07_01_TensorFlow

1. TensorFlow ?

- Google이 만든 Machine Learning Library
- Open Source Library
- 수학적 계산을 하기 위한 Library
- Data Flow Graph를 이용
- TensorFlow는 Node와 Edge로 구성된 방향성 있는 Graph
  - Node : 데이터의 입출력과 수학적 계산
  - Edge : Tensor를 Node로 실어 나르는 역할
  - Tensor : 동적 크기의 다차원 배열을 지칭

1.1 TensorFlow 설치

1.1.1 CPU 버전

- Python에서 설치

```
$ pip install tensorflow==1.5
```

- Anaconda에서 설치

```
$ conda install tesnsorflow==1.5
```

1.1.2 GPU 버전

1. https://developer.nvidia.com/cuda-toolkit-archive에서 CUDA 10.0 다운로드 및 설치
2. https://developer.nvidia.com/rdp/cudnn-download에서 for CUDA 10.0 다운로드 및 압축 해제
   - 압축 해제된 CUDA에서 lib, include, bin 폴더 등의 파일을 전체 복사
   - 1.에서 설치된 경로에 붙여넣기
3. 환경 변수 확인
   - 잘 되어있지 않다면 아까 추가한 lib, include, bin 폴더의 경로 추가
4. tensorflow-gpu 설치
   - Python에서

```
$ pip install tensorflow-gpu

# 업그레이드 시
$ pip install --upgrade tensorflow-gpu
```

   - Anaconda에서

```
$ conda install tensorflow-gpu
```

2. TensorFlow 기초

2.1 출력

- Node는 숫자 연산과 데이터 입출력을 담당

```
my_node = tf.constant("Hello World")
sess = tf.Session()

# Session을 이용해서 Node를 실행시켜야지 Node가 가지고 있는 데이터를 출력 함
print(sess.run(my_node).decode())   # 입력한 데이터 출력 .decode()
```

2.2 constant

- 선언과 동시에 초기화

```
node1 = tf.constant(10, dtype = tf.float32)
node2 = tf.constant(20, dtype = tf.float32)

node3 = node1 + node2

## 그래프를 실행시키기 위해 runner역할을 하는 session 객체 필요
sess = tf.Session()

print(sess.run(node3))
print(sess.run([node1, node2, node3]))
```

```
30.0
[10.0, 20.0, 30.0]
```

2.3 placeholder

- 선언과 동시에 초기화하는 것이 아니라 일단 선언 후 나중에 값을 입력

```
node1 = tf.placeholder(dtype = tf.float32)
node2 = tf.placeholder(dtype = tf.float32)

node3 = node1 + node2

sess = tf.Session()
result = sess.run(node3, feed_dict = {node1 : 10, node2 : 20})

print(result)
```

```
30.0
```

2.4 cast

```
node1 = tf.constant([10, 20, 30], dtype = tf.int32)
print(node1)    # Tensor("Const_4:0", shape=(3,), dtype=int32)

node2 = tf.cast(node1, dtype = tf.float32)
print(node2)    # Tensor("Cast_4:0", shape=(3,), dtype=float32)

sess = tf.Session()
print(sess.run(node1))  # [10 20 30]
print(sess.run(node2))  # [10. 20. 30.]
```

