
# Feature-based Linear Classifiers

How to put features into a classifier


# Feature-Based Linear Classifiers

- Linear classifiers are a linear function from feature sets $\{f_i\}$ to classes $\{c\}$
- At test time, we consider each class $c$ for a datum $d$
  - We generate a feature set $\{f_i\}$ for an observed datum-class pair $(c,d)$
  - Each feature $f_i$ has a weight $\lambda_i$
  - We then score each possible class assignment: $\text{vote(c)} = \Sigma\lambda_i f_i(c,d)$
  - We choose the class $c$ which maximizes $\Sigma\lambda_i f_i(c,d)$
- At training time we have observed $(c,d)$ pairs from labeled examples
  - We generate sets of features $\{f_i(c,d)\}$ for them
  - We use information about what features occur and don't occur to set a weight $\lambda_i$ for each feature


# Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \wedge w_{-1} = \text{“in”} \wedge \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \wedge \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \wedge \text{ends}(w, \text{“c”})]$


- Models will assign to each feature a *weight:*
  - A positive weight votes that this configuration is likely correct
  - A negative weight votes that this configuration is likely incorrect


# Feature-Based Linear Classifiers

- Maxent (softmax, multiclass logistic, exponential, log-linear, Gibbs) models:
  - Make a probabilistic model from the linear combination $\Sigma\lambda_i f_i(c,d)$

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

Makes votes positive

Normalizes votes

  - P(LOCATION|*in Québec*) = $e^{1.8}e^{-0.6}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.586$
  - P(DRUG|*in Québec*) = $e^{0.3}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.238$
  - P(PERSON|*in Québec*) = $e^{0}/(e^{1.8}e^{-0.6} + e^{0.3} + e^{0}) = 0.176$
  - The weights are the parameters of the probability model, combined via a "soft max" function


# Feature-Based Linear Classifiers

- Maxent (exponential, log-linear, logistic, Gibbs) models:
  - Given this model form, we will choose parameters $\{\lambda_i\}$ that *maximize the conditional likelihood* of the data according to this model.
  - We construct not only classifications, but probability distributions over classifications.
    - There are other (good!) ways of discriminating classes – SVMs, boosting, even perceptrons – but these methods are not as trivial to interpret as distributions over classes.


# Feature Expectations

- We will crucially make use of two *expectations*
  - actual or predicted counts of a feature firing:

- 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:

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


# Building a Maxent Model

- We 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*", POS, syntactic structure, relation between two phrases, etc.
- We might simply encode each $\Phi$ feature as a unique String
  - A datum will give rise to a set of Strings: the active $\Phi$ features
  - Each feature $f_i(c, d) \equiv [\Phi(d) \wedge c = c_j]$ gets a real number weight
- We concentrate on $\Phi$ features but the math uses $i$ indices of $f_i$


# Building a Maxent Model

- Features are often added during model development to target errors
  - Often, the easiest thing to think of are features that mark bad combinations
- Then, for any given feature weights, we want to be able to calculate:
  - Data conditional likelihood
  - Derivative of the likelihood wrt each feature weight
    - Uses expectations of each feature according to the model
- We can then find the optimum feature weights (discussed later).


# Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning


# Introduction

- So far we've looked at "generative models"
  - Language models, IBM alignment models, PCFGs
- But there is now much use of conditional or discriminative probabilistic models in NLP, Speech, IR (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
  - They generate the observed data from the hidden stuff
  - All the classic StatNLP models:
    - *n*-gram models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models

$P(c,d)$


# Joint vs. Conditional Models

- Discriminative (conditional) models take the data as given, and put a probability/score over hidden structure given the data:
  - Logistic regression, conditional loglinear or maximum entropy models, conditional random fields
  - Also, SVMs, (averaged) perceptron, etc. are discriminative classifiers (but not directly probabilistic)

$P(c|d)$


# Conditional vs. Joint Likelihood

- A *joint* model gives probabilities P(*d*,*c*) = P(*c*)P(*d*|*c*) 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.

Christopher Manning


# Conditional models work well: Word Sense Disambiguation

| Training Set | |
|---|---|
| Objective | Accuracy |
| Joint Like. | 86.8 |
| Cond. Like. | 98.5 |

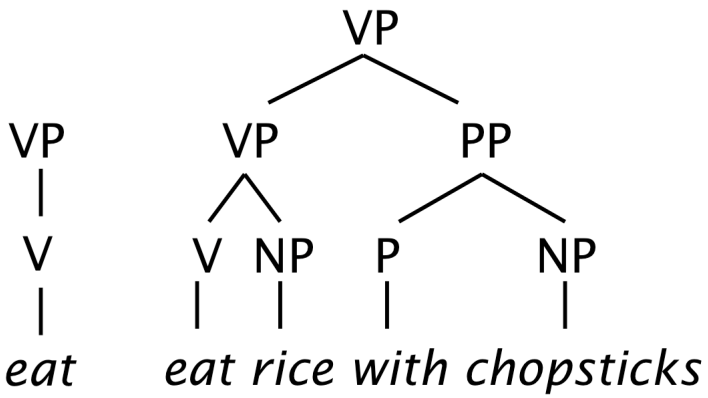
| Test Set | |
|---|---|
| Objective | Accuracy |
| Joint Like. | 73.6 |
| Cond. Like. | 76.1 |

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

- Even with exactly the same features, changing from joint to conditional estimation increases performance
- That is, we use the same smoothing, and the same word-class features, we just change the numbers (parameters)

