Structured Learning with Inexact Search

\[ x = \begin{cases} +1 & \text{if } y = +1 \\ -1 & \text{if } y = -1 \end{cases} \]

Liang Huang
The City University of New York (CUNY)

includes joint work with S. Phayong, Y. Guo, and K. Zhao
Structured Perceptron (Collins 02)

- 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

- 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 \(y \neq z\)

```
the man bit the dog
DT  NN  VBD  DT  NN
```
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 \( \leq \frac{R^2}{\delta^2} \)

\[ R: \text{diameter} \]

Rosenblatt => Collins
1957 => 2002
Convergence with Exact Search

training example

output space

\{N,V\} x \{N,V\}

current model

update

\(w^{(k+1)}\)

\(w^{(k)}\)

standard perceptron converges with exact search

correct label

time  flies

N  V
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 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

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

\[\Delta \Phi(x, y, z)\]

exact
inference

update weights
if \(y \neq z\)

\(x\)

\(y\)

\(z\)

update weights
if \(y \neq z\)

\(w\)

\(u\)

perceptron update:
\[w^{(k+1)} = w^{(k)} + \Delta \Phi(x, y, z)\]

\(u \cdot w^{(k+1)} = u \cdot w^{(k)} + u \cdot \Delta \Phi(x, y, z) \geq \delta\) margin (by induction)

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

\[\|u\| \cdot \|w^{(k+1)}\| \geq u \cdot w^{(k+1)} \geq k\delta\]

\[\|w^{(k+1)}\| \geq k\delta\] (part 1: upperbound)
Geometry of Convergence Proof pt 2

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

violation: incorrect label scored higher

perceptron update:

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

= \[
\|w^{(k)}\|^2 + \|\Delta \Phi(x, y, z)\|^2 + 2 \frac{w^{(k)} \cdot \Delta \Phi(x, y, z)}{2} \leq R^2
\]

diameter

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

parts 1+2 => update bounds:

\[
k \leq \frac{R^2}{\delta^2}
\]
Violation is All we need!

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

all violations

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

```
1: repeat
2: for each example (x, y) in D do
3:   (x, y', z) = FINDVIOLATION(x, y, w)
4:   if z != y then  ▷ (x, y', z) is a violation
5:     w ← w + ΔΦ(x, y', z)
6: until converged
```

standard perceptron

violations

all possible updates

update weights if y ≠ z

find violation

same mistake bound as before!
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

<table>
<thead>
<tr>
<th>time</th>
<th>flies</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>V</td>
</tr>
</tbody>
</table>

output space

\(\{N, V\} \times \{N, V\}\)

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
  - Output space: \{N,V\} x \{N,V\}

- Standard update doesn't converge because it doesn't guarantee violation.

- Early update: incorrect prefix scores higher: a violation!

- Table:
  | ✓ | ✓ | ⋮ | ✓ | × |
  | update | skip |
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!

Violation guaranteed:
- incorrect prefix scores higher up to this point

Correct label falls off beam (pruned)

Standard 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)

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

\[
\text{the } \text{man} \text{ bit} \text{ the } \text{dog}
\]

\[
\text{DT} \text{ NN} \text{ VBD} \text{ DT} \text{ NN}
\]

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

incremental dependency parsing

\[
\text{the } \text{man} \text{ bit} \text{ the } \text{dog}
\]

\[
\text{the } \text{man} \text{ bit} \text{ the } \text{dog}
\]

\[
\text{bit}
\]

\[
\text{man}
\]

\[
\text{dog}
\]

\[
\text{the}
\]

\[
\text{the}
\]

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

<table>
<thead>
<tr>
<th>Beam Size</th>
<th>Max-violation</th>
<th>Early</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97.2</td>
<td>97</td>
<td>97.2</td>
</tr>
<tr>
<td>2</td>
<td>97</td>
<td>97.0</td>
<td>97.0</td>
</tr>
<tr>
<td>3</td>
<td>96.8</td>
<td>96.6</td>
<td>96.6</td>
</tr>
<tr>
<td>4</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>5</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>6</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>7</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>8</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>9</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>10</td>
<td>96.6</td>
<td>96.4</td>
<td>96.4</td>
</tr>
</tbody>
</table>

% 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

<table>
<thead>
<tr>
<th>beam</th>
<th>iter</th>
<th>time</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>-</td>
<td>6</td>
<td>162m</td>
</tr>
<tr>
<td>early</td>
<td>4</td>
<td>6</td>
<td>37m</td>
</tr>
<tr>
<td>max-violation</td>
<td>2</td>
<td>3</td>
<td>26m</td>
</tr>
</tbody>
</table>

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)!

<table>
<thead>
<tr>
<th></th>
<th>test</th>
<th>standard</th>
<th>early</th>
<th>max-violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>79.1</td>
<td>92.1</td>
<td>92.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

30
Thank you!

% of bad updates in standard perceptron

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 minibatch
- 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 $R^2/\delta^2$)
- 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
on incremental dependency parser w/ max-violation
Comparison w/ McDonald et al 2010

The graph shows the accuracy over wall-clock time for different settings:
- **Baseline**
- **m=24, p=1**
- **m=24, p=4**
- **m=24, p=12**

Accuracy is plotted against wall-clock time (hours), with the x-axis ranging from 0 to 8 hours. The y-axis shows accuracy ranging from 91.4 to 92.4.
Intrinsic and Extrinsic Speedups

![Graph showing speedups for different methods against the number of processors. The methods include minibatch (extrinsic and intrinsic), and IPM (extrinsic and intrinsic).]
Tagging - Perceptron

- standard update with exact search

![Graphs showing accuracy and speedup ratio for different settings.](image-url)
Tagging vs. Parsing

![Graphs showing speedup ratio and speedups vs. number of processors for minibatch (extrinsic), minibatch (intrinsic), IPM (extrinsic), and IPM (intrinsic).]

- The graphs illustrate the performance of different models with varying numbers of processors.
- The y-axis represents the speedup ratio or speedups, and the x-axis is the number of processors.
- The graphs compare the performance of taggers and parsers under different conditions.
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)