딥러닝을 이용한
감성분석기 개발

정상근

2017-11

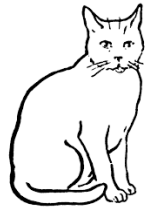
Deep Learning for NLP

SYMBOLS TO VECTOR

Deep Learning 처리 단위 = Vector

Symbol 이 아닌 Vector 가 처리의 기본 단위

[Representation]



Cat

One-Hot Representation

[0, 0, 0, **1**, 0, …]

Cat

Distributed Representation

[**34.2, 93.2, 45.3**, …]

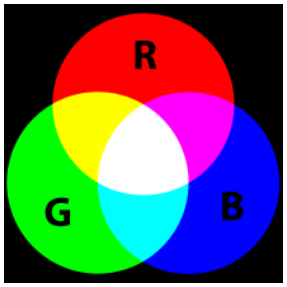
비교 – 영상 / 음성 / 자연어

Object

Vector

[영상]

Analog / Digital Converter



RGB = [32,54,11]

[음성]

Analog / Digital Converter

Wave = [12,35,45,10,…]

[자연어]

문자화

"사과"

Vector 가 아님!

자연어의 경우 이미
한번의 '*Digital*'화가
이루어졌다고도 볼 수 있음

자연어의 Vector 화

Object

Vector

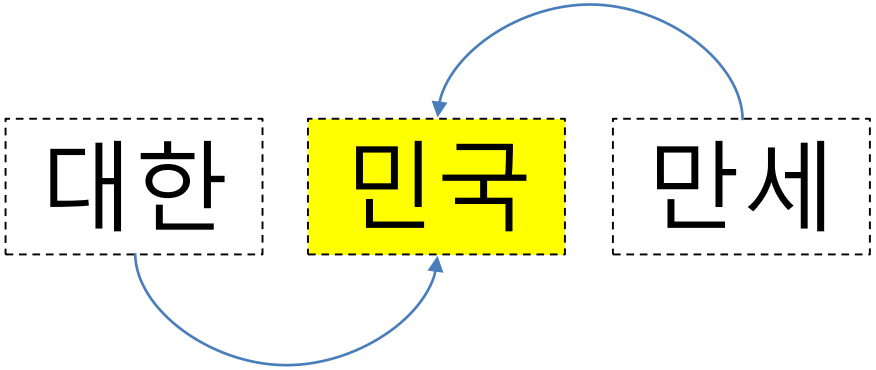
어떻게 자연어를 유의미한 Vector 로 변환 할 것인가?

- Neural Language Modeling
- Word2Vec(Skipgram, CBOW)
- Glove
- Sentence2Vec
- Doc2Vec
- …

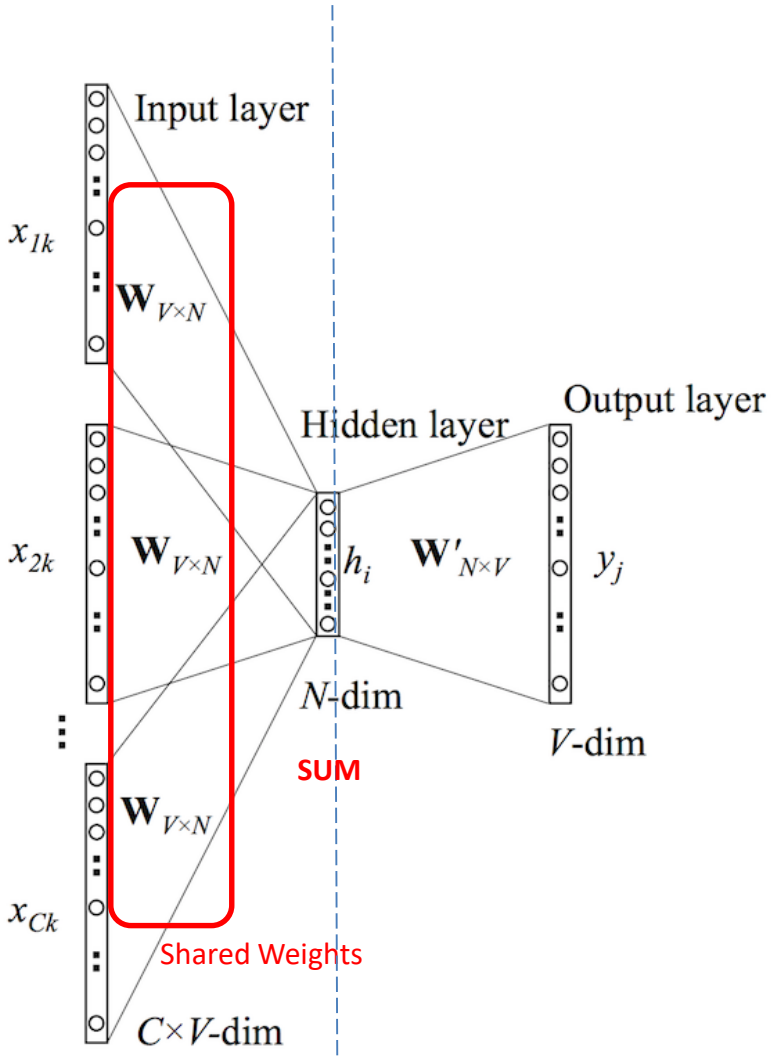
Word2Vec : CBOW

[Idea]

'단어'란 주변의 단어로 정의된다.



Continuous Bag-of-words Architecture



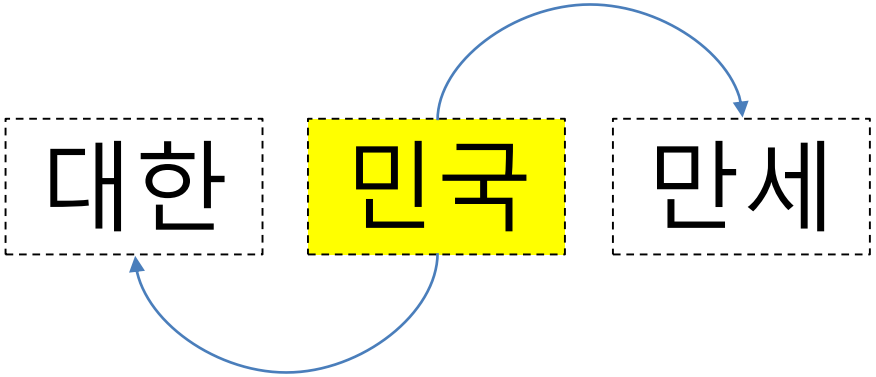
- Predicts the current word given the context

http://alexminnaar.com/word2vec-tutorial-part-ii-the-continuous-bag-of-words-model.html

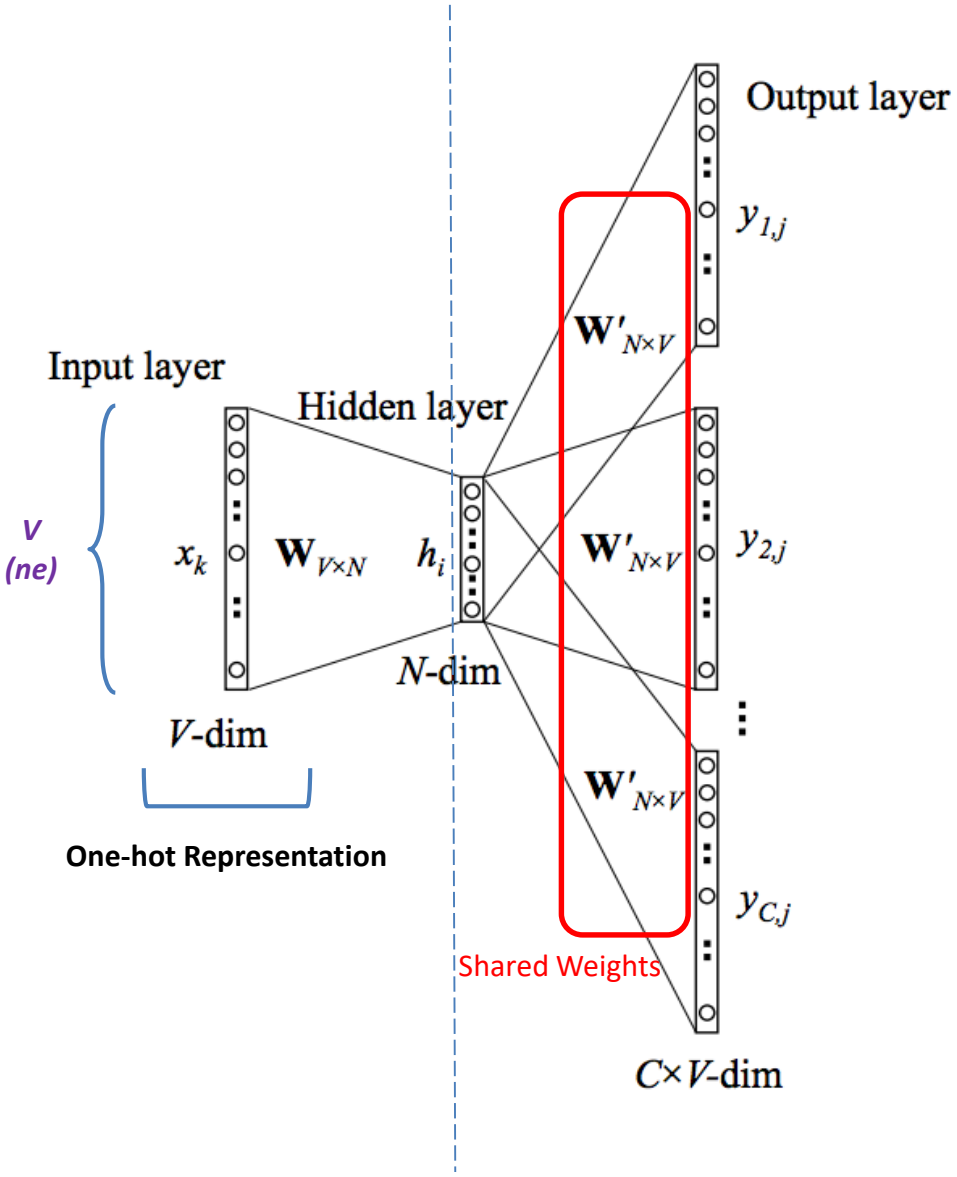
Word2Vec : Skipgram

[Idea]

주변의 '단어'를 잘 설명하는 무엇이
그 단어를 정의한다.



Skip-gram Architecture



- Predicts the surrounding words given the current word

http://alexminnaar.com/word2vec-tutorial-part-ii-the-continuous-bag-of-words-model.html

Word2Vec 결과물 : Semantic Guessing

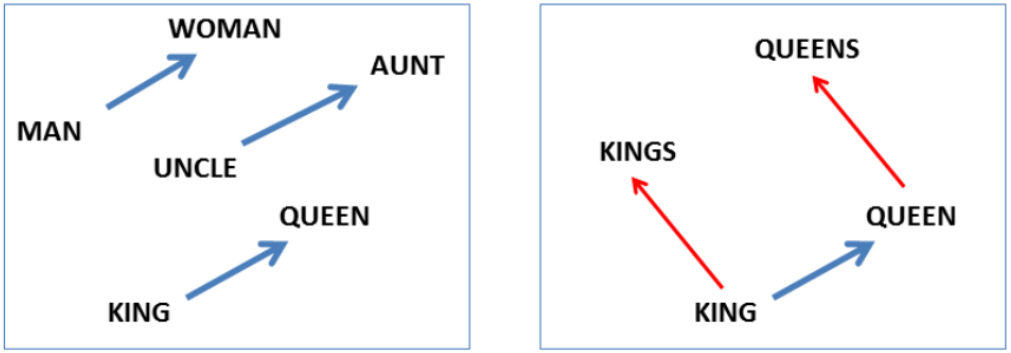


Figure 2: Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In high-dimensional space, multiple relations can be embedded for a single word.

