CNSCC.361 Artificial Intelligence

Lab session 3 Data Pre-processing

AIM OF THE SESSION:

- Data Standartisation
- Anomaly Detection
- Data Normalisation

Data Pre-processing

Data pre-processing is an important step in converting Data to Knowledge. Usually, the data gathered from different sensors and observations can be highly correlated and difficult to distinguish. This can lead to a variety of problems, e.g. initial poorly controlled process of collecting data samples, like out-of-range values (e.g., Income: −100), impossible data combination (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Also, data can have very big dimensionality which makes it very difficult to examine and represent (curse of dimensionality) by common techniques for visualisation. Therefore, we use pre-processing to achieve desirable adjustment of the initial raw data and prepare it for further processing like clustering, classification machine learning algorithms, etc.

In this labsession, we will cover:

1. Standardization
2. Anomaly Detection
3. Normalization

Standardisation and normalisation are examples of a data pre-processing technique that is practically compulsory at the start of the process. These pre-processing operations re-scale the values of features (input variables) into a more familiar range, and make different features comparable in terms of their values and ranges. Thus, the two techniques can reduce the influence of differences in the magnitudes of the input variables.

Anomaly detection is an important part of the pre-processing. If anomalies exist in the observed data, i.e. outliers, novelties, noise, and exceptions, machine learning algorithms will produce misleading results. Therefore, anomalies have to be removed before further processing in order to obtain the valid results. It is also important to remove outliers before applying normalization, because otherwise they will highly influence the result.

In this Lab session we will consider pre-processing data on the example of the Flame dataset (http://cs.joensuu.fi/sipu/datasets/). The data has 2 attributes. In order to load this file, please, use the *load* command. However, for a better demonstration of the pre-processing techiniques, two of the data samples were replaced manually.

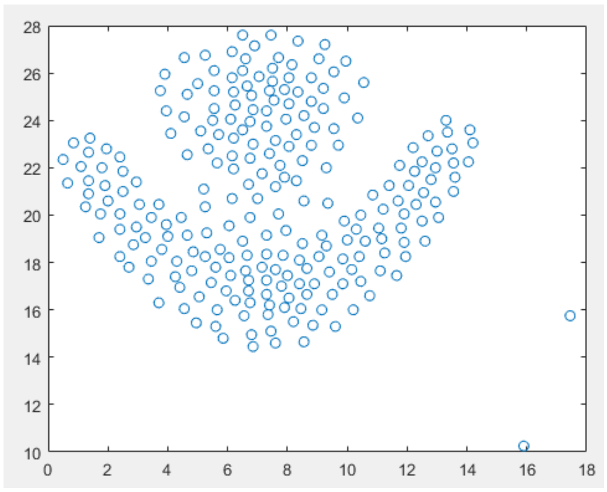
```
load Flame_data.mat;
```

Table 1: Example of data

IDX	**X**	**Y**
…	…	…
37	4.85	18.45
38	4.30	18.05
39	3.35	17.30
40	3.70	16.30
…	…	…

We can plot the data with the following command

```
plot(Data(:,1),Data(:,2),'o');
```



Data Standardization

Standardisation re-scales the numerical values of different features of the data into a more convenient rang. It is necessary to standardize variables in order to make the data comparable by values and ranges and reduce the effect of differences in the magnitudes or scales of the input variables. This is especially important in cases where the distance measure that is used, e.g. Euclidean, is insensitive to the different features.

In order to apply standardization we need to perform an algorithm that is realizing the formula as given in the lecture:

$$x_{std} = \frac{x - \mu}{\sigma}$$

Note, that the operation is performed for each feature separately.

Let's develop code that will realize the standardization operation step by step.

Step 1. Obtain the size of the dataset.

```
[L,W] = size(Data); % L - number of rows; W - number of
attributes
```

Step 2. Obtain the mean and standard divation of the dataset.

```
Miu=mean(Data); % Miu is the mean of the dataset, which is a 1 x
W dimensional vector;
Sigma=std(Data); % Sigma is the standard deviation of the
dataset, , which is a 1 x W dimensional vector;
```

Step 3. Standardise each attribute of the dataset:

```
Data_stand=zeros(L,W); % Create a variable for storing the
standardised data
for i=1:1:W
    Data_stand(:,i)=(Data(:,i)-Miu(i))./Sigma(i); % Perform
standardisation for each attribute
end
```

Exercise 1:

a) Convert the script into a function named *standard()*

b) Develop code that is calculating the mean and standard divation (rather than using the Matlab built-in functions).

Show to the TA your result and observations.

Anomaly Detection

Anomaly detection (or outlier detection) is the identification of values that differ greatly in relation to the rest. In the context of AI, having high quality data is important for properly training predictive and classification models. Standardisation can be used as a simple method to identify possible outliers by examing how far values are from the origin.

One can use the so-called “3sigma” rule to identify the anomlies, which is based on the Chebyshev inequality:

$$P\left(\|\mu - x\| \leq n\sigma\right) \geq 1-\frac{1}{n^2}$$
$$P\left(\|\mu - x\| \geq n\sigma\right) \leq \frac{1}{n^2}$$

which means that at least $1-\frac{1}{n^2}$ of the data are within the range of $\left[-n\sigma, n\sigma\right]$.

If $n=3$, it means that the vast majority of data ($>\frac{8}{9}$ in any case, rising up to >99.7 if data follows a Gaussian distribution) are within the range of $\left[-3\sigma, 3\sigma\right]$.

Using the standardised data, one can identify the anomalies easily using the following condition:

$$IF\left(x_{std}>3\right)OR\left(x_{std}<-3\right)$$
$$THEN\left(x is\, anomalous\right)$$

Plot the standardized data using the following command:

```
plot(Data_stand(:,1),Data_stand (:,2),'o');
```

You can observe that the majority of the values will be in the interval of $[-3, 3]$ and only outliers will have values larger than 3 by absolute value as described in the Lecture.

Exercise 2:

Develop code that can automatically give the indices of the data samples, which are beyond the interval of $[-3, 3]$.

Tips.
a) Use the build-in function `find`;
b) Use the following command to find out how to use this build-in function;

```
help find
```

Normalization

Normalization means adjusting values measured on different scales to a notionally common scale. Normalization converts the nominal numerical values on different features (dimensions of the input data vector) to the interval $[0,1]$:

$$x_{norm} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Note, that the operation is performed for each feature separately.

To find the minimum values of the dataset for each feature, you can use the following code:

```
Data_min=min(Data); % Data_min is the minimum values of different
attributes of the dataset, which is a 1 x W dimensional vector;
```

Similarly, to find the maximum values of the dataset for each feature, you can use the following code:

```
Data_max=max(Data); % Data_max is the maximum values of different
attributes of the dataset, which is a 1 x W dimensional vector;
```

Exercise 3:

Implement the normalization using `Data_min` and `Data_max`.

Tip.
Please refer to the step 3 of standardisation and use the similar form.

Exercise 4:

Plot both the original data and the normalised data side by side.