Modeling Learning Trajectories with Batch Gradient Descent

Joe Pater and Robert Staubs

Department of Linguistics

NECPhon, MIT, 10/26/2013

Overview

1. A gradual batch model: replication of Jaeger's (2007) replication of Boersma and Levelt (2000)
2. Comparison with on-line models
3. Derivation of MaxEnt gradient descent
4. Regularization and hidden structure

Gradient descent: a gradual batch model

- There seems to be a general assumption in the phonological literature that 'batch' implies 'non-gradual'.
- From our soon-to-be-revised unpublished ms.:

'Another important avenue for further research is the development of on-line learning algorithms in this framework. Such methods may be more cognitively plausible than the batch method we have explored and would certainly be more useful in modeling the course of human language acquisition.'

Gradient descent: a gradual batch model

- In using gradient ascent/descent for the modeling of human learning trajectories, we are following a suggestion of Goldwater and Johnson's (2003):

'Gradient ascent is a popular but not very efficient optimization algorithm which may produce human-like learning curves, although we have not investigated this here'

Gradient descent: a gradual batch model

- In this we are preceded by Jaeger (2007), who shows that a sampling version of Stochastic Gradient Ascent is nearly identical to Boersma's (1998) Gradual Learning Algorithm for Stochastic OT.
- Jaeger's (2007) model samples from the training data (on-line/'Stochastic') and from the probability distribution defined by the grammar ('sampling version')
- The version of gradient descent that we are using eliminates both of these kinds of sampling, which has great practical consequences for the modeler (see also Moreton, Pater and Pertsova 2013)

Gradient descent: a gradual batch model

- We'll start with the familiar on-line update (=HG-GLA, Perceptron), and show how the two kinds of sampling can be straightforwardly eliminated in a MaxEnt model
- The update takes the difference between the vectors of constraint violations of two representations, and scales it by the learning rate (multiplies it by a small number)
- Constraints, violations with negative integers, and difference vector:

		NoCoda	Faith
Training data (Winner)	/bait/, [bait]	–1	
Learner's grammar (Loser)	/bait/, [bai]		–1
Difference (T - L)		–1	+1

Gradient descent: a gradual batch model

- Jaeger (2007) gets the output of the learner's grammar, a 'Loser', by sampling. We instead take the probability weighted sum of violation vectors to represent the learner's expectation.

/bait/	NoCoda 6	Faith 4	H	e^H	p
[bait]	–1		–6	0.002	0.119
[bai]		–1	–4	0.018	0.881
	–0.119	–0.881			

Gradient descent: a gradual batch model

- Jaeger (2007) also gets the training datum, a 'Winner', by sampling. We instead take the probability weighted sum of violation vectors to represent the training data.

/bait/		NoCoda	Faith
[bait]	0.9	–1	
[bai]	0.1		–1
		–0.9	–0.1

Gradient descent: a gradual batch model

- And then we just do the same update, but now it's over all of the training data, and over the learner's entire expectation, rather over a sample of each.

	NoCoda	Faith
Training data	–0.9	–0.1
Learner's grammar	–0.119	–0.881
Difference (T - L)	–0.781	+0.781

