7 장: 딥재귀신경망

다층퍼셉트론 신경망은 입력이 출력방향으로만 활성화되고 은닉뉴런이 과거의 정보를 기억하지는 못한다. 이러한 망은 입력이 들어온 문맥을 기억할 수 없는 단점이 있다. 언어처리와 같이 앞에서 무슨 단어가 나왔는지의 문맥이 다음에 나올 단어를 예측하는데 도움이 되는 문제 해결에는 큰 단점으로 작용한다. 이러한 문제는 일찍이 인지과학자들에 의해서 인식되었다. 본 장에서는 문맥을 기억할 수 있는 재귀신경망 (Recurrent Neural Network, RNN)을 살펴본다. 7.2 절에서는 재귀신경망을 학습하기 위한 Back Propagation through Time (BPTT) 학습 방법을 살펴본다. 7.3 절에서는 재귀신경망을 딥구조로 확장한 딥재귀신경망 구조 (Deep Recurrent Neural Network, DRNN)를 살펴본다. 7.4 절에서는 메모리를 명시적으로 저장하는 뉴런을 가진 장단기메모리 신경망 (Long Short-Term Memory, LSTM)을 살펴본다.

7.1 재귀신경망 (Recurrent Neural Networks, RNN)

지금까지 살펴본 딥러닝 모델은 입력 패턴들의 시간적인 순서를 고려하지 않았다. 즉 시간 t 때에 입력 $\mathbf{x}(t)$이 주어지면 대응되는 출력 $\mathbf{y}(t)$을 생성하며, 이 때 t 시간의 출력은 t 시간의 입력에만 의존적이다. 이러한 학습 모델은 정적인 입출력 사상을 학습한다.

$$\mathbf{y}(t)=f(\mathbf{x}(t);\mathbf{w})$$

정적인 신경망의 단점은 과거 역사나 단기기억 능력을 가지지 못한다. 그러나 언어생성, 동작생성, 음악 생성 등과 같은 시계열 데이터는 시간에 따라 변하는 동적인 입출력 매핑을 학습할 필요가 있다.

$$\mathbf{y}(t)=f(\mathbf{x}(t),\mathbf{x}(t-1),...,\mathbf{x}(1);\mathbf{w})$$

여기서 $\mathbf{x}(t)$와 $\mathbf{y}(t)$는 시간에 따라 변하는 패턴들이다. 재귀신경망은 비정적인 입출력 사상을 학습하는 동적 시스템 모델이다.

재귀신경망은 뉴런들의 출력이 다시 입력으로 되먹임되는 재귀적인 연결 구조를 갖는 신경망이다. 되먹임의 구조는 직접적일 수도 있고 다른 뉴런을 통해서 간접적일 수도 있다. 재귀 신경망은 시간적인 동역학을 가지고 있어서 입력의 시간적 구조를 파악할 수 있으며 이로 인해서 같은 입력 패턴에 대해서 시간에 따라 다른 출력을 생성할 수 있다. 그림 7.1 은 전형적인 재귀신경망 구조를 보여준다. 이는 다층퍼셉트론의 변형으로서 은닉뉴런층만이 재귀적인 구조를 갖는다. 이 구조는 $t-1$ 시간에서의 은닉뉴런 활성값들 $\mathbf{h}(t-1)$이 t 시간에 복제되어 계속 과거의 정보를 축적하여 나가는 모델이다.

$$\mathbf{y}(t)=f(\mathbf{x}(t),\mathbf{h}(t);\mathbf{w})$$
$$\mathbf{h}(t)=\mathbf{g}(\mathbf{h}(t-1);\mathbf{w})$$

여기서 $\mathbf{h}(t-1)$는 시간 $t-1$ 에서의 은닉층 활성화 벡터이다. 은닉층의 활성값들이 재귀적으로 계속 축적되기 때문에 시간 t 에서의 은닉활성화는 그 시간까지의 과거 역사를 다 기록하는 효과가 있다.

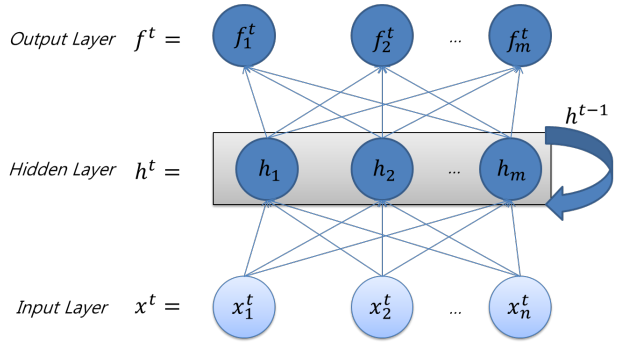


그림 7.1 재귀신경망의 구조 (시간 t)

