Technical documentation of the WW-M-SVM

December 18, 2023

Contents

1 Introduction

This technical documentation describes the implementation of the multi-class SVM (M-SVM) introduced by Weston and Watkins (WW-M-SVM) [9]. This machine can be seen as an instance of the generic model described in [6].

2 Theoretical framework

Let us consider a Q-category pattern classification problem, with $Q \geqslant 3$. We make the hypothesis that the covariates x live in a domain $\mathcal{X}$. The set of categories $\mathcal{Y}$ is identified with the set of indexes of the categories, i.e., $[\![1, Q]\!]$ (no structure is assumed on $\mathcal{Y}$). Let $d_m = \{(x_i, y_i) : 1 \leqslant i \leqslant m\}$ be a set of m labelled examples ($d_m \in (\mathcal{X} \times \mathcal{Y})^m$). Let κ be a real-valued positive type function/kernel [1] on $\mathcal{X}^2$ and let $\left(\mathbf{H}_\kappa, \langle \cdot, \cdot \rangle_{\mathbf{H}_\kappa}\right)$ be its reproducing kernel Hilbert space (RKHS). The architecture considered to perform the discriminant analysis, i.e., the function class $\mathcal{H}$ on which function selection is performed based on d_m, is the set of functions $h = (h_k)_{1 \leqslant k \leqslant Q}$ from $\mathcal{X}$ into $\mathbb{R}^Q$ given by:

$$\forall x \in \mathcal{X},\ h(x) = \left(\langle \bar{h}_k, \kappa_x \rangle_{\mathbf{H}_\kappa} + b_k\right)_{1 \leqslant k \leqslant Q},$$

where $\left(\bar{h}_k\right)_{1 \leqslant k \leqslant Q} \in \mathbf{H}_\kappa^Q$ and $(b_k)_{1 \leqslant k \leqslant Q} \in \mathbb{R}^Q$.

We denote $\mathbb{R}^{Qm}(d_m)$ the subset of $\mathbb{R}^{Qm}$ made up of the vectors $v = (v_t)_{1 \leqslant t \leqslant Qm}$ satisfying: $\left(v_{(i-1)Q+y_i}\right)_{1 \leqslant i \leqslant m} = 0_m$. For the sake of simplicity, the components of the vectors of $\mathbb{R}^{Qm}(d_m)$ are written with two indices, i.e., v_{ik} in place of $v_{(i-1)Q+k}$, for i in $[\![1, m]\!]$ and k in $[\![1, Q]\!]$. For n in $\mathbb{N}^*$, let $\mathcal{M}_{n,n}(\mathbb{R})$ be the algebra of $n \times n$ matrices over $\mathbb{R}$. Let $\mathcal{M}_{Qm,Qm}(d_m)$ be the subset of $\mathcal{M}_{Qm,Qm}(\mathbb{R})$ made up of the symmetric matrices $M = (m_{tu})_{1 \leqslant t,u \leqslant Qm}$ satisfying: $\forall j \in [\![1, m]\!]$, $\left(m_{t,(j-1)Q+y_j}\right)_{1 \leqslant t \leqslant Qm} = 0_{Qm}$. Once more for the sake of simplicity, the components of the matrices of $\mathcal{M}_{Qm,Qm}(d_m)$ are written with four indices, i.e., $m_{ik,jl}$ in place of $m_{(i-1)Q+k,(j-1)Q+l}$, for (i, j) in $[\![1, m]\!]^2$ and (k, l) in $[\![1, Q]\!]^2$.

3 Training the machine

3.1 Analytical expression of the primal

The *training set* d_m being given, The optimal hyperplanes are solution of the following quadratic programming (QP) problem:

Problem 1

$$\min_{h\in\mathcal{H},\ \xi\in\mathbb{R}^{Qm}(d_m)} \left\{ \frac{1}{2}\sum_{k=1}^{Q} \|\bar{h}_k\|^2_{\mathbf{H}_\kappa} + C\sum_{i=1}^{m}\sum_{k=1}^{Q}\xi_{ik} \right\}$$

subject to:

$$\begin{cases} h_{y_i}(x_i) - h_k(x_i) \geqslant 1 - \delta_{y_i,k} - \xi_{ik} & (1\leqslant i\leqslant m),\ (1\leqslant k\leqslant Q) \\ \xi_{ik} \geqslant 0 & (1\leqslant i\leqslant m),\ (1\leqslant k\leqslant Q) \end{cases},$$

where δ is the Kronecker symbol and the parameter C (the *soft margin parameter*), specifies the desired trade-off between training performance and capacity control. Its value, fixed *a priori*, is taken in $(0,+\infty]$. The objective function, hereafter denoted J_p, can be directly connected to a result of uniform convergence of the empirical risk (see for instance [5]).