:: DNN 을 통해 Symbol 을 공간상에 Mapping 가능하게 됨으로써 Symbol 들 간의 관계를 '수학적' 으로 추측해 볼 수 있는 여지가 있음

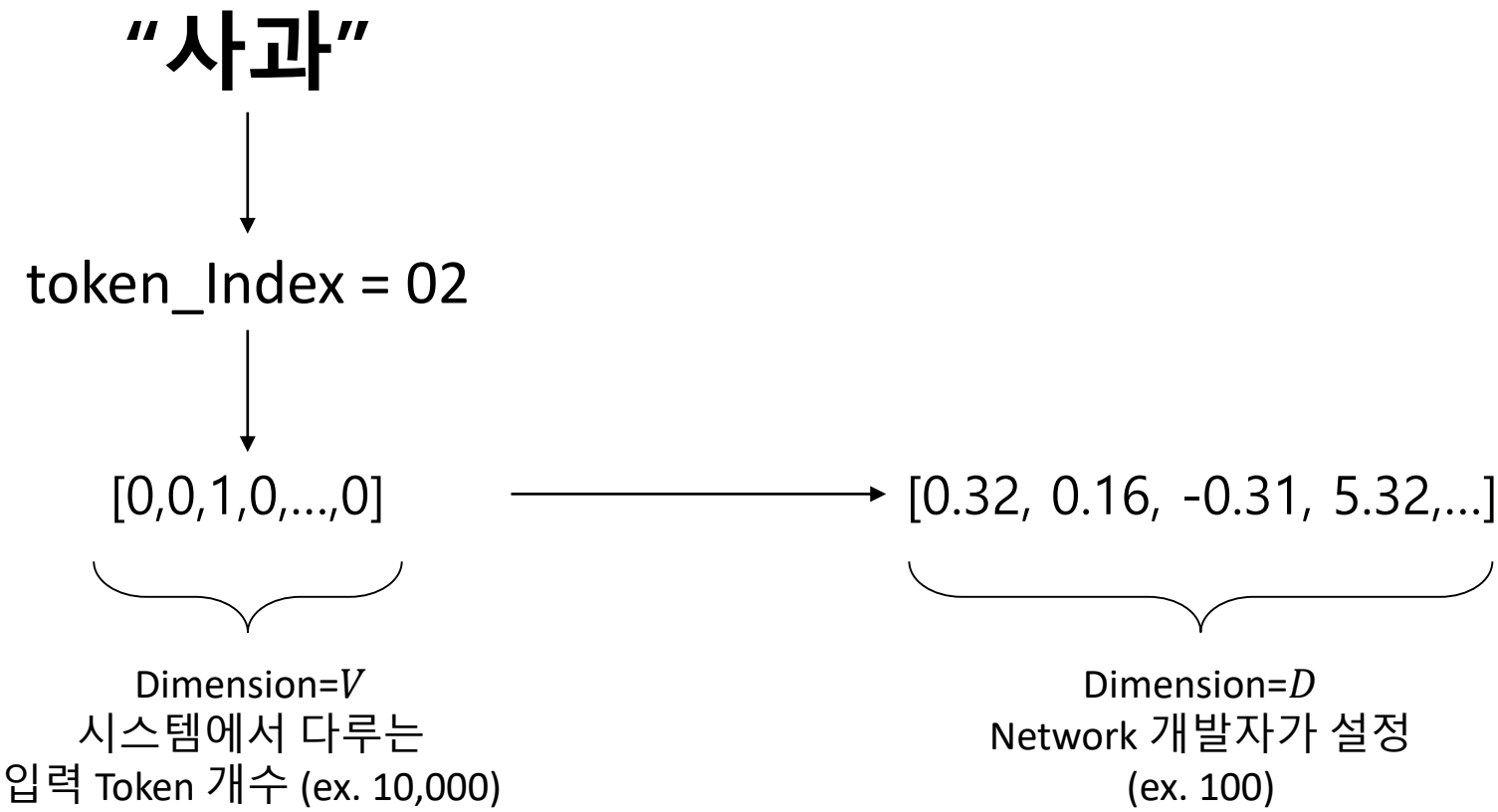
Ex) King – Man + Woman ≈ Queen

:: List of Number 가 Semantic Meaning 을 포함하고 있음을 의미

Microsoft, "Linguistic Regularities in Continuous Space Word Representations", 2013

Token Embedding (1)

- ✓ 언어마다, 혹은 다루는 데이터마다 자연어처리 단위가 달라 질 수 있음
- ✓ 경우에 따라 Word 일수도, Character 일 수도 있음
- ✓ 이를 위해 처리 단위를 **Token** 이라는 용어 사용



[remind] matrix multiplication

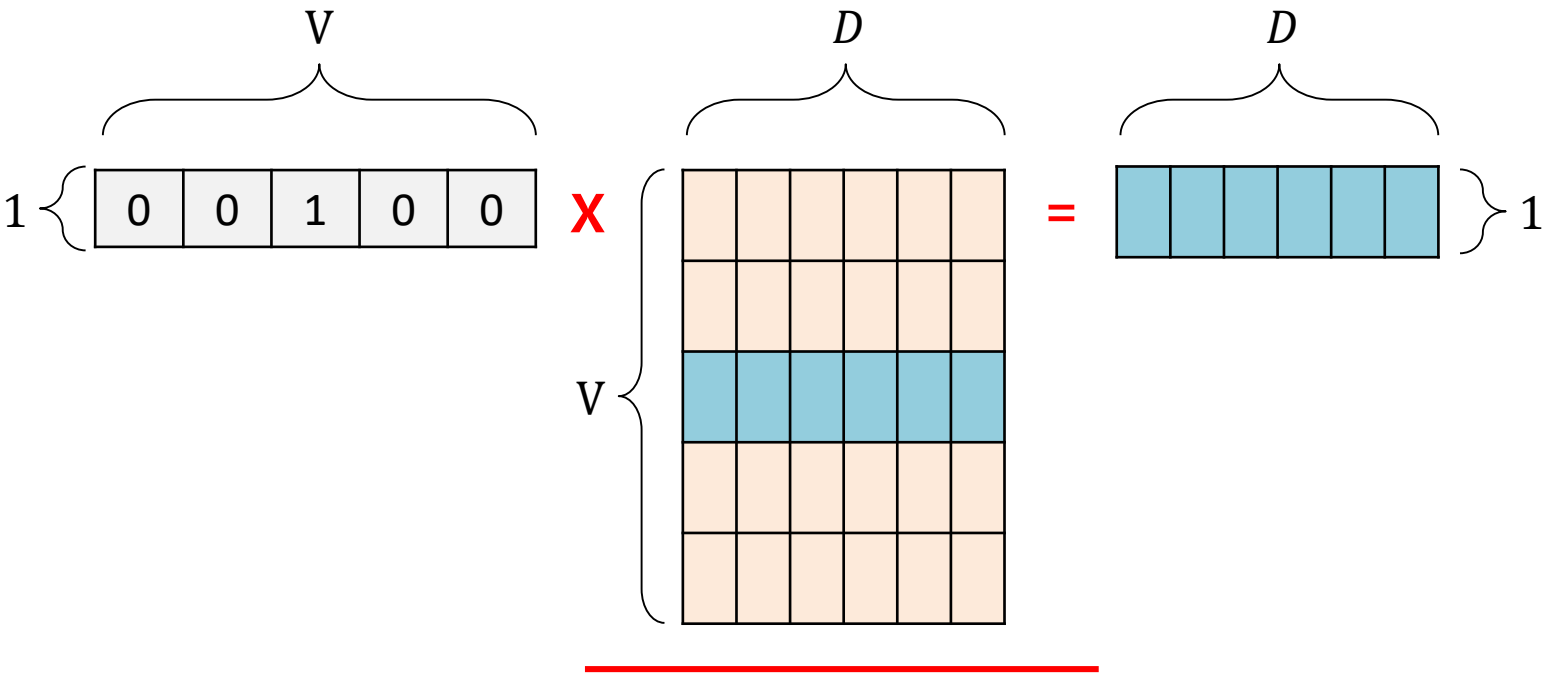
$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mk} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & \cdots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{k1} & B_{k2} & \cdots & B_{kn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \cdots & C_{kn} \end{bmatrix}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21} + \ldots + A_{1k}B_{k1}$$

Image from : https://serendip3.wordpress.com

Token Embedding (2)

Embedding Lookup 은 Matrix multiplication 을 통해 구현



- ✓ 이 Matrix ($[V, D]$) 를 하는 것이 바로 Token Embedding 기술
- ✓ 이러한 Token Embedding matrix 를 미리 학습(pretrained) 하여 배포해 놓은 것이 있다면, 다운로드하여 사용할 수 있음

- ✓ Embedding matrix 까지 학습시키는 것을 Neural Network 에 포함시킬 수도 있고, 아니면 기존의 것을 단순 연결하여 사용할 수도 있음

구현

감성분석

감성분석

- 다음에 또 가려 구요! **Positive**
- 이게 좀 비좁은 느낌을 가져다 줄지도! **Negative**
- 역시나 비싼 호텔인가 싶었어요. **Neutral**
- 무료 Wi-Fi **Objective**

Class 설계 (4개 Class)

- ✓ Positive
- ✓ Negative
- ✓ Neutral
- ✓ Objective

감성 분석 데이터 (1)

- https://air.changwon.ac.kr/
- [English] Won-Sik Bae and Jeong-Won Cha, Automatic Opinion Relations Extraction for Sentiment Analysis, Journal of KIISE: Software and Applications, vol. 40, no. 5, pp. 473-481, 2013. (in Korean)
- [Korean] 배원식, 차정원, “정서분석을 위한 의견관계 자동추출”, 한국정보과학회논문지: 소프트웨어 및 응용, 제40권, 제5호, pp. 473-481, 2013.
- 문장 길이가 128 character 미만인 것만 활용

감성 분석 데이터 (2)

N21 Dataset

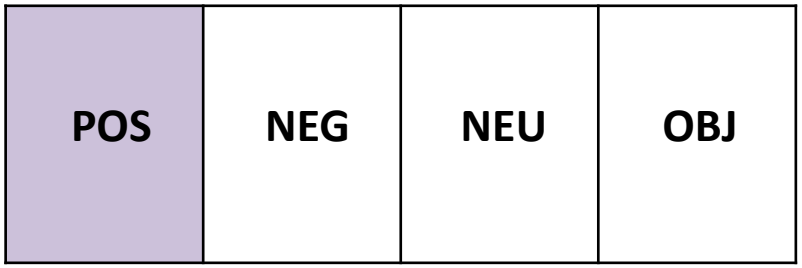
OBJ	플러툰(에프터눈티로유명한)호텔까지도 10분정도거리이며 열기구타는곳도 10분정도 도보위치에 있습니다.
POS	하지만 대체적으로 만족
OBJ	풀장
POS	정말 특별한 체험을 할 수 있을 것이다.
POS	인터넷은 무선, 유선 양쪽 다 좋았구요.
OBJ	한 사람에 평균 300이 들지 않았기 때문이다.
POS	일상의 피로를 날려버리기위한 장소로서는 최적인 곳이 아닐까 싶더라구요.
NEG	비싼가격에 낙후된 시설, 방에서 곰팡이 냄새같은 것이 남.
OBJ	또한 Nation 메트로 역에 가깝게 위치하고 있구요.
OBJ	개선할 점.
OBJ	하지만 우리는 호텔 내의 business center를 이용해서 비행기 시간 등을 체크할 수 있었습니다.
NEU	역에서 가깝고 새로운 호텔이라 처음으로 숙박했는데 신축이 아니라 맨션을 개조한 듯한 느낌이었습니다.
OBJ	여기도 빨리 예약할 필요가 있습니다.
NEG	입구는 호텔이라기보다 작은 오피스 빌딩 로비같은 인상이었습니다.

Tab 문자로 구분 (\t)

- Train Data : ./applications/sentiment_analysis/data/train.sent_data.txt
- Test Data : ./applications/sentiment_analysis/data/test.sent_data.txt

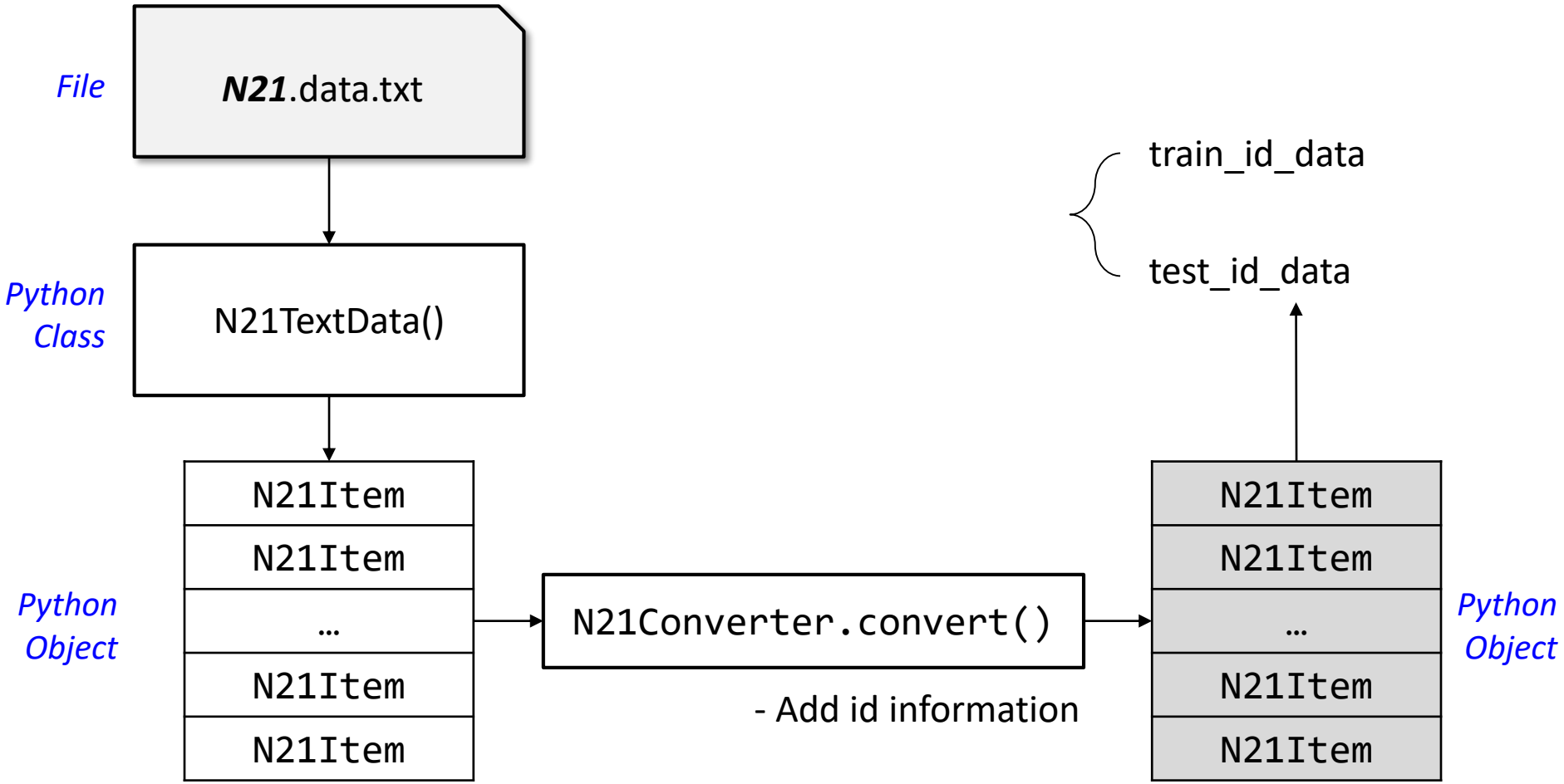
Reference Design

POS	인터넷은 무선, 유선 양쪽 다 좋았구요.



One-Hot
Representation

[N21] Data Processing Flow

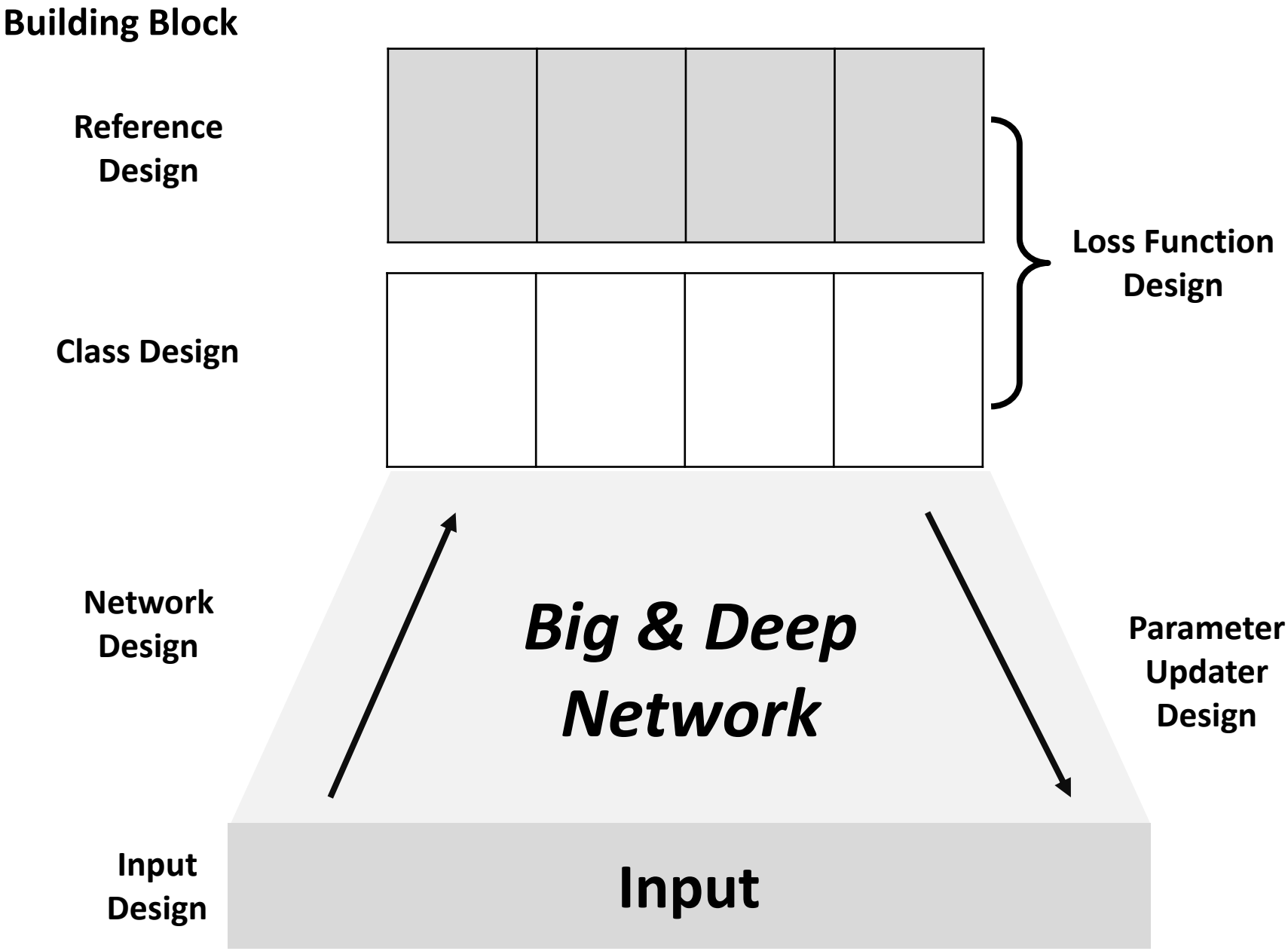


Code is available

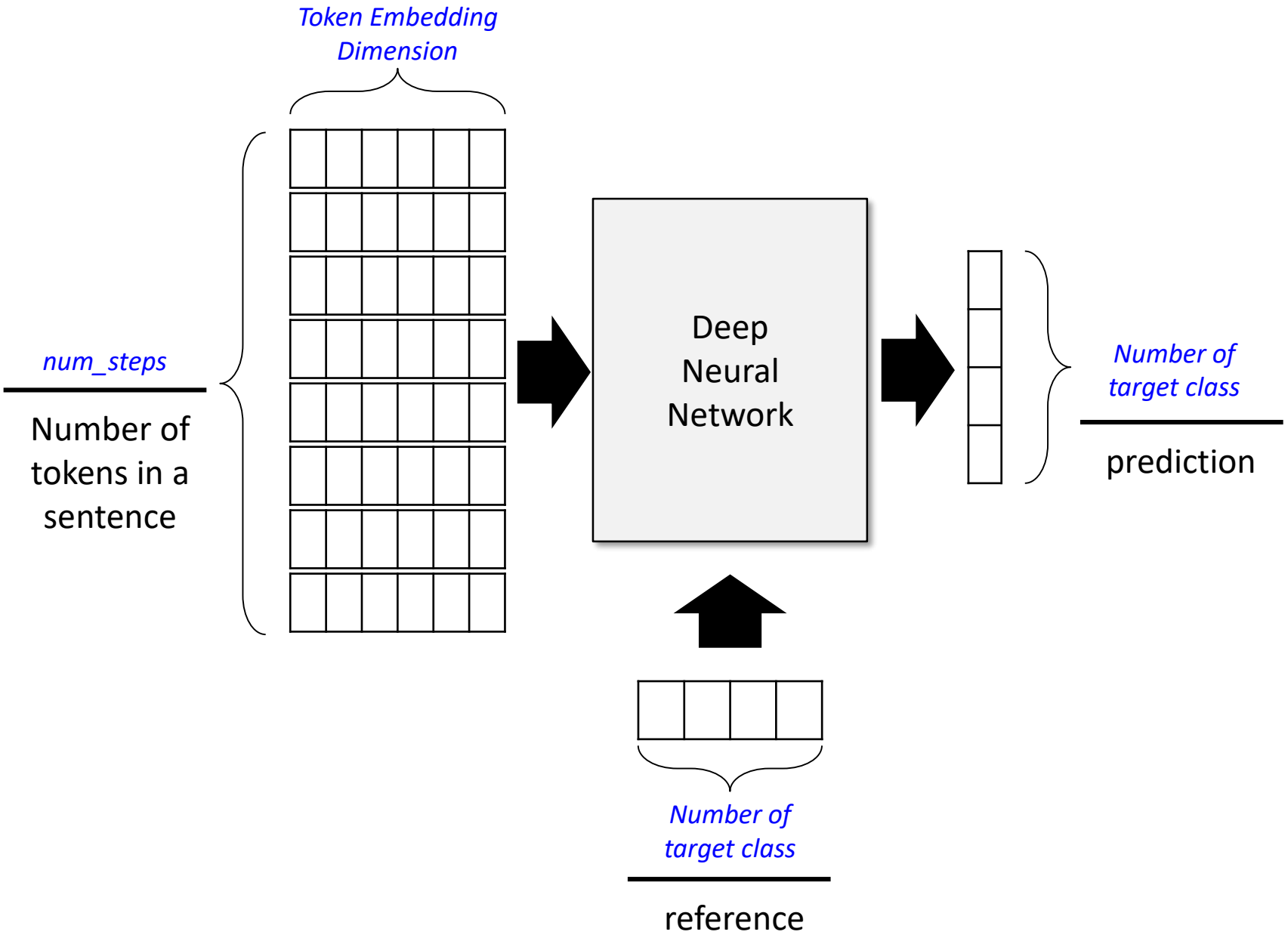
https://github.com/hugman/deep_learning/tree/master/course/nlp

./applications./sentiment_analysis/dataset : load_data()

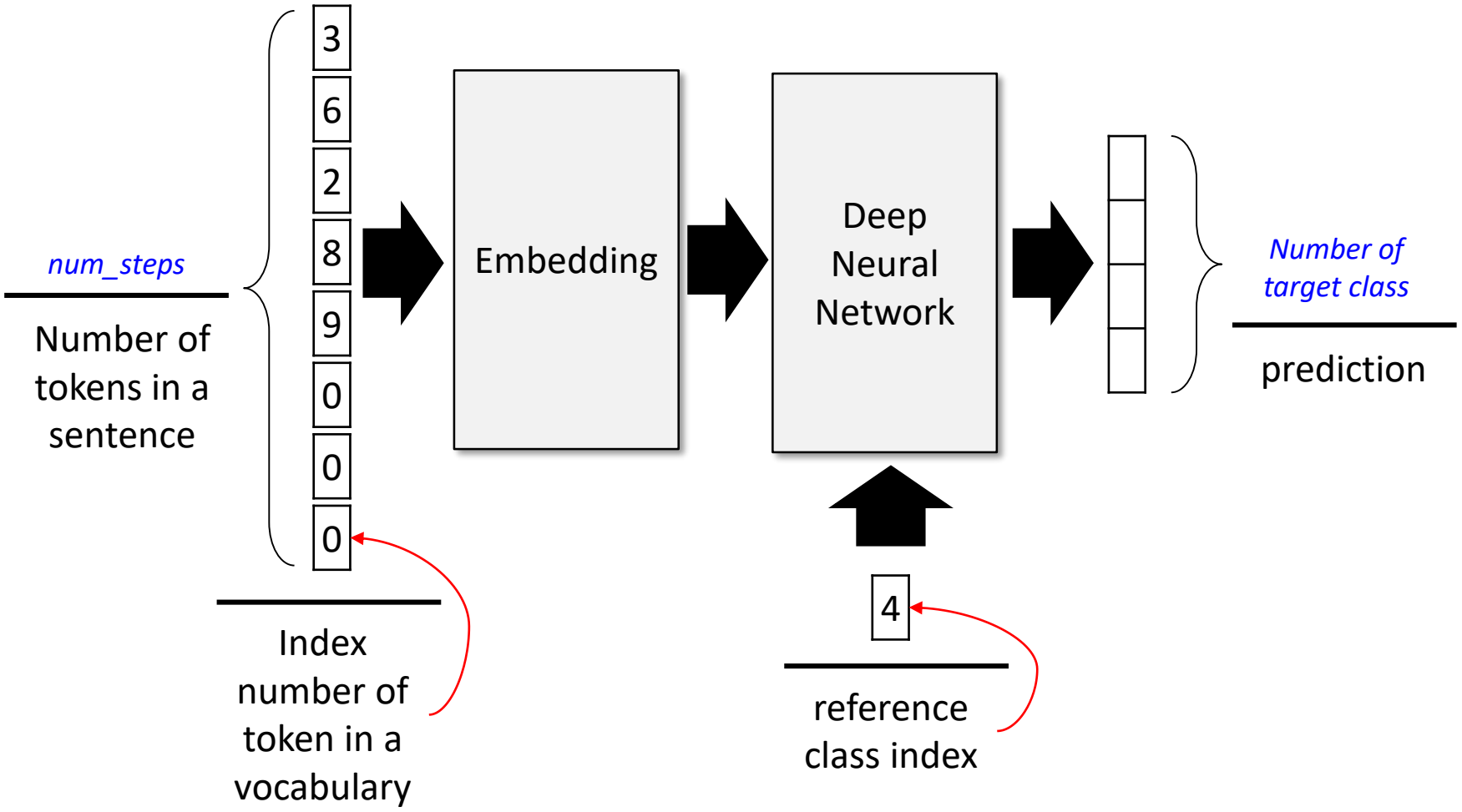
```
# vocab loader
token_vocab_fn  = os.path.join( os.path.dirname(__file__), 'data',
'token.vocab.txt')
token_vocab     = Vocab(token_vocab_fn, mode='token')
target_vocab_fn = os.path.join( os.path.dirname(__file__), 'data',
'target.vocab.txt')
target_vocab    = Vocab(target_vocab_fn, mode='target')

# load train data
train_data_fn  = os.path.join( os.path.dirname(__file__), 'data',
'train.sent_data.txt')
train_txt_data = N21TextData(train_data_fn)

# convert text data to id data
train_id_data  = N21Converter.convert(train_txt_data, target_vocab, token_vocab)
```