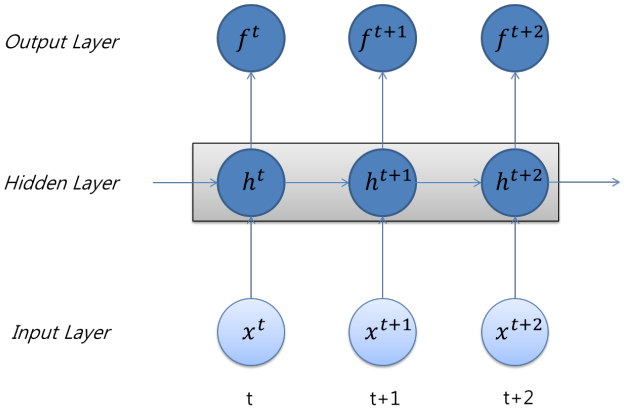


그림 7.2 재귀신경망의 시간적 펼침 (시간 t, $t+1$, $t+2$, $\cdots$)

재귀신경망은 $t-1$ 시간에 활성화된 은닉 뉴런 활성값이 t 시간의 활성화에 입력으로 작용하는 재귀적 활성화를 수행한다. h_j^{t-1} 가 시각 $t-1$ 때의 j 번째 은닉 뉴런의 활성값이라고 하면, 시각 t 에서의 은닉뉴런의 활성값 h_j^t 는 다음으로 주어진다.

$$\text{sum}_j^t = \sum_i w_{ji} x_i^t + \sum_{j \in \text{Hiddens}} w'_{ji} h_j'^{t-1}$$

$$h_j^t = h_j^t(\text{sum}_j^t) = \frac{1}{1+\exp(-\text{sum}_j^t)}$$

재귀망이 아닐 경우 은닉뉴런의 총 입력 식 sum_j^t 에서 첫번째 항으로만 구성되고 두번째 항이 없다는 점이 다르다. 재귀신경망의 출력층의 값들 $\boldsymbol{f_k^t}$ 은 은닉층의 값들 $\boldsymbol{h_j^t}$ 에 기반하여 다음과 같이 주어진다.

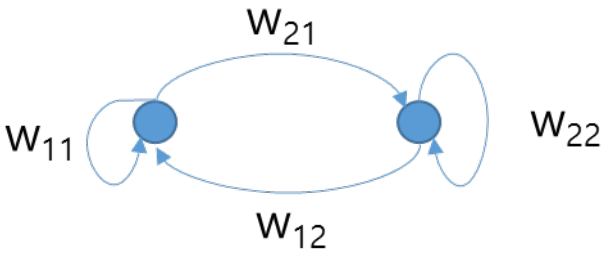
$$f_k^t = f_k^t(\text{sum}_k^t) = \frac{1}{1+\exp(-\text{sum}_k^t)}$$

$$\text{sum}_k^t = \sum_{j \in \text{Hiddens}} w_{kj} h_j^t$$

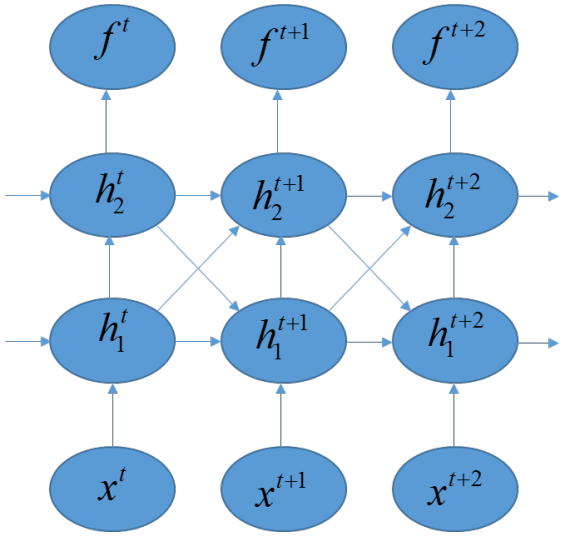
다층퍼셉트론이나 이에 기반한 딥러닝 모델들은 모두 정적인 학습 모델로서 입력벡터로부터 출력벡터로의 고정된 매핑을 수행한다. 반면에 재귀신경망은 원리적으로 과거 입력의 전체 히스토리로부터 출력노드로의 동적인 매핑을 수행한다. 재귀적 연결을 통해서 과거 입력들의 기억이 네트워크의 내부 상태로 지속적으로 남아있을 수 있게 한다.

