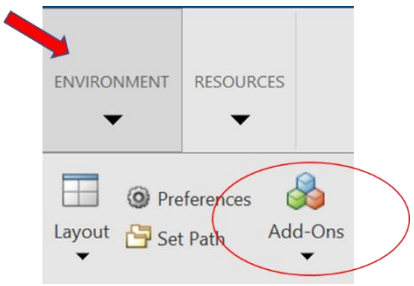
Module-Lab 11: Probabilistic Machine Learning

Intro: In this practical module we will concentrate on SVM and NN techniques applied to the classification problem studied in the Lab10 that is, based on the IDOL dataset. It is desirable that student make use of the LibSVM, which is available here: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Exercise 1

a) Download the LIBSVM packages available in the section <*Download LIBSVM*> as zip or tar files format. Unzip the files and respective directories into a given folder eg, C:\Users\Smith\ \ACP_codes\libsvm-3.25

b) Open Matlab. Using command line, check whether a C/C++ compiler is configured by typing the following command:
>> mex -setup



It may be the case that you do not have a compiler in your PC then, the **MinGW64** compiler should be installed. To do so, via the Matlab menus, go to <Environment> and click on <Add-Ons>.

Next, click on <get add-ons> and then search for "mingw64"

… finally, install the compiler.

c) After the installation, type **`>> mex -setup`** to verify everything is okay. You should get a message like this one:

```
MEX configured to use 'MinGW64 Compiler (C)' for C language compilation.

To choose a different language, select one from the following:
 mex -setup C++
 mex -setup FORTRAN
```

➔ Now, we are able to compile the library. Go to the Matlab command line and run the file named "make.m" that is in the folder **…\libsvm-3.25\matlab**

d) Develop a code, in Matlab (ie, a .m file), to train and test your algorithms using the LibSVM. Firstly, load the training and test sets comprised in the .mat files
'IDOL_trainingset.mat'
'IDOL_testset.mat'

Secondly, call the libSVM's function named ***svmtrain***. As a guide, you can use this piece of code:

```
SVM_par = '-s 0 -t 0 -e 0.01 -m 4000 -b 1 -q';    [See Appendix 1]
C_model = svmtrain(Xtr.label,Xtr.data,SVM_par);
```

Hint1: It shall be useful to add a path to the folder where the libSVM is installed for example,

```
addpath ..\..\..\libsvm-3.25\matlab\
```

Hint2: If it is taking too long to train the model, it is convenient to normalize the data, for example (See **Exe2**) using the simple normalization expressed by:

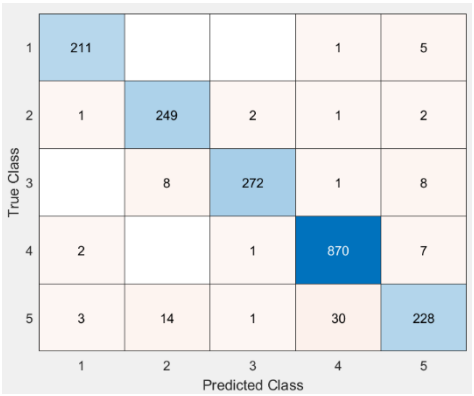
$X' = \frac{X_i - min_i}{max_i - min_i}$ where `i` indicates the respective `i`th feature

e) Check the following attributes of your SVM model:
C_model.Label
C_model.sv_indices //these are your support vectors
…

f) Now, to test your SVM model on the test set, you can take advantage of this code line:
```
[Yhat, acc1, Ylike] = svmpredict(Xte.label, Xte.data, C_model,'-b 1');
```

g) Obtain the confusion matrix by using the variable "Yhat" such as:
confusionmat(Xte.label,Yhat)

h) Calculate the Accuracy and the Balanced Accuracy.
Hint: you have to calculate the performance measures using the variable "Ylike" but, pay attention to the way the classes/categories have been organized by the libSVM through the parameter **C_model.Label**



Exercise 2

a) Repeat the Exercise1, namely (d),(e),…,(h) but, this time, using a simple normalized data.
Hint: $X' = \frac{X_i - min_i}{max_i - min_i}$ where i indicates the respective i^{th} feature

b) Important: during the test phase the test set must be normalized based on the "parameters" calculated on the training set i.e., the minimum (min_i) and maximum (max_i) values, per feature, must be applied to the test set.

c) Compare the results obtained in Exe1 and check the elapsed time necessary to train the model with and without normalization. In Matlab we can use the commands ***tic*** and ***toc*** to have an estimation of the elapsed time.

Exercise 3

a) Train the SVM without the "probabilistic" flag, ie *'b' = 0*
```
SVM_par = '-s 0 -t 0 -e 0.01 -m 4000 -b 0 -q';    [See Appendix 1]
C_model = svmtrain(Xtr.label,Xtr.data,SVM_par);
```

b) Now, perform the prediction stage on the training set
```
[Yhat, acc1, Ylike] = svmpredict(Xtr.label, Xtr.data, C_model,'-b 1');
```

c) Plot the histograms, per class, using the predicted variable "Ylike"

Hint: the LibSVM uses the one vs all principle thus, the variable **Ylike** has now 10 columns, according to the following convention, where "C' denotes class/category:

Ylike(:,**1**)	Ylike(:,**2**)	**3**	**4**	**5**	**6**	**7**	**8**	**9**	Ylike(:,**10**)
C**1**vsC**2**	C**1**vsC**3**	C**1**vsC**4**	C**1**vsC**5**	C**2**vsC**3**	C**2**vsC**4**	C**1**vsC**2**	C**2**vsC**5**	C**3**vsC**4**	C**4**vsC**5**
'5'vs'4'	'5'vs'2'	'5'vs'1'	'5'vs'3'	…	…	…	…	…	'1'vs'3'

d) Using the Sigmoid function on the "non-probability" outputs, verify whether the values are the same compared to the "probabilistic" case obtained in exe1(d) or exe2.

Exercise 4

a) Repeat Exe1 but this time use a **Neural Network** instead. See exe2 of the Lab.session 8.

Appendix 1

```
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default
0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for
probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default
1)

The k in the -g option means the number of attributes in the input data.
```

For more details please visit the webpage:
https://www.csie.ntu.edu.tw/~cjlin/libsvm/

// ----------