Structured Perceptron with Inexact Search

Liang Huang
Information Sciences Institute
University of Southern California
liang.huang.sh@gmail.com

Suphan Fayong
Dept. of Computer Science
University of Southern California
suphan.ying@gmail.com

Yang Guo
Bloomberg L.P.
New York, NY
yangg86@gmail.com

Abstract

Most existing theory of structured prediction assumes exact inference, which is often intractable in many practical problems. This leads to the routine use of approximate inference such as beam search but there is not much theory behind it. Based on the structured perceptron, we propose a general framework of "violation-fixing" perceptrons for inexact search with a theoretical guarantee for convergence under new separability conditions. This framework subsumes and justifies the popular heuristic "early-update" for perceptron with beam search (Collins and Roark, 2004). We also propose several new update methods within this framework, among which the "max-violation" method dramatically reduces training time (by 3 fold as compared to early-update) on state-of-the-art part-of-speech tagging and incremental parsing systems.

1 Introduction

Discriminative structured prediction algorithms such as conditional random fields (Lafferty et al., 2001), structured perceptron (Collins, 2002), max-margin markov networks (Taskar et al., 2003), and structural SVMs (Tsochantaridis et al., 2005) lead to state-of-the-art performance on many structured prediction problems such as part-of-speech tagging, sequence labeling, and parsing. But despite their success, there remains a major problem: these learning algorithms all assume exact inference (over an exponentially-large search space), which is needed to ensure their theoretical properties such as convergence. This exactness assumption, however, rarely holds in practice since exact inference is often intractable in many important problems such as machine translation (Liang et al., 2006), incremental parsing (Collins and Roark, 2004; Huang and Sagae, 2010), and bottom-up parsing (McDonald and Pereira, 2006; Huang, 2008). This leads to routine use of approximate inference such as beam search as evidenced in the above-cited papers, but the inexactness unfortunately abandons existing theoretical guarantees of the learning algorithms, and besides notable exceptions discussed below and in Section 7, little is known for *theoretical properties* of structured prediction under inexact search.

Among these notable exceptions, many examine how and which approximations break theoretical guarantees of existing learning algorithms (Kulesza and Pereira, 2007; Finley and Joachims, 2008), but we ask a deeper and practically more useful question: can we *modify* existing learning algorithms to *accommodate* the inexactness in inference, so that the theoretical properties are still maintained?

For the structured perceptron, Collins and Roark (2004) provides a partial answer: they suggest variant called "early update" for beam search, which updates on partial hypotheses when the correct solution falls out of the beam. This method works significantly better than standard perceptron, and is followed by later incremental parsers, for instance in (Zhang and Clark, 2008; Huang and Sagae, 2010). However, two problems remain: first, up till now there has been no theoretical justification for early update; and secondly, it makes learning extremely slow as witnessed by the above-cited papers because it only learns on partial examples and often requires 15–40 iterations to converge while normal perceptron converges in 5–10 iterations (Collins, 2002).

We develop a theoretical framework of "violation-fixing" perceptron that addresses these challenges. In particular, we make the following contributions:

- We show that, somewhat surprisingly, exact

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 142–151,
Montréal, Canada, June 3-8, 2012.

search is **not** required by perceptron convergence. All we need is that each update involves a "violation", i.e., the 1-best sequence has a higher model score than the correct sequence. Such an update is considered a "valid update", and any perceptron variant that maintains this is bound to converge. We call these variants "violation-fixing perceptrons" (Section 3.1).

- This theory explains why standard perceptron update may fail to work with inexact search, because violation is no longer guaranteed: the correct structure might indeed be preferred by the model, but was pruned during the search process (Sec. 3.2). Such an update is thus considered invalid, and experiments show that invalid updates lead to bad learning (Sec. 6.2).

- We show that the early update is always valid and is thus a special case in our framework; this is the first theoretical justification for early update (Section 4). We also show that (a variant of) LaSO (Daumé and Marcu, 2005) is another special case (Section 7).

- We then propose several other update methods within this framework (Section 5). Experiments in Section 6 confirm that among them, the max-violation method can learn equal or better models with dramatically reduced learning times (by 3 fold as compared to early update) on state-of-the-art part-of-speech tagging (Collins, 2002)[1] and incremental parsing (Huang and Sagae, 2010) systems. We also found strong correlation between search error and invalid updates, suggesting that the advantage of valid update methods is more pronounced with harder inference problems.

Our techniques are widely applicable to other structured prediction problems which require inexact search like machine translation and protein folding.

2 Structured Perceptron

We review the convergence properties of the standard structured perceptron (Collins, 2002) in our own notations that will be reused in later sections for non-exact search. We first define a new concept:

Algorithm 1 Structured Perceptron (Collins, 2002).

Input: data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$ and feature map $\mathbf{\Phi}$
Output: weight vector $\mathbf{w}$
Let: $\textsc{Exact}(x, \mathbf{w}) \triangleq \mathbf{argmax}_{s \in \mathcal{Y}(x)} \mathbf{w} \cdot \mathbf{\Phi}(x, s)$
Let: $\Delta\mathbf{\Phi}(x, y, z) \triangleq \mathbf{\Phi}(x, y) - \mathbf{\Phi}(x, z)$
1: **repeat**
2: **for each** example (x, y) **in** D **do**
3: $z \leftarrow \textsc{Exact}(x, \mathbf{w})$
4: **if** $z \neq y$ **then**
5: $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
6: **until** converged

Definition 1. The **standard confusion set** $C_{\text{s}}(D)$ for training data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$ is the set of triples (x, y, z) where z is a wrong label for input x:

$$C_{\text{s}}(D) \triangleq \{(x, y, z) \mid (x, y) \in D, z \in \mathcal{Y}(x) - \{y\}\}.$$

The rest of the theory, including separation and violation, all builds upon this concept. We call such a triple $S = \langle D, \mathbf{\Phi}, C \rangle$ a **training scenario**, and in the remainder of this section, we assume $C = C_{\text{s}}(D)$, though later we will define other confusion sets to accommodate other update methods.

