Advanced Human Language Technologies

Statistical Models of Language

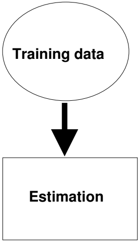
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

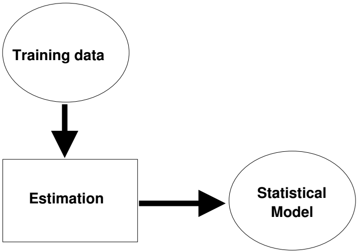
Outline

We model to make predictions

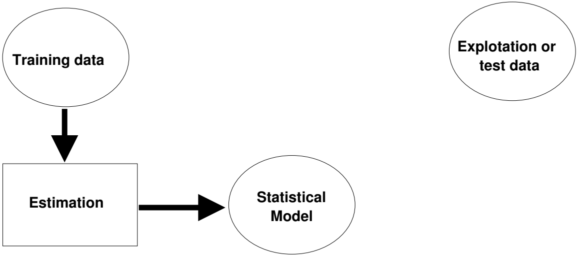
We model to make predictions



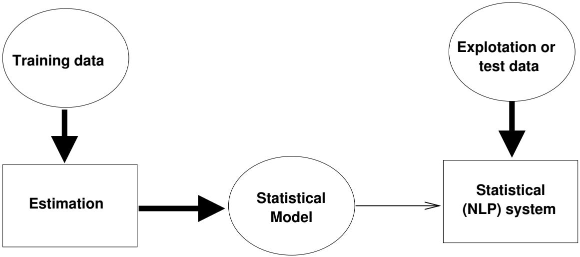
We model to make predictions



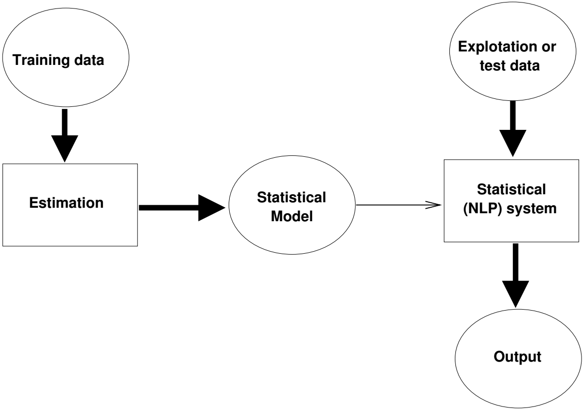
We model to make predictions



We model to make predictions



We model to make predictions



Outline

Prediction Models & Similarity Models

- **Prediction Models**: Oriented to *predict* probabilities of future events, knowing past and present.
- **Similarity Models**: Oriented to compute *similarities* between objects (may be used to predict, EBL). [

Outline

Outline

Prediction Models

Example: Noisy Channel Model (Shannon 48)

NLP Applications

Appl.	Input	Output	$p(i)$	$p(o \mid i)$
MT	L word sequence	M word sequence	$p(L)$	Translation model
OCR	Actual text	Text with mistakes	prob. of language text	model of OCR errors
PoS tagging	PoS tags sequence	word sequence	prob. of PoS sequence	$p(w \mid t)$
Speech recog.	word sequence	speech signal	prob. of word sequence	acoustic model

Given $\mathbf{o}$, we want to find the most likely $\mathbf{i}$

$$\operatorname*{argmax}_{\mathbf{i}} P(\mathbf{i} \mid \mathbf{o}) = \operatorname*{argmax}_{\mathbf{i}} P(\mathbf{o}, \mathbf{i}) = \operatorname*{argmax}_{\mathbf{i}} P(\mathbf{i}) P(\mathbf{o} \mid \mathbf{i})$$

Using Prediction Models

- Estimation: Using data to infer information about distributions. A.k.a. *learning*.
- Classification: Make predictions based on past behaviour
- In general, ML models estimate (i.e. *learn*) conditional probability distributions $\mathrm{P}(\text{target}|\text{features})$, e.g.:
  - language identification given word or subword features.
  - document category given words in it.
  - word PoS given context information.
  - ...
- Many NLP tasks require a posterior search step to find the best combination of predictions.