Building Block
Reference Design
Class Design
Loss Function Design
Network Design
Big & Deep Network
Parameter Updater Design
Input Design
Input

[N21 classification] DNN Interface

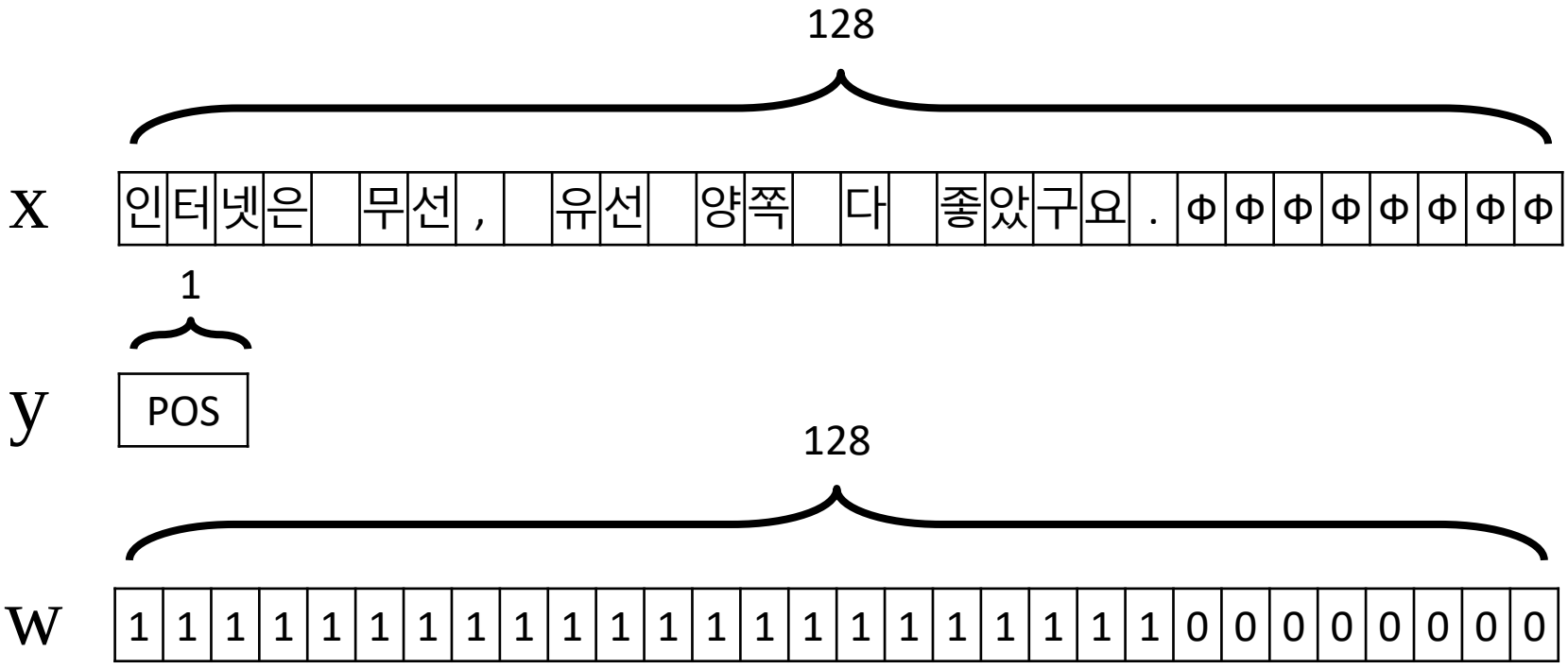


[N21 classification] DNN Interface with Embedding Library



Input Design

POS	인터넷은 무선, 유선 양쪽 다 좋았구요.

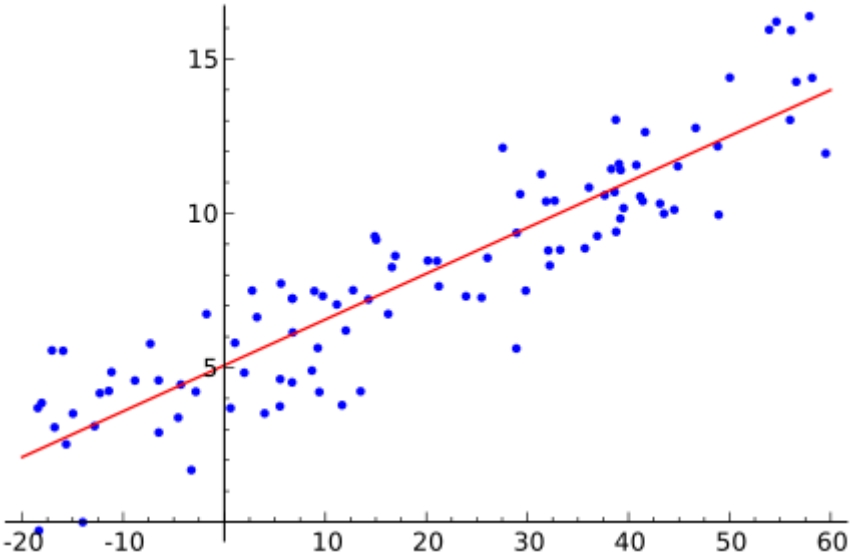


Φ : padding symbol
W : to mark padding positions

Batch Training – (batch_size)

핵심 질문 :

몇 개의 Example **을 살펴보고,** ***model 을 update*** **할 것인가?**



Parameter Update 를 한번
=
선을 다시 한번 새로 긋는것

↓

모든 점(훈련 예제)에 모두 잘 적용되는, 선을 찾는 것이 일반적

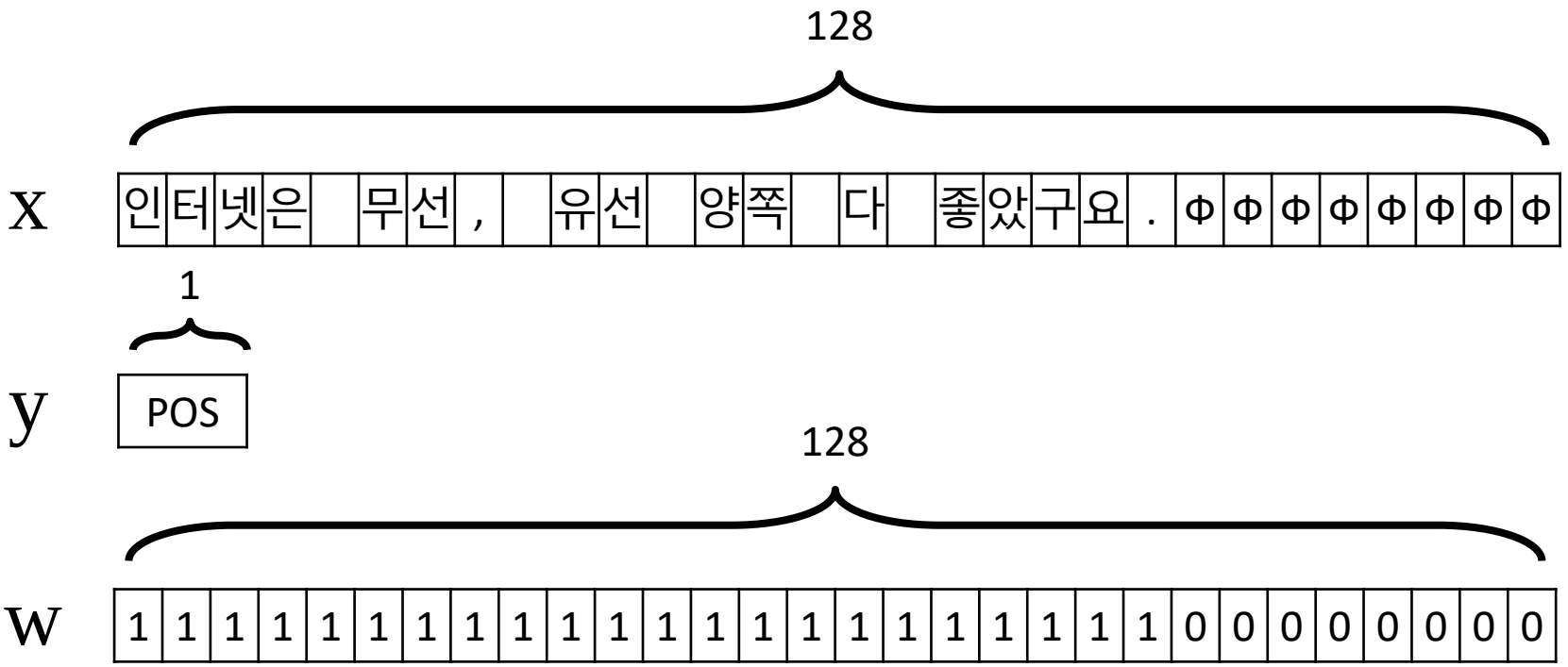
↓

$$\acute{\theta} = \theta + \alpha\Delta$$

Learning rate　　Update factor

https://en.wikipedia.org/wiki/Simple_linear_regression#/media/File:Linear_regression.svg