7.2 BPTT 망

재귀 신경망의 전방 패스를 시간 t 에 대해서 펼치면 t 개의 층을 가진 순방향 신경망처럼 생각할 수 있다. 그림 7.4는 그러한 예를 보여준다. 이 재귀신경망은 두 개의 뉴런으로 구성되며 두 뉴런 간에 그리고 각 뉴런 자신과 되먹임 구조를 갖는다(그림 7.4a). 이를 시간에 대해서 펼치면 그림 7.4b와 같은 전향 구조의 딥신경망이 된다.



(a)



(b)

그림 7.4 두 개의 뉴런으로 구성된 재귀신경망 구조. (a) 각 뉴런 자신의 그리고 두 뉴런 사이의 되먹임 구조. (b) 시간에 대해서 펼친 전향 구조의 딥신경망.

이 구조는 오류역전파 알고리듬을 사용하여 학습할 수 있으며 이를 시간펼침 역전파 알고리듬(Backpropagation through Time, BPTT)이라 한다. 학습은 BPTT 망의 가중치 값들을 변경하는 것이며 다음과 같이 표현할 수 있다.

$$w_{ji} \leftarrow w_{ji} - \eta \frac{\partial E}{w_{ji}}$$

여기서 η 는 학습률이며 $\frac{\partial E}{\partial w_{ji}}$ 는 네트워크 가중치에 대한 에러 미분값을 의미한다. BPTT망에서는 길이 T 인 시퀀스에 대해서 이 값들이 합한것이며 연쇄 법칙을 써서 표현하면 다음과 같다.

$$\begin{aligned}\frac{\partial E}{w_{ji}} &= \sum_{t=1}^{T} \frac{\partial E}{\partial \mathrm{net}_j^t} \frac{\partial \mathrm{net}_j^t}{\partial w_{ji}} \\ &= \sum_{t=1}^{T} \delta_j^t x_i^t\end{aligned}$$

위에서 우변의 둘째 항은 다음을 이용하였다.

$$\begin{aligned}\frac{\partial \mathrm{net}_j^t}{\partial w_{ji}} &= \frac{\partial}{\partial w_{ji}} \left(\sum_{i=1}^{I} w_{ji} x_i^t + \sum_{j'=1}^{H} w_{jj'} h_{j'}^{t-1} \right) \\ &= x_i^t\end{aligned}$$

첫번째의 항은 시간 t 에서의 뉴런 j 의 총입력에 대한 에러 변화율을 나타내며 이를 델타 δ_j^t 라고 정의하자.

$$\delta_j^t \triangleq \frac{\partial E}{\partial net_j^t}$$

이 델타는 시간 t 에 대한 인덱스가 추가된 것 외에는 표준 오류역전파 알고리듬에서의 델타와 같다. 한편 표준 오류역전파 알고리듬에서와는 달리 BPTT 망에서의 델타 δ_j^t는 출력층에서의 에러에 미치는 영향 δ_k^t (k는 출력뉴런) 뿐만 아니라 재귀적인 구조에 의해서 다음 시간 $(t+1)$에서의 은닉층에 대해 미치는 영향 $\delta_{j'}^{t+1}$(j' 는 은닉 뉴런) 또한 포함한다. 즉 델타가 다음과 같이 두 개의 항으로 구성된다.

$$\delta_j^t = f'(a_j^t) \left(\sum_{k=1}^{K} \delta_k^t w_{kj} + \sum_{j'=1}^{H} \delta_{j'}^{t+1} w_{j'j} \right)$$

에러 델타 계산은 $t = T$ 에서 출발하여 t 를 하나씩 줄이면서 순차적으로 위의 델타

갱신식을 적용하여 계산된다.

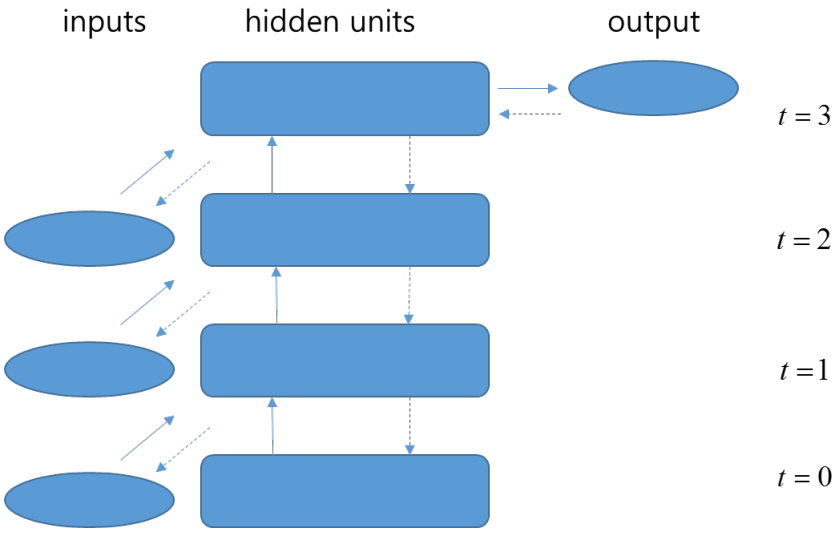


그림 7.5 길이 $T=3$ 인 입력벡터와 재귀신경망

길이 T 인 입력벡터 x 의 시퀀스를 I-H-K 재귀신경망에 제시한다. 다음을 정의하자.

