Maxent Models and Discriminative Estimation

Generative vs. Discriminative models

Christopher Manning

Introduction

- So far we've mainly looked at "generative models"
  - Language models, IBM alignment models, PCFGs
- But there is much use of conditional or discriminative 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 easy 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 ends up trivial to choose weights: just count! (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.

Conditional models work well: Word Sense Disambiguation

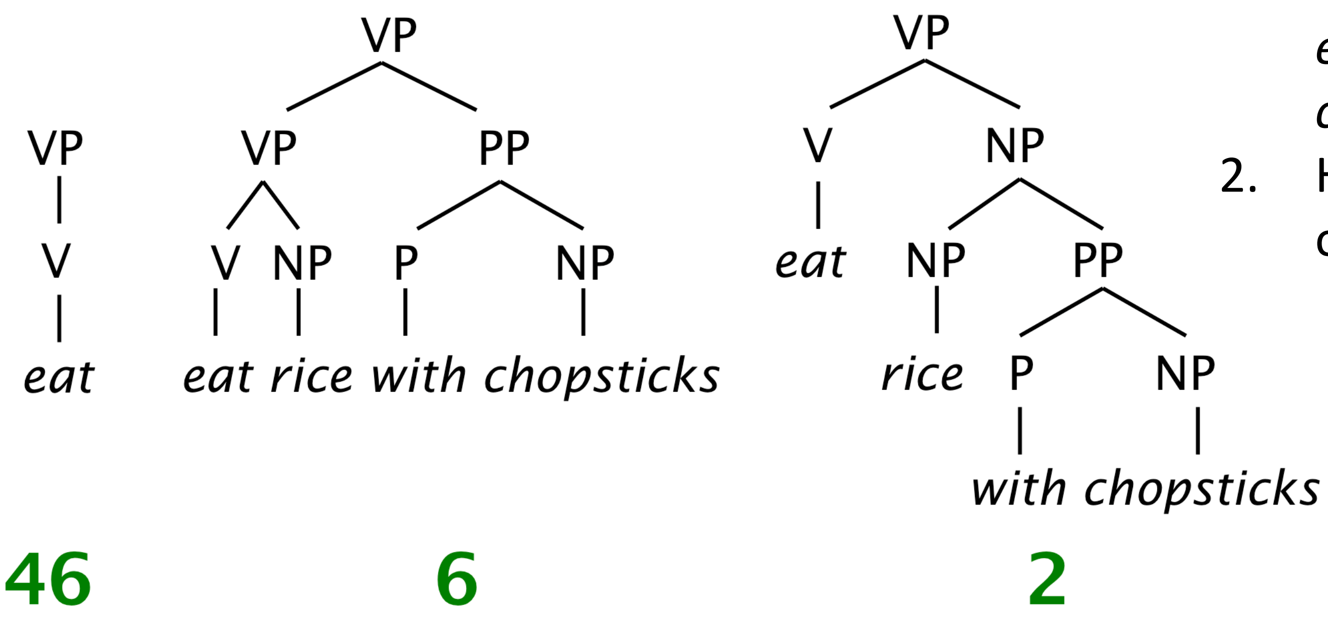
Training Set	
Objective	Accuracy
Joint Like.	86.8
Cond. Like.	98.5

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)

PCFGs Maximize Joint, not Conditional Likelihood



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

Based on an example by Mark Johnson

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

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 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)

