Deep Learning practice #1 report

컴퓨터소프트웨어학부 2018008768 윤정

[구현 과정]

```python
def generate_data(m):
    X = np.empty((0, 2))
    y = np.empty((0, 1))
    for i in range(m):
        x1 = random.uniform(-10, 10)
        x2 = random.uniform(-10, 10)
        X = np.append(X, np.array([[x1, x2]]), axis=0)
        if x1 + x2 > 0:
            y = np.append(y, np.array([[1]]), axis=0)
        else:
            y = np.append(y, np.array([[0]]), axis=0)
    return X, y
```

Input number m에 대해 (m, 2) 크기의 X와 0과 1로 구성된 (m, 1)크기의 y데이터를 생성하는 함수입니다. 이를 이용해서 X_train, X_test, y_train, y_test를 생성하였습니다.

Logistic regression을 Back propagation을 이용하여 W = [w1, w2] 와 b를 구하였습니다.

구한 parameter W, b값에 대하여 Cost와 Accuracy를 계산하였습니다.

[결과]

m = 10000, n = 1000, K = 5000, alpha = 0.1 일 때 결과값 예시

```
w1:  1.0063315893593348 w2:  1.003248183628516
b:  0.9952927452631634
w1:  1.9445216436035218 w2:  1.9371572197295848
b:  0.20705284654549377
w1:  2.372949981118904 w2:  2.3665652013907206
b:  0.07071334306245322
w1:  2.6862021082374836 w2:  2.6801664879529192
b:  0.032555762600047715
w1:  2.940537184954099 w2:  2.934651599212572
b:  0.02053109568425003
w1:  3.1578237583077224 w2:  3.152002770080998
b:  0.016951925719231077
w1:  3.349242529314346 w2:  3.343442287386669
b:  0.01638141500982106
w1:  3.5213844031693746 w2:  3.5155796011134406
b:  0.016923002893755507
w1:  3.6785018318469094 w2:  3.6726766645554734
b:  0.017861211484757934
w1:  3.823517625719268 w2:  3.817661681012643
b:  0.018913713823563407
Final w1:  3.958276244783336 Final w2:  3.9523824964817313
Final b:  0.019965691009181843
```

```
predict(X_train, y_train, m, W, b)
```

```
Cost:  0.02097648120414138
Accuracy:  99.96000000000001
```

```
predict(X_test, y_test, n, W, b)
```

```
Cost:  0.02253936569178261
Accuracy:  99.9
```

Train data m을 변화시키며 예측한 결과값입니다.

	m=10, n=1000, K=5000	m=100, n=1000, K=5000	m=10000, n=1000, K=5000
Cost (with 'm' train samples)	0.0009584001296777594	0.016491897332443217	0.020976481204144138
Cost (with 'n' test samples)	0.14412792127476914	0.056890544343576696	0.022539365691782261

	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	99.96000000000001
Accuracy (with 'n' test samples)	94.69999999999999	97.8	99.9

K를 변화시키며 예측한 결과값입니다.

	m=10000, n=1000, K=10	m=10000, n=1000, K=100	m=10000, n=1000, K=5000
Cost (with 'm' train samples)	0.0949457670447307	0.0651663408604182	0.0210463265298375532
Cost (with 'n' test samples)	0.11006085070911938	0.07922155420362213	0.0232131015454389278927

	m=10000, n=1000, K=10	m=10000, n=1000, K=100	m=10000, n=1000, K=5000
Accuracy (with 'm' train samples)	95.55	97.69	99.87
Accuracy (with 'n' test samples)	94.69999999999999	96.7	99.7

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

	alpha=0.01	alpha=0.05	alpha=0.1	alpha=0.3	alpha=0.5
Cost (with 'n' test samples)	0.04741681902172714	0.027312061464464435	0.023213101545438927	0.011563252372891719	0.013881695996955564
Accuracy (with 'n' test samples)	99.0	99.9	99.7	100.0	99.9

[분석]

Train data의 크기가 커질수록 train data를 이용하여 predict한 결과값과 test data를 이용하여 predict한 결과값의 차이가 줄어들었습니다. 이로 인해 train data가 커질수록 새로운 데이터에 대한 예측의 정확도가 높아짐을 알 수 있었습니다.

또한, 학습 반복횟수인 K (과제에서는 epoch으로 표현)을 증가시킬수록 cost는 감소하며 accuracy는 증가하는 것을 알 수 있었습니다. 이를 통해 학습 반복횟수가 증가할수록 더 정확한 W와 b값을 추정할 수 있음을 알 수 있었습니다.

마지막으로, 학습 진행속도를 위한 alpha 값을 각각 0.01, 0.05, 0.1, 0.3, 0.5를 넣어 예측해 본 결과, 본 실험에서는 0.3이 가장 최적의 알파 값이었음을 알 수 있었습니다. 또한, alpha값이 클수록 각 단계(매 500에폭마다)의 변화가 더 크게 일어남을 볼 수 있었습니다.