Gradient descent: a gradual batch model

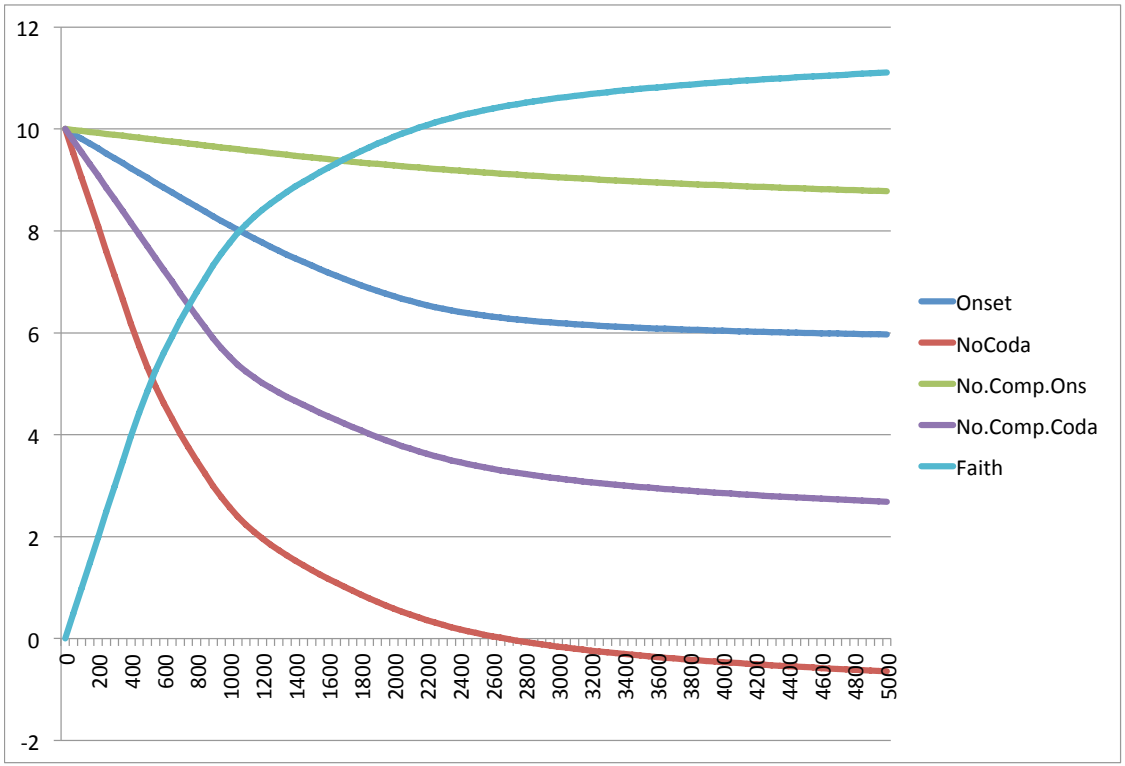
- There's also no need to sample from the set of tableaux. We again take the probability weighted sum, for both the training data and the learner's expectation.
- Probabilities of syllable types in Dutch (Boersma and Levelt 2000 via Jaeger 2007)

