EM Algorithm

CS 690N, Spring 2018

Advanced Natural Language Processing

http://people.cs.umass.edu/~brenocon/anlp2018/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

Announcements

- Feedback #1: due next Tuesday
- HW #1: due in two weeks

Today

- EM algorithm to learn latent variable probabilistic models
- Examples
  - (Unsupervised) Naive Bayes
  - Saul&Pereira's "Aggregate Bigram" Model
- Why does EM work (or not)?

- MLE for N-gram Markov model. How to combat sparsity?
  - Backoff and interpolation:
    combine different Markov orders
  - Smoothing (pseudocounts, discounting):
    observed data counts for less
  - Kneser-Ney smoothing: best n-gram LM
- Latent (hidden) variables
  - Generalizable word attributes
  - Long-distance dependencies
  - (Other forms of linguistic structure...)

Latent-variable generative models

Text

$$P(w, z \mid \theta)$$

Parameters

(Sometimes) latent quantity to help explain it

- Document category
- World context
- Grammatical category
- Semantic structure
- Real-valued embedding
- Why?

Easy stuff you can do

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

More stuff you can do

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

Multinomial Naive Bayes

Easy stuff you can do

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

More stuff you can do

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

Multinomial Naive Bayes

- ...is a class-conditional 0th-order Markov model
  for fixed number of doc categories and word types, with parameters:
  - $\boldsymbol{\phi_k}$ word distribution for each class **k**
  - $\boldsymbol{\mu}$ prior distribution over labels

Easy stuff you can do

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

More stuff you can do

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

Multinomial Naive Bayes

- ...is a class-conditional 0th-order Markov model
  for fixed number of doc categories and word types, with parameters:
  - $\boldsymbol{\phi_k}$ word distribution for each class **k**
  - $\boldsymbol{\mu}$ prior distribution over labels
- Generative story. For each document *d*:
  - P(z): Draw label $\mathbf{z_d} \sim \mathbf{Categ}(\boldsymbol{\mu})$
  - P(w|z): For t=1,2,...: Draw next word $\mathbf{w_{d,t}} \sim \mathbf{Categ}(\boldsymbol{\phi_z})$

