early-stopping

November 29, 2023

[1]:
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
import seaborn as sns
```

[2]:
```
df=pd.read_csv('/content/data.csv')
df
```

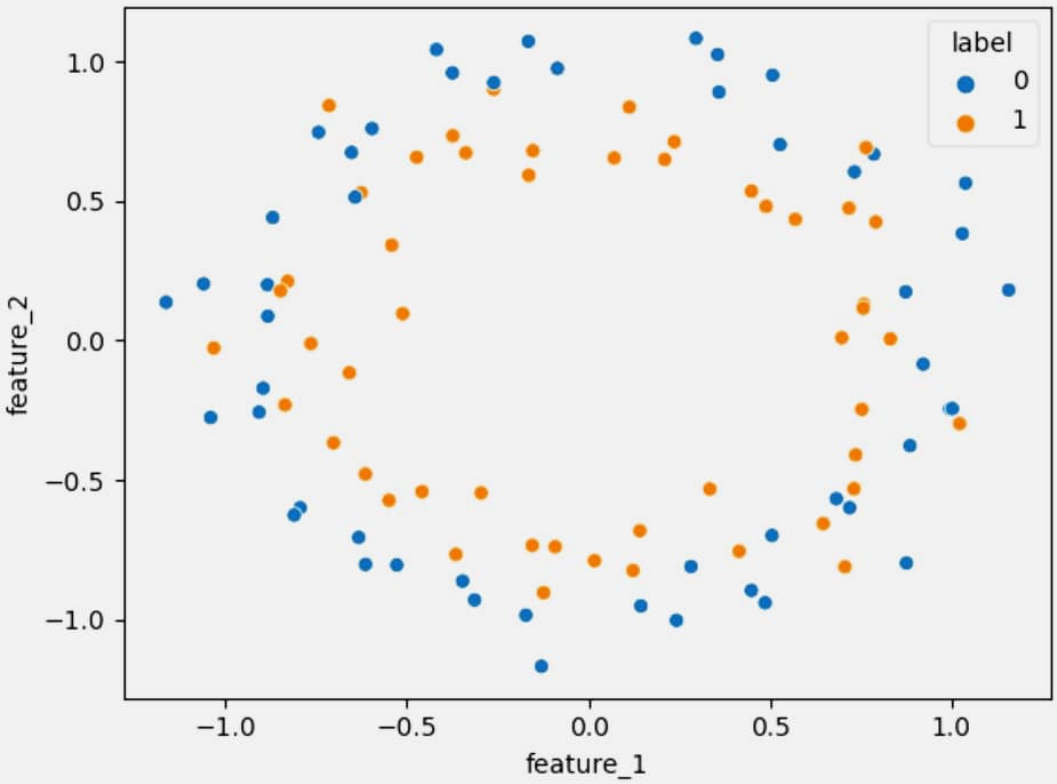
[2]:

	feature_1	feature_2	label
0	-0.365031	-0.766134	1
1	0.791425	0.424866	1
2	-0.626055	0.529933	1
3	0.719383	-0.598984	0
4	0.875863	-0.796937	0
..	…	…	…
95	0.236494	0.712223	1
96	-0.541747	0.342009	1
97	-0.260651	0.924460	0
98	0.415090	-0.754813	1
99	0.756847	0.115987	1

[100 rows x 3 columns]

[3]:
```
sns.scatterplot(x='feature_1', y='feature_2', hue='label', data=df)