x_i^t : 시간 t 에서의 입력 i 의 값

net_j^t: 시간 t 에서의 뉴런 j 의 네트워크 입력값

h_j^t: 시간 t 에서의 뉴런 j 의 활성값

w_{ji}: 유닛 i 로부터 유닛 j 로의 연결 가중치

길이 T 인 입력벡터 x 의 시퀀스에 대해서 전방 패스를 실시한다. $t=1$ 에서 출발하여 t 를 증가시키면서 반복적으로 가중치 변경식을 적용한다.

$$\mathrm{net}_j^t = \sum_{i=1}^{I} w_{ji} x_i^t + \sum_{j'=1}^{H} w_{jj'} h_{j'}^{t-1}$$

$$h_j^t = f_j(\mathrm{net}_j^t)$$

학습은 오류역전파 알고리듬을 사용하여 이루어진다.

1. 문맥 유닛들의 활성값을 0 으로 초기화한다. $x_i(1)=0$
2. 입력 $\mathbf{x}(t)$을 제시한다.
3. 출력값을 계산한다.
4. 오류역전파 알고리듬을 이용하여 가중치를 변경한다.
5. 위의 스텝 2로 가서 반복한다.

또 다른 형태의 재귀망은 $t-1$ 시간의 출력 활성값들 $\mathbf{y}(t-1)$을 복제하여(은닉뉴런 활성값 $h_j^{(t-1)}$이 아니라) t 시간의 입력값으로 사용하는 것이다.

$$\mathbf{y}(t) = f(\mathbf{x}(t), f(t-1); \mathbf{w})$$

뉴런의 재귀신경망은 뉴런들의 활성값이 그 출력값에 직접적으로나 간접적으로 의존하는 뉴런들을 가진 신경망이다. 재귀신경망에서는 신호 전달 경로가 사이클을 가진다. 다른 뉴런을 거치지 않고 바로 되먹임이 있으면 직접 피드백이라 하고, 다른 뉴런을 거쳐서 되먹임이 생기면 간접 피드백이라 한다. 재귀망은 과거의 모든 입력 정보가 녹아있는 내부적 상태 표현을 통해서 시간적 순서를 처리한다. 뉴런들의 출력과 그 전의 층의 뉴런들의 입력 사이에 피드백 루프가 존재함으로써 기억이 형성된다. 연결성 갱신을 동기화할 경우 이는 한단계 지연에 의해 기억 요소를 갖게 된다. 재귀신경망의 입력은 벡터들의 시퀀스로 주어진다.

$$S_p = x_0^p(1),\ x_0^p(2),\ \cdots,\ x_0^p(T_p),\ x_0^p(T_p) \in R^m,\ t = 1,\ \ldots,\ T_p$$

여기서 T_p는 p 번째 시퀀스의 길이이다.

다층퍼셉트론을 기본적인 신경망 구조로 사용할 경우 어떤 뉴런들이 피드백되는지에 따라서 다양한 재귀네트워크를 정의할 수 있다. 조단 네트워크에서처럼 퍼셉트론의 출력 유닛들을 입력으로 피드백할 수도 있고, 엘만 네트워크에서처럼 은닉유닛들을 입력으로 피드백할 수도 있다. 다수의 은닉층을 갖는 다층신경망의 경우 여러층에 걸쳐서 피드백이 존재할 수 있다. 이러한 재귀망의 행동은 시간에 대해 펼침으로써 재현될 수 있다. 즉 재귀신경망을 다층의 전방향 신경망 구조로 볼 수 있다.

p 번째 시퀀스를 인식하기 위해서 재귀신경망은 다음과 같은 정보처리 과정을 거친다. 먼저 망의 상태를 $\mathbf{x}^p(0)$로 초기화한다. 매 시간 단계마다 입력노드의 값들 $\mathbf{u}^p(t)$과 그전 시간대의 상태노드 값들 $\mathbf{x}^p(t-1)$을 이용하여 모든 뉴런들의 현재 출력값들을 계산한다. 새로운 시퀀스를 처리할 때는 매번 초기 상태를 리셋함으로써 과거 시퀀스의 메모리를 지워버린다.

