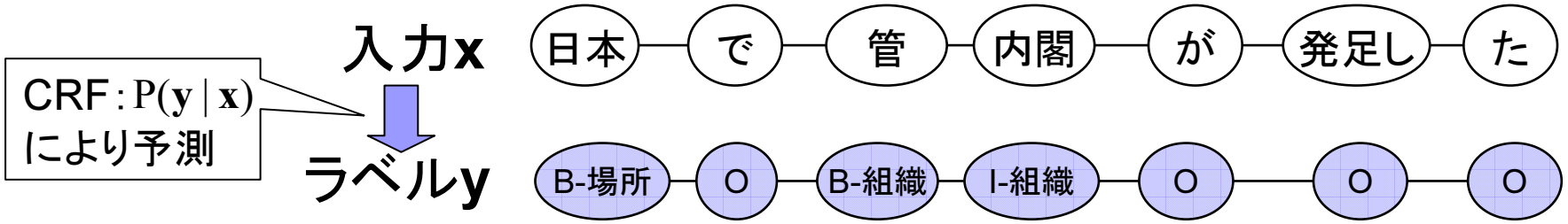
言語処理学会年次大会発表資料

# Newton-CG法による条件付き確率場のバッチ学習

坪井祐太[*1], 海野裕也[*1],
鹿島久嗣[*2], 岡崎直観[*2]
*1:日本アイ・ビー・エム株式会社, *2: 東京大学

# 研究概要：直鎖条件付き確率場（CRF）の高速なバッチ学習法の提案

- 直鎖CRF： 入力(文)からラベル列を予測するモデル
  - 応用：品詞タグ付け、句チャンキング、固有表現抽出など




- オンライン学習とバッチ学習

| | 学習頻度 | 利点 | 欠点 |
|---|---|---|---|
| **オンライン学習** | 1データごと | 性能の立ち上がりが早い | 問題依存の設定が必要（停止条件・更新率など） |
| **バッチ学習** | 全データごと | 設定は問題非依存で高精度（black box利用可） | 性能の立ち上がりが遅い |

# 本研究の貢献

1. CRF学習へのNewton-CG法の適用
   - 動的計画法によるヘシアン・ベクトル積計算
     : 計算・空間量$O(T|Y|^2)$、ただしTは列長。
2. CRF学習でのNewton-CG法の高速化
   - 周辺確率$P(y_{t-1}, y_t|\mathbf{x})$の再利用による指数関数呼び出し回数の削減

# CRF学習は関数最小化問題

- CRF:入力**x**に対する出力列**y**の条件付き確率のモデル:

$$P_{\theta}(\mathbf{y}\mid\mathbf{x})=\frac{\exp\left(\boldsymbol{\theta}^{T}\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\right)}{\sum_{\tilde{\mathbf{y}}\in\mathbf{Y}}\exp\left(\boldsymbol{\theta}^{T}\boldsymbol{\Phi}(\mathbf{x},\tilde{\mathbf{y}})\right)}$$

$\boldsymbol{\theta}\in\mathrm{R}^{d}$（d次元のモデルパラメータ）

$\boldsymbol{\Phi}(\mathbf{x},\mathbf{y}):\mathbf{X}\times\mathbf{Y}\rightarrow\mathrm{R}^{d}$

（素性関数）

- 正則化項付きの負の対数尤度関数の最小化

**（最小化したい）目的関数:** $f(\boldsymbol{\theta})=-\sum_{(\mathbf{x},\mathbf{y})\in D}\ln P_{\theta}(\mathbf{y}\mid\mathbf{x})+\frac{\|\boldsymbol{\theta}\|^{2}}{2\sigma^{2}}$

*D*は訓練データ集合

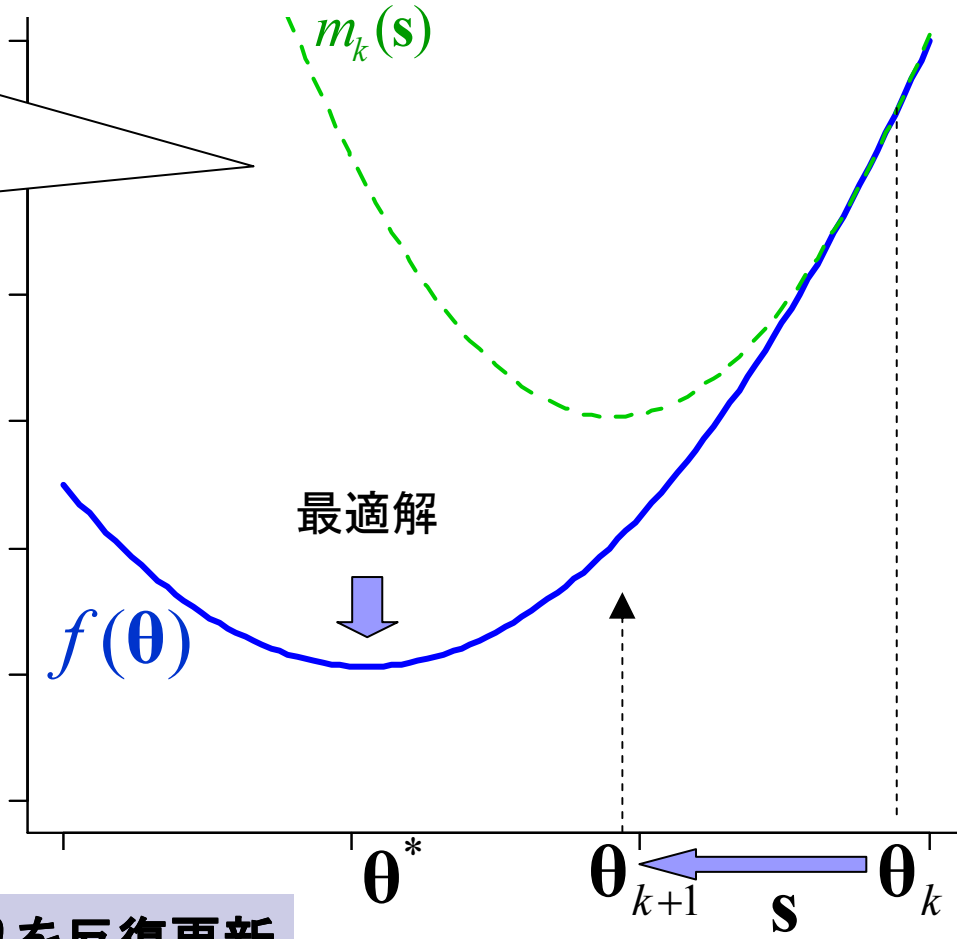
# 関数の最小化：Newton法(1)

モデル（2次曲線）で目的関数を近似

$$f(\boldsymbol{\theta}_k + \mathbf{s}) \approx m_k(\mathbf{s}) = f(\boldsymbol{\theta}_k) + \mathbf{g}_k{}^T(\mathbf{s} - \boldsymbol{\theta}_k) + (\mathbf{s} - \boldsymbol{\theta}_k)^T \mathbf{H}_k (\mathbf{s} - \boldsymbol{\theta}_k)$$

$\mathbf{g}_k = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_k) \in \mathrm{R}^d$
は勾配ベクトル

$\mathbf{H}_k \equiv \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_k) \in \mathrm{R}^{d \times d}$
はヘシアン行列




**θを反復更新**

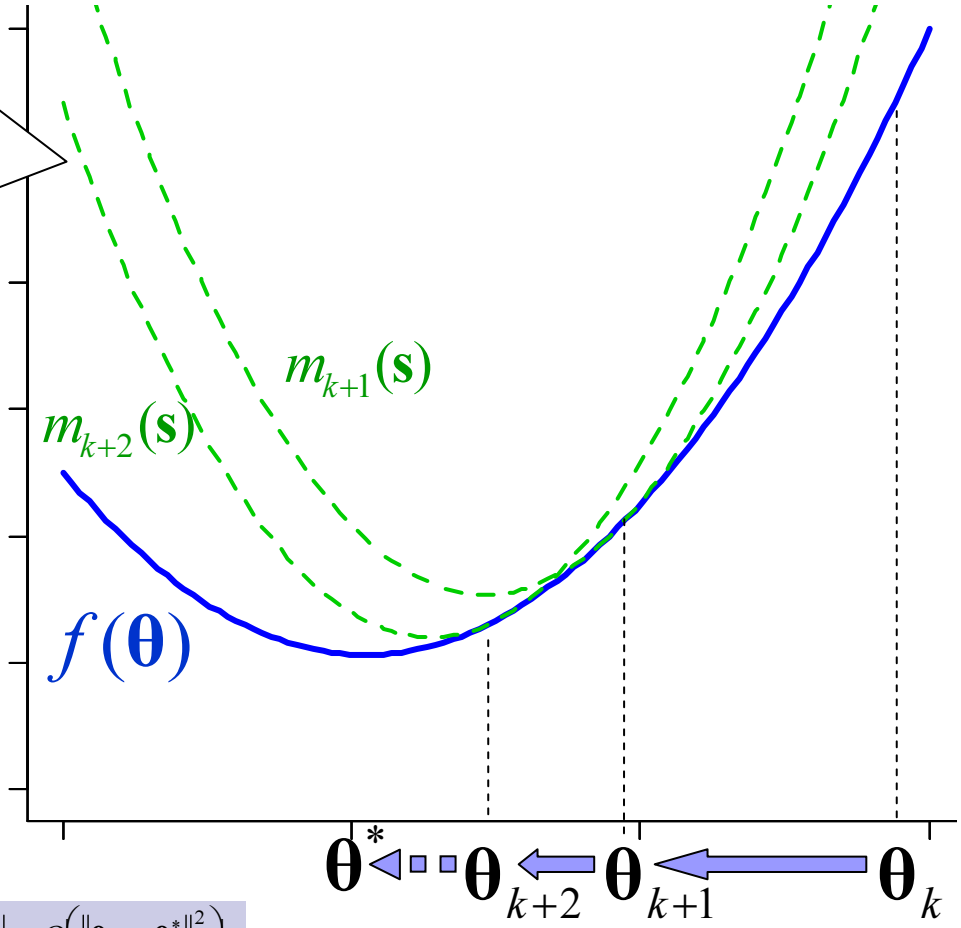
# 関数の最小化：Newton法(2)