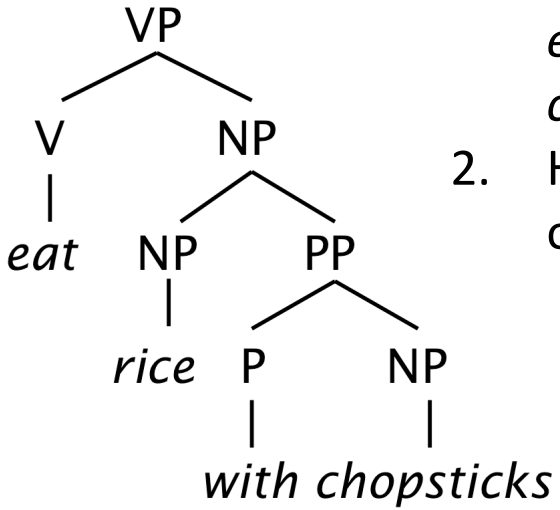
Christopher Manning


# PCFGs Maximize Joint, not Conditional Likelihood










1. What parse for *eat rice with chopsticks?*
2. How can you get the other parse?

**46** **6** **2**

Based on an example by Mark Johnson


# Maxent Models and Discriminative Estimation

Maximizing the likelihood


# Exponential Model Likelihood

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

$$\log P(C \mid D, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c \mid d, \lambda) = \sum_{(c,d) \in (C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$


# The Likelihood Value

- The (log) conditional likelihood of iid* data (*C*,*D*) according to a maxent model is a function of the data and the parameters λ:

$$\log P(C \mid D, \lambda) = \log \prod_{(c,d) \in (C,D)} P(c \mid d, \lambda) = \sum_{(c,d) \in (C,D)} \log P(c \mid d, \lambda)$$

- If there aren't many values of *c*, it's easy to calculate:

$$\log P(C \mid D, \lambda) = \sum_{(c,d) \in (C,D)} \log \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

*A fancy statistics term meaning "independent and identically distributed". You normally need to assume this for anything formal to be derivable, even though it's never quite true in practice.


# The Likelihood Value

- We can separate this into two components:

$$\log P(C \mid D, \lambda) = \sum_{(c,d) \in (C,D)} \log \exp \sum_i \lambda_i f_i(c,d) \; - \sum_{(c,d) \in (C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)$$

$$\log P(C \mid D, \lambda) = N(\lambda) \; - \; M(\lambda)$$

- We can maximize it by finding where the derivative is 0
- The derivative is the difference between the derivatives of each component


# The Derivative I: Numerator

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

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

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

Derivative of the numerator is: the empirical count($f_i$, $c$)


# The Derivative II: Denominator

$$\frac{\partial M(\lambda)}{\partial \lambda_i} = \frac{\partial \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \frac{\partial \sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \frac{1}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c',d)}{1} \frac{\partial \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

$$= \sum_{(c,d)\in(C,D)} \sum_{c'} \frac{\exp \sum_i \lambda_i f_i(c',d)}{\sum_{c''} \exp \sum_i \lambda_i f_i(c'',d)} \frac{\partial \sum_i \lambda_i f_i(c',d)}{\partial \lambda_i}$$

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

= predicted count($f_i$, $\lambda$)


# The Derivative III

$$\frac{\partial \log P(C \mid D, \lambda)}{\partial \lambda_i} = \text{actual count}(f_i, C) - \text{predicted count}(f_i, \lambda)$$

- The optimum parameters are the ones for which each feature's predicted expectation equals its empirical expectation. The optimum distribution is:
  - Always unique (but parameters may not be unique)
  - Always exists (if feature counts are from actual data).
- These models are also called maximum entropy models because we find the model having maximum entropy and satisfying the constraints: $E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$