재귀신경망은 입력 시퀀스를 대응되는 출력 시퀀스로 변환하는데 사용될 수 있다. 이 경우 딥재귀신경망이 L 개의 층으로 구성되어 있다고 하면 대응되는 감독학습 데이터는 다음과 같이 주어진다.

$$\begin{array}{cccccc} \mathbf{x}_0^p(1) & \mathbf{x}_0^p(2) & \ldots & \mathbf{x}_0^p(T_p-1) & \mathbf{x}_0^p(T_p) \\ \downarrow & \downarrow & \ldots & \downarrow & \downarrow \\ \mathbf{x}_L^p(1) & \mathbf{x}_L^p(2) & \ldots & \mathbf{x}_L^p(T_p-1) & \mathbf{x}_L^p(T_p) \end{array}$$

다른 경우에는 시퀀스의 중간값들에 대해서 대응되는 출력값을 모르거나 관심이 없을 수 있다. 예를 들어서, 단어들의 시퀀스로 주어지는 문장에 대해서 이 문장의 문법이 맞는지 틀리는지만 인식하고자 할 수 있다. 이 경우에 학습 데이터는 다음의 형식을 취한다.

$$\begin{array}{cccccc} \mathbf{x}_0^p(1) & \mathbf{x}_0^p(2) & \ldots & \mathbf{x}_0^p(T_p-1) & \mathbf{x}_0^p(T_p) \\ & & & & \downarrow \\ & & & & \mathbf{x}_L^p(T_p) \end{array}$$

BPTT 학습

재귀네트워크의 시간적 동력학은 시간에 대해 펼친 전방네트워크로 변환될 수 있다. 이렇게 펼친 신경망은 딥신경망과 같으며 이를 오류역전파 알고리듬을 사용하여 학습할 수 있다. 학습 데이터는 다음과 같이 입력 패턴의 시퀀스와 대응되는 출력 패턴으로 주어진다. 출력 패턴의 경우 일부 대응되는 패턴이 주어지지 않는 것을 허용한다. 시퀀스 분류 문제의 경우에는 입력 시퀀스에 대해서 시간 T_p에서의 최종 목표값만 주어진다.

$$S_p = x_0^p(1), x_0^p(2), ..., x_0^p(T_p), \quad x_0^p(t) \in R^m, \quad t = 1, ..., T_p$$
$$D_p = \mathbf{d}^p(t_1), \mathbf{d}^p(t_2), ..., \mathbf{d}^p(t_{K_p}), \quad \mathbf{d}^p(t) \in R^{n(L)}, \quad t = t_1, ..., t_{K_p} \in [1, T_p]$$

학습 데이터는 다음과 같이 주어진다.

$$L_e = \{(S_p, D_p) \mid S_p \in S(R^m), D_p \in S(R^{n(L)} \cup \{\varepsilon\}), p = 1, ..., N\}$$

여기서 는 목표 시퀀스에서 대응되는 위치의 목표값이 주어지지 않았음을 표시한다. 학습을 위해서 손실 함수를 다음과 같이 정의한다.

$$\varepsilon(L_e, \mathbf{W}) = \sum_{p=1}^{N} E_{\mathbf{W}}(S_p, D_p) = \sum_{p=1}^{N} \sum_{i=1}^{K_p} e_{\mathbf{W}}^p(t_i) = \sum_{p=1}^{N} \sum_{i=1}^{K_p} \frac{1}{2} \| \mathbf{x}_L^p(t_i) - \mathbf{d}^p(t_i) \|^2$$

이 손실함수는 학습집합에 있는 모든 학습예에 대해서서 망의 시퀀스와 목표 시퀀스의 차이를 측정한다. 이 차이는 두 벡터간의 유클리드 거리로 계산된다.

Backpropagation through Time

학습 데이터가 주어지면 딥재귀신경망은 시간오류역전파를 이용하여 학습할 수 있다. 시퀀스의 길이가 알려져 있으며 이는 전방네크워크의 깊이에 해당한다. 재귀네트워크의 똑같은 가중치 벡터들이 복제된 것이 전향네크워크의 가중치 벡터를 형성하며 이는 다른 층에서의 변경값들의 합에 해당하는 만큼 같은 크기만큼 변경된다. 즉 모든 층에서의 $\mathbf{w}(t)$ 가 같은 값으로 복제되어 유지된다.

길이 T_p 인 시퀀스를 학습하는 재귀네트워크를 N_R이라고 하자. 이를 시간에 대해서 펼친 전방 네트워크를 N_F 라고 하자. 이 두개의 망은 다음의 관계를 갖는다.

