Minibatch and Parallelization for Online Large Margin Structured Learning

Kai Zhao and Liang Huang

NAACL 2013

読み手: 岡崎　直観

背景・研究の動機

- 多くのNLPタスクでオンライン構造学習が使われている
  - バッチ学習と比べて速い，スケーラブル，省メモリ，実装が単純（1-bestもしくはk-best解が求まれば十分な場合がある）
- オンライン学習は並列化したときの扱いが難しい
  - 学習事例ごとに重みを更新するため，事例間に依存関係がある
  - Iterative Parameter Mixing (IPM) (McDonald+ 2010) が提案されているが，並列化の効果が薄い（例: 10CPU以上を使っても3倍程度しか速くならない）
  - バッチ学習では勾配の計算の並列化が容易（→ CRF++）
- 本研究の目的は，”mini-batch”というアイディアでオンライン学習を並列化し，学習を効率化・高速化する

本研究の貢献

- Mini-batchによりパーセプトロンとMIRAを拡張
  - MIRAでは複数の事例による制約に基づきパラメータ更新を行うため，よいマージンが得られる可能性がある
  - これらのアルゴリズムの収束を理論的に証明する

- 学習が速くなり精度も向上することを実証
  - 並列化しなくても学習が速くなった
  - タスクの精度も若干良くなった
  - 12CPUを使って並列化したMini-batch MIRAは約9倍速くなった

パーセプトロンの場合

MIRAまで説明すると時間がかかるので省略

Structured Perceptron (Collins 2002)

1. $\boldsymbol{w} = 0$
2. **repeat**:
3. **for** (x_i, y_i) **in** D:
4. $z \leftarrow \underset{z \in \mathrm{Gen}(x_i)}{\mathrm{argmax}}\ \boldsymbol{w} \cdot \boldsymbol{\Phi}(x_i, y_i)$
5. **if** $z \neq y_i$:
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z)$
7. **until convergence**
8. **return** $\boldsymbol{w}$

訓練データ

重みベクトル

素性関数

x_iに対して割り当て可能なラベルを列挙する関数

2値分類の場合は簡単

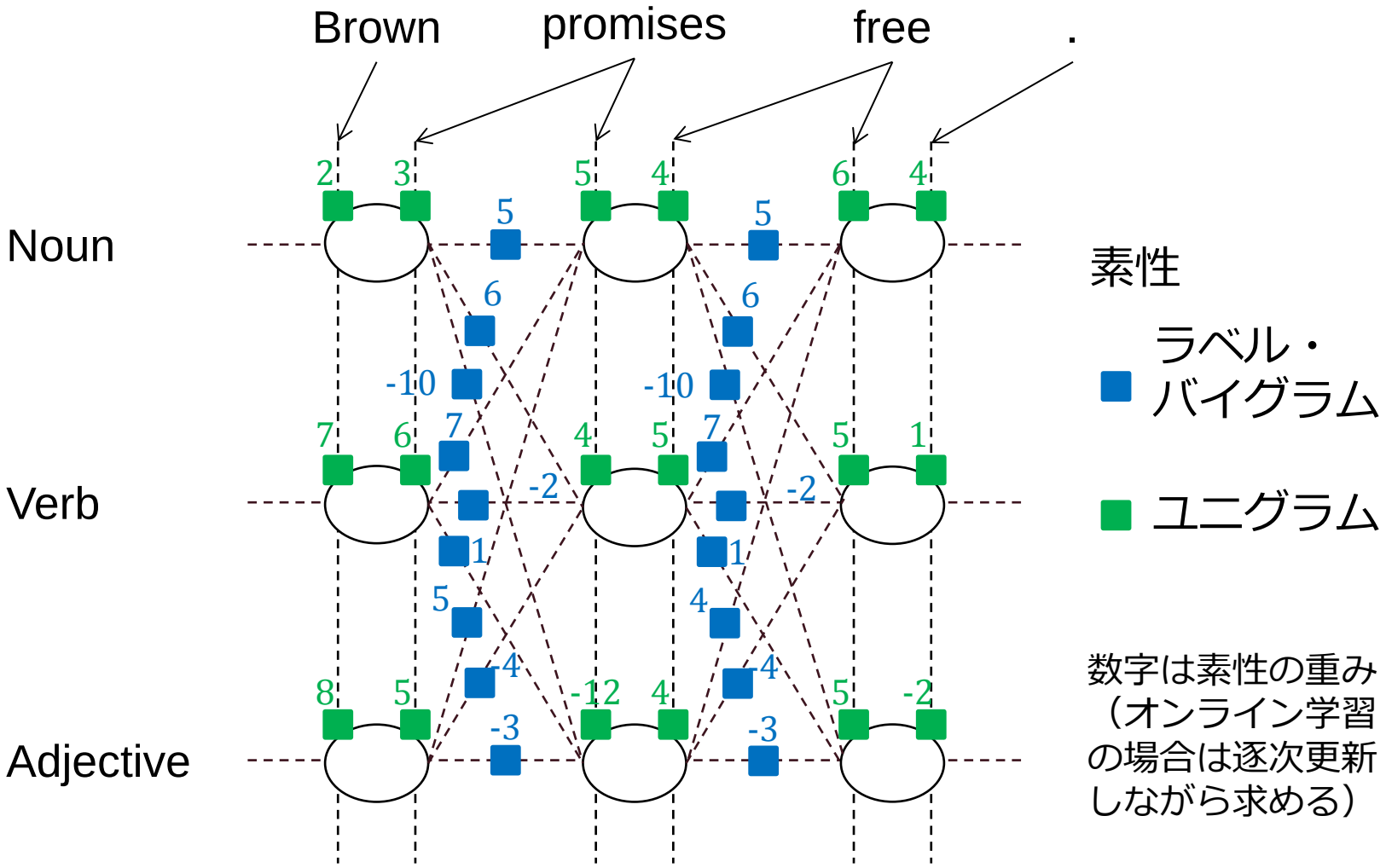
1. $\boldsymbol{w} = 0$
2. **repeat**:
3. **for** $(\boldsymbol{x_i}, y_i)$ **in** D:
4. $z \leftarrow \operatorname{sign} \boldsymbol{w} \cdot \boldsymbol{x_i}$
5. **if** $z \neq y_i$:
6. $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{x_i}$
7. **until convergence**
8. **return** $\boldsymbol{w}$