CV	44.81%	CCVC	1.98%
CVC	32.05%	CCV	1.38%
VC	11.99%	VCC	0.42%
V	3.85%	CCVCC	0.26%
CVCC	3.25%		

Tab. 1: Frequency of syllable types in Dutch

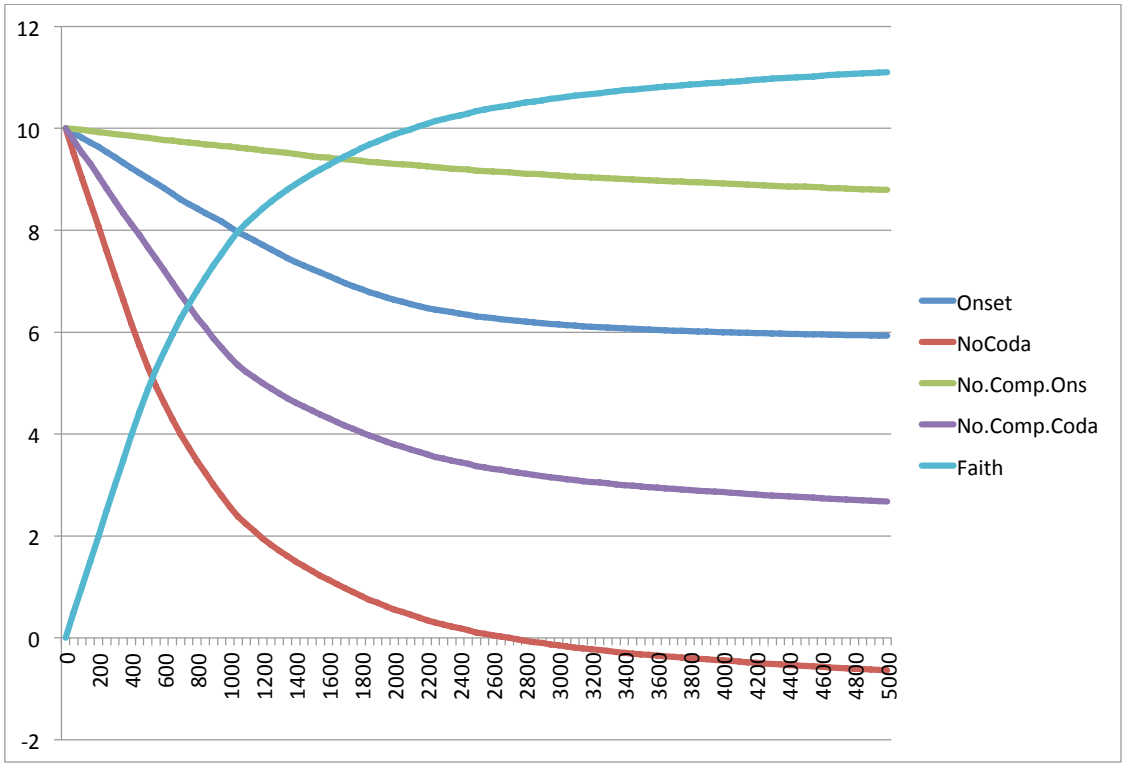
Gradient descent: a gradual batch model

Weights of 5 syllable structure constraints and Faith over 'time'



Comparison with on-line models

Here's the average of 10 runs of Stochastic Gradient Descent, with sampling from probabilities within and over tableaux in the training data (but not within the tableaux in the learner's expectation).



Where's the gradient?

- Gradient descent finds a minimum according to some loss function
- The gradient we use here corresponds to negative log likelihood loss
- The exponential form of MaxEnt grammar gives a simple gradient analogous to the perceptron rule or HG-GLA (Boersma & Pater 2014)

Deriving the gradient

Start with form of likelihood, applying properties of logarithms.

$$\begin{aligned}
\frac{\partial}{\partial w} \log \mathcal{L}(w; y) &= \frac{\partial}{\partial w} \log \prod_i P(y_i|w) \\
&= \frac{\partial}{\partial w} \sum_i \log P(y_i|w) \\
&= \frac{\partial}{\partial w} \sum_i \log \frac{e^{w^T v_i}}{\sum_j e^{w^T v_j}} \\
&= \sum_i \left[\frac{\partial}{\partial w} \log e^{w^T v_i} - \frac{\partial}{\partial w} \log \sum_j e^{w^T v_j} \right]
\end{aligned}$$

Gradient breaks into a numerator term and denominator term. The denominator contains a model probability in it.

$$\frac{\partial}{\partial w} \log \mathcal{L}(w; y) = \sum_i \left[v_i - \frac{\frac{\partial}{\partial w} \sum_j e^{w^T v_j}}{\sum_j e^{w^T v_j}} \right]$$

$$= \sum_i \left[v_i - \sum_j \frac{v_j e^{w^T v_j}}{\sum_j e^{w^T v_j}} \right]$$

$$= \sum_i \left[v_i - \sum_j v_j P(y_j | w) \right]$$

The denominator term has the form of an expectation.

$$\frac{\partial}{\partial w} \log \mathcal{L}(w; y) = \sum_i \left[v_i - \sum_j v_j P(y_j|w) \right]$$
$$= \sum_i v_i - E_P[v_j]$$

- As the examples we started with showed, gradient descent can handle non-categorical distributions
- K-L divergence loss (error minimization) is a generalization of negative log likelihood loss

