Deep Learning practice #2-1 report

컴퓨터소프트웨어학부 2018008768 윤정

[구현 과정]

```python
def logistic_regression_2_regualization(X_train, y_train, m, alpha, epoch):
    w1 =  w2 = b1 = b2 = 1
    W1 = np.array([[w1]])
    W2 = np.array([[w2]])
    X = np.array([[math.radians(x[0]) for x in X_train]])
    for i in range(epoch):
        Z1 = np.dot(X.T, W1) + b1 #(10000, 1) (1, 1) -> (10000, 1)
        A1 = sigmoid(Z1)
        Z2 = np.dot(A1, W2.T) + b2 #(10000, 1) (1, 1) -> (10000, 1)
        A2 = sigmoid(Z2)

        dZ2 = A2 - y_train # (10000, 1)
        dW2 = np.dot(dZ2.T, A1) / m # (1, 10000) (10000, 1) -> (1, 1)
        db2 = np.sum(dZ2) / m
        W2 = W2 - alpha * dW2
        b2 = b2 - alpha * db2

        dZ1 = np.dot(dZ2, W2) * sigmoid(Z1) * (1 - sigmoid(Z1)) #(10000, 1) (1, 1)
        dW1 = np.dot(X, dZ1) / m #(1, 10000) (10000, 1)
        db1 = np.sum(dZ1) / m
        W1 = W1 - alpha * dW1
        b1 = b1 - alpha * db1
        if i % 500 == 0:
            print(i, "th result")
            print("w1: ", W1[0][0], "b1: ", b1)
            print("w2: ", W2[0][0], "b2: ", b2)
    return W1, b1, W2, b2
```

Practice 1-2에서 진행했던 것과 동일하게 input X와 {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 실험을 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:  0.9980165839771242 b1:  1.0019764321990512
w2:  0.6612649464905885 b2:  0.6651180663455694
500 th result
w1:  2.354391944232708 b1:  -7.381520059086915
w2:  -9.47839403264392 b2:  4.657415539826407
1000 th result
w1:  2.9606558136046157 b1:  -9.312415181421747
w2:  -11.693819084296901 b2:  5.721560443388819
1500 th result
w1:  3.335573235478897 b1:  -10.49994734566138
w2:  -12.959420410398517 b2:  6.327941767118357
2000 th result
w1:  3.6178846885258547 b1:  -11.393060256493236
w2:  -13.870256291465754 b2:  6.761804169000376
2500 th result
w1:  3.8485150868155835 b1:  -12.12238068772574
w2:  -14.592301120577384 b2:  7.1036741388014635
3000 th result
w1:  4.0455958911481105 b1:  -12.745506685635142
w2:  -15.196144037270118 b2:  7.387976585405543
3500 th result
w1:  4.21888775028573 b1:  -13.293374001328864
w2:  -15.718563117593703 b2:  7.632694923219604
4000 th result
w1:  4.374301233003262 b1:  -13.78469286601261
w2:  -16.181229688330337 b2:  7.848438906888983
4500 th result
w1:  4.515711738980613 b1:  -14.231722709037411
w2:  -16.59802446956535 b2:  8.042006302263555
w1:  4.645560523718329 b1:  -14.642183610232033
w2:  -16.977663986762128 b2:  8.217684446308573
Train result
Cost:  0.014578705977754394
Accuracy:  99.98
Test result
Cost:  0.016844615188822784
Accuracy:  99.9
```

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)	0.671444826246355	0.11022152007795825	0.05207759044485469	0.03323181182795685	0.01662077096127
Accuracy (with 'n' test samples)	72.7	99.1	99.8	98.3	100.0

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)	0.00130054369998	0.005304403602337	0.014578705977754
Cost (with 'n' test samples)	0.092721242010896	0.034993528607039	0.016844615188822

	m=10, n=1000, K=5000	m=100, n=1000, K=5000	m=10000, n=1000, K=5000
Accuracy (with 'm' train samples)	100.0	100.0	99.98
Accuracy (with 'n' test samples)	96.39999	98.1	99.9

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)	0.691027501062590	0.523459438328402	0.014578705977754
Cost (with 'n' test samples)	0.689818730212681	0.525056743681524	0.016844615188822

	m=10000, n=1000, K=10	m=10000, n=1000, K=100	m=10000, n=1000, K=5000
Accuracy (with 'm' train samples)	50.07	88.9	99.98
Accuracy (with 'n' test samples)	53.300000000000004	88.3	99.9

[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 m, 반복 수 K를 변화시켰을 때 결과는 전반적으로 Practice#1-2와 비슷한 경향을 보였습니다. 레어어 한 개를 가지고 진행했던 저번 실험에서는 m = 10000, n = 1000, K = 5000일 때 Cost는 대략 0.183(test data 기준), 정확도는 95.8999 정도가 나왔습니다.

같은 train data수와 같은 반복 수에서 레이어를 하나 더 증가시킴으로 더 좋은 성능을 낼 수 있었습니다.