A fast implementation of semi-Markov conditional random fields

La The Vinh, Sungyoung Lee, and Young-Koo Lee

Dept. of Computer Engineering,
Kyung Hee University
{vinhlt,sylee,yklee}@oslab.khu.ac.kr

Abstract. Recently, Conditional Random Fields (CRF) model has been used and proved to be a good model for sequential modeling. It, however, lacks the capability of duration modeling. Therefore, some researchers introduced semi Markov Conditional Random Fields (semi-CRF) to take into account the duration distribution and showed some improvements. Nevertheless, the training algorithms for semi-CRF require quite a high complexity making semi-CRF impractical in some large-scale problems. Therefore, in this work we propose a fast implementation of the training algorithm in order to reduce the complexity required by semi-CRF. Our theoretical analysis as well as experiments' result show a noticeable improvement in computation time, which is about ten times less than that of the original algorithm.

Keywords: Conditional Random Fields, Semi-Markov Model

1 Introduction

Recently, Conditional Random Fields(CRF) was introduced as a discriminative model for sequential data and produced a much better result than the well-known existing generative model so-called Hidden Markov Model [3]. Nonetheless, both conventional HMM and first order linear chain CRF are limited to the Markovian property, which assumes that the current state depends only on the previous state. Because of this assumption, the model is not able to capture the duration distribution as well as the long-range transitions of states. To solve the problem, Sarawagi and Cohen proposed a semi Markov Conditional Random Fields (semi-CRF) model in [5]. However, Sarawagi and Cohen's model increases the complexity of forward/backward as well as gradient estimation algorithms by D times, where D were the maximum duration length. Although there is some other work about semi-CRF [2], [6], none of them shows any improvement in the computation complexity of the model. In [4] the author proposed a method to make semi-CRF scalable. In that work, the author utilized a Naive Bayes classifier to filter out some candidates making the computation much more faster. However, there may be a trade-off since the removed candidates may affect the final accuracy of the model in some applications. Therefore, we are going to overcome the above limitations by developing fast gradient estimation algorithms for semi-CRF while keeping the model's behavior [5] unchanged.

2 Semi-Markov Conditional Random Fields

Hereafter, we assume that the observation and the corresponding label sequence of length T are given in the form

$$X = \{x_1, x_2, ..., x_T\}, \tag{1}$$

$$Y = \{y_1, y_2, ..., y_T\}. \tag{2}$$

In our work, each state of a semi-CRF is defined as

$$s_i = (y, b, e) \; i = 1, 2, ..., P, \tag{3}$$

where P is the length of sequence $S = s_1...s_P$, which is constructed from input labels $Y = (y_1, y_2, ..., y_T)$. y, b, e in turn are label, beginning time, and ending time of state s_i. The beginning and ending time must satisfy the following constraints.

$$s_i.b \leq s_i.e \; i = 1, 2, ..., P, \tag{4}$$

$$s_i.e + 1 = s_{i+1}.b \; i = 1, 2, ..., P - 1, \tag{5}$$

$$s_1.b = 1, \tag{6}$$

$$s_P.e = T. \tag{7}$$

Now, the likelihood of S given X is estimated by

$$P(S|X) = \frac{\prod_{i=1}^{P} \Psi(s_{i-1}, s_i, X)}{Z_X}, \tag{8}$$

$$Z_X = \sum_{S'} \prod_{i=1}^{P'} \Psi(s'_{i-1}, s'_i, X), \tag{9}$$

where $\Psi(s_{i-1}, s_i, X)$ encodes the potential of the transition from s_{i-1} to s_i.In equations (8) and (9), we can consider the product of potential functions Ψ over all transitions of a sequence as the potential of the sequence, then we can rewrite (8) in the form

$$P(S|X) = \frac{Pol(S)}{\sum_{S'} Pol(S')}, \tag{10}$$

where

$$Pol(S) = \prod_{i=1}^{P} \Psi(s_{i-1}, s_i, X) \tag{11}$$

is the potential of the sequence $S = s_1, s_2, ..., s_P$. The potential function can be defined as followings

$$\Psi(s_{i-1}, s_i, X) = \begin{pmatrix} e^{Q^T(s_{i-1}, s_i, X)} \times \\ e^{Q^D(s_{i-1}, s_i, X)} \times \\ e^{Q^O(s_{i-1}, s_i, X)} \end{pmatrix}, \tag{12}$$

where Q^T, Q^D and Q^O are the weighted transition, duration and observation potential functions, respectively. These functions may have different forms in different applications. Nevertheless, we present below examples of the functions which could be applied in some well-known applications, namely Name Entity Recognition (NER), Activity Recognition (AR).

