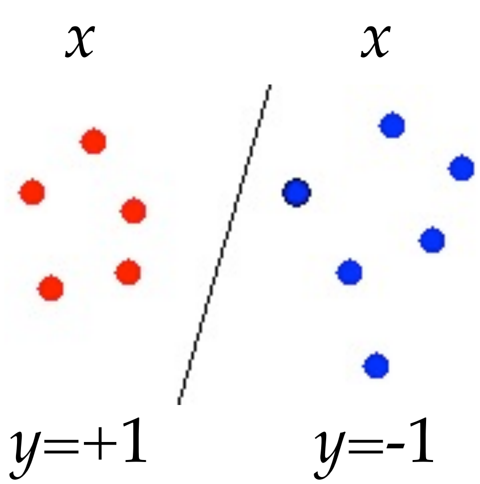
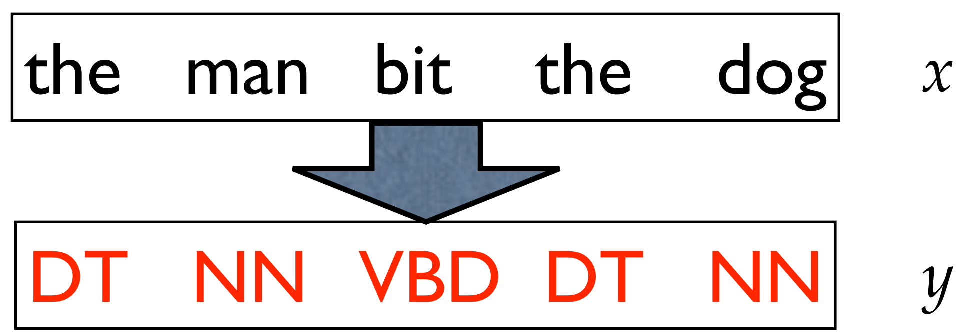
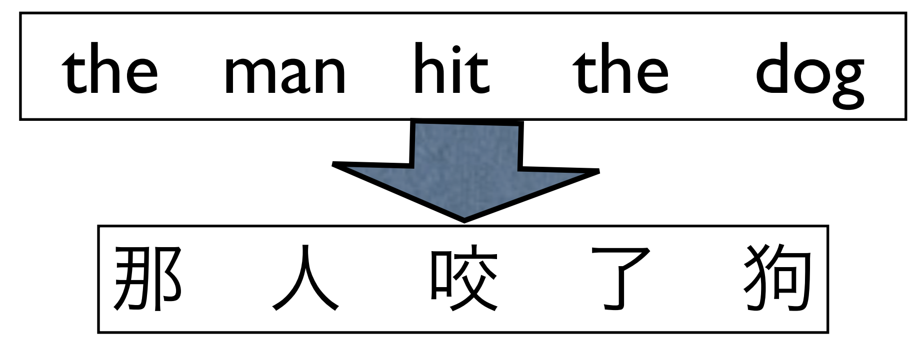
Structured Learning with Inexact Search







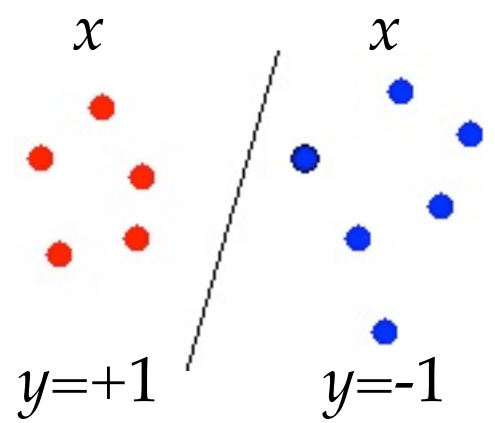
Liang Huang

The City University of New York (CUNY)

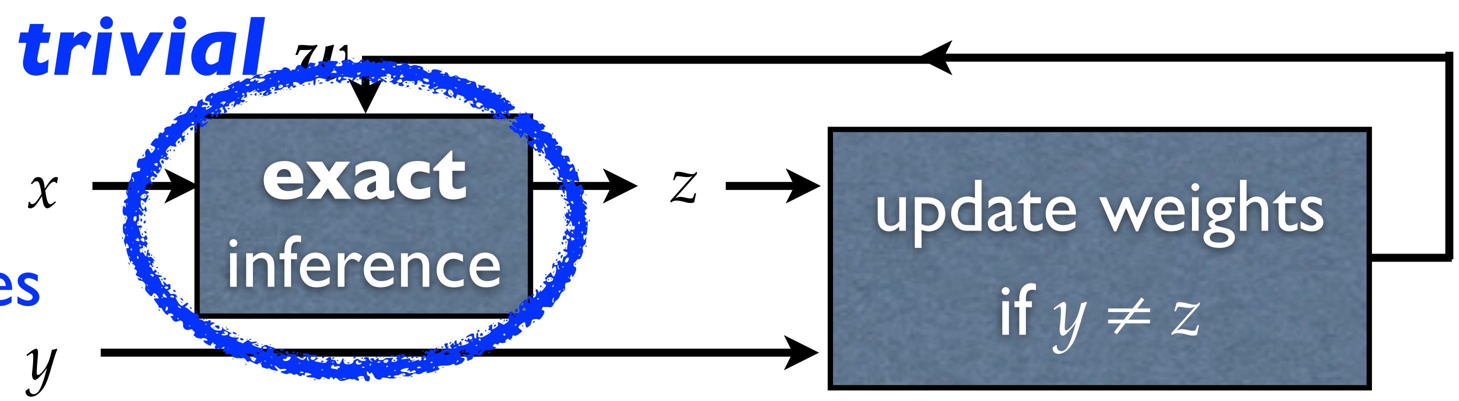
includes joint work with S. Phayong, Y. Guo, and K. Zhao

Structured Perceptron (Collins 02)

binary classification

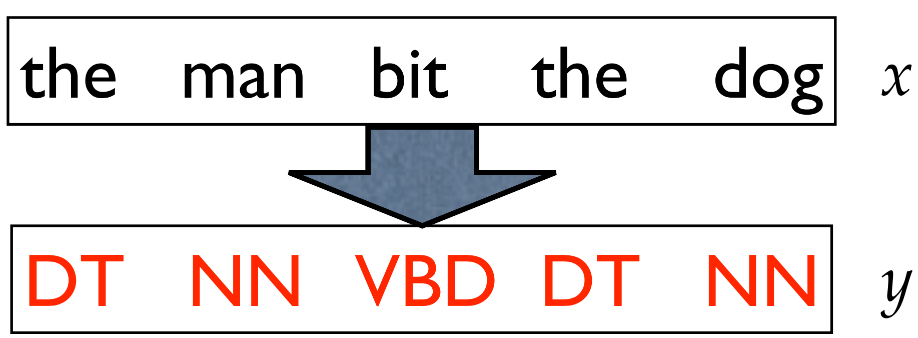


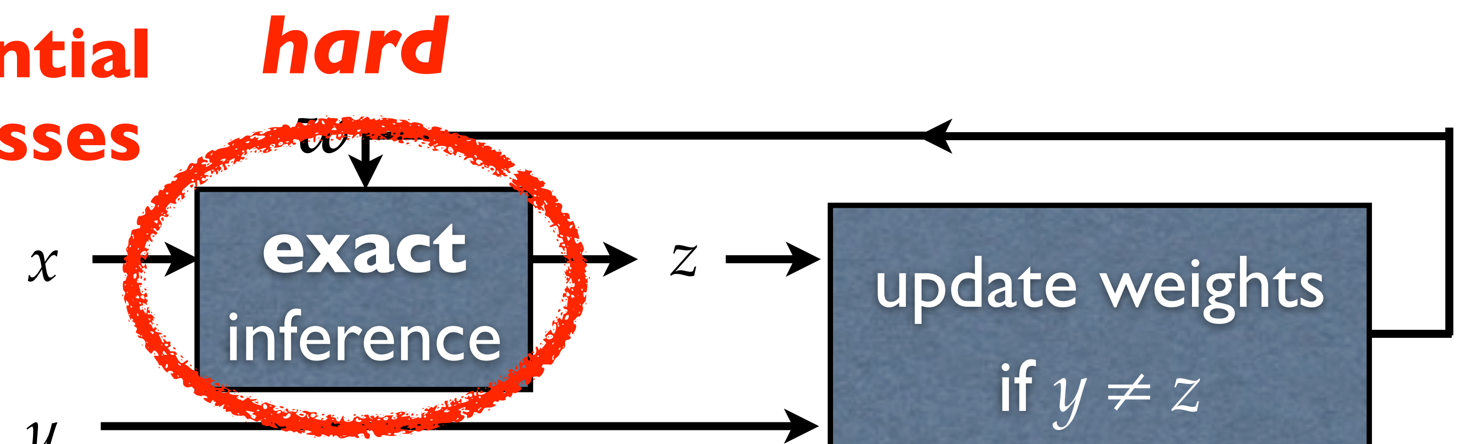
constant
# of classes



structured classification

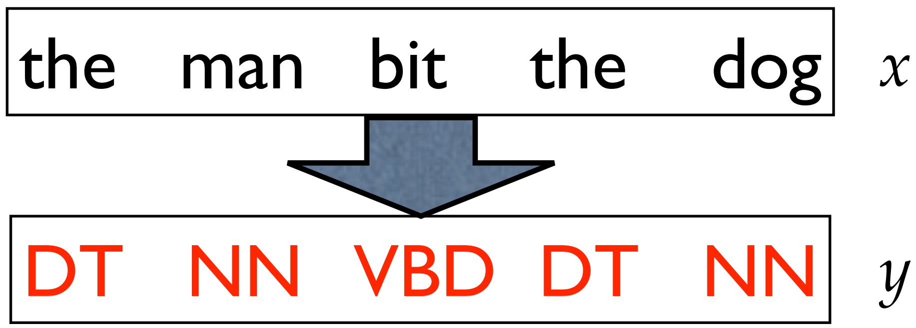
exponential
# of classes

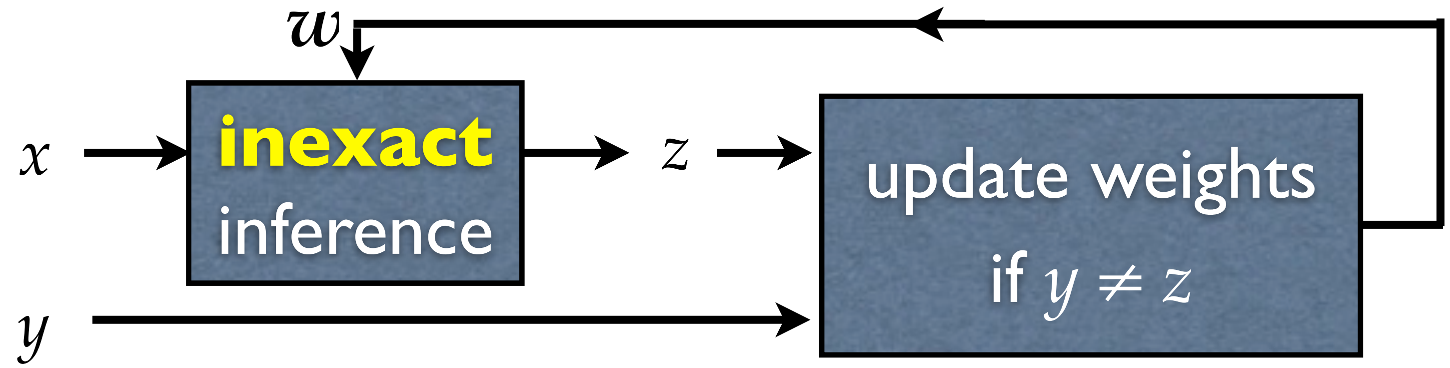


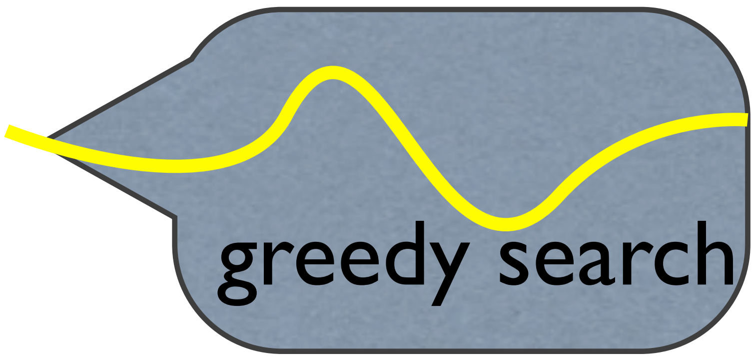


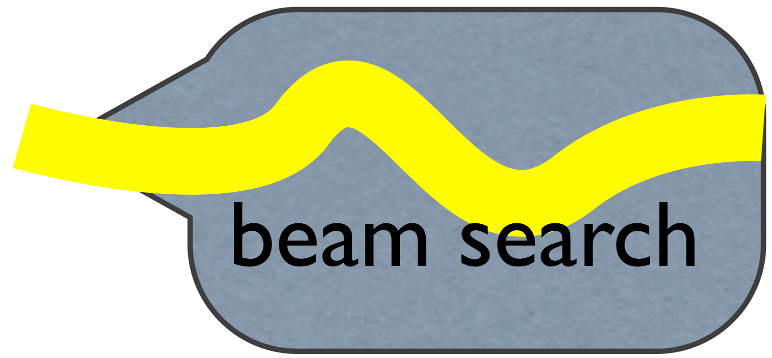
- challenge: search efficiency (exponentially many classes)
  - often use dynamic programming (DP)
  - but still too slow for repeated use, e.g. parsing is $O(n^3)$
  - and can't use non-local features in DP

Perceptron w/ Inexact Inference





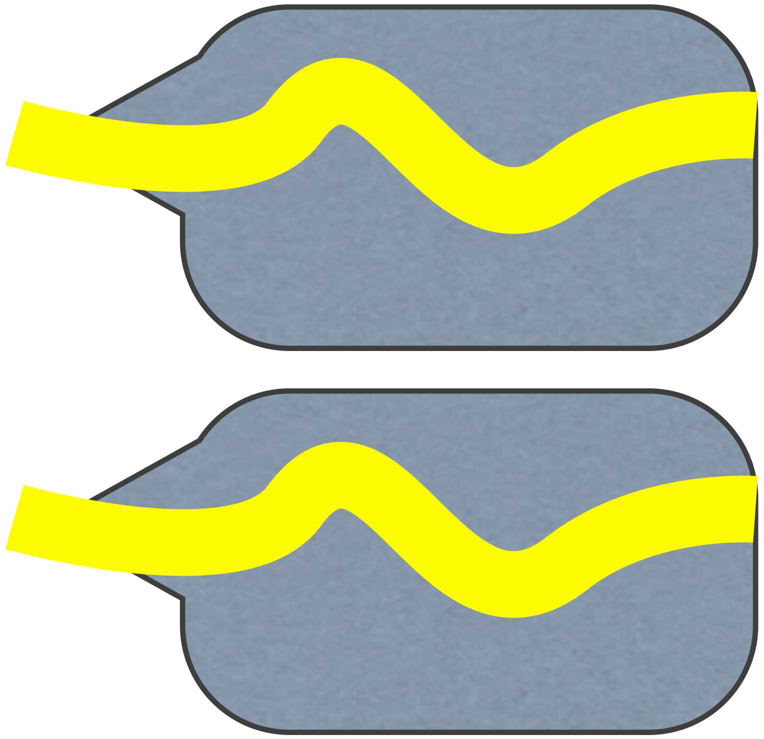


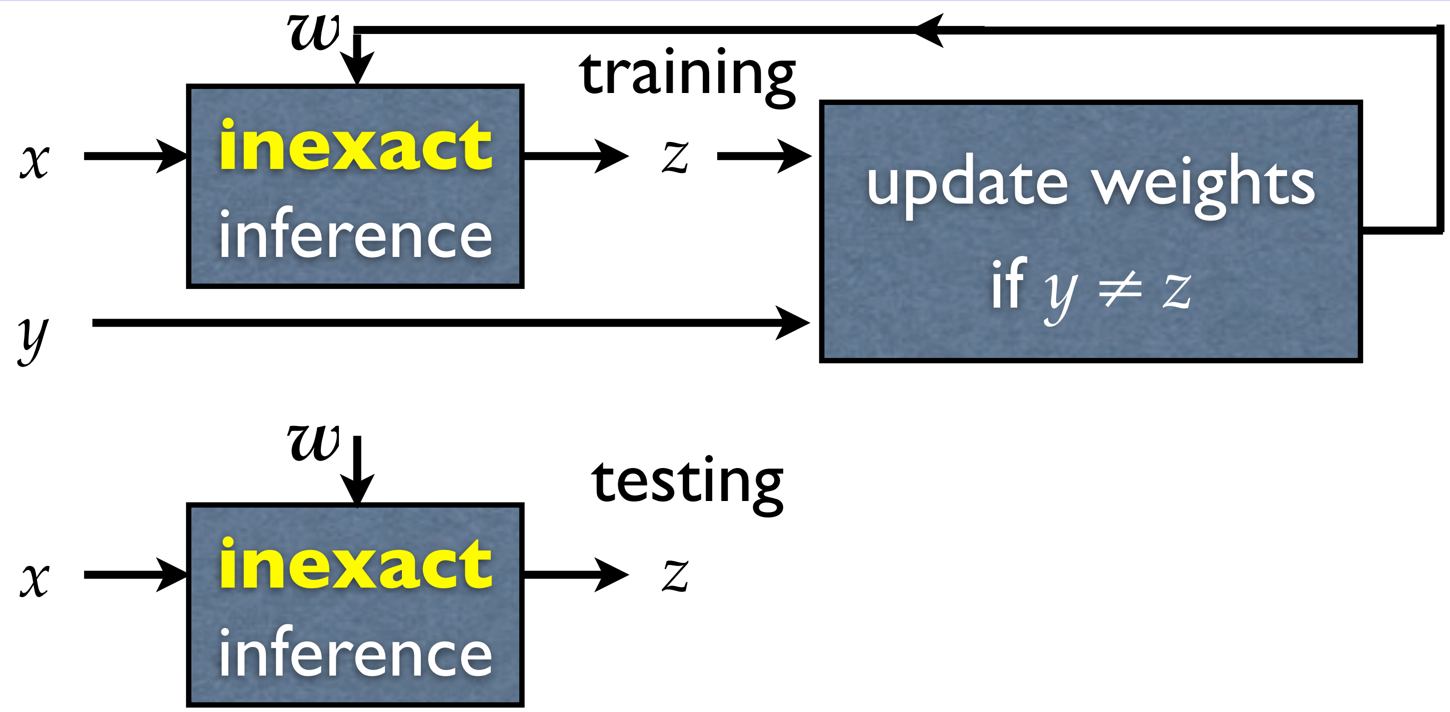


does it still work???

- routine use of inexact inference in NLP (e.g. beam search)
- how does structured perceptron work with inexact search?
  - so far most structured learning theory assume exact search
  - would search errors break these learning properties?
  - if so how to modify learning to accommodate inexact search?

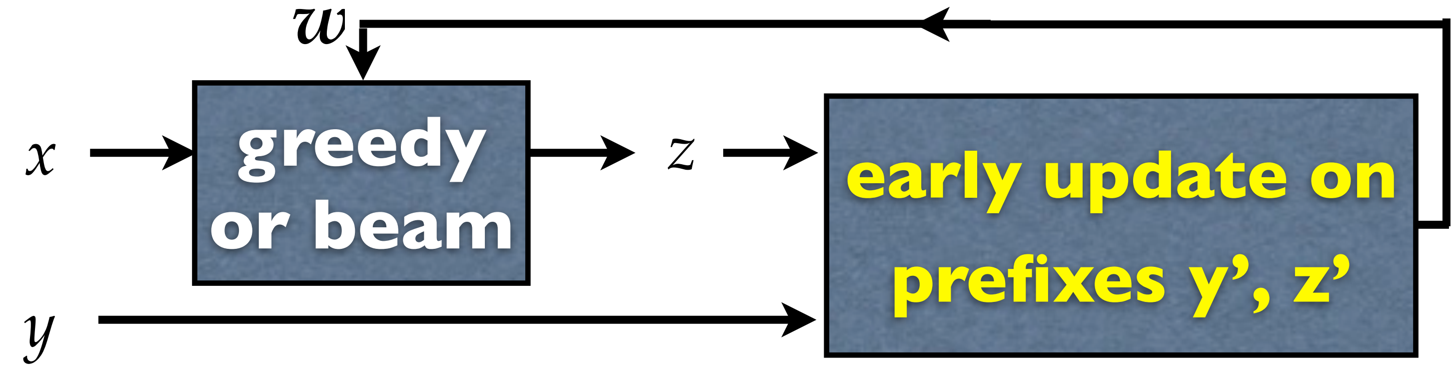
Idea: Search-Error-Robust Model





- train a "search-specific" or "search-error-robust" model
  - we assume the same "search box" in training and testing
  - model should "live with" search errors from search box
- exact search => convergence; greedy => no convergence
  - how can we make perceptron converge w/ greedy search?

