http://www.youtube.com/watch?v=OR_-Y-eIlQo

UNSUPERVISED LEARNING

David Kauchak
CS 457 – Spring 2011

some slides adapted from Dan Klein

Admin

- Assignment 4 grades
- Assignment 5 part 1
- Quiz next Tuesday

Final project

- Read the entire handout
- Groups of 2-3 people
  - e-mail me asap if you're looking for a group
- research-oriented project
  - must involve some evaluation!
  - must be related to NLP
- Schedule
  - Tuesday 11/15 project proposal
  - 11/24 status report 1
  - 12/1 status report 2
  - 12/9 writeup (last day of classes)
  - 12/6 presentation (final exam slot)
- There are lots of resources out there that you can leverage

Supervised learning: summarized

- Classification
  - Bayesian classifiers
    - Naïve Bayes classifier (linear classifier)
  - Multinomial logistic regression (linear classifier)
    - aka Maximum Entropy Classifier (MaxEnt)
- Regression
  - linear regression (fit a line to the data)
  - logistic regression (fit a logistic to the data)

Supervised learning: summarized

- NB vs. multinomial logistic regression
  - NB has stronger assumptions: better for small amounts of training data
  - MLR has more realistic independence assumptions and performs better with more data
  - NB is faster and easier
- Regularization
- Training
  - minimize an error function
  - maximize the likelihood of the data (MLE)

A step back: data

Why do we need computers for dealing with natural text?


The Official Google Blog

Insights from Googlers into our products, technology, and the Google culture.

We knew the web was big...

7/25/2008 10:12:00 AM

We've known it for a long time: the web is big. The first Google index in 1998 already had 26 million pages, and by 2000 the Google index reached the one billion mark. Over the last eight years, we've seen a lot of big numbers about how much content is really out there. Recently, even our search engineers stopped in awe about just **how** big the web is these days -- when our systems that process links on the web to find new content hit a milestone: 1 trillion (as in 1,000,000,000,000) unique URLs on the web at once!

How do we find all those pages? We start at a set of well-connected initial pages and follow each of their links to new pages. Then we follow the links on those new pages to even more pages and so on, until we have a huge list of links. In fact, we found even more than 1 trillion individual links, but not all of them lead to unique web pages. Many pages have multiple URLs with exactly the same content or URLs that are auto-generated copies of each other. Even after removing those exact duplicates, we saw a trillion unique URLs, and the number of individual web pages out there is growing by several billion pages per day.

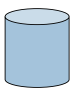
Web is just the start…

e-mail

247 billion e-mails a day

27 million tweets a day

corporate databases



Blogs: **126 million** different blogs

http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/

Corpora examples

- Linguistic Data Consortium
  - http://www.ldc.upenn.edu/Catalog/byType.jsp
- Dictionaries
  - WordNet – 206K English words
  - CELEX2 – 365K German words
- Monolingual text
  - Gigaword corpus
    - 4M documents (mostly news articles)
    - 1.7 trillion words
    - 11GB of data (4GB compressed)

Corpora examples

- Monolingual text continued
  - Enron e-mails
    - 517K e-mails
  - Twitter
  - Chatroom
  - Many non-English resources
- Parallel data
  - ~10M sentences of Chinese-English and Arabic-English
  - Europarl
    - ~1.5M sentences English with 10 different languages

Corpora examples

- Annotated
  - Brown Corpus
    - 1M words with part of speech tag
  - Penn Treebank
    - 1M words with full parse trees annotated
  - Other Treebanks
    - Treebank refers to a corpus annotated with trees (usually syntactic)
    - Chinese: 51K sentences
    - Arabic: 145K words
    - many other languages…
    - BLIPP: 300M words (automatically annotated)

Corpora examples

- Many others…
  - Spam and other text classification
  - Google n-grams
    - 2006 (24GB compressed!)
    - 13M unigrams
    - 300M bigrams
    - ~1B 3,4 and 5-grams
  - Speech
  - Video (with transcripts)

