M-SVM^2 user's guide

Yann Guermeur

November 16, 2010

Contents

1 Introduction **3**

2 Architecture of the software **3**
2.1 Programs 3
2.2 How to compile 3

3 Solving multi-class problems **3**
3.1 Four simple examples 3
3.2 Structure of the files containing the data 4
3.3 Training the M-SVM^2 4
3.4 Testing the M-SVM^2 5
3.5 Stopping criterion 7

4 General comments **8**

1 Introduction

This software is an implementation of the multi-class SVM (M-SVM) named M-SVM^2 introduced in [6] (the initial version of the paper can be found in the subdirectory Tech). It is written in C ANSI, and thus can be used under the various releases of UNIX, Linux, IRIX, etc. It is the property of CNRS and is governed by the CeCILL-B license as circulated by CEA, CNRS and INRIA at the following URL `http://www.cecill.info`. It has been registered at the "Agence pour la Protection des Programmes" (APP) under number `IDDN.FR.001.370001.000.S.P.2010.000.30000`. The training algorithm is a variant of Rosen's gradient projection method [9] which implements a decomposition strategy. Notations used in this guide are those of [6].

2 Architecture of the software

2.1 Programs

This application is made up of two main programs: `train_SVM` performs the training of the M-SVM, whereas `eval_SVM` is used to assess the model performance on the training and test sets.

2.2 How to compile

`train_SVM` and `eval_SVM` can be compiled thanks to the commands:
`compile_train_SVM` and `compile_eval_SVM`
(the corresponding makefiles are in the subdirectory `make`).

3 Solving multi-class problems

3.1 Four simple examples

A simple way to become familiar with the use of the software consists in running it on the four examples provided, named "Gaussians", "iris", "Lee", and "toy". The corresponding training and test sets are located in the subdirectory `Data`. The first of these examples is the three-Gaussian problem that illustrates [5]. The second one corresponds to the iris data set of Fisher, available at the UCI repository [1]. The third one is the numerical example used in [7]. At last, the fourth problem is a 3-class toy

problem borrowed from [3][1]. In order to select any of the four problems, it suffices to use the corresponding script, either `configure.gaussians`, `configure.iris`, `configure.Lee` or `configure.toy`. Once this is done, the files `Fichcom/train_SVM.com` and `Fichcom/eval_SVM.com` (see below) contain the appropriate parameters, and the file of dual variables is initialized with a feasible solution (by default the null vector). Suffice it to use the command `execute_train_SVM` to start training. While training is underway, the command `execute_eval_SVM` can be used to assess its progress and compute the training and test performance.

3.2 Structure of the files containing the data

The files containing the data must be text files, with a specific structure. We illustrate this structure on Elisseeff's toy problem. The name of the corresponding file is `Data/toy.app`.

`1000` ⟵ number of points in the set
`2` ⟵ number of predictors (components of the vectors coding the input data)
`0.781323 0.298303 1` ⟵ description of the first example: two components of the input vector plus the label of the category, here 1.
`...`

3.3 Training the M-SVM[2]

Training is initiated with the command
`execute_train_SVM`

In order to specify the nature of the problem to be solved, the file
`Fichcom/train_SVM.com`
must preliminary be filled. It is made up of seven lines. Its structure, illustrated on the aforementioned toy problem, is as follows:

`3` ⟵ number of categories for the problem
`1` ⟵ nature of the kernel
`100.0` ⟵ value of the soft margin parameter C
`4` ⟵ size of the *chunk* (see the technical documentation)
`Data/toy.app` ⟵ name of the file where the training data is stored

[1]The programs to generate the corresponding data sets and display them are located in the subdirectory `Toy`.

`Alpha/toy.alpha` ⟵ name of the file containing the initial values of the dual variables α_{ik}

`Alpha/toy.alpha` ⟵ name of the file where the updated values of the dual variables α_{ik} are stored during training

Currently, only three types of kernels are implemented: a linear kernel, a RBF (Gaussian) kernel and a polynomial one. A value of 1 for the "nature of the kernel" corresponds to a linear kernel (Euclidean inner product), whereas a value of 2 corresponds to a Gaussian kernel and a value of 3 corresponds to a polynomial kernel. The parameters: width of the Gaussian kernel and degree of the polynomial kernel, can be changed by modifying adequately the functions `gaussian` and `polynomial` in the file `algebre.c`. This file can also be used to store the functions corresponding to the new kernels a user could find useful to add.

A feasible solution of the dual problem (which can be used to initialize the vector of dual variables α) is the null vector. Care must be taken to the fact that the dummy variables α_{iy_i} must always remain equal to 0.

During training, the following pieces of information are displayed:

`*** Iteration: 1000` ⟵ number of gradient ascent steps since the beginning of training

`Number of support vectors: 632` ⟵ number of training examples for which at least one of the dual variables α_{ik} is positive (different from 0)

Note that the current value of the (dual) objective function can also be displayed. This is obtained by uncommenting the call to the function `compute_objective_function` in the `main` function of `train_SVM`. Furthermore, the file of dual variables is regularly copied in the `Save_alpha` subdirectory.

