

RNN(Recurrent Neural Network)

2018/05/24

강준하

Index

1. RNN
2. BPTT
3. LSTM
4. char-rnn tensorflow

1. RNN

Why RNN?



Bitcoin



음악

어는 꽃봉오리 같아서 내 머릿속에서 어느 정도 익숙한 꽃으로 펴어 날 때도 있었고 내가 전혀 모르는 형태로 피어나기도 했다. 익숙한 꽃의 형태를 취하면 나는 고무되어 참여하는 시늉을 해보였고 그렇지 않으면 무지를 숨기기 위해서 뒤로 물러났다. 그럴 때면 신경이 곤두섰다.

'저들이 무슨 이야기를 하는지 도무지 모르겠어. 누구에 대해 이야기를 하는 것인지 모르겠어.' 그들의 대화는 내게 의미 없는 소음일 뿐이었다. 세상은 수없이 많은 사람과 수없이 많은 사건과 사상으로 구성되어 있다는 사실을 깨달았다. 애써 노력해온 한밤중의 독서로는 그들의 토론을 따라가기가 턱없이 부족했고 니노와 갈리아니 선생님과 카를로와 아르마도에게 "그래. 맞아. 나도 알아"라고 말할 수 있다.

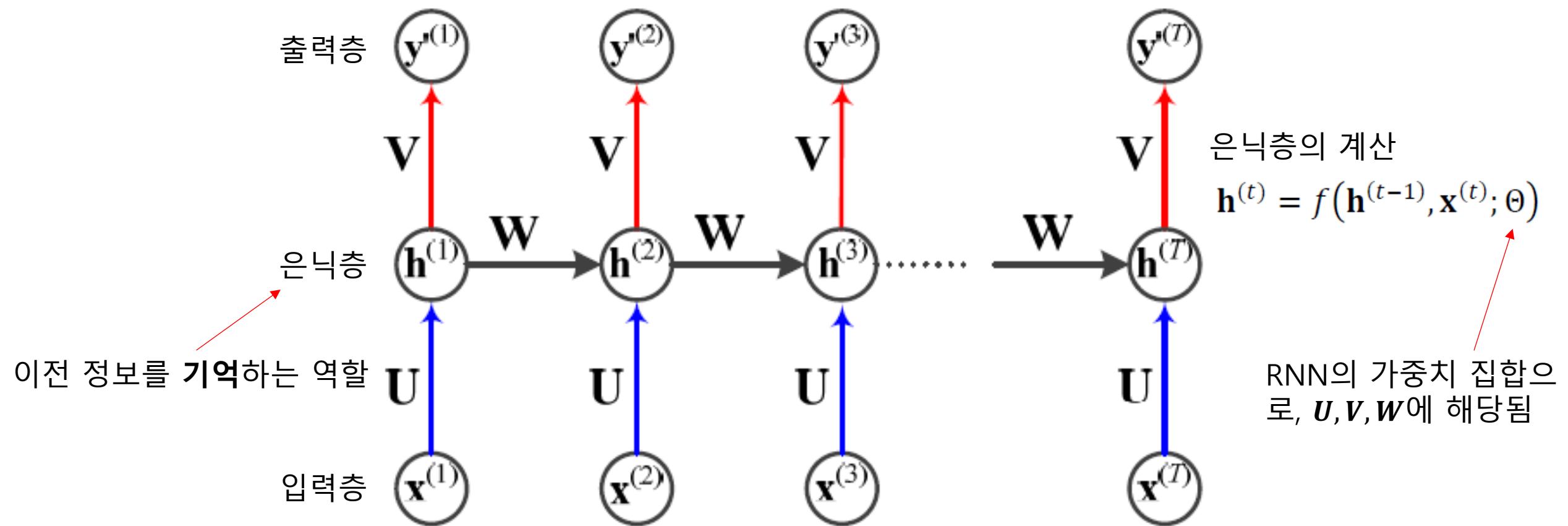
— 소설

RNN

- **Recurrent Neural Network**
- 순차적인 데이터를 다루는 신경망이 가져야 할 조건
 1. 시간성 : 특징을 순서대로 한 번에 하나씩 입력해야 함
 2. 가변 길이 : 길이가 T 인 샘플을 처리하려면 은닉층이 T 번 나타나야 함(T 는 가변적)
 3. 문맥 의존성 : 이전 특징 내용을 기억하고 있다가 적절한 순간에 활용해야 함