モデル（2次曲線）で目的関数を近似

$$f(\boldsymbol{\theta}_k + \mathbf{s}) \approx m_k(\mathbf{s})$$
$$= f(\boldsymbol{\theta}_k) + \mathbf{g}_k{}^T(\mathbf{s} - \boldsymbol{\theta}_k)$$
$$+ (\mathbf{s} - \boldsymbol{\theta}_k)^T \mathbf{H}_k (\mathbf{s} - \boldsymbol{\theta}_k)$$

$\mathbf{g}_k = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_k) \in \mathrm{R}^d$
は勾配ベクトル

$\mathbf{H}_k \equiv \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_k) \in \mathrm{R}^{d \times d}$
はヘシアン行列




**最適解付近では2次収束**　$\left\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\right\| = O\left(\left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|^2\right)$

# 関数の最小化：Newton法(3)

- 目的関数を2次のテーラー展開で近似

$$f(\boldsymbol{\theta}_k + \mathbf{s}) \approx \mathrm{m}_k(\mathbf{s}) = f(\boldsymbol{\theta}_k) + \mathbf{g}_k{}^T(\mathbf{s} - \boldsymbol{\theta}_k) + (\mathbf{s} - \boldsymbol{\theta}_k)^T \mathbf{H}_k(\mathbf{s} - \boldsymbol{\theta}_k)$$

⇩ 2次関数$\mathrm{m}_k$を最小にするsの条件

$$\nabla_{\mathbf{s}} \mathrm{m}_k(\mathbf{s}) \approx \mathbf{g}_k + \mathbf{H}_k \mathbf{s} = 0 \text{（Newton方程式）}$$

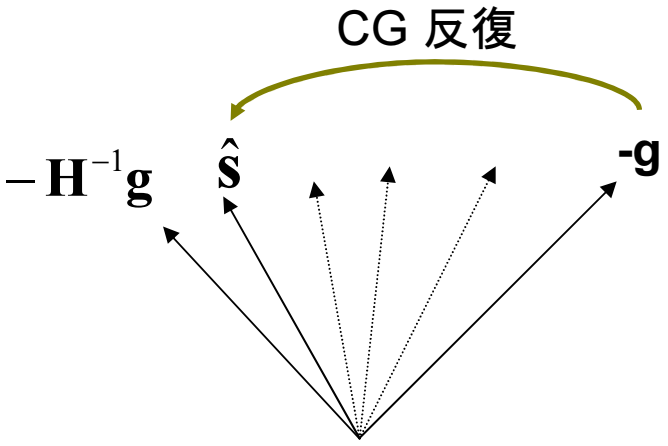
- 2次関数$\mathrm{m}_k$を最小化するステップ $\mathbf{s}^* = -\mathbf{H}_k^{-1}\mathbf{g}_k$
（Newtonステップ）で$\theta$を更新($\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$ )
- 高次元問題でのNewton法の課題
→巨大なd×d のヘシアン行列保持・逆行列の計算

# 関数の最小化：Newton-CG法

- Newton方程式を共役勾配(CG)法で解く

~~$\mathbf{s}^* = -\mathbf{H}_k^{-1}\mathbf{g}_k$~~

$$\Rightarrow \min_{\mathbf{s}} f(\boldsymbol{\theta}_k) + \mathbf{g}_k{}^T\mathbf{s} + \mathbf{s}^T\mathbf{H}_k\mathbf{s}$$




- Newton-CG法の利点
  - 適応的な精度調整(Truncated Newton method)
    - 最適解近くで正確なNewtonステップが求まれば十分
  - ヘシアン行列は不要（Hessian-free method）
    - 必要：d次元のヘシアン・ベクトル積 $\mathbf{H}_k\mathbf{r} \in \mathrm{R}^d$

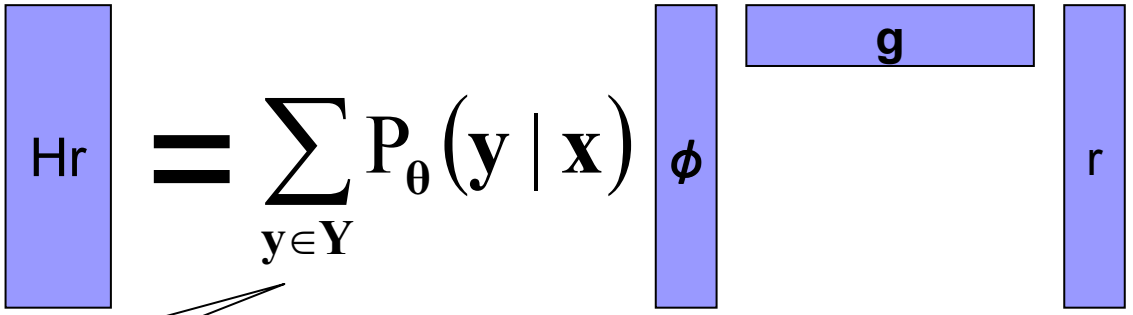
本研究の貢献

# CRFのヘシアン・ベクトル積：

$$\mathbf{Hr} = \sum_{(\mathbf{x},\mathbf{y})\in D} \mathbf{H}(\mathbf{x},\mathbf{y})\mathbf{r} + \frac{\mathbf{r}}{\sigma^2}$$

- 事例(**x**,**y**)毎のヘシアン・ベクトル積に分解可能

$$\mathbf{H}(\mathbf{x},\mathbf{y})\mathbf{r} = \sum_{\mathbf{y}\in\mathbf{Y}} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}\mid\mathbf{x})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\mathbf{g}(\mathbf{x},\mathbf{y})^T\mathbf{r}$$




$$\sum_{\mathbf{y}\in\mathbf{Y}} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}\mid\mathbf{x})$$

指数個の和
→動的計画法！

ただし、$\mathbf{g}(\mathbf{x},\mathbf{y}) = \boldsymbol{\Phi}(\mathbf{x},\mathbf{y}) - \mathrm{E}_{\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}[\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})]$

# CRFのヘシアン・ベクトル積の効率的な計算方法(1): ポイント

- 効率的な計算のポイント
  1. 行列を生成しない計算順
     - 行列×ベクトル→ベクトル×スカラー
  2. 動的計画法
     - 素性関数と確率の分解（マルコフ性を仮定）

$$\mathbf{\Phi}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} \phi(y_{t-1}, y_t \mid \mathbf{x}) \qquad \text{（素性関数の分解）}$$

$$\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}) = \mathrm{P}_{\boldsymbol{\theta}}(y_0 \mid y_1) \times \cdots \times \mathrm{P}_{\boldsymbol{\theta}}(y_{t-2} \mid y_{t-1}) \times \mathrm{P}_{\boldsymbol{\theta}}(y_{t-1}, y_t) \times \mathrm{P}_{\boldsymbol{\theta}}(y_{t+1} \mid y_t) \times \cdots \times \mathrm{P}_{\boldsymbol{\theta}}(y_T \mid y_{T-1})$$

（P(y|x)の分解 – 注: x省略）

# CRFのヘシアン・ベクトル積の効率的な計算方法(2): 計算順

$$\mathbf{H}(\mathbf{x},\mathbf{y})\mathbf{r} = \sum_{\mathbf{y}\in\mathbf{Y}} \mathrm{P}_{\theta}(\mathbf{y}\mid\mathbf{x})\left(\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\mathbf{g}(\mathbf{x},\mathbf{y})^{\mathrm{T}}\right)\mathbf{r}$$

ベクトル ＝ 行列 × ベクトル

$$\mathbf{H}(\mathbf{x},\mathbf{y})\mathbf{r} = \sum_{\mathbf{y}\in\mathbf{Y}} \mathrm{P}_{\theta}(\mathbf{y}\mid\mathbf{x})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\left(\mathbf{g}(\mathbf{x},\mathbf{y})^{\mathrm{T}}\mathbf{r}\right)$$

ベクトル ＝ ベクトル × スカラー

**ポイント1**
$\psi(\mathbf{x},\mathbf{y})^{\mathrm{T}}\mathbf{r}$
を先に
計算する
ことで
行列を
計算する
ことを避け
る

# CRFのヘシアン・ベクトル積の効率的な計算方法(3): 点毎に分解

$$\sum_{\mathbf{y}\in\mathbf{Y}} \mathrm{P}_{\theta}(\mathbf{y}\mid\mathbf{x})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})^{\mathrm{T}} = \sum_{t=1}^{T}\sum_{y_{t-1},y_t}\phi(\mathbf{x},y_{t-1},y_t)\left[\sum_{\tilde{\mathbf{y}}:\tilde{y}_{t-1}=y_{t-1}\wedge\tilde{y}_t=y_t}\Pr(\tilde{\mathbf{y}}\mid\mathbf{x})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})^{\mathrm{T}}\right]$$

