**Coding과제 1**

목차

1. Diabetes.csv 모델 소스코드
2. 실행 결과

학번 20140269 이름 고혁훈



* 과제의 목적

Diabetes 데이터를 가지고 모델을 돌려본다. 여러 번 돌려봐서 accuracy가 높은 것을 찾아본다. Softmax도 써보고 sigmoid도 써본다.

Weight 및 parameter 제출한다

* 배운 점

여러 번 돌리다 보면 성능이 높아질 때도 낮아질 때도 있다. 똑같은 스텝으로 학습해도 매번 돌릴 때마다 성능이 특정 구간에 빠져 성능이 들쭉날쭉하다. 초기 파라미터 세팅 값이 무엇이냐에 도달 포인트가 달라지는 것 같다. 확실히 init parameter를 제어해야겠다.

이번 경우에는 클래스가 두 개라서 마지막 레이어 노드를 한 개도 가능하고 두개도 가능한데, 둘 다 돌릴 때마다 성능이 다르다. 둘 다 학습했을 때 결과는 비슷하게 나온다. 노드가 1개이든 2개이든 큰 차이는 없는 것 같다.

반면 layer 개수를 늘렸을 때는 1개 layer에서 0.78을 넘기 힘들었던 것이 0.80까지 올라가는 것을 보았다. 이는 layer가 늘면 성능도 더 좋아지는 것 같다. 대신 레이어가 늘어나는 만큼 epoch도 1000까지 늘려줘야 하는 것 같다.

* 소스코드 - sigmoid

import numpy as np

import tensorflow as tf

#자료 불러옴

xy = np.loadtxt('data-lab-2-5-csv.csv',delimiter=',',dtype=np.float32)

x\_data = xy[:,:-1]

y\_data = xy[:,[-1]]

# logistic

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(1, activation='sigmoid')

])

# compile

model.compile(optimizer = "adam", loss="binary\_crossentropy",metrics=["binary\_accuracy"])

# train

history = model.fit(x\_data, y\_data, epochs=150)

* 실행결과 - sigmoid

1. 모델 summary

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 1) 9

=================================================================

Total params: 9

Trainable params: 9

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. epoch결과 일부

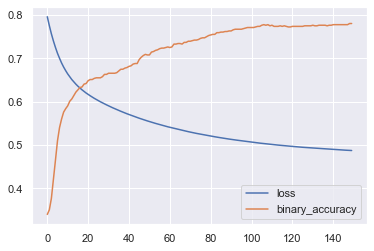
Epoch 149/150

24/24 [==============================] - 0s 1ms/step - loss: 0.4875 - binary\_accuracy: 0.7800

Epoch 150/150

24/24 [==============================] - 0s 1ms/step - loss: 0.4872 - binary\_accuracy: 0.7800

1. fit 결과 도표



1. 예측결과

print(y\_data[0])

model.predict(x\_data[0:1])

[0.]

array([[0.42964232]], dtype=float32)

1. **Accuracy, weight**

* 최고 accuracy

Epoch 149/150

24/24 [==============================] - 0s 1ms/step - loss: 0.4875 - binary\_accuracy: 0.7800

* model.weights

[<tf.Variable 'dense\_4/kernel:0' shape=(8, 1) dtype=float32, numpy=

array([[-0.6114759 ],

[-2.2263365 ],

[-0.08641764],

[-0.5845429 ],

[-0.3207873 ],

[-2.090146 ],

[-0.39614138],

[-0.35214156]], dtype=float32)>,

<tf.Variable 'dense\_4/bias:0' shape=(1,) dtype=float32, numpy=array([0.2473545], dtype=float32)>]

* 소스코드 - softmax

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(2, activation='softmax')

])

model.compile(optimizer = "adam", loss="sparse\_categorical\_crossentropy",metrics=["accuracy"])

history = model.fit(x\_data, y\_data, epochs=150)

* 실행결과 - softmax

1. 모델 summary

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_2 (Dense) (None, 2) 18

=================================================================

Total params: 18

Trainable params: 18

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. epoch결과 일부

Epoch 79/300

24/24 [==============================] - 0s 959us/step - loss: 0.4880 - accuracy: 0.7800

Epoch 80/300

24/24 [==============================] - 0s 959us/step - loss: 0.4874 - accuracy: 0.7800

Epoch 81/300

24/24 [==============================] - 0s 959us/step - loss: 0.4872 - accuracy: 0.7826

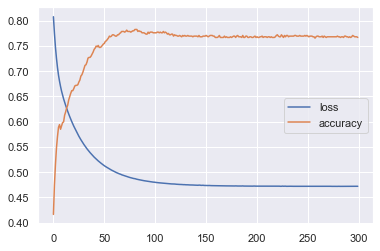
Epoch 82/300

24/24 [==============================] - 0s 959us/step - loss: 0.4865 - accuracy: 0.7826