Our Contributions



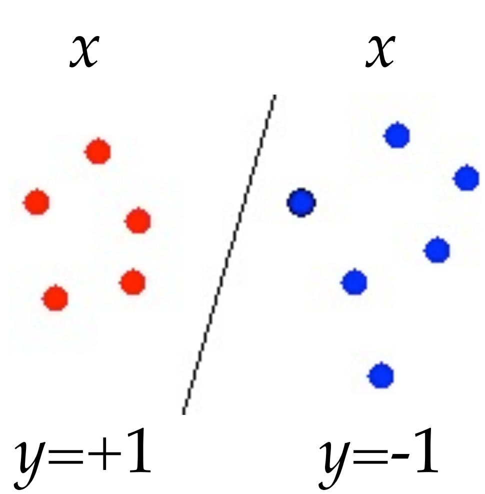
- theory: a framework for perceptron w/ inexact search
  - explains previous work (early update etc) as special cases
- practice: new update methods within the framework
  - converges faster and better than early update
  - real impact on state-of-the-art parsing and tagging
  - more advantageous when search error is severer

In this talk...

- Motivations: Structured Learning and Search Efficiency
- Structured Perceptron and Inexact Search
  - perceptron does not converge with inexact search
  - early update (Collins/Roark '04) seems to help; but why?
- New Perceptron Framework for Inexact Search
  - explains early update as a special case
  - convergence theory with *arbitrarily* inexact search
  - new update methods within this framework
- Experiments

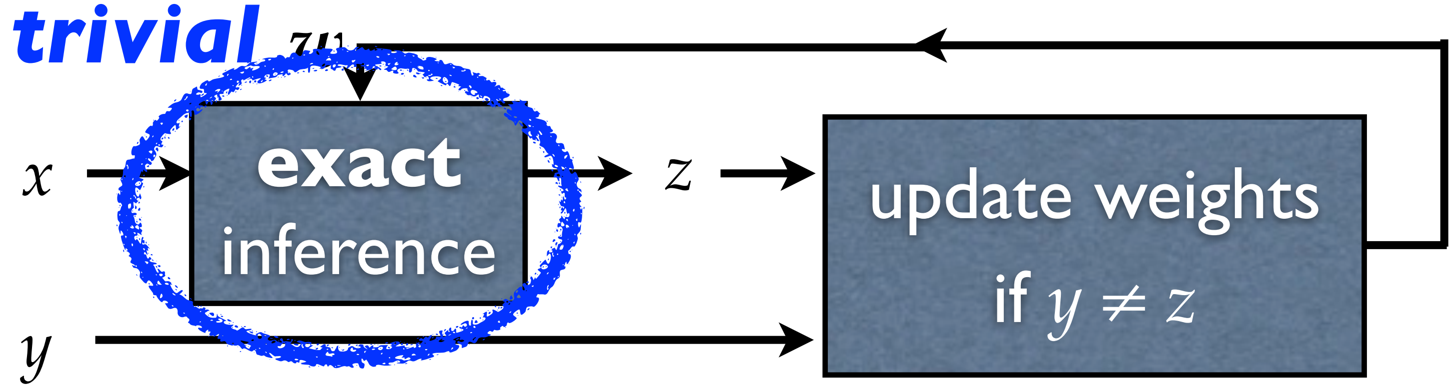
Structured Perceptron (Collins 02)

- simple generalization from binary/multiclass perceptron
- online learning: for each example (x, y) in data
  - inference: find the best output z given current weight w
  - update weights when if y ≠ z



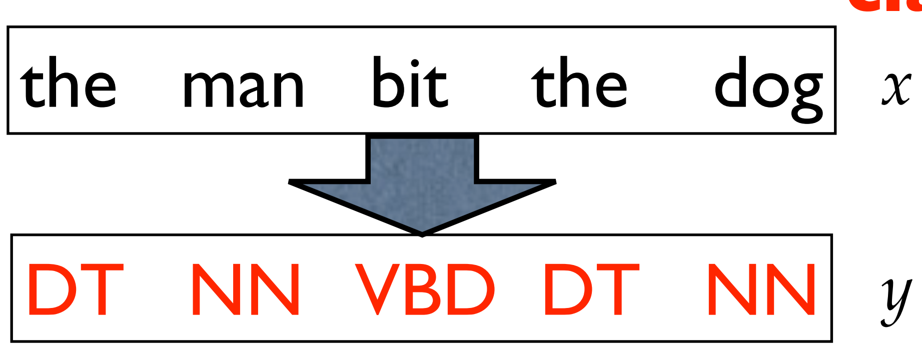
constant classes

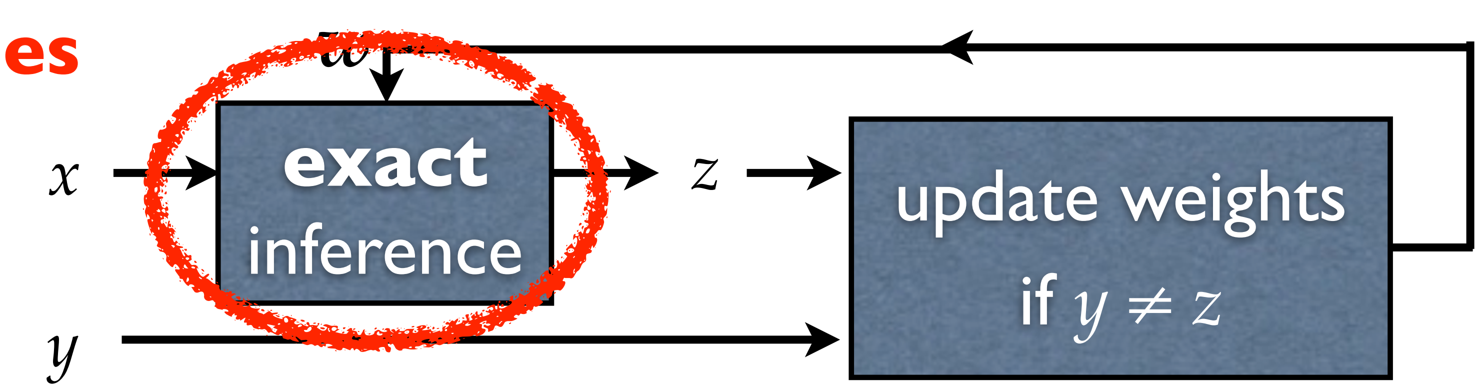
trivial



exponential classes

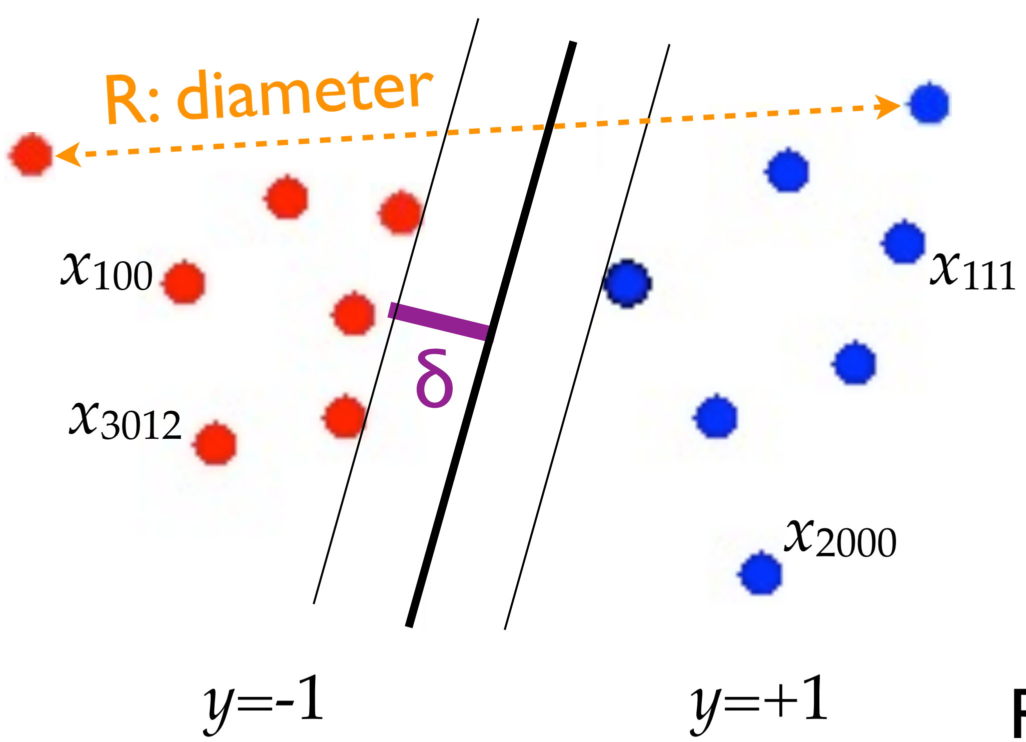
hard



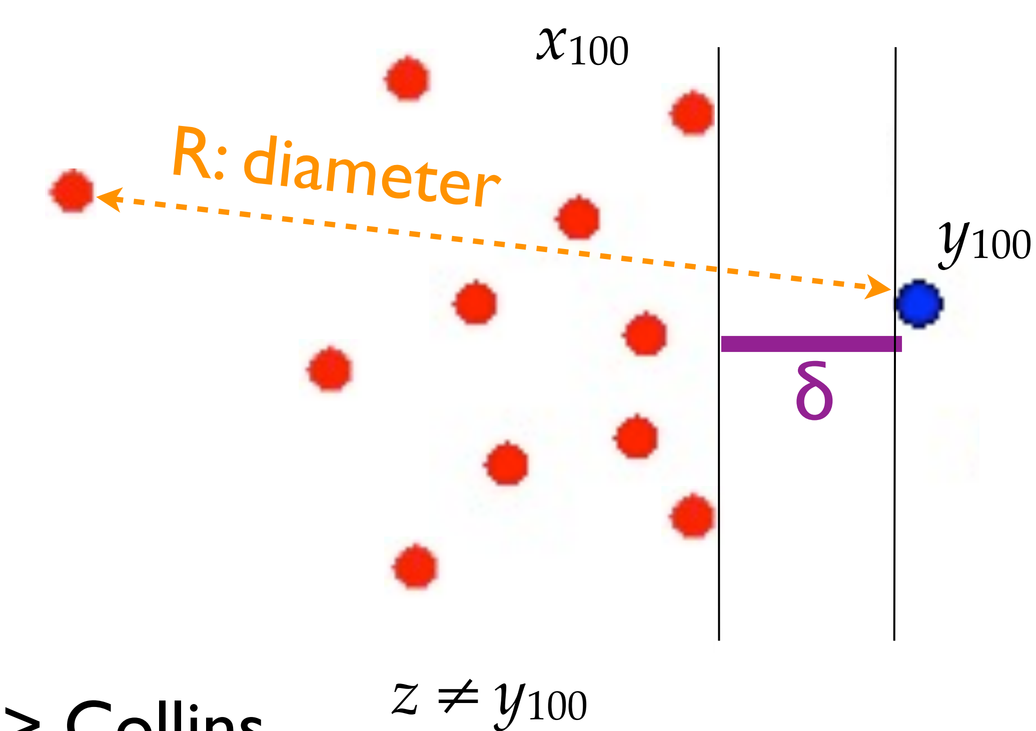


Convergence with Exact Search

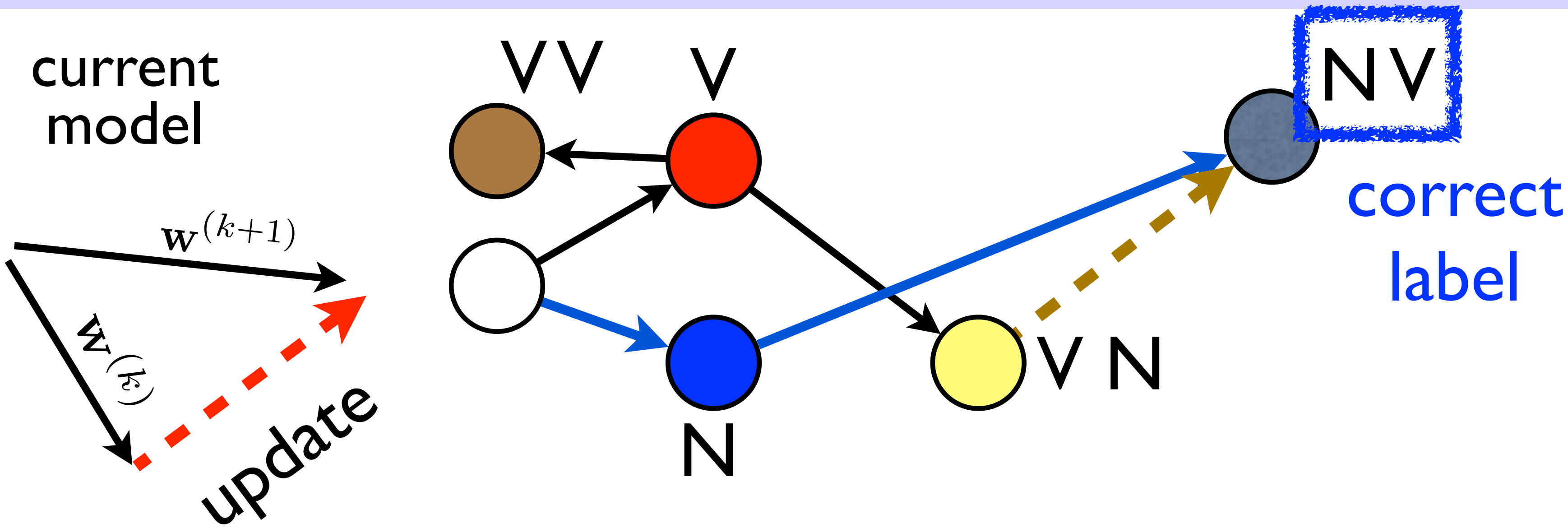
- linear classification: converges iff. data is separable
- structured: converges iff. data separable & search exact
  - there is an oracle vector that correctly labels all examples
  - one vs the rest (correct label better than all incorrect labels)
- theorem: if separable, then **# of updates ≤ R^2 / δ^2** R: diameter



Rosenblatt => Collins
1957 2002



Convergence with Exact Search



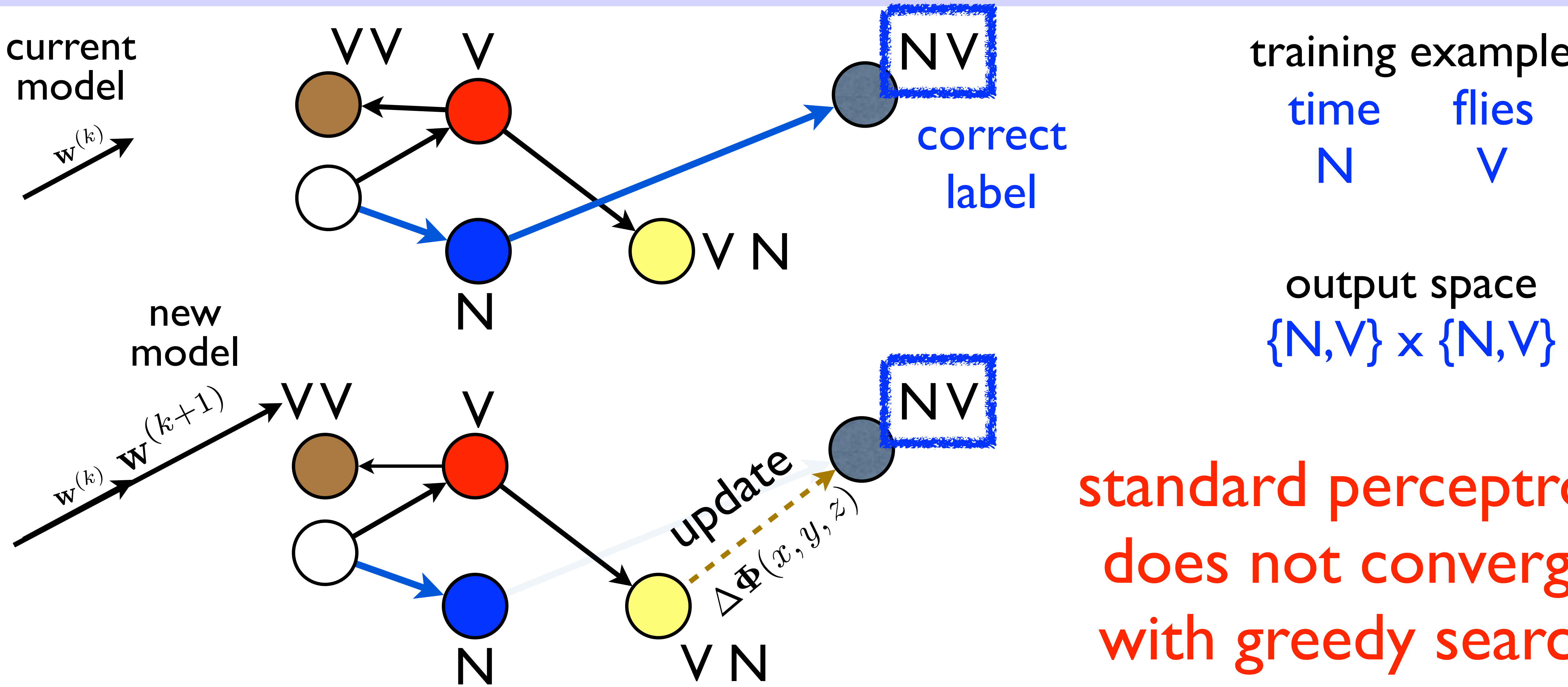
training example

time	flies
N	V

output space
{N,V} x {N,V}

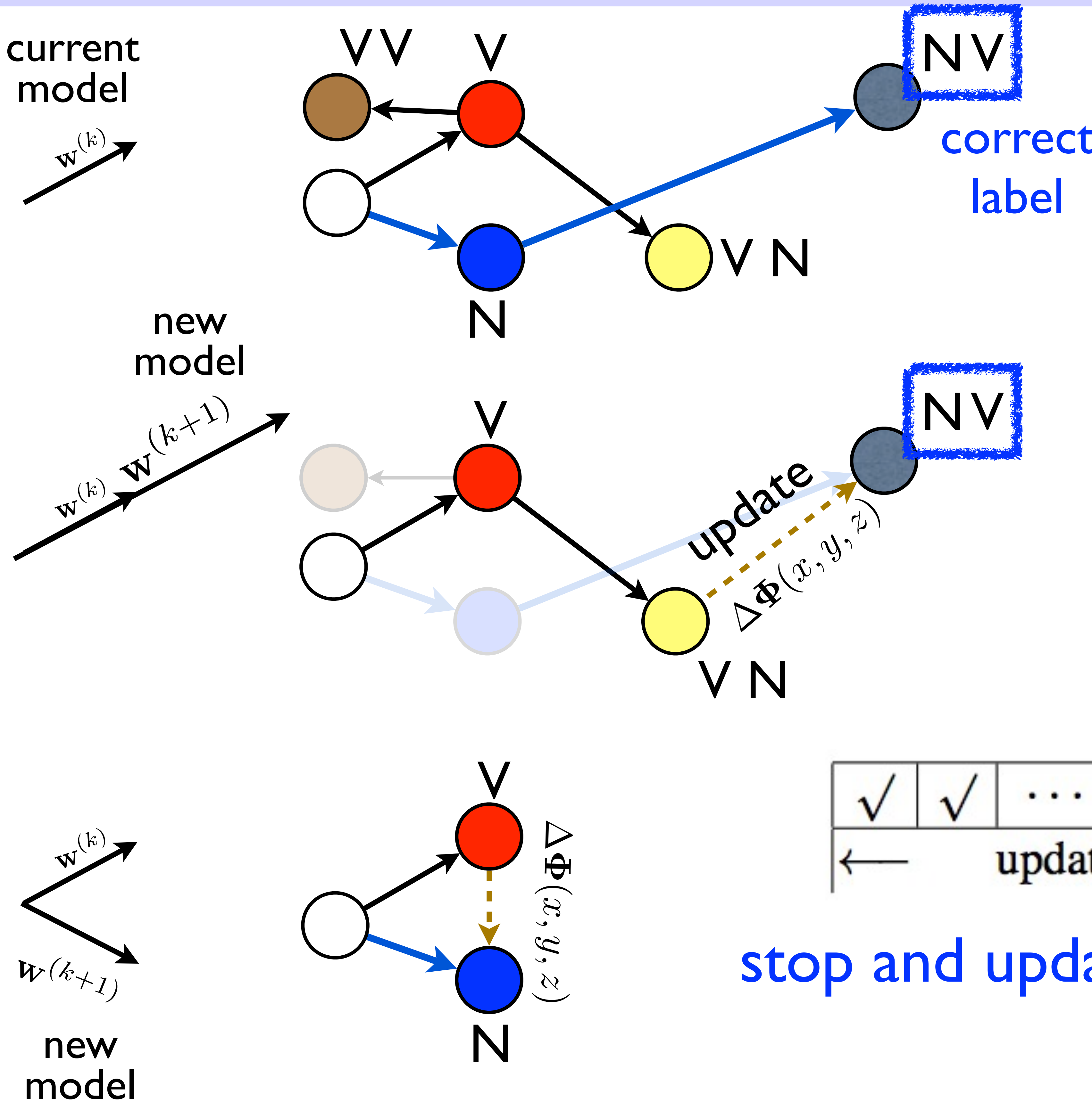
standard perceptron converges with exact search

