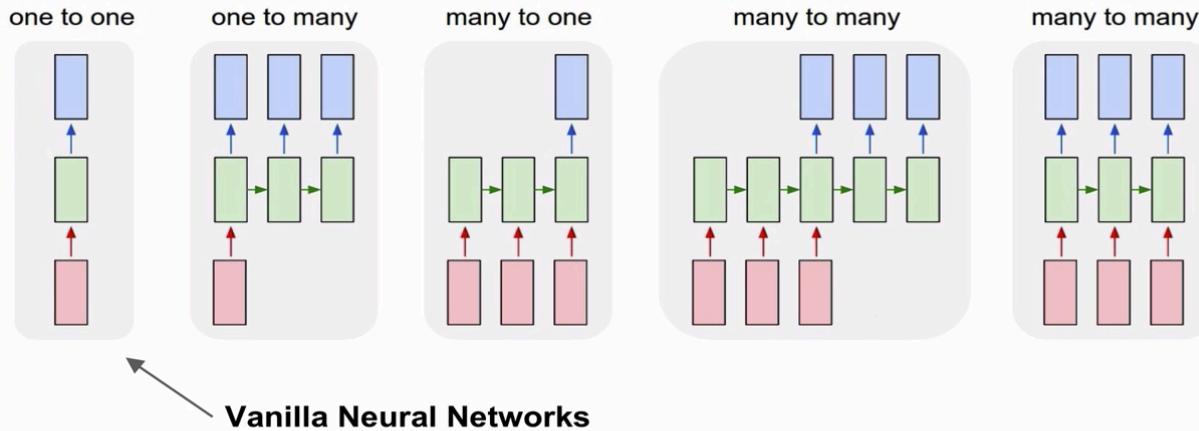


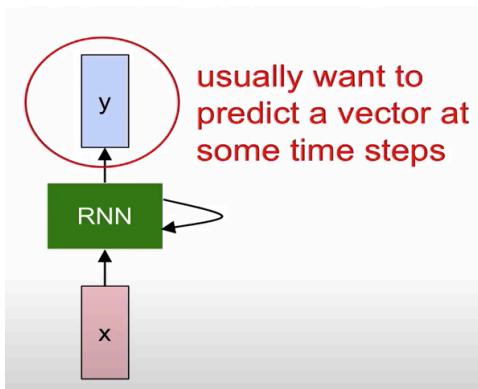
RNN

정재영

Recurrent Networks offer a lot of flexibility:



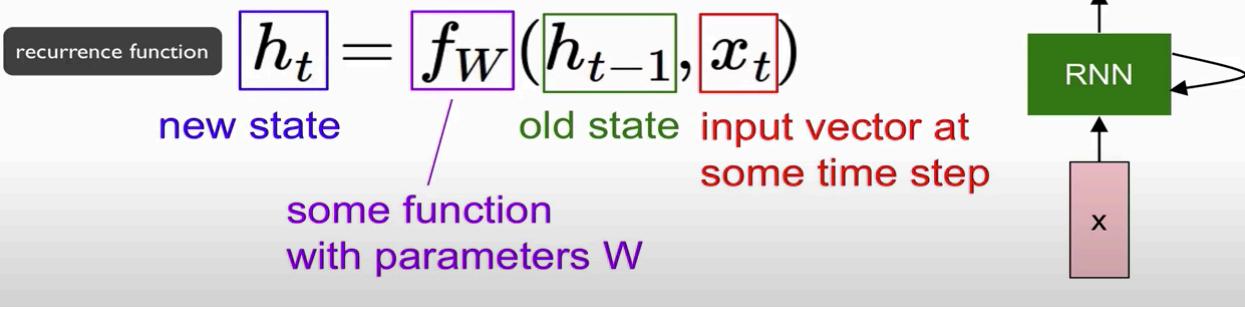
우리가 기존에 알아보았던 신경망은 위 그림의 맨 왼쪽에 있는 형태였다. RNN은 위 그림 모든 형태로 이루어질 수 있다. 2번째 그림은 이미지 캡셔닝에서 사용한다. 3번째는 감정 분류에서 사용한다. 가령 단어들로 구성된 시퀀스가 있을 때, 감정이 부정적인지 긍정적인지를 분류해낸다. 4번째 그림은 기계 번역에서 적용될 수 있다. 가령 영어 단어를 한국어 단어로 번역하는 경우가 그것이다. 마지막은 비디오 분류에서 사용된다. 이것은 현재의 프레임과 더불어 이미 지나간 과거 프레임에 대한 함수여야 한다. 즉, 여기서는 현재 프레임 + 지나간 프레임의 함수로 RNN은 이루어진다.



매 타임 스텝마다 x 인풋을 받는 RNN이 있다고 해보자. RNN은 내부적으로 어떤 상태를 갖는데, 우리는 이것을 어떤 함수로 (매 시간마다 인풋을 받는 것에 대한 함수)로 변형해줄 수 있다. 여기서 가중치가 업데이트 됨으로써 새로운 인풋이 들어올 때마다 다른 산출값을 내놓을 수 있다.

그래서 우리는 특정 타임 스텝에서의 예측을 얻기를 원한다.

We can process a sequence of vectors x by applying a recurrence formula at every time step:

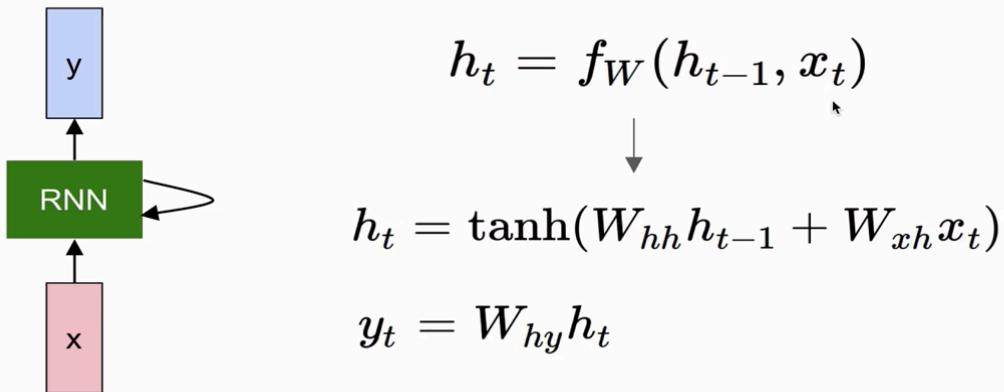


따라서 우리는 위 같은 recurrence 함수를 적용할 수 있게 된다. 이것을 적용함으로써, 그 시퀀스를 처리할 수 있게 된다. 새로운 상태를 h_t 라고 해보자. 그리고 이것은 파라미터 w 에 대한 함수와 직전 시점에 은닉 스텝과 현재 시점에 인풋 인자로 이루어진다. 이것은 파라미터 w 에 대한 함수이니까 w 를 변경하게 된다. 그러므로 우리는 이 RNN이 우리가 바라는 특정 행동을 얻을 수 있도록 가중치 값을 학습시켜 나가게 된다.

주의할 점은 매 타임 스텝마다 동일한 함수에서, 동일한 모수 셋이 사용되어야 한다. 그래야 인풋의 시퀀스의 사이즈, 아웃풋의 시퀀스의 사이즈에 영향을 받지 않는다.

바닐라 RNN

The state consists of a single “hidden” vector h :



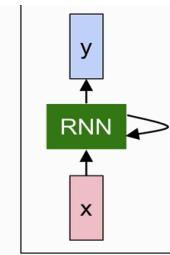
- 상태가 단일의 히든 벡터 h 로만 구성.
- 직전 상태와 현재의 입력값의 함수로 표현된다.
- 두번째 식을 보라: x_t 는 가중치 값이 x 에서 h 로 가니까 W_{xh} 를 곱해주었고, 직전의 상태 h_{t-1} 은 현재의 히든 층과 직전의 히든 층의 영향을 받기 때문에 가중치를 h_h 사이즈로 표현을 했다. 그리고 이렇게 더해진 것에 \tanh 를 적용하는 것이 현재의 state가 되는 것이다. 즉, 이것은 어떤 history와 현재의 입력값을 통해 상태가 변환된다는 것을 알려주고, y 값은 그러면 W_{hy} 가 된다.

사례

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

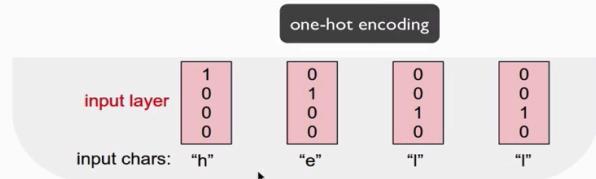


우리가 학습시키려는 것은 “hello”라는 단어. 여기에는 4 가지의 캐릭터가 들어가 있다. 우리는 이 모델에 h 다음에는 e가 나와야 한다는 것 등을 학습시켜야 한다.

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



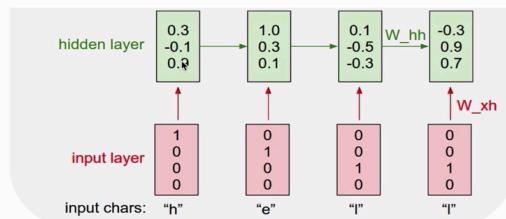
4 개의 캐릭터로 이루어졌기 때문에 위 같이 구성되고, 이 그림은 ‘hello’를 원핫인코딩으로 표현해주고 있다.

Character-level language model example

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

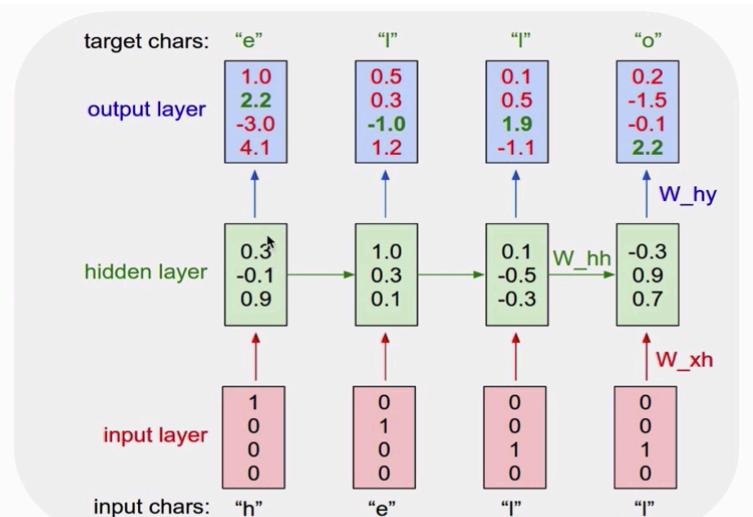


최초에 h를 히든 값으로 넣어주면 위 같은 임의로 가정된 히든 층에서 어떤 출력값이 나온다. 주목할 점은 위로 가는 것은 W_{xh} , 옆으로 가는 건 W_{hh} 이다. (인풋 레이어)

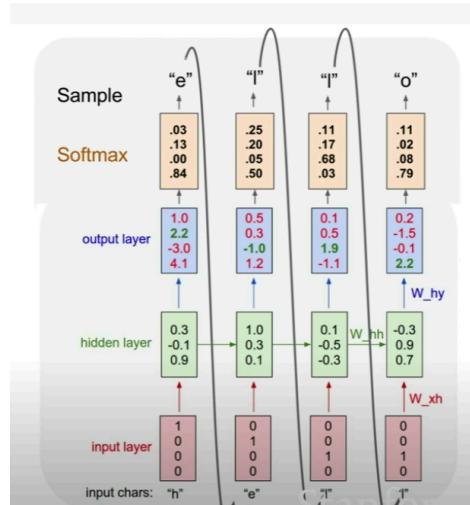
Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