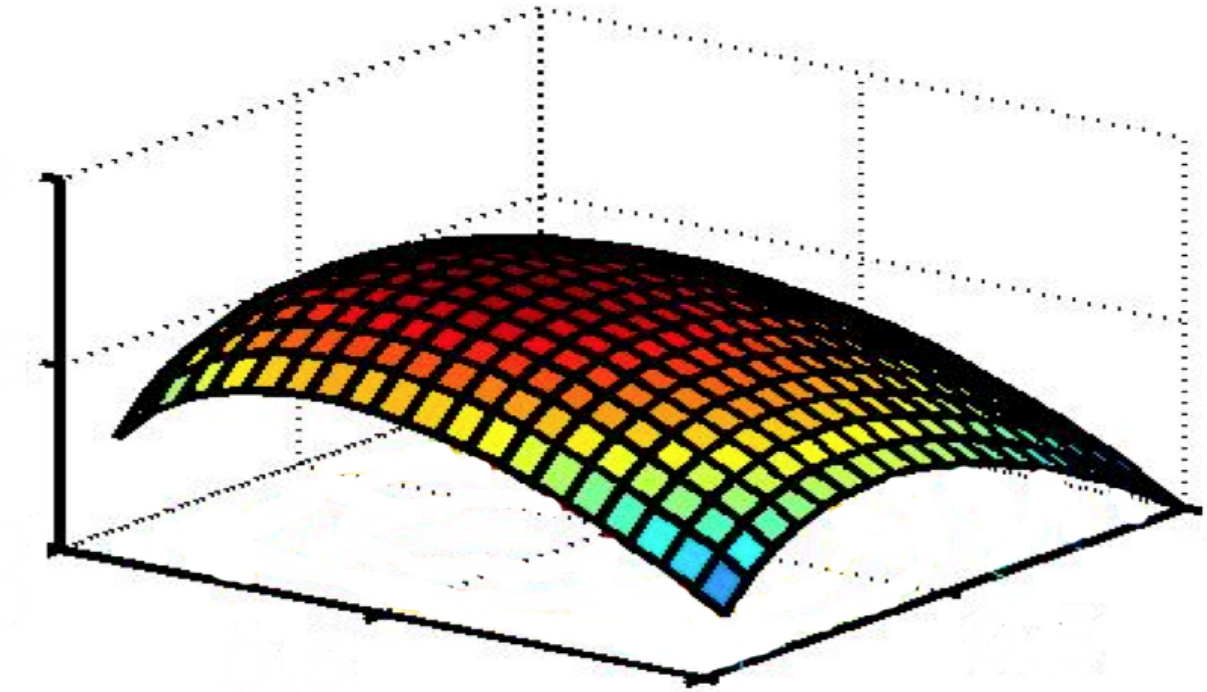

# Finding the optimal parameters

- We want to choose parameters $\lambda_1, \lambda_2, \lambda_3, \ldots$ that maximize the conditional log-likelihood of the training data

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

- To be able to do that, we've worked out how to calculate the function value and its partial derivatives (its gradient)


# A likelihood surface




# Finding the optimal parameters

- Use your favorite numerical optimization package….
  - Commonly (and in our code), you **minimize** the negative of *CLogLik*

1. Gradient descent (GD); Stochastic gradient descent (SGD)
2. Iterative proportional fitting methods: Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS)
3. Conjugate gradient (CG), perhaps with preconditioning
4. Quasi-Newton methods – limited memory variable metric (LMVM) methods, in particular, L-BFGS


# Named Entity Recognition


# Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.


# Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.


# Named Entity Recognition (NER)

- A very important sub-task: find and classify names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation


# Named Entity Recognition (NER)

- The uses:
  - Named entities can be indexed, linked off, etc.
  - Sentiment can be attributed to companies or products
  - A lot of relations (*employs, won, born-in*) are between named entities
  - For question answering, answers are often named entities.
- Concretely:
  - Many web pages tag various entities, with links to bio or topic pages, etc.
    - Reuters' OpenCalais, Evri, AlchemyAPI, Yahoo's Term Extraction, …
  - Apple/Google/Microsoft/… smart recognizers for document content


# Maximum entropy sequence models

Maximum entropy Markov models (MEMMs) or Conditional Markov models


# The Named Entity Recognition Task

Task: Predict entities in a text

| | |
|---|---|
| Foreign | ORG |
| Ministry | ORG |
| spokesman | O |
| Shen | PER |
| Guofang | PER |
| told | O |
| Reuters | ORG |
| : | : |

}


# Sequence problems

- Many problems in NLP have data which is a sequence of characters, words, phrases, lines, or sentences …
- We can think of our task as one of labeling each item

| VBG | NN | IN | DT | NN | IN | NN |
|---|---|---|---|---|---|---|
| Chasing | opportunity | in | an | age | of | upheaval |

**POS tagging**

| PERS | O | O | O | ORG | ORG |
|---|---|---|---|---|---|
| Murdoch | discusses | future | of | News | Corp. |

**Named entity recognition**

| B | B | I | I | B | I | B | I | B | B |
|---|---|---|---|---|---|---|---|---|---|
| 而 | 相 | 对 | 于 | 这 | 些 | 品 | 牌 | 的 | 价 |

**Word segmentation**


**Text segmen-tation**


# MEMM inference in systems

- For a Conditional Markov Model (CMM) a.k.a. a Maximum Entropy Markov Model (MEMM), the classifier makes a single decision at a time, conditioned on evidence from observations and previous decisions
- A larger space of sequences is usually explored via search

Local Context

Decision Point

| -3 | -2 | -1 | 0 | +1 |
|---|---|---|---|---|
| ORG | ORG | O | ??? | ??? |
| Xerox | Corp. | fell | 22.6 | % |

Features

| | |
|---|---|
| $W_0$ | 22.6 |
| $W_{+1}$ | % |
| $W_{-1}$ | fell |
| $C_{-1}$ | O |
| $C_{-1}$-$C_{-2}$ | ORG-O |
| hasDigit? | true |
| … | … |

(Borthwick 1999, Klein et al. 2003, etc.)


# Example: NER

- Scoring individual labeling decisions is no more complex than standard classification decisions
  - We have some assumed labels to use for prior positions
  - We use features of those and the observed data (which can include current, previous, and next words) to predict the current label

Local Context

Decision Point

| -3 | -2 | -1 | 0 | +1 |
|---|---|---|---|---|
| ORG | ORG | O | ??? | ??? |
| Xerox | Corp. | fell | 22.6 | % |

Features

| | |
|---|---|
| $W_0$ | 22.6 |
| $W_{+1}$ | % |
| $W_{-1}$ | fell |
| $C_{-1}$ | O |
| $C_{-1}$-$C_{-2}$ | ORG-O |
| hasDigit? | true |
| … | … |


