Machine Learning in Engineering

Tutorial/Assignment 1 (**100 pts**)

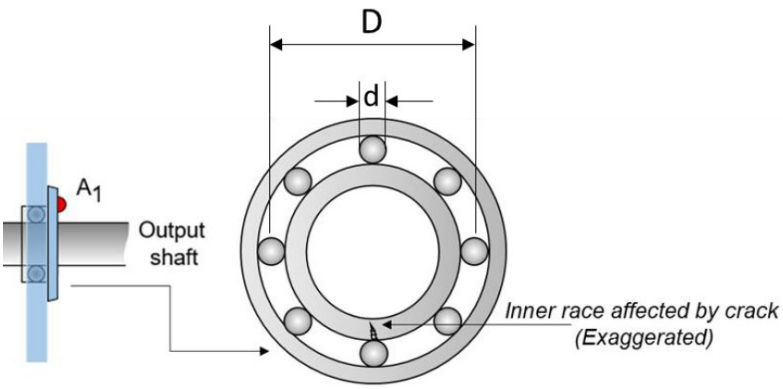
Writing instruction*: all derivations/codes are to be written in a .pdf file format (word/latex/written by hand then exported to the pdf format). When asked about* the *derivation of equations and formulations, illustrate the transition from one formulation step to another using a combination of explanatory text and mathematical formulas. This process involves a clear explanation of the reasoning behind each transformation or manipulation of equations, accompanied by mathematical expressions to substantiate the transitions.*

If asked to code specific algorithm in Python, then code is to be explained line by line (well documented). An example of this is given in Figure 2 at the end of this document. Please note that you are not allowed to use existing subroutines, but code it from scratch. You are only allowed to use linear algebra and probability libraries such as scipy, random, numpy, cvxopt, pyplot, and similar. Also, you are allowed to use optimization subroutines for solving quadratic programming problems. The final results are to be plotted or tabulated. Add a discussion paragraph to each result (figures, confusion matrices) you add to the pdf file. Focus on the explanation ***why*** *is something* ***happening*** *and* ***what we see*** *in the figure/matrix.*

Note that using artificial intelligence tools (including ChatGPT) to code the required algorithm (write text), or copying existing open-source codes from platforms like Google and similar, are strictly prohibited for the completion of all exercises.

1. Total: 40pts

Localized faults in a rolling element bearing may occur in the outer race, inner race, cage, or any of the rolling elements. In the following figure we may see an example of a defect in the inner race.



Ball bearing cross-section (Magnified)

Given the data sets on the following webpage

https://github.com/mathworks/RollingElementBearingFaultDiagnosis-Data/tree/master/train_data

a) Visualize one inner and one outer fault data set in the time domain, given the *train_data* set. (2.5pts) Is there any difference between these two signals? If yes, explain what kind of difference? (2.5pts)
b) To distinguish inner from outer fault, one requires a specific type of feature by the help of which these two signals can be uniquely identified (separated). What kind of features would you propose to use for classification of these two signals? Think like an engineer (You cannot use the solution under item c).). (10pts)
c) Estimate the mean, standard deviation, skewness and kurtosis of all inner and outer fault signals in the *train_data* set by using the statistical library *statistics.py*. Plot inner and outer fault class in 2D by using the mean and standard deviation as features. (3pts) Denote classes by different symbols and colors. Repeat the same plot, but this time by using skewness and kurtosis as relevant features. (3pts) What do you notice about these two plots? Are all proposed features relevant? If not, why not? (4pts)
d) Repeat the same analysis as in c) only this time divide the overall time interval to 10 equal subintervals and compute previously mentioned statistics for each of the subintervals. Plot the mean and standard deviation of the corresponding subintervals by distinguishing the classes as you did in c) (note that each subinterval is one sample point for the specific class). (5pts) Repeat the same analysis only this time given skewness and kurtosis as relevant features. (5pts) What can you conclude now? Is there any difference in feature representation compared to the plots in c) ? (5pts)

2. Total: 60pts

a) Code the dual form of the supervised hard margin SVM algorithm in Python from scratch (not allowed to use any code from the internet or existing libraries, see Lecture 2 for the theory of the dual SVM algorithm). You may use existing functions from libraries (as mentioned earlier) for the optimization part. The SVM code (function/subroutine) has to work for a general dataset represented by two classes and two features. (25pts)

b) Given the skewness and kurtosis features of bearing faults presented in Ex. 1c), predict the fault type (inner or outer) if a new data point from *test_data* is given. Plot the obtained hyperplane and represent it as a two dimensional plot (5pts). Is the obtained hyperplane correct? Plot the predicted labels for the points that are not seen by training (*test_data set*), and compare them to the exact labels (5pts). use different colors or markers for this purpose. Search in the literature for the confusion matrix, and use this metric (example shown in Fig. 1) to asses performance of the method. Compare estimate to the ground truth (5pts).

c) Repeat the training under b), only this time use the skewness and kurtosis features defined in Ex. 1d) on subintervals (10pts). Compare the obtained hyperplanes to the ones obtained in Ex. 2 b). What do you conclude? (5 pts). Plot the predicted labels for unseen points (those not used in the training algorithm, *train_data*) (2.5pts). Estimate the new confusion matrix, and compare to the one in Ex. 2c) (2.5pts).

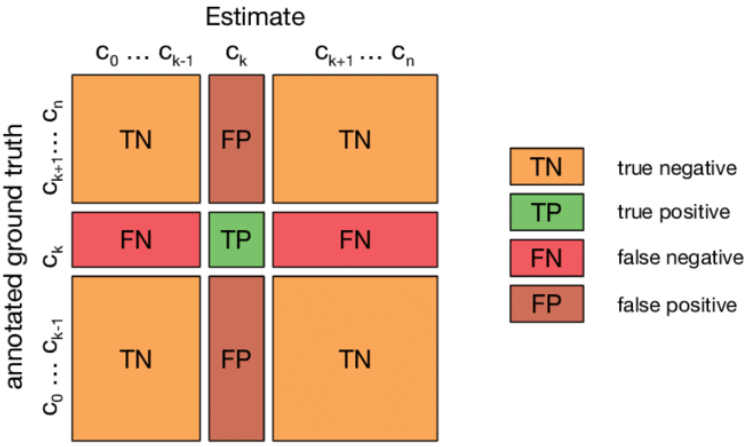


Figure 1: Confusion matrix example

1. The light position indicator is made using the same image processing package OpenCV. From the previous exercises, the image is taken and converted to a gray-scaled CV image.

```
1          cv_bridge::CvImagePtr ros_image_ptr;
2          ros_image_ptr = cv_bridge::toCvCopy(myimage,sensor_msgs::image_encodings::MONO8);
```

2. The image is blurred such that the sharp edges of light bodies are averaged/rounded. Such that light sources are smoothed removing the 'rays' created by the glare.

```
1        blur( ros_image_ptr->image, blurred, Size(5,5) );
```

3. The threshold parameter to determine whether or not the light is on or off is reused. Now it is used as an input for the CV function threshold() using a binary thresholding which keeps the pixels above the threshold value as white and all the rest as black.

```
1    threshold(blurred,thresh,threshold_,255,THRESH_BINARY);
```

4. (a) CV function 'findContours()' is used to find connected points along a boundary, this could also be done with Canny edge finder or other edge-finding methods but contours offers advantages for the next steps, the found contours around the light sources are stored in a vector.

```
1          findContours( thresh, contours, hierarchy, RETR_TREE, ...
               CHAIN_APPROX_SIMPLE );
```

Figure 2: Example of presenting code using snippets