension of $\hat{\theta}$. The third term is ignored. Hence in its raw form the AIC maximizes

$$\arg\max_p \left\{ \max_{\theta\in\Theta_p} \{\log(p(\{\mathbf{u}_n\}_{n=1}^N))\} - p \right\} \tag{51}$$

As can be seen, however, (at least) these problems can be identified:

- The integration in the first term of (50) is ignored.
- It is not clear why $\mathbf{J}^{-1}$ should be the covariance in the second term of (50) when θ_0 is true.
- It is unclear why the third term in (50) can be ignored.
- It is unexplained why the integration in (48) should be over $\mathbf{u}$ when in fact $\mathbf{u}$ is known.

There is a "corrected" form of the AIC for finite data sizes – that is, finite N. It is

$$\arg\max_p \left\{ \max_{\theta\in\Theta_p} \{\log(p(\{\mathbf{u}_n\}_{n=1}^N))\} - \frac{Np}{N-p} \right\} \tag{52}$$

The AIC is probably the first attempt to address the issue of model-order selection, and should be complimented for that; and in fact it works reasonably well for small N. But its development is a Swiss cheese.

3.2 MDL

Rissanen originally developed the MDL with an idea from information theory. A nice intuition is from a notional example. Consider we have an alphabet of 2 letters (OK: here "letters" means bits), N data from this alphabet, and two coding strategies:

1. Treat all symbols are equally likely. N data can be represented by N bits.

2. Randomly[4] generate 2^{10} symbol-probability choices $\{\{p_{i,n}\}_{i=1}^{32}\}_{n=1}^{1024}$, in which $p_{i,n}$ is the probability of symbol i under model n, and of course we must have $\sum_{i=1}^{32} p_{i,n} = 1$. Then for the N data perform a Huffman coding procedure for each $\{p_{i,n}\}_{i=1}^{32}$. Use the shortest coded symbol stream, which should be less than N. Since you must also encode the identity of the code used, the number of coded bits is $\min_n\{N\bar{L}_n\}+10$.

Clearly there is more "overhead" needed in the second strategy[5]; but if the data fits it better (shorter coded length) by enough compared to the overhead, then it might be a better strategy. Suppose we used 2^{20} $\{p_{i,n}\}$'s – presumably the best $\bar{L}_n$ should be lower than for 2^{10}, but is it worth the extra 10 bits needed to tell the decoder which codebook we used?

As I indicated, RIssanen originally was motivated by the ideas above – find the best encoding of the data – which is reminiscent both of Kolmogorov complexity theory and of "universal" source coding. But I find Djuric's 1998 paper the most appealing way to develop MDL. Djuric starts with (1) and takes $Pr(\mathcal{H}_i)$ uniform (and hence ignorable). He then writes

$$p(\mathbf{u}|\mathcal{H}_i) \;=\; \int p(\mathbf{u}|\theta, \mathcal{H}_i) p(\theta|\mathcal{H}_i) d\theta \tag{53}$$

and takes $p(\theta|\mathcal{H}_i)$ uniform as well. Let us put this into a form that we can use:

$$p(\mathbf{u}|\mathcal{H}_i) \;=\; \int p(\theta|\mathcal{H}_i) e^{N\left[\frac{1}{N}\sum_{n=1}^{N} \log(p(\mathbf{u}|\theta,\mathcal{H}_i))\right]} d\theta \tag{54}$$

We have to discuss Laplace's method of integral approximation now. Consider

$$I(t) \;=\; \int_V f(\mathbf{y}) e^{-tg(\mathbf{y})} d\mathbf{y} \tag{55}$$

[4]For uniformity this would be according to the Dirichlet density and model.

[5]We are not interested in the overhead to compute the codes, although this may be considerable; we are only interested in the encoded length.

where $g(y)$ attains its minimum at $\mathbf{y} = \mathbf{c}$ which is an interior point[6] of V. Then since we know $\nabla g(\mathbf{y})|_{\mathbf{y}=\mathbf{c}} = 0$ we can approximate

$$I(t) \;\longrightarrow\; \int_{\mathcal{B}(\mathbf{c})} f(\mathbf{c}) e^{-t\left[g(c) - \frac{1}{2}(\mathbf{y}-\mathbf{c})^T \mathbf{G}(\mathbf{y}-\mathbf{c})\right]} d\mathbf{y} \tag{56}$$

as $t \to \infty$, where $\mathcal{B}(\mathbf{c})$ is a small ball surrounding $\mathbf{c}$ and

$$\mathbf{G} \;\equiv\; \nabla^2 g(\mathbf{y})|_{\mathbf{y}=\mathbf{c}} \tag{57}$$

is the Hessian. We get

$$I(t) \;\longrightarrow\; f(\mathbf{c}) e^{-tg(\mathbf{c})} \sqrt{\left|\frac{2\pi}{t\mathbf{G}}\right|} \tag{58}$$

after integrating and recognizing the multivariate Gaussian form of the integral.

For us doing the MDL derivation we have the correspondences from our problem to the Laplace integral and solution in (55)-(58) given by

$$\begin{aligned}
f(\cdot) &\;\leftarrow\; p(\theta|\mathcal{H}_i) \text{ (uniform)} && (59)\\
t &\;\leftarrow\; N \text{ (the number of samples)} && (60)\\
\mathbf{c} &\;\leftarrow\; \hat{\theta} \text{ (the MLE)} && (61)\\
\mathbf{y} &\;\leftarrow\; \theta && (62)\\
g(\cdot) &\;\leftarrow\; -\frac{1}{N}\sum_{n=1}^{N} \log(p(\mathbf{u}_n|\theta, \mathcal{H}_i)) && (63)\\
\mathbf{G} &\;\leftarrow\; +\mathbf{J}_1 && (64)
\end{aligned}$$

where $\mathbf{J}_1$ is the FIM for one snapshot of data, and recall the negative sign in the definition of the FIM when the second-derivative is used. Consequently we can write

$$\begin{aligned}
\log(p(\mathbf{u}|\mathcal{H}_i)Pr(\mathcal{H}_i)) \;\to\; & \log(p(\hat{\theta}|\mathcal{H}_i)) + \log(p(\mathbf{u}|\hat{\theta}, \mathcal{H}_i)) - \frac{p}{2}\log(2\pi) \\
& - \frac{1}{2}\log(|N\mathbf{J}_1|) + \log(Pr(\mathcal{H}_i)) \qquad (65)
\end{aligned}$$

Ignoring the terms that don't scale with N – meaning the first, third and fifth terms – we have at last the task to look for

$$\arg\max_{p} \left\{ \max_{\theta \in \Theta_p} \{\log(p(\{\mathbf{u}_n\}_{n=1}^{N}))\} - \frac{1}{2}\log(|\mathbf{J}|) \right\} \tag{66}$$

[6]The situation that $\mathbf{c}$ is on the boundary of V is also treatable by Laplace's method, but is not at issue here.

where $\mathbf{J} = N\mathbf{J}_1$ is the FIM of the full data.

One interpretation of (66) is that the penalty term is the maximized logarithm of (43) – with a zero exponent. That is, it is perhaps a fair point of comparison of the maximized likelihood against what it should be.

I am very fond of Djuric's development, and of the "full" result (66). Nonetheless it is worth mentioning that one might consider setting $\mathbf{J} = N\sigma^2\mathbf{I}$. In that case we have

$$\log(|\mathbf{J}|) \; = \; p\log(N) \; + \; p\log(\sigma^2) \tag{67}$$

Again ignoring the terms not increasing with N, we have the *original* MDL

$$\arg\max_{p}\left\{\max_{\theta\in\Theta_p}\{\log(p(\{\mathbf{u}_n\}_{n=1}^{N}))\} \; - \; \frac{p}{2}\log(N)\right\} \tag{68}$$

which is certainly very simple but gives no visibility into models of the same order. It is worth mentioning that Rissanen, in later papers, enhanced his development to incorporate the FIM.

3.3 BIC

The BIC is actually equivalent to the form (68) of the MDL. It is "derived" by assuming the model is from the exponential family.

ECE 6123
Advanced Signal Processing
The SVD and its SP Application

Peter Willett

Fall 2017

1 Least Squares Formulation of Wiener Filtering

1.1 The Equations

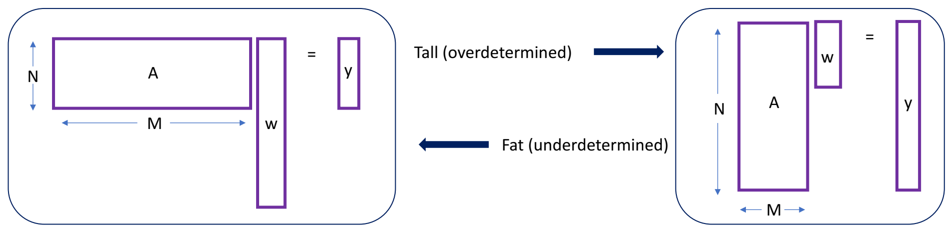
Let's suppose we arrange the data into a matrix:

$$\begin{pmatrix} \longleftarrow & \mathbf{u}_1^H & \longrightarrow \\ \longleftarrow & \mathbf{u}_2^H & \longrightarrow \\ & \vdots & \\ \longleftarrow & \mathbf{u}_2^H & \longrightarrow \end{pmatrix} \begin{pmatrix} w[1] \\ w[2] \\ \vdots \\ w[M] \end{pmatrix} \equiv \mathbf{A}\mathbf{w} = \mathbf{y}^* \equiv \begin{pmatrix} y[1]^* \\ y[2]^* \\ \vdots \\ y[N]^* \end{pmatrix} \qquad (1)$$

The Wiener goal is actually the least-squares goal: choose $\mathbf{w}$ to minimize the error

$$J(\mathbf{w}) \equiv ||\mathbf{y} - \mathbf{d}||^2 = \sum_{n=1}^{N} |e[n]|^2 = \sum_{n=1}^{N} |d[n] - y[n]|^2 \qquad (2)$$

A lot depends on whether the matrix $\mathbf{A}$ is short and fat or tall and skinny. In the short / fat case the linear system (1) is underdetermined, meaning there are more variables in $\mathbf{w}$ than there are equations to match $\mathbf{y}$ to $\mathbf{d}$.That means that we can make $J(\mathbf{w}) = 0$ with multiple $\mathbf{w}$'s – which one should we choose? In the tall / skinny case (1) is likewise **over**determined, meaning that in any nontrivial case we cannot find $\mathbf{w}$ such that $J(\mathbf{w}) = 0$ – and in that case it makes sense to find the minimizing $\mathbf{w}$. The situations are illustrated below.



If $N = M$ and there is no triviality (linear dependence in columns of $\mathbf{A}$) then we have a unique solution – this is the least interesting case and we will ignore it from now on.

1.2 The Overdetermined Case

Presumably this is familiar. To minimize (2) we apply the *p.o.o.* and see that optimally

$$\mathbf{A}^H(\mathbf{d} - \mathbf{A}\mathbf{w}) \;=\; 0 \tag{3}$$

or

$$\mathbf{w} \;=\; (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d} \tag{4}$$

unless $(\mathbf{A}^H\mathbf{A})$ is singular. We could write

$$\begin{aligned}
\mathbf{A}^H\mathbf{A} \;&=\; \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_N \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \longleftarrow & \mathbf{u}_1^H & \longrightarrow \\ \longleftarrow & \mathbf{u}_2^H & \longrightarrow \\ & \vdots & \\ \longleftarrow & \mathbf{u}_2^H & \longrightarrow \end{pmatrix} && (5) \\
&=\; \sum_{n=1}^{N} \mathbf{u}\mathbf{u}^H && (6) \\
&=\; N\hat{\mathbf{R}} && (7)
\end{aligned}$$

where the last equation assumes that the covariance matrix $\mathbf{R}$ is estimated by simple averaging. In a similar way we could write

$$\begin{aligned}
\mathbf{A}^H\mathbf{d} \;&=\; \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_N \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} d[1]^* \\ d[2]^* \\ \vdots \\ d[N]^* \end{pmatrix} && (8) \\
&=\; \sum_{n=1}^{N} \mathbf{u}d[n]^* && (9) \\
&=\; N\hat{\mathbf{p}} && (10)
\end{aligned}$$

where again the last equation assumes that the cross-correlation vector $\mathbf{p}$ is estimated by simple averaging. Written in this way we have optimally

$$\mathbf{w} \;=\; (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d} \;=\; (N\hat{\mathbf{R}})^{-1}(N\hat{\mathbf{p}}) \;=\; \hat{\mathbf{R}}^{-1}\hat{\mathbf{p}} \tag{11}$$

meaning that the solution we get by direct dumb least-squares is identical to the Wiener solution with block averaging estimates for the covariances.

2 The Singular Value Decomposition

2.1 Relationship to Eigendecompositions

Let us assume a matrix $\mathbf{A}$ whose dimension is N (rows) by M (columns): $N \times M$. Unless $M = N$ we have no eigendecomposition. But suppose we form left and right products (which are square and of respective dimensions $M \times M$ and $N \times N$. Now eigenstuff is available:

$$(\mathbf{A}^H\mathbf{A})\mathbf{V} = \mathbf{V}\mathbf{\Gamma} \tag{12}$$

$$(\mathbf{A}\mathbf{A}^H)\mathbf{U} = \mathbf{U}\mathbf{\Lambda} \tag{13}$$

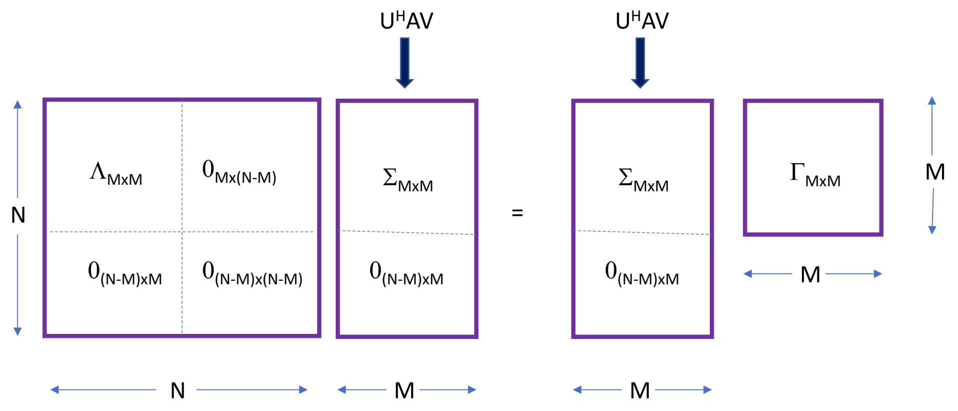
where $\mathbf{V}$ is unitary (means: $\mathbf{V}^H\mathbf{V} = \mathbf{I}$) and of dimension $M \times M$; and likewise $\mathbf{U}$ too is unitary ($\mathbf{U}^H\mathbf{U} = \mathbf{I}$) and of dimension $N \times N$. The matrices $\mathbf{\Gamma}$ and $\mathbf{\Lambda}$ are diagonal with nonnegative elements. Since the rank of $\mathbf{A}$ is $\min\{M, N\}$ this is also the rank of $(\mathbf{A}^H\mathbf{A})$ and $(\mathbf{A}\mathbf{A})^H$. Hence in the short / fat case $N < M$ there are[1] $M - N$ zeros on the diagonal of $\mathbf{\Gamma}$; and likewise in the tall / skinny case $N > M$ there are $N - M$ zeros on the diagonal of $\mathbf{\Lambda}$.

What is interesting is to form the identity

$$\mathbf{U}^H\mathbf{A}\mathbf{A}^H\mathbf{A}\mathbf{V} = \mathbf{U}^H\mathbf{A}\mathbf{A}^H\mathbf{A}\mathbf{V} \tag{14}$$

$$\mathbf{\Lambda}\mathbf{U}^H\mathbf{A}\mathbf{V} = \mathbf{U}^H\mathbf{A}\mathbf{V}\mathbf{\Gamma} \tag{15}$$

where to get (15) we've substituted (13) on the LHS and (12) on the RHS. The situation is as illustrated below.



[1]There could be more zeros if $\mathbf{A}$ is rank-deficient, meaning that some $\mathbf{u}_n$'s are linearly dependent; but this is a trivial case and would be dilatory to explore.

In the above figure we've assumed for concreteness that $N > M$; there is no loss of generality in doing that in this section. Note that we have inserted the fact that the last $N - M$ rows of $\mathbf{U}^H\mathbf{A}\mathbf{V}$ have to be zero: the LHS tells us that it must be so. We've also (slightly) changed notation to denote only the northwest $M \times M$ block of the premultiplying matrix on the LHS to be $\mathbf{\Lambda}$. Now, we can also write

$$\mathbf{\Lambda}\mathbf{\Sigma} = \mathbf{\Sigma}\mathbf{\Gamma} \tag{16}$$

and which implies that $\mathbf{\Sigma}$ is the (unnormalized) matrix of eigenvectors of $\mathbf{\Lambda}$ (or $\mathbf{\Gamma}$). Since $\mathbf{\Lambda}$ is a diagonal matrix we know that its eigenvectors are the Cartesian basis vectors: that is, $\mathbf{\Sigma}$ itself has to be diagonal.

And that's what we wanted to show. Now we know that we have

$$\mathbf{U}^H\mathbf{A}\mathbf{V} = \begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix} \tag{17}$$

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{V}^H \tag{18}$$

in the case that $N > M$ and

$$\mathbf{U}^H\mathbf{A}\mathbf{V} = \begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix} \tag{19}$$

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{V}^H \tag{20}$$

in the case $N < M$. Equations (18) and (20) represent the singular value decomposition (SVD) of the matrix $\mathbf{A}$: the product of a unitary $N \times N$ matrix, a diagonal matrix of dimension $N \times M$ and another $M \times M$ unitary matrix. It's quite general. As will be seen very shortly the matrices can be computed via appropriate eigendecompositions; but there are ways to compute them directly that are far more efficient, especially if $N \gg M$ or $N \ll M$. The SVD is a primary tool in many signal processing tasks; we will soon see an example in the adaptive filtering venue, and then more helping us with spectral estimation.

Again for the case $N > M$ we can also explore

$$\mathbf{A}\mathbf{A}^H = \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{V}^H\mathbf{V}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{U}^H \tag{21}$$

$$= \mathbf{U}\begin{pmatrix} \mathbf{\Sigma}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\mathbf{U}^H \tag{22}$$

and

$$\mathbf{A}^H\mathbf{A} = \mathbf{V}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{V}^H \tag{23}$$

$$= \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^H \tag{24}$$

meaning that $\mathbf{\Sigma}^2$ contains the eigenvalues of $(\mathbf{A}\mathbf{A}^H)$ (or $(\mathbf{A}^H\mathbf{A})$). For the case $N < M$ we have

$$\begin{aligned}
\mathbf{A}\mathbf{A}^H &= \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{V}^H\mathbf{V}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{U}^H & (25)\\
&= \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^H & (26)
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{A}^H\mathbf{A} &= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{V}^H & (27)\\
&= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\mathbf{V}^H & (28)
\end{aligned}$$

which are the same as (22) and (24), just reversed due to the matrix size.

2.2 The Pseudo-Inverse

The pseudo-inverse, or Moore-Penrose inverse, is defined as

$$\mathbf{A}^\dagger \equiv \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\mathbf{U}^H \tag{29}$$

if $N > M$ or

$$\mathbf{A}^\dagger \equiv \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} \\ \mathbf{0} \end{pmatrix}\mathbf{U}^H \tag{30}$$

if $N < M$. If some elements of $\mathbf{\Sigma}$ are zero the modification is obvious; and if $M = N$ (and $\mathbf{A}$ is full rank) it is easy to see that $\mathbf{A}^\dagger = \mathbf{A}^{-1}$. So what?

