International Journal of Wavelets, Multiresolution
and Information Processing
Vol. 10, No. 6 (2012) 1250059 (16 pages)

DOI: 10.1142/S0219691312500592

ACCELERATING GENERALIZED ITERATIVE SCALING BASED ON STAGGERED AITKEN METHOD FOR ON-LINE CONDITIONAL RANDOM FIELDS[a]

HEE-DEOK YANG
School of Computer Engineering, Chosun University
Seosuk-dong, Dong-ku, Gwangju 501-759, Korea
heedeok_yang@chosun.ac.kr

HEUNG-IL SUK
Department of Computer Science and Engineering
Korea University, Anam-dong,
Seongbuk-ku, Seoul 136-713, Korea
hisuk@image.korea.ac.kr

SEONG-WHAN LEE[b]
Department of Brain and Cognitive Engineering
Korea University, Anam-dong, Seongbuk-ku
Seoul 136-713, Korea
swlee@image.korea.ac.kr

Received 23 February 2012
Revised 7 March 2012
Published 8 December 2012

In this paper, a convergent method based on Generalized Iterative Scaling (GIS) with staggered Aitken acceleration is proposed to estimate the parameters for an on-line Conditional Random Field (CRF). The staggered Aitken acceleration method, which alternates between the acceleration and non-acceleration steps, ensures computational simplicity when analyzing incomplete data. The proposed method has the following advantages: (1) It can approximate parameters close to the empirical optimum in a single pass through the training examples; (2) It can reduce the computing time by approximating the Jacobian matrix of the mapping function and estimating the relation between the Jacobian and Hessian in order to replace the inverse of the objective function's Hessian matrix. We show the convergence of the penalized GIS based on the staggered Aitken acceleration method, compare its speed of convergence with those

[a]A preliminary version of this paper has been presented at the International Conference on Machine Learning and Cybernetics 2010, Qingdao, China in July 2010.
[b]Corresponding author.

of other stochastic optimization methods, and illustrate experimental results with two public datasets.

Keywords: On-line conditional random field; Aitken acceleration; sequence labeling.

AMS Subject Classification: 22E46, 53C35, 57S20

1. Introduction

The task of estimating parameters in statistical modeling, i.e. Neural Networks (NNs), Support Vector Machines (SVMs), CRFs, etc. has received a great deal of interest from the machine learning community. It is important to reduce the convergence time and to convert off-line algorithms to on-line ones without compromising the stability of models.[30]

In many real-world applications, a system's training examples continuously change over time, i.e. the system needs to incorporate new examples into the model. Traditional batch learning methods are not appropriate in such circumstances, whereas on-line learning methods make it possible to learn a model incrementally with new examples. In other words, on-line learning methods are the most suitable for dynamic environments. As a result, over the past several decades, batch learning algorithms have been successfully converted to on-line ones. For example, the Winnow algorithm, Learn++ and Learn++.MT are on-line learning algorithms based on NNs.[11,14,18]

Recently there has been of great interest in CRFs with its success on various applications, e.g., text processing,[20] bioinformatics,[23] and computer vision.[32] The conventional batch CRF has usually used GIS, Conjugate Gradient (CG), and Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS)[12] for estimating the parameters of the model. For converting a conventional batch CRF to an on-line one, several methods have been successfully applied.[8,22,27]

In general, the stochastic method used in NNs has been applied to conventional batch CRFs in order to convert them to on-line ones. One type is the Stochastic Gradient Descent (SGD), which has been most successfully applied to CRFs. The learning algorithms for batch CRFs are generally based on second-order information requiring computation of the inversion of the Hessian.[8,27] In order to reduce the computation time of the inversion of the Hessian in on-line learning algorithms for CRFs, the Hessian-vector product,[17,27] Quasi-Newton SGD (SGD-QN),[1] Stochastic Meta-Descent (SMD)[27] and Componentwise Triple Jump method for Penalized generalized Iterative Scaling (CTJPIS)[8] were applied.

Hsu *et al.*[8] proposed an efficient on- and off-line algorithm for CRF learning by approximating the Jacobian of the mapping function's matrix with CTJPIS. Vijayakumar *et al.*[27] applied an SMD with Hessian-vector products to estimate the parameters of a CRF. Bordes *et al.*[1] developed a variant of the second-order SGD, which iterates as fast as the first-order SGD but needs less iteration. Schraudolph *et al.*[22] developed a stochastic variant of the L-BFGS method, which is suitable for

on-line optimization of convex functions for CRF learning. Crammer and Singer[4] proposed the Margin Infused Relaxed Algorithm (MIRA) which is suitable for CRF on-line learning.

In general, on-line learning algorithms satisfy certain requirements[1,4,8,22,27]: (1) On-line learning algorithms should converge to the empirical minimum with small training examples. (2) The per-iteration time complexity of the on-line learning algorithm should be $O(n)$ or less, where n is the dimension of parameters.

In this paper, we focus on penalized GIS based on staggered Aitken acceleration[7,8] and parameter updating rules used in Periodic Step-size Adaption (PSA)[8] for estimating CRF parameters in an incremental way.

For estimating parameters with small training examples, the SGD algorithm is applied. The global objective function of the SGD can be approximated as the sum of the object function calculated from independently drawn small batches. As mentioned in Refs. 3 and 8, an SGD can approximate parameters close to the empirical optimum by splitting the parameter update into independently scheduled small batches.

In order to address the per-iteration time complexity of on-line learning algorithms, Aitken-based acceleration is applied. It can reduce the computing time by approximating the Jacobian matrix of the mapping function and estimating the relation between the Jacobian and the Hessian in order to replace the inverse of the objective function's Hessian matrix. In this paper, the staggered Aitken acceleration method, which has been widely used to speed up the convergence of learning algorithms, is applied.

