Lab Report 3

Gradient Descent

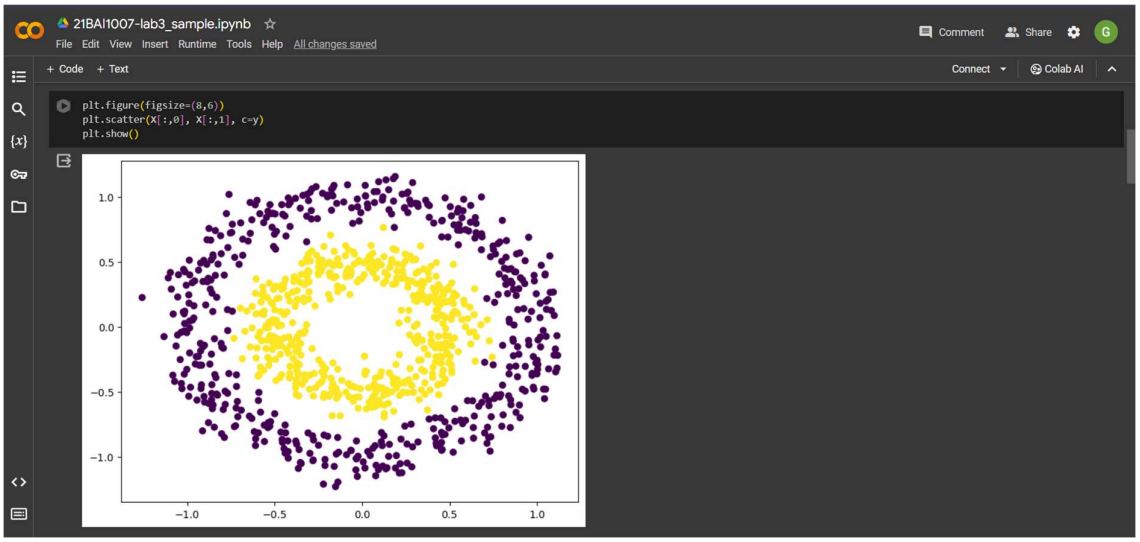
Goutham Krishnan

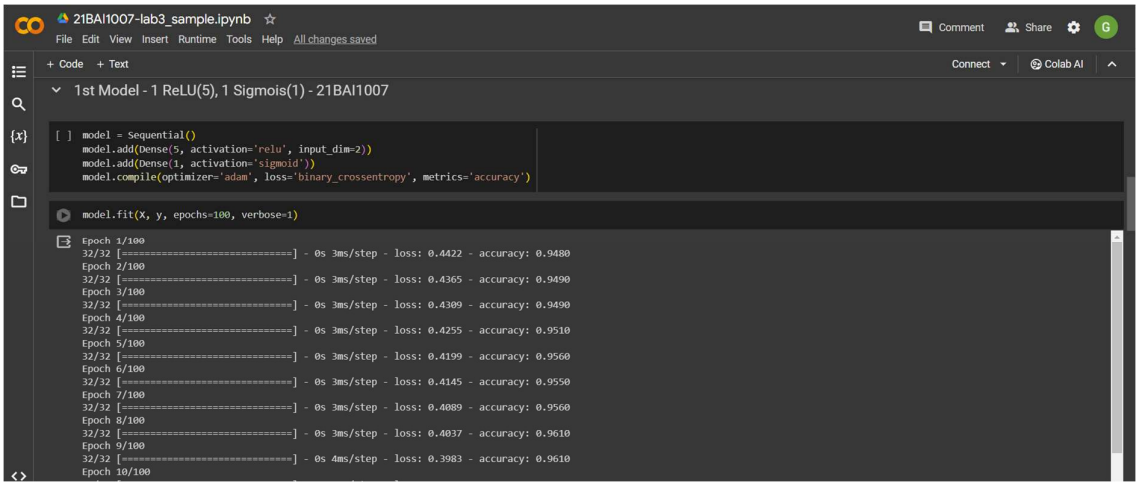
21BAI1007

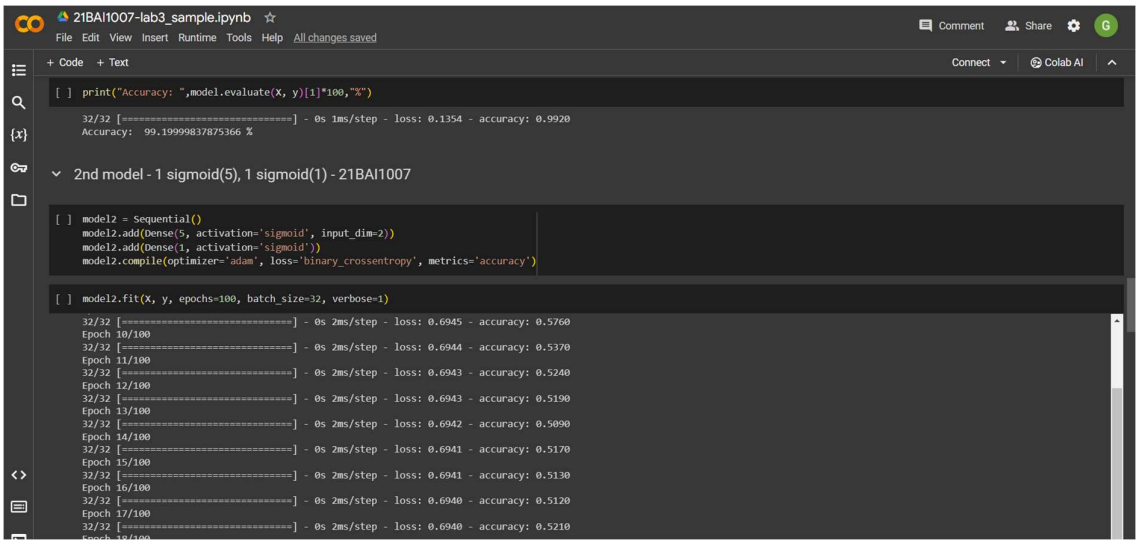
Aim

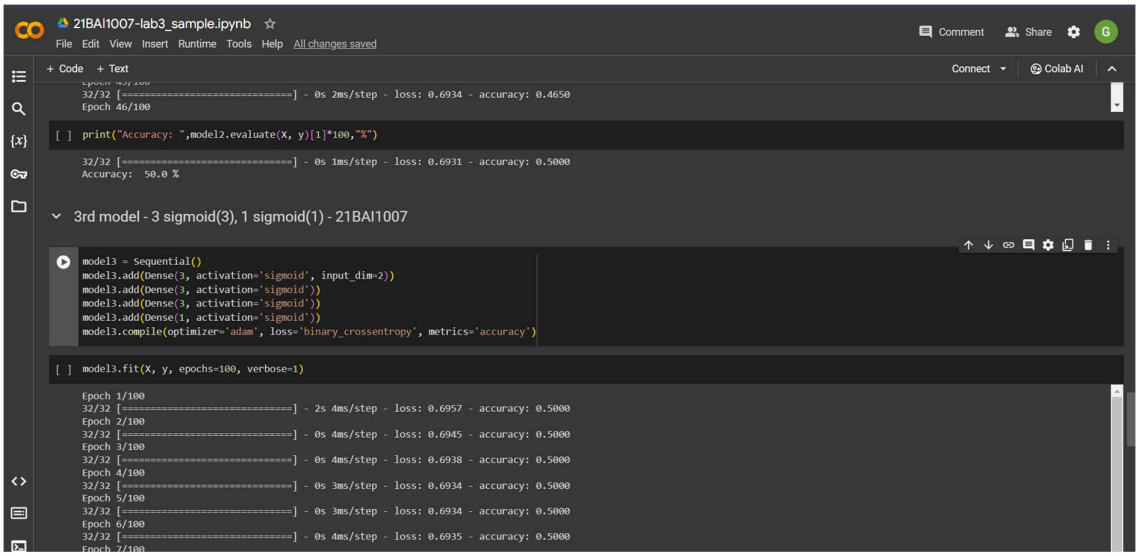
1. Implement 3 different neural networks as listed below
    - 2 layers. 1st layer : ReLU (5 perceptrons), 2nd Layer : Sigmoid (1 perceptron)
    - In the above layers, both sigmoid
    - 4 layers with first three layers : Sigmoid (3) and the last layer : sigmoid (1)
2. Change the activation functions and implement the code given.
3. Implement a code to compare GD, SGD, Minibatch SGD for any input/neural model of your choice.

Q1. Implementing 3 different neural networks (sigmoid, ReLU)




21BAI1007-lab3_sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share G
+ Code + Text
Connect Colab AI
1st Model - 1 ReLU(5), 1 Sigmois(1) - 21BAI1007
model = Sequential()
model.add(Dense(5, activation='relu', input_dim=2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
model.fit(X, y, epochs=100, verbose=1)
Epoch 1/100
32/32 [==========] - 0s 3ms/step - loss: 0.4422 - accuracy: 0.9480
Epoch 2/100
32/32 [==========] - 0s 3ms/step - loss: 0.4365 - accuracy: 0.9490
Epoch 3/100
32/32 [==========] - 0s 3ms/step - loss: 0.4309 - accuracy: 0.9490
Epoch 4/100
32/32 [==========] - 0s 3ms/step - loss: 0.4255 - accuracy: 0.9510
Epoch 5/100
32/32 [==========] - 0s 3ms/step - loss: 0.4199 - accuracy: 0.9560
Epoch 6/100
32/32 [==========] - 0s 3ms/step - loss: 0.4145 - accuracy: 0.9550
Epoch 7/100
32/32 [==========] - 0s 3ms/step - loss: 0.4089 - accuracy: 0.9560
Epoch 8/100
32/32 [==========] - 0s 3ms/step - loss: 0.4037 - accuracy: 0.9610
Epoch 9/100
32/32 [==========] - 0s 4ms/step - loss: 0.3983 - accuracy: 0.9610
Epoch 10/100


21BAI1007-lab3_sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share G
+ Code + Text
Connect Colab AI
print("Accuracy: ",model.evaluate(X, y)[1]*100,"%")
32/32 [==========] - 0s 1ms/step - loss: 0.1354 - accuracy: 0.9920
Accuracy: 99.19999837875366 %
2nd model - 1 sigmoid(5), 1 sigmoid(1) - 21BAI1007
model2 = Sequential()
model2.add(Dense(5, activation='sigmoid', input_dim=2))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
model2.fit(X, y, epochs=100, batch_size=32, verbose=1)
32/32 [==========] - 0s 2ms/step - loss: 0.6945 - accuracy: 0.5760
Epoch 10/100
32/32 [==========] - 0s 2ms/step - loss: 0.6944 - accuracy: 0.5370
Epoch 11/100
32/32 [==========] - 0s 2ms/step - loss: 0.6943 - accuracy: 0.5240
Epoch 12/100
32/32 [==========] - 0s 2ms/step - loss: 0.6943 - accuracy: 0.5190
Epoch 13/100
32/32 [==========] - 0s 2ms/step - loss: 0.6942 - accuracy: 0.5090
Epoch 14/100
32/32 [==========] - 0s 2ms/step - loss: 0.6941 - accuracy: 0.5170
