Lecture 19: Exam 2 Review

Mark Hasegawa-Johnson

ECE 537, Fall 2022

Outline

Admin Issues

- Exam is in class, Monday. If you need remote exam or conflict exam, tell me.
- One page handwritten notes, front and back.
- No calculators.

Outline

Dynamic Time Warping

- Similarity of two words is defined to be the maximum, among all possible alignments, of the average similarity of the aligned spectra.
- This is computed by dynamic programming (DP):

$$A_{i,k} = \max\left(A_{i-1,k}, A_{i,k-1}, a_{i,k} + A_{i-1,k-1}\right)$$

- Similarity of any pair of spectra is $a_{i,k} = \frac{2}{2+\rho_{i,k}^2}$,

$$\rho_{i,k} = \sqrt{\sum_{d=1}^{5}\left(\ln\left(\frac{E_0^{(i)}}{E_d^{(i)}}\right) - \ln\left(\frac{E_0^{(k)}}{E_d^{(k)}}\right)\right)^2}$$

Outline

1. Admin Issues
2. Velichko & Zagoruyko: DTW
3. Atal: LPC
4. Rabiner: HMM

All-Pole Filters

$$H(z) = \frac{1}{1-P(z)} = \frac{1}{1-\sum_{k=1}^{p} a_k z^{-1}} = \frac{1}{\prod_{m=1}^{p}(1-p_m z^{-1})}$$

This can be factored into second order sections with complex conjugate pole pairs; then we can get the impulse response as:

$$H(z) = \frac{C_1}{1-p_1 z^{-1}} + \frac{C_1^*}{1-p_1^* z^{-1}}$$
$$h[n] = C_1 p_1^n u[n] + C_1^* (p_1^*)^n u[n]$$

LPC

- Orthogonality principle: a_k minimizes

$$\sum_{n=0}^{N-1} e^2[n] = \sum_{n0}^{N-1}\left(s[n] - \sum_{m=1}^{p} a_m s[n-m]\right)^2$$

if and only if $e[n] \perp s[n-k]$, meaning

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

- p linear equations in p unknowns:

$$\vec{a} = \Phi^{-1}\vec{c}$$

Parcor Coefficients

- **Cholesky Decomposition:** find L_m such that $\Phi_m = L_m L_m^T$.
- Then the equation $\Phi\vec{a} = \vec{c}$ can be solved in two steps:

$$\vec{q}_m = L_m^{-1}\vec{c}_m$$
$$\vec{a}_m = L_m^{-T}\vec{q}_m$$

- The elements of $\vec{q}_m$ are unchanged from one filter order to the next, so you can perform that first step iteratively, for increasing values of m.
- The Parcor coefficients, $k_m = q_m/\sqrt{\epsilon_m}$, are bounded to $|k_m| < 1$ if and only if the filter is stable, so you can guarantee stability by quantizing k_m instead of a_m.

Pitch Predictor

The pitch predictor turns a Gaussian white-noise-like signal, $v[n]$, into a signal with pitch periodicity, by

$$d[n] = v[n] + \sum_{m=1}^{3} \beta_m d[n - (P + m - 2)]$$

where P, the pitch period, needs to be calculated by some other method, e.g., by finding the peak of the autocorrelation.

Perceptual Noise Weighting

Noise can be perceptually weighted, with a notch at each formant frequency, so that the quantizer is encouraged to shift noise toward the formants, where it will be masked:

$$1 - R(z) = \frac{1 - P_A(z)}{1 - P_B(z)} = \frac{\prod_{i=1}^{p}(1 - p_i z^{-1})}{\prod_{i=1}^{p}(1 - \alpha p_i z^{-1})},$$

Methods for quantizing the residual

After removing the pitch periodicity from the residual, we need a way to quantize what's left.

- Adaptive center-clipping: use bits to code the high-amplitude samples.
- Multi-pulse LPC: build up $v[n]$ one impulse at a time.
- Tree-coding and CELP: Just find the excitation that gives the best speech, who cares whether or not it's related to the true LPC residual.

Outline

The Forward Algorithm

Definition: $\alpha_t(i) \equiv p(\vec{o}_1, \ldots, \vec{o}_t, q_t = i|\lambda)$. Computation:

1. **Initialize:**

$$\alpha_1(i) = \pi_i b_i(\vec{o}_1), \quad 1 \le i \le N$$

2. **Iterate:**

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(\vec{o}_t), \quad 1 \le j \le N, \; 2 \le t \le T$$

3. **Terminate:**

$$p(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The Baum-Welch Algorithm: Derivation

The Baum-Welch algorithm maximizes $P(O|\lambda)$ iteratively. Each iteration finds $\bar{\lambda}$ that maximizes

$$Q(\lambda,\bar{\lambda}) = \sum_Q P(Q|O,\lambda) \ln P(O,Q|\bar{\lambda})$$

There are some constraints that need to be satisfied. If these constraints are of the form $C_i(\lambda)=0$, then we can construct a Lagrangian of the form

$$\mathcal{L}(\bar{\lambda}) = Q(\lambda,\bar{\lambda}) - \sum_i \kappa_i C_i(\bar{\lambda}),$$

Find the value of $\bar{\lambda}$ that maximizes that, in terms of the Lagrange multipliers κ_i. Then set κ_i to the values that cause $C_i(\bar{\lambda})=0$.

The Baum-Welch Algorithm: Result

1. **Transition Probabilities:**

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}$$

2. **Gaussian Observation PDFs:**

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) \vec{o}_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

$$\bar{\Sigma}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) (\vec{o}_t - \vec{\mu}_i)(\vec{o}_t - \vec{\mu}_i)^T}{\sum_{t=1}^{T} \gamma_t(i)}$$

The Scaled Forward Algorithm

1. **Initialize:**

$$\hat{\alpha}_1(i) = \frac{1}{c_1}\pi_i b_i(\vec{o}_1)$$

2. **Iterate:**

$$\tilde{\alpha}_t(j) = \sum_{i=1}^{N} \hat{\alpha}_{t-1}(i) a_{ij} b_j(\vec{o}_t)$$

$$c_t = \sum_{j=1}^{N} \tilde{\alpha}_t(j)$$

$$\hat{\alpha}_t(j) = \frac{1}{c_t}\tilde{\alpha}_t(j)$$

3. **Terminate:**

$$\ln p(O|\lambda) = \sum_{t=1}^{T} \ln c_t$$