아웃풋 레이어가 나오면, 위 같이 파란 박스 안의 초록색으로 정답값이 표현된다. (아웃풋 레이어) 가령 위 그림에서 “h”에 해당하는 열은 아웃풋 레이어에서 정답값이 [1000] = [‘h’, ‘e’, ‘l’, ‘o’]에서 ‘e’를 예측하고 있다. 따라서 제대로 예측하고 있다. 나머지도 마찬가지로 제대로 예측하고 있음을 확인해보라. (두번째는 제대로 예측이 안되는 것 같다)

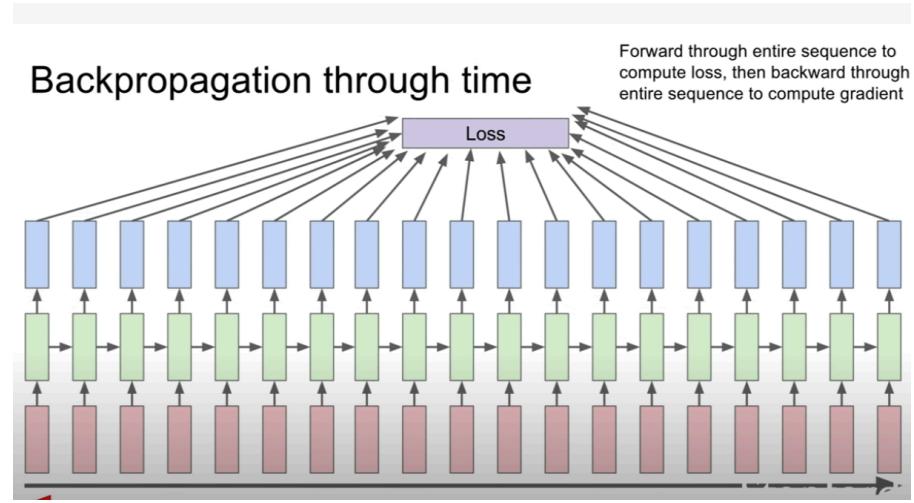


마찬가지로, 아웃풋 레이어를 통해 loss를 구하고, 그 뒤에 역전파를 계산한다.

이 W_{hh} , W_{xh} , W_{hy} 등이 여러 번 쓰인다는 것은 위 그림에서 쉽게 확인할 수 있다. 앞에서 보았듯이, “모든 타임 스텝에서 동일한 recurrence 함수와 동일한 모수가 사용된다.” 그래서 각각의 과정에서 W_{hy} , ... 등에서 모두 동일한 모수와 재발생 함수가 사용된다. 그래서 인풋, 아웃풋 시퀀스의 개수에 상관없이 처리가 가능하다.

RNN에서의 역전파 문제

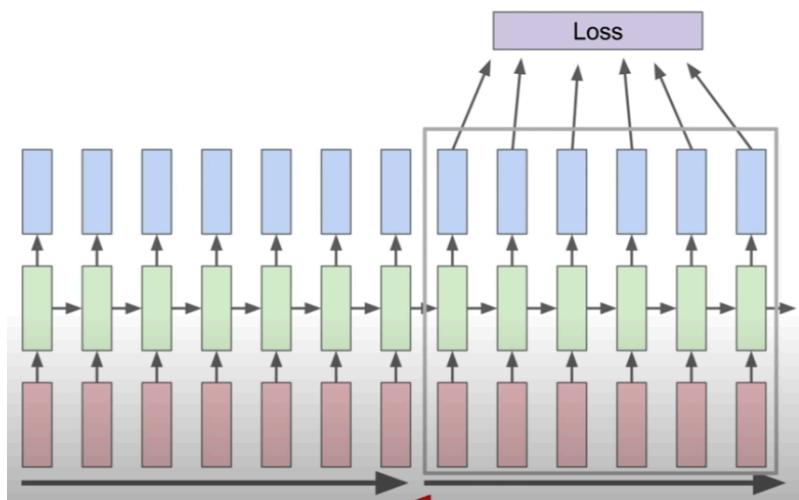
위 사례처럼 단지 4 개의 문자에 대해서만 모델에게 학습을 시킨다면, 문제는 없을 것이다. 하지만 우리는 한 번에 많은 단어를 처리해야 할 수도 있다.



우리는 모든 시간 스텝에서 순전파를 수행하여 손실을 구한 뒤, 모든 시간 스텝에서 역전파를 수행하여 gradient를 구하려 할 것이다. 하지만 이것은 문제가 될 수 있다. 매우 긴 시퀀스를 학습시키고자 한다면, 가령 위키피디아 전체의 텍스트를 신경망 언어 모델에 학습시키고자 한다면, 이것은 모든 시간에서 그래디언트를 구해야 하기 때문에 매우 느릴 것이다. 설상가상으로 이 모델은 수렴하지 않을 것이다.

잘려진 역전파 Truncated Backpropagation

실제에서 많이 사용되는 것은 이 잘려진 역전파이다. 이것은 전체 시퀀스가 아닌 일부의 미니 배치만을 이용하는 방법이다. 이것을 사용하면, 입력값이 매우 크고 잠재적으로 무한하더라도, 그래디언트를 구할 때 위 같은 문제가 발생하지 않는다. 그러면, 우리는 RNN을 다툼에 있어서, 순전파 시에는 모든 시간 스텝에 대해서 수행 할지라도, 역전파 시에는 오직 선택된 배치만을 가지고 수행한다.



코드로 이해하기 (잘려진 역전파로 RNN 구현)

[min-char-rnn.py](#) gist

Data I/O

```
"""
Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
BSD License
"""

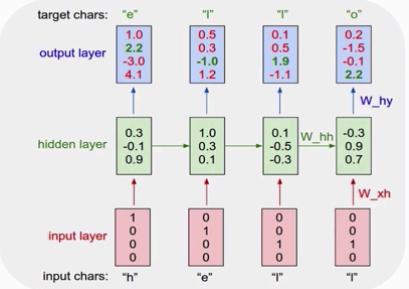
import numpy as np
# data I/O
data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }
```

chars와 index를 연관시켜준 것에 주목해보자. 이것을 통해 전자를 후자로, 후자를 전자로 변환할 수 있게 미리 만들어준다.