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

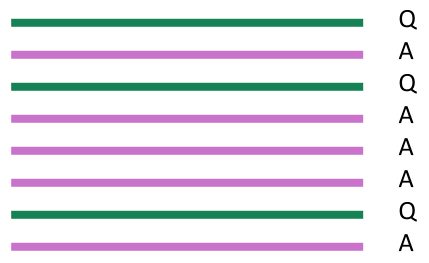
POS tagging

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

Word segmentation

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

Named entity recognition



Text segmentation

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

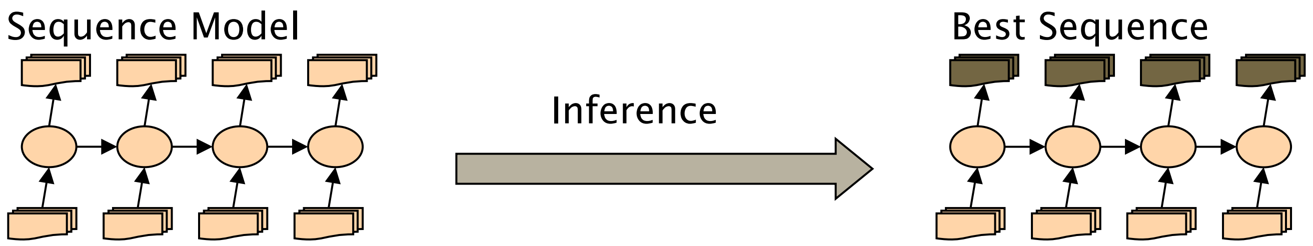
Decision Point

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

Features

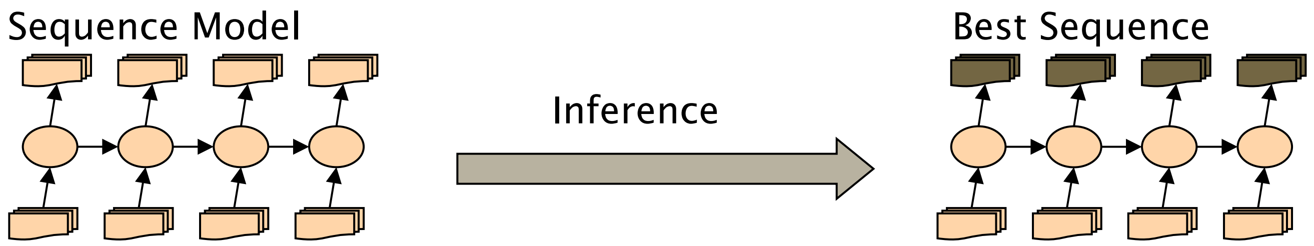
W_0	22.6
W_{+1}	%
W_{-1}	fell
C_{-1}	O
C_{-1}-C_{-2}	ORG-O
hasDigit?	true
…	…

Greedy Inference



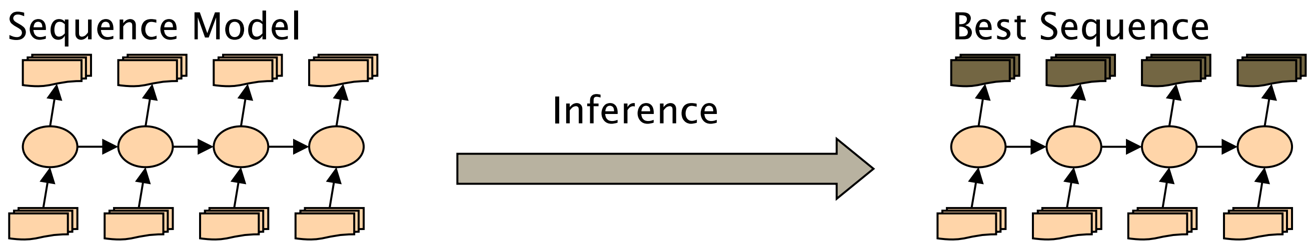
- Greedy inference:
  - We just start at the left, and use our classifier at each position to assign a label
  - The classifier can depend on previous labeling decisions as well as observed data
- Advantages:
  - Fast, no extra memory requirements
  - Very easy to implement
  - With rich features including 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).

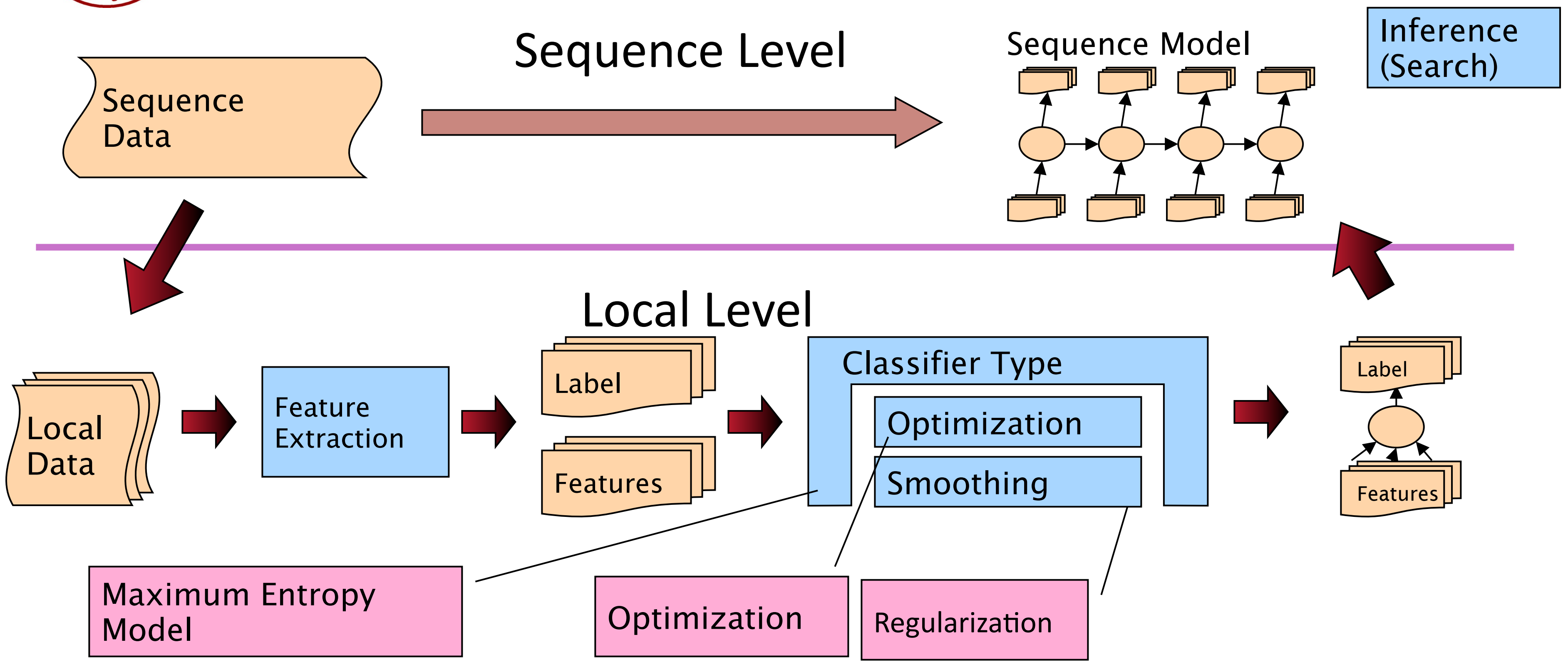
CRFs [Lafferty, Pereira, and McCallum 2001]