# Named Entity Recognition Evaluation

Task: Predict entities in a text

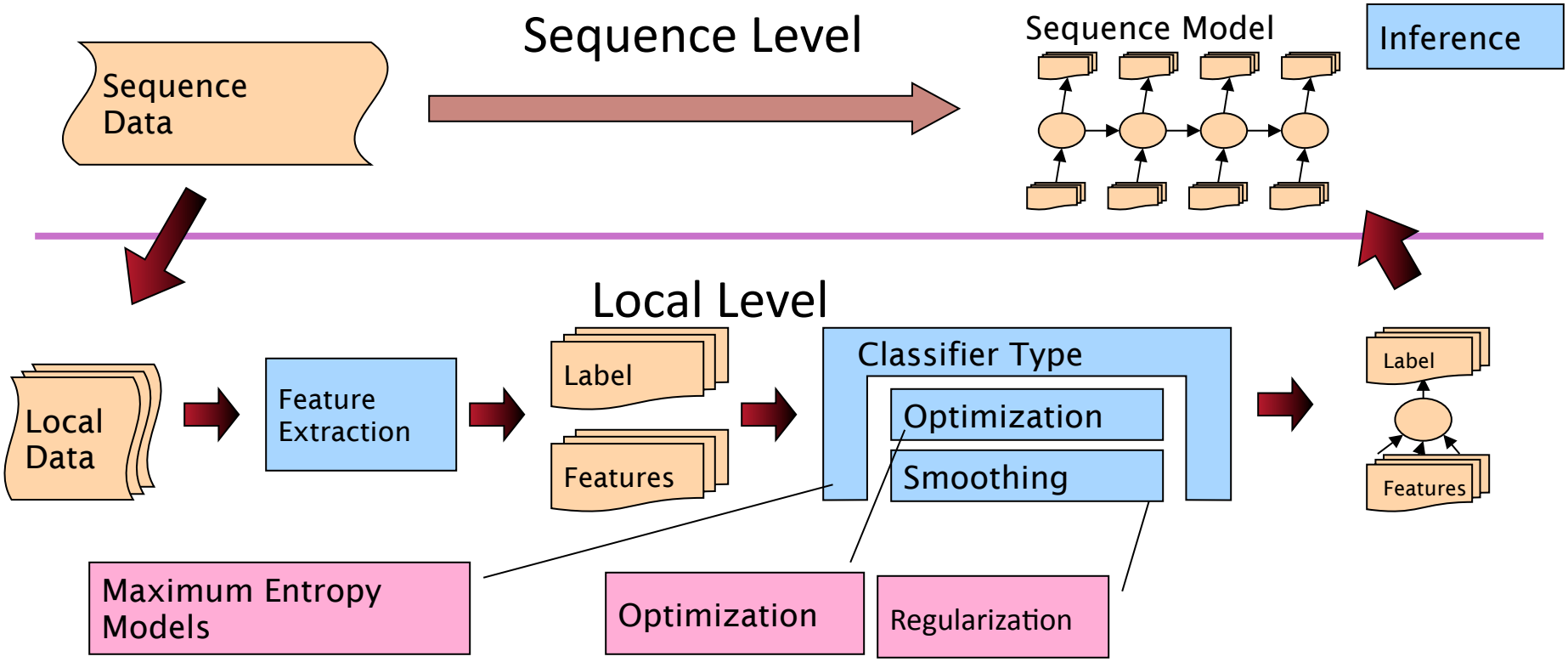
| | |
|---|---|
| Foreign | ORG |
| Ministry | ORG |
| spokesman | O |
| Shen | PER |
| Guofang | PER |
| told | O |
| Reuters | ORG |
| : | : |

} Standard evaluation is per entity, *not* per token

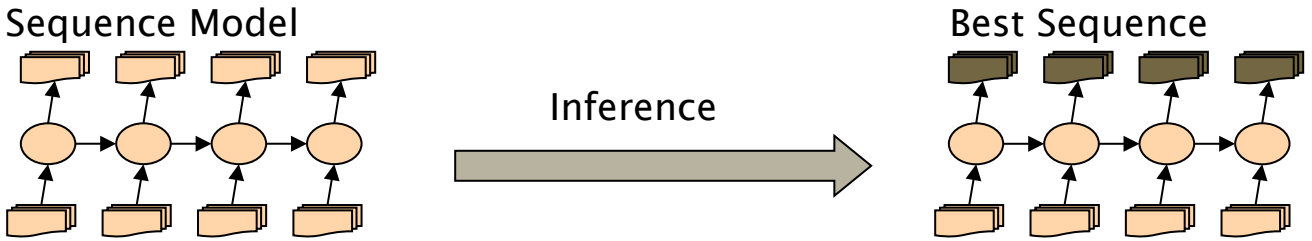

# Precision/Recall/F1 for IE/NER

- Recall and precision are straightforward for tasks like IR and text categorization, where there is only one grain size (documents)
- The measure behaves a bit funnily for IE/NER when there are *boundary errors* (which are *common*):
  - First Bank of Chicago announced earnings …
- This counts as both a fp and a fn
- Selecting *nothing* would have been better
- Some other metrics (e.g., MUC scorer) give partial credit (according to complex rules)