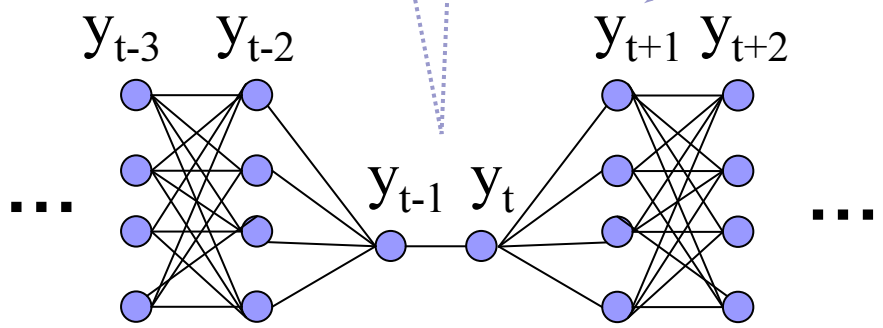
簡単のため $\phi$ で説明

$y_{t-1}y_t$をあるラベル組に固定した列 $\tilde{\mathbf{y}}$ すべての和

Naïveに計算すると計算量$O(T^2)$→乗算の再帰に分解

$$\sum_{\tilde{\mathbf{y}}:\tilde{y}_{t-1}=y_{t-1}\wedge\tilde{y}_t=y_t}\Pr(\tilde{\mathbf{y}}\mid\mathbf{x})\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})^{\mathrm{T}}$$

$$= \Pr(y_{t-1},y_t\mid\mathbf{x})\left(\sum_{y_{t-2}}\left[\Pr(y_{t-2}\mid y_{t-1}\mathbf{x})\left(\phi(\mathbf{x},y_{t-2},y_{t-1})^{\mathrm{T}}+\sum_{y_{t-3}}[\cdots]\right)\right]+\phi(\mathbf{x},y_{t-1},y_t)^{\mathrm{T}}+\sum_{y_{t+1}}\left[\Pr(y_{t+1}\mid y_t\mathbf{x})\left(\phi(\mathbf{x},y_t,y_{t+1})^{\mathrm{T}}+\sum_{y_{t+2}}[\cdots]\right)\right]\right)$$



# CRFのヘシアン・ベクトル積の効率的な計算方法(4): 動的計画法

- $y_{t-1}$, $y_t$を固定した全列和は表参照で計算

$$\sum_{\tilde{\mathbf{y}}:\tilde{y}_{t-1}=y_{t-1}\wedge\tilde{y}_t=y_t} \Pr(\tilde{\mathbf{y}}\mid\mathbf{x})\mathbf{g}(\mathbf{x},\mathbf{y})^{\mathrm{T}}\mathbf{r} =$$

$$\mathrm{P}_{\boldsymbol{\theta}}(\tilde{y}_{t-1},\tilde{y}_t\mid\mathbf{x})\left[\mathrm{A}[t-1,\tilde{y}_{t-1}]+\boldsymbol{\phi}(\mathbf{x},\tilde{y}_{t-1},\tilde{y}_t)^{\mathrm{T}}\mathbf{r}+\mathrm{B}[t,\tilde{y}_t]-\mathrm{E}_{\mathrm{P}_{\boldsymbol{\theta}}}[\boldsymbol{\Phi}(\mathbf{x},\mathbf{y})]^{\mathrm{T}}\mathbf{r}\right]$$

- 表A, Bの計算・サイズはO(|Y|$^2$T)

$$\mathrm{A}[t,y_t]=\sum_{y_{t-1}\in Y}\mathrm{P}_{\boldsymbol{\theta}}(y_{t-1}\mid y_t,\mathbf{x})\boldsymbol{\phi}(\mathbf{x},y_{t-1},y_t)^{\mathrm{T}}\mathbf{r}+\mathrm{A}[t-1,y_{t-1}]$$

（終端条件略）

$$\mathrm{B}[t,y_t]=\sum_{j\in Y}\mathrm{P}_{\boldsymbol{\theta}}(y_{t+1}\mid y_t,\mathbf{x})\boldsymbol{\phi}(\mathbf{x},y_t,y_{t+1})^{\mathrm{T}}\mathbf{r}+\mathrm{B}[t+1,y_{t+1}]$$

# ヘシアン・ベクトル積の計算時間

- 計算時間の大部分はラベル組の条件付き確率計算が占める

$$\mathrm{A}[t, y_t] = \sum_{y_{t-1} \in Y} \mathrm{P}_{\boldsymbol{\theta}}(y_{t-1} \mid y_t, \mathbf{x}) \boldsymbol{\phi}(\mathbf{x}, y_{t-1}, y_t)^{\mathrm{T}} \mathbf{r} + \mathrm{A}[t-1, y_{t-1}]$$

$$\mathrm{B}[t, y_t] = \sum_{j \in Y} \mathrm{P}_{\boldsymbol{\theta}}(y_{t+1} \mid y_t, \mathbf{x}) \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t+1})^{\mathrm{T}} \mathbf{r} + \mathrm{B}[t+1, y_{t+1}]$$

→理由：確率計算時の指数関数呼び出しコストは加算・積算の15-40倍

# Newton-CG法でのヘシアン・ベクトル積計算の効用→計算時間に最も影響する確率計算の回数を大幅に削減可能

- Newton-CG法のCG反復（各kで数回から数百回）
  - Newtonステップを探すために点列$\mathbf{r}_1 \cdots \mathbf{r}_\ell \cdots$に対して$\mathbf{H}_k \mathbf{r}_\ell$を評価
  - 反復中はパラメータ$\boldsymbol{\theta}$は変わらない

  →条件付き確率$\mathrm{P}_{\boldsymbol{\theta}}(y_{t-1} \mid y_t, \mathbf{x}), \mathrm{P}_{\boldsymbol{\theta}}(y_{t+1} \mid y_t, \mathbf{x})$も変わらない
- CRFの勾配計算での周辺確率を再利用し、条件付き確率を計算可能

$$\mathrm{P}_{\boldsymbol{\theta}}(y_{t-1} \mid y_t, \mathbf{x}) = \mathrm{P}_{\boldsymbol{\theta}}(y_{t-1}, y_t \mid \mathbf{x}) / \mathrm{P}_{\boldsymbol{\theta}}(y_t \mid \mathbf{x})$$
$$\mathrm{P}_{\boldsymbol{\theta}}(y_{t+1} \mid y_t, \mathbf{x}) = \mathrm{P}_{\boldsymbol{\theta}}(y_t, y_{t+1} \mid \mathbf{x}) / \mathrm{P}_{\boldsymbol{\theta}}(y_t \mid \mathbf{x})$$

勾配計算の中間結果を再利用可能

**CG反復中、コストの大きいラベル組確率の計算は一回のみでOK**

# 本研究の貢献(再掲)

1. CRF学習へのNewton-CG法の適用
   - 動的計画法によるヘシアン・ベクトル積計算
     : 計算・空間量$O(T|Y|^2)$、ただしTは列長。
2. CRF学習でのNewton-CG法の高速化
   - 周辺確率$P(y_{t-1}, y_t|\mathbf{x})$の再利用による指数関数呼び出し回数の削減

# 基本句チャンキングタスクと固有表現抽出における比較実験: 比較手法

- バッチ学習: 準ニュートン法(LBFGS)
  ヘシアンの逆行列 $\mathbf{H}^{-1}$ を2m個のベクトルで近似
  （以降、**LBFGS(m=xx)** ）
- オンライン学習: 確率的勾配法(SGD)
  1事例（文）だけで勾配を評価し、パラメータ更新:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \left( \mathbf{g}(x, y) + \frac{\boldsymbol{\theta}_k}{n\sigma^2} \right)$$

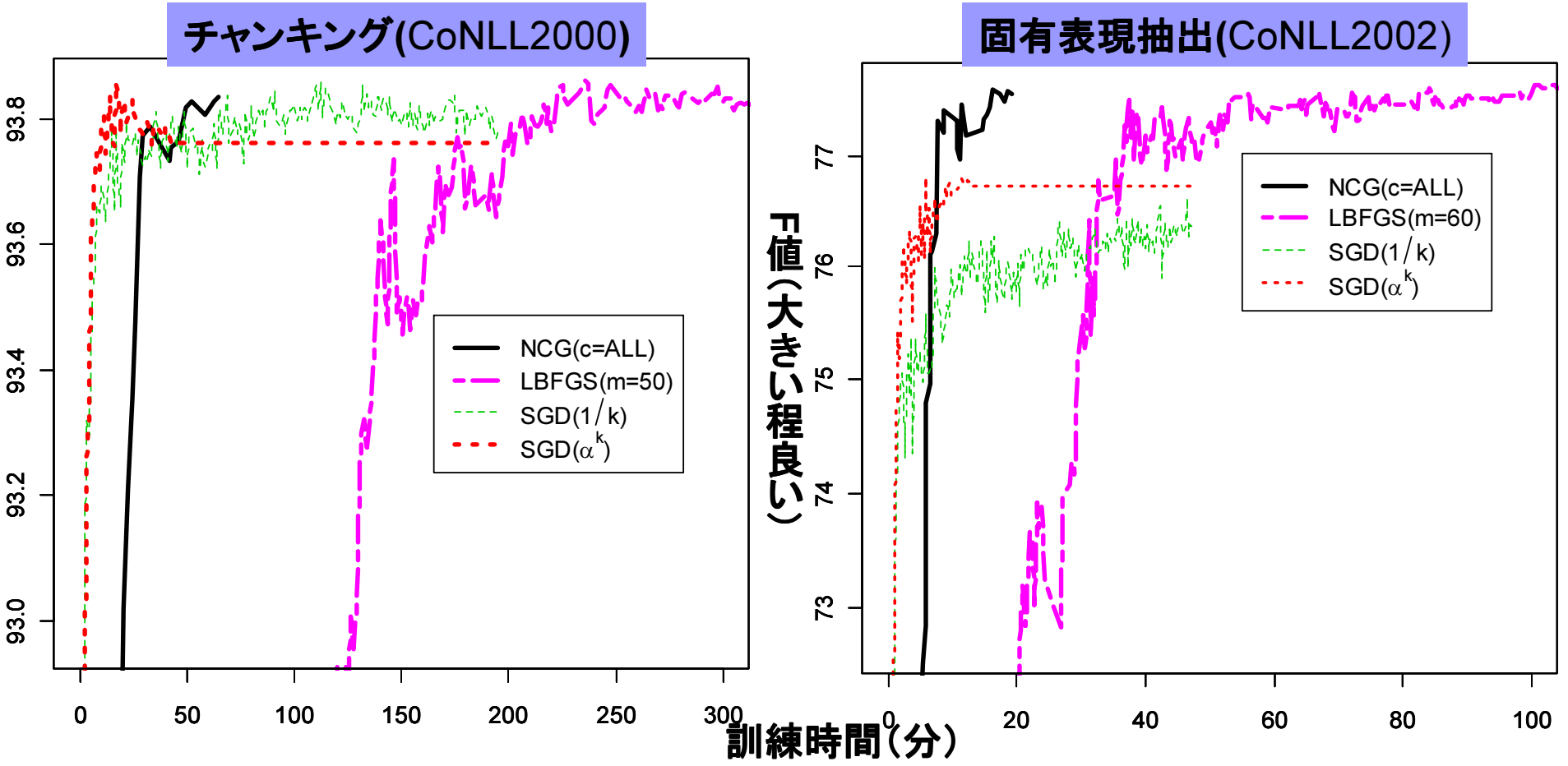
**更新率**

1. 逆数で減衰 $\eta_k = \dfrac{\eta_0}{1 + \dfrac{k}{n}}$
   以降、**SGD(1/k)**
   （Darken and Moody 1992, Collins et al. 2008）
2. 指数で減衰 $\eta_k = \eta_0 \alpha^{k/n} \quad (0 < \alpha < 1)$
   以降、**SGD( $\alpha^k$ )**
   (Tsuruoka et al. 2009)

