Master in Artificial Intelligence

Advanced Human Language Technologies

Statistical Models of Language

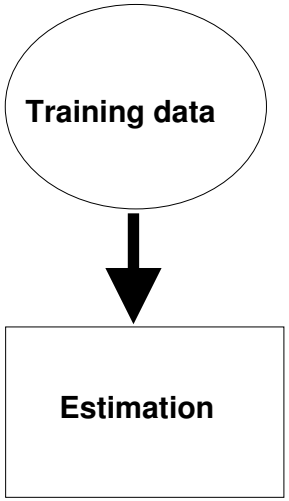
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

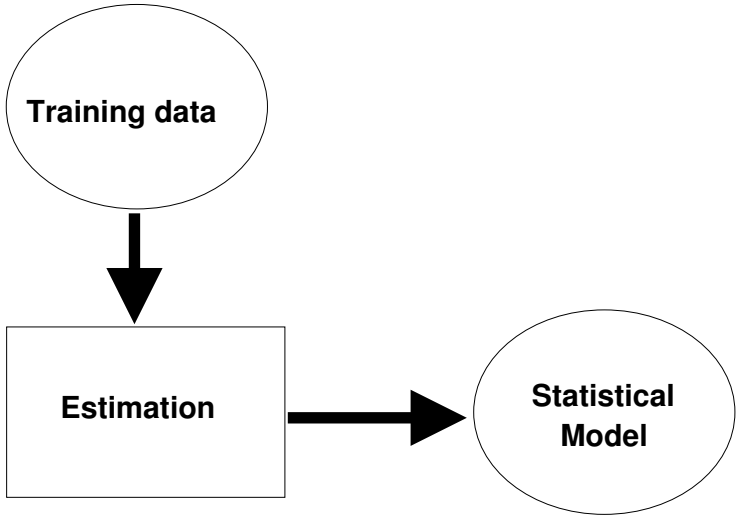
Outline

We model to make predictions

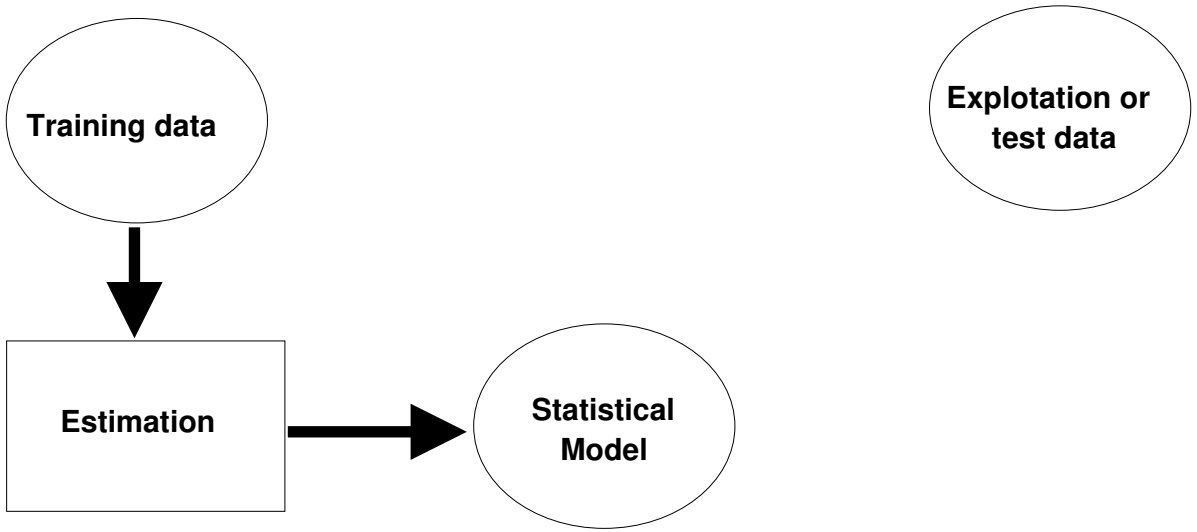
We model to make predictions



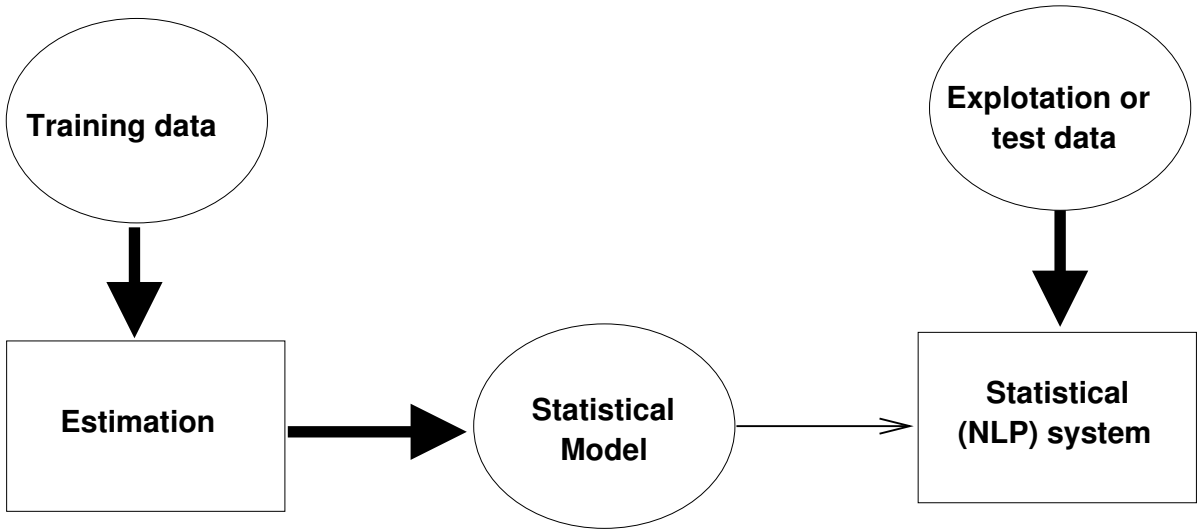
We model to make predictions



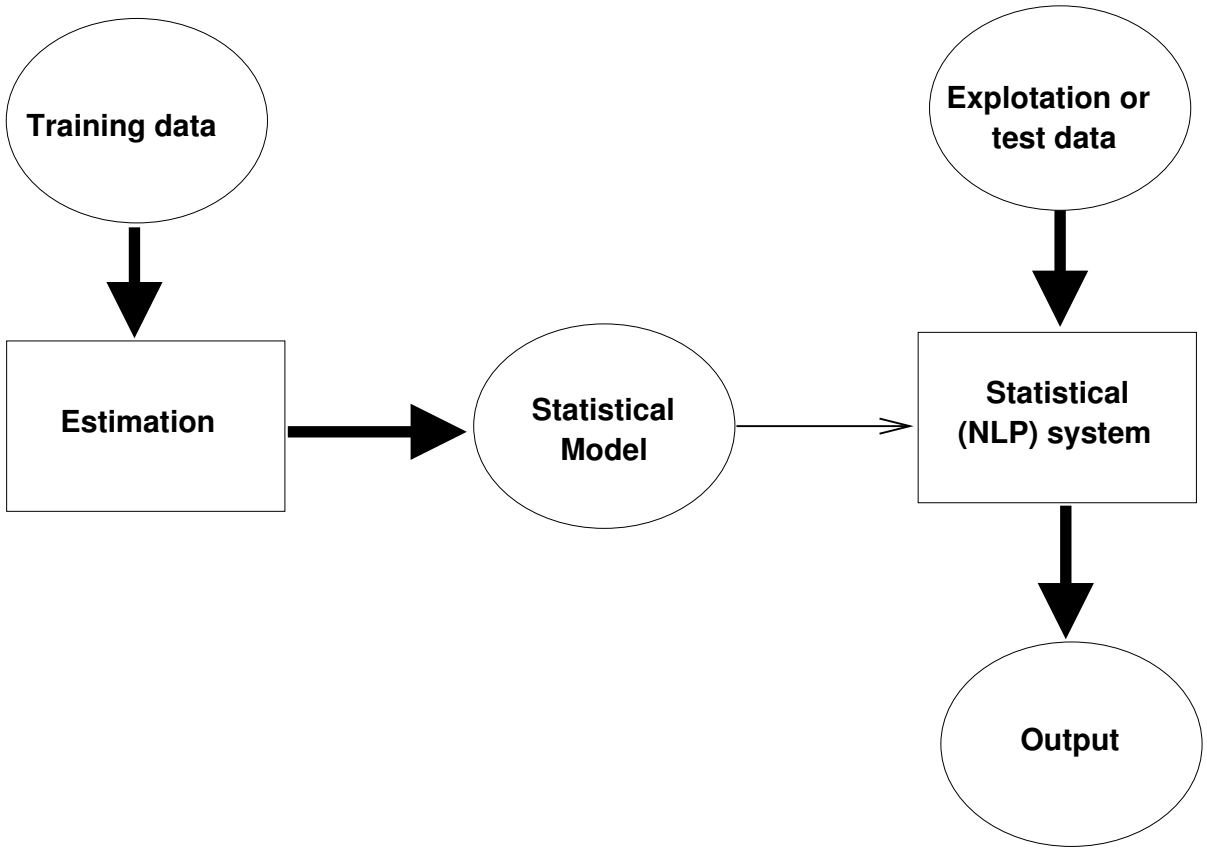
We model to make predictions



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).

Similarity Models

- Objects represented as feature-vectors, feature-sets, distribution-vectors, ...
- Used to group objects (clustering, data analysis, pattern discovery, ...)
- If classified objects are available, similarity may be used as a prediction (example-based ML techniques).
- Example: Document representation
  - Documents are represented as vectors in a high dimensional $\mathbb{R}^n$ space.
  - Dimensions are word forms, lemmas, NEs, n-grams, ...
  - Values may be either binary or real-valued (count, frequency, ...)