事例を素性ベクトルで表す（素性関数は不要）

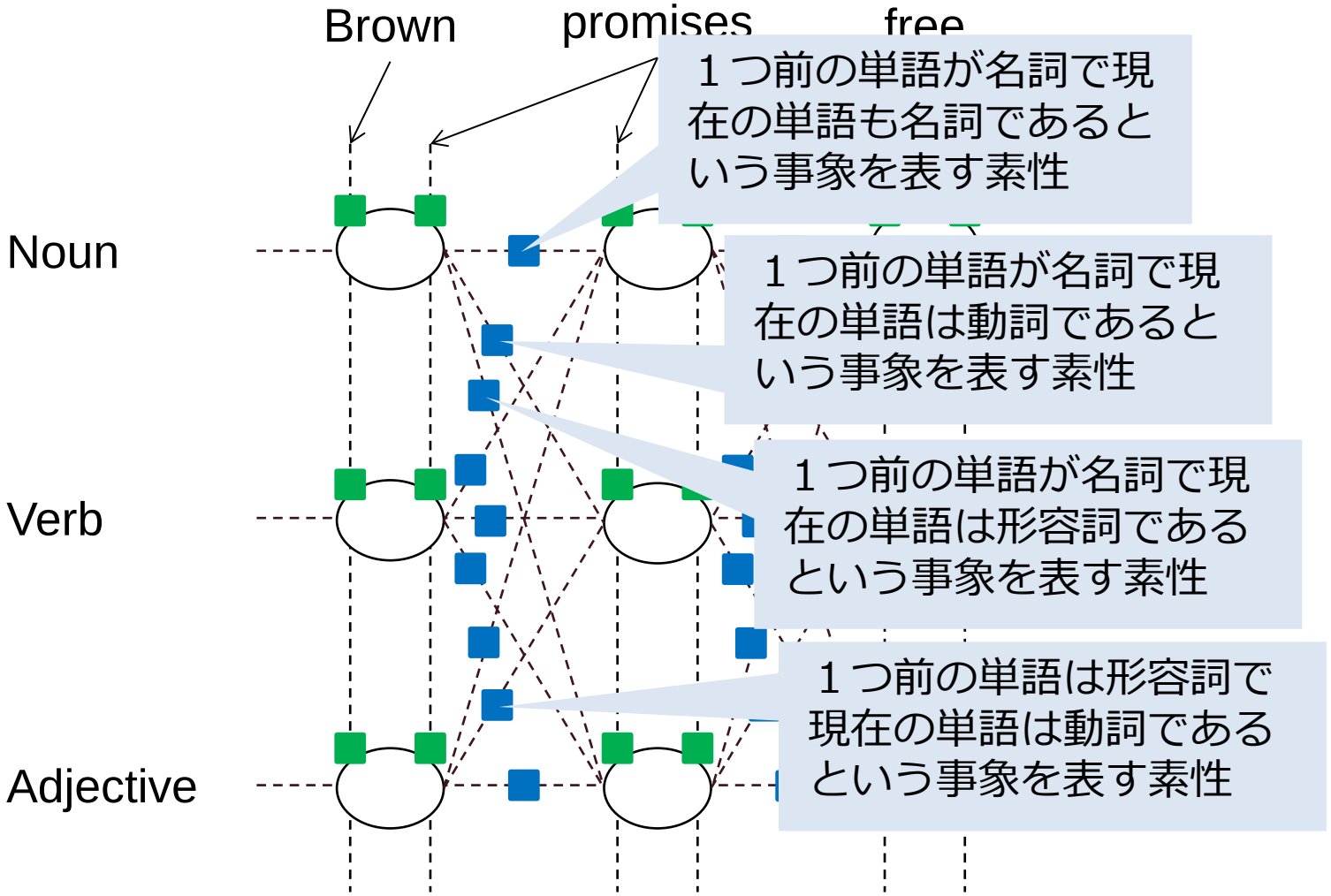
$y_i \in \{+1, -1\}$

$$z = \begin{cases} +1 & (\text{if } \boldsymbol{w} \cdot \boldsymbol{x}_i > 0) \\ -1 & (\text{otherwise}) \end{cases}$$

