7. 다중 클래스 분류, 8. 프로그래밍 프레임워크 소개

날짜	@2023년 12월 18일

▼ 목차

Softmax Regression
- 1️⃣ Softmax Layer
- 2️⃣ Softmax Examples

Softmax 분류기 훈련시키기
- 1️⃣ Understanding Softmax
- 2️⃣ Loss Function
  - 🚀 Example
- 3️⃣ Gradient Descent with Softmax
  - 🚀 Cost Function

Deep Learning Frameworks

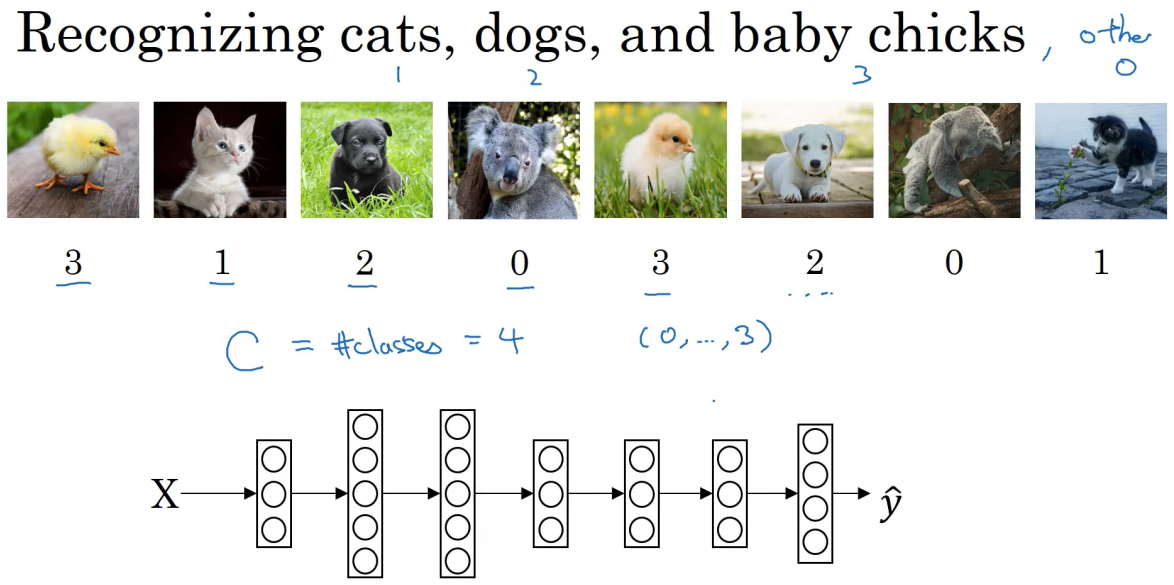
Tensorflow

Softmax Regression

💠 이진 분류에 로지스틱 회귀를 사용하듯, 다중 분류에는 소프트맥스 회귀를 사용함

1️⃣ Softmax Layer

- 다중 분류는 출력층에서 입력값 X가 주어졌을 때 각 클래스로 분류될 확률이 주어져야 함
  - 아래 경우 출력값인 $\hat{y}$는 (4,1) 벡터가 되며, 각 요소의 합이 1이 될 것임



Andrew Ng

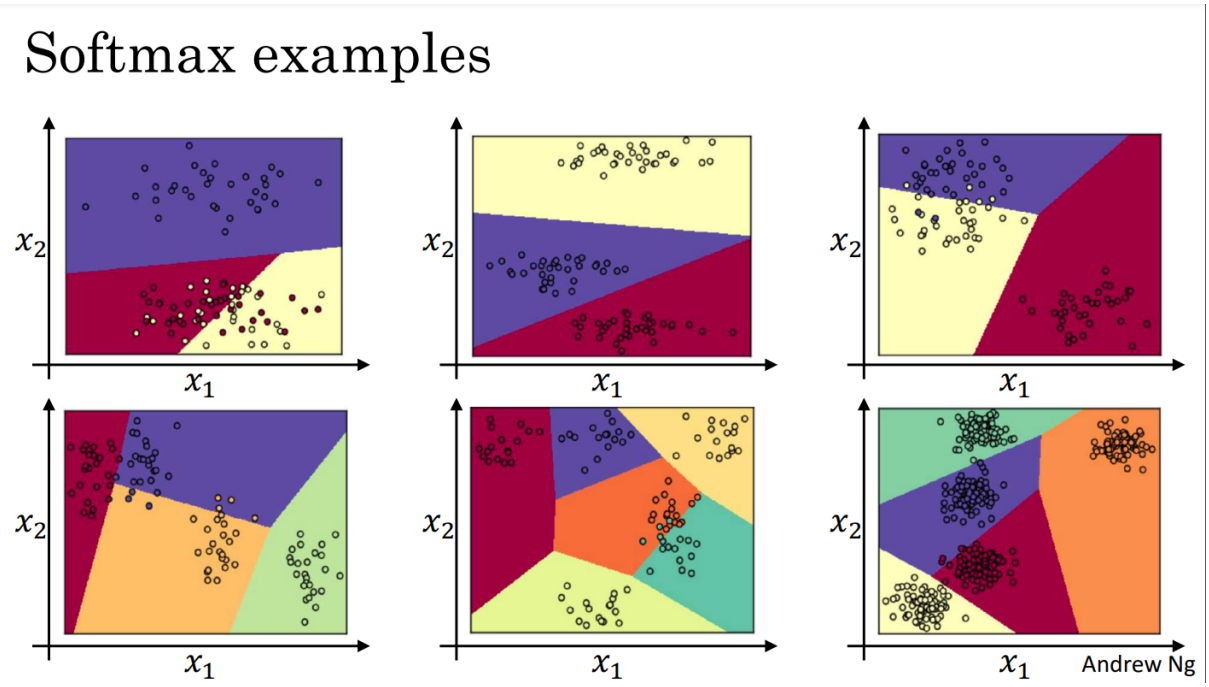
🔹 소프트맥스 활성화 함수를 적용하여 입력값 X가 각 클래스로 분류될 확률을 구할 수 있음

