정규문법 추론을 위한 이산 순환신경망의 개선된 학습방법

정 현 기† · 정 순 호††

요 약

본 연구는 정규문법을 추론하기 위한 DRNN의 학습성능을 향상시키기 위하여 2가지 측면에서 개선점을 제시한다. 첫 번째는 DRNN의 입력인 학습패턴의 측면으로, 학습패턴의 입력순서를 여러 가지로 정돈하여 실험을 통해 학습성능을 분석하고, 가장 향상된 학습성능을 가진 학습패턴의 입력순서를 제시한다. 두 번째는 DRNN 학습알고리즘의 측면으로, 학습이 실패할 때 최대 epoch까지 진행되는 비효율성을 수리적 분석을 통해 짧은 시간에 수렴되게 알고리즘을 수정하고, 실험을 통해 이 학습알고리즘이 학습효과를 향상시킬 수 있음을 보인다.

Improved learning method of Discrete Recurrent Neural Network inferring Regular Grammars

Hyeon-Ki Jung† · Soon-Ho Jung††

ABSTRACT

In this paper, we describe two considerations that improve the learning performance of Discrete Second-Order Recurrent Neural Network(DRNN) inferring regular grammars. One is due to the viewpoint of training patterns which are inputs of DRNN. It is to investigate learning performances according to a variety of input order and decide the best one. The other is due to the viewpoint of learning algorithm. It is to find reason with which learning is failed and modify algorithm in order to reduce total learning time. We suggest the modified learning method and show the improved result of experiments.

1. 서 론

최근에 신경망의 한 모델인 Recurrent Neural Networks(RNNs)을 사용하여 Finite State Automata를 더욱 효율적으로 학습하기 위한 연구가 진행되어왔다[1]. 이들 연구에서의 RNN은 기존의 First-Order RNN(FRNN)에 비해 문법의 추론에서 더욱 성공적인 학습이 가능한 Second-Order RNN(SRNN)을 모델로 사용하게 된다[2]. 이 연구에서는 Giles[3]의 SRNN 모델을 기반으로 Zeng[1]이 제안한 학습알고리즘을 적용한 Discrete SRNN(DRNN) 모델을 이용하여 문법의 학습을 설명하고자 한다. 이 DRNN의 특징은 Pseudo-Gradient Learning Method와 Discrete Function을 사용하여 기존 SRNN[3]의 학습성능과 유사하면서도 Discrete된 값을 통해 불안전한 상태가 발생되지 않고 학습이 이루어지게 된다. 이 DRNN의 학습성능을 향상시키는 방법을 2가지 측면에서 고려하게 되는데, 첫 번째는 DRNN의 학습입력집합에 관한 것으로, 입력패턴을 크기와 모양에 따라 여러 가지 입력순서로 정돈하여 그에 따른 학습효과를 분석, 평가하여 가장

† 준 회 원 : 부경대학교 대학원 전자계산학과
†† 정 회 원 : 부경대학교 컴퓨터멀티미디어공학부 교수
논문접수 : 1999년 3월 30일, 심사완료 : 1999년 11월 3일

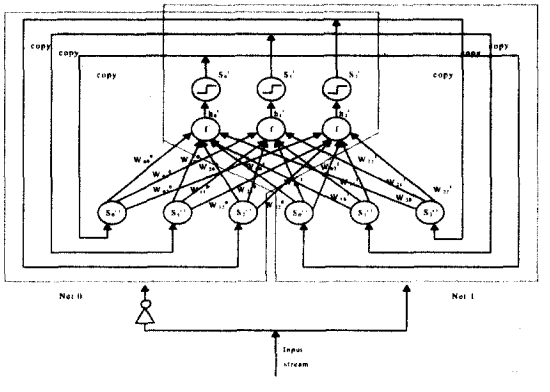
학습성능을 향상시키는 방안을 제안하는 것이다. 두 번째는 DRNN Pseudo-Gradient Learning rule의 학습과정에서 학습이 실패한 원인을 제시하고 수리적 관점에서 그 문제점을 분석하여 개선된 학습알고리즘을 제안할 것이다. 2장에서는 실험을 위해 사용된 DRNN 모델의 구조와 학습방법에 관해서 소개하고, 3장에서는 학습입력패턴의 정돈시 필요한 여러 요소와 방법에 관한 내용과 DRNN 학습알고리즘의 학습과정에서 발생하는 문제점을 수리적 관점에서 분석할 것이다. 그리고 4장에서는 실험과정과 결과를 통해 학습성능을 평가하고 마지막 5장에서 결론을 언급하고자 한다.