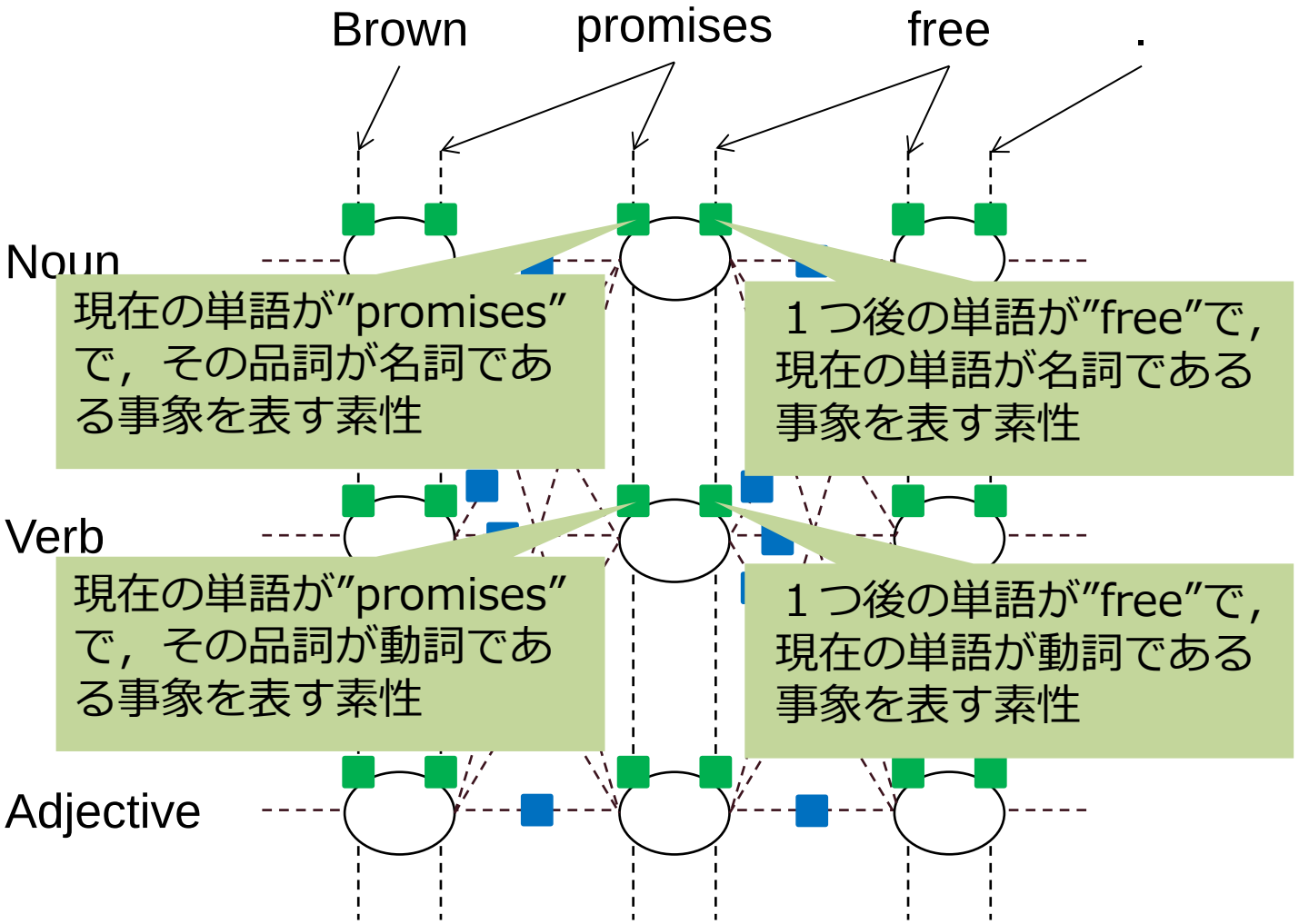
構造化パーセプトロン（品詞タグ付け）



ラベルバイグラム素性の例

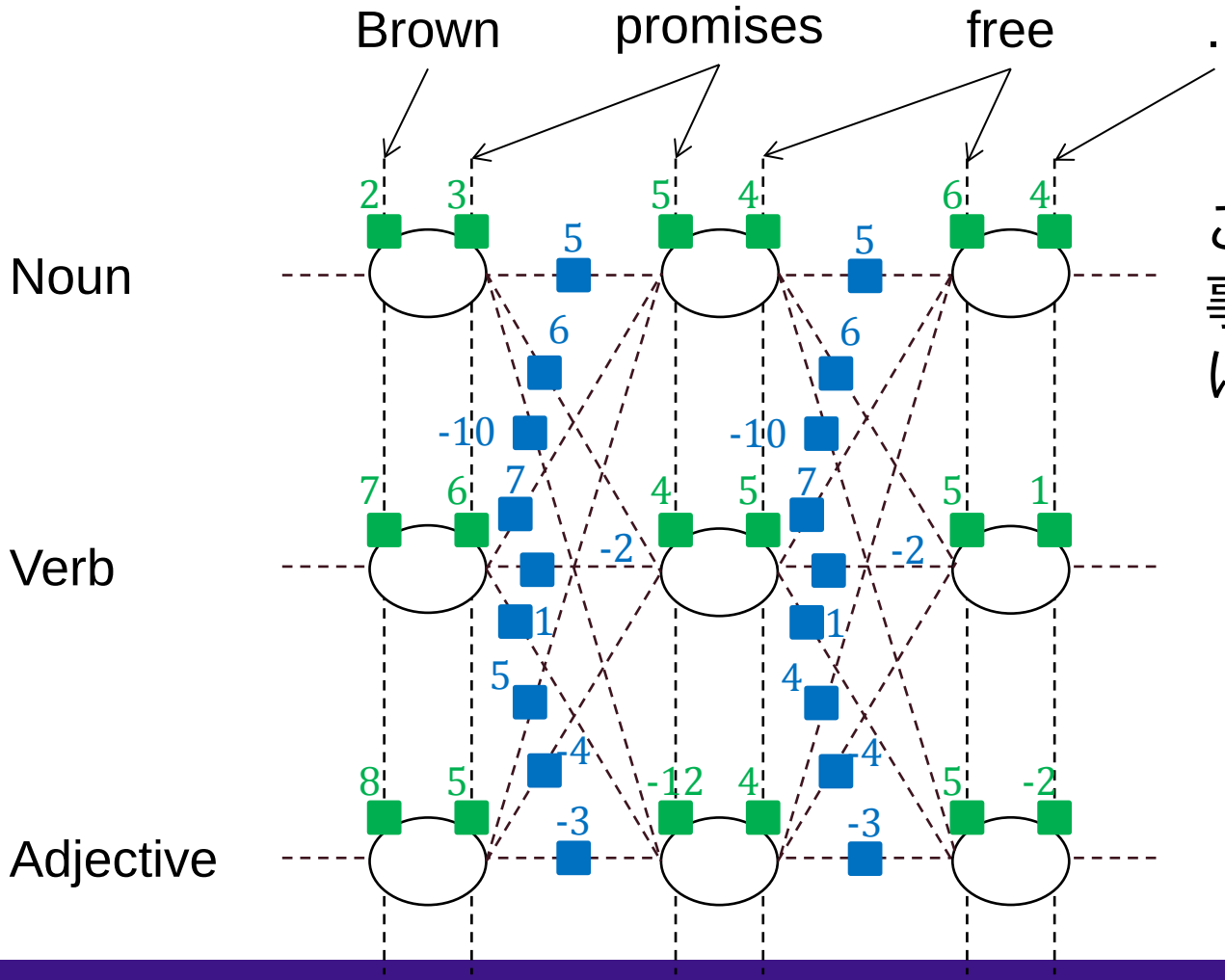


ユニグラム素性の例


Brown
promises
free
.
Noun
現在の単語が”promises”で，その品詞が名詞である事象を表す素性
１つ後の単語が”free”で，現在の単語が名詞である事象を表す素性
Verb