# Inference in Systems

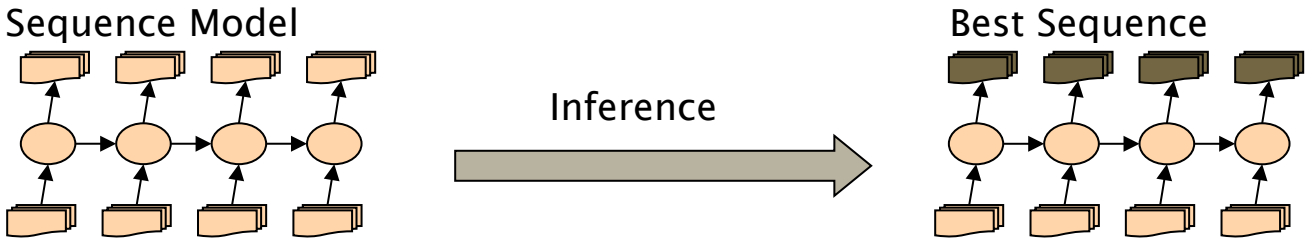



# Greedy Inference



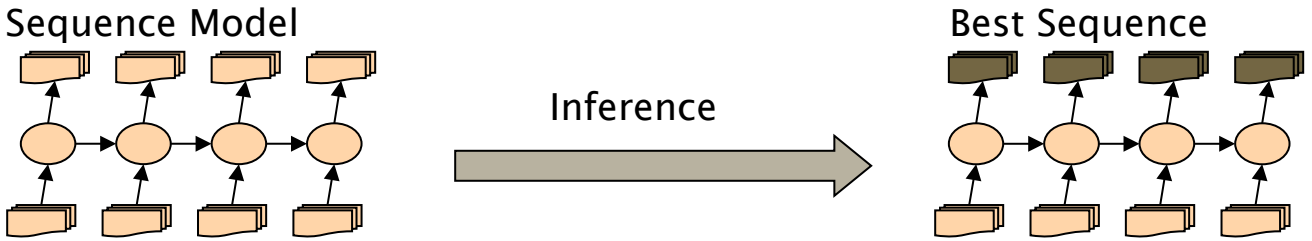

- Greedy inference:
  - We just start at the left, and use our classifier at each position to assign a label
  - The classifier can depend on previous labeling decisions as well as observed data
- Advantages:
  - Fast, no extra memory requirements
  - Very easy to implement
  - With rich features including observations to the right, it can perform quite well
- Disadvantage:
  - Greedy. We make commit errors we cannot recover from


# Beam Inference




- Beam inference:
  - At each position keep the top $k$ complete sequences.
  - Extend each sequence in each local way.
  - The extensions compete for the $k$ slots at the next position.
- Advantages:
  - Fast; beam sizes of 3–5 are almost as good as exact inference in many cases.
  - Easy to implement (no dynamic programming required).
- Disadvantage:
  - Inexact: the globally best sequence can fall off the beam.


# Viterbi Inference




- Viterbi inference:
  - Dynamic programming or memoization.
  - Requires small window of state influence (e.g., past two states are relevant).
- Advantage:
  - Exact: the global best sequence is returned.
- Disadvantage:
  - Harder to implement long-distance state-state interactions (but beam inference tends not to allow long-distance resurrection of sequences anyway).


# CRFs [Lafferty, Pereira, and McCallum 2001]

- Another sequence model: Conditional Random Fields (CRFs)
- A whole-sequence conditional model rather than a chaining of local models.

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)}$$

- The space of $c$'s is now the space of sequences
  - But if the features $f_i$ remain local, the conditional sequence likelihood can be calculated exactly using dynamic programming
- Training is slower, but CRFs avoid causal-competition biases
- These (or a variant using a max margin criterion) are seen as the state-of-the-art these days … but in practice usually work much the same as MEMMs.


# Maxent Models and Discriminative Estimation

The maximum entropy model presentation


# Aside: logistic regression

- Maxent models in NLP are essentially the same as multiclass logistic regression models in statistics (or machine learning)
  - If you haven't seen these before, don't worry, this presentation is self-contained!
  - If you have seen these before you might think about:
    - The parameterization is slightly different in a way that is advantageous for NLP-style models with tons of sparse features (but statistically inelegant)
    - The more general form of feature functions in this presentation
      - The features are more general: $f_i(c, d) \equiv [\Phi(d)$ **if** $c = c_j]$ is logistic regr.
      - When is it useful to have $f$ be more generally also a function of the class?


# Maximum Entropy Models

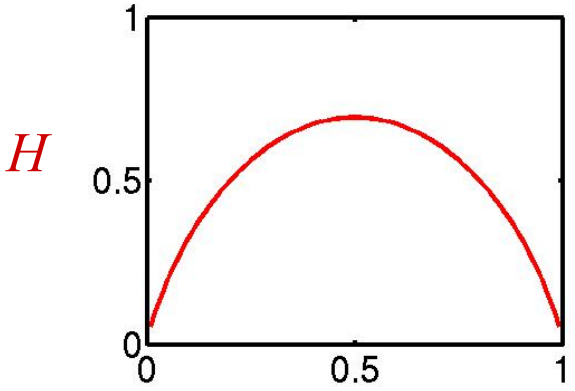
- An equivalent approach:
  - Lots of distributions out there, most of them very spiked, specific, overfit.
  - We want a distribution which is uniform except in specific ways we require.
  - Uniformity means high entropy – we can search for distributions which have properties we desire, but also have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


# (Maximum) Entropy

- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty ("surprise"):
  - Event $x$
  - Probability $p_x$
  - "Surprise" $\log(1/p_x)$
- Entropy: expected surprise (over $p$):

$$\mathrm{H}(p) = E_p\left[\log_2 \frac{1}{p_x}\right] = -\sum_x p_x \log_2 p_x$$



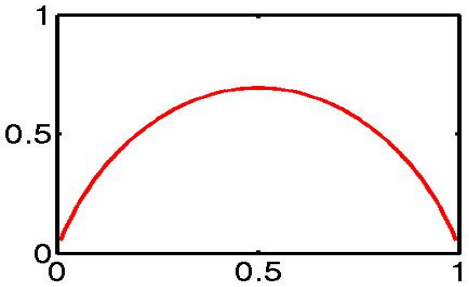

A coin-flip is most uncertain for a fair coin.