To further improve the performance, the learning rate of the SGD is automatically determined with a step-lengthening method.[7] An SGD with a small learning rate may converge to the empirical optimum slowly, whereas one with a larger learning rate can converge fast. However, this does not always guarantee an empirical optimum. It is difficult to select the proper learning rate. The staggered Aitken acceleration method based on the step-lengthening method adaptively selects the learning rate per iteration, as with SMD.[7]

The rest of this paper is organized as follows: Section 2 reviews related work on stochastic optimization methods for on-line CRF learning. Section 3 further reviews the basic CRF method. Section 4 provides details of the staggered Aitken-based penalized GIS methods for on-line CRF learning. Section 5 presents the experimental results and discusses their implications. Section 6 concludes this paper.

2. Related Work

The representative on-line learning methods for CRFs are further reviewed in this section. We focus on stochastic optimization methods. Refer to Refs. 8, 24, 26 and 29 and Refs. 13, 15, 22 and 25, respectively, for a detailed review on on-line learning methods and for a review on the stochastic optimization methods.

2.1. *Stochastic optimization methods*

An optimization problem is to minimize an objective function

$$\boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D}), \tag{2.1}$$

where $\boldsymbol{\theta}$ is the set of parameters of the model and $\boldsymbol{D}$ is the set of training examples.

In order to estimate $\boldsymbol{\theta}$ that minimizes an objective function, Eq. (2.1) can be solved by taking a partial derivative with respect to the parameters as described in Refs. 2, 8, 24, 26, 27 and 29

$$\frac{\partial \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D})}{\partial \boldsymbol{\theta}} = \nabla \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D}) = 0. \tag{2.2}$$

There are numerous methods that can solve Eq. (2.2) as described in Refs. 13 and 15. One of them is a Gradient Descent (GD) algorithm, which takes the gradient for the entire set of training examples and has generally been used in off-line learning. In the GD algorithm, the parameter set is updated as follows:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta^t \nabla \boldsymbol{L}(\boldsymbol{\theta}^t, \boldsymbol{D}), \quad t \geq 0, \tag{2.3}$$

where η^t is the learning rate (step size) in updating the parameter estimates at time t.

In order to achieve the empirical optimum in off-line learning, Eq. (2.3) can be solved with Newton's method, etc. as

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \boldsymbol{H}^{-1}(\boldsymbol{\theta}^t) \nabla \boldsymbol{L}(\boldsymbol{\theta}^t, \boldsymbol{D}), \tag{2.4}$$

where $\boldsymbol{H}^{-1}$ is the inverse of the objective function's Hessian matrix and is defined as $\boldsymbol{H}(\boldsymbol{\theta}) = \frac{\delta^2}{\delta \boldsymbol{\theta} \delta \boldsymbol{\theta}} \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D})$.

As shown in Eq. (2.4), in order to estimate the parameter set with off-line learning algorithms which use second-order information, we need to compute the inverse of the objective function's Hessian matrix and this is computationally intensive. From Eqs. (2.3) and (2.4), the learning rate η converges to $\boldsymbol{H}^{-1}$ and is generally calculated as in Ref. 8,

$$\eta^t = \frac{\boldsymbol{H}^{-1}(\boldsymbol{\theta})}{t+1}, \quad t \geq 0. \tag{2.5}$$

An SGD is an on-line version of GD. It considers the gradient of a subset $\boldsymbol{B} \subseteq \boldsymbol{D}$ at each iteration. Many algorithms for on-line optimization of convex functions are based on SGD or variants thereof.

An SGD-QN is a variant of SGD which uses second-order information and splits the parameter update into independently scheduled components. The SGD-QN iterates as fast as a first-order SGD but can achieve a similar performance with less iterations.[1] As shown in Eq. (2.4), the second-order SGD uses the inverse of the objective function's Hessian matrix, which is time-consuming. In order to overcome

this problem, a diagonal scaling matrix such as Ref. 1 is used

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta^t \boldsymbol{A} \nabla \boldsymbol{L}(\boldsymbol{\theta}^t, \boldsymbol{B}^t), \tag{2.6}$$

where $\boldsymbol{A}$ is a diagonal scaling matrix estimated on-the-fly.[1]

An SMD is a variant of SGD based on estimating the learning rate separately. Unlike the methods mentioned above, in SDM, the learning rate, $\boldsymbol{\eta}$, is a vector value. Therefore, Eq. (2.6) can be written as for SMD

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \boldsymbol{\eta}^t \cdot \nabla \boldsymbol{L}(\boldsymbol{\theta}^t, \boldsymbol{B}^t), \tag{2.7}$$

where $\cdot$ denotes Hadamard (componentwise) multiplication.

In SMD, $\boldsymbol{\eta}^t$ is generally calculated as in Refs. 27 and 28

$$\boldsymbol{\eta}^{t+1} = \boldsymbol{\eta}^t \cdot \max\left[\frac{1}{2}, 1 - \mu \nabla \boldsymbol{L}(\boldsymbol{\theta}^{t+1}, \boldsymbol{B}^t) \cdot \boldsymbol{v}^{t+1}\right], \tag{2.8}$$

where μ is a scalar tuning parameter and the auxiliary vector $\boldsymbol{v}$ is computed as Refs. 27 and 28

$$\boldsymbol{v}^{t+1} = \alpha \boldsymbol{v}^t - \boldsymbol{\eta}^t \cdot (\nabla \boldsymbol{L}(\boldsymbol{\theta}^t, \boldsymbol{B}^t) + \alpha \boldsymbol{H}^t \boldsymbol{v}^t), \tag{2.9}$$

where α is a time scale constant. In Eq. (2.9), the Hessian can be eliminated by computing $H^t v^t$ with a Hessian vector product utilizing the differential method as described in Refs. 17 and 28.