Definition 2. The training scenario $S = \langle D, \mathbf{\Phi}, C \rangle$ is said to be **linearly separable** (i.e., dataset D is linearly separable in C by representation $\mathbf{\Phi}$) with **margin** $\delta > 0$ if there exists an *oracle vector* $\mathbf{u}$ with $\|\mathbf{u}\| = 1$ such that it can correctly classify all examples in D (with a gap of at least δ), i.e., $\forall (x, y, z) \in C, \mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z) \geq \delta$. We define the *maximal margin* $\delta(S)$ to be the maximal such margin over all unit oracle vectors:

$$\delta(S) \triangleq \max_{\|\mathbf{u}\|=1} \min_{(x,y,z) \in C} \mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z).$$

Definition 3. A triple (x, y, z) is said to be a **violation** in training scenario $S = \langle D, \mathbf{\Phi}, C \rangle$ with respect to weight vector $\mathbf{w}$ if $(x, y, z) \in C$ **and** $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y, z) \leq 0$.

Intuitively, this means model $\mathbf{w}$ is possible to mislabel example x (though not necessarily to z) since y is not its single highest scoring label under $\mathbf{w}$.

Lemma 1. *Each update triple (x, y, z) in Algorithm 1 (line 5) is a violation in $S = \langle D, \mathbf{\Phi}, C_{\text{s}}(D) \rangle$.*

[1] Incidentally, we achieve the best POS tagging accuracy to date (97.35%) on English Treebank by early update (Sec. 6.1).

Proof. $z = \text{EXACT}(x, \mathbf{w})$, thus for all $z' \in \mathcal{Y}(x)$, $\mathbf{w} \cdot \mathbf{\Phi}(x, z) \geq \mathbf{w} \cdot \mathbf{\Phi}(x, z')$, i.e., $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y, z) \leq 0$. On the other hand, $z \in \mathcal{Y}(x)$ and $z \neq y$, so $(x, y, z) \in C_{\text{s}}(D)$. □

This lemma basically says exact search guarantees violation in each update, but as we will see in the convergence proof, violation itself is more fundamental than search exactness.

Definition 4. The **diameter** $R(S)$ for scenario $S = \langle D, \mathbf{\Phi}, C \rangle$ is $\max_{(x,y,z) \in C} \|\Delta\mathbf{\Phi}(x, y, z)\|$.

Theorem 1 (convergence, Collins)**.** *For a separable training scenario $S = \langle D, \mathbf{\Phi}, C_{\text{s}}(D) \rangle$ with $\delta(S) > 0$, the perceptron algorithm in Algorithm 1 will make finite number of updates (before convergence):*

$$err(S) \leq R^2(S)/\delta^2(S).$$

Proof. Let $\mathbf{w}^{(k)}$ be the weight vector **before** the kth update; $\mathbf{w}^{(0)} = \mathbf{0}$. Suppose the kth update happens on the triple (x, y, z). We will bound $\|\mathbf{w}^{(k+1)}\|$ from two directions:

1. $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)$. Since scenario S is separable with max margin $\delta(S)$, there exists a unit oracle vector $\mathbf{u}$ that achieves this margin. Dot product both sides with $\mathbf{u}$, we have

$$\begin{aligned} \mathbf{u} \cdot \mathbf{w}^{(k+1)} &= \mathbf{u} \cdot \mathbf{w}^{(k)} + \mathbf{u} \cdot \Delta\mathbf{\Phi}(x, y, z) \\ &\geq \mathbf{u} \cdot \mathbf{w}^{(k)} + \delta(S) \end{aligned}$$

   by Lemma 1 that $(x, y, z) \in C_{\text{s}}(D)$ and by the definition of margin. By induction, we have $\mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta(S)$. Since for any two vectors $\mathbf{a}$ and $\mathbf{b}$ we have $\|\mathbf{a}\|\|\mathbf{b}\| \geq \mathbf{a} \cdot \mathbf{b}$, thus $\|\mathbf{u}\|\|\mathbf{w}^{(k+1)}\| \geq \mathbf{u} \cdot \mathbf{w}^{(k+1)} \geq k\delta(S)$. As $\mathbf{u}$ is a unit vector, we have $\|\mathbf{w}^{(k+1)}\| \geq k\delta(S)$.

