Statistical model training

DTW, EM, and HMM training

- DTW: no training per se
  - each example = its own model
  - does deal with sequences
- EM estimates parameters for hidden variables
  - iteratively weights with posterior estimates
  - as described so far, no sequences
- HMM training uses EM to estimate parameters
  - iteratively weights with posterior estimates
  - applies to full sequences

HMM recognition->training

- Conditional independence assumptions
  - made inference feasible
  - led to full likelihood, Viterbi estimates
- Assumption: separate acoustic/language models
  - permitted Bayes rule combination
  - need to estimate associated parameters
- EM needed for sequences
  - goal is to maximize likelihood for entire sequence
  - optimize over all possible state sequences
  - don't know where speech classes start/stop

HMM training(1)

- Start with EM auxiliary function
  - states are the hidden variables
  - maximizing Aux also maximizes likelihood

$$Aux = \sum_{Q} P(Q \mid X_1^N, \Theta_{old}) \log[P(X_1^N, Q \mid \Theta)]$$

$$= \sum_{Q} P(Q \mid X_1^N, \Theta_{old}) \log[P(X_1^N \mid Q, \Theta) P(Q \mid \Theta)]$$

- Aux = E(log joint prob of observed, hidden)
  - observed = sequence of feature vectors
  - hidden=sequence of states
  - maximize for each model M by adjusting θ
  - iterate

HMM training(2)

- Use conditional independence assumptions
  - Replace P(data|states) by framewise product of emission probs
  - Replace P(state sequence) by framewise product of transition probs (and first frame prior)

$$Aux = \sum_{n=1}^{N} \sum_{k=1}^{L} P(q_k^n \mid X_1^N, \Theta_{old}) \log P(x_n \mid q_k^n, \Theta)$$

$$+ \sum_{k=1}^{L} P(q_k^1 \mid X_1^N, \Theta_{old}) \log P(q_k^1 \mid \Theta)$$

$$+ \sum_{n=2}^{N} \sum_{k=1}^{L} \sum_{l=1}^{L} P(q_l^n, q_k^{n-1} \mid X_1^N, \Theta_{old}) \log P(q_l^n \mid q_k^{n-1}, \Theta)$$

HMM training(3)

- Optimize terms separately (separate parameters)
  - First term: take partial derivative, set to zero,

  solve equations, get local maximum
  - Other terms: need to use Lagrangian constraint
    - State priors sum to 1 for all possible classes
    - State transition probs sum to 1 for all possible transitions
    - For mixture Gaussian case, all weights sum to 1
    - In all cases, take partial derivatives including the constraint term, set to zero, solve

HMM training(4)- summary

(1) Choose form for local prob estimators for state emission densities (e.g., Gaussian)

(2) Choose initialization for parameters

(3) Given the parameters, compute $P(q_j^n \mid X_1^N, \Theta_{old})$ for each state and time, and $P(q_j^n, q_i^{n-1} \mid X_1^N, \Theta_{old})$ for each state transition and time

(4) Given these probabilities, re-estimate parameters to maximize Aux

(5) Assess and return to (3) if not good enough

But wait, there's more

- Each parameter estimator needs posterior estimate (e.g., prob of a state at a particular time given the feature vector sequence)
- This requires recursion to estimate these values
- This recursion is called the forward-backward method, or Baum-Welch training

Forward and backward recursions

- Forward recursion was defined before:

$$\alpha_n(l \mid M) = P(X_1^n, q_l^n \mid M) = \sum_{k=1}^{L} \alpha_{n-1}(k \mid M) P(q_l^n \mid q_k^{n-1}) P(x_n \mid q_l^n)$$

- Backward recursion defined so that product is joint probability of observed sequence and a particular state at time n:

$$\beta_n(l \mid M) = P(X_{n+1}^N \mid q_l^n, X_1^n, M) = \sum_{k=1}^{L} \beta_{n-1}(k \mid M) P(q_k^{n+1} \mid q_l^n) P(x_{n+1} \mid q_k^{n+1})$$

State probability at time n

$$P(q_k^n \mid X_1^N, M) = \frac{P(X_1^N, q_k^n \mid M)}{P(X_1^N \mid M)} = \frac{P(X_1^N, q_k^n \mid M)}{\sum_l P(X_1^N, q_l^n \mid M)}$$

$$= \frac{\alpha_n(k \mid M)\beta_n(k \mid M)}{\sum_l \alpha_n(l \mid M)\beta_n(l \mid M)}$$

- This can be used to update parameter values for emission densities (e.g., means and variances)
- The new density estimators can then be used to do new forward and backward recurrences
- Etc., etc.