RNN

- Recurrent Neural Network

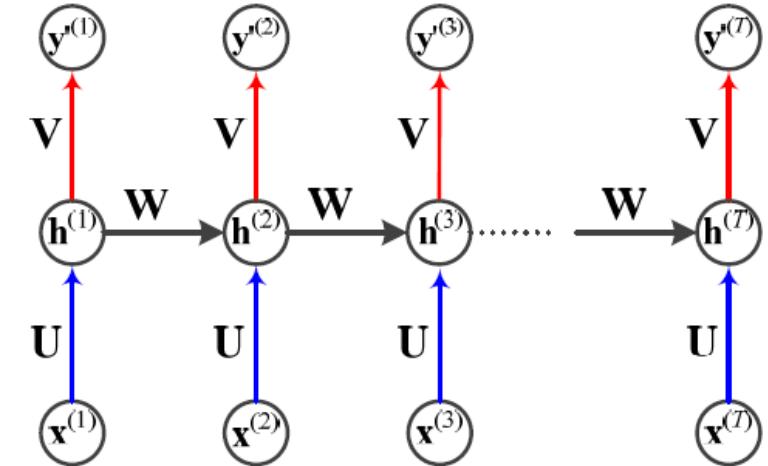


RNN – 구조의 특징과 장점

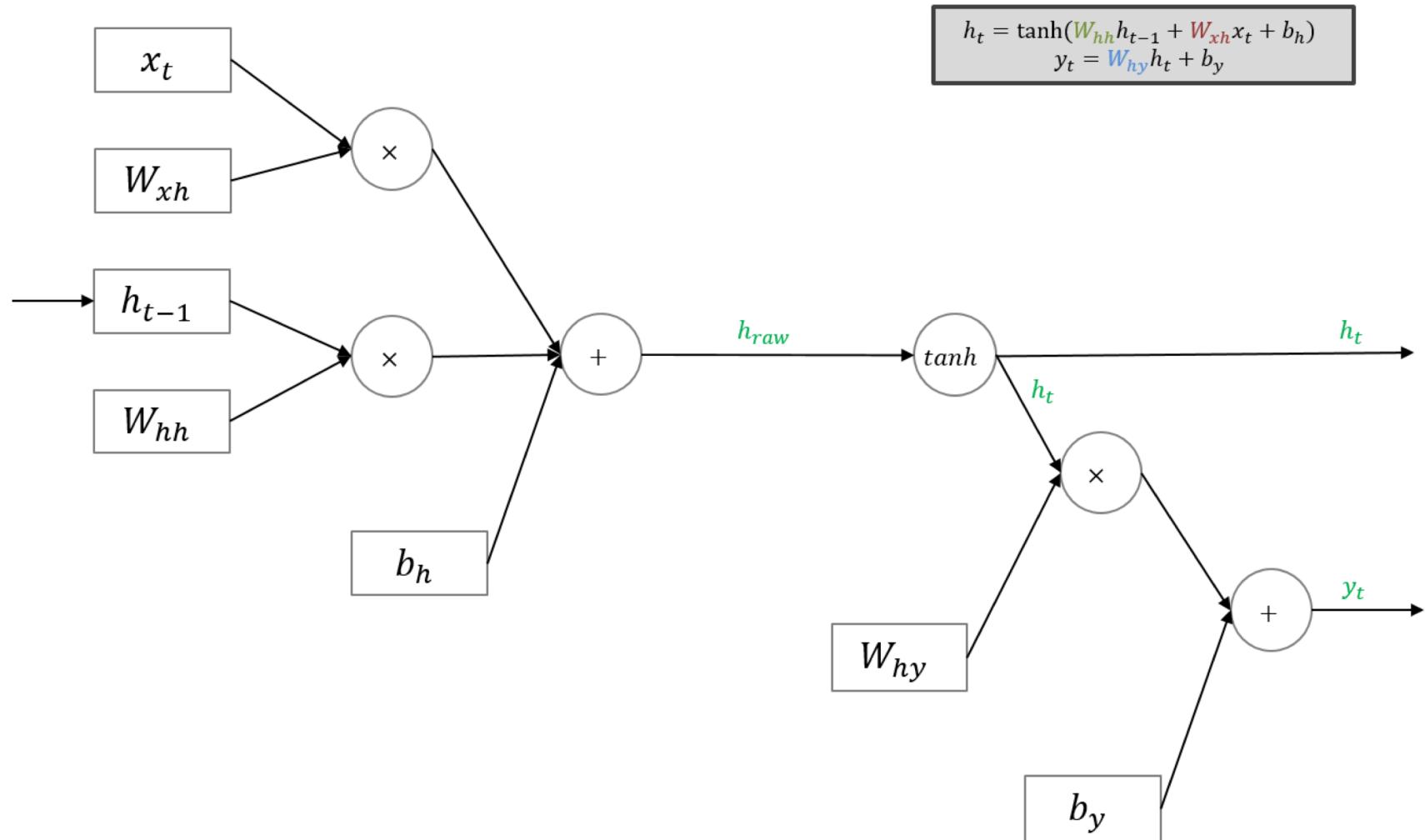
- 매개변수(가중치)를 공유함
- 이를 통해,
 1. 학습 과정에서 추정할 매개변수의 수를 획기적으로 줄임
 2. 매개변수의 개수가 특징 벡터의 길이 T 와 무관하게 일정
 3. 특징이 나타나는 순서가 바뀌어도 같거나 유사한 출력 생성 가능
- Ex) '어제 이 책을 샀다.' 와 '이 책을 어제 샀다.'는 같은 의미의 문장

RNN – 동작

- $\mathbf{h}^{(t)} = \tau(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b})$
- $\mathbf{y}'^{(t)} = \sigma(\mathbf{V}\mathbf{h}^{(t)} + \mathbf{c})$
- τ : tanh function, σ : logistic sigmoid function

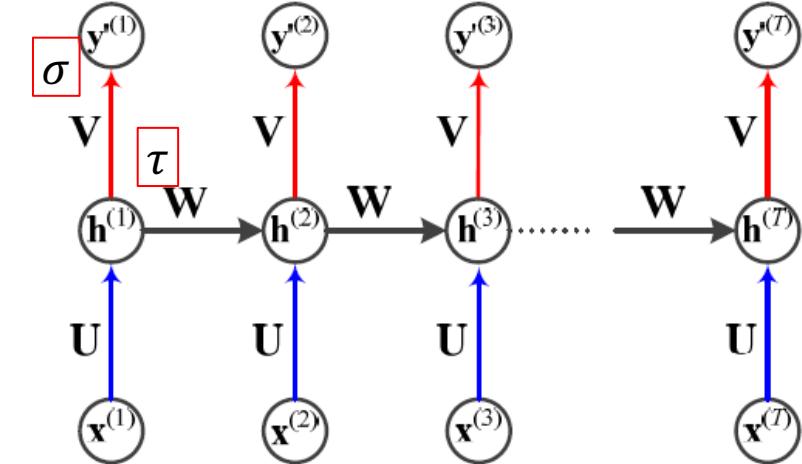


RNN – 동작



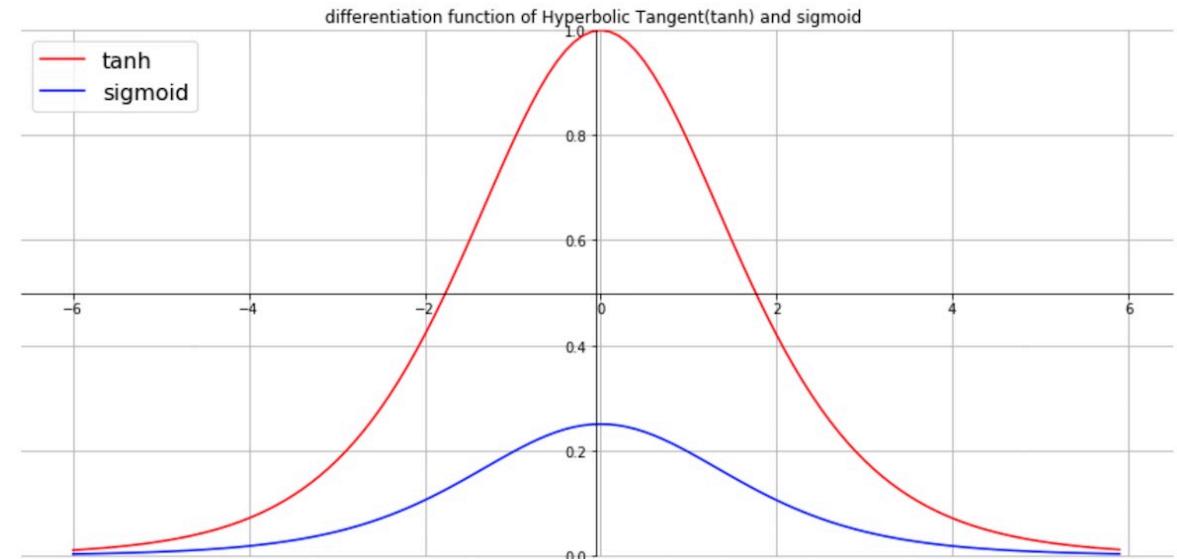
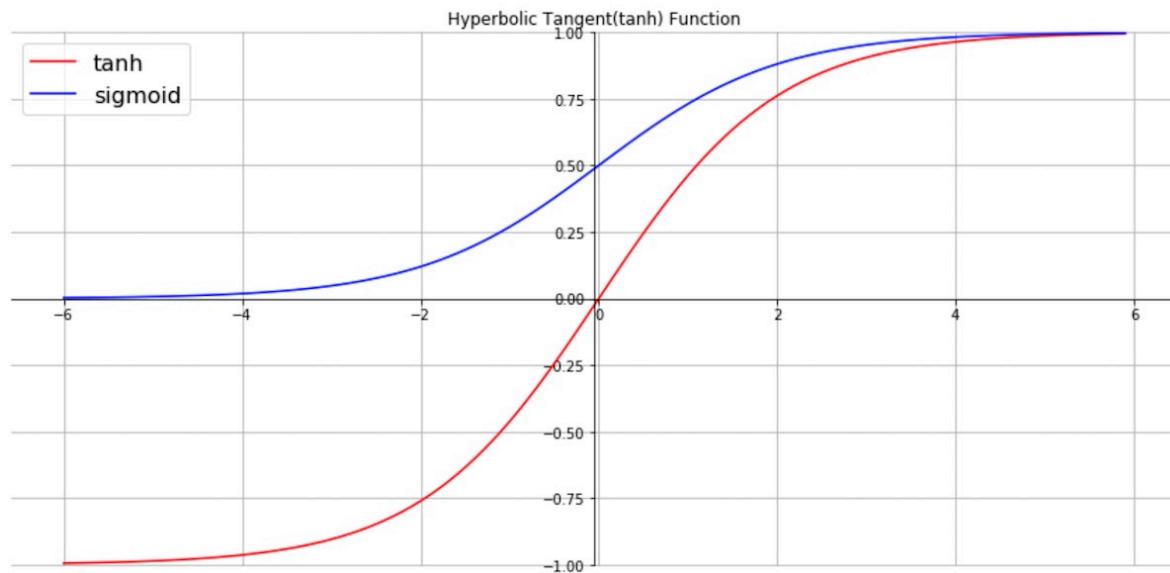
RNN – 동작