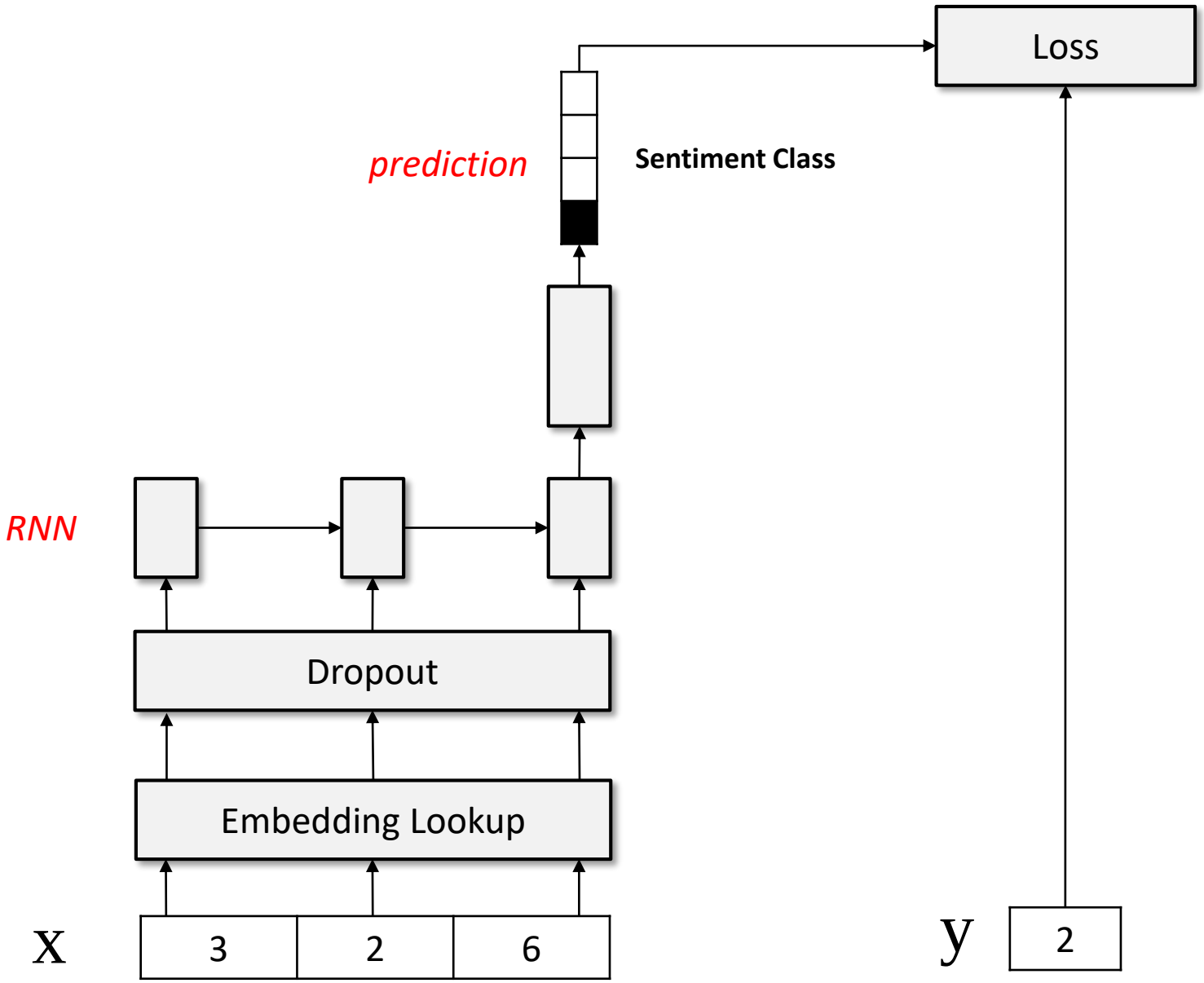
Tensorflow Interface Implementation



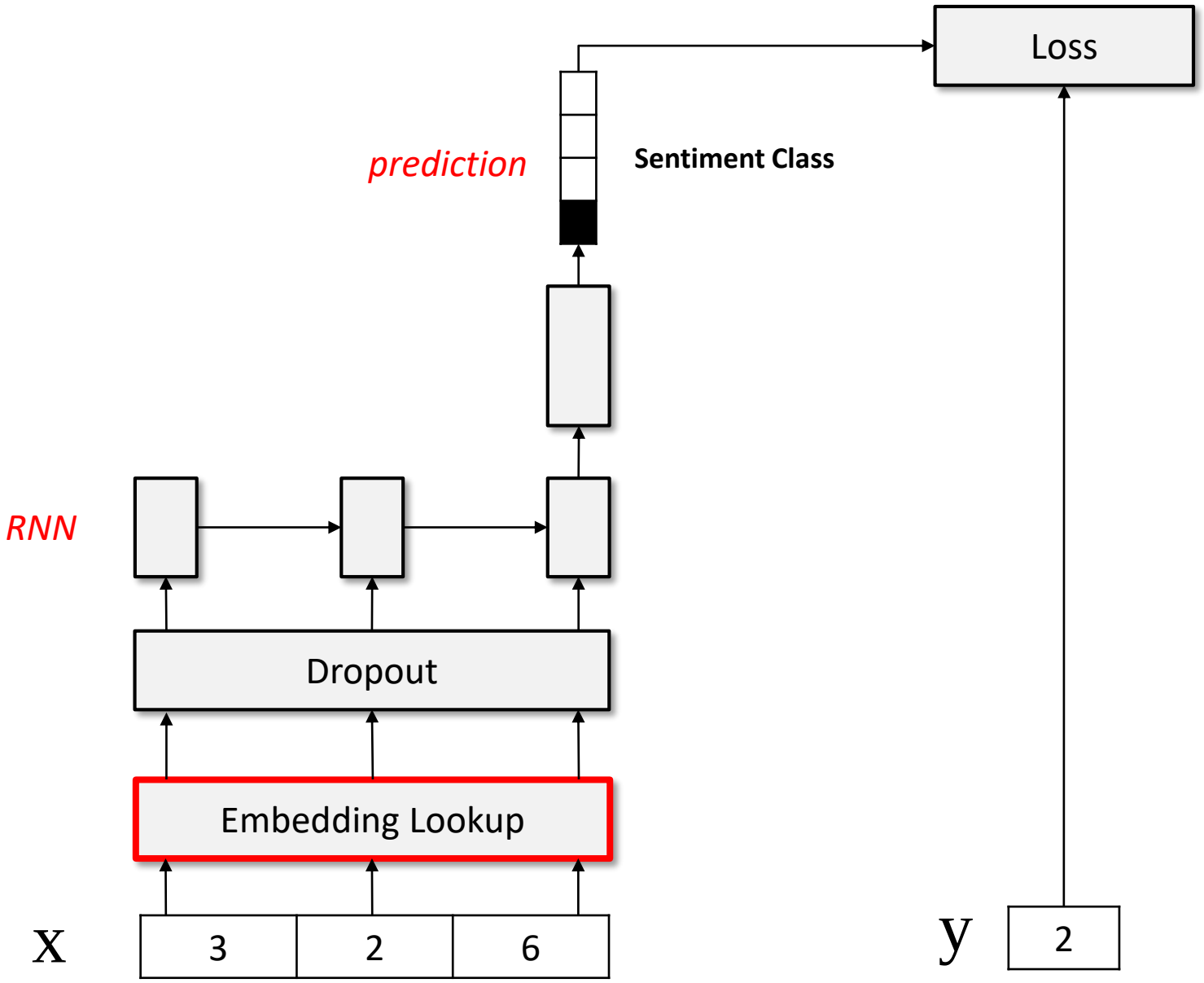

```
x = tf.placeholder(tf.int32, [None, hps.num_steps], name="pl_tokens")
y = tf.placeholder(tf.int32, [None                ], name="pl_target")
w = tf.placeholder(tf.int32, [None, hps.num_steps], name="pl_weight")
```

batch_size

Neural Network Design



Neural Network Design | embedding



Tensorflow Implementation | Embedding | Exercise

```
def _embedding(x)
```

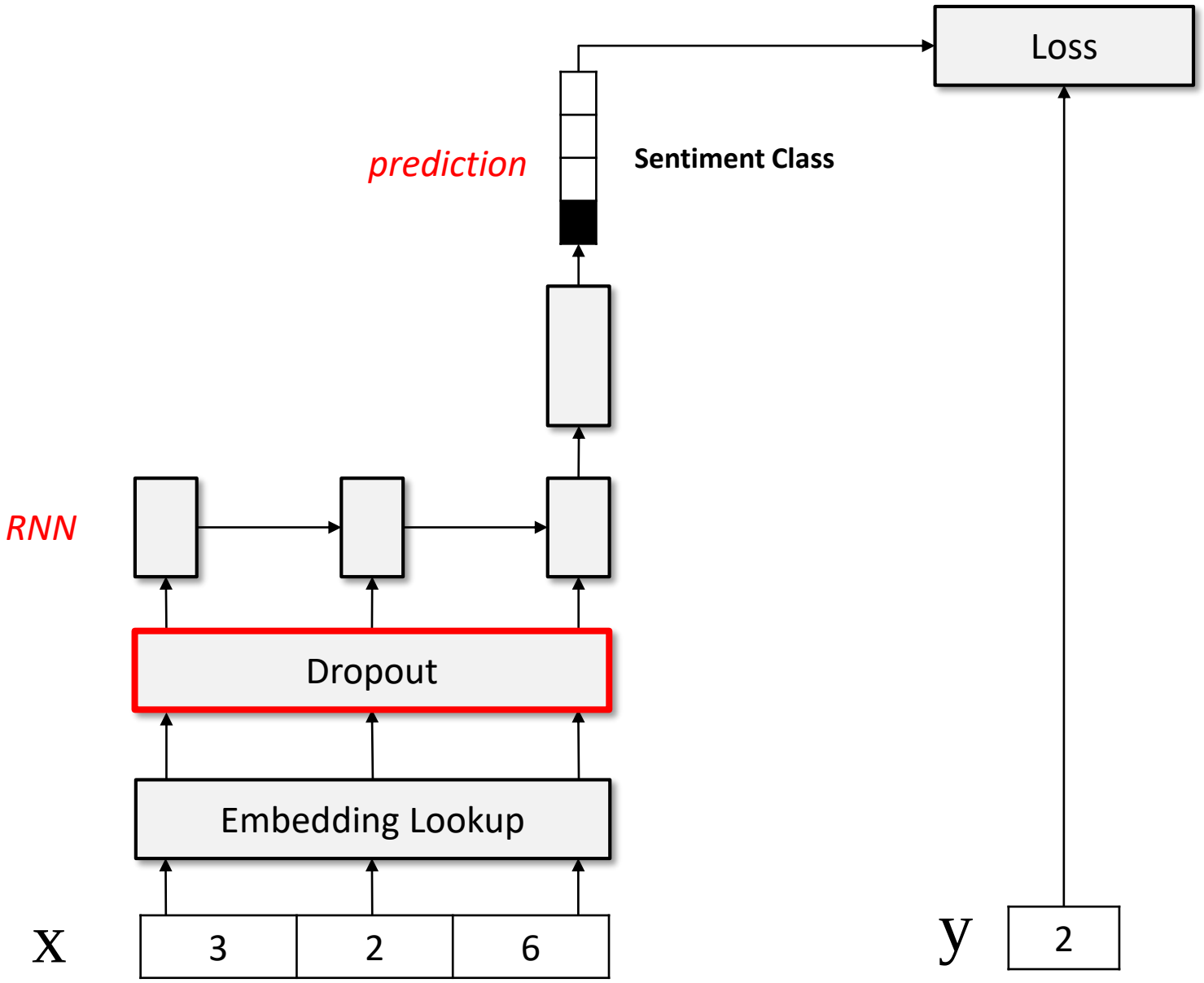
```
Input :  Tensor("model/pl_tokens:0", shape=(?, 128), dtype=int32)
Return :  a list of <tf.Tensor shape=(?, 50) dtype=float32>"
```

Keywords
tf.initializers.variance_scaling
tf.get_variable
tf.nn.embedding_lookup
tf.unstack

Tensorflow Implementation | Embedding

```
def _embedding(x)
    # character embedding
    shape       = [hps.vocab_size, hps.emb_size]
    initializer = tf.initializers.variance_scaling(distribution="uniform",
                                                   dtype=tf.float32)
    emb_mat     = tf.get_variable("emb", shape,
                                  initializer=initializer,
                                  dtype=tf.float32)

    # [batch_size, sent_len, emb_dim]
    input_emb   = tf.nn.embedding_lookup(emb_mat, x)

    # split input_emb -> num_steps
    step_inputs = tf.unstack(input_emb, axis=1)
```

Neural Network Design | dropout



Tensorflow Implementation | Dropout | Exercise

```
def _sequence_dropout(step_inputs, keep_prob)
    # apply dropout to each input
    # input : a list of input tensor which shape is [None, input_dim]
    with tf.name_scope('sequence_dropout') as scope:

            << Implement this part >>

    return step_outputs
```

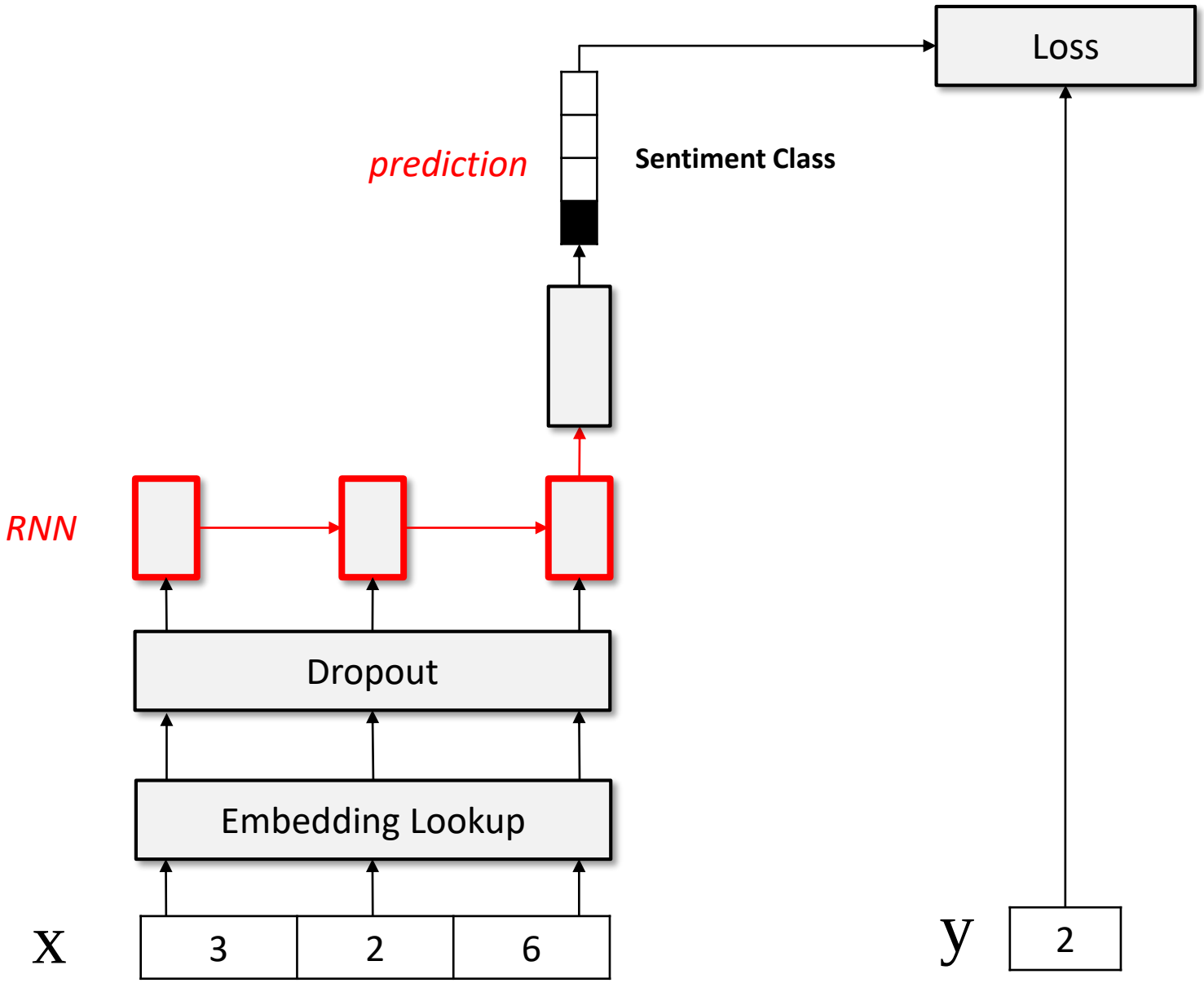
Return : a list of <tf.Tensor shape=(?, 50) dtype=float32>"