Outline

Outline

Finding good estimators: MLE

Maximum Likelihood Estimation (MLE)

- Choose the alternative that maximizes the probability of the observed outcome.
- $\bar{\mu}_n$ is a MLE for $E(X)$
- s_n^2 is a MLE for σ^2
- Zipf's Laws. Data sparseness. Smoothing tecnhiques.

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	0.10	0.15	0	0.08	0.03	0	0.40
on	0.06	0.25	0.10	0.15	0	0	0.04	0.60
total	0.10	0.35	0.25	0.15	0.08	0.03	0.04	1.0

Working Example: N-gram models

- Predict the next element in a sequence (e.g. next character, next word, next PoS, next stock value, ...), given the *history* of previous elements: $P(w_n \mid w_1 \ldots w_{n-1})$
- Markov assumption: Only *local* context (of size $n-1$) is taken into account. $P(w_i \mid w_{i-n+1} \ldots w_{i-1})$
- bigrams, trigrams, four-grams ($n = 2, 3, 4$).
  Sue swallowed the large green <?>
- Parameter estimation (number of equivalence classes)
- Parameter reduction: stemming, semantic classes, PoS, ...

Model	Parameters
bigram	$20,000^2 = 4 \times 10^8$
trigram	$20,000^3 = 8 \times 10^{12}$
four-gram	$20,000^4 = 1.6 \times 10^{17}$

Language model sizes for a 20,000 words vocabulary

N-gram model estimation

Estimate the probability of the target feature based on observed data. The prediction task can be reduced to having good estimations of the n-gram distribution:

$$P(w_n \mid w_1 \ldots w_{n-1}) = \frac{P(w_1 \ldots w_n)}{P(w_1 \ldots w_{n-1})}$$

- **MLE (Maximum Likelihood Estimation)**
  $P_{MLE}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n)}{N}$
  $P_{MLE}(w_n \mid w_1 \ldots w_{n-1}) = \frac{C(w_1 \ldots w_n)}{C(w_1 \ldots w_{n-1})}$
  - No probability mass for unseen events
  - Data sparseness, Zipf's Law
  - Unsuitable for NLP (widely used, though)

Brief Parenthesis: Zipf's Laws

Zipf's Laws (1929)

- Word frequency is inversely proportional to its rank (speaker/hearer minimum effort) $f \sim 1/r$
- Number of senses is proportional to frequency root $m \sim \sqrt{f}$
- Frequency of intervals between repetitions is inversely proportional to the length of the interval $F \sim 1/I$
- Frequency based approaches are hard, since most words are rare
  - Most common 5% words account for about 50% of a text
  - 90% least common words account for less than 10% of the text
  - Almost half of the words in a text occurr only once

Outline

Outline

Laplace's Law

LAPLACE'S LAW (adding one count)

General rule:

$$P_{LAP}(X = x) = \frac{C(X = x) + 1}{N + B}$$

N*: number of observations of* X
B*: number of potentially observable values for* X

N-gram probability:

$$P_{LAP}(w_1 \ldots w_n) = \frac{C(w_1 \ldots w_n) + 1}{N + B}$$

N*: total number of n-gram observations*
B*: number of potentially observable different n-grams*

N-gram conditional probability:

$$P_{LAP}(w_n \mid w_1 \ldots w_{n-1}) = \frac{C(w_1 \ldots w_n) + 1}{C(w_1 \ldots w_{n-1}) + B}$$

B*: number of potentially observable* w_n *values*

For large values of B *too much probability mass is assigned to unseen events.*

Lidstone's Law

LIDSTONE'S LAW (adding λ counts, with $\lambda < 1$)

General rule:

$$P_{LAP}(X = x) = \frac{C(X = x) + \lambda}{N + B\lambda}$$

N: *number of observations of* X
B: *number of potentially observable values for* X

N-gram probability:

$$P_{LAP}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n) + \lambda}{N + B\lambda}$$

N: *total number of n-gram observations*
B: *number of potentially observable different n-grams*

N-gram conditional probability:

$$P_{LAP}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n) + \lambda}{C(w_1 \dots w_{n-1}) + B\lambda}$$

B: *number of potentially observable* w_n *values*

Equivalent to linear interpolation between MLE and uniform prior: with $\mu = N/(N + B\lambda)$; $P_{LID}(X = x) = \mu \frac{C(X = x)}{N} + (1 - \mu)\frac{1}{B}$