- 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 they usually work much the same as MEMMs.

Inference in Systems



Maxent Models and Discriminative Estimation

The maximum entropy model presentation

Logistic regression

- Maxent models in NLP are essentially the same as multiclass logistic regression models in statistics (or machine learning)
  - 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: $f_i(c, d) \equiv [\Phi(d) \textbf{ if } c = c_j]$ is logistic regr.
      - With f a function of the class, we can do more → structured prediction
  - If you haven't seen these before, don't worry, this presentation is self-contained!

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)

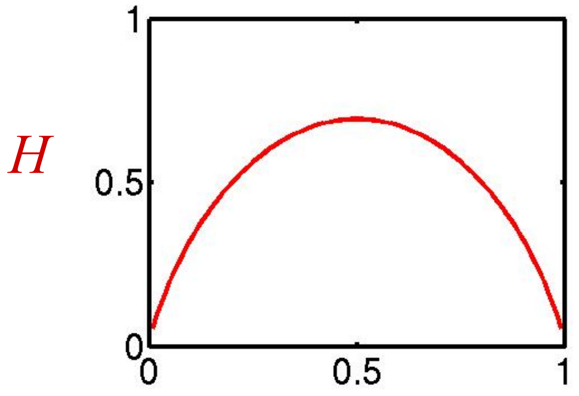
(Maximum) Entropy

- Entropy: the uncertainty of a distribution.
- Quantifying uncertainty ("surprise"):

• Event	x
• Probability	p_x
• "Surprise"	$\log(1/p_x)$

- Entropy: expected surprise (over *p*):

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



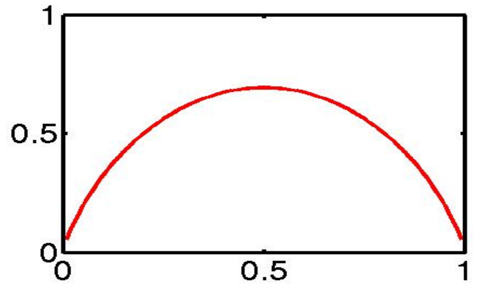
A coin-flip is most uncertain for a fair coin.

Maxent Examples I

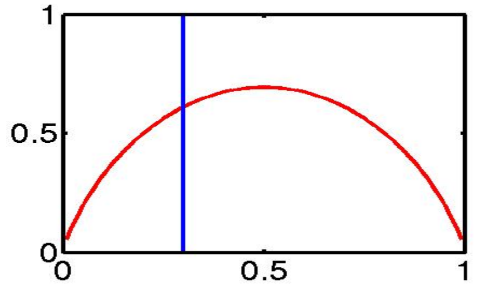
- What do we want from a distribution?
  - Minimize commitment = maximize entropy.
  - Resemble some reference distribution (data).
- Solution: maximize entropy *H*, subject to feature-based constraints:

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

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

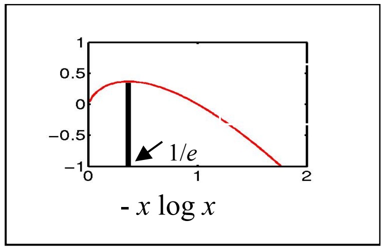


Unconstrained, max at 0.5

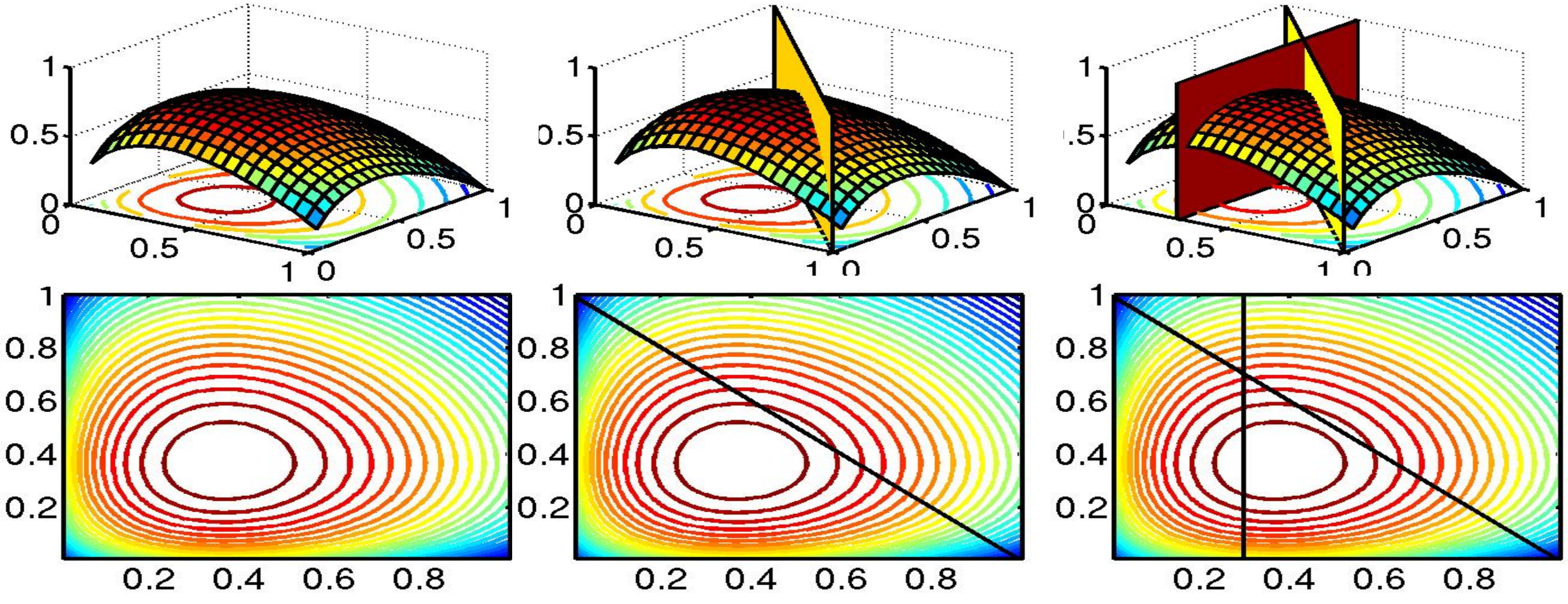


Constraint that $p_{\text{HEADS}} = 0.3$

Maxent Examples II



$H(p_H p_T,)$ $p_H + p_T = 1$ $p_H = 0.3$



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

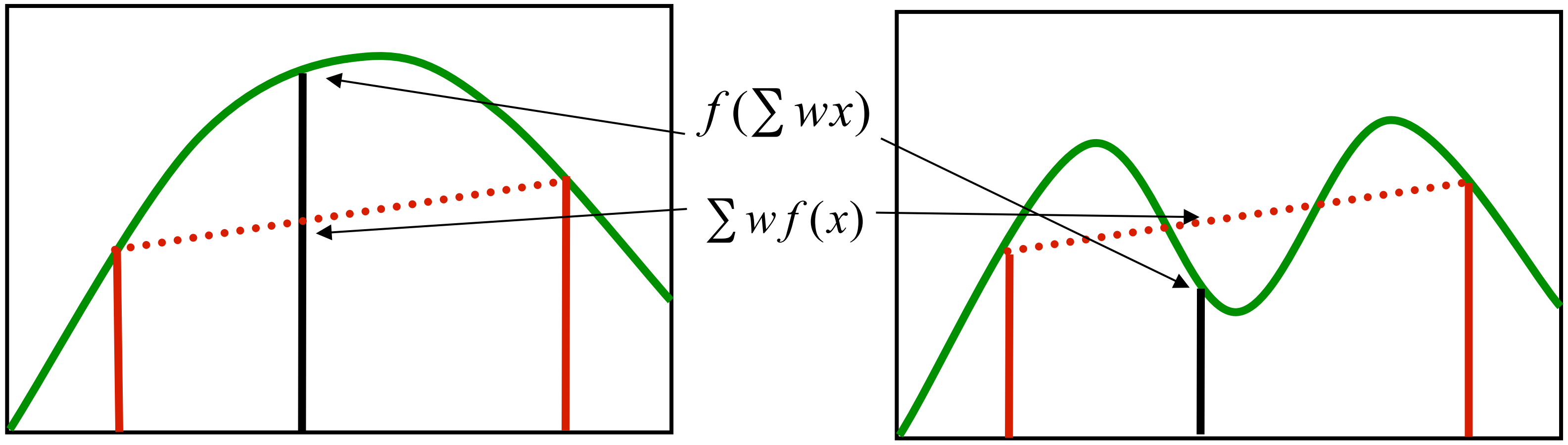
- … and proper nouns are more frequent than common nouns, so we add f_P = {NNP, NNPS}, with E[f_P] =24/36