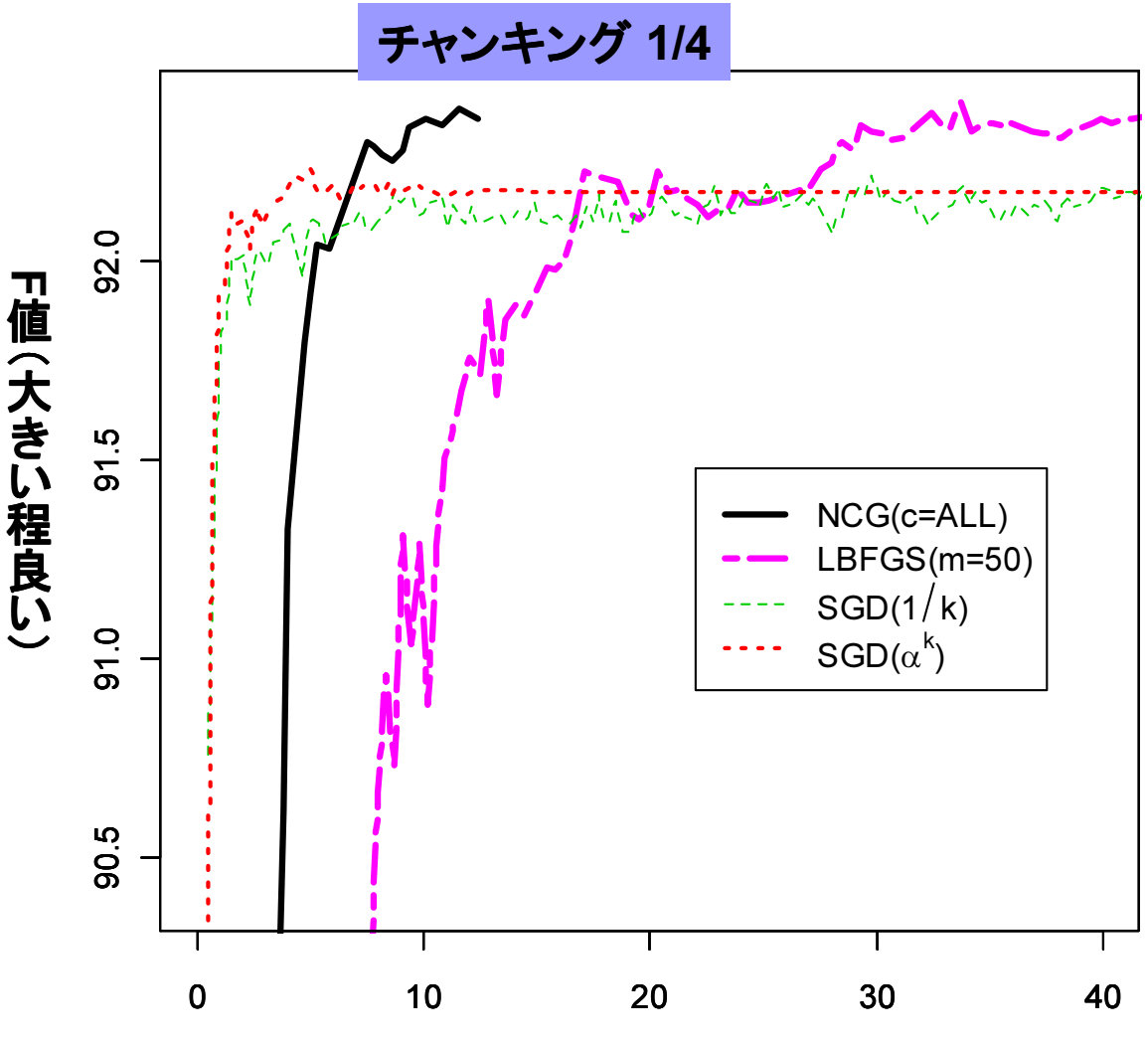
$\eta_0$ : 初期更新率

# 基本句チャンキングと固有表現抽出タスクの訓練時間とF値

- 提案手法はタスクによらず安定した性能を示した



# 小規模データ(チャンキング 訓練2,234文)での訓練時間とF値




- 提案法はデータサイズによらず安定した性能を示した

# まとめ

- CRFの高速バッチ学習法の提案
  - 既存のバッチ学習(LBFGS)より数倍高速
  - オンライン学習法と同等の速度かつ安定した性能（タスク・データサイズ非依存）

