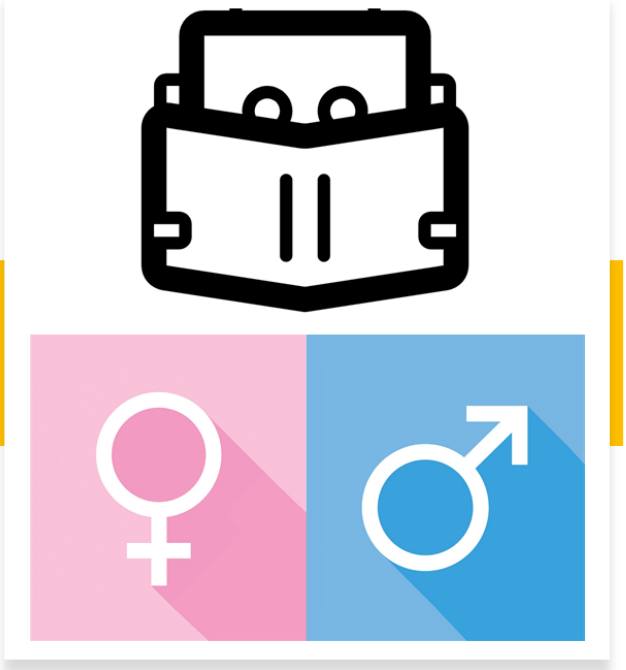

Gender Classification Model with Deep Learning

Submission by

Matan Ben Nagar & Yaara Bark

[The entire project can be found on our Github repository: shorturl.at/lprzY]

Added to the submission are two files:

1. **Logistic Regression.ipynb**
2. **MLP.ipynb**

You can press “Run all” and get the results that we got here

Handling Dataset

In this assignment, my partner and I have tried to establish a connection between different physical attributes of a person, to predict his sex. We were using the Cardiovascular Disease dataset from Kaggle (provided in this link: https://www.kaggle.com/sulianova/cardiovascular-disease-dataset).

Dataset:

- Our dataset contains 70,000 examples
- At first we took the only columns that were relevant for our research such as: [age, gender, height, weight, smoke]

First steps we implemented:

1. Data cleaning:

   - Removing unnecessary features that supposedly don't have impact or relation to what we were trying to predict (such as cholesterol, gluc and so on…)

   - Detect duplicates rows in our dataset and remove them

2. Data preprocessing:

   - Detecting Null values, and if such exist remove the entire row

   - Standarization: Converting column data so that it ranges around the same values as other columns. Bring the values close together.

3. Normalization: Making sure all data in columns if of the same type (integer)

```
(             age  gender  height  weight  smoke
 id
 0      51.091667       1     168    62.0      0
 1      56.188889       0     156    85.0      0
 2      52.380556       0     165    64.0      0
 3      48.952778       1     169    82.0      0
 4      48.538889       0     156    56.0      0
 ...          ...     ...     ...     ...    ...
 99993  53.444444       1     168    76.0      1
 99995  62.780556       0     158   126.0      0
 99996  52.961111       1     183   105.0      0
 99998  62.308333       0     163    72.0      0
 99999  57.055556       0     170    72.0      0

 [70000 rows x 5 columns],
 0    45530
 1    24470
 Name: gender, dtype: int64)
```

Logistic Regression

Gender classification is a classification problem, because of that we chose Logistic Regression as our predictive algorithm.

Understanding the data

Our first challenge was to realize how our data dimensions are supposed to look like. We also struggeled to under what the tf.placeholder, tf.Variable functions do. After we gained some deeper understandig it became clear.

X_data **Y_data**

```
[ 52.525     , 176.        ,  72.        ,   0.        ] ──────────────────────► [1]
[ 64.66111111, 145.        ,  68.        ,   0.        ] ──────────────────────► [0]
[ 58.8       , 159.        ,  66.        ,   0.        ] ──────────────────────► [1]
...,                                                                               ...
[ 52.96111111, 183.        , 105.        ,   0.        ] ──────────────────────► [1]
[ 62.30833333, 163.        ,  72.        ,   0.        ] ──────────────────────► [0]
[ 57.05555556, 170.        ,  72.        ,   0.        ] ──────────────────────► [0]
```

We then used pandas to split our database into the feature data and label data.

Loss and Updating

For the loss function we used the cross entropy function which measures **the probability error in discrete classification tasks in which each class is independent and not mutually exclusive**. Our update function used the GradientDescentOptimizer which aims to minimise the value of the error Variable, which is defined earlier as the square of the differences (a common error function). The 0.01 is the step it takes to try learn a better value.

Training – Steps and Learning rate

- Finally we started training our model, while printing the loss we get each round. To our surprise, sometime the loss was going down and sometimes it was going up. We realized that our step was too big, and so we attempted to make it smaller. The issue was fixed and the loss was steadily going down.
- The pace at which the loss was going down was too slow and we had to increase the total number of steps in order to get a smaller loss.

Testing

At the end of the training, we then split the remaining data into X_test, Y_test and sampled the results. To create a confusion matrix, we saved our predictions inside an array that held 2 values: {0,1}. Everytime the logistic regression returned a values, we checked if it was bigger or smaller than 0.5.

We have been at this proccess a few times, each time trying to change some of the variables, in order to get better results.

Results

Confusion Matrix:

Predicted	0	1	All
Actual			
0	12386	495	12881
1	5608	1511	7119
All	17994	2006	20000

```
final W:  [[-0.05389499]
 [ 0.01243871]
 [ 0.00525585]
 [ 0.7089657 ]]
final b:  [-0.19884658]
final Loss:  0.6047912
```

Neural Network

In this part we chose to use MLP(multilayer perceptrons).

Number of Neurons

As for the Neural Network, we used the same learning rate, optimizer function and loss function, with additional a hidden layer - with 10 neurons, for each one of the neurons we created number of weights as the number of features (In this case we have 4 features)

We had different attempts at the hidden layer variable and from what we gathered, it is recommened to place more neurons than the numbers of features, but less than 2 times the features. So we were training our model with 10,8,7,6,5 neuron.

	10 Neurons	8 Neurons	6 Neurons	5 Neurons
Accuracy	0.6684	0.67415	0.6702	0.67475
Loss	0.6168465	0.6122621	0.6203663	0.6173222
Confusion Matrix				

Predicted / Actual	0	1	All
0	12006	875	12881
1	5757	1362	7119
All	17763	2237	20000

Predicted / Actual	0	1	All
0	12070	811	12881
1	5706	1413	7119
All	17776	2224	20000

Predicted / Actual	0	1	All
0	12183	698	12881
1	5898	1221	7119
All	18081	1919	20000

Predicted / Actual	0	1	All
0	12242	639	12881
1	5866	1253	7119
All	18108	1892	20000

Keeping it fair:

While training this MLP model, we used a similar number of steps and learning rate (same as we did in the Logistic Regression part) all so that we could make a comparison based on the algorithm itself and not because we trained this model more than the other.

Results

Best results were achieved with 7 neurons

```
W:
  [[-1.3847809e-01  1.3938725e-01 -1.0154356e-01 -2.7380412e-02
  -1.0218264e-01  3.9868183e-03  5.6921523e-02 -1.5789089e-01
   1.0810335e-01 -2.7794974e-02]
 [ 7.1620129e-02 -9.9392466e-02 -7.3364012e-02  6.9725044e-02
   1.2915339e-01  7.7837378e-02 -8.2035199e-02  1.0904910e-01
   9.3240879e-02  1.6889591e-02]
 [-1.6410878e-02 -6.8567269e-02  5.9085667e-02 -3.4192830e-02
  -4.2779654e-02 -2.6598994e-02 -1.7565911e-04  3.4967672e-02
   1.3469948e-02 -1.3091291e-01]
 [ 4.4590598e-01  1.1338469e-02  5.2864697e-02  5.8577311e-01
  -5.6242847e-01 -2.4651256e-04 -7.3394194e-02  2.3330674e-01
   3.4019712e-02  4.4340692e-02]]
b:
  [-0.02022235  0.1         0.0998994  -0.04331133  0.22104895  0.11047985
  0.1         0.04598803  0.11650713  0.1       ]
Loss:
  0.59475523
```

Predicted	0	1	All
Actual			
0	12444	437	12881
1	5396	1723	7119
All	17840	2160	20000

Logistic Regression VS MLP:

Model	Test acuracy	Train loss
Logistic Regression	67%	0.8832
MLP	70%	0.5947