CS303B: AI Lab6

Aim: To understand how to perform classification using SVM in Matlab.

LIBSVM is very popular C++ implementation of support vector machines. It also comes with a Matlab wrapper, which enables its function called within Matlab. It will allow you to fast prototype classification systems using SVMs.

LibSVM can be downloaded from: https://github.com/cjlin1/libsvm

On Windows systems, pre-built binary files are already in the directory '..\windows', so no need to conduct installation. Binary files only for 64bit MATLAB on Windows are provided.

On mac systems, you can get binary files from our blackboard.
https://bb.sustech.edu.cn/webapps/blackboard/content/listContentEditable.jsp?content_id=_133557_1&course_id=_1800_1
Both windows and mac binary files only for 64bit MATLAB are provided on blackboard.

The dataset

In this lab, we will use the iris dataset, which comes with Matlab as a built-in dataset called 'fisheriris'. (If you are already familiar with the iris dataset, please skip this section)

The description of this dataset could be found in

http://archive.ics.uci.edu/ml/datasets/Iris

The dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

Each sample is described by a feature vector containing the following attributes: (features or measurements)
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Name of the three classes:
-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Load in the dataset and data scaling

Load the dataset 'fisheriris' using the load command in Matlab.

1. Load fisheriris. This will create two variables. *meas* is a matrix with each row containing the measures of the four attributes for each sample, and *species* is a column vector containing the ground truth of each sample, i.e., the type of iris plant. Out of the 150 items, 1-50 are setosa, 51-100 are versicolor and 101-150 are virginica.
2. Scale the data in each column of *meas* (i.e., each measurement) into the range of [0, 1], by the formula (x-min)/(max-min), with x being the measurements.
   You may try the following code:
   ```
   for i=1:max(size(meas,2))
   t = meas(:,i);
   t = (t-min(t(:)))./(max(t(:))-min(t(:)));
   meas(:,i) = t;
   end
   ```

LIBSVM setup

1. Unzip the LIBSVM into the current working folder in Matlab.
2. In the folder of \ libsvm-master, there is a subfolder called matlab. Read the README file in that folder.
3. Follow the instructions to install LIBSVM. (--basically running make.m)
   You can replace 'CFLAGS' with 'COMPFLAGS' if encountering the error 'gcc: error: \-fexceptions: No such file or directory'. Alternatively, you can use the compiled files in the folder 'windows'. You can also get the compiled file on blackboard.
4. There are plenty of instructions in the README file on how to use SVM to do classification, please make sure that you read all of the instructions.

Classification -- Linear Kernel SVM

1. The main function for training a SVM is *svmtrain*(). It has three input variables: *training_label_vector*, *training_instance_matrix*: *libsvm_options*
   -training_label_vector:
     An m by 1 vector of training labels (type must be double).
   -training_instance_matrix:
     An m by n matrix of m training instances with n features.
   -libsvm_options:
   A string of training options in the same format as that of LIBSVM.

   Thus to use SVM with a linear Kernel to do some classification task using the *fisheriris* dataset: You can try the following code: (we use one against the rest strategy). Let classify: 'virginica' against the *rest* (versicolour+ setosa).

```
load fisheriris; %% and scale using the above scaling code
whos;
% preparing the vectors for 'virginica' against the rest
a = zeros(size(species));
for i=1:max(size(a))
a(i) = isequal(species{i},'virginica');
%% in the vector a, 1 means 'virginica'; and ‘0’ means others
end

cvo = cvpartition(a,'k',2);
%% split the data into two fold. (one fold for training and the rest
for %testing
trIdx = cvo.training(1); %% get the index of training samples
teIdx = cvo.test(1);   %% get the index of the test samples
training_label_vector = a(trIdx); %% creating the training label
ground %truth
training_instance_matrix = meas(trIdx,:); %% creating the training
data %matrix
test_label_vector = a(teIdx); %% creating the testing label
ground %truth
test_instance_matrix = meas(teIdx,:); %% creating the test data
```

2. Now training SVM using a linear kernel

   ```
   model = svmtrain(training_label_vector, training_instance_matrix, '-t
   0');
   ```
3. Now doing classification on the test data:

   ```
   [predict_label, accuracy, dec_values] = svmpredict(test_label_vector,
   test_instance_matrix, model);
   ```

   You are getting there … , what is your classification accuracy?

4. Can you create a confusion matrix from the results of 'virginica' vs. the rest? What are the false positive rate, and false negative rate?

<u>Classification— RBF kernel</u>

1. Now try the rbf kernel using the code:
   ```
   model = svmtrain(training_label_vector, training_instance_matrix, '-t 1
   -g 0.07');
   ```
2. Now do classification on the test set using the trained RBF-SVM

3. What is the accuracy? Is it better than the one that you obtained using SVM with a linear kernel?
4. Try different value of g?

Finally…

Make sure you **backup** your work so that you may need it in next week's workshop.