A PSA is a variant of the second-order SGD by estimating the learning rate separately.[8] In order to approximate the Hessian, the Jacobian matrix of the mapping function is approximated and the relation between the Jacobian and the Hessian is explored.[8]

3. Conditional Random Field

3.1. *CRF framework*

In a CRF, the probability of label sequence $\boldsymbol{y}$ given an observation sequence $\boldsymbol{x}$ is estimated form the normalized product of potential functions. The product of potential functions are represented as in Ref. 10

$$\exp\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle, \tag{3.1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product and $\boldsymbol{\theta}$ is the set of weights of each function $\phi(\cdot)$ which is described as

$$\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i,j \in \mathcal{E}} \phi_{ij}(\boldsymbol{x}, y_i, y_j) + \sum_{i \in \mathcal{N}} \phi_i(\boldsymbol{x}, y_i), \tag{3.2}$$

where $\mathcal{E}$ is the set of edges, $\mathcal{N}$ is the set of nodes, $\phi_{ij}(\boldsymbol{x}, y_i, y_j)$ indicates whether a feature value is observed between two nodes or not, and $\phi_i(\boldsymbol{x}, y_i)$ indicates whether a feature value is observed at a particular node or not.

From Eq. (3.2), the probability of label sequence $\boldsymbol{y}$ given an observation sequence $\boldsymbol{x}$ is calculated by

$$p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{\exp\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}, \boldsymbol{y})\rangle}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})}, \tag{3.3}$$

where $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a normalization factor

$$Z_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}, \boldsymbol{y})\rangle. \tag{3.4}$$

3.2. *CRF parameter learning*

The CRF parameter learning is based on the principle of maximum entropy. Maximum likelihood training selects parameters that maximize the log-likelihood of the training examples. The log-likelihood for a CRF is defined as

$$\begin{aligned} \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D}) &= \sum_{k} \log p_{\boldsymbol{\theta}}(\boldsymbol{y_k} \mid \boldsymbol{x_k}) \\ &= \sum_{k} (\langle \boldsymbol{\theta}, \phi(\boldsymbol{x_k}, \boldsymbol{y_k})\rangle - \log Z_{\boldsymbol{\theta}}(\boldsymbol{x_k})), \end{aligned} \tag{3.5}$$

where $\boldsymbol{x_k}$ is a kth observation sequence in the training set and $\boldsymbol{y_k}$ is the corresponding label sequence.

To avoid overfitting and perform conventional computation, many previous researches assume that the parameter $\boldsymbol{\theta}$ in CRF follows a Gaussian distribution with a fixed variance σ. Therefore, from Eq. (3.5), the penalized log-likelihood function $\boldsymbol{L}$ can be rewritten as

$$\boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D}) = \frac{\|\boldsymbol{\theta} - \mu\|^2}{2\sigma^2} - \sum_{k} (\langle \boldsymbol{\theta}, \phi(\boldsymbol{x_k}, \boldsymbol{y_k})\rangle - \log Z_{\boldsymbol{\theta}}(\boldsymbol{x_k})). \tag{3.6}$$

The gradient of $\boldsymbol{L}$ for the direction of $\boldsymbol{\theta}_i$ is defined as in Refs. 8 and 10

$$\nabla_i \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{D}) = E\phi_i - \tilde{E}\phi_i + \frac{\boldsymbol{\theta}_i - \mu}{\sigma^2}, \tag{3.7}$$

where $\tilde{E}\phi_i$ and $E\phi_i$ are the empirical and model expectations of ϕ_i, respectively, and μ is usually set to zero.

The weight $\boldsymbol{\theta}_i$ of the ith feature function with GIS is updated as in Refs. 8 and 10

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_i, \tag{3.8}$$

where $\delta\boldsymbol{\theta}_i$ is calculated as in Refs. 8 and 10

$$\delta\boldsymbol{\theta}_i = \frac{1}{S} \log \frac{\tilde{E}\phi_i}{E\phi_i + \dfrac{\boldsymbol{\theta}_i}{\sigma^2}}, \tag{3.9}$$

where $S = \max_k \sum_i \phi_i(\boldsymbol{y_k}, \boldsymbol{x_k})$ is the maximum number of feature occurrences in an observation sequence.

4. Staggered Aitken Method for Accelerating GIS in On-line CRF

In an SGD for an on-line CRF, an approximated objective function is used to compute the gradient of $\boldsymbol{L}$ for the direction. From Eqs. (3.7), (3.8) and (3.9), the penalized GIS mappings can be defined by solving $\nabla_i \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{B}) = 0$[8,10] as

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \frac{1}{S} \log \frac{\tilde{E}\phi_i}{E\phi_i + \dfrac{\boldsymbol{\theta}_i}{\sigma^2}}. \tag{4.1}$$

As shown in Eqs. (2.5) and (3.6), the SGD for on-line learning involves two factors: (1) Selecting the proper learning rate; (2) calculating the Gradient. The first problem can be solved using the staggered Aitken acceleration method. The Aitken acceleration method has been applied to linearly convergent algorithms such as a fixed point algorithm. The GIS can be considered as a fixed point algorithm. Therefore, the GIS algorithm for the SGD can be sped up with the Aitken acceleration method.[7] The second problem can be solved with Eqs. (3.8), (3.9) and (4.1).

Let $\boldsymbol{\theta}$ be a vector in space $\mathbb{R}^d$ and F be a mapping function, $F : \mathbb{R}^d \to \mathbb{R}^d$. The fixed point iteration mapping can solve the equation as[5,7,8]

$$\boldsymbol{\theta}^{t+1} = F(\boldsymbol{\theta}^t), \quad t \geq 0. \tag{4.2}$$

Let $\boldsymbol{\theta}^* = \boldsymbol{\theta}^\infty$ be the optimum value. If F is differentiable then $\boldsymbol{\theta}^*$ is a fixed point of F and a neighborhood of $\boldsymbol{\theta}^*$ can be extracted.[7,8] From Eq. (4.2) and the following Eq. (4.3)[7]

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^* + \boldsymbol{J}^t(\boldsymbol{\theta}^t - \boldsymbol{\theta}^*), \tag{4.3}$$

we can obtain that

$$\begin{aligned} \Delta^{t+1} &= \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \\ &\approx \boldsymbol{J}^t \Delta^t \\ &= \boldsymbol{J}^t(\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}), \end{aligned} \tag{4.4}$$

where $\boldsymbol{J}$ is the gradient of F at θ.