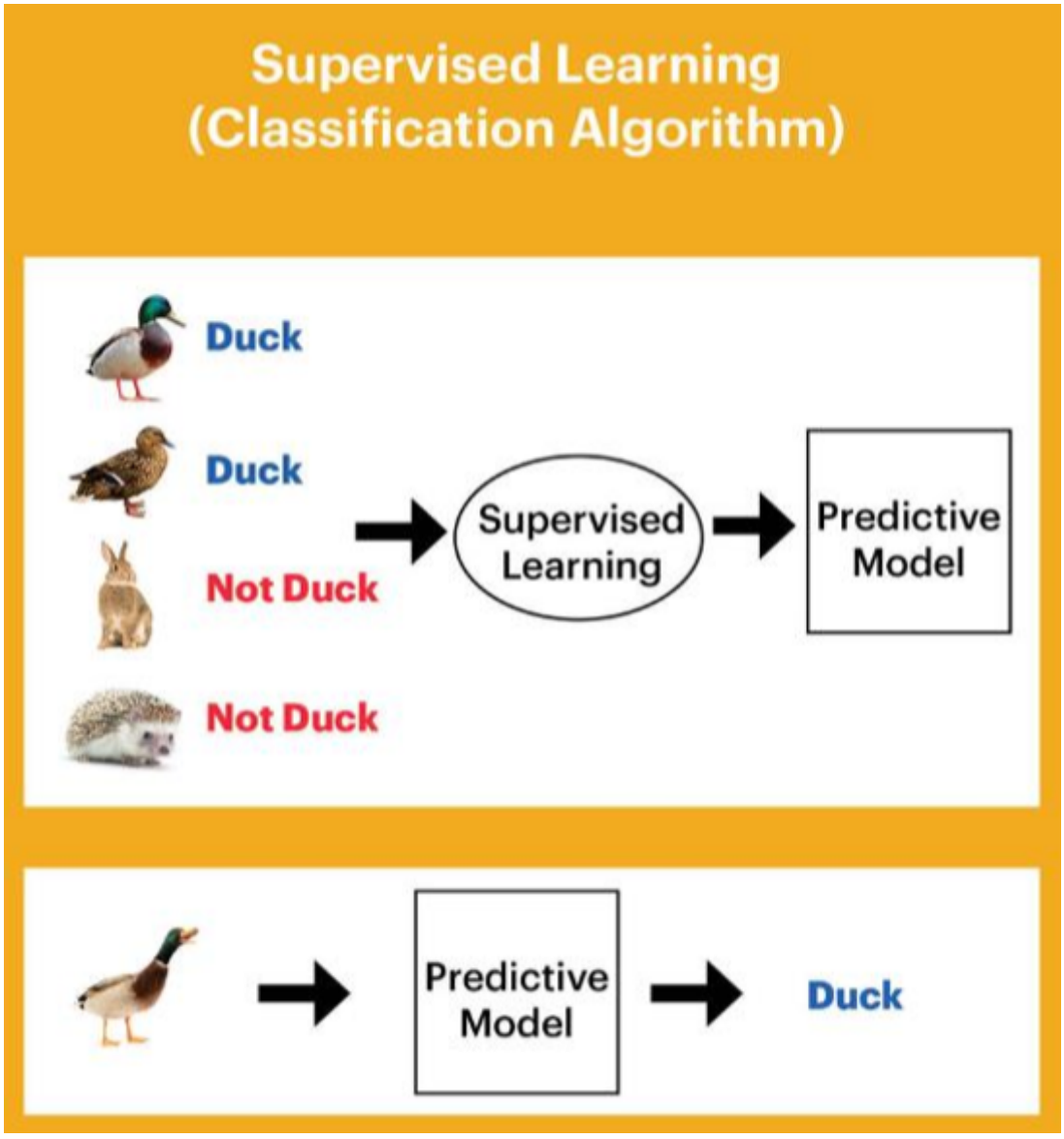
3. Machine Learning

- 프로그램 자체가 **데이터를 기반으로 학습**을 통해 배우는 능력을 가지는 프로그래밍

3.1 Learning의 종류

3.1.1 Supervised Learning(지도 학습)

- Training Set이라고 불리는 Label화 된 데이터를 통해 학습

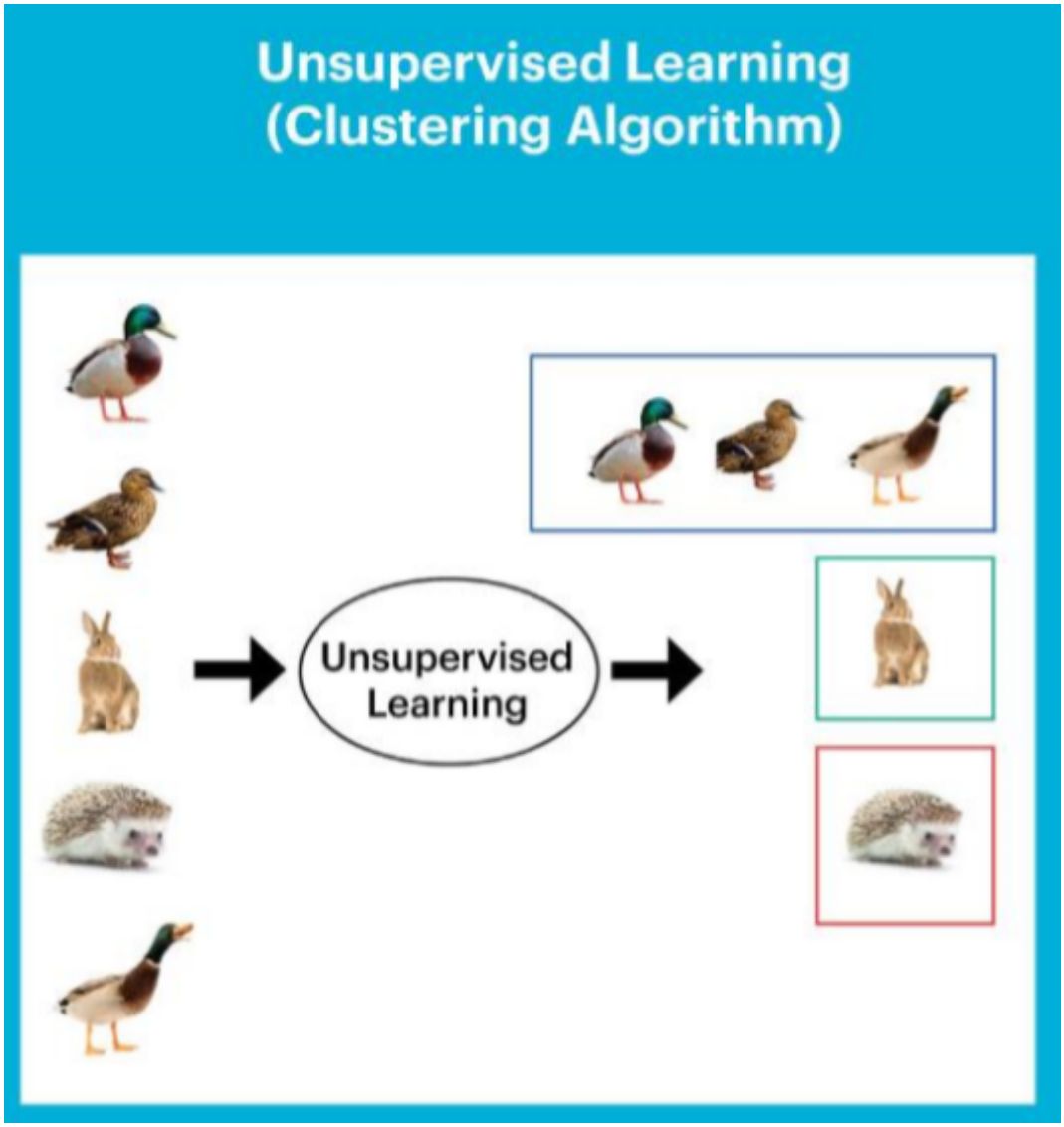


- Linear Regression(선형 회귀) - 공부 시간 : 시험 점수
- Logistic Regression(로지스틱 회귀)
    - Binary Classification(이항 분류) - 공부 시간 : 합격/불합격

- Multinomial Classification(다향 분류) - 공부 시간 : 학점

3.1.2 Unsupervised Learning(비지도 학습)

- Label화 되지 않은 데이터를 통해 학습
- 데이터를 이용해 스스로 학습



4. Linear Regression

- Linear Regression의 가장 큰 목표는 가설의 완성

$$\text{가설}(Hypothesis) = Wx + b$$

4.1 Training Data Set 준비

```
import tensorflow as tf

x = [1, 2, 3]
y = [1, 2, 3]
```

4.2 Weight(W) & Bias(b) 준비

```
W = tf.Variable(tf.random_normal([1]), name="weight")
b = tf.Variable(tf.random_normal([1]), name="bias")
```

- Hypothesis(가설)
  - 최종 목적은 Training Data에 가장 근접한 Hypothesis를 만드는 것(W와 b를 결정)
  - 잘 만들어진 가설은 W가 1에 b가 0에 가까워야 함

4.3 Cost(loss) Function

$$cost(W,b) = \frac{1}{n}\sum_{i=1}^{n}(H(x^i) - y^i)^2$$

- 목적은 cost 함수를 최소로 만드는 W와 b를 구하는 것

```
cost = tf.reduce_mean(tf.square(H - y))
```

- **Cost Function Minimize**

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)
```

4.4 Training

```
# runner 생성
sess = tf.Session()

# 실행전 global variable 초기화
sess.run(tf.global_variables_initializer())

# 학습 진행
for step in range(3000):
    _, w_val, b_val, cost_val = sess.run([train, W, b, cost])
