

Statistical NLP

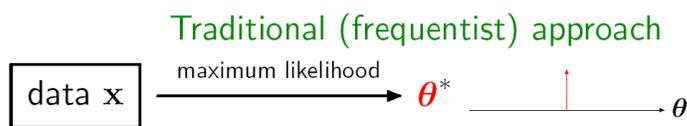
Spring 2008



Lecture 3: Language Models II

Dan Klein – UC Berkeley

Parameter Estimation



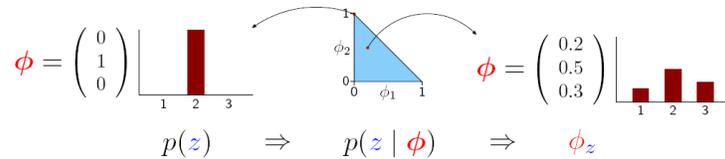
An example:

coin flips (data): $x = (\text{H}) (\text{T}) (\text{H}) (\text{H})$
probability of (H) (parameter): $\theta = \frac{3}{4}$? $\frac{4}{6}$?

Multinomial Distributions

Most parameters in NLP are distributions $p(z)$ over discrete objects $z \in \{1, \dots, K\}$.

Possible $p(z)$ s are the points ϕ on the simplex. For $K = 3$:



A random draw z from a multinomial distribution is written:

$$z \sim \text{Multinomial}(\phi).$$

If we draw some number of independent samples from $\text{Multinomial}(\phi)$, the probability of observing c_z counts of observation z is proportional to:

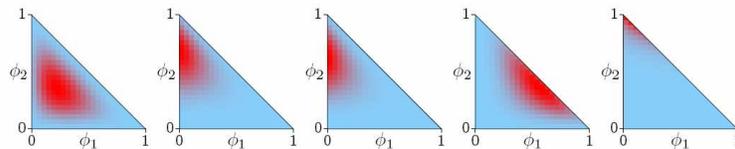
$$\phi_1^{c_1} \dots \phi_K^{c_K}$$

Dirichlet Distributions

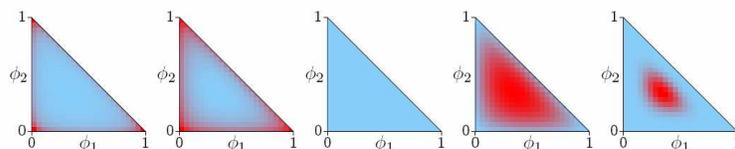
A **Dirichlet distribution** is a distribution over multinomial parameters ϕ in the simplex.

Like a Gaussian, there's a notion of mean and variance.

Different means:



Different variances:



Dirichlet Distributions

A Dirichlet is specified by **concentration parameters**:

$$\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K), \alpha_z \geq 0$$

$$\text{Mean: } \left(\frac{\alpha_1}{\sum_z \alpha_z}, \dots, \frac{\alpha_n}{\sum_z \alpha_z} \right)$$

Variance: larger $\boldsymbol{\alpha}$ s \rightarrow smaller variance

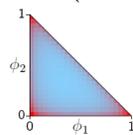
A Dirichlet draw $\boldsymbol{\phi}$ is written $\boldsymbol{\phi} \sim \text{Dirichlet}(\boldsymbol{\alpha})$,

which means $p(\boldsymbol{\phi} | \boldsymbol{\alpha}) \propto \phi_1^{\alpha_1-1} \dots \phi_K^{\alpha_K-1}$

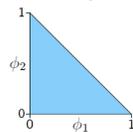
$$\text{Mode: } \left(\frac{\alpha_1-1}{\sum_z (\alpha_z-1)}, \dots, \frac{\alpha_n-1}{\sum_z (\alpha_z-1)} \right)$$

Dirichlet Distributions

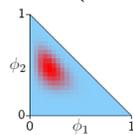
Dirichlet(.5,.5,.5)



Dirichlet(1,1,1)



Dirichlet(5,10,8)



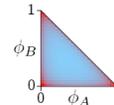
Posterior Updating

Model:

$$\phi \sim \text{Dirichlet}_3(\underbrace{0.5, 0.5, 0.5}_{\alpha})$$

$$\mathbf{x} \sim \text{Multinomial}_3^7(\phi)$$

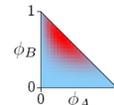
Prior: $p(\phi) \propto \phi_A^{0.5-1} \phi_B^{0.5-1} \phi_C^{0.5-1}$