In a linear problem, $\boldsymbol{J}$ is independent of $\boldsymbol{\theta}$ and is generally continuous and approaches $\boldsymbol{J}(\boldsymbol{\theta}^*)$ as t increases.[7]

By ignoring changes in $\boldsymbol{J}$ over successive iterations, we have

$$\begin{aligned} \boldsymbol{\theta}^* &= \boldsymbol{\theta}^t + \sum_{j=1}^{\infty} \Delta^{t+j} \\ &\approx \boldsymbol{\theta}^t + \sum_{j=1}^{\infty} \boldsymbol{J}^j \Delta^t \\ &= \boldsymbol{\theta}^t + \boldsymbol{J}(\boldsymbol{I} - \boldsymbol{J})^{-1} \Delta^t \\ &= \boldsymbol{\theta}^{t-1} + (\boldsymbol{I} - \boldsymbol{J})^{-1} \Delta^t \\ &= \boldsymbol{\theta}^{t-1} + (\boldsymbol{I} - \boldsymbol{J})^{-1}(\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}). \end{aligned} \tag{4.5}$$

Refer to Ref. 7 for the detailed derivation of Eq. (4.5). The eigenvalue of $\boldsymbol{J}$ at $\boldsymbol{\theta}^*$ determines the convergence of the sequence $\boldsymbol{\theta}$.[5] In other words, the convergence rate of Eq. (4.5) is governed by the eigenvalues of $\boldsymbol{J}(\boldsymbol{\theta}^*)$.[5] Let the eigendecomposition of $\boldsymbol{J}$ be $\boldsymbol{J} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^{-1}$ with eigenvalues Λ_j for $j = 1, \ldots, p$ (with $\lambda = \Lambda_1$), where p denotes a dimension. Thus, λ can be estimated by $\lambda^t = \Delta^t/\Delta^{t-1}$.[5,7]

Equation (4.5) is reduced by replacing $(\boldsymbol{I} - \boldsymbol{J})^{-1}$ with the estimated eigenvalues of $\boldsymbol{J}$[7]

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{t-1} + r^t\Delta^t, \tag{4.6}$$

where r^t is a step length factor usually set by $r^t = \frac{1}{1-\lambda^t}$ and Δ^t can be estimated by the EM or other linearly convergent method.

Hämmerlin and Hoffmann[6] suggested to set $r^t = 2/(2-\lambda_{\max}-\lambda_{\min})$, where $\lambda_{\max}$ is the largest eigenvalue and $\lambda_{\min}$ is the smallest eigenvalue. However, sometimes the algorithm oscillates without converging.[7]

Hsu *et al.*[8] set $r^t = (1-\gamma^t)^{-1}$ which is a variant of the Steffensen method.[16] They named this method "Triple Jump" extrapolation and γ^t is calculated as[6,8]

$$\gamma^t = \frac{\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t}{\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}}. \tag{4.7}$$

Hesterberg[7] provided three estimates of γ as

$$\gamma^{(1)} = \frac{\Delta^t \cdot \Delta^{t-1}}{|\Delta^{t-1}|^2}, \tag{4.8}$$

$$\gamma^{(2)} = \frac{|\Delta^t|}{|\Delta^{t-1}|}, \tag{4.9}$$

$$\gamma^{(3)} = \frac{|\Delta^t|^2}{\Delta^t \cdot \Delta^{t-1}}. \tag{4.10}$$

For certain constants c_j and b_j, Eqs. (4.8), (4.9) and (4.10) can be rewritten as[7]

$$\gamma^{(1)} = \frac{\sum\limits_j b_j^2\Lambda_j}{\sum\limits_j b_j^2}, \tag{4.11}$$

$$\gamma^{(2)} = \frac{\sum\limits_j b_j\Lambda_j}{\sum\limits_j b_j}, \tag{4.12}$$

$$\gamma^{(3)} = \frac{\sum\limits_j b_j^2\Lambda_j^2}{\sum\limits_j b_j^2\Lambda_j}, \tag{4.13}$$

where

$$\Delta^{t-1} = \sum_j c_j \Lambda_j^{t-1} \boldsymbol{Q}_j = \sum_j b_j^2, \tag{4.14}$$

$$\Delta^{t-1} \cdot \Delta^t = \sum b_j^2 \Lambda_j, \tag{4.15}$$

$$\Delta^t \cdot \Delta^t = \sum b_j^2 \Lambda_j^2. \tag{4.16}$$

As shown in Eqs. (4.11), (4.12) and (4.13), $\gamma^{(1)} \leq \gamma^{(2)} \leq \gamma^{(3)}$.[7] Hesterberg proved that the three parameters are equivalent if Δ^t and Δ^{t-1} are parallel. In most cases, the third estimate $\gamma^{(3)}$ is the most accurate.[7]

Hesterberg[7] proposed the staggered Aitken acceleration method, which alternates between the acceleration and non-acceleration steps. Subsequently, they can reduce the computation time. Hsu *et al.*[8] also used this method, viz. PSA. As described by Eqs. (4.7) and (4.9), Hsu *et al.*'s method is similar to $\gamma^{(2)}$ of Hesterberg. In this paper, we use Eq. (4.10) instead of Eq. (4.9). We also exploit the PSA concept to adaptively determine the step size in learning.

Finally, in order to update the learning rate $\boldsymbol{\eta}$ adaptively, we apply the update constraint rule in Ref. 8. The learning rate is updated as in Refs. 8 and 25

$$\boldsymbol{\eta}_i^{t+1} = \boldsymbol{v}_i \cdot \boldsymbol{\eta}_i^t, \tag{4.17}$$

where v_i is a future discount factor calculated as[8]

$$\boldsymbol{v}_i = \frac{m + u_i}{m + \kappa + n}, \tag{4.18}$$

where $m = \frac{\alpha+\beta}{\alpha-\beta}\kappa$ and $n = \frac{2(1-\alpha)}{\alpha-\beta}\kappa$, where κ is a positive constant governing the upper bound of $|\gamma_s|$, and u_i is calculated as[8]

$$u_i = \text{sgn}(\gamma_s)\min(|\gamma_s|, \kappa), \tag{4.19}$$

where γ_s is calculated with Eq (4.10).