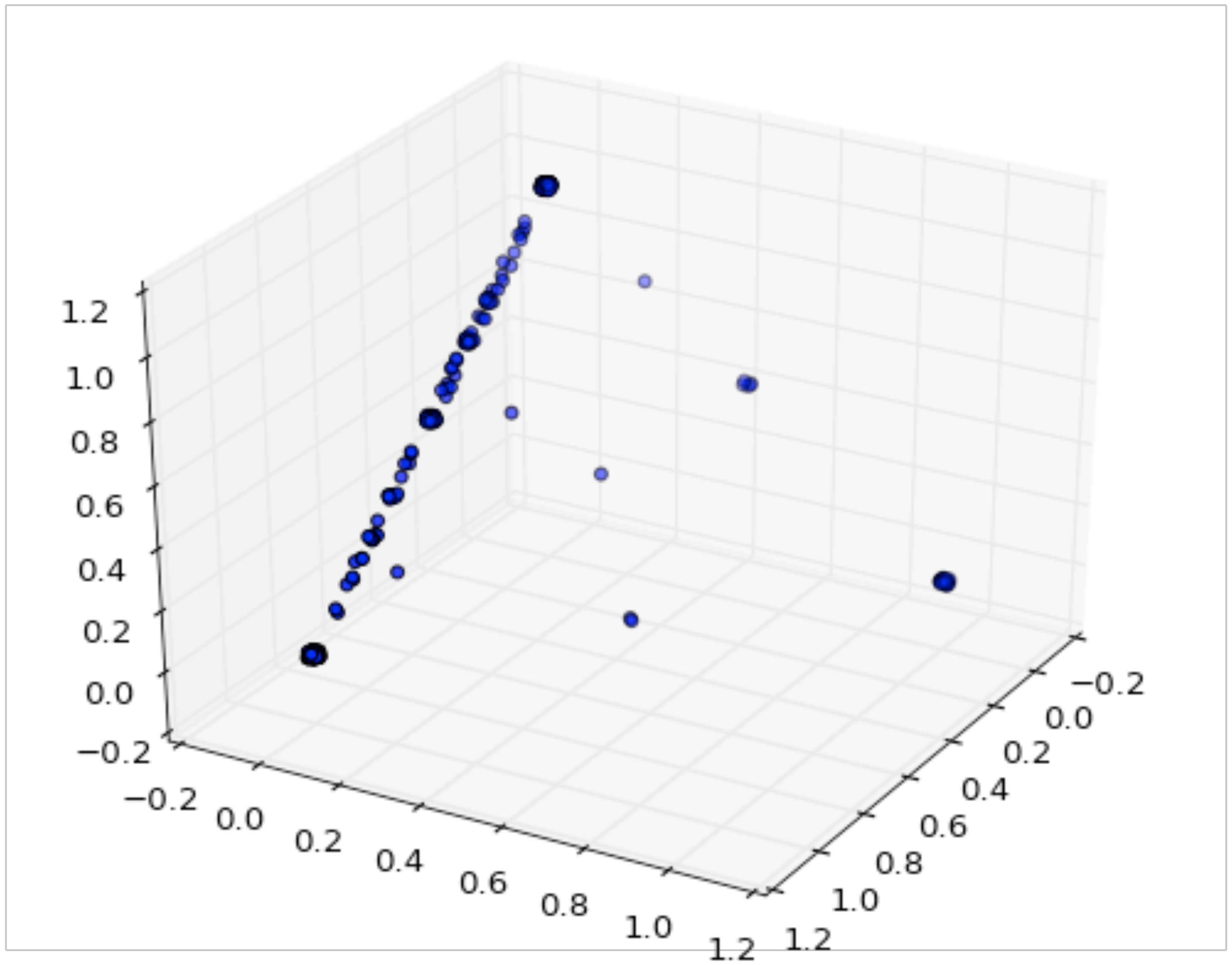
Easy stuff you can do

- Supervised learning: $\mathbf{argmax_\theta\ P(w^{train}, z^{train} \mid \theta)}$
- Prediction (via posterior inference): $\mathbf{P(z \mid w^{input}, \theta)}$

More stuff you can do

- Latent (unsupervised) learning: $\mathbf{argmax_\theta\ P(w^{train} \mid \theta)}$
- Language modeling (via marginal inference): $\mathbf{P(w^{input} \mid \theta)}$

- Supervised classification with MNB:
  - Training: known (w,z), learn params
  - Testing: fix params, known w, want z
- Unsupervised learning (soft clustering)
  - known w, jointly learn z and params
  - Can learn latent structure in data



1987 NYT data
one point per doc
"congress", "religious", "reagan"
probabilities per doc (normalized)

Expectation-Maximization

Expectation-Maximization

- For latent-variable learning (unsupervised or semi-supervised)
    - **w**: known (training data)
    - **z**: unknown "nuisance" variable: need to infer
    - θ: want to learn
    - Learning goal: maximize marginal likelihood
      $\text{argmax}_\theta\ P(w \mid \theta) = \text{argmax}_\theta\ \Sigma_z\ P(w,z \mid \theta)$

Expectation-Maximization

- For latent-variable learning (unsupervised or semi-supervised)
  - **w**: known (training data)
  - **z**: unknown "nuisance" variable: need to infer
  - θ: want to learn
  - Learning goal: maximize marginal likelihood
    $\text{argmax}_\theta P(w \mid \theta) = \text{argmax}_\theta \Sigma_z P(w,z \mid \theta)$
- ... for models where parameter learning would be easy if you had z.
  - Why is this the case for our model?