Let's begin with the case $N > M$. We have

$$\begin{aligned}
\mathbf{A}^\dagger\mathbf{A} &= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{V}^H & (31)\\
&= \mathbf{V}\mathbf{I}\mathbf{V}^H & (32)\\
&= \mathbf{I}_{M\times M} & (33)
\end{aligned}$$

Now let's examine the case $N < M$. We now have

$$\begin{aligned}
\mathbf{A}\mathbf{A}^\dagger &= \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{V}^H\mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} \\ \mathbf{0} \end{pmatrix}\mathbf{U}^H & (34)\\
&= \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{\Sigma}^{-1} \\ \mathbf{0} \end{pmatrix}\mathbf{U}^H & (35)\\
&= \mathbf{U}\mathbf{I}\mathbf{U}^H & (36)\\
&= \mathbf{I}_{N\times N} & (37)
\end{aligned}$$

We will use these shortly.

2.3 The SVD and the Overdetermined Case

Here we have $N > M$, the tall / skinny situation. From (33) we write

$$\begin{aligned}
\mathbf{w} &= \mathbf{A}^{\dagger}\mathbf{d} && (38)\\
&= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{d} && (39)\\
&= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{U}_1^H \\ \mathbf{U}_2^H \end{pmatrix}\mathbf{d} && (40)\\
&= \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}_1^H\mathbf{d} && (41)
\end{aligned}$$

For what it is worth, we could start with (4) and use the SVD to get

$$\begin{aligned}
\mathbf{w} &= (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d} && (42)\\
&= \left[\mathbf{V}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{U}\begin{pmatrix} \mathbf{\Sigma} \\ \mathbf{0} \end{pmatrix}\mathbf{V}^H\right]^{-1}\mathbf{V}\begin{pmatrix} \mathbf{\Sigma} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{d} && (43)\\
&= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\mathbf{U}^H\mathbf{d} && (44)\\
&= \mathbf{A}^{\dagger}\mathbf{d} && (45)\\
&= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{U}_1^H \\ \mathbf{U}_2^H \end{pmatrix}\mathbf{d} && (46)\\
&= \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}_1^H\mathbf{d} && (47)
\end{aligned}$$

The message: The SVD solves the overdetermined case.

2.4 The SVD and the Underdetermined Case

In the overdetermined case there is no solution to (1), so we found the solution to minimize the error[2]. In the underdetermined (short / fat $\mathbf{A}$) case there is a whole subspace of $\mathbf{w}$'s that solves (1) – which one should we choose? Unless there are other concerns, a good choice might be to select the $\mathbf{w}$ with minimum length. So we have the optimization problem

$$\text{Minimize } \mathbf{w}^H\mathbf{w} \text{ subject to } \mathbf{A}\mathbf{w} = \mathbf{d} \qquad (48)$$

We use Lagrange multipliers, and find

$$\mathbf{w} - \mathbf{A}^H\lambda = 0 \qquad (49)$$

[2]. . . or the residuals.

at optimality. Substituting for the constraint we have

$$\begin{aligned}\mathbf{A}\mathbf{A}^H\lambda &= \mathbf{d} && (50)\\ \lambda &= (\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{d} && (51)\end{aligned}$$

hence

$$\mathbf{w} = \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{d} \qquad (52)$$

Note that the matrix can be assumed in nontrivial cases to be nonsingular since $N < M$. Let us substitute for the SVD.

$$\begin{aligned}\mathbf{w} &= \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{d} && (53)\\ &= \mathbf{V}\begin{pmatrix}\boldsymbol{\Sigma}\\ \mathbf{0}\end{pmatrix}\mathbf{U}^H\left[\mathbf{U}\begin{pmatrix}\boldsymbol{\Sigma} & \mathbf{0}\end{pmatrix}\mathbf{V}^H\mathbf{V}\begin{pmatrix}\boldsymbol{\Sigma}\\ \mathbf{0}\end{pmatrix}\mathbf{U}^H\right]^{-1}\mathbf{d} && (54)\\ &= \mathbf{V}\begin{pmatrix}\boldsymbol{\Sigma}\\ \mathbf{0}\end{pmatrix}\mathbf{U}^H\left[\mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^H\right]^{-1}\mathbf{d} && (55)\\ &= \mathbf{V}\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\\ \mathbf{0}\end{pmatrix}\mathbf{U}^H\mathbf{d} && (56)\\ &= \mathbf{A}^\dagger\mathbf{d} && (57)\\ &= \begin{pmatrix}\mathbf{V}_1\\ \mathbf{V}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\\ \mathbf{0}\end{pmatrix}\mathbf{U}^H\mathbf{d} && (58)\\ &= \mathbf{V}_1\boldsymbol{\Sigma}^{-1}\mathbf{U}^H\mathbf{d} && (59)\end{aligned}$$

The message is the same: use the SVD.

2.5 Summary: Applying the Pseudo-Inverse

From (45) and (57) it is clear that the SVD – specifically the pseudo-inverse – can be used to solve both the overdetermined and underdetermined cases: It is always a safe choice. Perhaps more important[3], even if the rows of a short / fat **A** or the columns of a tall / skinny **A** are linearly dependent, the pseudo-inverse works fine. The only significant difference is that some of the elements of $\boldsymbol{\Sigma}$ are zero, and when the pseudo-inverse is formed these remain zero when $\boldsymbol{\Sigma}^{-1}$ is formed. Note that (47) and (59) are not mathematically necessary to include, but computationally they can save effort.

[3]We haven't shown this here because it is messy and irritating, but it it trivial to do.

3 The Normalized LMS Adaptive Filter

This is a nice twist on the LMS that uses the theory we've learnt about the SVD. Suppose we want to make a change in $\mathbf{w}_n \to \mathbf{w}_{n+1}$ such that

$$\mathbf{w}_{n+1}^H \mathbf{u}_n = d[n] \tag{60}$$

meaning that the filter error *would* have been zero if the filter had been clairvoyant enough to see $\mathbf{u}_{n+1}$ before it happened. To some extent this seems like making a "rear-view mirror" change. However, the intuition seems solid: it would appear that the filter is moving in the right direction by such a move. Now the concern is that (60) is *too easy*: $\mathbf{w}_{n+1}$ is a vector with M elements, and we are offering only a rank-one constraint by(60).

Let us define

$$\delta_{n+1} \equiv \mathbf{w}_{n+1} - \mathbf{w}_n \tag{61}$$

Inserting this to (60) gives us

$$(\delta_{n+1} + \mathbf{w}_n)^H \mathbf{u}_n = d[n] \tag{62}$$

or

$$\delta_{n+1}^H \mathbf{u}_n = e[n] \tag{63}$$
$$\mathbf{u}_n^H \delta_{n+1} = e[n]^* \tag{64}$$

where $e[n]$ is the true (not clairvoyant) filter error. This (63) is really a restatement of (60), but it allows us to see that this is really an underdetermined system, albeit one that is *very* underdetermined down to $N = 1$. If we were to use the pseudo-inverse to "solve" (63) we would find the solution that minimizes $||\delta_{n+1}||$ – and this seems like a reasonable thing to do.

Applying the SVD we have according to (64) $\mathbf{u}_n^H$ taking the role "$\mathbf{A}$"; $N = 1$ and M is the length of the filter tap-weight vector. Since $\mathbf{U}$ and $\mathbf{V}$ are matrices of eigenvectors (unitary matrices, meaning both orthogonal and normalized) the SVD is

$$\mathbf{U} = 1 \text{ a scalar} \tag{65}$$
$$\mathbf{\Sigma} = \begin{pmatrix} ||\mathbf{u}_n|| & 0 & 0 & \ldots & 0 \end{pmatrix} \text{ a row vector with } (M-1) \text{ zeros} \tag{66}$$
$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{pmatrix} \text{ where } \mathbf{V}_1 \text{ is a column-vector} \tag{67}$$
$$\mathbf{V}_1 = \frac{\mathbf{u}_n}{||\mathbf{u}_n||} \tag{68}$$

and to be clear: $||\mathbf{x}|| \equiv \sqrt{\mathbf{x}^H\mathbf{x}}$ defines the norm. Clearly $\mathbf{V}$ is $M \times M$; but only the first column is important. Applying the pseudo-inverse, then, we have from (59)

$$\delta_{n+1} = \mathbf{V}_1\mathbf{\Sigma}^{-1}\mathbf{U}^H\mathbf{e}^* \tag{69}$$

$$= \frac{\mathbf{u}_n}{||\mathbf{u}_n||^2}e[n]^* \tag{70}$$

which means

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \frac{1}{||\mathbf{u}_n||^2}\mathbf{u}_n e[n]^* \tag{71}$$

which for obvious reasons is called the *normalized* LMS (NLMS) update. Notice that (71) is very much like the usual LMS filter update, except that μ is replaced by $\frac{1}{||\mathbf{u}_n||^2}$. One can see that the NLMS update is in a sense more robust than LMS: a large $\mathbf{u}_n$ can force the LMS tap-weight vector $\mathbf{w}_{n+1}$ to make a large step. If that large $\mathbf{u}_n$ were really just an outlying sample (something non-Gaussian, say) then it is doubly-harmful to LMS: both $\mathbf{u}_n$ and $e[n]$ will be large. On the other hand, NLMS de-weights large $\mathbf{u}_n$'s, and that is in a practical sense quite appealing. It is also appealing that there is no need to study convergence to make suggestions for μ, as we had to do with LMS: the step-size is given. The text devotes much time to convergence nonetheless, and that is useful if inserted to a real application.

A concern that is raised in the text actually relates to the opposite of the robustness issues: what happens when $\mathbf{u}_n$ is very small? It is easy to see that the update then can be large. The proposal is rather a bandage:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \left(\frac{\tilde{\mu}}{\delta + ||\mathbf{u}_n||^2}\right)\mathbf{u}_n e[n]^* \tag{72}$$

The result is a far less beautiful algorithm. But it is probably quite practical.

4 Low-Rank Matrix Approximation

The Frobenius norm for a matrix is a logical extension of the vector L_2-norm to matrices:

$$||A||_F^2 \equiv \sum_{n=1}^{N}\sum_{m=1}^{M}|A_{n,m}|^2 \tag{73}$$

meaning that it is the sum of (magnitude-) squares of all the elements. An equivalent way to express the Frobenius norm is

$$||A||_F^2 = Tr\left(\mathbf{A}^H\mathbf{A}\right) \tag{74}$$

If we apply the SVD of $\mathbf{A}$ we get

$$\begin{aligned} ||A||_F^2 &= Tr\left(\mathbf{V\Sigma U}^H\mathbf{U\Sigma V}^H\right) && (75) \\ &= Tr\left(\mathbf{V\Sigma}^2\mathbf{V}^H\right) && (76) \\ &= Tr\left(\mathbf{V}^H\mathbf{V\Sigma}^2\right) && (77) \\ &= Tr\left(\mathbf{\Sigma}^2\right) && (78) \\ &= \sum_{i=1}^{\min\{M,N\}} \sigma_i^2(\mathbf{A}) && (79) \end{aligned}$$

that is, the Frobenius norm is the sum of squares of the singular values.

The low-rank approximation problem is to find $\hat{\mathbf{A}}_o$ to minimize

$$\hat{\mathbf{A}}_o = \arg\min_{\hat{\mathbf{A}}}\{||\mathbf{A}-\hat{\mathbf{A}}||_F^2\} \qquad (80)$$

with the constraint that the rank of $\hat{\mathbf{A}}_o$ is $R < \min\{M,N\}$. Let us assume that the the singular values of $\mathbf{A}$ have been ordered such that we have

$$\sigma_1^2(\mathbf{A}) \geq \sigma_2^2(\mathbf{A}) \geq \ldots \geq \sigma_{\min\{M,N\}}^2(\mathbf{A}) \qquad (81)$$

whence it is relatively easy to see that the solution is

$$\hat{\mathbf{A}}_o = \sum_{i=1}^{R} \sigma_i(\mathbf{A})\mathbf{u}_i\mathbf{v}_i^H \qquad (82)$$

where

$$||\mathbf{A}-\hat{\mathbf{A}}_o||_F^2 = \sum_{i=R+1}^{\min\{M,N\}} \sigma_i^2(\mathbf{A}) \qquad (83)$$

That is, just choose $\hat{\mathbf{A}}_o$ to align with the space corresponding to the R largest singular values of $\mathbf{A}$.

To see this, suppose that $R=1$. We have that $\hat{\mathbf{A}} = \alpha\mathbf{bc}^H$ where $\mathbf{b}$ and $\mathbf{c}$ are unit length. Now write

$$\begin{aligned} ||\mathbf{A}-\hat{\mathbf{A}}||_F^2 &= Tr\left((\mathbf{A}-\alpha\mathbf{bc}^H)^H(\mathbf{A}-\alpha\mathbf{bc}^H)\right) && (84) \\ &= Tr\left(\mathbf{AA}^H\right) - 2\Re\left\{\alpha Tr\left(\mathbf{A}^H\mathbf{bc}^H\right)\right\} \\ &\qquad + |\alpha|^2 Tr\left(\mathbf{bc}^H\mathbf{cb}^H\right) && (85) \\ &= Tr\left(\mathbf{AA}^H\right) - 2\Re\left\{\alpha Tr\left(\mathbf{A}^H\mathbf{bc}^H\right)\right\} \end{aligned}$$

$$+\ |\alpha|^2 Tr\left(\mathbf{b}^H\mathbf{b}\mathbf{c}^H\mathbf{c}\right) \tag{86}$$

$$= Tr\left(\mathbf{A}\mathbf{A}^H\right) - 2\Re\left\{\alpha Tr\left(\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H\mathbf{b}\mathbf{c}^H\right)\right\} + |\alpha|^2 \tag{87}$$

$$= \sum_{i=1}^{\min\{M,N\}} \sigma_i^2(\mathbf{A}) - 2\Re\left\{\alpha Tr\left(\mathbf{c}^H\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^H\mathbf{b}\right)\right\}$$

$$+\ |\alpha|^2 \tag{88}$$

Neither $\mathbf{c}^H\mathbf{V}$ nor $\mathbf{U}^H\mathbf{b}$ can be larger than unity in magnitude. They are maximized when $\mathbf{c}$ and $\mathbf{b}$ are aligned to columns in $\mathbf{V}$ and $\mathbf{U}$, respectively. And the middle term is maximized when aligned to the maximum singular value, yielding

$$||\mathbf{A} - \hat{\mathbf{A}}||_F^2 = ||\mathbf{A}||_F^2 - \sigma_1^2(\mathbf{A}) \tag{89}$$

We can continue the process with succeeding rank-one matrices to ascertain (82) and (83).

ECE 6123
Advanced Signal Processing:
Markov Chain Monte Carlo

Peter Willett

Fall 2017

1 Importance Sampling

1.1 Estimation of Small Probabilities

Suppose we want to estimate a small probability

$$\alpha \equiv Pr(x \in \Omega) \tag{1}$$

This may sound trivial, and it would be if we were interested, say, in the $\Omega = \{x : x > \tau\}$. But suppose it is not so simple, and Ω is the set of noise samples[1] that produces an error in an OFDM system with LDPC, zero-forcing equalization and carrier-offset recovery. We have no hope of an analytic probability calculation, all we can do is simulate and count the errors. That is, we estimate

$$\hat{\alpha} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}(x_i \in \Omega) \tag{2}$$

where $\mathcal{I}$ is the indicator, N is the number of Monte Carlo trials, these indexed by i. It is vary simple to see that

$$\begin{aligned} \mathcal{E}\{\hat{\alpha}\} &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{E}\{\mathcal{I}(x_i \in \Omega)\} && (3)\\ &= \int_{\Omega} p(x)dx = \alpha && (4) \end{aligned}$$

which is good news, but

$$\frac{Var\{\hat{\alpha}\}}{(\mathcal{E}\{\hat{\alpha}\})^2} \approx \frac{\alpha/N}{\alpha^2} = \frac{1}{N\alpha} \tag{5}$$

[1]Don't worry if this OFDM stuff means nothing to you; the point is that it's a complicated event.

which is not good news. Equation (5) means that if you want the standard deviation of $\hat{\alpha}$ to be (say) less than 10% of its value, you need $N > 100/\alpha$ MC trials; and if α is 10^{-8} this can be a chore.

Fortunately we have *importance sampling* to help. Consider a new estimator

$$\hat{\alpha} \;=\; \sum_{i=1}^{N} \mathcal{I}(x_i \in \Omega)\frac{p(x_i)}{q(x_i)} \tag{6}$$

where $p(\cdot)$ is the true probability density governing whatever is random about your problem and $q(\cdot)$ some other "importance" pdf that the samples used to estimate $\hat{\alpha}$ are actually drawn from. For example, we might have $\Omega = \{(x_1, x_2) : (x_1 - 10)^2 + (x_2 - 15)^2 \leq 1\}$ and $p(\cdot)$ bivariate Gaussian with mean zero and unity variance. It is fairly clear that $\hat{\alpha}$ from (2) will include exactly no indicators that "happen" for any reasonable value of N – it is useless. But suppose we use (6) with $q(\cdot)$ to mean a Gaussian pdf with mean $(10, 15)$ and variance of 0.5: then many indicators will fire, and each of them will force the inclusion to the sum in (6) of many relatively small values determined by the *importance weights* $p(x_i)/q(x_i)$.

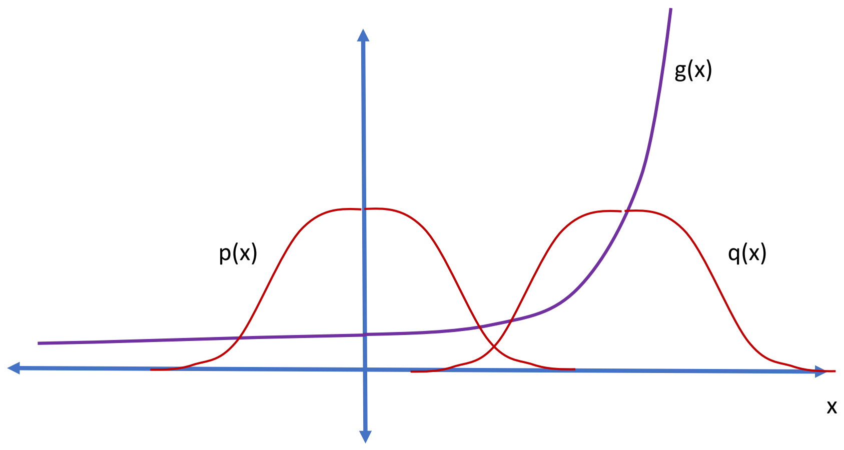
The variance of (6) is easily seen to be

$$Var(\hat{\alpha}) \;=\; \frac{1}{N}\left(\int_{\Omega} \frac{p(x)^2}{q(x)}dx - \alpha^2\right) \tag{7}$$

which is illuminating for two reasons. The first is that it is minimized (actually cut down to zero) by

$$q(x) \;=\; p(x|x \in \Omega) \;=\; \frac{p(x)\mathcal{I}(x \in \Omega)}{\alpha} \tag{8}$$

which gives us the helpful information that if we already knew the answer we could easily use MC techniques to find the answer. This actually really *is* useful, since it tells us that there is no "magic bullet" for importance-sampling – choosing a good $q(\cdot)$ is an art form. But (7) also suggests intuition: if we want to have a low variance we should try to reduce the variation of $p(\cdot)/q(\cdot)$ of Ω as much as we can. By that logic choosing $q(\cdot)$ to have mean $(9, 14)$ and unity variance in the previous example may be better than the $q(\cdot)$ given; and mean mean $(11, 16)$ worse.