Transition probabilities at time n

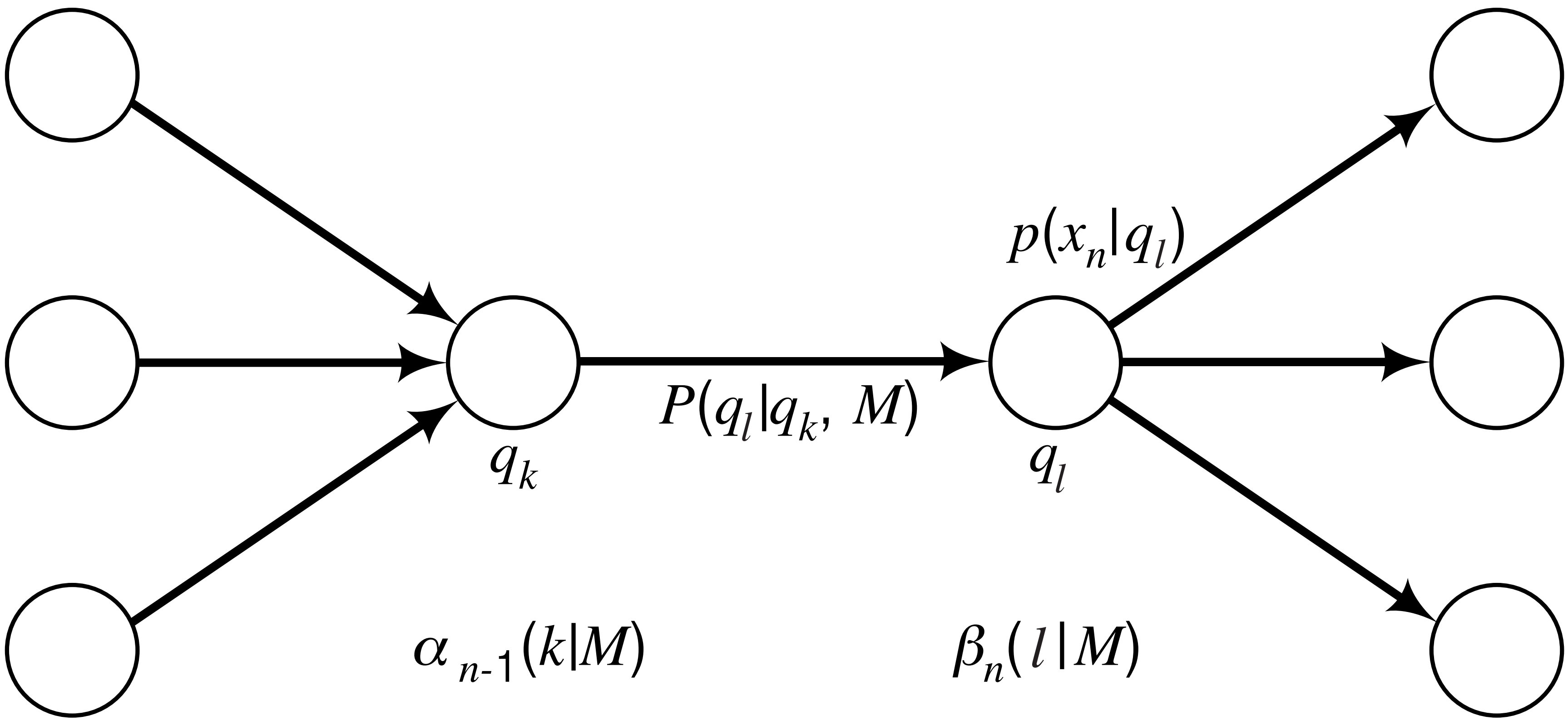
$$P(q_l^n \mid q_k^{n-1}, M) = \frac{P(q_l^n, q_k^{n-1} \mid M)}{P(q_k^{n-1} \mid M)} = \frac{P(q_l^n, q_k^{n-1} \mid M)}{\sum_l P(q_l^n, q_k^{n-1} \mid M)}$$

$$= \frac{\sum_{n=2}^{N} \beta_n(l \mid M) P(x_n \mid q_l^n) P(q_l^n \mid q_k^{n-1}) \alpha_{n-1}(k \mid M)}{\sum_{l=1}^{L(M)} \sum_{n=2}^{N} \beta_n(l \mid M) P(x_n \mid q_l^n) P(q_l^n \mid q_k^{n-1}) \alpha_{n-1}(k \mid M)}$$

Gets estimate of total probability for all paths that contain this transition

- Like emission density estimate, this one can be iterated for improved estimates
- Practical point: for most systems, transition probabilities have little effect

Transition probabilities at time n



Assumptions required for transition probability estimator

- No dependence on previous state for observations in current and later frames
- No dependence on past observations for current state and observation, given previous state
- That being said, the posterior is derived from acoustic probabilities over the entire utterance

Gaussian example

- Best estimator for mean is

$$\mu_j = \frac{\sum_{n=1}^{N} P(q_j^n \mid X_1^N, \Theta_{old}, M) x_n}{\sum_{n=1}^{N} P(q_j^n \mid X_1^N, \Theta_{old}, M)}$$

- Substituting recursion values for posterior

$$= \frac{\sum_{n=1}^{N} \alpha_n(j \mid M) \beta_n(j \mid M) x_n}{\sum_{n=1}^{N} \alpha_n(j \mid M) \beta_n(j \mid M)}$$

Viterbi training

- Previously: full likelihood ASR ≈ best path ASR (Viterbi approximation)
- Prob sum -> max (or min of –log P)
- Can also approximate for training
- Assume state sequence estimate is ground truth for each iteration -> posterior probs are either zero or one
- At training time, choice of model is known (i.e., you know what the word is)

Viterbi training steps

(1) Choose form for local prob estimators for state emission densities (e.g., Gaussian)

(2) Choose initialization for parameters

(3) Find most likely state sequence for each model

(4) Given this sequence, re-estimate parameters

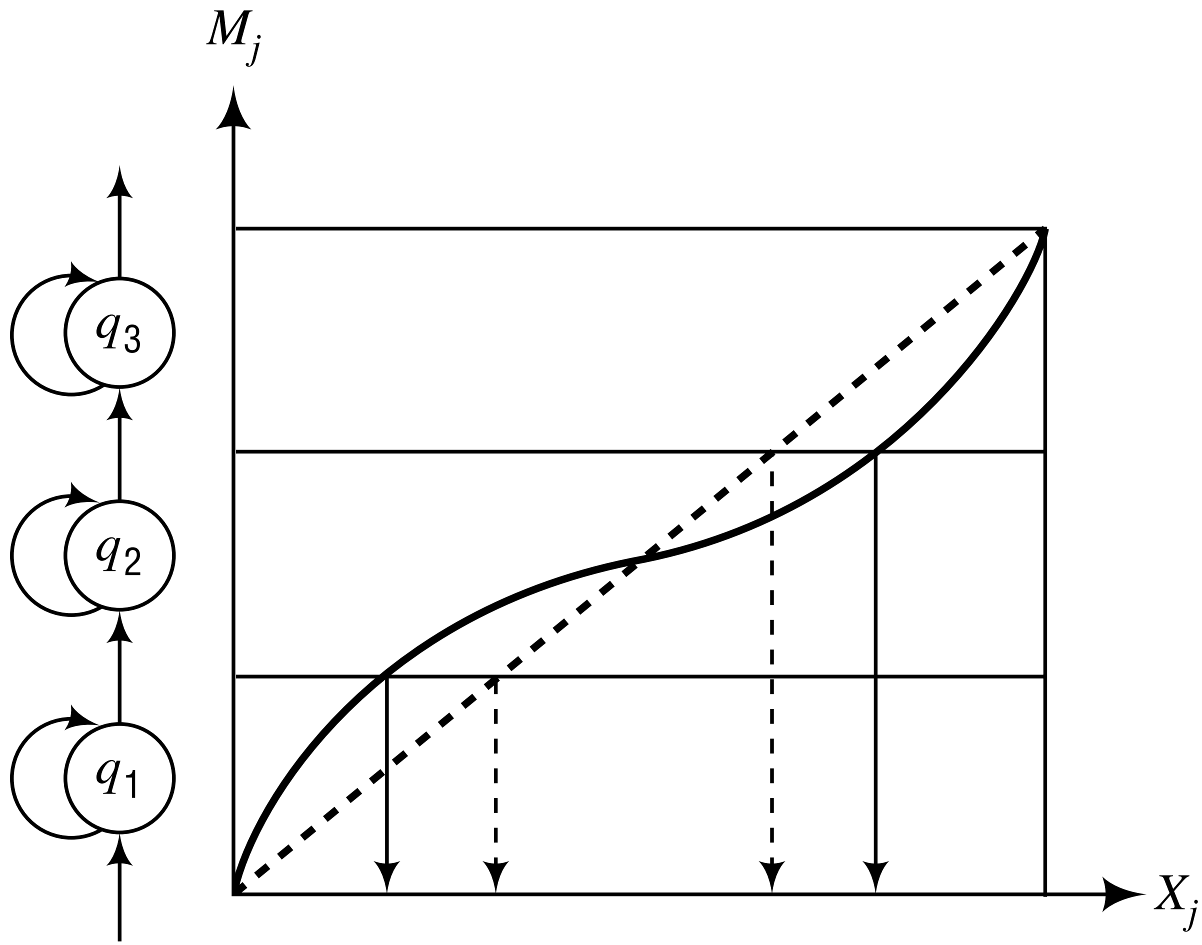
(5) Assess and return to (3) if not good enough

Note: Step (3) is called forced (or Viterbi) alignment.