Expectation-Maximization

- For latent-variable learning (unsupervised or semi-supervised)
  - **w**: known (training data)
  - **z**: unknown "nuisance" variable: need to infer
  - θ: want to learn
  - Learning goal: maximize marginal likelihood
    $\text{argmax}_\theta P(w \mid \theta) = \text{argmax}_\theta \Sigma_z P(w,z \mid \theta)$
- ... for models where parameter learning would be easy if you had z.
  - Why is this the case for our model?
- EM is a "meta"-algorithm
  - Initialize parameters.
  - Iterate until convergence (or stop early):
    - (E step): Infer $Q(z) := P(z \mid w, \theta)$
      - *[Guess based on your model]*
    - (M step): Learn new $\theta := \text{argmax}_\theta E_Q[\log P(w,z \mid \theta)]$
      - *[Learn from what you guessed]*

Expectation-Maximization

- For latent-variable learning (unsupervised or semi-supervised)
  - **w**: known (training data)
  - **z**: unknown "nuisance" variable: need to infer
  - θ: want to learn
  - Learning goal: maximize marginal likelihood
    $\text{argmax}_\theta P(w \mid \theta) = \text{argmax}_\theta \Sigma_z P(w,z \mid \theta)$
- ... for models where parameter learning would be easy if you had z.
  - Why is this the case for our model?
- EM is a "meta"-algorithm
  - Initialize parameters.
  - Iterate until convergence (or stop early):
    - (E step): Infer $Q(z) := P(z \mid w, \theta)$
      - *[Guess based on your model]*
    - (M step): Learn new $\theta := \text{argmax}_\theta E_Q[\log P(w,z \mid \theta)]$
      - *[Learn from what you guessed]*
- Turns out to converge and gives a local maximum solution to the original marginal likelihood learning goal

EM performance

- Guaranteed to find a locally maximum likelihood solution. Guaranteed to converge.
  - But can take a while
- Dependent on initialization

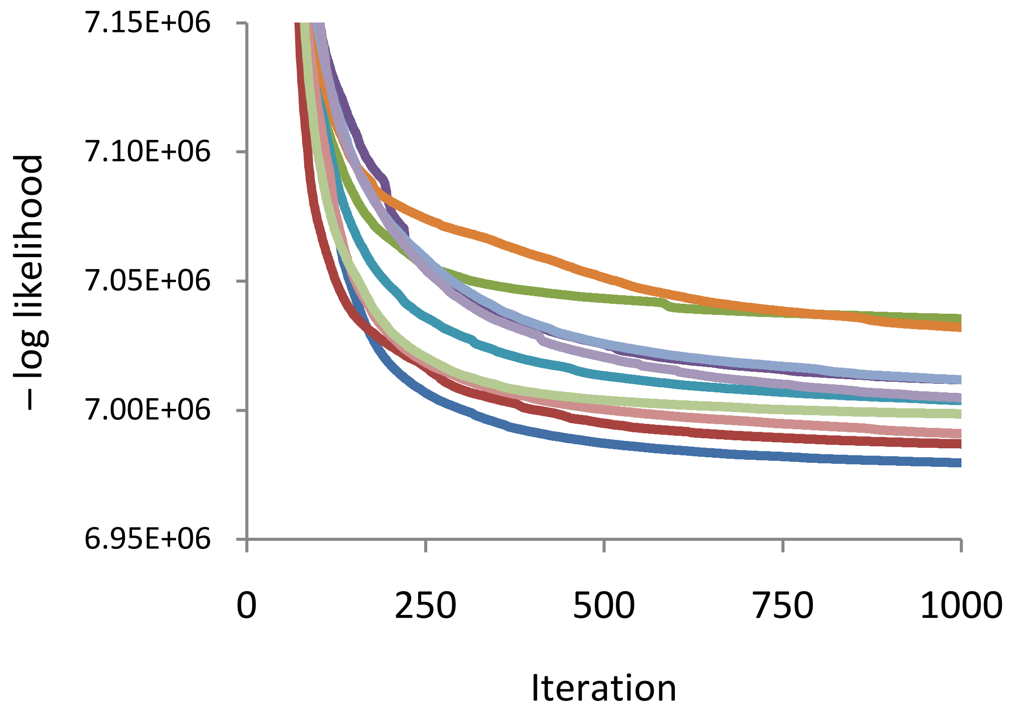


Figure 1: Variation in negative log likelihood with increasing iterations for 10 EM runs from different random starting points.

