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
    - Naive 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)

**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?

"We know the web was big..."

"We now know it's a megatrend that the web is big. The first Google crawl in 1998 already had 20 million pages, and by 2000 the Google index reached the one billion mark. Over the last eight years, we have been adding two billion pages per year to the index. This means that the average user is looking at over 200 million unique URLs on the web every day."

"How do we find all those pages? We start at a set of well-connected initial pages and follow each of the links to new pages. Then we divide and conquer, covering one topic at a time. Ingredients: crawling, ranking, algorithms. etwa 1,265,000,000,000 unique URLs on the web today!"

"After 200 million unique pages, we can't fit all of them into a single database. So we have to break it into smaller databases. This also allows us to do more advanced queries and search results to the user."

"With the growth of the web, we are now looking at the web as a whole. The web is not just a collection of individual pages, but rather a single, interconnected system."

"When we talk about "size" of the web, we mean the number of individual pages that we can access. The number of pages on the web is growing at a rate of several billion pages per day."
Web is just the start...

e-mail

247 billion e-mails a day

twitter

27 million tweets a day

corporate databases

Blogs: 126 million different blogs


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...
      - BUPT: 300M words (automatically annotated)
Corpora examples

- Many others...
  - Spam and other text classification
  - Google n-grams
    - 2006 (24GB compressed!)
    - 13M unigrams
    - 300M bigrams
    - ~18 3,4 and 5-grams
  - Speech
  - Video (with transcripts)

Problem

<table>
<thead>
<tr>
<th>Labeled</th>
<th>Unlabeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penn Treebank 1M words with full parse trees annotated</td>
<td></td>
</tr>
<tr>
<td>[Image] 1 trillion web pages</td>
<td></td>
</tr>
<tr>
<td>[Image] 247 billion e-mails a day</td>
<td></td>
</tr>
</tbody>
</table>

Unsupervised learning

Unsupervised 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/clustering each example to closest center
  - Recalculate centers as the mean of the points in a cluster, \( c \):

\[
\mu_c = \frac{1}{|c|} \sum_{x \in c} x
\]
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

Learning/Training

Grammar

$S \rightarrow \text{NP VP}$ 0.9
$S \rightarrow \text{VP}$ 0.1
$\text{NP} \rightarrow \text{Det AN}$ 0.5
$\text{NP} \rightarrow \text{NP PP}$ 0.3
$\text{NP} \rightarrow \text{Prop N}$ 0.2
$A \rightarrow \epsilon$ 0.4
$\text{A} \rightarrow \text{Adj A}$ 0.4
$\text{PP} \rightarrow \text{Prep NP}$ 0.1
$\text{VP} \rightarrow \text{V NP}$ 0.7
$\text{VP} \rightarrow \text{VP PP}$ 0.3

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

English

Parsing other data sources

What if we wanted to parse sentences from the web?

web

1 trillion web pages

Idea 1

Penn Treebank

Learning/Training

Penn Grammar

$S \rightarrow \text{NP VP}$ 0.9
$S \rightarrow \text{VP}$ 0.1
$\text{NP} \rightarrow \text{Det AN}$ 0.5
$\text{NP} \rightarrow \text{NP PP}$ 0.3
$\text{NP} \rightarrow \text{Prop N}$ 0.2
$A \rightarrow \epsilon$ 0.4
$\text{A} \rightarrow \text{Adj A}$ 0.4
$\text{PP} \rightarrow \text{Prep NP}$ 0.1
$\text{VP} \rightarrow \text{V NP}$ 0.7
$\text{VP} \rightarrow \text{VP PP}$ 0.3

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

English

Idea 1

Penn Grammar

Learning/Training

Penn Grammar

$S \rightarrow \text{NP VP}$ 0.7
$S \rightarrow \text{VP}$ 0.1
$\text{NP} \rightarrow \text{Det AN}$ 0.3
$\text{NP} \rightarrow \text{NP PP}$ 0.2
$\text{NP} \rightarrow \text{Prop N}$ 0.2
$A \rightarrow \epsilon$ 0.4
$A \rightarrow \text{Adj A}$ 0.4
$\text{PP} \rightarrow \text{Prep NP}$ 0.1
$\text{VP} \rightarrow \text{V NP}$ 0.7
$\text{VP} \rightarrow \text{VP PP}$ 0.3

How well will this work?
What if we wanted to parse "sentences" from Twitter?

