6.864 (Fall 2007): Lecture 6
Log-Linear Models

Michael Collins, MIT

The Language Modeling Problem

- w_i is the i'th word in a document
- Estimate a distribution $P(w_i|w_1, w_2, \ldots w_{i-1})$ given previous "history" $w_1, \ldots, w_{i-1}$.
- E.g., $w_1, \ldots, w_{i-1} =$

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

Trigram Models

- Estimate a distribution $P(w_i|w_1, w_2, \ldots w_{i-1})$ given previous "history" $w_1, \ldots, w_{i-1} =$

Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- **Trigram estimates:**

$$\begin{aligned} P(\text{model}|w_1, \ldots w_{i-1}) \quad = \quad & \lambda_1 P_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\ & \lambda_2 P_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\ & \lambda_3 P_{ML}(\text{model}) \end{aligned}$$

where $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, $P_{ML}(y|x) = \frac{Count(x,y)}{Count(x)}$

Trigram Models

$$\begin{aligned} P(\text{model}|w_1, \ldots w_{i-1}) \quad = \quad & \lambda_1 P_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) + \\ & \lambda_2 P_{ML}(\text{model}|w_{i-1} = \text{statistical}) + \\ & \lambda_3 P_{ML}(\text{model}) \end{aligned}$$

- Makes use of only bigram, trigram, unigram estimates
- Many other "features" of $w_1, \ldots, w_{i-1}$ may be useful, e.g.,:

$$\begin{array}{lcl} P_{ML}(\text{model} & | & w_{i-2} = \text{any}) \\ P_{ML}(\text{model} & | & w_{i-1} \text{ is an adjective}) \\ P_{ML}(\text{model} & | & w_{i-1} \text{ ends in "ical"}) \\ P_{ML}(\text{model} & | & author = \text{Chomsky}) \\ P_{ML}(\text{model} & | & \text{"model" does not occur somewhere in } w_1, \ldots w_{i-1}) \\ P_{ML}(\text{model} & | & \text{"grammatical" occurs somewhere in } w_1, \ldots w_{i-1}) \end{array}$$

A Naive Approach

$P(\text{model}|w_1, \ldots w_{i-1}) =$
$\lambda_1 P_{ML}(\text{model}|w_{i-2} = \text{any}, w_{i-1} = \text{statistical}) +$
$\lambda_2 P_{ML}(\text{model}|w_{i-1} = \text{statistical}) +$
$\lambda_3 P_{ML}(\text{model}) +$
$\lambda_4 P_{ML}(\text{model}|w_{i-2} = \text{any}) +$
$\lambda_5 P_{ML}(\text{model}|w_{i-1} \text{ is an adjective}) +$
$\lambda_6 P_{ML}(\text{model}|w_{i-1} \text{ ends in "ical"}) +$
$\lambda_7 P_{ML}(\text{model}|author = \text{Chomsky}) +$
$\lambda_8 P_{ML}(\text{model}|\text{"model" does not occur somewhere in } w_1, \ldots w_{i-1}) +$
$\lambda_9 P_{ML}(\text{model}|\text{"grammatical" occurs somewhere in } w_1, \ldots w_{i-1})$

This quickly becomes very unwieldy...

A Second Example: Part-of-Speech Tagging

INPUT:
Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT:

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.

N = Noun
V = Verb
P = Preposition
Adv = Adverb
Adj = Adjective
. . .

A Second Example: Part-of-Speech Tagging

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- There are many possible tags in the position ??
{NN, NNS, Vt, Vi, IN, DT, . . . }

- The task: model the distribution

$$P(t_i|t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

A Second Example: Part-of-Speech Tagging

Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/?? from which Spain expanded its empire into the rest of the Western Hemisphere .

- The task: model the distribution

$$P(t_i|t_1, \ldots, t_{i-1}, w_1 \ldots w_n)$$

where t_i is the i'th tag in the sequence, w_i is the i'th word

- Again: many "features" of $t_1, \ldots, t_{i-1}, w_1 \ldots w_n$ may be relevant

$$P_{ML}(\text{NN} \mid w_i = \text{base})$$
$$P_{ML}(\text{NN} \mid t_{i-1} \text{ is } \texttt{JJ})$$
$$P_{ML}(\text{NN} \mid w_i \text{ ends in "e"})$$
$$P_{ML}(\text{NN} \mid w_i \text{ ends in "se"})$$
$$P_{ML}(\text{NN} \mid w_{i-1} \text{ is "important"})$$
$$P_{ML}(\text{NN} \mid w_{i+1} \text{ is "from"})$$