  - Vector-space algebra and metrics can be used

Prediction Models

- Estimation: Using data to infer information about distributions
  - Parametric / non-parametric estimation
  - Finding good estimators: MLE, MEE, ...
  - Explicit / implicit models
- Classification: Predictions based on past behaviour
  - Predict most likely target given classification features (implies independence assumptions!)
  - Granularity of equivalence classes (bins): discrimination power *vs.* statistical reliability
- In general, ML models estimate (i.e. *learn*) conditional probability distributions $\mathrm{P}(\text{target}|\text{features})$
- Many NLP tasks require a posterior search step to find the best combination of predictions.

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 $\mathbf{i}$

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

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.

$P(a, b)$	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

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

Observations:

$p(\text{en} \vee \text{à}) = 0.6$

$P(a, b)$	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		$\underbrace{\quad\quad}_{0.6}$						1.0

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

Observations:

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

$P(a, b)$	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		$\underbrace{\quad\quad}_{\mathbf{0.6}}$						1.0

Finding good estimators: MEE

Maximum Entropy Estimation (MEE)

- Choose the alternative that maximizes the entropy of the obtained distribution, maintaning the observed probabilities.

Observations:

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

$P(a, b)$	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		$\underbrace{\quad\quad}_{\mathbf{0.6}}$						1.0

Outline

Outline

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

Notation

- $\mathrm{C}(w_1 \ldots w_n)$: Observed occurrence count for n-gram $w_1 \ldots w_n$.
- N: Number of observed n-gram occurrences

$$\mathrm{N} = \sum_{w_1 \ldots w_n} \mathrm{C}(w_1 \ldots w_n)$$

- N_k: Number of classes (n-grams) observed k times.
- B: Number of equivalence classes or bins (number of potentially observable n-grams).