2. On the other hand, since $\|\mathbf{a} + \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 + 2\,\mathbf{a} \cdot \mathbf{b}$ for any vectors $\mathbf{a}$ and $\mathbf{b}$,we have

$$\begin{aligned} \|\mathbf{w}^{(k+1)}\|^2 &= \|\mathbf{w}^{(k)} + \Delta\mathbf{\Phi}(x, y, z)\|^2 \\ &= \|\mathbf{w}^{(k)}\|^2 + \|\Delta\mathbf{\Phi}(x, y, z)\|^2 \\ &\quad + 2\,\underline{\mathbf{w}^{(k)} \cdot \Delta\mathbf{\Phi}(x, y, z)} \\ &\leq \|\mathbf{w}^{(k)}\|^2 + R^2(S) + \underline{0}. \end{aligned}$$

   By Lemma 1, the update triple is a violation so that $\mathbf{w}^{(k)} \cdot \Delta\mathbf{\Phi}(x, y, z) \leq 0$, and that $(x, y, z) \in C_{\text{s}}(D)$ thus $\|\Delta\mathbf{\Phi}(x, y, z)\|^2 \leq R^2(S)$ by the definition of diameter. By induction, we have $\|\mathbf{w}^{(k+1)}\|^2 \leq kR^2(S)$.

Algorithm 2 Local Violation-Fixing Perceptron.

Input: training scenario $S = \langle D, \mathbf{\Phi}, C \rangle$
repeat
  for each example (x, y) **in** D **do**
    $(x, y', z) = \text{FINDVIOLATION}(x, y, \mathbf{w})$
    if $z \neq y$ **then** ▷ (x, y', z) is a violation
      $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y', z)$
until converged

Combining the two bounds, we have $k^2\delta^2(S) \leq \|\mathbf{w}^{(k+1)}\|^2 \leq kR^2(S)$, thus $k \leq R^2(S)/\delta^2(S)$. □

3 Violation is All We Need

We now draw the central observation of this work from part 2 of the above proof: note that exact search (**argmax**) is **not** required in the proof, instead, it just needs a *violation*, which is a much weaker condition.[2] Exact search is simply used to ensure violation. In other words, if we can guarantee violation in each update (which we call **"valid update"**), it does not matter whether or how exact the search is.

3.1 Violation-Fixing Perceptron

This observation leads us to two generalized variants of perceptron which we call "violation-fixing perceptrons". The local version, Algorithm 2 still works on one example at a time, and searches for one violation (if any) in that example to update with. The global version, Algorithm 3, can update on any violation in the dataset at any time. We state the following generalized theorem:

Theorem 2. *For a separable training scenario S the perceptrons in Algorithms 2 and 3 both converge with the same update bounds of $R^2(S)/\delta^2(S)$ as long as the* FINDVIOLATION *and* FINDVIOLATIONINDATA *functions always return violation triples if there are any.*

Proof. Same as the proof to Theorem 1, except for replacing Lemma 1 in part 2 by the fact that the update triples are guaranteed to be violations. (Note a violation triple is by definition in the confusion set, thus we can still use separation and diameter). □

These generic violation-fixing perceptron algorithms can been seen as "interfaces" (or APIs),

[2] Crammer and Singer (2003) further demonstrates that a convex combination of violations can also be used for update.

Algorithm 3 Global Violation-Fixing Perceptron.

Input: training scenario $S = \langle D, \mathbf{\Phi}, C \rangle$

repeat
  $(x, y, z) \leftarrow$ FINDVIOLATIONINDATA$(C, \mathbf{w})$
  if $x = \epsilon$ **then break** ▷ no more violation?
  $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y, z)$
until converged

data $D = \{(x, y)\}$:

x	fruit	flies	fly	.
y	N	N	V	.

search space: $\mathcal{Y}(x) = \{\mathtt{N}\} \times \{\mathtt{N}, \mathtt{V}\} \times \{\mathtt{N}, \mathtt{V}\} \times \{.\}$.
feature map: $\mathbf{\Phi}(x, y) = (\#_{\mathtt{N}\rightarrow\mathtt{N}}(y),\ \#_{\mathtt{V}\rightarrow.}(y))$.

iter	label z	$\Delta\mathbf{\Phi}(x, y, z)$	$\mathbf{w}\cdot\Delta\mathbf{\Phi}$		new $\mathbf{w}$
0					$(0, 0)$
1	N N N .	$(-1, +1)$	0	$\surd$	$(-1, 1)$
2	N V N .	$(+1, +1)$	0	$\surd$	$(0, 2)$
3	N N N .	$(-1, +1)$	2	$\times$	$(-1, 3)$
4	N V N .	$(+1, +1)$	2	$\times$	$(0, 4)$

... infinite loop ...

Figure 1: Example that standard perceptron does not converge with greedy search on a *separable* scenario (e.g. $\mathbf{u} = (1, 2)$ can separate D with exact search).

where later sections will supply different implementations of the FINDVIOLATION and FINDVIOLATIONINDATA subroutines, thus establishing alternative update methods for inexact search as special cases in this general framework.

3.2 Non-Convergence with Inexact Search

What if we can not guarantee valid updates? Well, the convergence proof in Theorems 1 and 2 would break in part 2. This is exactly why standard structured perceptron may not work well with inexact search: with search errors it is no longer possible to guarantee violation in each update. For example, an inexact search method explores a (proper) subset of the search space $\mathcal{Y}'_{\mathbf{w}}(x) \subsetneq \mathcal{Y}(x)$, and finds a label

$$z = \underset{s \in \mathcal{Y}'_{\mathbf{w}}(x)}{\mathbf{argmax}}\ \mathbf{w} \cdot \mathbf{\Phi}(x, s).$$

It is possible that the correct label y is outside of the explored subspace, and yet has a higher score: $\Delta\mathbf{\Phi}(x, y, z) > 0$ but $y \notin \mathcal{Y}'_{\mathbf{w}}(x)$. In this case, (x, y, z) is *not* a violation, which breaks the proof. We show below that this situation actually exists.

