Maximum likelihood parameter estimation

- For some observed data $O = \langle o_1 \cdots o_n \rangle$, and a model, here a bigram model, the data likelihood for a particular set of parameters $\Theta = \{P(o^k|o^j)\}$ is:

$$L(O|\Theta) = \prod_{i=1}^{n} P(o_i|o_{i-1}) = \prod_{j=1}^{V} \prod_{k=1}^{V} P(o^k|o^j)^{\#(o^j o^k)}$$

- People often use the log because its easier to manipulate, and the log is monotonic with the likelihood:

$$LL(O|\Theta) = \sum_{i=1}^{n} \log P(o_i|o_{i-1}) = \sum_{j=1}^{V} \sum_{k=1}^{V} \#(o^j o^k) \log P(o^k|o^j)$$

- We can work out how to maximize this likelihood using calculus (assignment)

Maximum likelihood parameter estimation

- For an HMM with observed state data X, and s states, we do the same:

$$\begin{aligned} L(O, X|\Theta) &= \prod_{i=1}^{n} P(x_i|x_{i-1}) P(o_i|x_i) \\ &= \prod_{j=1}^{s} \prod_{k=1}^{s} P(x^k|x^j)^{\#(x^j x^k)} \prod_{k=1}^{s} \prod_{m=1}^{V} P(o^m|x^k)^{\#(x^k \to o^m)} \\ &= a_{x_0 x_1} a_{x_1 x_2} a_{x_2 x_3} \cdots a_{x_{n-1} x_n} b_{x_1 o_1} b_{x_2 o_2} \cdots b_{x_n o_n} \end{aligned}$$

- We can maximize this likelihood by setting the parameters in Θ, and get the same form of relative frequency estimates
- But if our state sequence is *unobserved* we can't do that directly

HMM maximum likelihood parameter estimation

- However, we can work out the likelihood of being in different states at different times, given the current model and the observed data:

$$P(X_t = x^k|O, \Theta) = \frac{\alpha_k(t)\beta_k(t)}{\sum_{j=1}^{s} \alpha_j(t)\beta_j(t)}$$

- Given, these probabilities, something we could do is *sample* from this distribution and generate pseudo-data which is complete.
- From this data $\langle O, \hat{X} \rangle$, we could do ML estimation as before – since it is complete data
- And with sufficient training data, this would work fine.

HMM maximum likelihood parameter estimation

- For the EM algorithm, we do something just slightly subtler. We work out the *expected* number of times we made each state transition and emitted each symbol from each state. This is conceptually just like an observed count, but it'll usually be a non-integer
- We then work out new parameter estimates as relative frequencies just like before.

Parameter reestimation formulae

$$\begin{aligned} \hat{\pi}_i &= \text{expected frequency in state } i \text{ at time } t = 1 \\ &= \gamma_i(1) \\ \hat{a}_{ij} &= \frac{\text{expected num. transitions from state } i \text{ to } j}{\text{expected num. transitions from state } i} \\ &= \frac{\sum_{t=1}^{T} p_t(i, j)}{\sum_{t=1}^{T} \gamma_i(t)} \\ \hat{b}_{ik} &= \frac{\text{expected num. times } k \text{ observed in state } i}{\text{expected num. transitions from } i} \\ &= \frac{\sum_{\{t: o_t = k, 1 \le t \le T\}} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)} \end{aligned}$$

EM Algorithm

- Changing the parameters in this way *must* have increased (or at any rate not decreased) the likelihood of this completion of the data: we're setting the parameters on the pseudo-observed data to maximize the likelihood of this pseudo-observed data
- But, then, we use these parameter estimates to compute new expectations (or, to sample new complete data)
- Since this new data completion is directly based on the current parameter settings, it is at least intuitively reasonable to think that the model should assign it higher likelihood than the old completion (which was based on different parameter settings)

We're guaranteed to get no worse

- Repeating these two steps iteratively gives us the EM algorithm
- One can prove rigorously that iterating it changes the parameters in such a way that the data likelihood is non-decreasing (**??**)
- But we can get stuck in local maxima or on saddle points, though
  - For a lot of NLP problems with a lot of hidden structure, this is actually a *big* problem

Information extraction evaluation

- Example text for IE:

  Australian Tom Moody took six for 82 but Chris Adams , 123 , and Tim O'Gorman , 109 , took Derbyshire to 471 and a first innings lead of 233 .
- Boxes shows attempt to extract person names (correct ones in purple)
- What score should this attempt get?
- A stringent criterion is exact match precision/recall/F_1

Precision and recall

- Precision is defined as a measure of the proportion of selected items that the system got right:
$$\text{precision} = \frac{tp}{tp + fp}$$
- Recall is defined as the proportion of the target items that the system selected:
$$\text{recall} = \frac{tp}{tp + fn}$$

These two measures allow us to distinguish between excluding target items and returning irrelevant items.

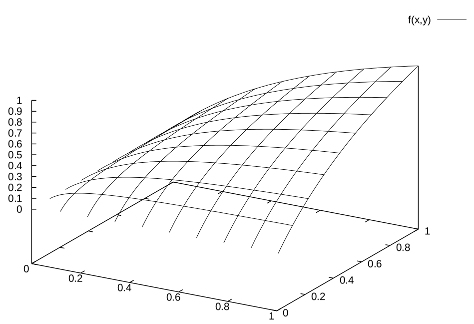
They still require human-made "gold standard" judgements.

Combining them: The F measure

Weighted harmonic mean: The F measure (where $F = 1 - E$):
$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$
where P is precision, R is recall and α weights precision and recall. (Or in terms of β, where $\alpha = 1/(\beta^2 + 1)$.)

A value of $\alpha = 0.5$ is often chosen.
$$F = \frac{2PR}{R + P}$$
At break-even point, when $R = P$, then $F = R = P$

The F measure ($\alpha = 0.5$)



Ways of averaging

Precision	Recall	Arithmetic	Geometric	Harmonic	Minimum
80	10	45	28.3	17.8	10
80	20	50	40.0	32.0	20
80	30	55	49.0	43.6	30
80	40	60	56.6	53.3	40
80	50	65	63.2	61.5	50
80	60	70	69.3	68.6	60
80	70	75	74.8	74.7	70
80	80	80	80.0	80.0	80
80	90	85	84.9	84.7	80
80	100	90	89.4	88.9	80

Other uses of HMMs: Information Extraction (Freitag and McCallum 1999)

- IE: extracting instance of a relation from text snippets
- States correspond to fields one wishes to extract, token sequences in the context that are good for identifying the fields to be extracted, and a background "noise" state
- Estimation is from tagged data (perhaps supplemented by EM reestimation over a bigger training set)
- The Viterbi algorithm is used to tag new text
- Things tagged as fields to be extracted are returned

Information Extraction (Freitag and McCallum 1999)

- State topology is set by hand. Not fully connected
- Use simpler and more complex models, but generally:
  - Background state
  - Preceding context state(s)
  - Target state(s)
  - Following context state(s)
- Preceding context states connect only to target state, etc.

Information Extraction (Freitag and McCallum 1999)

- Each HMM is for only one field type (e.g., "speaker")
- Use different HMMs for each field (*bad:* no real notion of multi-slot structure)
- Semi-supervised training: target words (generated only by target states) are marked
- Shrinkage/deleted interpolation is used to generalize parameter estimates to give more robustness in the face of data sparseness
- Some other work has done multi-field extraction over more structured data (Borkar et al. 2001)

Information Extraction (Freitag and McCallum 1999)

- Tested on seminar announcements and corporate acquisitions data sets
- Performance is generally equal to or better than that of other information extraction methods
- Though probably more suited to semi-structured text with clear semantic sorts, than strongly NLP-oriented problems
- HMMs tend to be especially good for robustness and high recall

Information extraction

- Getting particular fixed semantic relations out of text (e.g., buyer, sell, goods) for DB filling
- Statistical approaches have been explored recently, particularly use of HMMs (Freitag and McCallum 2000)
- States correspond to elements of fields to extract, token sequences in the context that identify the fields to be extracted, and background "noise" states
- Estimation is from labeled data (perhaps supplemented by EM reestimation over a bigger training set)
- Structure learning used to find a good HMM structure
- The Viterbi algorithm is used to tag new text
- Things tagged as within fields are returned

Information extraction: locations and speakers