`Date 25/04/23`

Name:- Krishna Mundada

Roll No:- 45

Batch:- E3

Practical 5

Topic:- Regularization

Write a program to implement L1, L2 regularization, Early stopping, Dropout on a feed forward neural network on sonar dataset. Use Sequential layer and create a dense network.

1. Use 3 hidden layers.
2. Don't Use autoencoder and produce results.
3. Iteration = 100
4. Early stopping (stop when 5 consecutive values are same)
5. Apply drop-out on 1st hidden layer.

Print Table to show each case train and test accuracy.

DL lab: 6

In [ ]:
```
import pandas as pd
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [ ]:
```
dataframe = pd.read_csv("./sonar.csv", header=None)
dataset = dataframe.values

# split into input (X) and output (Y) variables
X = dataset[:,0:60].astype(float)
Y = dataset[:,60]
```

In [ ]:
```
# encode class values as integers
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X,encoded_Y,test_size=0.2, random_state=0)
```

1. Base

In [ ]:
```
models = {}
```

In [ ]:
```
def create_baseline():
        # create model
        model = Sequential()
        model.add(Dense(60, input_shape=(60,), activation='relu'))
        model.add(Dense(40, activation='relu'))
        model.add(Dense(30, activation='relu'))
        model.add(Dense(50, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        # Compile model
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
```

In [ ]:
```
model1 = create_baseline()
model1.fit(X_train,y_train,epochs=100)
```

In [ ]:
```
model_name = "base"
models[model_name] = {}

model = model1

print("on traning data: ")
train = model.evaluate(X_train,y_train)
models[model_name]["train"] = round(train[1]*100,4)

print("on testing data: ")
test = model.evaluate(X_test,y_test)
models[model_name]["test"] = round(test[1]*100, 4)

print("data: ")
models[model_name]
```

```
on traning data:
6/6 [==============================] - 0s 8ms/step - loss: 0.0061 - accuracy: 1.0000
on testing data:
2/2 [==============================] - 0s 7ms/step - loss: 0.4936 - accuracy: 0.8571
data:
```

Out[ ]:
```
{'train': 100.0, 'test': 85.7143}
```

2. L1 regularization

In [ ]:
```
def create_regularization():
        # create model
        model = Sequential()
        model.add(Dense(60, input_shape=(60,), activation='relu'))
        model.add(Dense(40, activation='relu'))
        model.add(Dense(30, activation='relu'))
        model.add(Dense(50, activation='relu'))
        model.add(Dense(1, activation='sigmoid', kernel_regularizer=keras.regularizers.l1(0.1)))
        # Compile model
```

```
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
```

In [ ]:
```
model2 = create_regularization()
model2.fit(X_train,y_train,epochs=100)
```

In [ ]:
```
model_name = "l1"
models[model_name] = {}

model = model2

print("on traning data: ")
train = model.evaluate(X_train,y_train)
models[model_name]["train"] = round(train[1]*100,4)

print("on testing data: ")
test = model.evaluate(X_test,y_test)
models[model_name]["test"] = round(test[1]*100, 4)

print("data: ")
models[model_name]
```

```
on traning data:
6/6 [==============================] - 0s 6ms/step - loss: 0.1850 - accuracy: 0.9819
on testing data:
2/2 [==============================] - 0s 8ms/step - loss: 0.4501 - accuracy: 0.8095
data:
```

Out[ ]:
```
{'train': 98.1928, 'test': 80.9524}
```

3. L2 Regularization

In [ ]:
```
def create_regularization_2():
        # create model
        model = Sequential()
        model.add(Dense(60, input_shape=(60,), activation='relu'))
        model.add(Dense(40, activation='relu'))
        model.add(Dense(30, activation='relu'))
        model.add(Dense(50, activation='relu'))
        model.add(Dense(1, activation='sigmoid', kernel_regularizer=keras.regularizers.l2(0.1)))
        # Compile model
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        return model
```

In [ ]:
```
model3 = create_regularization_2()
model3.fit(X_train,y_train,epochs=100)
```

In [ ]:
```
model_name = "l2"
models[model_name] = {}

model = model3

print("on traning data: ")
train = model.evaluate(X_train,y_train)
models[model_name]["train"] = round(train[1]*100,4)

print("on testing data: ")
```

```
test = model.evaluate(X_test,y_test)
models[model_name]["test"] = round(test[1]*100, 4)

print("data: ")
models[model_name]
```

```
on traning data:
6/6 [==============================] - 0s 9ms/step - loss: 0.0596 - accuracy: 1.0000
on testing data:
2/2 [==============================] - 0s 11ms/step - loss: 0.3794 - accuracy: 0.8333
data:
```

Out[ ]:
```
{'train': 100.0, 'test': 83.3333}
```

4. Early stopping

In [ ]:
```
model4 = create_baseline()
early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
model4.fit(X_train,y_train,epochs=100, callbacks=[early_stopping])
```

In [ ]:
```
model_name = "early stopping"
models[model_name] = {}

model = model4

print("on traning data: ")
train = model.evaluate(X_train,y_train)
models[model_name]["train"] = round(train[1]*100,4)

print("on testing data: ")
test = model.evaluate(X_test,y_test)
models[model_name]["test"] = round(test[1]*100, 4)

print("data: ")
models[model_name]
```

```
on traning data:
6/6 [==============================] - 0s 6ms/step - loss: 0.0117 - accuracy: 1.0000
on testing data:
2/2 [==============================] - 0s 8ms/step - loss: 0.3495 - accuracy: 0.9048
data:
```

Out[ ]:
```
{'train': 100.0, 'test': 90.4762}
```

5. Drop out

In [ ]:
```
def create_drop_out():
        # create model
  model = Sequential()
  model.add(Dense(60, input_shape=(60,), activation='relu'))
  model.add(Dense(40, activation='relu'))
  model.add(keras.layers.Dropout(0.2))
  model.add(Dense(30, activation='relu'))
  model.add(Dense(50, activation='relu'))
  model.add(Dense(1, activation='sigmoid'))
  # Compile model
```

```
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

In [ ]:
```
model5 = create_drop_out()
model5.fit(X_train,y_train,epochs=100)
```

In [ ]:
```
model_name = "Drop out"
models[model_name] = {}

model = model5

print("on traning data: ")
train = model.evaluate(X_train,y_train)
models[model_name]["train"] = round(train[1]*100,4)

print("on testing data: ")
test = model.evaluate(X_test,y_test)
models[model_name]["test"] = round(test[1]*100, 4)

print("data: ")
models[model_name]
```

```
on traning data:
6/6 [==============================] - 0s 9ms/step - loss: 0.0083 - accuracy: 1.0000
on testing data:
2/2 [==============================] - 0s 11ms/step - loss: 0.6086 - accuracy: 0.8571
data:
```

Out[ ]:
```
{'train': 100.0, 'test': 85.7143}
```

In [ ]:
```
df = pd.DataFrame(models)
df
```

Out[ ]:

	base	l1	l2	early stopping	Drop out
train	100.0000	98.1928	100.0000	100.0000	100.0000
test	85.7143	80.9524	83.3333	90.4762	85.7143

In [ ]: