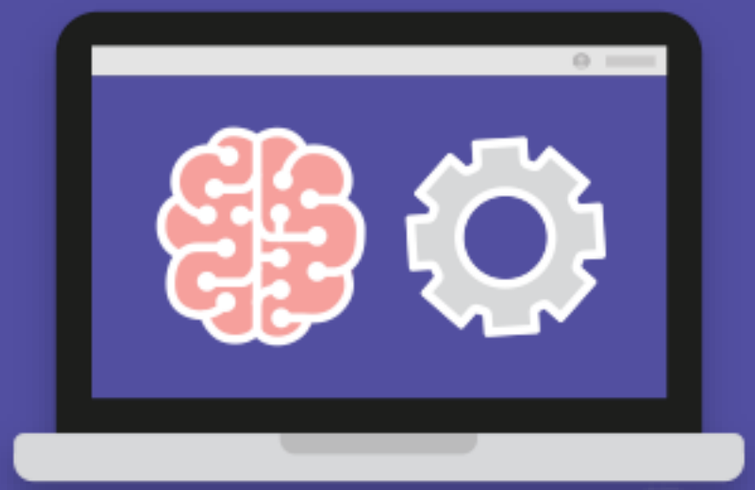
/* elice */

인공지능/머신러닝 기초

Module 11: 텐서플로우와 인공신경망



텐서플로우
(Tensorflow)

텐서플로우 (Tensorflow)

유연하고, **효율적**이며, **확장성**이 있는 **딥러닝 프레임워크**
대형 **클러스터 컴퓨터**부터 **스마트폰**까지 다양한 디바이스에서 동작
지원하는 언어 : Python 2/3, C/C++ 등

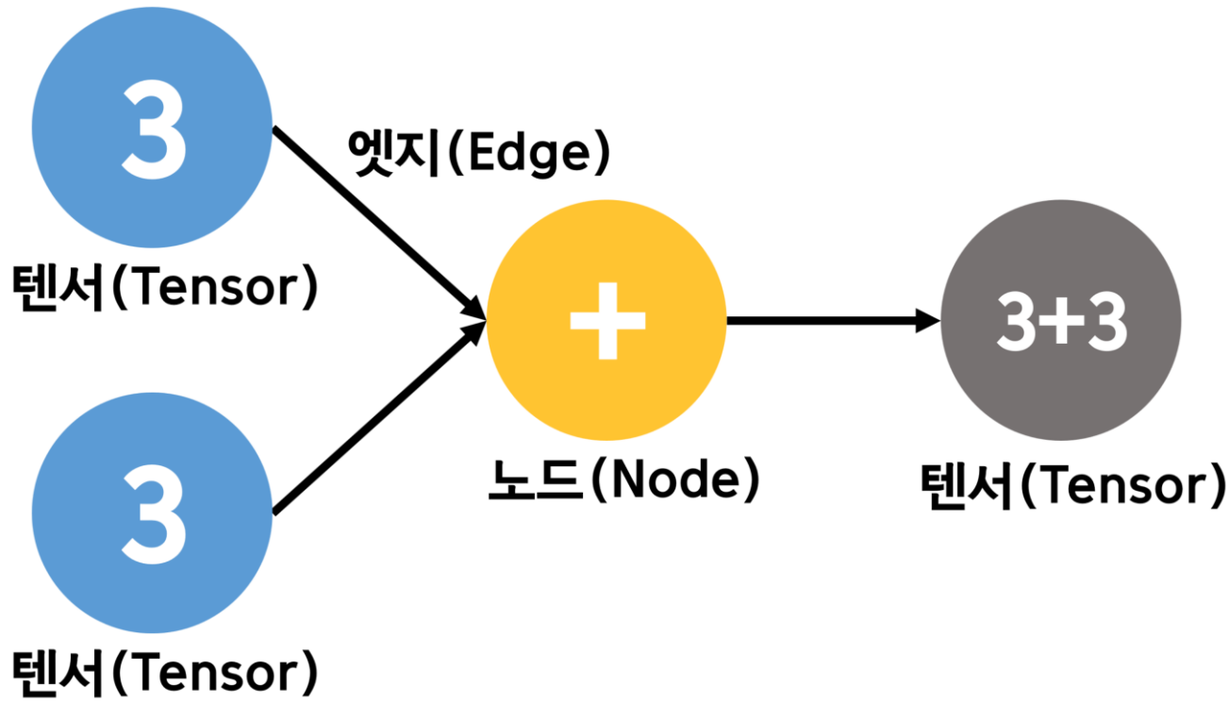
텐서(Tensor)?