Regularization

- Regularization imposes restrictions on solutions beyond fit to data
- This type of restriction is implemented slightly differently in different types of learning:
  1. Priors for learning from likelihood
  2. Regularization terms for batch learning by maximum likelihood
  3. Decay terms for gradual learning

$$\mathcal{L}(w; y) \propto \prod_i e^{-\frac{(w_i - \mu_i)^2}{2\sigma_i^2}} \quad \text{Gaussian prior} \quad (1)$$

$$\log \mathcal{L}(w; y) \propto -\sum_i \frac{(w_i - \mu_i)^2}{2\sigma_i^2} \quad \text{L2 regularization} \quad (2)$$

$$\frac{\partial}{\partial w_j} \log \mathcal{L}(w; y) \propto -\frac{(w_j - \mu_j)}{\sigma_j^2} \quad \text{decay} \quad (3)$$

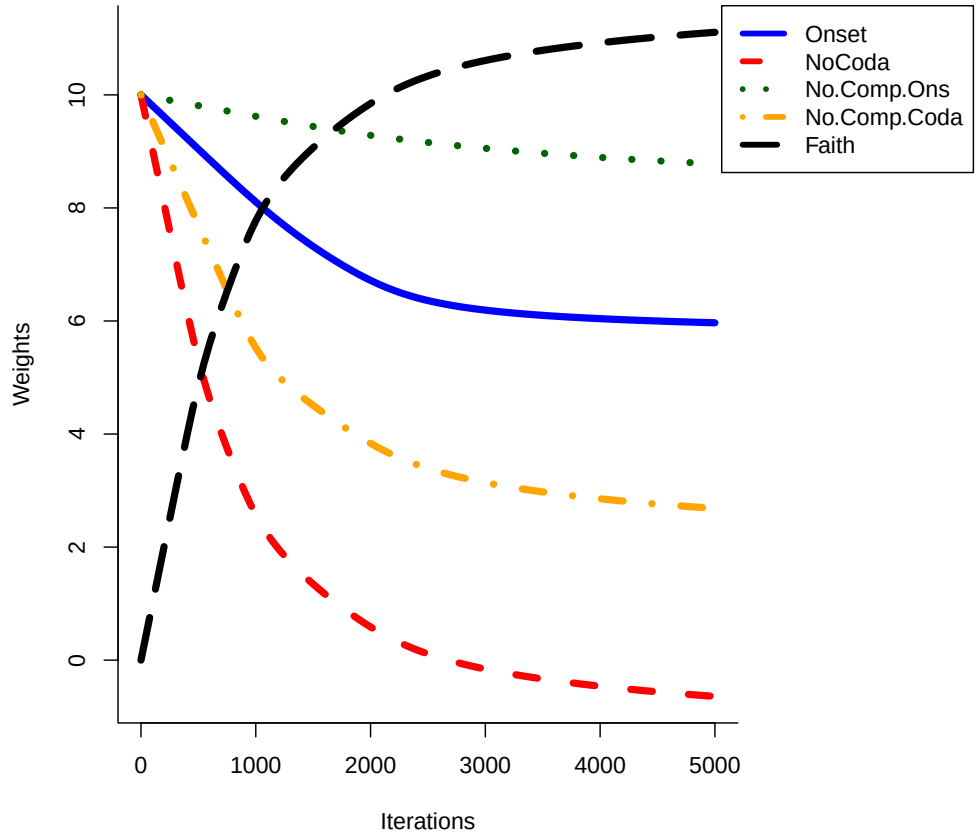
Encourages weights close to set values – penalizes sum of squared differences.

$$\mathcal{L}(w; y) \propto \prod_i e^{-\frac{|w_i - \mu_i|}{\lambda}} \quad \text{Laplacian prior} \quad (4)$$

