10. Maximum Entropy Models
Gerald Penn
Department of Computer Science, University of Toronto

(slides borrowed from Chris Manning and Dan Klein)
Introduction

- Much of what we’ve looked at has been “generative”
  - PCFGs, Naive Bayes for WSD
- In recent years there has been extensive use of *conditional* or *discriminative* probabilistic models in NLP, IR, Speech (and ML generally)
- Because:
  - They give high accuracy performance
  - They make it easy to incorporate lots of linguistically important features
  - They allow automatic building of language independent, retargetable NLP modules
Joint vs. Conditional Models

- We have some data $\{(d, c)\}$ of paired observations $d$ and hidden classes $c$.
- **Joint (generative) models** place probabilities over both observed data and the hidden stuff (generate the observed data from hidden stuff):
  - Many of the best known StatNLP models:
    - $n$-gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars
- **Discriminative (conditional) models** take the data as given, and put a probability over hidden structure given the data:
  - Logistic regression, conditional log-linear or maximum entropy models, conditional random fields, (SVMs, ...)
Bayes Net/Graphical Models

- Bayes net diagrams draw circles for random variables, and lines for direct dependencies
- Some variables are observed; some are hidden
- Each node is a little classifier (conditional probability table) based on incoming arcs

Naive Bayes

Generative

Logistic Regression

Discriminative
Conditional models work well: Word Sense Disambiguation

<table>
<thead>
<tr>
<th>Training Set</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td><strong>Accuracy</strong></td>
<td></td>
</tr>
<tr>
<td>Joint Like.</td>
<td>86.8</td>
<td></td>
</tr>
<tr>
<td>Cond. Like.</td>
<td>98.5</td>
<td></td>
</tr>
</tbody>
</table>

- Even with *exactly the same features*, changing from joint to conditional estimation increases performance.

<table>
<thead>
<tr>
<th>Test Set</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td><strong>Accuracy</strong></td>
<td></td>
</tr>
<tr>
<td>Joint Like.</td>
<td>73.6</td>
<td></td>
</tr>
<tr>
<td>Cond. Like.</td>
<td>76.1</td>
<td></td>
</tr>
</tbody>
</table>

(Klein and Manning 2002, using Senseval-1 Data)
Features

- In these slides and most MaxEnt work: *features* are elementary pieces of evidence that link aspects of what we observe $d$ with a category $c$ that we want to predict.
- A feature has a (bounded) real value: $f: C \times D \rightarrow \mathbb{R}$
- Usually features specify an indicator function of properties of the input and a particular class (*everyone we present is*). They pick out a subset.
  - $f_i(c, d) \equiv [\Phi(d) \land c = c_j]$ [Value is 0 or 1]
- We will freely say that $\Phi(d)$ is a feature of the data $d$, when, for each $c_j$, the conjunction $\Phi(d) \land c = c_j$ is a feature of the data-class pair $(c, d)$. 
Features

- For example:
  - $f_1(c, w_{i:t}) \equiv [c = \text{“NN”} \land \text{islower}(w_0) \land \text{ends}(w_0, \text{“d”})]$  
  - $f_2(c, w_{i:t}) \equiv [c = \text{“NN”} \land w_{-1} = \text{“to”} \land t_{-1} = \text{“TO”}]$  
  - $f_3(c, w_{i:t}) \equiv [c = \text{“VB”} \land \text{islower}(w_0)]$

- Models will assign each feature a weight

- Empirical count (expectation) of a feature:
  \[ \text{empirical } E(f_i) = \sum_{(c,d) \in \text{observed}(C,D)} f_i(c,d) \]

- Model expectation of a feature:
  \[ E(f_i) = \sum_{(c,d) \in (C,D)} P(c,d) f_i(c,d) \]
Feature-Based Models

- The decision about a data point is based only on the features active at that point.