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

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

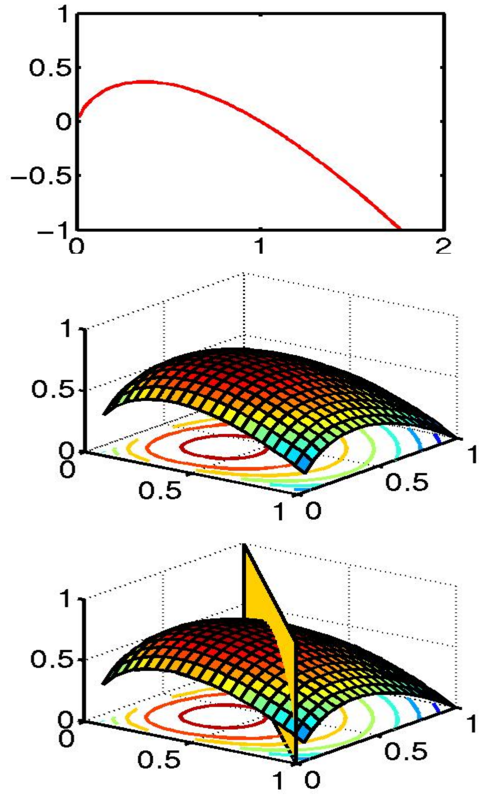


Convex　　　　　　　　Non-Convex

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.



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

	A	a
B	⬛	⬛
b	⬛	⬛

	A	a
B	🟥	
b	🟥	

	A	a
B	🟫	
b	🟫	

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	λ_A	
b	λ_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.

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**

Local Context

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

Feature Interaction

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

Empirical

	A	a
B	1	1
b	1	0

	A	a
B	⬛	⬛
b	⬛	⬛

All = 1

	A	a
B	1/4	1/4
b	1/4	1/4

	A	a
B	0	0
b	0	0

	A	a
B	🟥	
b	🟥	

A = 2/3

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

	A	a
B	λ_A	
b	λ_A	

	A	a
B	🟫	🟫
b		

B = 2/3

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

	A	a
B	$\lambda_A+\lambda_B$	λ_B
b	λ_A	

Feature Interaction

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

Empirical

	A	a
B	1	1
b	1	0

	A	a
B	🟥	
b	🟥	

	A	a
B	🟫	🟫
b		

	A	a
B	🟩	
b		

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	🟦	🟦
b	🟦	

	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**

Conditional Maxent Models for Classification

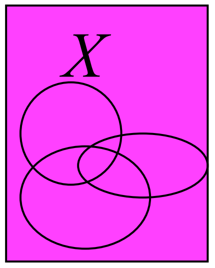
The relationship between conditional and joint maxent/ exponential models

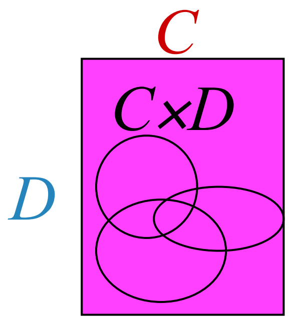
Classification

- What do these joint models of P(*X*) have to do with conditional models P(*C*|*D*)?
- Think of the space *C*×*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*×*D*):

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

- Generally impractical: can't enumerate *X* efficiently.





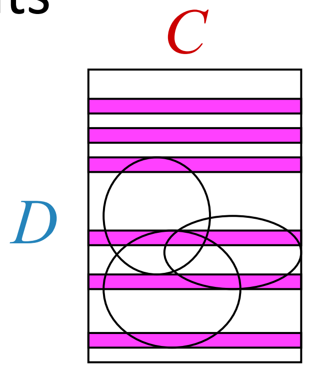
Classification II

- *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?