2. 배 경

이 연구에 사용되는 모델로 Zeng에 의해 소개된 DRNN을 사용하는데[1], 이 모델의 구조와 Pseudo-Gradient learning method에 관해 간략히 소개하면 다음과 같다.

2.1 DRNN의 구조

상태수가 3인(S_0, S_1, S_2) DRNN은 입력인 0과 1에 의해 제어되는 두 개의 분리된 networks인 net0과 net1로 이루어져 있으며 (그림 1)과 같이 보여진다. 길이가 L인 패턴의 입력은 다음과 같이 정의된다.



(그림 1) 상태수가 3인 Discrete SRNN

$$P_i = x^1 x^2 \cdots x^{L-1} x^L \ (L\text{은 string의 길이}).$$

여기서 weights w_{ij}^n은 -1과 1사이의 값을 가진다.

$$h_i^t = f(\sum_j w_{ij}^{x^t} S_j^{t-1}) \tag{1}$$

$$S_i^t = D(h_i^t) \tag{2}$$

time step t에서 상태 S_i^t는 식 (1)에 의해 식 (2)과 같이 표현되며, 여기서 f는 sigmoid function이며 D는 discrete function으로 식 (3)과 (4)로 표현된다.

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

$$D(x) = \begin{cases} 0.8 & x \geq 0.5 \\ 0.2 & x < 0.5 \end{cases} \tag{4}$$

2.2 DRNN의 학습방법

Zeng[1]의 DRNN은 식 (3)의 $D(x)$ 함수로 인해 analog 값 대신 discrete된 값을 사용하므로 변화량을 구할 수 없다. 대신 그 값의 추정치를 사용하게 되고 이를 Pseudo-gradient라고 한다.

길이가 L인 스트링에서 error E는

$$E = \frac{1}{2}(h_0^L - T)^2 \tag{5}$$

이며 여기서 T는 $target = \begin{cases} 1 & \text{if } \text{“}legal\text{”} \\ 0 & \text{if } \text{“}illegal\text{”} \end{cases}$ 이다.

학습과정에서 weights update의 변화량을 표현하면

$$w_{ij}^n = w_{ij}^n - \alpha \frac{\widehat{\partial E}}{\partial w_{ij}^n} \tag{6}$$

이고 위 (6)식의 $\frac{\widehat{\partial E}}{\partial w_{ij}^n}$는 식 (5)의 weights에 관한 변화량을 구하는 것으로 아래의 식과 같다.

$$\frac{\widehat{\partial E}}{\partial w_{ij}^n} = (h_0^L - T)\frac{\widehat{\partial h_0^L}}{\partial w_{ij}^n} \tag{7}$$

여기서 $\frac{\widehat{\partial h_0^L}}{\partial w_{ij}^n}$은 t=1에서 L까지 변화를 통해 계산되고 그 식은 다음과 같이 표현된다.

$$\frac{\widehat{\partial h_k^t}}{\partial w_{ij}^n} = f' \cdot (\sum_l w_{kl}^{x^t} \frac{\widehat{\partial h_l^{t-1}}}{\partial w_{ij}^n} + \delta_{ki}\delta_{nx^t} S_j^{t-1}) \tag{8}$$

각 time step에서 (8)식에 의해 pseudo-gradient값이 계산되고 time step L일 때 식 (7)과 (6)에 의해 weights update가 되며 이런 일련의 과정을 error 허용범위 안의 값을 가질 때까지 반복하여 학습이 이루어진다.

3. 학습방법의 개선

앞서 기술한 DRNN의 학습성능을 향상시키기 위하여

이번 장에서는 학습패턴의 입력순서를 여러 가지로 정돈하는 방법과, 학습알고리즘의 개선을 위해 기존학습알고리즘의 비효율성을 수리적 측면에서 분석하는 과정을 소개한다.

