

(Computer)

In this problem we'll use LIBSVM, a library for Support Vector Machines. A version of this library is available directly from the course web-page: <http://www.svcl.ucsd.edu/courses/ece175/data/libsvm-mat-2.82-2.zip> This should work under Windows architectures. In case it doesn't, or if you have another architecture, here is how to download and install LIBSVM with MATLAB interface:

1. Go to LIBSVM's website:
`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`
download LIBSVM and extract the software package to an appropriate location. (Detailed instructions for installation can be found inside the ZIP file, this provides only a brief overview.)
2. Once extracted, you can compile LIBSVM to be used in command line interface and in MATLAB. Here is how to get the latter. You should start MATLAB program and go to the `matlab` subdir (inside `libsvm-<version>` dir).
3. On Unix systems type `make`, at the MATLAB prompt (`>>`), to build the matlab executable files (`mex`): `svmtrain.mexglx` and `svmpredict.mexglx`. Note that it assumes MATLAB is installed in `'/usr/local/matlab'`, if not, please change `MATLABDIR` in `Makefile`.
4. Add the path of the installation to your MATLAB environment using the MATLAB command `addpath('<installation path here>')`;

LIBSVM is now ready to use. Here is how to use LIBSVM with MATLAB interface:

1. The command to train the SVM model is
`model = svmtrain(training_label_vector,
training_instance_matrix, ['libsvm_options']);`

where:

- `training_label_vector`: An `m` by 1 vector of training labels.
- `training_instance_matrix`: An `m` by `n` matrix of `m` training instances with `n` features.
- `libsvm_options`: A string of training options

2. The command to predict the classes using the model trained above
`[predicted_label, accuracy, decision_values] =
svmpredict(testing_label_vector, testing_instance_matrix,
model);`

where:

- `testing_label_vector`: An `m` by 1 vector of prediction labels.

-testing_instance_matrix: An m by n matrix of m testing instances with n features.
-model: The output of svmtrain.

3. Relevant libsvm options

-t kernel_type :set type of kernel function (default 2)
 0 – linear: $u \cdot v$
 2 – radial basis function: $\exp(-\gamma * |u - v|^2)$
-g gamma : set gamma in kernel function (default 1/k)
-c cost : set the parameter C
-v n : n-fold cross validation mode

Usage Examples

1. LIBSVM by default chooses a gaussian kernel, but can also be forced to do so by using the option '-t 2'. To choose linear kernel use '-t 0'
2. To set the cost use the option '-c <value>'
3. To set the gamma parameter use the option '-g <value>'
4. To perform 2-fold cross-validation use the option '-v 2'. Note: While performing validation no model will be retruned instead the validation classification accuracy is returned. You can use different values of model parameter to find the best classification accuracy.
5. Example 1: Train a model using `trainrow`, `labelTrain` with gaussian kernel and $C = 1, \gamma = 0.4$ use
`model = svmtrain(labelTrain, trainrow, ['-t 2 -c 1 -g 0.4']);`
6. Example 2: Perform 2-fold cross-validation using two different C values
`accuracy1 = svmtrain(labelTrain, trainrow, ['-t 2 -c 1 -g 0.4 -v 2']);`
`accuracy2 = svmtrain(labelTrain, trainrow, ['-t 2 -c 4 -g 0.4 -v 2']);`
7. Example 1: Predict/Classify `testrow`, `labelTest` use
`[predicted_label, accuracy, decision_values] = svmpredict(labelTest, testrow, model);`
accuracy(1) will give the classification accuracy.