# Deep Learning practice #3-1 report

컴퓨터소프트웨어학부 2018008768 윤정

**[구현 과정]**

```python
def generate_data(m):
    X = np.empty((0, 1))
    y = np.empty((0, 1))
    for i in range(m):
        degree_value = random.uniform(0, 360)
        cosine_value = math.cos(math.radians(degree_value))
        X = np.append(X, np.array([[degree_value]]), axis=0)
        if cosine_value > 0:
            y = np.append(y, np.array([[1]]), axis=0)
        else:
            y = np.append(y, np.array([[0]]), axis=0)
    return X, y
```

input X와 X의 코사인 값의 부호에 대응되는 {0, 1} 로 구성된 y를 생성하였습니다.

Logistic regression을 Back propagation을 이용하여 W1, b1, W2, b2 를 구하였습니다.

이 과정에서, 계산상의 에러 방지 및 더 좋은 성능을 위해 정규화를 진행하였습니다. degree value인 x값을 radian value값으로 변환한 뒤 train, predict를 진행하였습니다.

구한 parameter W1, b1, W2, b2 값에 대하여 Cost와 Accuracy를 계산하였습니다.

**[결과]**

m = 10000, n = 1000, K = 5000, alpha = 0.9 일 때 결과값 (매 500회마다 출력, 최종 w와 b)

```
0 th result
w1:  0.9879096023466333 b1:  0.9969761780967076
w2:  0.6640717024830809 b2:  0.6600173917256679
500 th result
w1:  3.315319147839785 b1:  -3.874373599755111
w2:  -4.268707634339381 b2:  3.658587937716865
1000 th result
w1:  4.676011536779855 b1:  -5.798844926384464
w2:  -5.580539732288041 b2:  4.923663701937613
1500 th result
w1:  5.487344352817046 b1:  -6.96624818423901
w2:  -6.244748687928985 b2:  5.576094265939145
2000 th result
w1:  6.094040486483692 b1:  -7.849742072447544
w2:  -6.695204933453797 b2:  6.020999938529877
2500 th result
w1:  6.589234927687981 b1:  -8.576673005066608
w2:  -7.038171942802305 b2:  6.360666128714749
3000 th result
w1:  7.012865233854604 b1:  -9.202155892222446
w2:  -7.316227842633927 b2:  6.636501279277417
3500 th result
w1:  7.386079851009958 b1:  -9.755625849601042
w2:  -7.55075369259283 b2:  6.8694142490846275
4000 th result
w1:  7.721558917339698 b1:  -10.254859867336249
w2:  -7.75402113578739 b2:  7.071447877276333
4500 th result
w1:  8.027560547017282 b1:  -10.711507418282471
w2:  -7.933730915184306 b2:  7.250177836198255
w1:  8.309247915152488 b1:  -11.132848324024495
w2:  -8.094728545893782 b2:  7.41037657845174
Train result
Cost:  0.4951824704320686
Accuracy:  74.22999999999999
Test result
Cost:  0.49978066161137624
Accuracy:  73.4
```

alpha값을 변화시키며 예측한 결과값입니다. (M = 10000, N = 1000, k = 5000)

| | alpha=0.01 | alpha=0.05 | alpha=0.1 | alpha=0.5 | alpha=0.9 |
|---|---|---|---|---|---|
| Cost (with 'n' test samples) | 0.68644838<br>20666747 | 0.59486271<br>49325524 | 0.53241209<br>76808924 | 0.50647192<br>70836949 | 0.4976575<br>82533408 |
| Accuracy (with 'n' test samples) | 65.6000 | 70.7 | 72.2 | 73.7 | 74.2 |

가장 작은 cost를 가지는 0.9를 본 실험에서 알파 값으로 사용하였습니다.

Train data m을 변화시키며 예측한 결과값입니다. (alpha=0.9)

| | m=10, n=1000, K=5000 | m=100, n=1000, K=5000 | m=10000, n=1000, K=5000 |
|---|---|---|---|
| Cost (with 'm' train samples) | 0.478997625493468 | 0.504260476713385 | 0.495182470432068 |
| Cost (with 'n' test samples) | 0.706387573309050 | 0.521914037184811 | 0.499780661611376 |

| | m=10, n=1000, K=5000 | m=100, n=1000, K=5000 | m=10000, n=1000, K=5000 |
|---|---|---|---|
| Accuracy (with 'm' train samples) | 70.0 | 77.0 | 74.2299999 |
| Accuracy (with 'n' test samples) | 50.1 | 72.5 | 73.4 |

K를 변화시키며 예측한 결과값입니다. (alpha = 0.9)

| | m=10000, n=1000, K=10 | m=10000, n=1000, K=100 | m=10000, n=1000, K=5000 |
|---|---|---|---|
| Cost (with 'm' train samples) | 0.692105283194714 | 0.681870419711400 | 0.495182470432068 |
| Cost (with 'n' test samples) | 0.694080612373324 | 0.681434568464428 | 0.499780661611376 |

| | m=10000, n=1000, K=10 | m=10000, n=1000, K=100 | m=10000, n=1000, K=5000 |
|---|---|---|---|
| Accuracy (with 'm' train samples) | 50.84999999 | 68.63 | 74.2299999 |
| Accuracy (with 'n' test samples) | 47.09999999 | 68.600000001 | 73.4 |

**[분석]**

학습 진행속도를 위한 alpha 값을 각각 0.01, 0.05, 0.1, 0.5, 0.9로 예측해 본 결과, m=10000, n=1000, k=5000의 환경에서는 0.9가 가장 최적의 알파 값이었음을 알 수 있었습니다.

Train data m을 변화시켰을 때 train data의 크기가 커질수록 train samples와 test samples간의 cost, accuracy 값 차이가 줄어들었습니다.

또한, 학습 반복 수인 K값을 늘리면 늘릴수록 학습의 성능이 향상되는 것을 볼 수 있었습니다.

이번 과제는 sin함수의 결과값을 바탕으로 한 저번 과제보다 성능이 더 좋지 않게 나왔는데, 이는 0~360도 구간에서 cos함수가 양수, 음수가 나뉘는 구간이 더 많아서 학습이 더 어려웠기 때문이라고 판단되었습니다.