

인공지능을 위한 머신러닝 알고리즘

14. Theano를 통한 머신러닝 구현

CONTENTS

1

Theano란?

2

Theano로 GPU 프로그래밍 실습하기

3

Theano로 신경망 구현하기

학습 목표

- Theano의 기본적인 문법과 Symbolic Expression을 이해할 수 있다.
- Theano의 GPU 연산 과정을 이해할 수 있다.
- Theano를 사용하여 신경망을 구현해보고 실행할 수 있다.



1. Theano란?

■ Theano의 특징

- ◉ LISA Lab(<https://mila.umontreal.ca/en/>)에서 만든 Python 기반 오픈소스 Package

<http://deeplearning.net/software/theano/>

❖ 장점

- **Symbolic** 연산 철학으로 간결하고 빠르게 모델 구현 가능
- **Symbolic** 미분이 가능하므로 역전파 등을 직접 구현할 필요가 없음
- 동일한 코드를 **CPU**와 **GPU**에서 모두 사용 가능
- **Python** 기반이므로, **numpy**, **scipy** 등 다양한 **Python** 패키지와의 연동할 수 있음

❖ 단점

- 복잡하고 알기 어려운 에러 메시지

■ Symbolic expression 정의 - (1) scalar

```
from theano import tensor as T
```

```
x = T.scalar()
```

```
y = T.scalar()
```

```
z = x + y
```

```
w = z * x
```

```
a = T.sqrt(w)
```

```
b = T.exp(a)
```

```
c = a ** b
```

```
d = T.log(c)
```

← Symbolic 변수 정의

← Symbolic Expression

← Symbolic Expression ($a = w^2$)

← Symbolic Expression ($b = e^a$)

← Symbolic Expression ($c = a^b$)

← Symbolic Expression ($d = \ln(c)$)

■ Symbolic expression 정의 - (2) vector & matrix

```
from theano import tensor as T
```

```
x = T.matrix()
```

```
y = T.matrix()
```

```
a = T.vector()
```

```
b = T.vector()
```

```
c = T.dot(x, y)
```

```
d = T.dot(x, a)
```

```
e = T.dot(a, b)
```

```
f = a * b
```

```
h = e + f
```

Symbolic 변수 정의

Symbolic Expression (matrix-matrix product)

Symbolic Expression (matrix-vector product)

Symbolic Expression (vector-vector product)

Symbolic Expression (element-wise product)

Symbolic Expression (broadcasting)

■ Symbolic expression 정의 - (3) theano function

```
>>> from theano import tensor as T
>>> x = T.scalar()
>>> y = T.scalar()
>>> from theano import function
>>> f = function([x, y], x + y)
>>> f(1., 2.)
array(3.0)
```

컴파일

첫 번째 인수는 입력 symbolic 변수들의 리스트

두 번째 인수는 출력 symbolic 변수

scalar를 사용하므로 scalar 값 입력

출력된 scalar 값

■ Symbolic 미분 연산

$Y = -\ln(3x^2 + 5x)$ 의 미분


```
>>> from theano import tensor as T
>>> x = T.scalar()
>>> y = -1*T.log(3*x**2+5*x)
>>> y_prime = T.grad(y,x)
>>> from theano import function
>>> f = function([x], y_prime)
>>> f(1)
array(-1.375, dtype=float32)
```

Symbolic
미분

자동으로 $Y = -\ln(3x^2 + 5x)$ 의
도함수 계산



복잡한 역전파 계산을
직접 구현할 필요가 없음

A person's hands are holding a smartphone, which is lit up. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark banner is at the bottom, containing a yellow icon and text.

2. Theano로 GPU 프로그래밍 실습하 기

■ shared variables

shared variable은 데이터를 **RAM**에서 **GPU**의 **VRAM**로 옮기는 명령어

```
>>> from theano import shared  
>>> x = shared(0.)  
>>> x.get_value()  
0.0
```



givens

- ◉ **symbolic** 변수에 **shared** 데이터를 대입
- ◉ **Theano** 함수 $f = x + y$ 가 존재할 때, **symbolic** 변수 x, y 의 값 **1, 2**를 설정하는 방법

```
>>> f = function([x, y], x + y)
```

```
>>> f(1., 2.)
```

```
array(3.0)      RAM → VRAM → GPU 연산
```

```
>>> x_val = theano.shared(1)
```

VRAM → GPU 연산

```
>>> y_val = theano.shared(2)
```

```
>>> f = function([x, y], x + y, givens = [(x, x_val), (y, y_val)])
```

```
>>> f()
```



updates

- GPU 연산 결과를 이용해 **shared** 데이터를 수정

```
>>> x_val = theano.shared(1)
>>> f = function([], x_val, updates = (x_val, x_val+1))
>>> f()
```

실행 시 RAM을 거치지 않고 GPU 내에서 **x_val**을 1씩 증가시킴



■ shared variable 예제

```
>>> from theano import shared
>>> x = shared(0.)
>>> from theano.compat.python2x import OrderedDict
>>> updates = OrderedDict()
>>> updates[x] = x + 1
>>> f = function([], updates=updates)
>>> f()
[]
>>> x.get_value()
1.0
>>> x.set_value(100.)
>>> f()
[]
>>> x.get_value()
101.0
```

A person's hands are shown holding a smartphone, with the screen glowing. The background is dark with out-of-focus, colorful bokeh lights in shades of yellow, orange, and blue. A semi-transparent dark blue banner is at the bottom, featuring a yellow decorative element on the left and the title text in yellow.

