



Smoothing/Priors/ Regularization for Maxent Models



Smoothing: Issues of Scale

- Lots of features:
 - NLP maxent models can have well over a 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:

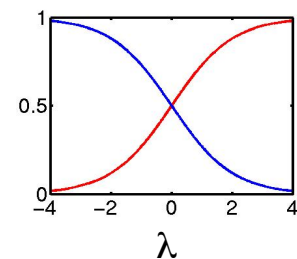
Heads	Tails
<i>h</i>	<i>t</i>

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

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

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

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



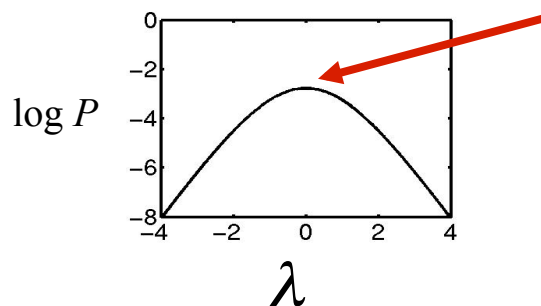


Smoothing: Issues

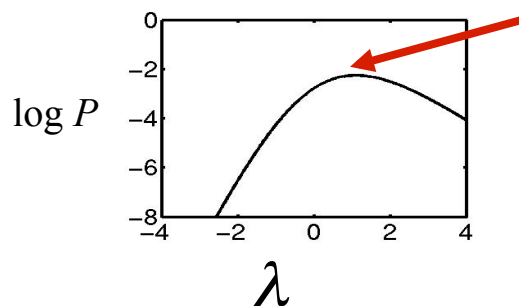
- The data likelihood in this model is:

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

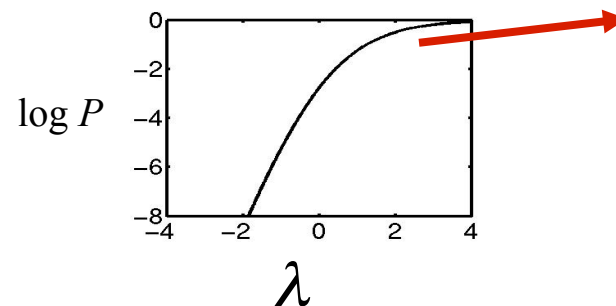
$$\log P(h, t \mid \lambda) = h\lambda - (t + h) \log(1 + e^\lambda)$$



Heads	Tails
2	2



Heads	Tails
3	1

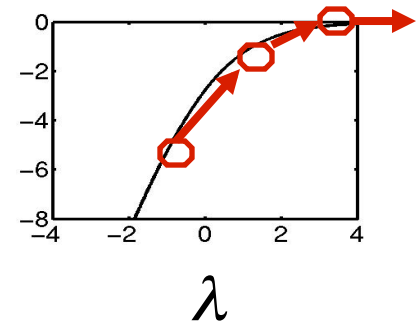


Heads	Tails
4	0



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.



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:

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

Posterior

Prior

Evidence

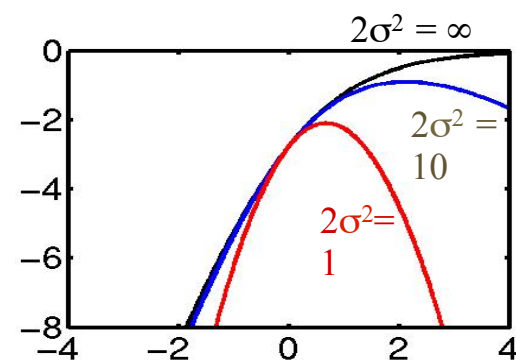


Smoothing: Priors

- 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 to far from their mean prior value (usually $\mu=0$).
- $2\sigma^2=1$ works surprisingly well.



They don't even capitalize my name anymore!





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!

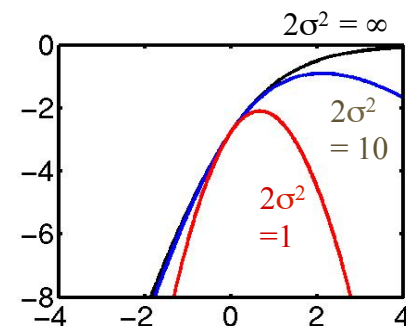
- Change the objective:

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

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

- Change the derivative:

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





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!

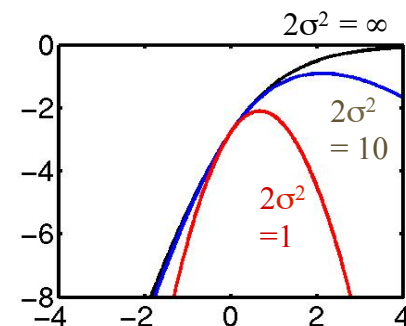
- Change the objective:

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

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

- Change the derivative:

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



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

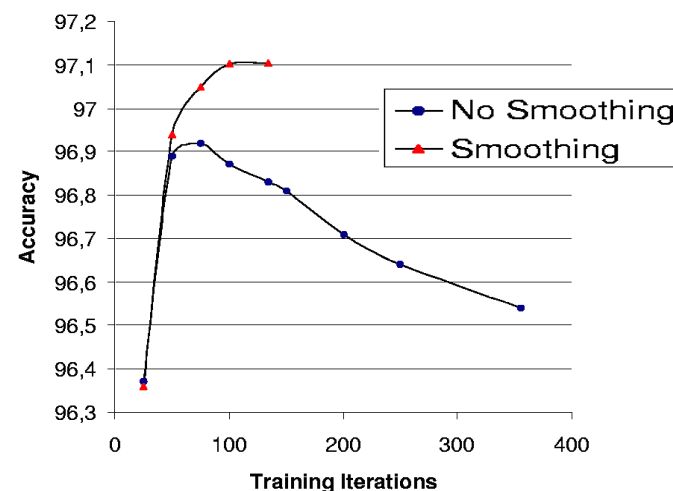


Example: POS Tagging

- 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

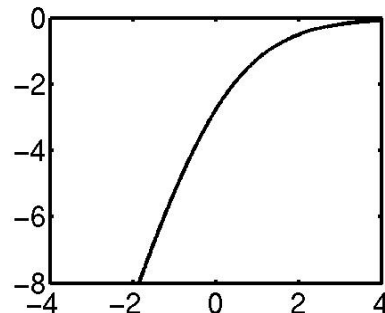
- Talking of “priors” and “MAP estimation” is Bayesian language
- In frequentist statistics, people will instead talk about using “regularization”, and in particular, a gaussian prior is “ L_2 regularization”
- The choice of names makes no difference to the math



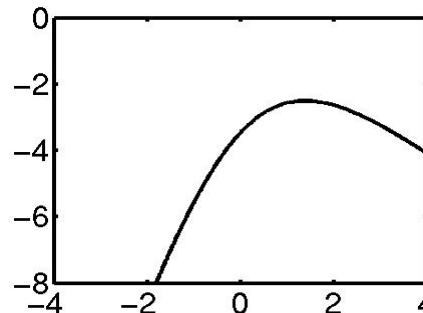
Smoothing: Virtual Data

- 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.
- 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.
- We recommend: don't use count cutoffs unless absolutely necessary for memory usage reasons.



Smoothing/Priors/ Regularization for Maxent Models