웨어러블 라이프로그 학습을 위한 심층 재귀 신경망의 시간 의존성 개선

Term Dependency Improvement of Deep Recurrent Neural Networks for Wearable Lifelog Learning

서울대학교 자유전공학부

손 성 호

웨어러블 라이프로그 학습을 위한 심층 재귀 신경망의 시간 의존성 개선

Term Dependency Improvement of Deep Recurrent Neural Networks for Wearable Lifelog Learning

이 논문을 학생설계전공 인공지능공학
졸업논문으로 제출함.
2015년 12월

서울대학교 자유전공학부

학번 : 2010-13378
이름 : 손성호
긴급연락처 : 010.3577.1991

손성호의 학생설계전공 인공지능공학(공학사) 졸업논문을 인준함.

2015년 12월
지도교수 장병탁 (인)

웨어러블 라이프로그 학습을 위한 심층 재귀 신경망의 시간 의존성 개선

Term Dependency Improvement of Deep Recurrent Neural Networks for Wearable Lifelog Learning

서울대학교 자유전공학부 손성호
geronest@gmail.com

요약

웨어러블 라이프로그는 휴대 장착이 가능한 기기를 통해 일상생활의 다양한 센서 정보를 수집한 시계열 데이터로, 이를 효율적으로 학습하여 높은 정확도의 예측을 하는 것은 실생활과 밀접한 인공지능 서비스를 구현하는 데 필수적이다. 기존 연구에서는 Recurrent Neural Network의 병렬화를 쉽게 하고 긴 시간 의존성을 학습할 수 있도록 여러 개의 데이터 instance가 하나의 은닉 상태에 대응되는 학습 구조를 설계하였는데, 이는 거꾸로 짧은 시간 의존성을 학습하기 어렵다는 한계를 가진다. 이 연구에서는 희소 은닉 상태 Deep Recurrent Neural Network의 성능을 개선하기 위해 은닉 상태가 대응되는 시계열 데이터의 크기를 상황에 따라 조절할 수 있게 하고, 이를 통해 신경망이 다양한 길이의 시간에 대한 의존성을 학습할 수 있도록 하였다.

1. 연구 배경

웨어러블 라이프로그(wearable lifelog)는 Google Glass나 스마트워치와 같은 웨어러블 기기로 수집한 시계열 데이터로서, 기기 사용자의 시선이나 주변 환경의 소리, GPS 정보, 가속도 센서 정보 등을 포함하고 있다[15]. 웨어러블 라이프로그를 학습하면 기기 사용자의 행동패턴 예측이나 주변 환경 인식을 기반으로 한 서비스 추천 등에 활용할 수 있으므로, 이를 효율적으로 학습할 수 있는 알고리즘을 찾아내는 것은 개인 맞춤형 인공지능 서비스를 구현하는 데 필수적이다.

시간에 따라 데이터가 변화하는 양상을 학습할 때에는 신경망 알고리즘 중에서 Deep Recurrent Neural Network(이하 DRNN)을 사용할 수 있는데, 이는 기존의 Deep Neural Network(이하 DNN)이 시계열 데이터를 학습할 수 없기 때문이다. DRNN은 DNN의 은닉 노드에 자기 자신을 가리키는 weight를 추가하여 직전의 은닉 상태가 현재 은닉 상태에 영향을 줄 수 있도록 함으로써 시간을 기준으로 한 데이터의 변화양상을 학습할 수 있다. 그러나 일반적인 DRNN은 직전의 은닉 상태로부터만 영향을 받게 되므로 긴 시간에 대한 의존성이 잘 반영되지 않는 한편 연산의 병렬화가 어렵다는 단점이 있었다. 이를 해결하기 위해 DRNN의 은닉 상태가 주어진 데이터의 매 instance가 아닌 여러 개의 instance에 대응되도록 하여 수행 속도와 정확도에서 개선을 보인 사례가 있었다[14].

본 연구에서는 DRNN에 희소 은닉 상태를 적용한 기존의 연구 결과를 토대로 더욱 다양한 길이의 시간에 대한 의존성을 유연하게 학습할 수 있는 알고리즘 구조를 탐구하였다. 기존 연구의 알고리즘에서는 신경망의 은닉 상태가 학습에 사용하는 구간의 길이가 고정되어 있었던 점에 착안하여, 학습하는 데이터의 경향을 분석해 희소 은닉 상태에 대응되는 구간의 길이를 변화시키는 알고리즘을 구상하였다. 이를 기반으로 데이터 학습을 수행하였으며, 기존 연구에서 다루었던 신경망들의 성능을 함께 비교하였다.