Tensor = Multidimensional Arrays = Data

딥러닝에서 텐서는 다차원 배열로 나타내는 데이터

예를 들어, RGB 이미지는 삼차원 배열로 나타나는 텐서

플로우(Flow)



플로는 데이터의 흐름을 의미

텐서플로우에서 계산은 데이터 플로우 그래프로 수행

그래프를 따라 데이터가 노드를 거쳐 흘러가면서 계산

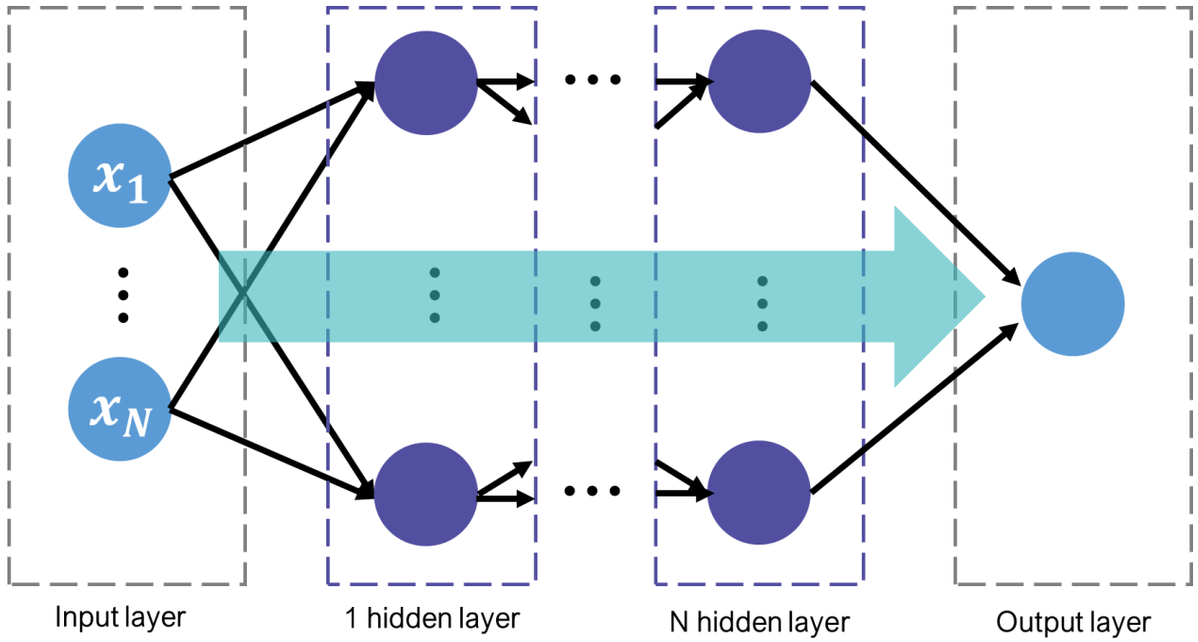
텐서 + 플로우

Tensor

feature	label
12, 13, 33	0
32, 25, 53	1
1, 36, 100	5
⋮	⋮

+

Flow



딥러닝에서 데이터를 의미하는 **텐서(tensor)**와
데이터 플로우 그래프를 따라 연산이 수행되는 **형태(Flow)**의 합

Tensorflow 기초 사용법

상수 선언 하기

```
import tensorflow as tf


# 상수형 텐서 선언

tensor_a = tf.constant(value, dtype=None, shape=None, name=None)
```

value : 반환되는 상수값

shape : Tensor의 차원

dtype : 반환되는 Tensor 타입

name : 상수 이름

상수 선언 하기

```
import tensorflow as tf


# 모든 원소 값이 0인 텐서 생성
tensor_b = tf.zeros(shape, dtype=tf.float32, name=None)


# 모든 원소 값이 1인 텐서 생성
tensor_c = tf.ones(shape, dtype=tf.float32, name=None)
```

