# EM algorithm and applications

Karthik Visweswariah[1]

[1]IBM Research, India

IISc, 2015

# Parametric density estimation

- Given data $\mathbf{x}$ and a density $p(x|\theta)$
- Find $\theta$ that explains data
- Maximimum likelhihood estimation
  - Choose parameter $\theta$ that maximizes $\prod_{i=1}^{n} p(x_i|\theta)$
  - Equivalently maximizes $\sum_{i=1}^{n} \log p(x_i|\theta)$
- Gaussian with identity covariance, mean as parameters
  - Choose $\mu$ that minimizes $\sum_i (x_i - \mu)^2$
  - Convex, unique global maximum, calculate in closed form
  - Set gradient to zero: $\mu = \frac{1}{n} \sum_i x_i$

# Parametric density estimation (contd.)

- Gaussian with mean and covariance as parameters
  - Choose $\mu$, $\Sigma$ that minimizes $\sum_i \frac{1}{2} \log |\Sigma| + (x_i - \mu)^T \Sigma^{-1} (x_i - \mu)$
  - Again: convex, unique global maximum, calculate in closed form
  - Set gradient to zero: $\mu = \frac{1}{n} \sum_i x_i$
  - $\Sigma = \frac{1}{n} (x_i - \mu)(x_i - \mu)^T$

# Parametric density estimation (contd.)

- Mixture of Gaussians, with means, covariances, mixture weights as parameters
  - Choose $\mu_j$, $\Sigma_j$, $\pi_j$ that maximizes:

    $\sum_i \log \sum_j \pi_j \frac{1}{|\Sigma_j|^{1/2}} \exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma^{-1}(x_i - \mu_j))$
  - No longer convex, no global maximum, no closed form soluion

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i, s|\theta)}{Q_i(s)}$

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i, s|\theta)}{Q_i(s)}$
- Jensen's inequality implies log of an average is greater than average of log

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i, s|\theta)}{Q_i(s)}$
- Jensen's inequality implies log of an average is greater than average of log
- $\sum_i \log \sum_s P(x_i, s|\theta) \geq \sum_i \sum_s Q_i(s) \log \frac{P(x_i, s|\theta)}{Q_i(s)}$

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i, s|\theta)}{Q_i(s)}$
- Jensen's inequality implies log of an average is greater than average of log
- $\sum_i \log \sum_s P(x_i, s|\theta) \geq \sum_i \sum_s Q_i(s) \log \frac{P(x_i, s|\theta)}{Q_i(s)}$
- If we can choose $Q_i(s)$ so that equality holds for our current parameter estimate, can increase objective by increasing the lower bound

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i,s|\theta)}{Q_i(s)}$
- Jensen's inequality implies log of an average is greater than average of log
- $\sum_i \log \sum_s P(x_i, s|\theta) \geq \sum_i \sum_s Q_i(s) \log \frac{P(x_i,s|\theta)}{Q_i(s)}$
- If we can choose $Q_i(s)$ so that equality holds for our current parameter estimate, can increase objective by increasing the lower bound
- How can we have $\sum_i \log \sum_s P(x_i, s|\theta_0) = \sum_i \sum_s Q_i(s) \log \frac{P(x_i,s|\theta_0)}{Q_i(s)}$

# Expectation maximization

- Maximize $\sum_i \log \sum_s P(x_i, s|\theta)$
- $\sum_i \log \sum_s P(x_i, s|\theta) = \sum_i \log \sum_s Q_i(s) \frac{P(x_i,s|\theta)}{Q_i(s)}$
- Jensen's inequality implies log of an average is greater than average of log
- $\sum_i \log \sum_s P(x_i, s|\theta) \geq \sum_i \sum_s Q_i(s) \log \frac{P(x_i,s|\theta)}{Q_i(s)}$
- If we can choose $Q_i(s)$ so that equality holds for our current parameter estimate, can increase objective by increasing the lower bound
- How can we have $\sum_i \log \sum_s P(x_i, s|\theta_0) = \sum_i \sum_s Q_i(s) \log \frac{P(x_i,s|\theta_0)}{Q_i(s)}$
- Need $\frac{P(x_i,s|\theta_0)}{Q_i(s)}$ to be independent of $s$

