Lstm

**1)배경**

**기존 rnn이 긴 간격의 정보를 저장하는데 역전파가 시간도 오래 걸리고, 기울기 소실 문제가 발생하여(장기의존성 문제) 이를 해결하기 위해 hochreiter의 1991년 연구를 참고해 고안해냈다**.

**2)구조**

**Cell state**

**기존 rnn과 다르게 추가된 이전 정보를 계속해서 전달해주는 역할을 하는 state이다.**

**(장기의존성문제를 해결하는 핵심)**

**\*forget gate\***

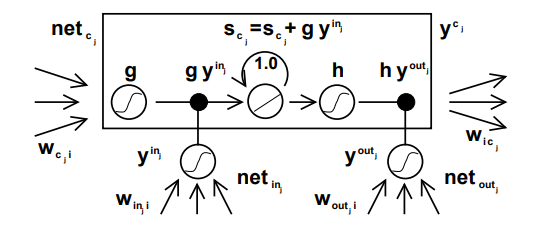
**이전 hidden state에서 받아온 정보중 필요한 정보만 걸러서 cell state에 넣어주는 역할**

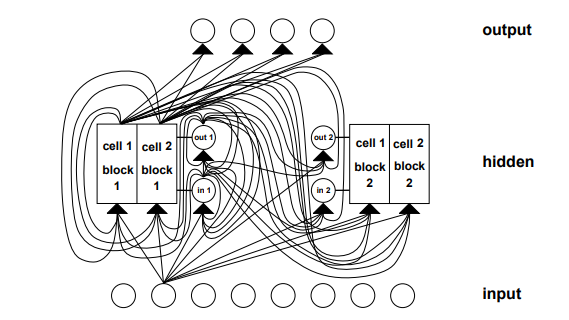
**Input gate**

**현재 시점에서 얻은 정보에 기존 cell state에 추가해주는 역할**

**Output gate**

**Input gate에서 얻은 정보 중에 hidden state에 얼마나 전달할 건지를 정하는 역할**





**5)**

**Experiment1**

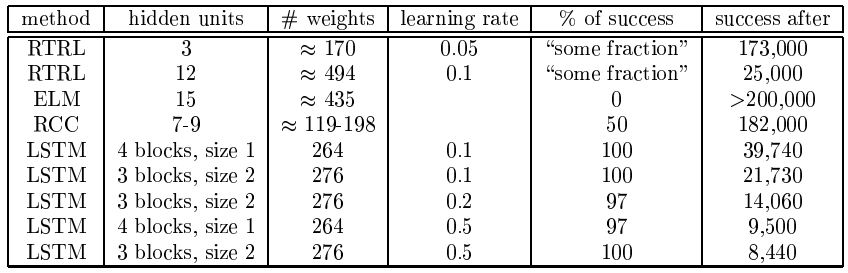
**실험데이터**

7개의 input unit과 output unit을 사용했으며,.

Fahlman에 따라 256개의 훈련데이터 문자열과 256개의 테스트데이터 문자열을 사용했다.

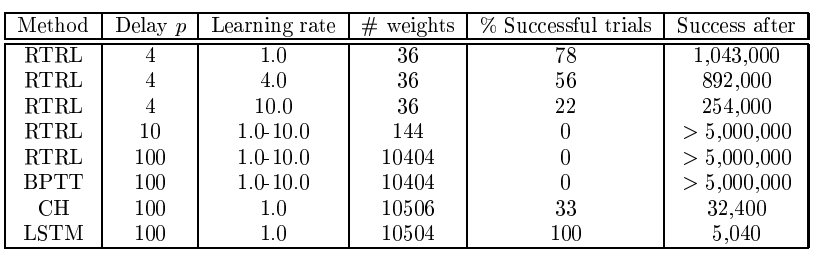
훈련데이터 셋은 랜덤하게 생성되었고 훈련 예시도 그 중에서 랜덤하게 선택되었다. 테스트 데이터도 훈련데이터와 마찬가지로 선택되었다. 또한 훈련데이터와 테스트데이터는 겹치는 게 없이 사용된다.

처음엔 모든 활성화 초기값이 0이다. 만약 문자열 글자들이 테스트 데이터와 훈련데이터 모두에서 일치되게 예측된다면 이 실험은 성공으로 고려된다.

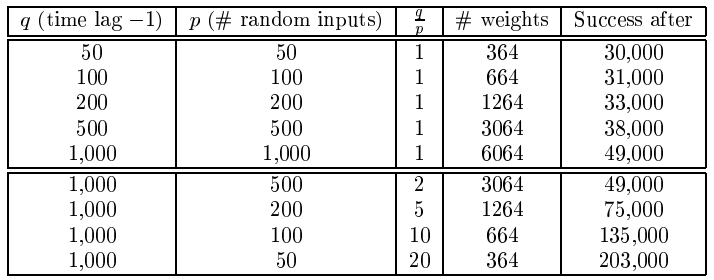


결과: 다른 방법에 비해서 빠르고 정확하다

**Experiment2**



결과: RTRL, BPTT가 delay p를 10과 100으로 각각 늘렸을 때 문제를 풀지 못하는 반면에 LSTM이 학습도 빠르고 정확한 결과를 보여줬다.



