The Expectation-Maximization Algorithm and Statistical Machine Translation

Aaron Elkiss

EECS/SI 767

Outline

- Generative Models and Maximum Likelihood Estimation (MLE)
- The Problem: Incomplete Data
- The Algorithm: Expectation-Maximization (EM)
- Statistical Machine Translation: Model 1
- Hidden Markov Models: The Forward-Backward Algorithm
- Is EM Maximum-Likelihood?

Generative Models

- "Tell a Story" about how the data came to be
- Example: A Silly Generative Model
  - $G =$ have a good day
  - $B =$ get up on the wrong side of bed
  - $P(G, B) = P(G|B)P(B)$
  - Contrast with discriminative models

Probability vs. Likelihood

- Consider $P(x|\theta)$:
- Probability - hold θ (the model) fixed: $f(x) = P(X = x|\theta)$
- Likelihood - hold x (the observation) fixed: $f(\theta) = P(x|\Theta = \theta) = L(\theta|x)$

Maximum Likelihood

- Our Model: $P(G, B) = P(G|B)P(B)$
- Given a set of observations (G_i, B_i) we should pick the parameters $P(G|B)$, $P(G|\neg B)$, $P(B)$ for our model such that the parameters *maximize* the *likelihood* of the data

Why Use Maximum Likelihood Estimation?

- Bayesian Argument
- x - fixed observations
- θ - models under consideration
- $P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$
- or: $P(\theta|x) \propto L(\theta|x)P(\theta)$
- If all models are equally likely: $P(\theta|x) \propto L(\theta|x)$
- This is often a bad assumption!

A Maximum-Likelihood Estimate: Coin Tossing

- Observation: $x = (T, T, H, H, T, H, H, H, H, T, H, H, H, H, H, T)$
- ($n = 16$, $k = c(H) = 11$)
- Our model has one parameter, $P(H) = \alpha$
- Find $\arg\max_\alpha P(x|\alpha) = \arg\max_\alpha \binom{k}{n-k} \alpha^k (1-\alpha)^{n-k}$

A Maximum-Likelihood Estimate: Coin Tossing

$$\frac{d}{d\alpha}\binom{k}{n-k}\alpha^k(1-\alpha)^{n-k}$$

$$=\binom{k}{n-k}(k\alpha^{k-1}(1-\alpha)^{n-k}-(n-k)\alpha^k(1-\alpha)^{n-k-1})$$

$$=\binom{k}{n-k}\alpha^{k-1}(1-\alpha)^{n-k-1}(k(1-\alpha)-(n-k)\alpha)$$

- Solutions: $\alpha = 0$, $\alpha = 1$, $\alpha = \frac{k}{n}$

A Maximum-Likelihood Estimate: Coin Tossing

- What if you observe $x = (H, H)$?
- Discuss!

Maximum Likelihood

- Count-and-divide (or *relative-frequency*) estimation works in general
- Why? Proof sketch (Prescher):
- Maximizing $L(\theta|x)$ is equivalent to minimizing $D(\theta||\hat{\theta})$ where $\hat{\theta}$ is the relative-frequency estimate
- $D(\hat{\theta}||\hat{\theta}) = 0$ and $D(\cdot||\cdot) \geq 0$

The Problem: Incomplete Data

- Back to the earlier example:
- $G =$ have a good day
- $B =$ get up on the wrong side of bed
- $P(G, B) = P(G|B)P(B)$
- What if we only observe G?

The Problem: Incomplete Data

- Suppose we observe $x = (\neg G, \neg G, G, G, \neg G, \neg G, G, \neg G)$
- ($n = 8$, $G = 3$)
- If we had the complete data (a series of (G, B) observations), we could provide a maximum likelihood estimate for $P(B), P(G|B), P(G|\neg B)$
- If we had the parameters, we could find the expected complete data

The Solution: EM

- Pick a random initial guess for the parameters $\lambda = P(B), p_1 = P(G|B), p_2 = p(G|\neg B)$
- *E step*: Using our guess for λ, p_1, p_2, find the expected series of (G, B) observations
- *M step*: Using the expected series of (G, B) observations, update our guess of λ, p_1, p_2 with a new a maximum likelihood estimate
- Later we will (try to) show that this actually works!

The Solution: EM

- Pick a random initial guess for λ, p_1, p_2, say $\lambda = 0.4$, $p_1 = 0.8$, $p_2 = 0.3$
- *E-step*: Calculate expected counts: $\mathrm{E}[c(G,B)] = c(G)P(B|G)$

$$= c(G)\frac{P(B,G)}{P(G)} = c(G)\frac{P(B)P(G|B)}{P(G|B)P(B) + P(G|\neg B)P(\neg B)}$$

$$= c(G)\frac{\lambda p_1}{\lambda p_1 + (1-\lambda)p_2} = 1.92$$

- Similarly for $\mathrm{E}[c(G,\neg B)]$, $\mathrm{E}[c(\neg G,B)]$, $\mathrm{E}[c(G,\neg B)]$

The Solution: EM

- *M-step*: Calculate a new maximum likelihood estimate of λ, p_1, p_2

$$\lambda' = \frac{\mathrm{E}[c(G,B)] + \mathrm{E}[c(\neg G,B)]}{\mathrm{E}[c(G,B)] + \mathrm{E}[c(\neg G,B)] + \mathrm{E}[c(G,\neg B)] + \mathrm{E}[c(\neg G,\neg B)]}$$

- Similarly for p_1, p_2
- Iterate!