# Maxent Examples I

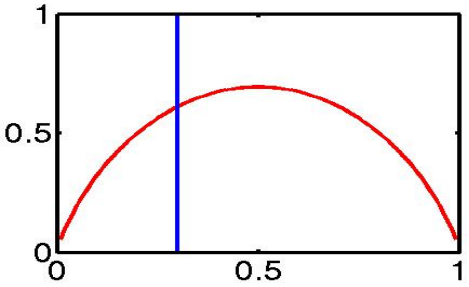
- What do we want from a distribution?
  - Minimize commitment = maximize entropy.
  - Resemble some reference distribution (data).
- Solution: maximize entropy *H*, subject to feature-based constraints:

$$E_p[f_i] = E_{\hat{p}}[f_i] \Leftrightarrow \sum_{x \in f_i} p_x = C_i$$

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

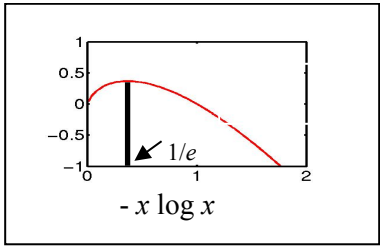



Unconstrained, max at 0.5

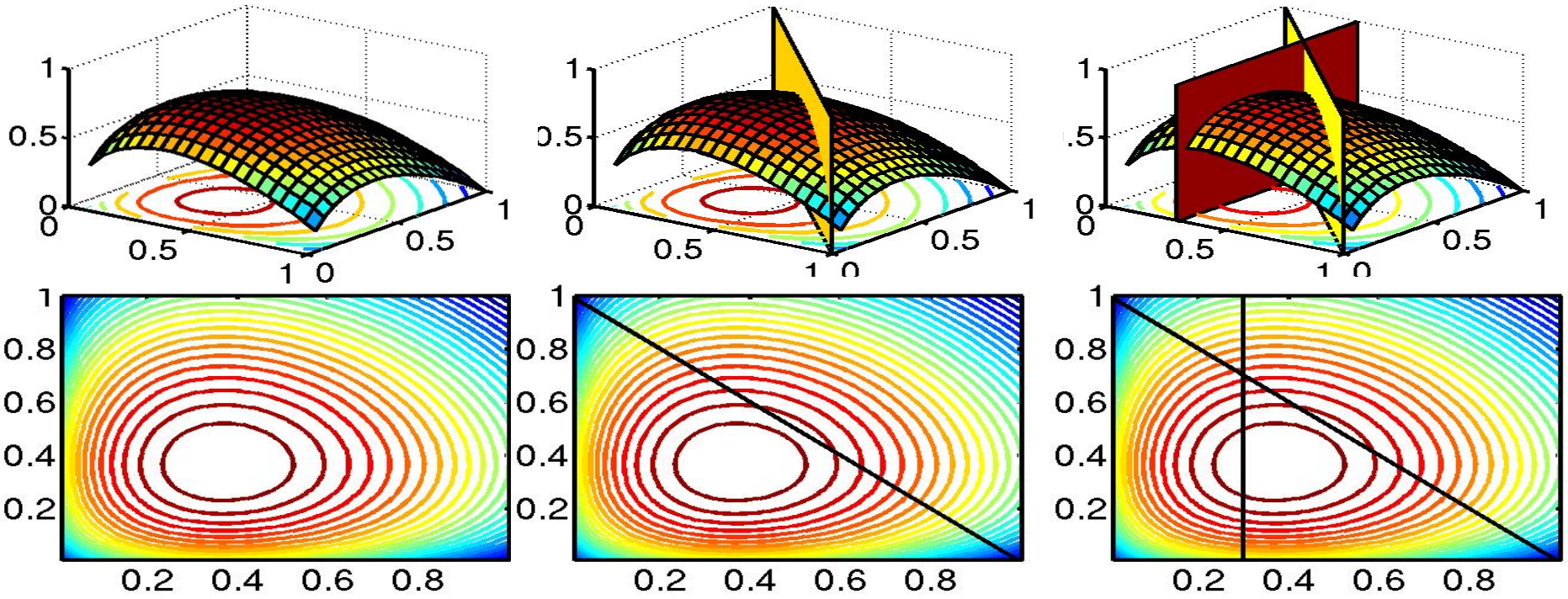



Constraint that $p_{\text{HEADS}} = 0.3$

# Maxent Examples II




$\mathrm{H}(p_{\mathrm{H}} p_{\mathrm{T}},)$ $p_{\mathrm{H}} + p_{\mathrm{T}} = 1$ $p_{\mathrm{H}} = 0.3$




# Maxent Examples III

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

| NN | NNS | NNP | NNPS | VBZ | VBD |
|---|---|---|---|---|---|

- … 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 |
|---|---|---|---|---|---|