Overview

- Log-linear models
- The maximum-entropy property
- Smoothing, feature selection etc. in log-linear models

The General Problem

- We have some **input domain** $\mathcal{X}$
- Have a finite **label set** $\mathcal{Y}$
- Aim is to provide a **conditional probability** $P(y \mid x)$ for any x, y where $x \in \mathcal{X}$, $y \in \mathcal{Y}$

Language Modeling

- x is a "history" $w_1, w_2, \ldots w_{i-1}$, e.g.,

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- y is an "outcome" w_i

Feature Vector Representations

- Aim is to provide a conditional probability $P(y \mid x)$ for "decision" y given "history" x
- A **feature** is a function $f(x, y) \in \mathbb{R}$
  (Often **binary features** or **indicator functions** $f(x, y) \in \{0, 1\}$).
- Say we have m features f_k for $k = 1 \ldots m$
  $\Rightarrow$ A **feature vector** $\mathbf{f}(x, y) \in \mathbb{R}^m$ for any x, y

Language Modeling

- x is a "history" $w_1, w_2, \ldots w_{i-1}$, e.g.,

  Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- y is an "outcome" w_i

- Example features:

$$f_1(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model} \text{ and } w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ is an adjective} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-1} \text{ ends in "ical"} \\ 0 & \text{otherwise} \end{cases}$$

$$f_7(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ author = Chomsky} \\ 0 & \text{otherwise} \end{cases}$$

$$f_8(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ "model" is not in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_9(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model}, \text{ "grammatical" is in } w_1, \ldots w_{i-1} \\ 0 & \text{otherwise} \end{cases}$$

Defining Features in Practice

- We had the following "trigram" feature:

$$f_3(x,y) = \begin{cases} 1 & \text{if } y = \texttt{model},\ w_{i-2} = \texttt{any},\ w_{i-1} = \texttt{statistical} \\ 0 & \text{otherwise} \end{cases}$$

- In practice, we would probably introduce one trigram feature for every trigram seen in the training data: i.e., for all trigrams (u, v, w) seen in training data, create a feature

$$f_{N(u,v,w)}(x,y) = \begin{cases} 1 & \text{if } y = w,\ w_{i-2} = u,\ w_{i-1} = v \\ 0 & \text{otherwise} \end{cases}$$

  where $N(u, v, w)$ is a function that maps each (u, v, w) trigram to a different integer

The POS-Tagging Example

- Each x is a "history" of the form $\langle t_1, t_2, \ldots, t_{i-1}, w_1 \ldots w_n, i \rangle$
- Each y is a POS tag, such as $NN, NNS, Vt, Vi, IN, DT, \ldots$
- We have m features $f_k(x, y)$ for $k = 1 \ldots m$

For example:

$$f_1(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } y = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$\ldots$$

The Full Set of Features in [Ratnaparkhi 96]

- Word/tag features for all word/tag pairs, e.g.,

$$f_{100}(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } y = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

- Spelling features for all prefixes/suffixes of length ≤ 4, e.g.,

$$f_{101}(x, y) = \begin{cases} 1 & \text{if current word } w_i \text{ ends in } \texttt{ing} \text{ and } y = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{102}(h, t) = \begin{cases} 1 & \text{if current word } w_i \text{ starts with } \texttt{pre} \text{ and } y = \texttt{NN} \\ 0 & \text{otherwise} \end{cases}$$

The Full Set of Features in [Ratnaparkhi 96]

- Contextual Features, e.g.,

$$f_{103}(x, y) = \begin{cases} 1 & \text{if } \langle t_{i-2}, t_{i-1}, y \rangle = \langle \text{DT, JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{104}(x, y) = \begin{cases} 1 & \text{if } \langle t_{i-1}, y \rangle = \langle \text{JJ, Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{105}(x, y) = \begin{cases} 1 & \text{if } \langle y \rangle = \langle \text{Vt} \rangle \\ 0 & \text{otherwise} \end{cases}$$

$$f_{106}(x, y) = \begin{cases} 1 & \text{if previous word } w_{i-1} = \textit{the} \text{ and } y = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{107}(x, y) = \begin{cases} 1 & \text{if next word } w_{i+1} = \textit{the} \text{ and } y = \text{Vt} \\ 0 & \text{otherwise} \end{cases}$$

The Final Result

- We can come up with practically any questions (*features*) regarding history/tag pairs.
- For a given history $x \in \mathcal{X}$, each label in $\mathcal{Y}$ is mapped to a different feature vector

$$\begin{aligned}
\mathbf{f}(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{Vt}) &= 1001011001001100110 \\
\mathbf{f}(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{JJ}) &= 0110010101011110010 \\
\mathbf{f}(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{NN}) &= 0001111101001100100 \\
\mathbf{f}(\langle \text{JJ, DT, } \langle \text{ Hispaniola, } \ldots \rangle, 6 \rangle, \text{IN}) &= 0001011011000000010 \\
&\ldots
\end{aligned}$$