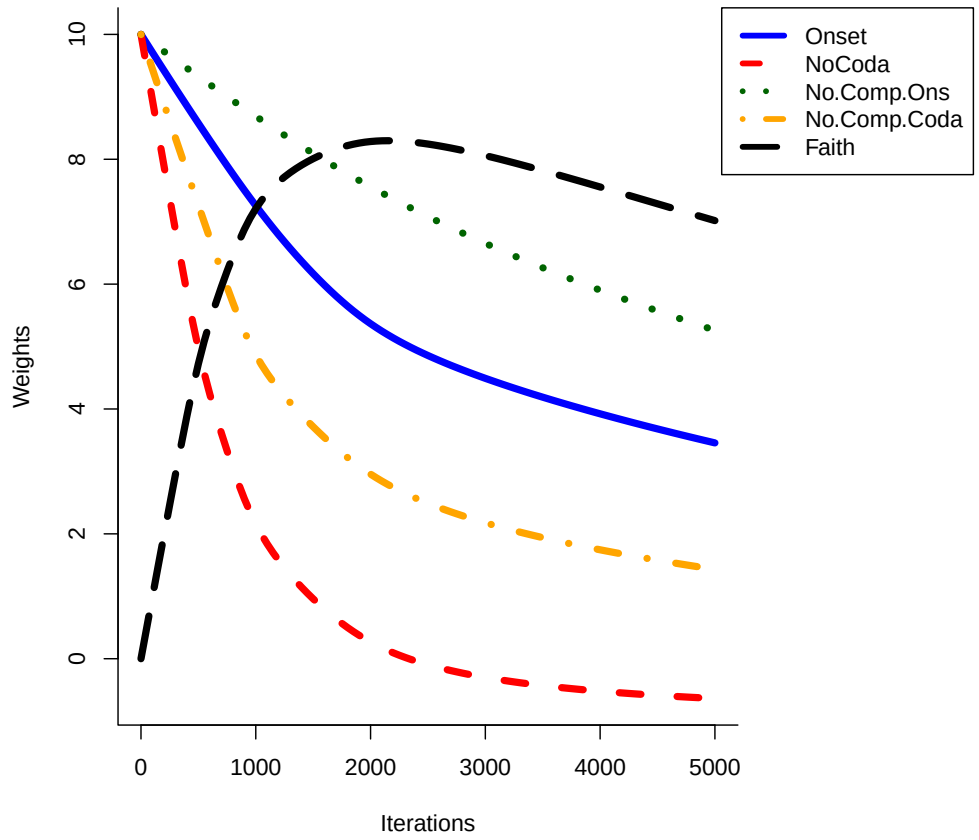
$$\log \mathcal{L}(w; y) \propto -\sum_i \frac{|w_i - \mu_i|}{\lambda_i} \quad \text{L1 regularization} \quad (5)$$

$$\frac{\partial}{\partial w_j} \log \mathcal{L}(w; y) \propto -\lambda_j \operatorname{sgn}(w_j) \quad \text{decay} \quad (6)$$

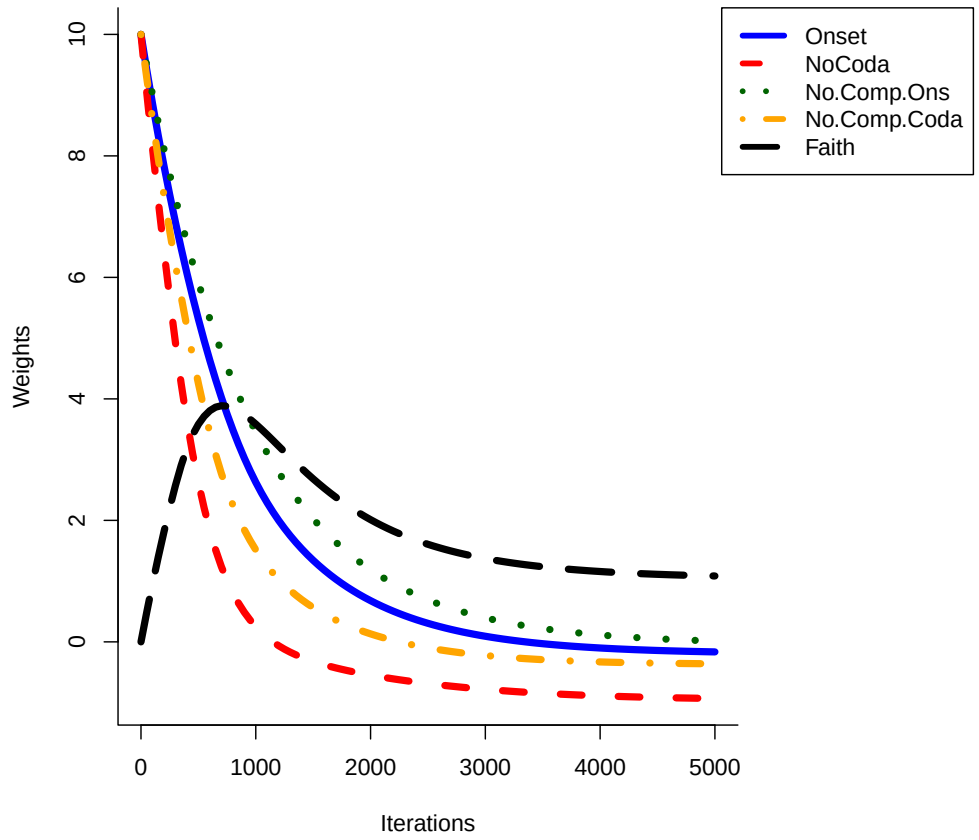
Encourages weights to equal set values – penalizes sum of absolute differences.