2. 신경망 알고리즘의 개선

2.1. 기존 연구

은닉 노드 각각이 자신을 가리키는 weight를 가지며 직전의 은닉 노드를 참조하는 DRNN은 시간 의존성을 학습할 수 있는 대표적인 학습 모델이다. 그러나 DRNN은 은닉 노드의 weight가 학습 데이터의 instance마다 갱신되어야 하므로 학습하는 데 오랜 시간이 걸리며, 학습할 수 있는 시간 의존성이 짧은 편이다[2, 3, 13]. 이는 긴 시간 의존성을 가지는 요소가 학습하는 시계열 데이터에 있으면 이를 반영한 학습을 하기가 어려움을 의미한다. 신경망의 학습에 널리 이용되는 방법인 경사 하강(gradient descent) 기법 자체가 긴 시간 의존성을 학습하기에 적절하지 않다는 지적이 있는데, 이는 갱신의 기준이 개별 instance인 경우 back-propagation through time 과정에서 vanishing gradient 현상이 쉽게 일어나 시

간축 상에서 멀리 떨어져 있는 instance들 사이의 상관관계를 학습하기가 어려워지기 때문이다[3, 11].

Recurrent Neural Network(이하 RNN)가 긴 시간 의존성을 학습하기 어렵다는 한계를 극복하기 위한 다양한 시도들이 기존 연구에서 제안되었다. Nonlinear AutoRegressive with eXogenous inputs RNN 모델은 신경망이 내놓는 결과값을 지연시킴으로써 계산된 경사 정보가 더 짧은 경로를 통해 전파되도록 하고, 이를 통해 신경망의 긴 시간 의존성을 높여 기존의 RNN보다 약 두 배에서 세 배 정도 오래 정보를 유지할 수 있다고 알려졌다[8]. 한편 데이터의 시간 의존성이 계층적인 구조를 이룬다는 가정에 따라 여러 계층으로 이루어진 RNN을 만들고, 각 계층이 서로 다른 길이의 시간 의존성에 반응하게 하여 학습 정확도를 높인 사례도 있었다[5]. Long Short-Term Memory 기법은 학습 중인 instance에 따라 연결 여부가 결정되는 별개의 memory cell을 정의하여 그 안에 특정 error flow를 보존함으로써 vanishing gradient 현상을 방지하고 긴 시간 의존성을 확보하였다[7]. DRNN의 vanishing gradient 현상을 극복하기 위해서 역전파된 gradient 값의 크기를 적절한 선에서 조절하는 vanishing gradient regularization과 같은 기법 또한 소개되었으나, 이는 역으로 exploding gradient의 가능성을 높이는 것으로 알려졌다[11].

기존의 연구들은 주로 신경망 구조 자체에 수정을 가하여 vanishing gradient 현상을 완화하거나 지연시킴으로써 RNN의 시간 의존성을 개선하고자 하였다. 그러나 RNN이 가지는 또 다른 한계인 긴 처리 시간에 대한 적극적인 개선을 이루지 못하였다는 공통점을 보인다. 이는 기존의 시도들이 웨어러블 기기와 같이 실시간으로 축적되는 데이터를 빠르게 학습하여 결과를 내보내야 하는 환경에서는 적절한 대책이 될 수 없음을 시사한다.

2.2. 희소 은닉 상태 DRNN

희소 은닉 상태 DRNN은 하나의 은닉 상태가 시간축 내에서 연속되는 여러 개의 instance에 대응되며, 은닉층의 학습을 진행할 때 이 instance들에 대응되는 하나의 은닉 노드 단위로 계산을 진행하여 학습한다. 이러한 방식은 기존 DRNN에서 한계로 지적되었던 긴 시간 의존성을 학습할 수 있게 하며, 은닉층의 학습 빈도를 instance의 참조 범위를 늘린 것에 반비례하여 줄이게 되므로 처리 속도 또한 개선할 수 있게 되었다. 이 참조 범위의 확대로 인한 처리 속도 개선은 학습 과정에서의 행렬 연산 숫자 자체가 줄어들기 때문에 일어나는 것으로, 은닉층의 학습에 이용되는 데이터 구간의 크기에 비례하여 행렬 연산의 병렬화 또한 가능한 점을 고려하면 희소 은닉 상태 DRNN의 처리 속도는 더욱 빨라짐을 알 수 있다. 은닉층의 가중치 갱신에 이용되는 학습 데이터 구간의 최대 크기는 확률적 경사 하강(stochastic gradient descent)의 minibatch 크기와 같은데, 이 경우 DNN과 동등한 수준의 병렬화가 가능하여 학습 속도가 매우 빨라진다[14].