- 마지막 선형 출력값(z)들을 각각 지수화시켜 임시변수 $t = e^z$를 만든 뒤,
  모든 값의 합이 1이 되도록 각 임시변수를 모든 임시변수들의 합으로 나눠 정규화시킴

$$a_i = \frac{e_i^z}{\sum_{i=1}^{C} e_i^z}$$

2️⃣ Softmax Examples

🔹 은닉층이 없을 때, 소프트맥스는 다음과 같은 선형 경계를 그려 다중 분류를 수행함



Softmax 분류기 훈련시키기

1️⃣ Understanding Softmax

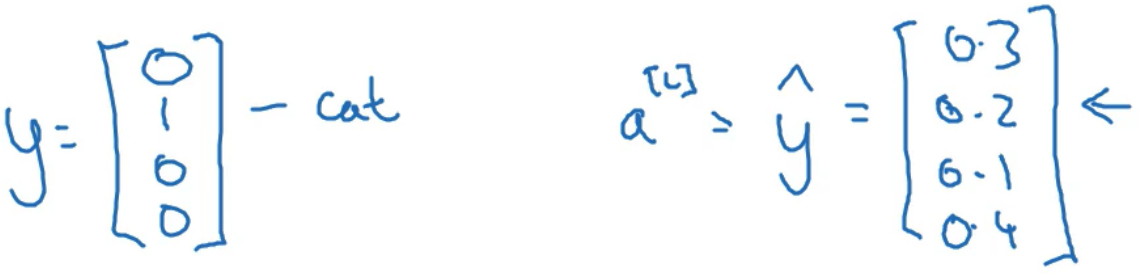
💠 소프트맥스는 로지스틱 회귀를 일반화한 것으로, $C = 2$인 경우 로지스틱 회귀와 같아짐

- 소프트맥스는 z의 원소가 가장 큰 곳에 1, 나머지에는 0을 갖는 벡터로 대응시키는 하드맥스(Hardmax)와 대비되는 개념

2️⃣ Loss Function

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{C} y_j \log \hat{y}_j$$

🚀 Example



- 위와 같은 예제가 있을 때, 해당 손실함수를 사용하면 y_j가 0인 경우가 제외되어 유일하게 $-y_2 \log \hat{y}_2 = -\log 0.2$만이 남게 됨
- 학습 알고리즘의 목표는 손실함수의 값을 작게 하는 것이므로, 출력값 $\hat{y}_2$(클래스가 2일 확률)를 가능한 한 크게 만드는 것이 중요하며 이는 최대 우도 추정과 유사함

3️⃣ Gradient Descent with Softmax

🚀 Cost Function

$$\mathcal{J}(w^{[1]}, b^{[1]}, \cdots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

- 소프트맥스와 손실함수를 결합한 역전파의 값은 다음과 같음

$$dz^{[L]} = \frac{\partial J}{\partial z^{[L]}} = \hat{y} - y$$

Deep Learning Frameworks

Deep learning frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Theano
- Torch

Choosing deep learning frameworks
- Ease of programming (development and deployment)
- Running speed
- Truly open (open source with good governance)

Andrew Ng

Tensorflow

🥵 강의에서는 Tensorflow 1를 사용하고 있어,
현재 최신 버전인 Tensorflow 2와는 상충하는 부분이 많음에 주의

```
import numpy as np
import tensorflow as tf

############## 1
w = tf.Variable(0, dtype=tf.float32)
cost = tf.add(tf.add(w ** 2, tf.multiply(-10, w)), 25) #cost
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost

init = tf.global_variables_initializer()
session = tf.Session()
sesion.run(init)
print(session.run(w))

session.run(train)
print(session.run(w))
```

```
for i in range(1000) :
    session.run(train)
print(session.run(w))

############## 2
coefficients = np.array([[1.], [-10.], [25.]])

w = tf.Variable(0, dtype=tf.float32)
x = tf.placeholder(tf.float32, [3, 1])

#cost = tf.add(tf.add(w ** 2, tf.multiply(-10, w)), 25)
cost = x[0][0] * w**2 + x[1][0] * w + x[2][0]
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost

init = tf.global_variables_initializer()
session = tf.Session()
sesion.run(init)
print(session.run(w))

session.run(train, feed_dict={x:coefficients})
print(session.run(w))
```

- Tensorflow 2를 이용한 딥러닝 실습 코드는 다음과 같음

```
import numpy as np
import tensorflow as tf

coefficients = np.array([[1.], [-10.], [25.]])

w = tf.Variable(0, dtype=tf.float32)
x = tf.Variable(coefficients, dtype=tf.float32) ###

def cost_function() :
    return x[0][0] * w**2 + x[1][0] * w + x[2][0]
```

```
optimizer = tf.optimizers.SGD(0.01)
optimizer.minimize(cost_function, var_list=[w])

print(w.numpy())
```