Outline

Absolute Discounting

ABSOLUTE DISCOUNTING (discount δ counts, with $0 < \delta < 1$)

General rule:

$$P_{ABS}(X = x) = \begin{cases} \frac{C(X=x)-\delta}{N} & \text{if } C(w_1 \ldots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{N} & \text{otherwise} \end{cases}$$

N_0: *number of possible values for* X *observed* 0 *times*

N-gram probability:

$$P_{ABS}(w_1 \ldots w_n) = \begin{cases} \frac{C(w_1 \ldots w_n)-\delta}{N} & \text{if } C(w_1 \ldots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{N} & \text{otherwise} \end{cases}$$

N-gram conditional probability:

$$P_{ABS}(w_n \mid w_1 \ldots w_{n-1}) = \begin{cases} \frac{C(w_1 \ldots w_n)-\delta}{C(w_1 \ldots w_{n-1})} & \text{if } C(w_1 \ldots w_n) > 0 \\ \frac{(B-N_0)\delta/N_0}{C(w_1 \ldots w_{n-1})} & \text{otherwise} \end{cases}$$

N_0: *number of possible values for* w_n *observed* 0 *times*

Linear Discounting

LINEAR DISCOUNTING (discount a proportion α of counts)

General rule:

$$P_{\mathrm{LIN}}(X = x) = \begin{cases} (1-\alpha)\frac{C(X=x)}{N} & \text{if } C(X = x) > 0 \\ \alpha / N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible values for* X *observed* 0 *times*

N-gram probability:

$$P_{\mathrm{LIN}}(w_1 \dots w_n) = \begin{cases} (1-\alpha)\frac{C(w_1 \dots w_n)}{N} & \text{if } C(w_1 \dots w_n) > 0 \\ \alpha / N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible n-grams observed* 0 *times*

N-gram conditional probability:

$$P_{\mathrm{LIN}}(w_n \mid w_1 \dots w_{n-1}) = \begin{cases} (1-\alpha)\frac{C(w_1 \dots w_n)}{C(w_1 \dots w_{n-1})} & \text{if } C(w_1 \dots w_n) > 0 \\ \alpha / N_0 & \text{otherwise} \end{cases}$$

N_0: *number of possible values for* w_n *observed* 0 *times*

Outline

Combining Estimators

Combining estimators for conditional n-gram probabilities

Linear Interpolation:

$$
\begin{aligned}
P_{LI}(w_n \mid w_{n-2}, w_{n-1}) &= \lambda_1 P_1(w_n) \\
&+ \lambda_2 P_2(w_n \mid w_{n-1}) \\
&+ \lambda_3 P_3(w_n \mid w_{n-2}, w_{n-1})
\end{aligned}
$$

Backing-off:

$$
P_{BO}(w_n \mid h_i) = \begin{cases} \alpha_{h_i} \dfrac{C(h_i, w_n)}{C(h_i)} & \text{if } C(h_i, w_n) > k \\ \delta_{h_{i-1}} P_{BO}(w_n \mid h_{i-1}) & \text{otherwise} \end{cases}
$$

$h_i = w_{n-i} \dots w_{n-1}$: *recent history of length* i.
α_{h_i}: *remaining proportion after discount.*
$\delta_{h_{i-1}}$: *mass assigned to back-off distribution.*

- *Constant:* $\alpha_{h_i} = 1 - \delta_{h_{i-1}} = K; \ \forall h$
- *Katz back-off (based on Good-Turing estimation)*

Outline

Outline

Example: Identifying Sentence Boundaries

Example

The president lives in Washington, D.C. The presidents met in Washington D.C. in 2010. Mr. Wayne is young. Mr. Wayne is a Ph.D. I got 98.5%! What?

Goal: given a text, identify tokens that end a sentence.