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

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

$$E(f_i) = \sum_{(c,d)\in(C,D)} P(c,d) f_i(c,d)$$
$$= \sum_{(c,d)\in(C,D) \wedge \hat{P}(d)>0} P(c,d) f_i(c,d)$$



Classification III

- But if we've constrained the *D* marginals

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

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

$$\begin{aligned} P(c,d) &= P(c \mid d)P(d) \\ &= P(c \mid d)\hat{P}(d) \end{aligned}$$

Smoothing/Priors/ Regularization for Maxent Models

Smoothing: Issues of Scale

- Lots of features:
  - NLP maxent models can have ten million features.
  - Even storing a single array of parameter values can have a substantial memory cost.
- Lots of sparsity:
  - Overfitting very easy – we need smoothing!
  - Many features seen in training will never occur again at test time.
- Optimization problems:
  - Feature weights can be infinite, and iterative solvers can take a long time to get to those infinities.

Smoothing: Issues

- Assume the following empirical distribution:

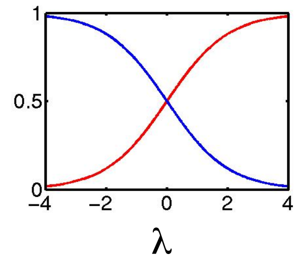
Heads	Tails
h	*t*

- Features: {Heads}, {Tails}
- We'll have the following model distribution:

$$p_{\text{HEADS}} = \frac{e^{\lambda_H}}{e^{\lambda_H} + e^{\lambda_T}} \qquad p_{\text{TAILS}} = \frac{e^{\lambda_T}}{e^{\lambda_H} + e^{\lambda_T}}$$

- Really, only one degree of freedom ($\lambda = \lambda_H - \lambda_T$)

$$p_{\text{HEADS}} = \frac{e^{\lambda_H} e^{-\lambda_T}}{e^{\lambda_H} e^{-\lambda_T} + e^{\lambda_T} e^{-\lambda_T}} = \frac{e^{\lambda}}{e^{\lambda} + e^{0}} \qquad p_{\text{TAILS}} = \frac{e^{0}}{e^{\lambda} + e^{0}}$$



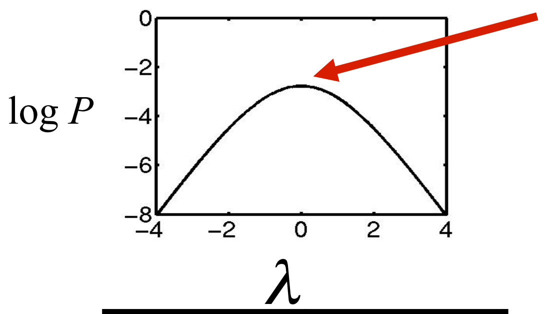
Christopher Manning

Smoothing: Issues

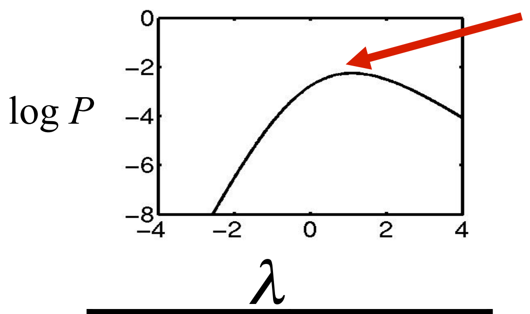
- The data likelihood in this model is:

$$\log P(h,t \mid \lambda) = h \log p_{\text{HEADS}} + t \log p_{\text{TAILS}}$$

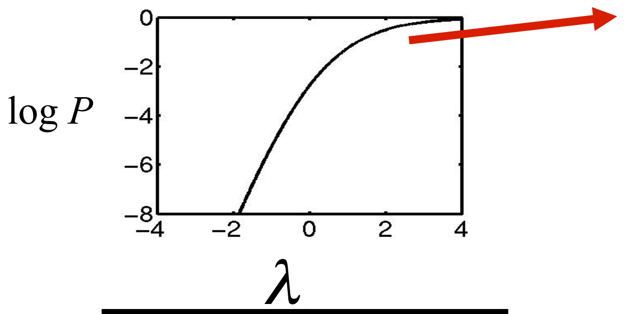
$$\log P(h,t \mid \lambda) = h\lambda - (t+h)\log(1+e^{\lambda})$$



Heads	Tails
2	2



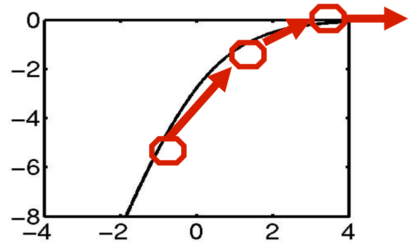
Heads	Tails
3	1



Heads	Tails
4	0

Smoothing: Early Stopping

- In the 4/0 case, there were two problems:
  - The optimal value of λ was ∞, which is a long trip for an optimization procedure
  - The learned distribution is just as spiked as the empirical one – no smoothing
- One way to solve both issues is to just stop the optimization early, after a few iterations:
  - The value of λ will be finite (but presumably big)
  - The optimization won't take forever (clearly)
  - Commonly used in early maxent work
    - Has seen a revival in deep learning ☺