No Convergence w/ Greedy Search



standard perceptron does not converge with greedy search

Early update (Collins/Roark 2004) to rescue



training example

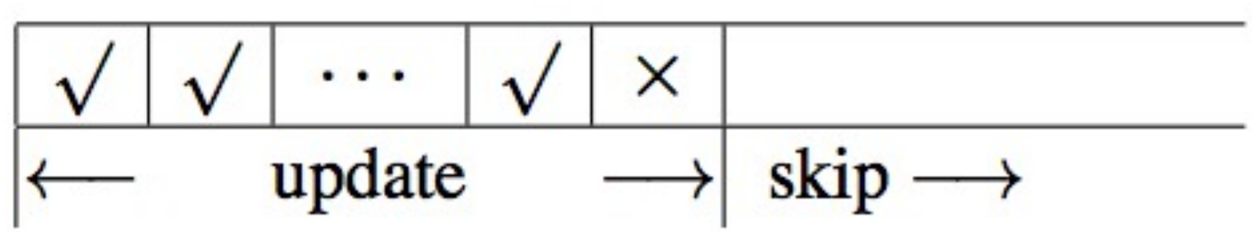
time flies

N V

output space

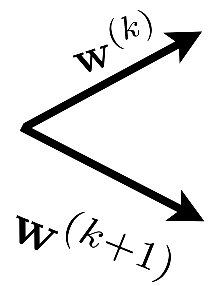
{N,V} x {N,V}

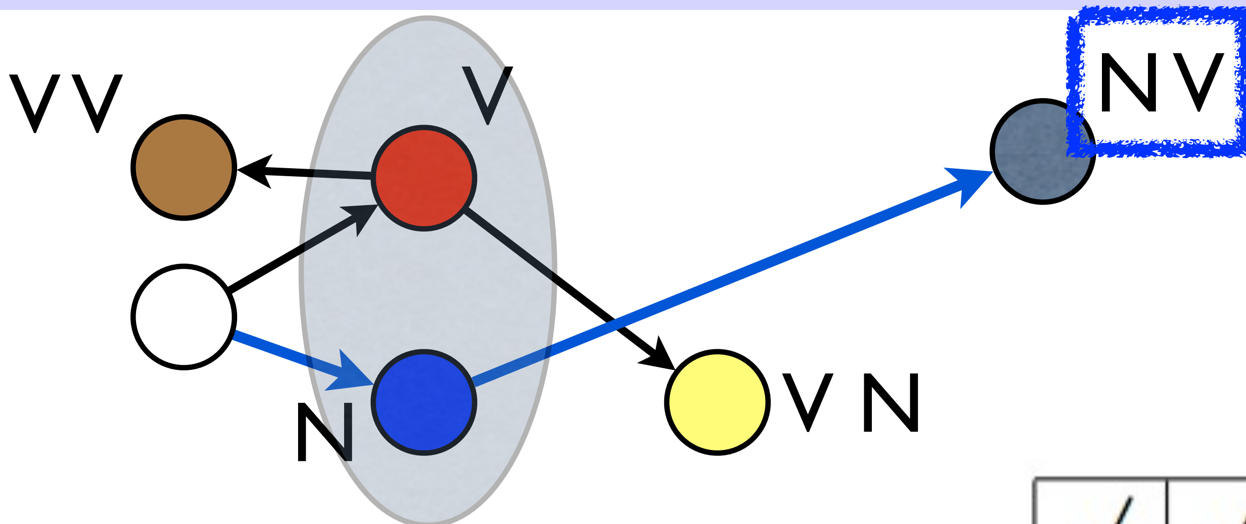
standard perceptron does not converge with greedy search

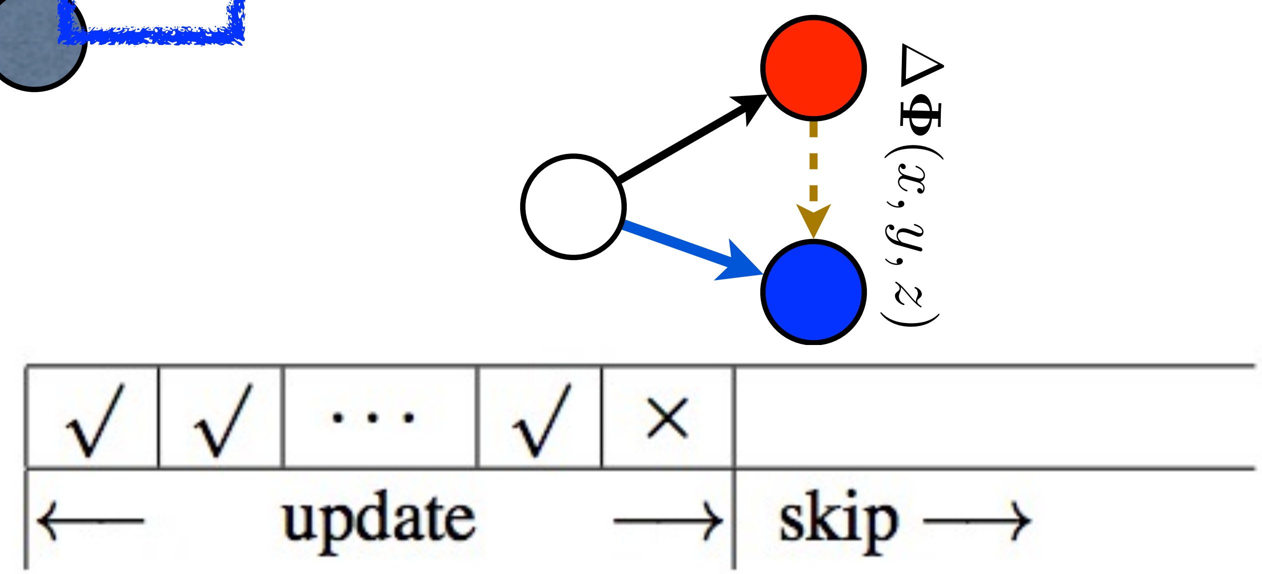


stop and update at the first mistake

Why?



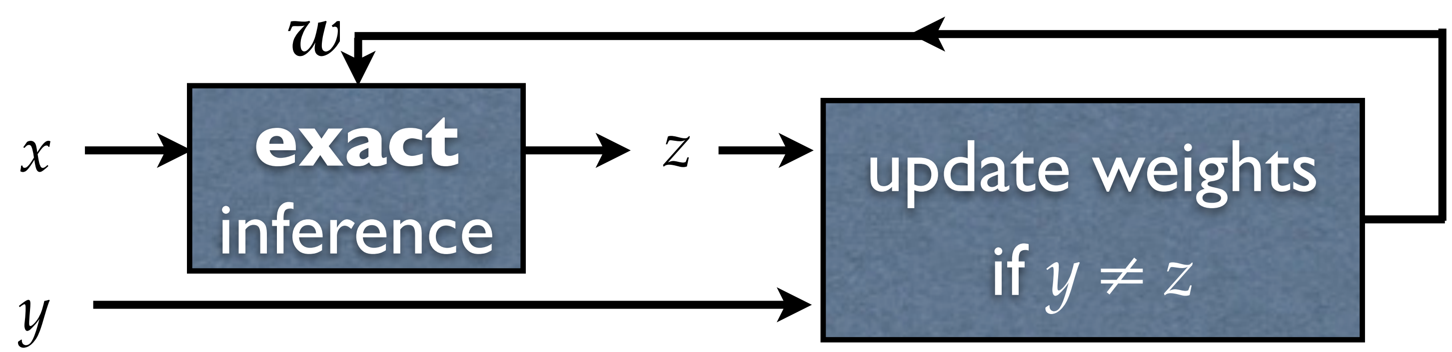


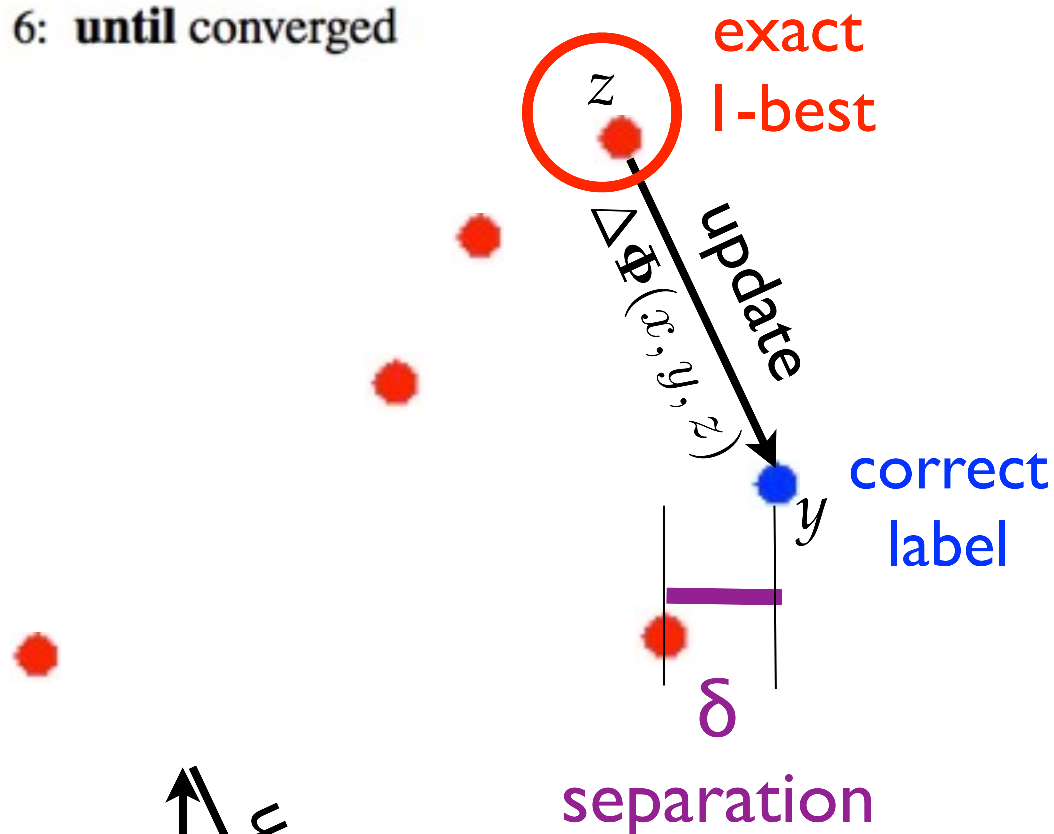


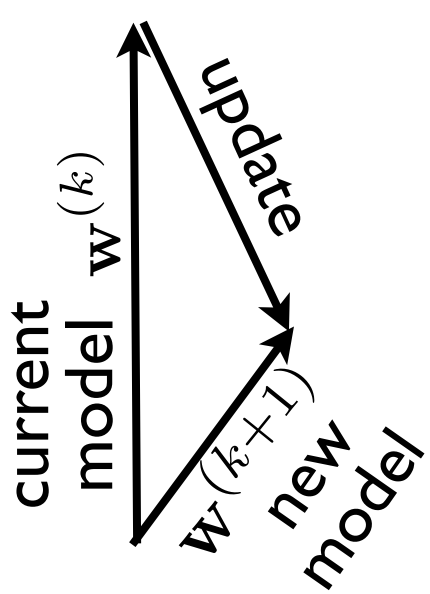
- why does inexact search break convergence property?
  - what is required for convergence? exactness?
- why does early update (Collins/Roark 04) work?
  - it works well in practice and is now a standard method
  - but there has been no theoretical justification
- we answer these Qs by inspecting the convergence proof

Geometry of Convergence Proof pt 1

repeat
  for each example (x, y) **in** D **do**
    $z \leftarrow \textsc{Exact}(x, \mathbf{w})$
    if $z \neq y$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
until converged







unit oracle
vector **u**

perceptron update:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)$$

$$\mathbf{u} \cdot \mathbf{w}^{(k+1)} = \mathbf{u} \cdot \mathbf{w}^{(k)} + \boxed{\mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z) \geq \delta \quad \text{margin}}$$

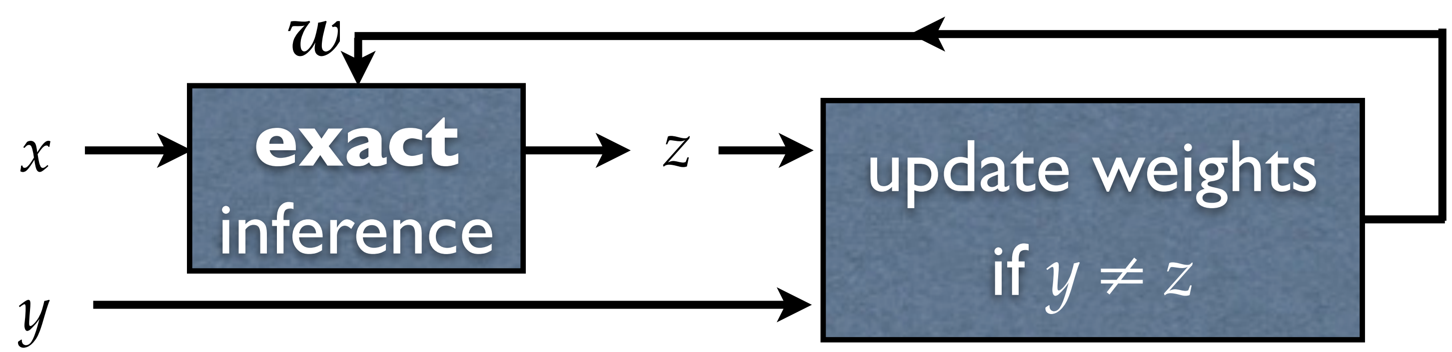
$$\mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta \qquad \text{(by induction)}$$

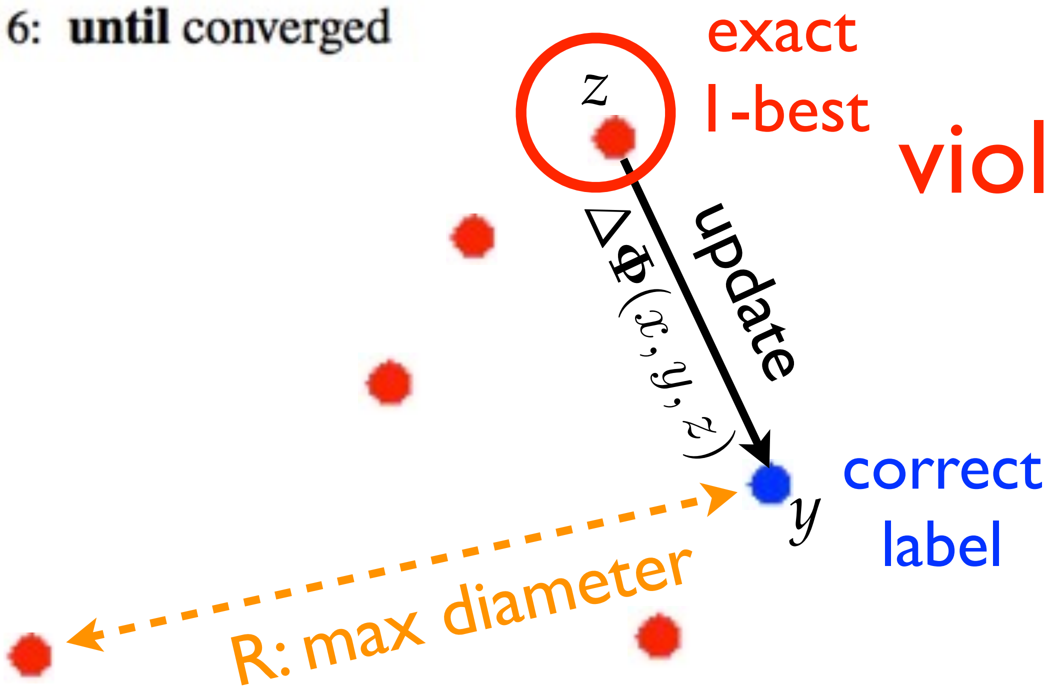
$$\|\mathbf{u}\| \|\mathbf{w}^{(k+1)}\| \geq \mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta$$

$$\|\mathbf{w}^{(k+1)}\| \geq k\delta \qquad \text{(part 1: upperbound)}$$

Geometry of Convergence Proof pt 2

repeat
  for each example (x, y) **in** D **do**
    $z \leftarrow \textsc{Exact}(x, \mathbf{w})$
    if $z \neq y$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
until converged





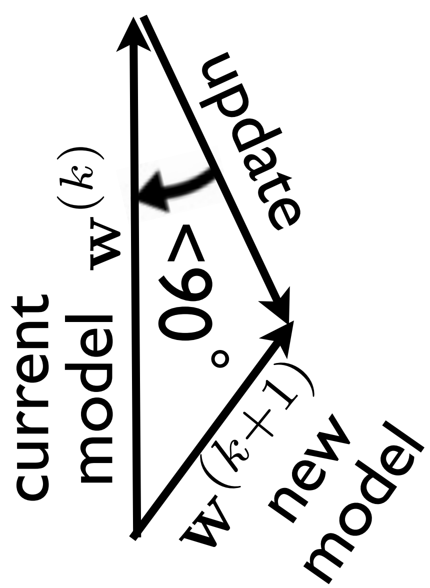
violation: incorrect label scored higher

perceptron update:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)$$

$$\|\mathbf{w}^{(k+1)}\|^2 = \|\mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)\|^2$$

$$= \|\mathbf{w}^{(k)}\|^2 + \underbrace{\|\Delta\mathbf{\Phi}(x, y, z)\|^2}_{\leq R^2 \text{ diameter}} + 2 \underbrace{\mathbf{w}^{(k)} \cdot \Delta\mathbf{\Phi}(x, y, z)}_{\leq 0 \text{ violation}}$$

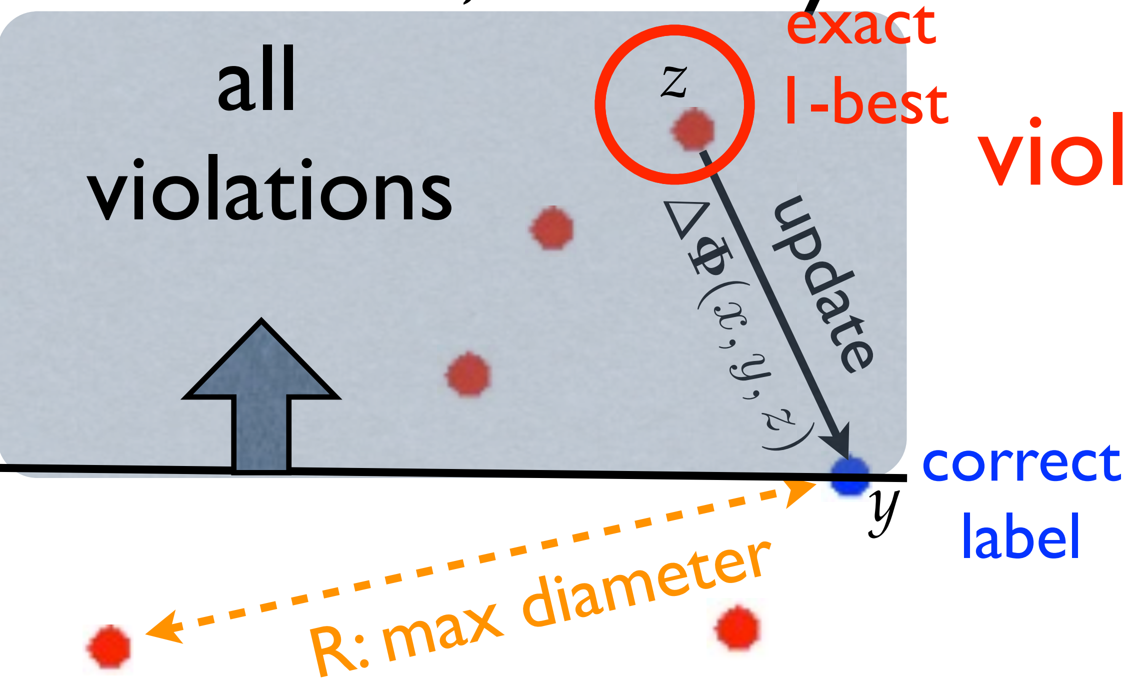


by induction: $\|\mathbf{w}^{(k+1)}\|^2 \leq kR^2$ (part 2: upperbound)

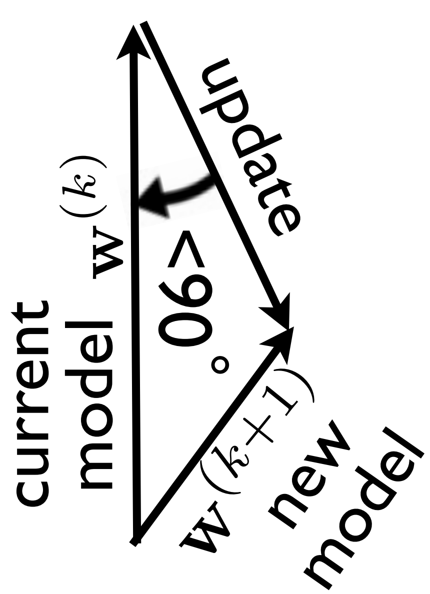
parts 1+2 => update bounds: $\boxed{k \leq R^2/\delta^2}$

Violation is All we need!