Mixture Models

- This is an instance of a *mixture model*
- Hidden parameter controls which model to use
- Models could be anything - e.g. Gaussian

Jason Eisner's Ice Cream Spreadsheet

Statistical Machine Translation: Model 1

- Model 1: $P(a, f|e) = \prod_{j=1}^{m} t(f_j|e_{a_j})$
- where
  - a is the alignment
  - f is the foreign sentence and f_j is the jth word
  - e is the English sentence and e_{a_j} is the English word f_j aligns to
  - m is the sentence length
  - $t(f_j|e_{a_j})$ is the probability of foreign word f_j given English word e_{a_j}
- Note the parameters of the model are the $t(\cdot, \cdot)$ values, the observed data are sentences f and e and the hidden data are the alignments a
- Not a very good model - but, easy to train!

EM for Model 1

- Suppose we have two sentence pairs: “b c” and “x y” are a pair as are “b” and “y”
- First: Set parameter values randomly
- *E step*: Compute expected counts $\mathrm{E}[c(t(f_j, e_{a_j}))]$ for all word pairs (f_j, e_{a_j})
  - To do this we have to compute $P(a, f|e)$ from our guess at $t(\cdot, \cdot)$
  - Then normalize to $P(a|e, f)$ and use to compute expected counts

EM for Model 1

- *M step*: Compute new maximum likelihood estimate of $t(\cdot,\cdot)$ from the expected counts
$$t(w_e|w_f) = \frac{\mathrm{E}[c(t(w_e|w_f))]}{\sum_{w \in W_e} \mathrm{E}[c(t(w|w_f))]}$$
where W_e is the set of all English words

- Iterate!

A More Sophisticated Model: The HMM

(Och and Ney 2000)

$$p(f_j, a_j | f_1^{j-1}, a_1^{j-1}, e_1^I) = \alpha(a_j | a_{j-1}, I) t(f_j | e_{a_j})$$

where α are the alignment or state transition probabilities and t are the translation or output probabilities. Hence

$$p(f_1^J | e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} [\alpha(a_j | a_j - 1, I) t(f_j | e_{a_j})]$$

Compare to probability of output sequence for normal HMM:

$$p(o_1^T) = \sum_{X_1^T} \prod_{t=1}^{T} a_{X_t X_{t+1}} b_{X_t X_{t+1} o_t}$$

Training HMMs: Forward-Backward

- Observed data: Output symbols
- Hidden data: State sequence
- Strategy: Collect partial counts for (output, state) pairs over all possible state sequences
- But there are too many possible state sequences!
- Solution: The Markov property!

Training HMMs: Markov Properties

- Forward probability: $P(o_1^{t-1}, X_t = i) = \alpha_i(t) = \sum_{j=1}^{n} s_j(t-1) a_{ij} b_{ijo_t}$
- Backward probability: $P(o_t^T | X_t = i) = \beta_i(t) = \sum_{j=1}^{n} a_{ij} b_{ijo_t} \beta_j(t+1)$

Training HMMs: E step

$$p_t(i,j) = \frac{\alpha_i(t)a_{ij}b_{ijo_t}\beta_j(t+1)}{\sum_{m=1}^{n}\alpha_m(t)\beta_m(t)}$$

- Expected count of transitions from i to j with k observed: $\mathrm{E}[c(i,j)] = \sum_{\{t:o_t=k\}} p_t(i,j)$
- Expected count of transitions from i to j: $\mathrm{E}[c(i,j)] = \sum_{t=1}^{T} p_t(i,j)$
- Expected count of transitions from i: $\mathrm{E}[c(i)] = \sum_{t=1}^{T}\sum_{j=1}^{n} p_t(i,j)$

Training HMMs: M step

$$\hat{a}_{ij} = \frac{\mathrm{E}[c(i,j)]}{\mathrm{E}[c(i)]}$$

$$\hat{b}_{ijk} = \frac{\mathrm{E}[c(i,j,k)]}{\mathrm{E}[c(i,j)]}$$

EM and Maximum Likelihood

- Consider average per-symbol log-likelihood $H = \sum_{y \in Y} \hat{p}(y) \log p(y)$
- The earlier proof from Prescher we sketched showed $L(\hat{p}|p) \propto H$
- We then consider

$$L(\hat{p}|p_{\lambda'}) - L(\hat{p}|p_{\lambda}) = \sum_{y} \hat{p}(y) \log \frac{p_{\lambda'}(y)}{p_{\lambda}(y)}$$

- See Berger for full proof - break $p_{\lambda'}(y)$ into $\sum_h p_{\lambda'}(y, h)$, apply Bayes' theorem, bound ≥ 0 using log-sum inequality
- Doesn't give a terribly intuitive derivation of the algorithm in terms of expected counts