Symbolic computing of LS-SVM based models

S. Mehrkanoon, L. Jiang, C. Alzate, & J. A. K. Suykens *

Department of Electrical Engineering, K.U. Leuven, ESAT-SCD,
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium.
e-mail: Siamak.Mehrkanoon@esat.kuleuven.be

Abstract. This paper introduces a software tool *SYM-LS-SVM-SOLVER* written in Maple to derive the dual system and the dual model representation of LS-SVM based models, symbolically. *SYM-LS-SVM-SOLVER* constructs the Lagrangian from the given objective function and list of constraints. Afterwards it obtains the KKT (Karush-Kuhn-Tucker) optimality conditions and finally formulates a linear system in terms of the dual variables. The effectiveness of the developed solver is illustrated by applying it to a variety of problems involving LS-SVM based models.

1 Introduction

Support Vector Machines (SVMs) is a powerful methodology for solving pattern recognition and function estimation problems [1, 2]. In this method one maps the data into a high dimensional feature space and then constructs an optimal separating hyperplane in this space. It leads to solving quadratic programming problems [3]. Least squares support vector machines (LS-SVMs) on the other hand have been given by [4] for function estimation, classification, problems in unsupervised learning and others [5]. In this case, the problem formulation involves equality instead of inequality constraints.

LS-SVM core models are formulated in the primal in terms of high-dimensional feature maps, equality constraints and an L_2 loss function. In most cases, solving the primal problem directly is not possible due to the high dimensionality of the variables involved in the optimization problem. Through the constrained optimization framework, it is possible to obtain a dual system where the problem is recast in terms of kernel evaluations (the so-called kernel trick) and which grows with the number of data points [4]. Building the dual is a systematic process: first write the Lagrangian, then obtain the Karush-Kuhn-Tucker (KKT) optimality conditions and finally wrap up and formulate a system in terms of the dual variables that fulfills all KKT conditions. Fig. 1 shows an illustration of building models based upon LS-SVM core models; as outlined in [5].

2 Development of Symbolic Solver

In order to be able to work with a symbolic solver for LS-SVM model, at first the model should be transformed to the symbolic expressions i.e. in the matrix

*This work was supported by GOA/10/09 MaNet , CoE EF/05/006 (OPTEC), FWO: G0226.06, G.0302.07, G.0588.09, SBO POM, IUAP P6/04 (DYSCO, 2007-2011). Carlos Alzate is a postdoctoral fellow of the Research Foundation - Flanders (FWO). Johan Suykens is a professor at the K.U.Leuven, Belgium.

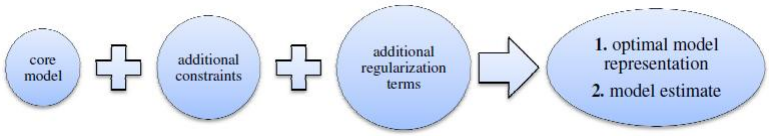

Fig. 1: Illustration of advanced LS-SVM models.

or vector notation. It should be noted that this stage is done by the user before utilizing the symbolic solver. An example is provided to clarify this procedure.

Let us consider a given training set $\{x_i, y_i\}_{i=1}^{N}$ with input data $x_i \in \mathbb{R}^d$ and output data $y_i \in \{-1, 1\}$. The LS-SVM model for classification [5], can be rewritten in a matrix form as follows

$$\begin{aligned} \underset{w,b,e}{\text{minimize}} \quad & \frac{1}{2}w^T w + \frac{\gamma}{2}e^T e \\ \text{subject to} \quad & Y\Big[\Phi w + b1_N\Big] = 1_N - e \end{aligned} \tag{1}$$

where $\gamma \in \mathbb{R}^+, b \in \mathbb{R}$, $e \in \mathbb{R}^N$, $w \in \mathbb{R}^h$, $Y = \text{diag}(y_1, y_2, \ldots, y_N) \in \mathbb{R}^{N\times N}$, $1_N \in \mathbb{R}^N$, $\Phi \in \mathbb{R}^{N\times h}$ with

$$\Phi = \big[\phi(x_1)\cdots\phi(x_N)\big]^T,$$

$\phi(\cdot) : \mathbb{R}^d \longrightarrow \mathbb{R}^h$ is the feature map and h is the dimension of the feature space.