→小・中規模タスクでは最適なCRF学習手法

# 補足資料

# 今後の課題

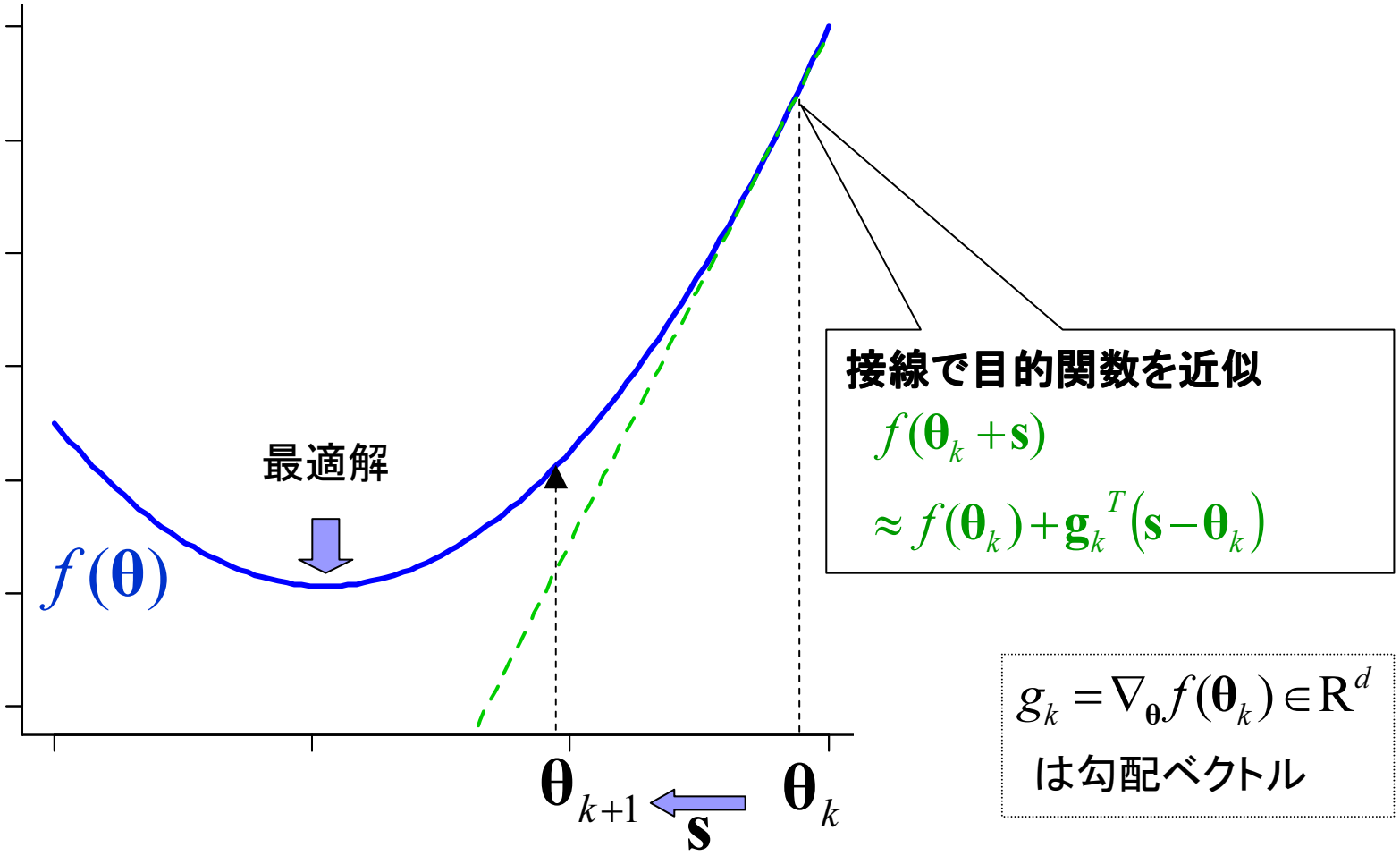
- スパース学習への展開
- オンライン学習への展開

# 注：論文と発表資料との記号の違い

- 説明のため、以下は論文とは違う記号を使用しています。

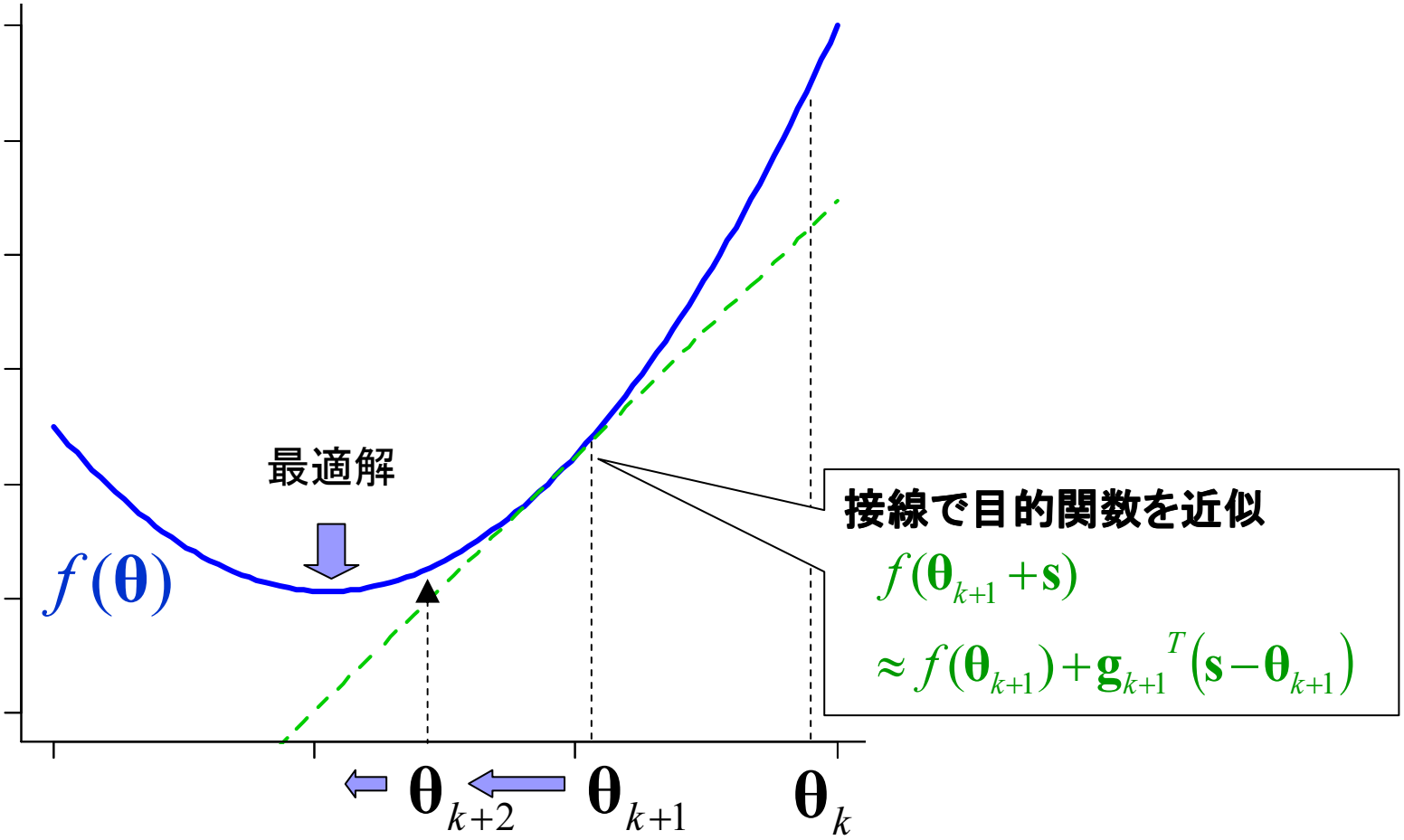
| 論文 | 本発表資料 |
|---|---|
| $\psi(x,y)$ | $g(x,y)$ |
| $\mathbf{Hd}$ | $\mathbf{Hr}$ |
| $\mathbf{H}_{(x,y)}$ | $\mathbf{H}(x,y)$ |

# 関数の最小化：最急降下法(1)




**接線に沿ってθを更新**

# 関数の最小化：最急降下法(2)




勾配gが0に近づくまでθを反復更新

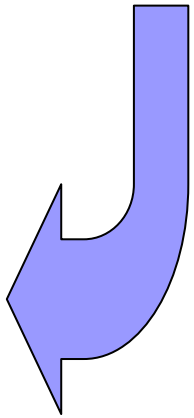
# Newton-CG法擬似コード

- TN methods $k = 0, \boldsymbol{\theta}_0 = \mathbf{0}$
  1. 訓練データすべてを使用して勾配 $\mathbf{g}_k$ 計算
     a. Forward-backward アルゴリズムで周辺確率計算
     b. 周辺確率を用いて $\mathbf{g}(\mathbf{x}, \mathbf{y})$ を計算
  2. CG反復
     a. 訓練データすべてを使用して
        ヘシアン・ベクトル積計算
        i. Forward-backwardアルゴリズムで
           周辺確率計算
        ii. 周辺確率を用いて $\mathbf{H}_k(\mathbf{x}, \mathbf{y})\mathbf{r}_\ell$ を計算
     b. $\mathbf{H}_k\mathbf{r}_\ell + \mathbf{g}_k \approx 0$ ならば $\mathbf{s}_k = \mathbf{r}_\ell$ , otherwise $\ell = \ell + 1$
  3. Update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{s}_k$
  4. k = k+1 and goto step 1

$\boldsymbol{\theta}_k$ は固定

# Newton-CG法高速化 擬似コード

- TN methods $k = 0, \boldsymbol{\theta}_0 = \mathbf{0}$
  1. 訓練データすべてを使用して勾配 $\mathbf{g}_k$ 計算
     a. Forward-backward アルゴリズムで周辺確率計算
     b. **周辺確率を用いて $\mathbf{g}(\mathbf{x},\mathbf{y})$ を計算**
  2. CG反復
     a. 訓練データすべてを使用して
        ヘシアン・ベクトル積計算
        i. **周辺確率を用いて $\mathbf{H}_k(\mathbf{x},\mathbf{y})\mathbf{r}_\ell$ を計算**
     b. $\mathbf{H}_k\mathbf{r}_\ell + \mathbf{g}_k \approx 0$ならば $\mathbf{s}_k = \mathbf{r}_\ell$, otherwise $\ell = \ell + 1$
  3. Update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{s}_k$
  4. k = k+1 and goto step 1



周辺確率
を再利用

# 提案手法のメモリー使用量

- 周辺確率$P(y_{t-1},y_t|\mathbf{x})$を再利用するために各事例に対してO(T|Y|^2)必要（1次のマルコフモデルの場合）
- c個の事例の周辺確率を覚えるには、O(cT|Y|^2)必要
  - c << n の場合は提案するNewton-CG法の高速化の効果は限定的。ただし、nは全事例数
    - 注：オリジナルのNewton-CG法は周辺確率を覚える必要が無い

# 信頼区間法＆Newton-CG法（1）

- Newton-CG法の大域最適解への収束を保障するために、本研究では信頼区間法とNewton-CG法を組み合わせた。
- Δは信頼区間を示す。モデルと目的関数との違いρの値に基づきΔは更新される。
- 信頼区間法と組み合わせるときにはステップが信頼区間Δに収まるようにCG法は制約付最小化問題を解く

**Algorithm 1** A trust-region Newton-CG method

Given $\boldsymbol{\theta}_0 = 0, \Delta_0 = ||\boldsymbol{g}_0||, \upsilon_0 = 10^{-4}$:
**for** $k = 0, 1, 2, \cdots$ **do**
  **if** converged **then**
    **return** $\boldsymbol{\theta}_k$ and stop.
  **end if**
  CG iteration: Find $\boldsymbol{s}_k$ by approximately solving a subproblem:

$$\min_{\boldsymbol{s}} m_k(\boldsymbol{s}) = f_k + (\boldsymbol{g}_k)^\top \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{H}_k \boldsymbol{s}$$
$$\text{subject to } ||\boldsymbol{s}|| \leq \Delta_k.$$

  Set $\rho_k = \frac{f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}_k + \boldsymbol{s}_k)}{m_k(0) - m_k(\boldsymbol{s}_k)} = \frac{f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}_k + \boldsymbol{s}_k)}{-(\boldsymbol{g}_k)^\top \boldsymbol{s} - \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{H}_k \boldsymbol{s}}$
  Update $\Delta_{k+1} = \Lambda(\rho_k)\Delta_k$. Note that

$$\Lambda(\rho) = \begin{cases} \delta_1 & \text{if } \rho < \upsilon_1 \\ 1 & \text{if } \upsilon_1 \leq \rho < \upsilon_2 \\ \delta_2 & \text{if } \upsilon_2 \leq \rho < \upsilon_3 \\ \delta_3 & \text{if } \rho > \upsilon_3, \end{cases}$$

  where we set $\delta_1 = 0.5, \delta_2 = 2, \delta_3 = 1.01, \upsilon_1 = 0.01, \upsilon_2 = 0.95,$ and $\upsilon_3 = 1.05$.
  **if** $\rho_k > \upsilon_0$ **then**
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{s}_k$
  **else**
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k$
  **end if**
**end for**