Keywords
tf.nn.dropout

Tensorflow Implementation | Dropout

```
def _sequence_dropout(step_inputs, keep_prob)
    # apply dropout to each input
    # input : a list of input tensor which shape is [None, input_dim]
    with tf.name_scope('sequence_dropout') as scope:
            step_outputs = []
            for t, input in enumerate(step_inputs):
                    step_outputs.append( tf.nn.dropout(input, keep_prob) )
    return step_outputs
```

Neural Network Design | RNN based N21 encoding



Tensorflow Implementation | RNN based N21 encoding | Exercise

```
def sequence_encoding_n21_rnn(step_inputs, cell_size, scope_name)
```

```
Input :  a list of <tf.Tensor shape=(?, 50), dtype=float32>
Return :  a list of <tf.Tensor, shape=(?, 100)>
```

Keywords
tf.contrib.rnn.GRUCell
tf.contrib.rnn.static_rnn

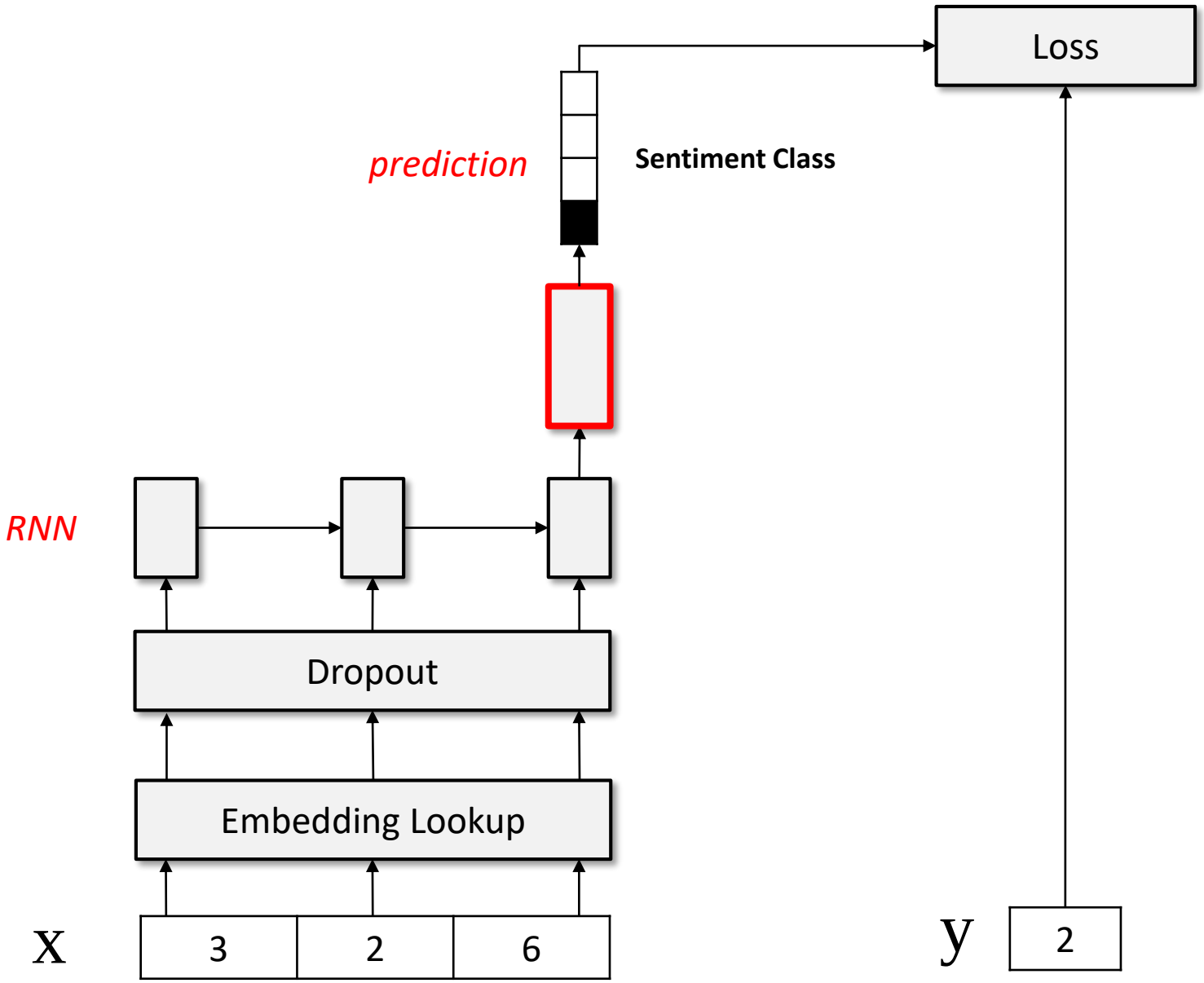
Tensorflow Implementation | RNN based N21 encoding

```
def sequence_encoding_n21_rnn(step_inputs, cell_size, scope_name)
    # rnn based N21 encoding (GRU)
    step_inputs = list( reversed( step_inputs ) )
    f_rnn_cell = tf.contrib.rnn.GRUCell(cell_size, reuse=None)
    _inputs    = tf.stack(step_inputs, axis=1)
    step_outputs, final_state = tf.contrib.rnn.static_rnn(f_rnn_cell,
                                                          step_inputs,
                                                          dtype=tf.float32,
                                                          scope=scope_name
                                                         )
    out = step_outputs[-1]
    return out
```

Why *reversed(step_inputs)* ?

: To place padding symbols in the left part of RNN inputs

Neural Network Design | to class



Tensorflow Implementation | to class | Exercise

```
def _to_class(input, num_class)
```

```
Input :  tf.Tensor, shape=(?, 100), dtype=float32)
Return :  tf.Tensor, shape=(?, 4), dtype=float32
```

Keywords
tensorflow.contrib.layers.python.layers.linear

Tensorflow Implementation | to class

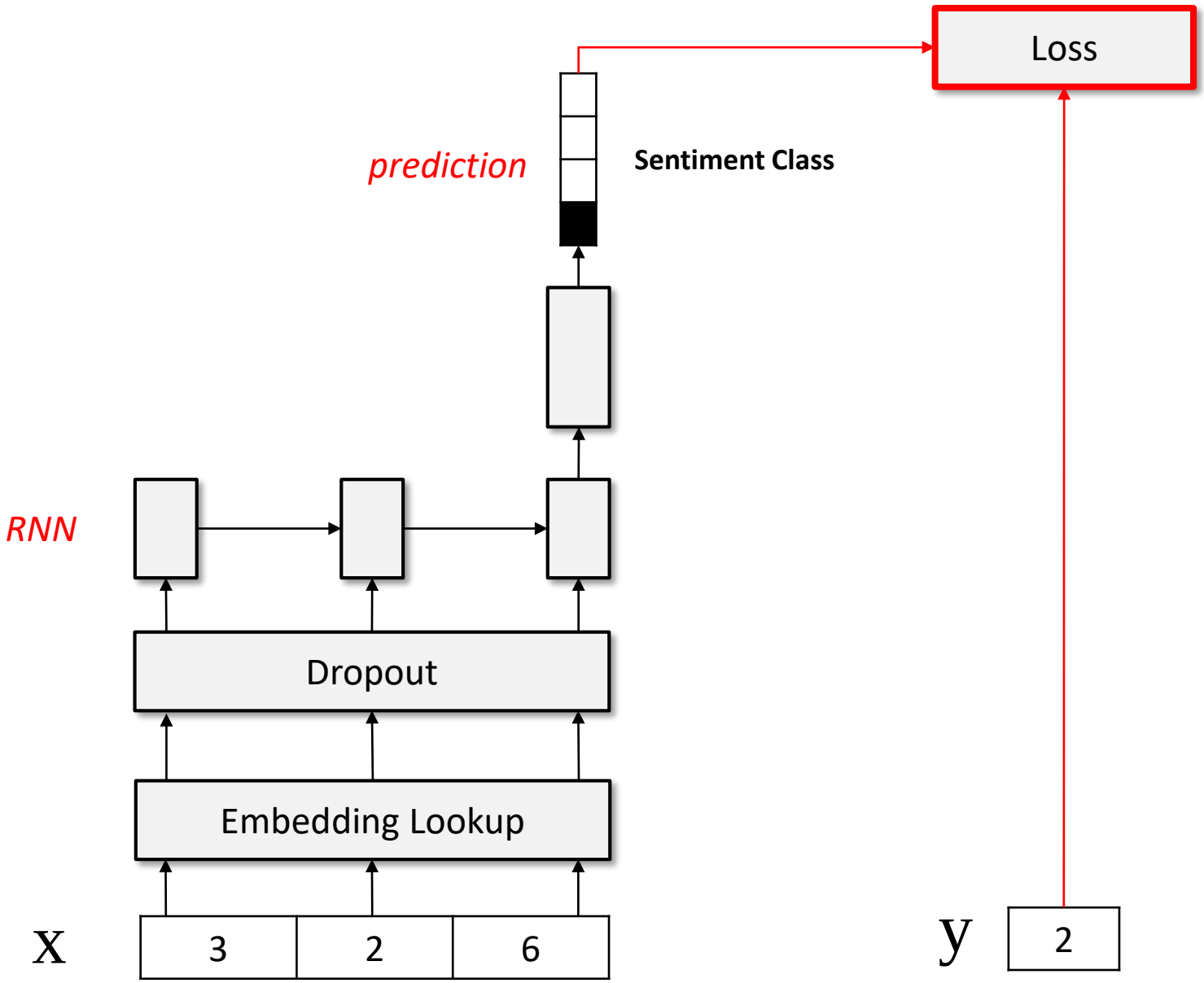
```
def _to_class(input, num_class)
```

```
    # out = [batch_size, 4]
    out = linear(input, num_class, scope="Rnn2Sentiment")
    return out
```

```
# output of the neural network
```

```
    out_probs = tf.nn.softmax(out, name="out_probs")
    out_pred  = tf.argmax(out_probs, 1, name="out_pred")
```

Neural Network Design | Loss calculation



Tensorflow Implementation | loss calculation | Exercise

```
def _loss(out, ref)
```

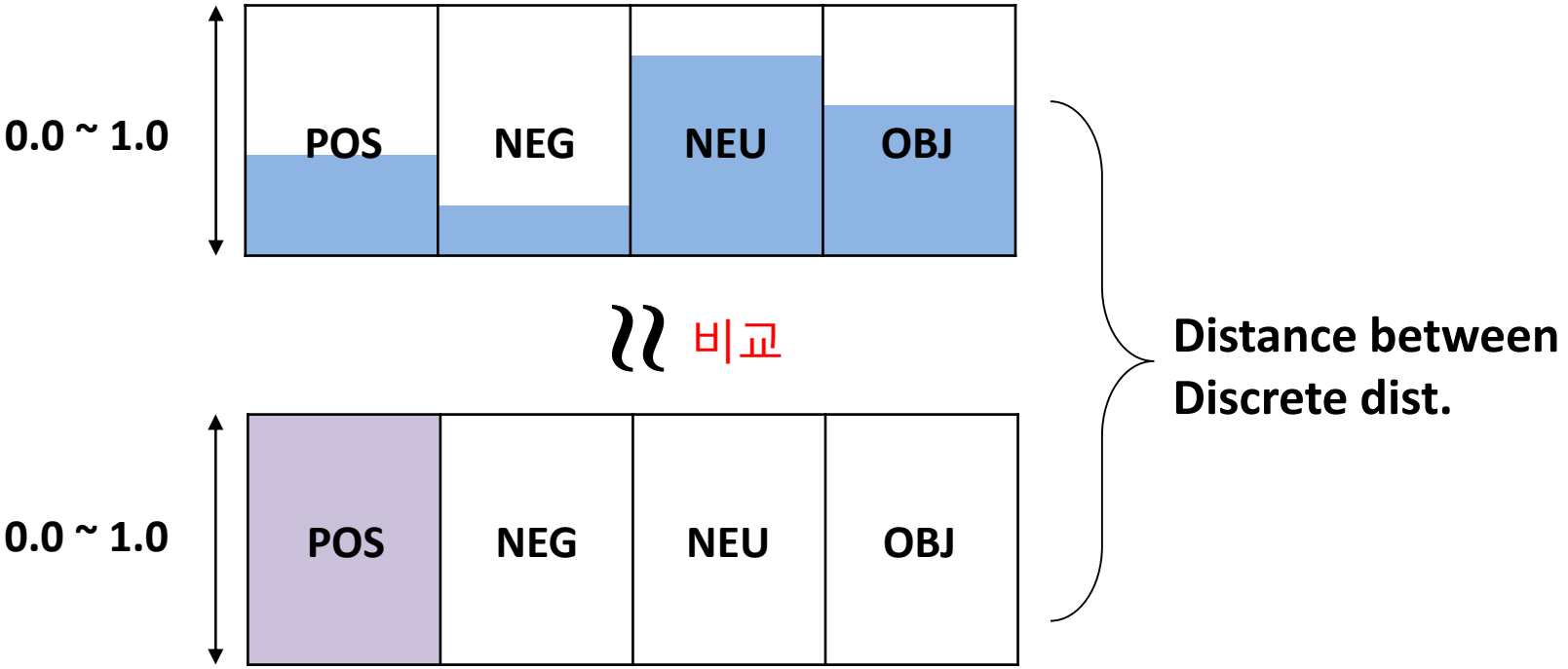
```
Input out:  tf.Tensor, shape=(?, 4)
Input ref:  Input out:  tf.Tensor, shape=(?, 4)

Return :  tf.Tensor, shape=(), dtype=float32
```