The approach on which the LS-SVM solver is based can be summarized as follows: (1) constructing the Lagrangian, (2) taking derivatives of the Lagrangian with respect to the primal and dual variables and setting them equal to zero, (3) elimination of primal variables (or part of it), (4) expressing the solution in terms of the Lagrange multipliers, (5) obtaining the dual representation of the model. The Maplet of the code is designed, (see Fig. 2), containing windows, textbox regions and other visual interfaces, which gives the user point-and-click access. It is an alternative to the worksheet. Users can perform the *SYM-LS-SVM-SOLVER* Package without having to get involved in the Maple syntax.

3 *SYM-LS-SVM-SOLVER* Package

A specific module, denoted by *SYM_LS_SVM_SOLVER*, is designed for the symbolic solver for LS-SVMs. This module is composed of four main procedures denoted by *Pro_Lag*, *Pro_KKT*, *Pro_Dual system* and *Pro_Dual Model*.

```
> print(SYM_LS_SVM_SOLVER);
```

module()

export Pro_Lag, Pro_KKT, Pro_Dual System, Pro_Dual Model;

end module

More details of these procedures are discussed in the following subsections.

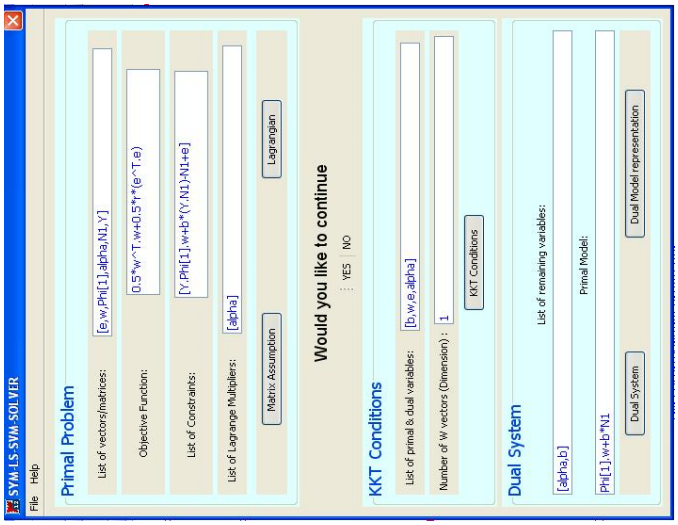

Fig. 2: The GUI for SYM-LS-SVM-SOLVER

3.1 Procedure *Pro-Lag*

The aim of this procedure is to form the Lagrangian from a given primal problem. The arguments of the Pro_Lag procedure are thus the objective function, list of constraints and Lagrange multipliers, respectively. It should be noticed that in our code the vectors are considered as a special case of matrices. Also the possibility that the users can define the type of the matrix is provided.

Example 1. Consider the LS-SVM model (1). One initially reads the package into memory using the *'with'* command. A second task is to utilize the *'assume'* command to specify the matrix variables. If the variable has additional properties such as being symmetric or positive definite, the *additionally* function can be used which adds additional assumptions without removing previous assumptions.

```
> with(SYM_LS_SVM_SOLVER);
> assume(w::Matrix,e::Matrix,Phi::Matrix,
> N1::Matrix,alpha::Matrix,Y::Matrix),additionally(Y::symmetric);
> L[1]:=Pro_Lag(0.5*w^T.w+0.5*gamma*(e^T.e),
> [Y.Phi.w+b*(Y.N1)=N1-e],[alpha]);
```

$$\mathscr{L}_1 = 0.5\, w^T w + 0.5\, \gamma e^T e - \alpha^T \cdot Y \cdot \Phi \cdot w - b\alpha^T \cdot Y \cdot N1 + \alpha^T \cdot N1 - \alpha^T \cdot e$$

Note that *N1* is a vector of all ones and equals 1_N.