plt.show()
```



[4]:
```
X=df.drop('label',axis=1)
y=df['label']
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10,␣
↪random_state=42)
```

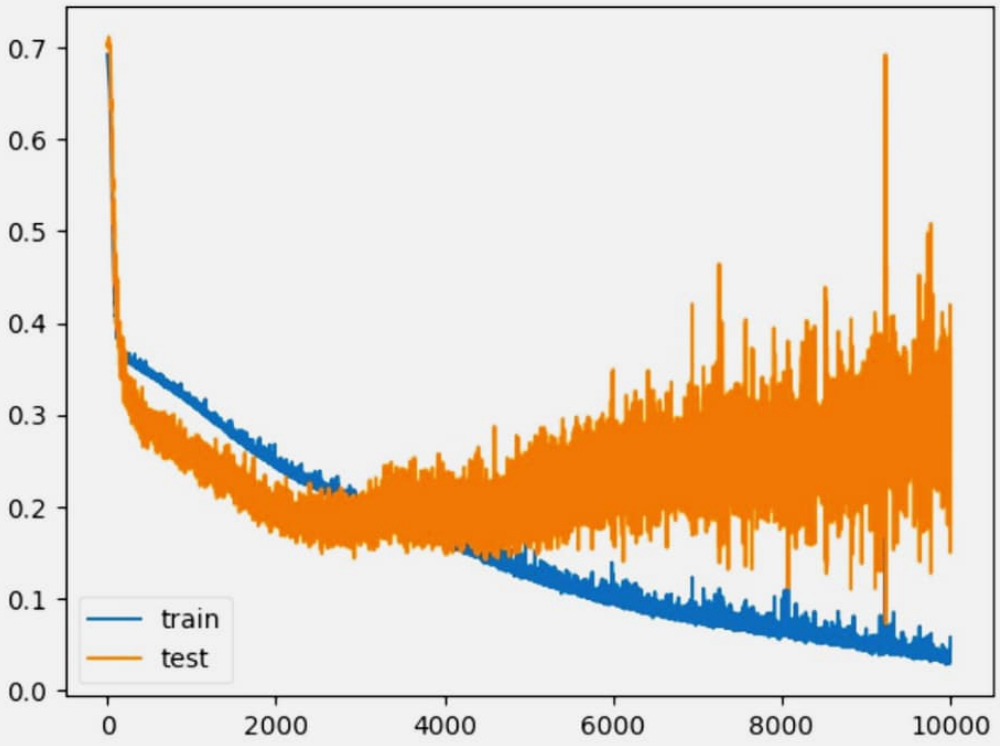
[6]:
```
model = Sequential()
model.add(Dense(256, input_dim=2, activation='relu'))# input_dim=2 vaneko␣
↪features 2 ota xa . 1st layer ma 256 nodes
model.add(Dense(32, activation='relu')) # second layer ko nodes 32 ota rakheko
model.add(Dense(1, activation='sigmoid')) #output layer ma euta node
```

[7]:
```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
```

[8]:
```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪epochs=10000, verbose=0)
```

[9]:
```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
```

```
plt.show()
```



about 2300 epoch paxi
over fitting va xa

1 EARLY STOPPING

[26]:
```
model = Sequential()

model.add(Dense(256, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

[27]:
```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
```

[28]:
```
callback = EarlyStopping(
    monitor="val_loss",
    min_delta=0.0001,
    patience=200,
    verbose=1,
    mode="auto",
    restore_best_weights=True
)
```

previous best Val loss - current Val loss

0.6926 - 0.6933 = - 0.0007

[29]:
```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),␣
↪epochs=10000, callbacks=callback)
```

```
Epoch 1/10000
3/3 [==============================] - 1s 91ms/s
0.5000 - val_loss: 0.6926 - val_accuracy: 0.500
Epoch 2/10000
3/3 [==============================] - 0s 17ms/
0.4889 - val_loss: 0.6933 - val_accuracy: 0.500
Epoch 3/10000
3/3 [==============================] - 0s 16ms/step
0.5111 - val_loss: 0.6967 - val_accuracy: 0.5000
Epoch 4/10000
3/3 [==============================] - 0s 16
0.5333 - val_loss: 0.6972 - val_accuracy: 0.5000
Epoch 5/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6900 - accuracy:
0.5333 - val_loss: 0.6993 - val_accuracy: 0.3000
Epoch 6/10000
3/3 [==============================] - 0s 16ms/step - loss: 0.6893 - accuracy:
0.5556 - val_loss: 0.7004 - val_accuracy: 0.3000
Epoch 7/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6886 - accuracy:
0.6000 - val_loss: 0.7024 - val_accuracy: 0.4000
Epoch 8/10000
3/3 [==============================] - 0s 17ms/step - loss: 0.6876 - accuracy:
0.6111 - val_loss: 0.7014 - val_accuracy: 0.4000
Epoch 9/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6868 - accuracy:
0.5889 - val_loss: 0.7023 - val_accuracy: 0.4000
Epoch 10/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6863 - accuracy:
0.5778 - val_loss: 0.7019 - val_accuracy: 0.4000
Epoch 11/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6857 - accuracy:
0.5444 - val_loss: 0.7018 - val_accuracy: 0.4000
Epoch 12/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6851 - accuracy:
0.5444 - val_loss: 0.7035 - val_accuracy: 0.4000
Epoch 13/10000
3/3 [==============================] - 0s 15ms/step - loss: 0.6842 - accuracy:
0.5444 - val_loss: 0.7050 - val_accuracy: 0.4000
Epoch 14/10000
3/3 [==============================] - 0s 16ms/step - loss: 0.6836 - accuracy:
0.5444 - val_loss: 0.7062 - val_accuracy: 0.4000
Epoch 15/10000
3/3 [==============================] - 0s 17ms/step - loss: 0.6830 - accuracy:
```

