

Lecture 14: Linear Prediction

Mark Hasegawa-Johnson

ECE 537: Speech Processing

- ① Linear prediction
- ② Solving for the LPC Coefficients: General Form
- ③ Implementations: Autocovariance and Autocorrelation Methods
- ④ Gain of the synthesis filter
- ⑤ Roots of the LPC Polynomial as Formant Estimates
- ⑥ Example
- ⑦ Summary

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example
- 7 Summary

Review: Source-Filter Model

$$s[n] = h[n] * u[n]$$

... where ...

- $u[n]$ is the speech excitation function, e.g., derivative of airflow through the glottis, or sound pressure fluctuations created by turbulence at a fricative constriction.
- $h[n]$ is the impulse response of the vocal tract.
- **Today's problem:** $h[n]$ might be hundreds of samples long. Can't we represent it using fewer numbers? After all, it represents only the ringing response of ~ 4 formants...

Resonance

- Resonance is a second-order feedback system.
- The k^{th} resonance can be represented by a second-order IIR filter:

$$s_k[n] = As_{k-1}[n] + Bs_k[n-1] + Cs_k[n-2]$$

- We can represent four or five resonances by just cascading the filters:

$$s[n] = h_5[n] * h_4[n] * h_3[n] * h_2[n] * h_1[n] * u[n]$$

The linear predictive speech synthesis model

The combination of LTI systems is an LTI system, so we can represent the combination of $p/2$ resonant filters by a single p^{th} -order IIR filter:

$$s[n] = \sum_{k=1}^p a_k s[n-k] + Gu[n]$$

... where $e[n] = Gu[n]$ is the excitation, and G is the filter gain.
The spectrum is:

$$S(z) = H(z)E(z) = \frac{E(z)}{1 - \sum_{k=1}^p a_k z^{-k}}$$

... and $H(z) = 1/(1 - \sum a_k z^{-k})$ is called the **LPC Synthesis Filter**.

The prediction filter

We can model the synthesis process by saying that $e[n]$, the excitation, is the difference between $s[n]$ and its predicted value, $\tilde{s}[n]$, given by

$$s[n] = \tilde{s}[n] + e[n]$$
$$\tilde{s}[n] = \sum_{k=1}^p a_k s[n - k]$$

The filter $P(z)$ is called the **LPC Prediction Filter**:

$$\tilde{S}(z) = P(z)S(z)$$
$$P(z) = \sum_{k=1}^p a_k z^{-k}$$

The analysis filter

Another way to say the same thing: We model the excitation, $e[n]$, as being the part of $s[n]$ that cannot be predicted based on a linear combination of the previous p samples:

$$\begin{aligned} e[n] &= s[n] - \tilde{s}[n] \\ &= s[n] - \sum_{k=1}^p a_k s[n-k] \end{aligned}$$

The filter $A(z)$ is called the **LPC Analysis Filter**:

$$E(z) = A(z)S(z)$$

$$A(z) = 1 - P(z) = 1 - \sum_{k=1}^p a_k z^{-k}$$

$$H(z) = \frac{1}{1 - P(z)} = \frac{1}{A(z)}$$

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example
- 7 Summary

By what criterion should we choose a_k ?

We will choose a_k to minimize $e[n] = s[n] - \tilde{s}[n]$. Why does this make sense?

- Speech production model:
 - The impulse, every time the glottis closes, can't be predicted from recent past ($\tilde{s}[n] = \sum_{k=1}^p a_k s[n-k]$) if p is less than a pitch period. If we can zero out all the other samples of $e[n]$, we will be left with a series of impulses, one per pitch period.
 - Unvoiced excitation: $e[n] =$ unpredictable white noise.
Remove everything that's predictable.
- Low-bit-rate speech coding: If we can make $e[n]$ very small, we might be able to code only a few of its samples, with a few bits each, and leave the rest as zeros.

Minimizing the MSE

- Shift or window the speech signal

$$s_n[m] = s[m + n] \quad \text{or} \quad s_n[m] = s[m + n]w[m]$$

- Define the short-time sum-squared error:

$$\begin{aligned}\mathcal{E}_n &= \sum_m e_n^2[m] = \sum_m (s_n[m] - \tilde{s}_n[m])^2 \\ &= \sum_m \left(s_n[m] - \sum_{k=1}^p a_k s_n[m - k] \right)^2\end{aligned}$$

- Differentiate with respect to a_i , set the derivative to zero:

$$\sum_m s_n[m - i]s_n[m] = \sum_{k=1}^p a_k \sum_m s_n[m - i]s_n[m - k], \quad 1 \leq i \leq p$$