Algorithm 4 Greedy Search.

Let: NEXT$(x, z) \triangleq \{z \circ a \mid a \in \mathcal{Y}_{|z|+1}(x)\}$ ▷ set of possible one-step extensions (successors)
BEST$(x, z, \mathbf{w}) \triangleq \mathbf{argmax}_{z' \in \text{NEXT}(x,z)}\ \mathbf{w} \cdot \mathbf{\Phi}(x, z')$ ▷ best one-step extension based on history

function GREEDYSEARCH$(x, \mathbf{w})$
  $z \leftarrow \epsilon$ ▷ empty sequence
  for $i \in 1 \ldots |x|$ **do**
    $z \leftarrow$ BEST$(x, z, \mathbf{w})$
  return z

Theorem 3. *For a separable training scenario $S = \langle D, \mathbf{\Phi}, C_s(D) \rangle$, if the* $\mathbf{argmax}$ *in Algorithm 1 is not exact, the perceptron might not converge.*

Proof. See the example in Figure 1. □

This situation happens very often in NLP: often the search space $\mathcal{Y}(x)$ is too big either because it does not factor locally, or because it is still too big after factorization, which requires some approximate search. In either case, updating the model $\mathbf{w}$ on a non-violation (i.e., "invalid") triple (x, y, z) does not make sense: it is **not** the model's problem: $\mathbf{w}$ does score the correct label y higher than the incorrect z; rather, it is a problem of the search, or its interaction with the model that prunes away the correct (sub)sequence during the search.

What shall we do in this case? Collins and Roark (2004) suggest that instead of the standard full update, we should only update on the prefix $(x, y_{[1:i]}, z_{[1:i]})$ up to the point i where the correct sequence falls off the beam. This intuitively makes a lot of sense, since up to i we can still guarantee violation, but after i we may not. The next section formalizes this intuition.

4 Early Update is Violation-Fixing

We now proceed to show that early update is always valid and it is thus a special case of the violation-fixing perceptron framework. First, let us study the simplest special case, greedy search (beam=1).

4.1 Greedy Search and Early Update

Greedy search is the simplest form of inexact search. Shown in Algorithm 4, at each position, we commit to the single best action (e.g., tag for the current word) given the previous actions and continue to the

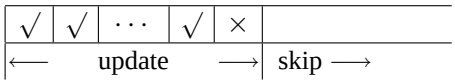

Figure 2: Early update at the first error in greedy search.

Algorithm 5 Early update for greedy search adapted from Collins and Roark (2004).

Input: training scenario $S = \langle D, \mathbf{\Phi}, C_{\mathrm{g}}(D) \rangle$
repeat
  for each example (x, y) **in** D **do**
    $z \leftarrow \epsilon$ ▷ empty sequence
    for $i \in 1 \dots |x|$ **do**
      $z \leftarrow \textsc{Best}(x, z, \mathbf{w})$
      if $z_i \neq y_i$ **then** ▷ first wrong action
        $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{\Phi}(x, y_{[1:i]}, z)$ ▷ early update
        break ▷ skip the rest of this example
until converged

next position. The notation $\mathcal{Y}_i(x)$ denotes the set of possible actions at position i for example x (for instance, the set of possible tags for a word).

The early update heuristic, originally proposed for beam search (Collins and Roark, 2004), now simplifies into "update at the first wrong action", since this is exactly the place where the correct sequence falls off the singleton beam (see Algorithm 5 for pseudocode and Fig. 2). Informally, it is easy to show (below) that this kind of update is always a valid violation, but we need to redefine confusion set.

Definition 5. The **greedy confusion set** $C_{\mathrm{g}}(D)$ for training data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{n}$ is the set of triples $(x, y_{[1:i]}, z_{[1:i]})$ where $y_{[1:i]}$ is a i-prefix of the correct label y, and $z_{[1:i]}$ is an incorrect i-prefix that agrees with the correct prefix on all decisions except the last one:

$$C_{\mathrm{g}}(D) \triangleq \{(x, y_{[1:i]}, z_{[1:i]}) \mid (x, y, z) \in C_{\mathrm{s}}(D), \\ 1 \leq i \leq |y|, z_{[1:i-1]} = y_{[1:i-1]}, z_i \neq y_i\}.$$

We can see intuitively that this new defintion is specially tailored to the early updates in greedy search. The concepts of separation/margin, violation, and diameter all change accordingly with this new confusion set. In particular, we say that a dataset D is **greedily separable** in representation $\mathbf{\Phi}$ if and only if $\langle D, \mathbf{\Phi}, C_{\mathrm{g}}(D) \rangle$ is linearly separable, and we say (x, y', z') is a **greedy violation** if $(x, y', z') \in C_{\mathrm{g}}(D)$ and $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y', z') \leq 0$.

Algorithm 6 Alternative presentation of Alg. 5 as a Local Violation-Fixing Perceptron (Alg. 2).

1: **function** FINDVIOLATION($x, y, \mathbf{w}$)
2: $z \leftarrow \epsilon$ ▷ empty sequence
3: **for** $i \in 1 \dots |x|$ **do**
4: $z \leftarrow \textsc{Best}(x, z, \mathbf{w})$
5: **if** $z_i \neq y_i$ **then** ▷ first wrong action
6: **return** $(x, y_{[1:i]}, z)$ ▷ return for early update
7: **return** (x, y, y) ▷ success ($z = y$), no violation

We now express early update for greedy search (Algorithm 5) in terms of violation-fixing perceptron. Algorithm 6 implements the FINDVIOLATION function for the generic Local Violation-Fixing Perceptron in Algorithm 2. Thus Algorithm 5 is equivalent to Algorithm 6 plugged into Algorithm 2.