# Maxent Examples IV

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

| NN | NNS | NNP | NNPS | VBZ | VBD |
|---|---|---|---|---|---|
| 8/36 | 8/36 | 8/36 | 8/36 | 2/36 | 2/36 |

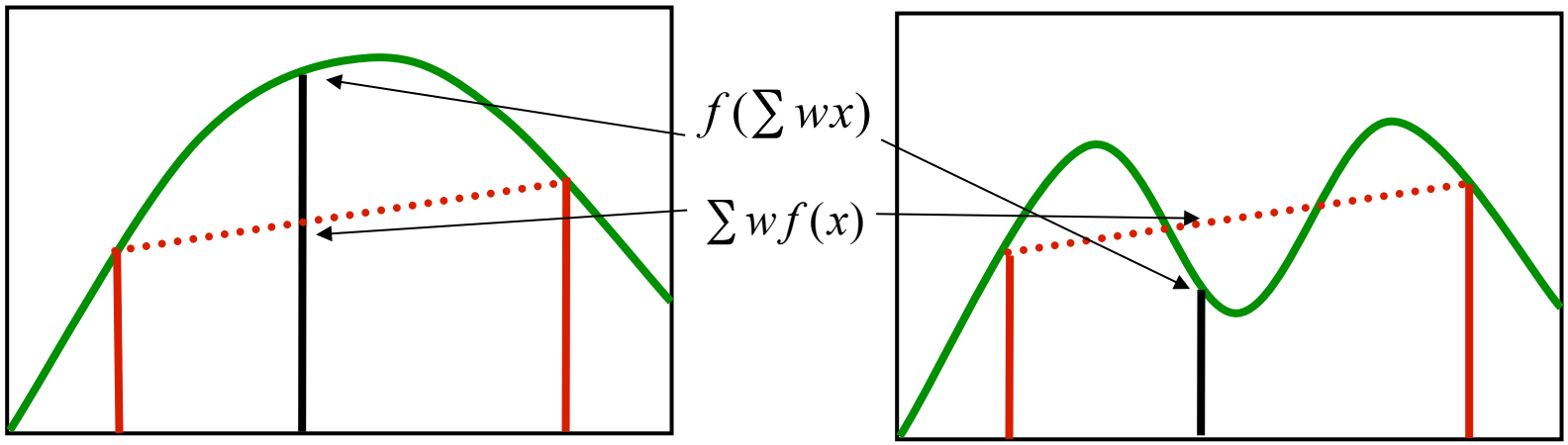
- … and proper nouns are more frequent than common nouns, so we add $f_P$ = {NNP, NNPS}, with E[$f_P$] =24/36

| | | | | | |
|---|---|---|---|---|---|
| 4/36 | 4/36 | 12/36 | 12/36 | 2/36 | 2/36 |

- … we could keep refining the models, e.g., by adding a feature to distinguish singular vs. plural nouns, or verb types.


# Convexity

$$f(\sum_i w_i x_i) \geq \sum_i w_i f(x_i) \quad \sum_i w_i = 1$$



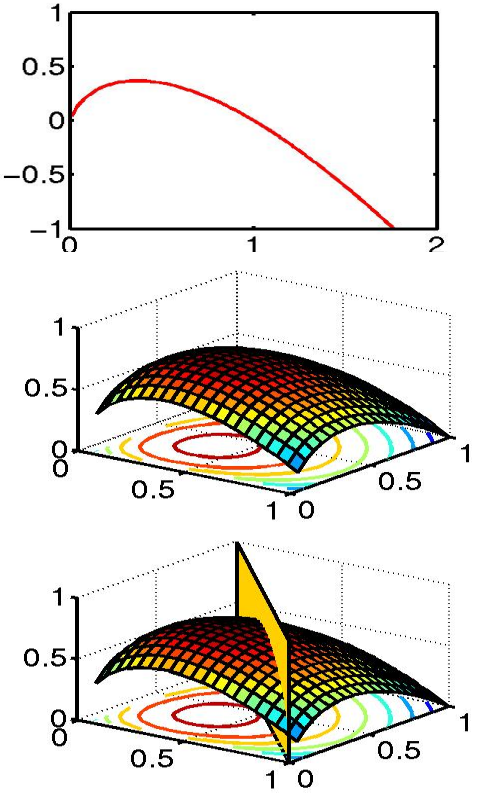

Convex

Non-Convex

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


# Convexity II

- 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.




# Maxent Models and Discriminative Estimation

The maximum entropy model presentation


# Feature Overlap/ Feature Interaction

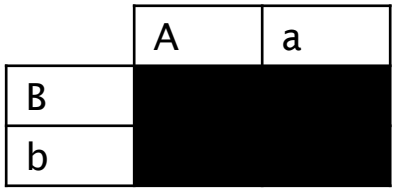
How overlapping features work in maxent models

# Feature Overlap

- Maxent models handle overlapping features well.

