Structured Prediction with Perceptron: Theory and Algorithms

Kai Zhao    Liang Huang

Graduate Center, the City University of New York

slides at: http://kaizhao.me
What is Structured Prediction?

- binary classification: output is binary
- multiclass classification: output is a (small) number
- structured classification: output is a structure (seq., tree, graph)
- part-of-speech tagging, parsing, summarization, translation
- exponentially many classes: search (inference) efficiency is crucial!

NLP is all about structured prediction!
An Example of Bad Structured Prediction
Learning: Unstructured vs. Structured

- **Binary/Multiclass**
  - Naive Bayes
  - Conditional

- **Structured Learning**
  - HMMs
  - Conditional
  - CRFs
  - Online+ Viterbi

- **Generative**
  - (Count & divide)

- **Discriminative**
  - (Expectations)
  - (Argmax)
  - (Loss-augmented argmax)
Why Perceptron (Online Learning)?

- because we want scalability on big data!
- learning time has to be linear in the number of examples
  - can make only constant number of passes over training data
  - only online learning (perceptron/MIRA) can guarantee this!
- SVM scales between $O(n^2)$ and $O(n^3)$; CRF no guarantee
- and inference on each example must be super fast
- another advantage of perceptron: just need argmax
Perceptron: from binary to structured

binary perceptron (Rosenblatt, 1959)

2 classes

multiclass perceptron (Freund/Schapire, 1999)

constant # of classes

structured perceptron (Collins, 2002)

exponential # of classes
Scalability Challenges

- inference (on one example) is too slow (even w/ DP)
- can we sacrifice search exactness for faster learning?
- would inexact search interfere with learning?
- if so, how should we modify learning?
Outline

• Overview of Structured Learning
  • Challenges in Scalability

• Structured Perceptron
  • convergence proof

• Structured Perceptron with Inexact Search

• Latent-Variable Structured Perceptron
Generic Perceptron

• perceptron is the simplest machine learning algorithm
• online-learning: one example at a time
• learning by doing
  • find the best output under the current weights
  • update weights at mistakes

\[
\begin{align*}
    x_i & \rightarrow \text{inference} \\
    w & \rightarrow z_i \\
    y_i & \rightarrow \text{update weights}
\end{align*}
\]
Structured Perceptron

**Inputs:** Training set \((x_i, y_i)\) for \(i = 1 \ldots n\)

**Initialization:** \(W = 0\)

**Define:**
\[
F(x) = \underset{y \in \text{GEN}(x)}{\text{argmax}} \Phi(x, y) \cdot W
\]

**Algorithm:**
For \(t = 1 \ldots T, i = 1 \ldots n\)
\[
z_i = F(x_i)
\]
If \((z_i \neq y_i)\)
\[
W \leftarrow W + \Phi(x_i, y_i) - \Phi(x_i, z_i)
\]

**Output:** Parameters \(W\)

*the man bit the dog*
Example: POS Tagging

- gold-standard: DT NN VBD DT NN
- the man bit the dog
- current output: DT NN NN DT NN
- the man bit the dog

- assume only two feature classes
- tag bigrams
- word/tag pairs

- weights ++: (NN, VBD) (VBD, DT) (VBD → bit)
- weights --: (NN, NN) (NN, DT) (NN → bit)
Inference: Dynamic Programming

exact inference

update weights if $y \neq z$

tagging: $O(nT^3)$

CKY parsing: $O(n^3)$
Perceptron vs. CRFs

- perceptron is online and Viterbi approximation of CRF
- simpler to code; faster to converge; ~same accuracy

**CRFs**
(Lafferty et al, 2001)

\[ \sum_{(x, y) \in D} \sum_{z \in \text{GEN}(x)} \frac{\exp(w \cdot \Phi(x, z))}{Z(x)} \]

**stochastic gradient descent (SGD)**

**structured perceptron**
(Collins, 2002)

**hard/Viterbi CRFs**

\[ \text{for } (x, y) \in D, \argmax_{z \in \text{GEN}(x)} w \cdot \Phi(x, z) \]
Perceptron Convergence Proof