Likelihood: $p(\mathbf{x} | \phi) = \phi_A^2 \phi_B^4 \phi_C^1$



Posterior: $p(\phi | \mathbf{x}) \propto \phi_A^{2.5-1} \phi_B^{4.5-1} \phi_C^{1.5-1}$



Result: $p(\phi | \mathbf{x}) = \text{Dirichlet}(2.5, 4.5, 1.5)$

Parameter Estimates

- Given observations $w_1 \dots w_N$
 - Maximum likelihood

$$\hat{P}(w) = P_{\text{ML}}(w) = \frac{c(w)}{\sum_{w'} c(w')}$$

- With prior $\text{Dir}(k\theta)$, mean posterior (also posterior predictive dist) is:

$$P_{\text{dir}}(w) = \frac{c(w) + k\theta_w}{\sum_{w'} [c(w') + k\theta_w]}$$

- Note again how prior shows up as *pseudocounts*
- BUT, most estimators are not so simply described
- Would be nice to maintain uncertainty over parameters (but it's usually inconvenient)

Parameter Estimation

- Given data, we want to produce estimates of conditional probabilities
- Maximum likelihood estimates won't get us very far for bigrams and up

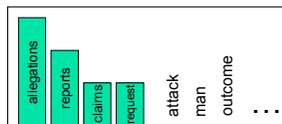
$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

- Need to *smooth* these estimates
- General method (procedurally)
 - Take your empirical counts
 - Modify them in various ways to improve estimates
- General method (mathematically)
 - Often can give estimators a formal statistical interpretation
 - ... but not always
 - Stuff that works not always the same as stuff we can explain (yet!)

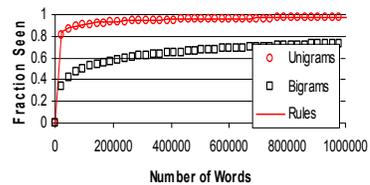
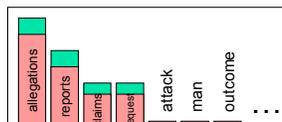
Smoothing

- Dealing with sparsity well: smoothing / shrinkage
 - For most histories $P(w | h)$, relatively few observations
 - Very intricately explored for the speech n-gram case
 - Easy to do badly

$P(w | \text{denied the})$
 3 allegations
 2 reports
 1 claims
 1 request
 7 total



$P(w | \text{denied the})$
 2.5 allegations
 1.5 reports
 0.5 claims
 0.5 request
 2 other
 7 total



Priors on Parameters

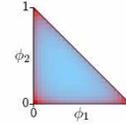
- Most obvious formal solution: use MAP estimate instead of ML estimate for a multinomial $P(X)$
- Maximum likelihood estimate: $\max P(X|\theta)$

$$\theta_{ML} = \frac{c(x)}{\sum_{x'} c(x')}$$

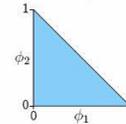
- MAP estimate: $\max P(\theta|X)$
 - Dirichlet priors are a convenient choice
 - Specified by a center θ' and strength k , $\text{Dir}(k\theta')$
 - Mean is center, higher strength means lower variance
 - MAP estimate is then (though usually use the mean)

$$P_{\text{DIR}}(w) = \frac{c(w) + k\theta_w}{\sum_{w'} [c(w') + k\theta_w]}$$

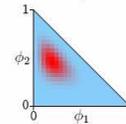
Dirichlet(.5,.5,.5)



Dirichlet(1,1,1)



Dirichlet(5,10,8)



Smoothing: Add-One, Etc.

- With a uniform (aka symmetric) prior, get estimates of the form

$$P_{\text{add-}\delta}(w|w_{-1}) = \frac{c(w_{-1}, w) + \delta}{\sum_{w'} [c(w_{-1}, w') + \delta]}$$

- Add-one smoothing especially often talked about
- Or, for each bigram distribution, can use a prior *centered on the empirical unigram*:

$$P_{\text{dir}}(w|w_{-1}) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{(\sum_{w'} c(w_{-1}, w')) + k}$$

- Can consider hierarchical formulations in which trigram is centered on smoothed bigram estimate, etc [MacKay and Peto, 94]
- Problem: works quite poorly! (Why?)