# Expectation maximization (contd.)

- Need $\frac{P(x_i, s|\theta_0)}{Q_i(s)}$ to be independent of $s$
- Take $Q_i(s) = cP(x_i, s|\theta_0)$
- Since $\sum_s Q_i(s) = 1$,

$$Q_i(s) = \frac{P(x_i, s|\theta_0)}{\sum_{s'} P(x_i, s'|\theta_0)}$$

# Expectation maximization (contd.)

- Need $\frac{P(x_i, s|\theta_0)}{Q_i(s)}$ to be independent of $s$
- Take $Q_i(s) = cP(x_i, s|\theta_0)$
- Since $\sum_s Q_i(s) = 1$,

$$Q_i(s) = \frac{P(x_i, s|\theta_0)}{\sum_{s'} P(x_i, s'|\theta_0)}$$

- This is just the posterior probability of $s$ given $x_i$

# Expectation maximization (contd.)

- Need $\frac{P(x_i, s|\theta_0)}{Q_i(s)}$ to be independent of $s$
- Take $Q_i(s) = cP(x_i, s|\theta_0)$
- Since $\sum_s Q_i(s) = 1$,

$$Q_i(s) = \frac{P(x_i, s|\theta_0)}{\sum_{s'} P(x_i, s'|\theta_0)}$$

- This is just the posterior probability of $s$ given $x_i$
- $Q_i(s) = P(s|x_i, \theta_0)$

# Expectation Maximization: putting it together

- How has this helped? Transformed to a form that hopefully we know how to maximize
- E-Step:
  Set $Q_i(s) = P(s|x_i, \theta_0)$
- M-Step:
  Choose $\theta$ to maximize
  $\sum_i \sum_s Q_i(s) \log P(x_i, s|\theta)$
- Maximisation problem is the same as in the non-mixture case
- If posterior is peaky, basically reduces to k-means type approach

# Expectation Maximization: Back to GMM case

- Update equation for the means, mixing weight, covariance:

$$\mu_j = \frac{\sum_i Q_i(j) x_i}{\sum_i Q_i(j)}$$

$$\Sigma_j = \frac{\sum_i Q_i(j)(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_i Q_i(j)}$$

$$\pi_j = \frac{\sum_i Q_i(j)}{\sum_s \sum_i Q_i(s)}$$

- Replace data by weighted data and essentially do usual updates

# Expectation Maximization: what about Hidden Markov Models

- Sequences instead of individual items
- E-Step: Set $Q(s^n) = P(s^n | x^n, \theta_0)$
- M-Step:
  Choose $\theta$ to maximize $\sum_{s^n} Q_i(s^n) \log P(x^n, s^n | \theta)$
- Computationally tractable with forward backward; to update models for a given state $s$, just need to weight data $x_t$ with the posterior probability of being in a certain state $s$ at time $t$
- Similar approach applies to unsupervised adaptation where we don't even know the word sequence
  - Usually make approximation that most likely path gets all the posterior

# Initialisation

- Need to initialize parameters (or posteriors)
- Divide data randomly into states
- For HMMs divide by evenly dividing data to states
- For GMMs could consider starting with a single component, and splitting till we get to the desired number

## Switch to a different application: Translation

- Corpus: Parallel sentences of English and French
- Goal: Translate French into English
- Same approach: Choose $e$ that maximises $P(f|e)P(e)$
- $P(e)$ is the same language model that we discussed earlier
- How do we model $P(f|e)$
- Agenda: Setup a simple enough model so you can code up the EM algorithm to estimate $P(f|e)$

# IBM Model 1

- Simplest possible model for $P(f|e)$

# IBM Model 1

- Simplest possible model for $P(f|e)$
- Sets up intermediate variables: Alignments
- $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m)$
- $a_i$ gives the English word generating the $i$th French word

# IBM Model 1

- Simplest possible model for $P(f|e)$
- Sets up intermediate variables: Alignments
- $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m)$
- $a_i$ gives the English word generating the $i$th French word
- Example: $e$: Prime Minister Modi visited the United States
  $f$: Premier ministre Modi a visité les États-Unis
  $a$: 1, 2, 3, 4, 4, 5, 6, 6