1.2 Importance Sampling for Moments

Consider the situation as above, in which we wish to calculate the expected value of a function $g(x)$ under the pdf $p(x)$. A direct MC implementation will probably not work very well, since the "active" part of $g(x)$ occurs where samples from $p(x)$ are rare. Suppose instead we simulate under $q(x)$ as also indicated in the plot. Then we get

$$\bar{g} \;=\; \frac{1}{N}\sum_{i=1}^{N} g(x_i)\frac{p(x_i)}{q(x_i)} \tag{9}$$

so that

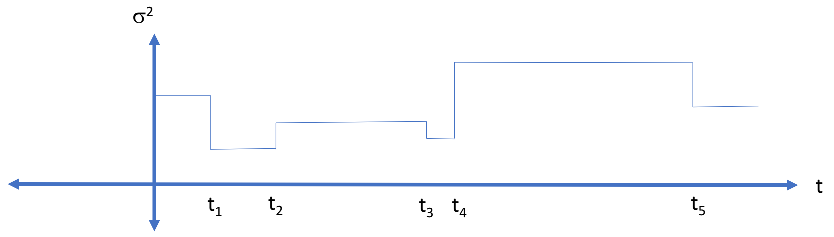
$$\mathcal{E}\{\bar{g}(x)\} \;=\; \int g(x)\frac{p(x)}{q(x)}q(x)dx \;=\; \mathcal{E}\{g(x)\} \tag{10}$$

meaning that the importance-sampling estimator is unbiased for this case, too.

2 Motivation for Markov Chain Monte Carlo

2.1 Segmentation

Consider the problem that we are given a record of N data: $\{u[n]\}$ is zero mean, independent and Gaussian. There are M segments to the data, such that if $t_{i-i} \leq n < t_i$ then the variance of $u[n]$ is σ_i^2 – see below. The problem is that we don't know the t_i's (but naturally assume $t_0 = 0$ and $t_M = N-1$) and we don't know the σ's. What do we do?



Suppose we *did* know the t_i's. Then solving for σ_i is fairly simple:

$$\hat{\sigma}_i^2 \ = \ \frac{1}{t_i - t_{i-1}} \sum_{n=t_{i-1}}^{t_i - 1} u[n]^2 \tag{11}$$

is the maximum-likelihood estimate (MLE). But finding the t_i's is more of a problem.

Define $\mathbf{t}_{\bar{i}} \equiv \{t_0, \ldots, t_{i-1}, t_{i+1}, \ldots, t_M\}$ and assume the prior information is (on $\mathbf{t}$, say) is uniform. We write

$$\begin{aligned}
p(t_i|\mathbf{t}_{\bar{k}}, \mathbf{u}) &= \frac{p(\mathbf{t}|\mathbf{u})}{p(\mathbf{t}_{\bar{i}}|\mathbf{u})} && (12)\\
&\propto p(\mathbf{t}|\mathbf{u}) && (13)\\
&\propto p(\mathbf{u}|\mathbf{t}) && (14)\\
&\propto p(\{u[n]\}_{n=t_{i-1}}^{t_i-1}|\sigma_i^2) \times p(\{u[n]\}_{n=t_i}^{t_{i+1}-1}|\sigma_{i+1}^2) && (15)\\
&= \left(\prod_{n=t_{i-1}}^{t_i-1} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{u[n]^2}{2\sigma_i^2}}\right)\left(\prod_{n=t_i}^{t_{i+1}-1} \frac{1}{\sqrt{2\pi\sigma_{i+1}^2}} e^{-\frac{u[n]^2}{2\sigma_{i+1}^2}}\right) && (16)
\end{aligned}$$

where (14) follows from a assumption of uniformity on $\mathbf{t}$ and (15) from the fact that none of the other segments depends on t_k, only the one preceding and succeeding it. Note that (15) is a set of $t_{i+1} - t_{i-1} - 1$ likelihoods that can be normalized to give a probability mass function. An algorithm follows:

1. Generate some initial $t_i(0)$'s. The initial set does not matter, but a uniform spacing is probably best. Set the iteration counter $k = 1$.

2. Calculate $\{\hat{\sigma}_i^2(k)\}_{i=1}^M$ according to (11).

3. Set $i = 1$.

4. Draw $t_i(k)$ according to

$$t_i(k) \sim p(\{u[n]\}_{n=t_{i-1}(k)}^{t_i-1}|\sigma_i^2(k)) \times p(\{u[n]\}_{n=t_i}^{t_{i+1}(k-1)-1}|\sigma_{i+1}^2(k)) \tag{17}$$

   This is from (15) and is made explicit in (16).

5. Set $i \leftarrow i + 1$ and if $i < M$ go to 4.

6. Set $k \leftarrow k + 1$ and go to 2 if $k \leq K$.

Here K is the number of iterations to perform, and K_b is some number that will be elided as "burn-in" samples. At the end, estimate

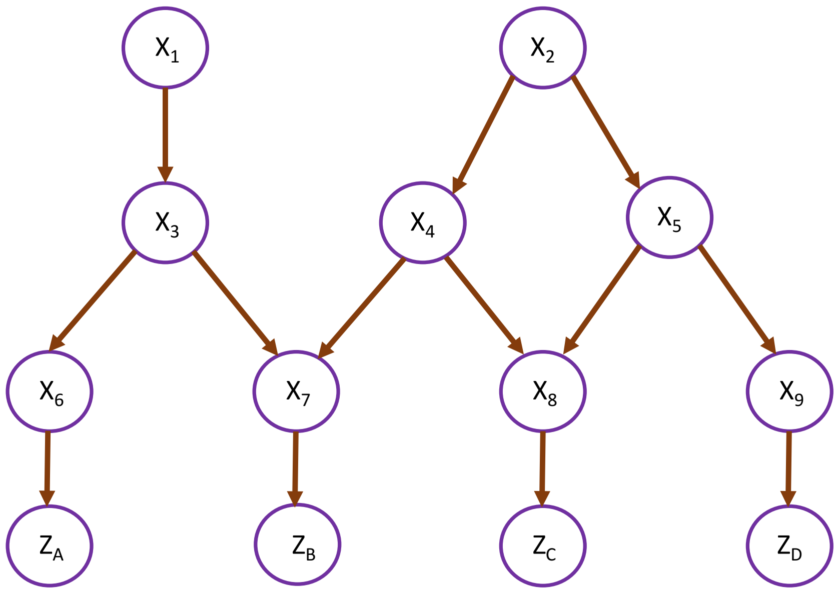
$$\hat{t}_i = \frac{1}{K - K_b} \sum_{k=K_b+1}^{K} t_i(k) \tag{18}$$

for the average. Actually we could take the variance as well to determine our posterior variance, as we shall see. But what gives us any right to do such an operation and expect any meaning at the end?

2.2 Bayesian Inference Networks

Actually the procedure just discussed[2] is *Gibbs sampling*, which is far more general. An example, perhaps the canonical one, is the Bayesian Inference Network (BIN), pictured below. The arrows indicate known conditional probabilities, which are assumed known. Each "node" x_i is a hidden state variable, and the z's are observations – and of these it is possible that only a subset is known. For example, x_2 might be a stock valuation: underpriced (0), fairly-priced (1) or overpriced (2) – clearly this is a hidden node that you are interested in. Maybe x_4 is institutional interest in the stock (yes / no); and x_1 is the company's growth potential. Observation z_a might be existence of a dividend, and z_c is the company's price-to-earnings ratio – these are both something you can observe. Finally, let's say that z_b is whether there have been buys of the stock by company insiders – this is something you might know, but might not.

[2]Actually it was not quite Gibbs sampling, since the MLE step for the σ's has no place with Gibbs.



The same approach as in the previous segmentation example can be used there. It works best if the nodes can only take on a finite number of values, but that is not necessary. Specifically, do the following:

1. Initialize the instantiated observation nodes – those z's that you know – to their true values. These will never change, of course.
2. Initialize all other nodes to random values: that is, the $x^{(0)}{}_i$'s and also the un-instantiated z's. Set $k = 1$.
3. For all (uninstantiated) nodes calculate

$$p(x_i) \;=\; \kappa \left(\prod_{j \in S_p} p(x_i | x_j^{(k-1)}) \right) \left(\prod_{j \in S_c} p(x_j^{(k-1)} | x_i) \right) \tag{19}$$

   for all possible values of x_i, where κ normalizes the sum over these to unity. The set S_p indicates the *parent* nodes of x_i and S_c the child nodes; see below for an example.

4. Draw new x_i's from the pmf's calculated in the previous step.
5. Set $k \leftarrow k + 1$ and go to 3 if $k \leq K$.

For example, for x_4 we have $S_p = \{x_2\}$ and $S_c = \{x_7, x_8\}$; that is,

$$\begin{aligned} p(x_4 | \mathbf{x}_{\bar{4}}^{(k-1)}, \mathbf{z}) &\propto p(x_4, \mathbf{x}_{\bar{4}}^{(k-1)} | \mathbf{z}) && (20) \\ &\propto p(x_4 | x_2^{(k-1)}) \times p(x_7^{(k-1)} | x_4) \times p(x_8^{(k-1)} | x_4) && (21) \end{aligned}$$

Note that although in this algorithm it seems like one generates all of the $\mathbf{x}^{(k)}$ based on $\mathbf{x}^{(k-1)}$, it is a perfectly legal algorithm that simply uses whatever the present node values might be, whether updated or not; that is, (19) may use a combination of $x_j^{(k)}$'s and $x_j^{(k-1)}$'s.

3 Metropolis-Hastings Algorithm

3.1 Theory

Consider these steps, which form the MH (meta-) algorithm:

1. Initialize x^0. Set $n = 1$.
2. Generate y according to $q(y|x^n)$.
3. Generate u uniform on $(0, 1)$.
4. Form
$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)q(x^n|y)}{\pi(x^n)q(y|x^n)}\right\} \tag{22}$$
In this step $\pi(\cdot)$ is the probability (mass function or density) of the system you are investigating.
5. If $u \leq \alpha(x^n, y)$ set $x^{n+1} = y$. Otherwise keep $x^{n+1} = x^n$.
6. Increment $n \leftarrow n + 1$. Go to 2 unless finished with iterations.

The choice of $q(\cdot|\cdot)$ is a matter of tuning. The superscript for x refers to the iteration number.

The probability $p(\cdot)$ is assumed to be available. This latter may seem strange, but in many problems the overall probability is explicit and what is sought is a marginal probability of some component. Turning the BIN previously pictured the overall probability is actually fairly simple to write. In fact it is

$$\begin{aligned}\pi(\mathbf{x}, \mathbf{z}) \;=\; & p(z_A|x_6)p(z_B|x_7)p(z_C|x_8)p(z_D|x_9) \\ & \times p(x_6|x_3)p(x_7|x_3, x_4)p(x_8|x_4, x_5)p(x_9|x_5) \\ & \times p(x_3|x_1)p(x_4|x_2)p(x_5|x_2)p(x_1)p(x_2)\end{aligned} \tag{23}$$

which is, as advertised, simple; but $p(x_4|z_A, z_C)$ is not at all simple and would involve a great deal of awkward summation. Another example is nonlinear filtering: it is (relatively) easy to write $\pi(x_1, \ldots, x_t, z_1, \ldots, z_t)$; but it is not easy to find $p(x_t|z_1, \ldots, z_t)$. These are cases in which $\pi(\cdot)$ is explicit; our goal is to generate samples of $\mathbf{x}$ (like $\{x_1, \ldots, x_t\}$) such that we can trivially investigate subsets of them (like x_t).

Let us first note that we have

$$\frac{\alpha(u,v)}{\alpha(v,u)} = \frac{\pi(v)q(u|v)}{\pi(u)q(v|u)} \tag{24}$$

or

$$\pi(u)q(v|u)\alpha(u,v) = \pi(v)q(u|v)\alpha(v,u) \tag{25}$$

since either the numerator or denominator of the LHS must have been "clamped" at unity.

Now, (22) instructs us that we have

$$\begin{aligned} p(x^{n+1}|x^n) &= \alpha(x^n, x^{n+1})q(x^{n+1}|x^n) \\ &\quad + \delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{26}$$

Multiply (26) by $\pi(x^n)$ and we get

$$\begin{aligned} \pi(x^n)p(x^{n+1}|x^n) &= \pi(x^n)\alpha(x^n, x^{n+1})q(x^{n+1}|x^n) \\ &\quad + \pi(x^n)\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{27}$$

and inserting (25) we have

$$\begin{aligned} \pi(x^n)p(x^{n+1}|x^n) &= \pi(x^{n+1})\alpha(x^{n+1}, x^n)q(x^n|x^{n+1}) \\ &\quad + \pi(x^n)\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^n, y)q(y|x^n)dy\right) \end{aligned} \tag{28}$$

The δ-function allows us to switch the terms, hence we have

$$\begin{aligned} &\pi(x^n)p(x^{n+1}|x^n) \\ &\quad = \pi(x^{n+1})\alpha(x^{n+1}, x^n)q(x^n|x^{n+1}) \\ &\qquad + \pi(x^{n+1})\delta(x^{n+1} - x^n)\left(1 - \int \alpha(x^{n+1}, y)q(y|x^{n+1})dy\right) \qquad (29) \\ &\quad = \pi(x^{n+1})p(x^n|x^{n+1}) \qquad (30) \end{aligned}$$

Equation (30) tells us that we can identify $\pi(x^n)$ and $p(x^{n+1}|x^n)$ as respectively the stationary and transition probability mass functions (or densities) of a Markov chain. Thus, discarding *burn-in* (transient) x^n's at the beginning, we know that the x^n's that we accumulate by following the given procedure are distributed according to $\pi(\cdot)$. That is, we can do things like take an average over one of the dimensions to get an expected value.

It is interesting that the *correctness* of the MH algorithm does not depend on the choice of $q(\cdot|\cdot)$. Nonetheless, the *efficiency* is strongly connected to $q(\cdot|\cdot)$: an "aggressive" $q(\cdot|\cdot)$ can cover a lot of ground, but may end up rejecting many putative samples via $\alpha(\cdot,\cdot)$; similarly a timid $q(\cdot|\cdot)$ does not waste samples but may take many iterations to explore its space.

3.2 Special Cases of Metropolis-Hastings

3.2.1 Metropolis Sampler

This is probably the original version, and uses a conditional density $q(\cdot|\cdot)$ such that

$$q(v|u) \;=\; q(u|v) \tag{31}$$

An example of such a density is the (obvious) Gaussian: under $q(v|u)$, v is Gaussian with mean u. If the Metropilis version is used we have

$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)}{\pi(x^n)}\right\} \tag{32}$$

The statements recently made about aggressive or timid $q(\cdot|\cdot)$ are very clear in light of (32).

3.2.2 Independence Sampler

The independence sampler uses

$$q(v|u) \;=\; q(v) \tag{33}$$

and hence

$$\alpha(x^n, y) \;=\; \min\left\{1, \frac{\pi(y)q(x^n)}{\pi(x^n)q(y)}\right\} \tag{34}$$

Equation (33) means that its mode of exploration does not depend on its current knowledge at all – the second pair of MC's in MCMC are suspect. But it works, and can be thought of as a way to understand the celebrated "bootstrap" particle filter.

3.2.3 Gibbs Sampler

This one is slightly trickier to understand. The key is to acknowledge that x^n and y are actually multi-dimensional. Let us use y_i to refer to the i^{th} dimension of y; and $y_{\bar{i}}$ to refer to all dimensions $\underline{\text{except}}$ the i^{th}. Then according to the Gibbs idea we use

$$q(y_i|x^n_{\bar{i}}) \;=\; \pi(y_i|x^n_{\bar{i}}) \tag{35}$$

meaning that we draw a new i^{th} dimension using the *true* pdf based on all other dimensions (which are unchanged). In the BIN formulation we can see how this is accomplished: for our example, we know how to draw x_4 based on $x_{\bar{4}}$; and this amounts to (21). Now what is especially interesting about the Gibbs sampler is that we have

$$\alpha(x_{\bar{i}}^n, y_i) \;=\; \min\left\{1, \frac{\pi(x_i|x_{\bar{i}}^n)\pi(y_i|x_{\bar{i}}^n)}{\pi(y_i|x_{\bar{i}}^n)\pi(x_i|x_{\bar{i}}^n)}\right\} \;=\; 1 \tag{36}$$

which means that the Gibbs Sampler never "rejects".

4 Particle Filters

The *particle filter*, an approach to the solution of the *Chapman-Kolmogorov* equation (CKE) via a Monte Carlo (MC) method, has evolved considerably over the last years, and there are many versions. The basic idea is most easily explained using the first version that was *feasible*, known as the bootstrap filter or the *sequential importance sampling/resampling (SIR or SIS).*

Consider the general Markov chain "target" pdf model, which may be nonlinear and/or non-Gaussian,

$$p(x_{[1:t]}) = p(x_1)\prod_{k=2}^{t} p(x_k|x_{k-1}) \tag{37}$$

in which x_k is the n_x-dimensional target state at time k,

$$x_{[1:t]} \equiv \{x_1, x_2, \dots, x_t\} \tag{38}$$

and $p(x_k|x_{k-1})$ is the *state transition pdf.* The assumption of white process noise is what allows one to write (37) in the product form. As usual, the measurement sequence must be – conditioned on the state – independent (i.e., white measurement noise), in which case

$$p(z_{[1:t]}|x_{[1:t]}) = \prod_{k=1}^{t} p(z_k|x_k) \tag{39}$$

which again is a fairly arbitrary collection of conditionally-independent pdfs.

To develop the idea of the bootstrap filter, consider the following. Using the Monte Carlo method, N samples $\{x_{[1:t]}^i\}_{i=1}^N$ — *multiscan particles*, which are $n_x t$ vectors — are drawn from the prior density function $p(x_{[1:t]})$ shown in (37) Then the likelihood of each such sample is evaluated according to $p(z_{[1:t]}|x_{[1:t]})$ in (39). We can normalize these likelihoods to "weights"

$$\omega^i_{[1:t]} = \frac{1}{c} p(z_{[1:t]}|x^i_{[1:t]}) \qquad i = 1, \ldots, N \tag{40}$$

where c is chosen such that $\sum_{i=1}^N \omega^i_{[1:t]} = 1$, i.e.,

$$c \equiv \sum_{i=1}^N p(z_{[1:t]}|x^i_{[1:t]}) \tag{41}$$

The weights in (40) are probabilities, assuming equal priors for the samples $\{x^i_{[1:t]}\}_{i=1}^N$. Then one can claim that the desired posterior $p(x_{[1:t]}|z_{[1:t]})$ is well represented by the point-mass pdf (or pmf)

$$\hat{p}[x_{[1:t]}|z_{[1:t]}] = \sum_{i=1}^N \omega^i_{[1:t]} \delta(x_{[1:t]} - x^i_{[1:t]}) \tag{42}$$

assuming N, the number of particles, is sufficiently large.

