deep_hw1

March 4, 2019

1 Exercise 1

1.1 gradient descent

gradient descent numpy .
W, X, Y step . step W W cost .

hint: W . (bias)

```
In [125]: import numpy as np

          def grad(W, X, Y, learning_rate = 0.1, step = 100):
              for step in range(step):
                  # insert your code


                  if step % 10 == 0:
                      print(step, W, cost)
              return W
```

W .
W 2 .

```
In [2]: import numpy as np

        X = np.array([[1], [2], [3]])
        Y = np.array([[2], [4], [6]])
        W = np.random.random((1,1))

        W = grad(W, X, Y)
        print("W:",W)
```

```
0 [[0.93634556]] 18.56138772846749
10 [[1.99801944]] 6.435540111509881e-05
20 [[1.99999631]] 2.231308193680736e-10
30 [[1.99999999]] 7.736314477536554e-16
40 [[2.]] 2.682313429311266e-21
50 [[2.]] 9.315477649130816e-27
```

```
60 [[2.]] 0.0
70 [[2.]] 0.0
80 [[2.]] 0.0
90 [[2.]] 0.0
100 [[2.]] 0.0
W: [[2.]]
```

2 Exercise 2

2.0.1

AND . W gradient descent

- AND .

x1	x2	y
0	0	0
1	0	0
0	1	0
1	1	1

```
In [42]: # insert your code
         #  size
         #     W  W_and  .
```

W AND . learning_rate step .

```
In [52]: def AND(x1, x2):
             x = np.array([x1, x2])
             y = np.matmul(x, W_and)
             if y >= 0.5:
                 return 1
             else:
                 return 0
```

```
In [58]: print(AND(0,0))
         print(AND(1,0))
         print(AND(0,1))
         print(AND(1,1))
```

```
0
0
0
1
```

OR NAND .

- OR .

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	1

- NAND .

x1	x2	y
0	0	1
1	0	1
0	1	1
1	1	0

W_or

```
In [ ]: # insert your code
        #     W  W_or  .
```

```
In [60]: def OR(x1, x2):
             x = np.array([x1, x2])
             y = np.matmul(x, W_or)
             if y >= 0.5:
                 return 1
             else:
                 return 0
```

```
In [61]: print(OR(0,0))
         print(OR(1,0))
         print(OR(0,1))
         print(OR(1,1))
```

```
0
1
1
1
```

NAND AND .

```
In [ ]: # make nand function
        # def NAND(x1, x2):
```

```
In [66]: print(NAND(0,0))
         print(NAND(1,0))
         print(NAND(0,1))
         print(NAND(1,1))
```

```
1
1
1
0
```

XOR .

- XOR .

x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

```
In [ ]: # insert your code
        #     W  W_xor  .
```

```
In [77]: def XOR(x1, x2):
             x = np.array([x1, x2])
             y = np.matmul(x, W_xor)
             if y >= 0.5:
                 return 1
             else:
                 return 0
```

```
In [78]: print(XOR(0,0))
         print(XOR(1,0))
         print(XOR(0,1))
         print(XOR(1,1))
```

```
0
0
1
1
```

. XOR .

.

AND, OR, NAND XOR .

hint: . . .
XOR

```
In [ ]: # make XOR gate
        # def XOR(x1,x2):
```

```
In [81]: print(XOR(0,0))
         print(XOR(1,0))
         print(XOR(0,1))
         print(XOR(1,1))
```

```
0
1
1
0
```

3 Example1

3.0.1 tensorflow

XOR tensorflow .
tensorflow .

3.1 Step 1: Define a computation graph

```
In [4]: import tensorflow as tf

        a = tf.placeholder(tf.int32, name="input_a")
        b = tf.placeholder(tf.int32, name="input_b")

        c = tf.add(a, b, name="add")
        d = tf.multiply(a, b, name="multiply")
        e = tf.subtract(c, d, name="subtract")
        out = tf.add(b, e, name="output")
```

tensorflow operation(Node) tensor(edge) .
Session run .

3.2 Step 2: Run the graph

```
In [5]: sess = tf.Session()
        input_data = { a: 7, b: 3 }
        result = sess.run(out, feed_dict=input_data)
        print("out:",result)
```

```
out: -8
```

4 Exercise 3

4.0.1 tensorflow

.

tensorflow Operations

In	Out
1, 2, 3	15
-1, -2, 3	-3
123, 456, 789	44613795

```
In [ ]: # input your code
```