Simplify by writing it in terms of autocovariance

Define the short-time autocovariance to be:

$$\phi_n[i, k] = \sum_m s_n[m - i]s_n[m - k]$$

Then the minimum-MSE values of a_k , given by

$$\sum_m s_n[m - i]s_n[m] = \sum_{k=1}^p a_k \sum_m s_n[m - i]s_n[m - k], \quad 1 \leq i \leq p$$

can be written as

$$\phi_n[i, 0] = \sum_{k=1}^p a_k \phi_n[i, k]$$

p linear equations in p unknowns

The minimum-MSE LPC coefficients, p unknowns, are the solution of p simultaneous linear equations, so we can write them all as a single matrix equation:

$$\mathbf{r} = \mathbf{R}\mathbf{a}$$

... which has the solution $\mathbf{a} = \mathbf{R}^{-1}\mathbf{r}$, where

$$\mathbf{r} = \begin{bmatrix} \phi_n[1, 0] \\ \vdots \\ \phi_n[p, 0] \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \phi_n[1, 1] & \cdots & \phi_n[p, 1] \\ \vdots & \ddots & \vdots \\ \phi_n[1, p] & \cdots & \phi_n[p, p] \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix}$$

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods**
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example
- 7 Summary

p linear equations in p unknowns

Remember that the general solution is:

$$\mathbf{a} = \mathbf{R}^{-1}\mathbf{r}$$

where $\mathbf{R}$ is a $p \times p$ matrix. Computing its inverse is $\mathcal{O}\{p^3\}$ (I've read that solutions exist that are $\mathcal{O}\{p^{2.3}\}$, but normal Gaussian elimination is $\mathcal{O}\{p^3\}$). We might ask: Is there any way to reduce the computation?

Autocorrelation solution

Suppose we window the signal, then compute its autocorrelation:

$$R_n[i - k] = \sum_m s_n[m - i]s_n[m - k], \quad s_n[m] = s[n + m]w[m]$$

This function is called the autocorrelation, and it has the property that it only depends on $|i - k|$:

$$R_n[i - k] = \sum_m s_n[m - |i - k|]s_n[m]$$

Autocorrelation $\Rightarrow$ Toeplitz matrix

Since $\phi_n[i, k] = R_n[i - k]$ only depends on $|i - k|$, the matrix $\mathbf{R}$ has a special structure called **Toeplitz**:

$$\mathbf{r} = \mathbf{R}\mathbf{a}$$

$$\mathbf{R} = \begin{bmatrix} R_n[0] & R_n[1] & \cdots & R_n[p-1] \\ R_n[1] & R_n[0] & \cdots & R_n[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_n[p-1] & R_n[p-2] & \cdots & R_n[0] \end{bmatrix}$$

An equation $\mathbf{a} = \mathbf{R}^{-1}\mathbf{r}$ with a Toeplitz $\mathbf{r}$ can be solved in $\mathcal{O}\{p^2\}$ using a special equation called the Levinson-Durbin recursion.

What are the disadvantages of the autocorrelation solution?

The disadvantage of autocorrelation is that we are predicting each sample only using previous samples of the **windowed** speech signal:

$$e_n[m] = s_n[m] - \sum_k a_k s_n[m - k]$$

$$s_n[m] = s[n + m]w[m]$$

- The sample at the beginning of the window, $s_n[0]$, is being predicted based on zero history— $s_n[-p \dots -1] = 0$!
- Since the error has this built-in artifact, the coefficients will have some built-in error.
- This problem is most important if $w[n]$ is short.

The autocovariance solution: Predict from samples outside the window

The autocovariance solution is to predict each sample possibly using samples from before the start time of the window:

$$s_n[m] = s[n + m]$$

$$s_n[-1] = s[n + m - 1] \neq 0$$

$$e_n[m] = s[n + m] - \sum_{k=1}^p a_k s[n + m - k], \quad 0 \leq m \leq L - 1$$

- Autocovariance gives more accurate coefficients for short windows, but. . .
- The matrix is no longer Toeplitz (e.g., $\phi_n[1, 1] \neq \phi_n[2, 2]$), so solving $\mathbf{a} = \mathbf{R}^{-1}\mathbf{r}$ is an $\mathcal{O}\{p^3\}$ operation.
- More important in real-world applications: autocorrelation gives an $H(z)$ that is guaranteed stable, but autocovariance doesn't.

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter**
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example
- 7 Summary

The LPC synthesis model

- Speech is synthesized from its own prediction:

$$s[n] = e[n] + \sum_{k=1}^p a_k s[n-k]$$

$$e[n] = Gu[n]$$

- The gain is related to the prediction error as:

$$\begin{aligned}\mathcal{E}_n &= \sum_m e^2[m] \\ &= G^2 \sum_m u^2[m]\end{aligned}$$

So if we know $\sum u^2[m]$, then

$$G = \sqrt{\frac{\mathcal{E}_n}{\sum_m u^2[m]}}$$

What is $\sum u^2[m]$?