3.1 학습패턴의 정돈

학습에 사용되는 입력패턴으로는 Tomita grammars가 사용되어진다[4]. 이것은 모두 7개의 문법으로 구성되어 있으며 transition diagram을 통해 학습에 사용되는 positive 예와 negative 예를 표현 할 수 있고[2], 학습집합은 이 예들을 모두 포함하게 된다.

T_1 = 1^*
T_2 = $(10)^*$
T_3 = an even number of consecutive 1's is always followed by an even number of consecutive 0's
T_4 = any string not containing '000'
T_5 = $[(01|10)(01|10)]^*$
T_6 = Mod-3 $[(N_1 - N_0) \bmod 3 = 0]$
T_7 = $0^*1^*0^*1^*$

이전의 연구[1-2][5][7][9-11]에서는 학습집합의 원소인 입력패턴을 random한 순서로 제공하였는데, 이 논문에서는 모든 입력순서의 종류를 조사하여 그에 따른 성능을 비교 분석하여 가장 최선의 방법을 찾고자 한다.

학습패턴의 입력순서의 종류는 크게 두 가지로 분류할 수 있다. 첫 번째는 학습패턴을 정돈된 방법(Ordered)과 그렇지 못한 방법(Random)으로 분류하고 두 번째는 positive 예와 negative 예를 혼합해서 학습하는 방법(Mixed)과 positive 예와 negative 예를 분리해서 학습하는 방법(Separated)으로 나눌 수 있다. 또다시 정돈된 방법(Ordered)은 입력스트링의 길이에 대해서 짧은 것에서 긴 순서로 정돈하는 방법(ascending)과 긴 것에서 짧은 순서로 정돈하는 방법(descending)으로 나눌 수 있다. 이런 학습패턴의 모양과 크기에 따라 학습패턴의 모든 입력순서의 종류는 <표 1>과 같이 6가지 방법으로 나뉘어 진다. 이 6가지 방법 중에는 기존 연구에서 사용하는 Random한 방법도 포함한다. 이 6가지 방법을 [MA], [MD], [MR], [SA], [SD], [SR]이라고 하고, 이 방법들로 Tomita grammars로부터 얻어진 패턴들로 기존의 학습알고리즘을 이용해서 기존 방법을 포함한 새로이 분류된 방법들에 대한 DRNN의 성능을 평가한다. 이 결과에 따라 최선의 결과를 가진 입력순서 방법이 [MA]이 됨을 실험을 통하여 밝힐 것이다.

〈표 1〉 학습패턴의 입력순서

	Ordered		(R)andom
	(A)scending	(D)escending	(R)
(M)ixed	[MA]: P · N을 ascending하게 정렬한 다음 P와 N을 교대로 학습	[MD]: P · N을 descending하게 정렬한 다음 교대로 학습	[MR]: P · N을 random하게 교대로 학습
(S)epa-rated	[SA]: P · N을 ascending하게 정렬한 다음 P와 N중 한쪽을 먼저 학습	[SD]: P · N을 descending하게 정렬한 다음 P와 N중 한쪽을 먼저 학습	[SR]: P · N을 길이에 상관없이 정렬한 다음 P와 N중 한쪽을 먼저 학습
P와 N은 positive 예와 negative 예를 표현			

3.2 개선된 알고리즘

기존 학습알고리즘에서 학습이 수렴되지 않는 경우에 발생하는 비효율성의 원인을 수학적 분석을 통해 설명하고, 학습성능을 향상시킬 수 있는 알고리즘을 제안한다. 먼저 학습이 수렴되지 않는 경우를 설명하여 보면 식 (7)에서 $(h_0^L - T)$항과 $\frac{\partial \widetilde{h_0^L}}{\partial w_{ij}^n}$ 항의 곱으로 계산되어지고 있는데 $|h_0^L - T|$가 에러 허용치보다 큼에도 불구하고 $\frac{\partial \widetilde{h_0^L}}{\partial w_{ij}^n}$ 항의 값이 0이 되어서 식 (7)의 $\frac{\partial E}{\partial w_{ij}^n}$ 값이 0이 되므로 식 (6)에서 weight w_{ij}^n의 변화가 없게 된다.

