CSE 5526 – Spring 2018

Introduction to Neural Networks

Programming Assignment 3: SVM

Non-coding RNAs (ncRNAs) have a multitude of roles in a cell, many of which remain to be discovered. However, it is difficult to detect novel ncRNAs in biochemical screening. Recently, some studies show that computational methods can accurately detect ncRNAs, which can be treated as supervised classification. To perform the classification, for each instance, an 8-dimensional feature is used as input to a classifier, including the length of genomic sequence and nucleotide frequencies:

1. A feature value computed by the "Dynalign algorithm"
2. The length of shorter sequence
3. 'A' frequencies of sequence 1
4. 'U' frequencies of sequence 1
5. 'C' frequencies of sequence 1
6. 'A' frequencies of sequence 2
7. 'U' frequencies of sequence 2
8. 'C' frequencies of sequence 2

This project asks you to train a support vector machine (SVM) to determine if a genomic sequence is an ncRNA.

You should use LIBSVM for the project, which is a popular open-source SVM toolbox that has been implemented in many programing languages, such as C/C++, JAVA, and MATLAB. Download LIBSVM from this webpage:

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

The training data and the test data can be downloaded from the class website.

For each file, the data are organized as LIBSVM format:

```
<label>    <index1>:<value1>    <index2>:<value2>  ...
…
…
```

Each line contains an instance. <label> is a bipolar value indicating the class label. <index> is an integer starting from 1. <value> is a real number corresponding to a feature value which has been scaled to [0, 1]. If an index is omitted, it means that the corresponding value is zero. For MATLAB users, you need to use *libsvmread* function to load the data to the workspace.

Do the following:

1. Classification using linear SVMs. Train a set of linear SVMs with different values of the parameter *C* using **the training data set**. Vary *C* from the exponential sequence $C = (2^{-4}, 2^{-3}, 2^{-2}\ldots, 2^{7}, 2^{8})$ and for each *C* value train an SVM. Use each trained SVM to classify **the test data**. Plot the classification accuracy with respect to different *C*.
2. Classification using RBF kernel SVM: $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\alpha\|\mathbf{x}_1 - \mathbf{x}_2\|^2)$. There are two steps to implement the task.
    1) Use 5-fold cross validation to choose the best *C* and α. To do so, first randomly choose 50% (i.e., 1000 instances) of the training set as the cross validation set. Then, divide the cross validation set into 5 subsets of equal size (i.e. each containing 200 instances). Each subset in turn is used to validate the classifier trained on the remaining 4 subsets. So you have 5 trained SVMs and 5 validation subsets. The cross-validation accuracy is the average accuracy over the 5 validation subsets. Note that, do NOT use the build-in cross validation option in LIBSVM (i.e., option "-v 5"). You need to write some additional code/script necessary to divide the data and search over the parameters *C* and α. For both *C* and α, try the values in the following exponential steps: $(2^{-4}, 2^{-3},\ldots, 2^{7}, 2^{8})$. Note that you should try all possible pairs of values for parameters *C* and α. That is, try each of the 13 values listed above for *C* and α, and train and evaluate on the cross validation set for a total of 13x13=169 different models. Show a matrix of your cross validation results, where the entry (*i*, *j*) of the matrix corresponds to the classification accuracy on the **cross validation set** with the *i*th value of *C* and the *j*th value of α.
    2) Use **the whole training set** to train an SVM with the best *C* and α values you have found in the previous step. Use the trained SVM to classify the test set and show the classification accuracy.

Note: There is a guide for LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf and the README file in your downloaded package contains very helpful information for using LIBSVM.

What you need to turn in:

(1). 1-2 page summary report

(2). test results of your implementation

(3). your source program or script