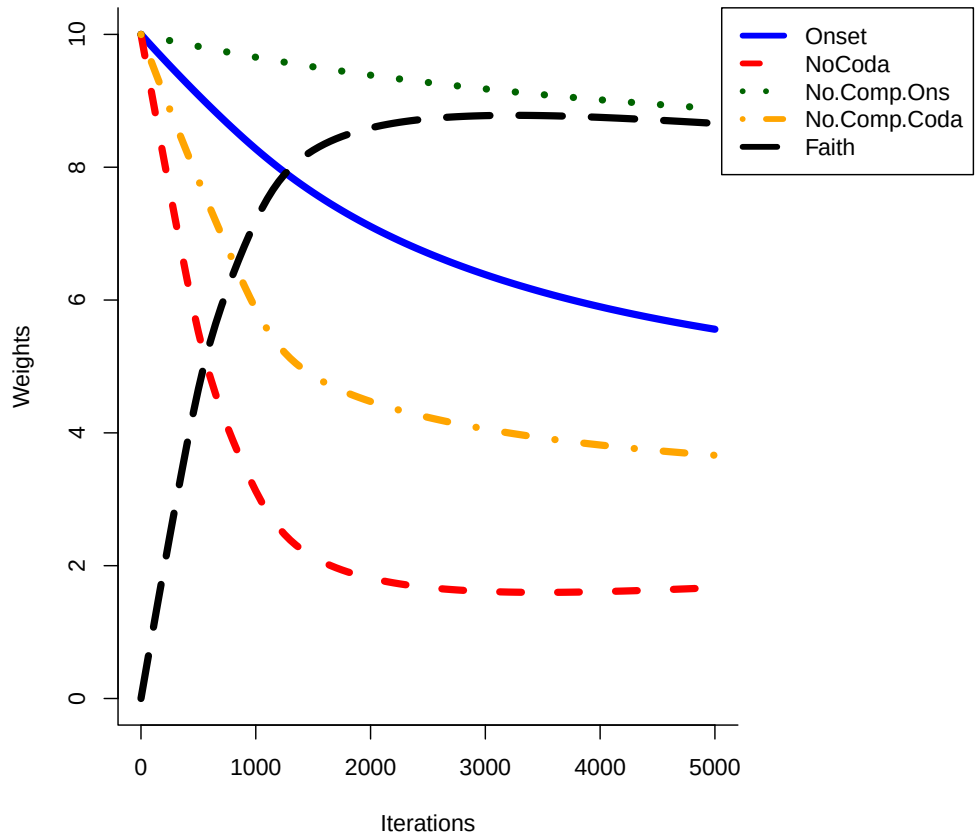
Dutch problem with no regularization.