That the representation is reasonable for the posterior mean

$$\begin{aligned} \hat{x}_{[1:t]} &\equiv \int x_{[1:t]} \hat{p}(x_{[1:t]}|z_{[1:t]}) dx_{[1:t]} \\ &= \int \sum_{i=1}^N \omega^i_{[1:t]} x_{[1:t]} \delta(x_{[1:t]} - x^i_{[1:t]}) dx_{[1:t]} \\ &= \sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]} \end{aligned} \tag{43}$$

is straightforward to see as follows. Consider the expected value of (43) over the samples $\{x^i_{[1:t]}\}_{i=1}^N$

$$\begin{aligned} E\left\{\hat{x}_{[1:t]}|z_{[1:t]}\right\} &= E\left\{\sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]}\right\} \\ &= \int \sum_{i=1}^N \omega^i_{[1:t]} x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \\ &= \int \sum_{i=1}^N \frac{1}{c} p(z_{[1:t]}|x^i_{[1:t]}) x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \\ &\approx \int \sum_{i=1}^N \frac{1}{N p(z_{[1:t]})} p(z_{[1:t]}|x^i_{[1:t]}) x^i_{[1:t]} p(x^i_{[1:t]}) dx^i_{[1:t]} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N}\sum_{i=1}^{N}\int x^i_{[1:t]} p(x^i_{[1:t]}|z_{[1:t]})dx^i_{[1:t]} \\
&= E\left\{x_{[1:t]}|z_{[1:t]}\right\} \qquad (44)
\end{aligned}
$$

The approximation in (44) is that

$$
\begin{aligned}
p(z_{[1:t]}) &= \int p(z_{[1:t]}|x_{[1:t]})p(x_{[1:t]})dx_{[1:t]} \\
&\approx \int p(z_{[1:t]}|x_{[1:t]})\sum_{i=1}^{N}\frac{1}{N}\delta(x_{[1:t]} - x^i_{[1:t]})dx_{[1:t]} \\
&= \frac{1}{N}\sum_{i=1}^{N} p(z_{[1:t]}|x^i_{[1:t]}) = \frac{c}{N} \qquad (45)
\end{aligned}
$$

The above forms the basis for particle filter-based track-likelihood evaluation and track-testing

$$
\begin{aligned}
p(z_{[1:t]}) &= p(z(t)|z_{[1:t-1]})p(z_{[1:t-1]}) \\
&= p(z_{[1:t-1]})\int p(z_t|x_t)p(x_t|z_{[1:t-1]})dx_t \\
&\approx p(z_{[1:t-1]})\int p(z_t|x_t]\frac{1}{N}\sum_{i=1}^{N}\delta[x_t - x^i_t]dx_t \\
&= p(z_{[1:t-1]})\frac{1}{N}\sum_{i=1}^{N} p(z_t|x^i_t) \\
&= \prod_{k=1}^{t}\left(\frac{1}{N}\sum_{i=1}^{N} p(z_k|x^i_k)\right) \qquad (46)
\end{aligned}
$$

similar to the χ^2 statistic that one might use in a Kalman filter context, but for more general models. The track likelihood (46) is recognizable as the product of *unnormalized* particle weights (which have to be calculated anyway), and certainly the second formula in (46) shows that it can be evaluated iteratively.

Consequently an efficient means to generate $\{x^i_{[1:t]}\}_{i=1}^N$ — recall that for an n_x-dimensional state, each $x^i_{[1:t]}$ is $n_x t$ dimensional — is key. The steps are as follows:

Prediction: Given $x^i_{[1:t-1]}$ we draw x^i_t from it according to

$$
x^i_t \sim p(x^i_t|x^i_{t-1}) \qquad (47)
$$

and thence augment $x^i_{[1:t-1]}$ to $x^i_{[1:t]}$.

Update: Calculate

$$\begin{aligned}\omega^i_{[1:t]} &= \frac{1}{c}p(z_{[1:t]}|x^i_{[1:t]}) \\ &= \frac{1}{c'}p(z_t|x^i_t)\omega^i_{[1:t-1]} \qquad (48)\end{aligned}$$

The last line above will, in general, require a new normalization.

Note that (47) and (48) imply that it is not necessary to work with $n_x t$-dimensional particles $\{x^i_{[1:t]}\}_{i=1}^N$ and weights $\{\omega^i_{[1:t]}\}_{i=1}^N$. Instead, as a practical matter all that one need retain is their value at time t: $\{x^i_t\}_{i=1}^N$ and weights $\{\omega^i(t)\}_{i=1}^N$, which may be taken as a statement that $\{x^i_t\}_{i=1}^N$ and $\{\omega^i(t)\}_{i=1}^N$ are sufficient for $x(t)$ given observations $\{z_k\}_{k=1}^t$. With reference to (43), any moment of $x(t)$ can be approximated from these alone.

To be specific about these steps, if one were to use a particle filter to estimate in a linear/Gaussian situation (which one never would — one would use a Kalman filter) one would *predict* by drawing the i^{th} particle at time t as $x^i_t = Fx^i_{t-1} + v^i_t$, where v^i_t is simply the realization of a Gaussian random vector with covariance Q. The weight for the i^{th} particle is the likelihood

$$\omega^i(t) = \frac{1}{c}\frac{1}{\sqrt{|2\pi R|}}e^{-\frac{1}{2}[z_t - Hx^i_t]' R^{-1}[z_t - Hx^i_t]} \qquad (49)$$

with appropriate normalization.

Unfortunately there is a problem — *particle degeneracy* — namely, the tendency for all the weights save one to go to zero. This tendency — really a compulsion in any nontrivial case — arises from the product in (48), and it severely limited the acceptance of Monte Carlo approaches in their early days. *Fortunately* there is a solution: *resampling.* The especially easy resampling in the bootstrap filter is to sample *with replacement* from $\{x^i_t\}_{i=1}^N$ according to probabilities $\{\omega^i(t)\}_{i=1}^N$. We note that in a mathematical sense this sampling is optional, and in some implementations it is performed only when degeneracy seems to be occurring, not necessarily at every update step. It is important to realize that such resampling can (and usually does) result in many repeated (i.e., copied) particles, those corresponding to the largest likelihoods (the largest $\omega^i(t)$'s). This is an evanescent concern, since the next prediction step in (47) adds different "noise" to each of these.

We therefore restate the operation of the bootstrap filter as

Prediction: For $i = 1, \ldots, N$ draw $\tilde{x}^i_t$ from x^i_{t-1} according to

$$\tilde{x}^i_t \sim p(x^i_t|x^i_{t-1}) \qquad (50)$$

Update: For $i = 1, \ldots, N$ calculate and normalize to unity-sum the weights

$$\omega^i(t) = \frac{1}{c} p(z_t | \tilde{x}_t^i) \tag{51}$$

Resampling: Draw $\{x_t^i\}_{i=1}^N$ from $\{\tilde{x}_t^i\}_{i=1}^N$ according to the pmf $\{\omega^i(t)\}_{i=1}^N$.

This shows that multiscan particles, while helpful to intuition, are not part of a practical particle filter system. Note that the fact that the resampling operation at $t-1$ used $\{\omega^i(t-1)\}_{i=1}^N$, makes it inappropriate to use these weights again in the update step for $\{\omega^i(t)\}_{i=1}^N$: only the present likelihoods are used.

One interesting variation on the particle filter is having $x^i(t)$ drawn according to some alternative *proposal density* $q(x_t^i | x_{t-1}^i)$ for the prediction instead of the prior (or transition) density $p(x_t^i | x_{t-1}^i)$. Then this "incorrect" prediction step can be exactly canceled in the typical importance sampling manner by using the appropriate importance weight. That is, the three particle filtering steps become

Importance-Weighted Prediction: Draw $\tilde{x}_t^i$ from x_{t-1}^i, according to

$$\tilde{x}^i(t) \sim q(\tilde{x}_t^i | x_{t-1}^i) \tag{52}$$

Importance-Weighted Update: Calculate

$$\omega_{[1:t]}^i = \frac{1}{c} p(z_t | \tilde{x}_t^i) \frac{p(\tilde{x}_t^i | x_{t-1}^i)}{q(\tilde{x}_t^i | x_{t-1}^i)} \tag{53}$$

Resampling: Draw $\{x_t^i\}_{i=1}^N$ from $\{\tilde{x}_t^i\}_{i=1}^N$ via the pmf $\{\omega^i(t)\}_{i=1}^N$.

The resampling step is unaffected. That (53) is appropriate, is easily checked by a development similar to (44)

A properly chosen $q(\cdot|\cdot)$ can "steer" particles to be predicted to places where they are likely to be corroborated by measurements, namely, (52) becomes

$$\tilde{x}_t^i \sim q(\tilde{x}_t^i | x_{t-1}^i; z_t) \tag{54}$$

This is as opposed to the standard procedure (47) and (48) where many predicted particles can be, in effect, wasted with essentially-zero weights, leaving relatively few (or no!) "working" particles near where the measurement makes them likely. Common choices for the proposal density include the transition density with inflated noise, the EKF and UKF.

The key step – which must be credited to Gordon, Salmond & Smith – in making particle filters practical was resampling and that estimation is performed recursively in time, as opposed to the batch approach with the notional "multiscan particles" discussed at the beginning of this section. In addition to proposal density improvements there have been many interesting variations. One of the most widely accepted is the auxiliary particle filter that directly uses the measurement at time t to guide the proposal density selection. It is worth noting that success in dealing with data association problems has proven elusive to particle filters — this is perhaps despite the interpretation of measurement origin uncertainty as a form of non-Gaussian noise. Nonetheless, multiple model systems (of the sort for which the IMM would be appropriate) can be treated using the Rao-Blackwellization policy, which splits the inference task into parts that are "easy" to solve (like filtering with a known mode sequence) relegated to quick algorithms, with more difficult ones (like mode estimation) that are assigned to particles.

5 The Ensemble Kalman Filter

The EnKF is only really of interest in problems that are extremely large-scale in their state space, as happens in geophysics and meteorology, for example. If the state space has dimension of several thousand (or more!) we might decide to implement a Kalman filter in parallel Monte Carlo form. That is, many processors each are given responsibility for a few particles – and the total number of particles across all processors may be less than the state dimension. It is typical to specify

$$\begin{aligned}\text{dimension of observations } (m) &\ll \text{number of particles } (N)\\ &\ll \text{dimension of state } (n) \qquad (55)\end{aligned}$$

for an EnKF to be useful.

Now, consider we have a collection of particles $\{\mathbf{x}^i_{t-1|t-1}\}_{i=1}^N$ that represent the state at time $t-1$, and we sample them forward[3] to $\{\mathbf{x}^i_{t|t-1}\}_{i=1}^N$ according to

$$\mathbf{x}^i_{t|t-1} \;=\; \mathbf{F}\mathbf{x}^i_{t-1|t-1} \;+\; \mathbf{v}^i_t \qquad (56)$$

where $\mathbf{v}^i_t$ are *iid* $\mathcal{N}(0,\mathbf{Q})$. The natural particle-filter next step is to weight each particle by the observation likelihood

$$p(\mathbf{z}_t|\mathbf{x}^i_{t|t-1}) \;=\; \mathcal{N}(\mathbf{H}\mathbf{x}^i_{t|t-1},\mathbf{R}) \qquad (57)$$

As we have discussed earlier, this will require resampling in order to work well, and resampling is not well suited to parallel implementation.

[3]This is obvious: coast and add noise.

Another approach is to take the ensemble covariance

$$\hat{\mathbf{Q}} \;=\; \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}^i_{t|t-1})(\mathbf{x}^i_{t|t-1})^T \;-\; \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^i_{t|t-1}\right)^2 \tag{58}$$

and thence compute the "optimal" posterior ensemble of particles

$$\mathbf{x}^i_{t|t} \;=\; \mathbf{x}^i_{t|t-1} \;+\; \hat{\mathbf{Q}}\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{Q}}\mathbf{H}^T+\mathbf{R}\right)^{-1}\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right) \tag{59}$$

Unfortunately all the resulting particles are correlated, since all are the result of the same measurement noise that gave rise to $\mathbf{z}_t$. That is,

$$\begin{aligned}
&\mathcal{E}\left\{\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right)\left(\mathbf{z}_t-\mathbf{H}\mathbf{x}^j_{t|t-1}\right)^T\right\} && (60)\\
&=\; \mathcal{E}\Big\{\Big((\mathbf{z}_t-\mathbf{H}\mathbf{x}_t)-(\mathbf{H}\mathbf{x}^i_{t|t-1}-\mathbf{H}\mathbf{x}_t)\Big) && \\
&\qquad\times\Big((\mathbf{z}_t-\mathbf{H}\mathbf{x}_t)-(\mathbf{H}\mathbf{x}^j_{t|t-1}-\mathbf{H}\mathbf{x}_t)\Big)^T\Big\} && (61)\\
&=\; \mathbf{R} && (62)
\end{aligned}$$

and this is just wrong.

The EnKF idea is to replace (59) by

$$\mathbf{x}^i_{t|t} \;=\; \mathbf{x}^i_{t|t-1} \;+\; \hat{\mathbf{Q}}\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{Q}}\mathbf{H}^T+\mathbf{R}\right)^{-1}\left(\mathbf{z}^i_t-\mathbf{H}\mathbf{x}^i_{t|t-1}\right) \tag{63}$$

where

$$\mathbf{z}^i_t \;=\; \mathbf{z}_t \;+\; \mathbf{w}^i_t \tag{64}$$

and $\mathbf{w}^i_t$ are *iid* $\mathcal{N}(0,\mathbf{R})$, meaning that we actually add noise[4] to the measurement before doing the Kalman update: each particle $\mathbf{x}^i_{t|t-1}$ is updated via its own noisy measurement $\mathbf{z}^i_t$. The *ensemble* posterior covariance is obviously correct, too.

The implementation of the EnKF uses, instead of (58),

$$\hat{\mathbf{Q}} \;=\; \frac{1}{N-1}\mathbf{A}_t\mathbf{A}_t^T \tag{65}$$

where

$$\mathbf{A}_t \;\equiv\; \begin{pmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ (\mathbf{x}^1_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) & (\mathbf{x}^2_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) & \cdots & (\mathbf{x}^N_{t|t-1}-\bar{\mathbf{x}}_{t|t-1}) \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix} \tag{66}$$

[4]This is the right thing to do. But I am still emotionally offended by the idea that the right thing to do is to add noise. It almost implies that the Kalman filter is not optimal.

and we also define

$$\mathbf{D} \equiv \begin{pmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \mathbf{z}_t^1 - \mathbf{z}_t & \mathbf{z}_t^2 - \mathbf{z}_t & \dots & \mathbf{z}_t^N - \mathbf{z}_t \\ \downarrow & \downarrow & \dots & \downarrow \end{pmatrix} \tag{67}$$

for the observation that has this artificial noise added. Note that the expected value of the particles has to be calculated, but this is simple since it is just the mean of the local means at each of the parallel processing units. Now (63) becomes

$$\mathbf{A}_t = \mathbf{A}_{t-1} + \mathbf{A}_{t-1}(\mathbf{H}\mathbf{A}_{t-1})^T \Big((\mathbf{H}\mathbf{A}_{t-1})(\mathbf{H}\mathbf{A}_{t-1})^T + \mathbf{R} \Big)^{-1} (\mathbf{D} - (\mathbf{H}\mathbf{A}_{t-1})) \tag{68}$$

which implies that each particle can be updated in relative isolation, with the exception that the $m \times m$ covariance matrix must be calculated and inverted. Note that $(\mathbf{H}\mathbf{A}_{t-1})$ has dimension $m \times N$ – no need for a large matrix computation.

ECE 6123
Advanced Signal Processing: Multi-Rate Signal Processing, Multi-Resolution Decomposition and Wavelets

Peter Willett

Fall 2017

1 Decimation and Interpolation

1.1 Decimation

This is a review from basic Digital Signal Processing, but please bear with it. First, consider the *decimation* operation

$$y[n] \;=\; x[nD] \tag{1}$$

in which D is some integer – say, for $D = 2$ this amounts to forming $y[n]$ from the even-indexed samples of $x[n]$. The transform relationship comes from doing this in two steps:

$$y[n] = u[nD] \quad \text{where} \quad u[n] \;=\; \begin{cases} x[n] & n = mD \\ 0 & \text{else} \end{cases} \tag{2}$$

We begin with the *noble identity*

$$\sum_{p=0}^{N-1} e^{j2\pi pm/N} \;=\; N \sum_{q=-\infty}^{\infty} \delta[m - qN] \tag{3}$$

which is easy to show via the geometric series formula (special case of summing N 1's when $m = qN$).

Now we write

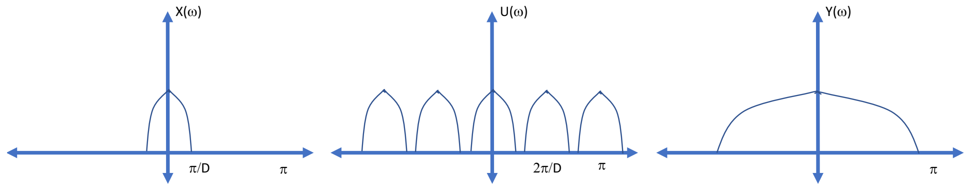
$$u[n] = \frac{1}{D} \sum_{k=0}^{N-1} e^{j2\pi kn/D} x[n] \tag{4}$$

so

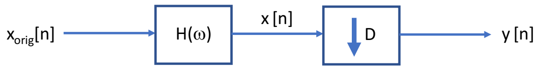
$$\begin{aligned} U(z) &= \sum_{n=-\infty}^{\infty} \frac{1}{D} \sum_{k=0}^{N-1} e^{j2\pi kn/D} x[n] z^{-n} && (5) \\ &= \frac{1}{D} \sum_{k=0}^{N-1} X(ze^{-j2\pi k/D}) && (6) \\ U(\omega) &= \frac{1}{D} \sum_{k=0}^{N-1} X\left(\omega - 2\pi k/D\right) && (7) \end{aligned}$$

It is easy to see that we have

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} u[nD] z^{-n} && (8) \\ &= U(z^{1/D}) && (9) \\ &= \frac{1}{D} \sum_{k=0}^{N-1} X(z^{1/D} e^{-j2\pi k/D}) && (10) \\ Y(\omega) &= U(\omega/D) && (11) \\ &= \frac{1}{D} \sum_{k=0}^{N-1} X\left(\frac{\omega - 2\pi k}{D}\right) && (12) \end{aligned}$$



See above for an illustration. Note that in the above figure the bandwidth of $X(\omega)$ is constrained to be less than π/D; if this were not so we would have *aliasing*. We are not interested in aliasing in the current discussion. And in any case we could pre-filter (with an "anti-aliasing" filter) the signal $x[n]$ to make sure that no frequency components above π/D remain, as in the figure below.



It is useful to note that if $h[n]$ is *finite impulse-response* (FIR) of length L then while each $y[n]$ requires L operations, that means that only L/D operations are needed per sample of $x[n]$. It is also worth mentioning that the case $D = 2$

$$\begin{aligned} Y(z) &= \frac{1}{2}\left(X(z^{1/2}) + X(-z^{1/2})\right) && (13) \\ Y(\omega) &= \frac{1}{2}\left(X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2}+\pi\right)\right) && (14) \end{aligned}$$

will especially interest us.

1.2 Interpolation

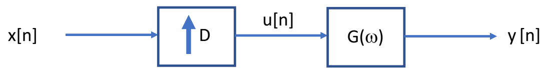
With reference to the previous discussion, the *interpolation* essentially refers to starting with $y[n]$ and re-formulating $u[n]$. Switching input to $x[n]$ we thence have

$$u[n] = \begin{cases} x[m] & n = mD \\ 0 & \text{else} \end{cases} \qquad (15)$$

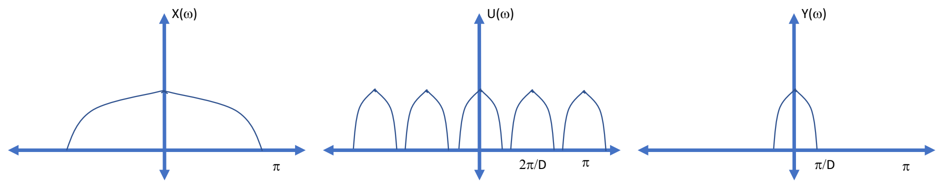
meaning that between each sample of $x[n]$ we simply insert $(D-1)$ 0's. It's obvious that we have

$$\begin{aligned} U(z) &= \sum_{n=-\infty}^{\infty} u[n]z^{-n} && (16) \\ &= \sum_{m=-\infty}^{\infty} u[m]z^{-nD} && (17) \\ &= X(z^D) && (18) \\ U(\omega) &= X(\omega D) && (19) \end{aligned}$$

The system is as shown below.



And the spectra are as shown here.