이러한 과정은 (그림 2)의 (a)와 같이 어떤 가중치가 급격히 증가한 A 지점에서 발생한다. 또한 그와 대응된 (b)의 Error 변화 그래프에서는 B지점과 같다. 따라서 이후의 학습에서는 Error가 발생하여도 weights의 수정이 불가능하게 된다. 이러한 경우의 원인을 살펴보기 위하여 식 (8)의 Pseudo-gradient learning rule을 time step 0에서 L까지 전개하게 되고 그 과정은 아래와 같다.

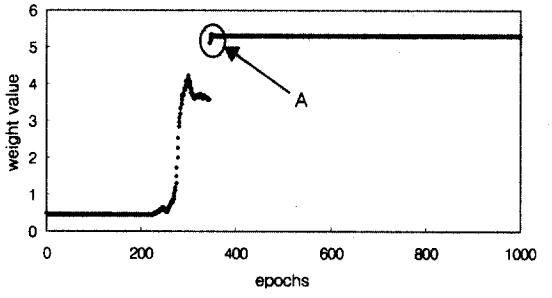
$$\frac{\partial \widetilde{h_0^L}}{\partial w_{ij}^n} = f'_{(L)} \cdot \left(\sum_l w_{0l}^{x^L} \frac{\partial \widetilde{h_l^{L-1}}}{\partial w_{ij}^n} + \delta_{0i}\delta_{nx^L} S_j^{L-1}\right)$$

$$= f'_{(L)} \cdot \left(\sum_l w_{0l}^{x^L} \cdot \left(f'_{(L-1)} \cdot \left(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \cdot \frac{\partial \widetilde{h_{k(1)}^{L-2}}}{\partial w_{ij}^n} + \delta_{li}\delta_{nx^{L-1}} S_j^{L-2}\right)\right) + \delta_{0i}\delta_{nx^L} S_j^{L-1}\right)$$

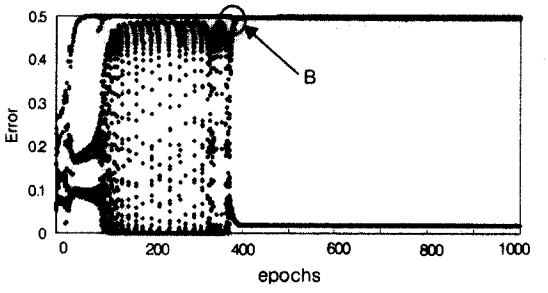
$$= f'_{(L)} \cdot f'_{(L-1)} \cdot \left(\sum_l w_{0l}^{x^L} \left(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \frac{\partial \widetilde{h_{k(1)}^{L-2}}}{\partial w_{ij}^n}\right)\right)$$

$$+ f'_{(L)} \cdot f'_{(L-1)} \cdot \left(\sum_l w_{0l}^{x^L} \cdot \delta_{li}\delta_{nx^{L-1}} S_j^{L-2}\right)$$

$$+ f'_{(L)} \cdot \delta_{0i}\delta_{nx^L} S_j^{L-1}$$



(a) 학습실패인 경우의 w_{ij}^{n}의 변화 그래프



(b) 학습이 실패한 경우의 Error 변화 그래프

(그림 2) 학습이 실패한 경우의 weight와 Error 변화 그래프

$\frac{\widetilde{\partial h_{k(1)}^{L-2}}}{\partial w_{ij}^{n}}$에 대해 다시 전개하여 정리하면

$$
\begin{aligned}
&= f'_{(L)} \cdot f'_{(L-1)} \cdot f_{(L-2)} \cdot \Big(\sum_l w_{0l}^{x^l} \cdot \\
&\quad \Big(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \cdot \Big(\sum_l w_{0l}^{x^l} \cdot \frac{\widetilde{\partial h_{k(2)}^{L-3}}}{\partial w_{ij}^{n}}\Big)\Big)\Big) \\
&+ f'_{(L)} \cdot f'_{(L-1)} \cdot f'_{(L-2)} \cdot \Big(\sum_l w_{0l}^{x^L} \cdot \\
&\quad \Big(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \cdot \delta_{k(1)i}\delta_{nx^{L-2}} S_j^{L-3}\Big)\Big) \\
&+ f'_{(L)} \cdot f'_{(L-1)} \cdot \Big(\sum_l w_{0l}^{x^L} \cdot \delta_{li}\delta_{nx^{L-1}} S_j^{L-2}\Big) \\
&+ f'_{(L)} \cdot \delta_{0i}\delta_{nx^L} S_j^{L-1}
\end{aligned}
$$