- 왜 은닉층에는 tanh 사용하고 출력층에는 sigmoid 사용?
- 출력층에서 sigmoid를 이용하는 이유는 이제까지와 똑같음 (to gain non-linearity)
- 은닉층에서는 2차 미분 형태 또한 지속적으로 값이 남아있을 수 있는 활성함수를 사용해야 vanishing gradient 문제를 피할 수 있었음



RNN – 동작

- $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- $\frac{d}{dx} \tanh(x) = 1 - \tanh^2(x)$



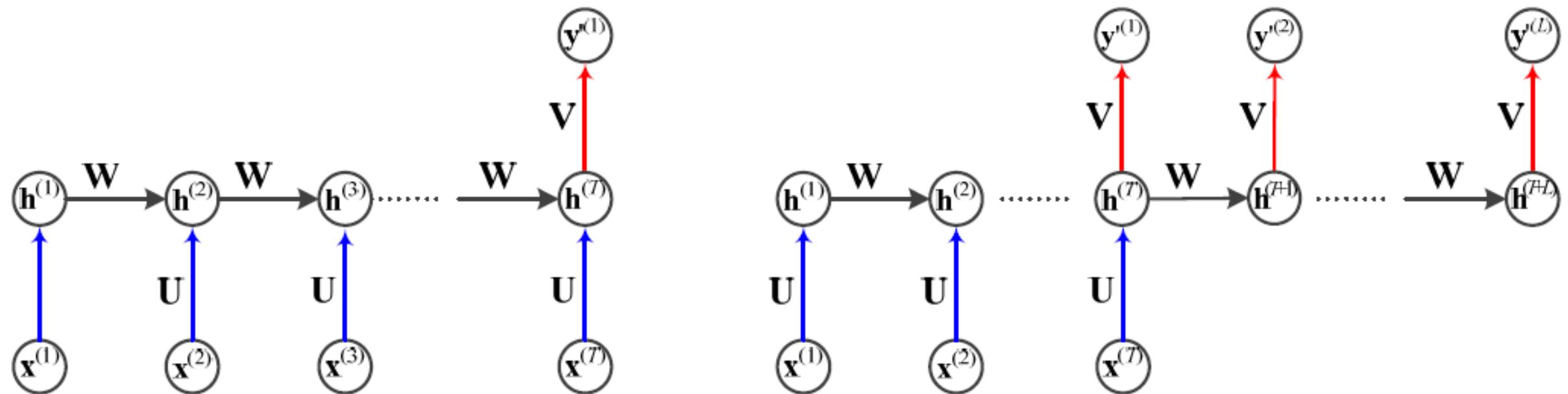
RNN – 동작

- 예상되는 질문 : 어차피 깊어지면 vanishing gradient가 발생하
긴 마찬가지 아닌가?
- Sigmoid보단 덜하다는데 의미를 둬야함
- RNN의 은닉층을 쌓는 구조에서는 선형함수(ReLU)를 사용하면
안됨!

RNN – 동작

- 선형 함수인 $h(x) = cx$ 를 활성 함수로 사용한 3층 네트워크를 가정
- 이를 식으로 나타내면 $y(x) = h(h(h(x)))$
- 이는 $y(x) = c \times c \times c \times x$ 처럼 세번의 곱셈을 수행하지만 실제로는 $y(x) = ax, a = c^3$
- 즉, **은닉층이 없는** 네트워크로 표현 가능
- 그래서 층을 쌓는 혜택을 얻고 싶다면 활성함수로는 반드시 비선형 함수를 사용해야 함

RNN – 여러가지 구조



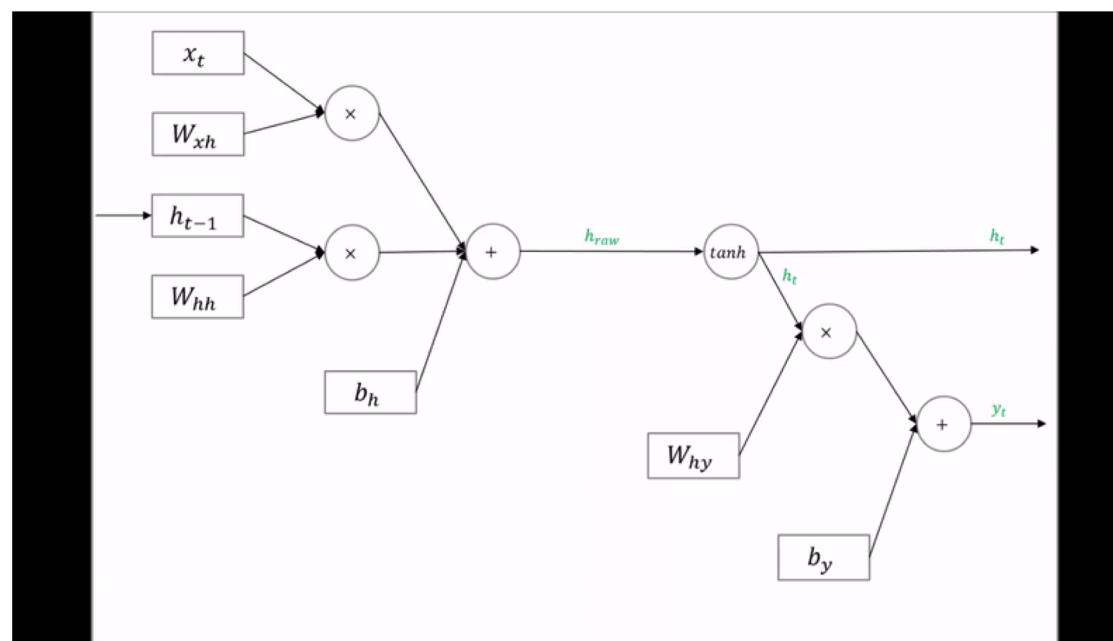
RNN – 한계

- 은닉층은 이전 기억을 저장하는데, 선별적으로 기억하는 기능은 존재하지 않음
- 모든 순간의 입력을 같은 비중으로 기억하기 때문
- 이로 인해 특정 시점 t 의 입력 $x^{(t)}$ 에 대한 기억은 시간이 가면 갈수록 희미해짐 (거리가 멀어지면 vanishing gradient 필연적으로 발생)
→ LSTM 등장

