Journal of Machine Learning Research 12 (2011) 2293-2296 Submitted 5/10; Revised 3/11; Published 7/11

MSVMpack: A Multi-Class Support Vector Machine Package

Fabien Lauer FABIEN.LAUER@LORIA.FR
Yann Guermeur YANN.GUERMEUR@LORIA.FR
LORIA – Equipe ABC
Campus Scientifique, BP 239
54506 Vandœuvre-lès-Nancy cedex, France

Editor: Mikio Braun

Abstract

This paper describes MSVMpack, an open source software package dedicated to our generic model of *multi-class* support vector machine. All four multi-class support vector machines (M-SVMs) proposed so far in the literature appear as instances of this model. MSVMpack provides for them the first unified implementation and offers a convenient basis to develop other instances. This is also the first parallel implementation for M-SVMs. The package consists in a set of command-line tools with a callable library. The documentation includes a tutorial, a user's guide and a developer's guide.

Keywords: multi-class support vector machines, open source, C

1. Introduction

In the framework of polytomy computation, a multi-class support vector machine (M-SVM) is a support vector machine (SVM) dealing with all the categories simultaneously. Four M-SVMs can be found in the literature: the models of Weston and Watkins (1998), Crammer and Singer (2001), Lee et al. (2004), and the M-SVM^2 of Guermeur and Monfrini (2011). The proposed software implements them all in a single package named MSVMpack. Its design paves the way for the implementation of our generic model of M-SVM and the integration of additional functionalities such as model selection algorithms. The current version offers a parallel implementation with the possibility to use custom kernels. This software package is available for Linux under the terms of the GPL at `http://www.loria.fr/~lauer/MSVMpack/` and provides two command-line tools with a C application programming interface without dependencies beside a linear programming solver.

2. Multi-Class Support Vector Machines

We consider Q-category classification problems where $\mathcal{X}$ is the description space and the set $\mathcal{Y}$ of the categories can be identified with $[\![1,Q]\!]$. Let κ be a real-valued positive type function (Berlinet and Thomas-Agnan, 2004) on $\mathcal{X}^2$ and let $(\mathbf{H}_\kappa, \langle\cdot,\cdot\rangle_{\mathbf{H}_\kappa})$ be the corresponding reproducing kernel Hilbert space. Let $\mathcal{H} = \mathbf{H}_\kappa^Q$ and $\mathcal{H} = (\mathbf{H}_\kappa + \{1\})^Q$. $\mathcal{H}$ is the class of functions $h = (h_k)_{1\leqslant k\leqslant Q}$ from $\mathcal{X}$ to $\mathbb{R}^Q$ that can be written as $h(\cdot) = \overline{h}(\cdot) + b = \left(\overline{h}_k(\cdot) + b_k\right)_{1\leqslant k\leqslant Q}$, where $\overline{h} = \left(\overline{h}_k\right)_{1\leqslant k\leqslant Q} \in \mathcal{H}$ and $b = (b_k)_{1\leqslant k\leqslant Q} \in \mathbb{R}^Q$. A function h assigns the category y to x if and only if $y = \operatorname{argmax}_{1\leqslant k\leqslant Q} h_k(x)$ (cases of ex æquo are dealt with by introducing a dummy category). $\mathcal{H}$ is endowed with the norm

Reference	M-SVM type	M	p	K_1	K_2	K_3
Weston and Watkins (1998)	WW	I_{Qm}	1	1	1	0
Crammer and Singer (2001)	CS	$\frac{1}{Q-1}I_{Qm}$	1	1	1	1
Lee et al. (2004)	LLW	I_{Qm}	1	0	$\frac{1}{Q-1}$	0
Guermeur and Monfrini (2011)	MSVM2	$M^{(2)}$	2	0	$\frac{1}{Q-1}$	0

Table 1: Specifications of the M-SVMs with their type as used by the MSVMpack interface.