Smoothing 1 - Adding Counts

- **Laplace's Law** (adding one)

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

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

- **Lidstone's Law**

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

  - Usually $\lambda = 0.5$, *Expected Likelihood Estimation*.
  - Equivalent to linear interpolation between MLE and uniform prior, with $\mu = N/(N + B\lambda)$,

$$P_{LID}(w_1 \dots w_n) = \mu \frac{C(w_1 \dots w_n)}{N} + (1 - \mu)\frac{1}{B}$$

Smoothing 2 - Discounting Counts

- **Absolute Discounting**

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

- **Linear Discounting**

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

Combining Estimators

- **Simple 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**

$$P_{BO}(w_i \mid h) = \begin{cases} (1 - \alpha_h) \dfrac{C(h, w_i)}{C(h)} & \text{if } C(h, w_i) > k \\ \delta_{h'} P_{BO}(w_i \mid h') & \text{otherwise} \end{cases}$$

(where $h = w_{i-n+1} \ldots w_{i-1}$, $h' = w_{i-n+2} \ldots w_{i-1}$)

Different options to determine α_h and $\delta_{h'}$ (e.g. $\alpha_h = \delta_{h'} \quad \forall h$)

Outline

Outline

MEM Overview

- Maximum Entropy: alternative estimation technique.
- Able to deal with different kinds of evidence
- ME principle:
  - Do not assume anything about non-observed events.
  - Find the most uniform (maximum entropy, less informed) probability distribution that matches the observations.
- Example:

$p(x,y)$	0	1	
a	?	?	
b	?	?	
total	0.6		1.0

Observations

$p(x,y)$	0	1	
a	0.5	0.1	
b	0.1	0.3	
total	0.6		1.0

One possible $p(x,y)$

$p(x,y)$	0	1	
a	0.3	0.2	
b	0.3	0.2	
total	0.6		1.0

Max.Entropy $p(x,y)$

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.:

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

Example

- Example:

$$\varepsilon = \{a, b\} \times \{0, 1\}$$

$p(x, y)$	0	1	
a	?	?	
b	?	?	
total	0.6		1.0

- Observed fact: $p(a, 0) + p(b, 0) = 0.6$
- Encoded as a constraint: $E_p(f_1) = 0.6$ where:
  - $f_1(x, y) = \begin{cases} 1 & \text{if } y = 0 \\ 0 & \text{otherwise} \end{cases}$
  - $E_p(f_1) = \sum_{(x,y) \in \{a,b\} \times \{0,1\}} p(x, y) f_1(x, y)$

Outline

Probability Model

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

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

- Maximum entropy model

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

Conditional Probability Model

- For NLP applications, we are usually interested in conditional distributions $P(Y|X)$, thus, the ME model is

$$p^* = \underset{p \in P}{\operatorname{argmax}}\, H(p) = \underset{p \in P}{\operatorname{argmax}}\, H(Y \mid X)$$

where:

$$\begin{aligned} H(Y \mid X) &= \sum_{x \in X} p(x) H(Y \mid X = x) \\ &= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y \mid x) \log p(y \mid x) \\ &= -\sum_{x \in X, y \in Y} p(x, y) \log p(y \mid x) \\ &= -\sum_{x \in X, y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)} \end{aligned}$$

Parameter Estimation

Example: Maximum entropy model for translating *in* to French

- No constraints

$P(x)$	dans	en	à	au-cours-de	pendant	
	0.2	0.2	0.2	0.2	0.2	
total						1.0

- With constraint $p(dans) + p(en) = 0.3$

$P(x)$	dans	en	à	au-cours-de	pendant	
	0.15	0.15	0.233	0.233	0.233	
total	**0.3**					1.0

- With constraints

  $p(dans) + p(en) = 0.3; \quad p(en) + p(\grave{a}) = 0.5$

...Not so easy !