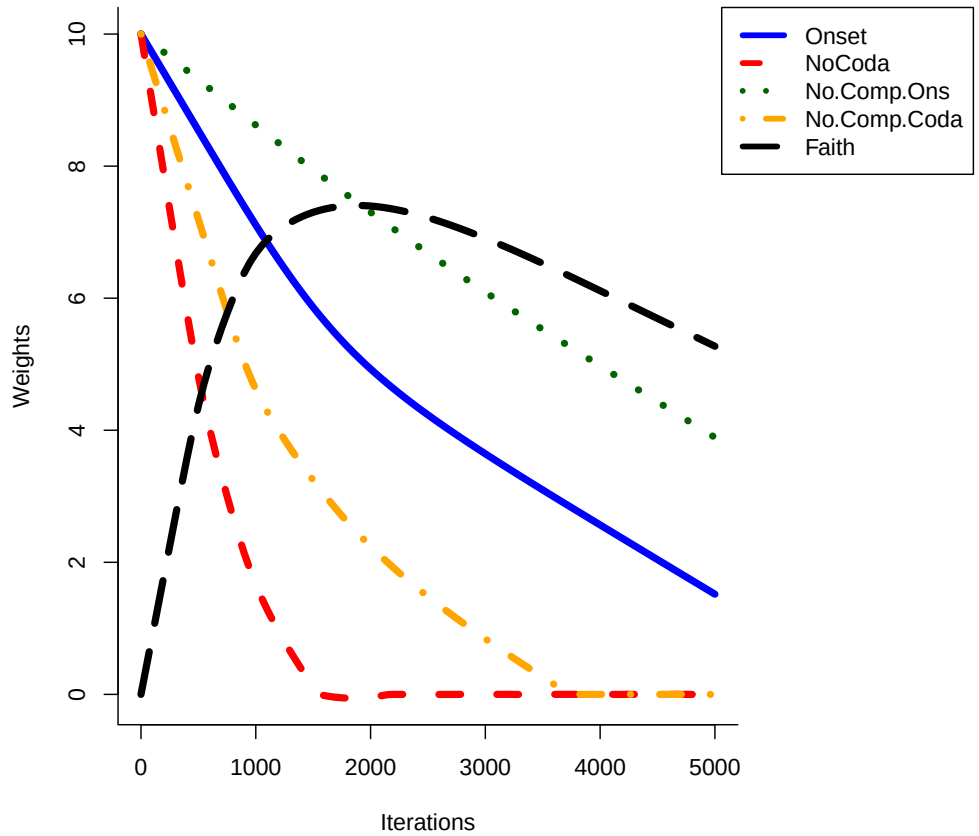
Dutch problem with L2 regularization, $\sigma^2 = 100$, $\mu = 0$. Lower weights.



Dutch problem with L2 regularization, $\sigma^2 = 10$, $\mu = 0$. Lower weights.



Dutch problem with L2 regularization, $\sigma^2 = 50$, $\mu = 0$ for Faithfulness, $\mu = 10$ for Markedness.



Dutch problem with L1 regularization, $\lambda = 0.1$, $\mu = 0$. More sparsity (zero weights).

Hidden structure

- The probability of an overt structure y is the sum of the probabilities of the hidden structures z that correspond to it (e.g. Eisenstat NECPhon 2; Pater et al. NECPhon 3; Pater et al. 2012)
- Recall: the gradient without hidden structure is the difference between the violations of the observed forms and the expected violations under the model

- With hidden structure the observed violations are not known
- One strategy is to use the expected violations of hidden structures for the observed structure under the model's probabilities

No hidden structure:

$$\sum_i v_i - E_P[v_j]$$

Hidden structure:

$$\sum_i E_P[v|y_i] - E_P[v_j]$$

This approach thus has the form of Expectation-Maximization.