- binary classification: converges iff. data is separable
- structured prediction: converges iff. data is separable
  - 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 $\leq \frac{R^2}{\delta^2}$

Novikoff => Freund & Schapire => Collins
1962                 1999                  2002
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

**perceptron update:**

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

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

(by induction)

\[
\|w^{k+1}\| \geq k\delta
\]

(part 1: lowerbound)
Geometry of Convergence Proof pt 2

summary: the proof uses 3 facts:
1. separation (margin)
2. diameter (always finite)
3. violation (guaranteed by exact search)

violation: incorrect label scored higher

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

by induction:
\[ \|w^{k+1}\|^2 \leq kR^2 \]  
\[ \|w^{k+1}\| \leq \sqrt{kR} \]  
\[ \|w^{k+1}\| \geq k\delta \]  

bound on # of updates:
\[ k \leq \frac{R^2}{\delta^2} \]
Outline

• Overview of Structured Learning
  • Challenges in Scalability
• Structured Perceptron
  • convergence proof
• Structured Perceptron with Inexact Search
• Latent-Variable Perceptron
Scalability Challenge 1: Inference

- **challenge:** search efficiency (exponentially many classes)
- **often use** dynamic programming (DP)
- **but DP** is still too slow for repeated use, e.g. parsing $O(n^3)$
- **Q:** can we sacrifice search exactness for faster learning?
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?

Q: does perceptron still work???
Bad News and Good News

- bad news: no more guarantee of convergence
  - in practice perceptron degrades a lot due to search errors
- good news: new update methods guarantee convergence
  - new perceptron variants that “live with” search errors
  - in practice they work really well w/ inexact search

A: it no longer works as is, but we can make it work by some magic.
Convergence with Exact Search

current model

w^{(k+1)}

update

w^{(k)}

Structured perceptron converges with exact search

training example

time
N
flies
V

output space

\{N,V\} \times \{N,V\}
No Convergence w/ Greedy Search

Which part of the convergence proof no longer holds?

the proof only uses 3 facts:
1. separation (margin)
2. diameter (always finite)
3. violation (guaranteed by exact search)
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

update weights if \(y \neq z\)

perceptron update:
\[
|w^{(k+1)}|^2 = |w^{(k)} + \Delta \Phi(x, y, z)|^2 \\
= |w^{(k)}|^2 + |\Delta \Phi(x, y, z)|^2 + 2 \cdot w^{(k)} \cdot \Delta \Phi(x, y, z) \\
\leq R^2 \\
\leq 0
\]

violation

inexact search doesn’t guarantee violation!
Observation: 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

```python
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
```
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
- Q: how can we always guarantee violation?
Solution 1: Early update (Collins/Roark 2004)

- Current model: \(w^{(k)}\)
- New model: \(w^{(k+1)}\)

Training example: 
- Time: N
- Flies: V

Output space: \(\{N,V\} \times \{N,V\}\)

Standard perceptron does not converge with greedy search.