Example 2. Consider the following problem,

$$\begin{aligned} \underset{w,b,e,\widehat{Y}}{\text{minimize}} \quad & \frac{1}{2} w^T w + \gamma e^T e + \eta (\widehat{Y} - Y^*)^T (\widehat{Y} - Y^*) \\ \text{subject to} \quad & Y - \widehat{Y} = e \\ & \widehat{Y} = \Phi w + b 1_N \end{aligned} \tag{2}$$

```
> assume(e::Matrix,w::Matrix,
> Y::Matrix,Yhat::Matrix,alpha[1]::Matrix,alpha[2]::Matrix,
> Phi::Matrix,N1::Matrix, Ystr::Matrix);
> L[2]:=Pro_Lag(0.5*(w^T.w)+gamma*(e^T.e)+eta*((Yhat-Ystr)^T.
> (Yhat-Ystr)),[Y-Yhat-e,Yhat-Phi[1].w-b*N1],[alpha[1],alpha[2]]);
```

$$\mathscr{L}_2 = \frac{1}{2} w^T w + \gamma\, e^T e + \eta\, (Yhat - Ystr)^T \cdot (Yhat - Ystr) + {\alpha_1}^T \cdot Y - {\alpha_1}^T \cdot Yhat - {\alpha_1}^T \cdot e + {\alpha_2}^T \cdot Yhat - {\alpha_2}^T \cdot \Phi \cdot w - b{\alpha_2}^T \cdot N1$$

Example 3. As another example, we consider the data visualization model, see ([6]),

```
> with(SYM_LS_SVM_SOLVER);
> assume(z::Matrix,N1::Matrix,P[D]::Matrix);
> dims:=2;
> for k from 1 to dims do
> assume(w[k]::Matrix,e[k]::Matrix,Phi[k]::Matrix,v[k]::Matrix,
> alpha[k]::Matrix,C[k]::Matrix,M[k]::Matrix,Omega[k]::Matrix,
> beta[1,k]::Matrix,e[1,k]::Matrix); end do;
> L[3]:=Pro_Lag(-0.5*gamma*z^T.z+0.5*(z-P[D].z)^T.(z-P[D].z)+
> (gamma/2)*(sum(w[j]^T.w[j],j=1..dims))+0.5*eta*(sum(e[j]^T.e[j],
> j=1..dims)),[seq(v[j]^T.z-Phi[j].w[j]-b[j]*N1=e[j],j=1..dims),
> seq(C[j]^T.z=q[j]+e[1,j],j=1..dims)],
> [seq(alpha[j],j=1..dims),seq(beta[1,j],j=1..dims)]);
```

$$\mathscr{L}_3 = -0.5\,\gamma\, z^T z + 0.5\,(z - P_D \cdot z)^T \cdot (z - P_D \cdot z) + 0.5\,\gamma\,\left({w_1}^T w_1 + {w_2}^T w_2\right) + 0.5\,\eta\,\left({e_1}^T e_1 + {e_2}^T e_2\right) + {\alpha_1}^T \cdot {v_1}^T \cdot z - {\alpha_1}^T \cdot \Phi_1 \cdot w_1 - b_1 {\alpha_1}^T \cdot N1 - {\alpha_1}^T \cdot e_1 + {\alpha_2}^T \cdot {v_2}^T \cdot z - {\alpha_2}^T \cdot \Phi_2 \cdot w_2 - b_2 {\alpha_2}^T \cdot N1 - {\alpha_2}^T \cdot e_2 + {\beta_{1,1}}^T \cdot {C_1}^T \cdot z - {\beta_{1,1}}^T \cdot q_1 - {\beta_{1,1}}^T \cdot e_{1,1} + {\beta_{1,2}}^T \cdot {C_2}^T \cdot z - {\beta_{1,2}}^T \cdot q_2 - {\beta_{1,2}}^T \cdot e_{1,2}$$

3.2 Procedure *Pro-KKT*

After obtaining the Lagrangian, the task is to take derivatives of this function with respect to the primal variables and Lagrange multipliers. In our code, Procedure *Pro-KKT* sets the derivatives of the Lagrangian to zero which leads to the system of linear equations.

