Structured Prediction with Perceptron: Theory and Algorithms

Kai Zhao
Dept. Computer Science
Graduate Center, the City University of New York
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!
Learning: Unstructured vs. Structured

binary/multiclass
- naive bayes
- Conditional

structured learning
- HMMs
- Conditional

- logistic regression (maxent)

- Online+ Viterbi

- perceptron
- max margin
- SVM

- structured perceptron
- max margin
- structured SVM

- generative
  (count & divide)

- discriminative
  (expectations)

- Conditional

- CRFs

- (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
- Update weights if $y \neq z$
- Exact inference
- Trivial inference

**Multiclass Perceptron** (Freund/Schapire, 1999)

- Constant # of classes
- Update weights if $y \neq z$
- Exact inference
- Easy inference

**Structured Perceptron** (Collins, 2002)

- Exponential # of classes
- Update weights if $y \neq z$
- Exact inference
- Hard inference

Examples:
- The man bit the dog
- 那人咬了狗
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
Structured Perceptron

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

Initialization: \(W = 0\)

Define: \(F(x) = \arg \max_{y \in \text{GEN}(x)} \Phi(x, y) \cdot W\)

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
- \(x_i \rightarrow y_i \rightarrow z_i \rightarrow W\)
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)

hard/Viterbi CRFs

structured perceptron
(Collins, 2002)

for \((x, y) \in D\), \(\text{argmax}_{z \in \text{GEN}(x)} w \cdot \Phi(x, z)\)
**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 $\leq \frac{R^2}{\delta^2}$

\[ R: \text{diameter} \]

Novikoff $\Rightarrow$ Freund & Schapire $\Rightarrow$ 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

update current model \(w^{(k)}\) to \(w^{(k+1)}\)

exact inference

update weights

\(x \rightarrow z\) if \(y \neq z\)

\(y \rightarrow z\)

perceptron update:

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

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

(by induction)

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

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

violation

by induction:
\[ \|\mathbf{w}^{k+1}\|^2 \leq k R^2 \]
\[ \|\mathbf{w}^{k+1}\| \leq \sqrt{k} R \]

combine with:
\[ \|\mathbf{w}^{k+1}\| \geq k \delta \]

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???

\[
\text{the man bit the dog} \quad \text{DT NN VBD DT NN}
\]

\[
x \rightarrow \text{inexact inference} \rightarrow z \quad \text{update weights if } y \neq z
\]

greedy search
beam search
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

structured perceptron converges with exact search
No Convergence w/ Greedy Search

current model

new model

training example
time flies
N V

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

structured perceptron
does not converge
with 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

violation: incorrect label scored higher

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

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!

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)
Violating-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 \(z \neq y\) then (\(x, y', z\) is a violation)
5: \(w \leftarrow w + \Delta \Phi(x, y', z)\)
6: until converged
```

same mistake bound as before!

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

Training example:

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

Current model:

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

New model:

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

Update:

- Update:
  - \(\Delta \phi(x, y, z)\)

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

- Training example:
  - Time: flies
  - Label: N, V

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

- Standard update:
  - Doesn’t converge
  - Because it doesn’t guarantee violation

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

- Table:
  - \( \begin{array}{c|c|c|c|c}
      \ \checkmark \ & \ \checkmark \ & \cdots \ & \checkmark \ & \times \\
      \leftarrow \ & \text{update} \ & \rightarrow \ & \text{skip} \ & \rightarrow
  \end{array} \)
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!

![Diagram showing early update and standard update with correct and incorrect labels.](image-url)
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

the man bit the dog

machine translation

the man bit the dog

那人咬了狗
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 and training time vs. beam size](image)

- **best accuracy vs. beam size**
  - max-violation
  - early
  - standard

- tagging accuracy on held-out vs. training time (hours)
  - beam=1 (greedy)
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)
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)

![Graphs showing parsing and tagging accuracy](image)

- Parsing:
  - $O(n^4) \Rightarrow O(nb)$
  - $b=8$

- Tagging:
  - $O(nT^3) \Rightarrow O(nb)$
  - $b=1$

![Graph showing % of invalid updates](image)

% of invalid updates in standard update
Exp 3: Bottom-up Parsing

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

(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

![Graphs showing comparison of four experiments](image-url)
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...

(Bush talked with Sharon) $\rightarrow$ (computer) $\rightarrow$ (argmax(state, size))

parallel text

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

transliteration

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

QA pairs

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

- Binary/multiclass
  - Naive Bayes
  - Conditional

- Structured learning
  - HMMs
  - EM (forward-backward)

- Logistic regression (maxent)
  - CRFs

- Perceptron
  - Online+ Viterbi
  - Max margin
  - Structured perceptron
  - Structured SVM
  - Latent perceptron
  - Latent structured SVM
Latent Structured Perceptron

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

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

\begin{itemize}
  \item \textit{training example}
  \begin{align*}
    \text{那人咬了狗} & \quad x \\
    \text{the man bit the dog} & \quad y \\
  \end{align*}

  \item \textit{during online learning...}
  \begin{align*}
    \text{那人咬了狗} & \quad x \\
    \text{the dog bit the man} & \quad y \\
  \end{align*}
\end{itemize}

\text{(Liang et al 2006; Yu et al 2013)}
Unconstrained Search

- example: beam search phrase-based decoding

*Bushi yu Shalong juxing le huitan***

Bush 💡... meeting

Bush 💡... talks

Bush 💡... talk

Sharon 💡... Shalong

Sharon 💡... Sharon
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

强迫解码空间

highest-scoring gold derivation

the man bit the dog

during online learning...

全搜索空间

highest-scoring derivation

那 人 咬 了 狗

the dog bit the man

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

(Liang et al 2006; Yu et al 2013)

problem: search errors

reward correct
penalize wrong
Search Error: Gold Derivations Pruned

gold derivation lattice

Bush

held talks

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

Model

- correct sequence falls off beam (pruned)
- violation guaranteed: incorrect prefix scores higher *up to this point*
- standard update (no guarantee!)
Early Update w/ Latent Variable

- the gold-standard derivations are *not* annotated
- we treat any reference-producing derivation as good

Gold derivation lattice

- Model
- Early update
- Violation guaranteed: incorrect *prefix* scores higher *up to this point*
- All correct derivations fall off
- Stop decoding
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

biggest violation

last valid update

invalid update!

model

full (standard)

std

local

standard update is invalid

best in the beam

worst in the beam

$h_{i_e}^-$ $h_{i_e}^+$ $h_{i_e}^*$ $h_{i_*}^-$ $h_{i_*}^+$ $h_{i_*}^*$

$y_{|x|}$ $h_{|x|}^+$

correct sequences

all fall off beam
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”

- standard latent-variable perceptron
- Local update is invalid
- MaxForce
- early
- MERT
- local
- standard

BLEU

Number of iteration

Ratio

beam size

Ratio of invalid updates
+non-local feature

20 18 16 14 12 10 8 6 4 2

50% 60% 70% 80% 90%

20 18 16 14 12 10 8 6 4 2

26 25 24 23 22 21 20 19 18 17
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