[min-char-rnn.py](#) gist

Initializations

```
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
```



recall:

여기서는 히든 사이즈가 100, seq_length는 가령 입력된 시퀀스 길이가 매우 길다면, 우리가 이것을 한꺼번에 올리고 역전파를 하는 것은 불가능하니까. 위 같이 제약된 청크 25를 지정해준다.

그 다음에 W_{xh} , W_{hh} , W_{hy} ... 등을 랜덤을 이용하여 정의해주고, b_h , b_v 를 이용하여 바이어스를 더해준다.

1. 메인 루프

```
34     smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
35     while True:
36         # prepare inputs (we're sweeping from left to right in steps seq_length long)
37         if p+seq_length+1 >= len(data) or n == 0:
38             hprev = np.zeros((hidden_size,1)) # reset RNN memory
39             p = 0 # go from start of data
40             inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
41             targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
42
```

메모리를 리셋, 0 - 25까지 인풋에 정수값 넣어주고, 타겟에는 1 - 25+1(26)까지 (1 개를 더 준 이유는 예측값을 넣어주는 공간을 부여)

```
91     targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
92
93     # sample from the model now and then
94     if n % 100 == 0:
95         sample_ix = sample(hprev, inputs[0], 200)
96         txt = ''.join(ix_to_char[ix] for ix in sample_ix)
97         print '----\n%s\n----' % (txt, )
98
99     # forward seq_length characters through the net and fetch gradient
```

샘플링해주기.

```
96     txt = ''.join(ix_to_char[ix] for ix in sample_ix)
97     print '----\n%s\n----' % (txt, )
98
99     # forward seq_length characters through the net and fetch gradient
100    loss, dWxh, dWhh, dWhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
101    smooth_loss = smooth_loss * 0.999 + loss * 0.001
102    if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
103
104    # perform parameter update with Adagrad
```

손실 구하는 코드. 이것을 구할 때는 인풋, 타겟 전 단계의 은닉 상태 벡터이다. 이것은 25 개의 배치로 진행하고, 이들 중 가장 끝 쪽에 존재하는 것을 계속 추적한다. 역전파 시 은닉 상태 벡터가 각 배치에서 배치로 이어질 때 잘 역전파할 수 있도록 구해준다.

```
93
94     # perform parameter update with Adagrad
95     for param, dparam, mem in zip([Wxh, Whh, Why, bh, by],
96                                   [dWxh, dWhh, dWhy, dbh, dby],
97                                   [mWxh, mWhh, mWhy, mbh, mby]):
98         mem += dparam * dparam
99         param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
100
101    p += seq_length # move data pointer
```

2. 손실 함수

순전파:

```

27 def lossFun(inputs, targets, hprev):
28     """
29     inputs,targets are both list of integers.
30     hprev is Hx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
34     hs[-1] = np.copy(hprev)
35     loss = 0
36     # forward pass
37     for t in xrange(len(inputs)):
38         xs[t] = np.zeros((vocab_size, 1)) # encode in 1-of-k representation
39         xs[t][inputs[t]] = 1
40         hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
41         ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
42         ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
43         loss += -np.log(ps[t][targets[t], 0]) # softmax (cross-entropy loss)
    
```

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$

$y_t = W_{hy}h_t$

Softmax classifier

인풋만 1, 나머지를 0으로 해주기 위해 원핫인코딩 해준다. 인풋 개수는 25. h_t , y_t 를 구한다.

역전파:

```

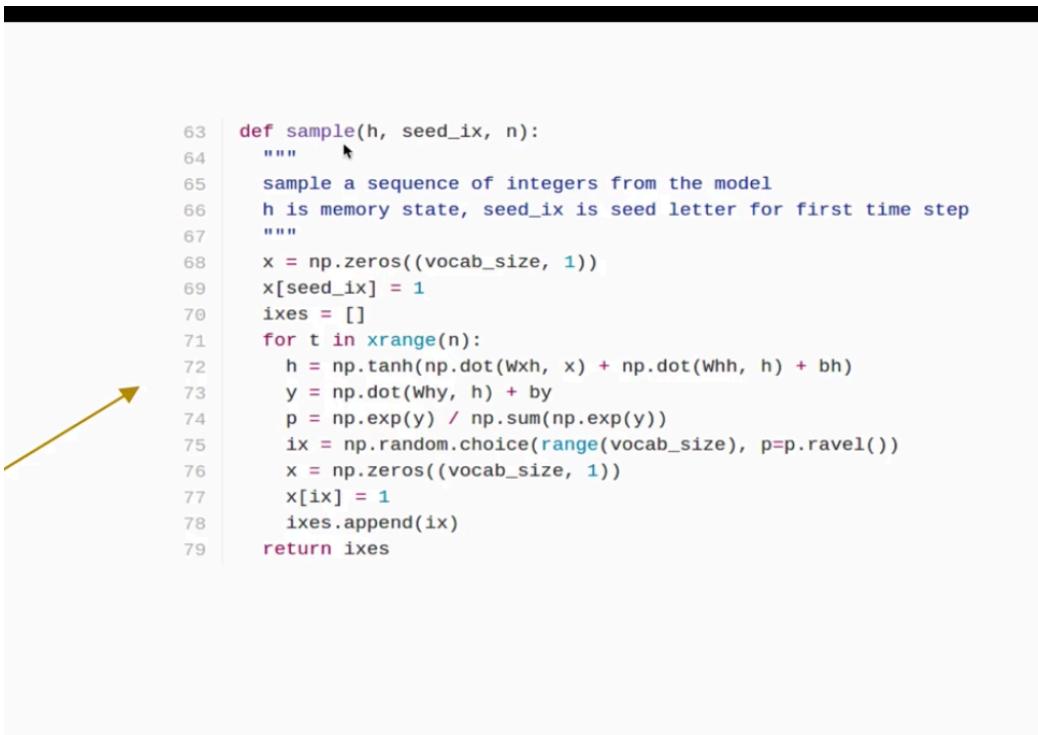
44     # backward pass: compute gradients going backwards
45     dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
46     dbh, dyb = np.zeros_like(bh), np.zeros_like(by)
47     dhnext = np.zeros_like(hs[0])
48     for t in reversed(xrange(len(inputs))):
49         dy = np.copy(ps[t])
50         dy[targets[t]] -= 1 # backprop into y
51         dWhy += np.dot(dy, hs[t].T)
52         dyb += dy
53         dh = np.dot(Why.T, dy) + dhnext # backprop into h
54         ddraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
55         dbh += ddraw
56         dWxh += np.dot(ddraw, xs[t].T)
57         dWhh += np.dot(ddraw, hs[t-1].T)
58         dhnext = np.dot(Whh.T, ddraw)
59     for dparam in [dWxh, dWhh, dWhy, dbh, dyb]:
60         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61     return loss, dWxh, dWhh, dWhy, dbh, dyb, hs[len(inputs)-1]
    
```

recall:

리버스 함수로 25에서 1까지 진행시키고, 각 단계에서 소프트맥스 등을 이용해서 그래디언트를 구해나가게 한다.