Parameter estimation

- Exponential models

$$p(y \mid x) = \frac{1}{Z(x)} \prod_{j=1}^{k} \alpha_j^{f_j(x,y)} \qquad \alpha_j > 0, \quad Z(x) = \sum_{y} \prod_{i=1}^{k} \alpha_i^{f_i(x,y)}$$

- Can also be formuled as

$$p(y \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{j=1}^{k} \lambda_j f_j(x,y)\right) \qquad \text{(i.e. } \lambda_i = \ln \alpha_i\text{)}$$

- Each model parameter weights the influence of a feature.
- Optimal parameters can be computed with:
  - Generalized Iterative Scaling (GIS) [Darroch & Ratcliff 72]
  - Improved Iterative Scaling (IIS) [Della Pietra et al. 96]
  - Limited Memory BFGS (**LM-BFGS**) [Malouf 03]

Example: Text Categorization

- Probabilistic model over $W \times C$ (Words $\times$ Categories).
  A document is a set of words: $d = (w_1, w_2 \ldots w_N)$.
  Each combination $w, c \in W \times C$ is a feature:

$$f_{w,c}(d, c') = \begin{cases} \frac{N(w,d)}{N(d)} & \text{if } c = c' \\ 0 & \text{otherwise} \end{cases}$$

- Disambiguation: Select class with highest $P(c \mid d)$

$$P(c \mid d) = \frac{1}{Z(d)} \exp\left(\sum_i \lambda_i f_i(d, c)\right)$$

MEM 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).

ME Models are a particular case of **Log-Linear models**

Outline

Log-Linear Models

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

where

- $\mathbf{f}(x, y)$ is a feature vector representing x and y
- $\mathbf{w}$ are the parameters of the model
- $\mathbf{w} \cdot \mathbf{f}(x, y)$ is a score for x and y
- $Z(x) = \sum_y \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)$ is a normalizer (sums over all possible values y for x); it's sometimes called the *partition function*

Features, Indicator Features

- $\mathbf{f}(x, y)$ is a vector of d features representing x and y

$$(\mathbf{f}_1(x, y), \ldots, \mathbf{f}_j(x, y), \ldots, \mathbf{f}_d(x, y))$$

- What's in a feature $\mathbf{f}_j(x, y)$?
  - Anything we can compute using x and y
  - Anything that is informative for (or against) x belonging to class y
  - Indicator features: binary-valued features looking at a single simple property

$$\mathbf{f}_j(c, b) = \begin{cases} 1 & \text{if prefix}(c) = Mr \text{ and } b = \text{no} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_k(c, b) = \begin{cases} 1 & \text{if uppercase}(\text{next}(c)) \text{ and } b = \text{yes} \\ 0 & \text{otherwise} \end{cases}$$

Features, Parameters, Inner Products

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

- $\mathbf{f}(x, y) \in \mathbb{R}^d$ is a feature vector with d features
- $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector, with d parameters

- Inner products (*a.k.a.* dot products)

$$\mathbf{w} \cdot \mathbf{f}(x, y) = \sum_{i=1}^{d} \mathbf{w}_i \, \mathbf{f}_i(x, y)$$

Log-linear Models

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

where

- $\mathbf{f}(x, y)$ is a feature vector representing x and y
  - Arbitrary features of x and y are allowed
  - They are provided for the application in turn
- $\mathbf{w}$ are the parameters of the model

- Two problems:
  - How to make predictions using $P(y \mid x)$
  - How to estimate the parameters $\mathbf{w}$?

Log-linear Models: Name

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

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

- Partition function: $Z(x) = \sum_y \exp\left(\mathbf{w} \cdot \mathbf{f}(x, y)\right)$
- $\log Z(x)$ 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?

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

Log-linear Models: Making Predictions

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

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

Log-linear Models: Making Predictions

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

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

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

Log-linear Models: Computing Probabilities

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

- Sometimes we will be interested in computing $P(y \mid x)$
  - It can be used as a measure of confidence, e.g.

    $P(\texttt{yes} \mid c) = 0.51$ *versus*

    $P(\texttt{yes} \mid c) = 0.99$
- We need to compute:

$$Z(x) = \sum_{y=\{1,\ldots,L\}} \exp(\mathbf{w} \cdot \mathbf{f}(x, y))$$

- Fast 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:

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

- Let's 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

- We pose it as an optimization problem
- Find:

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

where

- But low-frequency features may end up having large weights (i.e. overfitting)
- We need a regularization factor that penalizes solutions with a large norm (similar to norm-minimization in SVM):

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

- 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 $\Rightarrow$ We use iterative techniques, i.e. gradient-based optimization
- Very fast algorithms exist (e.g. LBFGS)

Parameter Estimation in Log-Linear Models : Gradient step

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

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

  - Compute step size

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

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

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

- until convergence ($||\delta|| < \epsilon$)

Log-linear Models: Computing the Gradient

$$\frac{\partial L'(\mathbf{w})}{\partial \mathbf{w}_j} = \frac{1}{m} \sum_{k=1}^{m} \mathbf{f}_j(x^{(k)}, y^{(k)}) - \sum_{k=1}^{m} \sum_{y \in \{1,\ldots,L\}} P(y|x^{(k)}; \mathbf{w})\ \mathbf{f}_j(x^{(k)}, y) - \lambda \mathbf{w}_j$$

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

Maximum log-likelihood log-linear models correspond to Maximum Entropy models

Example: Identifying Sentence Boundaries

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

- Candidate: punctuation sign + context

  `c = < sign, prefix, suffix, previous, next >`

- Assume access to *annotated* data:

b	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:
  - $\mathrm{P}(\texttt{yes} \mid \mathrm{c})$: conditional probability of c being end of sentence
  - $\mathrm{P}(\texttt{no} \mid \mathrm{c})$: conditional probability of c *not* being e.o.s.
  - Obviously, $\mathrm{P}(\texttt{yes} \mid \mathrm{c}) + \mathrm{P}(\texttt{no} \mid \mathrm{c}) = 1$
  - Predict yes if $\mathrm{P}(\texttt{yes} \mid \mathrm{c}) > 0.5$
- How to model $\mathrm{P}(\texttt{yes} \mid \mathrm{c})$ and $\mathrm{P}(\texttt{no} \mid \mathrm{c})$?

Example system: Identifying Sentence Boundaries

(Reynar and Ratnaparkhi '97)

- Candidate: punctuation sign + context

  `c = < sign, prefix, suffix, previous, next >`

- **Goal**: estimate $\mathrm{P}(\texttt{yes} \mid \texttt{c})$ and $\mathrm{P}(\texttt{no} \mid \texttt{c})$
- Feature templates:
  1. The `prefix`
  2. The `suffix`
  3. The word `previous`
  4. The word `next`
  5. Whether `prefix` or `suffix` are in ABBREVIATIONS
     - ABBREVIATIONS: list of all training tokens that contain a . and are *not* sentence boundaries
  6. Whether `previous` or `next` are in ABBREVIATIONS
- Actual features are generated by applying each template to each training example

Example system: Identifying Sentence Boundaries

(Reynar and Ratnaparkhi '97)

Feature Templates

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

`< b=no punc=. pref=Mr suff= prev=2010. next=Wayne >`

Generated Features

$$f_1(c, b) = \begin{cases} 1 & \textbf{if } \text{pref}(c) = \texttt{Mr} \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c, b) = \begin{cases} 1 & \textbf{if } \text{suff}(c) = \texttt{NULL} \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c, b) = \begin{cases} 1 & \textbf{if } \text{prev}(c) = \texttt{2010.} \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c, b) = \begin{cases} 1 & \textbf{if } \text{next}(c) = \texttt{Wayne} \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5(c, b) = \begin{cases} 1 & \textbf{if } (\text{abbr}(\text{pref}(c)) \textbf{ or } \text{abbr}(\text{suff}(c))) \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(c, b) = \begin{cases} 1 & \textbf{if } (\text{abbr}(\text{prev}(c)) \textbf{ or } \text{abbr}(\text{next}(c))) \\ & \quad \textbf{and } b = \texttt{no} \\ 0 & \text{otherwise} \end{cases}$$

Example System: Identifying Sentence Boundaries

(Reynar and Ratnaparkhi '97)

training sentences	test accuracy
500	96.5%
1000	97.3%
2000	97.3%
4000	97.6%
8000	97.6%
16000	97.8%
39441	98.0%

- Corpus: Wall Street Journal, English