現在の単語が”promises”で，その品詞が動詞である事象を表す素性
１つ後の単語が”free”で，現在の単語が動詞である事象を表す素性
Adjective

構造化パーセプトロンの学習例(1/3)

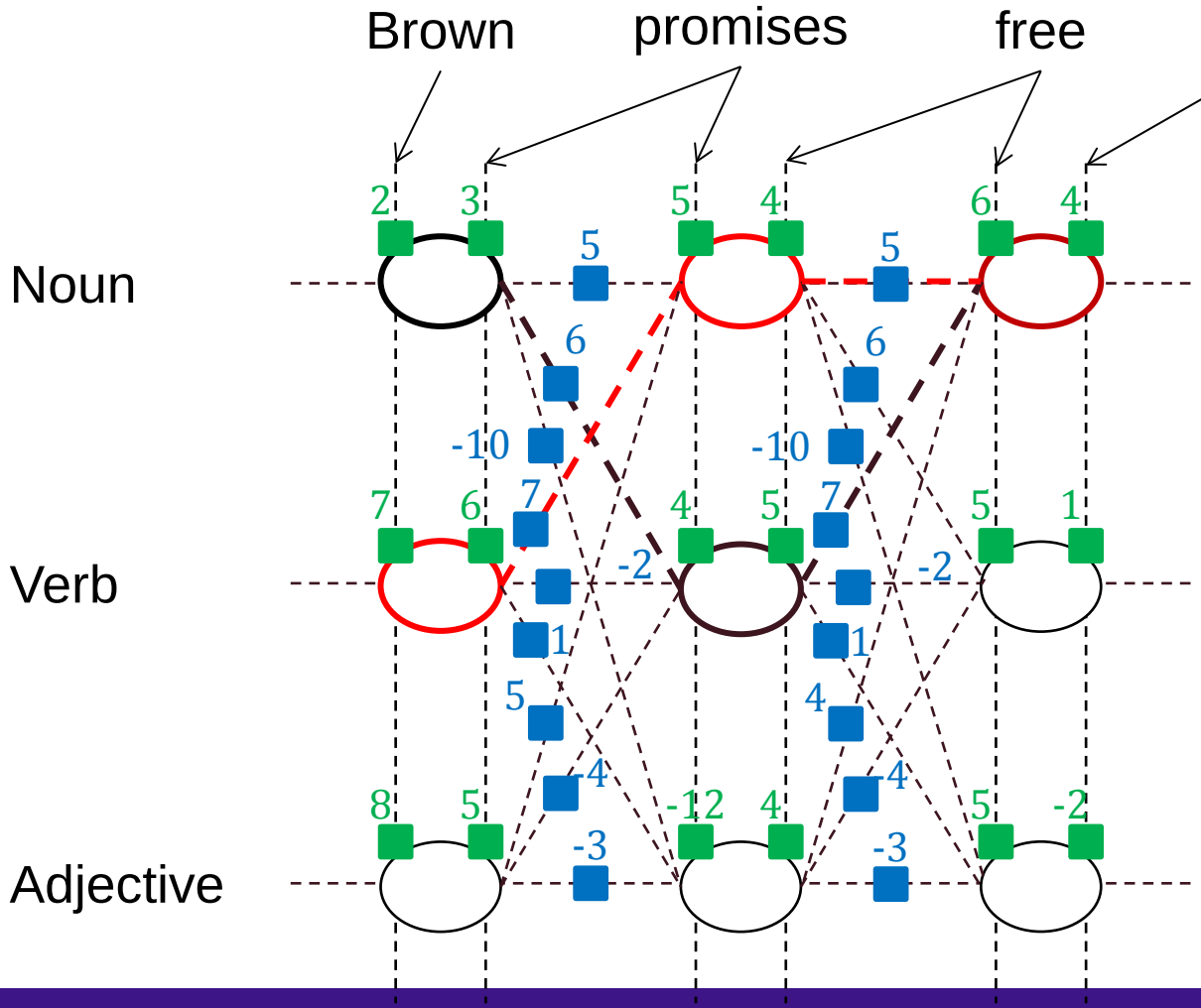
- $(\boldsymbol{x}_i, \boldsymbol{y}_i) = (\text{Brown promises change, Noun Verb Noun})$