<table>
<thead>
<tr>
<th>Data</th>
<th>Label</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSINESS: Stocks</td>
<td>BUSINESS</td>
<td>{..., stocks, hit, a, yearly, low, ...}</td>
</tr>
<tr>
<td>hit a yearly low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MONEY</td>
<td>{..., P=restructure, N=debt, L=12, ...}</td>
</tr>
<tr>
<td>... to restructure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bank:MONEY debt.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NN</td>
<td>{W=fall, PT=JJ PW=previous}</td>
</tr>
</tbody>
</table>

Text Categorization  
Word-Sense Disambiguation  
POS Tagging
Example: Text Categorization

(Zhang and Oles 2001)

- Features are a word in document and class (they do feature selection to use reliable indicators)
- Tests on classic Reuters data set (and others)
  - Naïve Bayes: 77.0% $F_1$
  - Linear regression: 86.0%
  - Logistic regression: 86.4%
  - Support vector machine: 86.5%
- Emphasizes the importance of regularization (smoothing) for successful use of discriminative methods (not used in most early NLP/IR work)
Example: POS Tagging

- Features can include:
  - Current, previous, next words in isolation or together.
  - Previous (or next) one, two, three tags.
  - Word-internal features: word types, suffixes, dashes, etc.

<table>
<thead>
<tr>
<th>Local Context</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3 DT The</td>
<td>W₀ 22.6</td>
</tr>
<tr>
<td>-2 NNP Dow</td>
<td>W₁ %</td>
</tr>
<tr>
<td>-1 VBD fell</td>
<td>W₋₁ fell</td>
</tr>
<tr>
<td>0 ??? 22.6</td>
<td>T₋₁ VBD</td>
</tr>
<tr>
<td>+1 ??? %</td>
<td>T₋₁₋₋₁ NNP-VBD</td>
</tr>
</tbody>
</table>

(Ratnaparkhi 1996; Toutanova et al. 2003, etc.)
Example: NER Interaction

Previous-state and current-signature have interactions, e.g. \( P=PERS-C=Xx \) indicates \( C=PERS \) much more strongly than \( C=Xx \) and \( P=PERS \) independently.

This feature type allows the model to capture this interaction.

### Local Context

<table>
<thead>
<tr>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Other</td>
<td>???</td>
</tr>
<tr>
<td>Word</td>
<td>at</td>
<td>Grace</td>
</tr>
<tr>
<td>Tag</td>
<td>IN</td>
<td>NNP</td>
</tr>
<tr>
<td>Sig</td>
<td>x</td>
<td>Xx</td>
</tr>
</tbody>
</table>

### Feature Weights

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>-0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>&lt;G</td>
<td>0.45</td>
<td>-0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>-0.70</td>
<td>-0.92</td>
</tr>
<tr>
<td>Current signature</td>
<td>Xx</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-Xx</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>Prev-cur-next sig</td>
<td>x-Xx-Xx</td>
<td>-0.69</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-Xx</td>
<td>-0.20</td>
<td>0.82</td>
</tr>
</tbody>
</table>

**Total:** -0.58 2.68
Other MaxEnt Examples

- Sentence boundary detection (Mikheev 2000)
  - Is period end of sentence or abbreviation?
- PP attachment (Ratnaparkhi 1998)
  - Features of head noun, preposition, etc.
- Language models (Rosenfeld 1996)
  - $P(w_0|w_{-n},...,w_{-1})$. Features are word n-gram features, and trigger features which model repetitions of the same word.
- Parsing (Ratnaparkhi 1997; Johnson et al. 1999, etc.)
  - Either: Local classifications decide parser actions or feature counts choose a parse.
A joint model gives probabilities $P(c, d)$ and tries to maximize this joint likelihood.

- It turns out to be trivial to choose weights: just relative frequencies.

A conditional model gives probabilities $P(c|d)$. It takes the data as given and models only the conditional probability of the class.

