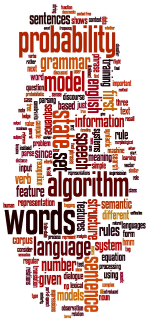


Maximum Entropy Models / Softmax Classifiers

Christopher Manning

Christopher Manning

Today's plan

1. Generative vs. discriminative models [15 mins]
2. Optimizing softmax/maxent model parameters [20 mins]
3. Named Entity Recognition [10 mins]
4. Maximum entropy sequence models [10 mins]

21

Generative vs. Discriminative models

Christopher Manning

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

---

Christopher Manning

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

  $P(c,d)$

  - They generate the observed data from the hidden stuff
  - All the classic 1990s StatNLP models:
    - *n*-gram language models, Naive Bayes classifiers, hidden Markov models, probabilistic context-free grammars, IBM machine translation alignment models

---

Christopher Manning

Joint vs. Conditional Models

- Discriminative (conditional) models take the data as given, and put a probability/score over hidden structure given the data:

  $P(c|d)$

  - Logistic regression, maximum entropy models, conditional random fields
  - Also, SVMs, (averaged) perceptron, feed forward neural networks, etc. are discriminative classifiers
    - but not directly probabilistic

Christopher Manning

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 give maximum joint likelihood on categorical data
- A *conditional* model gives probabilities P(*c*|*d*). It 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

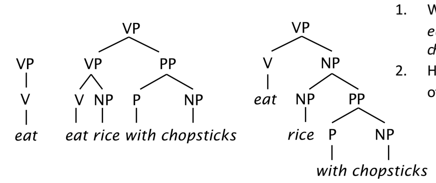
Test Set	
Objective	Accuracy
Joint Like.	73.6
Cond. Like.	76.1

- 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)

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

Christopher Manning

PCFGs Maximize Joint, not Conditional Likelihood



46 6 2

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

Based on an example by Mark Johnson

Optimizing softmax/maxent model parameters

Their likelihood and derivatives

Christopher Manning

Background: Feature Expectations

- We will crucially make use of two *expectations*
  - actual and predicted counts of a feature firing:
- Empirical expectation (count) 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:

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

Christopher Manning

Maxent/Softmax Model Likelihood

- Maximum (Conditional) Likelihood Models
  - Given a model form, we choose values of parameters λ_i to maximize the (conditional) likelihood of the data.
- For any given feature weights, we can calculate:
  - Conditional likelihood of training data

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

  - Derivative of the likelihood wrt each feature weight

Christopher Manning

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:

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

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

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

*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.

Christopher Manning

The Likelihood Value

- We can separate this into two components:

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

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

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

Christopher Manning

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)

Christopher Manning

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, λ)

Christopher Manning

The Derivative III

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

- 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: $E_p(f_j) = E_{\tilde{p}}(f_j), \forall j$

Christopher Manning

Finding the optimal parameters

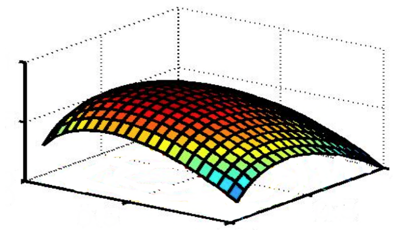
- We want to choose parameters $\lambda_1, \lambda_2, \lambda_3, \ldots$ that maximize the conditional log-likelihood of the training data

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

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

Christopher Manning

A likelihood surface



Christopher Manning

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)
   - Improved variants like Adagrad, Adadelta, RMSprop, NAG
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

Christopher Manning

Named Entity Recognition (NER)

- A very important NLP 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.

Christopher Manning

Named Entity Recognition (NER)

- A very important NLP 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.

Christopher Manning

Named Entity Recognition (NER)

- A very important NLP 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

The Named Entity Recognition Task

We	ORG	O
should	ORG	O
show	O	O
Neha	PER	B-PER
Eric	PER	B-PER
King	PER	I-PER
's	O	O
assignment	ORG	O
:	:	:

BIO/IOB notation

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 false positive and a false negative
- 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) a.k.a. Conditional Markov models

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

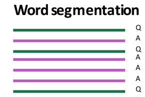
POS tagging

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

Christopher Manning

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.)