시퀀스 선언 하기

```
import tensorflow as tf


# start에서 stop까지 증가하는 num 개수 데이터
tensor_d = tf.linspace(start, stop, num, name=None)
```

start : 시작 값

stop : 끝 값

num : 생성할 데이터 개수

name : 시퀀스 이름

시퀀스 선언 하기

```
import tensorflow as tf


# start에서 stop까지 delta씩 증가하는 데이터
tensor_e = tf.range(start, limit=None, delta=None, name=None)
```

start : 시작 값

limit : 끝 값

delta : 증가량

name : 시퀀스 이름

난수 선언 하기

```
import tensorflow as tf


# 정규분포 생성
tensor_f = tf.random.normal(shape, mean=0.0, stddev=1.0,
                dtype=tf.float32, seed=None, name='normal')

# 균등분포 생성
tensor_g = tf.random.uniform(shape, minval=0, maxval=None,
                dtype=tf.float32, seed=None, name='uniform')
```

변수 선언 하기

```
import tensorflow as tf


# 변수 텐서 생성
tensor_f = tf.Variable(value, name=None)

# 일반적인 퍼셉트론의 가중치와 bias 생성
weight = tf.Variable(10)
bias = tf.Variable(tf.random.normal([10,10]))
```

텐서 연산자

```
import tensorflow as tf


# 단항 연산자
tf.negative(x)       # -x
tf.logical_not(x)    # !x
tf.abs(x)            # x의 절대값
```

텐서 연산자

```
import tensorflow as tf


# 이항 연산자
tf.add(x,y)         # x + y
tf.subtract(x,y)    # x - y
tf.multiply(x,y)    # x * y
tf.truediv(x,y)     # x / y
tf.math.mod(x,y)    # x % y
tf.math.pow(x,y)    # x ** y
```