Linear Interpolation

- Problem: $\hat{P}(w|w_{-1}, w_{-2})$ is supported by few counts
- Classic solution: mixtures of related, denser histories, e.g.:

$$\lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w)$$

- The mixture approach tends to work better than the Dirichlet prior approach for several reasons
 - Can flexibly include multiple back-off contexts, not just a chain
 - Good ways of learning the mixture weights with EM (later)
 - Not entirely clear why it works so much better
- All the details you could ever want: [Chen and Goodman, 98]

Held-Out Data

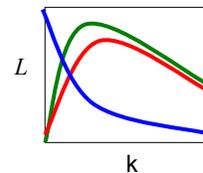
- Important tool for getting models to generalize:



- Set a small number of hyperparameters that control the degree of smoothing by maximizing the (log-)likelihood of held-out data
 - Can use any optimization technique (line search or EM usually easiest)

- Examples:

$$P_{dir}(w|w_{-1}, k) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{(\sum_{w'} c(w_{-1}, w')) + k}$$



$$P_{lin}(w|w_{-1}, \lambda, \lambda', \lambda'') = \lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w)$$

Held-Out Reweighting

- What's wrong with unigram-prior smoothing?
- Let's look at some real bigram counts [Church and Gale 91]:

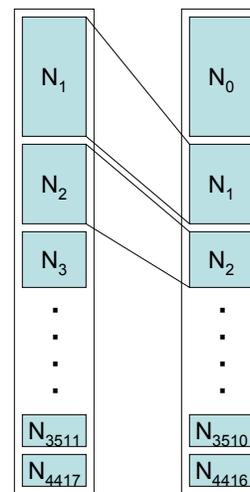
| Count in 22M Words | Actual c^* (Next 22M) | Add-one's c^* | Add-0.0000027's c^* |
|--------------------|-------------------------|-----------------|-----------------------|
| 1 | 0.448 | $2/7e-10$ | ~ 1 |
| 2 | 1.25 | $3/7e-10$ | ~ 2 |
| 3 | 2.24 | $4/7e-10$ | ~ 3 |
| 4 | 3.23 | $5/7e-10$ | ~ 4 |
| 5 | 4.21 | $6/7e-10$ | ~ 5 |

| | | | |
|--------------|------|--------------|----------|
| Mass on New | 9.2% | $\sim 100\%$ | 9.2% |
| Ratio of 2/1 | 2.8 | 1.5 | ~ 2 |

- Big things to notice:
 - Add-one vastly overestimates the fraction of new bigrams
 - Add-0.0000027 still underestimates the ratio $2^*/1^*$
- One solution: use held-out data to predict the map of c to c^*

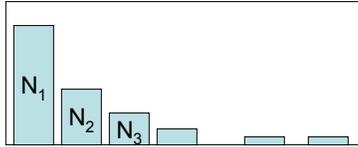
Good-Turing Reweighting I

- We'd like to not need held-out data (why?)
- Idea: leave-one-out validation
 - N_k : number types which occur k times in the entire corpus
 - Take each of the c tokens out of corpus in turn
 - c "training" sets of size $c-1$, "held-out" of size 1
 - What fraction of held-out tokens were unseen in training?
 - N_1/c
 - What fraction of held-out words were seen k times in training?
 - $(k+1)N_{k+1}/c$
 - Words with training count k seem to take up $(k+1)N_{k+1}/c$ fraction of future text
 - There are N_k words with training count k
 - Each should occur with probability:
 - $(k+1)N_{k+1}/(cN_k)$
 - ...or expected count $(k+1)N_{k+1}/N_k$

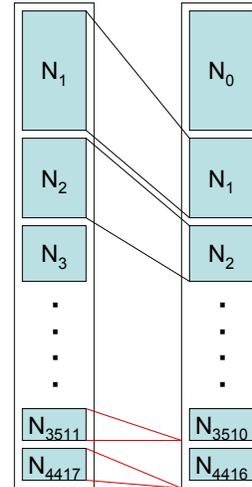
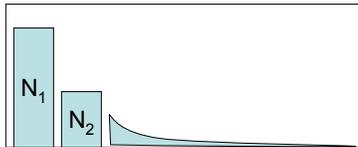


Good-Turing Reweighting II

- Problem: what about “the”? (say $c=4417$)
 - For small k , $N_k > N_{k+1}$
 - For large k , too jumpy, zeros wreck estimates



- Simple Good-Turing [Gale and Sampson]: replace empirical N_k with a best-fit power law once count counts get unreliable



Good-Turing Reweighting III

- Hypothesis: counts of k should be $k^* = (k+1)N_{k+1}/N_k$

| Count in 22M Words | Actual c^* (Next 22M) | GT's c^* |
|--------------------|-------------------------|------------|
| 1 | 0.448 | 0.446 |
| 2 | 1.25 | 1.26 |
| 3 | 2.24 | 2.24 |
| 4 | 3.23 | 3.24 |
| Mass on New | 9.2% | 9.2% |