Stop and update at the first mistake.
Early Update: Guarantees Violation

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} x {N,V}
```

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

early update: incorrect prefix scores higher: a violation!
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
Solution 2: Max-Violation  (Huang et al 2012)

- 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
Four Experiments

part-of-speech tagging

the man bit the dog

DT NN VBD DT NN

incremental parsing

the man bit the dog

bottom-up parsing w/ cube pruning

the man bit the dog

machine translation

the man bit the dog

那个人咬了狗
Max-Violation > Early >> Standard

- exp 1 on part-of-speech tagging w/ beam search (on CTB5)
- early and max-violation >> standard update at smallest beams
  - this advantage shrinks as beam size increases
- max-violation converges faster than early (and slightly better)

![Graph showing tagging accuracy vs. beam size and training time](image-url)
Max-Violation > Early >> Standard

- exp 1 on part-of-speech tagging w/ beam search (on CTB5)
- early and max-violation >> standard update at smallest beams
  - this advantage shrinks as beam size increases
- max-violation converges faster than early (and slightly better)

![Graph showing accuracy vs. beam size and training time](image-url)
Max-Violation > Early >> Standard

- exp 2 on incremental dependency parser (Huang & Sagae 10)
- standard update is horrible due to search errors
- early update: 38 iterations, 15.4 hours (92.24)
- max-violation: 10 iterations, 4.6 hours (92.25)

max-violation is 3.3x faster than early update
Why standard update so bad for parsing

- standard update works horribly with severe search error
- due to large number of *invalid* updates (non-violation)

\[ O(n^{11}) \Rightarrow O(nb) \]

\[ O(nT^3) \Rightarrow O(nb) \]
Exp 3: Bottom-up Parsing

- CKY parsing with cube pruning for higher-order features
- We extended our framework from graphs to hypergraphs

(UAS on Penn-YM dev)

(Zhang et al 2013)
Exp 4: Machine Translation

- standard perceptron works poorly for machine translation
  - b/c invalid update ratio is very high (search quality is low)
- max-violation converges faster than early update
- first truly successful effort in large-scale training for translation

(Yu et al 2013)
Comparison of Four Exps

- the harder your search, the more advantageous
Outline

• Overview of Structured Learning
  • Challenges in Scalability

• Structured Perceptron
  • convergence proof

• Structured Perceptron with Inexact Search

• Latent-Variable Perceptron
Learning with Latent Variables

- aka “weakly-supervised” or “partially-observed” learning
- learning from “natural annotations”; more scalable
- examples: translation, transliteration, semantic parsing...

parallel text

Bush talked with Sharon

x

布什 与 沙龙 会谈

latent derivation

y

computer

What is the largest state?

argmax(state, size)

QA pairs

コンピューター

ko n py u : ta :

transliteration

Alaska

(Liang et al 2006; Yu et al 2013; Xiao and Xiong 2013)

(Knight & Graehl, 1998; Kondrak et al 2007, etc.)

(Clark et al 2010; Liang et al 2013; Kwiatkowski et al 2013)
Learning Latent Structures

- Binary/multiclass learning
  - Naive Bayes
  - Perceptron
  - SVM

- Structured learning
  - HMMs
  - CRFs
  - Logistic regression (maxent)
  - Structured perceptron
  - Structured SVM

- Latent structures
  - EM (forward-backward)
  - Latent CRFs
  - Latent perceptron
  - Latent structured SVM
Latent Structured Perceptron

• no explicit positive signal

• hallucinate the “correct” derivation by current weights

**training example**

那 人 咬 了 狗

forced decoding space

highest-scoring gold derivation

the man bit the dog

**during online learning...**

那 人 咬 了 狗

full search space

the dog bit the man

w ← w + Φ(x, d*) − Φ(x, d̂)

(Liang et al 2006; Yu et al 2013)

reward correct

penalize wrong

highest-scoring derivation
Unconstrained Search

- example: beam search phrase-based decoding

Bushi yu Shalong juxing le huitan

???
Constrained Search

- forced decoding: must produce the exact reference translation

Bushi yu Shalong juxing le huitan → Bush held talks with Sharon

one gold derivation

gold derivation lattice

held talks

with Sharon
Search Errors in Decoding

- no explicit positive signal
- hallucinate the “correct” derivation by current weights