- **Voiced speech:** Each frame has one or two or three glottal closure events. Let's just say "one." Then $u[m] = \delta[m - n_0]$, where n_0 is whatever time at which the impulse occurs. Therefore:

$$\sum u^2[m] = 1$$

- **Unvoiced speech:** If $u[m]$ is zero-mean, unit-variance white noise, then

$$E \left[\sum_{m=0}^{L-1} u^2[m] \right] = \sum_{m=0}^{L-1} E [u^2[m]] = L$$

We can make $\sum u^2[m] = 1$ if we use a Gaussian noise process with a variance of

$$E [u^2[m]] = \frac{1}{L}$$

Finding G

Now that we have $\sum u^2[m] \approx 1$ in both cases, we can find G simply as

$$G^2 = \frac{\sum e^2[m]}{\sum u^2[m]} = \frac{\mathcal{E}_n}{1}$$

So the LPC synthesis process is:

- 1 Take an impulse train, or a variance- $\frac{1}{L}$ white noise process, and scale it up by $G = \sqrt{\mathcal{E}_n}$ to get $e[n]$.
- 2 Pass it through the synthesis filter to make

$$s[m] = e[m] + \sum_{k=1}^p a_k s[m-k]$$

How do we find $\mathcal{E}_n$?

Any of these formulas should give the same answer:

$$\begin{aligned}\mathcal{E}_n &= \sum_m (s_n[m] - \tilde{s}_n[m])^2 \\ &= \sum_m \left(s_n[m] - \sum_{k=1}^p a_k s_n[m-k] \right)^2 \\ &= \sum_m s_n^2[m] - \sum_{k=1}^p a_k \sum_m s_n[m] s_n[m-k] \\ &= \phi_n[0,0] - \sum_{k=1}^p a_k \phi_n[0,k]\end{aligned}$$

For the autocorrelation method, the last equation simplifies to

$$G^2 = R_n[0] - \sum_{k=1}^p a_k R_n[k] = R_n[0] - \mathbf{a}^T \mathbf{r}$$

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates**
- 6 Example
- 7 Summary

Roots of the LPC Polynomial

We can factor $A(z)$ as:

$$\begin{aligned} H(z) &= \frac{G}{A(z)} \\ &= \frac{G}{1 - \sum_k a_k z^{-k}} \\ &= \frac{G}{\prod_{i=1}^p (1 - z_i z^{-1})} \\ &= \frac{Gz^p}{\prod_{i=1}^p (z - z_i)} \end{aligned}$$

where z_i is the i^{th} root of the LPC polynomial.

Formant tracking

- The speech formants F_i , with bandwidths B_i , tend to produce very large peaks in the spectrum. Therefore, at least eight of the LPC roots tend to track the formants in complex conjugate pairs as $z_i = z_{p-i}^* = e^{-\sigma_i + j\omega_i}$, where

$$\omega_i = -\omega_{p-i} = \frac{2\pi F_i}{F_s}$$

$$\sigma_i = \sigma_{p-i} = \frac{\pi B_i}{F_s}$$

- It's usually good to set $p \approx 10$ or more, so that two of the poles can be used to track overall spectral tilt. These poles will usually wind up being real-valued, i.e., $\sigma_i = 0$. Typically one of them will have a small value of σ_i (representing an overall lowpass spectral shape), while the other has a much higher σ_i (representing the frequency at which the speech spectrum starts to drop off more quickly).

Example of formant tracking

For some formant-tracking examples, try running Praat:
<https://www.fon.hum.uva.nl/praat/>

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example**
- 7 Summary

Example

Consider a one-tap predictor:

$$s[n] = e[n] + as[n - 1]$$

What is the minimum-MSE predictor coefficient, a ? You have almost certainly seen such a predictor coefficient in one of your statistics or linear algebra courses in the past.

Outline

- 1 Linear prediction
- 2 Solving for the LPC Coefficients: General Form
- 3 Implementations: Autocovariance and Autocorrelation Methods
- 4 Gain of the synthesis filter
- 5 Roots of the LPC Polynomial as Formant Estimates
- 6 Example
- 7 Summary**

Summary

- LPC models the resonant structure of speech using a p^{th} -order linear predictor:

$$s[n] = e[n] + \sum_{k=1}^p a_k s[n-k]$$

- Minimizing the mean-squared error results in p linear equations in p unknowns, which can be solved as

$$\mathbf{r} = \mathbf{R}\mathbf{a}$$

- In almost all cases, the formant frequencies (and their bandwidths) are among the poles of the LPC synthesis filter:

$$H(z) = \frac{G}{A(z)} = \frac{G}{1 - \sum_k a_k z^{-k}} = \frac{Gz^p}{\prod_{i=1}^p (z - z_i)}$$