- We seek to maximize conditional likelihood.
- Harder to do (as we’ll see...)
- More closely related to classification error.
Feature-Based Classifiers

- “Linear” classifiers:
  - Classify from feature sets \( \{f_i\} \) to classes \( \{c\} \).
  - Assign a weight \( \lambda_i \) to each feature \( f_i \).
  - For a pair \( (c,d) \), features vote with their weights:
    - \( \text{vote}(c) = \sum \lambda_i f_i(c,d) \)
  - Choose the class \( c \) which maximizes \( \sum \lambda_i f_i(c,d) = \text{VB} \)
  - There are many ways to chose weights
    - Perceptron: find a currently misclassified example, and nudge weights in the direction of a correct classification
Feature-Based Classifiers

- Exponential (log-linear, maxent, logistic, Gibbs) models:
  - Use the linear combination $\sum \lambda_i f_i(c, d)$ to produce a probabilistic model:
    $$P(c|d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$
    - $P(\text{NN}|\text{to, aid, TO}) = \frac{e^{1.2} e^{-1.8}}{e^{1.2} e^{-1.8} + e^{0.3}} = 0.29$
    - $P(\text{VB}|\text{to, aid, TO}) = \frac{e^{0.3}}{e^{1.2} e^{-1.8} + e^{0.3}} = 0.71$
  - The weights are the parameters of the probability model, combined via a “soft max” function
  - Given this model form, we will choose parameters $\{\lambda_i\}$ that maximize the conditional likelihood of the data according to this model.
Other Feature-Based Classifiers

- The exponential model approach is one way of deciding how to weight features, given data.
- It constructs not only classifications, but probability distributions over classifications.
- There are other (good!) ways of discriminating classes: SVMs, boosting, even perceptrons – though these methods are not as trivial to interpret as distributions over classes.
Comparison to Naïve-Bayes

- Naïve-Bayes is another tool for classification:
  - We have a bunch of random variables (data features) which we would like to use to predict another variable (the class):
  - The Naïve-Bayes likelihood over classes is:

\[
\frac{P(c) \prod_{i} P(\phi_i | c)}{\sum_{c'} P(c') \prod_{i} P(\phi_i | c')} = \frac{\exp \left[ \log P(c) + \sum_{i} \log P(\phi_i | c) \right]}{\sum_{c'} \exp \left[ \log P(c') + \sum_{i} \log P(\phi_i | c') \right]} \frac{\exp \left[ \sum_{i} \lambda_{ic} f_{ic}(d, c) \right]}{\sum_{c'} \exp \left[ \sum_{i} \lambda_{ic'} f_{ic'}(d, c') \right]}
\]

- Naïve-Bayes is just an exponential model.
Comparison to Naïve-Bayes

The primary differences between Naïve-Bayes and maxent models are:

<table>
<thead>
<tr>
<th>Naïve-Bayes</th>
<th>Maxent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained to maximize joint likelihood of data and classes.</td>
<td>Trained to maximize the conditional likelihood of classes.</td>
</tr>
<tr>
<td>Features assumed to supply independent evidence.</td>
<td>Features weights take feature dependence into account.</td>
</tr>
<tr>
<td>Feature weights can be set independently.</td>
<td>Feature weights must be mutually estimated.</td>
</tr>
<tr>
<td>Features must be of the conjunctive $\Phi(d) \land c = c_i$ form.</td>
<td>Features need not be of this conjunctive form (but usually are).</td>
</tr>
</tbody>
</table>
Example: Sensors

**NB FACTORS:**
- $P(s) = 1/2$
- $P(+|s) = 1/4$
- $P(+|r) = 3/4$

**Reality**
- $P(+,+,r) = 3/8$
- $P(-,-,r) = 1/8$
- $P(-,-,s) = 3/8$
- $P(+,+,s) = 1/8$

**NB Model**

- **Raining?**
  - M1
  - M2

**NB FACTORS:**
- $P(s) = 1/2$
- $P(+|s) = 1/4$
- $P(+|r) = 3/4$

**PREDICTIONS:**
- $P(r,+,+) = (1/2)(3/4)(3/4)$
- $P(s,+,+) = (1/2)(1/4)(1/4)$
- $P(r|+,+) = 9/10$
- $P(s|+,+) = 1/10$
Example: Sensors