Keywords
tf.nn.sparse_softmax_cross_entropy_with_logits

Tensorflow Implementation | loss calculation

```
def _loss(out, ref)
    # out : [batch_size, num_class] float - unscaled logits
    # ref : [batch_size] integer
    # calculate loss function using cross-entropy
    batch_loss = tf.nn.sparse_softmax_cross_entropy_with_logits(
                                    logits=out,
                                    labels=ref,
                                    name="sentiment_loss") # [batch_size]
    loss = tf.reduce_mean(batch_loss)
    return loss
```

Loss Function Design



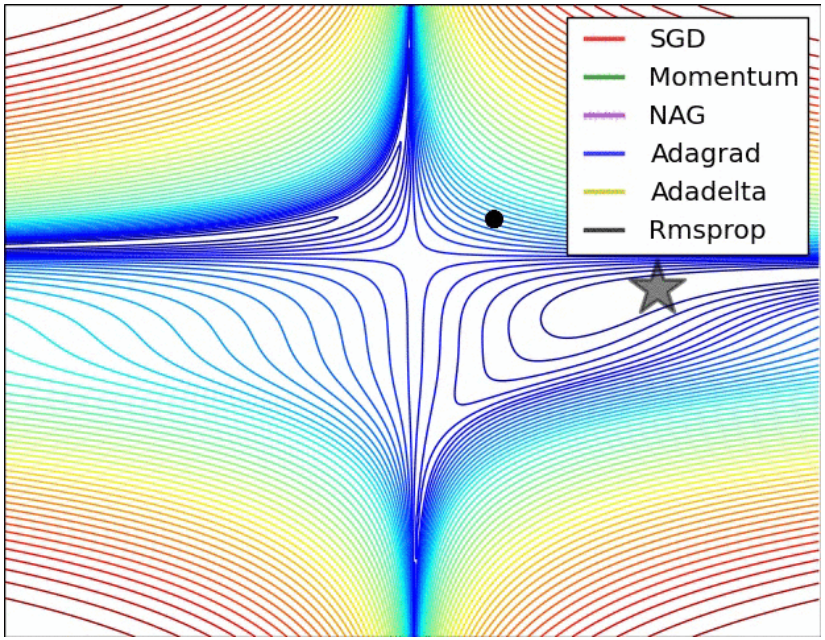
- ✓ Cross Entropy
- ✓ https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/api_docs/python/functions_and_classes/shard4/tf.nn.sparse_softmax_cross_entropy_with_logits.md

Tensorflow Implementation | parameter update

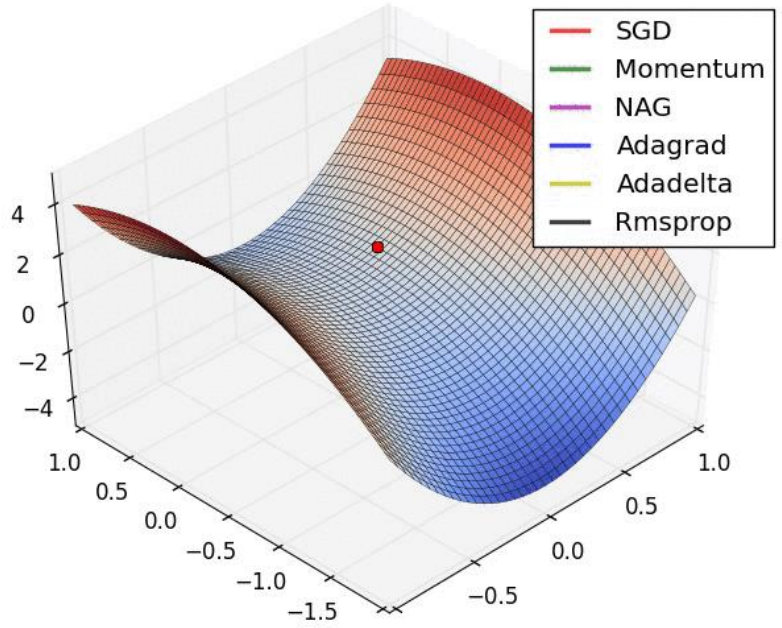
```
# optimizer settings

  optimizer       = tf.train.AdamOptimizer(hps.learning_rate)
  self.train_op   = optimizer.minimize(self.loss,
                                       global_step=self.global_step)
```

- ***train_op*** should be called outside of the network to update parameters

Parameter Updater Design



Left: Contours of a loss surface and time evolution of different optimization algorithms.

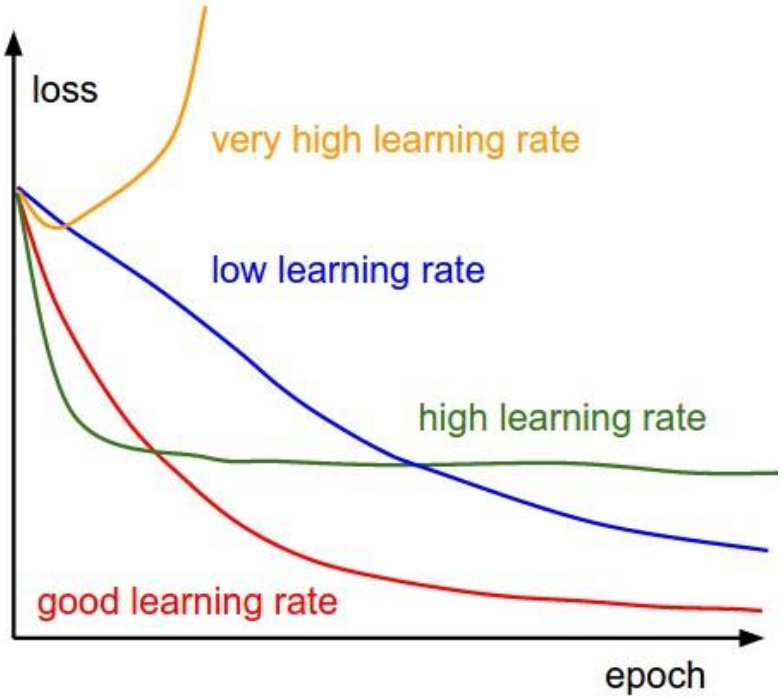


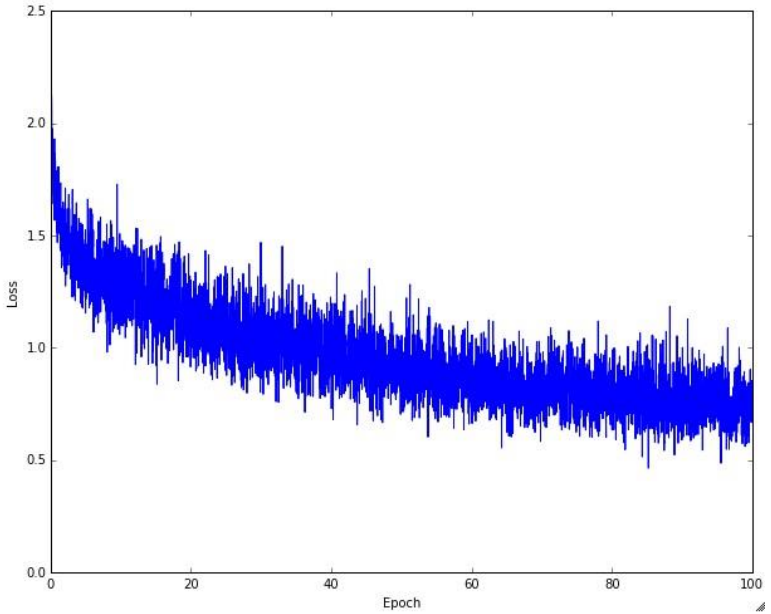
Right: A visualization of a saddle point in the optimization landscape, where the curvature along different dimension has different signs (one dimension curves up and another down).

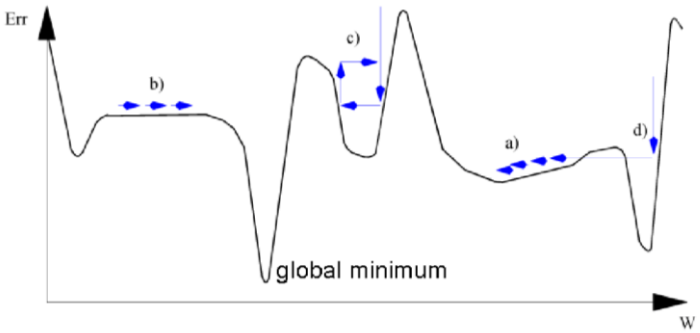
Image credit : Alec Radford

http://cs231n.github.io/neural-networks-3/

Tuning Learning Rate

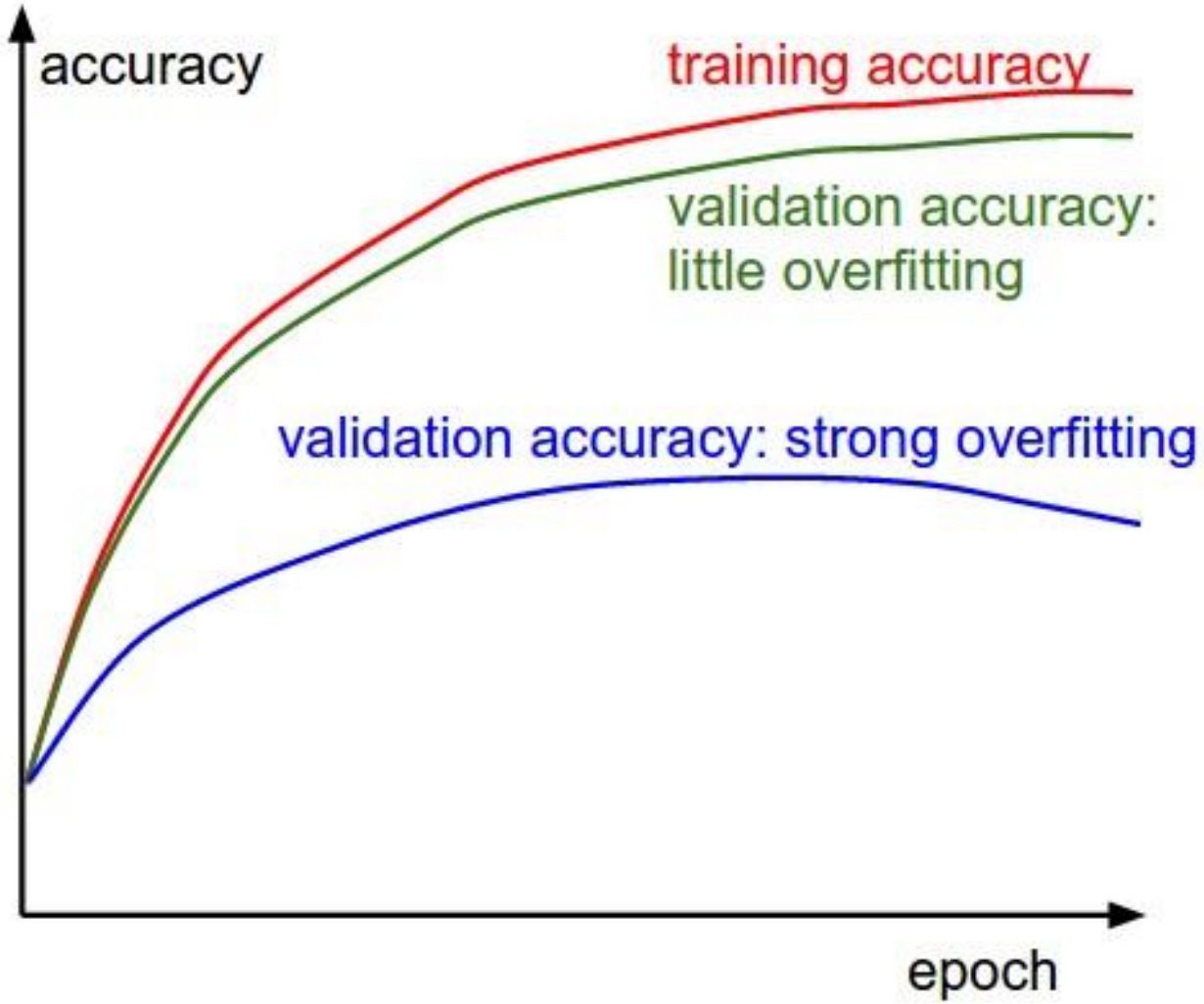






http://cs231n.github.io/neural-networks-3/

Train / Validation Accuracy



http://cs231n.github.io/neural-networks-3/

Training Process

```
# training process
while not sv.should_stop():
        fetches = [model.global_step, model.loss, model.train_op]
        a_batch_data = next( train_data_set.iterator )
        y, x, w = a_batch_data
        fetched = sess.run(fetches, {
                                                model.x: x,
                                                model.y: y,
                                                model.w: w,

                                                model.keep_prob: hps.keep_prob,
                                            }
                                )

        local_step += 1

        _global_step = fetched[0]
        _loss        = fetched[1]
```

Q/A

감사합니다.