Lemma 2. *Each triple $(x, y_{[1:i]}, z)$ returned at line 6 in Algorithm 6 is a greedy violation.*

Proof. Let $y' = y_{[1:i]}$. Clearly at line 6, $|y'| = i = |z|$ and $y'_i \neq z_i$. But $y'_j = z_j$ for all $j < i$ otherwise it would have returned before position i, so $(x, y', z) \in C_{\mathrm{g}}(D)$. Also $z = \textsc{Best}(x, z, \mathbf{w})$, so $\mathbf{w} \cdot \mathbf{\Phi}(x, z) \geq \mathbf{w} \cdot \mathbf{\Phi}(x, y')$, thus $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y', z) \leq 0$. □

Theorem 4 (convergence of greedy search with early update)**.** *For a separable training scenario $S = \langle D, \mathbf{\Phi}, C_{\mathrm{g}}(D) \rangle$, the early update perceptron by plugging Algorithm 6 into Algorithm 2 will make finite number of updates (before convergence):*

$$err(S) < R^2(S)/\delta^2(S).$$

Proof. By Lemma 2 and Theorem 2. □

4.2 Beam Search and Early Update

To formalize beam search, we need some notations:

Definition 6 (k-best)**.** We denote $\mathbf{argtop}_{z \in \mathcal{Z}}^{k} f(z)$ to return (a sorted list of) the top k **unique** z in terms of $f(z)$, i.e., it returns a list $\mathcal{B} = [z^{(1)}, z^{(2)}, \dots, z^{(k)}]$ where $z^{(i)} \in \mathcal{Z}$ and $f(z^{(1)}) \geq f(z^{(2)}) \geq \dots \geq f(z^{(k)}) \geq f(z')$ for all $z' \in \mathcal{Z} - \mathcal{B}$.

By unique we mean that no two elements are equivalent with respect to some equivalence relation, i.e., $z^{(i)} \equiv z^{(j)} \Rightarrow i = j$. This equivalence relation is useful for dynamic programming (when used with beam search). For example, in trigram tagging, two tag sequences are equivalent if they are of the

Algorithm 7 Beam-Search.

$\textsc{Best}_k(x, \mathcal{B}, \mathbf{w}) \triangleq \mathbf{argtop}^k_{z' \in \cup_{z \in \mathcal{B}} \textsc{Next}(z)} \mathbf{w} \cdot \mathbf{\Phi}(x, z')$
▷ top k (unique) extensions from the beam
1: **function** $\textsc{BeamSearch}(x, \mathbf{w}, k)$ ▷ k is beam width
2: $\mathcal{B}_0 \leftarrow [\epsilon]$ ▷ initial beam
3: **for** $i \in 1 \dots |x|$ **do**
4: $\mathcal{B}_i \leftarrow \textsc{Best}_k(x, \mathcal{B}_{i-1}, \mathbf{w})$
5: **return** $\mathcal{B}_{|x|}[0]$ ▷ best sequence in the final beam

Algorithm 8 Early update for beam search (Collins and Roark 04) as Local Violation-Fixing Perceptron.

1: **function** $\textsc{FindViolation}(x, y, \mathbf{w})$
2: $\mathcal{B}_0 \leftarrow [\epsilon]$
3: **for** $i \in 1 \dots |x|$ **do**
4: $\mathcal{B}_i \leftarrow \textsc{Best}_k(x, \mathcal{B}_{i-1}, \mathbf{w})$
5: **if** $y_{[1:i]} \notin \mathcal{B}_i$ **then** ▷ correct seq. falls off beam
6: **return** $(x, y_{[1:i]}, \mathcal{B}_i[0])$ ▷ update on prefix
7: **return** $(x, y, \mathcal{B}_{|x|}[0])$ ▷ full update if wrong final

same length and if they agree on the last two tags, i.e. $z \equiv z'$ iff. $|z| = |z'|$ and $z_{|z|-1:|z|} = z'_{|z|-1:|z|}$. In incremental parsing this equivalence relation could be relevant bits of information on the last few trees on the stack (depending on feature templates), as suggested in (Huang and Sagae, 2010). [3]

Algorithm 7 shows the pseudocode for beam search. It is trivial to verify that greedy search is a special case of beam search with $k = 1$. However, the definition of confusion set changes considerably:

Definition 7. The **beam confusion set** $C_{\mathrm{b}}(D)$ for training data $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^n$ is the set of triples $(x, y_{[1:i]}, z_{[1:i]})$ where $y_{[1:i]}$ is a i-prefix of the correct label y, and $z_{[1:i]}$ is an incorrect i-prefix that differs from the correct prefix (in at least one place):

$$C_{\mathrm{b}}(D) \triangleq \{(x, y_{[1:i]}, z_{[1:i]}) \mid (x, y, z) \in C_{\mathrm{s}}(D), 1 \le i \le |y|, z_{[1:i]} \neq y_{[1:i]}\}.$$

Similarly, we say that a dataset D is **beam separable** in representation $\mathbf{\Phi}$ if and only if $\langle D, \mathbf{\Phi}, C_{\mathrm{b}}(D)\rangle$ is linearly separable, and we say (x, y', z') is a **beam violation** if $(x, y', z') \in C_{\mathrm{b}}(D)$ and $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y', z') \le 0$.

