LIBSVM

A Library for Support Vector Machines

Information Management Lab.
Dept. of Industrial Engineering
Seoul National Univ.

Introduction

- LIBSVM is an integrated software for
  - support vector classification, (C-SVC, nu-SVC)
  - regression (epsilon-SVR, nu-SVR)
  - distribution estimation (one-class SVM)
  - multi-class classification.
- Main features of **LIBSVM**
  - Different SVM formulations
  - Efficient multi-class classification
  - Cross validation for model selection
  - Probability estimates
  - Various kernels (including precomputed kernel matrix)
  - Weighted SVM for unbalanced data
  - Both C++ and Java sources
  - GUI demonstrating SVM classification and regression
  - Python, R, MATLAB, Perl, Ruby, Weka, Common LISP, CLISP, Haskell, LabVIEW, and PHP interfaces. C# .NET code and CUDA extension is available.
    It's also included in some data mining environments: RapidMiner and PCP.
  - Automatic model selection which can generate contour of cross valiation accuracy.

download

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

Download LIBSVM

The current release (Version 3.1, April 2011) of **LIBSVM** can be obtained by downloading the zip file or tar.gz file. Please e-mail us if you have problems to download the file.

The package includes the source code of the library in C++ and Java, and a simple program for scaling training data. A README file with detailed explanation is provided. For **MS Windows** users, there is a subdirectory in the zip file containing binary executable files. Precompiled Java class archive is also included.

Please read the COPYRIGHT notice before using **LIBSVM**.

Example - input

classification, margin, learning

margin, learning, kernel

hidden, markov, viterbi, model, kernel

hidden, markov, margin, model

classification, markov, margin

Features: classification, margin, learning, kernel, hidden, markov, viterbi, model

<train.txt>

```
1  1:1  2:1  3:1  4:0  5:0  6:0  7:0  8:0
1  1:0  2:1  3:1  4:1  5:0  6:0  7:0  8:0
2  1:0  2:0  3:0  4:1  5:1  6:1  7:1  8:1
2  1:0  2:1  3:0  4:0  5:1  6:1  7:0  8:1
1  1:1  2:1  3:0  4:0  5:0  6:1  7:0  8:0
```

Example – learning options

- -s svm_type : set type of SVM (default 0)
  - 0 -- C-SVC
  - 1 -- nu-SVC
  - 2 -- one-class SVM
  - 3 -- epsilon-SVR
  - 4 -- nu-SVR
- -t kernel_type : set type of kernel function (default 2)
  - 0 -- linear: u'*v
  - 1 -- polynomial: (gamma*u'*v + coef0)^degree
  - 2 -- radial basis function: exp(-gamma*|u-v|^2)
  - 3 -- sigmoid: tanh(gamma*u'*v + coef0)
  - 4 -- precomputed kernel (kernel values in training_set_file)
- -d degree : set degree in kernel function (default 3)
- -g gamma : set gamma in kernel function (default 1/num_features)
- -r coef0 : set coef0 in kernel function (default 0)
- -c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
- -n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
- -p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
- -m cachesize : set cache memory size in MB (default 100)
- -e epsilon : set tolerance of termination criterion (default 0.001)
- -h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
- -b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
- -wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
- -v n : n-fold cross validation mode
- -q : quiet mode (no outputs)

Example - learning

```
D:₩workspace₩libsvm>java -classpath lib₩libsvm.jar svm_train -s 0 -t 0 train.txt model.txt
.*
optimization finished, #iter = 9
nu = 0.29331275720164607
obj = -0.7333332936417212, rho = -0.5999228395061729
nSV = 4, nBSV = 0
Total nSV = 4

D:₩workspace₩libsvm>
```

<model.txt>

```
svm_type c_svc
kernel_type polynomial
degree 3
gamma 0.125
coef0 0.0
nr_class 2
total_sv 5
rho -0.9596182839912281
label 1 2
nr_sv 3 2
SV
0.2894736842105263 1:1.0 2:1.0 3:1.0 4:0.0 5:0.0 6:0.0 7:0.0 8:0.0
0.7206477732793525 1:0.0 2:1.0 3:1.0 4:1.0 5:0.0 6:0.0 7:0.0 8:0.0
0.9898785425101212 1:1.0 2:1.0 3:0.0 4:0.0 5:0.0 6:1.0 7:0.0 8:0.0
-1.0 1:0.0 2:0.0 3:0.0 4:1.0 5:1.0 6:1.0 7:1.0 8:1.0
-1.0 1:0.0 2:1.0 3:0.0 4:0.0 5:1.0 6:1.0 7:0.0 8:1.0
```