Problem

Labeled

Penn Treebank
1M words with full parse trees annotated

☹

Unlabeled

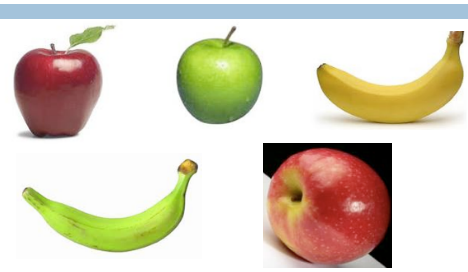
web

1 trillion web pages

e-mail

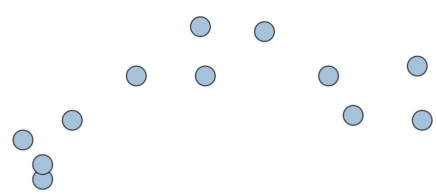
247 billion e-mails a day

Unsupervised learning



Unupervised learning: given data, but no labels

How would you group these points?

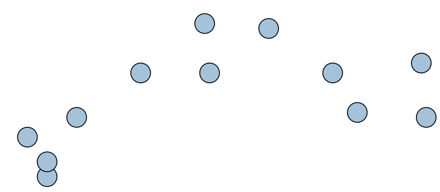


K-Means

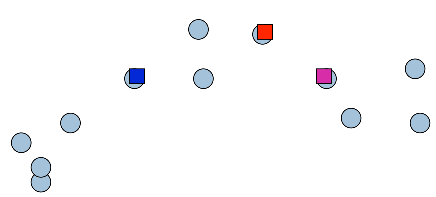
- Most well-known and popular clustering algorithm
- Start with some initial cluster centers
- Iterate:
  - Assign/cluster each example to closest center
  - Recalculate centers as the mean of the points in a cluster, c:

$$\vec{\mu}(c) = \frac{1}{|c|}\sum_{\vec{x}\in c}\vec{x}$$

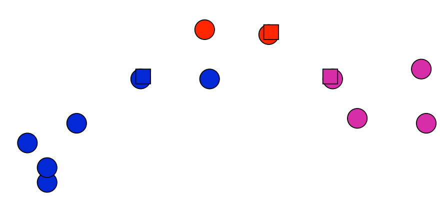
K-means



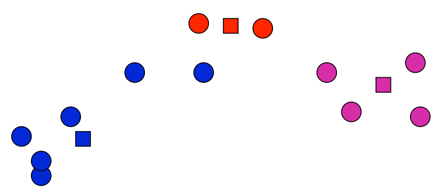
K-means: Initialize centers randomly



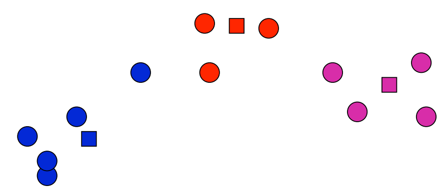
K-means: assign points to nearest center



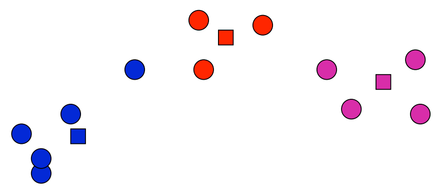
K-means: readjust centers



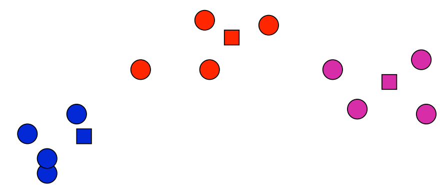
K-means: assign points to nearest center



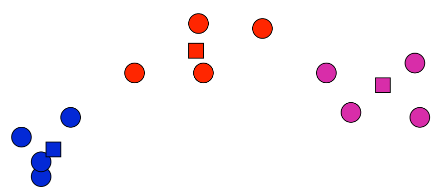
K-means: readjust centers



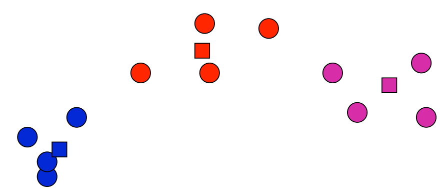
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center



No changes: Done

K-means variations/parameters

- Initial (seed) cluster centers
- Convergence
  - A fixed number of iterations
  - partitions unchanged
  - Cluster centers don't change
- K

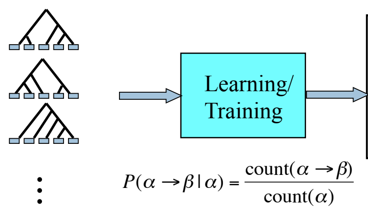
Hard vs. soft clustering

- Hard clustering: Each example belongs to exactly one cluster
- Soft clustering: An example can belong to more than one cluster (probabilistic)
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes

Learning a grammar

Parsed sentences

Grammar



S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

Parsing other data sources

What if we wanted to parse sentences from the web?

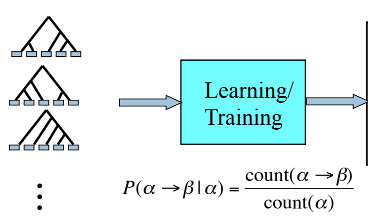
web

1 trillion web pages

Idea 1

Penn Treebank

Penn Grammar



S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

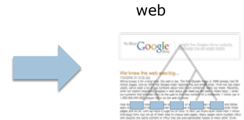
Idea 1

Penn Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

web



1 trillion web pages

How well will this work?

Parsing other data sources

What if we wanted to parse "sentences" from twitter?

27 million tweets a day

Idea 1

Penn Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

27 million tweets a day

Probably not going to work very well

Ideas?

Idea 2

Learning/ Training

Pseudo-Twitter grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

Idea 2

Pseudo-Twitter grammer

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

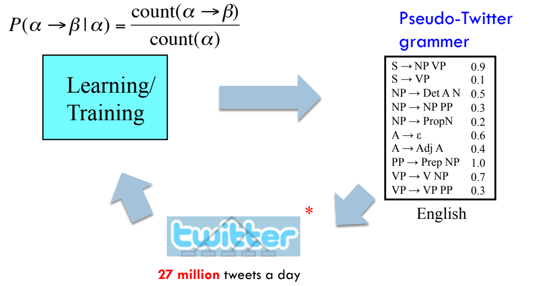
*

27 million tweets a day

Often, this improves the parsing performance

Idea 3

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$



Idea 3: some things to think about

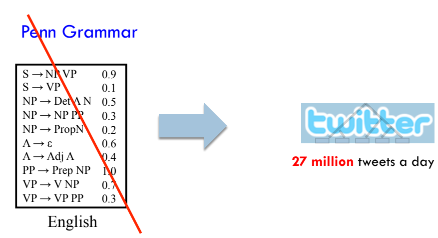
- How many iterations should we do it for?
  - When should we stop?
- Will we always get better?
- What does "get better" mean?

Idea 3: some things to think about

- How many iterations should we do it for?
  - We should keep iterating as long as we improve
- Will we always get better?
  - Not guaranteed for most measures
- What does "get better" mean?
  - Use our friend the development set
  - Does it increase the likelihood of the training data

Idea 4

What if we don't have any parsed data?



Idea 4

Randomly initialized grammar

S → NP VP	?
S → VP	?
NP → Det A N	?
NP → NP PP	?
NP → PropN	?
A → ε	?
A → Adj A	?
PP → Prep NP	?
VP → V NP	?
VP → VP PP	?

English

Pseudo-random

twitter

27 million tweets a day

Idea 4

Pseudo-random

twitter

Learning/ Training

Pseudo-Twitter grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

Idea 4

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

Learning/ Training

Pseudo-Twitter grammer

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

twitter *

27 million tweets a day

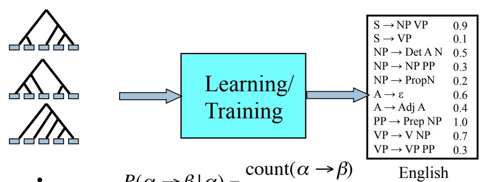
Idea 4

- Viterbi approximation of EM
  - Fast
  - Works ok (but we can do better)
  - Easy to get biased based on initial randomness
- What information is the Viterbi approximation throwing away?
  - We're somewhat randomly picking the best parse
  - We're ignoring all other possible parses
  - Real EM takes these into account

A digression

Why is this called Maximum Likelihood Estimation (MLE)?