- Candidate characters: . ! ?
- Candidate tokens: tokens containing candidate characters
- Given a candidate token in a *context* decide whether it ends a sentence or not

Example: Sentence Boundaries

- Object to classify: punctuation sign + context
  x = <sign, prefix, suffix, previous, next>
- Assume access to *annotated* data:

y	sign	prefix	suffix	prev	next
no	.	D	C.	Washington,	The
yes	.	D.C		Washington,	The
no	.	Mr		2010.	Wayne

- Let's take a probabilistic approach:
  - $P(\texttt{yes} \mid x)$: conditional probability of x being end of sentence,
  - $P(\texttt{no} \mid x)$: conditional probability of x *not* being e.o.s.
  - $P(\texttt{yes} \mid x) + P(\texttt{no} \mid x) = 1$
  - Predict yes if $P(\texttt{yes} \mid x) > 0.5$
- How to model $P(\texttt{yes} \mid x)$ and $P(\texttt{no} \mid x)$?

Outline

Log-Linear Models

Log-Linear models take the form:

$$P(y \mid x; \mathbf{w}) = \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)} = \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{\sum_{y} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}$$

where

- $\mathbf{f}(x, y) \in \mathbb{R}^d$ is a feature vector representing a *context* x and a *label* y
- $\mathbf{w} \in \mathbb{R}^d$ is a vector containing the *parameters* of the model
- $\mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{i=1}^{d} w_i f_i(x, y)$ is a *score* for x and y
- $Z(x) = \sum_y \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)$ is a normalizer (sum of scores for all possible values for y); a.k.a *partition function*

Features, Indicator Features

- $\mathbf{f}(x, y) \in \mathbb{R}^d$ is a vector of d features encoding some information about x and y

$$\mathbf{f}(x, y) = (\ f_1(x, y), \ldots, f_k(x, y), \ldots, f_d(x, y)\ )$$

- What's in a feature $f_k(x, y)$?
  - *Anything* we can compute using x and y (and *suitable* for the task at hand)
  - *Anything* informative for (or against) x belonging to class y.
  - Usually, they are indicator features: binary-valued features looking at a (simple) property.

$$f_1(x, y) = \begin{cases} 1 & \text{if prefix}(x) = \textit{Mr} \text{ and } y = \text{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x, y) = \begin{cases} 1 & \text{if uppercase}(\text{next}(x)) \text{ and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

Log-linear Models: Name

- Let's take the log of the conditional probability:

$$\begin{aligned}
\log P(y \mid x; \mathbf{w}) &= \log \frac{\exp(\mathbf{w} \cdot \mathbf{f}(x, y))}{\sum_y \exp(\mathbf{w} \cdot \mathbf{f}(x, y))} \\
&= \mathbf{w} \cdot \mathbf{f}(x, y) - \log \sum_y \exp(\mathbf{w} \cdot \mathbf{f}(x, y)) \\
&= \mathbf{w} \cdot \mathbf{f}(x, y) - \log Z(x)
\end{aligned}$$

- $\log Z(x)$ is a constant for a fixed x
- In the log space, computations are linear

Log-linear Models: Making Predictions

- Given x, what y in $\{1, \ldots, L\}$ is most appropriate?

$$\text{best\_label}(x) = \underset{y \in \{1,\ldots,L\}}{\operatorname{argmax}} \; P(y \mid x; \mathbf{w})$$

Log-linear Models: Making Predictions

- Given x, what y in $\{1, \ldots, L\}$ is most appropriate?

$$\begin{aligned} \text{best\_label}(x) &= \underset{y \in \{1,\ldots,L\}}{\operatorname{argmax}} \; P(y \mid x; \mathbf{w}) \\ &= \underset{y \in \{1,\ldots,L\}}{\operatorname{argmax}} \; \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)} \end{aligned}$$

Log-linear Models: Making Predictions

- Given x, what y in $\{1, \ldots, L\}$ is most appropriate?

$$
\begin{aligned}
\text{best\_label}(x) &= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} P(y \mid x; \mathbf{w}) \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)} \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)
\end{aligned}
$$

Log-linear Models: Making Predictions

- Given x, what y in $\{1, \ldots, L\}$ is most appropriate?

$$\begin{aligned}
\text{best_label}(x) &= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} P(y \mid x; \mathbf{w}) \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)} \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right) \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \mathbf{w} \cdot \mathbf{f}(x, y)
\end{aligned}$$

Log-linear Models: Making Predictions

- Given x, what y in $\{1, \ldots, L\}$ is most appropriate?

$$\begin{aligned}
\text{best_label}(x) &= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} P(y \mid x; \mathbf{w}) \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)} \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right) \\
&= \operatorname*{argmax}_{y \in \{1,\ldots,L\}} \mathbf{w} \cdot \mathbf{f}(x, y)
\end{aligned}$$

- Predictions only require simple dot products (linear)
- No need to exponentiate!