**Experiment3**

**실험데이터**

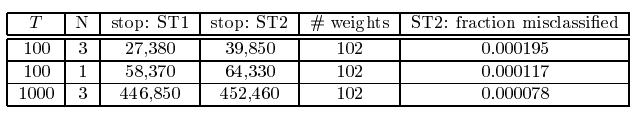
모든 가중치가 -0.1에서 0.1사이값으로 랜덤하게 초기화되어있다. 첫번째 INPUT GATE BIAS는 -1, 두번째는 -3, 세번째는 -5로 초기화되었다. 첫번째 OUTPUT GATE는 -2, 두번째는 -4, 세번째는 -6으로 초기화되었다.

학습율은 1.0이면 모든 활성화는 시작과 함께 0으로 초기화한다.

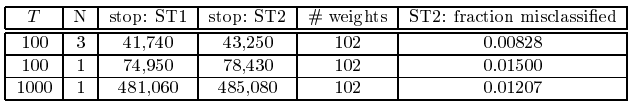
또한 아래 기준에 따라 학습을 멈추게 된다(문제가 해결되었다고 판단)

ST1: 랜덤하게 선택된 256개 데이터가 잘못 분류되지 말아야 한다.

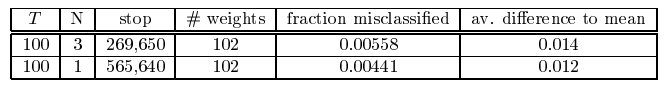
ST2: ST1을 만족하며, MAE가 0.01아래여야 한다. ST2의 경우 랜덤한 2560개로 구성된 추가 데이터 셋은 잘못 분류된 부분을 결정하는데 사용된다.



결과: 문제가 너무 간단해서 random weight guessing으로 더 빨리 해결되었다.



결과: 노이즈를 추가했지만 노이즈에 의해 불안한 결과가 나왔다.

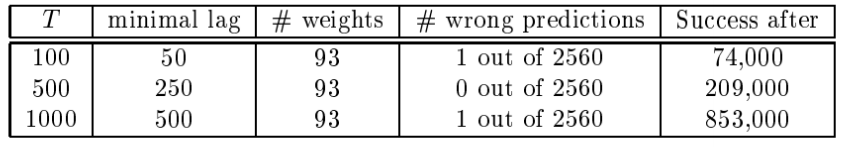


결과: 첫번째 테이블에 노이즈 추가된 문제.  random weight guessing으로 해결되지 않지만 LSTM으로 해결되었다.

**Experiment4**

**실험데이터**

학습율은 0.5이며 학습은 평균 오차값이 0.01이 아래이고 최근 2000개 데이터가 올바르게 처리되면 중단한다**.**



결과: 기존 RNN으로 해결하지 못한 문제를 LSTM으로 해결함, 장기의존성을 해결했다.

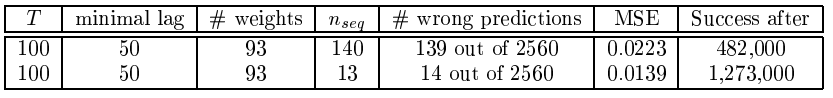
**Experiment5**

**실험데이터**

학습율은 0.1이다. 최근 2000개 훈련데이터가 절대오차가 0.04를 초과할 때의 nseq값보다 작은 조건인 nseq값이 140일때와 13일 때 두번 테스트한다.

nseq=140은 관련된 input들의 저장된 데이터를 학습하는데 충분하지만 정확한 결과를 도출해내기는 어렵다.

하지만 nseq=13은 꽤 만족할만한 결과를 도출해낸다.



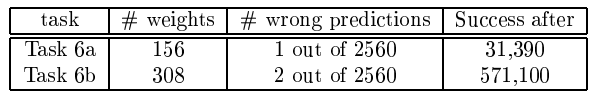
결과: LSTM이 non-integrative information processing도 가능하다는 것을 보여주었다.

**Experiment6**

**실험데이터**

학습율은 0.5(0.1)이다. 학습은 일단 평균오차가 0.1이하로 떨어지며 최근 2000개 데이터가 올바르게 분류되면 중단한다. 모든 가중치는 -0.1과 0.1사이값으로 초기화하며 6a의 경우 첫번째 input gate bias를 -2, 두번째를 -4로 초기화한다. 6b의 경우 첫번째 input gate bias를 -2, 두번째를 -4로 초기화하며, 세번째는 -6으로 초기화한다.

\*괄호 밖은 6a조건, 괄호 안은 6b조건



결과: Lstm이 긴 시간 간격에 따라 전달된 정보를 추출할 수 있다는 것을 보여준다.

**Lstm parameters by pytorch**

**Input\_size**: x의 feature의 수(열의 개수)

**Hidden size**: hidden state의 차원의 개수

**Num\_layers**: hidden state 레이어 개수(기본값은 1)

**Bias:** 바이어스 가중치 활성화 여부, 기본값은 true(활성화하겠다는 의미)

**Batch\_first::** output의 shape을 정하는 것 true면(batch, seq, feature), False면 (seq,batch,feature), 기본값은 false

**dropout:** 드롭아웃 비율. dropout이 0이 아니라면, last layer를 제외한 나머지 LSTM layer에 dropout을 적용한다 기본값:0

drop-out은 특정변수만 과도하게 집중 학습하여 생길 수 있는 과적합을 방지하기 위해서 사용한다. 비율에 따라 랜덤으로 뉴런을 제거하는데 보통 0.5를 많이 사용한다

**Bidirectional:** 양방향 lstm여부, 기본값 false(단일방향의 의미)

**Proj\_size:** 0보다 큰경우, 해당크기의 projection을 갖는 lstm을 사용한다. 기본값:0