この状態で訓練事例$(\boldsymbol{x}_i, \boldsymbol{y}_i)$を受け取ったとする

構造化パーセプトロンの学習例(2/3)

- $(\boldsymbol{x}_i, \boldsymbol{y}_i) = (\text{Brown promises change, Noun Verb Noun})$

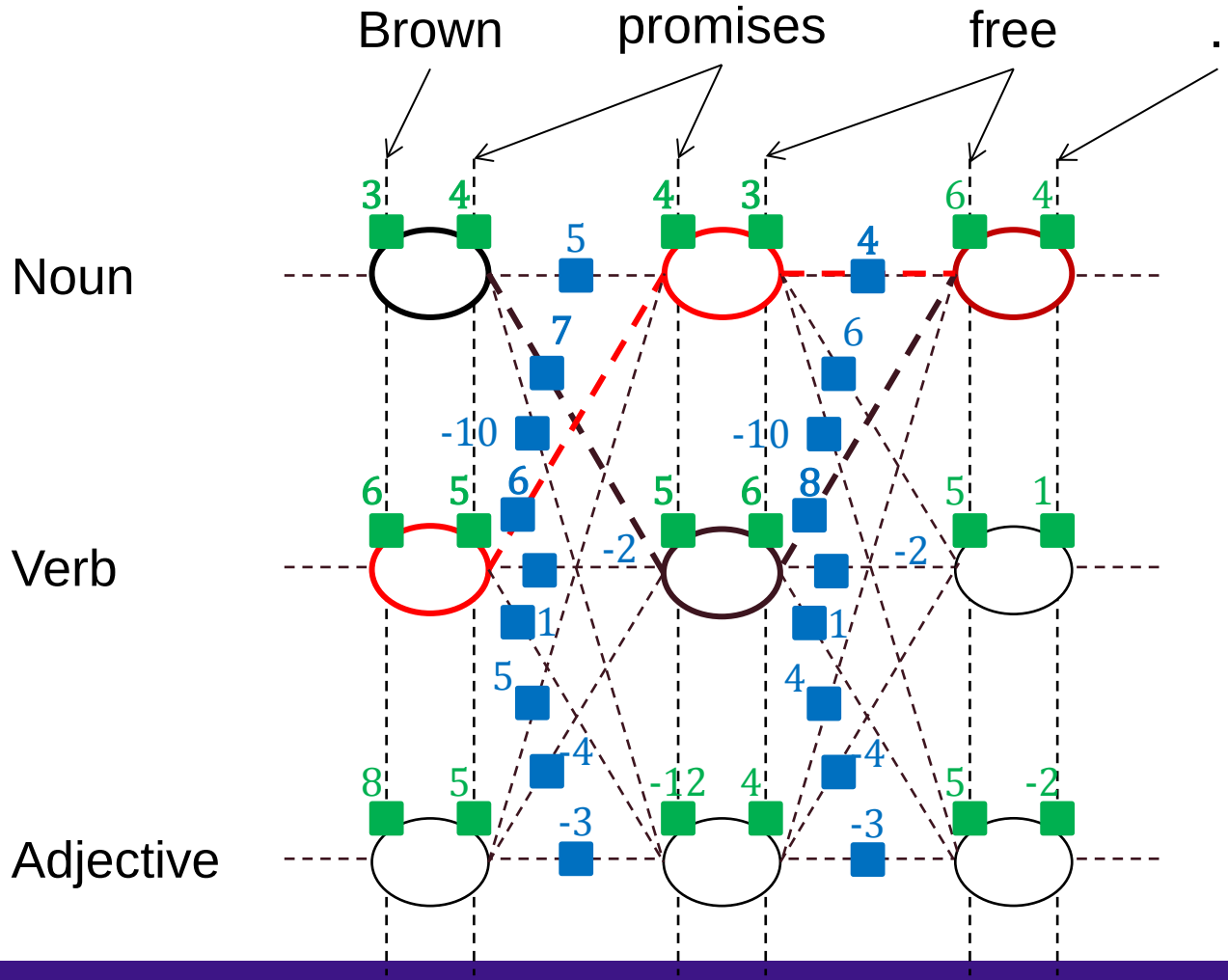


ビタビアルゴリズムによる品詞の予測(argmax):
Verb Noun Noun

訓練事例の正解:
Noun Verb Noun

構造化パーセプトロンの学習例(3/3)

