LPC Analysis
Prediction & Regression

• We hypothesize that there is some systematic relation between the values of two variables, X and Y.

• If this hypothesis is true, we can (partially) predict the observed values of one (Y) from observations of the other.

• When we do this we say we regress the values of the dependent variable, Y, onto the independent variable, X.
Linear Regression

• The simplest form of systematic relationship between 2 variables is a linear one.

• This means that the predicted values of $Y$ fall on a straight line, which can be characterized by a slope ($b$) and a $Y$-intercept ($I$).

$Y_{\text{pred}} = b \times X + I$

$\text{Res} = Y - Y_{\text{pred}}$

$Y = b \times X + I + \text{Res}$
Example: predicting coffee preferences

- Suppose I want to learn how to maximize my coffee pleasure.

- I can rate my morning coffee on a scale of 1-7 and record some the values of some potential predictor variables like:
  - days since ROAST
  - hours since GRINDing
  - DARKness of roast

- Then I can see how well these variables predict my coffee preferences.
Regression Results: ROAST

```matlab
>> plotReg (roast, rating);
```

![Plot with linear regression line and annotated slope and RsQ]

- Slope: -0.39375
- RsQ: 0.83531

Ypred
What is RSQ?

- RSQ = \( R^2 \), where \( R = \)correlation coefficient, which is a measure of the strength of the association between the two variables.

- Also equal to the percentage of the variance of \( Y \) that is explained by the variance in \( X \).

- RSQ = variance (\( Y_{\text{pred}} \))/ variance(\( Y \))

- 1-RSQ = variance(Res)/variance(\( Y \))
function [b res rsq] = plotReg (X,Y)

% Louis Goldstein
% 27 October 2009
%
% input arguments:
% X = independent variable column vector
% Y = dependent variable column vector
%
% output arguments:
% b = vector of regressions coefficients
% one for each column of X, plus a constant
% Res = vector of residuals
% Rsq = R^2

[b res rsq] = regress (X,Y);

% calculate line showing Y-Pred for range of X values
xline = linspace(0,max(X),100);
yline = b(1)*xline + b(2);

% plot data points and regression line
plot (X,Y, 'or', xline,yline,'b');

title (['slope = ', num2str(b(1)), '; RSQ = ', num2str(rsq)]);
How does regression work?

1) \[ Y = Xb + I \]

- **Matrix form:**

2) \[ Y = X*b \]
   
   \( X \) is a matrix = \([X_{\text{data}} \text{ ones}]\);  
   
   \( b \) is a column vector, one row for each col of \( X \).

3) \[ X^{-1}Y = X^{-1}X*b \]

4) \[ X^{-1}Y = b \]

5) \[ b = X\backslash Y \]
Regression in Matlab

function [b, res, Rsq] = regress (X,Y)

% Append a vector of ones to X
% The coefficient associated with this vector is the constant.
N = length(X);
X = [X ones(N,1)];

% Do the regression
b = X\Y;

% Calculate residuals
Ypredicted = X * b;
res = Y - Ypredicted;

% Calculate Rsq
SSreg = (std(Ypredicted) .^ 2) .* (N-1);
SStot = (std(Y) .^ 2) .* (N-1);
Rsq = SSreg/SStot;'
Results: GRIND

```matlab
>> plotReg (grind, rating);
```

- Reduced slope
- Reduced RSQ
Results: DARK

\[ \text{plotReg (dark, rating)}; \]

- Reduced slope
- Reduced RSQ
Multiple Regression

- If we want as good a prediction as possible of the dependent variable, we could attempt to fit a single linear equation to all the possible X-variables:

\[ Y = b_1 X_1 + b_2 X_2 + b_3 X_3 + ... + b_M X_M + K + \text{Res} \]

- b coefficients will be related to b coefficients of individual regressions (they will be equal, if variables are uncorrelated).

- RSQ will always be higher as variables are added, not every one will result in a significant increase.
Coffee multiple regression

- \([b \text{ Res RSQ}] = \text{regress([roast grind dark], rating)};\]
- **b coefficients:**
  - ROAST -0.3597
  - GRIND -0.1602
  - DARK 0.1402

- RSQ = 0.8946
Linear Prediction of Speech

• Because speech is often periodic (with a variety of frequencies), there will be predictability (correlation) between sample points at different lags.