5. Experimental Results and Analysis

5.1. *Experimental environments*

In our experiments, we considered two public datasets, i.e. CoNLL-2000 Base NP chunking[21] and BioNLP/NLPBA-2004.[9]

SGD, PSA, and the proposed method for on-line learning and L-BFGS for off-line learning were implemented and compared on two datasets. We used the softwares of Refs. 33–35 for SGD, PSA and L-BFGS, respectively. The proposed method was implemented by extending the software of Ref. 34.

To measure the accuracy of the proposed method, the F-score was used[8,27]

$$\text{F-score} = \frac{2pr}{p+r} \times 100, \tag{5.1}$$

where the precision p is the fraction of output instances which match the reference instances and recall r is the fraction of reference instances returned.[8,27]

The parameters used in our experiments were determined as follows:

- SGD: $\eta^0 = 0.1$, $\lambda = 0.1$.[33]
- PSA: $\kappa = 0.9$, $(\alpha, \beta) = (0.9999, 0.99)$, $\eta^0 = 0.1$.[34]
- L-BFGS: The original parameter in the software of CRF++.[35]
- The proposed method: $\kappa = 0.9$, $(\alpha, \beta) = (0.9999, 0.99)$, $\eta^0 = 0.1$.

All the experiments described in this paper were performed on an Intel Core 2 Duo CPU 2.66 GHz with 4 GB memory.

5.2. *Convergence performance comparison with training set*

We first evaluated the F-scores obtained on the training set with four methods, SGD, PSA, L-BGFS, and the proposed method, as a function of the number of passes through the training set on a logarithmic scale.[8,27] As shown in Fig. 1, the F-score achieved by the proposed method is as good as SGD, PSA and L-BFGS. We only plotted curves for the first 50 passes, because all of the on-line methods

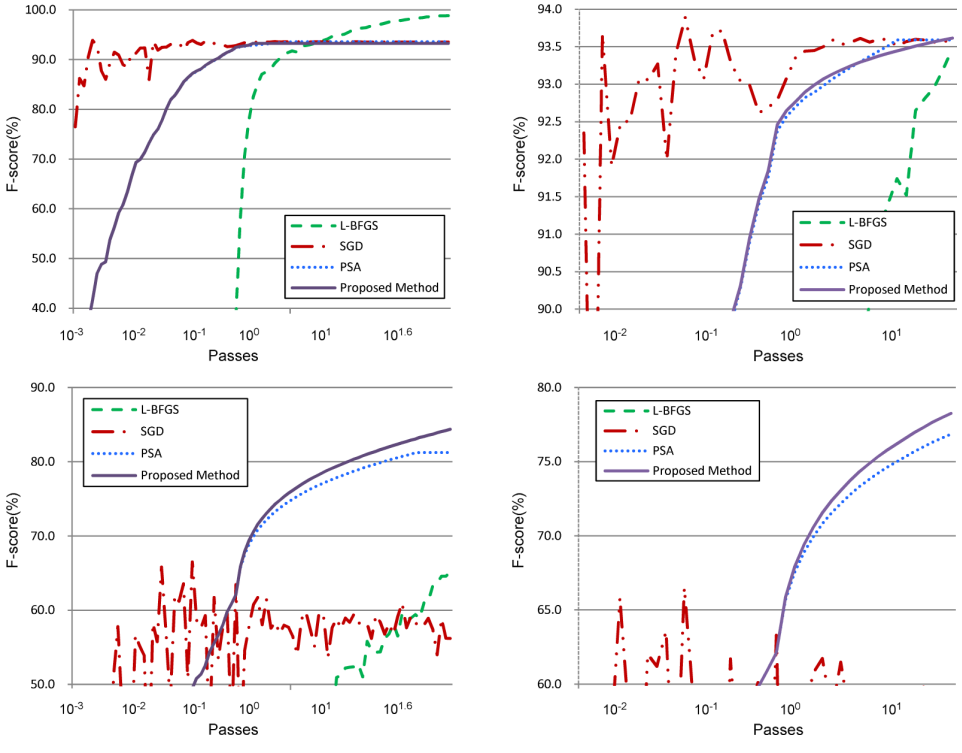

Fig. 1. The left curves show the F-scores obtained on the training set with respect to the number of passes through the training data on a logarithmic scale, and the right curves show an enlargement of the middle area of the left curves. Top row: CoNLL-2000 Base NP; Bottom row: BioNLP/NLPBA-2004.

converged to an empirical optimum around that time, although the off-line method, L-BFGS, required more passes to converge to the empirical optimum.

In CoNLL-2000 Base NP, the proposed method gets within 93.2% in one pass and within 90.0% in 0.35 passes. In other words, after the 3,100th training sequence, the F-score has nearly converged to the empirical optimum. The F-score of the plain SGD undergoes extreme oscillations over time. Meanwhile, the proposed method converges to the empirical optimum in a single pass. After four passes, the change of the proposed method's F-score is negligible. As shown by the enlarged curves, the convergence speed of the proposed method is slightly faster than PSA, since the learning rate of the former is calculated with Eq. (4.10) and that of the latter is calculated with Eq. (4.9).

Figure 2 shows the objective values obtained on the training set with respect to the number of passes through the training data. Again, we only plotted curves for the first 50 passes. The on-line methods, SGD, PSA, and the proposed method converged faster than the off-line method, L-BFGS, did. As shown by the curves, the proposed method converges to the empirical optimum in a single pass. After four passes, the change of the objective value of the proposed method is again negligible. L-BFGS and SGD took more than 50 and 20 passes to converge to an optimal value, respectively.