한편 희소 은닉 상태 DRNN의 단점으로는 은닉층의 가중치 갱신에 이용되는 데이터 구간이 크기 때문에 단시간 의존성의 학습이 잘 이루어지지 않는다는 것이 있다. 기존 연구에서는 음성 데이터의 경우 MFCC feature 와 같은 것을 이용하여 단기적 시간 의존성이 반영된 요소를 추출한 데이터를 먼저 준비한 뒤 이를 학습하는 방법을 제시하고 있으나, 단기적 시간 의존성을 반영하기에 적절한 데이터 구간의 크기와 긴 시간 의존성을 반영하기에 적절한 구간의 크기를 설정하는 문제가 남게 된다. 특히 학습과 데이터 인식에 사용하는 데이터 구간의 길이가 고정되어 있다는 점은 단일 데이터 내에서도 신경망의 데이터 인식 과정에서 긴 시간 의존성이 중요한 구간과 짧은 시간 의존성이 중요한 구간이 서로 다른 경우 유연하게 적응하지 못하는 원인이 된다.

2.3. 동적 참조범위(Dynamic Reference Range) 알고리즘

본 연구에서는 기존의 DRNN과 희소 은닉 상태 DRNN이 가지는 한계를 극복하기 위하여 희소 은닉 상태 DRNN에 동적 참조범위(Dynamic Reference Range, 이하 DRR) 알고리즘을 도입한 학습 모델을 제안한다. 동적 참조범위 알고리즘은 희소 은닉 상태 DRNN에서 은닉 상태가 학습하고 인식하는 데이터 구간, 즉 참조 범위의 길이를 학습 상황에 따라 스스로 조정하는 것을 골자로 한다. DRR 알고리즘이 도입된 희소 은닉상태 DRNN은 짧은 시간 의존성이 중요한 구간에서는 참조 범위를 줄이고 긴 시간 의존성이 중요한 구간에서는 참조 범위를 늘리게 되는데, 이를 통해 은닉 상태가 다양한 길이의 시간 의존성을 학습하는 한편 참조 범위가 짧게 고정된 경우에 생길 수 있는 처리 속도의 희생을 허용된 범위 안에서 줄일 수 있게 된다.

2.3.1. 알고리즘 구현

DRR 알고리즘을 구현할 때 중요한 것은 학습하는 데이터의 특성에 알맞은 기준을 통해 참조 범위를 조정하는 것이다. 본 연구에서 다루는 문제는 supervised sequence labeling에 해당하므로, 참조

범위 안에서 최대 빈도를 보이는 특정 label의 비율에 따라서 참조범위를 두 배 늘리거나 반으로 줄이는 방식으로 DRR을 구현하였다. 아래의 [표 1]에 제시된 변수 중 알고리즘 사용자가 사전에 설정할 수 있는 것으로는 range_max, range_min, lim_acc, lim_dec가 있다.

변수 이름	의미
data	학습할 데이터
index	학습할 범위의 시작위치
range	참조 범위의 길이
range_max	최대 참조 범위
range_min	최소 참조 범위
lim_acc	참조 범위 확대 기준
lim_dec	참조 범위 축소 기준
cnt_acc	참조 범위 확대 횟수
cnt_dec	참조 범위 축소 횟수

[표 2] pseudo code에서 쓰이는 변수들의 의미

[그림 1] Dynamic Reference Range 알고리즘의 적용 예시. 그림의 숫자들은 각각 하나의 instance를 분류하는 label을 의미하며, 숫자들 위의 구간은 은닉 상태가 학습할 때 참조하는 데이터 구간을 의미한다. 숫자들 위의 구간과 각 구간 내에서 최대 빈도를 보이는 label은 같은 색으로 설정되어 있다.

위의 [그림 1]은 range_min = 2, range_max = 16, lim_acc = 0.7, lim_dec = 0.6으로 설정된 DRR 알고리즘을 도입한 희소 은닉 상태 DRNN이 20개의 연속된 instance를 학습하는 과정에서 보이는 참조 구간 길이의 변화 예시를 나타낸다. 그림에서의 처음 두 instance가 분류된 label이 같은 값이기에 해당 구간 내에서의 최대 빈도 label의 비율은 lim_acc의 값을 초과하는 1이 되고, 그 다음 참조 구간의 길이는 두 배 늘어난 4가 된다. 두 번째 참조 구간에서는 최대 빈도 label의 비율이 lim_dec의 값보다 작은 0.5이므로 [그림 1]에서의 세 번째 참조 구간의 길이는 반으로 줄어든 2가 된다. 같은 방법으로 네 번째 참조 구간의 길이는 다시 4, 다섯 번째 참조 구간의 길이는 8이 되며, 여기서는 최대 빈도 label이 여덟 개의 instance 중 3개에서 나타나 lim_dec의 값보다 작은 비율 0.375를 보인다. 따라서 [그림 1]에 나타나지 않은 여섯 번째 참조 구간에서는 참조 구간의 길이가 반으로 줄어든 4가 될 것이다. 한편 참조 구간 내에서 최대 빈도 label의 비율이 계속해서 lim_acc의 값을 초과하더라도 참조 구간의 길이는 range_max를 초과하지 않는다.