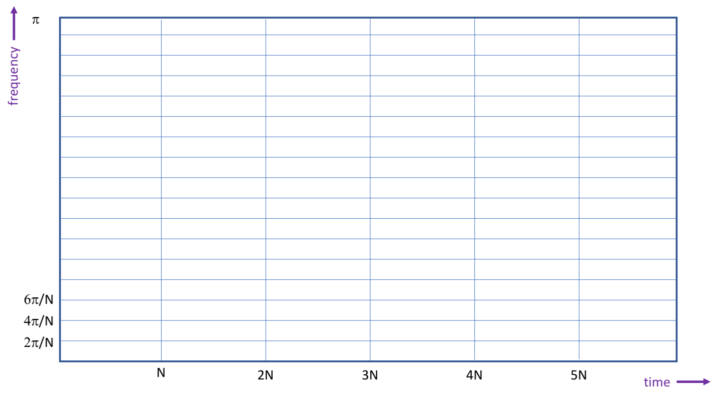


Note that replica spectra appear in $U(\omega)$; in many applications it is desirable to suppress these so we have above represented the final output $y[n]$ as being after another filter. It is commonly known as an "interpolation" filter since its function / effect is to insert smoothed values over the $(D-1)$ 0's that are in $u[n]$. Note, again, that since only every D^{th} sample of $u[n]$ is non-zero only L/D operations per output of $y[n]$ are needed for this interpolation operation.

2 Filter Banks

2.1 Transforming Data via the Block-DFT

A transformation of data that concentrates signal energy in a few samples makes for better signal understanding, representation, manipulation and coding. One such transformation is the block-DFT.



The illustration above is intended to illustrate the shape of the transformed components to the block-DFT. The block-DFT has several nice properties:

- It is invertible – no information is lost.
- It is orthogonal – if the input is white, the transformed components are white, too.
- It is efficient – via the FFT it requires only $\log_2(N)$ operations per output.

There is one disadvantage, however, and it is perhaps best illustrated in the notional plot just shown. It is this: high-frequency components correspond to features in the original signal that are of short duration. However, the time-swath of the DFT is the same for all frequencies: that is, a high-frequency component measures the amount of high-frequency energy over the entire block of N data. Since high-frequency components are by their nature fast-changing it would make more sense to have them measuring energy at those frequencies over shorter periods of time as compared to lower-frequency components that measure long-term trends and smooth features.

Now, a transformation that is invertible is really, in linear-algebraic terms, a change of basis. How is a DFT that? Consider

$$\mathbf{X} = \mathbf{W}\mathbf{x} \tag{20}$$

in which

$$\mathbf{W} \equiv \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & W_N^3 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & W_N^6 & \dots & W_N^{2(N-1)} \\ 1 & W_N^3 & W_N^6 & W_N^9 & \dots & W_N^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & W_N^{3(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{pmatrix} \tag{21}$$

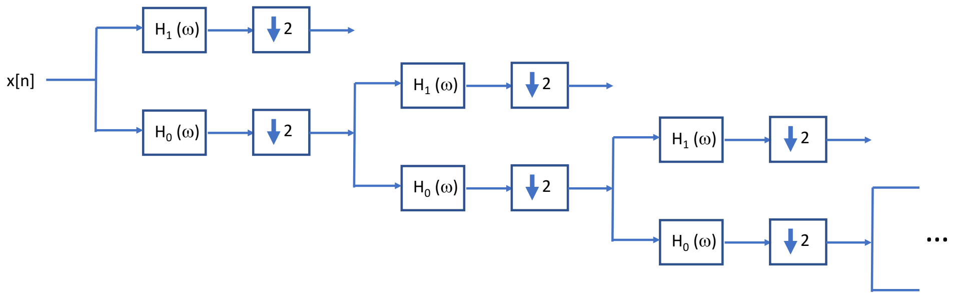
and of course

$$W_N \equiv e^{-j2\pi/N} \tag{22}$$

That is, the DFT operation is really a matrix/vector multiplication. We know that the FFT gives us an efficient way to implement it – better than the N^2 operations that a direct matrix/vector multiply would normally take – and we also know that $\mathbf{W}\mathbf{W}^H = N\mathbf{I}$, meaning that it is orthogonal. Are there other matrices that have the same properties?

2.2 Transforming Data via a Filter Bank

The question was just posed as to whether there any other transformation matrices that have the same three nice properties as the block-DFT. The answer is most certainly *yes* – see the figure below – and, even better, we have one such that does avoids the block-DFT's "disadvantage" in terms of compatibility of time-averaging to frequency. This an "octave" filter bank, and of course other decimation factors could be imagined.

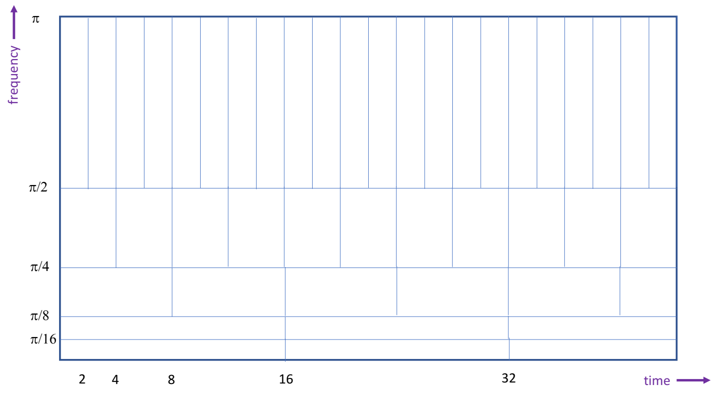


We will treat invertibility and orthogonality soon. But as for efficiency, let us assume that both filters $H_0(\omega)$ and $H_1(\omega)$ are of length L. Then the upper-most branch requires $L/2$ operations per input sample of $x[n]$ and so does the lower branch. The second level likewise $2L/4$. Overall, the computational load is

$$L\sum_{k=0}^{\infty} 2^{-k} \;=\; 2L \tag{23}$$

operations per input $x[n]$, assuming that the filter-back goes on "forever." In fact – and rather unusually for an FIR filter – we will not be interested in especially long filters. A value $L = 8$ is quite normal.

Let's just pretend that $H_0(\omega)$ is a perfect LPF with cutoff at $\pi/2$ and that $H_1(\omega)$ is a perfect HPF also at $H_0(\omega)$. Then the time-frequency plot would look like the below.



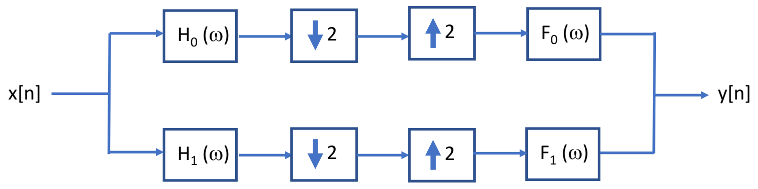
(Please note that only four levels of decimation have been represented here; in general this is arbitrary, and in principle it can go on . . . forever.) The point is that the representation may be more appropriate than the block DFT since components at higher frequencies use data over shorter time windows.

So we have efficiency and appropriateness. Now it is time to discuss invertibility and orthogonality. Before we begin, however, let us examine the notional idea that $H_0(\omega)$ and $H_1(\omega)$ are a perfect LPF and HPF. It seems relatively clear that such surgical splitting avoids aliasing and enables reconstruction. The cost, however, is that L would seem to need to be very large. But wait! The block DFT actually *allows* aliasing and uses filters of length N? It appears that *some* kinds of aliasing do not destroy information.

3 Perfect-Reconstruction Filter Banks

3.1 The Half-Band Condition

We wish to change the basis via a filter bank, but we demand that we lose no information as we do so – we could call this *invertibility* of *perfect reconstruction.* The basic building block for analysis is as below, and clearly we want $y[n] = x[n-l]$ for some l. Note that the two middle blocks (down-sample then up-sample) may look like they cancel; but they do not, since their back-to-back pair amounts to setting every other sample to zero.



Using what we have discovered about sample-rate conversion, we have after the up-sample operation on the upper branch

$$\frac{1}{2}\left(H_0(z)X(z) + H_0(-z)X(-z)\right) \tag{24}$$

which means

$$\begin{aligned} Y(z) &= \frac{1}{2}\left(H_0(z)F_0(z)X(z) + H_0(-z)F_0(z)X(-z)\right) \\ &\quad + \frac{1}{2}\left(H_1(z)F_1(z)X(z) + H_1(-z)F_1(z)X(-z)\right) \end{aligned} \tag{25}$$

which means that in order that we have $y[n] = x[n-l]$ we need

$$H_0(z)F_0(z) + H_1(z)F_1(z) = 2z^{-l} \tag{26}$$

$$H_0(-z)F_0(z) + H_1(-z)F_1(z) = 0 \tag{27}$$

This turns out to be way under-determined. So we adopt the common choice

$$F_0(z) \; = \; H_1(-z) \tag{28}$$

so that we require

$$F_1(z) \; = \; -H_0(-z) \tag{29}$$

in order to have (27) be satisfied.

It is interesting to substitute (28) and (29) into (26) and then to evaluate the result at both z and $-z$; we get

$$\begin{aligned} H_0(z)F_0(z) - H_0(-z)F_0(-z) &= 2z^{-l} && (30) \\ H_0(-z)F_0(-z) - H_0(z)F_0(z) &= 2(-1)^l z^{-l} && (31) \end{aligned}$$

which implies that l must be odd.

It is also convenient to define

$$\begin{aligned} P(z) &\equiv z^l H_0(z)F_0(z) && (32) \\ &= z^l H_0(z)H_1(-z) && (33) \end{aligned}$$

so since $(-z)^l = -z^l$ (as l is odd), we can write

$$P(z) + P(-z) = 2 \tag{34}$$

This is the *half-band* condition, and is perhaps familiar as the first Nyquist criterion for pulse-shaping from digital communications. The half-band condition is sufficient (and by no means necessary!) for perfect reconstruction. And in fact the half-band condition is itself underdetermined.

3.2 The Haar Example

Here we have

$$\begin{aligned} H_0(z) &= \frac{1}{\sqrt{2}}(1 + z^{-1}) && (35) \\ F_0(z) &= \frac{1}{\sqrt{2}}(1 + z^{-1}) && (36) \\ H_1(z) &= \frac{1}{\sqrt{2}}(1 - z^{-1}) && (37) \\ F_1(z) &= -\frac{1}{\sqrt{2}}(1 - z^{-1}) && (38) \end{aligned}$$

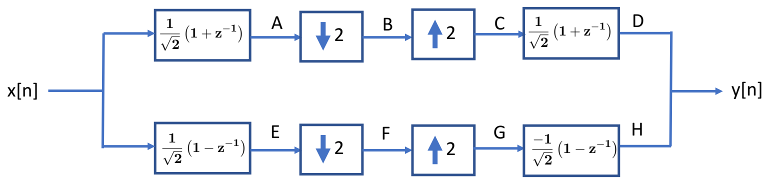
It is interesting that while (35) is the Haar filter[1] and that (37) & (38) follow from (28) & (29) applied to (35) & (36), the actual choice of (36) is really quite arbitrary. In fact, inserting (35) into (26) gives us

$$\begin{aligned} [F_0(z) - F_0(-z)] \ + \ z^{-1}[F_0(z) + F_0(-z)] \quad &= \quad 2\sqrt{2}z^{-l} && (39) \\ \text{(odd-indexed terms)} + z^{-1}\text{(odd-indexed terms)} \quad &= \quad \sqrt{2}z^{-l} && (40) \end{aligned}$$

This implies that there are only two adjacent non-zero terms in $F_0(z)$; it makes sense to choose a first-order $F_0(z)$, but we still have

$$f_0[1]z^{-1} + z^{-1}(f_0[0]) = 2\sqrt{2}z^{-l} \tag{41}$$

from (39). For symmetry and linear phase we choose (36).



x[n]
$\frac{1}{\sqrt{2}}(1+z^{-1})$
A
2
B
2
C
$\frac{1}{\sqrt{2}}(1+z^{-1})$
D
y[n]
$\frac{1}{\sqrt{2}}(1-z^{-1})$
E
2
F
2
G
$\frac{-1}{\sqrt{2}}(1-z^{-1})$
H

The figure above shows a scaled version of the Haar system. We have:

A: $\frac{1}{\sqrt{2}}\{\ldots, x[0]+x[-1],\ x[1]+x[0],\ x[2]+x[1],\ x[3]+x[2],\ x[4]+x[3],\ \ldots\}$

B: $\frac{1}{\sqrt{2}}\{\ldots, x[0]+x[-1],\ x[2]+x[1],\ x[4]+x[3],\ \ldots\}$

C: $\frac{1}{\sqrt{2}}\{\ldots, x[0]+x[-1],\ 0,\ x[2]+x[1],\ 0,\ x[4]+x[3],\ \ldots\}$

D: $\frac{1}{2}\{\ldots, x[0]+x[-1],\ x[0]+x[-1],\ x[2]+x[1],\ x[2]+x[1],\ x[4]+x[3],\ \ldots\}$

E: $\frac{1}{\sqrt{2}}\{\ldots, x[0]-x[-1],\ x[1]-x[0],\ x[2]-x[1],\ x[3]-x[2],\ x[4]-x[3],\ \ldots\}$

F: $\frac{1}{\sqrt{2}}\{\ldots, x[0]-x[-1],\ x[2]-x[1],\ x[4]-x[3],\ \ldots\}$

G: $\frac{1}{\sqrt{2}}\{\ldots, x[0]-x[-1],\ 0,\ x[2]-x[1],\ 0,\ x[4]-x[3],\ \ldots\}$

H: $\frac{1}{2}\{\ldots, x[-1]-x[0],\ x[0]-x[-1],\ x[1]-x[2],\ x[1]-x[2],\ x[3]-x[4],\ \ldots\}$

y: $\{\ldots, x[-1],\ x[0],\ x[1],\ x[2],\ x[3],\ \ldots\}$

[1]The Haar filter is just a running two-sampler average.

The last line (the final output $y[n]$) is obtained from adding the signals at D and H. Note that it is identical to the input $x[n]$ – perfect recovery! – except for a delay by a single time-step ($l = 1$).

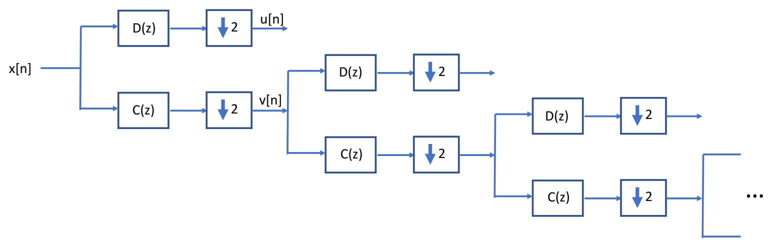
One more note on the Haar system is appropriate. Consider the octave filter bank structure, with the Haar filter and the change-of-basis interpretation (20). Stopping after three levels, the matrix $\mathbf{W}$ is

$$\mathbf{W} = \tag{42}$$
$$\begin{pmatrix}
a & -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a & -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a & -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a & -a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & -a & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & -a & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & -a & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a & -a \\
b & b & -b & -b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & b & b & -b & -b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & b & -b & -b & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & b & b & -b & -b \\
c & c & c & c & -c & -c & -c & -c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & c & c & c & -c & -c & -c & -c \\
d & d & d & d & d & d & d & d & -d & -d & -d & -d & -d & -d & -d & -d \\
d & d & d & d & d & d & d & d & d & d & d & d & d & d & d & d
\end{pmatrix}$$

where $a = 1/\sqrt{2}$, $b = 1/2$, $c = 1/\sqrt{8}$, $d = 1/4$. What is intended to be illustrated here is that the basis vector is the same at all levels, just translated within that level and dilated (by a factor of two) as the level is deepened. So if an artifact has a good projection onto (match with) some basis, the same artifact dilated by a factor of two would appear at a deeper level. This is why this is said to be a decomposition according to *scale*.

4 Orthogonal Filter Banks

To avoid too much subscripting, and to be in commonality with the literature, we'll switch from $H_0(z)$ & $H_1(z)$ to $C(z)$ & $D(z)$, as shown below.



It's worth expressing the output of the top two branches as a matrix-vector multiplication, shown in (43) for $L = 4$:

$$\begin{pmatrix} \vdots \\ u[n] \\ v[n] \\ u[n-1] \\ v[n-1] \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c_0 & c_1 & c_2 & c_3 & 0 & 0 & \dots \\ d_0 & d_1 & d_2 & d_3 & 0 & 0 & \dots \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & \dots \\ 0 & 0 & d_0 & d_1 & d_2 & d_3 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ x[n] \\ x[n-1] \\ x[n-2] \\ x[n-3] \\ x[n-4] \\ x[n-5] \\ \vdots \end{pmatrix} \tag{43}$$

For orthogonality we require

$$\sum_n c_n c_{n-2k} = \delta[k] \tag{44}$$

$$\sum_n c_n d_{n-2k} = 0 \tag{45}$$

$$\sum_n d_n d_{n-2k} = \delta[k] \tag{46}$$

As usual we have rather too much freedom to select the filters. For now assume that $\{c[n]\}$ is already picked. The Smith-Barnwell/Mintzer choice for $\{d[n]\}$ is

$$D(z) = -z^{-(L-1)}C(-z^{-1}) \tag{47}$$

$$= -z^{-(L-1)}(c_0 - c_1 z + c_2 z^2 - \dots + (-1)^{(L-1)} c_{L-1} z^{L-1}) \tag{48}$$

$$= (-1)^L c_{L-1} + \dots - c_2 z^{-(L-3)} + c_1 z^{-(L-2)} - c_0 z^{-(L-1)} \tag{49}$$

The Smith-Barnwell/Mintzer choice is not the only one, but it is fairly nice for the following reasons.

Smith Barnwell/Mintzer is Nice: Perfect Reconstruction

According to (33) we define, with $l = L - 1$,

$$\begin{aligned} P(z) &= z^{(L-1)} H_0(z) H_1(-z) && (50)\\ &= z^{(L-1)} C(z) D(-z) && (51)\\ &= z^{(L-1)} C(z)(-(-z)^{-(L-1)} C(z^{-1})) && (52)\\ &= C(z)C(z^{-1}) && (53) \end{aligned}$$

since $l = L - 1$ has to be odd. Now, notice that this refers to

$$p[n] = c[n] \star c[-n] \tag{54}$$

Looking at (44) and realizing that this is a constraint on the down-sampled $\{p[n]\}$, we have

$$P(z) + P(-z) = 2 \tag{55}$$

or

$$C(z)C(z^{-1}) + C(-z)C(-z^{-1}) = 2 \tag{56}$$

What this means is that (44) is the same as the *half-band condition* from (34). If we select $C(z)$ to satisfy (55) then we have both perfect reconstruction and one out of three conditions for orthogonality.

Smith Barnwell/Mintzer is Nice: Self-Orthogonality

We just found out that if (44) with the Smith-Barnwell-Mintzer condition (44) then we have perfect reconstruction (invertibility). We also have the same property for $\{d[n]\}$:

$$\begin{aligned} &D(z)D(z^{-1}) + D(-z)D(-z^{-1})\\ &\quad= (-z^{-(L-1)} C(-z^{-1}))(-z^{(L-1)} C(-z))\\ &\qquad+(z^{-(L-1)} C(z^{-1}))(z^{-(L-1)} C(z)) && (57)\\ &\quad= Q(z) + Q(-z) && (58) \end{aligned}$$

That is, if $C(z)$ is chosen to satisfy (56) then both (44) and (46) are satisfied.