Viterbi alignment uses DP

- DTW-like local distance is $-\log P(x_n \mid q_l^n)$
- Transition cost is $-\log P(q_l^n \mid q_k^{n-1})$
- Only consider models for transcribed words
- Backtracking straightforward
- Next slide, alignment cartoon

Viterbi (forced) alignment



Viterbi training minus/plus

- Adds another approximation
- Best path might not be the best choice to represent model against other models

But:

- Recognition often done with Viterbi, so it's a good match, since best path gets reinforced
- Transition probabilities particularly simple: just count

Gaussian example

- Means and variances computed from last alignment
- Equivalent to Baum-Welch example with posteriors only being zero or one
- For the mean, get the obvious

$$\mu_j = \frac{\sum_{\#\, frames \text{ labeled } j} x_n}{\#\text{ frames labeled j}}$$

Baum-Welch mean vs Viterbi

$$\mu_j = \frac{\sum_{n=1}^{N} P(q_j^n \mid X_1^N, \Theta_{old}, M)\, x_n}{\sum_{n=1}^{N} P(q_j^n \mid X_1^N, \Theta_{old}, M)}$$

$$\mu_j = \frac{\sum_{\#\ frames\ \text{labeled}\ j} x_n}{\#\ \text{frames labeled j}}$$

Emission probability estimators

- Gaussians
  - Strong assumption; better if full covariance used
- Tied Mixtures of Gaussians
  - Typically better use of parameters
- Independent Mixture of Gaussians
  - More parameters, needs more training data
- Neural Networks – quite different
- Discrete density estimators (using quantization)

Discrete probability estimators

- Vector quantization (VQ) training
  - make a table of feature vectors using clustering
  - commonly called a codebook – sometime >1
- Map each training frame x_n to codebook index y_j
- For both Baum-Welch and Viterbi, generate probability estimates for states given codebook entries

Discrete probability estimators(2)

- Baum-Welch case:

$$P(y_j \mid q_l^n, \Theta) = \frac{\sum_{n=1}^{N} P(q_l^n \mid X_1^N, \Theta_{old}, M)\delta_{nj}}{\sum_{n=1}^{N} P(q_l^n \mid X_1^N, \Theta_{old}, M)}$$

  where posteriors come from forward-backward

- E(#frames for codebook index j and state l) divided by E(#frames for state l)

Discrete probability estimators(3)

- Viterbi case:

$$P(y_j \mid q_l, \Theta) = \frac{\#\, frames \text{ labeled } l \text{ and } j}{\#\, frames \text{ labeled } l}$$

where counts come from the previous alignment

Initialization

- Needed for any form of EM
- Can start with manually annotated database
  - TIMIT
  - STP or Buckeye
- Can start with estimator probabilities from a previous task
- For Baum-Welch, can even use very simple segmentations

Smoothing

- To capture variability, want detailed models
- Insufficient data for some fine categories
- Smoothing is required
- Typically combine fine and coarse estimates
- Used for both acoustic and language models
- Common methods: backoff and interpolation

Backoff Smoothing

- Set threshold for number of training examples in a category to use for estimate
- If fewer examples, use a coarser category
- Example: triphone
  - Phone in context of a left and right phone
  - If not enough examples, use biphone (e.g., average of the left biphone value and right one)
- Simple, but often works well
- The subtlety is in picking thresholds

Smoothing by Interpolation

- Linearly interpolate between fine and coarse
- One approach: deleted interpolation
  - Learn weights from disjoint data
  - Can also jackknife through the data
  - Can set fine class weight to fraction of utterances for which fine class is better
  - Can also use EM to estimate the weights

A caution about probabilities

- I' ve treated each incidence of P() as a prob
- Often it' s really a density
- Density values often > 1
- Integrate to 1 over all possible values, not over all observed values

Summary

- Training of HMMs briefly covered
- Chapter 26 has a few things worked through in greater detail – try to follow the equations
- Papers from ICASSP, Interspeech (the combined ICSLP and Eurospeech) have more
- We had many assumptions
  - known to be wrong – long distance independence
  - If models are wrong, ML not the best
  - Increased importance of discriminant training