[그림 1]에서는 예시로 든 데이터를 이루는 instance가 20개밖에 되지 않아 참조 구간의 길이를 적절하게 조정하는 것이 비교적 간단했으나, instance의 개수가 많은 큰 용량의 데이터를 처리할 때에는 참조 구간의 길이를 더욱 극적으로 줄이거나 늘려야 하는 경우가 생길 수 있다. 이러한 필요에 맞추어 학습을 진행하기 전에 참조 구간의 길이를 여러 번 늘리거나 줄일 수 있도록 구현한 DRR 알고리즘을 Dynamic_Reference_Range_Loop으로, 참조 구간의 조정을 1회로 한정한 DRR 알고리즘을 Dynamic_Reference_Range_Single로 구분하였다. 각 알고리즘의 pseudo code는 아래와 같다.

Dynamic_Reference_Range_Loop

```
(data, index, range, lim_acc, lim_dec, cnt_acc,
cnt_dec, range_max, range_min)
cnt_acc = 0
cnt_dec = 0
while true
    most_label_occurrence =
      data [ index + 1 : index + range ] 에서
      가장 높은 빈도로 등장하는 label의 등장 횟수
    most_label_ratio =
      most_label_occurence / range
    if ( max_label_ratio <= lim_dec
        && range > range_min)
          range = range / 2
          cnt_dec = cnt_dec + 1
    else if ( max_label_ratio > lim_acc
            && range < range_max)
            range = range * 2
            cnt_acc = cnt_acc + 1
        else break
        ifend
    ifend
    if ( cnt_acc > 0 && cnt_dec > 0)
        break
    ifend
whileend
```

```
Dynamic_Reference_Range_Single
(data, index, range, lim_acc, lim_dec,
cnt_acc, cnt_dec, range_max, range_min)
most_label_occurrence =
    data [ index + 1 : index + range ] 에서
    가장 높은 빈도로 등장하는 label의 등장 횟수
most_label_ratio =
    most_label_occurence / range
if ( max_label_ratio <= lim_dec
    && range > range_min)
        range = range / 2
        cnt_dec = cnt_dec + 1
else if ( max_label_ratio > lim_acc
         && range < range_max)
        range = range * 2
        cnt_acc = cnt_acc + 1
     ifend
ifend
```

2.3.2. 시간 복잡도(time complexity)

Dynamic_Reference_Range_Single의 경우, most_label_occurrence를 계산하는 부분을 제외한 다른 모든 부분의 시간 복잡도는 O(1) 이다. most_label_occurrence를 구하는 부분의 시간 복잡도는 range의 크기와 데이터를 분류하는 클래스 개수(이하 n)에 비례하므로 O(range × n)이 된다.

Dynamic_Reference_Range_Loop의 경우 while 문 안의 내용을 반복하는 최대 횟수는 cnt_acc + cnt_dec의 최댓값과 같다. cnt_acc + cnt_dec 의 최댓값은 DRR 알고리즘이 사용할 수 있는 참조 범위 크기의 가짓수와 같으므로 $\log_2$(range_max) - $\log_2$(range_min) + 1 이 되며, 이 값을 m이라 하면 Dynamic_Reference_Range_Loop의 시간 복잡도는 O(range × n × m)이 된다. 즉 본 연구에서 구현한 DRR 알고리즘의 시간 복잡도는 신경망 알고리즘의 시간 복잡도에서 큰 비중을 차지하는 행렬 연산과 관계가 없으며, 따라서 신경망 알고리즘의 처리 시간에 미치는 영향이 미미하다고 볼 수 있다. 한편 DRR 알고리즘은 학습 과정에서 신경망을 이루는 다른 데이터나 변수들을 수정하지 않으므로, DRR 알고리즘을 도입한 희소 은닉 상태 DRNN은 range_max를 참조 범위 길이로 설정한 희소 은닉 상태 DRNN보다 더 짧은 처리 시간을 가질 수 없음을 예측할 수 있다.

2.3.3. 신경망 알고리즘에 대한 적용

DRR을 적용한 신경망을 이용하여 데이터의 학습을 진행할 때에는 학습 대상이 되는 데이터에 이미 labeling이 되어 있는 상태이므로 은닉 상태의 학습에 적절한 참조 범위를 설정하는 것이 쉽다. 따라서 은닉 상태를 갱신하기 전마다 Dynamic_Reference_Range_Loop 알고리즘을 실행하여 학습에 이용할 데이터의 범위를 최대한 적절하게 맞춰주는 것이 신경망의 성능을 높이는 데 유리하다.

한편 이미 학습한 신경망을 가지고 정확도를 측정할 때에는 주어지는 데이터에 미리 labeling이 되어 있지 않으므로 신경망이 데이터를 인식하기 전에 미리 참조범위를 조정하는 것이 불가능하다. 이 경우에는 신경망이 데이터를 인식하여 내놓은 labeling 결과를 기반으로 그다음 참조 범위를 점진적으로 조절하는 방법을 쓸 수 있으며, 이 경우에는 Dynamic_Reference_Range_Single 만을 1회 실행한다.