λ

Heads	Tails
4	0

Input

Heads	Tails
1	0

Output

Smoothing: Priors (MAP)

- What if we had a prior expectation that parameter values wouldn't be very large?
- We could then balance evidence suggesting large parameters (or infinite) against our prior.
- The evidence would never totally defeat the prior, and parameters would be smoothed (and kept finite!).
- We can do this explicitly by changing the optimization objective to maximum posterior likelihood:

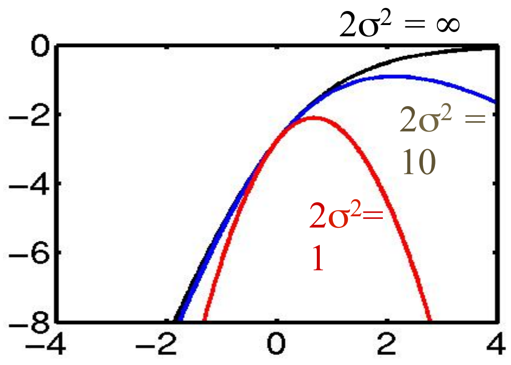
$$\underbrace{\log P(C,\lambda \mid D)}_{\text{Posterior}} = \underbrace{\log P(\lambda)}_{\text{Prior}} + \underbrace{\log P(C \mid D,\lambda)}_{\text{Evidence}}$$

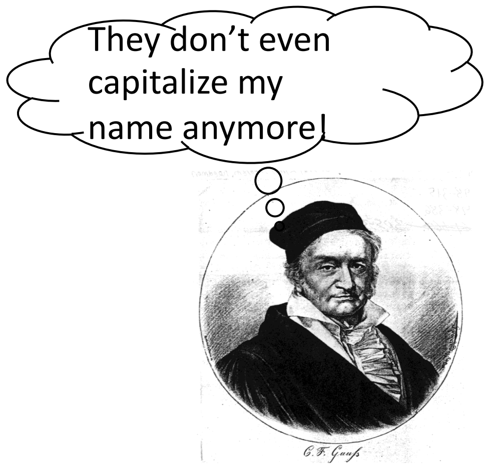
Smoothing: Priors

- Gaussian, or quadratic, or L_2 priors:
  - Intuition: parameters shouldn't be large.
  - Formalization: prior expectation that each parameter will be distributed according to a gaussian with mean μ and variance σ^2.

$$P(\lambda_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{(\lambda_i - \mu_i)^2}{2\sigma_i^2}\right)$$

  - Penalizes parameters for drifting too far from their mean prior value (usually μ=0).
  - $2\sigma^2$=1 works surprisingly well.



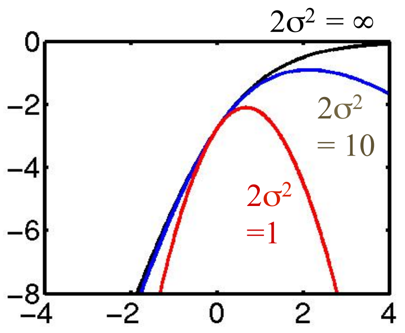


Smoothing: Priors

- If we use gaussian priors:
  - Trade off some expectation-matching for smaller parameters.
  - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
  - Accuracy generally goes up!
- Change the objective:

$$\log P(C,\lambda \mid D) = \log P(C \mid D,\lambda) + \log P(\lambda)$$

$$\log P(C,\lambda \mid D) = \sum_{(c,d)\in(C,D)} P(c \mid d,\lambda) - \sum_i \frac{(\lambda_i - \mu_i)^2}{2\sigma_i^2} + k$$



- Change the derivative:

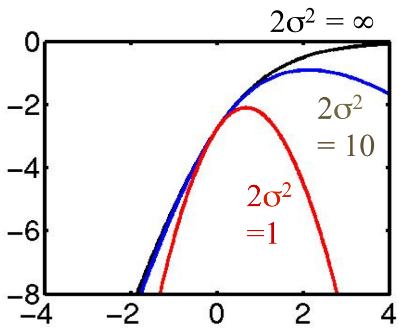
$$\partial \log P(C,\lambda \mid D) / \partial \lambda_i = \text{actual}(f_i, C) - \text{predicted}(f_i,\lambda) - (\lambda_i - \mu_i)/\sigma^2$$