Lecture Blog: www.hugman.re.kr
Lecture Video: https://goo.gl/7NL5hV
Lecture Slides: https://goo.gl/6NfR1V
Code Share: https://github.com/hugman
Facebook: https://goo.gl/1RML3C

정상근, Ph.D

Intelligence Architect

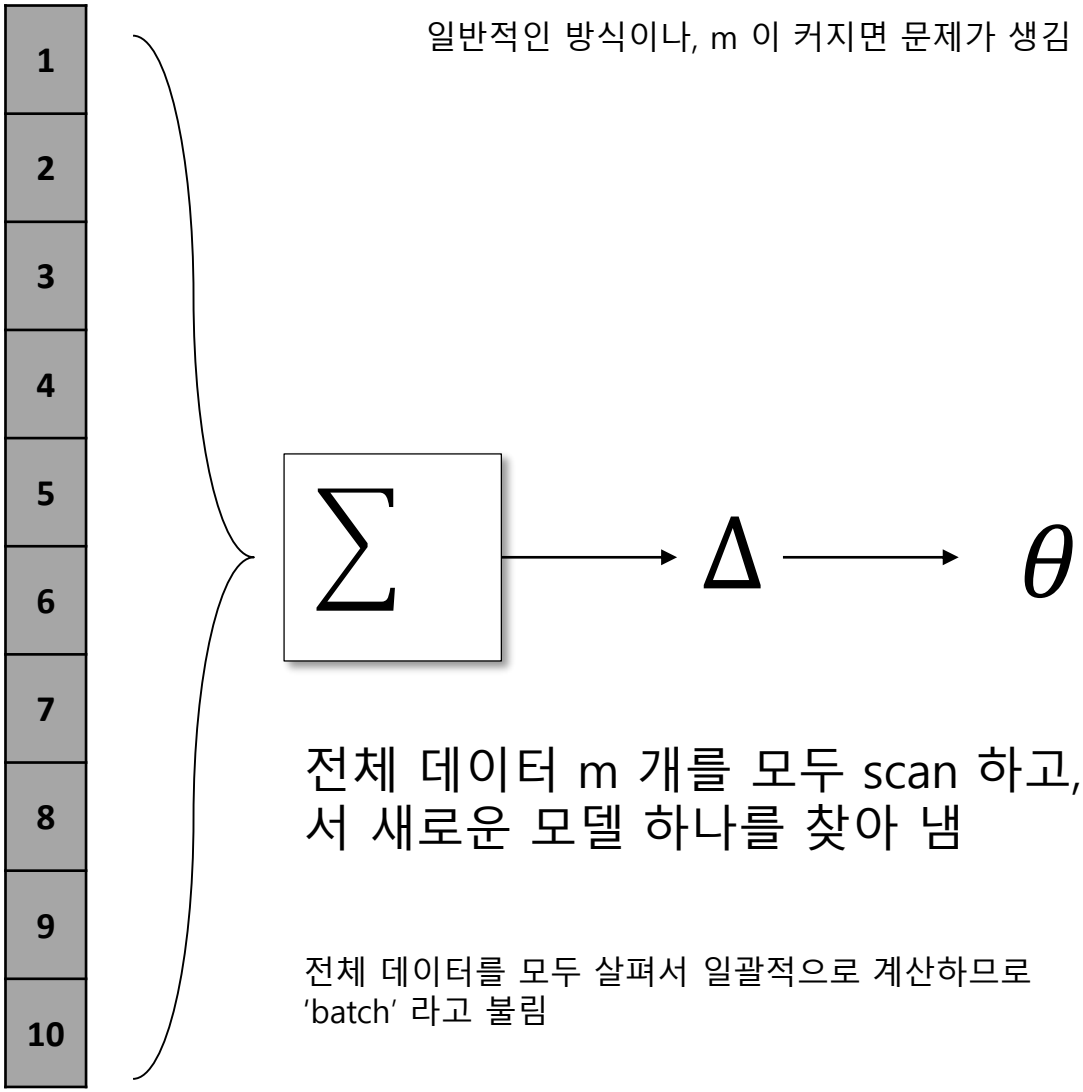
Senior Researcher, AI Tech. Lab. SKT Future R&D
Contact : hugmanskj@gmail.com, hugman@sk.com

Training – Batch Gradient Descent

m=10

Batch Gradient Descent

일반적인 방식이나, m 이 커지면 문제가 생김

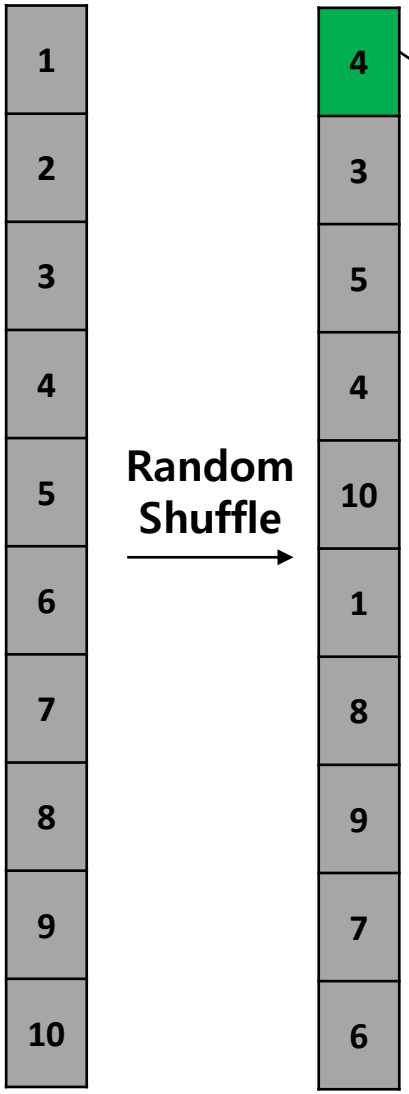


전체 데이터 m 개를 모두 scan 하고, 계산해서 새로운 모델 하나를 찾아 냄

전체 데이터를 모두 살펴서 일괄적으로 계산하므로 'batch' 라고 불림

Training – Stochastic Gradient Descent 방식

m=10



Stochastic Gradient Descent

한번의 iteration 에 오직 1개의 예제만 사용함

- 매 example 마다 그것에 맞는 방식으로 편향적으로 update 되므로 global minimum 을 찾을 수 없음
- 그러나 충분히 많은 데이터에서는, 이런 방식이 잘 작동함이 실험적으로 증명됨

오직 하나의 예제만 사용하여 계산

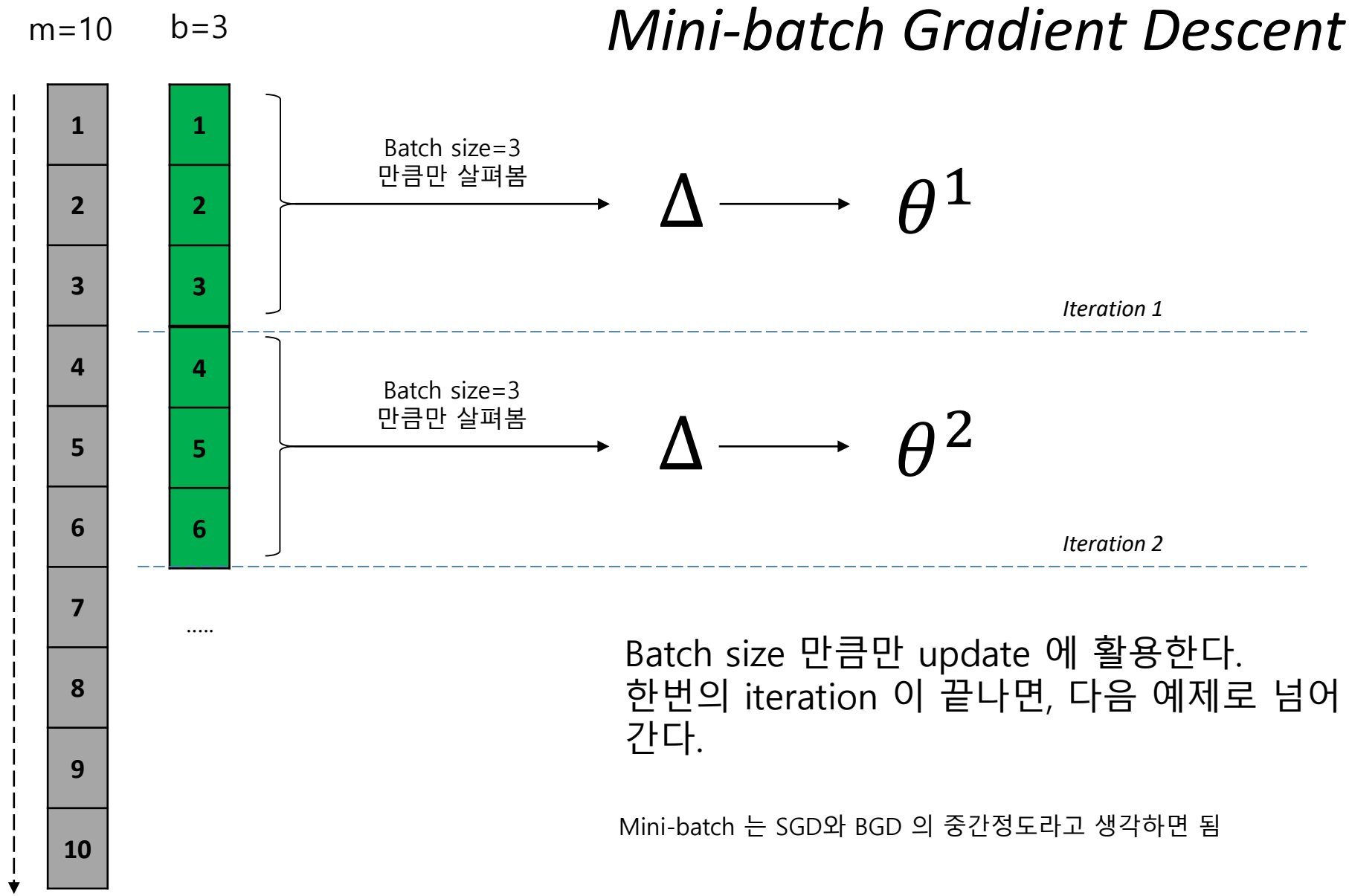
$\Delta \longrightarrow \theta$

매 훈련 iteration 마다

1) Data 를 randomly shuffle 하고
2) 가장 첫 번째 예제만 이용해서 update factor 를 찾아냄
3) 이 factor 를 이용해서 새로운 모델을 만듦

Shuffle 은 단순히 stochastically training example 을 selection 한다는 의미 정도로 생각하면 됨

Training – Mini-batch Gradient Descent 방식



Batch size 만큼만 update 에 활용한다.
한번의 iteration 이 끝나면, 다음 예제로 넘어간다.

Mini-batch 는 SGD와 BGD 의 중간정도라고 생각하면 됨

Training – Batch Gradient / Stochastic gradient / Mini-batch gradient

Batch Gradient Descent : use all ***m*** examples in each iteration

Stochastic Gradient Descent : use ***1*** example in each iteration

Mini-batch Gradient Descent : use ***b*** examples in each iteration

https://class.coursera.org/ml-003/lecture/view?lecture_id=106