- exact search is **not** really required by the proof
  - rather, it is only used to ensure violation!



violation: incorrect label scored higher

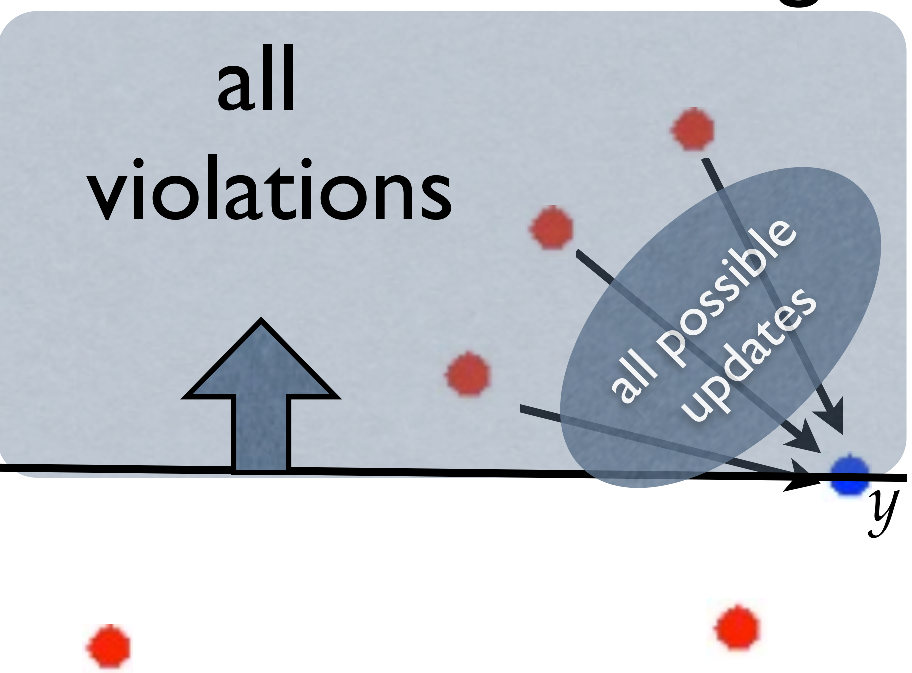


the proof only uses 3 facts:

1. separation (margin)
2. diameter (always finite)
3. violation (but no need for exact)

Violation-Fixing Perceptron

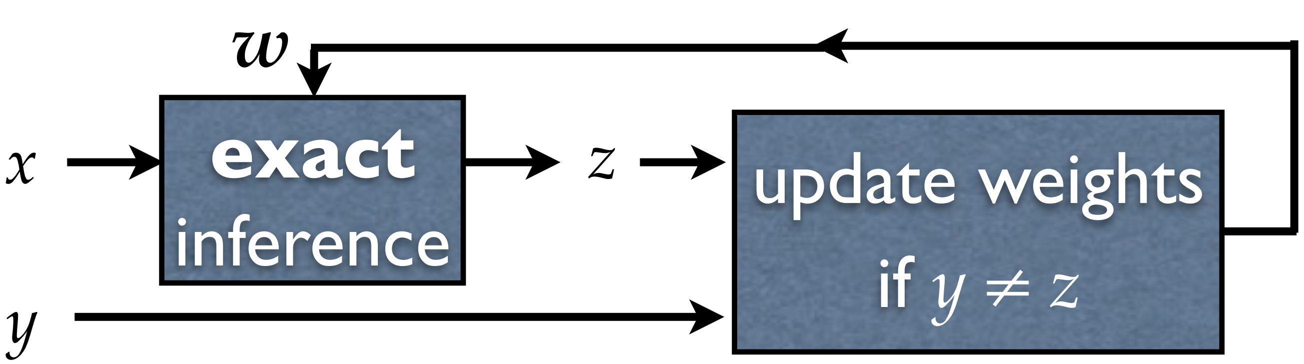
- if we guarantee violation, we don't care about exactness!
  - violation is good b/c we can at least fix a mistake



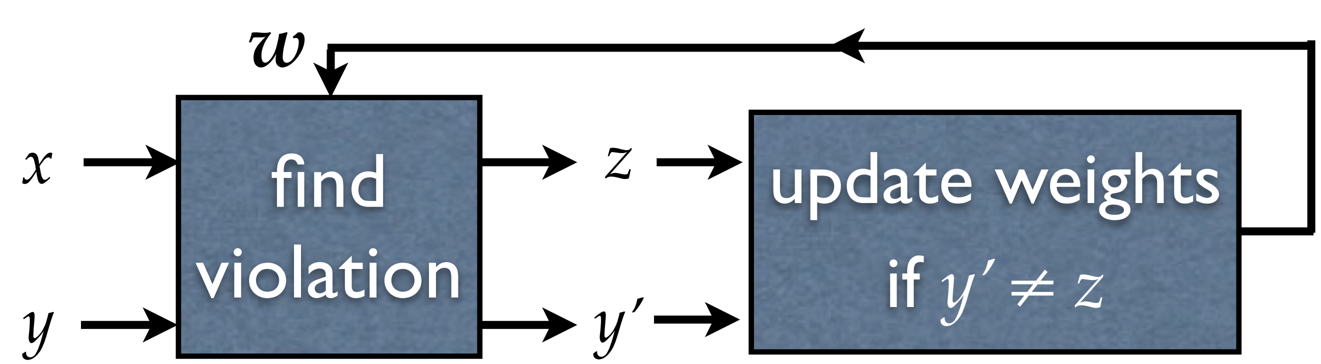
same mistake bound as before!

repeat
  for each example (x, y) **in** D **do**
    $(x, y', z) = \textsc{FindViolation}(x, y, \mathbf{w})$
    if $z \neq y$ **then** $\triangleright$ (x, y', z) is a viol
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y', z)$
until converged

standard perceptron

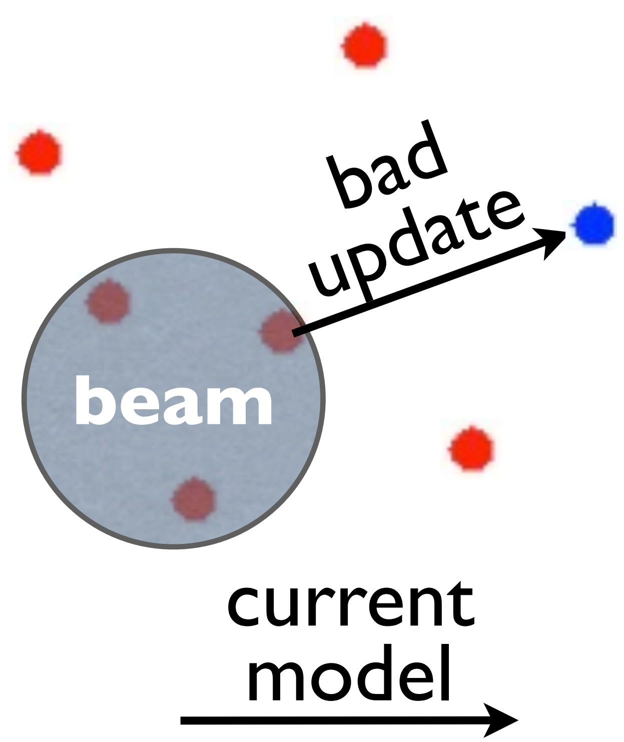


violation-fixing perceptron

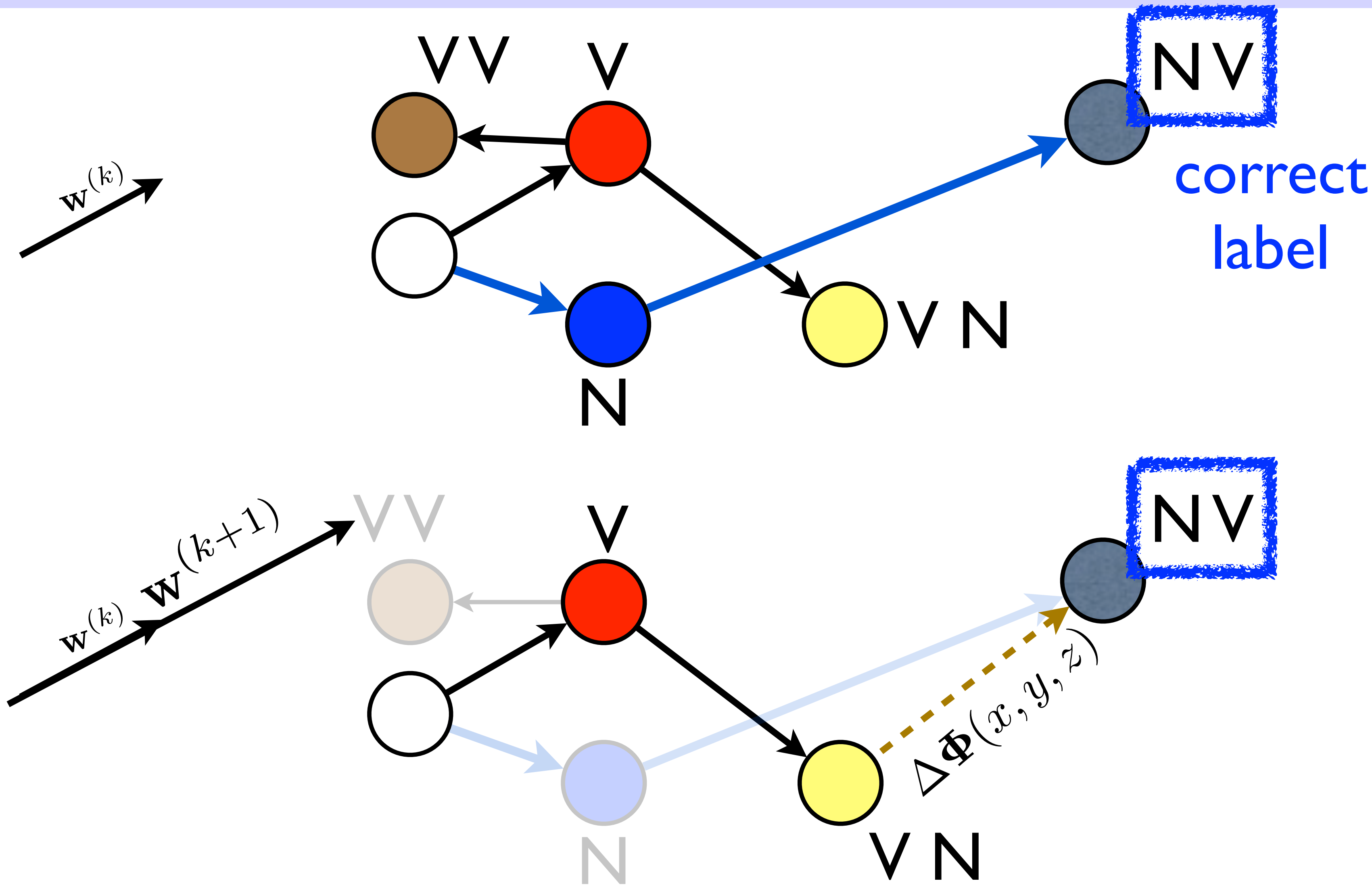


What if can't guarantee violation

- this is why perceptron doesn't work well w/ inexact search
  - because not every update is guaranteed to be a violation
  - thus the proof breaks; no convergence guarantee
- example: beam or greedy search
  - the model might prefer the correct label (if exact search)
  - but the search prunes it away
  - such a non-violation update is "bad" because it doesn't fix any mistake
  - the new model still misguides the search



Standard Update: No Guarantee



training example

time flies

N V

output space

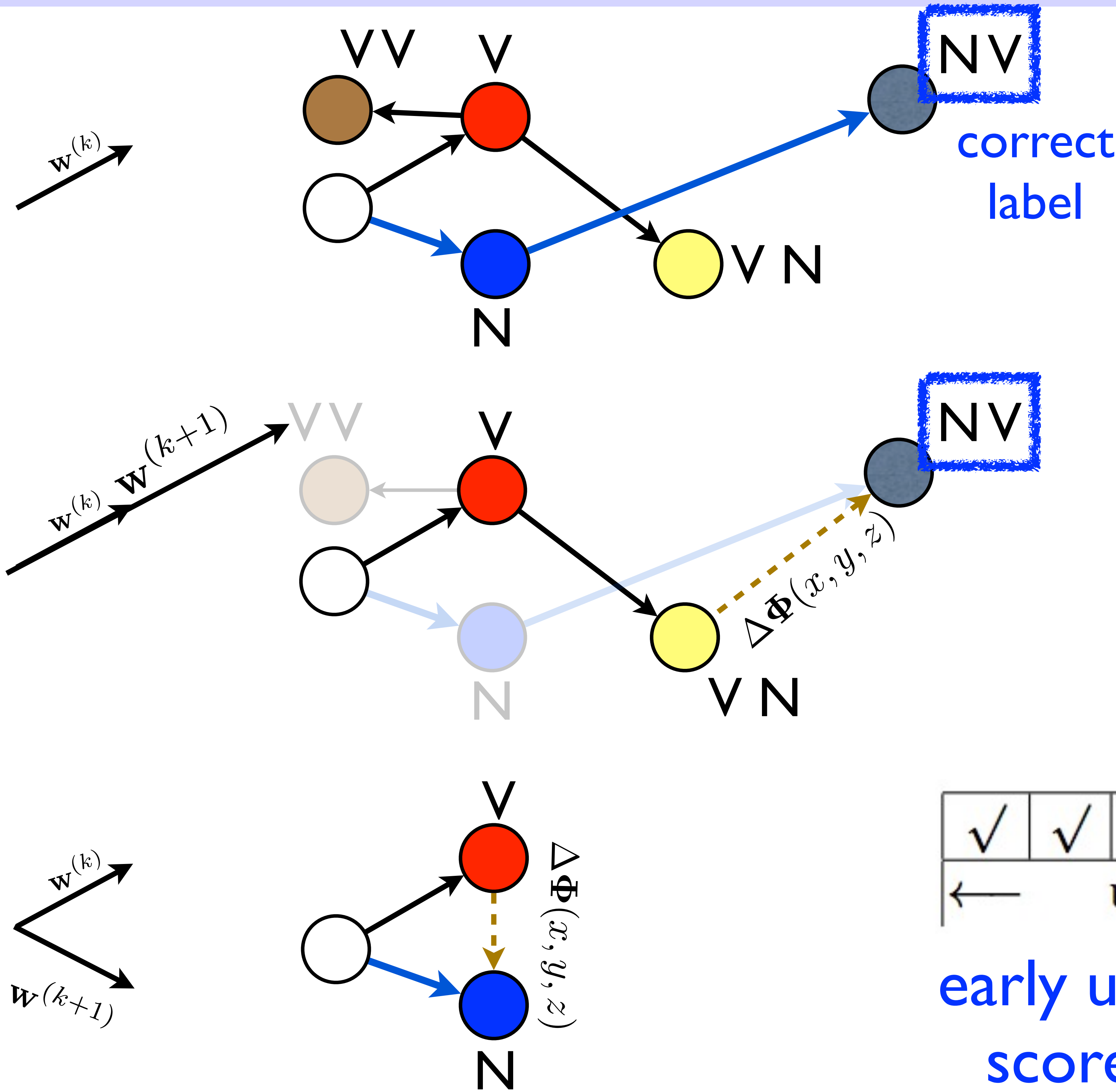
{N,V} x {N,V}

standard update doesn't converge b/c it doesn't guarantee violation

correct label scores higher.

non-violation: bad update!

Early Update: Guarantees Violation



training example

time	flies
N	V

output space

{N,V} x {N,V}

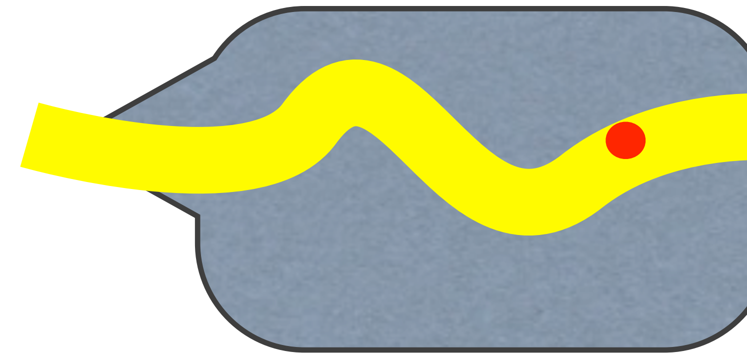
standard update doesn't converge b/c it doesn't guarantee violation

√	√	···	√	×	
⟵	update			⟶	skip ⟶

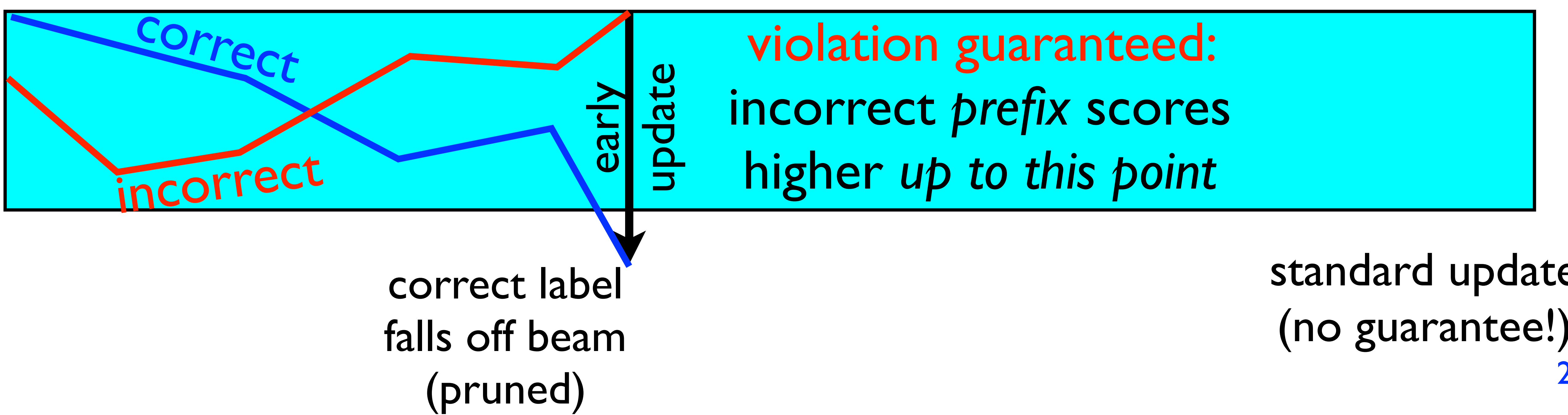
early update: incorrect prefix scores higher: a violation!

Early Update: from Greedy to Beam

- beam search is a generalization of greedy (where b=1)
  - at each stage we keep top b hypothesis
  - widely used: tagging, parsing, translation...