Smoothing: Priors

- If we use gaussian priors:
  - Trade off some expectation-matching for smaller parameters.
  - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
  - Accuracy generally goes up!
- Change the objective:

$$\log P(C,\lambda \mid D) = \log P(C \mid D,\lambda) + \log P(\lambda)$$

$$\log P(C,\lambda \mid D) = \sum_{(c,d)\in(C,D)} P(c \mid d,\lambda) - \sum_i \frac{\lambda_i^2}{2\sigma_i^2} + k$$



- Change the derivative:

$$\partial \log P(C,\lambda \mid D)/\partial\lambda_i = \text{actual}(f_i,C) - \text{predicted}(f_i,\lambda) - \lambda_i/\sigma^2$$

Taking prior mean as 0

Example: NER Smoothing

Because of smoothing, the more common prefix and single-tag features have larger weights even though entire-word and tag-pair features are more specific.

Local Context

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

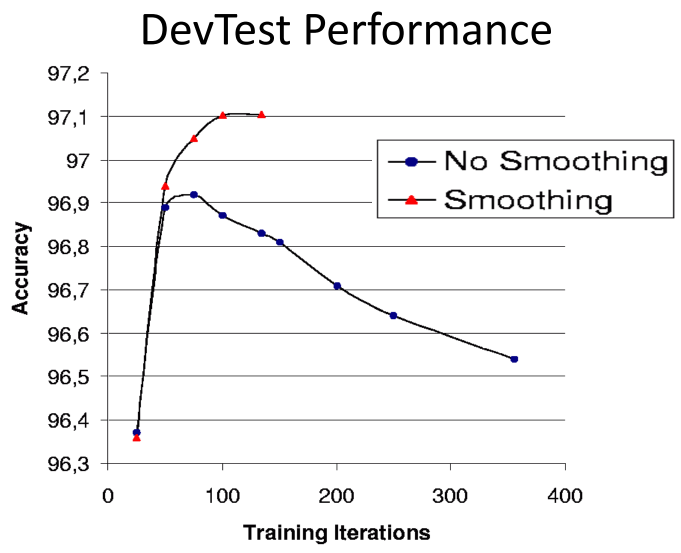
Feature Weights

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**

Example: POS Tagging

- From (Toutanova et al., 2003):

	Overall Accuracy	Unknown Word Acc
Without Smoothing	96.54	85.20
With Smoothing	97.10	88.20



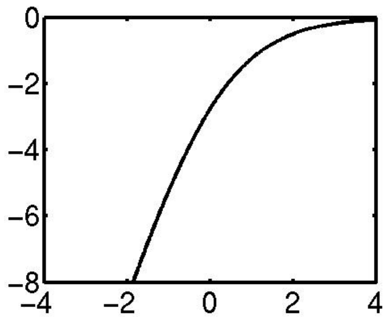
- Smoothing helps:
  - Softens distributions.
  - Pushes weight onto more explanatory features.
  - Allows many features to be dumped safely into the mix.
  - Speeds up convergence (if both are allowed to converge)!

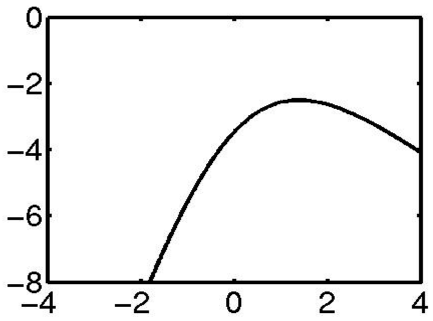
Smoothing: Regularization

- Talking of “priors” and “MAP estimation” is Bayesian language
- In frequentist statistics, people will instead talk about using “regularization”, and in particular, a gaussian prior is “L_2 regularization”
- The choice of names makes no difference to the math
- Recently, L_1 regularization is also very popular
  - Gives sparse solutions – most parameters become zero [Yay!]
  - Harder optimization problem (non-continuous derivative)

Smoothing: Virtual Data

- Another option: smooth the data, not the parameters.
- Example:





Heads	Tails
4	0

Heads	Tails
5	1

  - Equivalent to adding two extra data points.
  - Similar to add-one smoothing for generative models.
- For feature-based models, hard to know what artificial data to create!

Smoothing: Count Cutoffs

- In NLP, features with low empirical counts are often dropped.
  - Very weak and indirect smoothing method.
  - Equivalent to locking their weight to be zero.
  - Equivalent to assigning them gaussian priors with mean zero and variance zero.
  - Dropping low counts does remove the features which were most in need of smoothing…
  - … and speeds up the estimation by reducing model size …
  - … but count cutoffs generally hurt accuracy in the presence of proper smoothing.
- Don't use count cutoffs unless necessary for memory usage reasons. Prefer L_1 regularization for finding features to drop.

Smoothing/Priors/ Regularization for Maxent Models