- $(\boldsymbol{x}_i, \boldsymbol{y}_i) = (\text{Brown promises change, Noun Verb Noun})$



更新:

正解のラベル系列上の素性の重みを+1

予測された（不正解の）ラベル系列上の素性の重みを-1

定理: 半径がRの訓練データ$D = \{(x_i, y_i)\}_{i=1}^{N}$がマージン$\delta$で分離可能であるとき，パーセプトロン・アルゴリズムは$t \leq R^2/\delta^2$回のパラ―メータ更新で終了する

- 定義
  - 事例x_iに割り当て可能なラベルの集合を$\mathrm{Gen}(x_i)$と書く
  - 訓練データDの半径がRであるとは

$$\forall i, \forall z \in \mathrm{Gen}(x_i), \|\mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z)\| \leq R$$

  - 訓練データDがマージンδで分離可能とは

$$\forall i, \forall z \in \mathrm{Gen}(x_i); \|\mathbf{u} \cdot \mathbf{\Phi}(x_i, y_i) - \mathbf{u} \cdot \mathbf{\Phi}(x_i, z)\| \geq \delta$$

となるベクトル$\mathbf{u}$ (ただし$\|\mathbf{u}\| = \sqrt{\sum u_i^2} = 1$) が存在すること

定理: 半径がRの訓練データ$D = \{(x_i, y_i)\}_{i=1}^{N}$がマージン$\delta$で分離可能であるとき，パーセプトロン・アルゴリズムは$t \leq R^2/\delta^2$回のパラ―メータ更新で終了する

証明: $\boldsymbol{w}^0 = 0$から始めてt回目のパラメータ更新が訓練事例(x_i, y_i)に対して行われ，重みベクトル$\boldsymbol{w}^{t-1}$が$\boldsymbol{w}^t$に更新されるとき，

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} + \boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z), z = \underset{y \in \mathrm{Gen}(x_i)}{\mathrm{argmax}}\ \boldsymbol{w}^{t-1} \cdot \boldsymbol{\Phi}(x_i, y_i)$$

更新式の両辺に$\boldsymbol{u}$をかけると，

$$\boldsymbol{u} \cdot \boldsymbol{w}^t = \boldsymbol{u} \cdot \boldsymbol{w}^{t-1} + \boldsymbol{u} \cdot \boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{u} \cdot \boldsymbol{\Phi}(x_i, z) \geq \boldsymbol{u} \cdot \boldsymbol{w}^{t-1} + \delta \geq t\delta$$

帰納法により（$\boldsymbol{w}^0 = 0$）

内積と$\boldsymbol{u}$の定義より$\boldsymbol{u} \cdot \boldsymbol{w}^t \leq \|\boldsymbol{u}\|\|\boldsymbol{w}^t\| = \|\boldsymbol{w}^t\|$．したがって，

$$\|\boldsymbol{w}^t\| \geq \boldsymbol{u} \cdot \boldsymbol{w}^t \geq t\delta$$

更新式の両辺を二乗すると，

$$\|\boldsymbol{w}^t\|^2 = \|\boldsymbol{w}^{t-1}\|^2 + \|\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z)\|^2 + 2\boldsymbol{w}^{t-1} \cdot \underbrace{\left(\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z)\right)}_{y_i\text{の予測に失敗しているので負}}$$
$$\leq \|\boldsymbol{w}^{t-1}\|^2 + R^2 \leq tR^2$$

$\|\boldsymbol{w}^t\|^2$の上限と下限より，$t^2\delta^2 \leq \|\boldsymbol{w}^t\|^2 \leq tR^2 \Rightarrow t \leq R^2/\delta^2$ ■

Mini-batch Perceptron

1. $\boldsymbol{w} = 0$
2. Dを$[n/m]$個のmini-batch$D_1, \ldots, D_{[n/m]}$に分割
3. **repeat**:
4. **for** j **in range**$([n/m])$:
5. $C = \emptyset$
6. **for** (x_i, y_i) **in** D_j:
7. $z_i \leftarrow \underset{z_i \in \mathrm{Gen}(x_i)}{\mathrm{argmax}}\ \boldsymbol{w} \cdot \boldsymbol{\Phi}(x_i, y_i)$
8. **if** $z_i \neq y_i$:
9. $C \leftarrow C \cup \{(x_i, y_i, z_i)\}$
10. **for** (x_i, y_i, z_i) **in** C:
11. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \frac{1}{|C|}\left(\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z)\right)$
12. **until convergence**
13. **return** $\boldsymbol{w}$

n: 訓練事例数
m: mini-batch数

Mini-batch D_j (m個の訓練事例) のうち，現在の重み$\boldsymbol{w}$では分類を間違えてしまうものをCに格納する（重みの更新は行わない）

Cを用いて重みの更新をまとめて実行

どのような感じで動くのか

- $n = 20, m = 4$の場合（丸数字は訓練事例の番号，赤色は予測を間違えた事例とする）
  - D_1: ①→②→③→④: ①②④で$\boldsymbol{w}$を更新
  - D_2: ⑤→⑥→⑦→⑧: ⑤⑦⑧で$\boldsymbol{w}$を更新
  - D_3: ⑨→⑩→⑪→⑫: ⑩⑪で$\boldsymbol{w}$を更新
  - D_4: ⑬→⑭→⑮→⑯: ⑬⑮で$\boldsymbol{w}$を更新
  - D_5: ⑰→⑱→⑲→⑳: ⑱で$\boldsymbol{w}$を更新
- 各mini-batch内で全ての訓練事例を処理するまで重みベクトル$\boldsymbol{w}$の更新を行わない
  - mini-batch内でのargmaxの計算を並列化してもよい（重みベクトル$\boldsymbol{w}$を介した依存関係がないため）