**Idea 1**

Penn Grammar:

- $S \rightarrow NP \ VP$
- $S \rightarrow VP \ \ NP$
- $NP \rightarrow \ Det \ A \ N$
- $NP \rightarrow \ PropN \ \ NP$
- $A \rightarrow \ Adj \ A$
- $A \rightarrow \ Adj \ A \ A$
- $PP \rightarrow \ Prep \ NP$
- $VP \rightarrow \ V \ NP$
- $VP \rightarrow \ VP \ PP$

English:

27 million tweets a day

Probably not going to work very well

Ideas?

**Idea 2**

Pseudo-Twitter grammar:

- $S \rightarrow NP \ VP$
- $S \rightarrow VP \ NP$
- $NP \rightarrow \ Det \ A \ N$
- $NP \rightarrow \ PropN \ NP$
- $A \rightarrow \ Adj \ A$
- $A \rightarrow \ Adj \ A \ A$
- $PP \rightarrow \ Prep \ NP$
- $VP \rightarrow \ V \ NP$
- $VP \rightarrow \ VP \ PP$

English:

Often, this improves the parsing performance
Idea 3

\[ P(\alpha \rightarrow \beta | \alpha) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)} \]

Learning/Training

27 million tweets a day

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?

Penn Grammar

27 million tweets a day
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

Learning/Training

P(α → β|α) = count(α → β) / count(α)

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 0.0
A → Adj A 0.6
PP → Prep NP 0.4
VP → V NP 0.3
VP → VP PP 0.1

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

27 million tweets a day
A digression

Why is this called Maximum Likelihood Estimation (MLE)?

Parsed sentences

Learning/Training

Grammar

$S \rightarrow NP \ VP$

$S \rightarrow VP$

$NP \rightarrow \text{Det} \ A \ N$

$NP \rightarrow \text{Prop} \ N$

$A \rightarrow \varepsilon$

$A \rightarrow \text{Adj} \ A$

$PP \rightarrow \text{Prep} \ NP$

$VP \rightarrow V \ NP$

$VP \rightarrow VP \ PP$

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

MLE

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

$S \rightarrow NP \ VP$

$S \rightarrow VP$

$NP \rightarrow \text{Det} \ A \ N$

$NP \rightarrow \text{Prop} \ N$

$A \rightarrow \varepsilon$

$A \rightarrow \text{Adj} \ A$

$PP \rightarrow \text{Prep} \ NP$

$VP \rightarrow V \ NP$

$VP \rightarrow VP \ PP$

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

$\text{English}

Expectation Maximization (EM)

EM also tries to maximize 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

Attempt to maximize training data

EM

Training

MLE

Training

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

EM is a general framework

- Create an initial model, $\theta'$
  - Arbitrarily, randomly, or with a small set of training examples

- Use the model $\theta'$ to obtain another model $\theta$ such that
  $\sum \log P_{\theta}(\text{data}) > \sum \log P_{\theta'}(\text{data})$
  
  i.e. better models data (increased log likelihood)

- Let $\theta' = \theta$ 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!

**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

\[ P(\text{red}) = 0.75 \]
\[ P(\text{blue}) = 0.25 \]

**EM clustering**

Model: mixture of Gaussians

\[
A(x|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2} \sqrt{\det(\Sigma)}} \exp\left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)
\]

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

\[ p(x | \theta_c) \]

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, \( \theta_c \)

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

---

EM for parsing (Inside-Outside algorithm)

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

\[ p(x | \theta_c) \]

What is the probability of sentence being grammatical?