Clip 함수: -5와 5를 넘어서는 범위에 있는 것을 이 안으로 들어오게 클립핑하는 것인데, 이것을 사용하는 이유는 그래디언트가 explode 되는 단점을 RNN이 갖기 때문에 이것에 대해 조치를 취하기 위해 사용.

3. 샘플 함수

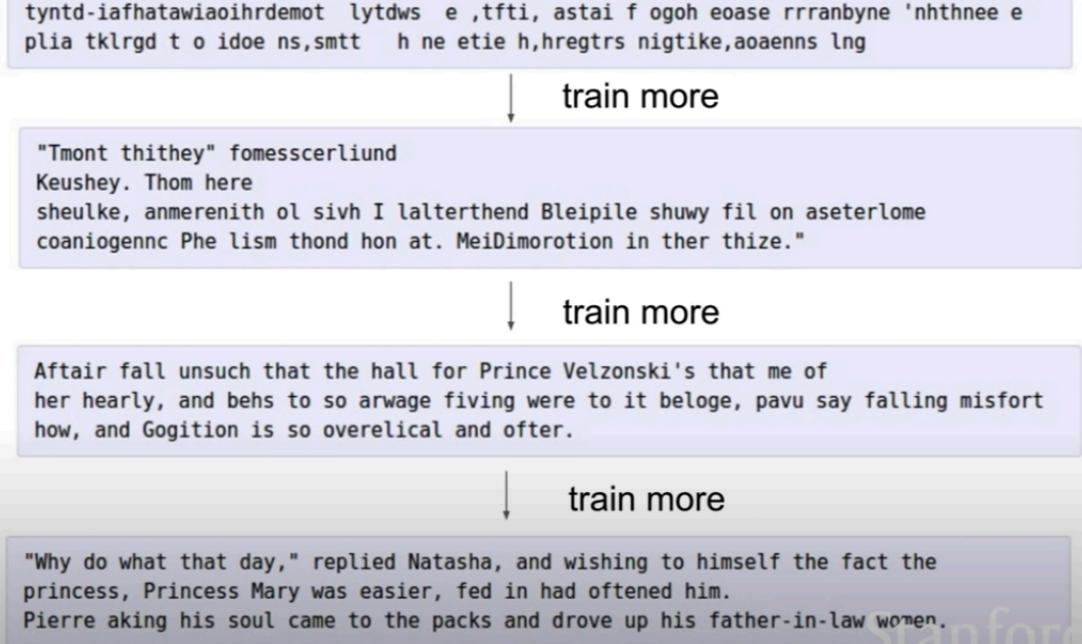


```
63 def sample(h, seed_ix, n):
64     """
65     sample a sequence of integers from the model
66     h is memory state, seed_ix is seed letter for first time step
67     """
68     x = np.zeros((vocab_size, 1))
69     x[seed_ix] = 1
70     ixes = []
71     for t in xrange(n):
72         h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
73         y = np.dot(Why, h) + by
74         p = np.exp(y) / np.sum(np.exp(y))
75         ix = np.random.choice(range(vocab_size), p=p.ravel())
76         x = np.zeros((vocab_size, 1))
77         x[ix] = 1
78         ixes.append(ix)
79     return ixes
```

이 함수에서는 실제로 RNN이 글자를 학습한 것과, 통계에 기반하여서 새로운 텍스트 데이터를 생성하도록 해 준다.

- (1) 원핫인코딩
- (2) 리커런스 식을 이용
- (3) 확률 구하기
- (4) 확률분포로부터 샘플링 초이스
- (5) 다시 원핫인코딩
- (6) 다음 타임 스텝으로 feed해준다

응용: 세익스피어의 “소네트”



응용2: Latac

위상대수학 교과서를 학습한 뒤 Latex을 기술한 경우.

For $\bigoplus_{m=1,\dots,m} \mathcal{L}_{m\bullet} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ?? Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', x'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of X' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on C as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^* = \mathcal{I}^* \otimes_{\text{Spec}(k)} \mathcal{O}_{S,a} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)^{\text{opp}}_{fppf}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ?? It may replace S by $X_{\text{spaces},\text{etale}}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ?? Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restcomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \widetilde{A}_2$ works.

Lemma 0.3. In Situation ?? Hence we may assume $q' = 0$.

Proof. We will use the property we see that p is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

응용3. 코딩

Generated C code

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNICA) +
                ((count & 0xffffffffffff8) & 0xffff) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
```

해석 가능한 셀 탐색하기

Searching for interpretable cells

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
    */
```

[Visualizing and Understanding Recurrent Networks, Andrej Karpathy*, Justin Johnson*, Li Fei-Fei]

이 셀cell은 해석이 가능한가? RNN에서 어떤 히든 상태 벡터의 특정 cell을 보니, 이것이 어떤 특정 측면에 대해서 규칙을 가지고 행동을 하는 것을 보인다.

