Learning Mixture Models with EM

Micha Elsner

March 19, 2014
This lecture

A broader perspective on EM:

- Build from k-means to the broader class of mixture models
- Introduce “soft” EM and contrast with k-means style “hard” EM
- Reading: Spitkovsky et al: soft vs hard EM in practice
Review: k-means

- k-means: model has $k$ clusters
- Objective is to place centers so that each point is close to a center
- Alternate:
  - E-step (classification): associate each point with closest center
  - M-step (estimation): place center at mean of associated points
- k-means can be seen as generative model similar to Naive Bayes
New modeling possibilities

This view of k-means creates opportunities for us:

- What if we choose a different $P(d|C)$?
- What if $P(C)$ isn’t uniform?

We can obtain more complex *mixture models* for clustering

**Mixture model**

A model which has $k$ components, $C_1 \ldots C_k$...

- Each component has an associated distribution over the data, $P(d|C_i)$
- And a prior probability $P(C_i)$

- Naive Bayes is a mixture model
Clustering documents

For instance, what if our data are emails? Clustering (let $d_i$ be the text of document $i$)

- Suppose $P(d|C)$ is a language model
- And $P(C)$ is a number which we will estimate

This is a “mixture of language models”

- Or in stats class, a “mixture of multinomials”
EM for the mixture of language models

Objective:

\[
\max_{P(C), C_1=LM, C_2=LM} \sum_{d_i \in D} \max_{C^* \in [C_1, C_2]} \log P_{LM}(d_i|C^*) P(C^*)
\]

E-step (classification):

- Calculate probability of document \(d_i\) under LM \(C_1\) and \(C_2\)
- Multiply by \(P(C_1)\) and \(P(C_2)\) respectively
- Pick \(C^*\): the LM with highest probability

M-step (estimation):

- Take all documents for which \(C^* = C_1\)
- Estimate LM \(C_1\) using these as training data
- We also need to estimate prior \(P(C_1)\)
  - Max-likelihood...

\[
P(\hat{C}_1) = \frac{\#(C_i^* = C_1)}{n}
\]
Recap: we can make k-means style clustering models out of two ingredients:

- A model of data given the tag, \( P(d|C) \)
- A prior over tags, \( P(C) \)

We need a classification procedure and a training procedure

- And we alternate these

These are called *mixture models*
Maximizing data likelihood

k-means-like objectives maximize joint prob of data and label:

$$\max_{P(C), C_1, C_2} \sum_{d_i \in D} \max_{C^* \in [C_1, C_2]} \log P(d_i, C^*)$$

Many popular techniques instead aim to maximize the data probability:

$$\max_{P(C), C_1, C_2} \sum_{d_i \in D} \log P(d_i; C_1, C_2)$$

Why?

- The labels aren’t “real”... so why maximize their probability?
- Maximizing the data probability is what we do in supervised learning
- k-means-style objectives don’t give “partial credit”
Partial credit

An example is predicted by multiple classes (it’s ambiguous):

- For instance “raths”
- k-means-like objective: \( P(raths, \text{NN}) \)
  - Probability of “raths” as a noun
- Data only objective:
  \[
P(raths) = P(raths|\text{NN}) + P(raths|\text{VB})
  \]
  - Probability of “raths” as any tag
- If raths *might be* a verb, we will consider this possibility
- Which helps us allow more verbs that look like “raths”
Objective with partial credit

Our new objective:

\[
\max_{P(C), C_1, C_2} \sum_{d_i \in D} \log P(d_i; C_1, C_2)
\]

In our mixture model, \( P(d) \) is a marginal:

\[
P(d; C_1, C_2) = \sum_{C \in [C_1, C_2]} P(d, C) = \sum_{C \in [C_1, C_2]} P(d|C)P(C)
\]

(We’ve seen this before in Naive Bayes)

So we’d maximize:

\[
\max_{P(C), C_1, C_2} \sum_{d_i \in D} \sum_{C} P(d_i|C)P(C)
\]

(We can’t take a log inside anymore since we’re in a sum)
k-means-like (hard assignments) objective:

$$\max_{P(C), C_1, C_2} \sum_{d_i \in D} \max_{C^* \in [C_1, C_2]} \log P(d_i, C^*)$$