Early stoping is triggered and now it waits for 200 epoches more to get less val loss than 0.6926 until the diff of previous best vloss(0.6926) - curent weight will be pos >= 0.0001

0.3081 - 0.3124 = - 0.0043

```
0.8444 - val_loss: 0.3235 - val_accuracy: 0.8000
Epoch 2016/10000
3/3 [==============================] - 0s 33ms/step - loss: 0.3601 - accuracy:
0.8444 - val_loss: 0.3243 - val_accuracy: 0.8000
Epoch 2017/10000
3/3 [==============================] - 0s 31ms/step - loss: 0.3602 - accuracy:
0.8333 - val_loss: 0.3211 - val_accuracy: 0.8000
Epoch 2018/10000
3/3 [==============================] - 0s 29ms/step - loss: 0.3601 - accuracy:
0.8333 - val_loss: 0.3205 - val_accuracy: 0.8000
Epoch 2019/10000
3/3 [==============================] - 0s 25ms/step - loss: 0.3603 - accuracy:
0.8333 - val_loss: 0.3244 - val_accuracy: 0.9000
Epoch 2020/10000
3/3 [==============================] - 0s 25ms/step - loss: 0.3610 - accuracy:
0.8444 - val_loss: 0.3266 - val_accuracy: 0.8000
Epoch 2021/10000
3/3 [==============================] - 0s 31ms/step - loss: 0.3606 - accuracy:
0.8333 - val_loss: 0.3251 - val_accuracy: 0.8000
Epoch 2022/10000
3/3 [==============================] - 0s 31ms/step - loss: 0.3600 - accuracy:
0.8333 - val_loss: 0.3252 - val_accuracy: 0.8000
Epoch 2023/10000
3/3 [==============================] - 0s 30ms/step - loss: 0.3602 - accuracy:
0.8333 - val_loss: 0.3249 - val_accuracy: 0.8000
Epoch 2024/10000
3/3 [==============================] - 0s 31ms/step - loss: 0.3597 - accuracy:
0.8333 - val_loss: 0.3221 - val_accuracy: 0.8000
Epoch 2025/10000
3/3 [==============================] - 0s 42ms/step - loss: 0.3598 - accuracy:
0.8333 - val_loss: 0.3170 - val_accuracy: 0.8000
Epoch 2026/10000
3/3 [==============================] - 0s 32ms/step - loss: 0.3598 - accuracy:
0.8444 - val_loss: 0.3139 - val_accuracy: 0.8000
Epoch 2027/10000
3/3 [==============================]
0.8444 - val_loss: 0.3087 - val_accu
Epoch 2028/10000
3/3 [==============================]
0.8444 - val_loss: 0.3124 - val_accurac
Epoch 2029/10000
3/3 [==============================]
0.8333 - val_loss: 0.3186 - val_accur
Epoch 2030/10000
3/3 [==============================] - 0s 35ms/st
0.8333 - val_loss: 0.3225 - val_accuracy: 0.8000
Epoch 2031/10000
3/3 [==============================] - 0s 31ms/step - loss: 0.3601 - accuracy:
```

early stoping trigger vho aba next 200 epochs samma yo loss vanda kunai kam loss aauxa ki aaudina herxa ra yo (0.3087 - current weights) ko values >= 0.001 aauxa ki nai herxa aayena vane traing stop hunca

2027 epoch paxi next 200 epoch samma ma kunai epoch ma val loss less than 0.3087 aayena. sabai ko previous best weight(0.3087) - current weight garda < 0.001 nai aayo (neg value) so training rokiyo

```
0.8556 - val_loss: 0.3239 - val_accuracy: 0.8000
Epoch 2224/10000
3/3 [==============================] - 0s 20ms/step - loss: 0.3593 - accuracy:
0.8556 - val_loss: 0.3183 - val_accuracy: 0.8000
Epoch 2225/10000
3/3 [==============================] - 0s 28ms/step - loss: 0.3593 - accuracy:
0.8444 - val_loss: 0.3132 - val_accuracy: 0.8000
Epoch 2226/10000
3/3 [==============================] - 0s 29ms/step - loss: 0.3593 - accuracy:
0.8444 - val_loss: 0.3154 - val_accuracy: 0.8000
Epoch 2227/10000
1/3 [=========>...] - ETA: 0s - loss: 0.3576 - accuracy:
0.8438Restoring model weights from the end of the best epoch: 2027.
3/3 [==============================] - 0s 34ms/step - loss: 0.3592 - accuracy:
0.8444 - val_loss: 0.3207 - val_accuracy: 0.8000
Epoch 2227: early stopping
```

[30]:
```
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```