- **Problem:** NB multi-counts the evidence.

\[
\frac{P(r|+...+)}{P(s|+...+)} = \frac{P(r) P(+|r)}{P(s) P(+|s)} \cdots \frac{P(+|r)}{P(+|s)}
\]

- **Maxent behavior:**
  - Take a model over \((M_1,...M_n,R)\) with features:
    - \(f_{ri}: M_i=+, R=r\) weight: \(\lambda_{ri}\)
    - \(f_{si}: M_i=+, R=s\) weight: \(\lambda_{si}\)
    - \(\exp(\lambda_{ri}-\lambda_{si})\) is the factor analogous to \(P(+|r)/P(+|s)\)
    - ... but instead of being 3, it will be \(3^{1/n}\)
    - ... because if it were 3, \(E[f_{ri}^n]\) would be far higher than the target of 3/8!
  - **NLP problem:** we often have overlapping features....
Example: Stoplights

**Reality**

- **Lights Working**
  - P(g,r,w) = 3/7
  - P(r,g,w) = 3/7
- **Lights Broken**
  - P(r,r,b) = 1/7

**NB Model**

- Working?
  - NS
  - EW

**NB FACTORS:**

- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0
Example: Stoplights

- What does the model say when both lights are red?
  - $P(b, r, r) = \frac{1}{7}(1)(1) = \frac{1}{7} = \frac{4}{28}$
  - $P(w, r, r) = \frac{6}{7}(\frac{1}{2})(\frac{1}{2}) = \frac{6}{28} = \frac{6}{28}$
  - $P(w|r, r) = \frac{6}{10}$

- We’ll guess that $(r, r)$ indicates lights are working!

- Imagine if $P(b)$ were boosted higher, to $\frac{1}{2}$:
  - $P(b, r, r) = \frac{1}{2}(1)(1) = \frac{1}{2} = \frac{4}{8}$
  - $P(w, r, r) = \frac{1}{2}(\frac{1}{2})(\frac{1}{2}) = \frac{1}{8} = \frac{1}{8}$
  - $P(w|r, r) = \frac{1}{5}$

- Changing the parameters bought conditional accuracy at the expense of data likelihood!
Exponential Model Likelihood

- Maximum Likelihood (Conditional) Models:
  - Given a model form, choose values of parameters to maximize the (conditional) likelihood of the data.

- Exponential model form, for a data set (C,D):

\[
\log P(C|D, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c|d, \lambda) = \sum_{(c,d) \in (C,D)} \log \left( \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)} \right)
\]
Building a Maxent Model

- Define features (indicator functions) over data points.
  - Features represent sets of data points which are distinctive enough to deserve model parameters.
    - Words, but also “word contains number”, “word ends with ing”
    - Usually features are added incrementally to “target” errors.
  - For any given feature weights, we want to be able to calculate:
    - Data (conditional) likelihood
    - Derivative of the likelihood wrt each feature weight
      - Use expectations of each feature according to the model
- Find the optimum feature weights (MaxEnt).
Digression: Lagrange's Method

Task: find the highest yellow point.

This is "constrained optimization."
Digression: Lagrange's Method

\[ F(x,y): \text{height of } (x,y) \text{ on surface.} \]
\[ G(x,y): \text{color of } (x,y) \text{ on surface.} \]
Maximize \( F(x,y) \) subject to constraint: \( G(x,y) = k \).
Suppose \( G(x,y) - k = 0 \) is given by an implicit function \( y = f(x) \).
(We're allowed to change coordinate systems, too.)
So we really want to maximize \( u(x) = F(x, f(x)) \).
Digression: Lagrange's Method

Maximize $F(x, f(x))$ so we want $\frac{du}{dx} = 0$:

$$\frac{du}{dx} = 0 = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \frac{df}{dx}$$