[3] Note that when checking whether the correct sequence falls off the beam (line 5), we could either store the whole (sub)sequence for each candidate in the beam (which is what we do for non-DP anyway), or check if the equivalence class of the correct sequence is in the beam, i.e. $[\![y_{[1:i]}]\!]_{\equiv} \in \mathcal{B}_i$, and if its backpointer points to $[\![y_{[1:i-1]}]\!]_{\equiv}$. For example, in trigram tagging, we just check if $\langle y_{i-1}, y_i\rangle \in \mathcal{B}_i$ and if its backpointer points to $\langle y_{i-2}, y_{i-1}\rangle$.

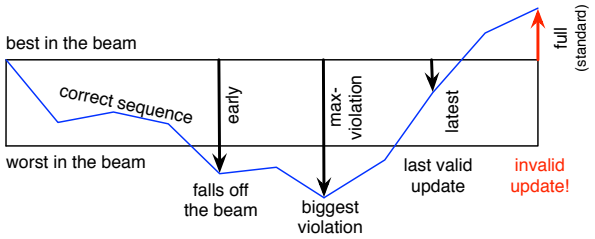

Figure 3: Illustration of various update methods: early, max-violation, latest, and standard (full) update, in the case when standard update is invalid (shown in red). The rectangle denotes the beam and the blue line segments denote the trajectory of the correct sequence.

It is easy to verify that beam confusion set is superset of both greedy and standard confusion sets: for all dataset D, $C_{\mathrm{g}}(D) \subsetneq C_{\mathrm{b}}(D)$, and $C_{\mathrm{s}}(D) \subsetneq C_{\mathrm{b}}(D)$. This means that beam separability is the strongest condition among the three separabilities:

Theorem 5. *If a dataset D is beam separable, then it is also greedily and (standard) linear separable.*

We now present early update for beam search as a Local Violation Fixing Perceptron in Algorithm 8. See Figure 3 for an illustration.

Lemma 3. *Each update (lines 6 or 7 in Algorithm 8) involves a beam violation.*

Proof. Case 1: early update (line 6): Let $z' = \mathcal{B}_i[0]$ and $y' = y_{[1:i]}$. Case 2: full update (line 8): Let $z' = \mathcal{B}_{|x|}[0]$ and $y' = y$. In both cases we have $z' \neq y'$ and $|z'| = |y'|$, thus $(x, y', z') \in C_{\mathrm{b}}(D)$. Also we have $\mathbf{w} \cdot \mathbf{\Phi}(x, z') \ge \mathbf{w} \cdot \mathbf{\Phi}(x, y')$ by defintion of **argtop**, so $\mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y', z') \le 0$. □

Theorem 6 (convergence of beam search with early update)**.** *For a separable training scenario $S = \langle D, \mathbf{\Phi}, C_{\mathrm{b}}(D)\rangle$, the early update perceptron by plugging Algorithm 8 into Algorithm 2 will make finite number of updates (before convergence):*

$$err(S) < R^2(S)/\delta^2(S).$$

Proof. By Lemma 3 and Theorem 2. □

5 New Update Methods for Inexact Search

We now propose three novel update methods for inexact search within the framework of violation-fixing perceptron. These methods are inspired by early update but addresses its very limitation of slow learning. See Figure 3 for an illustration.

1. **"hybrid" update.** When the standard update is valid (i.e., a violation), we perform it, otherwise we perform the early update.

2. **"max-violation" update**. While there are more than one possible violations on one example x, we choose the triple that is most violated:

$$(x, y^*, z^*) = \operatorname*{argmin}_{(x,y',z') \in C, z' \in \cup_i \{\mathcal{B}_i[0]\}} \mathbf{w} \cdot \Delta\mathbf{\Phi}(x, y', z').$$

3. **"latest" update**. Contrary to early update, we can also choose the latest point where the update is still a violation:

$$(x, y^*, z^*) = \operatorname*{argmax}_{(x,y',z') \in C, z' \in \cup_i \{\mathcal{B}_i[0]\}, \mathbf{w} \cdot \Delta\mathbf{\Phi}(x,y',z') > 0} |z'|.$$

All these three methods go beyond early update but can be represented in the Local Violation Fixing Perceptron (Algorithm 2), and are thus all guaranteed to converge. As we will see in the experiments, these new methods are motivated to address the major limitation of early update, that is, it learns too slowly since it only updates on prefixes and neglect the rest of the examples. Hybrid update is trying to do as much standard ("full") updates as possible, and latest update further addresses the case when standard update is invalid: instead of backing-off to early update, it uses the longest possible update.

6 Experiments

We conduct experiments on two typical structured learning tasks: part-of-speech tagging with a trigram model where exact search is possible, and incremental dependency parsing with arbitrary non-local features where exact search is intractable. We run both experiments on state-of-the-art implementations.

6.1 Part-of-Speech Tagging

Following the standard split for part-of-speech tagging introduced by Collins (2002), we use sections 00–18 of the Penn Treebank (Marcus et al., 1993) for training, sections 19–21 as a held-out development set, and sections 22–24 for testing. Our baseline system is a faithful implementation of the perceptron tagger in Collins (2002), i.e., a trigram model with spelling features from Ratnaparkhi (1996), except that we replace one-count words as <unk>. With standard perceptron and exact search, our baseline system performs slightly better than Collins (2002) with a beam of 20 (M. Collins, p.c.).