이 된다. 이 과정을 time step 0까지 전개하여 정리하면 아래의 식과 같다.

$$
\begin{aligned}
\frac{\widetilde{\partial h_0^L}}{\partial w_{ij}^{n}} &= (f'_{(L)} \cdots f'_{(1)}) \cdot \Big(\sum_l w_{0l}^{x^L} \Big(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \\
&\quad \Big(\cdots \Big(\sum_{k(L-1)} w_{k(L-2)k(L-1)}^{x^1} \cdot \frac{\widetilde{\partial h_{k(L-1)}^{0}}}{\partial w_{ij}^{n}}\Big)\cdots\Big)\Big)\Big) \\
&\quad + (f'_{(L)} \cdots f'_{(1)}) \cdot \Big(\sum_l w_{0l}^{x^L} \Big(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \\
&\quad \Big(\cdots \Big(\sum_{k(L-2)} w_{k(L-3)k(L-2)}^{x^2} \cdot \delta_{k(L-2)i}\delta_{nx^1} S_j^0\Big)\cdots\Big)\Big)\Big) \\
&\quad \vdots \\
&\quad + (f'_{(L)} \cdots f'_{(L-2)}) \cdot \Big(\sum_l w_{0l}^{x^L} \\
&\quad \Big(\sum_{k(1)} w_{lk(1)}^{x^{L-1}} \cdot \delta_{k(1)i}\delta_{nx^{L-2}} S_j^{L-3}\Big)\Big) \\
&\quad + (f'_{(L)} \cdot f'_{(L-1)}) \cdot \Big(\sum_l w_{0l}^{x^L} \big(\delta_{li}\delta_{nx^{L-1}} S_j^{L-2}\big)\Big) \\
&\quad + f'_{(L)} \cdot \delta_{0i}\delta_{nx^L} S_j^{L-1} \qquad (9)
\end{aligned}
$$

여기서 $f'_{(k)} = f'\big(\sum_j w_{0j}^{x^k} S_j^{k-1}\big)$이다.

이 식에서 각 항에 공통적으로 $f'_{(L)}$이 곱해지는데, f가 sigmoid 함수이므로 $f'_{(x)} = f_{(x)} \cdot (1 - f_{(x)})$이다. 따라서 $f_{(L)} = f\big(\sum_j w_{0j}^{x^L} S_j^{L-1}\big) = 0$ 또는 1일 때 $f'_{(L)} = f\big(\sum_j w_{0j}^{x^L} S_j^{L-1}\big) \cdot \big(1 - f\big(\sum_j w_{0j}^{x^L} S_j^{L-1}\big)\big) = 0$이 되므로 $|h_0^L - T| > \varepsilon$인 경우에도 $\frac{\widetilde{\partial h_0^L}}{\partial w_{ij}^{n}} = 0$이 되어 weights 수정이 불가능해진다.

이것은 기존의 알고리즘이 이러한 문제점 때문에 일찍이 $\frac{\widetilde{\partial h_0^L}}{\partial w_{ij}^{n}} = 0$이 되어 학습이 되지 않는데도 불구하고 최대 허용 epoch까지 학습이 진행되는 시간적 낭비를 가지게 된다. 이런 문제점을 보안하기 위해 학습이 실패하는 2가지 상황을 감지하여 새로이 학습할 수 있도록 개선된 내용을 기술하면 아래와 같다.

새로운 weights의 부여 시점.

1) $|h_0^L - T| > \varepsilon$면서 $\frac{\widetilde{\partial h_0^L}}{\partial w_{ij}^{n}} = 0$ 될 때.

2) 학습이 진행 중 최대 epoch에 도달할 때.

여기에 개선된 알고리즘을 적용 기존의 학습알고리즘과 성능을 비교하게 된다.

학습 알고리즘

```
select Tomita-Grammar
repeat up to MAXSEEDS seeds{
/* training phase */
  select initial weights
  repeat up to MAXEPOCHS epochs {
    success_string = 0;
    for n=1 to Training_Set_size {
      present string(n);
      pseudo_gradient_learning(n);
      evaluate error E;
      if ( E > error_tolerance ){
```