Log-linear Models: Computing Probabilities

- Sometimes we will be interested in computing $P(y \mid x)$, not just the argmax.

  $P(y \mid x)$ can be used as a measure of confidence in the prediction, e.g.:

  $$P(\text{yes} \mid x) = 0.51 \quad \textit{vs.} \quad P(\text{yes} \mid x) = 0.99$$

- Since: $P(y \mid x; \mathbf{w}) = \dfrac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{Z(x)}$

  we need to compute: $Z(x) = \sum_{y=\{1,\ldots,L\}} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)$

- Feasible as long as L is not too large

Parameter Estimation in Log-linear Models

- How to estimate model parameters $\mathbf{w}$ given a training set:

$$\left\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\right\}$$

- We define the conditional log-likelihood of the data:

$$L(\mathbf{w}) = \frac{1}{m} \sum_{k=1}^{m} \log P(y^{(k)} | x^{(k)}; \mathbf{w})$$

- $L(\mathbf{w})$ measures how well $\mathbf{w}$ explains the data. A good value for $\mathbf{w}$ will give a high value for $P(y^{(k)} | x^{(k)}; \mathbf{w})$ for all $k = 1 \ldots m$.
- We want $\mathbf{w}$ that *maximizes* $L(\mathbf{w})$

Parameter Estimation in Log-Linear Models

Estimating model parameters is an optimization problem, aiming to find:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w} \in \mathbb{R}^d} L(\mathbf{w})$$

But low-frequency features may end up having large weights (i.e. overfitting).
Thus, we need a regularization factor that penalizes solutions with a large norm (similar to norm-minimization in SVM), redefining $L(\mathbf{w})$ as:

$$L(\mathbf{w}) = \frac{1}{m} \sum_{k=1}^{m} \log P(y^{(k)} | x^{(k)}; \mathbf{w}) - \frac{\lambda}{2} ||\mathbf{w}||^2$$

where λ is a parameter to control the trade-off between fitting the data and model complexity. Tuned experimentally.

Parameter Estimation in Log-Linear Models

So we want to find:

$$
\begin{aligned}
\mathbf{w}^* &= \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} L(\mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{m} \sum_{k=1}^{m} \log P(y^{(k)}|x^{(k)};\mathbf{w}) - \frac{\lambda}{2}\|\mathbf{w}\|^2 \right)
\end{aligned}
$$

- In general there is no analytical solution to this optimization.
- ... but it is a convex function ⇒ nummerical optimization iterative techniques, i.e. gradient-based optimization, may be used.
- Very fast algorithms exist (e.g. LBFGS).

Parameter Estimation in Log-Linear Models : Gradient step

- Initialize $\mathbf{w} = \mathbf{0}$
- Repeat
  - Compute gradient $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_d)$, where:

$$\delta_j = \frac{\partial L(\mathbf{w})}{\partial w_j} \quad \forall j = 1 \ldots d$$

  - Compute step size

$$\beta^* = \operatorname*{argmax}_{\beta \in \mathbb{R}} L(\mathbf{w} + \beta \boldsymbol{\delta})$$

  - Move $\mathbf{w}$ in the direction of the gradient

$$\mathbf{w} \leftarrow \mathbf{w} + \beta^* \boldsymbol{\delta}$$

- until convergence ($||\boldsymbol{\delta}|| < \epsilon$)

Log-linear Models: Computing the Gradient

$$
\begin{aligned}
\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}_j} = & \frac{1}{m} \sum_{k=1}^{m} f_j(x^{(k)}, y^{(k)}) \\
& - \sum_{k=1}^{m} \sum_{y \in \{1,\ldots,L\}} P(y|x^{(k)}; \mathbf{w}) \; f_j(x^{(k)}, y) \\
& - \lambda w_j
\end{aligned}
$$

- First term: observed mean feature value
- Second term: expected feature value under current $\mathbf{w}$
- In the optimal, observed = expected

Outline

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

No observations (no constraints)

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.07	0.07	0.07	0.07	0.07	0.07	0.07	
on	0.07	0.07	0.07	0.07	0.07	0.07	0.07	
total								1.0