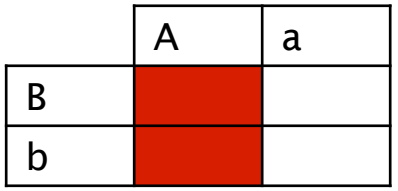
Empirical

| | A | a |
|---|---|---|
| B | 2 | 1 |
| b | 2 | 1 |



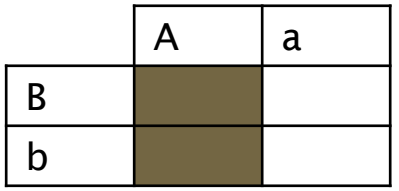
All = 1

| | A | a |
|---|---|---|
| B | 1/4 | 1/4 |
| b | 1/4 | 1/4 |



A = 2/3

| | A | a |
|---|---|---|
| B | 1/3 | 1/6 |
| b | 1/3 | 1/6 |



A = 2/3

| | A | a |
|---|---|---|
| B | 1/3 | 1/6 |
| b | 1/3 | 1/6 |

| | A | a |
|---|---|---|
| B | | |
| b | | |

| | A | a |
|---|---|---|
| B | $\lambda_A$ | |
| b | $\lambda_A$ | |

| | A | a |
|---|---|---|
| B | $\lambda'_A+\lambda''_A$ | |
| b | $\lambda'_A+\lambda''_A$ | |


# Example: Named Entity Feature Overlap

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

## Local Context

| | Prev | Cur | Next |
|---|---|---|---|
| State | Other | ??? | ??? |
| Word | at | Grace | Road |
| Tag | IN | NNP | NNP |
| Sig | x | Xx | Xx |

## Feature Weights

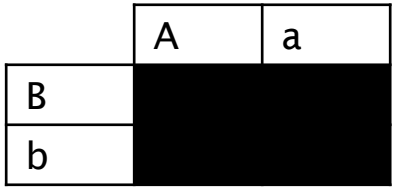
| Feature Type | Feature | PERS | LOC |
|---|---|---|---|
| Previous word | *at* | -0.73 | 0.94 |
| Current word | *Grace* | 0.03 | 0.00 |
| Beginning bigram | *<G* | 0.45 | -0.04 |
| Current POS tag | NNP | 0.47 | 0.45 |
| Prev and cur tags | IN NNP | -0.10 | 0.14 |
| Previous state | Other | -0.70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

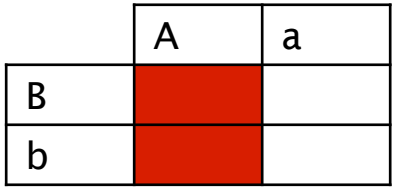
# Feature Interaction

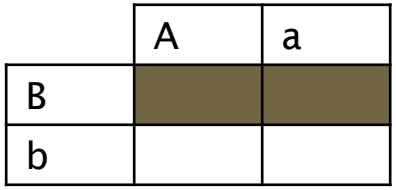
- Maxent models handle overlapping features well, but do not automatically model feature interactions.

Empirical

| | A | a |
|---|---|---|
| B | 1 | 1 |
| b | 1 | 0 |




All = 1

| | A | a |
|---|---|---|
| B | 1/4 | 1/4 |
| b | 1/4 | 1/4 |

A = 2/3

| | A | a |
|---|---|---|
| B | 1/3 | 1/6 |
| b | 1/3 | 1/6 |

B = 2/3

| | A | a |
|---|---|---|
| B | 4/9 | 2/9 |
| b | 2/9 | 1/9 |

| | A | a |
|---|---|---|
| B | 0 | 0 |
| b | 0 | 0 |

| | A | a |
|---|---|---|
| B | $\lambda_A$ | |
| b | $\lambda_A$ | |

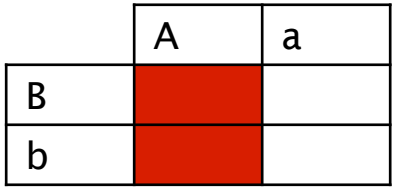
| | A | a |
|---|---|---|
| B | $\lambda_A+\lambda_B$ | $\lambda_B$ |
| b | $\lambda_A$ | |


# Feature Interaction

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

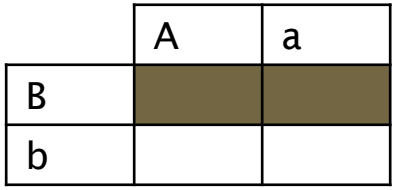
Empirical

| | A | a |
|---|---|---|
| B | 1 | 1 |
| b | 1 | 0 |



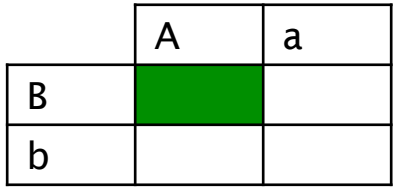
A = 2/3

| | A | a |
|---|---|---|
| B | 1/3 | 1/6 |
| b | 1/3 | 1/6 |



B = 2/3

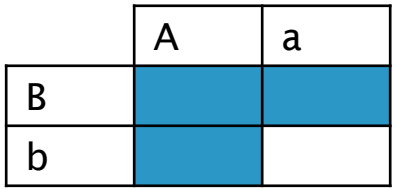
| | A | a |
|---|---|---|
| B | 4/9 | 2/9 |
| b | 2/9 | 1/9 |



AB = 1/3

| | A | a |
|---|---|---|
| B | 1/3 | 1/3 |
| b | 1/3 | 0 |

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



| | A | a |
|---|---|---|
| B | 1/3 | 1/3 |
| b | 1/3 | 0 |


# Feature Interaction

- For loglinear/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.
- Commonly, interaction terms are added by hand based on linguistic intuitions.


# 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

| | Prev | Cur | Next |
|---|---|---|---|
| State | Other | ??? | ??? |
| Word | at | Grace | Road |
| Tag | IN | NNP | NNP |
| Sig | x | Xx | Xx |

## Feature Weights

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