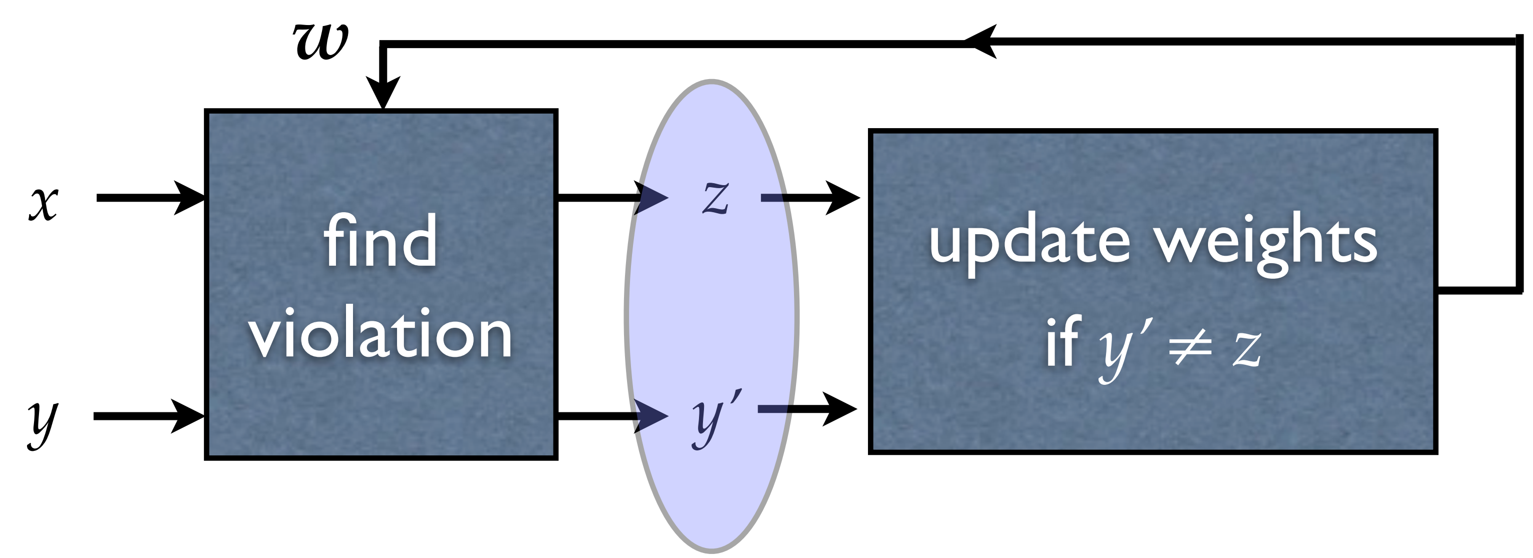


- early update -- when correct label first falls off the beam
  - up to this point the incorrect prefix should score higher
- standard update (full update) -- no guarantee!



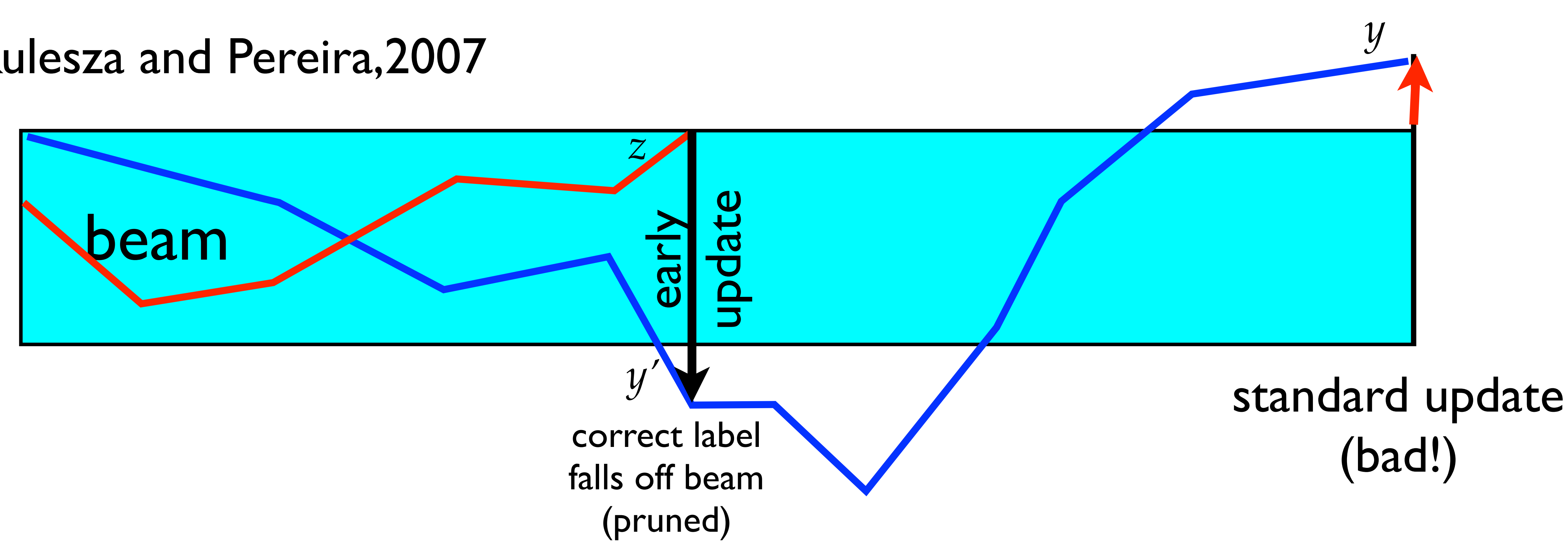
Early Update as Violation-Fixing

also new definition of
"beam separability":
a correct prefix should
score higher than
any incorrect prefix
of the same length
(maybe too strong)

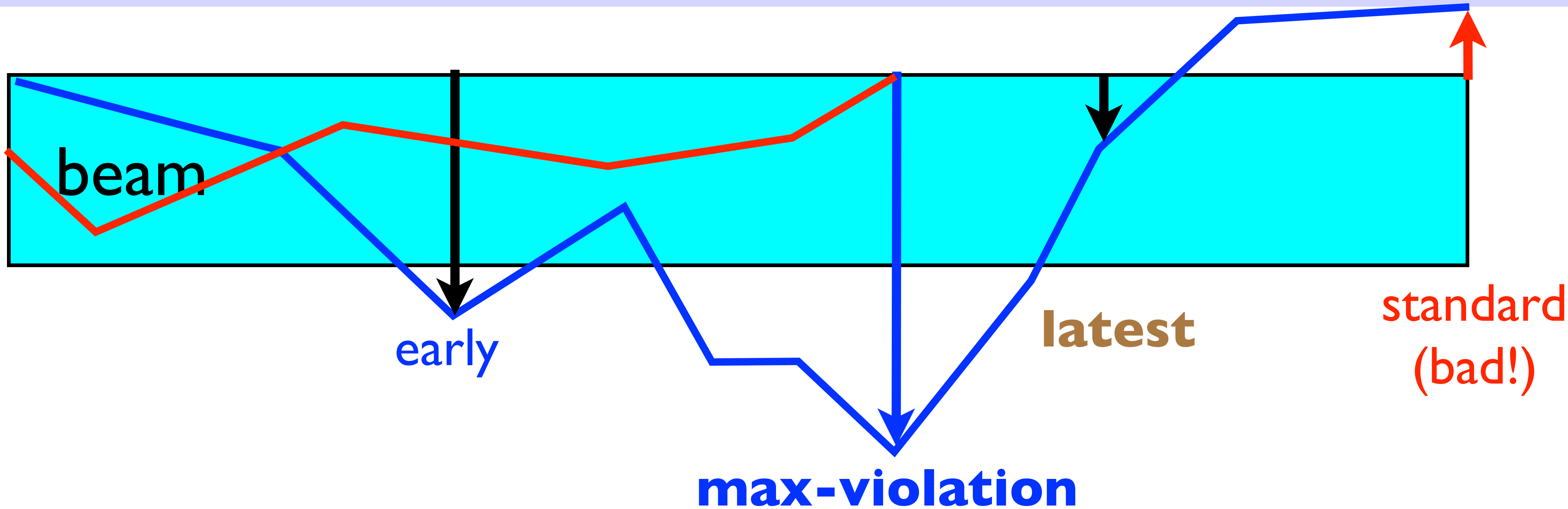


prefix violations

cf. Kulesza and Pereira,2007



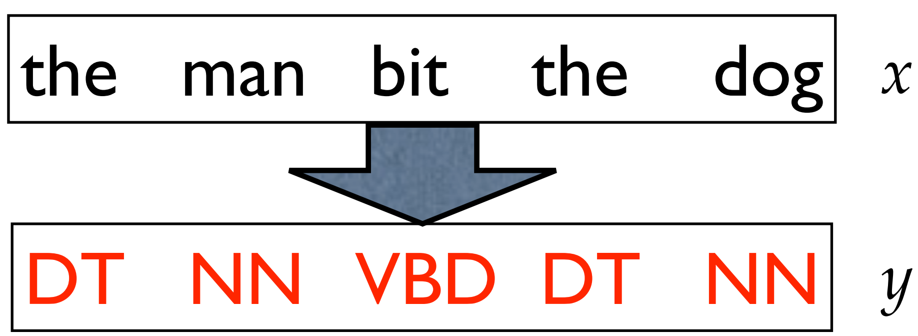
New Update Methods: max-violation, ...



- we now established a theory for early update (Collins/Roark)
- but it learns too slowly due to partial updates
- max-violation: use the prefix where violation is maximum
  - "worst-mistake" in the search space
- all these update methods are violation-fixing perceptrons

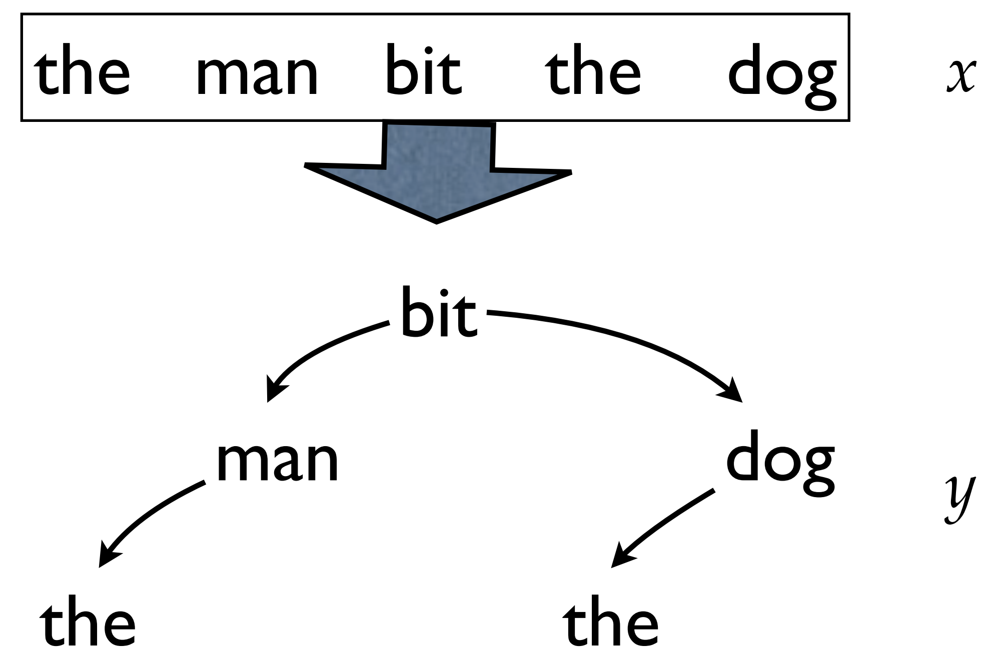
Experiments

trigram part-of-speech tagging



local features only,
exact search tractable
(proof of concept)

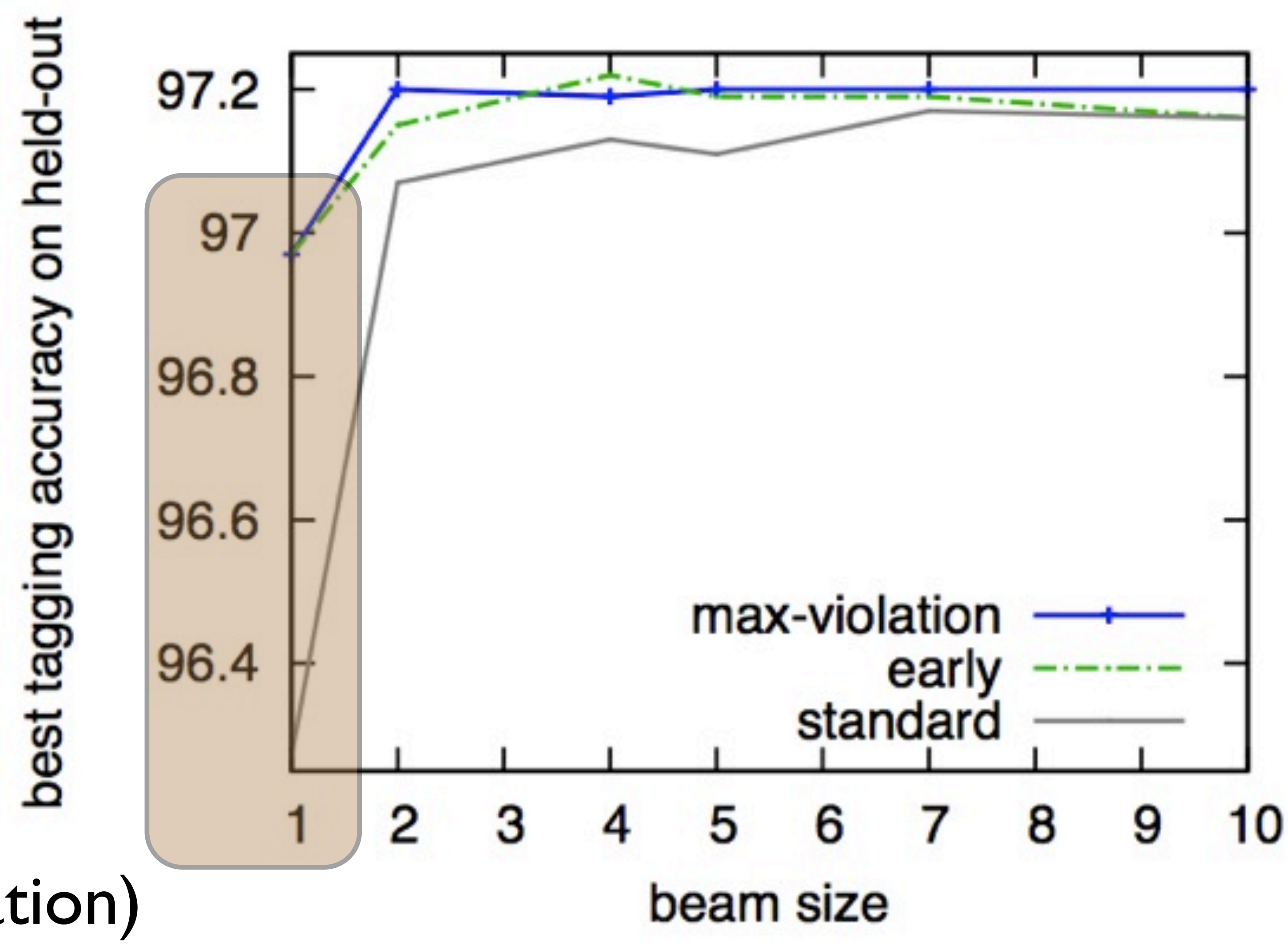
incremental dependency parsing



non-local features,
exact search intractable
(real impact)

1) Trigram Part of Speech Tagging

- standard update performs terribly with greedy search (b=1)
  - because search error is severe at b=1: half updates are bad!
  - no real difference beyond b=2: search error becomes rare

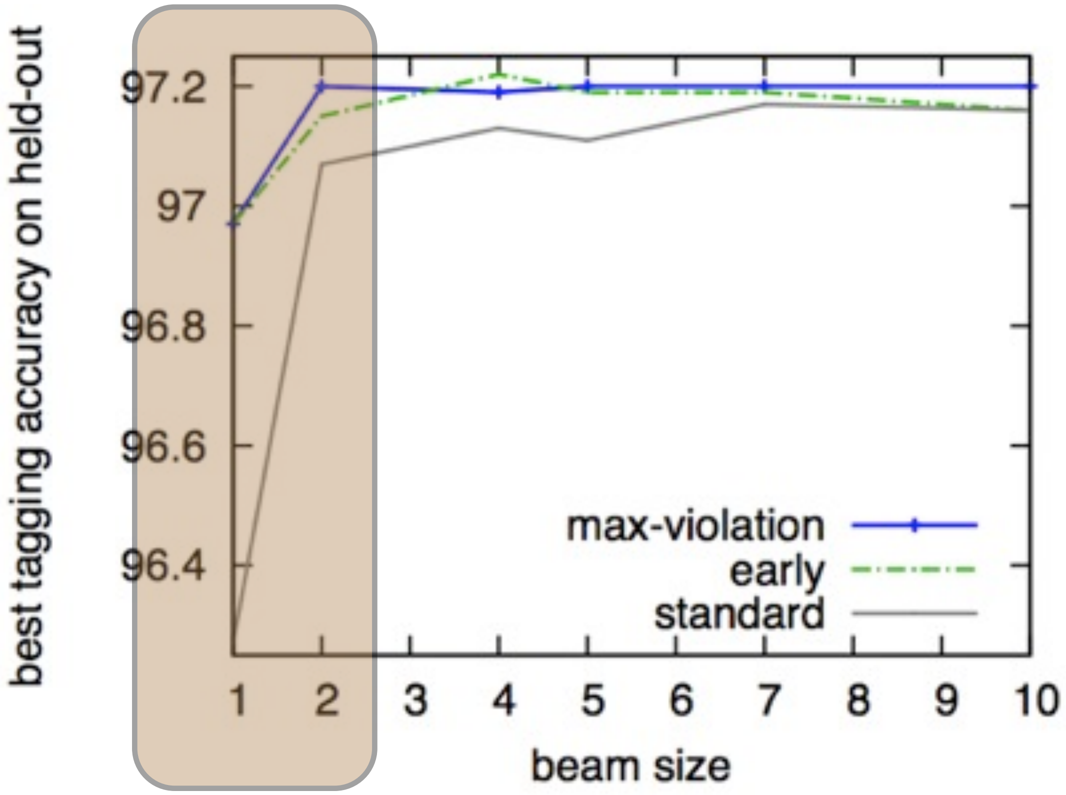


% of bad (non-violation) standard updates

53% 10% 1.5% 0.5%

Max-Violation Reduces Training Time

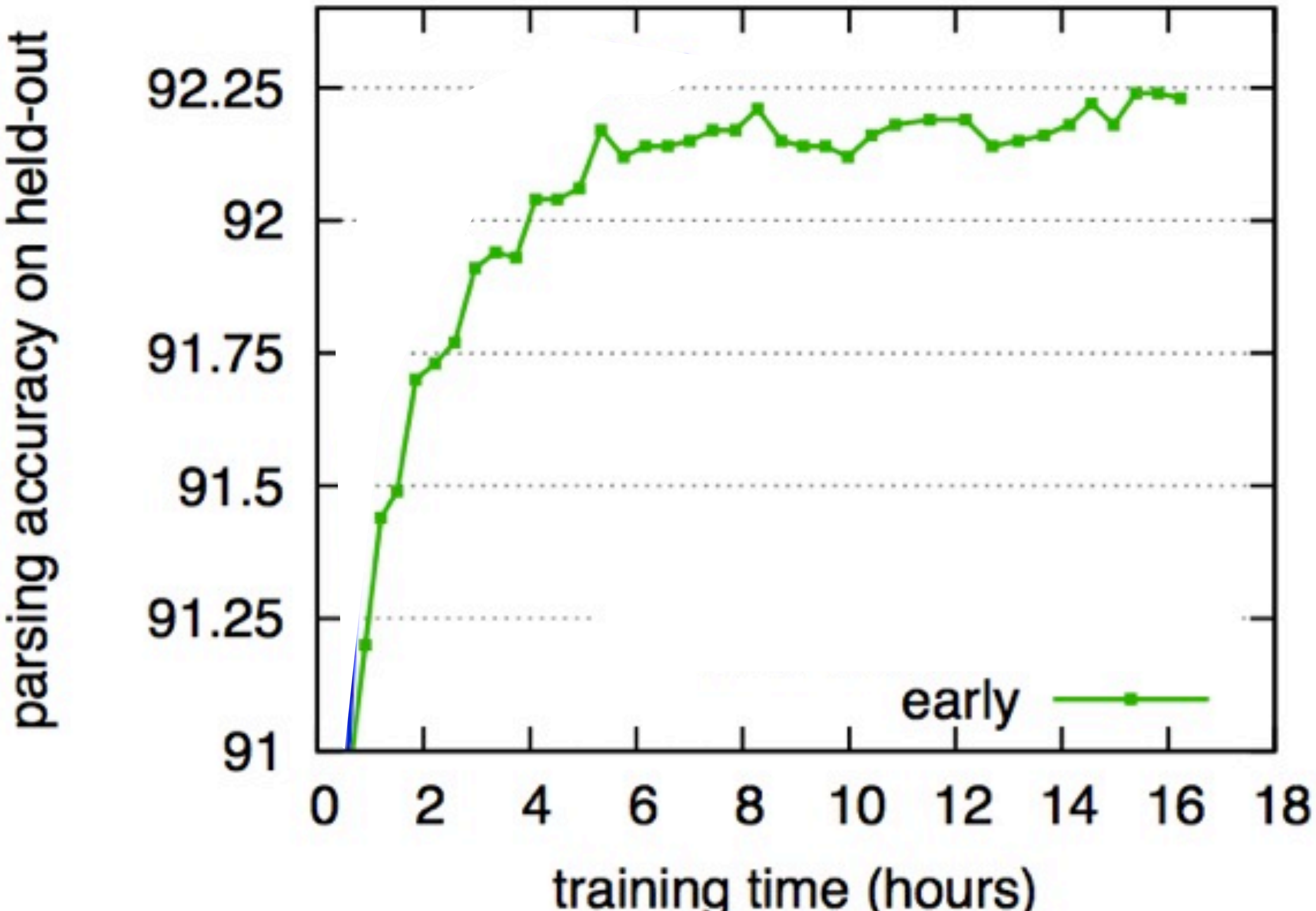
- max-violation peaks at b=2, greatly reduced training time
- early update achieves the highest dev/test accuracy
    - comparable to best published accuracy (Shen et al '07)
- future work: add non-local features to tagging



	beam	*iter*	*time*	*test*
standard	-	6	162m	97.28
early	4	6	37m	97.27
max-violation	2	3	26m	97.27
Shen et al (2007)				97.33