• We can do multiple regression of a sample speech samples, attempting to predict their values from the values of the M previous sample points:

\[
Y(k) = b(1)Y(k-1) + b(2)Y(k-2) + \ldots + b(M)Y(k-M) + \text{Res}
\]

• We perform this over a window of data during which correlations (and frequency components) are assumed not to change (5-10 ms).
Matrix Construction

Example: M=5, window = 20 points

<table>
<thead>
<tr>
<th>Y =</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0023</td>
</tr>
<tr>
<td>-0.0033</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0016</td>
</tr>
<tr>
<td>-0.0024</td>
</tr>
<tr>
<td>-0.0036</td>
</tr>
<tr>
<td>-0.0024</td>
</tr>
<tr>
<td>-0.0016</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0033</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0016</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0033</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0016</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0033</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
<tr>
<td>-0.0016</td>
</tr>
<tr>
<td>-0.0027</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X =</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0036 -0.0043 -0.0038 -0.0023 -0.0023</td>
</tr>
<tr>
<td>-0.0023 -0.0036 -0.0043 -0.0038 -0.0023</td>
</tr>
<tr>
<td>-0.0033 -0.0023 -0.0036 -0.0043 -0.0038</td>
</tr>
<tr>
<td>-0.0027 -0.0033 -0.0023 -0.0036 -0.0043</td>
</tr>
<tr>
<td>-0.0016 -0.0027 -0.0033 -0.0023 -0.0036</td>
</tr>
<tr>
<td>-0.0024 -0.0016 -0.0027 -0.0033 -0.0023</td>
</tr>
<tr>
<td>-0.0036 -0.0024 -0.0016 -0.0027 -0.0033</td>
</tr>
<tr>
<td>-0.0029 -0.0036 -0.0024 -0.0016 -0.0027</td>
</tr>
<tr>
<td>-0.0029 -0.0036 -0.0024 -0.0016 -0.0027</td>
</tr>
<tr>
<td>-0.0031 -0.0029 -0.0036 -0.0024 -0.0016</td>
</tr>
<tr>
<td>-0.0048 -0.0031 -0.0029 -0.0036 -0.0024</td>
</tr>
<tr>
<td>-0.0056 -0.0048 -0.0031 -0.0029 -0.0036</td>
</tr>
<tr>
<td>-0.0056 -0.0048 -0.0031 -0.0029 -0.0036</td>
</tr>
<tr>
<td>-0.0070 -0.0056 -0.0048 -0.0031 -0.0029</td>
</tr>
<tr>
<td>-0.0090 -0.0070 -0.0056 -0.0048 -0.0031</td>
</tr>
<tr>
<td>-0.0070 -0.0056 -0.0048 -0.0031 -0.0029</td>
</tr>
<tr>
<td>-0.0090 -0.0070 -0.0056 -0.0048 -0.0031</td>
</tr>
<tr>
<td>-0.0070 -0.0056 -0.0048 -0.0031 -0.0029</td>
</tr>
<tr>
<td>-0.0090 -0.0070 -0.0056 -0.0048 -0.0031</td>
</tr>
<tr>
<td>-0.0096 -0.0137 -0.0090 -0.0070 -0.0056</td>
</tr>
<tr>
<td>-0.0078 -0.0096 -0.0137 -0.0090 -0.0070</td>
</tr>
<tr>
<td>-0.0109 -0.0078 -0.0096 -0.0137 -0.0090</td>
</tr>
<tr>
<td>-0.0109 -0.0078 -0.0096 -0.0137 -0.0090</td>
</tr>
<tr>
<td>-0.0101 -0.0109 -0.0078 -0.0096 -0.0137</td>
</tr>
<tr>
<td>-0.0084 -0.0101 -0.0109 -0.0078 -0.0096</td>
</tr>
<tr>
<td>-0.0071 -0.0084 -0.0101 -0.0109 -0.0078</td>
</tr>
<tr>
<td>-0.0071 -0.0084 -0.0101 -0.0109 -0.0078</td>
</tr>
</tbody>
</table>
function [acoef, Res] = lpc_demo (signal, srate, winsize, M, ibeg)

% Input arguments:

% signal    vector containing time series data
% srate     sample rate in Hz
% winsize   window size in number of samples
% M         number of coefficients
% ibeg      beginning sample number

% iend is last sample in window
iend = ibeg+winsize-1;

% Y is a vector of winsize data points from signal
% Y is the dependent variable for the regression
Y = signal(ibeg:iend);