- N_F 는 시간 단위에 대응되는 N_R 의 카피를 하나의 층으로 가지고 있다.
- N_F 는 N_R 의 모든 뉴런의 카피를 가지고 있다.
- 각 시간 단위마다 층 l 의 뉴런 i 에서 층 $l+1$ 의 뉴런 j 로의 시냅스는 N_R 의 같은 시냅스에 대한 카피이다.

시간에 대한 재귀적 정보처리가 대응되는 시간에 대해 펼쳐진 전향네트워크에 해당된다. 따라서 에러 경사도 값은 학습은 전향 네트쿼크에서와 같이 계산될 수 있다. 펼쳐진 네트워크는 다층신경망이고 여기에서 에러경사도는 오류역전파 알고리듬을 그대로 쓸 수 있다. 단, 재귀 네트워크의 카피가 펼쳐진 네트워크에서 가중치값들을 공유해야 한다는 제약조건이 지켜져야 한다. 이 제약은 네트워크를 펼치는 동안 각 카피에 대해서 각 가중치에 대한 경사도를 축적함으로써 구현될 수 있다. BPTT 는 시간에 대해서 국부적이지 않다. 즉 각 시간 단위마다 모든 뉴런의 출력값들을 저장함으로써 전체 시퀀스가 처리되어야 한다. 그러나 이 알고리듬은 공간에 대해서는 국부적이다. 즉 각 뉴런은 구부적인 변수만을 이용한다. BPTT 는 MLP 에 적용되는 표준 오류역전파 알고리듬을 모듈화해서 구현할 수 있다.

7.3 딥재귀신경망(DRNN)

재귀신경망은 전통적인 칼만필터와 같은 동적시스템 모델과 유사하다. 즉 시스템의 상태변수와 출력변수들은 다음의 식에 의해서 갱신된다.

$$\boldsymbol{f}(t) = \boldsymbol{f}(\mathbf{h}(t), \mathbf{x}(t))$$
$$\mathbf{h}(t) = \boldsymbol{g}(\mathbf{h}(t-1), \mathbf{x}(t))$$

최근에는 재귀신경망의 은닉층을 여러층 쌓은 딥재귀신경망 모델이 제안되었다. 그림 7.6 은 딥재귀신경망의 구조이다. 입력층에 이어 3 층의 은닉층을 가지고 최종적으로 출력층을 갖는 재귀신경망 구조이다.

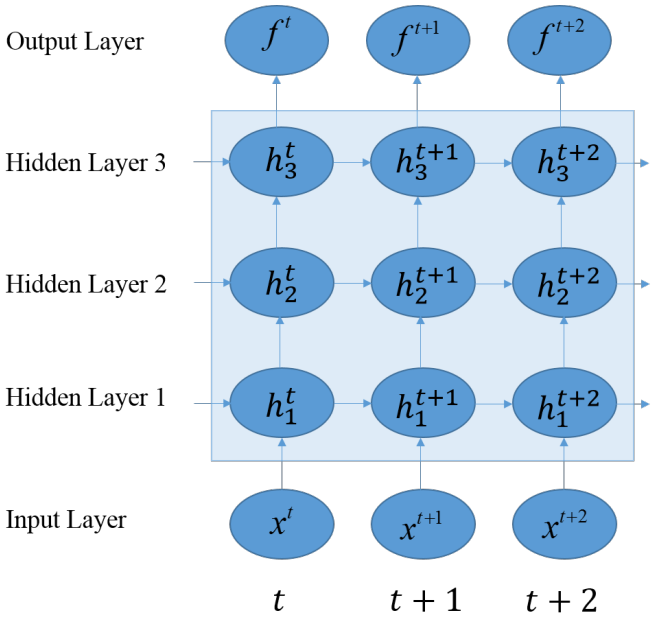


그림 7.6 딥재귀신경망

7.4 장단기메모리

재귀신경망을 위한 표준 학습 알고리듬은 긴 시간 지연을 허용하지 않는다. BPTT 에서의 시간에 대한 역전파 에러 신호가 급격히 증가되거나 급격히 줄어드는 문제가 있다. 에러 신호가 지수함수적으로 희석되는 문제를 살펴보자. RNN 에서는 에러 경사도가 급격히 줄어드는 문제가 있다.

시간 t 에서 u 유닛 로부터 q 스텝 앞에 있는 v 유닛으로의 에러는 다음과 같이 스케일된다.