3. 실험

본 연구에서 학습에 사용할 웨어러블 라이프로그는 스마트글래스를 활용해 2명의 피험자가 촬영한 영상에서 추출한 소리정보에서 사람의 소리 인지 주파수를 반영하는 것으로 알려진 MFCC 계수를 13차원으로 추출한 것이다[15]. 각각의 instance는 추출된 MFCC 계수에 따라 총 13개의 feature로 이루어져 있으며, 피험자가 영상을 촬영하던 당시의 상황에 따라 총 10개의 클래스로 분류되었다. 본 연구에서는 총 3,542,714개의 instance가 약 600MB 정도의 용량을 차지하는 원본 데이터에서 100분의 1에 해당하는 약 6MB의 데이터를 추출하여 실험을 진행하였다.

웨어러블 라이프로그 데이터를 이용한 실험을 진행하기 전 모의 실험을 진행하여 실험 과정이 올바르게 수행되고 있는지, 기존의 학습 기법과 본 연구에서 제안한 DRR 기법이 도입된 희소 은닉 상태 DRNN이 어떤 차이를 보이는지를 확인한다. 이를 위해 시계열 데이터의 속성을 가진 모의 데이터와 그렇지 않은 모의 데이터를 임의로 생성하여 사용한다. 각 신경망의 성능을 평가할 때에는 크로스 엔트로피와 데이터 클래스 분류의 정확도를 측정하며, 신경망이 데이터를 학습하는 데 걸리는 시간을 함께 기록하여 서로 비교한다.

각 실험을 수행하는 프로그램은 MATLAB을 이용하여 구현하였으며, 실험에서 사용한 학습 모델들은 같은 크기의 은닉 층과 은닉 노드, 학습률(learning rate)과 반복횟수를 갖도록 설정하였다.

3.1. 모의 데이터 실험

시계열 속성을 보이지 않는 모의 데이터(이하 모의

실험 데이터 1)는 1 이상 1000 이하의 자연수를 클래스 1로, 1001 이상 2000 이하의 자연수를 클래스 2로 지정한 것이다. 모의 시계열 데이터(이하 모의실험 데이터 2)는 1에서 10까지 숫자가 증가하는 벡터에는 클래스 1을, 10에서 1까지 숫자가 감소하는 벡터에는 클래스 2를 지정한 것이다.

모의실험에서는 각 신경망이 50개, 25개의 은닉 노드를 가진 은닉층 2개를 갖도록 구성하였으며, DRNN이 적용된 모델에서는 1-truncated back-propagation through time (이하 bptt)을 사용하였다. 실험 결과를 나타낸 표에서는 편의상 '희소 은닉 상태 DRNN'은 SRNN(Sparse Hidden State Deep Recurrent Neural Network)로 표기하며, 'DRR을 도입한 희소 은닉상태 DRNN'은 DRR로만 표기한다. DRR 알고리즘에서 lim_acc의 값은 0.9, lim_dec의 값은 0.6으로 두었다.

알고리즘	참조 범위	크로스 엔트로피	테스트 정확도	평균 처리 시간(초)
DNN	128	36.078	74.6%	0.11
DRNN	1	0.114	74.7%	3.68
SRNN	2	0.16	73.9%	1.85
SRNN	4	0.246	73.8%	0.95
DRR	2 ~ 4	0.533	74.5%	1.14

[표 3] 모의실험 데이터 1을 이용한 실험 결과

모의실험 1의 경우, 시계열 데이터의 속성이 반영되지 않은 데이터를 사용하여 학습을 진행했기 때문에 처리 시간이 압도적으로 짧으면서도 테스트 정확도가 두 번째로 높은 DNN의 성능이 월등함을 알 수 있다. 즉 DRNN이나 희소 은닉 상태 DRNN, DRR을 도입한 희소 은닉 상태 DRNN 등은 시계열 데이터의 속성을 띠지 않는 데이터를 학습할 때 DNN을 대체할 만큼의 우월한 성능을 보이지 못한다. 다만 DRNN이 도입된 학습 모델끼리 비교했을 때에는 DRR을 도입한 알고리즘이 평균 처리 시간이 두 번째로 짧으면서도 두 번째로 높은 테스트 정확도를 보인다.