Epoch 15/100
32/32 [==========] - 0s 2ms/step - loss: 0.6941 - accuracy: 0.5130
Epoch 16/100
32/32 [==========] - 0s 2ms/step - loss: 0.6940 - accuracy: 0.5120
Epoch 17/100
32/32 [==========] - 0s 2ms/step - loss: 0.6940 - accuracy: 0.5210


21BAI1007-lab3_sample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share G
+ Code + Text
Connect Colab AI
32/32 [==========] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.4650
Epoch 46/100
print("Accuracy: ",model2.evaluate(X, y)[1]*100,"%")
32/32 [==========] - 0s 1ms/step - loss: 0.6931 - accuracy: 0.5000
Accuracy: 50.0 %
3rd model - 3 sigmoid(3), 1 sigmoid(1) - 21BAI1007
model3 = Sequential()
model3.add(Dense(3, activation='sigmoid', input_dim=2))
model3.add(Dense(3, activation='sigmoid'))
model3.add(Dense(3, activation='sigmoid'))
model3.add(Dense(1, activation='sigmoid'))
model3.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
model3.fit(X, y, epochs=100, verbose=1)
Epoch 1/100
32/32 [==========] - 2s 4ms/step - loss: 0.6957 - accuracy: 0.5000
Epoch 2/100
32/32 [==========] - 0s 4ms/step - loss: 0.6945 - accuracy: 0.5000
Epoch 3/100
32/32 [==========] - 0s 4ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 4/100
32/32 [==========] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000
Epoch 5/100
32/32 [==========] - 0s 3ms/step - loss: 0.6934 - accuracy: 0.5000
Epoch 6/100
32/32 [==========] - 0s 4ms/step - loss: 0.6935 - accuracy: 0.5000

21BAI1007-lab3_sample.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Colab AI

```
32/32 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.4960
Epoch 24/100
32/32 [==============================] - 0s 3ms/step - loss: 0.6932 - accuracy: 0.4970
Epoch 25/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.4820
Epoch 26/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.4730
Epoch 27/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6934 - accuracy: 0.4960
Epoch 28/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5000
Epoch 29/100
32/32 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.4920
```

```
print("Accuracy: ",model3.evaluate(X, y)[1]*100,"%")
```

```
32/32 [==============================] - 1s 1ms/step - loss: 0.6900 - accuracy: 0.6050
Accuracy:  60.50000190734863 %
```

```
Start coding or generate with AI.
```

Q1 extension. Implementing 3 different neural networks (sigmoid, ReLU) for the diabetes.csv Pima dataset

21BAI1007-diabetes-Dataset.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Colab AI

Implement 3 different neural networks as given below on the PIMA indian diabetes dataset - 21BAI1007

(i) 2 layers - ReLU(5), Sigmoid(1) (ii) 2 layers - Sigmoid(5), Sigmoid(1) (iii) 4 layers - Sigmoid(3), Sigmoid(3), Sigmoid(3), Sigmoid(1)