Table 1 shows the average execution time in seconds per pass after processing the training examples for 50 passes with various γ. As shown in the results, the

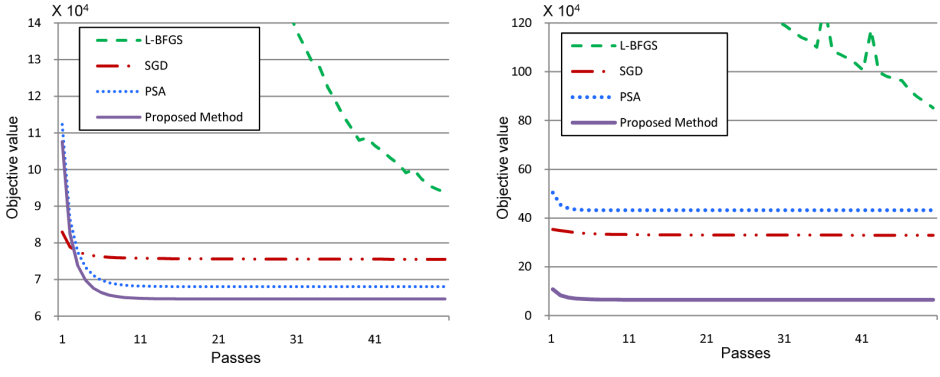

Fig. 2. The curves show the objective values obtained on the training set with respect to the number of passes through the training data. Left column: CoNLL-2000 Base NP; Right column: BioNLP/NLPBA-2004.

Table 1. Comparing average execution time in seconds per pass after processing the training examples for 50 passes with various γ.

Dataset	Number of features	$\gamma^{(1)}$	$\gamma^{(2)}$ (PSA)	$\gamma^{(3)}$ (Proposed method)
CoNLL-2000 Base NP	7448606	1953.7	1945.8	1954.5
BioNLP/NLPBA-2004	4583414	1930.4	1852.7	1927.3

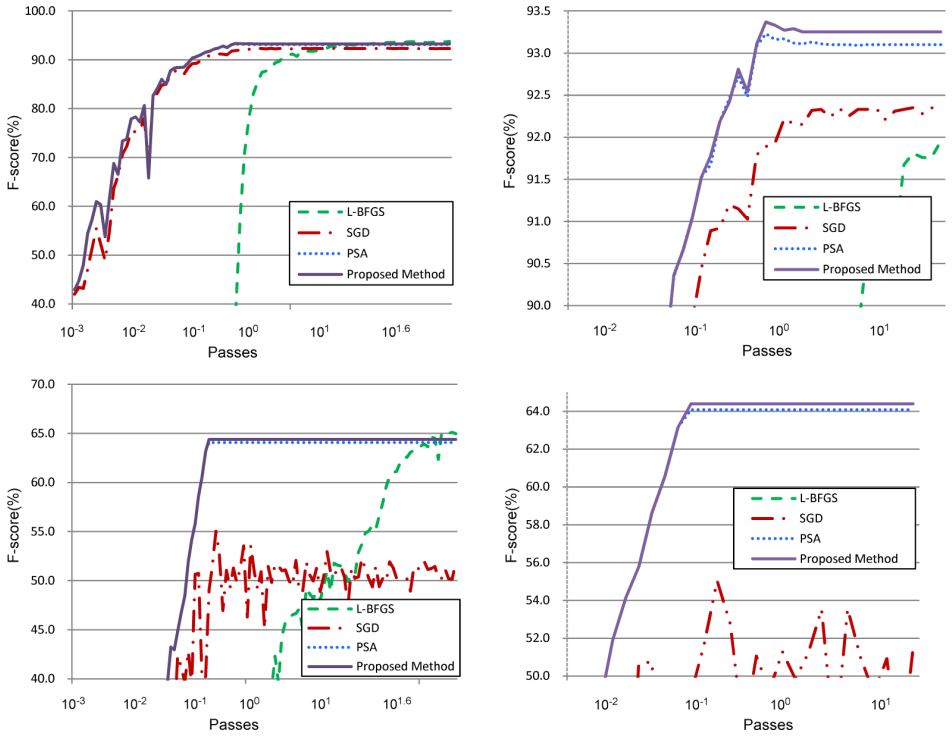

Fig. 3. The left curves show the F-scores obtained on the test set with respect to the number of passes through the training data on a logarithmic scale, and the right curves show an enlargement of the middle area of the left curves. Top row: CoNLL-2000 Base NP; Bottom row: BioNLP/NLPBA-2004.

execution times of $\gamma^{(1)}$ and $\lambda^{(3)}$ was similar. The execution speed of $\gamma^{(2)}$ is slightly faster than those of $\gamma^{(1)}$ and $\gamma^{(3)}$. However, the execution time of $\gamma^{(3)}$ is low enough to process the training examples on-line.

5.3. *Accuracy performance comparison with test set*

Figure 3 shows the F-scores obtained on the test set with four methods, SGD, PSA, L-BGFS, and the proposed method, as a function of the number of passes through the training set on a logarithmic scale. The proposed method converges faster than SGD and PSA. In CoNLL-2000 Base NP, the proposed method gets within 93.1% in one pass. As shown by the enlarged curves, the convergence speed of the proposed method is slightly faster than PSA. In CoNLL-2000 Base NP, the proposed method gets within 90.0% in 0.1 passes. In other words, after the 890th training sequence, the F-score has nearly converged to the empirical optimum. In BioNLP/NLPBA-2004, the proposed method gets within 64.1% in one pass. The curve of the plain SGD fluctuates significantly. The F-score of a single pass of the proposed method is better than the other methods.

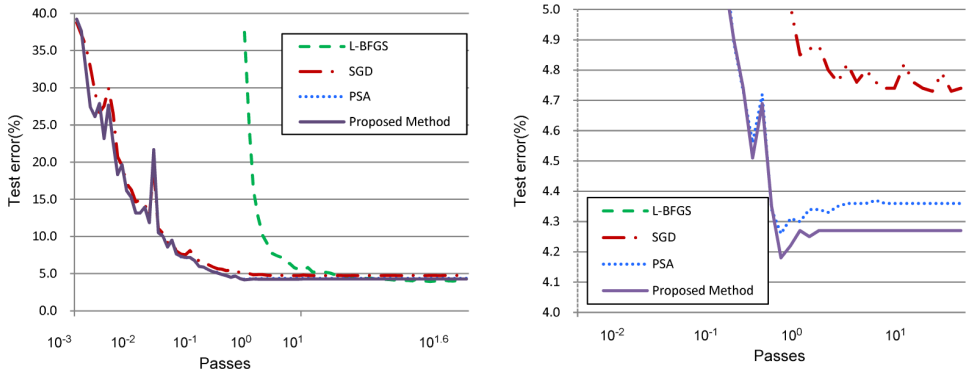

Fig. 4. The left curve shows the test error obtained on the test set of CoNLL-2000 Base NP with respect to the number of passes through the training set on a logarithmic scale, and the right curve shows an enlargement of the middle area of the left curve.