# 信頼区間法＆Newton-CG法(2)

- 信頼区間制約付CG法では、信頼区間を超えたステップとなったときは、信頼区間内に射影して終了する
- **Hr**は周辺確率を再利用して計算（高速化）

**Algorithm 2** Conjugate gradient procedure

Input: $\boldsymbol{\theta}_k$, $\boldsymbol{g}_k$ and the marginal probabilities used in the $\boldsymbol{g}_k$ evaluation.
Set $\boldsymbol{p}_0 = 0, \boldsymbol{q}_0 = \boldsymbol{g}_k$, $\boldsymbol{r}_0 = -\boldsymbol{q}_0$, and tolerance $\epsilon_k = \min(0.5, ||\boldsymbol{g}_k||)\,||\boldsymbol{g}_k||$.
**for** $\ell = 0, 1, 2, \cdots$ **do**
  Evaluate $\boldsymbol{H}_k \boldsymbol{r}_\ell$ using the marginal probabilities.
  **if** $\boldsymbol{r}_\ell^\top \boldsymbol{H}_k \boldsymbol{r}_\ell \leq 0$ **then**
    Find the positive root $\tau$ of $||\boldsymbol{p}_\ell + \tau \boldsymbol{r}_\ell|| = \Delta$
    **return** $\boldsymbol{s}_k = \boldsymbol{p}_\ell + \tau \boldsymbol{r}_\ell$ and stop
  **end if**
  $a_\ell = \frac{||\boldsymbol{q}_\ell||^2}{\boldsymbol{r}_\ell^\top \boldsymbol{H}_k \boldsymbol{r}_\ell}$.
  $\boldsymbol{p}_{\ell+1} = \boldsymbol{p}_\ell + a_\ell \boldsymbol{r}_\ell$.
  **if** $||\boldsymbol{p}_{\ell+1}||^2 \geq \Delta_k^2$ **then**
    Find the positive root $\tau$ of $||\boldsymbol{p}_\ell + \tau \boldsymbol{r}_\ell|| = \Delta$
    **return** $\boldsymbol{s}_k = \boldsymbol{p}_\ell + \tau \boldsymbol{r}_\ell$ and stop
  **end if**
  $\boldsymbol{q}_{\ell+1} = \boldsymbol{q}_\ell + a_\ell \boldsymbol{H}_k \boldsymbol{r}_\ell$.
  **if** $||\boldsymbol{q}_{\ell+1}|| \leq \epsilon_k$ **then**
    **return** $\boldsymbol{s}_k = \boldsymbol{p}_{\ell+1}$ and stop.
  **end if**
  $b_\ell = \frac{||\boldsymbol{q}_{\ell+1}||^2}{||\boldsymbol{q}_\ell||^2}$.
  $\boldsymbol{r}_{\ell+1} = -\boldsymbol{q}_{\ell+1} + b_\ell \boldsymbol{r}_\ell$.
**end for**

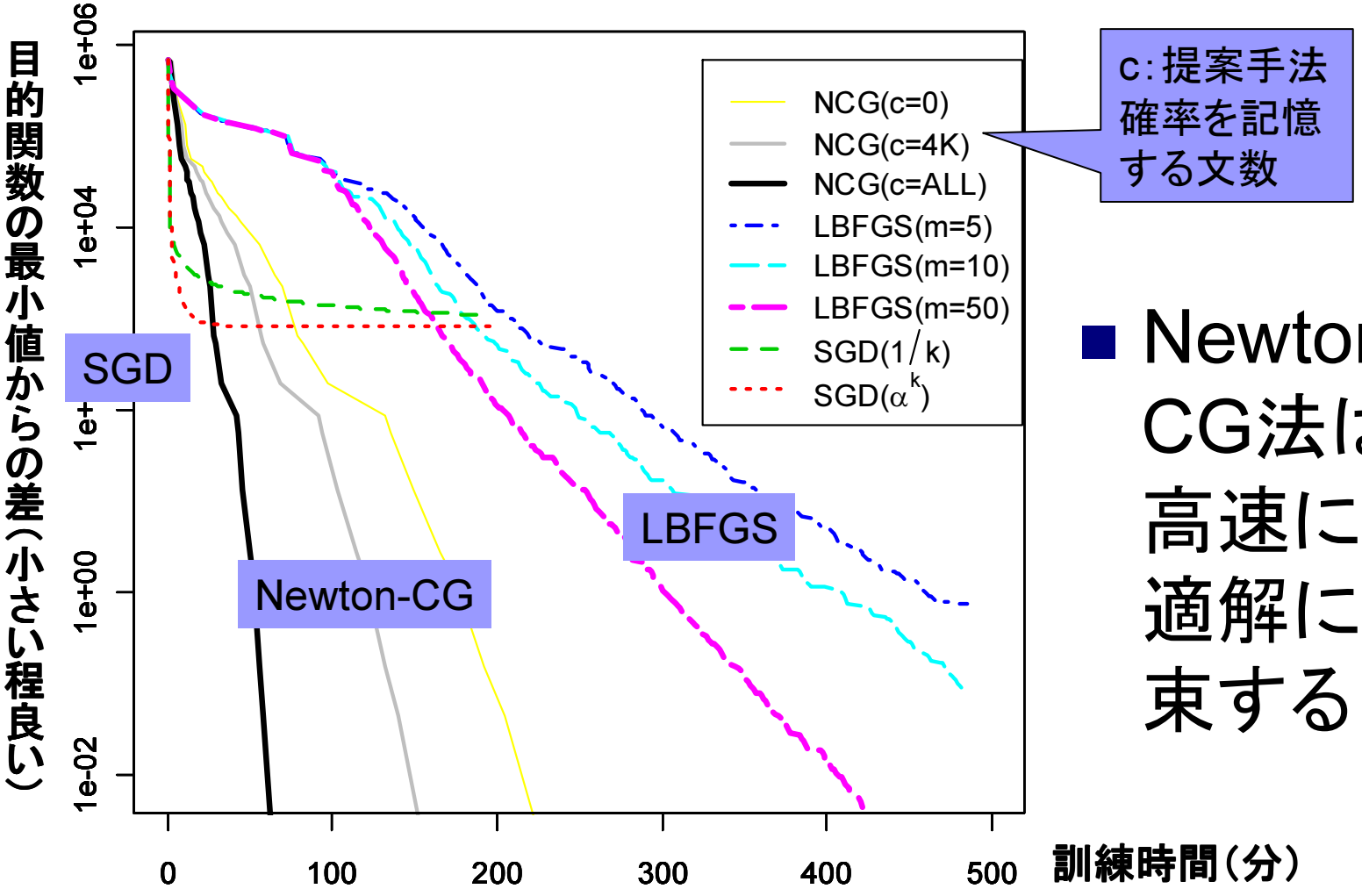
# 基本句チャンキング(CoNLL2000)と固有表現抽出(CoNLL2002)の詳細

- 各タスクの統計情報

| タスク | 言語 | 訓練 | 開発 | テスト | \|X\| | \|Y\| |
|---|---|---|---|---|---|---|
| **チャンキング** | 英語 | 8,936 | N/A | 2,012 | 338,539 | 23 |
| **固有表現抽出** | スペイン語 | 8,322 | 1,914 | 1,516 | 99,135 | 9 |

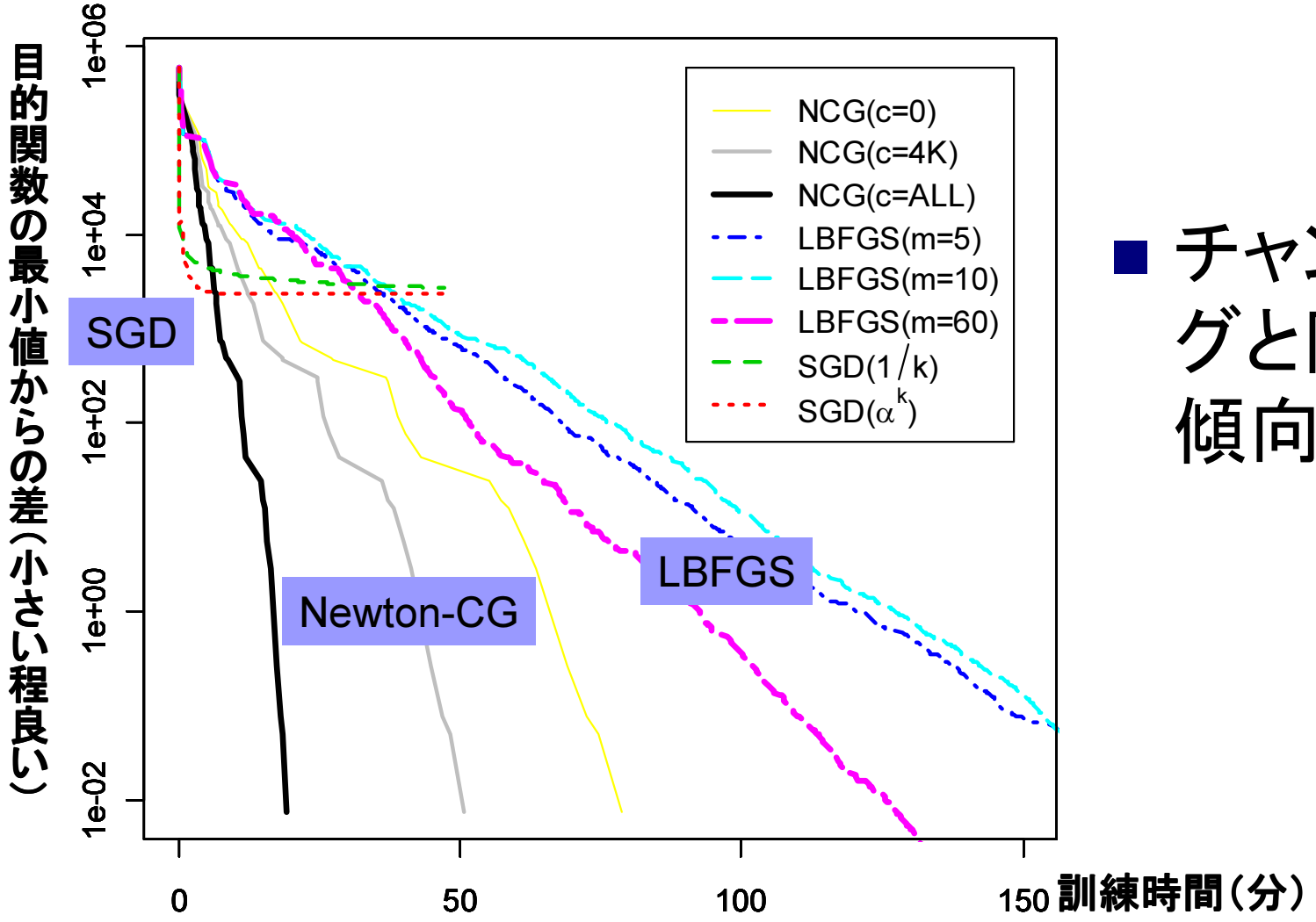
- 使用した観測素性
  - チャンキング: CRF++ソフトウェアに含まれている素性
  - 固有表現抽出: Altun 2003のS3素性