• training example
  
  那人咬了狗
  
  the man bit the dog

  forced decoding space
  
  $d^*$
  
  highest-scoring gold derivation
  
  • during online learning...
  
  那人咬了狗
  
  the dog bit the man

  full search space
  
  beam
  
  highest-scoring derivation

  $w \leftarrow w + \Phi(x, d^*) - \Phi(x, \hat{d})$

  reward correct
  
  penalize wrong

(Liang et al 2006; Yu et al 2013)
Search Error: Gold Derivations Pruned

gold derivation lattice

held talks

Bush

held

talks

with Sharon

real decoding beam search

should address search errors here!
Fixing Search Error 1: Early Update

- **early update** (Collins/Roark’04) when the correct falls off beam
- up to this point the incorrect prefix should score higher
- that’s a “violation” we want to fix; proof in (Huang et al 2012)
- standard perceptron does not guarantee violation
- the correct sequence (pruned) might score higher at the end!
- “invalid” update b/c it reinforces the model error

![Diagram showing early update and standard update](image_url)
Early Update w/ Latent Variable

- The gold-standard derivations are not annotated.
- We treat any reference-producing derivation as good.

The model stops decoding when all correct derivations fall off.

Violation guaranteed: incorrect prefix scores higher up to this point.
Fixing Search Error 2: Max-Violation

- early update works but learns slowly due to partial updates
- **max-violation**: use the prefix where violation is maximum
  - “worst-mistake” in the search space
- now extended to handle latent-variable
Latent-Variable Perceptron

- best in the beam
- worst in the beam
- correct sequence
- falls off the beam
- max-violation
- biggest violation
- latest
- full update (standard)
- invalid update!
- model
- std
- local
- standard update is invalid
- correct sequences
- all fall off beam
- correct seqs
Roadmap of Techniques

- Structured perceptron (Collins, 2002)
  - Latent-variable perceptron (Zettlemoyer and Collins, 2005; Sun et al., 2009)
  - Perceptron w/ inexact search (Collins & Roark, 2004; Huang et al. 2012)
  - Latent-variable perceptron w/ inexact search (Yu et al. 2013; Zhao et al. 2014)

- MT: syntactic parsing, semantic parsing, transliteration
Experiments: Discriminative Training for MT

- standard update (Liang et al’s “bold”) works poorly
- b/c invalid update ratio is very high (search quality is low)
- max-violation converges faster than early update

this explains why Liang et al ’06 failed
std ~ “bold”; local ~ “local”

![Graph showing BLEU scores and ratio of invalid updates over iterations and beam sizes.](image)
Open Problems in Theory

- latent-variable structured perceptron:
  - does it converge? under what conditions?
  - special case: POS tagging (Sun et al., 2009)

- latent-variable structured perceptron with inexact search
  - does it converge? under what conditions?

\[ y = 1 \]
\[ y = -1 \]

\[ x_{100} \quad x_{111} \quad x_{2000} \quad x_{3012} \]

\[ R \]
\[ \delta \]

\[ \text{oracle} \]

\[ \text{(Novikoff, 1962)} \]

\[ \text{#updates} \leq \frac{R^2}{\delta^2} \]

\[ z \neq y_{100} \]

\[ y_{100} \]

\[ \text{oracle} \]

\[ \text{(Collins, 2002)} \]
Open Problems in Theory

- latent-variable structured perceptron:
  - does it converge? under what conditions?
- latent-variable structured perceptron with inexact search
  - does it converge? under what conditions?

Easy to prove but unrealistic

\[
\text{#updates} \leq \frac{R^2}{\delta^2}
\]

(Collins, 2002)
Open Problems in Theory

- latent-variable structured perceptron:
  - does it converge? under what conditions?
- latent-variable structured perceptron with inexact search
  - does it converge? under what conditions?

\[
\text{easy to prove but unrealistic} \quad \text{ideal situation but hard to prove}
\]

\[
\text{oracle vector}\quad (\text{Sun et al, 2009})
\]

\[
\#\text{updates} \leq \frac{R^2}{\delta^2}
\]

???
Final Conclusions

- online structured learning is simple and powerful
- search efficiency is the key challenge
- search errors do interfere with learning
  - but we can use violation-fixing perceptron w/ inexact search
- we can extend perceptron to learn latent structures

![Diagram showing correct sequence, worst in the beam, falls off the beam, biggest violation, last valid update, full (standard), and invalid update!](image)