Developers Manual

The code is written in C++ under Windows Cygwin, and it can also run on Linux or Mac OS X. The current vision is 1.0. In this version the program only provide two classes classification, and the multi-class classification will be added in version 1.1.

Quick Start

There are two executable binary files: svm-train and svm-predict. The usage is as follows:

```
$ ./svm-train.exe
Usage: svm-train [options] <train_file> <model_file>
options:
-c cost : the C in c-svc (default 1)
-t kernel_type : (default 2)
        0 -- linear
        1 -- polynomial : ( gama*x'*y + coef0 )^degree
        2 -- radial basis function : exp( -gama*|x-y|^2)
        3 -- sigmoid : tanh( gama*x'*y + coef0 )
-d degree : degree in polynomial kernel (default 3)
-g gama : gama in ploynomial, rbf, and sigmoid kernel (default 1/k, k = the demension of train vectors)
-r coef0 : coef0 in polynomial and sigmoid kernel (default 0)
-m cache size : cache memory size in MB (default 40)
-e epsilon : the epsilon for termination (default 0.001)

$ ./svm-predict.exe
Usage: svm-predict <test-file> <model-file> <predict-file>
```

The *train file* and *test file* have the same format (one training example per line):

classNumber1[space or tab]index1:value1[space or tab]index2:value2

The *class number* and the *index* must be integers. The *value* is a double. The *indexes* in each line must be in **ascending order**. In the predict-file are the values of $w^T x+b$ for all test data (one per line).

Use SVM in Your Own Program

Simply include the SVM.h, and call the four static methods of class SVM.

```
static void train( const SVM_Input &input, SVM_Model &outModel );
static double predict( const SVM_Model &model, const SpElem *x );
/**
  * load svm model from a file. Notice that you must delete xspace
  * and the end of the program
  */
static bool load( const char* infile, SVM_Model &outModel, SpElem *&xspace );
```

```
    /**
      * save svm model to a file.
      */
    static bool save( const char *outfile, const SVM_Model &model );
```

Add Your Own Kernel

Modify the code in KernelMatrix.h and KernelMatrix.cpp. For example, if you want to add a kernel called MyKernel, firstly add the class MyKernel in KernelMatrix.h

```
class MyKernel : public Kernel
{
public:
    double kernel( const SVM_Input *input, int i, int j ) const
    {
        …
    }
};
```

Secondly modify KUtil::kernel and KernelMatrix::KernelMatrix in KernelMatrix.cpp.