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

Observations (constraints):

$$p(\text{en} \vee \text{à}) = 0.6$$

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	0.15	0.15	0.04	0.04	0.04	0.04	
on	0.04	0.15	0.15	0.04	0.04	0.04	0.04	
total		0.6 (en + à)						1.0

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

Observations (constraints):

$$p(\text{en} \vee \text{à}) = 0.6; \quad p((\text{en} \vee \text{à}) \wedge \text{in}) = 0.4$$

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.04	0.20	0.20	0.04	0.04	0.04	0.04	
on	0.04	0.10	0.10	0.04	0.04	0.04	0.04	
total		0.6 (en + à)						1.0

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

Observations (constraints):

$$p(\text{en} \vee \text{à}) = 0.6; \quad p((\text{en} \vee \text{à}) \wedge \text{in}) = 0.4; \quad p(\text{in}) = 0.5$$

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.02	0.20	0.20	0.02	0.02	0.02	0.02	**0.5**
on	0.06	0.10	0.10	0.06	0.06	0.06	0.06	
total		$\underbrace{\quad\quad}_{0.6}$						1.0

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

Observations (constraints):

$p(\text{en} \vee \text{à}) = 0.6; \quad p((\text{en} \vee \text{à}) \wedge \text{in}) = 0.4; \quad p(\text{in}) = 0.5; \quad p(\text{sur}) = 0.1$

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.02	0.20	0.20	0.02	0.02	0.02	0.02	0.5
on	0.06	0.10	0.10	0.06	0.06	0.06	0.06	
total		0.6 (en + à)		**0.1**				1.0

Example

Example: Maximum entropy model for translating English prepositions *in* and *on* to French.

Observations (constraints):

$$p(\text{en} \vee \text{à}) = 0.6; \quad p((\text{en} \vee \text{à}) \wedge \text{in}) = 0.4; \quad p(\text{in}) = 0.5; \quad p(\text{sur}) = 0.1$$

$P(x, y)$	dans	en	à	sur	au-cours-de	pendant	selon	
in	0.02	0.20	0.20	0.02	0.02	0.02	0.02	0.5
on	0.06	0.10	0.10	0.06	0.06	0.06	0.06	
total				**0.1**				1.0
		0.6 (en + à)						

Not so easy...

Maximum entropy Models

MaxEnt models: **dual formulation** of **Log-Linear models**.

ME principles:

- Do not assume anything about non-observed events.
- Find the most uniform (maximum entropy, less informed) probability distribution that matches the observations.

ME Modeling

- Observed facts are constraints for the desired model p.
- Constraints take the form of feature functions:

$$f_i : \varepsilon \rightarrow \{0, 1\}$$

- The desired model p must satisfy the constraints:
  The expectation predicted by model p for any feature f_i must match the observed expectation for f_i
  i.e.:

$$\begin{aligned} E_p(f_i) &= E_{\tilde{p}}(f_i) \quad \forall i \\ \sum_{x \in \varepsilon} p(x) f_i(x) &= \sum_{x \in \varepsilon} \tilde{p}(x) f_i(x) \quad \forall i \end{aligned}$$

Probability Model

- There is an infinite set P of probability models consistent with observations:

$$P = \{p \mid E_p(f_i) = E_{\tilde{p}}(f_i), \quad \forall i\}$$

- Maximum entropy model

$$\begin{aligned} p^* &= \operatorname*{argmax}_{p \in P} H(p) \\ &= \operatorname*{argmax}_{p \in P} \left(- \sum_{x \in \varepsilon} p(x) \log p(x) \right) \end{aligned}$$

Parameter estimation

- ME models are exponential models, same as log-linear models:

$$P(y \mid x) = \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}{\sum_{y \in L} \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)}$$

- Each model parameter weights the influence of a feature.
- Same convex optimization algorithms are used (e.g. **LM-BFGS**,)
- Optimized cost function is different, but optimum corresponds to the same $\mathbf{w}$ than a log-linear model.

Outline

Example: Identifying Sentence Boundaries

Goal: given a text, identify tokens that end a sentence.