Parameter Vectors

- Given features $f_k(x, y)$ for $k = 1 \ldots m$, also define a **parameter vector** $\mathbf{v} \in \mathbb{R}^m$
- Each (x, y) pair is then mapped to a “score”

$$\mathbf{v} \cdot \mathbf{f}(x, y) = \sum_k v_k f_k(x, y)$$

Language Modeling

- x is a “history” $w_1, w_2, \ldots w_{i-1}$, e.g.,

  Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order of statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical

- Each possible y gets a different score:

$$\begin{array}{rr} \mathbf{v} \cdot \mathbf{f}(x, model) = 5.6 & \mathbf{v} \cdot \mathbf{f}(x, the) = -3.2 \\ \mathbf{v} \cdot \mathbf{f}(x, is) = 1.5 & \mathbf{v} \cdot \mathbf{f}(x, of) = 1.3 \\ \mathbf{v} \cdot \mathbf{f}(x, models) = 4.5 & \ldots \end{array}$$

Log-Linear Models

- We have some input domain $\mathcal{X}$, and a finite label set $\mathcal{Y}$. Aim is to provide a conditional probability $P(y \mid x)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- A feature is a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
  (Often binary features or indicator functions $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$).
- Say we have m features f_k for $k = 1 \ldots m$
  $\Rightarrow$ A feature vector $\mathbf{f}(x, y) \in \mathbb{R}^m$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- We also have a **parameter vector** $\mathbf{v} \in \mathbb{R}^m$

- We define

$$\boxed{P(y \mid x, \mathbf{v}) = \frac{e^{\mathbf{v} \cdot \mathbf{f}(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x,y')}}}$$

More About Log-Linear Models

- Why the name?

$$\log P(y \mid x, \mathbf{v}) = \underbrace{\mathbf{v} \cdot \mathbf{f}(x, y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x, y')}}_{\text{Normalization term}}$$

- Maximum-likelihood estimates given training sample (x_i, y_i) for $i = 1 \ldots n$, each $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$:

$$\mathbf{v}_{ML} = \text{argmax}_{\mathbf{v} \in \mathbb{R}^m} L(\mathbf{v})$$

where

$$\begin{aligned} L(\mathbf{v}) &= \sum_{i=1}^{n} \log P(y_i \mid x_i) \\ &= \sum_{i=1}^{n} \mathbf{v} \cdot \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x_i, y')} \end{aligned}$$

Calculating the Maximum-Likelihood Estimates

- Need to maximize:

$$L(\mathbf{v}) = \sum_{i=1}^{n} \mathbf{v} \cdot \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x_i, y')}$$

- Calculating gradients:

$$\begin{aligned} \frac{dL(\mathbf{v})}{d\mathbf{v}} &= \sum_{i=1}^{n} \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} \mathbf{f}(x_i, y') e^{\mathbf{v} \cdot \mathbf{f}(x_i, y')}}{\sum_{z' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x_i, z')}} \\ &= \sum_{i=1}^{n} \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \mathbf{f}(x_i, y') \frac{e^{\mathbf{v} \cdot \mathbf{f}(x_i, y')}}{\sum_{z' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x_i, z')}} \\ &= \underbrace{\sum_{i=1}^{n} \mathbf{f}(x_i, y_i)}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \mathbf{f}(x_i, y') P(y' \mid x_i, \mathbf{v})}_{\text{Expected counts}} \end{aligned}$$

Gradient Ascent Methods

- Need to maximize $L(\mathbf{v})$ where

$$\frac{dL(\mathbf{v})}{d\mathbf{v}} = \sum_{i=1}^{n} \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \mathbf{f}(x_i, y') P(y' \mid x_i, \mathbf{v})$$

Initialization: $\mathbf{v} = 0$

Iterate until convergence:

- Calculate $\Delta = \frac{dL(\mathbf{v})}{d\mathbf{v}}$
- Calculate $\beta_* = \text{argmax}_\beta L(\mathbf{v} + \beta \Delta)$ (Line Search)
- Set $\mathbf{v} \leftarrow \mathbf{v} + \beta_* \Delta$

Conjugate Gradient Methods

- (Vanilla) gradient ascent can be very slow
- Conjugate gradient methods require calculation of gradient at each iteration, but do a line search in **a direction which is a function of the current gradient, and the previous step taken**.
- Conjugate gradient packages are widely available
  In general: they require a function

$$\texttt{calc\_gradient}(\mathbf{v}) \rightarrow \left(L(\mathbf{v}), \frac{dL(\mathbf{v})}{d\mathbf{v}} \right)$$

and that's about it!

