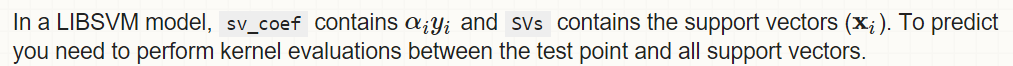
**Q: The output of training C-SVM is like the following. What do they mean?**   
  
optimization finished, #iter = 219   
nu = 0.431030   
obj = -100.877286, rho = 0.424632   
nSV = 132, nBSV = 107   
Total nSV = 132

obj is the optimal objective value of the dual SVM problem. rho is the bias term in the decision function sgn(w^Tx - rho). nSV and nBSV are number of support vectors and bounded support vectors (i.e., alpha\_i = C). nu-svm is a somewhat equivalent form of C-SVM where C is replaced by nu. nu simply shows the corresponding parameter. More details are in [libsvm document](http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf).



**Q: How could I generate the primal variable w of linear SVM?** 

Let's start from the binary class and assume you have two labels -1 and +1. After obtaining the model from calling svmtrain, do the following to have w and b:

w = model.SVs' \* model.sv\_coef;

b = -model.rho;

if model.Label(1) == -1

w = -w;

b = -b;

end

If you do regression or one-class SVM, then the if statement is not needed.

For multi-class SVM, we illustrate the setting in the following example of running the iris data, which have 3 classes

> [y, x] = libsvmread('../../htdocs/libsvmtools/datasets/multiclass/iris.scale');

> m = svmtrain(y, x, '-t 0')

m =

Parameters: [5x1 double]

nr\_class: 3

totalSV: 42

rho: [3x1 double]

Label: [3x1 double]

ProbA: []

ProbB: []

nSV: [3x1 double]

sv\_coef: [42x2 double]

SVs: [42x4 double]

sv\_coef is like:

+-+-+--------------------+

|1|1| |

|v|v| SVs from class 1 |

|2|3| |

+-+-+--------------------+

|1|2| |

|v|v| SVs from class 2 |

|2|3| |

+-+-+--------------------+

|1|2| |

|v|v| SVs from class 3 |

|3|3| |

+-+-+--------------------+

so we need to see nSV of each classes.

> m.nSV

ans =

3

21

18

Suppose the goal is to find the vector w of classes 1 vs 3. Then y\_i alpha\_i of training 1 vs 3 are

> coef = [m.sv\_coef(1:3,2); m.sv\_coef(25:42,1)];

and SVs are:

> SVs = [m.SVs(1:3,:); m.SVs(25:42,:)];

Hence, w is

> w = SVs'\*coef;

For rho,

> m.rho

ans =

1.1465

0.3682

-1.9969

> b = -m.rho(2);

because rho is arranged by 1vs2 1vs3 2vs3.

- Function: void svm\_get\_sv\_indices(const struct svm\_model \*model, int \*sv\_indices)

This function outputs indices of support vectors into an array called sv\_indices.

The size of sv\_indices is the number of support vectors and can be obtained by calling svm\_get\_nr\_sv.

Each sv\_indices[i] is in the range of [1, ..., num\_traning\_data].

- Function: int svm\_get\_nr\_sv(const struct svm\_model \*model)

This function gives the number of total support vector.

struct svm\_model

{

struct svm\_parameter param; /\* parameter \*/

int nr\_class; /\* number of classes, = 2 in regression/one class svm \*/

int l; /\* total #SV \*/

struct svm\_node \*\*SV; /\* SVs (SV[l]) \*/

double \*\*sv\_coef; /\* coefficients for SVs in decision functions (sv\_coef[k-1][l]) \*/

double \*rho; /\* constants in decision functions (rho[k\*(k-1)/2]) \*/

double \*probA; /\* pairwise probability information \*/

double \*probB;

int \*sv\_indices; /\* sv\_indices[0,...,nSV-1] are values in [1,...,num\_traning\_data] to indicate SVs in the training set \*/

/\* for classification only \*/

int \*label; /\* label of each class (label[k]) \*/

int \*nSV; /\* number of SVs for each class (nSV[k]) \*/

/\* nSV[0] + nSV[1] + ... + nSV[k-1] = l \*/

/\* XXX \*/

int free\_sv; /\* 1 if svm\_model is created by svm\_load\_model\*/

/\* 0 if svm\_model is created by svm\_train \*/

};