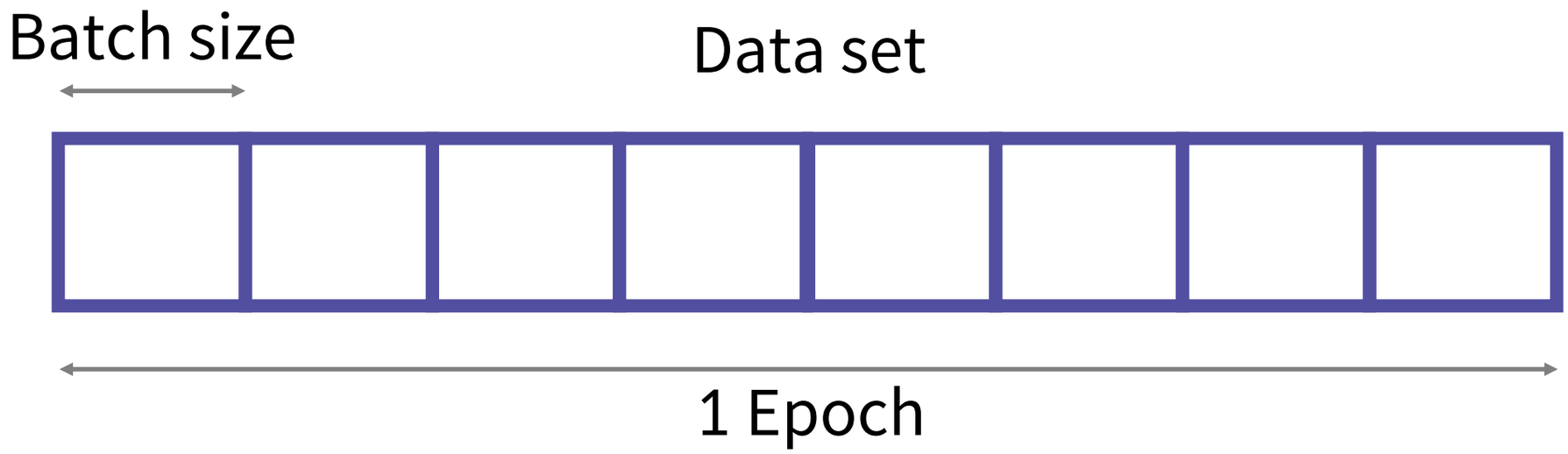
딥 러닝 구현하기

데이터 준비 하기

Epoch: 한 번의 epoch는 전체 데이터 셋에 대해
한 번 학습을 완료한 상태

Batch: batch(보통 mini-batch라고 표현)는 나눠진
데이터 셋을 뜻하며 iteration는 epoch를
나누어서 실행하는 횟수.

데이터 준비 하기



Ex) 총 데이터가 1000개, batch size = 100

1 iteration = 100개 데이터에 대해서 학습

1 epoch = 100/batch size = 10 iteration

데이터 준비 하기

```
import tensorflow as tf
import numpy as np


data = np.random.sample((100,2))
labels = np.random.sample((100,1))
# numpy array로부터 데이터셋 생성
dataset = tf.data.Dataset.from_tensor_slices((data, labels))
dataset = dataset.batch(32)
```

Dataset API를 사용하여 딥러닝 모델 용 dataset을 생성

딥러닝 모델 구축을 위한 Keras

Keras는 딥러닝 모델을 만들기 위한 **고수준의 API** 요소를
제공하는 모델 수준의 **라이브러리**

Keras API

- 동일한 코드로 CPU와 GPU에서 실행
- 사용하기 쉬운 API를 가지고 있어 딥러닝 모델의 프로토타입을 빠르게 구현
- 합성곱 신경망, 순환 신경망을 지원하며 이 둘을 자유롭게 조합하여 가능
- MIT 라이선스를 따르므로 상업적인 프로젝트에도 자유롭게 사용할 수 있습니다.

딥러닝 모델 생성 함수

인공신경망 모델을 만들기 위한 함수

tf.keras.models.Sequential()

신경망 모델의 layer 구성에 필요한 함수

tf.keras.layers.Dense(units, activation)

- units : 레이어 안의 Node의 수
- activation : 적용할 activation 함수

딥러닝 모델 구축하기

```
import tensorflow as tf


model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_dim=2, activation=‘sigmoid’),
    tf.keras.layers.Dense(10, activation=‘sigmoid'),
    tf.keras.layers.Dense(1, activation='sigmoid'),
])
```

tf.keras.layers를 추가하여 hidden layer를 쌓음

딥러닝 모델 학습 및 평가

```
model.compile(loss='mean_squared_error’, optimizer=‘SGD')
model.fit(dataset, epochs=100)


dataset_test = tf.data.Dataset.from_tensor_slices((data_test,
                                                   labels_test))
dataset_test = dataset.batch(32)


model.evaluate(dataset_test)
predicted_labels_test = model.predict(data_test)
```