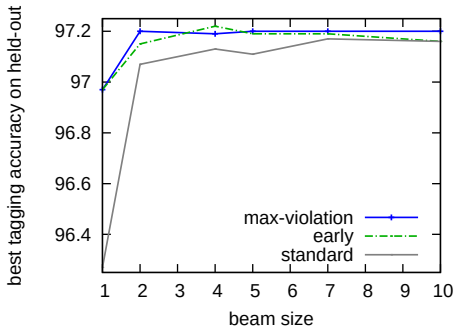

Figure 4: POS tagging using beam search with various update methods (hybrid/latest similar to early; omitted).

	$b = 1$		$b = 2$		$b = 7$	
method	it	dev	it	dev	it	dev
standard	12	96.27	6	97.07	**4**	97.17
early	13	**96.97**	6	97.15	7	97.19
max-viol.	**7**	**96.97**	**3**	**97.20**	**4**	**97.20**

Table 1: Convergence rate of part-of-speech tagging. In general, max-violation converges faster and better than early and standard updates, esp. in smallest beams.

We then implemented beam search on top of dynamic programming and experimented with standard, early, hybrid, and max-violation update methods with various beam settings ($b = 1, 2, 4, 7, 10$). Figure 4(a) summarizes these experiments. We observe that, first of all, the standard update performs poorly with the smallest beams, esp. at $b = 1$ (greedy search), when search error is the most severe causing lots of invalid updates (see Figure 5). Secondly,max-violation is almost consistently the best-performing method (except for $b = 4$). Table 1 shows convergence rates, where max-violation update also converges faster than early and standard methods. In particular, at $b = 1$, it achieves a 19% error reduction over standard update, while converg-

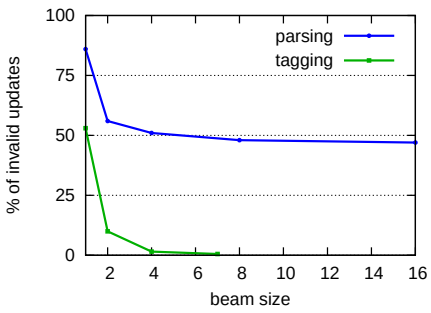

Figure 5: Percentages of invalid updates for standard update. In tagging it quickly drops to 0% while in parsing it converges to $\sim 50\%$. This means search-wise, parsing is much harder than tagging, which explains why standard update does OK in tagging but terribly in parsing. The harder the search is, the more needed valid updates are.

method	b	it	time	dev	test
standard*	∞	6	162 m	97.17	97.28
early	4	6	37 m	**97.22**	**97.35**
hybrid	5	5	30 m	97.18	97.19
latest	7	5	45 m	97.17	97.13
max-viol.	**2**	3	**26 m**	97.20	97.33
standard	20	Collins (2002)			97.11
guided	3	Shen et al. (2007)			97.33

Table 2: Final test results on POS tagging. *:baseline.

method	b	it	time	dev	test
early*	8	38	15.4 h	92.24	92.09
standard		1	0.4 h	*78.99*	*79.14*
hybrid		11	5.6 h	92.26	91.95
latest		9	4.5 h	92.06	91.96
max-viol.		12	5.5 h	**92.32**	**92.18**
early	8	Huang & Sagae 2010			92.1

Table 3: Final results on incremental parsing. *: baseline.

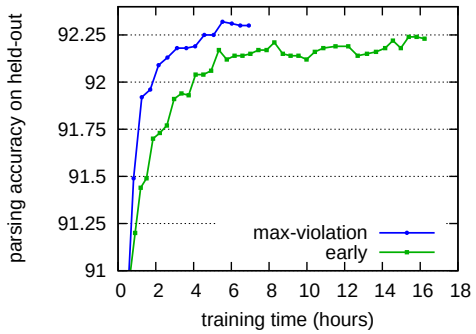

Figure 6: Training progress curves for incremental parsing ($b = 8$). Max-violation learns faster and better: it takes 4.6 hours (10 iterations) to reach 92.25 on held-out, compared with early update's 15.4 hours (38 iterations), even though the latter is faster in each iteration due to early stopping (esp. at the first few iterations).

ing twice as fast as early update.[4] This agrees with our intuition that by choosing the "most-violated" triple for update, the perceptron should learn faster.

Table 2 presents the final tagging results on the test set. For each of the five update methods, we choose the beam size at which it achieves the highest accuracy on the held-out. For standard update, its best held-out accuracy 97.17 is indeed achieved by exact search (i.e., $b = +\infty$) since it does not work well with beam search, but it costs 2.7 hours (162 minutes) to train. By contrast, the four valid update methods handle beam search better. The max-violation method achieves its highest held-out/test accuracies of 97.20 / 97.33 with a beam size of only 2, and only 26 minutes to train. Early update achieves the highest held-out/test accuracies of 97.22 / **97.35** across all update methods at the beam size of 4. This test accuracy is even better than Shen et al. (2007), the best tagging accuracy reported on the Penn Treebank to date.[5,6] To conclude, with valid update methods, we can learn a better tagging model with 5 times faster training than exact search.

6.2 Incremental Parsing

While part-of-speech tagging is mainly a proof of concept, incremental parsing is much harder since non-local features rules out exact inference.

