Homework #3

2017012251 윤영훈

선행과정

```
In [1]: import random
        import numpy as np
        import matplotlib.pyplot as plt
```

XOR data

```
In [2]: x_seeds = np.array([(0,0), (1,0), (0,1), (1,1)], dtype=float)
        y_seeds = np.array([0,1,1,0])

In [3]: N = 1000
        idxs = np.random.randint(0,4,N)

In [4]: X = x_seeds[idxs]
        Y = y_seeds[idxs]

In [5]: X += np.random.normal(scale = 0.25, size = X.shape)
```

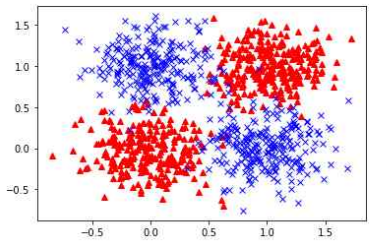
Plot data

```
In [6]: idxs_1 = np.where(Y==1)
        idxs_0 = np.where(Y==0)

In [7]: X_0 = X[idxs_0]
        Y_0 = Y[idxs_0]

In [8]: X_1 = X[idxs_1]
        Y_1 = Y[idxs_1]

In [9]: #plt.clf()
        plt.plot(X_0[:,0], X_0[:,1], "r^")
        plt.plot(X_1[:,0], X_1[:,1], "bx")
        plt.show()
```



Model

```
In [10]: class shallow_neural_network():
             def __init__(self, num_input_features, num_hiddens):
                 self.num_input_features = num_input_features
                 self.num_hiddens = num_hiddens

                 self.W1 = np.random.normal(size = (num_hiddens, num_input_features))
                 self.b1 = np.random.normal(size = num_hiddens)
                 self.W2 = np.random.normal(size = num_hiddens)
                 self.b2 = np.random.normal(size = 1)

             def sigmoid(self, z):
                 return 1/(1 + np.exp(-z))

             def predict(self, x):
                 z1 = np.matmul(self.W1, x) + self.b1
                 a1 = np.tanh(z1)
                 z2 = np.matmul(self.W2, a1) + self.b2
                 a2 = self.sigmoid(z2)
                 return a2, (z1, a1, z2, a2)

In [11]: model = shallow_neural_network(2,3)
```

random함수를 사용하기 위한 import random, numpy를 사용하기 위한 import numpy as np, plot graph 작성을 위한 import matplotlib.pyplot as plt 총 3가지 import 선언

Homework#3는 Train Code의 for-loop만 제거하는 과제이기 때문에 Plot data, Model Code는 변경하지 않음

XOR data의 x_seeds numpy array의 dtype이 기존 강의 PDF에서는 dtype=np.float 이었지만 실행 결과 error 발생

```
In [2]: x_seeds = np.array([(0,0), (1,0), (0,1), (1,1)], dtype=np.float)
        y_seeds = np.array([0,1,1,0])
```

```
<ipython-input-2-c6a779e32712>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  x_seeds = np.array([(0,0), (1,0), (0,1), (1,1)], dtype=np.float)
```

➔ dtype=np.float → dtype=float 로 수정하여 error 해결 (numpy version에 따른 error로 사료됨).

dtype 제외 나머지 요소들은 변경하지 않음

Train : remove for-loops

	기존 Train	변경 후 Train

①	`# dW2 - todo: remove for-loops`
	`for i in range(model.num_hiddens):`
	`  dW2[i] += a1[i]*diff`
		`# dW2 : remove for-loops completed`
		`dW2 += a1*diff`

②	`# db1 - todo: remove for-loops`
	`for i in range(model.num_hiddens):`
	`  db1[i] += (1-a1[i]**2)*model.W2[i]*diff`
		`# db1 : remove for-loops completed`
		`db1 += (1-a1**2)*model.W2*diff`

③	`# db2 - todo: remove for-loops`
	`for i in range(model.num_hiddens):`
	`  for j in range(model.num_input_features):`
	`  dW1[i,j] += x[j]*(1-a1[i]**2)*model.W2[i]*diff`
		`# db2 : remove for-loops completed`
		`dW1 += np.outer((1-a1**2)*model.W2*diff, x, out=None)`

①, ②의 경우 dW2, a1 / db1, a1, model.W2이 각각 서로 같은 index i에 대해 for-loop가 적용됨

➔ numpy array 전체에 대한 연산을 적용할 수 있음. 이를 통해 최종적으로 for-loops 제거

```
In [36]: a = np.array([1,2])
         b = np.array([5,1])

In [37]: a+b
Out[37]: array([6, 3])
```

numpy array 전체에 대한 연산 예시

③의 경우 사용하는 index가 I, j 2개여서 dW1, x, a1, model.W2의 index가 서로 호환되지 않아 ①, ②와 같은 전체에 대한 연산을 적용할 수 없음

➔ numpy.outer 사용하여 index가 통일되도록 a, b에 각각 알맞은 값을 대입하여 최종적으로 for-loop 제거

`numpy.outer(a, b, out=None)` [source]

Compute the outer product of two vectors.

Given two vectors, a = [a0, a1, ..., aM] and b = [b0, b1, ..., bN], the outer product [1] is:

```
[[a0*b0  a0*b1 ... a0*bN ]
 [a1*b0    .
 [ ...          .
 [aM*b0           aM*bN ]]
```

a	b
(1 - a1**2) * model.W2 * diff	x

변경된 Train으로 cost check

```
In [13]: for epoch in range(100):
             cost = train(X, Y, model, 1.0)
             if epoch%10 == 0:
                 print(epoch, cost)
```

```
0 [1.02361387]
10 [0.60741637]
20 [0.53821186]
30 [0.45367961]
40 [0.38513224]
50 [0.34099542]
60 [0.31380234]
70 [0.29671977]
80 [0.28552195]
90 [0.27769447]
```

변경된 Train으로도 cost 값이 정상적으로 감소하는 것을 확인할 수 있음

Test

Test

```
In [14]: model.predict((1,1))[0].item()
Out[14]: 0.05507978735458496

In [15]: model.predict((1,0))[0].item()
Out[15]: 0.9089329184307066

In [16]: model.predict((0,1))[0].item()
Out[16]: 0.9111184212597205

In [17]: model.predict((0,0))[0].item()
Out[17]: 0.05754848590969091
```

XOR		
1	1	0
1	0	1
0	1	1
0	0	0

최종적으로 for-loops를 모두 제거한 Train으로도 Test 값이 정상적으로 XOR에 부합하게 출력되는 것을 확인할 수 있음