$\|\cdot\|_{\bar{\mathcal{H}}}$ given by:

$$\forall \bar{h} \in \bar{\mathcal{H}},\ \left\|\bar{h}\right\|_{\bar{\mathcal{H}}} = \sqrt{\sum_{k=1}^{Q} \langle \bar{h}_k, \bar{h}_k \rangle_{\mathbf{H}_\kappa}} = \left\| \left(\left\|\bar{h}_k\right\|_{\mathbf{H}_\kappa} \right)_{1 \leqslant k \leqslant Q} \right\|_2 .$$

With these definitions at hand, our generic definition of a Q-category M-SVM is:

Definition 1 (Generic model of M-SVM, Definition 4 in Guermeur, forthcoming) *Let $((x_i, y_i))_{1 \leqslant i \leqslant m} \in (\mathcal{X} \times [\![1, Q]\!])^m$ and $\lambda \in \mathbb{R}_+^*$. Let $\xi \in \mathbb{R}^{Qm}$ be a vector such that for $(i,k) \in [\![1,m]\!] \times [\![1,Q]\!]$, ξ_{ik} is its component of index $(i-1)Q+k$, with $(\xi_{iy_i})_{1 \leqslant i \leqslant m} = 0_m$. A Q-*category M-SVM *is a classifier obtained by solving a convex quadratic programming (QP) problem of the form*

$$\min_{h,\xi} J(h,\xi) = \|M\xi\|_p^p + \lambda \left\|\bar{h}\right\|_{\bar{\mathcal{H}}}^2$$

$$s.t. \begin{cases} \forall i \in [\![1,m]\!],\ \forall k \in [\![1,Q]\!] \setminus \{y_i\},\ K_1 h_{y_i}(x_i) - h_k(x_i) \geqslant K_2 - \xi_{ik} \\ \forall i \in [\![1,m]\!],\ \forall (k,l) \in ([\![1,Q]\!] \setminus \{y_i\})^2,\ K_3 (\xi_{ik} - \xi_{il}) = 0 \\ \forall i \in [\![1,m]\!],\ \forall k \in [\![1,Q]\!] \setminus \{y_i\},\ (2-p)\xi_{ik} \geqslant 0 \\ (1-K_1) \sum_{k=1}^{Q} h_k = 0, \end{cases}$$

where $p \in \{1,2\}$, $(K_1, K_3) \in \{0,1\}^2$, $K_2 \in \mathbb{R}_+^*$ and the matrix M is such that $\|M\xi\|_p$ is a norm of ξ.

Extending to matrices the notation used to designate the components of ξ and using δ to denote the Kronecker symbol, let us define the general term of $M^{(2)} \in \mathcal{M}_{Qm,Qm}(\mathbb{R})$ as:

$$m^{(2)}_{ik,jl} = (1 - \delta_{y_i,k})(1 - \delta_{y_j,l})\, \delta_{i,j} \left(\delta_{k,l} + \frac{\sqrt{Q}-1}{Q-1} \right).$$

This allows us to summarize the characteristics of the four M-SVMs in Table 1. The potential of the generic model is discussed in Guermeur (forthcoming).

3. The Software Package

MSVMpack includes a C application programming interface (API) and two command-line tools: one for training an M-SVM and one for making predictions with a trained M-SVM. The following discusses some algorithmic issues before presenting these tools and the API.

3.1 Training Algorithm

As in the bi-class case, an M-SVM is trained by solving the Wolfe dual of its instantiation of the QP problem in Definition 1. The corresponding dual variables α_{ik} are the Lagrange multipliers of the constraints of correct classification. The implemented QP algorithm is based on the Frank-Wolfe method (Frank and Wolfe, 1956), in which each step of the descent is obtained as the solution of a linear program (LP). The LP solver included in MSVMpack is `lp_solve` (Berkelaar et al., 2009). In order to make it possible to process large data sets, a decomposition method is applied and only a small subset of the data is considered in each iteration.

Let J_d be the dual objective function and let $\alpha = (\alpha_{ik})$ be a (feasible) solution of the dual problem obtained at some point of the training procedure. The quality of α is measured thanks to the computation of an upper bound $U(\alpha)$ on the optimum $J(h^*, \xi^*) = J_d(\alpha^*)$ that goes to this optimum. The stopping criterion is defined as a large enough value for $J_d(\alpha)/U(\alpha)$. In MSVMpack, the bound $U(\alpha)$ is obtained by solving the primal problem with $\bar{h}$ being the function associated with the current α. This partial optimization requires little computation except in the case of the M-SVM^2, for which another QP problem has to be solved. However, the computational burden may increase for a large number of classes (e.g., $Q > 20$).

3.2 Practical Use and Experiments

In its most simple form, the command line '`trainmsvm trainingdata -m WW`' is used to train an M-SVM, where the `-m` flag allows one to choose the type of M-SVM model according to Table 1. Then, this model can be applied to a test set by using '`predmsvm testdata`'. The complete list of options and parameters for these command-line tools can be found in the documentation or simply obtained by calling them without argument.

Table 2 shows a comparison of MSVMpack with other implementations of M-SVMs on a subset of the USPS database with 500 instances from 10 classes and the whole CB513 data set with 84119 instances from 3 classes. For the latter, the numbers reflect the average error and total times over a 5-fold cross validation, and the implementations that failed due to a lack of memory are not included in the Table. We refer the reader to the documentation for the details of the experimental setup and additional comparisons on other data sets.

3.3 Calling the Library from Other Programs

The "Developer's guide" section of the documentation presents the API reference and an example program including MSVMpack functionalities through this API. The library defines specific data structures for M-SVM models and data sets. It also provides wrapper functions, which act according to the M-SVM model type, for example, call the corresponding training function. The standard workflow for a train-and-test sequence is: call `MSVM_make_model()` to initialize the model; call `MSVM_make_dataset()` for each data set to load; call `MSVM_train()` to train the model; and call `MSVM_classify_set()` to test the trained classifier.

4. Ongoing and Future Developments

MSVMpack implements the four M-SVMs proposed in the literature. Current work focuses on the explicit implementation of our generic model of M-SVM, which will make it possible to study new machines thanks to a simple choice of the values of the hyperparameters M, p, and $(K_t)_{1 \leqslant t \leqslant 3}$. Future

Data set	M-SVM	Software	Test error	Training time	Testing time
USPS_500	WW	Spider (Matlab)	10.20 %	4m 19s	0.2s
$Q = 10$		BSVM (C++)	10.00 %	0.2s	0.1s
$m = 500$		MSVMpack (C)	10.40 %	2.5s	0.1s
$\mathcal{X} \subset \mathbb{R}^{256}$	CS	MCSVM (C)	9.80 %	0.5s	0.3s
test set:		MSVMpack	9.80 %	30s	0.1s
$m = 500$	LLW	SMSVM (R)	12.00 %	5m 58s	0.1s
		MSVMpack	11.40 %	1m 22s	0.1s
	$MSVM^2$	MSVMpack	12.00 %	22s	0.1s
CB513	WW	BSVM	23.96 %	9h 48m 40s	46m 29s
$Q = 3$		MSVMpack	23.72 %	1h 05m 11s	1m 51s
$m = 84119$	CS	MCSVM	23.55 %	22h 52m 10s	2h 08m 33s
$\mathcal{X} \subset \mathbb{Z}^{260}$		MSVMpack	23.63 %	1h 00m 36s	2m 06s
test:	LLW	MSVMpack	25.65 %	1h 14m 21s	2m 33s
5-fold CV	$MSVM^2$	MSVMpack	23.47 %	6h 44m 50s	2m 49s

Table 2: Relative performance of different M-SVM implementations on two data sets.

work will consider including automatic tuning procedures for the regularization parameter λ, and relaxing the hypothesis on the norm of the penalizer.

References

M. Berkelaar, K. Eikland, and P. Notebaert. *An Open Source (Mixed-Integer) Linear Programming System*, 2009. Software available at `http://lpsolve.sourceforge.net/`.

A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Kluwer Academic Publishers, Boston, 2004.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1–2):95–110, 1956.

Y. Guermeur. A generic model of multi-class support vector machine. *International Journal of Intelligent Information and Database Systems*, forthcoming.

Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius-margin bound applies. *Informatica*, 22(1):73–96, 2011.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.