알고리즘	참조 범위	크로스 엔트로피	테스트 정확도	평균 처리 시간(초)
DNN	128	62.847	51.2%	0.12
DRNN	1	0.24	95.0%	3.92
SRNN	4	0.329	90.0%	1.03
SRNN	8	1.838	79.6%	0.54
DRR	4 ~ 8	2.275	91.9%	0.56

[표 4] 모의 실험 데이터 2를 이용한 실험 결과

모의실험 2에서는 시계열 데이터의 속성을 갖는 데이터를 사용하여 학습을 진행하였기 때문에 시계열 데이터를 제대로 학습하지 못하는 DNN의 성능 향상이 미미했음을 확인할 수 있다. DRNN은 가장 긴 평균 처리시간과 함께 가장 높은 테스트 정확도를 보이며, 희소 은닉 상태 DRNN에서는 참조 범위가 늘어날수록 평균 처리 시간과 테스트 정확도가 감소하는 경향이 확인된다. 한편 DRR을 도입한 희소 은닉 상태 DRNN 알고리즘은 두 번째로 높은 테스트 정확도와 두 번째로 짧은 평균 처리시간을 보인다.

3.2. 웨어러블 라이프로그 데이터 실험

웨어러블 라이프로그 데이터 실험에서는 각 신경망이 2500개, 1500개, 500개의 은닉 노드를 가진 은닉층 3개를 갖도록 구성하였으며, 10-truncated bptt를 사용하였다. DRR 알고리즘에서 lim_acc의 값은 0.8, lim_dec의 값은 0.4로 두었다.

알고리즘	참조 범위	크로스 엔트로피	테스트 정확도	평균 처리 시간(초)
DNN	1024	136.893	9.5%	46.3
SRNN	128	43.261	32.3%	952.2
SRNN	256	92.107	32.0%	652.3
SRNN	512	212.969	32.6%	514.7
SRNN	1024	412.338	29.4%	476.3
DRR	128 ~512	197.527	31.5%	557.5
DRR	2 ~ 1024	328.125	33.8%	463.1

[표 5] 웨어러블 라이프로그 데이터를 이용한 실험 결과

웨어러블 라이프로그 데이터를 이용한 실험 결과는 모의실험 2와 미세하게 다른 경향을 보였다. 희소 은닉 상태 DRNN의 경우 참조 범위가 늘어날수록 평균 처리시간은 줄어든 반면 테스트 정확도는 참조 범위의 크기가 512일 때 가장 높았으며, DRR을 도입한 희소 은닉 상태 DRNN의 경우 참조 범위가 최소 2, 최대 1024인 경우에 최소 128, 최대 512인 경우보다 더 높은 테스트 정확도와 더 짧은 평균 처리시간을 보였다. 웨어러블 라이프로그 데이터는 시계열 데이터이므로 모의실험 2의 결과에서처럼 DNN은 학습 이후에도 매우 낮은 테스트 정확도를 보였다.

4. 실험 결과 분석 및 논의

4.1. DRR 알고리즘을 통한 성능 향상

DRR 알고리즘을 도입한 희소 은닉 상태 DRNN은 앞서 진행했던 대부분의 실험에서 DRR 알고리즘을 도입하지 않은 희소 은닉 상태 DRNN에 비해 향상된 성능을 보여주었다.

기존의 희소 은닉 상태 DRNN은 처리 시간을 줄이고 긴 시간 의존성을 학습하기 위해 참조 범위를 늘

리면 테스트 정확도를 희생하게 되는 경향을 보였다. 이는 [표 3]에 정리된 모의실험 2의 결과에서 두드러지는데, 실험에서 사용한 데이터가 같은 패턴을 보이는 짧은 구간을 여러 개 이어붙인 형태로 생성되었기 때문에 긴 시간 의존성의 학습이 필요하지 않았기 때문으로 볼 수 있다. 웨어러블 라이프로그의 경우 데이터에 서로 다른 길이의 시간 의존성이 병존하여 참조 범위의 크기와 테스트 정확도가 완전히 반비례하지 않았다.

반면 DRR을 도입한 희소 은닉 상태 DRNN은 DRR을 도입하지 않은 경우와 비교했을 때 더욱 뛰어나거나 버금가는 테스트 정확도를 보이며, 처리 속도 측면에서도 기존의 알고리즘이 최대 길이의 참조 범위를 이용했을 때 보이는 수준에서 크게 뒤처지지 않는다. 기존의 알고리즘이 한쪽을 취하면 다른 한쪽을 희생해야 했던 장점 두 가지를 모두 얻을 수 있게 된 것이다. 이는 '짧은 시간 의존성이 중요한 구간에서는 참조 범위를 줄이고, 긴 시간 의존성이 중요한 구간에서는 참조 범위를 늘린다'는 DRR의 기본 아이디어가 유효했음을 보여준다.

웨어러블 라이프로그 데이터를 이용한 실험에서는 대부분의 학습 기법들이 30% 근방의 정확도를 보인다. 이는 학습에 사용한 데이터의 instance가 분류될 수 있는 클래스가 10종류로 많은 편이고, 실험 환경 특성상 테스트 정확도가 일정 수준에 도달할 때까지 학습 반복 횟수를 충분히 늘리지 못했기 때문이다. 즉 [표 4]의 실험결과가 신경망 알고리즘을 이용해 웨어러블 데이터로그를 학습하는 것 자체의 유용성을 입증하기에는 무리가 있으나, 서로 다른 알고리즘 사이에 테스트 정확도나 평균 처리 시간 차원에서 비교 가능한 차이가 나타나므로 DRR이 도입된 희소 은닉 상태 DRNN의 성능을 기존 알고리즘과 비교한다는 본 연구의 취지는 충족된다고 볼 수 있다.