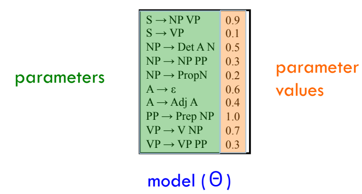
Parsed sentences Grammar



$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

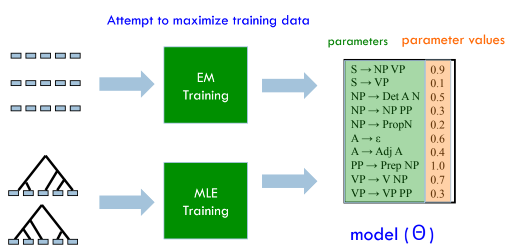
MLE

- Maximum likelihood estimation picks the values for the model parameters that maximize the likelihood of the training data



Expectation Maximization (EM)

- EM also tries to maximized the likelihood of the training data
  - EM works without labeled training data, though!
- However, because we don't have labeled data, we cannot calculate the exact solution in closed form



EM is a general framework

- Create an initial model, θ'
  - Arbitrarily, randomly, or with a small set of training examples
- Use the model θ' to obtain another model θ such that

  $\sum_i \log P_\theta(\text{data}_i) > \sum_i \log P_{\theta'}(\text{data}_i)$ i.e. better models data (increased log likelihood)
- Let θ' = θ and repeat the above step until reaching a local maximum
  - Guaranteed to find a better model after each iteration

Where else have you seen EM?

EM shows up all over the place

- Training HMMs (Baum-Welch algorithm)
- Learning probabilities for Bayesian networks
- EM-clustering
- Learning word alignments for language translation
- Learning Twitter friend network
- Genetics
- Finance
- Anytime you have a model and unlabeled data!

E and M steps: creating a better model

Expectation: Given the current model, figure out the expected probabilities of the each example

$p(x \mid \theta_c)$

What is the probability of each point belonging to each cluster?

What is the probability of sentence being grammatical?

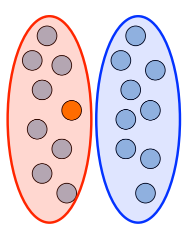
Maximization: Given the probabilities of each of the examples, estimate a new model, θ_c

Just like maximum likelihood estimation, except we use fractional counts instead of whole counts

EM clustering

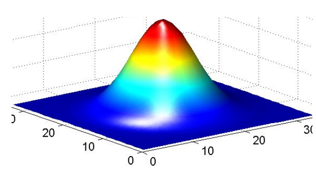
- We have some points in space
- We would like to put them into some known number of groups (e.g. 2 groups/clusters)
- Soft-clustering: rather than explicitly assigning a point to a group, we'll probabilistically assign it

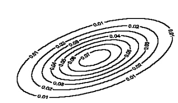
P(red) = 0.75
P(blue) = 0.25



EM clustering
Model: mixture of Gaussians

$$N[x;\mu,\Sigma] = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}}\exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$





Covariance determines the shape of these contours

- Fit these Gaussian densities to the data, one per cluster

E and M steps: creating a better model

Expectation: Given the current model, figure out the expected probabilities of the data points to each cluster

$p(x \mid \theta_c)$ What is the current probability of each point belonging to each cluster?