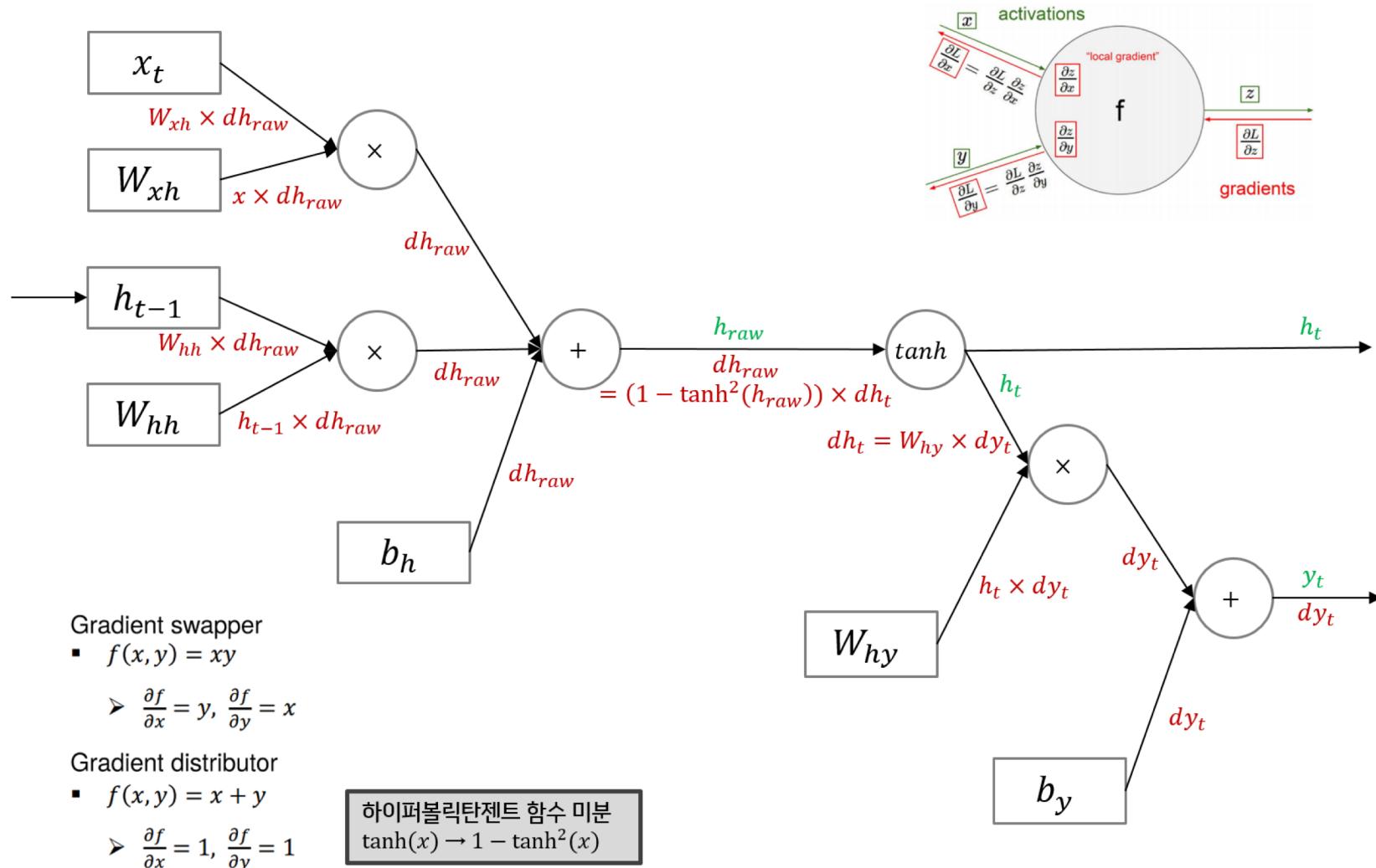
2. BPTT

BPTT

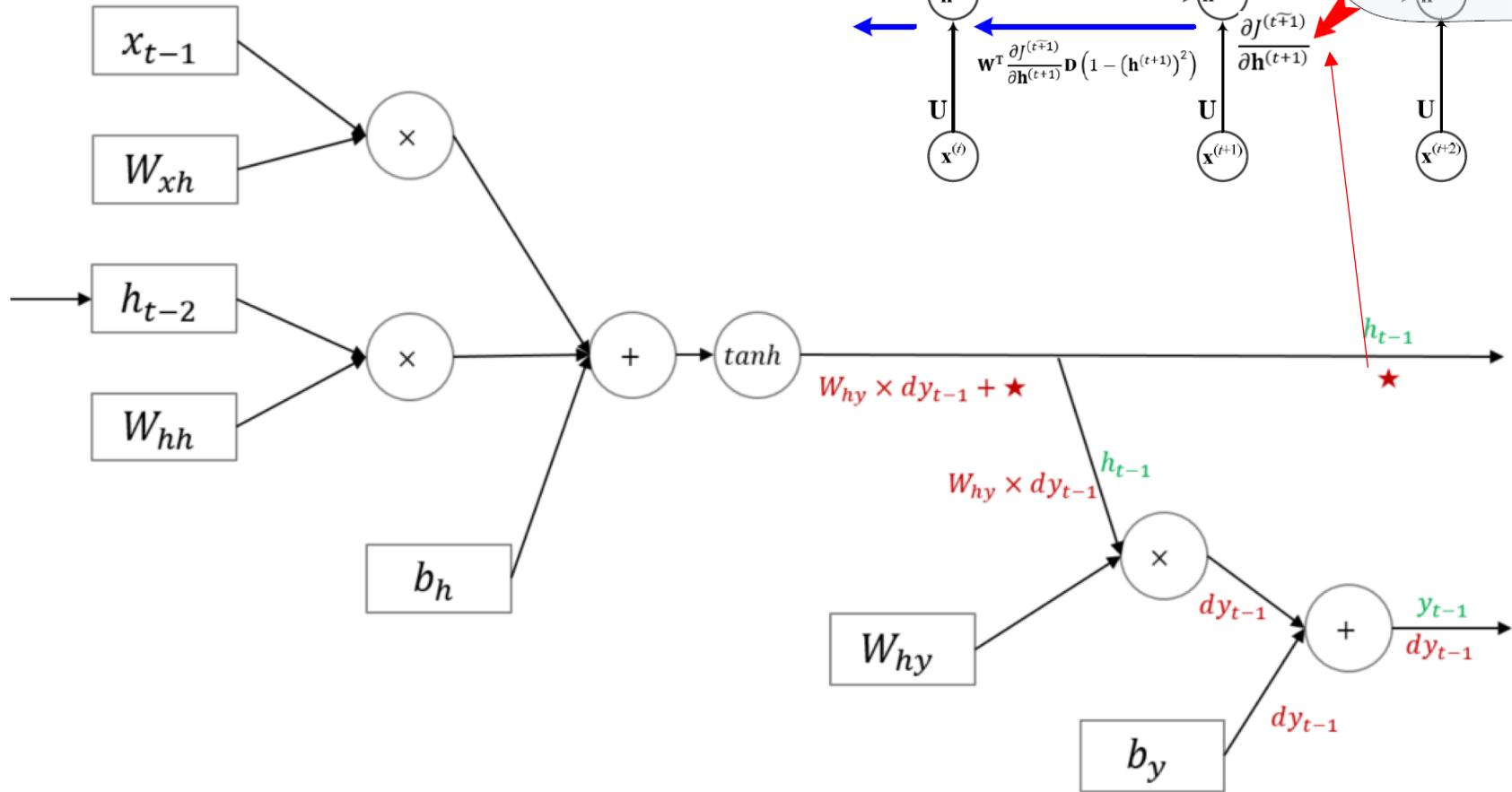
- BackPropagation Through Time
- RNN에서 사용하기 위해 Backpropagation을 적절하게 변형한 알고리즘



BPTT



BPTT



3. LSTM

LSTM

- Long Short Term Memory
- 어떤 입력층이 다른 출력층에 얼마나 영향을 끼치는지 정도가 다 다를 것
- 이를 학습하기 위한 구조