Christopher Manning

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

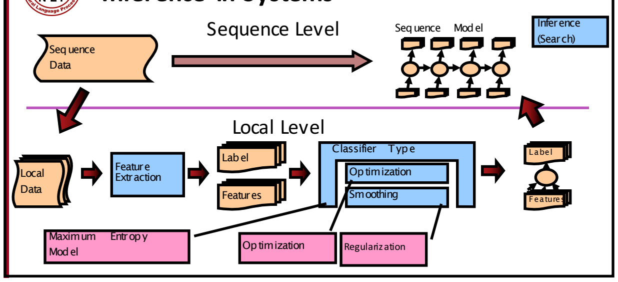
-3	-2	-1	0	+1
ORG	ORG	O	???	???
Xerox	Corp.	fell	22.6	%

Features

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

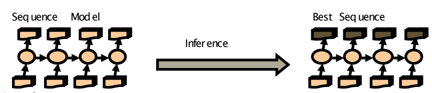
Christopher Manning

Inference in Systems



Christopher Manning

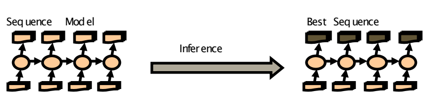
Greedy Inference



- 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

Christopher Manning

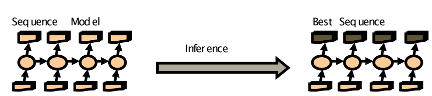
Beam Inference



- 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.

Christopher Manning

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).

Christopher Manning

CRFs [Lafferty, Pereira, and McCallum 2001]

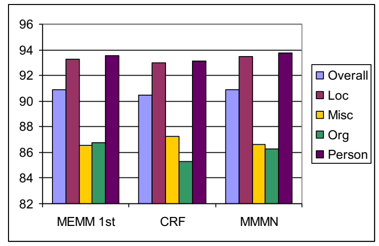
- Another sequence model: Conditional Random Fields (CRFs)
- A whole-sequence conditional model rather than a chaining of local models.

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

- 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.

Christopher Manning

CoNLL 2003 NER shared task Results on English Devset



Smoothing/Priors/ Regularization for Maxent Models

Christopher Manning

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.

Christopher Manning

Smoothing: Issues

- Assume the following empirical distribution:

Heads	Tails
h	t

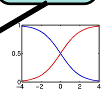
- Features: {Heads}, {Tails}
- We'll have the following softmax model distribution:

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

Logistic regression!

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

$$p_{\text{HEADS}} = \frac{e^{\lambda_H} e^{-\lambda_T}}{e^{\lambda_H} e^{-\lambda_T} + e^{\lambda_T} e^{-\lambda_T}} = \frac{e^{\lambda}}{e^{\lambda} + e^{0}} = \frac{e^{\lambda}}{e^{\lambda} + 1} \qquad p_{\text{TAILS}} = \frac{e^{0}}{e^{\lambda} + e^{0}} = \frac{1}{1 + e^{\lambda}}$$



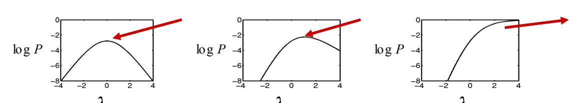
Christopher Manning

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

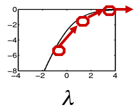
Heads	Tails
3	1

Heads	Tails
4	0

Christopher Manning

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

Christopher Manning

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:

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

Posterior Prior Evidence

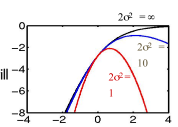
Christopher Manning

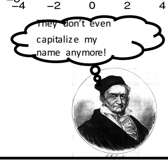
Smoothing: Priors

- 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.





Christopher Manning

Smoothing: Priors

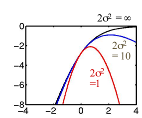
- If we use gaussian priors / L_2 regularization:
  - Trade off some expectation-matching for smaller parameters.
  - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
  - Accuracy generally goes up!
- Change the objective:

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

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

- Change the derivative:

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



Christopher Manning

Smoothing: Priors

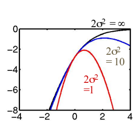
- If we use gaussian priors / L_2 regularization :
  - Trade off some expectation-matching for smaller parameters.
  - When multiple features can be recruited to explain a data point, the more common ones generally receive more weight.
  - Accuracy generally goes up!
- Change the objective:

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

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

- 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

Christopher Manning

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.

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

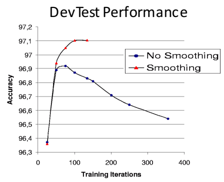
	Prev	Cur	Next
State	Other	???	???
Word	at	Grace	Road
Tag	IN	NNP	NNP
Sig	x	Xx	Xx

Christopher Manning

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)!

Christopher Manning

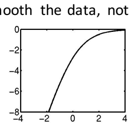
Smoothing / Regularization

- 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)

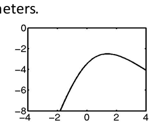
Christopher Manning

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!

Christopher Manning

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