- Katz Smoothing
 - Use GT discounted *bigram* counts (roughly – Katz left large counts alone)
 - Whatever mass is left goes to empirical unigram

$$P_{\text{katz}}(w|w') = \frac{c^*(w, w')}{c(w')} + \alpha \hat{P}(w)$$

Kneser-Ney: Discounting

- Kneser-Ney smoothing: very successful but slightly ad hoc estimator
- Idea: observed n-grams occur more in training than they will later:

| Count in 22M Words | Avg in Next 22M | Good-Turing c^* |
|--------------------|-----------------|-------------------|
| 1 | 0.448 | 0.446 |
| 2 | 1.25 | 1.26 |
| 3 | 2.24 | 2.24 |
| 4 | 3.23 | 3.24 |

- Absolute Discounting
 - Save ourselves some time and just subtract 0.75 (or some d)
 - Maybe have a separate value of d for very low counts

$$P_{\text{ad}}(w|w') = \frac{c(w, w') - d}{c(w')} + \alpha \hat{P}(w)$$

Kneser-Ney: Continuation

- Something's been very broken all this time
 - Shannon game: There was an unexpected ____?
 - delay?
 - Francisco?
 - "Francisco" is more common than "delay"
 - ... but "Francisco" always follows "San"
- Solution: Kneser-Ney smoothing
 - In the back-off model, we don't want the probability of w as a unigram
 - Instead, want the probability that w is *allowed in this novel context*
 - For each word, count the number of bigram types it completes

$$P'(w) \propto |w' : c(w, w') > 0|$$

Kneser-Ney

- Kneser-Ney smoothing combines these two ideas
 - Absolute discounting

$$P(w|w') = \frac{c(w, w') - d}{c(w')} + \alpha P'(w)$$

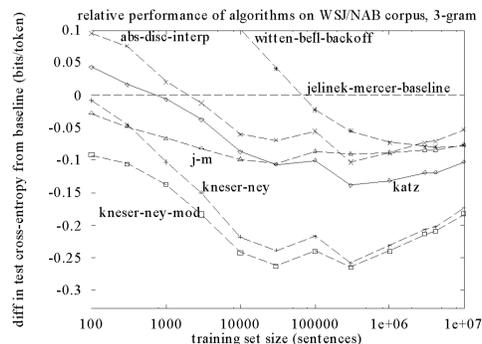
- Lower order models take a special form

$$P'(w) \propto |w' : c(w, w') > 0|$$

- KN smoothing repeatedly proven effective
 - But we've never been quite sure why
 - And therefore never known how to make it better
- [Teh, 2006] shows KN smoothing is a kind of approximate inference in a hierarchical Pitman-Yor process (and better approximations are superior to basic KN)

What Actually Works?

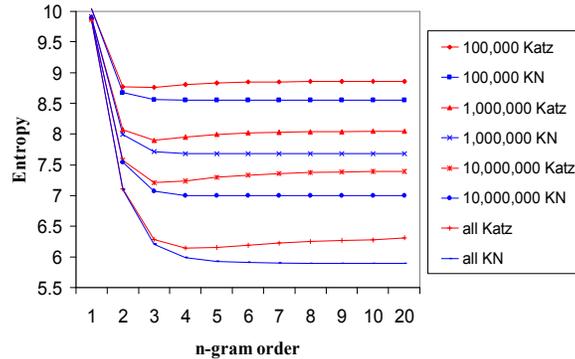
- Trigrams:
 - Unigrams, bigrams too little context
 - Trigrams much better (when there's enough data)
 - 4-, 5-grams often not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- Good-Turing-like methods for count adjustment
 - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell
- Kneser-Ney equalization for lower-order models
- See [Chen+Goodman] reading for tons of graphs!



[Graphs from Joshua Goodman]

Data >> Method?

- Having more data is better...



- ... but so is using a better model
- Another issue: $N > 3$ has huge costs in speech recognizers (though using $N \gg 3$ is helpful for MT)

Beyond N-Gram LMs

- Lots of ideas we won't have time to discuss:
 - Caching models: recent words more likely to appear again
 - Trigger models: recent words trigger other words
 - Topic models
- A few recent ideas
 - Syntactic models: use tree models to capture long-distance syntactic effects [Chelba and Jelinek, 98]
 - Discriminative models: set n-gram weights to improve final task accuracy rather than fit training set density [Roark, 05, for ASR; Liang et. al., 06, for MT]
 - Structural zeros: some n-grams are syntactically forbidden, keep estimates at zero [Mohri and Roark, 06]
 - Bayesian document and IR models [Daume 06]