We use the standard split for parsing: secs 02–21 for training, 22 as held-out, and 23 for testing. Our baseline system is a faithful reimplementation of the beam-search dynamic programming parser of Huang and Sagae (2010). Like most incremental parsers, it used early update as search error is severe.

[4] for tagging (but not parsing) the difference in per-iteration speed between early update and max-violation update is small.

[5] according to ACL Wiki: http://aclweb.org/aclwiki/.

[6] Note that Shen et al. (2007) employ contextual features up to 5-gram which go beyond our local trigram window. We suspect that adding *genuinely* non-local features would demonstrate even better the advantages of valid update methods with beam search, since exact inference will no longer be tractable.

We first confirm that, as reported by Huang and Sagae, early update learns very slowly, reaching 92.24 on held-out with 38 iterations (15.4 hours).

We then experimented with the other update methods: standard, hybrid, latest, and max-violation, with beam size $b = 1, 2, 4, 8$. We found that, first of all, the standard update performs horribly on this task: at $b = 1$ it only achieves 60.04% on held-out, while at $b = 8$ it improves to 78.99% but is still vastly below all other methods. This is because search error is much more severe in incremental parsing (than in part-of-speech tagging), thus standard update produces an enormous amount of invalid updates even at $b = 8$ (see Figure 5). This suggests that the advantage of valid update methods is more pronounced with tougher search problems. Secondly, max-violation learns much faster (and better) than early update: it takes only 10 iterations (4.6 hours) to reach 92.25, compared with early update's 15.4 hours (see Fig. 6). At its peak, max-violation achieves 92.18 on test which is better than (Huang and Sagae, 2010). To conclude, we can train a parser with only 1/3 of training time with max-violation update, and the harder the search is, the more needed the valid update methods are.

7 Related Work and Discussions

Besides the early update method of Collins and Roark (2004) which inspired us, this work is also related to the **LaSO** method of Daumé and Marcu (2005). LaSO is similar to early update, except that after each update, instead of skipping the rest of the example, LaSO *continues* on the same example with the correct hypothesis. For example, in the greedy case LaSO is just replacing the **break** statement in Algorithm 5 by

$$8\text{'}: \quad z_i = y_i$$

and in beam search it is replacing it with

$$8\text{'}: \quad \mathcal{B}_i = [y_{[1:i]}].$$

This is beyond our Local Violation-Fixing Perceptron since it makes more than one updates on one example, but can be easily represented as a Global Violation-Fixing Perceptron (Algorithm 3), since we can prove any further updates on this example is a violation (under the new weights). We thus establish LaSO as a special case within our framework.[7]

More interestingly, it is easy to verify that the greedy case of LaSO update is equivalent to training a local **unstructured** perceptron which **independently** classifies at each position based on history, which is related to SEARN (Daumé et al., 2009).

Kulesza and Pereira (2007) study perceptron learning with approximate inference that *overgenerates* instead of *undergenerates* as in our work, but the underlying idea is similar: by learning in a harder setting (LP-relaxed version in their case and prefix-augmented version in our case) we can learn the simpler original setting. Our "beam separability" can be viewed as an instance of their "algorithmic separability". Finley and Joachims (2008) study similar approximate inference for structural SVMs.

Our max-violation update is also related to other training methods for large-margin structured prediction, in particular the cutting-plane (Joachims et al., 2009) and subgradient (Ratliff et al., 2007) methods, but detailed exploration is left to future work.

8 Conclusions

We have presented a unifying framework of "violation-fixing" perceptron which guarantees convergence with inexact search. This theory satisfingly explained why standard perceptron might not work well with inexact search, and why the early update works. We also proposed some new variants within this framework, among which the max-violation method performs the best on state-of-the-art tagging and parsing systems, leading to better models with greatly reduced training times. Lastly, the advantage of valid update methods is more pronounced when search error is severe.

Acknowledgements

We are grateful to the four anonymous reviewers, especially the one who wrote the comprehensive review. We also thank David Chiang, Kevin Knight, Ben Taskar, Alex Kulesza, Joseph Keshet, David McAllester, Mike Collins, Sasha Rush, and Fei Sha for discussions. This work is supported in part by a Google Faculty Research Award to the first author.

[7] It turns out the original theorem in the LaSO paper (Daumé and Marcu, 2005) contains a bug; see (Xu et al., 2009) for corrections. Thanks to a reviewer for pointing it out.

References

Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research (JMLR)*, 3:951–991.

Daumé, Hal, John Langford, and Daniel Marcu. 2009. Search-based structured prediction.

Daumé, Hal and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of ICML*.

Finley, Thomas and Thorsten Joachims. 2008. Training structural SVMs when exact inference is intractable. In *Proceedings of ICML*.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the ACL: HLT*, Columbus, OH, June.

Huang, Liang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*.

Joachims, T., T. Finley, and Chun-Nam Yu. 2009. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59.

Kulesza, Alex and Fernando Pereira. 2007. Structured learning with approximate inference. In *NIPS*.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Liang, Percy, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of COLING-ACL*, Sydney, Australia, July.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*.

Ratliff, Nathan, J. Andrew Bagnell, and Martin Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Proceedings of AIStats*.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*.

Shen, Libin, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL*.

Taskar, Ben, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *Proceedings of NIPS*. MIT Press.

Tsochantaridis, I., T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Xu, Yuehua, Alan Fern, and Sungwook Yoon. 2009. Learning linear ranking functions for beam search with application to planning. *Journal of Machine Learning Research (JMLR)*, 10:1349–1388.

Zhang, Yue and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*.