We also know $G(x, f(x)) - k = 0$:

$$\frac{\partial G}{\partial x} + \frac{\partial G}{\partial y} \frac{df}{dx} = 0 \quad \Rightarrow \quad \frac{df}{dx} = -\frac{\partial G}{\partial x} \frac{\partial G}{\partial y}$$

So:

$$\frac{du}{dx} = \frac{\partial F}{\partial x} \frac{\partial G}{\partial y} - \frac{\partial F}{\partial y} \frac{\partial G}{\partial x} = 0$$

Let:

$$-\lambda = \frac{\partial F}{\partial x} \frac{\partial G}{\partial y} = \frac{\partial F}{\partial y} \frac{\partial G}{\partial x}$$
Lagrange Multipliers

These constants are called *Lagrange Multipliers*. They allow us to convert constraint optimization problems into unconstrained optimization problems:

\[-\lambda := \frac{\partial F}{\partial x} = \frac{\partial F}{\partial y} = -\frac{\partial G}{\partial x} = \frac{\partial G}{\partial y}\]

We don't actually care about $\Lambda$ - we want its derivatives to be 0:

\[0 = \frac{\partial F}{\partial x_i} + \lambda \frac{\partial G}{\partial x_i} \text{ for all } i\]
So what is/are G?

\[ \Lambda(x, y; \lambda) = F(x, y) + \sum_j \lambda_j G_j(x, y) \]

This generalizes to having multiple constraints - use one Lagrange multiplier for each.

We'll be searching over probability distributions \( p \) instead of \( (x, y) \).

But what should our constraints be? Answer:

\[ E_p(f_j) - E_{\bar{p}}(f_j) = 0 \]

Up to the sensitivity of our feature representation, \( p \) acts like what we see in our training data.
So what is F?

\[ \Lambda(x, y; \lambda) = F(x, y) + \sum_j \lambda_j G_j(x, y) \]

This generalizes to having multiple constraints - use one Lagrange multiplier for each.

We'll be searching over probability distributions \( p \) instead of \( (x, y) \).

But what should we maximize as a function of \( p \)? Answer...
Maximize Entropy!

- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty ("surprise"):  
  - Event \( x \)
  - Probability \( p_x \)
  - "Surprise" \( \log(1/p_x) \)
- Entropy: expected surprise (over \( p \)):
  \[
  H(p) = E_p \left[ \log_2 \frac{1}{p_x} \right] 
  \]
  \[
  H(p) = - \sum_x p_x \log_2 p_x 
  \]

A coin-flip is most uncertain for a fair coin.
Maximum Entropy Models

- Lots of distributions out there, most of them very spiked, specific, overfit.
- We want a distribution which is uniform except in specific ways we require.
- Uniformity means high entropy – we can search for distributions which have properties we desire, but also have high entropy.

*Ignorance is preferable to error and he is less remote from the truth who believes nothing than he who believes what is wrong* – Thomas Jefferson (1781)
Maxent Examples I

- What do we want from a distribution?
  - Minimize commitment = maximize entropy.
  - Resemble some reference distribution (data).
- Solution: maximize entropy $H$, subject to feature-based constraints:

\[ E_p[f_i] = E_{\tilde{p}}[f_i] \]

- Adding constraints (features):
  - Lowers maximum entropy
  - Raises maximum likelihood of data
  - Brings the distribution further from uniform
  - Brings the distribution closer to data
Maxent Examples II

\( H(p_H p_T, ) \quad p_H + p_T = 1 \quad p_H = 0.3 \)
Maxent Examples III

- Let's say we have the following event space:

<table>
<thead>
<tr>
<th>NN</th>
<th>NNS</th>
<th>NNP</th>
<th>NNPS</th>
<th>VBZ</th>
<th>VBD</th>
</tr>
</thead>
</table>