Epoch 83/300

24/24 [==============================] - 0s 959us/step - loss: 0.4860 - accuracy: 0.7826

1. fit 결과 도표



1. 예측결과

print(y\_data[0])

model.predict(x\_data[0:1])

[0.]

array([[0.6405423 , 0.35945767]], dtype=float32)

1. **Accuracy, weight**

* 최고 accuracy
* Epoch 82/300
* 24/24 [==============================] - 0s 959us/step - loss: 0.4865 - accuracy: 0.7826
* model.weights

[<tf.Variable 'dense\_14/kernel:0' shape=(8, 2) dtype=float32, numpy=

array([[ 0.7726362 , -0.10641171],

[ 1.6965446 , -1.8706745 ],

[ 0.02106451, 0.37481353],

[ 0.40401825, -0.10073677],

[ 0.22873777, -0.11498593],

[ 1.4381422 , -1.1998248 ],

[ 0.9907157 , 0.07795638],

[ 0.22539772, 0.12526822]], dtype=float32)>,

<tf.Variable 'dense\_14/bias:0' shape=(2,) dtype=float32, numpy=array([-0.10332352, 0.10332356], dtype=float32)>]

* 실행결과 – sigmoid + layer 증가

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(8, activation='relu'),

tf.keras.layers.Dense(4, activation='relu'),

tf.keras.layers.Dense(2, activation='relu'),

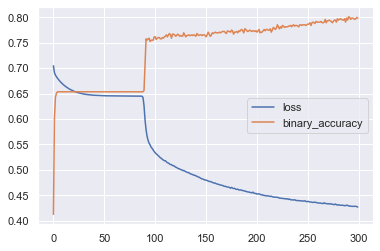
tf.keras.layers.Dense(1, activation='sigmoid')

])

* Accuracy 0.80

Epoch 291/300

24/24 [==============================] - 0s 1ms/step - loss: 0.4290 - binary\_accuracy: 0.8011



* Weight

[<tf.Variable 'dense\_15/kernel:0' shape=(8, 8) dtype=float32, numpy=

array([[-0.2379802 , 0.581052 , -0.20999284, 0.14334556, -0.8027803 ,

0.4944544 , -0.2333827 , -0.13113791],

[ 0.62185824, 0.62741053, 0.16601647, -0.45407158, -0.57169265,

0.1334911 , 0.5628964 , 0.52011037],

[-0.08953063, 0.00520621, -0.02584101, -0.4998019 , -0.0187176 ,

0.25509948, -0.29177976, -0.81669486],

[-0.44449157, -0.5097548 , -0.29234803, -0.05520047, -0.7922143 ,

0.5876517 , 0.63043964, -0.5080687 ],

[-0.26138198, 0.39202344, 0.20054159, -0.56013745, -0.8069655 ,

0.589866 , -0.34802336, -0.4804043 ],

[ 0.58953255, 0.8067826 , -0.2766758 , -0.33899072, -0.3427777 ,

-0.60100144, -0.48340917, -0.14081873],

[ 0.07715072, 0.16371733, -0.8314803 , -0.48185882, -0.05525413,

0.85402197, -0.27984235, 0.10119109],

[ 0.37662026, -0.3165575 , -0.12647164, 1.0532209 , -0.27676216,

0.70150304, 0.13055255, 0.3750276 ]], dtype=float32)>,

<tf.Variable 'dense\_15/bias:0' shape=(8,) dtype=float32, numpy=

array([ 0.58775264, 0.234865 , -0.31053048, 0.00184566, -0.18167815,

0.2762637 , 0.3139097 , 0.61635995], dtype=float32)>,

<tf.Variable 'dense\_16/kernel:0' shape=(8, 4) dtype=float32, numpy=

array([[-0.38207015, 0.32221478, -1.0626985 , 1.1372423 ],

[-1.1974181 , 0.19298644, 0.39193273, 0.34069195],

[ 0.19093393, -0.24620405, -0.04667053, -0.6322159 ],

[ 1.7752773 , -0.6367983 , 0.00919275, -1.019534 ],

[ 0.05976728, -1.703134 , -0.70152116, -0.9388381 ],

[ 1.1036714 , 1.0367936 , -0.1700741 , -2.017544 ],

[ 0.25799468, -0.22921693, 0.7830306 , 0.70457447],

[ 0.07268799, -0.16070814, -0.6441986 , 0.46166697]],

dtype=float32)>,

<tf.Variable 'dense\_16/bias:0' shape=(4,) dtype=float32, numpy=array([ 0.01543884, -0.10423391, 0.16067018, 0.4542276 ], dtype=float32)>,

<tf.Variable 'dense\_17/kernel:0' shape=(4, 2) dtype=float32, numpy=

array([[-1.7509553 , -1.3101975 ],

[-1.4199821 , -1.6638746 ],

[-0.8663813 , 1.1061872 ],

[ 1.6605636 , -0.04156792]], dtype=float32)>,

<tf.Variable 'dense\_17/bias:0' shape=(2,) dtype=float32, numpy=array([ 0.01557585, -0.14693934], dtype=float32)>,

<tf.Variable 'dense\_18/kernel:0' shape=(2, 1) dtype=float32, numpy=

array([[-1.1528689],

[-1.883187 ]], dtype=float32)>,

<tf.Variable 'dense\_18/bias:0' shape=(1,) dtype=float32, numpy=array([2.4535294], dtype=float32)>]

* 실행결과 – softmax+ layer 증가

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(8, activation='relu'),

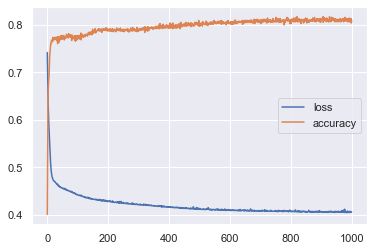
tf.keras.layers.Dense(4, activation='relu'),

tf.keras.layers.Dense(2, activation='softmax')

])

Epoch 992/1000

24/24 [==============================] - 0s 1ms/step - loss: 0.4052 - accuracy: 0.8142



* Weight

[<tf.Variable 'dense\_22/kernel:0' shape=(8, 8) dtype=float32, numpy=

array([[-1.0679356e+00, -4.5867455e-01, -2.4928129e-01, 3.0380368e-01,

-4.9014807e-01, -3.7941965e-01, 9.2594899e-02, 6.8617582e-01],

[-3.4238771e-01, -4.4150347e-01, 5.8021200e-01, 5.7393730e-01,

2.8387051e-02, 2.4478337e-01, -6.6318738e-01, 2.7969098e-01],

[-6.8051994e-02, -5.0008488e-01, -1.4496271e-01, -1.1317427e+00,

1.6438772e-01, 4.1129661e-01, -4.0930599e-01, 4.2087927e-01],

[-8.1399739e-01, 2.5273490e-01, 2.9978731e-01, -2.0041114e-01,

-5.5447556e-02, 3.2411501e-01, 4.4334075e-01, -7.9410642e-01],

[ 1.6387593e-02, -1.3789847e-01, 1.5578571e-01, 4.6317449e-01,

-4.4951954e-01, 2.8450093e-01, -7.3308659e-01, 4.0504694e-01],

[-1.1424785e+00, 5.0872761e-01, 9.6995227e-02, -3.2160813e-01,

7.2903359e-01, -1.6433300e-01, -5.3482568e-01, -5.3851801e-01],

[-1.4946501e-01, -1.0176933e+00, -3.6032736e-01, -2.0587060e-01,

-4.4078314e-01, -5.4720575e-01, -1.8530451e-01, -6.3395721e-01],

[ 3.5205624e-01, -2.2359590e-01, 4.0767029e-01, -5.5667222e-01,

-7.0213655e-04, -1.0989077e+00, 1.1829969e+00, 1.1583945e-01]],

dtype=float32)>,

<tf.Variable 'dense\_22/bias:0' shape=(8,) dtype=float32, numpy=

array([-0.18192951, -0.05923195, 0.27890223, -0.07316122, 0.35520595,

-0.0326535 , 0.21331179, -0.18359233], dtype=float32)>,

<tf.Variable 'dense\_23/kernel:0' shape=(8, 4) dtype=float32, numpy=

array([[ 0.28389174, 0.64759916, 0.00622787, -1.2733691 ],

[-0.12409856, 0.5713052 , -0.65637004, -0.59574115],

[ 0.66351306, -0.86580074, 0.40037718, -0.749498 ],

[ 0.38940072, -0.4160218 , 0.9136223 , 0.8403244 ],

[ 0.55550754, -0.46626887, 0.6852102 , 0.20422077],

[ 0.40514302, 0.73858476, 0.4375885 , -0.58901715],

[-0.64482 , 2.2319617 , -0.4764065 , -0.48584932],

[-0.3216152 , 0.50212944, -0.0677949 , 0.82397395]],

dtype=float32)>,

<tf.Variable 'dense\_23/bias:0' shape=(4,) dtype=float32, numpy=array([-0.34888217, -0.1855588 , 0.11608436, 0.18941581], dtype=float32)>,

<tf.Variable 'dense\_24/kernel:0' shape=(4, 2) dtype=float32, numpy=

array([[-0.2193013 , -0.8677016 ],

[-1.3828671 , 1.9855688 ],

[ 0.31914425, -0.7665798 ],

[ 2.1553316 , -3.068916 ]], dtype=float32)>,

<tf.Variable 'dense\_24/bias:0' shape=(2,) dtype=float32, numpy=array([-0.31554496, 0.31554493], dtype=float32)>]