3. Theano로 신경망 구현하기

신경망 코드

```
import numpy
import theano
import theano.tensor as T
rng = numpy.random
```

$N = 400$ ← 데이터 샘플 개수
 $feats = 784$ ← 특징 차원 크기

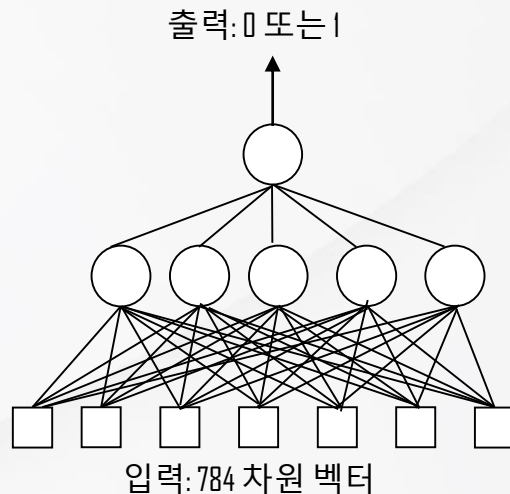
```
D = (rng.randn(N, feats).astype(numpy.float32), rng.randint(size = N, low=0, high=2).astype(numpy.float32))
```

$D = \{X, Y\}$

$N = 400$

0.12	0.49	1.65	-1.45	2.12	0.21	-2.12	0.23

각 특징 값들은 평균 0, 분산 1인 가우시안 분포에서 무작위로 샘플링



$N = 400$

1
1
0
0
1
1

레이블

■ 신경망 코드

```
training_steps = 10,000
```

```
x = T.matrix('x')
```

```
y = T.vector('y')
```

```
w_1 = theano.shared(rng.randn(784, 300), name = 'w1')
```

```
b_1 = theano.shared(numpy.zeros(300,), name = 'b1')
```

```
w_2 = theano.shared(rng.randn(300,), name = 'w2')
```

```
b_2 = theano.shared(0., name = 'b2')
```

```
print w_1.get_value(), b_1.get_value()
```

```
print w_2.get_value(), b_2.get_value()
```

Parameters ϑ

784

300		
0.22	-1.21	2.01

w_1

300

0	0	0
---	---	---

b_1

300

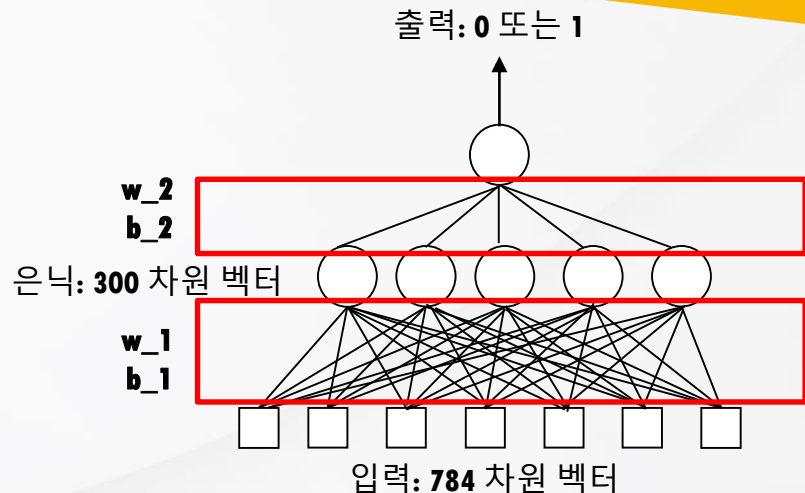
0	0	0
---	---	---

w_2

1

0

b_2



신경망 코드

```
p_1 = T.nnet.sigmoid(T.dot(T.nnet.sigmoid(T.dot(x, w_1)+b_1), w_2)+b_2) ← 타겟이 일 확률
```

```
prediction = p_1 > 0.5
```

```
xent = -y*T.log(p_1) - (1-y)*T.log(1-p_1) ← Cross entropy
```

```
cost = xent.mean() + 0.01 * ((w_1**2).sum() ← L2 regularization를 적용한 손실 함수  
+ (w_2**2).sum() + (b_1**2).sum() + (b_2**2).sum())
```