- ... and the following empirical data:

| 3  | 5  | 11 | 13  | 3  | 1  |

- Maximize $H$:

| $1/e$ | $1/e$ | $1/e$ | $1/e$ | $1/e$ | $1/e$ |

- ... want probabilities: $E[NN, NNS, NNP, NNPS, VBZ, VBD] = 1$

| $1/6$ | $1/6$ | $1/6$ | $1/6$ | $1/6$ | $1/6$ | $1/6$ |
Maxent Examples IV

- Too uniform!
- N* are more common than V*, so we add the feature $f_n = \{\text{NN, NNS, NNP, NNPS}\}$, with $E[f_n] = 32/36$

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>NNS</th>
<th>NNP</th>
<th>NNPS</th>
<th>VBZ</th>
<th>VBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>8/36</td>
<td>8/36</td>
<td>8/36</td>
<td>8/36</td>
<td>2/36</td>
<td>2/36</td>
</tr>
</tbody>
</table>

- ... and proper nouns are more frequent than common nouns, so we add $f_p = \{\text{NNP, NNPS}\}$, with $E[f_p] = 24/36$

<table>
<thead>
<tr>
<th></th>
<th>NN</th>
<th>NNS</th>
<th>NNP</th>
<th>NNPS</th>
<th>VBZ</th>
<th>VBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>4/36</td>
<td>4/36</td>
<td>12/36</td>
<td>12/36</td>
<td>2/36</td>
<td>2/36</td>
</tr>
</tbody>
</table>

- ... we could keep refining the models, e.g. by adding a feature to distinguish singular vs. plural nouns, or verb types.
Digression: Jensen's Inequality

\[ f(\sum_i w_i x_i) \geq \sum_i w_i f(x_i) \text{ where } \sum_i w_i = 1 \]

Convexity guarantees a single, global maximum because any higher points are greedily reachable.
Convexity

- Constrained \( H(p) = - \sum x \log x \) is convex:
  - \( - x \log x \) is convex
  - \( - \sum x \log x \) is convex (sum of convex functions is convex).
- The feasible region of constrained \( H \) is a linear subspace (which is convex)
- The constrained entropy surface is therefore convex.
- The maximum likelihood exponential model (dual) formulation is also convex.
The Kuhn-Tucker Theorem

\[ \Lambda(p;\lambda)=H(p)+\sum_j \lambda_j(E_{p}f_j-E_{\bar{p}}f_j) \]

When the components of this are convex, we can find the optimal \( p \) and \( \lambda \) by first calculating:

\[ p_{\lambda}=\arg\max_{p} \Lambda(p;\lambda) \]

with \( \lambda \) held constant, then solving the “dual:”

\[ \bar{\lambda}=\arg\max_{\lambda} \Lambda(p_{\lambda},\lambda). \]

The optimal \( p \) is then \( p_{\bar{\lambda}} \).
The Kuhn-Tucker Theorem

\[ \Lambda(p; \lambda) = H(p) + \sum_j \lambda_j (E_p f_j - E_{\bar{p}} f_j) \]

For us, there is an analytic solution to the first part:

\[ p_\lambda(c|d) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)} \]

So the only thing we have to do is find \( \lambda \), given this.
Digression: Log-Likelihoods

- The (log) conditional likelihood is a function of the iid data \((C, D)\) and the parameters \(\lambda\):
  \[
  \log P(C \mid D, \lambda) = \log \prod_{(c, d) \in (C, D)} P(c \mid d, \lambda) = \sum_{(c, d) \in (C, D)} \log P(c \mid d, \lambda)
  \]

- If there aren’t many values of \(c\), it’s easy to calculate:
  \[
  \log P_\lambda(C \mid D, \lambda) = \sum_{(c, d) \in (C, D)} \log \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}
  \]

- We can separate this into two components:
  \[
  \log P_\lambda(C \mid D, \lambda) = \sum_{(c, d) \in (C, D)} \log \exp \sum_i \lambda_i f_i(c, d) - \sum_{(c, d) \in (C, D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c', d)
  \]
  \[
  \log P(C \mid D, \lambda) = N(\lambda) - M(\lambda)
  \]