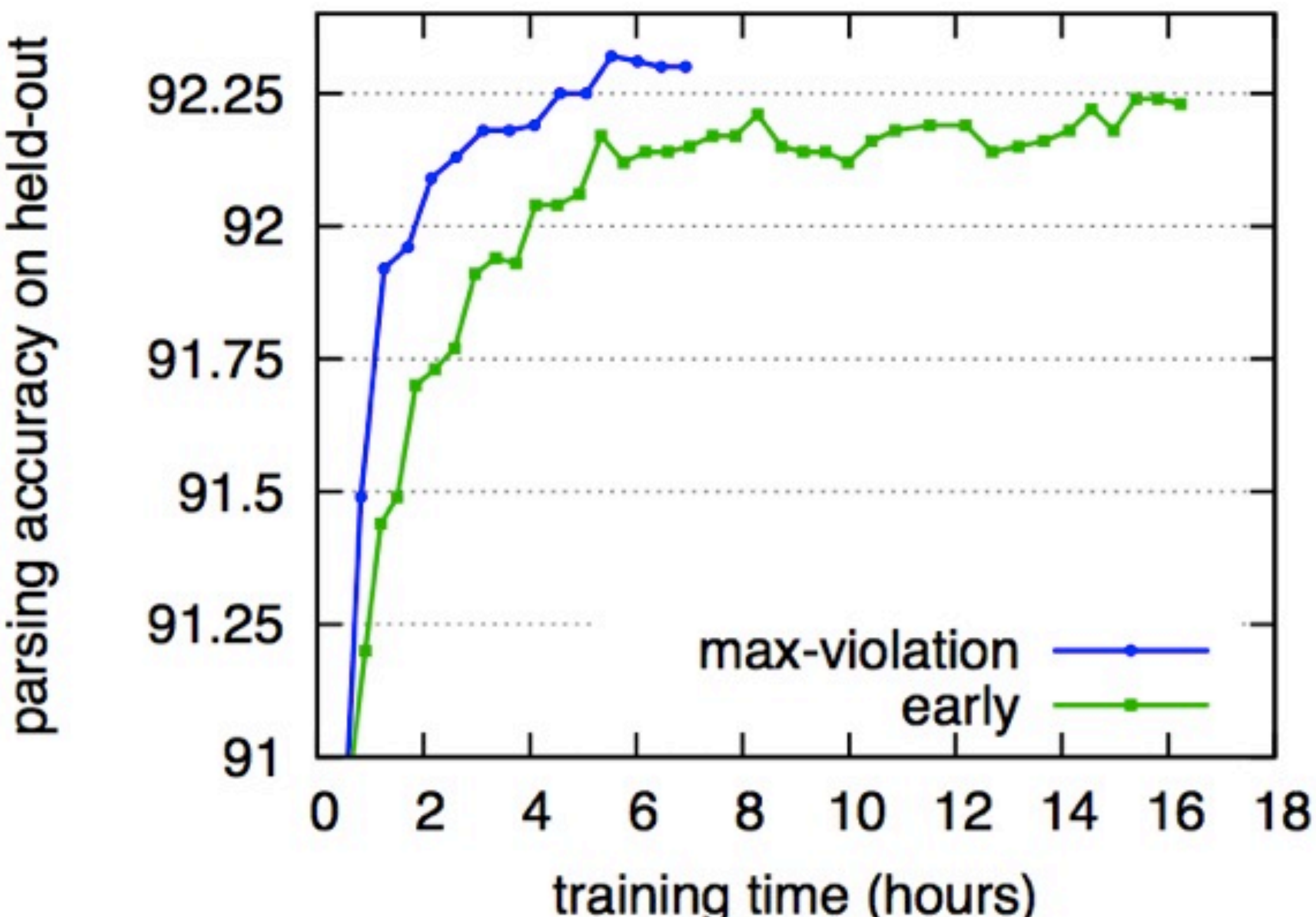
2) Incremental Dependency Parsing

- DP incremental dependency parser (Huang and Sagae 2010)
- non-local history-based features rule out exact DP
  - we use beam search, and search error is severe
  - baseline: early update. extremely slow: 38 iterations



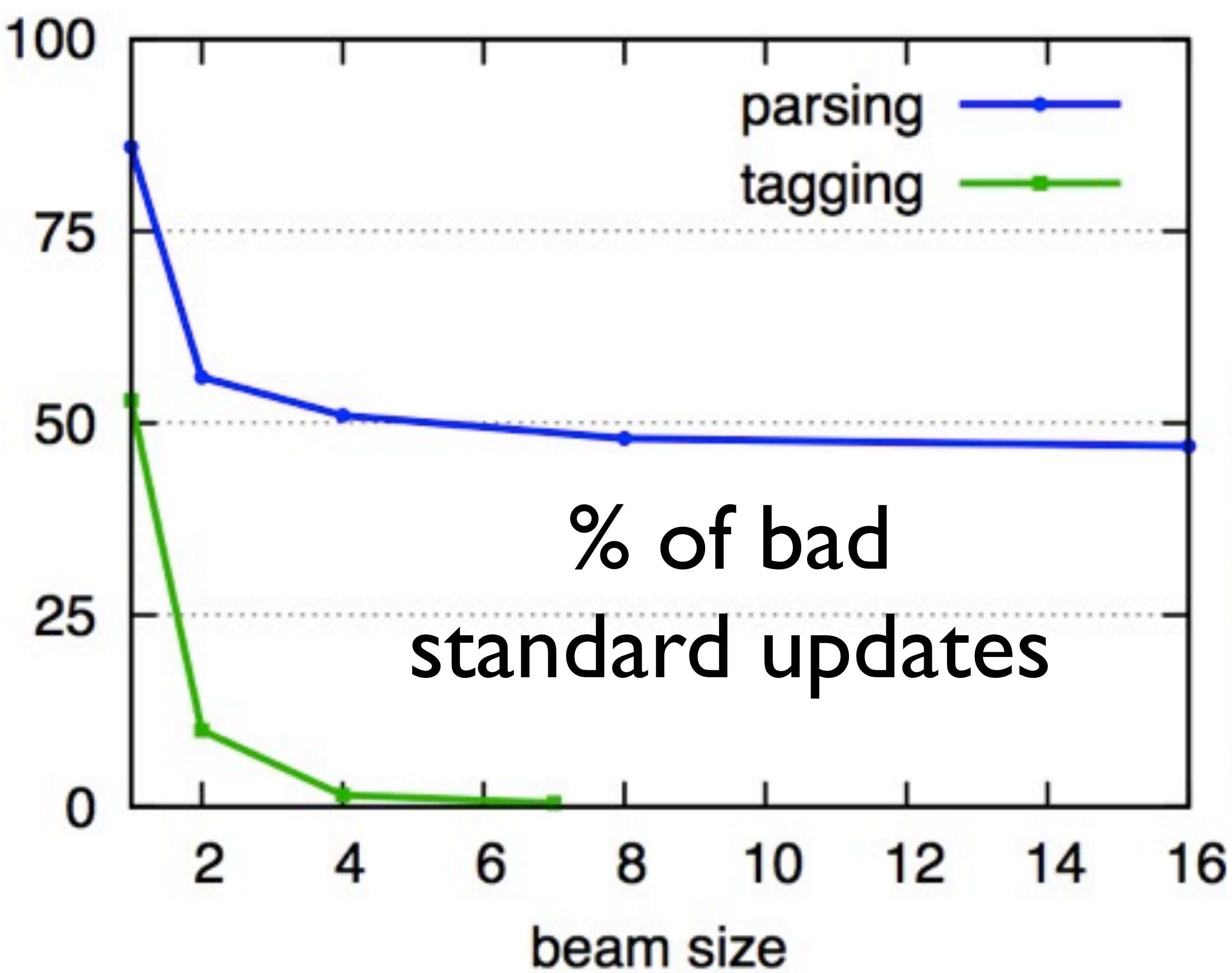
Max-violation converges much faster

- early update: 38 iterations, 15.4 hours (92.24)
- max-violation: 10 iterations, 4.6 hours (92.25)
  12 iterations, 5.5 hours (92.32)

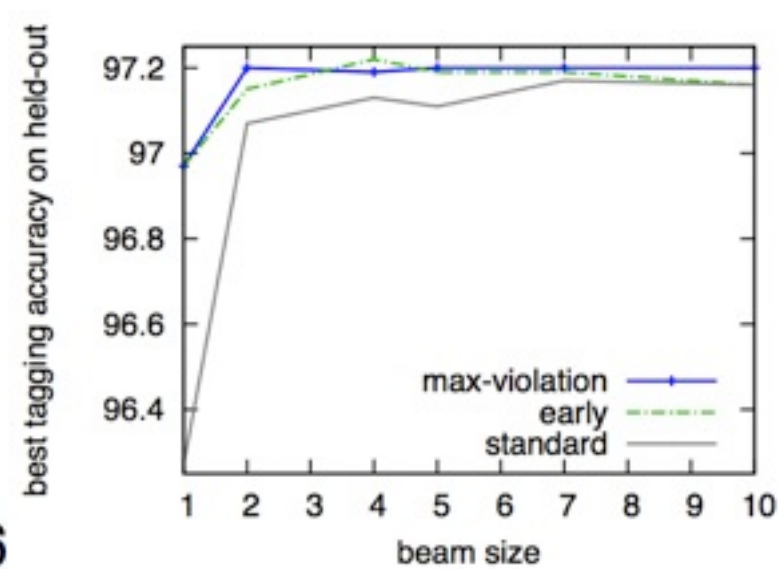


Comparison b/w tagging & parsing

- search error is much more severe in parsing than in tagging
- standard update is OK in tagging except greedy search (b=1)
- but performs horribly in parsing even at large beam (b=8)
  - because ~50% of standard updates are bad (non-violation)!



take-home message:
our methods are more helpful
for harder search problems!



	test
standard	**79.1**
early	92.1
max-violation	92.2

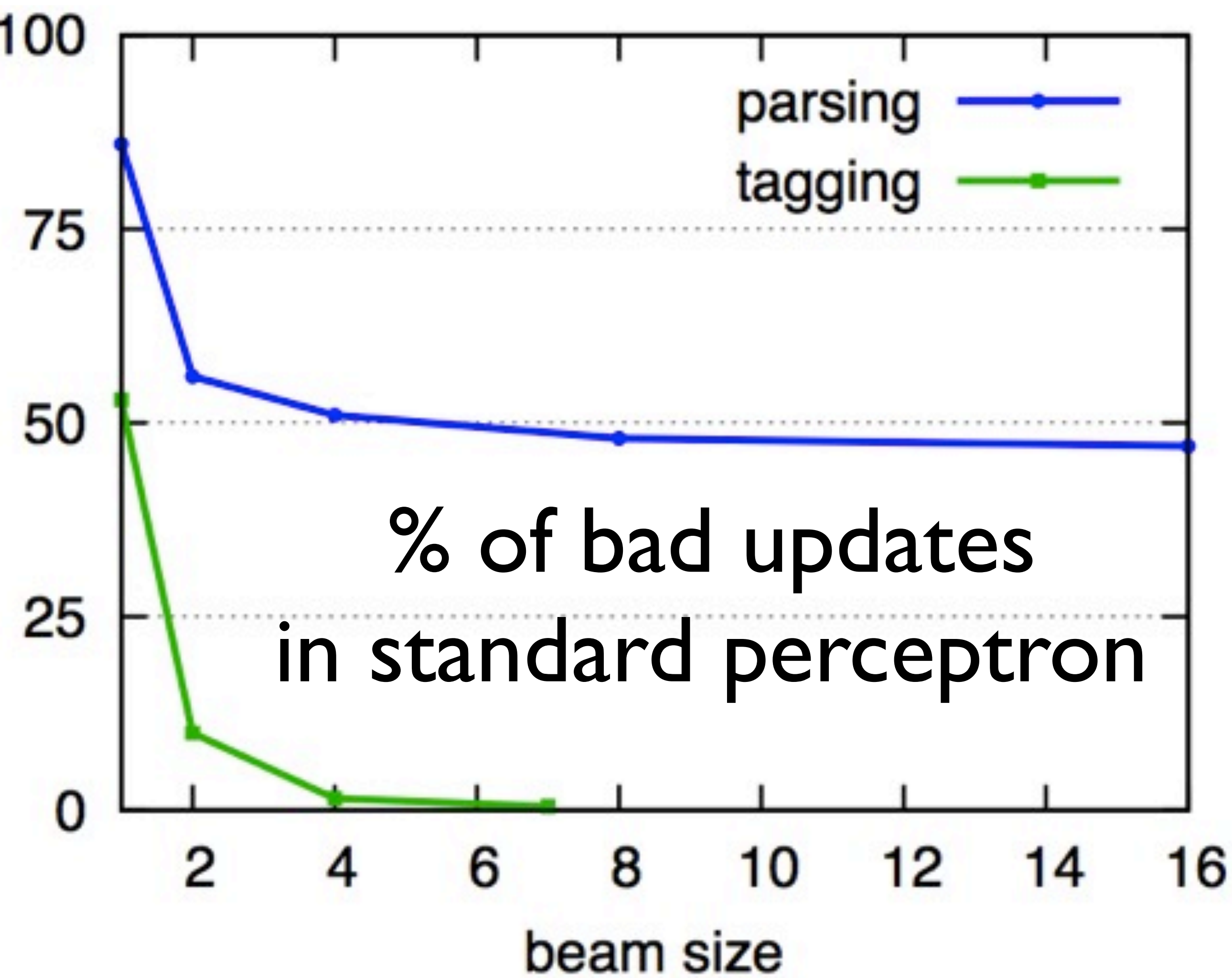
Related Work and Discussions

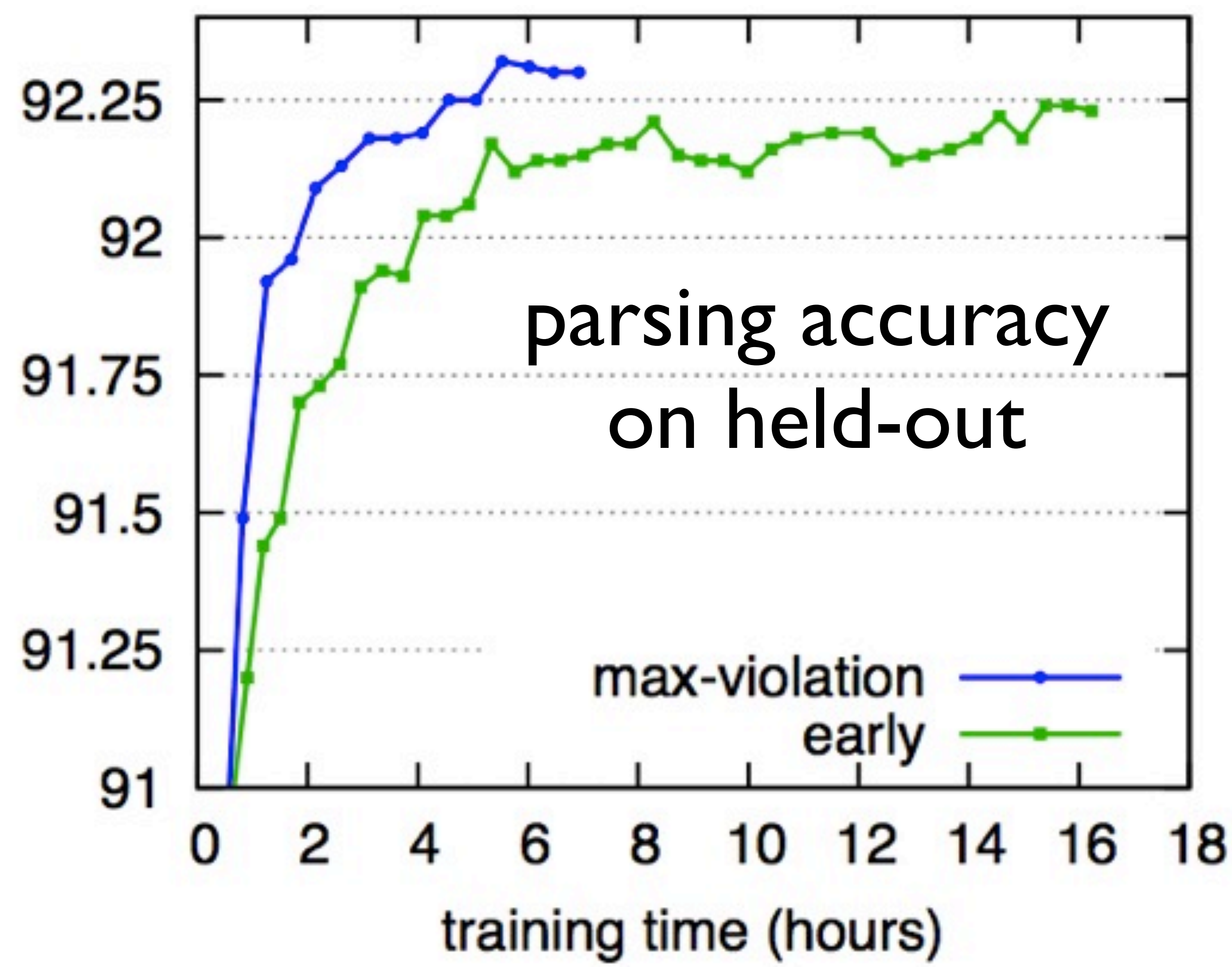
- our "violation-fixing" framework include as special cases
  - early-update (Collins and Roark, 2004)
  - a variant of LaSO (Daume and Marcu, 2005)
  - not sure about Searn (Daume et al, 2009)
- "beam-separability" or "greedy-separability" related to:
  - "algorithmic-separability" of (Kulesza and Pereira, 2007)
  - but these conditions are too strong to hold in practice
- under-generating (beam) vs. over-generating (LP-relax.)
  - Kulesza & Pereira and Martins et al (2011): LP-relaxation
  - Finley and Joachims (2008): both under and over for SVM

Conclusions

- Structured Learning with Inexact Search is Important
- Two contributions from this work:
  - theory: a general violation-fixing perceptron framework
    - convergence for inexact search under new defs of *separability*
    - subsumes previous work (early update & LaSO) as special cases
  - practice: new update methods within this framework
    - "max-violation" learns faster and better than early update
      - dramatically reducing training time by 3-5 folds
      - improves over state-of-the-art tagging and parsing systems
    - our methods are more helpful to harder search problems! :)

Thank you!