Johnson 2007, "Why doesn't EM find good HMM POS-taggers?"

Aggregate Bigram Model

Saul and Pereira 1997

- Superficially similar to, but different than, a Hidden Markov Model
- Graphical model / generative story: intermediate state
- Linear algebra: low-rank approximation of standard bigram model (compare: Mnih and Hinton 2007's log-bilinear model)

Assumption 1: Markov

$$p(w_1..w_T) = \prod_t p(w_t \mid w_{t-1})$$

Assumption 2: latent variable

$$p(w_t \mid w_{t-1}) = \sum_{z \in 1..K} \underbrace{p(z \mid w_{t-1})}_{\text{next latent state}} \; \underbrace{p(w_t \mid z)}_{\text{generate word}}$$

next latent state
("transition" probs)

Params to learn:
For every word, prob of which state next?

generate word
("emission" probs)

Params to learn:
For every state, prob of which word to emit?

Train with EM

The EM algorithm for aggregate Markov models is particularly simple. The E-step is to compute, for each bigram $w_1 w_2$ in the training set, the *posterior* probability

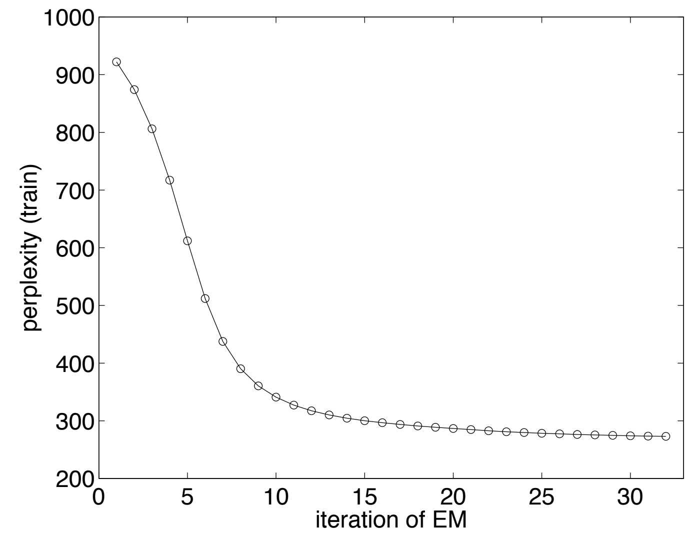
$$P(c|w_1, w_2) = \frac{P(w_2|c)P(c|w_1)}{\sum_{c'} P(w_2|c')P(c'|w_1)}. \qquad (2)$$

Eq. (2) gives the probability that word w_1 was assigned to class c, based on the observation that it was followed by word w_2. The M-step uses these posterior probabilities to re-estimate the model parameters. The updates for aggregate Markov models are:

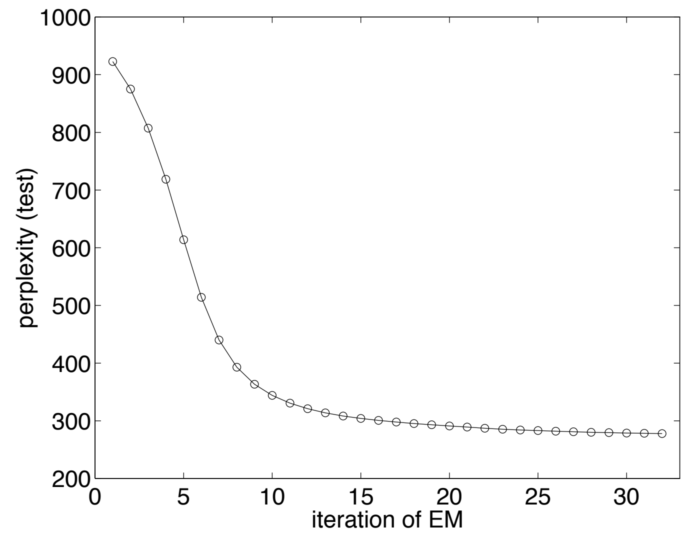
$$P(c|w_1) \leftarrow \frac{\sum_w N(w_1, w)P(c|w_1, w)}{\sum_{wc'} N(w_1, w)P(c'|w_1, w)}, \qquad (3)$$