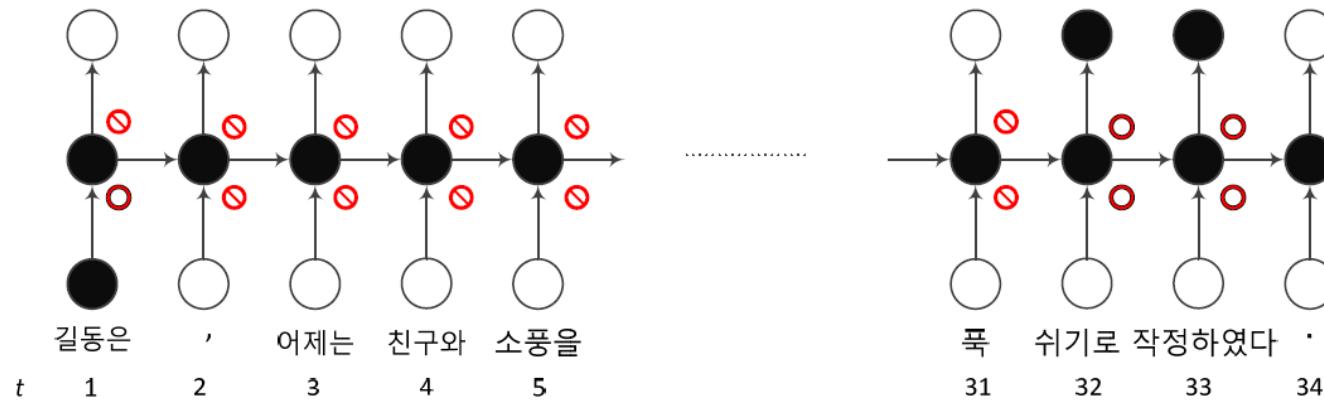
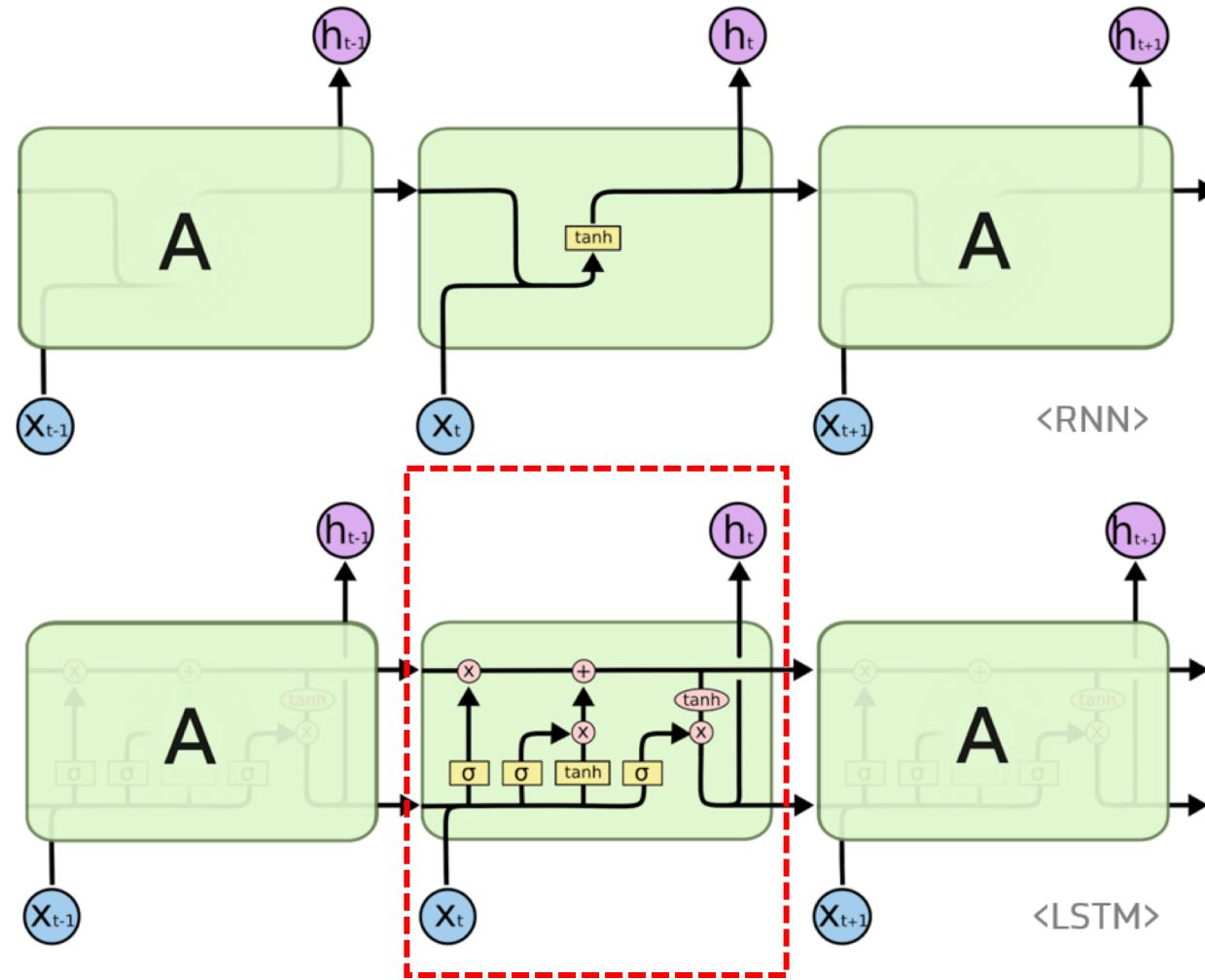
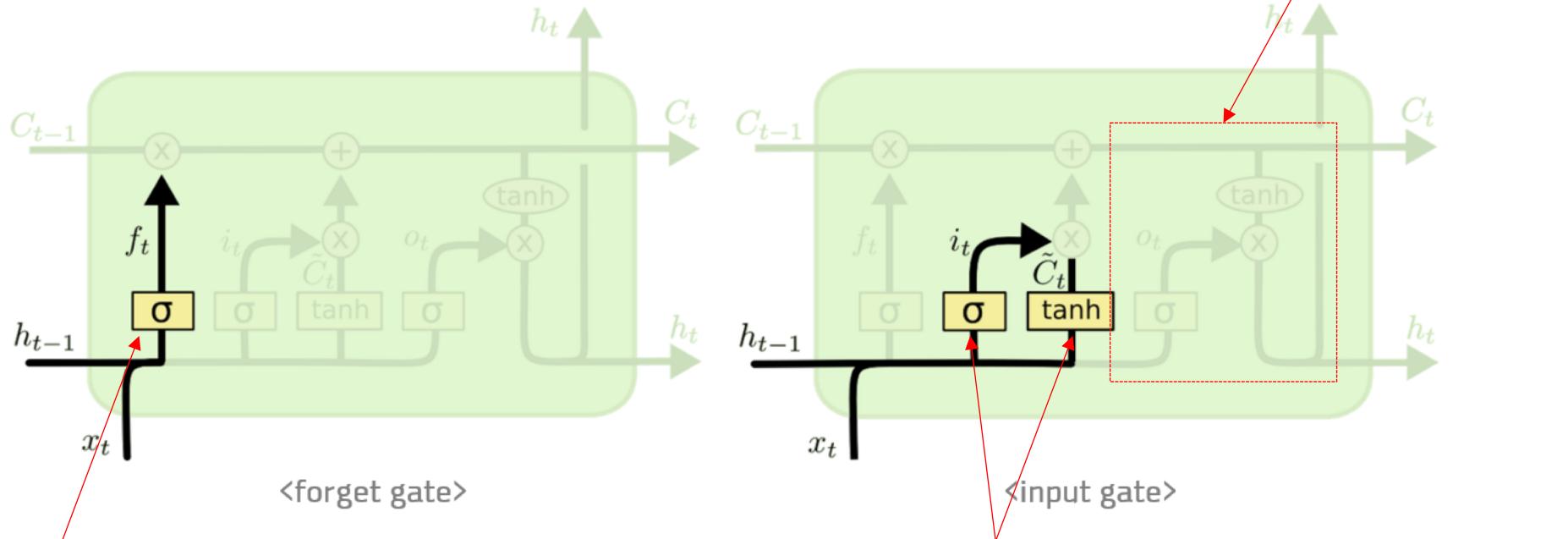