- The derivative is the difference between the derivatives of each component
LL Derivative I: Numerator

\[
\frac{\partial N(\lambda)}{\partial \lambda_i} = \frac{\partial}{\partial \lambda_i} \sum_{(c,d) \in (C,D)} \log \exp \sum_i \lambda_{ci} f_i(c,d)
\]

\[
= \sum_{(c,d) \in (C,D)} \frac{\partial \sum_i \lambda_i f_i(c,d)}{\partial \lambda_i}
\]

\[
= \sum_{(c,d) \in (C,D)} f_i(c,d)
\]

Derivative of the numerator is: the empirical count\((f_i, c)\)
\[
\frac{\partial M(\lambda)}{\partial \lambda_i} = \frac{\partial}{\partial \lambda_i} \sum_{(c,d) \in (C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c', d) \\
= \sum_{(c,d) \in (C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \frac{\partial}{\partial \lambda_i} \sum_{c'} \exp \sum_i \lambda_i f_i(c', d) \frac{\partial}{\partial \lambda_i} \sum_{c'} \exp \sum_i \lambda_i f_i(c', d) \\
= \sum_{(c,d) \in (C,D)} \sum_{c'} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'', d)} \sum_{c'} \frac{\partial}{\partial \lambda_i} \sum_{c'} \exp \sum_i \lambda_i f_i(c', d) \\
= \sum_{(c,d) \in (C,D)} \sum_{c'} \sum_{c''} \frac{P(c'|d, \lambda)}{f_i(c', d)} = \text{predicted count}(f_i, \lambda)
\]
Our choice of constraint is vindicated: with our choice of $\rho_\lambda$, these correspond to the stable equilibrium points of the log conditional likelihood with respect to $\lambda$.

The optimum distribution is:
- Always unique (but parameters may not be unique)
- Always exists (if feature counts are from actual data).
Fitting the Model

- To find the parameters $\lambda_1, \lambda_2, \lambda_3 \ldots$
  write out the conditional log-likelihood of the training data and maximize it

$$CLogLik\left(D\right) = \sum_{i=1}^{n} \log P(c_i|d_i)$$

- The log-likelihood is concave and has a single maximum; use your favorite numerical optimization package

- Good large scale techniques: conjugate gradient or limited memory quasi-Newton
Fitting the Model
Generalized Iterative Scaling

- A simple optimization algorithm which works when the features are non-negative
- We need to define a slack feature to make the features sum to a constant over all considered pairs from $D \times C$.

Define

$$M = \max_{i, c} \sum_{j=1}^{m} f_j(d_i, c)$$

Add new feature

$$f_{m+1}(d, c) = M - \sum_{j=1}^{m} f_j(d, c)$$
Generalized Iterative Scaling

- Compute empirical expectation for all features:
  \[
  E_{\tilde{p}}(f_j) = \frac{1}{N} \sum_{i=1}^{n} f_j(d_i, c_i)
  \]

- Initialize \( \lambda_j = 0, j=1\ldots m+1 \)

- Repeat
  - Compute feature expectations according to current model
    \[
    E_{p^t}(f_j) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} P(c_k | d_i) f_j(d_i, c_k)
    \]
  - Update parameters:
    \[
    \lambda_{j(t+1)} = \lambda_{j(t)} + \frac{1}{M} \log \left( \frac{E_{\tilde{p}}(f_j)}{E_{p^t}(f_j)} \right)
    \]

- Until converged
Classification

- What do these joint models of $P(X)$ have to do with conditional models $P(C|D)$?
- Think of the space $C \times D$ as a complex $X$.
  - $C$ is generally small (e.g., 2-100 topic classes)
  - $D$ is generally huge (e.g., space of documents)
- We can, in principle, build models over $P(C,D)$.
- This will involve calculating expectations of features (over $C \times D$):

$$E(f_i) = \sum_{(c,d) \in (C,D)} P(c,d)f_i(c,d)$$

- Generally impractical: can’t enumerate $d$ efficiently.
Classification II

- $D$ may be huge or infinite, but only a few $d$ occur in our data.
- What if we add one feature for each $d$ and constrain its expectation to match our empirical data?