$$Q^T(s_{i-1}, s_i, X) = \sum_{y',y} w^T(y', y)\delta(s_{i-1}.y = y', s_i.y = y), \tag{13}$$

where $w^T(y', y)$ is the weight of transition from y' to y and δ is given by

$$\delta(X) = \begin{cases} 1 \text{ if X is true} \\ 0 \text{ if X is false} \end{cases}. \tag{14}$$

$$Q^D(s_{i-1}, s_i, X) = \sum_{y,d} G^D(y, d)\delta(s_i.y = y, d = s_i.e - s_i.b + 1) \tag{15}$$

$$= \sum_{y,d} w^D(y)\frac{(d - m_y)^2}{2\sigma_y^2}\delta(s_i.y = y, d = s_i.e - s_i.b + 1), \tag{16}$$

where $w^D(y)$ is the duration weight of y. m_y and σ_y are the average and the standard deviation of state y's duration respectively, which can be easily extracted from training data.

$$Q^O(s_{i-1}, s_i, X) = \sum_{y,t_1,t_2} \begin{pmatrix} G_y(y, t_1, t_2)\times \\ \delta(s_i.y = y, s_i.b = t_1, s_i.e = t_2) \end{pmatrix}, \tag{17}$$

where

$$G_y(y, t_1, t_2) = \sum_{t=t_1}^{t_2} \sum_o w^O(y, o)\delta(x_t = o), \tag{18}$$

where $w^O(y, o)$ is the weights of the observation given that input symbol o is observed in state with label y. For our convenience in the presentation of the following equations, we denote $G(y, t_1, t_2) = G_y(y, t_1, t_2) + G^D(y, t_2 - t_1 + 1)$ as a combined potential function.

As we noted above that the potential functions may have different definitions. The above equations are just examples of them, which generally can be applied to a large number of applications including language processing[1], gene prediction [2], activity recognition [7], etc.

2.1 Forward algorithm

Forward algorithm is used to compute the normalization factor Z_X efficiently by using the dynamic programming method.
To compute Z_X in (9) let we denote

$$\alpha(y, t) = \sum_{S^t \in \Gamma_t^y} Pol(S^t), \tag{19}$$

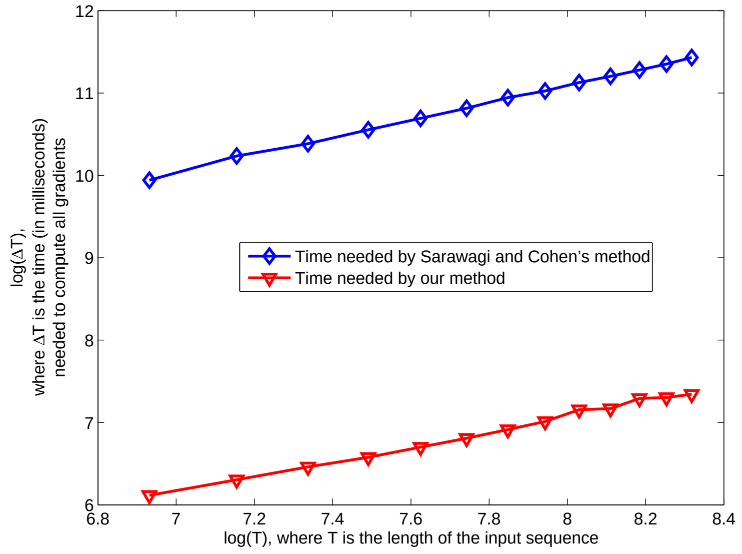

Fig. 1: Average time needed for computing all the gradients. Herein, the number of labels (M) is 4, the maximum duration (D) is 16, the number of input values (or often known as size of codebook) (V) is 128, the length of the input sequence (T) changes from 1024 to 4096 with a step of 256. Therefore, the number of gradients is $M + M^2 + MV = 532$