그림 8-14 입력 게이트와 출력 게이트를 이용한 입출력 제어

LSTM



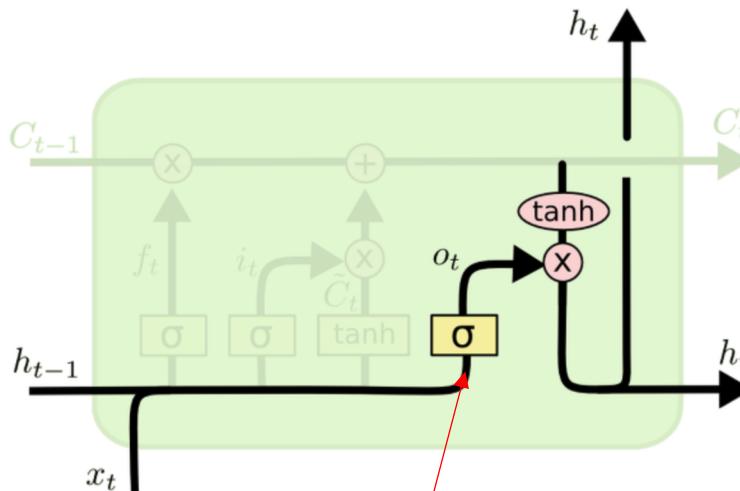
LSTM



sigmoid의 출력 범위는 0~1
0이면 이전 상태의 정보는 잊고,
1이면 이전 상태의 정보를 온전
히 기억하는 것

sigmoid의 출력 범위는 0~1,
tanh의 출력 범위는 -1~1
각각 강도와 반영 방향을 제시하
는 역할

LSTM



sigmoid의 범위가 0~1인 점을 여기
서도 이용해 출력되는 정도를 조절

LSTM

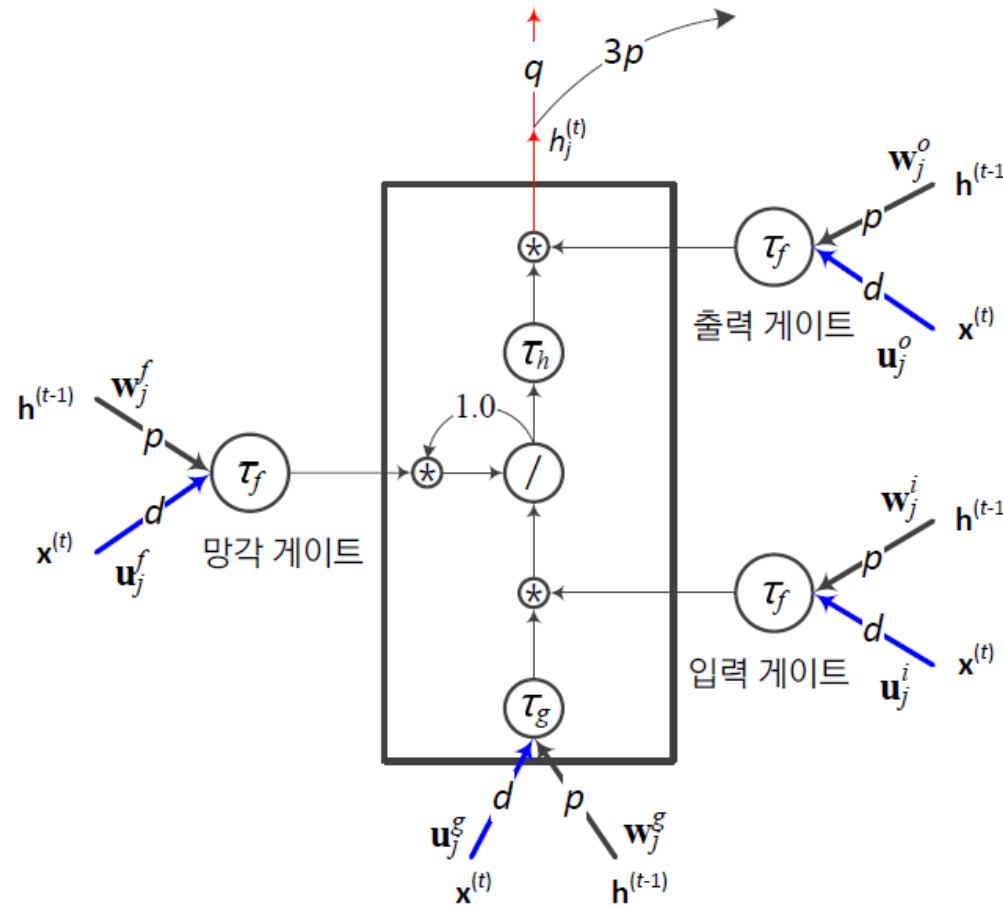
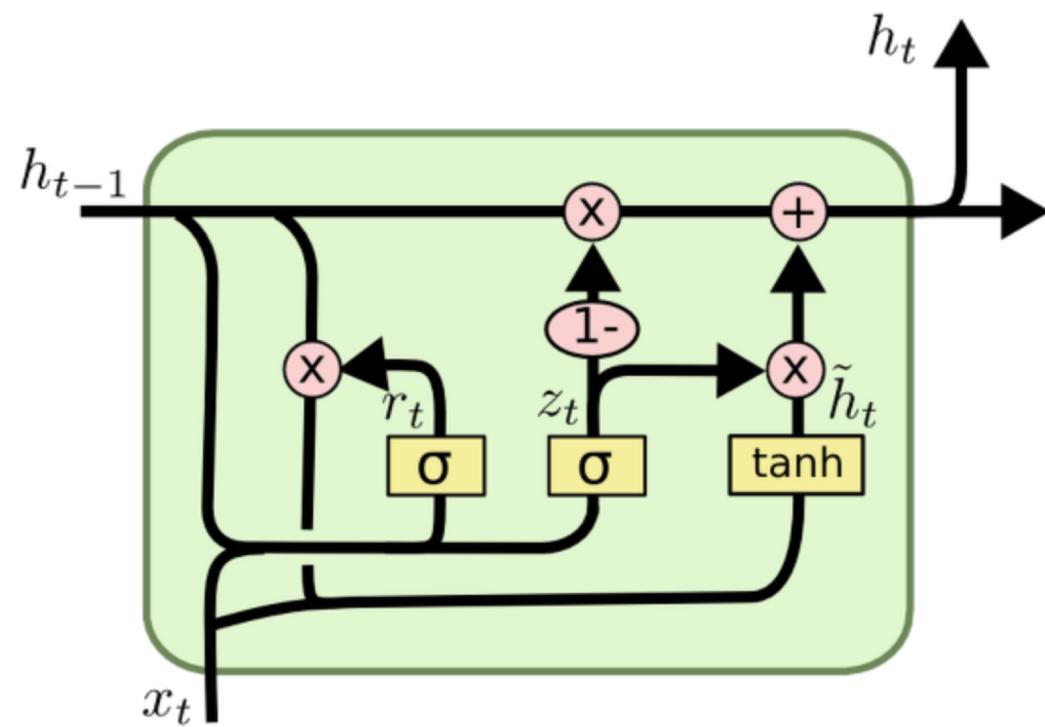


그림 8-18 망각 게이트가 추가된 LSTM 메모리 블록

변칙 LSTM – GRU



변칙 LSTM

- Greff, et al. "LSTM : A Search Space Odyssey", 2015 에 의하면 여러 변칙 구조들을 자세히 분석하였지만 거의 비슷했다고 함
- Jozefowicz, et al. "An Empirical Exploration of Recurrent Network Architectures", 2015 에서는 약 만 가지 이상의 RNN architecture를 실험해 봤는데, 몇몇 변칙패턴들은 LSTM보다도 성능이 좋았다고 함
- 근데 모든 경우에서 GRU나 LSTM을 이길 수는 없었다고 함

For Tensorflow

- RNN 기본 Cell
 - `rnnLayer = tf.contrib.rnn.BasicRNNCell(num_units=n_neurons)`
- LSTM Cell
 - `lstm_cell = tf.contrib.rnn.LSTMCell(num_units=n_neurons, use_peepholes=True)`
- GRU Cell
 - `gru_cell = tf.contrib.rnn.GRUCell(num_units=n_neurons)`

4. char-rnn tensorflow

Unreasonable Effectiveness of RNN

- Andrej Karpathy – The Unreasonable Effectiveness of Recurrent Neural Networks
- <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- Shakespeare, Wikipedia, Mathematics (Latex), Linux Source Code, Generating Baby Names, etc.