(1) quote 탐지 셀

Searching for interpretable cells

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

quote detection cell

인용문을 파란색으로 탐지

2. 줄 개수 탐지

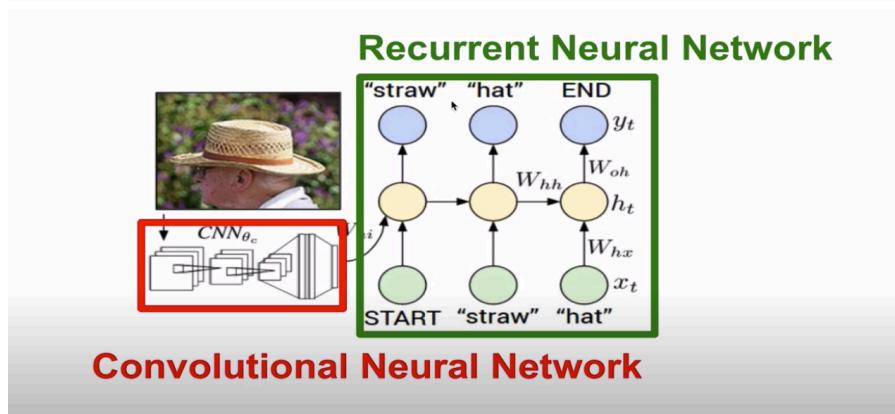
Searching for interpretable cells

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact  
that it plainly and indubitably proved the fallacy of all the plans for  
cutting off the enemy's retreat and the soundness of the only possible  
line of action--the one Kutuzov and the general mass of the army  
demanded--namely, simply to follow the enemy up. The French crowd fled  
at a continually increasing speed and all its energy was directed to  
reaching its goal. It fled like a wounded animal and it was impossible  
to block its path. This was shown not so much by the arrangements it  
made for crossing as by what took place at the bridges. When the bridges  
broke down, unarmed soldiers, people from Moscow and women with children  
who were with the French transport, all--carried on by vis inertiae--  
pressed forward into boats and into the ice-covered water and did not,  
surrender.
```

line length tracking cell

3. 이미지 캡셔닝