회귀 모델과 분류 모델

회귀 모델

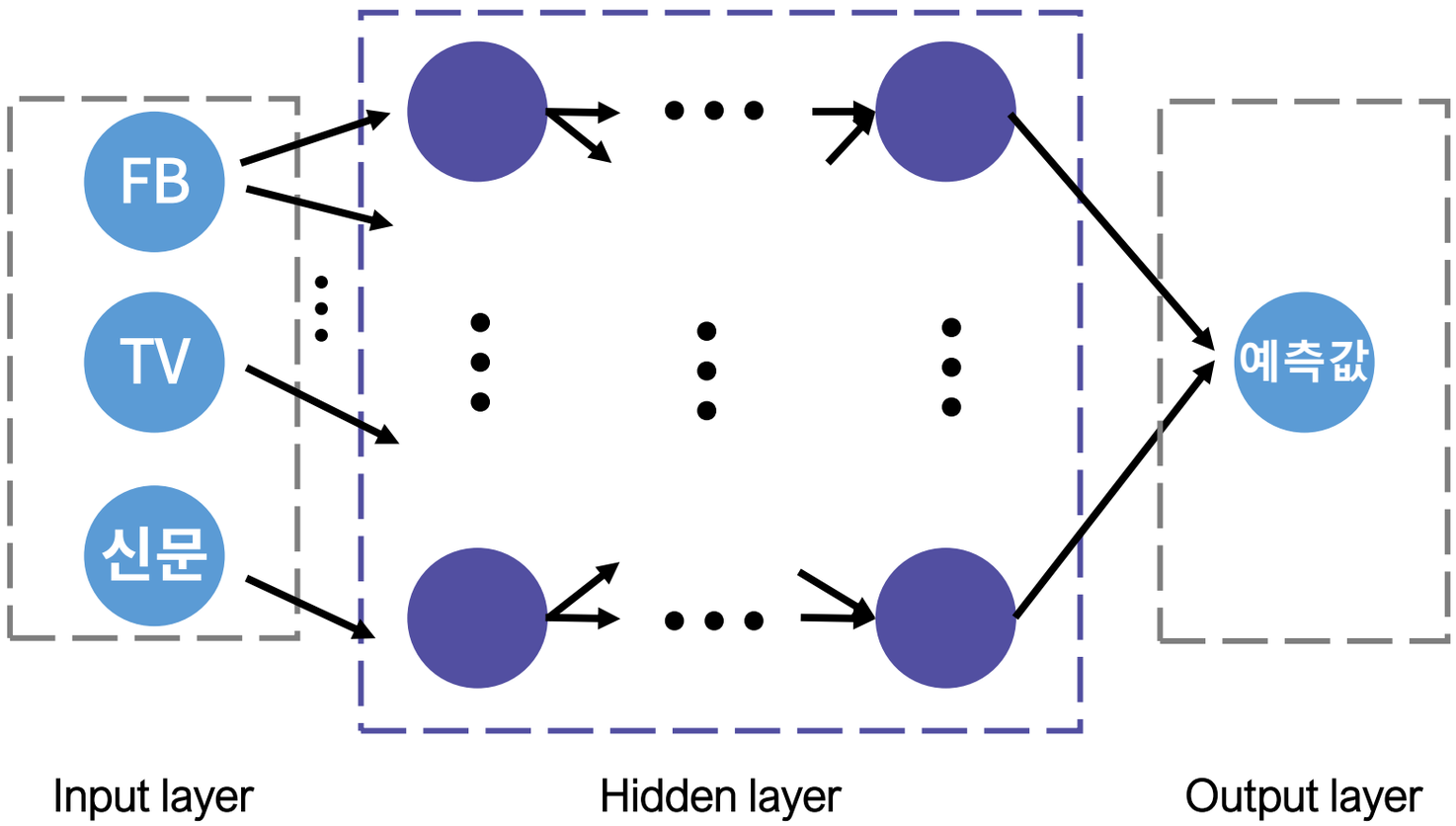
광고료 대비 판매량 데이터

FB TV 신문 판매량

X_1	X_2	X_3	Y
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
⋮	⋮	⋮	⋮

회귀 모델

FB	TV	신문
X_1	X_2	X_3
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4
8.7	48.9	75
57.5	32.8	23.5
⋮	⋮	⋮



판매량
Y
22.1
10.4
9.3
18.5
12.9
7.2
11.8
⋮

Loss function: $\mathrm{L}(\boldsymbol{W}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\text{판매량} - \text{예측값}(FB_i, TV_i, \text{신문}_i; \boldsymbol{W})\right\|^2$