# 目的関数の収束時間の比較: 基本句チャンキング(CoNLL2000 訓練8,936文)




- Newton-CG法は高速に最適解に収束する

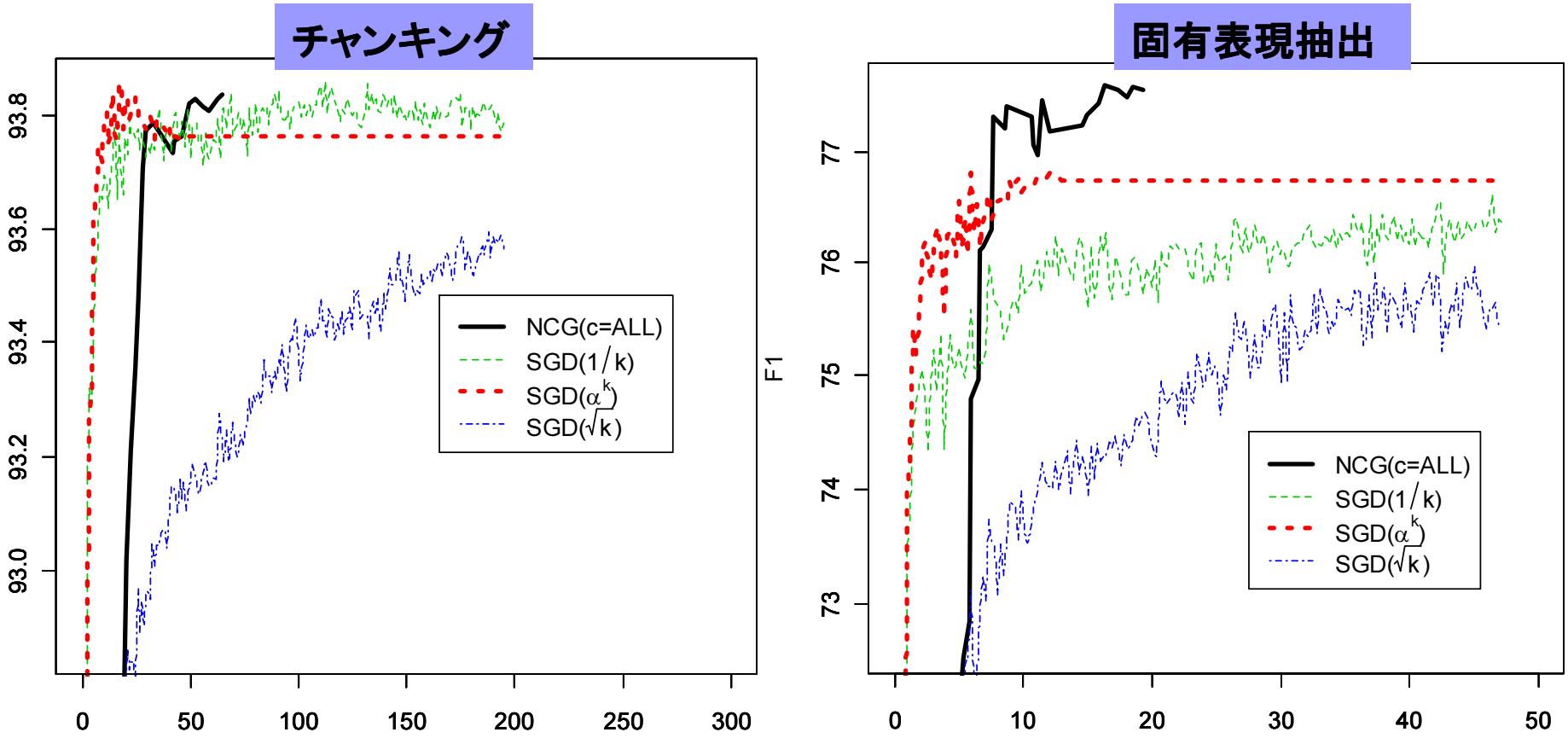
# 目的関数の収束時間の比較: 固有表現抽出(CoNLL2002 訓練8,322)




- チャンキングと同様の傾向

# 確率的勾配法(SGD)の更新率の影響

- $\sqrt{k}$ に比例した更新率ではさらに収束が遅い



# 確率的勾配法(SGD)の実装

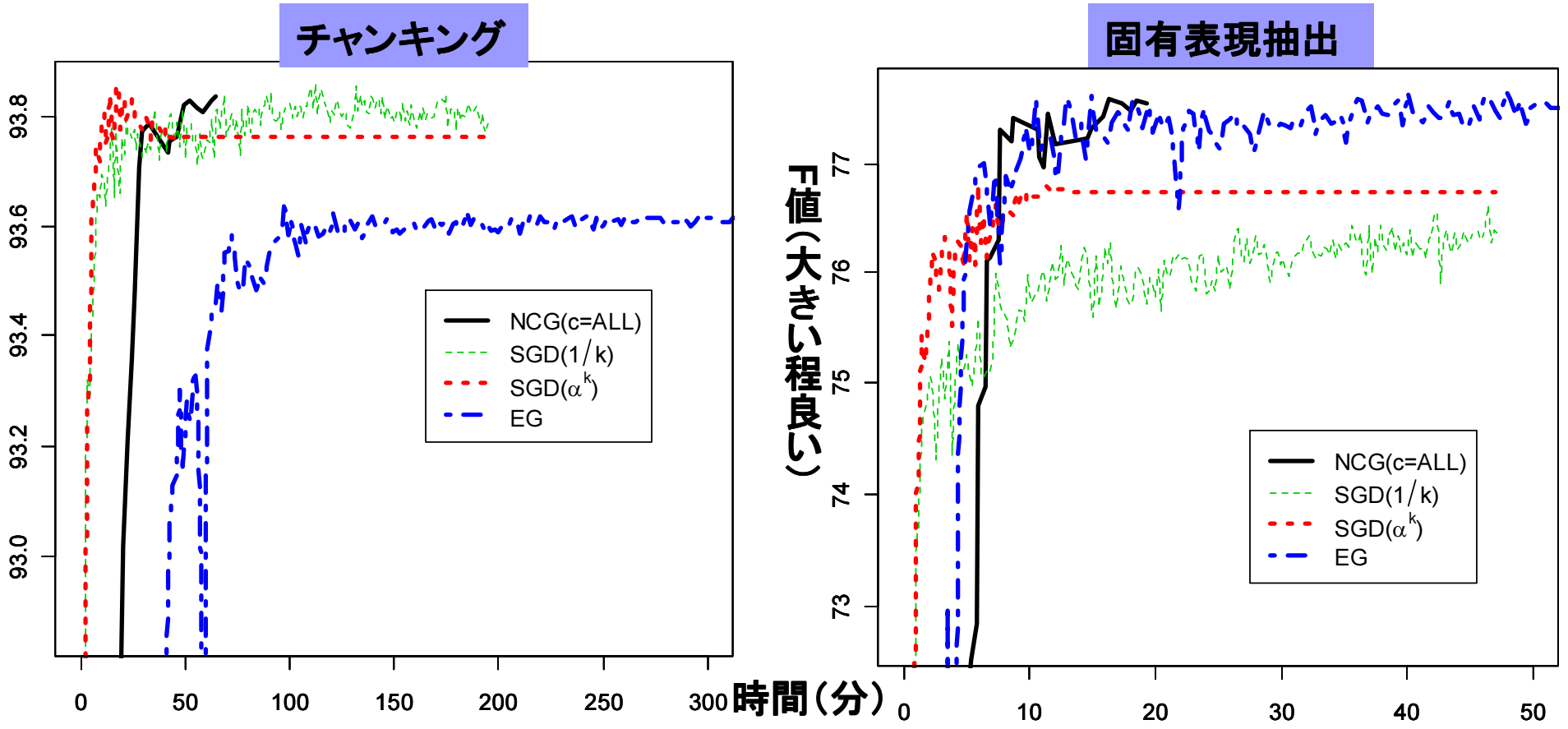
- 更新率の初期値$\eta_0$選択手法
  - 候補$\{0.5, 0.1, 0.05, 0.01\}$ を$\eta_0$としてそれぞれ500文で学習
  - 別の500文で目的関数を計測し、最も目的関数が下がった候補を$\eta_0$として選択
  - 本処理は実験での訓練時間に含む
- パラメータのスパース更新のための実装
  - スカラーとベクトルの組で表現: $\boldsymbol{\theta} = a\mathbf{v}$
  - 密ベクトルの更新を避ける:

aだけ更新

$$\boldsymbol{\theta}_{k+1} = \left(1 - \frac{\eta_k}{n\sigma^2}\right)\boldsymbol{\theta}_k - \eta_k \mathbf{g}(x, y)$$