% matrix X contains the independent variables for the regression.
% It contains M columns, each of which contains
% elements of signal (y) delayed progressively by one sample

for i=1:M
    X(:,i) = signal(ibeg-i:iend-i);
end

% perform the regression
% the coefficients are the weights that are applied to each
% of the M columns in order to predict Y.
% Since the columns are delayed versions of Y,
% the weights represent the best prediction of Y from the
% previous M sample points.
[acoef, Res] = regress(X,Y);
% The predicted signal is given by subtracting the residual % from Y
Pred = Y - Res;

% Plot original window and predicted
subplot (211), plot (signal(ibeg:iend))
title ('Original Signal')
subplot (212), plot (Pred)
title ('Predicted Signal')
subplot
pause

% Plot original window and residual
subplot (211), plot (signal(ibeg:iend))
title ('Original Signal')
subplot (212), plot (Res)
title ('Residual')
subplot
pause

% Get rid of the M+1st term returned by the regression.
% It just represents the mean (DC level of the signal).
acoef(M+1) = [ ];
Test with sinusoid

\[
s1000 = \text{make\_tone}(10000,1000);
\]
\[
test = s1000+.01*\text{randn}(\text{size}(test));
\]
a = lpc_sine (test', 10000, 100, 12, 25);

Original Signal

Predicted Signal
a = lpc_sine ('test', 10000, 100, 12, 25);
\[
a = \text{lpc\_sine (test', 10000, 100, 12, 25)};
\]
\[
\text{bar(abs(a))}
\]
Interpretation of Regression

(1) \( Y = b_1 X_1 + b_2 X_2 + b_3 X_3 + \ldots + b_M X_M + K + \text{Res} \)

(2) \( Y = a_1 Y_1 + a_2 Y_2 + a_3 Y_3 + \ldots + a_M Y_M + K + \text{Res} \)

- Residual is the signal after periodic short-time resonances (due to the vocal tract are removed), so it represents what is not predictable by a short-time filter, which are the pulses of the source.

- By re-arranging (2), we can create an inverse filter that takes a speech sample as input and produces a sequence of E pulses (ignore K).

(3) \( E = \text{Res} = Y - a_1 Y_1 - a_2 Y_2 - a_3 Y_3 + \ldots - a_M Y_M \)
Inverse and Forward Filters

- The inverse filter thus has the following form:

\[ A(z) = 1 -a_1 z^{-1} -a_2 z^{-2} -a_3 z^{-3} + ... -a_M z^{-M} \]

- \( E = A(z) \ast Y \)

- But we can also rearrange this to derive the forward filter, that will generate the output \( Y \) from \( E \) and the filter:

\[ Y = E / A(z) \]

- A coefficients now in the denominator
The returned coefficients (acoef) can be used to predict Y(k) from preceding M samples:
E1: $Y(k) = a(1)Y(k-1) + a(2)Y(k-2) + \ldots + a(M)Y(k-M) + \text{Res}$
We want what is called the INVERSE filter, which will take Y as input and output the Res signal.
That is, it will take all the structure out of the signal and give us an impulse sequence. This can be obtained from equation E1 by moving Y(k) and Res to the other side of E1:
E2: $\text{Res} = 1*Y(k) - a(1)*Y(k-1) - a(2)*Y(k-2) - \ldots - a(M)*Y(k-M)$
Thus, the first coefficient of the inverse filter is 1; the rest are the negatives of the coefficients returned.
Note -acoef' is used to convert from a column to row vector:
acoef = [1 -acoef'];

Plotting the magnitude of the transfer function for the INVERSE FILTER:
freqz will return value of the transfer function, h, for a given filter numerator and denominator at npoints along the frequency scale.
The frequency of each point (in rad/sample) is returned in w.
num = acoef;
den = 1;
npoints = 100;
[h, w] = freqz(num,den,npoints);
plot (w*srate./(2*pi), 20*log10(abs(h)))
grid
xlabel ('Frequency in Hz')
ylabel ('Magnitude of Transfer Function')
title ([''INVERSE FILTER: M =', num2str(M), ' winsize = ', num2str(winsize), ' Beginning sample = ', num2str(ibeg)])
pause
% Plot magnitude of transfer function if the coefficients are
% used as coefficients of the FORWARD filter (in denominator).
% freqz will return value of the transfer function, h,
% for a given filter numerator and denominator
% at npoint along the frequency scale.
% The frequency of each point (in rad/sample) is returned in w.
num = 1;
den = acoef;
npoints = 100;
[h, w] = freqz(num, acoef, npoints);
plot (w*srate./(2*pi), 20*log10(abs(h)))
grid
xlabel ('Frequency in Hz')
ylabel ('Log Magnitude of Transfer Function')
title ([ 'FORWARD FILTER: M = ', num2str(M), '   winsize = ', num2str(win_size), '   Beginning sample = ', num2str(ibeg)])
Example