Solving Problem 1 amounts to finding a saddle point of the following Lagrangian function:

$$L(h,\xi,\alpha,\beta) = \frac{1}{2}\sum_{k=1}^{Q}\|\bar{h}_k\|^2_{\mathbf{H}_\kappa} + C\sum_{i=1}^{m}\sum_{k=1}^{Q}\xi_{ik}$$

$$-\sum_{i=1}^{m}\sum_{k=1}^{Q}\alpha_{ik}\left\{h_{y_i}(x_i) - h_k(x_i) - 1 + \delta_{y_i,k} + \xi_{ik}\right\} - \sum_{i=1}^{m}\sum_{k=1}^{Q}\beta_{ik}\xi_{ik}$$

where $\alpha \in \mathbb{R}_+^{Qm}(d_m)$ and $\beta \in \mathbb{R}_+^{Qm}(d_m)$ are the vectors of the Lagrange multipliers. For technical reasons, among which the fact that the RKHS can be infinite-dimensional, Problem 1 is solved in the form of its Wolfe dual.

3.2 Analytical expression of the dual

At the optimum, the gradient of the Lagrangian function with respect to the primal variables is null. As a consequence:

$$\nabla_{\bar{h}_k} L(h^*,\xi^*,\alpha^*,\beta^*) = 0_{\mathbf{H}_\kappa} \quad (1\leqslant k\leqslant Q)$$

and thus:

$$\bar{h}_k^* = \sum_{\{i:y_i=k\}} \sum_{l\neq k} \alpha_{il}^* \kappa_{x_i} - \sum_{\{i:y_i\neq k\}} \alpha_{ik}^* \kappa_{x_i} \ \ (1 \leqslant k \leqslant Q). \tag{1}$$

Note that a consequence of Equation (1) is that $\sum_{k=1}^{Q} \bar{h}_k^* = 0_{\mathbf{H}_\kappa}$. Similarly,

$$\frac{\partial}{\partial b_k} L\left(h^*, \xi^*, \alpha^*, \beta^*\right) = 0 \ \ (1 \leqslant k \leqslant Q)$$

and consequently:

$$\sum_{\{i:y_i=k\}} \sum_{l\neq k} \alpha_{il}^* - \sum_{\{i:y_i\neq k\}} \alpha_{ik}^* = 0 \ \ (1 \leqslant k \leqslant Q).$$

From

$$\frac{\partial}{\partial \xi_{ik}} L\left(h^*, \xi^*, \alpha^*, \beta^*\right) = 0 \ \ (1 \leqslant i \leqslant m), \ (1 \leqslant k \leqslant Q), \ k \neq y_i$$

we get:

$$\alpha_{ik}^* + \beta_{ik}^* = C, \ \ (1 \leqslant i \leqslant m), \ (1 \leqslant k \leqslant Q), \ k \neq y_i,$$

with the consequence that

$$\alpha_{ik}^* \in (0, C) \Longrightarrow b_{y_i}^* - b_k^* = 1 - \langle \bar{h}_{y_i}^* - \bar{h}_k^*, \kappa_{x_i} \rangle_{\mathbf{H}_\kappa}. \tag{2}$$

These equations make it possible to eliminate the primal variables in the expression of the Lagrangian function at the optimum. This gives us the expression of the objective function of the dual problem (objective function to be minimized):

$$J_d(\alpha) = \frac{1}{2} \alpha^T H \alpha - 1_{Qm}^T \alpha,$$

where $H = (h_{ik,jl})_{1\leqslant i,j\leqslant m, 1\leqslant k,l\leqslant Q}$ is the matrix of $\mathcal{M}_{Qm,Qm}(d_m)$ with general term

$$h_{ik,jl} = \left(\delta_{y_i,y_j} - \delta_{y_i,l} - \delta_{y_j,k} + \delta_{k,l}\right) \kappa(x_i, x_j) \tag{3}$$

and 1_{Qm} is the vector of $\mathbb{R}^{Qm}(d_m)$ of general term $1_{ik} = 1 - \delta_{y_i,k}$. The expression of the dual problem is thus the following one:

Problem 2

$$\min_{\alpha \in \mathbb{R}_+^{Qm}(d_m)} J_d(\alpha)$$

subject to:

$$\begin{cases} 0 \leqslant \alpha_{ik} \leqslant C & (1 \leqslant i \leqslant m), \ (1 \leqslant k \leqslant Q) \\ \sum_{\{i:y_i=k\}} \sum_{l\neq k} \alpha_{il} - \sum_{\{i:y_i\neq k\}} \alpha_{ik} = 0 & (1 \leqslant k \leqslant Q-1) \end{cases}.$$

4 Assessment of the machine

The assessment of the machine involves the computation of the values of the primal variables and the two objective functions (to check that the duality gap is indeed negligible).

4.1 Computation of the primal variables

The analytical expression of the components of vector $\bar{h}^* = \left(\bar{h}_k^*\right)_{1 \leqslant k \leqslant Q}$ is provided by Formula (1). The value of the vector $b^* = \left(b_k^*\right)_{1 \leqslant k \leqslant Q}$ can be derived from the Karush–Kuhn–Tucker (KKT) conditions (Formula (2)), by means of the computation of the gradient of the objective function of the dual problem. The algebraic expression of this gradient is $\nabla J_d(\alpha) = H\alpha - 1_{Qm}$ (so that it belongs to $\mathbb{R}^{Qm}\,(d_m)$). Its components are given by:

$$\forall (i,k) \in [\![1,m]\!] \times [\![1,Q]\!], \quad \frac{\partial}{\partial \alpha_{ik}} J_d(\alpha) =$$

$$\sum_{\{j:y_j=y_i\}} \sum_{l \neq y_i} \alpha_{jl} \kappa\left(x_j, x_i\right) - \sum_{\{j:y_j \neq y_i\}} \alpha_{jy_i} \kappa\left(x_j, x_i\right) - \sum_{\{j:y_j=k\}} \sum_{l \neq k} \alpha_{jl} \kappa\left(x_j, x_i\right) + \sum_{\{j:y_j \neq k\}} \alpha_{jk} \kappa\left(x_j, x_i\right) + \delta_{y_i,k} - 1.$$

Combining this formula with Formula (1) produces at the optimum

$$\forall (i,k) \in [\![1,m]\!] \times [\![1,Q]\!], \quad \frac{\partial}{\partial \alpha_{ik}} J_d\left(\alpha^*\right) = \langle \bar{h}_{y_i}^* - \bar{h}_k^*, \kappa_{x_i} \rangle_{\mathbf{H}_\kappa} + \delta_{y_i,k} - 1, \quad (4)$$

whose substitution into Formula (2) finally provides the desired expression of the components of vector b^*:

$$\alpha_{ik}^* \in (0, C) \Longrightarrow b_{y_i}^* - b_k^* = -\frac{\partial}{\partial \alpha_{ik}} J_d\left(\alpha^*\right) \quad (5)$$

(keeping in mind that without loss of generality, we can enforce the constraint $\sum_{k=1}^{Q} b_k^* = 0$). With the vectors $\bar{h}^*$ and b^* at hand, the value of the vector ξ^* is obtaine once more through a direct application of the KKT conditions:

$$\alpha_{ik}^* = C \Longrightarrow \xi_{ik}^* = \left(1 + h_k^*\left(x_i\right) - h_{y_i}^*\left(x_i\right)\right)_+. \quad (6)$$

4.2 Computation of the objective functions

The quadratic terms of both functions are equal, i.e.,

$$\sum_{k=1}^{Q} \|\bar{h}_k^*\|_{\mathbf{H}_\kappa}^2 = {\alpha^*}^T H \alpha^*.$$

5 Solving the dual problem

To solve the dual problem, one can use the Frank-Wolfe algorithm [4]. The principle of this linearization method is the following:

5.1 The Frank-Wolfe algorithm

We are interested in problems with linear constraints of the form:

$$\min_t f(t)$$

subject to:

$$\begin{cases} At = b \\ t \geqslant 0 \end{cases}.$$

The method is iterative and generates, starting from a feasible solution $t^{(0)}$, a series of points $t^{(0)}, t^{(1)}, \ldots, t^{(n)}, \ldots$ where, for each n, $t^{(n+1)}$ is deduced from $t^{(n)}$ as follows:

(1) solve the linear programming (LP) problem $LP(t^{(n)})$ given by:

$$\min_u \left\{ \nabla f(t^{(n)})^T u \right\}$$

subject to:

$$\begin{cases} Au = b \\ u \geqslant 0 \end{cases}.$$

(2) Let $u^{(n)}$ be an extreme point of the polyhedron which is an optimal solution of $LP(t^{(n)})$. Then $t^{(n+1)}$ is chosen so as to minimize f on the interval $\left[t^{(n)}, u^{(n)}\right]$.

5.2 Application to the WW-M-SVM

Applying this algorithm to train the WW-M-SVM raises no difficulties. One can for instance choose $\alpha^{(0)} = 0_{Qm}$. The linear program to be solved is:

Problem 3

$$\min_{\gamma \in \mathbb{R}_+^{Qm}(d_m)} \left\{ \nabla J_d(\alpha^{(n)})^T \gamma \right\}$$

subject to:

$$\begin{cases} 0 \leqslant \gamma_{ik} \leqslant C & (1 \leqslant i \leqslant m),\ (1 \leqslant k \leqslant Q) \\ \sum_{\{i:y_i=k\}} \sum_{l \neq k} \gamma_{il} - \sum_{\{i:y_i \neq k\}} \gamma_{ik} = 0 & (1 \leqslant k \leqslant Q-1) \end{cases}$$

with

$$\nabla J_d(\alpha^{(n)})^T \gamma = {\alpha^{(n)}}^T H \gamma - 1_{Qm}^T \gamma.$$

Let $\theta^{(n)} \in [0,1]$ be the coefficient of the optimal convex combination between $\alpha^{(n)}$ and $\gamma^{(n)}$, i.e.,

$$\theta^{(n)} = \text{Argmin}_{\theta \in [0,1]} J_d \left((1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right).$$

The analytical expression of $\theta^{(n)}$ can be obtained as follows:

$$J_d \left((1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right) =$$

$$\frac{1}{2} \left\{ (1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right\}^T H \left\{ (1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right\} - 1_{Qm}^T \left\{ (1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right\}.$$

$$\frac{\partial}{\partial \theta} J_d \left((1-\theta)\alpha^{(n)} + \theta \gamma^{(n)} \right) = \theta \left\{ \gamma^{(n)} - \alpha^{(n)} \right\}^T H \left\{ \gamma^{(n)} - \alpha^{(n)} \right\} + \nabla J_d(\alpha^{(n)})^T \left\{ \gamma^{(n)} - \alpha^{(n)} \right\}.$$

In the non degenerate case where $\gamma^{(n)} \neq \alpha^{(n)}$, the derivative considered is an affine function of θ with a positive slope. According to the definition of $\gamma^{(n)}$, its intercept is nonpositive. Thus, the single value for which the derivative is null is nonnegative. However, it can be superior to 1. In that case, the optimal value of θ on $[0,1]$ is simply 1. This finally gives us:

$$\theta^{(n)} = \min \left\{ -\frac{\nabla J_d(\alpha^{(n)})^T \left\{ \gamma^{(n)} - \alpha^{(n)} \right\}}{\left\{ \gamma^{(n)} - \alpha^{(n)} \right\}^T H \left\{ \gamma^{(n)} - \alpha^{(n)} \right\}}, \ 1 \right\}.$$

5.3 Applying a decomposition method to the algorithm

The main difficulty met when one tries to solve the dual problem, let it be with the Frank-Wolfe algorithm or other standard methods, rests in the handling of the Hessian matrix H. Equation (3) shows that its components are simple multiples of the components of the Gram matrix $K \in M_{m,m}(\mathbb{R})$, of general term $k_{ij} = \kappa(x_i, x_j)$. In the case of real-world problems, this latter matrix can be huge, to an extent that it can fail to fit in memory. Furthermore, its computation can be time consuming. A way to bypass this difficulty consists in applying to the algorithm chosen a *decomposition technique*. This approach was already applied in the early works dealing with SVMs, works exposed in [2]. The chunking method used was the one described in [8], Addendum I, in the particular case of a linear model. The main decomposition techniques introduced afterwards are all based on the solution of the dual problem in the case when the value of some of (most

of actually) the variables are fixed. Hereafter, this case is considered in a general way.