> *The president lives in Washington, D.C. The presidents met in Washington D.C. in 2010. Mr. Wayne is young. Mr. Wayne is a Ph.D. I got 98.5%! What?*

- Candidate characters: . ! ?
- Candidate tokens: tokens containing candidate characters (e.g. *D.C.*, *2010.*, *young.*, *Mr.*, *Ph.D.*, *98.5%!*, *What?*)
- Given a candidate character in a *context* decide whether the token ends a sentence.

Identifying Sentence Boundaries: Formulation

- Object to classify: punctuation sign + context
  x = `<sign, prefix, suffix, previous, next>`
- Class: $y \in \{\texttt{yes}, \texttt{no}\}$
- Probabilistic approach:
  - **Goal**: estimate $P(\texttt{yes} \mid x)$ and $P(\texttt{no} \mid x)$
  - $P(\texttt{yes} \mid x) + P(\texttt{no} \mid x) = 1$
  - Predict yes if $P(\texttt{yes} \mid x) > 0.5$

Identifying Sentence Boundaries: Data set

- Assume access to *annotated* data:

  The president lives in Washington, D.C. The presidents met in Washington D.C. in 2010. Mr. Wayne is young. Mr. Wayne is a Ph.D. I got 98.5%! What?

- Data set

y	sign	prefix	suffix	prev	next
no	.	D	C.	Washington,	The
yes	.	D.C		Washington,	The
no	.	D	C.	Washington	in
no	.	D.C		Washington	in
yes	.	2010		in	Mr.
no	.	Mr		2010.	Wayne
yes	.	young		is	Mr.
no	.	Mr		young.	Wayne
no	.	Ph	D.	a	I
yes	.	Ph.D		a	I
no	.	98	5%!	got	What?
yes	!	98.5%!		got	What?
yes	?	What		98.5%!	<eof>

Identifying Sentence Boundaries: Feature templates

- Feature templates:
  1. The `prefix`: part of the token before the `sign`.
  2. The `suffix`: part of the token after the `sign`.
  3. The `previous` token.
  4. The `next` token.
  5. Whether `prefix` or `suffix` are in ABBREVIATIONS
  6. Whether `previous` or `next` are in ABBREVIATIONS
- ABBREVIATIONS: list of all tokens in training data that contain `sign` and are *not* sentence boundaries.
- Actual features are generated by applying *each template* to *each training example*.

Identifying Sentence Boundaries: Feature generation

Feature Templates

1. The `prefix`
2. The `suffix`
3. The `previous` token
4. The `next` token
5. Whether `prefix` or `suffix` are in Abbreviations
6. Whether `previous` or `next` are in Abbreviations

Training Data Example 1

$y =$ `no`; $x =$ `<sign=. pref=Mr suff= prev=2010. next=Wayne>`

Generated Features

$$f_{1\_Mr\_no}(x,y) = \begin{cases} 1 & \textbf{if } \text{pref}(x) = \texttt{Mr} \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{2\_null\_no}(x,y) = \begin{cases} 1 & \textbf{if } \text{suff}(x) = \texttt{NULL} \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{3\_2010.\_no}(x,y) = \begin{cases} 1 & \textbf{if } \text{prev}(x) = \texttt{2010.} \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{4\_Wayne\_no}(x,y) = \begin{cases} 1 & \textbf{if } \text{next}(x) = \texttt{Wayne} \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{5\_no}(x,y) = \begin{cases} 1 & \textbf{if } (\text{abbr}(\text{pref}(x)) \\ & \quad \textbf{or } \text{abbr}(\text{suff}(x))) \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{6\_no}(x,y) = \begin{cases} 1 & \textbf{if } (\text{abbr}(\text{prev}(x)) \\ & \quad \textbf{or } \text{abbr}(\text{next}(x))) \\ & \quad \textbf{and } y = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

Identifying Sentence Boundaries: Feature generation

FEATURE TEMPLATES

1. The `prefix`
2. The `suffix`
3. The `previous` token
4. The `next` token
5. Whether `prefix` or `suffix` are in ABBREVIATIONS
6. Whether `previous` or `next` are in ABBREVIATIONS