定理: 半径がRの訓練データ$D = \{(x_i, y_i)\}_{i=1}^{N}$がマージン$\delta$で分離可能であるとき，mini-batchパーセプトロン・アルゴリズムも$t \leq R^2/\delta^2$回のパラ―メータ更新で終了する

証明: $\boldsymbol{w}^0 = 0$から始めてt回目のパラメータ更新が集合$C_t = \{c_1, \ldots, c_a\}$, $c_k = (x_i, y_i, z_i)$に対して実行され，$\boldsymbol{w}^{t-1}$が$\boldsymbol{w}^t$に更新されるとき，

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} + \frac{1}{|C_t|} \sum_{c_k \in C_t, (x_i, y_i, z_i) \leftarrow c_k} \left(\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z) \right) \equiv \boldsymbol{w}^{t-1} + \frac{1}{a} \sum_i \boldsymbol{v}_i$$

更新式の両辺に$\boldsymbol{u}$をかけると，

$$\boldsymbol{u} \cdot \boldsymbol{w}^t = \boldsymbol{u} \cdot \boldsymbol{w}^{t-1} + \frac{1}{a} \sum_i \boldsymbol{u} \cdot \boldsymbol{v}_i \geq \boldsymbol{u} \cdot \boldsymbol{w}^{t-1} + \frac{1}{a} \cdot a\delta = \boldsymbol{u} \cdot \boldsymbol{w}^{t-1} + \delta \geq t\delta$$

通常のパーセプトロンと同様

更新式の両辺を二乗すると，

$$\|\boldsymbol{w}^t\|^2 = \|\boldsymbol{w}^{t-1}\|^2 + \left\| \frac{1}{a} \sum_i \boldsymbol{v}_i \right\|^2 + \frac{2}{a} \boldsymbol{w}^{t-1} \cdot \sum_i \boldsymbol{v}_i \leq \|\boldsymbol{w}^{t-1}\|^2 + \left\| \frac{1}{a} \sum_i \boldsymbol{v}_i \right\|^2$$

$$\leq \|\boldsymbol{w}^{t-1}\|^2 + \frac{1}{a} \sum_i \|\boldsymbol{v}_i\|^2 \leq \|\boldsymbol{w}^{t-1}\|^2 + \frac{1}{a} \sum_i R^2 = \|\boldsymbol{w}^{t-1}\|^2 + R^2 \leq tR^2$$

イェンゼンの不等式

Mini-batch Perceptronの並列化

1. $\boldsymbol{w} = 0$
2. Dを$\lceil n/m \rceil$個のmini-batch$D_1, \ldots, D_{\lceil n/m \rceil}$に分割
3. D_jを$\lceil m/p \rceil$個のグループ$D_{j,1}, \ldots, D_{j,\lceil m/p \rceil}$に分割
4. **repeat**:
5. **for** j **in range**($\lceil n/m \rceil$):
6. **for** k **in range**($\lceil m/p \rceil$):
7. $C_k = \emptyset$
8. **for** (x_i, y_i) **in** $D_{j,k}$:
9. $z_i \leftarrow \underset{z_i \in \mathrm{Gen}(x_i)}{\mathrm{argmax}}\ \boldsymbol{w} \cdot \boldsymbol{\Phi}(x_i, y_i)$
10. **if** $z_i \neq y_i$:
11. $C_k \leftarrow C_k \cup \{(x_i, y_i, z_i)\}$
12. $C \leftarrow \bigcup_k C_k$
13. **for** (x_i, y_i, z_i) **in** C:
14. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \frac{1}{|C|}\left(\boldsymbol{\Phi}(x_i, y_i) - \boldsymbol{\Phi}(x_i, z)\right)$
15. **until convergence**
16. **return** $\boldsymbol{w}$