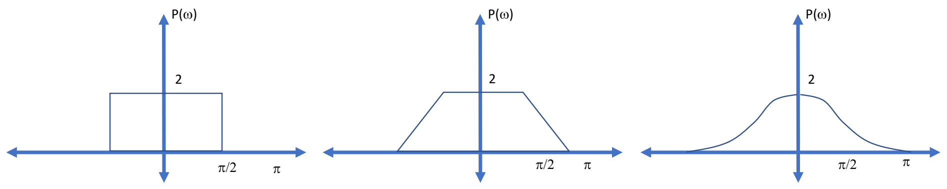
Smith Barnwell/Mintzer is Nice: Cross-Orthogonality

Just as good, we have

$$\begin{aligned} &C(z)D(z^{-1}) + C(-z)D(-z^{-1})\\ &\quad= C(z)(-z^{(L-1)} C(-z)) + C(-z)(z^{-(L-1)} C(z)) && (59)\\ &\quad= 0 && (60) \end{aligned}$$

so (45) is satisfied as well. That is, we have orthogonality!

This "half-band" condition – introduced as a sufficient condition for perfect reconstruction (invertibility) in (34) and rediscovered as a by-product of the Smith-Barnwell/Mintzer choice in (55) that also gives orthogonality – is also known as the first Nyquist condition in digital communications. It is perhaps worth mentioning that any filter satisfying the half-band condition gives rise to a structure commonly known as a *quadrature mirror filter* (QMF) bank. Below we see three possible configurations for a viable $P(z)$.



On the left is the rather obvious brick-wall filter. This is fine, but even to approximate it requires a very large L: no good. The middle is better, and it becomes clear how aliasing is not the deal-killer we thought it might be. On the right is the "raised-cosine" filter that uniquely satisfies both first and second Nyquist criteria. Here we have

$$\begin{aligned} P(\omega) &= 1 + \cos(\omega) && (61)\\ &= \frac{1}{2}e^{j\omega} \;+\; 1 \;+\; \frac{1}{2}e^{-j\omega} && (62)\\ P(z) &= \frac{1}{2}z \;+\; 1 \;+\; \frac{1}{2}z^{-1} && (63) \end{aligned}$$

and since $P(z) = C(z)C(z^{-1})$ this means

$$C(z) = \frac{1}{\sqrt{2}}(1 + z^{-1}) \tag{64}$$

That is, the raised-cosine filter and the Haar filter are the same thing.

As a side note, it is interesting to ask whether filters can be orthogonal *and* linear-phase. A linear-phase filter structure

$$\{c_0,\ c_1,\ c_2,\ ,\ldots,\ \pm c_2,\ \pm c_1,\ \pm c_0\ \} \tag{65}$$

meaning that the impulse response is either even or odd symmetric. Clearly the Haar filter works, it is even symmetric and hence linear-phase. For a filter of length $(L = 4)$ we interrogate (44) for $k = 1$ and find it implies

$$\pm 2c_0c_1 \;=\; 0 \tag{66}$$

which means that any such filter has only two identical non-zero coefficients, and since $L = 4$ it means $c_1 = 0$. Similar analysis for $L = 6,\ 8, \ldots$ finds the same conclusion: c_0 is the only non-zero coefficient. While this is slightly different from the Haar filter it possesses no new richness, so we do not pursue it: aside from $L = 2$ (Haar) linear-phase is out of the question if orthogonality is desired.

5 Daubechies Filters

5.1 The Max-Flat Idea

The half-band conditions (resulting from the choice (28)) and even the Smith-Barnwell/Mintzer choice are decent but non-unique ways to get perfect reconstruction and orthogonality, respectively. But even the latter does not specify $C(z)$, only the half-band condition that $C(\omega)$ must satisfy. Daubechies came up with a set of conditions that are often thought to give the "best" QMF. Her idea is to look for a filter that is both short (small L) and decently frequency-selective.

5.2 The Really Technical Development

The development is rather indirect. Here goes. Consider the function

$$(1-y)^{-p} = \sum_{k=0}^{\infty} \begin{pmatrix} p+k-1 \\ k \end{pmatrix} y^k \tag{67}$$

We will truncate this to p terms

$$\begin{aligned} B(y) &= \sum_{k=0}^{p-1} \begin{pmatrix} p+k-1 \\ k \end{pmatrix} y^k && (68) \\ &= 1 + py + \begin{pmatrix} p+1 \\ 2 \end{pmatrix} y^2 + \ldots + \begin{pmatrix} 2p-1 \\ p-1 \end{pmatrix} y^{p-1} && (69) \end{aligned}$$

Now

$$\begin{aligned} \tilde{P}(y) &\equiv 2(1-y)^p B(y) && (70) \\ &= 2(1-y)^p((1-y)^{-p} + \mathcal{O}(y^p)) && (71) \\ &= 2 + \mathcal{O}(y^p) && (72) \end{aligned}$$

Now, notice from (72) we have

$$\tilde{P}'(y)|_{y=0} = \tilde{P}''(y)|_{y=0} = \ldots = \tilde{P}^{(p-1)}(y)|_{y=0} = 0 \tag{73}$$

and likewise we have

$$\tilde{P}'(y)|_{y=1} = \tilde{P}''(y)|_{y=1} = \ldots = \tilde{P}^{(p-1)}(y)|_{y=1} = 0 \tag{74}$$

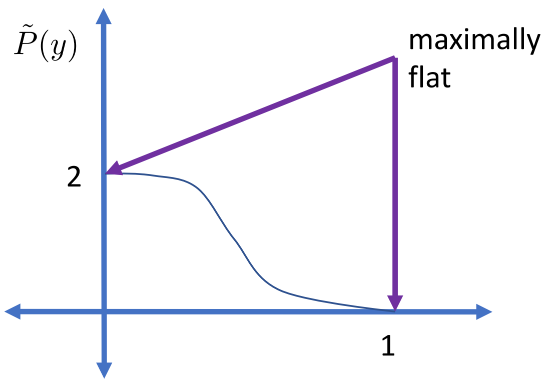
from (70). Similarly, from (72) we have

$$\tilde{P}(0) = 2 \tag{75}$$

and

$$\tilde{P}(1) = 0 \tag{76}$$

from (70). These are the *maximum-flatness* conditions: the function is flat and very smoothly so at both $y = 0$ and $y = 1$, and decreases from "passband" to "stopband" between. A notion is plotted below.



From (69) and (70) we have that $\tilde{P}(y)$ is a polynomial in y of degree $2p-1$. As such, $\tilde{P}'(y)$ is a polynomial in y of degree $2p-2$. And (73) and (74) tell us what it must be:

$$\tilde{P}'(y) \;=\; Cy^{p-1}(1-y)^{p-1} \tag{77}$$

Since $\tilde{P}'(y) = 0$ and hence $\tilde{P}'(1-y) = 0$ as well, and since we know

$$\begin{aligned}(\tilde{P}(y) + \tilde{P}(1-y))|_{y=0} &= \tilde{P}(y)|_{y=0} + \tilde{P}(y)|_{y=1} & (78)\\ &= 2 & (79)\end{aligned}$$

we can say

$$\tilde{P}(y) + \tilde{P}(1-y) \;=\; 2 \tag{80}$$

which is looking very close to our half-band condition, except in y as opposed to z.

Now substitute

$$y \longleftarrow \left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right) \tag{81}$$

Note

$$\begin{aligned} 1-y &= \frac{1}{4}\left(4-(-z+2-z^{-1})\right) && (82)\\ &= \frac{1}{4}\left(2+z+z^{-1}\right) && (83)\\ &= \left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right) && (84)\end{aligned}$$

So we substitute

$$P(z) = \tilde{P}(y)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} \tag{85}$$

Now we have

$$\begin{aligned} &P(z)+P(-z)\\ &= \tilde{P}(y)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} + \tilde{P}(y)\Big|_{y=\left(\frac{1+z}{2}\right)\left(\frac{1+z^{-1}}{2}\right)} && (86)\\ &= \tilde{P}(y)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} + \tilde{P}(1-y)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} && (87)\\ &= \left(\tilde{P}(y)+\tilde{P}(1-y)\right)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} && (88)\\ &= 2 && (89)\end{aligned}$$

so the half-band condition is indeed satisfied by the Daubechies filters!

5.3 How to Make a Daubechies Filter

All we need to do now is to find find one. We need to write

$$\tilde{P}(y) = 2(1-y)^p\left(\sum_{k=0}^{p-1}\binom{p+k-1}{k}y^k\right) \tag{90}$$

from (70) and (68). Then we write

$$\begin{aligned} P(z) &= \tilde{P}(y)\Big|_{y=\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)} && (91)\\ &= 2\left(\frac{1+z}{2}\right)^p\left(\frac{1+z^{-1}}{2}\right)^p\\ &\quad\times\sum_{k=0}^{p-1}\binom{p+k-1}{k}\left(\frac{1-z}{2}\right)^k\left(\frac{1-z^{-1}}{2}\right)^k && (92)\end{aligned}$$

from (90), (85) and (84). Finally we must use

$$P(z) \;=\; C(z)C(z^{-1}) \tag{93}$$

to extract $C(z)$ from $P(z)$.

So let's try $p = 1$. From (92) we easily get

$$P(z) \;=\; \frac{1}{2}\left((1+z)(1+z^{-1})\right) \tag{94}$$

and it is easy to apply (94) to (93) to get

$$C(z) \;=\; \frac{1+z^{-1}}{\sqrt{2}} \tag{95}$$

which is the Haar filter!

To show something a little more interesting, let us try $p = 2$. We get

$$\begin{aligned} P(z) \;&=\; \frac{1}{8}(1+z)^2(1+z^{-1})^2\left(1+2\left(\frac{1-z}{2}\right)\left(\frac{1-z^{-1}}{2}\right)\right) && (96)\\ &=\; \frac{-1}{16}(1+z)^2(1+z^{-1})^2\left(z-4+z^{-1}\right) && (97)\\ &=\; \frac{-1}{16(2-\sqrt{3})}(1+z)^2(1+z^{-1})^2 \\ &\qquad \times\left((1-(2-\sqrt{3})z)(1-(2-\sqrt{3})z^{-1})\right) && (98) \end{aligned}$$

meaning that we have

$$C(z) \;=\; \frac{1}{\sqrt{32}}\left((1+\sqrt{3}) \;+\; (3+\sqrt{3})z^{-1} \;+\; (3-\sqrt{3})z^{-2} \;+\; (1-\sqrt{3})z^{-3}\right) \tag{99}$$

This is the D4 filter.

6 Wavelets

6.1 The Telescoping Subspaces

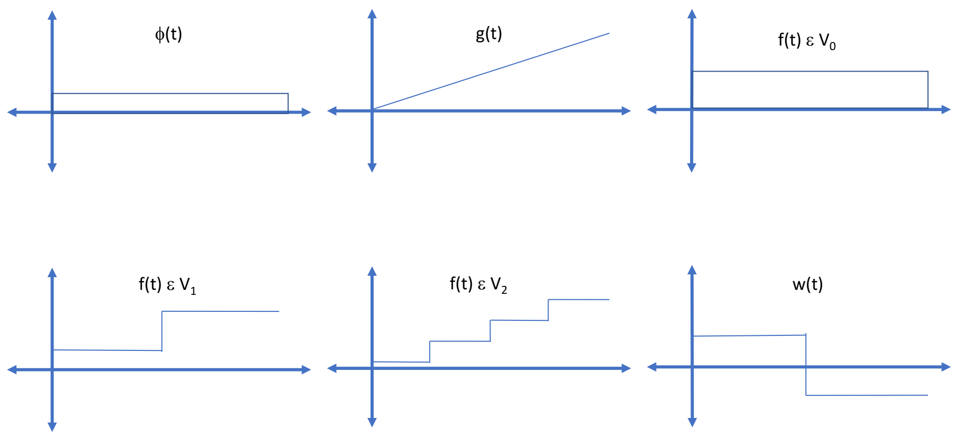
Wavelets are the continuous-time (or -space) version of multi-resolution decomposition. Begin with *telescoping* subspaces

$$\mathcal{V}_0 \subset \mathcal{V}_1 \subset \mathcal{V}_2 \subset \mathcal{V}_3 \subset \mathcal{V}_4 \subset \ldots \tag{100}$$

and require that if $f(t) \in V_j$ then

1. $f(t-k) \in \mathcal{V}_j, \quad \forall k \in \mathcal{I}$; and
2. $f(2t-k) \in \mathcal{V}_{j+1}, \quad \forall k \in \mathcal{I}$.

Also assume that there is *scaling function* $\phi(t)$ such that for $\{\phi(t-k)\}_{k\in\mathcal{I}}$ is a basis for $\mathcal{V}_0$.



An example is given above. We seek to approximate the ramp-function $g(t)$ (top middle) in a telescoping series of spaces that are formed by the scaling function $\phi(t)$, top left. The approximations in $\mathcal{V}_0$, $\mathcal{V}_1$ and $\mathcal{V}_2$ are shown in top right, bottom left and bottom middle, respectively. It is clear that the deeper one gets the better the approximation. We also define the *wavelet space* $\mathcal{W}_0$ with basis $w(t)$ (bottom right), such that

$$\mathcal{V}_j \bigcup \mathcal{W}_j \;=\; \mathcal{V}_{j+1} \tag{101}$$

and

$$\mathcal{V}_j \bigcap \mathcal{W}_j \;=\; \emptyset \tag{102}$$

The function $w(t)$ is called the *mother wavelet.*

6.2 Relationship to Multi-Resolution Decomposition

Now, $\mathcal{V}_0 \subset \mathcal{V}_1$ means that

$$\phi(t) \;=\; \sum_n c_n \phi(2t-n) \tag{103}$$

If we also have $\{\phi(t-k)\}$ orthogonal – and hence $\{\phi(2t-k)\}$ orthogonal – we can write

$$c_n \;=\; \int \phi(t)\phi(2t-n)dt \tag{104}$$

Expressing the orthogonality requirement using this, we have

$$\begin{aligned}
\delta[k] &= \int \phi(t)\phi(t-k)dt && (105)\\
&= \int \left(\sum_m c_m \phi(2t-m)\right)\left(\sum_n c_n \phi(2(t-k)-n)\right)dt && (106)\\
&= \sum_n c_n c_{n-2k} && (107)
\end{aligned}$$

It is very interesting that (107) is identical to (44) – that is, the condition for a telescoping basis based on orthogonal functions is the same as the condition for a multi-resolution decomposition filter to be orthogonal. Let us go a little further, and note that since $\mathcal{W}_j \subset \mathcal{V}_{j+1}$ we can write

$$w(t) = \sum_k d_k \phi(2t-k) \qquad (108)$$

where

$$d_k = \int w(t)\phi(2t-k)dt \qquad (109)$$

If we desire orthogonality of $\{w(t-m)\}$ we have

$$\begin{aligned}
\delta[k] &= \int \phi(t)\phi(t-k)dt && (110)\\
&= \int \left(\sum_m d_k \phi(2t-m)\right)\left(\sum_n d_n \phi(2(t-k)-n)\right)dt && (111)\\
&= \sum_n d_n d_{n-2k} && (112)
\end{aligned}$$

and similarly, if orthogonality of $\mathcal{W}_j$ to $\mathcal{V}_j$ is desired we have

$$\begin{aligned}
0 &= \int \phi(t)w(t-k)dt && (113)\\
&= \int \left(\sum_m c_k \phi(2t-m)\right)\left(\sum_n d_n \phi(2(t-k)-n)\right)dt && (114)\\
&= \sum_n c_n d_{n-2k} && (115)
\end{aligned}$$

That is, (107), (112) & (115) – demanded for orthogonality of the telescoping representation – are identical to (44), (46) & (45) for orthogonality of a multi-resolution decomposition.

6.3 How to Make the Mother Wavelet and Scaling Function

So what are $\phi(t)$ and $w(t)$? The relation

$$\phi(t) \;=\; \sum_k c_k \phi(2t-k) \tag{116}$$

provides the answer. Take the (continuous-time) Fourier transform

$$\begin{aligned}
\Phi(\Omega) \;&=\; \int_{-\infty}^{\infty} \phi(t) e^{-j\Omega t} dt & (117)\\
&=\; \int_{-\infty}^{\infty} \left(\sum_k c_k \phi(2t-k) \right) e^{-j\Omega t} dt & (118)\\
&=\; \sum_k c_k e^{-j\left(\frac{\Omega}{2}\right)k} \int_{-\infty}^{\infty} \phi(2t-k) e^{-j\left(\frac{\Omega}{2}\right)(2t-k)} dt & (119)\\
&=\; \frac{1}{2} C\left(\frac{\Omega}{2}\right) \Phi\left(\frac{\Omega}{2}\right) & (120)
\end{aligned}$$

where

$$C(\omega) \;\equiv\; \sum_k c_k e^{-j\omega k} \tag{121}$$

which of course repeats with period 2π. We are not interested in the factor of $\frac{1}{2}$ in (120) since we normalize to have unit energy; so let us drop it. We also have for the mother wavelet

$$W(\Omega) \;=\; \frac{1}{2} D\left(\frac{\Omega}{2}\right) \Phi\left(\frac{\Omega}{2}\right) \tag{122}$$

Note that as $k \to \infty$ we have $\frac{\Omega}{2^k} \to 0$. We arbitrarily set $\Phi(0) = 1$ – any non-zero constant will do – so we have

$$\begin{aligned}
\Phi(\omega) \;&=\; \prod_{k=1}^{\infty} C\left(\frac{\omega}{2^k}\right) & (123)\\
W(\omega) \;&=\; D\left(\frac{\omega}{2}\right) \prod_{k=2}^{\infty} C\left(\frac{\omega}{2^k}\right) & (124)
\end{aligned}$$

which explicitly define the (Fourier transforms of the) scaling function and mother wavelet in terms of chosen multi-resolution filter function.

6.4 Compactness of the Scaling Function and Mother Wavelet

First, let us observe that (123) and (124) require that $C(\omega) = 0$ else $\Phi(\omega)$ goes on forever. Let us also define $c(t)$ via

$$\begin{aligned} C(\omega) &= \mathcal{F}\left[c(t)\right] && (125)\\ &= \mathcal{F}\left[\sum_k c_k \delta(t-k)\right] && (126)\\ &= \int_{-\infty}^{\infty} \sum_k c_k \delta(t-k) e^{-j\omega t} dt && (127)\\ &= \sum_k c_k \delta(t-k) e^{-j\omega k} && (128)\end{aligned}$$

This is not especially useful except to tell is that $c(t)$ is time-limited if $\{c_n\}$ is FIR – $c(t)$ has support only on $[0, L)$ (actually $[0, L-1)$). Then (123) implies

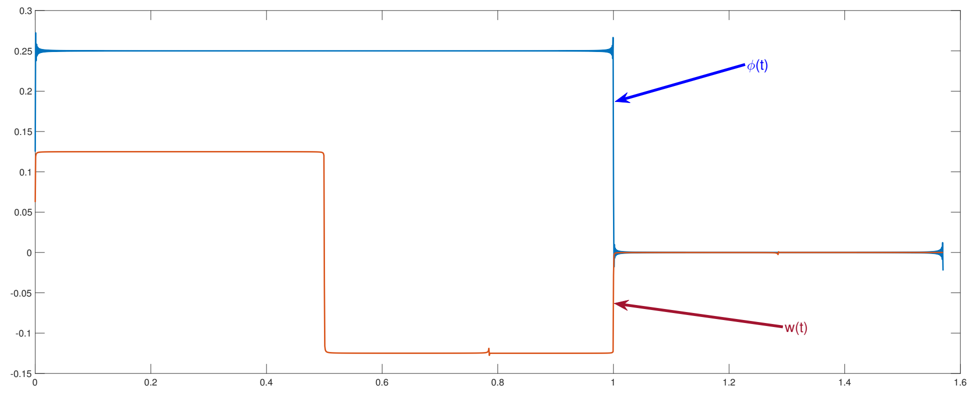
$$\phi(t) = c(2t) \star c(4t) \star c(8t) \star c(16t) \star c(32t) \star \dots \quad (129)$$

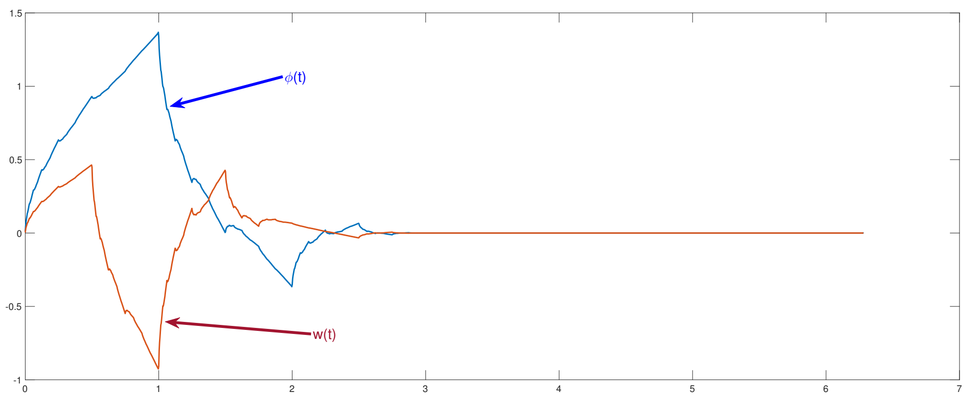
meaning that the support of $\phi(t)$ can be no greater than of length

$$\frac{L}{2} + \frac{L}{4} + \frac{L}{8} + \frac{L}{16} + \dots = L \quad (130)$$

That is, the scaling function $\phi(t)$ is supported only on $[0, L)$ – it is *compact*! The same can be said for the mother wavelet $w(t)$.