where $\Gamma_t^y = \{S = s_1, s_2, ..., s_q\}$ is a set of all semi-Markov sequences, which have an original label sequence $(y_1, y_2, ..., y_t)$ with the last label is y. Thus, every $S^t = s_1, s_2, ..., s_q \in \Gamma_t^y$ satisfies $s_q.e = t$, $s_q.y = y$. Our forward algorithm is implemented in the following equations

$$Z_X = \sum_y \alpha(y, T), \tag{20}$$

$$\alpha(y, t) = \sum_{d=1}^{D} \left(\lambda(y, t-d) e^{G(y, t-d+1, t)} \right), \tag{21}$$

$$\lambda(y, t) = \sum_{y'} \alpha(y', t) e^{w^T(y', y)}. \tag{22}$$

2.2 Backward algorithm

Similarly to the forward algorithm, let we denote

$$\beta(y, t) = \sum_{S^{T-t+1} \in \Omega_t^y} Pol(S^{T-t+1}), \tag{23}$$

where $\Omega_t^y = \{S = s_1, s_2, ..., s_q\}$ is a set of all semi-Markov sequences, which have an original label sequence $(y_t, y_{t+1}, ..., y_T)$ with the first label is y. The backward algorithm is described in the below equations

$$Z_X = \sum_y \beta(y, 1), \tag{24}$$

$$\beta(y,t) = \sum_{d=1}^{D} \left(\zeta(y, t+d) e^{G(y,t,t+d-1)} \right), \tag{25}$$

where

$$\zeta(y,t) = \sum_{y'} \beta(y', t) e^{w^T(y,y')}. \tag{26}$$

2.3 Parameter estimation

The goal of parameter estimation is to choose appropriate values for the model weights (w^T, w^D and w^O) so that the likelihood of the observation data $P(S|X)$ is maximized. Take the logarithm form of $P(S|X)$ we have

$$L(S|X) = \sum_{i=1}^{P} \begin{pmatrix} Q^T(s_{i-1}, s_i, X) + \\ Q^D(s_{i-1}, s_i, X) + \\ Q^O(s_{i-1}, s_i, X) \end{pmatrix} - log(Z_X). \tag{27}$$

To find the optimal parameter values w we have to solve$\frac{dL}{dw^*} = 0$. From (27) we know that

$$\frac{dL}{dw^*} = \sum_{i=1}^{P} \frac{dQ^*(s_{i-1}, s_i, X)}{dw^*} - \frac{1}{Z_X} \frac{dZ_X}{dw^*}. \tag{28}$$

Herein we use Q^* and w^* with the superscript to refer to any kind of the potential function and weight (* can be D, T, or O). Computing the first term of the right side is trivial, Z_X is calculated by using forward/backward variables. Therefore, here we mainly focus on evaluating $\frac{dZ_X}{dw^*}$ for different kind of weights. From (9) and (12) we have

$$\frac{dZ_X}{dw^*} = \sum_{S^T} \left(\left(\sum_{i=1}^{P} \frac{dQ^*(s_{i-1}, s_i, X)}{dw^*} \right) \prod_{i=1}^{P} \Psi(s_{i-1}, s_i, X) \right). \tag{29}$$

Gradient of the transition weight Since

$$\frac{dQ^T(s_{i-1}, s_i, X)}{dw^T(y', y)} = \delta(s_{i-1}.y = y', s_i.y = y), \tag{30}$$

we have

$$\frac{dZ_X}{dw^T(y', y)} = \sum_{t=1}^{T} \alpha(y', t) \beta(y, t+1) e^{w^T(y', y)}. \tag{31}$$

Gradient of the duration weight From the definition of the duration potential function, it is clear that

$$\frac{dQ^D(s_i, s_{i-1}, X)}{dw^D(y)} = \sum_{y,d} \delta(s_i.y = y, s_i.e - s_i.b + 1 = d) \frac{(d - m_y)^2}{2\sigma_y^2}. \tag{32}$$

As a result

$$\frac{dZ_X}{dw^D(y)} = \sum_{d=1}^{D} \sum_{t=1}^{T} \frac{(d - m_y)^2}{2\sigma_y^2} \theta(y, t, d), \tag{33}$$

where

$$\theta(y, t, d) = \lambda(y, t-1)\zeta(y, t+d) e^{G(y,t,t+d-1)} \tag{34}$$

can be considered to be the sum of potential values of all sequences $Y = y_1, y_2, ..., y_T$, which have d labels y from time t, or equivalently $y_t = y_{t+1} = ... = y_{t+d-1} = y$.

Gradient of the observation weight From (17) and (18) we have

$$\frac{dQ^O(s_{i-1}, s_i, X)}{dw^O(y, o)} = \sum_{k=s_i.b}^{s_i.e} \delta(s_i.y = y, x_k = o). \tag{35}$$

Combining (35) and the definition of θ in (34) leads to

$$\frac{dZ_X}{dw^O(y, o)} = \sum_{\substack{k, t, d \\ k \in [t, t+d-1]}} \theta(y, t, d) \delta(x_k = o). \tag{36}$$

3 Evaluation

In this section, we show that our proposed algorithm achieves a remarkable improvement in terms of the complexity in comparison with the previous work theocratically as well as practically. In the following experiments, we use C++ programming language with Microsoft Visual Studio IDE to implement the algorithms.

3.1 Complexity Analysis

As it is described in [5], the required complexity for computing each gradient is $O(TM^2D)$, where T, M, D are the length of the input sequence, the number of label values, and the maximum duration of a label, respectively. Because of this, the estimation of gradients for all N model's parameters takes $O(NTM^2D)$.

In our solution, gradients are computed by using (31), (33), and (36). It is obvious that if α, λ, β, ζ, and θ are cached then the maximum time needed is about $O(TD)$. Therefore, for optimizing N parameters, our algorithm needs only $O(NTD)$ to complete calculating all gradients. Nevertheless, we need to take into account the extra time of estimating the cached variables. As it is shown in the forward and backward algorithm, α, λ, β, and ζ can be computed with $O(2TM(M+D))$, meanwhile from (34) we see that θ takes $O(TMD)$. Totally, for caching of these variables, we need a complexity of around $O(2TM^2 + 3TMD)$. Herein, we take a numerical example to compare $O(NTM^2D)$, which is the estimated complexity in [5] and $O(NTD + 2TM^2 + 3TMD)$, our algorithm complexity. Let $T = 1000$, $D = 100$, $M = 8$ and $N = 10000$ then the former is about 64×10^9, the latter is about 10^9.

In addition, Fig. 1 illustrates another comparison of the two complexities with N=532, M=4, D=16 and T changes from 1024 to 4096, time is measured in millisecond. The blue represents the amount of time which is needed by the method proposed by Sarawagi and Cohen in [5], time consumed by our algorithm is marked by the red. Undoubtedly, there is a remarkable improvement in our complexity.

4 Conclusion

In this paper, we have presented a fast implementation for estimating the gradients of semi-CRF in the training phase, increasing the computation performance of the model. Although, semi-CRF is a powerful discriminative model for sequential data. It, however, is limited in use because of the high computation complexity. Therefore, we believe that our contribution will make semi-CRF more practical especially for large-scale applications.

Acknowledgments. This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2009-(C1090-0902-0002)). This work was also sponsored by the IT R&D program of MKE/KEIT, [10032105, Development of Realistic Multiverse Game Engine Technology], and by the Basic Science Research Program funded by National Research Foundation (2009-0076798).

References

1. Andrew, G.: A hybrid markov/semi-markov conditional random field for sequence segmentation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) (2006)
2. Doherty, M.K.: Gene Prediction with Conditional Random Fields. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institue of Technology (2007)
3. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceeding of International Conference on Machine Learning (ICML) (2001)

4. Okanohara, D., Miyao, Y., Tsuruoka, Y., Tsujii, J.: Improving the scalability of semi-markov conditional random fields for named entity recognition. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. pp. 465–472 (2006)
5. Sarawagi, S., Cohen, W.: Semi-markov conditional random fields for information extraction. In: Proceedings of Advances in Neural Information Processing Systems (NIPS) (2004)
6. Truyen, T., Phung, D., Bui, H., Venkatesh, S.: Hierarchical semi-markov conditional random fields for recursive sequential data. In: Proceeding of International Conference on Advances in Neural Information Processing (NIPS) (2008)
7. Vail, D.L., Veloso, M.M., Lafferty, J.D.: Conditional random fields for activity recognition. In: Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (2007)