4.2. DRR 알고리즘에서의 parameter 설정 문제

DRR 알고리즘의 pseudo code를 제시할 때 밝혔던 바와 같이, DRR을 도입할 때 사용자가 사전에 지정해 주어야 하는 알고리즘의 변수로는 range_max, range_min, lim_acc, lim_dec의 네 가지가 있다. 이 변수들을 적절하게 설정하는 것이 알고리즘의 성능에 큰 영향을 미치는데, 이는 다른 변인을 통제하고 DRR 관계 변수만을 바꾸어 모의실험 데이터 2를 가지고 실험한 아래의 [표 5]에서 확인할 수 있다.

range_max	range_min	lim_acc	lim_dec	테스트 정확도	평균 처리 시간(초)
8	4	0.9	0.6	91.4%	0.552
8	4	0.9	0.7	92.6%	0.569
8	4	0.9	0.8	93.0%	0.668
8	4	0.7	0.6	91.4%	0.546
8	2	0.9	0.6	92.3%	0.559
16	4	0.9	0.6	83.1%	0.369

[표 6] 모의 실험 데이터 2를 사용하고 DRR 관계 변수를 바꾸어 실험한 결과

lim_acc와 lim_dec는 DRR 알고리즘이 상황에 따라서 참조 범위를 조절하는 기준이 된다. 따라서 이 두 변수의 값을 적절하게 설정해 주지 않으면 참조 범위가 적절한 때 늘어나거나 줄어들지 못하므로 DRR 알고리즘을 도입하더라도 제대로 된 성능 향상을 기대할 수 없다. [표 5]의 실험 결과는 lim_acc과 lim_dec의 값이 적절하게 설정되었을 때 테스트 정확도가 개선될 수 있음을 보여준다.

반면 range_max와 range_min은 참조 범위가 늘어나고 줄어들 수 있는 한계를 정하는 기준이 된다. 이때 DRR 알고리즘의 성능에 주로 영향을 미치는 변수는 주로 range_max라 할 수 있다. range_max의 값이 지나치게 작은 경우 평균 처리 시간이 길어지고, 그 값이 시계열 데이터가 가지고 있는 시간 의존성의 범위에 비해 지나치게 크게 설정되면 평균 처리 시간이 줄어드는 반면 테스트 정확도 측면에서 손실이 일어날 수 있기 때문이다. [표 5]에서는 참조 범위의 하한이 적절하게 늘어나자 테스트 정확도가 개선되고, 상한이 부적절하게 늘어나자 평균 처리 시간이 줄어든 반면 정확도는 오히려 떨어지는 것을 확인할 수 있다.

4.3. 추후 과제

본 연구에서는 편의상 DRR을 구현할 때 참조 범위가 두 배씩 늘어나거나 반으로 줄어들도록 하고, 참조 범위가 변화하는 기준을 알고리즘의 most_label_ratio, 즉 참조 범위 내에서 최대 빈도를 가지는 label의 비율로 설정하였다. 이렇게 구현한 DRR을 희소 은닉 상태 DRNN에 도입하는 것만으로도 위에서 보인 바와 같은 성능 향상을 이끌어낼 수 있었으나, 이는 supervised sequence labeling 문제에서 활용하기에 적합한 방식이다. 같은 방식으로 구현한 DRR 알고리즘이 본 연구에서 사용했던 데이터와 다른 특성을 보이는 시계열 데이터에서도 성능 향상을 이끌어낼 수 있으리라는 보장은 없는 것이다. 즉, 더욱 다양한 특성을 보이는 다른 시계열 데이터들에 맞춘 DRR 알고리즘을 구현하는 것이 추후 연구에서 다룰 만한 과제가 될 것이다.

한편 풀고자 하는 문제에 적합한 형태의 DRR 알고리즘을 구현했더라도 알고리즘에 관계된 변수들을 적절히 설정하는 것이 중요한 문제가 된다. 4.2에서 보인 바와 같이 parameter에 어떤 값을 설정하느냐가 신경망의 성능 향상에 큰 영향을 미칠 수 있기 때문이다. 따라서 DRR 알고리즘의 구체적인 형태를 주었을 때 학습에 적합한 parameter를 찾는 방법을 연구하는 것 또한 중요한 연구 과제가 될 수 있다.