분류 모델

네이버 영화 댓글 평점 데이터

document	label
와.. 연기가 진짜 개쩔구나.. 지루할거라고 생각했는데 몰입해서 봤다.. 그래 이런게 진짜 영화지	1(긍정)
안개 자욱한 밤하늘에 떠 있는 초승달 같은 영화.	1(긍정)
아 더빙.. 진짜 짜증나네요 목소리	0(부정)
원작의 긴장감을 제대로 살려내지못했다.	0(부정)
사랑을 해본사람이라면 처음부터 끝까지 웃을수 있는영화	1(긍정)
⋮	⋮

Preprocessing

Sklearn의 CountVectorizer

토큰 빈도수 카운트 벡터로 변환

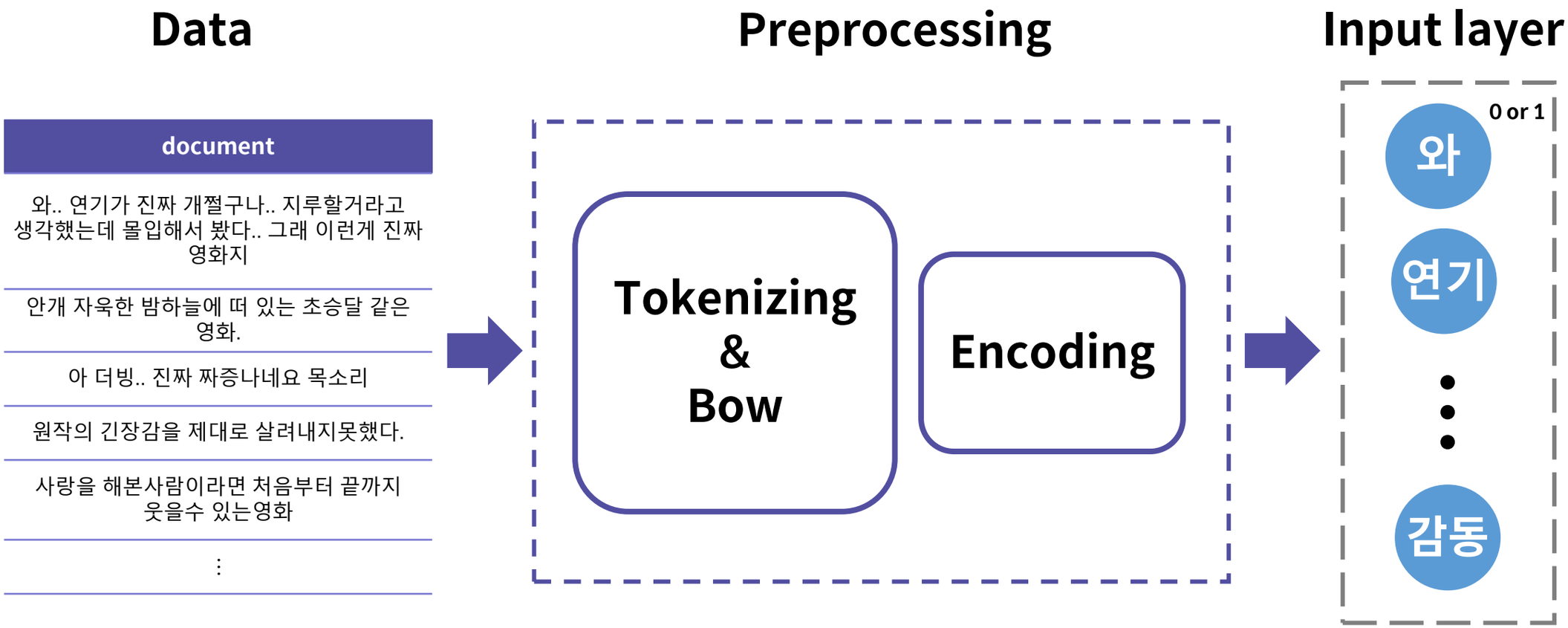
```
corpus = [ 'This is the first document.',
'This document is the second document.',
'And this is the third one.',
'Is this the first document?',... ]
```