Overview

- Log-linear models
- The maximum-entropy property
- Smoothing, feature selection etc. in log-linear models

Maximum-Entropy Properties of Log-Linear Models

- We define the set of distributions which satisfy linear constraints implied by the data:

$$\mathcal{P} = \{p \ : \ \underbrace{\sum_i \mathbf{f}(x_i, y_i)}_{\text{Empirical counts}} = \underbrace{\sum_i \sum_{y \in \mathcal{Y}} p(y \mid x_i) \mathbf{f}(x_i, y)}_{\text{Expected counts}}\}$$

here, p is an $n \times |\mathcal{Y}|$ vector defining $P(y \mid x_i)$ for all i, y.

- Note that at least one distribution satisfies these constraints, i.e.,

$$p(y \mid x_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}$$

Maximum-Entropy Properties of Log-Linear Models

- The **entropy** of any distribution is:

$$H(p) = -\left(\frac{1}{n} \sum_i \sum_{y \in \mathcal{Y}} p(y \mid x_i) \log p(y \mid x_i)\right)$$

- Entropy is a measure of "smoothness" of a distribution
- In this case, entropy is maximized by uniform distribution,

$$p(y \mid x_i) = \frac{1}{|\mathcal{Y}|} \quad \text{for all } y, x_i$$

The Maximum-Entropy Solution

- The maximum entropy model is

$$p_* = \text{argmax}_{p \in \mathcal{P}} H(p)$$

- Intuition: find a distribution which
  1. satisfies the constraints
  2. is as smooth as possible

Maximum-Entropy Properties of Log-Linear Models

- Consider the distribution

$$P(y|x, \mathbf{v}^*)$$

defined by the maximum-likelihood estimates $\mathbf{v}^* = \arg\max L(\mathbf{v})$

- Then $P(y|x, \mathbf{v}^*)$ is the maximum-entropy distribution

Is the Maximum-Entropy Property Useful?

- Intuition: find a distribution which
  1. satisfies the constraints
  2. is as smooth as possible

- One problem: the constraints are define by *empirical counts* from the data.

- Another problem: no formal relationship between maximum-entropy property and generalization(?) (at least none is given in the NLP literature)

Overview

- Log-linear models
- The maximum-entropy property
- Smoothing, feature selection etc. in log-linear models

Smoothing in Maximum Entropy Models

- Say we have a feature:

$$f_{100}(h,t) = \begin{cases} 1 & \text{if current word } w_i \text{ is } \texttt{base} \text{ and } t = \texttt{Vt} \\ 0 & \text{otherwise} \end{cases}$$

- In training data, `base` is seen 3 times, with `Vt` every time

- Maximum likelihood solution satisfies

$$\sum_i f_{100}(x_i, y_i) = \sum_i \sum_y p(y \mid x_i, \mathbf{v}) f_{100}(x_i, y)$$

$\Rightarrow p(\text{Vt} \mid x_i, \mathbf{v}) = 1$ for any history x_i where w_i = `base`
$\Rightarrow \mathbf{v}_{100} \rightarrow \infty$ at maximum-likelihood solution (most likely)
$\Rightarrow p(\text{Vt} \mid x, \mathbf{v}) = 1$ for any test data history x where w = `base`

A Simple Approach: Count Cut-Offs

- [Ratnaparkhi 1998] (PhD thesis): include all features that occur 5 times or more in training data. i.e.,

$$\sum_i f_k(x_i, y_i) \geq 5$$

for all features f_k.

Gaussian Priors

- Modified loss function

$$L(\mathbf{v}) \;=\; \sum_{i=1}^{n} \mathbf{v} \cdot \mathbf{f}(x_i, y_i) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{\mathbf{v} \cdot \mathbf{f}(x_i, y')} - \sum_{k=1}^{m} \frac{v_k^2}{2\sigma^2}$$

- Calculating gradients:

$$\frac{dL(\mathbf{v})}{d\mathbf{v}} \;=\; \underbrace{\sum_{i=1}^{n} \mathbf{f}(x_i, y_i)}_{\text{Empirical counts}} \; - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \mathbf{f}(x_i, y') P(y' \mid x_i, \mathbf{v})}_{\text{Expected counts}} - \frac{1}{\sigma^2}\mathbf{v}$$

- Can run conjugate gradient methods as before
- Adds a penalty for large weights