Maximization: Given the probabilistic assignment of all the points, estimate a new model, θ_c

Do MLE of the parameters (i.e. Gaussians), but use fractional counts based on probabilities (i.e. $p(x \mid \Theta_c)$)

EM example

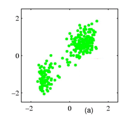


Figure from Chris Bishop

EM example

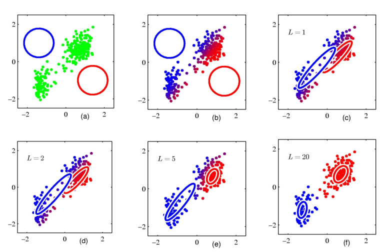


Figure from Chris Bishop

EM for parsing (Inside-Outside algorithm)

Expectation: Given the current model, figure out the expected probabilities of the each example

$p(x \mid \theta_c)$ What is the probability of sentence being grammatical?

Maximization: Given the probabilities of each of the examples, estimate a new model, θ_c

Just like maximum likelihood estimation, except we use fractional counts instead of whole counts

Expectation step

p(sentence)$_{grammar}$

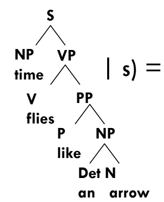
p(*time flies like an arrow*)$_{grammar}$ = ?

Note: This is the language modeling problem

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Most likely parse?

p(

| S) = p(S → NP VP | S) * p(NP → time | NP)

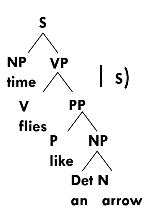
* p(VP → V PP | VP)

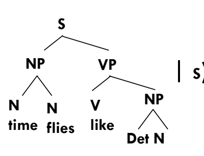
* p(V → flies | V) * …

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Sum over all the possible parses!
Often, we really want: p(*time flies like an arrow* | S)

p(

| S) + p(

| S) + …

Expectation step

p(*time flies like an arrow*)$_{grammar}$ = ?

Sum over all the possible parses!
Often, we really want: p(*time flies like an arrow* | S)

how can we calculate this sum?

Expectation step

p(*time flies like an arrow*)$_{\text{grammar}}$ = ?

Sum over all the possible parses!
Often, we really want: p(*time flies like an arrow* | S)

CKY parsing except sum over possible parses instead of max

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6*.0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S:.05*.5*.000864 =.0000216

S:.03*.0135*.032 =.00001296

For any entry, sum over the possibilities!

$S \rightarrow VP\ PP \quad 0.03$

$S \rightarrow Verb\ NP\ 0.05$

Maximization step

- Calculate the probabilities of the grammar rules using partial counts

MLE

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

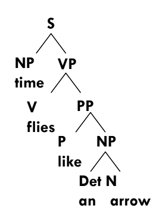
EM

?

Maximization step

Say we're trying to figure out VP -> V PP

MLE



count this as one occurrence

EM

$$p(VP \rightarrow V\ PP \mid \text{time flies like an arrow, S})$$

fractional count based on the sentence and how likely the sentence is to be grammatical

Maximization step

$$p(VP \rightarrow V\ PP \mid \text{time flies like an arrow, S})$$

$$= \frac{p(VP \rightarrow V\ PP, \text{time flies like an arrow} \mid \text{S})}{\text{p(time flies like an arrow} \mid \text{S)}}$$

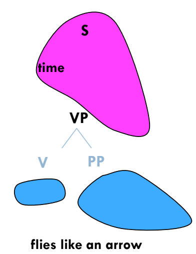
def. of conditional probability

$$= \frac{p(VP \rightarrow V\ PP)\, p(\text{time VP} \mid \text{S})\ \text{p(left - side} \mid \text{V) p(right - side} \mid \text{PP)}}{\text{p(time flies like an arrow} \mid \text{S)}}$$

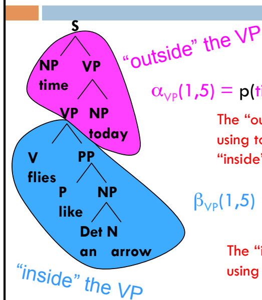
conditional independence as specified by the PCFG

Maximization step

$$\frac{p(VP \rightarrow V\ PP)\, p(\text{time VP} \mid \text{S})\ \text{p(left - side} \mid \text{V) p(right - side} \mid \text{PP)}}{\text{p(time flies like an arrow} \mid \text{S)}}$$



Inside & Outside Probabilities



$\alpha_{VP}(1,5)$ = p(time VP today | S)

The "outside" probabilities we can calculate using top-down approach (after we have the "inside" probabilities

$\beta_{VP}(1,5)$ = p(flies like an arrow | VP)

The "inside" probabilities we can calculate using a CKY-style, bottom-up approach

EM grammar induction

- The good:
  - We learn a grammar
  - At each step we're guaranteed to increase (or keep the same) the likelihood of the training data
- The bad
  - Slow: $O(m^3n^3)$, where m = sentence length and n = non-terminals in the grammar
  - Lot's of local maxima
  - Often have to use more non-terminals in the grammar than are theoretically motivated (often ~3 times)
  - Often non-terminals learned have no relation to traditional constituents

But…

- If we bootstrap and start with a reasonable grammar, we can often obtain very interesting results

Penn Grammar

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

EM: Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

- In machine translation, we train from pairs of translated sentences
- Often useful to know how the words align in the sentences
- Use EM: learn a model of P(french-word | english-word)

Idea?

EM: Finding Word Alignments

Expectation: Given the current model, figure out the expected probabilities of the each example

$p(x \mid \theta_c)$ What is the probability of this word alignment?

Maximization: Given the probabilities of each of the examples, estimate a new model, θ_c

Just like maximum likelihood estimation, except we use fractional counts instead of whole counts:

count the fractional counts of one word aligning to another

EM: Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

All word alignments equally likely

All P(french-word | english-word) equally likely

EM: Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

"la" and "the" observed to co-occur frequently,
so P(la | the) is increased.

EM: Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

"house" co-occurs with both "la" and "maison", but
P(maison | house) can be raised without limit, to 1.0,
while P(la | house) is limited because of "the"

(pigeonhole principle)

EM: Finding Word Alignments

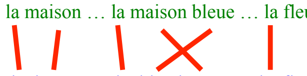
… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

settling down after another iteration

EM: Finding Word Alignments

… la maison … la maison bleue … la fleur …



… the house … the blue house … the flower …

Inherent hidden structure revealed by EM training!
For details, see

- "A Statistical MT Tutorial Workbook" (Knight, 1999).
  - 37 easy sections, final section promises a free beer.
- "The Mathematics of Statistical Machine Translation" (Brown et al, 1993)
- Software: GIZA++

Statistical Machine Translation

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

P(maison | house) = 0.411
P(maison | building) = 0.027
P(maison | manson) = 0.020
…

Estimating the model from training data

EM summary

- EM is a popular technique in NLP
- EM is useful when we have lots of unlabeled data
  - we may have some labeled data
  - or partially labeled data
- Broad range of applications
- Can be hard to get it right, though…

Human Parsing

- How do humans do it?
- How might you try and figure it out computationally/ experimentally?

Human Parsing

- Read these sentences
- Which one was fastest/slowest?

John put the dog in the pen with a lock.

John carried the dog in the pen with a bone in the car.

John liked the dog in the pen with a bone.

Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.
  - John put the dog in the pen with a lock.
  - John carried the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

Human Parsing

- Computational parsers can be used to predict human reading time as measured by tracking the time taken to read each word in a sentence.
- Psycholinguistic studies show that words that are more probable given the preceding lexical and syntactic context are read faster.
  - John put the dog in the pen with a lock.
  - John carried the dog in the pen with a bone in the car.
  - John liked the dog in the pen with a bone.
- Modeling these effects requires an ***incremental*** statistical parser that incorporates one word at a time into a continuously growing parse tree.

Garden Path Sentences

- People are confused by sentences that seem to have a particular syntactic structure but then suddenly violate this structure, so the listener is "lead down the garden path".
  - The horse raced past the barn fell.
    - vs. The horse raced past the barn broke his leg.
  - The complex houses married students.
  - The old man the sea.
  - While Anna dressed the baby spit up on the bed.
- Incremental computational parsers can try to predict and explain the problems encountered parsing such sentences.