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \begin{cases} f_v'(\text{net}_v(t-1))w_{vu} & q=1 \\ f_v'(\text{net}_v(t-1))\sum_{l=1}^{n} \frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} w_{vl} & q>1 \end{cases}$$

$l_0 = u$ 로부터 $l_q = v$ 로의 경로에 대해서 모두 합하면

$$\frac{\partial \vartheta_v(t-q)}{\partial \vartheta_u(t)} = \sum_{l_1=1}^{n} \cdots \sum_{l_{q-1}=1}^{n} \prod_{m=1}^{q} f_{l_m}'(\text{net}_{l_m}(t-m)) w_{l_m l_{m-1}}$$

만약 오른쪽의 항의

$$\left| f_{l_m}^{'}(\text{net}_{l_m}(t-m))w_{l_m l_{m-1}} \right|$$

절대값이 1 보다 크면 에러가 q에 대해서 지수함수적으로 증가하고, 이 값이 1 보다 작으면 에러가 q 에 대해서 지수함수적으로 작아진다. 어떻게 에러 경사도가 지수함수적으로 줄어드는 것을 방지할 수 있을 것인가? 시간에 대해서 꾸준히 활성화되는 선형적인 활성화 함수를 필요로 한다. LSTM 은 가중치 1 을 갖는 identity function 을 사용한다.

$$f_j^{'}(\text{net}_j(t))w_{jj} = 1.0$$

이를 상수 에러 카루셀(CEC)이라고 한다.

CEC 는 두 가지 이슈가 있다. 하나는 입력 가중치 문제이다. 유닛 j가 만약 하나의 외부 입력 i를 갖는다고 하면, i 는 j를 on 시키고 계속 활성화를 유지하려고 한다면 w_{ij} 는 입력을 저장하고 (j를 on 시키기 위해), 입력을 보호해야 한다(j가 off 되는 것을 방지하기 위해). 이 문제는 학습을 어렵게 만든다. 한 가지 방법은 입력 게이트를 추가하여 CEC 를 부적합한 입력으로부터 보호하는 것이다. 두 번째 문제는 출력 가중치 문제이다. 유닛 j가 만약 하나의 외부 출력 k 를 갖는다면, w_{kj} 는 가끔은 CEC 로부터 출력값을 얻고, 가끔은 j가 k를 방해하지 않도록 방지해야 한다. 이 또한 학습을 어렵게 만든다. LSTM 은 출력 게이트를 추가해서 언제 저장된 데이터를 읽을 것인지를 제어한다.

LSTM 메모리 셀은 다음과 같다.

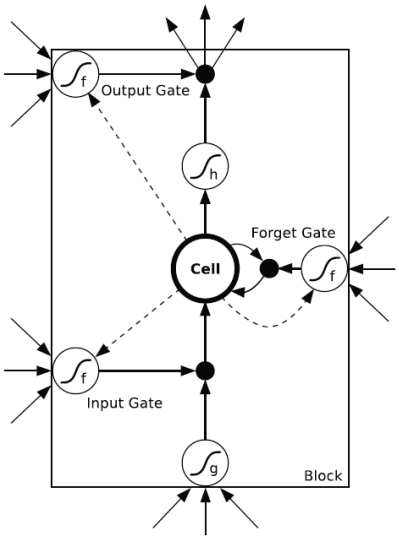


그림 7.7 LSTM 메모리 셀

LSTM 은 후에 망각 게이트를 셀에 추가했다. LSTM 은 셀을 메모리 블록에 연결한다. 다수의 메모리 셀에 대해서 같은 입력, 출력 게이트를 사용한다. 그림 7.7 의 구조는 메모리 블록들로만 구성된 은닉 층을 가지고 서로 완전히 연결된 구조이다. 보통의 은닉 뉴런도 사용할 수 있다. 학습은 BPTT 에 의해서 메모리 셀을 시간에 대해서 펼쳐서 에러를 역전파하면서 학습할 수 있다.

때로는 메모리 셀이 상수 편향값으로 오용되기도 한다. 이 문제를 해결하는 방법은 에러가 줄어들 때까지 학습을 시킨 후에 메모리 셀을 추가하는 것이다. 또 다른 방법은 오용되는 상태를 멀리하여 출력값을 억제하도록 출력 게이트 편향을 조절하는 것이다. 또 다는 문제는 내부 상태가 떠다는 것이다. 만약 입력값들이 대부분 같은 부호이면, 메모리 내용이 시간에 대해서 떠다니는 경향이 있다. 이는 에러 경사도가 사라지도록 한다. 이에 대한 간단한 해는 입력 게이트에 대한 초기 편향값을 0 에 가까지 주는 것이다.