Unreasonable Effectiveness of RNN

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

Unreasonable Effectiveness of RNN

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m_n} = 0$, 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 the 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 $x, x', s'' \in S'$ 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 \mathcal{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 \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

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

is a unique morphism of algebraic stacks. Note that

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

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 $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

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

When in this case of 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 \'etale 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 restrocomposes 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 $\mathcal{J}_{n,0} \circ \overline{A}_2$ 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

Unreasonable Effectiveness of RNN

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

Unreasonable Effectiveness of RNN

- Karpathy was used Torch(lua) for implementation...
- Let's make it with tensorflow!
- <https://github.com/solaris33/char-rnn-tensorflow>

Preprocessing

```
8  class TextLoader():
9      def __init__(self, data_dir, batch_size, seq_length, encoding='utf-8'):
10         self.data_dir = data_dir
11         self.batch_size = batch_size
12         self.seq_length = seq_length
13         self.encoding = encoding
14
15         input_file = os.path.join(data_dir, "input.txt")
16         vocab_file = os.path.join(data_dir, "vocab.pkl")
17         tensor_file = os.path.join(data_dir, "data.npy")
18
19         if not (os.path.exists(vocab_file) and os.path.exists(tensor_file)):
20             print("reading text file")
21             self.preprocess(input_file, vocab_file, tensor_file)
22         else:
23             print("loading preprocessed files")
24             self.load_preprocessed(vocab_file, tensor_file)
25             self.create_batches()
26             self.reset_batch_pointer()
```

TextLoader class initialization
File check & making

Preprocessing

```
28     def preprocess(self, input_file, vocab_file, tensor_file):
29         with codecs.open(input_file, "r", encoding=self.encoding) as f:
30             #             with codecs.open(input_file, 'rb') as f:           Preprocessing
31                 data = f.read()                                     단어별로 나눠서 저장
32             counter = collections.Counter(data)
33             count_pairs = sorted(counter.items(), key=lambda x: -x[1])
34             self.chars, _ = zip(*count_pairs)
35             self.vocab_size = len(self.chars)
36             self.vocab = dict(zip(self.chars, range(len(self.chars))))
37             with open(vocab_file, 'wb') as f:
38                 cPickle.dump(self.chars, f)
39             self.tensor = np.array(list(map(self.vocab.get, data)))
40             np.save(tensor_file, self.tensor)
```

Preprocessing

```
42     def load_preprocessed(self, vocab_file, tensor_file):
43         with open(vocab_file, 'rb') as f:
44             self.chars = cPickle.load(f)
45             self.vocab_size = len(self.chars)
46             self.vocab = dict(zip(self.chars, range(len(self.chars))))
47             self.tensor = np.load(tensor_file)
48             self.num_batches = int(self.tensor.size / (self.batch_size *
49                                         self.seq_length))
```

Preprocess되어 있는거 파일로 저장되어 있으니깐 불러오기

Preprocessing

```
51     def create_batches(self):
52         self.num_batches = int(self.tensor.size / (self.batch_size *
53                                         self.seq_length))
54
55         # When the data (tensor) is too small,                      Batch 크기에 맞춰서 쪼개기
56         # let's give them a better error message
57         if self.num_batches == 0:
58             assert False, "Not enough data. Make seq_length and batch_size small."
59
60         self.tensor = self.tensor[:self.num_batches * self.batch_size * self.seq_length]
61         xdata = self.tensor
62         ydata = np.copy(self.tensor)
63         ydata[:-1] = xdata[1:]
64         ydata[-1] = xdata[0]
65         self.x_batches = np.split(xdata.reshape(self.batch_size, -1),
66                               self.num_batches, 1)
67         self.y_batches = np.split(ydata.reshape(self.batch_size, -1),
68                               self.num_batches, 1)
```

Preprocessing

```
70     def next_batch(self):
71         x, y = self.x_batches[self.pointer], self.y_batches[self.pointer]
72         self.pointer += 1
73         return x, y
74
75     def reset_batch_pointer(self):
76         self.pointer = 0
```

Batch 조절 관련 함수들

Training

```
13 # 학습에 필요한 설정값들을 지정합니다.  
14 data_dir = 'data/tinyshakespeare' # 셰익스피어 희곡 <리처드 3세> 데이터로 학습  
15 #data_dir = 'data/linux' # <Linux 소스코드> 데이터로 학습  
16 batch_size = 50 # Training : 50, Sampling : 1  
17 seq_length = 50 # Training : 50, Sampling : 1  
18 hidden_size = 128 # 히든 레이어의 노드 개수  
19 learning_rate = 0.002  
20 num_epochs = 2  
21 num_hidden_layers = 2  
22 grad_clip = 5 # Gradient Clipping에 사용할 임계값  
23  
24 # TextLoader를 이용해서 데이터를 불러옵니다.  
25 data_loader = TextLoader(data_dir, batch_size, seq_length)  
26 # 학습데이터에 포함된 모든 단어들을 나타내는 변수인 chars와 chars에 id를 부여해 dict 형태로 만든 vocab을 선언합니다.  
27 chars = data_loader.chars  
28 vocab = data_loader.vocab  
29 vocab_size = data_loader.vocab_size # 전체 단어개수
```

기본 변수들 설정

데이터 load

Training

```
31 # 인풋데이터와 타겟데이터, 배치 사이즈를 입력받기 위한 플레이스홀더를 설정합니다.  
32 input_data = tf.placeholder(tf.int32, shape=[None, None]) # input_data : [batch_size, seq_length])  
33 target_data = tf.placeholder(tf.int32, shape=[None, None]) # target_data : [batch_size, seq_length])  
34 state_batch_size = tf.placeholder(tf.int32, shape=[]) # Training : 50, Sampling : 1  
35  
36 # RNN의 마지막 히든레이어의 출력을 소프트맥스 출력값으로 변환해주기 위한 변수들을 선언합니다.  
37 # hidden_size -> vocab_size  
38 softmax_w = tf.Variable(tf.random_normal(shape=[hidden_size, vocab_size]), dtype=tf.float32)  
39 softmax_b = tf.Variable(tf.random_normal(shape=[vocab_size]), dtype=tf.float32)
```

input, target, batch size 초기화
Output layer weight와 bias 초기화

Training

```
41      # num_hidden_layers만큼 LSTM cell(히든레이어)를 선언합니다.  
42      cells = []  
43      for _ in range(0, num_hidden_layers):  
44          cell = tf.nn.rnn_cell.BasicLSTMCell(hidden_size)  
45          cells.append(cell)  
46  
47      # cell을 종합해서 RNN을 정의합니다.  
48      cell = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)
```