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.callbacks import Callback
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import RandomNormal
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv('diabetes.csv')
```

```
len(df)
```

```
768
```

21BAI1007-diabetes-Dataset.ipynb

File Edit View Insert Runtime Tools Help All changes saved

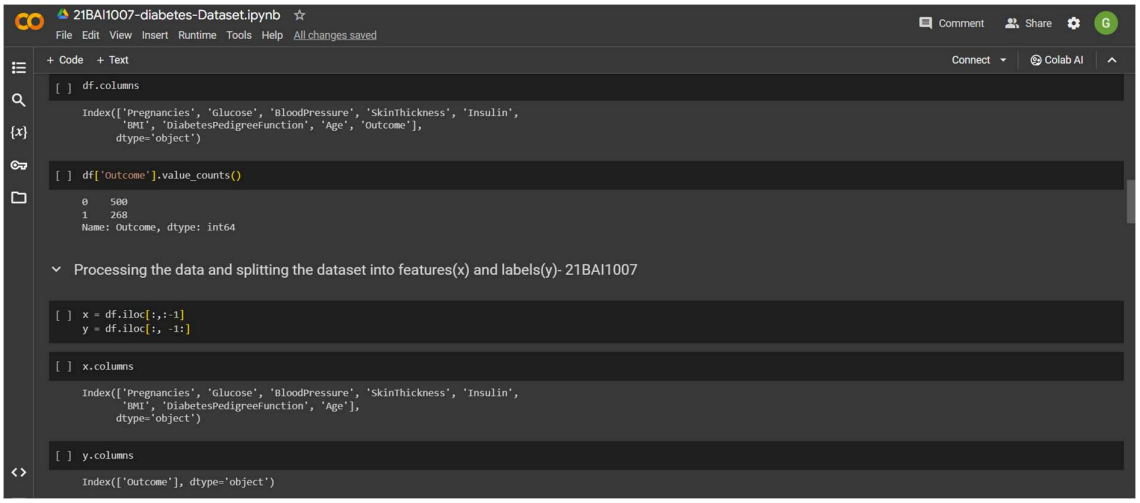
Comment Share

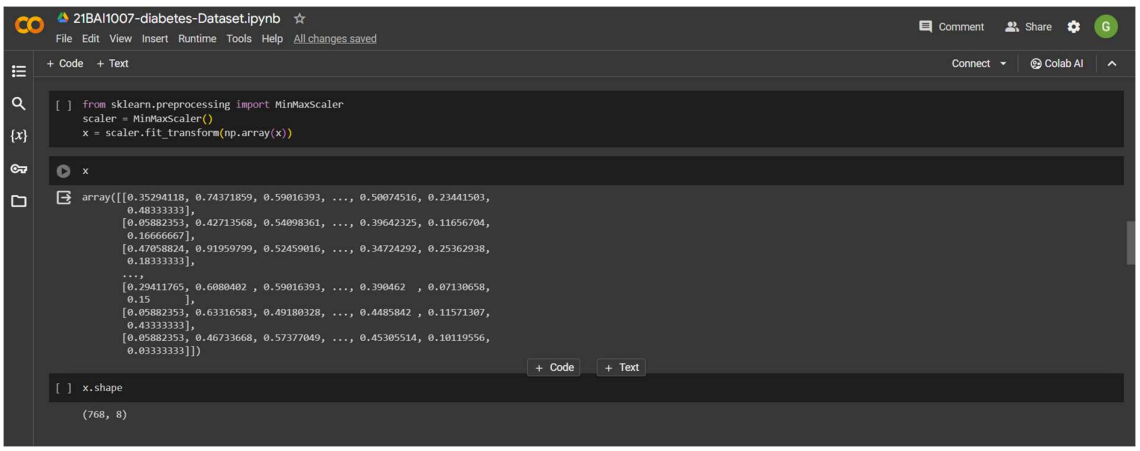
+ Code + Text

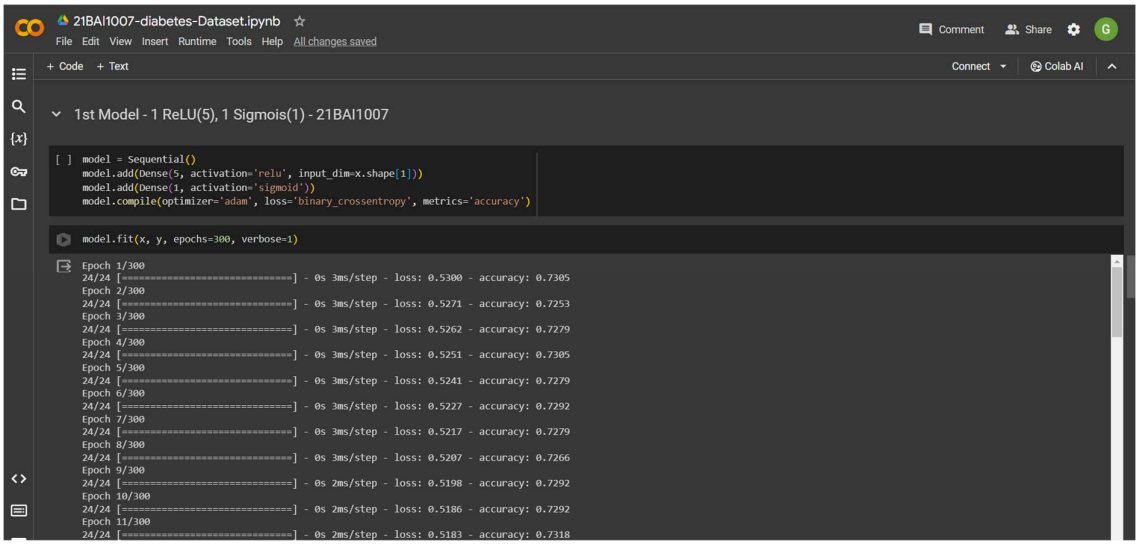
Connect Colab AI

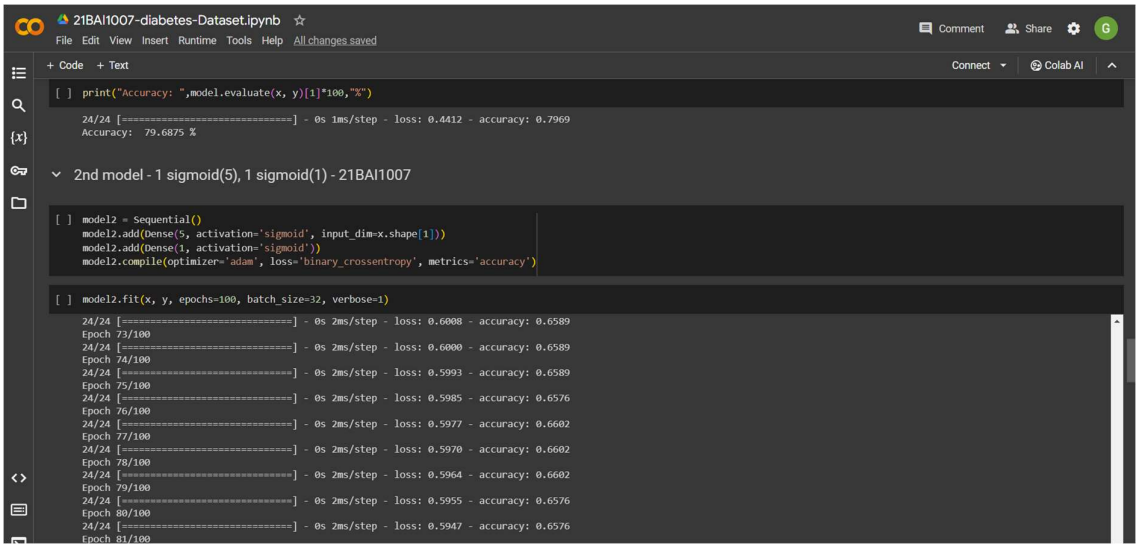
```
df.head(30)
```

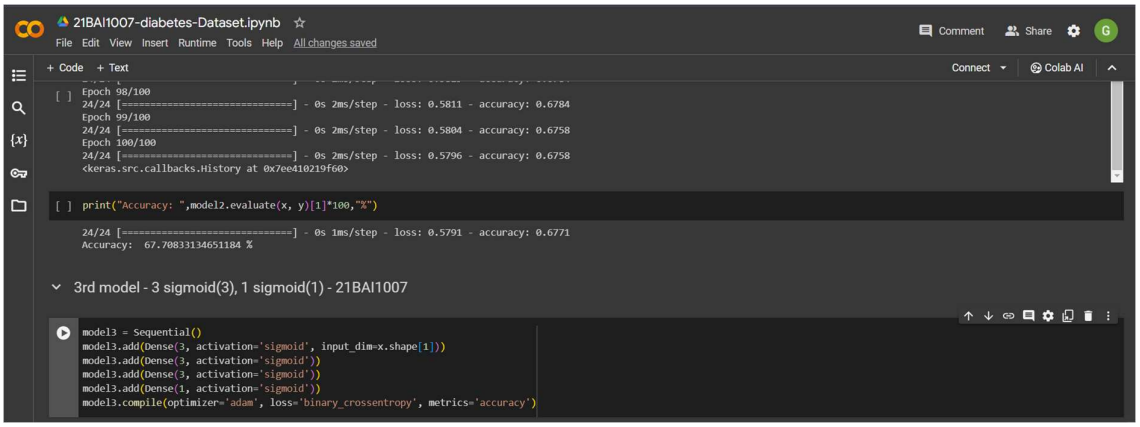
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1

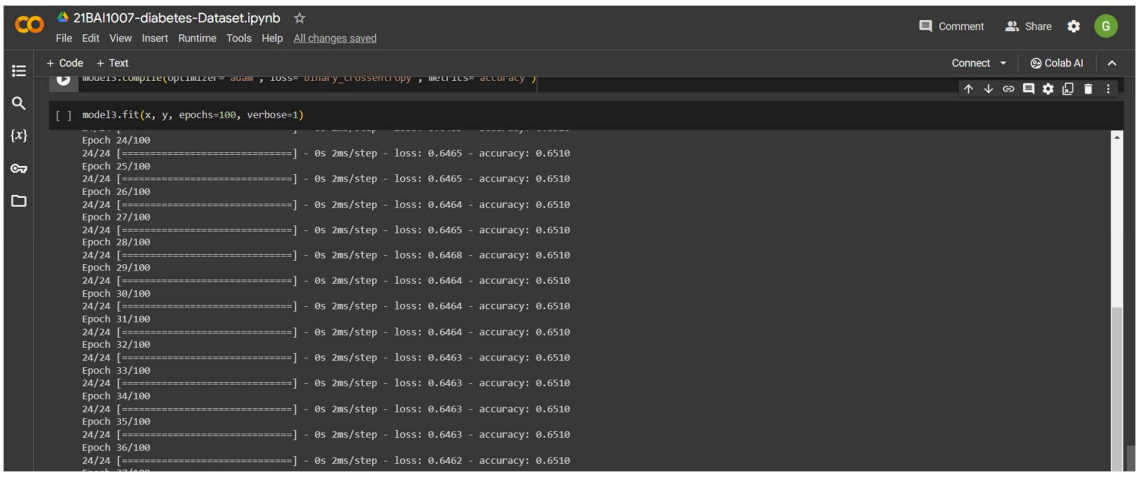

21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
[ ] df.columns
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
dtype='object')
[ ] df['Outcome'].value_counts()
0 500
1 268
Name: Outcome, dtype: int64
Processing the data and splitting the dataset into features(x) and labels(y)- 21BAI1007
[ ] x = df.iloc[:,:-1]
y = df.iloc[:, -1:]
[ ] x.columns
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI', 'DiabetesPedigreeFunction', 'Age'],
dtype='object')
[ ] y.columns
Index(['Outcome'], dtype='object')


21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
[ ] from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x = scaler.fit_transform(np.array(x))
x
array([[0.35294118, 0.74371859, 0.59016393, ..., 0.50074516, 0.23441503,
0.48333333],
[0.05882353, 0.42713568, 0.54098361, ..., 0.39642325, 0.11656704,
0.16666667],
[0.47058824, 0.91959799, 0.52459016, ..., 0.34724292, 0.25362938,
0.18333333],
...,
[0.29411765, 0.6080402 , 0.59016393, ..., 0.390462 , 0.07130658,
0.15],
[0.05882353, 0.63316583, 0.49180328, ..., 0.4485842 , 0.11571307,
0.43333333],
[0.05882353, 0.46733668, 0.57377049, ..., 0.45305514, 0.10119556,
0.03333333]])
+ Code + Text
[ ] x.shape
(768, 8)


21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
1st Model - 1 ReLU(5), 1 Sigmois(1) - 21BAI1007
[ ] model = Sequential()
model.add(Dense(5, activation='relu', input_dim=x.shape[1]))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
model.fit(x, y, epochs=300, verbose=1)
Epoch 1/300
24/24 [==========] - 0s 3ms/step - loss: 0.5300 - accuracy: 0.7305
Epoch 2/300
24/24 [==========] - 0s 3ms/step - loss: 0.5271 - accuracy: 0.7253
Epoch 3/300
24/24 [==========] - 0s 3ms/step - loss: 0.5262 - accuracy: 0.7279
Epoch 4/300
24/24 [==========] - 0s 3ms/step - loss: 0.5251 - accuracy: 0.7305
Epoch 5/300
24/24 [==========] - 0s 3ms/step - loss: 0.5241 - accuracy: 0.7279
Epoch 6/300
24/24 [==========] - 0s 3ms/step - loss: 0.5227 - accuracy: 0.7292
Epoch 7/300
24/24 [==========] - 0s 3ms/step - loss: 0.5217 - accuracy: 0.7279
Epoch 8/300
24/24 [==========] - 0s 3ms/step - loss: 0.5207 - accuracy: 0.7266
Epoch 9/300
24/24 [==========] - 0s 2ms/step - loss: 0.5198 - accuracy: 0.7292
Epoch 10/300
24/24 [==========] - 0s 2ms/step - loss: 0.5186 - accuracy: 0.7292
Epoch 11/300
24/24 [==========] - 0s 2ms/step - loss: 0.5183 - accuracy: 0.7318


21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
[ ] print("Accuracy: ",model.evaluate(x, y)[1]*100,"%")