Figure 4 shows the test error obtained on the test set of CoNLL-2000 Base NP, as a function of the number of passes through the training set on a logarithmic scale. The on-line methods, SGD, PSA, and the proposed method, converged faster than the off-line method, L-BFGS. Especially, the proposed method outperformed all of the other ones. Thus, it needs less passes than the other methods to converge to the minimum. After one pass, the test error of the proposed method had converged to 4.27%.

It is difficult to select a batch size that is efficient for all tasks.[8,27] As mentioned in Ref. 27, typically, small batches of examples, $5 \leq b \leq 20$, are efficient. Hsu *et al.*[8] mentioned that $b = \frac{0.5|D|}{1{,}000}$ is usually sufficient. We performed experiments with various batch sizes using the proposed method. Figure 5 shows the F-scores obtained on the test set with various batch sizes, 1, 5, 10, 50, as a function of the number of

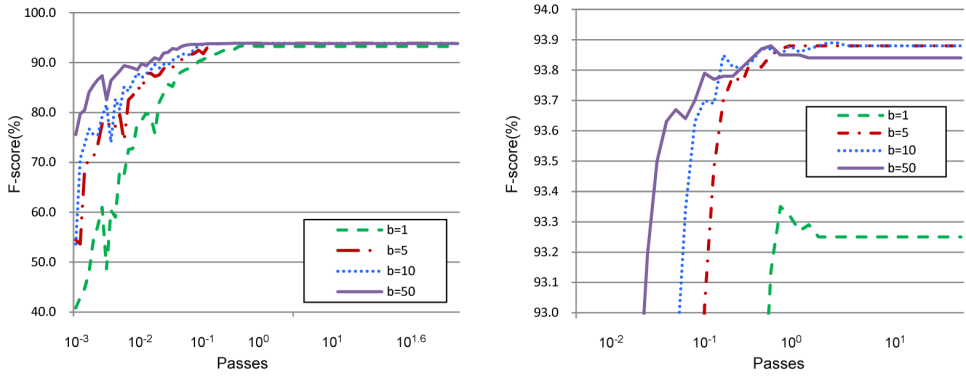

Fig. 5. The left curve shows the F-scores obtained on the test set of CoNLL-2000 Base NP with respect to the number of passes through the training data on a logarithmic scale with various batch sizes, and the right curve shows an enlargement of the middle area of the left curve.

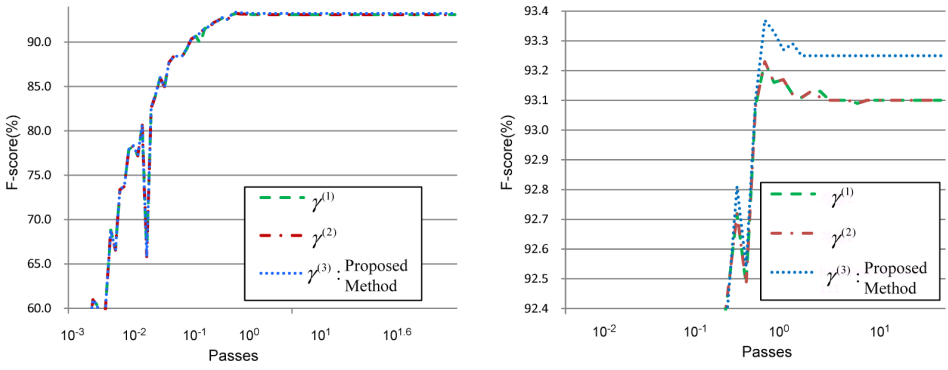

Fig. 6. The left curve shows the F-scores obtained on the test set of CoNLL-2000 Base NP with respect to the number of passes through the training data on a logarithmic scale with various λ and the right curve shows an enlargement of the middle area of the left curve.

passes through the training set on a logarithmic scale. As the batch size increases, the curve rises rapidly. In other words, as the batch size increases, the F-score of the initial portion converges at a good value. As shown in the enlargements of the final portion, the F-scores converge between 93.8 and 93.9%. In our experiments, when the batch sizes are 5 and 10, the F-scores converge at the largest value, 93.9%.

It is difficult to select the proper step size in stochastic optimization methods, as mentioned in Sec. 1. In order to estimate the step size adaptively, we can approximate γ with Eqs. (4.8), (4.9) and (4.10) as shown in Sec. 4. Figure 6 shows the F-score obtained on the test set with respect to the number of passes through the training data with various γ. Enlargement of the middle portion of the result reveals some differences in asymptotic convergence between Eqs. (4.8), (4.9) and (4.10). The F-score measured by Eqs. (4.8) and (4.10) is similar. On the other hand, the F-score with the $\gamma^{(3)}$ of Eq. (4.10) converges the fastest, since $\gamma^{(3)}$ is more stable than $\gamma^{(1)}$ and $\gamma^{(2)}$, as mentioned in Sec. 4.

6. Conclusions and Further Research

In this paper, a convergent method based on penalized GIS with staggered Aitken acceleration was proposed to efficiently estimate the parameters for an on-line CRF. The Aitken acceleration method ensures computational simplicity when analyzing incomplete data. The proposed method can approximate parameters close to the empirical optimum in a single pass through the training examples, and it can reduce the computing time by approximating the Jacobian matrix of the mapping function and estimating the relation between the Jacobian and Hessian in order to replace the inverse of the objective function's Hessian matrix. The staggered Aitken acceleration method alternates between the acceleration and non-acceleration steps. Thus, it can converge to the empirical optimum without oscillations as time passes.