3.4 Testing the M-SVM^2

Testing is initiated with the command

`execute_eval_SVM`

The structure of the file

`Fichcom/eval_SVM.com`

containing the parameters used by the program `eval_SVM`, is pretty similar to the structure of the file `Fichcom/train_SVM.com`. Here is an example,

corresponding once more to the toy problem:

`3` ⟵ number of categories for the problem
`1` ⟵ nature of the kernel
`100.0` ⟵ value of the soft margin parameter C
`Data/toy.app` ⟵ name of the file where the training data is stored
`Data/toy.test` ⟵ name of the file where the test data is stored
`Save_alpha/toy.alpha` ⟵ name of the file where the values of the dual variables α_{ik} are read
`Data/toy.output` ⟵ name of the file where the outputs of the M-SVM^2 will be stored

The program `execute_eval_SVM` displays several pieces of information regarding the feasibility of the current solution, the values of the objective functions and the training and test performance. Once more, we illustrate its behaviour on the toy problem.

```
The maximum value among the dual variables is:  3.407756e+02
```
⟵ largest value of the dual variables α_{ik} (which are unbounded)

```
Satisfaction of the equality constraints:

 8.931949e-10
-2.243681e-10
-6.880256e-10
```
⟵ values of the left-hand sides of the $Q-1$ equations defining the equality constraints of the training problem (dual formulation):

$$\forall k \in [\![1, Q-1]\!], \ \sum_{i=1}^{m} \sum_{l=1}^{Q} \left(\frac{1}{Q} - \delta_{k,l} \right) \alpha_{il} = 0.$$

```
Vector b before centering

b_1 = -9.008381e-01
b_2 = -8.480913e-01
b_3 =  1.125504e+00

Sum of the components of b:  -6.234254e-01
```

⟵ value of the vector $\mathbf{b} = (b_k)_{1 \leqslant k \leqslant Q}$ of the biases of the affine model. At the optimum, the sum of its components is null.

`1^T alpha = 2CQ 1^T xi = 9.690762e+04` ⟵ value of the quantities:

$$1_{Qm}^T \alpha = 2CQ 1_{Qm}^T \xi$$

```
            Dual objective function:  3.540782e+04
Estimated primal objective function:  1.801829e+05
(ratio 19.65%)
```

⟵ These three lines provide the user with the value of the dual objective function $J_{\text{M-SVM}^2,\text{d}}(\alpha)$, an upper bound on the value of the primal objective function and the value of the corresponding ratio (its limit as the number of iterations goes to infinity is 1).

```
Recognition rate:  62.40%

Confusion matrix:
```

164	91	0
0	351	153
127	5	109

```
Matthews coefficients:

C1:  0.45
C2:  0.51
C3:  0.24
```

⟵ These figures characterize the training performance. The definition of the Pearson's/Matthews' correlation coefficients can be found in [8]. Details regarding the test performance follow those regarding the training performance. They are displayed in the same way.

3.5 Stopping criterion

This application has been designed to process very large data sets. This is the reason why the training procedure incorporates a decomposition method. As a consequence, the cpu time needed to check the Kuhn-Tucker optimality conditions [4] is far superior to that of a gradient step. Thus, in order not to slow down the training algorithm drastically, no test is made in the program `train_SVM` regarding the optimality of the current feasible solution α. There

is only a variable named `nb_iter` which limits the number of iterations (steps in the gradient ascent). Its value can be set utterly arbitrarily. The computation of the feasibility gap [2] is performed by the program `eval_SVM` (see Section 3.4). A rule of sumb is that one can stop training as soon as the ratio exceeds 98.00%.

4 General comments

This application is intended to be used for academic research purposes only. It should evolve frequently. Please, feel free to report any suggestion you could have to improve the programs or this document to the following address: `Yann.Guermeur@loria.fr`

Acknowledgments Thanks are due to A. Iouditski for interesting discussions on the programming of M-SVMs.

References

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, 2000.

[3] A. Elisseeff. *Etude de la complexité et contrôle de la capacité des systèmes d'apprentissage : SVM multi-classe, réseaux de régularisation et réseaux de neurones multicouches*. PhD thesis, ENS Lyon, 2000. (in French).

[4] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Chichester, second edition, 1987.

[5] Y. Guermeur, M. Maumy, and F. Sur. Model selection for multi-class SVMs. In *ASMDA'05*, pages 507–517, 2005.

[6] Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius-margin bound applies. *INFORMATICA*, 2010. (conditionally accepted).

[7] Y. Lee and Z. Cui. Characterizing the solution path of multicategory support vector machines. *Statistica Sinica*, 16(2):391–409, 2006.

[8] B.W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta*, 405:442–451, 1975.

[9] J.B. Rosen. The gradient projection method for nonlinear programming. Part I. Linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, 1960.