TRAINING DATA EXAMPLE 2

$y =$ `yes` $x =$ `<punc=. pref=D.C suff= prev=, next=The>`

GENERATED FEATURES

$$f_{1\_D.C\_yes}(x,y) = \begin{cases} 1 & \text{if } \mathrm{pref}(x) = \texttt{D.C} \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{2\_null\_yes}(x,y) = \begin{cases} 1 & \text{if } \mathrm{suff}(x) = \texttt{NULL} \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{3\_,\_yes}(x,y) = \begin{cases} 1 & \text{if } \mathrm{prev}(x) = \texttt{,} \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{4\_The\_yes}(x,y) = \begin{cases} 1 & \text{if } \mathrm{next}(x) = \texttt{The} \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{5\_yes}(x,y) = \begin{cases} 1 & \text{if } (\mathrm{abbr}(\mathrm{pref}(x)) \\ & \quad\quad \textbf{or } \mathrm{abbr}(\mathrm{suff}(x))) \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{6\_yes}(x,y) = \begin{cases} 1 & \text{if } (\mathrm{abbr}(\mathrm{prev}(x)) \\ & \quad\quad \textbf{or } \mathrm{abbr}(\mathrm{next}(x))) \\ & \quad \textbf{and } y = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

Example: Text Categorization

Goal: given a text, classify it according to a set of predefined classes

> *British hurdler Sarah Claxton is confident she can win her first major medal at next month's European Indoor Championships in Madrid.*

- Candidate classes: `business`, `entertainment`, `politics`, `sport`, `tech`.
- Given a document decide which category (or categories) it belongs to.

Text Categorization: Formulation

- Object to classify: document (as a set of words)
  $x = $ `<word_1, word_2, ..., word_n>`
- Class: $y \in L = \{\texttt{biz}, \texttt{ent}, \texttt{pol}, \texttt{spo}, \texttt{tech}\}$
- Probabilistic approach:
  - **Goal**: estimate $P(y \mid x) \quad \forall y \in L$
  - $\sum_{y \in L} P(y \mid x) = 1$
  - Predict as output class:
    - Class with highest probability: $\text{argmax}_y P(y \mid x)$
    - Any class with probablity over a threshold: $\{y \mid P(y \mid x) > k\}$

Text Categorization: Data set

- Assume access to *annotated* data:

spo	British hurdler Sarah Claxton is confident she can win her first major medal at next month's European Indoor Championships in Madrid.
pol	The Labour Party will hold its 2006 autumn conference in Manchester and not Blackpool, it has been confirmed.
tech	Microsoft has warned PC users to update their systems with the latest security fixes for flaws in Windows programs.

Text Categorization: Feature templates

- Feature templates:
  1. The ocurrence of a `word` in the document
- Actual features are generated by applying the *template* to *each training example*.

Text Categorization: Feature generation

FEATURE TEMPLATES

1. A word occurring in the document.

TRAINING DATA EXAMPLE 1

$y = \text{spo}$

$x = \{$British hurdler Sarah Claxton confident win first major medal next month European Indoor Championships Madrid$\}$

GENERATED FEATURES

$$f_{1\_British\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{British} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_medal\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{medal} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_hurdler\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{hurdler} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_next\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{next} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

...

...

$$f_{1\_win\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{win} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_Madrid\_spo}(x, y) = \begin{cases} 1 & \textbf{if } \texttt{Madrid} \in x \\ & \quad \textbf{and } y = \text{spo} \\ 0 & \text{otherwise} \end{cases}$$

...

Text Categorization: Feature generation

Feature Templates

1. A `word` occurring in the document.

Training Data Example 2

$y = \texttt{tech}$

$x = \{$Microsoft warned PC users update systems latest security fixes flaws Windows programs$\}$

Generated Features

$$f_{1\_\text{Microsoft\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{Microsoft} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_\text{latest\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{latest} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_\text{warned\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{warned} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_\text{security\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{security} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

...

...

$$f_{1\_\text{systems\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{systems} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{1\_\text{programs\_tech}}(x,y) = \begin{cases} 1 & \textbf{if } \texttt{programs} \in x \textbf{ and } y = \texttt{tech} \\ 0 & \text{otherwise} \end{cases}$$

...

Outline

Log-linear Models Summary

- Advantages
  - Teoretically well founded
  - Enables combination of random context features
  - Better probabilistic models than MLE (no smoothing needed)
  - General approach (features, events and classes)
- Disadvantages
  - Implicit probabilistic model (joint or conditional probability distribution obtained from model parameters).