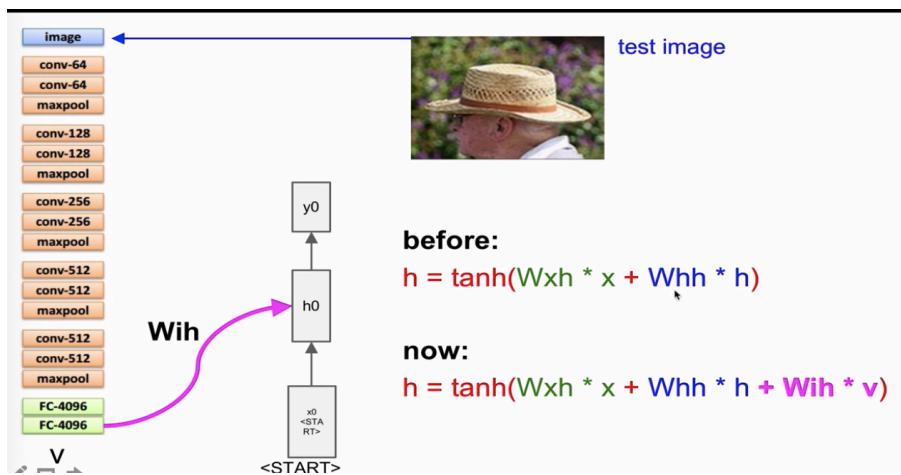


CNN에서는 이미지를 처리해주고, RNN에서는 시퀀스를 처리해준다. Convnet의 결과물은 분류화된 결과물이 나올 것인데, 그것을 다시 입력값으로 받아서 RNN에서 무언가를 처리해 준다. 자세히 보자:

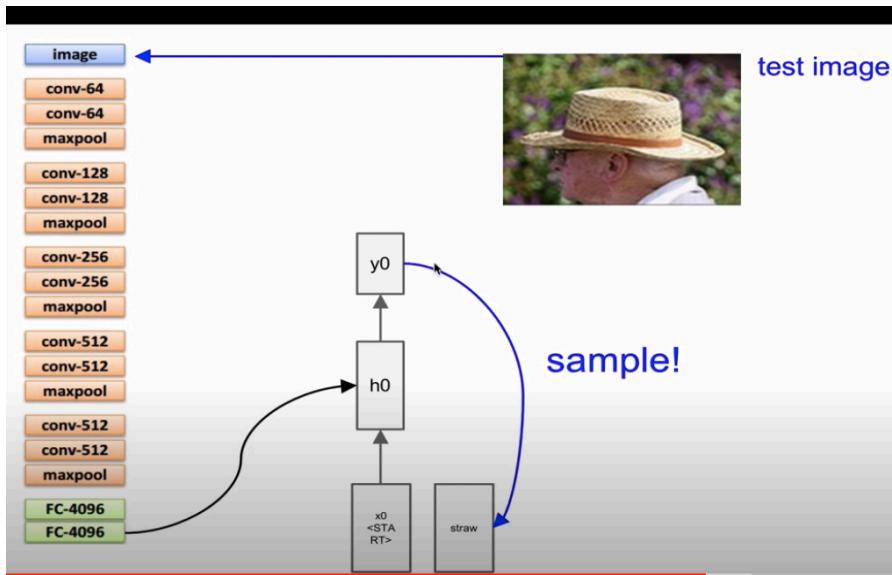
아래 같은 CNN에서 맨 마지막 FCnet과 소프트맥스를 없애준다.



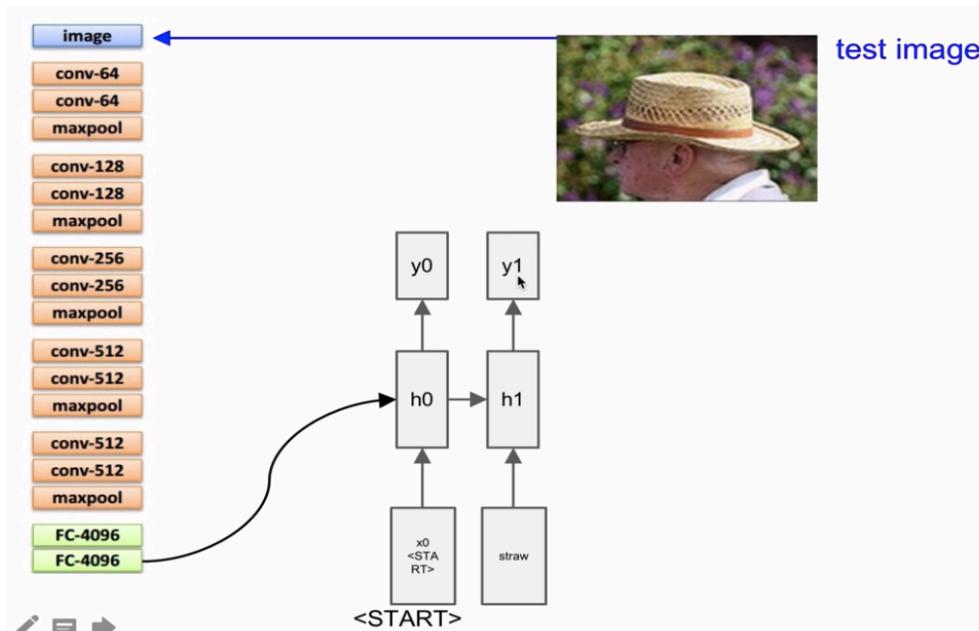
시퀀스를 처음 시작할 때, 즉 RNN을 시작할 때에는 아래 같이 스타트 벡터를 입력한다. 그 뒤에 h_0 , y_0 을 처리한다. 기본적으로 RNN은 아래에서 오는 것, 이전 상태에서 오는 것이 있기 때문에 이들을 더해준 뒤 활성화 해준다. 그런데 이미지 캡셔닝에서는 하나 더 추가되어 W_{ih} 도 있다. 이것은 <이미지 — 히든>. 즉 이미지에서 히든 층으로 온 것에 v (convnet의 top)을 곱해준 것이 된다.



위 같이 y_0 이 출력되면 다음 단계에서 이것이 인풋으로서 쓰인다.



다음 단계에서는 또 다른 것에 대해 학습한다. 그리고 y_1 에 hat임을 탐지하는 데 영향이 커다면, 다음 단계에서 hat을 인풋으로 삼는다.

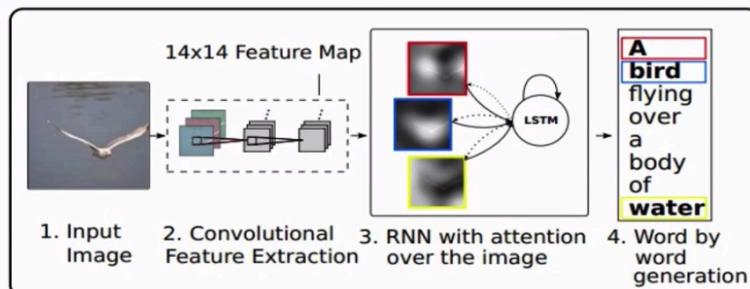


RNN의 후속 아키텍쳐 (Attention)

RNN은 기본적으로 이미지를 전체적으로 한번만 보는데, 어텐션은 이미지의 각 부분을 보고 각각 다른 단어를 추출해낸다.

Preview of fancier architectures

RNN attends spatially to different parts of images while generating each word of the sentence:



Show Attend and Tell, Xu et al., 2015

이것은 단순히 단어만 보는 것이 아니라, 다음에 어디를 봐야하는지 조차도 알려준다고 한다.

RNN에서는 은닉 상태가 존재하는데, LSTM에서는 cell 상태가 공존한다. LSTM은 실제로 더 많이 사용된다. 각각의 셀에는 gate가 있다.

- 아래 LSTM에서 f 는 이전 상태를 얼마나 잊게 해주는지를 정해주는 역할을 수행한다. 만약 f 가 1이라면 이전 상태 전체를 전달해주고, 0이면 reset 해준다.