The City University of New York

my parser with max-violation update is available at:
http://acl.cs.qc.edu/~lhuang/#software

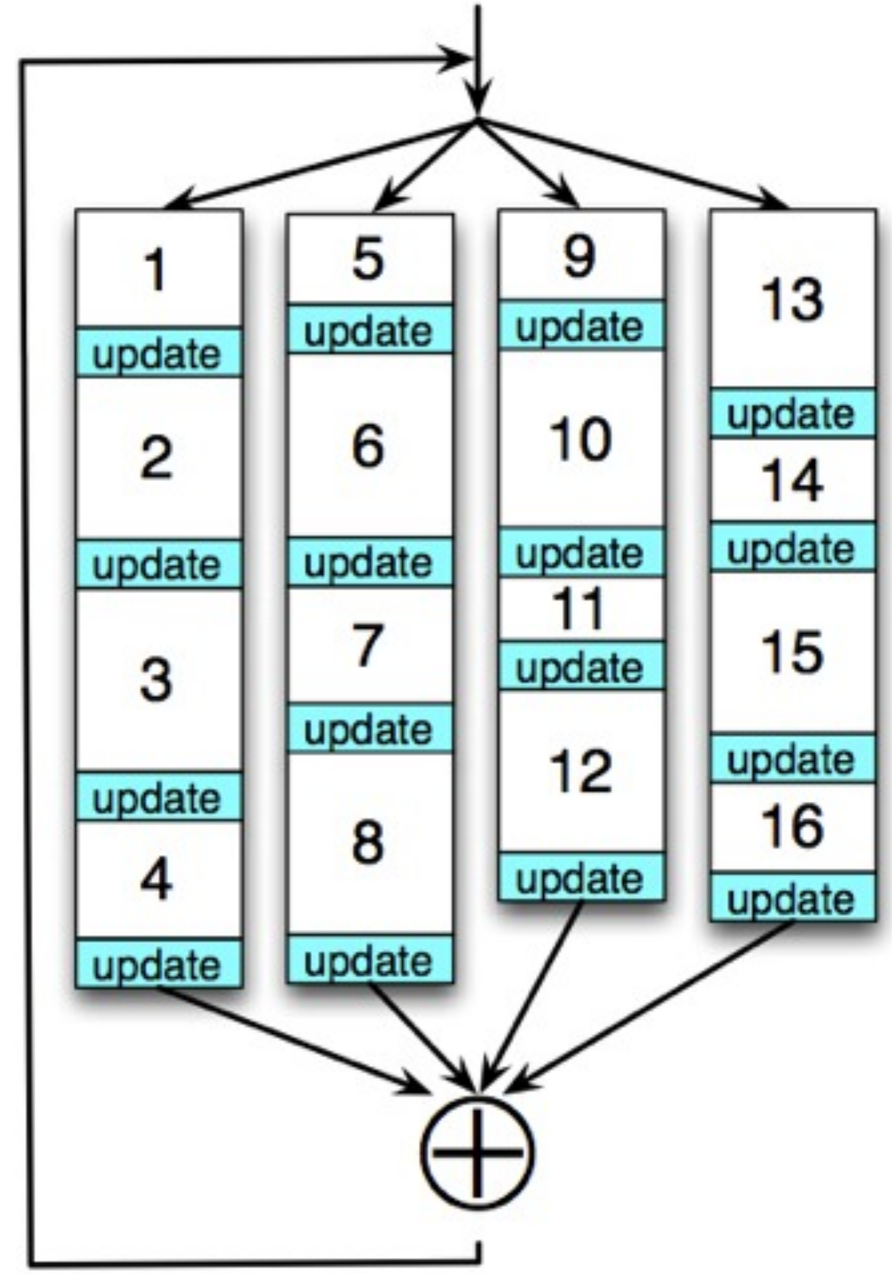
liang.huang.sh@gmail.com

Bonus Track: Parallelizing Online Learning

(K. Zhao and L. Huang, NAACL 2013)

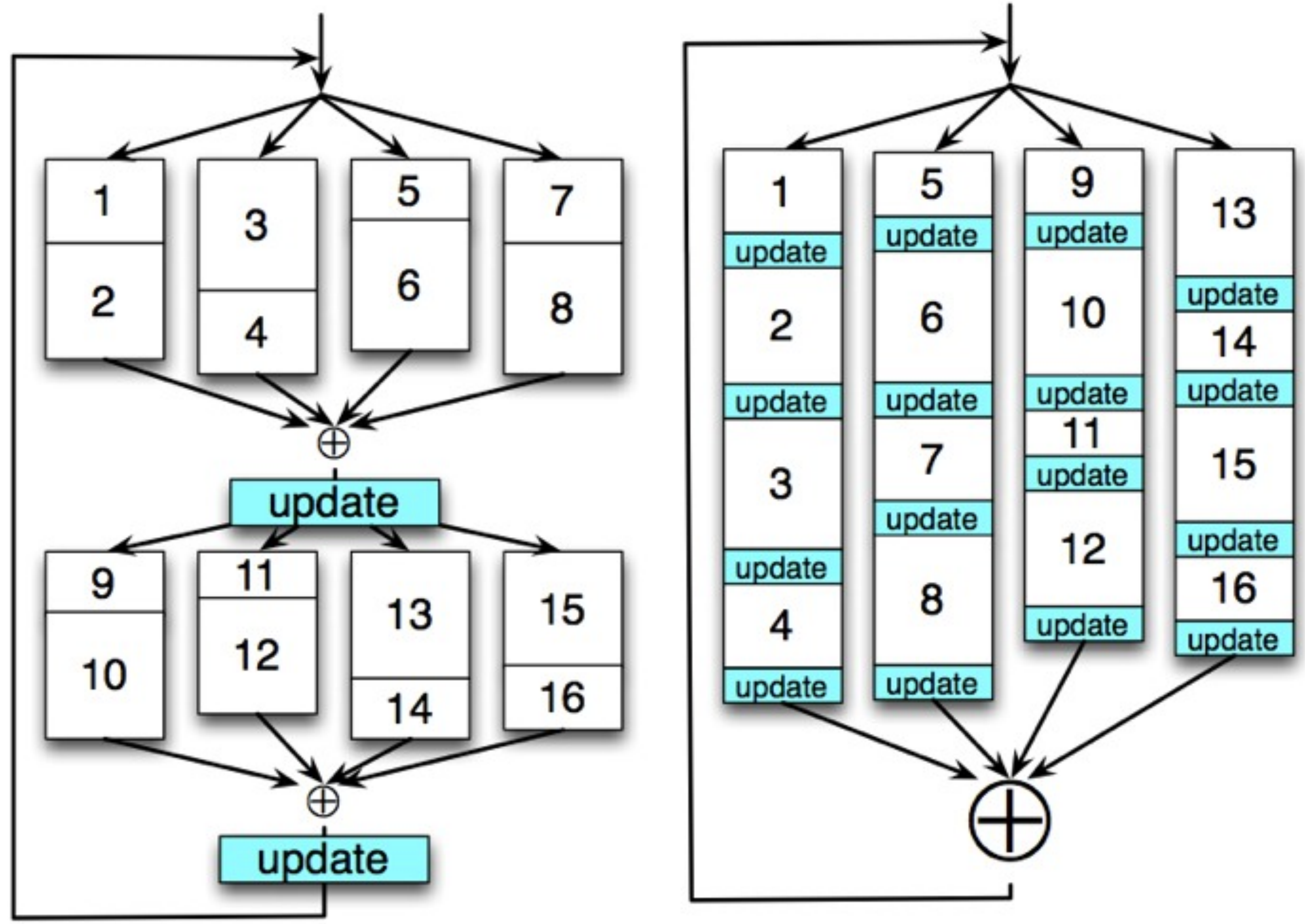
Perceptron still too slow

- even if we use very fast inexact search because
  - there is too much training data, and
  - has to go over the whole data many times to converge
- can we parallelize online learning?
  - harder than parallelizing batch learning (e.g. CRF)
  - losing dependency b/w examples
  - McDonald et al (2010): ~3-4x faster



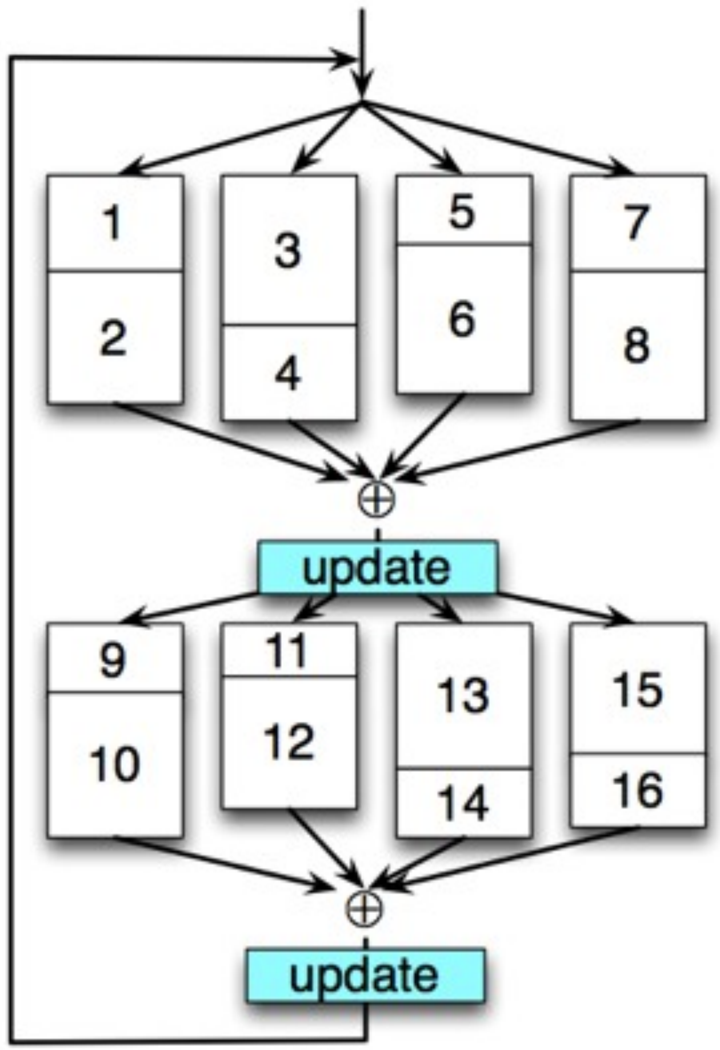
Minibatch Parallelization

- parallelize in each minibach
- do aggregate update after each minibatch
- becomes batch if minibatch size is the whole set



Minibach helps in serial also

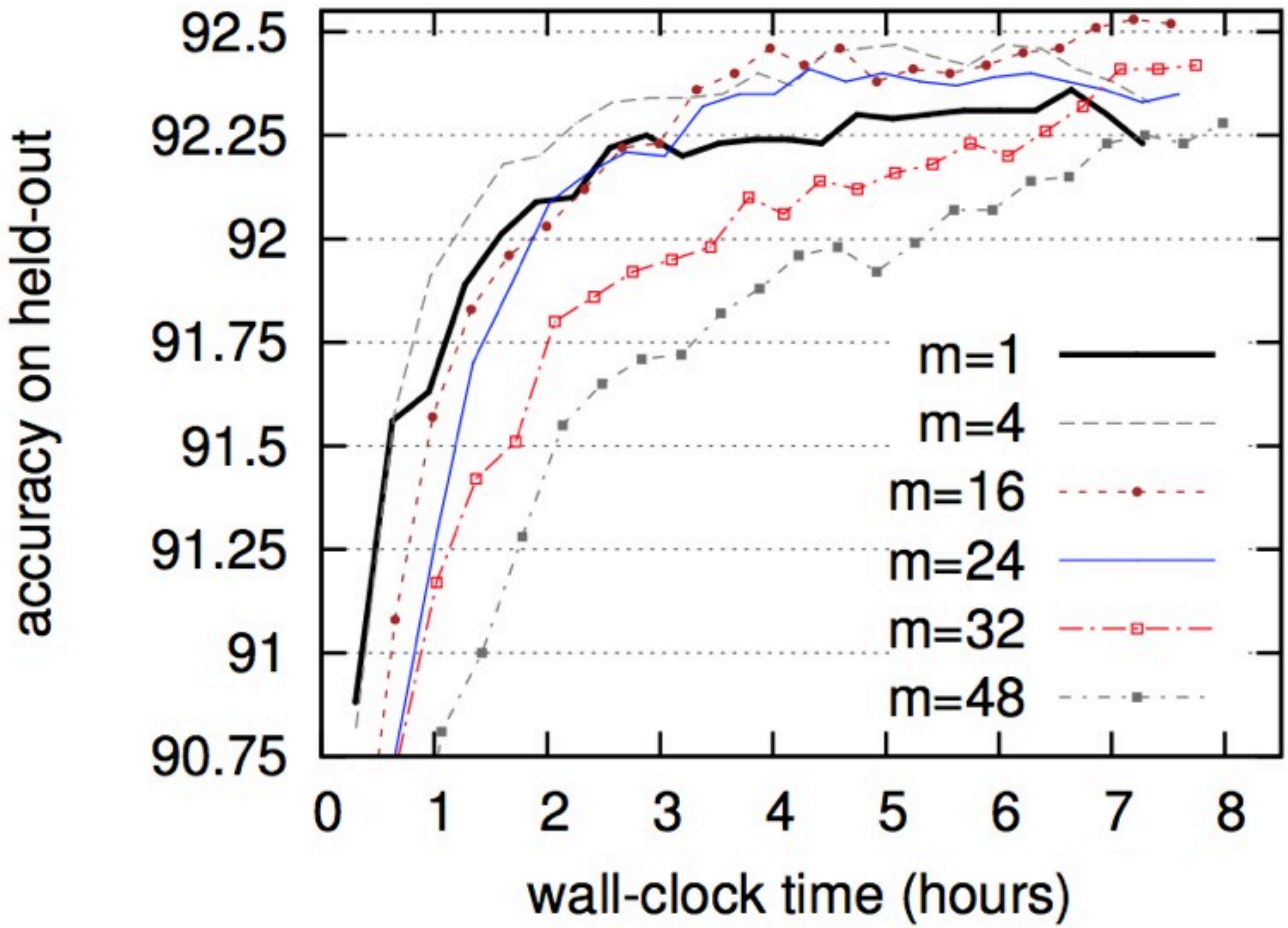
- minibatch perceptron
  - use average of updates within minibatch
  - “averaging effect” (cf. McDonal et al 2010)
  - easy to prove convergence (still R^2/δ^2)
- minibatch MIRA
  - optimization over more constraints
  - MIRA: online approximation of SVM
  - minibatch MIRA: better approximation
    - approaches SVM at maximum batch size
    - middle-ground b/w MIRA and SVM



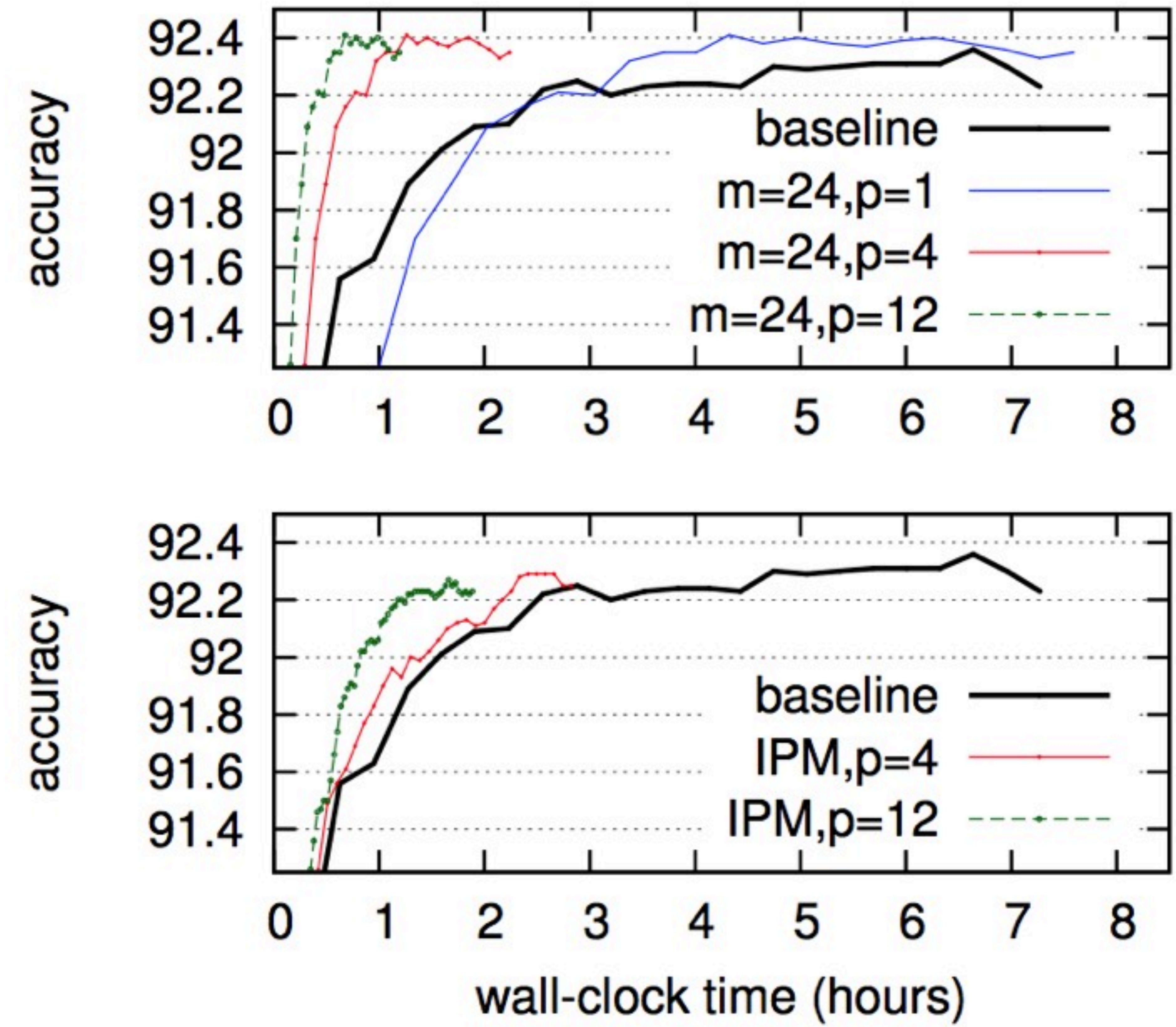
4x constrains
in each update

Parsing - MIRA - serial minibach

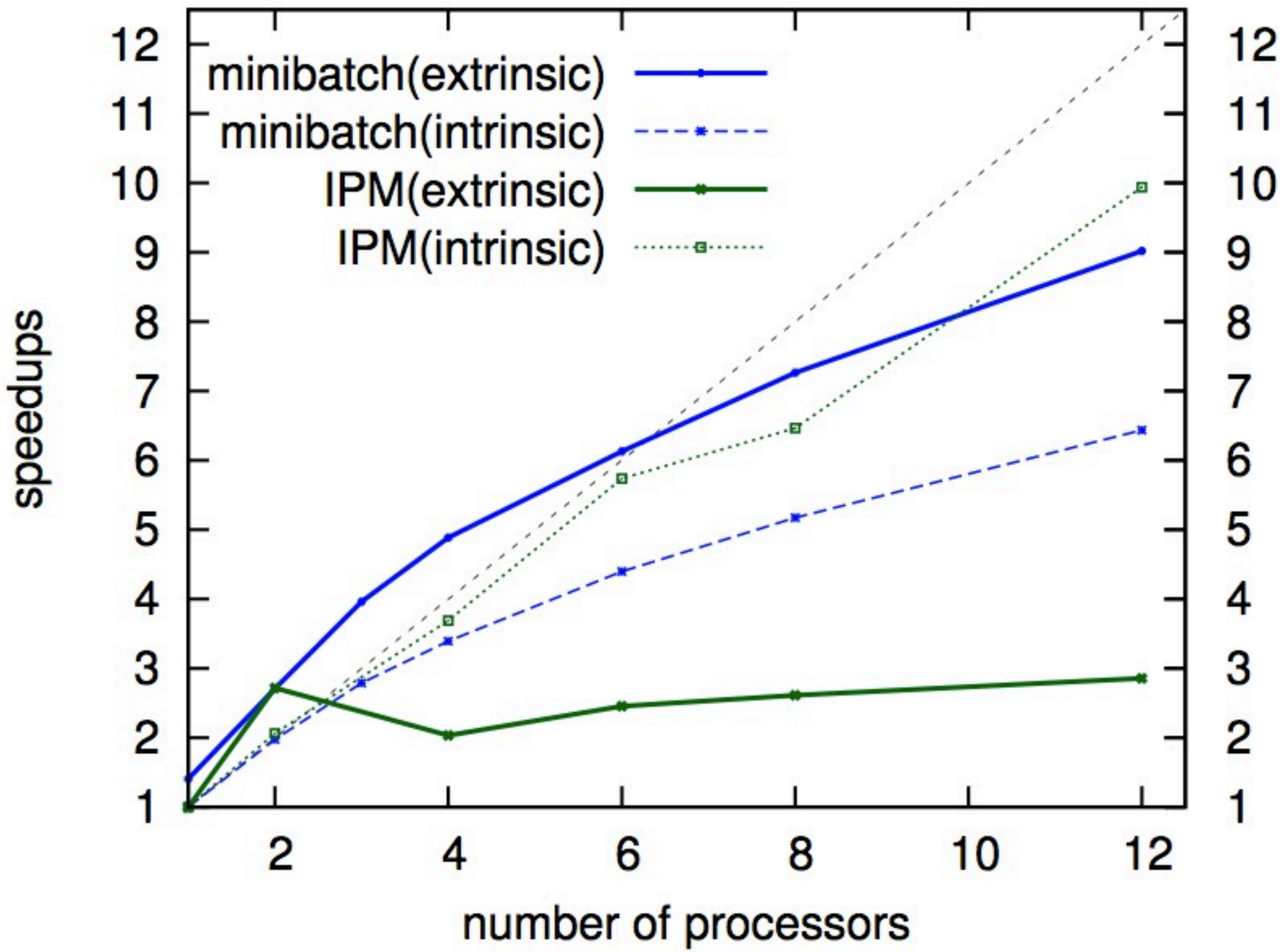
- on incremental dependency parser w/ max-violation