Examples of scaling functions and wavelets for the Daubechies-2 (i.e., Haar) and Daubechies-4 systems are given below





What is striking is that the Haar functions are exactly what one might think, and basically the same as in the earlier notional cartoon. The Daubechies-4 scaling function and mother wavelet are weird. But they are what they are.

ECE 6123
Advanced Signal Processing: EM, HMMs and BP

Peter Willett

Fall 2017

1 Expectation-Maximization (EM)

1.1 The Algorithm and Why it Works

Suppose we have a problem in which the variables can be divided as follows:

Z: The observation – known, of course.

X: The unknown parameters that are desired.

K: Some hidden random variables.

In fact we could have $\mathbf{X}$ as an unknown random variable (i.e., with a prior) but for the present discussion let's assume it is a parameter. Our goal is to find the maximum-likelihood estimator (MLE) of $\mathbf{X}$ based on $\mathbf{Z}$. Now, if you can easily write $p_{\mathbf{X}}(\mathbf{Z})$ by all means maximize it and skip this whole section. What we are interested in are cases in which $p_{\mathbf{X}}(\mathbf{Z}|\mathbf{K})$ and $p_{\mathbf{X}}(\mathbf{K})$ can both be written, but (with integration in the most general sense, possibly meaning a sum)

$$p_{\mathbf{X}}(\mathbf{Z}) = \int p_{\mathbf{X}}(\mathbf{Z}|\mathbf{K})p_{\mathbf{X}}(\mathbf{K})d\mathbf{K} \tag{1}$$

is irritating and complicated to evaluate, let alone maximize.

The EM approach has two steps. We begin with some sort of guess – and, yes, it can matter a lot – as to $\mathbf{X}^0$, and set $n = 0$.

***E-step*:** Here we form the "Q-function"

$$Q(\mathbf{X};\mathbf{X}^{(n)}) \equiv \int \log\left(p_{\mathbf{X}}(\mathbf{Z},\mathbf{K})\right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z})d\mathbf{K} \tag{2}$$

and the reason this is called the E-step should be obvious: it involves an expectation, albeit of an *un*expected function.

***M-step*:** We maximize (yes, that's why it's called the M-step) and form

$$\mathbf{X}^{(n+1)} \;=\; \arg\max_{\mathbf{X}} \left\{ Q(\mathbf{X};\mathbf{X}^{(n)}) \right\} \tag{3}$$

We then increment n and return to the E-step.

The reason this works is actually pretty simple. We have

$$\begin{aligned} Q(\mathbf{X};\mathbf{X}^{(n)}) &= \int \log\left(p_{\mathbf{X}}(\mathbf{Z},\mathbf{K})\right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) d\mathbf{K} & (4)\\ &= \int \left(\log\left(p_{\mathbf{X}}(\mathbf{K}|\mathbf{Z})\right) + \log\left(p_{\mathbf{X}}(\mathbf{Z})\right)\right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) d\mathbf{K} & (5)\\ &= \log\left(p_{\mathbf{X}}(\mathbf{Z})\right) + \int \log\left(p_{\mathbf{X}}(\mathbf{K}|\mathbf{Z})\right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) d\mathbf{K} & (6) \end{aligned}$$

Now consider any two pmf's or pdf's q_1 & q_2. Noting that

$$\ln(x) \;\le\; x-1 \tag{7}$$

with equality if and only if $x = 1$ (draw the graph!), we have

$$\begin{aligned} \int q_1 \log(q_2) - \int q_1 \log(q_1) &= \int q_1 \log\left(\frac{q_2}{q_1}\right) & (8)\\ &\le \log(2) \int q_1 \left(\frac{q_2}{q_1} - 1\right) & (9)\\ &\le \log(2) \int q_1 \left(\frac{q_2}{q_1} - 1\right) & (10)\\ &= 0 & (11) \end{aligned}$$

This means that

$$\int q_1 \log(q_2) \;\le\; \int q_1 \log(q_1) \tag{12}$$

with equality if and only if $q_1 = q_2$. Returning to (6) we see from (12) that if

$$Q(\mathbf{X}^{(n+1)};\mathbf{X}^{(n)}) \;>\; Q(\mathbf{X}^{(n)};\mathbf{X}^{(n)}) \tag{13}$$

then

$$\log\left(p_{\mathbf{X}^{(n+1)}}(\mathbf{Z})\right) \;>\; \log\left(p_{\mathbf{X}^{(n)}}(\mathbf{Z})\right) \tag{14}$$

since the second term must have decreased. This means that the change from $\mathbf{X}^{(n)}$ to $\mathbf{X}^{(n+1)}$ must have increased the likelihood that we are aiming to maximize. Note that although the M-step by tradition requires a maximization, the M could also stand for *majorization*: all that is really required for (14) is an increase in Q. There is no real need for a *maximization* if that turns out to be difficult or expensive. Note also that (14) tells us clearly that this is *hill-climbing* approach: there is no guarantee that a $\underline{\text{global}}$ maximum likelihood be found.

As a final note, many authors refer to $\mathbf{K} \bigcup \mathbf{Z}$ as the *complete data* and $\mathbf{Z}$ as the *in*complete data. I don't care for the nomenclature; but there it is.

1.2 The Gaussian Mixture Example

Sometimes you need to manufacture the $\mathbf{K}$'s yourself. Consider that you have N independent z_i's from the same Gaussian mixture pdf

$$p(z) = \sum_{m=1}^{M} \frac{p_m}{\sqrt{|2\pi\mathbf{R}|}} e^{-\frac{1}{2}(z-\mu_m)^T\mathbf{R}^{-1}(z-\mu_m)} \tag{15}$$

where the mixture priors $\{p_m\}$ and the means $\{\mu_m\}$ are both unknown. You could insert[1] $\mathbf{K} = \{k_i\}$ such that $k_i \in \{1, M\}$ and

$$\begin{aligned} Pr(k_i = m) &= p_m & (16)\\ \{k_i\} &\sim \text{independent and identically distributed} & (17)\\ p(z_i|k_i) &= \frac{1}{\sqrt{|2\pi\mathbf{R}|}} e^{-\frac{1}{2}(z_i-\mu_{k_i})^T\mathbf{R}^{-1}(z_i-\mu_{k_i})} & (18) \end{aligned}$$

In the EM formalism the first thing we need is $p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z})$. This is relatively easy:

$$\begin{aligned} p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) &= \prod_{i=1}^{N} p(k_i|\mathbf{Z}) & (19)\\ &= \prod_{i=1}^{N} p_{\mathbf{X}^{(n)}}(k_i|z_i) & (20)\\ &\equiv \prod_{i=1}^{N} w_i(k_i) & (21)\\ w_i(m) &= \frac{p_m^{(n)} p_{\mathbf{X}^{(n)}}(z_i|k_i = m)}{\sum_{l=1}^{M} p_l^{(n)} p_{\mathbf{X}^{(n)}}(z_i|k_i = l)} & (22)\\ &= \frac{p_m^{(n)} \frac{1}{\sqrt{|2\pi\mathbf{R}|}} e^{-\frac{1}{2}(z_i-\mu_m^{(n)})^T\mathbf{R}^{-1}(z_i-\mu_m^{(n)})}}{\sum_{l=1}^{M} p_l^{(n)} \sqrt{|2\pi\mathbf{R}|} e^{-\frac{1}{2}(z_i-\mu_l^{(n)})^T\mathbf{R}^{-1}(z_i-\mu_l^{(n)})}} & (23) \end{aligned}$$

[1]Actually this is the way you would generate such random variables.

The nomenclature involving w's is fairly common for the posterior probabilities. Now we have

$$\begin{aligned}
Q(\mathbf{X};\mathbf{X}^{(n)}) &= \int \log\left(p_{\mathbf{X}}(\mathbf{Z},\mathbf{K})\right) p_{\mathbf{X}^{(n)}}(\mathbf{K})d\mathbf{K} && (24)\\
&= \sum_{\mathbf{K}}\left(\sum_{i=1}^{N}\left(\left(\log(p_{k_i}) - \frac{1}{2}\log\left(|2\pi\mathbf{R}|\right)\right.\right.\right.\\
&\qquad \left.\left.\left. - \frac{1}{2}\left(z_i-\mu_{k_i}\right)^T\mathbf{R}^{-1}\left(z_i-\mu_{k_i}\right)\right)\prod_{i=1}^{M} w_i(k_i)\right)\right) && (25)\\
&= \sum_{m=1}^{M}\sum_{i=1}^{N}\left(\left(\log(p_m) - \frac{1}{2}\log\left(|2\pi\mathbf{R}|\right)\right.\right.\\
&\qquad \left.\left. - \frac{1}{2}\left(z_i-\mu_m\right)^T\mathbf{R}^{-1}\left(z_i-\mu_m\right)\right) w_i(m)\right) && (26)
\end{aligned}$$

Maximizing (26) over p_m subject to the constraint that these prior probabilities add to unity yields

$$\frac{\partial}{\partial p_m}\left(\sum_{i=1}^{N}\log(p_m)w_i(m) - \lambda p_m\right) = 0 \qquad (27)$$

or

$$p_m = \frac{\sum_{i=1}^{N} w_i(m)}{\sum_{i=1}^{N}\sum_{l=1}^{M} w_i(l)} \qquad (28)$$

where of course the denominator is most easily found by normalization. As for the μ_m's we take the gradient

$$\begin{aligned}
\nabla\left(\sum_{i=1}^{N}\left(\frac{1}{2}\left(z_i-\mu_m\right)^T\mathbf{R}^{-1}\left(z_i-\mu_m\right)\right)w_i(m)\right) &= 0 && (29)\\
\sum_{i=1}^{N}\left(\mathbf{R}^{-1}\left(z_i-\mu_m\right)\right)w_i(m) &= 0 && (30)
\end{aligned}$$

or

$$\mu_m = \frac{\sum_{i=1}^{N} w_i(m)z_i}{\sum_{i=1}^{N} w_i(m)} \qquad (31)$$

This is a nice easy recursion: Start with a guess about the μ_m's and p_m's. Then calculate the w's according to (23). Then update the parameters according to (28) & (31); and go back to getting new w's. Stop when you get tired of it – or more likely when the estimates stop moving. Note that this is a "soft" version of the celebrated k-means algorithm for clustering. It is also interesting to note that it *is* possible to estimate the covariance as well, and also to allow the covariances to be different across the various modes.

2 The Hidden Markov Model

2.1 Modeling for EM

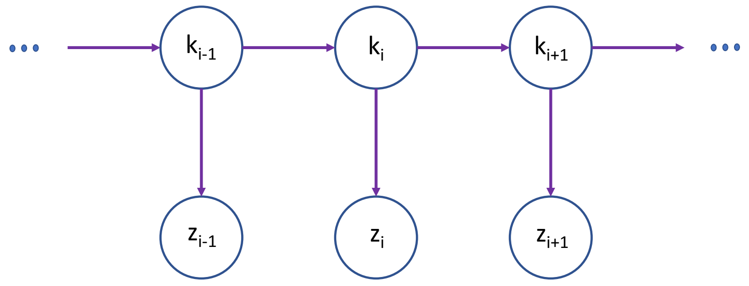
A Markov model has

$$p(\mathbf{Z}) = p(z_1)\prod_{i=2}^{N} p(z_i|z_{i-1}) \tag{32}$$

whereas a *hidden* Markov model (HMM) does not give direct access to the Markov process:

$$p(\mathbf{Z},\mathbf{K}) = \left(p(k_1)\prod_{i=2}^{N} p(k_i|k_{i-1})\right)\left(\prod_{i=1}^{N} p(z_i|k_i)\right) \tag{33}$$

A fragment of an HMM is pictured below.



I have gone out of my way to use non-standard HMM nomenclature (especially the k_i's) to emphasize the relationship to the EM algorithmic tools we have developed. We will define $\mathbf{X} = \{\mathbf{A}, \mathbf{B}, \mathbf{p}\}$ in which

$$A(m|n) = Pr(k_i = m|k_{i-1} = n) \tag{34}$$
$$B(l|n) = Pr(z_i = l|k_i = n) \tag{35}$$
$$p(m) = Pr(k_1 = m) \tag{36}$$

These are what we seek: the $M \times M$ matrix $\mathbf{A}$ and the $M \times L$ matrix $\mathbf{B}$, meaning that there are M "hidden" states and L kinds[2] of outputs. And of

[2]There is a perfectly good formulation of the HMM that allows for continuous-valued outputs; for simplicity of notation we will assume discreteness.

course this is a prime example of a problem that is absolutely panting for EM to come solve it.

2.2 The Forward-Backward Algorithm

In this section we seek an expression for the posterior probabilities of the state sequence. Define

$$\alpha(\mathbf{Z}_1^{i+1}, m) \equiv p(\mathbf{Z}_1^i, k_i = m) \tag{37}$$

We can write

$$\begin{aligned}
\alpha(\mathbf{Z}_1^{i+1}, m) &= \sum_{n=1}^{M} p(z_{i+1}, k_{i+1} = m, k_i = n, \mathbf{Z}_1^i) &(38)\\
&= \sum_{n=1}^{M} p(z_{i+1}, k_{i+1} = m | k_i = n, \mathbf{Z}_1^i) \\
&\qquad \times Pr(k_i = n, \mathbf{Z}_1^i) &(39)\\
&= \sum_{n=1}^{M} p(z_{i+1} | k_{i+1} = m, \mathbf{Z}_1^i, k_i = n) \\
&\qquad \times Pr(k_{i+1} = m | \mathbf{Z}_1^i, k_i = n) Pr(k_i = n, \mathbf{Z}_1^i) &(40)\\
&= \sum_{n=1}^{M} p(z_{i+1} | k_{i+1} = m) \\
&\qquad \times Pr(k_{i+1} = m | k_i = n) Pr(k_i = n, \mathbf{Z}_1^i) &(41)\\
&= \sum_{n=1}^{M} B(z_{i+1}|m) A(m|n) \alpha(\mathbf{Z}_1^i, n) &(42)
\end{aligned}$$

which is a nice recursive formula for $\alpha(\cdot|\cdot)$ when initialized with

$$\begin{aligned}
\alpha(\mathbf{Z}_1^1, m) &= Pr(z_1, k_i = m) &(43)\\
&= \frac{B(z_1|m)p(m)}{\sum_{l=1}^{M} B(z_1|l)p(l)} &(44)
\end{aligned}$$

This is the *forward* part of the forward-backward algorithm. Notice that if all we wanted was what amounted to a *filter* – that is, we want the posterior probability $p(k_i = m|\mathbf{Z}_1^i)$ – then all we need to do is one single "forward" pass and normalize the sum over m of the $\alpha(\mathbf{Z}_1^N, m)$'s to be unity. Similarly, If what we wanted was just $p(\mathbf{Z}_1^i)$ – that would give us the *likelihood* that we might use to test if the model is correct – then all we need do is sum

$\alpha(\mathbf{Z}_1^N, m)$ over m to marginalize that out. That is, in either of these cases we would be done. However, in order to estimate the model we require detailed information about $p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z})$; for that we need the *backward* pass. However, similar to the Kalman Smoother[3] all that is needed is one forward and one backward sweep [4] per iteration.

Similarly, define

$$\beta(\mathbf{Z}_{i+1}^N, m) \equiv p(\mathbf{Z}_{i+1}^N | k_i = m) \tag{45}$$

We can write

$$\begin{aligned}
\beta(\mathbf{Z}_i^N, m) &= p(\mathbf{Z}_i^N | k_{i-1} = m) && (46)\\
&= \sum_{n=1}^{M} p(\mathbf{Z}_{i+1}^N, z_i, k_i = n | k_{i-1} = m) && (47)\\
&= \sum_{n=1}^{M} p(\mathbf{Z}_{i+1}^N | z_i, k_i = n, k_{i-1} = m) \\
&\qquad \times p(z_i | k_i = n, k_{i-1} = m) \\
&\qquad \times Pr(k_i = n | k_{i-1} = m) && (48)\\
&= \sum_{n=1}^{M} Pr(\mathbf{Z}_{i+1}^N | k_i = n) p(z_i | k_i = n) \\
&\qquad \times Pr(k_i = n | k_{i-1} = m) && (49)\\
&= \sum_{n=1}^{M} \beta(\mathbf{Z}_{i+1}^N, n) B(z_i | n) A(n|m) && (50)
\end{aligned}$$

which is a nice recursive formula for $\beta(\cdot|\cdot)$ when initialized with

$$\beta(\mathbf{Z}_{N+1}^N, m) = \frac{1}{N} \tag{51}$$

This is the *backward* part of the forward-backward algorithm. It is typical to scale both forward and backward directions, since underflow often results.

One key fact that is especially interesting is that these quantities give us

[3]Actually there is a nice alternative derivation of the Kalman Smoother that is exactly Baum-Welch.

[4]In the EM (or Baum-Welch) algorithm the estimates for $\mathbf{X}^{(n)} = \{\mathbf{A}, \mathbf{B}, \mathbf{p}\}$ change each iteration. Naturally, therefore, each iteration requires a new forward and backward sweep to determine the requisite $p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z})$.

the marginal probabilities for state occupancies. That is,

$$
\begin{aligned}
Pr(k_i = m|\mathbf{Z}_1^N) &\propto p(k_i = m, \mathbf{Z}_1^N) && (52)\\
&= p(\mathbf{Z}_1^i, \mathbf{Z}_{i+1}^N, k_i = m) && (53)\\
&= p(\mathbf{Z}_{i+1}^N|\mathbf{Z}_1^i, k_i = m)p(\mathbf{Z}_1^i|k_i = m) && (54)\\
&= p(\mathbf{Z}_{i+1}^N|k_i = m)p(\mathbf{Z}_1^i|k_i = m) && (55)\\
&= \beta(\mathbf{Z}_{i+1}^N|k_i = m)\alpha(\mathbf{Z}_1^i|k_i = m) && (56)
\end{aligned}
$$

and normalizing (56) to sum to unity is obvious. We also see from (52) that we can write

$$
\begin{aligned}
p(\mathbf{Z}_1^N) &= \sum_{m=1}^{M} p(k_i = m, \mathbf{Z}_1^N) && (57)\\
&= \sum_{m=1}^{M} \beta(\mathbf{Z}_{i+1}^N|k_i = m)\alpha(\mathbf{Z}_1^i|k_i = m) && (58)
\end{aligned}
$$

which, interestingly, does not depend[5] on i. This is the likelihood of the whole sequence given the model, and can be useful for model testing.