<table>
<thead>
<tr>
<th>F</th>
<th>BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>290</td>
<td>60</td>
</tr>
<tr>
<td>2070</td>
<td>200</td>
</tr>
<tr>
<td>2960</td>
<td>400</td>
</tr>
<tr>
<td>3300</td>
<td>250</td>
</tr>
<tr>
<td>3750</td>
<td>200</td>
</tr>
</tbody>
</table>

```bash
>> iy=synthesize_only (10000,F,BW,100,.5);
```
```matlab
[acoef, Res] = lpc_demo (iy', 10000, 200, 12, 20);
```
 inverse filter

>> Frec=formants (acoef, 10000)
Frec =
1.0e+03 *
 0.2924     F
 2.0714     290
 2.9626     2070
 3.3014     2960
 3.7507     3300
 3.7507     3750

 forward filter
Choice of M

• We saw that a digital resonator (formant) can be represented with 2 feedback filter coefficients.
• So M needs to be at least equal to 2*NF, where NF is the expected number of formants, given the length of the VT and SR.
• Example:
  For a VT of 17 cm, formants are spaced 1000Hz apart, on average.
  So for a sr of 10000, there will be 5 formants or 10 coefficients
• $M = \frac{sr}{1000}$
• But the effect of the glottal resonator and other things (anti-resonance) need an additional 4 coefficients
• $M = (\frac{sr}{1000}) + 4$, and subtract 2 for a somewhat shorter vocal tract
Tuesday, March 24, 15
‘I, yo-yo, wow, why’
LPC of running speech: windowing

- We can measure how LPC parameters and the F and BW values we calculate from them change over the course of an utterance.
- Analysis frames need to be long enough to include at least one full cycle of the lowest VT resonance, and better if it is somewhat longer than that. F1 in low vowels is 200 Hz, which would make 5ms the minimum duration.
- We also want the formants to smoothly (except near stop closures and releases), and not to show jitter from one frame to frame. So the longer window, the better.
- But we also want to track meaningful VT changes smoothly, and if the window is too long, we could get discrete jumps from one frame to the next.
- **Solution:** Use a relatively long windows (e.g., 20 ms), but slide the windows through time at a faster rate (e.g., 10 ms), so successive windows overlap
Example

- We also want to weight the points near the middle of window more heavily in regression (reducing the effect of seeing the window’s frequencies in the output). So we multiply the signal in a window by a hamming weighting function.
function [A, G] = get_lpc (signal, srate, M, window, slide)

% get_lpc
% Louis Goldstein
% March 2005
%
% Output arguments:
% A filter coefficients (M+1 rows x nframes columns)
% G Gain coefficients (vector length = nframes)
%
% input arguments:
% signal signal to be analyzed
% srate sampling rate in Hz
% M LPC order (def. = srate/1000 +4)
% window size of analysis window in ms (def. = 20)
% slide no. of ms. to slide analysis window for each frame (def. = 10)

if nargin < 3, M = floor(srate/1000) + 4; end
if nargin < 4, window = 20; end
if nargin < 5, slide = 10; end

samp_tot = length(signal);
samp_win = fix((window/1000)*srate);
samp_slide = fix((slide/1000)*srate);
nframes = floor(samp_tot/samp_slide) - ceil((samp_win-samp_slide)/samp_slide);

A = []; G = [];

for i = 1:nframes
    begin = 1 + (i-1)*samp_slide;
    [Ai,Gi] = LPC (hamming(samp_win).*signal(begin:begin+samp_win-1),M);
    A = [A Ai']; G = [G Gi];
end

Tuesday, March 24, 15
LPC synthesis

• The LPC filter coefficients can be used to re-synthesize the speech (with an impulse source).

• This can be used to compress the data required to represent speech.

• But also can be used to independently manipulate, $f_0$, formants, intensity.

Original

Re-synthesized
Synthesis method

(1) Produce impulse train

(2) Take frames of samples and filter through the M-order filter defined by the LPC coefficients

(3) Concatenate frames (as in syn1).
Formant Analysis

Plot waveform, Excitation (Gain), Formants

Use gain to zero formant display during silence.