Comparison w/ McDonald et al 2010

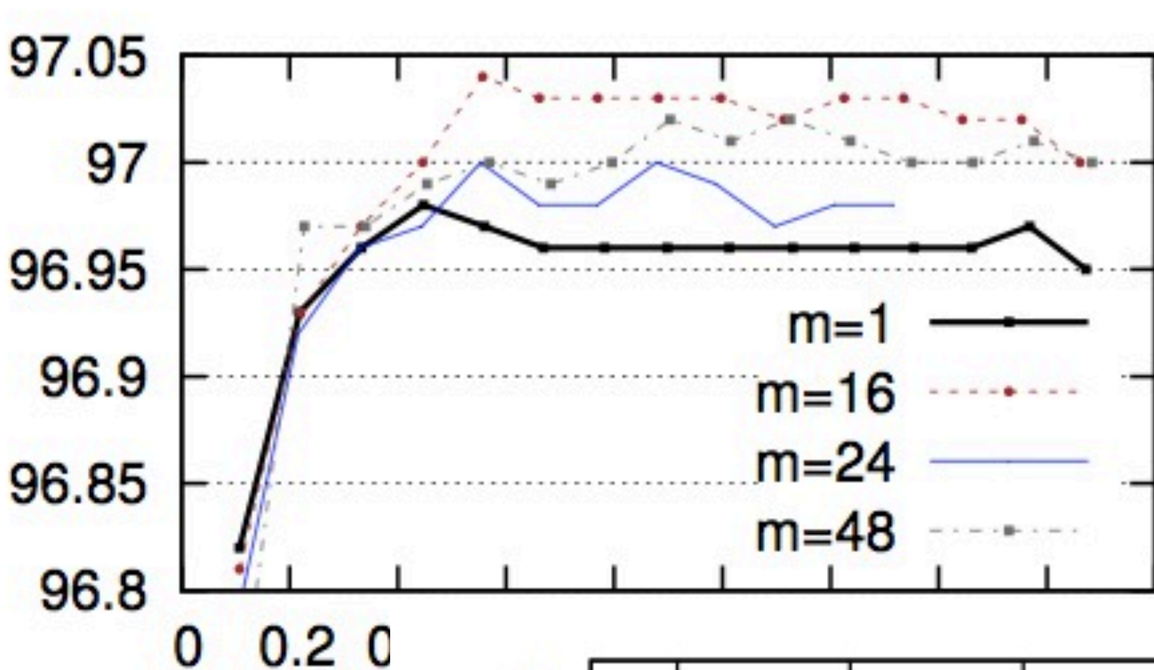


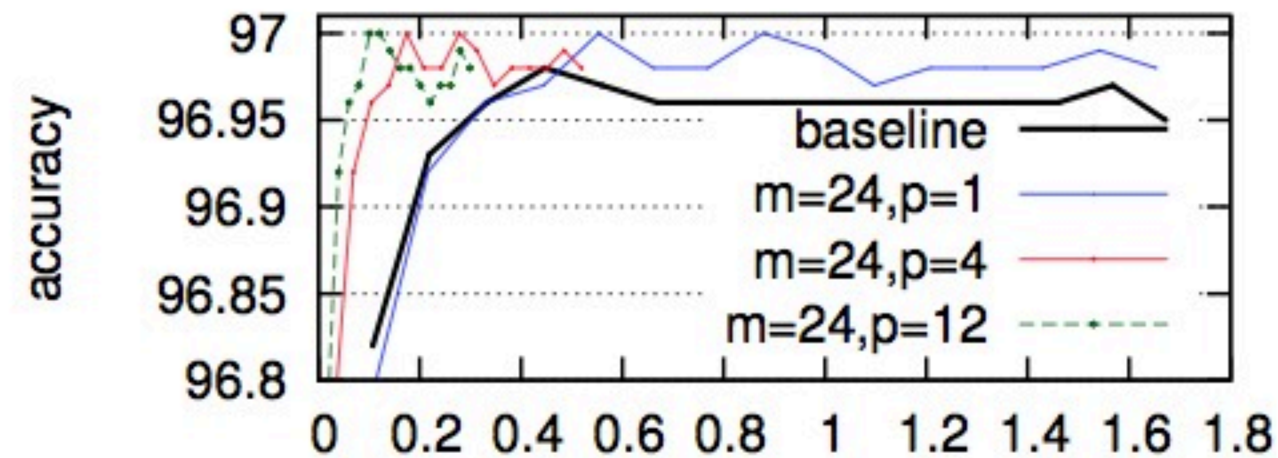
Intrinsic and Extrinsic Speedups

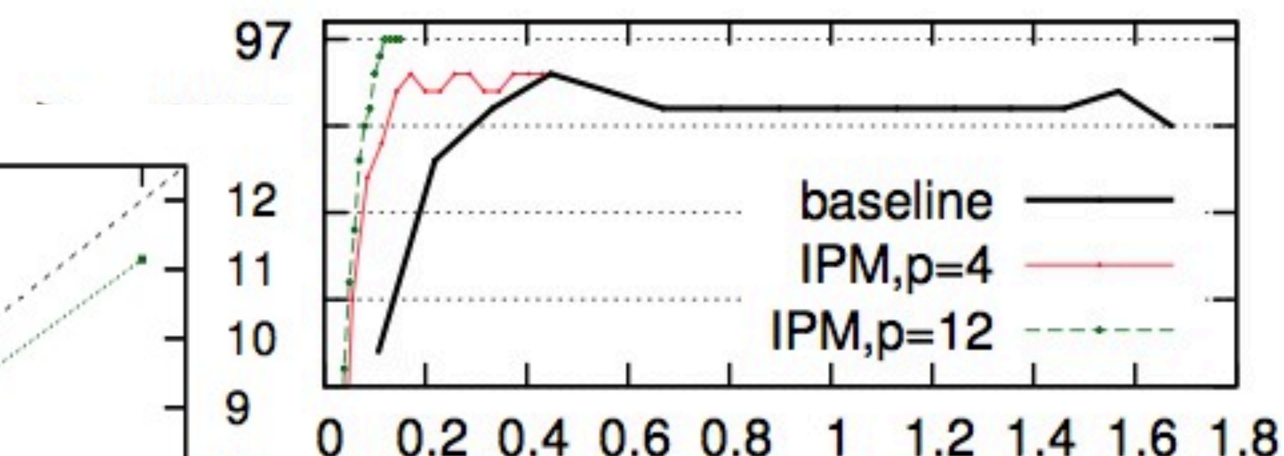


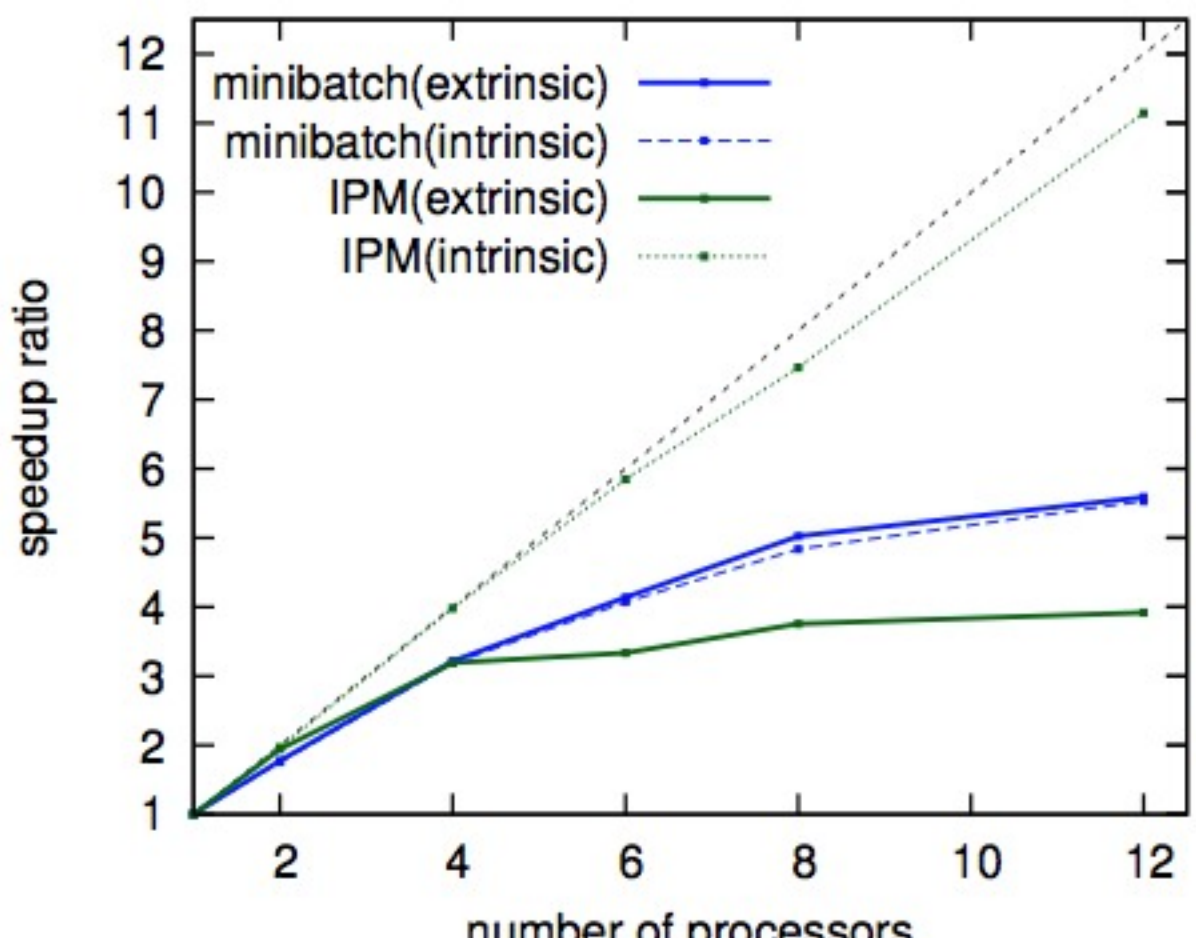
Tagging - Perceptron

- standard update with exact search

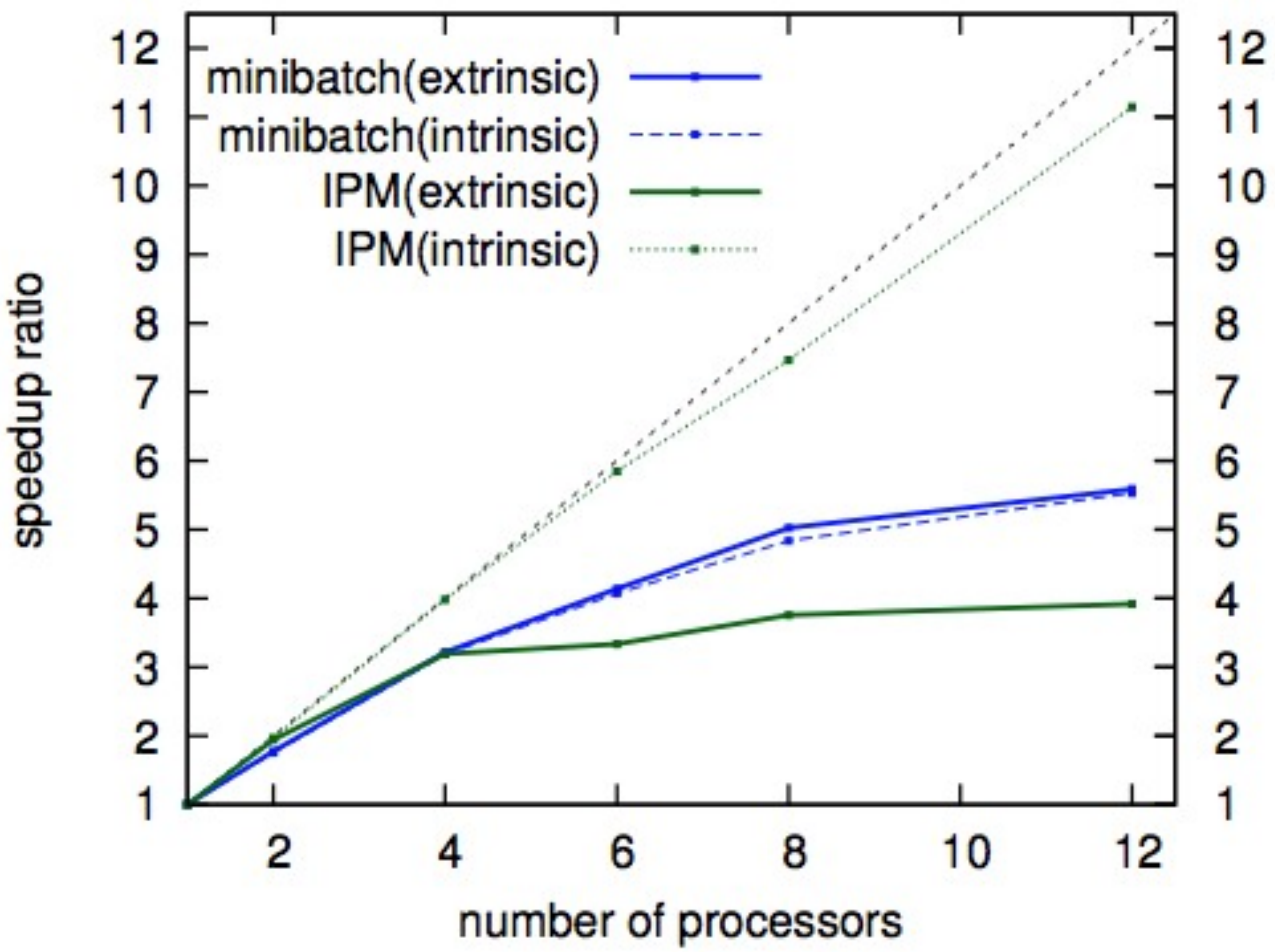


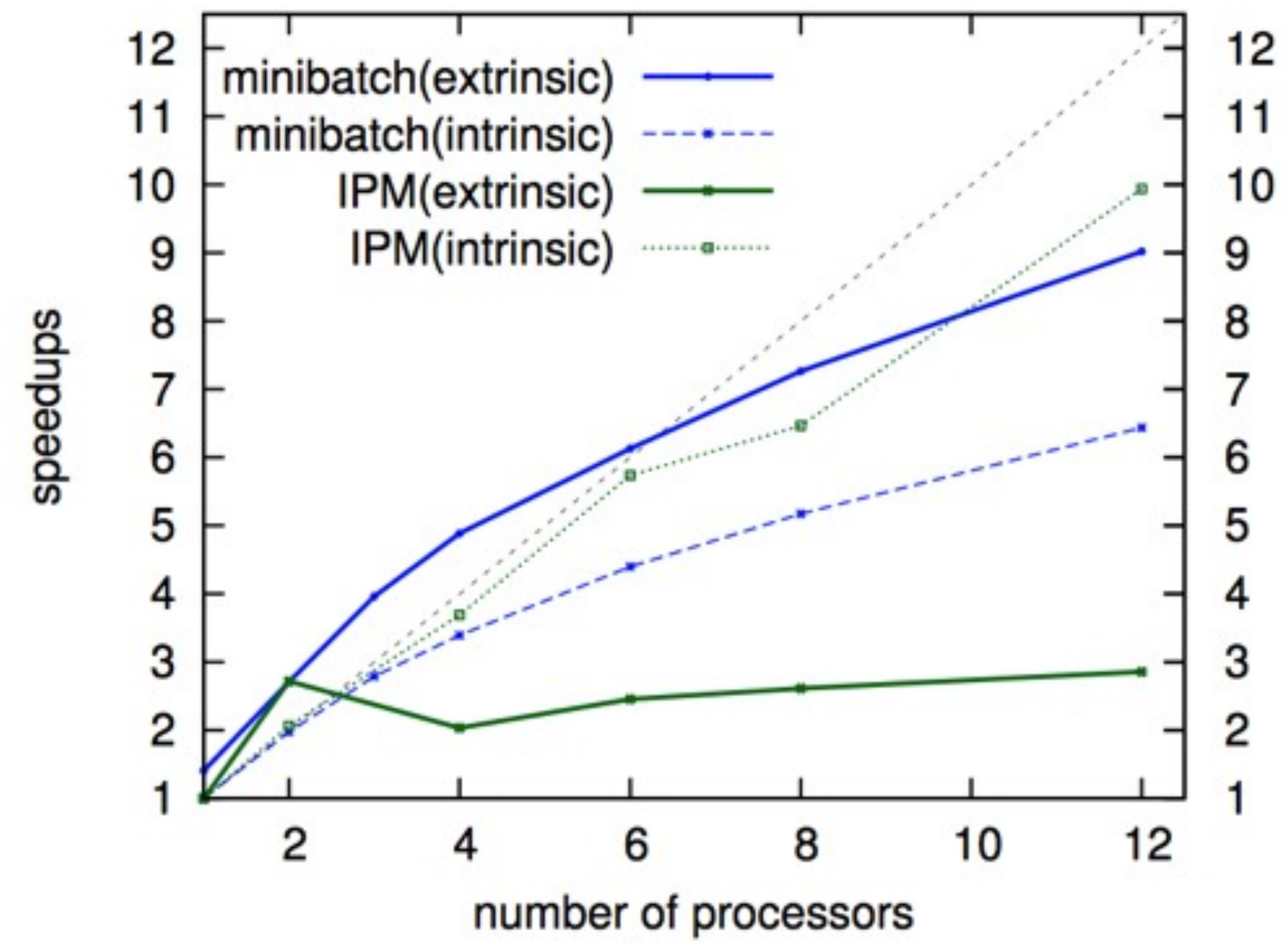


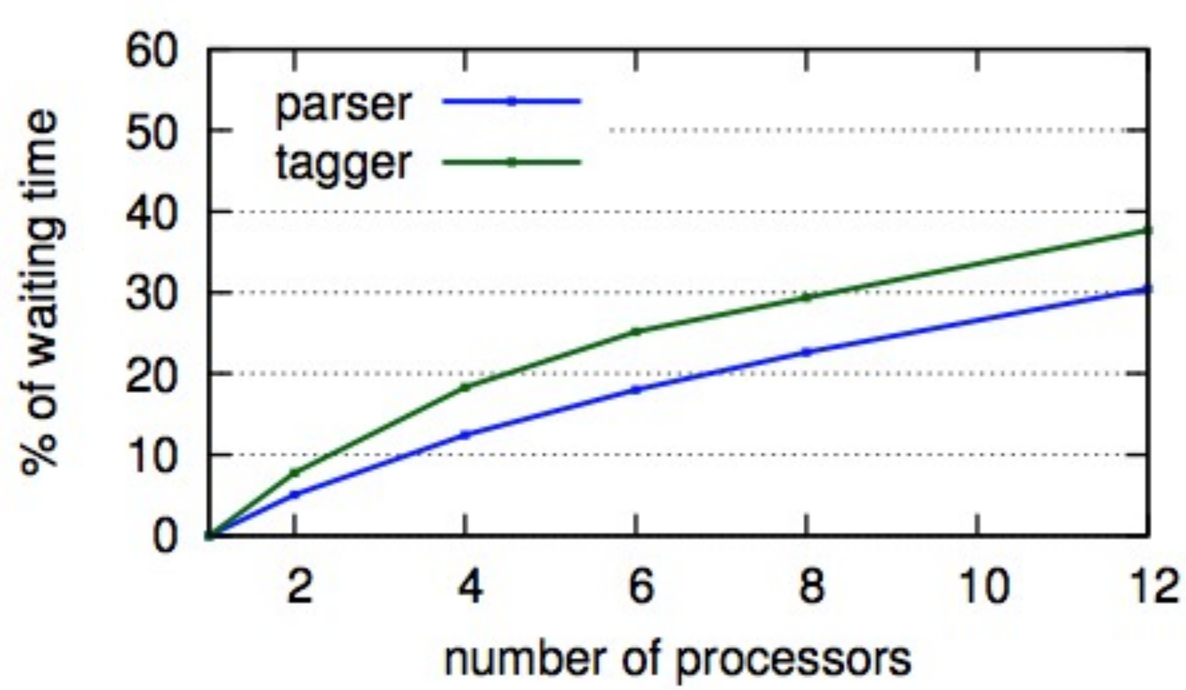




Tagging vs. Parsing







Conclusions

- Two Methods for Scaling Up Structured Learning
  - New variant of perceptron that allows fast inexact search
    - theory: a general violation-fixing perceptron framework
    - practice: new update methods within this framework
      - "max-violation" learns faster and better than early update
      - our methods are more helpful to harder search problems! :)
  - Minibatch parallelization offers significant speedups
    - much faster than previous parallelization (McDonald et al 2010)
    - even helpful in serial setting (MIRA with more constraints)