$$\forall (d) \in D \quad P(d) = \tilde{P}(d)$$

- Now, most entries of $P(c,d)$ will be zero.
- We can therefore use the much easier sum:

$$E(f_i) = \sum_{(c,d) \in (C,D)} P(c,d)f_i(c,d)$$

$$= \sum_{(c,d) \in (C,D) \land \tilde{P}(d) > 0} P(c,d)f_i(c,d)$$
Classification III

- But if we’ve constrained the $D$ marginals

\[ \forall (d) \in D \quad P(d) = \tilde{P}(d) \]

then the only thing that can vary is the conditional distributions:

\[ P(c,d) = P(c|d)P(d) = P(c|d)\tilde{P}(d) \]

- This is the connection between joint and conditional maxent / exponential models:
  - Conditional models can be thought of as joint models with marginal constraints.
  - Maximizing joint likelihood and conditional likelihood of the data in this model are equivalent!
Feature Overlap

- Maxent models handle overlapping features well.
- Unlike a NB model, there is no double counting!

<table>
<thead>
<tr>
<th>Empirical</th>
<th>All = 1</th>
<th>A = 2/3</th>
<th>A = 2/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A a</td>
<td>B 1/4</td>
<td>B 1/3</td>
<td>B 1/3</td>
</tr>
<tr>
<td>B 2 1</td>
<td>1/4</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td>b 2 1</td>
<td>1/4</td>
<td>1/6</td>
<td>1/6</td>
</tr>
</tbody>
</table>

A a

B 1/3

b 1/6

A A a

B λA

b λA

A A a

B λ′A + λ″A

b λ′A + λ″A
Example: NER Overlap

Grace is correlated with PERSON, but does not add much evidence on top of already knowing prefix features.

Local Context

<table>
<thead>
<tr>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Other</td>
<td>???</td>
</tr>
<tr>
<td>Word</td>
<td>at</td>
<td>Grace</td>
</tr>
<tr>
<td>Tag</td>
<td>IN</td>
<td>NNP</td>
</tr>
<tr>
<td>Sig</td>
<td>x</td>
<td>Xx</td>
</tr>
</tbody>
</table>

Feature Weights

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>-0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Beginning bigram</td>
<td>&lt;G</td>
<td>0.45</td>
<td>-0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>-0.70</td>
<td>-0.92</td>
</tr>
<tr>
<td>Current signature</td>
<td>Xx</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-Xx</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>Prev-cur-next sig</td>
<td>x-Xx-Xx</td>
<td>-0.69</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-Xx</td>
<td>-0.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
</tr>
</tbody>
</table>
**Feature Interaction**

- Maxent models handle overlapping features well, but do not automatically model feature interactions.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Empirical</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All = 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>b</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A = 2/3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1/3</td>
<td>1/6</td>
</tr>
<tr>
<td>b</td>
<td>1/3</td>
<td>1/6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B = 2/3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4/9</td>
<td>2/9</td>
</tr>
<tr>
<td>b</td>
<td>2/9</td>
<td>1/9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All = 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A = 2/3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>(\lambda_A)</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>(\lambda_A)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B = 2/3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>(\lambda_A + \lambda_B)</td>
<td>(\lambda_B)</td>
</tr>
<tr>
<td>b</td>
<td>(\lambda_A)</td>
<td></td>
</tr>
</tbody>
</table>
Feature Interaction

- If you want interaction terms, you have to add them:

<table>
<thead>
<tr>
<th>A</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

Empirical

- A disjunctive feature would also have done it (alone):
Feature Interaction

- For log-linear/logistic regression models in statistics, it is standard to do a greedy stepwise search over the space of all possible interaction terms.
- This combinatorial space is exponential in size, but that’s okay as most statistics models only have 4–8 features.
- In NLP, our models commonly use hundreds of thousands of features, so that’s not okay.
- Formerly, interaction terms were added by hand based on linguistic intuitions.