Experiments on two public datasets demonstrated that the proposed method can converge close to the empirical optimum parameters for an on-line CRF in a single pass.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Ministry of Education, Science, and Technology (MEST), under Grant 2012-005741. This work was also supported by the NRF Grant funded by the Korean government (MEST) (No. 2010-0015362).

References

1. A. Bordes, L. Bottou and P. Gallinari, SGD-QN: Careful quasi-Newton stochastic gradient descent, *J. Mach. Learning Research* **10** (2009) 1737–1754.
2. L. Bottou and O. Bousquet, The tradeoffs of large scale learning, *Proc. Advances in Neural Information Processing Systems* (Vancouver, Canada, 2008), pp. 161–168.
3. L. Bottou and Y. L. Cun, On-line learning for very large datasets: Research articles, *Appl. Stoch. Models in Business and Industry* **21**(2) (2005) 137–151.
4. K. Crammer and Y. Singer, Ultraconservative online algorithms for multiclass problems, *J. Mach. Learning Research* **3** (2001) 951–991.
5. A. P. Dempster, N. M. Laird and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Stat. Soc. Series B (Methodological)* **19** (1977) 1–38.
6. G. Hämmerlin and K. Hoffmann, *Numerical Mathematics* (Springer-Verlag, 1991).
7. T. Hesterberg, Staggered Aitken acceleration for EM, *Proc. Conf. on Statistical Computing Section of the American Statistical Association* (Minneapolis, USA, 2005), pp. 2101–2110.
8. C. Hsu, H. Huang, Y. Chang and Y. Lee, Periodic step-size adaptation in second-order gradient descent for single-pass on-line structured learning, *Mach. Learning* **77** (2009) 195–224.
9. J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi and N. Collier, Introduction to the bio-entity recognition task at JNLPBA, *Proc. Int. Joint Workshop on Natural Language Processing in Biomedicine and its Applications* (Geneva, Switzerland, 2004), pp. 70–75.
10. J. Lafferty, A. McCallum and F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence, *Proc. Int. Conf. on Machine Learning* (Williamstown, USA, 2001), pp. 282–289.
11. N. Littlestone, Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Mach. Learning* **2** (1988) 285–318.
12. D. C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Programming* **45**(3) (1989) 503–528.
13. G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions* (Wiley-Interscience, 1997).
14. M. Muhlbaier, A. Topalis and R. Polikar, Learn++.MT: A new approach to incremental learning, *Proc. Workshop on Multiple Classifiers Systems* (Cagliari, Italy, 2004), pp. 52–61.
15. J. Nocedal and S. Wright, *Numerical Optimization* (Springer-Verlag, 1999).
16. J. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables* (Academic Press, 1970).

17. B. Pearlmutter, Fast exact multiplication by the Hessian, *Neural Comput.* **6**(1) (1994) 147–160.
18. R. Polikar, J. Byorick, S. Krause and M. Moreton, Learn++: A classifier independent incremental learning algorithm for supervised neural networks, *Proc. Int. Joint Conf. on Neural Networks (2002)*, pp. 1742–1747.
19. A. Quattoni, S. Wang, L.-P. Morency, M. Collins and T. Darrell, Hidden conditional random fields, *IEEE Trans. on Pattern Anal. Mach. Intell.* **29**(10) (2007) 1848–1852.
20. D. Roth and W. Yih, Integer linear programming inference for conditional random fields, *Proc. Int. Conf. on Machine Learning* (Bonn, Germany, 2005), pp. 737–747.
21. E. Sang and S. Buchholz, Introduction to the CoNLL-2000 shared task: Chunking, *Proc. Int. Conf. on Computational Natural Language Learning* (Lisbon, Portugal, 2000), pp. 127–132.
22. N. Schraudolph, J. Yu and S. Gunter, A stochastic quasi-Newton method for online convex optimization, *Proc. Int. Conf. on Artificial Intelligence and Statistics* (San Juan, Puerto Rico, 2007), pp. 436–443.
23. B. Settles, Abner: An open source tool for automatically tagging genes, proteins, and other entity names in text, *Bioinformatics* **21**(14) (2005) 3191–3192.
24. A. Shapiro and Y. Wardi, Convergence analysis of gradient descent stochastic algorithms, *J. Optim. Theory Appl.* **91** (1996) 439–454.
25. J. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control* (Wiley-Interscience, 2003).
26. S. Vijayakumar, A. Souza and S. Schaal, Incremental online learning in high dimensions, *Neural Comput.* **17** (2005) 2602–2634.
27. S. Vishwanathan, N. Schraudolph, M. Schmidt and K. Murphy, Accelerated training of conditional random fields with stochastic gradient methods, *Proc. Int. Conf. on Machine Learning* (Pittsburgh, USA, 2006), pp. 969–976.
28. S. Vishwanathan, N. Schraudolph and A. Smola, Adaptation in reproducing kernel hilbert space, *J. Mach. Learning Research* **7** (2006) 1107–1133.
29. W. Wiegerinck and T. Heskes, How dependencies between successive examples affect on-line learning, *Neural Comput.* **8** (1996) 1743–1765.
30. H.-D. Yang and S.-W. Lee, Accelerating generalized iterative scaling using componentwise extrapolations for on-line conditional random fields, *Proc. Int. Conf. on Machine Learning and Cybernetics* (Qingdao, China, 2010), pp. 3169–3173.
31. H.-D. Yang and S.-W. Lee, Simultaneous spotting of signs and finger spellings based on hierarchical conditional random fields and boostmap embeddings, *Pattern Recogn.* **43**(8) (2010) 2858–2870.
32. H.-D. Yang, S. Sclaroff and S.-W. Lee, Sign language spotting with a threshold model based on conditional random fields, *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(7) (2009) 1264–1277.
33. http://leon.bottou.org/projects/sgd/.
34. http://aiia.iis.sinica.edu.tw/.
35. http://chasen.org/∼taku/software/crf++/.