Without loss of generality, we assume that the optimization is performed with respect to the dual variables associated with the N_B first points in the training set, the dual variables associated with the $N_H = m - N_B$ last points being fixed. The objective function can then be decomposed as follows:

$$J_d(\alpha) = \frac{1}{2} \begin{pmatrix} \alpha_B \\ \alpha_H \end{pmatrix}^T \begin{pmatrix} H_{BB} & H_{BH} \\ H_{HB} & H_{HH} \end{pmatrix} \begin{pmatrix} \alpha_B \\ \alpha_H \end{pmatrix} - 1_{Qm}^T \begin{pmatrix} \alpha_B \\ \alpha_H \end{pmatrix}.$$

This functional can still be rewritten as:

$$J_d(\alpha) = \frac{1}{2}\alpha_B^T H_{BB} \alpha_B - (1_{QN_B}^T - \alpha_H^T H_{HB})\alpha_B + \frac{1}{2}\alpha_H^T H_{HH} \alpha_H - 1_{QN_H}^T \alpha_H.$$

We have thus:

$$\nabla J_d(\alpha_B) = H_{BB}\alpha_B + H_{BH}\alpha_H - 1_{QN_B} = \begin{pmatrix} H_{BB} & H_{BH} \end{pmatrix} \alpha - 1_{QN_B}.$$

It springs from this last equation that the expression of the gradient of the objective function of the dual problem with respect to the free variables is unchanged. In practice, in the framework of the implementation of the basic Frank-Wolfe algorithm, the Hessian matrix H or the Gram matrix K appear at three different levels:

1. initially, K is computed, which provides us with H;
2. at each iteration, the computation of the gradient makes use of the computation of $H\alpha^{(k)}$;
3. at each iteration, the computation of $\theta^{(k)}$ is based on the computation of
$$\left\{\gamma^{(n)} - \alpha^{(n)}\right\}^T H \left\{\gamma^{(n)} - \alpha^{(n)}\right\}.$$

With the implementation of a decomposition method, the computation of H_{HH}, or more precisely K_{HH}, becomes useless. The aforementioned stages are replaced with the following ones:

1. after each split of the training set, K_{BB} and K_{BH} are computed;
2. for a given split, the Frank-Wolfe algorithm requires one single initial computation of $H_{BH}\alpha_H$;

3. for a given split, at each iteration of the Frank-Wolfe algorithm, the computation of the gradient makes use of the computation of $H_{BB}\alpha_B^{(n)}$;

4. for a given split, at each iteration of the Frank-Wolfe algorithm, the computation of $\theta^{(n)}$ is based on the computation of $\left\{\gamma_B^{(n)} - \alpha_B^{(n)}\right\}^T H_{BB} \left\{\gamma_B^{(n)} - \alpha_B^{(n)}\right\}$.

One can readily point out the gain in terms of computation and memory requirements made at each step (update of vector α), gain which is balanced by the fact that the number of steps increases (it is *a priori* all the higher as the number of free variables is lower). A decomposition method is utterly specified by the algorithm computing the *working set* for the different iterations. Many such algorithms can be found in literature. The interested reader will find detailed overviews on the subject in Chapters 10, 11 and 12 of [7], as well as Chapter 7 of [3].

Acknowledgments Thanks are due to W. da Rocha for carefully reading this documentation.

References

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

[2] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT'92*, pages 144–152, 1992.

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

[4] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Res. Logist. Quart.*, 3:95–110, 1956.

[5] Y. Guermeur. VC theory of large margin multi-category classifiers. *Journal of Machine Learning Research*, 8:2551–2594, 2007.

[6] Y. Guermeur. A generic model of multi-class support vector machine. *International Journal of Intelligent Information and Database Systems*, 6(6):555–577, 2012.

[7] B. Schölkopf, J.C. Burges, and A. Smola, editors. *Advances in Kernel Methods, Support Vector Learning*. The MIT Press, Cambridge, 1999.

[8] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, N.Y, 1982.

[9] 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.