$$P(w_2|c) \leftarrow \frac{\sum_w N(w, w_2)P(c|w, w_2)}{\sum_{ww'} N(w, w')P(c|w, w')}, \qquad (4)$$

Train with EM



(a)



(b)

Figure 1: Plots of (a) training and (b) test perplexity versus number of iterations of the EM algorithm, for the aggregate Markov model with $C = 32$ classes.

- Why evaluate on test data?
- Hyperparameters and under/overfitting for different models

Learned model

1	as cents made make take
2	ago day earlier Friday Monday month quarter reported said Thursday trading Tuesday Wednesday ⟨. . . ⟩
3	even get to
4	based days down home months up work years ⟨%⟩
5	those ⟨,⟩ ⟨—⟩
6	⟨.⟩ ⟨?⟩
7	eighty fifty forty ninety seventy sixty thirty twenty ⟨(⟩ ⟨·⟩
8	can could may should to will would
9	about at just only or than ⟨&⟩ ⟨;⟩
10	economic high interest much no such tax united well
11	president
12	because do how if most say so then think very what when where
13	according back expected going him plan used way
15	don’t I people they we you
16	Bush company court department more officials police retort spokesman
17	former the
18	American big city federal general house military national party political state union York
19	billion hundred million nineteen
20	did ⟨”⟩ ⟨’⟩
21	but called San ⟨:⟩ ⟨start-of-sentence⟩
22	bank board chairman end group members number office out part percent price prices rate sales shares use
23	a an another any dollar each first good her his its my old our their this
24	long Mr. year
25	business California case companies corporation dollars incorporated industry law money thousand time today war week ⟨)⟩ ⟨unknown⟩
26	also government he it market she that there which who
27	A. B. C. D. E. F. G. I. L. M. N. P. R. S. T. U.
28	both foreign international major many new oil other some Soviet stock these west world
29	after all among and before between by during for from in including into like of off on over since through told under until while with
30	eight fifteen five four half last next nine oh one second seven several six ten third three twelve two zero ⟨-⟩
31	are be been being had has have is it’s not still was were
32	chief exchange news public service trade

Table 2: Most probable assignments for the 300 most frequent words in an aggregate Markov model with $C = 32$ classes. Class 14 is absent because it is not the most probable class for any of the selected words.)

Latent variables combat sparsity

Chomsky (1957)

(1) Colorless green ideas sleep furiously.
(2) Furiously sleep ideas green colorless.

[T]he notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally 'remote' from English.

Pereira (2000)

By using this estimate for the probability of a string and an aggregate model with $C = 16$ trained on newspaper text, and by using the expectation–maximization (EM) method (Dempster *et al.* 1977), we find that

$$\frac{p(\text{Colourless green ideas sleep furiously})}{p(\text{Furiously sleep ideas green colourless})} \approx 2 \times 10^5.$$

Thus, a suitably constrained statistical model, even a very simple one, can meet Chomsky's particular challenge.

- Latent variables: let the model learn hidden structure in the data.
  - Typically for partial/un-supervised settings
- EM: a meta-algorithm for latent-variable learning
  - Use when observed-variable MLE is easy (e.g. count-estimated multinomial models) but marginal MLE is hard
  - Issues with local optima and convergence
- Alternatives
  - MCMC
  - Spectral learning