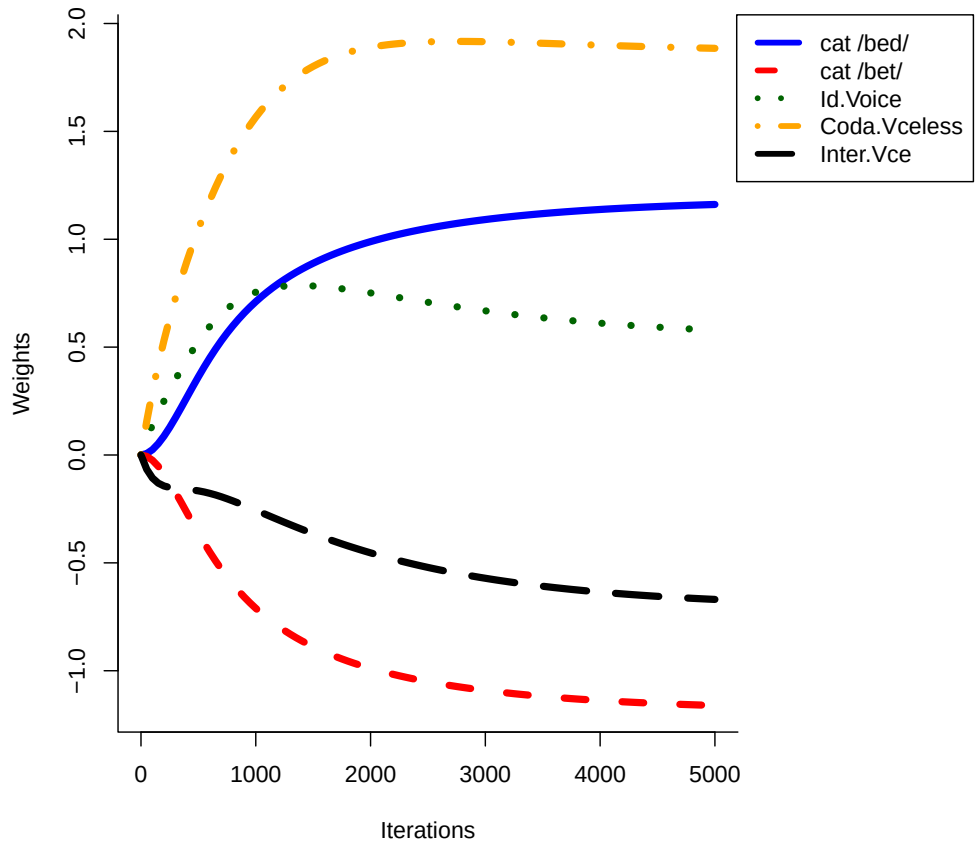
We replicate an example from Pater et al. (2012) of a language with final devoicing.

	SR	**Meaning**
a.	[bet]	cat
b.	[beda]	cats
c.	[mot]	dog
d.	[mota]	dogs

The hidden structure here is the assumed UR for the alternating word.

	UR options	SR	Meaning
a.	/bed/ /bet/	[bet]	cat
b.	/bed+a/ /bet+a/	[beda]	cats
c.	/mot/	[mot]	dog
d.	/mot+a/	[mota]	dogs

These URs are controlled by constraints which motivate their choice.



Devoicing problem with L2 regularization, $\sigma^2 = 10$, $\mu = 0$.
Starting weights at 0.

- Hidden structure introduces non-convexity—local minima—to the learning problem
- Different starting weights may end up in different optima
- Gradient descent for hidden structure still needs some sampling—but only over starting weights (or some other method for avoiding local minima)

Conclusions

- Why use a batch model?
- Because there is no sampling, there is a single learning curve given an initial set of weights
- An on-line model requires multiple runs and averaging to get a representative learning curve
- Multiple runs and averaging are inconvenient, and require a decision on an appropriate number of runs

Conclusions

- Why use on-line learning?
- Because you know the order of the data, and want to model its effects (though a sliding batch window is also a possibility)
- Because your problem is very large, in which case sampling can be more efficient
- Or because you need to sample because you aren't using MaxEnt
- Cognitive plausibility arguments are questionable

Resources

Scripts and example files to be available at Harmonic Grammar in R site:
`http://blogs.umass.edu/hgr/`

Software for Elliott Moreton's related weightless MaxEnt algorithm:
`http://www.unc.edu/~moreton/Software/Weightless/`