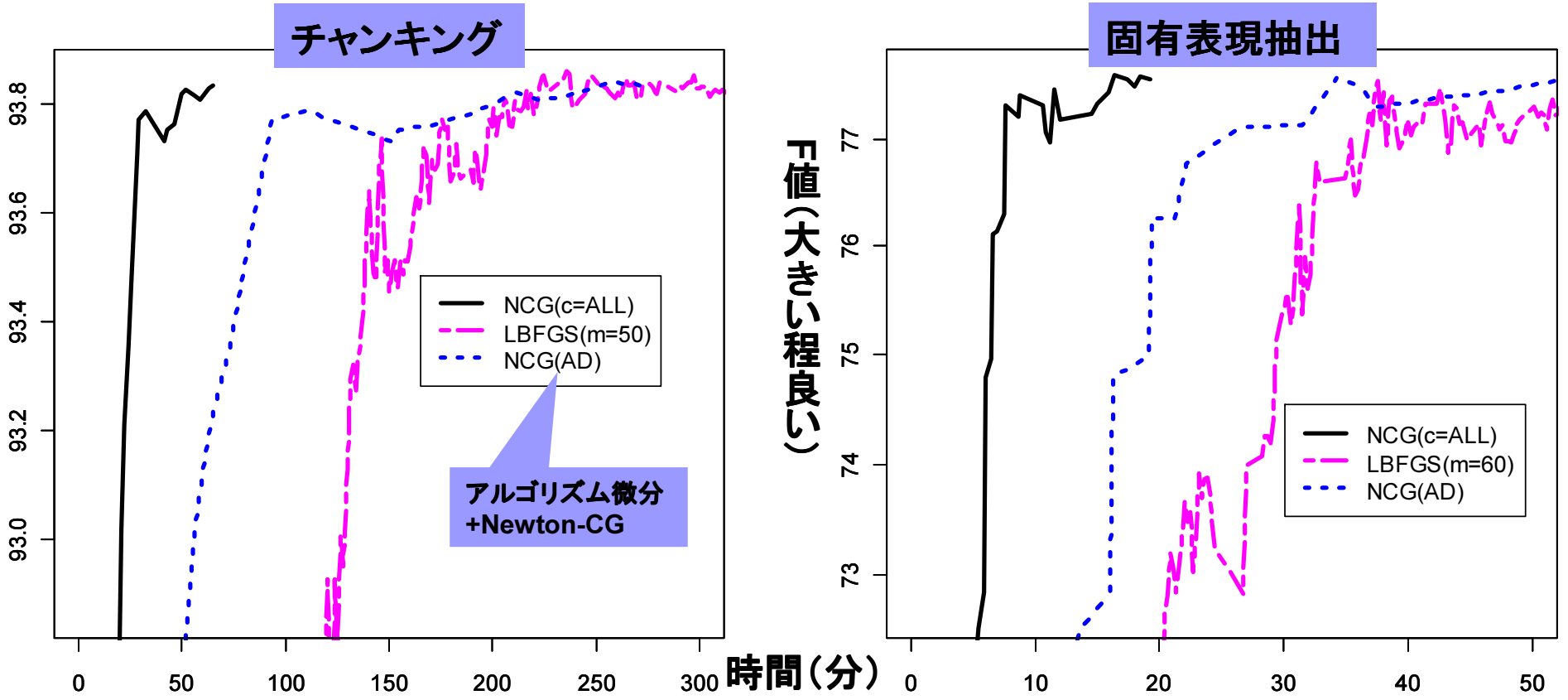
# Exponentiated Gradient (Collins et al. 2008)との比較

- 双対問題の最大化をオンラインで行うEGも、他のオンライン学習手法と同様にタスク依存の性能を示した



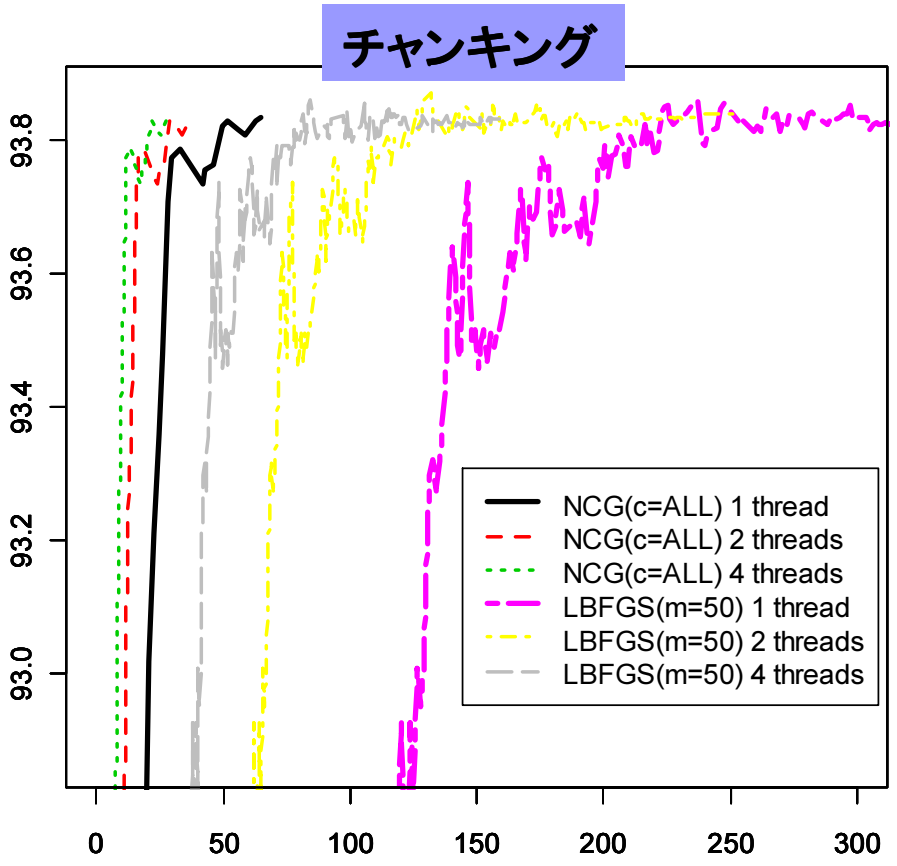
# アルゴリズム微分によるヘシアン・ベクトル積計算法(Pearlmutter 1994)と提案法との比較

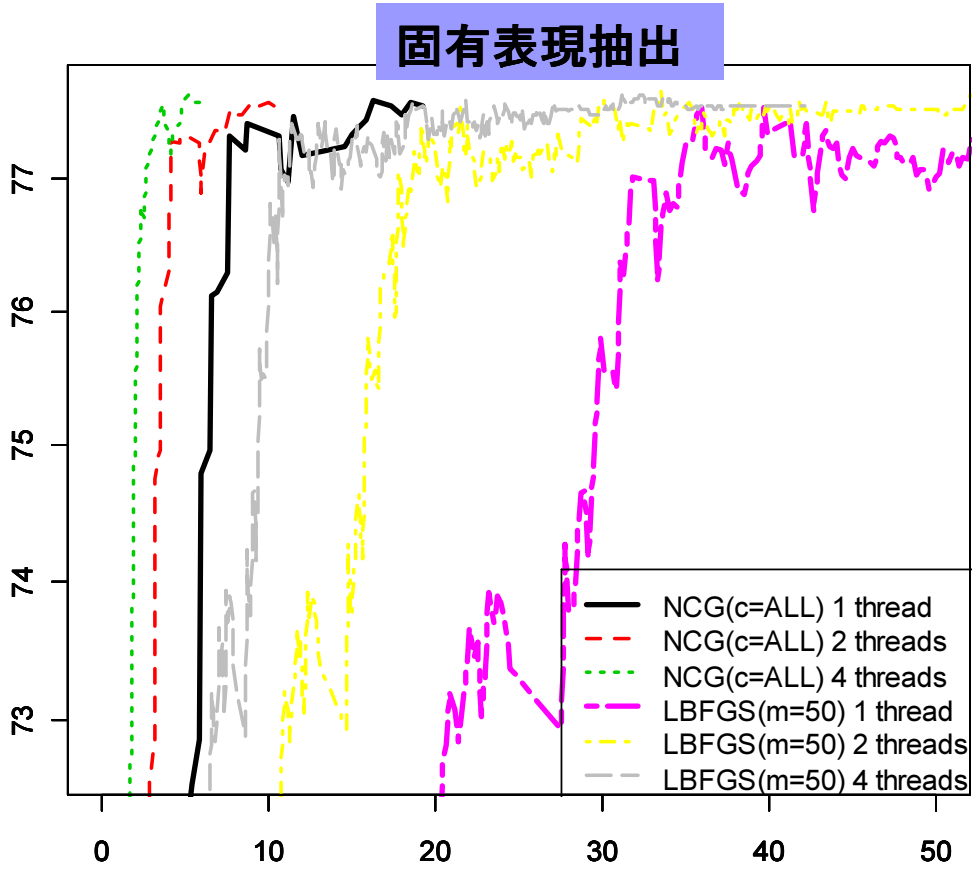
- アルゴリズム微分(AD)を使用したNewton-CG法は周辺確率の再利用ができないため、提案法より収束が遅い



# 勾配とヘシアン・ベクトル積の並列計算の効果（同一PC上 4コアCPU）

- 4スレッドのLBFGSよりもシングルスレッドのNewton-CGが速い



# Limited-memory BFGS (LBFGS)

- ニュートン・ステップ: $-\mathbf{H}^{-1}\mathbf{g}$
- BFGS formula

$$\mathbf{H}_{k+1}^{-1} = \left(\mathbf{I} - \rho\mathbf{u}\mathbf{v}'\right)\mathbf{H}_k^{-1}\left(\mathbf{I} - \rho\mathbf{u}\mathbf{v}'\right) + \rho\mathbf{u}\mathbf{u}'$$

$$\mathbf{u} = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k, \mathbf{v} = \mathbf{g}_{k+1} - \mathbf{g}_k, \rho = \frac{1}{\mathbf{v}'\mathbf{u}}$$

- LBFGS: 直近m更新の変化量を表すベクトル $\mathbf{s}_k, \mathbf{v}_k$ を用いて $\mathbf{H}_k^{-1}$ を近似

# 議論

- バッチ学習とオンライン学習の組み合わせ
  - 初期にオンライン学習を適用(早い立ち上がり)、後半にバッチ学習を適用(高精度の最適化)

  →オンラインからバッチへの切り替えのタイミングが難しく、最適なタイミングは問題依存