ANN(Artifitial Neuraul Network) Code review

1. First get data from kaggle(or UCI)
2. This data sheet(1000x13) is about churn of a company, and the goal of ours is to find out what kind of customer will have huge probability to leave this company.
3. Because RowNumber,CustomerId,Surname are unnecessary information(or it can be noises), so we get rid of them (X = dataset.iloc[:, 3:13].values) .
4. Extract the feature which is whether customer existed or not (y = dataset.iloc[:, 13].values).
5. Beware of categorical data! Categorical data should be transformed to a form that will not affect training. Here we use OneHotEncoder to encode the categorical data into the form like 0 0 1, 0 1 0,….. Also beware of dummy variable! Redundant encoded categorical data should be removed.
6. In order to train model, so split the data into training data and test data. Here we extract 20% of the data to be test data.
7. To standardize the range of independent variables and features, Scaling training and test data.
8. Build ANN layers. We use two hidden layers and one output layer. Output’s activation function is Sigmoid so that it can produce probability outcomes.
9. Finally, we use 10 batch and 100 epochs to deal with these 1000 data. And use cross validation to compare the results 10 times.

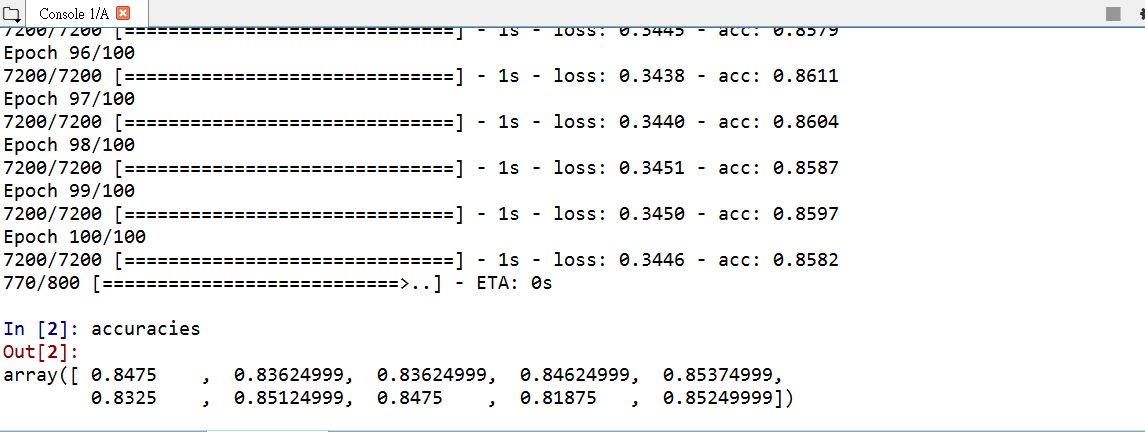
Data:

|  |
| --- |
| RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited  1,15634602,Hargrave,619,France,Female,42,2,0,1,1,1,101348.88,1  2,15647311,Hill,608,Spain,Female,41,1,83807.86,1,0,1,112542.58,0  3,15619304,Onio,502,France,Female,42,8,159660.8,3,1,0,113931.57,1  4,15701354,Boni,699,France,Female,39,1,0,2,0,0,93826.63,0  5,15737888,Mitchell,850,Spain,Female,43,2,125510.82,1,1,1,79084.1,0  6,15574012,Chu,645,Spain,Male,44,8,113755.78,2,1,0,149756.71,1  7,15592531,Bartlett,822,France,Male,50,7,0,2,1,1,10062.8,0  8,15656148,Obinna,376,Germany,Female,29,4,115046.74,4,1,0,119346.88,1  …  … |

Code:

|  |
| --- |
| **import** numpy **as** np  **import** matplotlib.pyplot **as** plt  **import** pandas **as** pd  # Importing the dataset  dataset = pd.read\_csv('Churn\_Modelling.csv')  X = dataset.iloc[:, 3:13].values  y = dataset.iloc[:, 13].values  # Encoding categorical data  **from** sklearn.preprocessing **import** LabelEncoder, OneHotEncoder  labelencoder\_X\_1 = LabelEncoder()  X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])  labelencoder\_X\_2 = LabelEncoder()  X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])  onehotencoder = OneHotEncoder(categorical\_features = [1])  X = onehotencoder.fit\_transform(X).toarray()  X = X[:, 1:]  # Splitting the dataset into the Training set and Test set  **from** sklearn.model\_selection **import** train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)  # Feature Scaling  **from** sklearn.preprocessing **import** StandardScaler  sc = StandardScaler()  X\_train = sc.fit\_transform(X\_train)  X\_test = sc.transform(X\_test)  **import** keras  **from** keras.models **import** Sequential #initial neural network  **from** keras.layers **import** Dense #build layer  **from** keras.wrappers.scikit\_learn **import** KerasClassifier  **from** sklearn.model\_selection **import** cross\_val\_score  **def** build\_classifier():  classifier = Sequential()  classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu', input\_dim = 11))  classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu'))  classifier.add(Dense(units = 1, kernel\_initializer = 'uniform', activation = 'sigmoid'))  classifier.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])  **return** classifier  classifier = KerasClassifier(build\_fn = build\_classifier, batch\_size = 10, epochs = 100)  accuracies = cross\_val\_score(estimator=classifier, X=X\_train, y=y\_train, cv=10) |

Result:



Summary:

Here, we use adam optimizer and cross validation 10 times to reach average 84% accuracy. To adjust the parameter, we can use other optimizer which can refer to [Deep Learning optimizers](https://medium.com/towards-data-science/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f), and also we can change batch size, epoch and then use cross validation to find better training model for our test set data.