# IBM Model 1

- Simplest possible model for $P(f|e)$
- Sets up intermediate variables: Alignments
- $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m)$
- $a_i$ gives the English word generating the $i$th French word
- Example: $e$: Prime Minister Modi visited the United States
  $f$: Premier ministre Modi a visité les États-Unis
  $a$: 1, 2, 3, 4, 4, 5, 6, 6
- To get $P(f|e)$ marginalize over all possible alignments

# IBM Model 1 (Contd.)

- Still need more assumptions on joint distribution

# IBM Model 1 (Contd.)

- Still need more assumptions on joint distribution
- Assumption:
  $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m) = \prod_i P(a_i) P(f_i | e_{a_i})$

# IBM Model 1 (Contd.)

- Still need more assumptions on joint distribution
- Assumption:
  $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m) = \prod_i P(a_i)P(f_i | e_{a_i})$
- Also assume any English word equally likely to generate a French word: $P(a_i) = 1/l$

# IBM Model 1 (Contd.)

- Still need more assumptions on joint distribution
- Assumption:
  $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m) = \prod_i P(a_i)P(f_i | e_{a_i})$
- Also assume any English word equally likely to generate a French word: $P(a_i) = 1/l$
- In words: Generate French sentence by picking one of the English words (at random) and generating a french word using $P(f|e)$

# IBM Model 1 (Contd.)

- Still need more assumptions on joint distribution
- Assumption:
  $P(f_1, f_2, ..., f_m, a_1, a_2, ..., a_m | e_1, ...e_l, m) = \prod_i P(a_i)P(f_i | e_{a_i})$
- Also assume any English word equally likely to generate a French word: $P(a_i) = 1/l$
- In words: Generate French sentence by picking one of the English words (at random) and generating a french word using $P(f|e)$
- "The Mathematics of Statistical Machine Translation", Brown et. al.

# IBM Model 1 (Contd.)

- Are these reasonable assumptions?

# IBM Model 1 (Contd.)

- Are these reasonable assumptions?
- Clearly not
  - Order usually preserved
  - Can pick same English word many times to generate French
  - Context might matter for translation, ...

# IBM Model 1 (Contd.)

- Are these reasonable assumptions?
- Clearly not
  - Order usually preserved
  - Can pick same English word many times to generate French
  - Context might matter for translation, ...
- "All models are wrong but some are useful" (Box, 1978)

# IBM Model 1 (Contd.)

- Are these reasonable assumptions?
- Clearly not
  - Order usually preserved
  - Can pick same English word many times to generate French
  - Context might matter for translation, ...
- "All models are wrong but some are useful" (Box, 1978)
- IBM Model 1 still used as a step in building state of the art transation systems
- Objective function gets to global maximum; but parameter values not unique

http://research-srv.microsoft.com/pubs/150581/acl11.pdf

# IBM Model 1: Parameter estimation

- Parameters are $P(f|e)$ for every French word/English word pair
- Given: Parallel corpus of English-French sentences
- If we were given the alignments; then ML estimates would be:

# IBM Model 1: Parameter estimation

- Parameters are $P(f|e)$ for every French word/English word pair
- Given: Parallel corpus of English-French sentences
- If we were given the alignments; then ML estimates would be:
  $P(f|e) = \frac{N(f,e)}{\sum_{f'} N(f',e)}$
- Where $N(f,e)$ is the number of times word $f$ is aligned to word $e$

# IBM Model 1: Parameter estimation

- Parameters are $P(f|e)$ for every French word/English word pair
- Given: Parallel corpus of English-French sentences
- If we were given the alignments; then ML estimates would be: $P(f|e) = \frac{N(f,e)}{\sum_{f'} N(f',e)}$
- Where $N(f,e)$ is the number of times word $f$ is aligned to word $e$
- Since alignments are unknown...

## IBM Model 1: EM for parameter estimation

- E-Step: Posterior probability calculation
  $P(a_i = j | e, f) = P(f_i | e_j) / \sum_{j'} P(f_i | e_{j'})$

# IBM Model 1: EM for parameter estimation

- E-Step: Posterior probability calculation
  $P(a_i = j|e, f) = P(f_i|e_j) / \sum_{j'} P(f_i|e_{j'})$
- M-step: Parameter updates
  Accumulate counts $N(f_i, e_j) += P(a_i = j|e, f)$
  Normalize to get updates
  $P(f|e) = \frac{N(f,e)}{\sum_{f'} N(f',e)}$

# IBM Model 1: EM for parameter estimation

- E-Step: Posterior probability calculation
  $P(a_i = j|e, f) = P(f_i|e_j) / \sum_{j'} P(f_i|e_{j'})$
- M-step: Parameter updates
  Accumulate counts $N(f_i, e_j) += P(a_i = j|e, f)$
  Normalize to get updates
  $P(f|e) = \frac{N(f,e)}{\sum_{f'} N(f',e)}$
- For a single pass, accumulate counts for each sentence rather than waiting till end of corpus

# IBM Model 1: EM for parameter estimation

- E-Step: Posterior probability calculation
  $P(a_i = j|e, f) = P(f_i|e_j)/\sum_{j'} P(f_i|e_{j'})$
- M-step: Parameter updates
  Accumulate counts $N(f_i, e_j) += P(a_i = j|e, f)$
  Normalize to get updates
  $P(f|e) = \frac{N(f,e)}{\sum_{f'} N(f',e)}$
- For a single pass, accumulate counts for each sentence rather than waiting till end of corpus
- Initialisation
  - Initialize posteriors: Each French word equally likely to be generate by any of the English words in a sentence
    $P(a_i = j|e, f) = 1/I$

# For your entertainment

- Convince yourself calculations above are right and increase likelihood in each step

## For your entertainment

- Convince yourself calculations above are right and increase likelihood in each step
- Get French-English corpus: http://www.statmt.org/europarl/v7/fr-en.tgz
- Write a program to implement parameter estimation for Model 1 and see (using Google translate) if it learns reasonable estimates for the probabilities
- For faster runs, limit to 10k sentences

# For your entertainment

- Convince yourself calculations above are right and increase likelihood in each step
- Get French-English corpus: http://www.statmt.org/europarl/v7/fr-en.tgz
- Write a program to implement parameter estimation for Model 1 and see (using Google translate) if it learns reasonable estimates for the probabilities
- For faster runs, limit to 10k sentences
- Simple data pre-processing:
  - Make all words lower case
  - Separate ',' and '.' from preceding words
  - Split on space to get words

## For your entertainment

- Convince yourself calculations above are right and increase likelihood in each step
- Get French-English corpus: http://www.statmt.org/europarl/v7/fr-en.tgz
- Write a program to implement parameter estimation for Model 1 and see (using Google translate) if it learns reasonable estimates for the probabilities
- For faster runs, limit to 10k sentences
- Simple data pre-processing:
  - Make all words lower case
  - Separate ',' and '.' from preceding words
  - Split on space to get words
- Plug into translation model with a Unigram language model to translate French to English

## Sample output

- $P(f|administrative)$
  (0.40002900254924834, 'administratives'),
  (0.27302381868066222, 'administrative'),
  (0.14887685235760370, "d'"),
  (0.035485862464094235, 'gestion')
- $P(f|commissioner)$
  (0.69527480962463710, 'commissaire'),
  (0.19461810645247485, 'monsieur'),
  (0.04607403873838770, 'madame'),
  (0.04515723792614397, 'le')

# Sample output (contd.)

- Even works ok for words that occured a couple of times
- $P(f|abnormal)$
  (0.3069741827337813, 'anormale'),
  (0.3069218340043971, 'expansion'),
  (0.018469264480013486, 'visions')
- But fails on the most frequent word
- $P(f|the)$
  (0.23114959886960706, 'la'),
  (0.17679169454165436, 'de'),
  (0.12015766094814723, 'le'),
  (0.08767385651554384, ',')

# Sample output: evolution with iteration

- $P(f|commissioner)$
  Iter 0: (0.07, ','), (0.04, '.'), (0.04, 'le')
  Iter 1: (0.28, 'commissaire'), (0.10, ','), (0.07, 'monsieur')
  Iter 9: (0.69, 'commissaire'), (0.19, 'monsieur'), (0.04, 'madame')

Thanks. Questions?