

Pattern Recognition 2020/2021

April 23rd, 2021

COVID-19 Prediction

*Pattern Recognition Project Assignment*

*Milestone I*

Authors:

Diogo João Durães Antunes, 2017251783, djantunes@student.dei.uc.pt

Vitalina Holubenko, 2017255810, vitalina@student.dei.uc.pt

**Table of Contents**

1. Introduction

With COVID-19 being a current and widespread pandemic, which has been affecting our daily lives and infected more than 140 million people worldwide, it has become a vastly researched topic, and thus, we have various datasets that are related to this subject.

In this assignment, we were tasked with developing a software that predicts the outcome of the COVID-19 disease by employing data regarding the patients affected by it, and other related background information. In this first milestone we will be solely treating the data regarding the patient, without utilizing the background datasets. The main objective is to design a binary classifier (described in **Scenario A**, in the statement) which will categorize the outcome for each infected patient (released or not released), making use exclusively of various features extracted from said patient’s information.

1. Data Set

The data set used in this milestone of the assignment consisted of various features regarding the infection cases of more than 10,000 patients, mostly from South Korea. These features include personal information about the patient, such as age, sex, and nationality, as well as information pertaining to the infection, such as number of contacts, various dates relative to the evolution of the case and, finally, the aforementioned case outcome. The data set is available under the name ‘*PatientInfo.csv’* at <https://www.kaggle.com/kimjihoo/coronavirusdataset>.

1. Methodology

All steps were implemented and tested in the R2020b version of MATLAB, with the help of the Statistical Pattern Recognition Toolbox (*STPRTool*).

3.1. Data Preprocessing

In order to make the provided data set appropriate for classification, various analysis and treatment techniques were employed to alter, complete and decrease the data, clarified in this section of the report.

Firstly, to import the data set we generated a script named ‘*import\_data.m*’, which simply reads the *.csv* file with the patient information and returns a table with its contents.

3.1.1. Data Conversion

One of the first restrictions encountered in the original *.csv* file was that most of the data was not in numeric form, such as the sex, age, country, province, city, case of infection, symptom onset date, confirmation date, released date, deceased date and current state of the patient. As a result, we decided to convert all features into numeric values (*double*).

To accomplish this task, we wrote the script ‘*convert\_data.m*’. Text information was converted to categorical information, using the MATLAB method *grp2idx*, which takes a vector of values (text), and returns a vector of the same size, made up of positive integers representing each one of the distinct values found (category). In order to convert the date information to numeric, we referred to a MATLAB method called *datenum*, which straightforwardly converts the date to a serial date number.

Lastly, we converted the resulting table to an array with the *table2array* method, which left us with a patients-by-features matrix.

3.1.2. Missing Data

Many of the features included in the data set in question were missing a significant amount of data, prompting us to develop a script with the purpose of filling in all the missing values (*NaN*). In this section we seek to explain said process, as well as the reasoning behind the decisions made regarding the method utilized for each feature.

**Sex.** The missing values in the sex feature were filled according to the female-to-male ratio. In other words, we calculated the distribution of the sexes in the existent data and filled the rest according to this distribution.

**Age.** To fill missing age data, we calculated the average age of all the existing values and used it as a constant for the patients with no age value, using the *fillmissing* function.

**City**. For this specific feature, we took the most common city in each province (using the *mode* function) and used it as a constant for every missing city value corresponding to said province. In the rare case that a province did not have any city associated with it, we simply added a new unique value, representing a new city, and attributed it to all patients living in that province.

**Infection case**. Using the same logic as with the antecedent feature, the most common case of infection in the city in which a patient resides is attributed to all the patients living in the same city without a value for this feature. If a city does not have any registered information for infection cases, add a new value to the categories and use it as the constant.

**Infected by (ID)**. This feature was filled with the constant 0, since there was not any legitimate way to speculate as to who infected a patient.

**Number of contacts**. Utilizing the same logic as with the age feature, we calculated the average number of contacts in with the existent data and applied it as a constant to the missing values.

**Symptom onset date**. This feature was filled according to the confirmation date, which contained no missing values. By calculating the mean of the difference between the confirmation date and the symptom onset date, we were able to roughly estimate the missing values for the latter.

**Released date**. This feature was filled in the same way as the previous one, with the exception of assigning the value 0 to any patient not released (isolated or deceased). Released patients without a date of release were attributed an estimation based on the confirmation date.

**Deceased date** - The same exact method was used to fill the missing data in this feature as with the released date, with the deceased patients without a corresponding being attributed an estimation based on the confirmation date, and the rest being filled with the value 0.

3.1.3. Data Structure

At this stage, before we can normalize the data and perform feature selection and reduction, we need to convert the three state classes into two, since it is only required to differentiate the released outcome from the other two (isolated or deceased).

Now the only thing that is left to do is to construct the data structure that is composed of the *X*, *y*, *dim*, *num\_data* and *name* elements.

The *data.X* component includes all the features from the original data set save for the patient ID and state: (1) *sex*, (2) *age*, (3) *country*, (4) *province*, (5) *city*, (6) *infection\_case*, (7) *infected\_by*, (8) *contact\_number*, (9) *symptom\_onset\_date*, (10) *confirmed\_date*, (11) *released\_date* and (12) *deceased\_date*.

  The *data.y* component represents the target vector that indicates the outcome state of the patient (1, for released, or 2, for isolated or deceased). Finally, *data.dim* is the total number of features in the current data set and *data.num\_data* is the total number of patients that we have a record of.

3.1.4. Data Scaling

To normalize the data features (mean 0 and standard deviation 1) we used the *scalestd* function, developed in the practical classes (‘*scalestd.m*’ script), in which we subtract mean value of a feature from all existing values in said feature and then divide them by its standard deviation.

3.1.5. Feature Selection and Reduction

Assessing Redundancy: Correlation Test

Concerning the study of the redundancy that present in the given data set, in which it is determined which features are close to being identical, we resorted to using the built-in MATLAB function *corrcoef,* which takes in the array *data.X* transposed and returns a matrix of correlation coefficients for the features in that array. We then applied the *heatmap* method, to visualize the heatmap chart from the resulting matrix.

As we can see in the figure above (figure 1), there are not any clearly redundant features aside from the symptom onset date and the confirmed date, for which the correlation coefficient has a value of 0.998917, quite close to reaching 1, which implies a direct correlation. This was probably caused by the method used in filling the missing symptom onset date data, heavily dependent on the confirmed date feature.

As for the rest of the features, we have not found that there is much similarity based on the values that were obtained in this step, resulting in the symptom onset date being the only discarded feature.

Assessing Relevancy: Kruskal-Wallis Test

After performing the Kruskal-Wallis test on the features present in the data structure, with the use of the function *kruskalwallis*, and obtaining the correspondent relevancy scores, the following features were selected: (4) *province*, (7) *infected\_by*, (10) *confirmation\_date* and (11) *released\_date*.

Assessing Variance: Principal Component Analysis

With the features manually selected in the previous sections, two techniques of feature reduction were implemented. The first of those, the principal component analysis (PCA), was implemented using the function *pca*, reducing the total number of features to 2.

When faced with the choice of the new dimensionality for the resulting data set, we consulted the features’ eigenvalues with the function *eigen*, applying the Kaiser rule by keeping only the features with values below 1.

Cassifiers

1. Experimental Results

Text.

1. Conclusion

As evident in the results obtained…