The Bayesian Justification for Gaussian Priors

- In *Bayesian* methods, combine the log-likelihood $P(data \mid \mathbf{v})$ with a prior over parameters, $P(\mathbf{v})$

$$P(\mathbf{v} \mid data) = \frac{P(data \mid \mathbf{v}) P(\mathbf{v})}{\int_{\mathbf{v}} P(data \mid \mathbf{v}) P(\mathbf{v}) d\mathbf{v}}$$

- The **MAP** (Maximum A-Posteriori) estimates are

$$\begin{aligned} \mathbf{v}_{MAP} &= \text{argmax}_{\mathbf{v}} P(\mathbf{v} \mid data) \\ &= \text{argmax}_{\mathbf{v}} \left(\underbrace{\log P(data \mid \mathbf{v})}_{\text{Log-Likelihood}} + \underbrace{\log P(\mathbf{v})}_{\text{Prior}} \right) \end{aligned}$$

- Gaussian prior: $P(\mathbf{v}) \propto e^{-\sum_k v_k^2 / 2\sigma^2}$
$\Rightarrow \log P(\mathbf{v}) = -\sum_k v_k^2 / 2\sigma^2 + C$

Experiments with Gaussian Priors

- [Chen and Rosenfeld, 1998]: apply maximum entropy models to language modeling: Estimate $P(w_i \mid w_{i-2}, w_{i-1})$
- Unigram, bigram, trigram features, e.g.,

$$f_1(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if trigram is } \texttt{(the,dog,laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if bigram is } \texttt{(dog,laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(w_{i-2}, w_{i-1}, w_i) = \begin{cases} 1 & \text{if unigram is } \texttt{(laughs)} \\ 0 & \text{otherwise} \end{cases}$$

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{e^{\mathbf{f}(w_{i-2}, w_{i-1}, w_i) \cdot \mathbf{v}}}{\sum_w e^{\mathbf{f}(w_{i-2}, w_{i-1}, w) \cdot \mathbf{v}}}$$

Experiments with Gaussian Priors

- In regular (unsmoothed) maxent, if all n-gram features are included, then it's equivalent to maximum-likelihood estimates!

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}$$

- [Chen and Rosenfeld, 1998]: with gaussian priors, get very good results. Performs as well as or better than standardly used "discounting methods" (see lecture 2).
- Note: their method uses development set to optimize σ parameters
- Downside: computing $\sum_w e^{\mathbf{f}(w_{i-2}, w_{i-1}, w) \cdot \mathbf{v}}$ is SLOW.

Feature Selection Methods

- Goal: find *a small number of features* which make good progress in optimizing log-likelihood
- A greedy method:

Step 1 Throughout the algorithm, maintain a set of active features. Initialize this set to be empty.

Step 2 Choose a feature from outside of the set of active features which has largest estimated impact in terms of increasing the log-likelihood and add this to the active feature set.

Step 3 Minimize $L(\mathbf{v})$ with respect to the set of active features. Return to **Step 2**.

Figures from [Ratnaparkhi 1998] (PhD thesis)

- The task: PP attachment ambiguity
- **ME Default:** Count cut-off of 5
- **ME Tuned:** Count cut-offs vary for 4-tuples, 3-tuples, 2-tuples, unigram features
- **ME IFS:** feature selection method

Experiment	Accuracy	Training Time	# of Features
ME Default	82.0%	10 min	4028
ME Tuned	83.7%	10 min	83875
ME IFS	80.5%	30 hours	387
DT Default	72.2%	1 min	
DT Tuned	80.4%	10 min	
DT Binary	-	1 week +	
Baseline	70.4%		

Table 8.2: Maximum Entropy (ME) and Decision Tree (DT) Experiments on PP attachment

Figures from [Ratnaparkhi 1998] (PhD thesis)

- A second task: text classification, identifying articles about acquisitions

Experiment	Accuracy	Training Time	# of Features
ME Default	95.5%	15 min	2350
ME IFS	95.8%	15 hours	356
DT Default	91.6%%	18 hours	
DT Tuned	92.1%	10 hours	

Table 8.4: Text Categorization Performance on the `acq` category

Summary

- Introduced log-linear models as general approach for modeling conditional probabilities $P(y \mid x)$.
- Optimization methods:
  - Iterative scaling
  - Gradient ascent
  - **Conjugate gradient ascent**
- Maximum-entropy properties of log-linear models
- Smoothing methods using Gaussian prior, and feature selection methods

References

[Ratnaparkhi 96] A maximum entropy part-of-speech tagger. In *Proceedings of the empirical methods in natural language processing conference*.