카운트 벡터로 변환한 결과

```
[[0 1 1 1 0 0 1 0 1]
[0 2 0 1 0 1 1 0 1]
[1 0 0 1 1 0 1 1 1]
[0 1 1 1 0 0 1 0 1]]
```

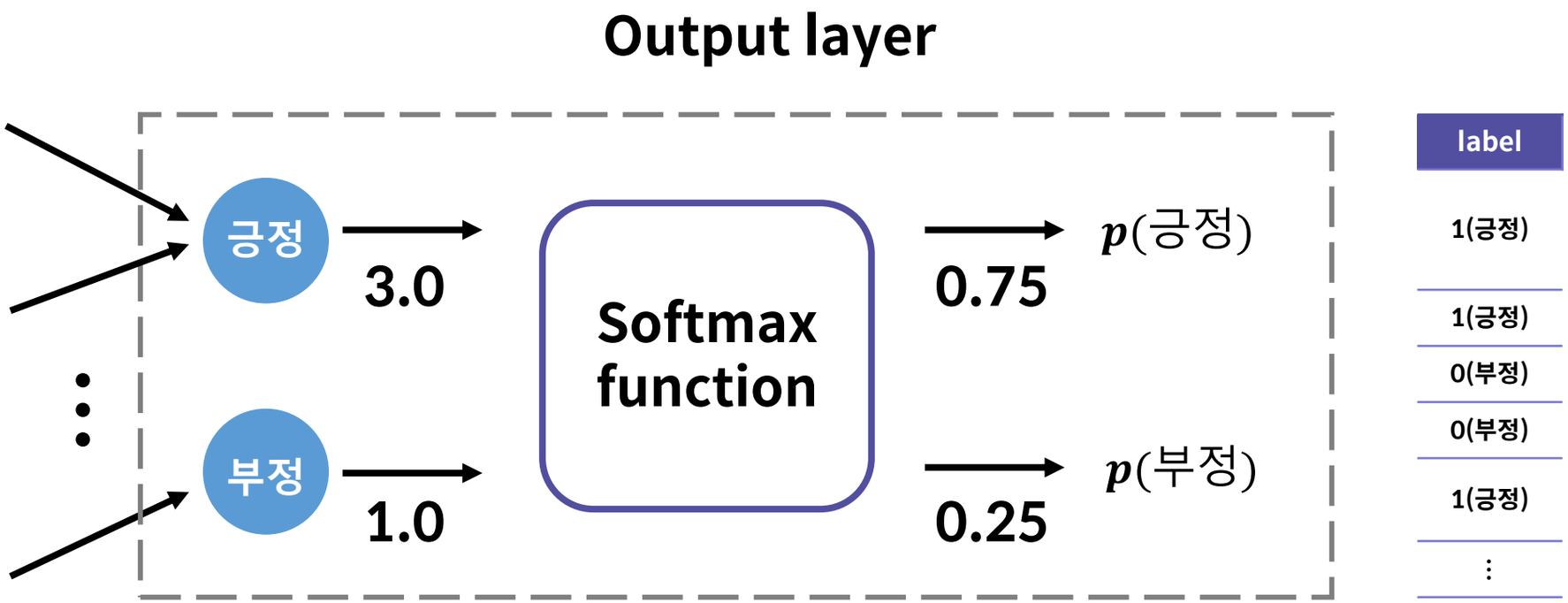
분류 모델

네이버 영화 댓글 긍정 부정 분류기



분류 모델

네이버 영화 댓글 평점 데이터



Cross-Entropy Loss function: $\mathrm{L}(\boldsymbol{W}) = -\frac{1}{N}\sum_{i=1}^{N} label * \log(p(\text{예측}; \boldsymbol{W}))$

최적화 기법

Loss function $L(W)$ **최소화**

선형 회귀에서의 **'거꾸로 된 산을 내려가기'** 활용

Gradient Descent(GD) 방식으로 최적화

인공 신경망에서는 다양한 형태의 GD기법들을 사용

SGD, Momentum, AdaGrad, RMSProp, Adam 등등