Now we'll need

$$
\begin{aligned}
w_1(m) &\equiv Pr_{\mathbf{X}^{(n)}}(k_i = m|\mathbf{Z}) && (59)\\
&= \beta(\mathbf{Z}_2^N|k_i = m)\alpha(\mathbf{Z}_1^1|k_i = m) && (60)
\end{aligned}
$$

from (56). We'll also need

$$
\begin{aligned}
p(k_{i-1} = n, k_i = m|\mathbf{Z}_1^N) &\propto p(k_{i-1} = n, k_i = m, \mathbf{Z}_1^N) && (61)\\
&= p(\mathbf{Z}_1^{i-1}, z_i, \mathbf{Z}_{i+1}^N, k_{i-1} = n, k_i = m) && (62)\\
&= p(\mathbf{Z}_{i+1}^N|\mathbf{Z}_1^{i-1}, z_i, k_{i-1} = n, k_i = m)\\
&\qquad \times p(z_i|\mathbf{Z}_1^{i-1}, k_{i-1} = n, k_i = m)\\
&\qquad \times p(k_i = m|\mathbf{Z}_1^{i-1}, k_{i-1} = n)\\
&\qquad \times p(\mathbf{Z}_1^{i-1}, k_{i-1} = n) && (63)\\
&= p(\mathbf{Z}_{i+1}^N|k_i = m)p(z_i|, k_i = m)\\
&\qquad \times p(k_i = m|k_{i-1} = n)\\
&\qquad \times p(\mathbf{Z}_1^{i-1}, k_{i-1} = n) && (64)\\
&= \beta(\mathbf{Z}_{i+1}^N|m)B(z_i|m)A(m|n)\alpha(\mathbf{Z}_1^{i-1}, n) && (65)
\end{aligned}
$$

[5]In fact the solution is the same for every i.

Finally, we'll need

$$\begin{aligned} p(k_i = m, z_i = n|\mathbf{Z}_1^N) &\propto p(k_i = m, \mathbf{Z}_1^N)\mathcal{I}(z_i = n) && (66) \\ &= p(\mathbf{Z}_{i+1}^N|k_i = m)p(k_i = m, \mathbf{Z}_1^i) \\ &\qquad \times \mathcal{I}(z_i = n) && (67) \\ &= \beta(\mathbf{Z}_{i+1}^N|m)\alpha(\mathbf{Z}_1^i, m)\mathcal{I}(z_i = n) && (68) \end{aligned}$$

And now we are ready to apply EM.

2.3 The Baum-Welch Algorithm

The Baum-Welch "re-estimation" algorithm was designed to estimate the parameters of an HMM based on one or preferably many sets of observations. The notation below assumes a single time series observation, but the extension to multiple observation sequences is obvious – and indeed estimation of the initial probability will be pretty poor if there is only one times-series observation, since there is only one exemplar. Note that there is no stipulation that the underlying Markov model be in "steady-state" – Baum-Welch works fine for non-stationary HMMs. The Baum-Welch procedure was discovered independently of EM; but it was later noted that it was exactly the EM algorithm applied to an HMM.

We begin by inserting (33) to (2). We have

$$\begin{aligned} Q(\mathbf{X}; \mathbf{X}^{(n)}) &= \int \log\left(p_{\mathbf{X}}(\mathbf{Z}, \mathbf{K})\right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) d\mathbf{K} && (69) \\ &= \sum_{\mathbf{K}} \left(\left(\log(p(k_1)) + \sum_{i=2}^{N} \log\left(A(k_i|k_{i-1})\right) \right.\right. \\ &\qquad \left.\left. + \sum_{i=1}^{N} \log\left(B(z_i|k_i)\right) \right) p_{\mathbf{X}^{(n)}}(\mathbf{K}|\mathbf{Z}) \right) && (70) \end{aligned}$$

Maximizing the Q-function over all these is quite simple; the only "subtlety" (and it isn't very subtle, really) is that we have to apply the Lagrange constraint that all probabilities sum to unity. We get

$$p(m) = w_1(m) \qquad (71)$$

using (60). We also have

$$A(m|n) = \kappa_{A(\cdot|n)} \sum_{i=2}^{N} p(k_{i-1} = n, k_i = m|\mathbf{Z}_1^N) \qquad (72)$$

where the probabilities are from (65) and $\kappa_{A(\cdot|n)}$ is such that

$$\sum_{m=1}^{M} A(m|n) = 1 \tag{73}$$

is normalized. Finally, we get

$$B(l|m) \ = \ \kappa_{B(\cdot|m)} \sum_{i=1}^{N} p(k_i = m, z_i = n|\mathbf{Z}_1^N) \tag{74}$$

where the probabilities are from (68) and $\kappa_{B(\cdot|m)}$ is such that

$$\sum_{l=1}^{L} B(l|m) = 1 \tag{75}$$

is normalized. Baum-Welch says keep doing this iteration until convergence.

ECE 6123
Advanced Signal Processing: Compressive Sensing and Sparseness

Peter Willett

Fall 2017

1 Sparse Representations

Why do image processors transform an image – via multi-resolution (wavelet) transform, discrete Fourier transform (2D-DFT) or its variant the discrete cosine transform (DCT) – prior to coding it for data compaction? It seems fairly intuitive: while in the original (image) domain the energy is distributed evenly amongst all pixels, in the transform domain this is no longer true. For example, it is common to find most of the energy in low spatial frequency components (larger image objects) and much less at higher frequencies (fine detail); and it makes sense to expend many bits to quantize the former and rather fewer to deal with the latter. In fact, *inverse water-filling* from rate-distortion arguments in information theory tell us to do exactly that, and even to ignore completely (no bits at all) components that are smaller than some threshold.

Taken to its limit, this describes a representation that is *sparse*. We may wish to write

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{e} \tag{1}$$

in which $\mathbf{x}$ is the observation vector[1] of dimension $M \times 1$, $\mathbf{A}$ is an a-priori fixed "dictionary" matrix[2] of dimension $M \times N$ (generally $M \ll N$), $\mathbf{s}$ is a vector of dimension $N \times 1$ that contains only S ($S \ll M$) non-zero elements and $\mathbf{e}$ is a small "noise" vector to account for the inaccuracy in the representation. This is clearly quite appealing: to code (approximately, anyway) the data vector $\mathbf{x}$ all we need is a few (S) elements of $\mathbf{s}$, since presumably the *de*-coder already knows $\mathbf{A}$.

[1]This is not quite a correct thing to write in all cases, but wait for Compressive Sensing to go into more detail.

[2]This is an *over-complete* dictionary: there are more columns of $\mathbf{A}$ than should be necessary to span the space, meaning that these columns are necessarily linearly dependent.

So how do we do this? Let's begin by considering the problem

$$\min_{\mathbf{s}} \{\|\mathbf{s}\|_2\} \text{ such that } \|\mathbf{x}-\mathbf{A}\mathbf{s}\|_2 \leq \varepsilon \tag{2}$$

in which $\|\cdot\|_2$ refers to the Euclidean distance (2-norm). With a few Lagrange multipliers and use of the Woodbury formula we have

$$\mathbf{s} \;=\; (\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{x} \tag{3}$$

where

$$\left\|(\mathbf{I}+\lambda^{-1}\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{x}\right\|_2 = \varepsilon \tag{4}$$

solves implicitly for λ. This is complicated and not what we want anyway – the point in showing it is to demonstrate that there will be no special sparsity associated with the problem, since $\mathbf{s}$ will in general be fully populated. A real sparse solution is found from

$$\min_{\mathbf{s}} \{\|\mathbf{s}\|_0\} \text{ such that } \|\mathbf{x}-\mathbf{A}\mathbf{s}\|_2 \leq \varepsilon \tag{5}$$

where the 0-norm is the number of non-zero elements. The problem with this solution is that in general one may need to test each of the 2^N combinations of non-zero elements of $\mathbf{s}$, and that is clearly not an option for computational reasons. Actually you can do OK with a greedy algorithm, that amounts to finding the best column of $\mathbf{A}$, then next-best, and so on: this is called matching-pursuit (MP) and has complexity $\mathcal{O}(MS)$. A variant of MP that works a little better is *orthogonal* matching pursuit (OMP): after a new column of $\mathbf{A}$ is discovered all non-zero coefficients in the set so far discovered are re-computed. This reduces the error somewhat, and the complexity remains approximately the same.

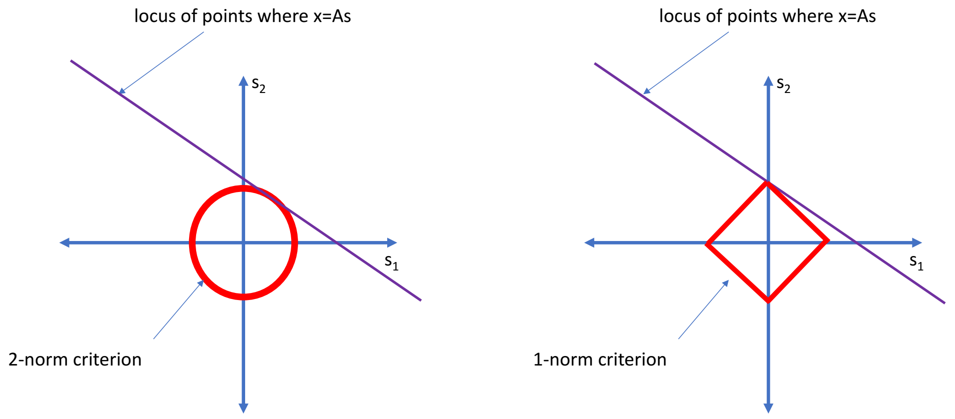
However, researchers have found that an in-between solution is perhaps preferable. Consider the problem

$$\min_{\mathbf{s}} \left\{\|\mathbf{s}\|_p\right\} \text{ such that } \mathbf{x}=\mathbf{A}\mathbf{s} \tag{6}$$

where of course

$$\|\mathbf{s}\|_p \;=\; \left(\sum_{m=1}^{M} |s[m]|^p\right)^{\frac{1}{p}} \tag{7}$$

This assumes that an exact solution (not necessarily sparse) exists, but the notion suggests the following cartoon.



On the left is what we have with $p = 2$ – minimizing the standard Euclidean norm does not encourage a sparse solution. On the right is the L_1 ("Manhattan distance") norm, and it is seen that a sparse solution is indeed the most likely result, since it will be at a "corner" of the constraint set. Actually a reformulation of (6) is what is actually posed:

$$\min_{\mathbf{s}} \left\{ \|\mathbf{x} - \mathbf{A}\mathbf{s}\|_2^2 \right\} \text{ such that } \|\mathbf{s}\|_1 \leq \varepsilon \tag{8}$$

where

$$\|\mathbf{s}\|_1 = \sum_{m=1}^{M} |s[m]| \tag{9}$$

Problem (8) is solved iteratively by a classical subgradient technique from statistics, and is called Least Absolute Shrinkage and Selection Operator (LASSO). And there seems to be some success with (6) using $p = 0.5$.

2 Compressive Sensing

Consider a communications application in which the channel is being probed. The channel is made up of multiple paths, such that

$$h[n] = \sum_{k=1}^{K} \alpha_k \delta[n - n_k] \tag{10}$$

and it is assumed here that the sampling rate is high enough that Nyquist rate sampling can incorporate all possible delays – hence we use discrete time to represent it. Naturally, you want to characterize the channel. One approach is simply to measure $h[n]$ – perhaps by inserting a very narrow pulse – and to look for peaks. One may need a lot of samples.

Another approach would be to apply a sinusoid of frequency ω_1; one observes

$$H(\omega_1) = \sum_{k=1}^{K} \alpha_k e^{-j\omega_1 t_k} \tag{11}$$

If one applies another sinusoid of frequency ω_2 one observes $H(\omega_2)$; and so on. Notice that all paths $\{\alpha_k, t_k\}$ contribute to all observations; and that one really needs only $L \geq K$ probing frequencies in order to have enough information to characterize the channel. Notice that we can write

$$L \text{ frequencies} \left\{ \begin{pmatrix} H(\omega_1) \\ H(\omega_2) \\ \vdots \\ H(\omega_L) \end{pmatrix} \right. =$$

$$\underbrace{\begin{pmatrix} e^{-j\omega_1 0} & e^{-j\omega_1} & e^{-j\omega_1 2} & \dots & e^{-j\omega_1 (N-1)} \\ e^{-j\omega_2 0} & e^{-j\omega_2} & e^{-j\omega_2 2} & \dots & e^{-j\omega_2 (N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{-j\omega_L 0} & e^{-j\omega_L} & e^{-j\omega_L 2} & \dots & e^{-j\omega_L (N-1)} \end{pmatrix}}_{N \text{ samples} = N \text{ possible paths}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_1 \\ 0 \\ \vdots \\ 0 \\ \alpha_2 \\ 0 \\ \vdots \end{pmatrix} \tag{12}$$

where the vector on the RHS is clearly sparse: we could write (12) as (1) with $\mathbf{e} = 0$ and the n^{th} column of the dictionary matrix containing the response of a path at time sample n to the various probing frequencies. Notice how many fewer samples (channel-probings) are needed.

At a somewhat more abstract level what we have done is to posit that we have

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{13}$$

with a sparse $\mathbf{s}$, but that in fact we observe

$$\mathbf{y} = \mathbf{B}\mathbf{x} \tag{14}$$

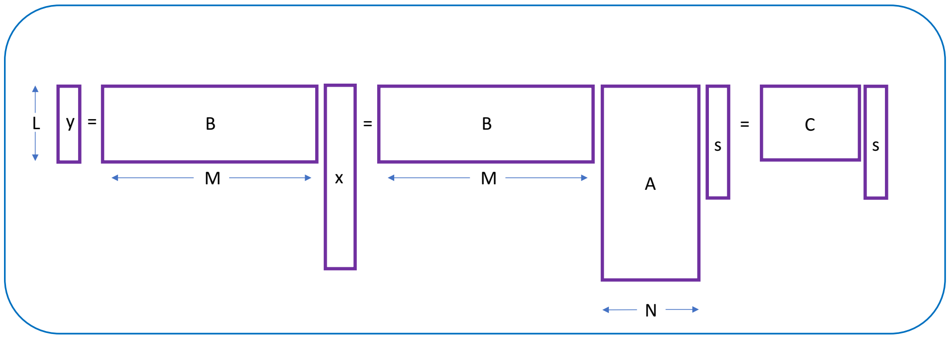
meaning that we have

$$\mathbf{y} = \mathbf{C}\mathbf{s} \tag{15}$$

where

$$\mathbf{C} = \mathbf{BA} \tag{16}$$

In the channel-probing example just given, $\mathbf{x}$ is the impulse response, which we do not know but would like to know. The *sparse* vector $\mathbf{s}$ is made up of mostly 0's but also the α's in the appropriate locations. These locations are delayed impulses, meaning that the i^{th} column of $\mathbf{A}$ is really just an impulse at the i^{th} delay – that is, it is $\delta[n - n_i]$, where n increments down the column. We do not observe $\mathbf{x}$ directly, of course; instead we observe $\mathbf{Bx}$ which is the response of the impulse response at a particular frequency (see (12)). We observe this at several frequencies; that is, we observe $\mathbf{y}$ which is made up of $H(\omega)$'s.

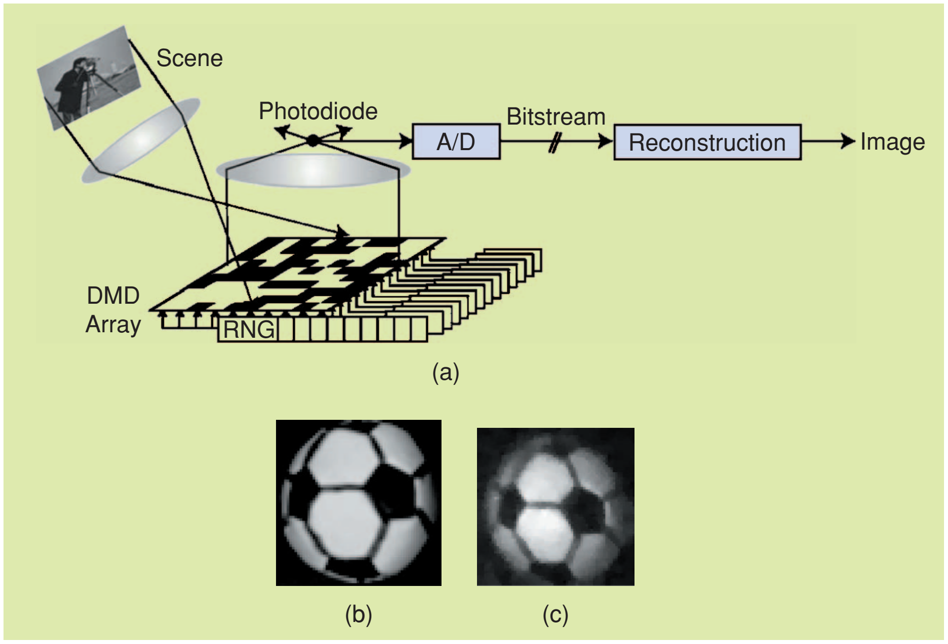


The notional figure is as above. The figure assumes that $N \ll M$ but this need not be true and we could have N quite large – in that case the matrix $\mathbf{C}$ would be "fat."

The above figure leads us to the rather interesting idea of the "single-pixel" camera. In the digital communications notional example the columns of $\mathbf{C}$ – the *dictionary* that we are trying to build our response from – is constructed carefully. That is, each column corresponds to the response that we would observe at the frequencies probed for a specific delay. But indeed much of the more recent success of such compressive sensing has come via columns that are designed haphazardly – using Matlab's *randnormal* function, for example.

Suppose that each row of $\mathbf{B}$ represents a pseudo-random photographic *mask*, and the corresponding element in the observation vector $\mathbf{y}$ is the amount of light received at the single pixel (a photo-diode?) as the true image $\mathbf{x}$ is applied to the mask. The matrix $\mathbf{A}$ can be anything, and often is assumed itself to be random. The after solving the sparse problem $\mathbf{y} = \mathbf{Cs}$ the resultant vector $\mathbf{s}$ is applied to the dictionary matrix $\mathbf{A}$ to render an approximation to the *actual image* $\mathbf{x}$. It seems to work quite well.

Please see the figure below, taken from "Compressive Sensing" by R. Baraniuk, *IEEE Signal Processing Magazine*, pp. 118-124, July 2007.



[FIG3] (a) Single-pixel, compressive sensing camera. (b) Conventional digital camera image of a soccer ball. (c) 64 $\times$ 64 black-and-white image $\hat{x}$ of the same ball ($N = 4{,}096$ pixels) recovered from $M = 1{,}600$ random measurements taken by the camera in (a). The images in (b) and (c) are not meant to be aligned.

In order that this work we need to make sure that we have the correct sparse vector $\mathbf{s}$ of sparseness S. Suppose that we have $\mathbf{y} = \mathbf{C}\mathbf{s}_1$ and $\mathbf{y} = \mathbf{C}\mathbf{s}_2$ for $\mathbf{d} = (\mathbf{s}_1 - \mathbf{s}_2) \neq \mathbf{0}$. Then we know that $\mathbf{C}\mathbf{d} = \mathbf{0}$ which implies that there is some group of $2S$ columns of $\mathbf{C}$ that are linearly dependent. To avoid this, we must insist that all such collections of $2S$ columns of $\mathbf{C}$ be linearly-independent. This amounts to the (more complicated) *restricted isometry property* (RIP) that gives fairly exact results.