- i 는 입력값이고,
- g 는 tanh니까 -1에서 1까지가 범위다. g 의 의미는 우리가 i 를 이 cell 상태에 얼마나 더해 줄 것인지를 결정해준다.

RNN:

$$h_t^l = \tanh W^l \begin{pmatrix} h_{t-1}^{l-1} \\ h_t^{l-1} \end{pmatrix}$$

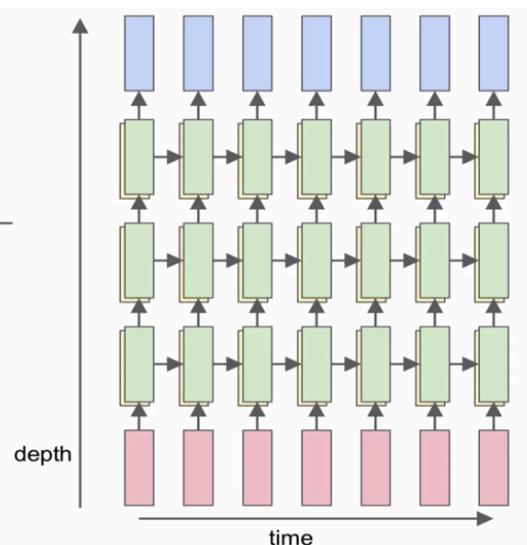
$h \in \mathbb{R}^n$ $W^l [n \times 2n]$

LSTM:

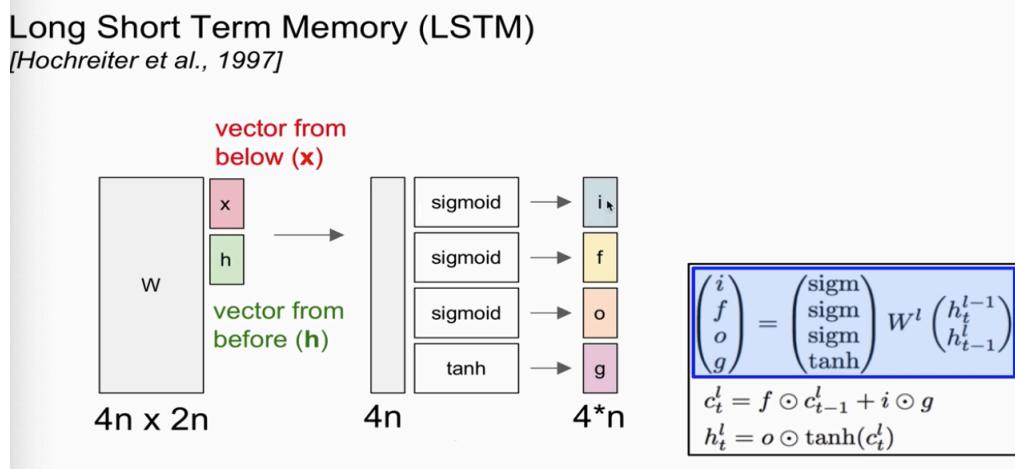
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_{t-1}^{l-1} \\ h_t^{l-1} \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

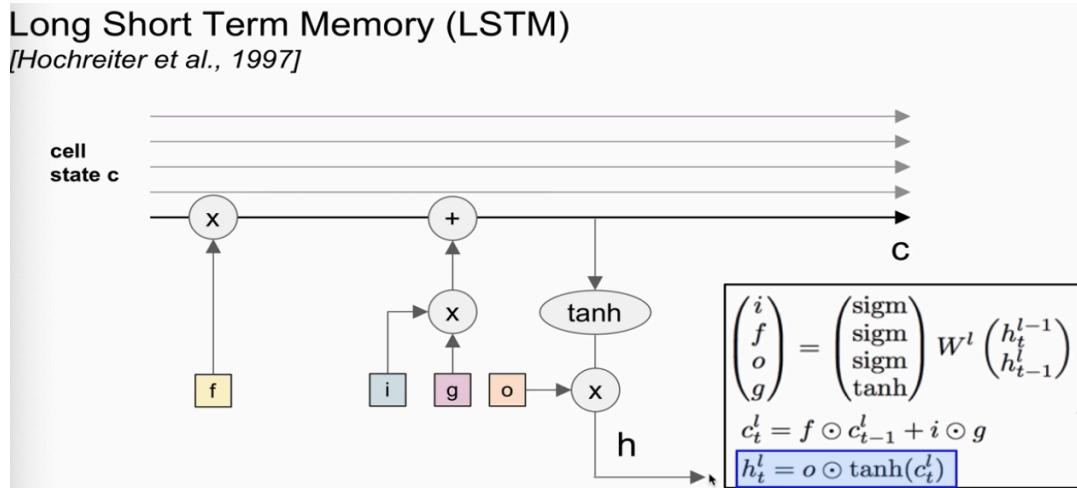
$$h_t^l = o \odot \tanh(c_t^l)$$



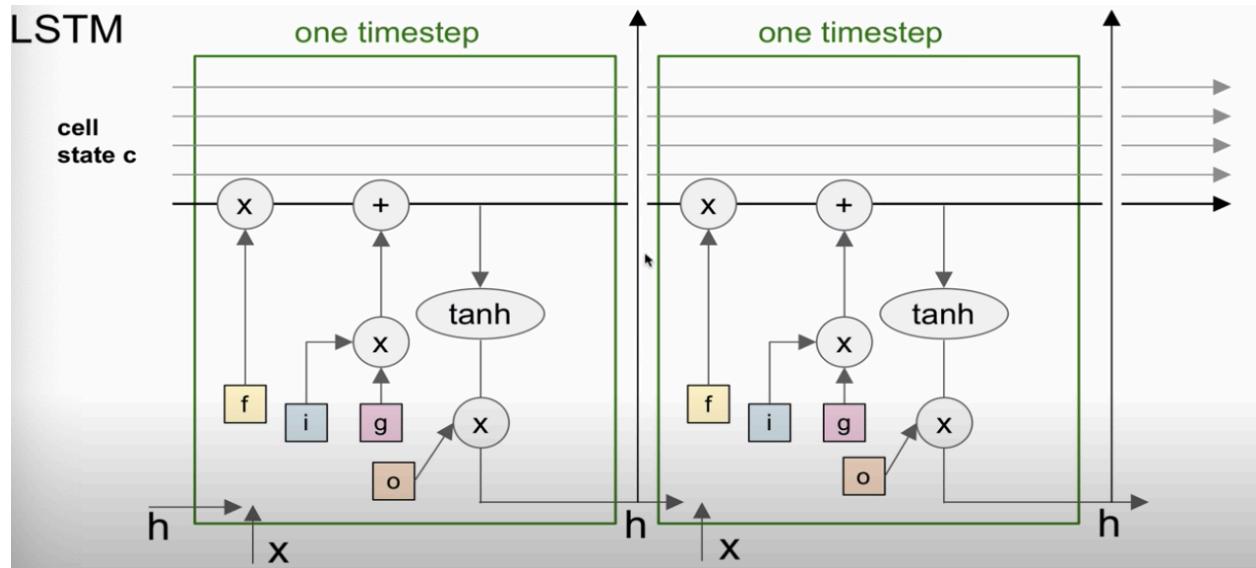
인풋에서의 벡터 x 와 전단계에서의 은닉 벡터 h 를 결합할 때 $4n$, $2n$ 을 곱하면 $4n$ 이 나오고 이것에 활성화를 취하면 아래 같이 산출값이 나온다.



Cell 상태가 쭉 훌러간다고 할 때, $f=0$ 이면 이전 상태를 반영 x . 따라서 셀 상태를 리셋해준다. 그 뒤, i , g 의 연산 단계에서는 입력값을 셀 상태에 어느 만큼 더해줄 것인지를 결정해주고, 그러면 현재의 셀 상태를 구한 것이다. 그 뒤 여기에 활성화를 취한 뒤, o 를 곱해주면, 비로소 은닉 상태를 구하게 된다. 현재의 셀 상태 중 어느 것을 다음의 은닉 셀에 전달할지를 결정해준다.



위에서 얻은 값은 양갈래 길로 나눠져서 이동한다. 위로 가면 더 높은 층, 예측 방향으로 가게 되는 것이다.



왜 현업에서 RNN 보다 LSTM이 더 선호되는가?

RNN에서 구해지는 y 값은 급진적으로 변한다. transformative. 반면에 LSTM에서는 cell state가 flow해서, 3 개의 게이트들이 조금씩 추가되는 additive 양상을 보인다.