24/24 [==========] - 0s 1ms/step - loss: 0.4412 - accuracy: 0.7969
Accuracy: 79.6875 %
2nd model - 1 sigmoid(5), 1 sigmoid(1) - 21BAI1007
[ ] model2 = Sequential()
model2.add(Dense(5, activation='sigmoid', input_dim=x.shape[1]))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')
[ ] model2.fit(x, y, epochs=100, batch_size=32, verbose=1)
24/24 [==========] - 0s 2ms/step - loss: 0.6008 - accuracy: 0.6589
Epoch 73/100
24/24 [==========] - 0s 2ms/step - loss: 0.6000 - accuracy: 0.6589
Epoch 74/100
24/24 [==========] - 0s 2ms/step - loss: 0.5993 - accuracy: 0.6589
Epoch 75/100
24/24 [==========] - 0s 2ms/step - loss: 0.5985 - accuracy: 0.6576
Epoch 76/100
24/24 [==========] - 0s 2ms/step - loss: 0.5977 - accuracy: 0.6602
Epoch 77/100
24/24 [==========] - 0s 2ms/step - loss: 0.5970 - accuracy: 0.6602
Epoch 78/100
24/24 [==========] - 0s 2ms/step - loss: 0.5964 - accuracy: 0.6602
Epoch 79/100
24/24 [==========] - 0s 2ms/step - loss: 0.5955 - accuracy: 0.6576
Epoch 80/100
24/24 [==========] - 0s 2ms/step - loss: 0.5947 - accuracy: 0.6576
Epoch 81/100


21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
Epoch 98/100
24/24 [==========] - 0s 2ms/step - loss: 0.5811 - accuracy: 0.6784
Epoch 99/100
24/24 [==========] - 0s 2ms/step - loss: 0.5804 - accuracy: 0.6758
Epoch 100/100
24/24 [==========] - 0s 2ms/step - loss: 0.5796 - accuracy: 0.6758
<keras.src.callbacks.History at 0x7ee410219f60>
[ ] print("Accuracy: ",model2.evaluate(x, y)[1]*100,"%")
24/24 [==========] - 0s 1ms/step - loss: 0.5791 - accuracy: 0.6771
Accuracy: 67.70833134651184 %
3rd model - 3 sigmoid(3), 1 sigmoid(1) - 21BAI1007
model3 = Sequential()
model3.add(Dense(3, activation='sigmoid', input_dim=x.shape[1]))
model3.add(Dense(3, activation='sigmoid'))
model3.add(Dense(3, activation='sigmoid'))
model3.add(Dense(1, activation='sigmoid'))
model3.compile(optimizer='adam', loss='binary_crossentropy', metrics='accuracy')


21BAI1007-diabetes-Dataset.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Colab AI
[ ] model3.fit(x, y, epochs=100, verbose=1)
Epoch 24/100
24/24 [==========] - 0s 2ms/step - loss: 0.6465 - accuracy: 0.6510
Epoch 25/100
24/24 [==========] - 0s 2ms/step - loss: 0.6465 - accuracy: 0.6510
Epoch 26/100
24/24 [==========] - 0s 2ms/step - loss: 0.6464 - accuracy: 0.6510
Epoch 27/100
24/24 [==========] - 0s 2ms/step - loss: 0.6465 - accuracy: 0.6510
Epoch 28/100
24/24 [==========] - 0s 2ms/step - loss: 0.6468 - accuracy: 0.6510
Epoch 29/100
24/24 [==========] - 0s 2ms/step - loss: 0.6464 - accuracy: 0.6510
Epoch 30/100
24/24 [==========] - 0s 2ms/step - loss: 0.6464 - accuracy: 0.6510
Epoch 31/100
24/24 [==========] - 0s 2ms/step - loss: 0.6464 - accuracy: 0.6510
Epoch 32/100
24/24 [==========] - 0s 2ms/step - loss: 0.6463 - accuracy: 0.6510
Epoch 33/100
24/24 [==========] - 0s 2ms/step - loss: 0.6463 - accuracy: 0.6510
Epoch 34/100
24/24 [==========] - 0s 2ms/step - loss: 0.6463 - accuracy: 0.6510
Epoch 35/100
24/24 [==========] - 0s 2ms/step - loss: 0.6463 - accuracy: 0.6510
Epoch 36/100
24/24 [==========] - 0s 2ms/step - loss: 0.6462 - accuracy: 0.6510

21BAI1007-diabetes-Dataset.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect Colab AI

```
Epoch 41/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6461 - accuracy: 0.6510
Epoch 42/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6461 - accuracy: 0.6510
Epoch 43/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6461 - accuracy: 0.6510
Epoch 44/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6461 - accuracy: 0.6510
Epoch 45/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6461 - accuracy: 0.6510
Epoch 46/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6459 - accuracy: 0.6510
Epoch 47/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6460 - accuracy: 0.6510
Epoch 48/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6458 - accuracy: 0.6510
Epoch 49/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6458 - accuracy: 0.6510
Epoch 50/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6458 - accuracy: 0.6510
Epoch 51/100
24/24 [==============================] - 0s 2ms/step - loss: 0.6457 - accuracy: 0.6510
```

```
print("Accuracy: ",model3.evaluate(x, y)[1]*100,"%")
```

```
24/24 [==============================] - 0s 1ms/step - loss: 0.6303 - accuracy: 0.6510
Accuracy:  65.10416865348816 %
```

Q2. Visualizing gradient descent for tanh and ReLU activation fucntions

21BAI1007-GradientDescent.ipynb

File Edit View Insert Runtime Tools Help Last edited on 4 February

Comment Share

+ Code + Text

Connect Colab AI

Visualizing gradient descent by building a model using tanh and ReLU activation functions

21BAI1007

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.callbacks import Callback
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.initializers import RandomNormal
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
```

```
from sklearn.datasets import make_circles

tf.random.set_seed(42)
np.random.seed(42)

X, y = make_circles(n_samples=1000, factor=0.5, noise=0.1)
```

```
plt.figure(figsize=(8,6))
plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
```

21BAI1007-GradientDescent.ipynb

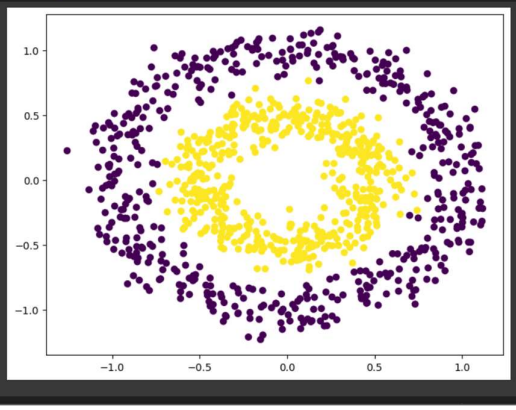
File Edit View Insert Runtime Tools Help Last edited on 4 February

Comment Share

+ Code + Text

Connect Colab AI

```
plt.figure(figsize=(8,6))
plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
```



21BAI1007-GradientDescent.ipynb

File Edit View Insert Runtime Tools Help Last edited on 4 February

Comment Share G

+ Code + Text

Connect Colab AI

```
# Illustrate weights across epochs
class WeightCapture(Callback):
    "Capture the weights of each layer of the model"
    def __init__(self, model):
        super().__init__()
        self.model = model
        self.weights = []
        self.epochs = []

    def on_epoch_end(self, epoch, logs=None):
        self.epochs.append(epoch) # remember the epoch axis
        weight = {}
        for layer in model.layers:
            if not layer.weights:
                continue
            name = layer.weights[0].name.split("/")[0]
            weight[name] = layer.weights[0].numpy()
        self.weights.append(weight)

def make_mlp(activation, initializer, name):
    "Create a model with specified activation and initalizer"
    model = Sequential([
        Input(shape=(2,), name=name+"0"),
        Dense(5, activation=activation, kernel_initializer=initializer, name=name+"1"),
        Dense(5, activation=activation, kernel_initializer=initializer, name=name+"2"),
        Dense(5, activation=activation, kernel_initializer=initializer, name=name+"3"),
        Dense(5, activation=activation, kernel_initializer=initializer, name=name+"4"),
        Dense(1, activation="sigmoid", kernel_initializer=initializer, name=name+"5")
    ])
    return model
```

21BAI1007-GradientDescent.ipynb

File Edit View Insert Runtime Tools Help Last edited on 4 February

Comment Share G

+ Code + Text

Connect Colab AI

```
def plotweight(capture_cb):
    "Plot the weights' mean and s.d. across epochs"
    fig, ax = plt.subplots(2, 1, sharex=True, constrained_layout=True, figsize=(8, 10))
    ax[0].set_title("Mean weight")
    for key in capture_cb.weights[0]:
        ax[0].plot(capture_cb.epochs, [w[key].mean() for w in capture_cb.weights], label=key)
    ax[0].legend()
    ax[1].set_title("S.D.")
    for key in capture_cb.weights[0]:
        ax[1].plot(capture_cb.epochs, [w[key].std() for w in capture_cb.weights], label=key)
    ax[1].legend()
    plt.show()
```

```
initializer = RandomNormal(mean=0, stddev=1)
batch_size = 32
n_epochs = 100
```

Model using tanh - 21BAI1007

```
# Sigmoid activation
model = make_mlp("tanh", initializer, "tanh")
capture_cb = WeightCapture(model)
capture_cb.on_epoch_end(-1)
model.compile(optimizer="rmsprop", loss="binary_crossentropy", metrics=["acc"])
print("Before training: Accuracy", accuracy_score(y, (model(X).numpy() > 0.5).astype(int)))
model.fit(X, y, batch_size=batch_size, epochs=n_epochs, callbacks=[capture_cb], verbose=0)
print("After training: Accuracy", accuracy_score(y, (model(X).numpy() > 0.5).astype(int)))
print(model.evaluate(X,y))
plotweight(capture_cb)
```

21BAI1007-GradientDescent.ipynb

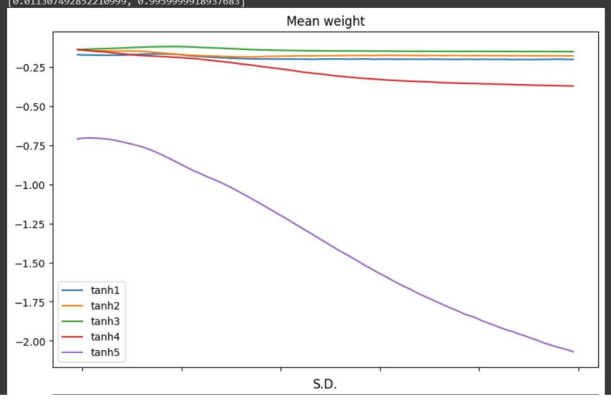
File Edit View Insert Runtime Tools Help Last edited on 4 February

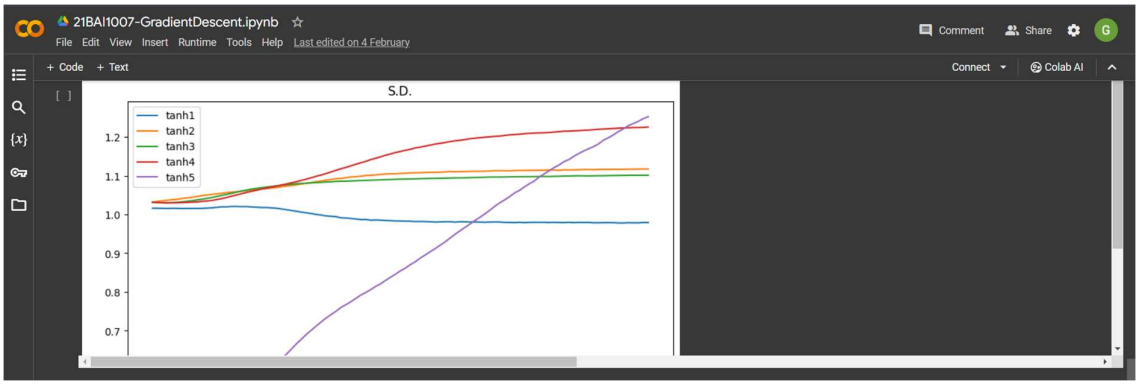
Comment Share G

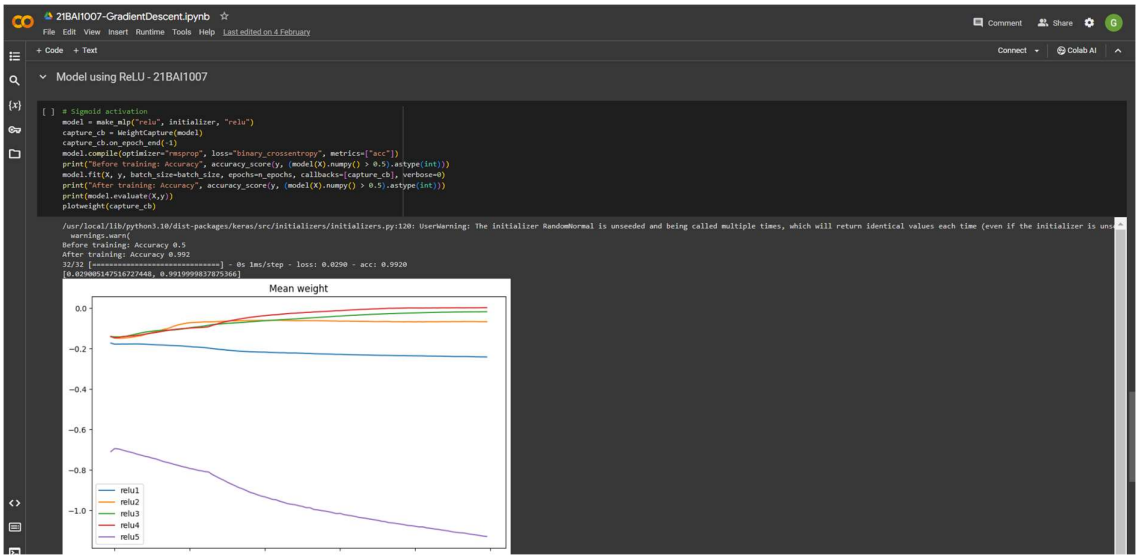
+ Code + Text

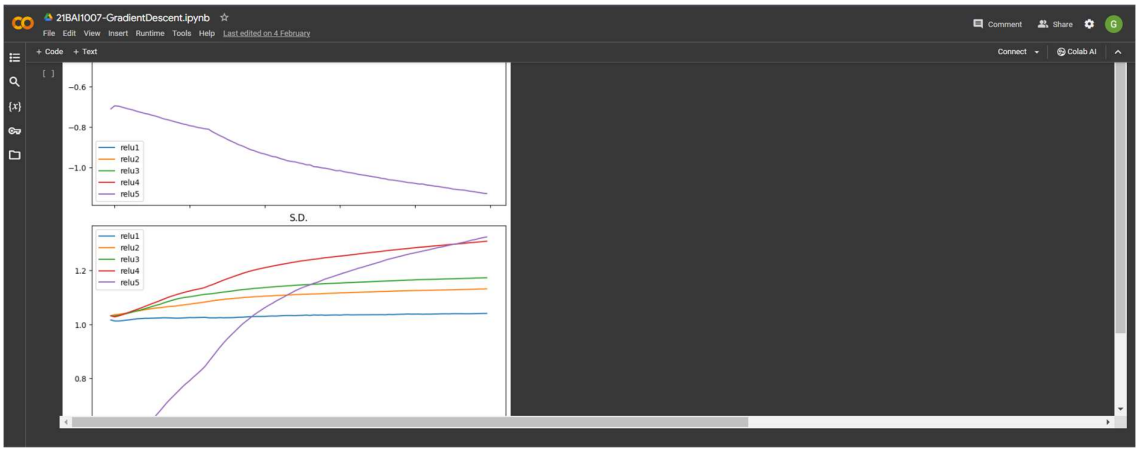
Connect Colab AI

```
/usr/local/lib/python3.10/dist-packages/keras/src/initializers/initializers.py:120: UserWarning: The initializer RandomNormal is unseeded and being called multiple times, which will
  warnings.warn(
Before training: Accuracy 0.508
After training: Accuracy 0.996
32/32 [==============================] - 0s 957us/step - loss: 0.0113 - acc: 0.9960
[0.011307492852210999, 0.9959999918937683]
```

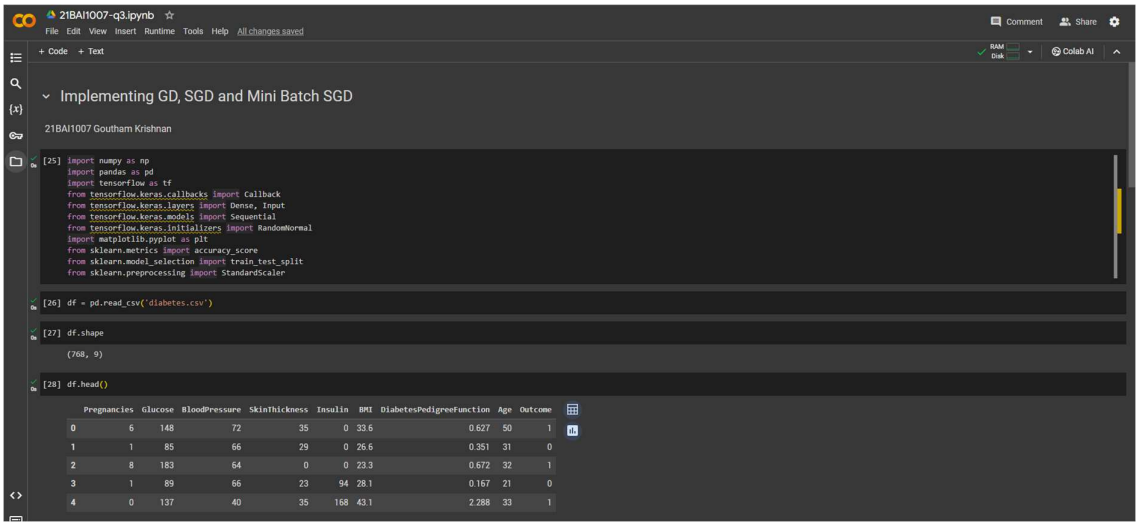



21BAI1007-GradientDescent.ipynb
File Edit View Insert Runtime Tools Help Last edited on 4 February
Comment
Share
+ Code + Text
Connect
Colab AI
S.D.
tanh1
tanh2
tanh3
tanh4
tanh5


21BAI1007-GradientDescent.ipynb
Model using ReLU - 21BAI1007
# Sigmoid activation
model = make_mlp("relu", initializer, "relu")
capture_cb = WeightCapture(model)
capture_cb.on_epoch_end(-1)
model.compile(optimizer="rmsprop", loss="binary_crossentropy", metrics=["acc"])
print("Before training: Accuracy", accuracy_score(y, (model(X).numpy() > 0.5).astype(int)))
model.fit(X, y, batch_size=batch_size, epochs=n_epochs, callbacks=[capture_cb], verbose=0)
print("After training: Accuracy", accuracy_score(y, (model(X).numpy() > 0.5).astype(int)))
print(model.evaluate(X,y))
plotweight(capture_cb)
/usr/local/lib/python3.10/dist-packages/keras/src/initializers/initializers.py:120: UserWarning: The initializer RandomNormal is unseeded and being called multiple times, which will return identical values each time (even if the initializer is uns
warnings.warn(
Before training: Accuracy 0.5
After training: Accuracy 0.992
32/32 [==============================] - 0s 1ms/step - loss: 0.0290 - acc: 0.9920
[0.029005147516727448, 0.9919999837875366]
Mean weight
relu1
relu2
relu3
relu4
relu5


21BAI1007-GradientDescent.ipynb
File Edit View Insert Runtime Tools Help Last edited on 4 February
Comment
Share
+ Code + Text
Connect
Colab AI
relu1
relu2
relu3
relu4
relu5
S.D.
relu1
relu2
relu3
relu4
relu5

Q3. Implementing Gradient Descent, Stochastic gradient descent and mini batch SGD



21BAI1007-q3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

Implementing GD, SGD and Mini Batch SGD

21BAI1007 Goutham Krishnan

```
[25] import numpy as np
     import pandas as pd
     import tensorflow as tf
     from tensorflow.keras.callbacks import Callback
     from tensorflow.keras.layers import Dense, Input
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.initializers import RandomNormal
     import matplotlib.pyplot as plt
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
```

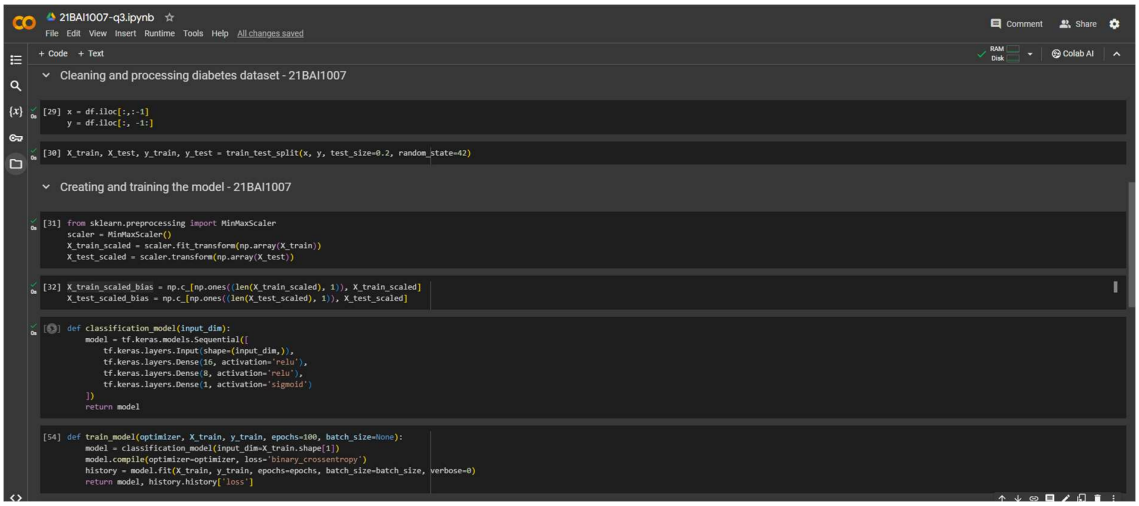
```
[26] df = pd.read_csv('diabetes.csv')
```

```
[27] df.shape
```

```
(768, 9)
```

```
[28] df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



21BAI1007-q3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

Cleaning and processing diabetes dataset - 21BAI1007

```
[29] x = df.iloc[:,:-1]
     y = df.iloc[:, -1:]
```

```
[30] X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

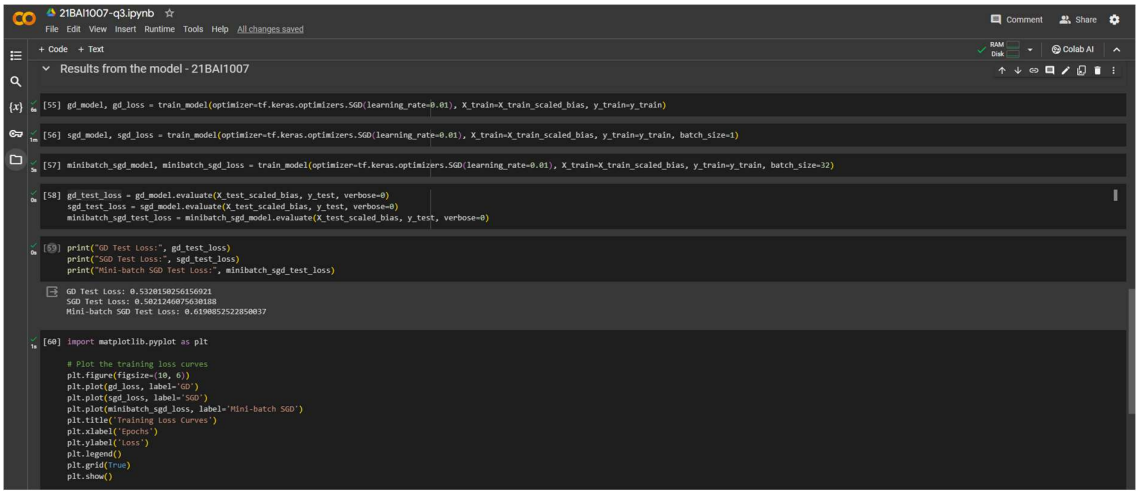
Creating and training the model - 21BAI1007

```
[31] from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     X_train_scaled = scaler.fit_transform(np.array(X_train))
     X_test_scaled = scaler.transform(np.array(X_test))
```

```
[32] X_train_scaled_bias = np.c_[np.ones((len(X_train_scaled), 1)), X_train_scaled]
     X_test_scaled_bias = np.c_[np.ones((len(X_test_scaled), 1)), X_test_scaled]
```

```
[ ] def classification_model(input_dim):
        model = tf.keras.models.Sequential([
            tf.keras.layers.Input(shape=(input_dim,)),
            tf.keras.layers.Dense(16, activation='relu'),
            tf.keras.layers.Dense(8, activation='relu'),
            tf.keras.layers.Dense(1, activation='sigmoid')
        ])
        return model
```

```
[54] def train_model(optimizer, X_train, y_train, epochs=100, batch_size=None):
         model = classification_model(input_dim=X_train.shape[1])
         model.compile(optimizer=optimizer, loss='binary_crossentropy')
         history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=0)
         return model, history.history['loss']
```



21BAI1007-q3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

Results from the model - 21BAI1007

```
[55] gd_model, gd_loss = train_model(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01), X_train=X_train_scaled_bias, y_train=y_train)
```

```
[56] sgd_model, sgd_loss = train_model(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01), X_train=X_train_scaled_bias, y_train=y_train, batch_size=1)
```

```
[57] minibatch_sgd_model, minibatch_sgd_loss = train_model(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01), X_train=X_train_scaled_bias, y_train=y_train, batch_size=32)
```

```
[58] gd_test_loss = gd_model.evaluate(X_test_scaled_bias, y_test, verbose=0)
     sgd_test_loss = sgd_model.evaluate(X_test_scaled_bias, y_test, verbose=0)
     minibatch_sgd_test_loss = minibatch_sgd_model.evaluate(X_test_scaled_bias, y_test, verbose=0)
```

```
[59] print("GD Test Loss:", gd_test_loss)
     print("SGD Test Loss:", sgd_test_loss)
     print("Mini-batch SGD Test Loss:", minibatch_sgd_test_loss)
```

```
GD Test Loss: 0.5320150256156921
SGD Test Loss: 0.5021246075630188
Mini-batch SGD Test Loss: 0.6190852522850037
```

```
[60] import matplotlib.pyplot as plt

     # Plot the training loss curves
     plt.figure(figsize=(10, 6))
     plt.plot(gd_loss, label='GD')
     plt.plot(sgd_loss, label='SGD')
     plt.plot(minibatch_sgd_loss, label='Mini-batch SGD')
     plt.title('Training Loss Curves')
     plt.xlabel('Epochs')
     plt.ylabel('Loss')
     plt.legend()
     plt.grid(True)
     plt.show()
```

21BAI1007-q3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```