CS 5310 Data Mining

Lab 1 – Chapter 7 Black Box Methods – Neural Networks and Support Vector Machines

Strain of Bacteria Identification with Mass Spectrometer Data

In this lab, we will use neural network method to analyze protein and peptides data from a mass spectrometer system and build a model to analyze the relationship between a patient clinic metrics and proteins’ mass spectrometer experimental data. We give some background information about the project as follows.

The project is about mining of big data on molecules making up and produced by beneficial microbes found in human. Currently, scientists are interested in studying microbiota that produce molecules involving in mind control (see <https://www.youtube.com/watch?v=mioR_WrkRaU> ). There is a state-of-the-art mass spectrometer system, Bruker's Tims TOF Pro, which can identify over 3,500 proteins, 20,000 peptides in about 15 minutes. It can also identify metabolites (molecules produced by living organisms including microbiota in human’s gut) that are important to health, including our mind, mood and mental health (more than gut feelings).

While the instrument is powerful in producing huge volume of data (about 3-5 G in a 20-minute run), data mining becomes a bottleneck, which is needed to improve the instrument performance, answer biological questions, and direct the development of analytical methods. The instrument comes with a third-party software (Peaks studio) that converts raw data into molecular structures. However, we need to extract much more information from the raw data by using machine learning means.

Two data files are provided for this lab. The first file is an Excel file that contains 22 patients’ clinic metrics data. We want to use the data in column “Schirmer collection (mm)” as the dependent variable for a regression analysis. All other columns in this file are not needed.

The independent variables will be drawn from a xlsx file named “Data\_SubjectStripTearSamples-Dec2020\_Report.xlsx”, which contains experimental data from those 22 subjects, in columns named “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, where italicized *X*’s are numbers. Data in each row are identity information about one protein and its mass spectrometer experimental data got from those 22 subjects. In this lab, we only use mass spectrometer experimental data from 22 subjects, so, all protein identity data should be removed, except for the column named “PG.ProteinGroups”, which we will use as names of the proteins. In other words, we should remove all columns except those 22 with a name in pattern “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, and the column named “PG.ProteinGroups”. Each retained row of proteomic experimental data cross all 22 patients will be used as an independent variable.

Firstly, we need to fill-in entries where data are “filtered”. These are the entries that have a string “filtered” instead of a numeric value. Because of this string, the data read into the data frame could be typed as “string” instead of a numeric type. We need to replace those “filtered” strings by value 0, and convert the type of columns to floating point type, if necessary.

We are interested in analyzing the relationship between the patient clinic metrics “Schirmer collection (mm)” and proteins’ mass spectrometer experimental data. So, transpose the matrix of the mass spectrometer experimental data so that data for each protein becomes a column, and the name of protein in “PG.ProteinGroups” becomes the column label. Note that after the transposition, the matrix should have 22 rows.

Identify proteins that have high correlations (>0.8 or <-0.8), and remove redundant proteins (i.e., for columns that have high correlation, retain only one of them). Retain the remaining columns.

Normalize every remaining column using min-max normalization.

Use “Schirmer collection (mm)” from the patients’ clinic data as the dependent variable, and all the columns from the mass spectrometer experimental data as independent variables, and build a neural network regression model to analyze the relation between “Schirmer collection (mm)” and proteins’ mass spectrometer experimental data.

Note that we do not split the dataset for training and testing in this lab. Instead, we train the model using the entire dataset and predict the results for the entire dataset.

Perform the following activities in Python.

1. Load the data in the patients’ clinic data file into a data frame.
2. Extract column “Schirmer collection (mm)” into an array. This will be the dependent variable of the model.
3. Load the data in the proteins’ mass spectrometer experimental data file into a data frame.
4. Remove columns except those 22 with a name in pattern “[*X*] Strip-No*X*\_Slot*X*-*XX*\_*XXXX*.d.PG.Quantity”, and the column named “PG.ProteinGroups”.
5. Zero fill-in entries with a string “filtered” and convert the type of columns to floating point type, if necessary.
6. Transpose the data frame so that data for each protein becomes a column, and the name of protein in variable “PG.ProteinGroups” becomes the column label. Please note that after this step, there should not be a row corresponding to column “PG.ProteinGroups” in the data frame before transposition. Columns in the transposed data frame will be the independent variables of the model.
7. Remove collinearity among the columns (i.e., proteins): For columns that have a correlation coefficient that is higher than 0.8 or less than -0.8, only one of them is kept in the data frame, and all others are removed. Note that in this lab, there is no need to control the number of columns you want to retain for modeling.
8. Normalize every remaining column using min-max normalization.
9. Create a line plot that displays the values of “Schirmer collection (mm)” and the average of the experimental data of all the proteins, using different colors.
10. Create a KerasRegressor with two hidden layers (See the sample codes following the instructions).
11. Create a visualization of the trained model using the plot\_model() function. Sample codes are provided after the instructions.
12. Use the model to make predictions for the entire dataset.
13. Compute the correlation coefficient and mean absolute error (MAE) between the true values of the first column and the predicted values.

Submit your codes (.py file) and results, including the line plot (Step 9), the visualization of the KerasRegressor model (Step 11), and the correlation coefficient and mean absolute error (MAE) between the true values of the dependent variable and the predicted values (Step 13), in a Word document, by the due date.

Please note that the following codes need to be customized in order to be used in your program. They are provided as hints to help your coding only.

**Sample codes**:

Build a KerasClassifier:

# define baseline model

# ZZZ is the total number of the independent variables in the data frame

# after removing co-linearity.

# NN1 and NN2 are two powers of 2 that are evenly distributed in the sequence

# of powers of 2 from 1 to the power of 2 that is the closest to ZZZ.

# NN2 must be smaller than NN1.

def baseline\_model():

# create model

model = Sequential()

model.add(Dense(NN1, input\_dim=ZZZ, activation='relu'))

model.add(Dense(NN2, activation='relu'))

model.add(Dense(1, activation='softmax'))

# Compile model

model.compile(loss='mean\_squared\_error’, optimizer='adam')

model.summary()

return model

estimator = KerasRegressor(build\_fn=baseline\_model, epochs=10, batch\_size=256, verbose=0)

Plot the model:

import tensorflow as tf

dot\_img\_file = 'model.png'

tf.keras.utils.plot\_model(model, to\_file=dot\_img\_file, show\_shapes=True)