**Maximization:** Given the probabilities of each of the examples, estimate a new model, \( \theta_c \)

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

---
Expectation step

\[ p(\text{sentence})_{\text{grammar}} \]
\[ p(\text{time flies like an arrow})_{\text{grammar}} = ? \]

Note: This is the language modeling problem

Expectation step

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

Most likely parse?

\[
p(\text{s}) = p(\text{s} \rightarrow \text{NP} \text{ VP} | \text{s}) \times p(\text{NP} \rightarrow \text{time} | \text{NP}) \\
\times p(\text{VP} \rightarrow \text{V} \text{ PP} | \text{VP}) \\
\times p(\text{V} \rightarrow \text{flies} | \text{V}) \times ... 
\]

Expectation step

Sum over all the possible parses!
Often, we really want: \[ p(\text{time flies like an arrow} | \text{S}) \]

\[
p(\text{s}) = p(\text{s} \rightarrow \text{NP} \text{ VP} | \text{s}) + p(\text{s} \rightarrow \text{NP} \text{ VP} | \text{s}) + ... 
\]

Expectation step

Sum over all the possible parses!
Often, we really want: \[ p(\text{time flies like an arrow} | \text{S}) \]

how can we calculate this sum?
Expectation step

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

Sum over all the possible parses!

Often, we really want: \( p(\text{time flies like an arrow} \mid S) \)

CKY parsing except sum over possible parses instead of max

Maximization step

- Calculate the probabilities of the grammar rules using partial counts

\[ \text{MLE} \quad \text{EM} \]

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

\[ ? \]

Probabilistic CKY Parser

Maximization step

Say we're trying to figure out \( VP \rightarrow V PP \)

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

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

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

\[ = \frac{p(\text{VP} \rightarrow \text{V PP}, \text{time flies like an arrow} | S)}{p(\text{time flies like an arrow} | S)} \quad \text{def. of conditional probability} \]

\[ = \frac{p(\text{VP} \rightarrow \text{V PP}) p(\text{time VP} | S) p(\text{left - side} | V) p(\text{right - side} | PP)}{p(\text{time flies like an arrow} | S)} \]

\[ \beta_{\text{V}}(1, 5) = p(\text{flies like an arrow} \mid \text{VP}) \]

Inside & Outside Probabilities

The “outside” 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

- $S \rightarrow NP \ VP$
- $S \rightarrow VP \ NP$
- $NP \rightarrow \text{Det} \ A \ N$
- $NP \rightarrow \text{PropN}$
- $A \rightarrow \epsilon$
- $A \rightarrow \text{Adj} \ A$
- $PP \rightarrow \text{Prep} \ NP$
- $VP \rightarrow V \ NP$
- $VP \rightarrow VP \ PP$

0.9
0.1
0.5
0.3
0.2
0.6
0.4
0.1
0.7
0.3

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 $P(\text{french-word} \ \mid \ \text{english-word})$

Idea?

EM: Finding Word Alignments

Expectation: Given the current model, figure out the expected probabilities of 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, $\theta_c$

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

All word alignments equally likely

All $P(\text{french-word} \ \mid \ \text{english-word})$ equally likely
EM: Finding Word Alignments

“la” and “the” observed to co-occur frequently, so \( P(\text{la} | \text{the}) \) is increased.

EM: Finding Word Alignments

“house” co-occurs with both “la” and “maison”, but \( P(\text{maison} | \text{house}) \) can be raised without limit, to 1.0, while \( P(\text{la} | \text{house}) \) is limited because of “the” (pigeonhole principle)

EM: Finding Word Alignments

settling down after another iteration

EM: Finding Word Alignments

Inherent hidden structure revealed by EM training!

Inherent hidden structure revealed by EM training!
Statistical Machine Translation

- la maison … la maison bleue … la fleur …
- the house … the blue house … the flower …

\[
P(\text{maison} \mid \text{house}) = 0.411 \\
P(\text{maison} \mid \text{building}) = 0.027 \\
P(\text{maison} \mid \text{mansion}) = 0.020 \\
\ldots
\]

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.

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.