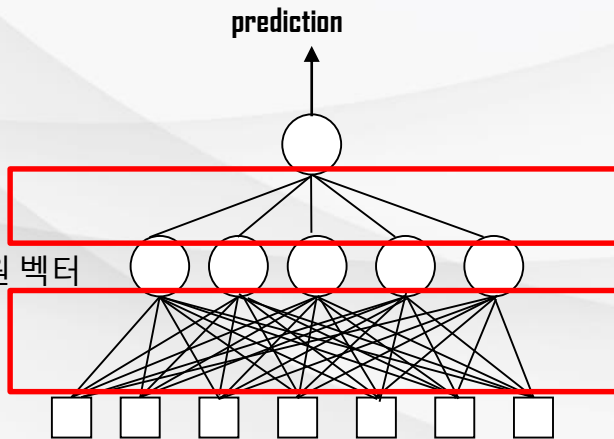
```
gw_1, gb_1, gw_2, gb_2 = T.grad(cost, [w_1, b_1, w_2, b_2]) ← 손실 함수에 대한  
파라미터의 변화량
```

```
p_1 = sigmoid(h*w_2 + b_2)
```

은닉: 300 차원 벡터

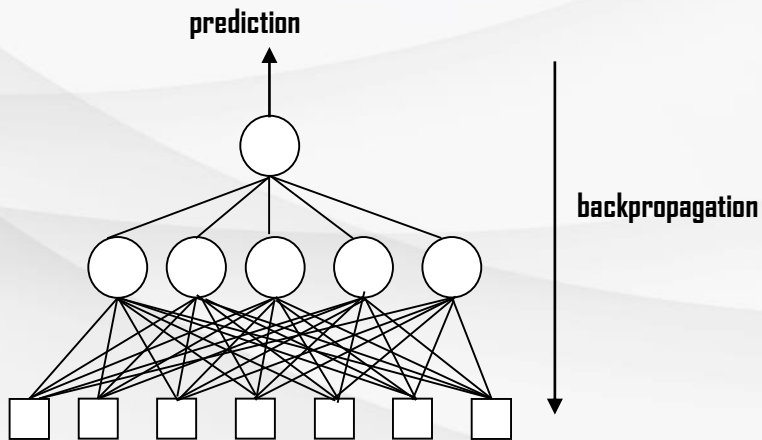
```
h = sigmoid(x*w_1 + b_1)
```

입력: 784 차원 벡터



■ 신경망 코드

```
train = theano.function(inputs = [x, y],      ← 컴파일
                        outputs = [prediction, xent],
                        updates = {w_1 : w_1-0.1*gw_1, b_1 : b_1-0.1*gb_1, ← 학습률:0.1
                                   w_2 : w_2-0.1*gw_2, b_2 : b_2-0.1*gb_2})
predict = theano.function(inputs = [x], outputs = prediction) ← 예측 함수
```



■ 신경망 코드

```
for i in range(training_steps):
```

```
    pred, err = train(D[0], D[1]) ← 훈련
```

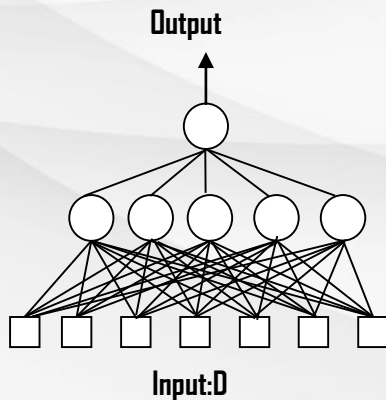
```
print "Final model:"
```

```
print w_1.get_value(), b_1.get_value()
```

```
print w_2.get_value(), b_2.get_value()
```

```
print "target values for D: ", D[1]
```

```
print "predictions on D: ", predict(D[0]) ← 예측
```





학습정리

지금까지 [Theano를 통한 머신러닝 구현]에 대해서 살펴보았습니다.

Theano란?

symbolic 연산 철학으로 간결하고 빠르게 모델 구현 가능
symbolic 미분이 가능하므로 역전파 등을 직접 구현할 필요가 없음 (grad 함수 사용)

```
x = T.scalar()    y = -1*T.log(3*x**2+5*x)    y_prime = T.grad(y,x)
```

Theano로 GPU 프로그래밍 실습하기

shared variables: RAM에서 GPU VRAM으로 데이터를 옮겨줌
givens: symbolic 변수에 shared variables를 대입
updates: GPU 연산 결과를 이용해 shared variables의 값을 수정

Theano로 신경망 구현하기

Theano를 사용하여 머신러닝 알고리즘을 구현할 경우, GPU를 사용하여 빠른 학습 가능
간결한 코드 작성 가능, 미분의 자동 계산으로 프로그래머의 일을 줄여줌