    if step % 300 == 0:
        print("{}, {}, {}".format(w_val, b_val, cost_val))
```

4.5 결과

```
print(sess.run(H))
```

4.6 소스

```
x = [1, 2, 3]
y = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name = "weight")
b = tf.Variable(tf.random_normal([1]), name = "bias")

H = W * x + b

cost = tf.reduce_mean(tf.square(H - y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.01)
train = optimizer.minimize(cost)

sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer())

for step in range(3000):
    _, w_val, b_val, cost_val = sess.run([train, W, b, cost])
    if step % 300 == 0:
        print("{}, {}, {}".format(w_val, b_val, cost_val))

print(sess.run(H))
```

5. Prediction

- Placeholder를 이용

```
import tensorflow as tf

x = tf.placeholder(dtype = tf.float32)
y = tf.placeholder(dtype = tf.float32)

x_data = [1, 2, 3, 4]
y_data = [4, 7, 10, 13]

W = tf.Variable(tf.random_normal([1]), name = "weight")
b = tf.Variable(tf.random_normal([1]), name = "bias")

H = W * x + b

cost = tf.reduce_mean(tf.square(H - y))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.01)
train = optimizer.minimize(cost)

sess = tf.Session()

sess.run(tf.global_variables_initializer())

for step in range(4200):
    _, cost_val = sess.run([train, cost], feed_dict = {x : x_data, y : y_data})
    if step % 300 == 0:
        print(cost_val)

# 예측
print(sess.run(H, feed_dict = {x : [300]}))
```