n: 訓練事例数
m: mini-batch数
p: 並列実行数

p個のスレッドで並列実行

1個のスレッドで実行

並列化方法のまとめ

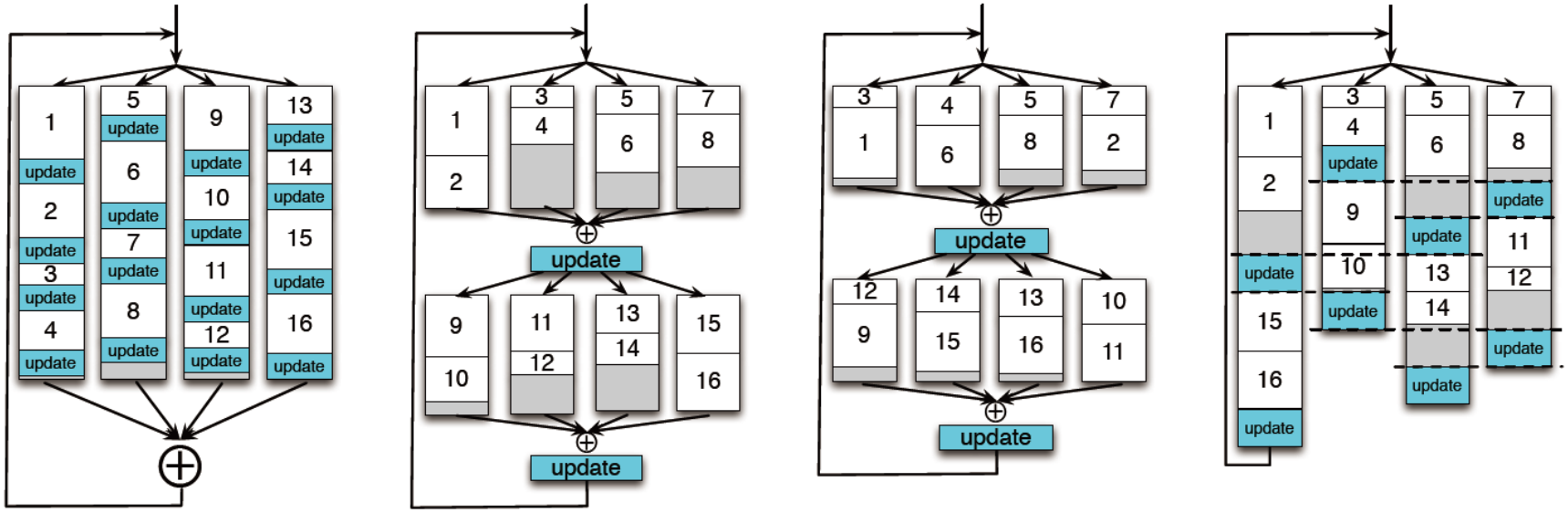


Figure 1: Comparison of various methods for parallelizing online learning (number of processors $p = 4$). (a) iterative parameter mixing (McDonald et al., 2010). (b) unbalanced minibatch parallelization (minibatch size $m = 8$). (c) minibatch parallelization after load-balancing (within each minibatch). (d) asynchronous minibatch parallelization (Gimpel et al., 2010) (not implemented here). Each numbered box denotes the decoding of one example, and $\oplus$ denotes an aggregate operation, i.e., the merging of constraints after each minibatch or the mixing of weights after each iteration in IPM. Each gray shaded box denotes time wasted due to synchronization in (a)-(c) or blocking in (d). Note that in (d) at most one update can happen concurrently, making it substantially harder to implement than (a)-(c).

Zhao and Huang (2013) より

評価実験

実験設定

- 既存の２つのタスクを学習させる
  - 係り受け解析
  - 品詞タグ付け
- Pythonで実装
  - multiprocessingモジュールを用いて並列化 (!)
- 実験環境
  - ８コアのCPU（3.1GHz）×２
  - 64GBの主記憶

Figure 2: Mini-batch MIRAによる係り受け解析（並列化なし）

- Mini-batchのサイズを大きくしていくと，反復が終了したときのモデルの精度が良くなる傾向
  - 沢山の訓練事例を使って重みベクトルを更新するため
- Mini-batchのサイズを適度に調整すれば，学習の速度（同じ精度を達成するまでに必要な時間）も速くなることがある
- Mini-batchを大きくしすぎると，学習が遅くなる
  - Mini-batch内の訓練事例間の情報が重みベクトルに反映されるのが遅れるため
  - ただ，Mini-batchを大きくするほど並列実行しやすくなるので，並列度を上げることで学習の遅さをカバーできるかもしれない

Figure 3: Mini-batch MIRAとIPMによる係り受け解析（並列化有り）

- 上段: Mini-batchのサイズを24としたとき，並列実行数を1, 4, 12と増やしていくと学習が速くなる

- 下段: IPMで並列実行数4, 12と増やしても，学習がそんなに速くならない

- まとめ: IPMよりも提案手法の方が学習の効率がよく，得られるモデルの精度も高い

Figure 4: 並列実行数と速度向上の関係

- Extrinsicな評価: 精度が92.27になるまでの時間を計測
  - 提案手法は線形をやや下回るような速度向上
  - IPMは並列実行数を増やしても速度向上が見られない
- Intrinsicな評価: 全訓練事例をひと通り処理するまでの時間を計測
  - 提案手法は並列実行数が2や4の時は線形な速度向上が見られる
  - 並列実行数が12になると線形な速度向上の50%くらいまで落ちる
    - 各スレッドの負荷が均一にならない
    - 間違えた事例の集約や重み更新は1スレッドで行い，残りの$p-1$スレッドは待つことになるため

Figures 5-8: パーセプトロンによる品詞タグ付けの実験

- Figure 5: Mini-batchにより精度が向上
- Figure 6: 並列実行をした場合，IPMよりも提案手法の方が速いが，係り受け解析の時ほどの差はない
- Figure 7: Intrinsicな評価では提案手法の方が遅いものの，Extrinsicな評価では提案手法の方が速い
- Figure 8: 各スレッドの負荷が均一になるように工夫すると，CPUの利用効率が上がる

まとめ

- Mini-batchによりオンライン学習（パーセプトロンとMIRA）を並列化するアルゴリズムを提案し，収束に関する理論的な解析を行った
- 評価実験により，既存手法（IPM）よりも学習が速く，高精度が達成できることを示した
- 重みの更新が並列に実行できない
  - Map-Reduceのreduce処理みたいに並列化すればよいと思う（効果は不明だが）
- 訓練事例をクラスタ上に分散配置する場合には向かない
  - クラスタ間で学習時の重みを効率よく交換する戦略が必要