RNN Cell (LSTM) 선언

선언한 Unit 종합하여 MultiRNNCell 선언

Training

```
50    # 인풋데이터를 변환하기 위한 Embedding Matrix를 선언합니다.  
51    # vocab_size -> hidden_size  
52    embedding = tf.Variable(tf.random_normal(shape=[vocab_size, hidden_size]), dtype=tf.float32)  
53    inputs = tf.nn.embedding_lookup(embedding, input_data)  
54  
55    # 초기 state 값을 0으로 초기화합니다.  
56    initial_state = cell.zero_state(state_batch_size, tf.float32)
```

Embedding Matrix : input data를 vocabulary 단위로 쪼갬
state 초기화

Training

```
58 # 학습을 위한 tf.nn.dynamic_rnn을 선언합니다.  
59 # outputs : [batch_size, seq_length, hidden_size]  
60 outputs, final_state = tf.nn.dynamic_rnn(cell, inputs, initial_state=initial_state, dtype=tf.float32)  
61 # ouputs을 [batch_size * seq_length, hidden_size]] 형태로 바꿉니다.  
62 output = tf.reshape(outputs, [-1, hidden_size])
```

RNN cell setting, output initialize

Training

```
64    # 최종 출력값을 설정합니다.  
65    # logits : [batch_size * seq_length, vocab_size]  
66    logits = tf.matmul(output, softmax_w) + softmax_b  
67    probs = tf.nn.softmax(logits)  
68  
69    # Cross Entropy 손실 함수를 정의합니다.  
70    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=target_data))  
71  
72    # 옵티마이저를 선언하고 옵티마이저에 Gradient Clipping을 적용합니다.  
73    # grad_clip(=5)보다 큰 Gradient를 5로 Clipping합니다.  
74    tvars = tf.trainable_variables()  
75    grads, _ = tf.clip_by_global_norm(tf.gradients(loss, tvars), grad_clip)  
76    optimizer = tf.train.AdamOptimizer(learning_rate)  
77    train_step = optimizer.apply_gradients(zip(grads, tvars))
```

Weight와 bias를 이용해 output 함수 구성

Loss function 구성 (Cross Entropy)

Optimizer 설정

Training

```
79     # 세션을 열고 학습을 진행합니다.
80     with tf.Session() as sess:
81         # 변수들에 초기값을 할당합니다.
82         sess.run(tf.global_variables_initializer())
83
84     for e in range(num_epochs):
85         data_loader.reset_batch_pointer()
86         # 초기 상태값을 지정합니다.
87         state = sess.run(initial_state, feed_dict={state_batch_size : batch_size})
88
89     for b in range(data_loader.num_batches):
90         # x, y 데이터를 불러옵니다.
91         x, y = data_loader.next_batch()
92         # y에 one-hot 인코딩을 적용합니다.
93         y = tf.one_hot(y, vocab_size)           # y : [batch_size, seq_length, vocab_size]
94         y = tf.reshape(y, [-1, vocab_size])    # y : [batch_size * seq_length, vocab_size]
95         y = y.eval()
96
97         # feed-dict에 사용할 값들과 LSTM 초기 cell state(feed_dict[c]) 값과 hidden layer 출력값(feed_dict[h])을 지정합니다.
98         feed_dict = {input_data : x, target_data: y, state_batch_size : batch_size}
99         for i, (c, h) in enumerate(initial_state):
100             feed_dict[c] = state[i].c
101             feed_dict[h] = state[i].h
102
103         # 한스텝 학습을 진행합니다.
104         _, loss_print, state = sess.run([train_step, loss, final_state], feed_dict=feed_dict)
105
106         print("{}(학습한 배치개수)/{}(학습할 배치개수), 반복(epoch): {}, 손실함수(loss): {:.3f}".format(
107             e * data_loader.num_batches + b,
108             num_epochs * data_loader.num_batches,
109             (e+1),
110             loss_print))
111
112     print("트레이닝이 끝났습니다!")
```

학습 진행
알파벳별로 one hot encoding

Training

```
115     # 샘플링 시작
116     print("샘플링을 시작합니다!")
117     num_sampling = 4000 # 생성할 글자(Character)의 개수를 지정합니다.
118     prime = u' '        # 시작 글자를 ' ' (공백)으로 지정합니다.
119     sampling_type = 1    # 샘플링 타입을 설정합니다.
120     state = sess.run(cell.zero_state(1, tf.float32)) # RNN의 최초 state값을 0으로 초기화합니다.
121
122     # Random Sampling을 위한 weighted_pick 함수를 정의합니다.
123     def weighted_pick(weights):
124         t = np.cumsum(weights)
125         s = np.sum(weights)
126         return(int(np.searchsorted(t, np.random.rand(1)*s)))
```

Training

```
122     # Random Sampling을 위한 weighted_pick 함수를 정의합니다.
123     def weighted_pick(weights):
124         t = np.cumsum(weights)
125         s = np.sum(weights)
126         return(int(np.searchsorted(t, np.random.rand(1)*s)))
127
128     ret = prime      # 샘플링 결과를 리턴받을 ret 변수에 첫번째 글자를 할당합니다.
129     char = prime[-1] # Char-RNN의 첫번째 인풋을 지정합니다.
130     for n in range(num_sampling):
131         x = np.zeros((1, 1))
132         x[0, 0] = vocab[char]
133
134         # RNN을 한스텝 실행하고 Softmax 행렬을 리턴으로 받습니다.
135         feed_dict = {input_data: x, state_batch_size : 1, initial_state: state}
136         [probs_result, state] = sess.run([probs, final_state], feed_dict=feed_dict)
137
138         # 불필요한 차원을 제거합니다.
139         # probs_result : (1,65) -> p : (65)
140         p = np.squeeze(probs_result)
141
142         # 샘플링 타입에 따라 3가지 종류로 샘플링 합니다.
143         # sampling_type : 0 -> 다음 글자를 예측할때 항상 argmax를 사용
144         # sampling_type : 1(defualt) -> 다음 글자를 예측할때 항상 random sampling을 사용
145         # sampling_type : 2 -> 다음 글자를 예측할때 이전 글자가 ' ' (공백) 이면 random sampling, 그럴지 않을 경우 argmax를 사용
146         if sampling_type == 0:
147             sample = np.argmax(p)
148         elif sampling_type == 2:
149             if char == ' ':
150                 sample = weighted_pick(p)
151             else:
152                 sample = np.argmax(p)
153         else:
154             sample = weighted_pick(p)
155
156         pred = chars[sample]
157         ret += pred      # 샘플링 결과에 현재 스텝에서 예측한 글자를 추가합니다. (예를들어 pred=L일 경우, ret = HEL -> HELL)
158         char = pred      # 예측한 글자를 다음 RNN의 인풋으로 사용합니다.
```

Result

```
1. junha@Junhai-MacBook-Pro: ~/Documents/workspace/study/char-rnn-tensorflow (zsh)
× ..ce/han... %1 × jupyter %2 × ..nha.git... %3 × ..earning... %4 × ..tyle-tra... %5 × ..nn-tens... %6
885(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.487
886(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.541
887(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.475
888(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.485
889(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.455
890(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.513
891(학습한 배치개수)/892(학습할 배치개수), 반복(epoch): 2, 손실함수(loss): 1.495
트레이닝이 끝났습니다!
샘플링을 시작합니다!
샘플링 결과:
sport
Tyaltion so.

Lord Clazentage's, or any wish didine,
I, thine which'd him those that have truth.

CLORUS:
And tridblam; men may's quarques
Whis is your heard the amefor breein, and
Mound him sortory stup he wome?

DUKE VINCENTIO:
The time to crom. I famelory memliar,
Beond nor oldie of you had things here advanu,
Whishall not mother, caster still laidon which will much be now
Thou swear? so Glace to this manust the death.

LAUDION:
A's midded it
Stails Marger of him enemish! Waster fair
For a powin: the crowle, my lieford!
```