Data likelihood (soft assignments) objective:

$$\max_{P(C), C_1, C_2} \sum_{d_i \in D} \sum_{C \in [C_1, C_2]} P(d_i, C)$$

Inner max replaced with a sum

- If nothing is very ambiguous, these are basically the same
We can still use EM to maximize the partial credit version:
- If we aren’t sure whether a data point belongs in $C_1$ or $C_2$
- We change both $C_1$ and $C_2$ to account for it...
- Change in a cluster: proportional to our belief $d_i$ in that cluster
Soft EM for the mixture of language models

Objective:

\[
\max_{P(C), C_1=LM, C_2=LM} \sum_{d_i \in D} \max_{C^* \in [C_1, C_2]} \log P_{LM}(d_i|C^*)P(C^*)
\]

E-step (classification):

- Calculate probability of document \(d_i\) under LM \(C_1\) and \(C_2\), \(P(d|C)\)
- Multiply by \(P(C_1)\) and \(P(C_2)\) respectively to get \(P(C, d)\)
- Old version: Pick \(C^*\): the LM with highest probability
  - New version: calculate \(P(C|d) = \frac{P(C,d)}{\sum_{C'} P(C',d)}\)
  - Assign \(P(C_1|d)\) proportion of document to \(C_1\) and \(P(C_2|d)\) to \(C_2\)
Partial counts

What does it mean to assign .4 of a document to a cluster?

- When we estimate, we sum over these partial counts

Old: training data for $C_1$ language model:
“debates of the Senate”
“this is the Senate”
Counts:
  - the = 2, Senate = 2, debates = 1, …

New: training data for $C_1$ language model: “debates of the Senate” (prob .4)
“this is the Senate” (prob .8)
Counts:
  - the = 1.2, Senate = 1.2, debates = .4, this = .8…

Probabilities:

$$\hat{P}(the) = \frac{\#(the)}{\#\bullet} = \frac{1.2}{4.8}$$

(Or smoothed however….)
Partial counts (2)

Need to modify the \# count operator:

- Now a *weighted* count operator
- Calculates *expected* number of times you see the event
- If each instance $i$ is a “real” datapoint with prob $p_i$

\[
\#(\text{the}) = E[\text{the}] = \sum_{w_i=\text{the}} p_i
\]

This is where the “expectation” part of Expectation/Maximization comes from
Mixture of Gaussians

Close cousin of k-means

- Except:
  - Prior $P(C)$ doesn’t have to be uniform
  - Clusters don’t have to be spherical...
  - Both a mean and a covariance
  - So they’re ellipses

- Often inferred with soft rather than hard EM
Vowel example
Vowel example
Vowel example
Vowel example
Vowel example
Vowel example

Iteration 10: new zs
Vowel example

![Iteration 15: parameters](image-url)
Soft vs hard

Spitkovsky: hard EM might be better when:

- Your model is terrible
  - Perhaps it gets the right answer, but competing “ambiguous” answers are very wrong
- You care about something other than likelihood
  - Ex trying to learn “correct” (linguist-style) syntax
- You will do inference with Viterbi
  - Can help to match learning algorithm with inference algorithm

Soft EM:

- Find better likelihoods
- Possibly escape poor local maxima
- Known good results for many problems
Spitkovsky’s argument

(Paraphrases talk slides)

- EM redistributes wealth (probability mass) according to a central plan— like Communism
  - Knows the true value of everything
  - Could work if underlying model is powerful enough
- But model might be weak— reserves a lot of mass for ludicrous analyses
  - Eventually, exponentially many free-loaders
  - “A dog of a probability distribution... wagged by its very long tail”
- Viterbi EM driven by greed— like capitalism
Spitkovsky’s conclusion
Review

- We can construct EM algorithms for any mixture model
- By specifying a conditional distribution and a prior
- EM has hard and soft variants
- Soft variant maximizes data likelihood
- By giving partial credit to multiple clusters for ambiguous data
- Soft variant works by dividing up counts for ambiguous item according to $P(C|d)$