새로운 입력들이 은닉층의 활성화를 덮어쓰고 처음 입력을 잊어버리게 되면서, 시간이 감에 따라 네트워크의 민감도가 줄어든다. 이 문제는 은닉층에 시그모이드 셀을 사용하는 대신에 LSTM 블록을 사용함으로써 일부 해결될 수 있다. LSTM 블록은 임의의 오랜 기간동안 메모리를 유지할 수 있고 필요시 또한 잊어버릴 수도 있도록 해준다. LSTM 은 다음의 요소들로 구성된다.

- w_{ji} : 유닛 i 로부터 유닛 j 로의 연결 가중치
- net_j^t : 시간 t 에서의 유닛 j 의 총 입력
- h_j^t : 시간 t 에서의 유닛 j 의 활성값
- i, f, o : 블록의 입력, 망각, 출력 게이트의 인덱스
- c : 셀의 인덱스
- w_{ci}, w_{cf}, w_{co} : 셀 c 로부터 입력, 망각, 출력 게이트로의 peephole 가중치
- s_c^t : 셀 c 의 시간 t 에서의 상태
- f, g, h: 게이트의 활성화 함수, 입력과 출력 게이트의 활성화 함수

입력뉴런수, 은닉뉴런수, 출력뉴런수를 각각 I, H, K 라고 하자. LSTM 은 BPTT 로 학습할 수 있다. 전향 패스에서는 표준 RNN 처럼 길이 T 인 입력 시퀀스에 대해서 $t=1$ 에서 출발하여 t 를 증가하면서 변경식을 적용한다. 후향 패스에서는 $t=T$ 에서 출발하여 $t=1$ 까지 줄이면서 에러 경사도를 계산한다. 최종 가중치 경사도값은 각 시간 단계에서 구한 경사도값들의 총합을 계산함으로써 구해진다.

재귀신경망은 뉴런의 출력이 다시 입력으로 되먹임되는 신경망 구조이다. 이는 이전 시간 $t=1$ 에서의 은닉 노드가 현재 시간 t 에서의 은닉 노드에 영향을 주는 동적인 모델이다.

연속적인 입력 데이터를 $\mathbf{x}$, 연속적인 출력 데이터를 $\mathbf{y}$, 연속적인 은닉상태 값을 $\mathbf{h}$ 라 하면

$$\mathbf{x} = (x_1, x_2, ..., x_n)$$
$$\mathbf{y} = (y_1, y_2, ..., y_n)$$
$$\mathbf{h} = (h_1, h_2, ..., h_n)$$

이로부터 은닉 상태 값 $\mathbf{h}_t$ 과 출력 값 $\mathbf{y}_t$ 은 다음 식에 의해서 구해진다.

$$h_t = H(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{ho}h_t + b_o$$

H 은 은닉 노드의 활성화 함수이고, W_{ih} 는 입력노드에서 은닉 노드로의 가중치 행렬, b_h 는 바이어스 값이다.

각 게이트와 셀 상태의 값은 다음과 같이 계산된다.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

참고문헌

- [1] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

- [2] L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.

- [3] W.-Y. Chen, Y.-F. Liao, and S.-H. Chen. Speech recognition with hierarchical recurrent neural networks. *Pattern Recognition*, 28(6):795 – 805, 1995.

- [4] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.

- [5] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. *Advances in Neural Information Processing Systems*, 8:493–499, 1996.

- [6] S. Ferna´ndez, A. Graves, and J. Schmidhuber. Sequence labelling in structured domains with hierarchi- cal recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, Hyderabad, India, January 2007.

- [7] J. Garofolo, N. I. of Standards, T. (US, L. D. Consortium, I. Science, T. Office, U. States, and D. A. R. P. Agency. *TIMIT Acoustic-phonetic Continuous Speech Corpus.* Linguistic Data Consortium, 1993.

- [8] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *To appear in ICASSP 2013*, 2013.

- [9] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

- [10] G. E. Hinton. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

- [11] S.HochreiterandJ.Schmidhuber.Longshort-termmemory.*Neuralcomputation*,9(8):1735–1780,1997.

- [12] M. Hutter. The human knowledge compression prize, 2006.

- [13] H. Jaeger. Long short-term memory in echo state networks: Details of a simulation study. Technical report, Jacobs University, 2012.

- [14] M. Mahoney. Adaptive weighing of context models for lossless data compression. *Florida Tech., Mel- bourne, USA, Tech. Rep*, 2005.

- [15] J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the*

27th International Con- ference on Machine Learning, pages 735–742, 2010.

- [16] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Pro- ceedings of the 28th International Conference on Machine Learning*, volume 46, page 68. Omnipress Madison, WI, 2011.

- [17] A. Mohamed, G. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14–22, 2012.

- [18] C. E. Shannon. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64, 1951.

- [19] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1017–1024, 2011.

- [20] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine learning*, pages 1096–1103, 2008.