Example - test

<test.txt>

```
1 1:1 2:1 3:1 4:0 5:0 6:0 7:0 8:0
1 1:0 2:1 3:1 4:0 5:1 6:0 7:0 8:0
1 1:1 2:0 3:0 4:1 5:0 6:1 7:0 8:0
2 1:0 2:0 3:0 4:1 5:1 6:1 7:1 8:1
2 1:0 2:1 3:0 4:0 5:1 6:1 7:2 8:1
```

```
D:₩workspace₩libsvm>java -classpath lib₩libsvm.jar svm_predict test.txt model.txt output.txt
Accuracy = 100.0% (5/5) (classification)

D:₩workspace₩libsvm>more output.txt
1.0
1.0
1.0
2.0
2.0
```

Kernel type

0 -- linear: $k(x, y) = x \cdot y$

1 -- polynomial: $k(x, y) = (x \cdot y + c)^d$

2 -- radial basis function: $k(x, y) = e^{-\Gamma * |x-y|^2}$

3 -- sigmoid: $k(x, y) = \tanh(\Gamma \cdot x \cdot y + c)$

4 -- precomputed kernel (kernel values in training_set_file)

<Precomputed kernel training set file example>

14891	0:01	12:26.0	13:39.6	03:45.0	04:31.1	06:15.1	07:44.3	07:26.7	08:37.3
14891	0:02	12:39.6	14:51.6	03:30.8	04:14.2	05:42.9	07:17.9	07:00.2	08:09.4
14837	0:03	01:45.0	02:30.8	04:22.7	04:08.2	08:18.1	09:11.5	07:12.2	08:16.6
15607	0:04	01:31.1	02:14.2	03:08.2	08:03.8	05:02.5	06:43.7	12:34.1	14:06.7
14837	0:05	02:15.1	02:42.9	06:18.1	04:02.5	13:05.5	13:31.9	07:03.4	08:13.7
14837	0:06	02:44.3	03:17.9	06:11.5	04:43.7	12:31.9	14:11.3	08:00.9	09:17.0
15607	0:07	01:26.7	02:00.2	03:12.2	09:34.1	05:03.4	07:00.9	15:11.8	16:38.0
15607	0:08	01:37.3	02:09.4	03:16.6	10:06.7	05:13.7	07:17.0	15:38.0	17:35.4
14837	0:09	02:10.3	02:52.0	04:59.9	04:31.0	09:45.7	10:41.5	07:41.9	08:53.1
15607	0:10	01:50.9	02:07.9	03:20.1	13:09.5	05:05.6	07:39.7	20:11.0	22:08.6
14891	0:11	14:44.6	16:20.6	03:45.4	04:33.6	06:11.4	07:46.1	07:24.0	08:38.1
14891	0:12	07:36.6	08:58.9	03:32.5	04:17.1	05:58.6	07:18.5	07:13.2	08:20.5
14891	0:13	09:59.1	11:32.8	03:24.7	04:29.6	05:32.2	07:02.2	07:27.5	08:36.8
15607	0:14	01:34.9	02:13.7	03:11.0	08:41.0	05:05.2	06:56.2	13:46.1	15:17.9
14891	0:15	08:40.9	10:06.8	03:25.0	04:09.4	05:39.1	07:00.8	07:00.3	08:06.7
15607	0:16	01:32.2	02:11.7	03:09.8	08:41.5	05:02.9	06:52.6	13:43.0	15:15.1
14891	0:17	09:40.0	11:12.7	03:15.6	04:12.2	05:14.0	06:40.5	07:02.2	08:09.2
14837	0:18	02:06.5	02:46.2	05:21.8	04:09.0	10:39.6	11:43.0	07:12.1	08:20.6
15607	0:19	01:49.8	02:32.4	03:13.4	08:00.1	05:12.6	06:54.0	12:38.7	14:10.2