본 연구에서는 실험을 수행하는 프로그램을 MATLAB으로 작성하였으며, 프로그램을 구동하는 컴퓨터의 성능상의 한계로 인하여 원본 데이터의 100분의 1만을 추출하여 실험에 사용하였다. 추후 연구에서는 연산의 병렬화가 가능한 환경에서 프로그램을 구현하고 실험을 진행하여 본 연구에서 제안한 알고리즘이 보이는 성능 개선이나 한계를 검증할 필요가 있다. 또한 고성능 컴퓨터에서 원본 데이터 전체를 사용하는 실험을 진행함으로써 DRR 알고리즘의 유용성을 확실히 검증할 필요가 있다.

5. 결론

희소 은닉 상태 Deep Recurrent Neural Network는 기존의 시도들과 달리 Recurrent Neural Network의 두 가지 한계인 긴 처리 시간과 짧은 시간 의존성 모두에서 개선된 성능을 보인다. 반면 희소 은닉 상태 DRNN은 은닉 상태가 학습하는 데이터 구간의 길이가 고정되어 있기 때문에 거꾸로 짧은 시간 의존성의 학습이 어렵다는 단점이 있다.

본 연구에서는 희소 은닉 상태 DRNN의 기존 한계점을 극복하고 성능을 개선하는 것을 목표로 하였다. 이를 위해 상황에 따라 학습하는 참조 범위의 길이를 늘이거나 줄일 수 있는 Dynamic Reference Range 알고리즘을 구상하여 희소 은닉 상태 DRNN에 도입하였다. 두 종류의 모의실험과 웨어러블 라이프로그 데이터를 활용한 실험에서 기존의 신경망 기반 학습 모델과 DRR을 도입한 희소 은닉 상태 DRNN의 성능을 비교하였으며, 시계열 데이터를 학습할 때 DRR 알고리즘에 적절한 변수가 설정된 경우 평균 처리 시간과 테스트 정확도 측면에서 성능 향상이 있음을 확인하였다.

한편 본 연구에서는 supervised sequence labeling 문제의 해결에 초점을 맞추고 DRR 알고리즘을 구현하였으며, 다른 문제나 데이터를 이용한 학습에서는 본 연구에서 제시한 형태의 DRR 알고리즘이 성능 향상에 이바지하기 힘들 수 있음을 밝혔다. 또한 신경망 알고리즘의 성능 향상에서 DRR 알고리즘과 관계된 변수들을 적절한 수치로 설정하는 것이 중요함을 밝히며 이를 추후 연구 과제로 제시하였다.

감사의 글

이 연구는 서울대학교 컴퓨터공학부 장병탁 교수님의 지도를 받아 진행되었으며, 논문의 작성 및 검토 과정에서 서울대학교 컴퓨터공학부 바이오지능 연구실의 장하영 선배와 곽동현 선배의 도움을 받았다. 도움을 아끼지 않으신 위 세 분께 감사드린다.

참고문헌

[1] Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), pp. 1-127.

[2] Bengio, Y., Frasconi, P., and Simard, P. 1993. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks 3*, pp. 1183-1195.

[3] Bengio, Y., Simard, P., and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), pp. 157-166.

[4] Chen, J. and Deng, L. 2013. A new method for learning deep recurrent nerual networks. *arXiv*:1311.6091 [cs.LG].

[5] El Hihi, S. and Bengio, Y. 1996. Hierarchical recurrent nerual networks for long-term dependencies. In *Advances in Neural Information Processing Systems 8*. MIT Press.

[6] Hermans, M. and Schrauwen, B. 2013. Training and analysing deep recurrent nerual networks. In *Advances in Nerual Information Processing Systems 26*, pp.190-198.

[7] Hochreiter, S. and Schmidhuber, J. 1997. Long Short-term memory. *Neural Computation*, 9(8), pp. 1735-1780.

[8] Lin, T., Horne, B., Tino, P., and Glies, C. 1996. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6), pp. 1329-1338.

[9] Martens, J. and Sutskever, I. 2011. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning 28*, pp. 1033-1040.

[10] Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. 2014. How to Construct Deep Recurrent Nerual networks. *arXiv*:1312.6026v5 [cs.NE].

[11] Pascanu, R., Mikolov, T., and Bengio, Y. 2013. On the difficulty of training Recurrent Neural Networks, In *Proceedings of the 30th International Conference on Machine Learning*.

[12] Sutskever, I. 2013. Training recurrent neural networks. Diss. University of Toronto.

[13] Williams, R. and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2), pp. 270-280.

[14] 곽동현, 이충연, 곽하녹, 장병탁. 2015. 희소 은닉 상태 Deep Recurrent Neural Network을 이용한 웨어러블 라이프로그 학습, 한국컴퓨터종합학술대회 논문집, pp.751-753.

[15] 이충연, 이상우, 곽동현, 장병탁. 2014. 웨어러블 라이프로그 기반 상황 인식과 응용 시나리오, 한국정보과학회 동계학술발표회 논문집, pp.337-339.