The built-in differentiator in Maple (i.e *diff* command) is not able to handle the derivative with respect to a vector or matrix (of known dimension, but unknown values). Therefore a special procedure so called *Pro_DIFF* is designed to do differential operations on generalized matrices symbolically, under the framework of LS-SVMs. *Pro_DIFF* has two parameters, the algebraic expression that has to be differentiated and differentiation variable respectively.

Most cases encountered when solving LS-SVMs are as follows,

$$\frac{\partial X^T A}{\partial X} = \frac{\partial A^T X}{\partial X} = A, \quad \frac{\partial A^T X B}{\partial X} = AB^T, \quad \frac{\partial X^T X}{\partial X} = 2X, \quad \frac{\partial X^T A X}{\partial X} = (A + A^T)X$$

Where A, B, X are symbols for matrices. For more details we refer to [7]. Having the Lagrangian function available from the *Pro_Lag*, we can call the

function *Pro_KKT* to generate the KKT optimality conditions. The parameters of *Pro_KKT* are thus the Lagrangian, list of differentiation variables and number of w vectors (the dimension of the problem) respectively. In order to illustrate the procedure we apply it to the example 2 and 3 of section 3.1, thus the KKT optimality conditions are as follows,

For example 2,

```
> Pro_KKT(L[2],[w,e,alpha[1],alpha[2],Yhat,b],1);
```

$$\frac{\partial L_2}{\partial w} = 2\,w - \Phi^T \cdot \alpha_2 = 0, \frac{\partial L_2}{\partial e} = 2\,\gamma\,e - \alpha_1 = 0, \frac{\partial L_2}{\partial \alpha[1]} = Y - Yhat - e = 0,$$

$$\frac{\partial L_2}{\partial \alpha[2]} = Yhat - \Phi \cdot w - bN1 = 0, \frac{\partial L_2}{\partial \widehat{Y}} = 2\,\eta\,Yhat - 2\,\eta\,Ystr - \alpha_1 + \alpha_2 = 0,$$

$$\frac{\partial L_2}{\partial b} = -{N1}^T \cdot \alpha_2 = 0.$$

For example 3,

```
> Pro_KKT(L[3],[seq(w[i],i=1..dims),seq(e[i],i=1..dims),
> seq(e[1,i],i=1..dims),seq(alpha[i],i=1..dims),
> seq(beta[1,i],i=1..dims),seq(b[i],i=1..dims),z],2);
```

$$\frac{\partial L_3}{\partial w_1} = \gamma\,w_1 - {\Phi_1}^T \cdot \alpha_1 = 0, \frac{\partial L_3}{\partial w_2} = \gamma\,w_2 - {\Phi_2}^T \cdot \alpha_2 = 0, \frac{\partial L_3}{\partial e_1} = 1.0\,\eta\,e_1 - \alpha_1 = 0,$$

$$\frac{\partial L_3}{\partial e_2} = 1.0\,\eta\,e_2 - \alpha_2 = 0, \frac{\partial L_3}{\partial e_{1,1}} = -\beta_{1,1} + 1.0\,\eta\,e_{1,1} = 0 \frac{\partial L_3}{\partial e_{1,2}} = -\beta_{1,2} + 1.0\,\eta\,e_{1,2} = 0,$$

$$\frac{\partial L_3}{\partial \alpha_1} = {v_1}^T \cdot z - \Phi_1 \cdot w_1 - b_1 N1 - e_1 = 0, \frac{\partial L_3}{\partial \alpha_2} = {v_2}^T \cdot z - \Phi_2 \cdot w_2 - b_2 N1 - e_2 = 0,$$

$$\frac{\partial L_3}{\partial \beta_{1,1}} = {C_1}^T \cdot z - q_1 - e_{1,1} = 0, \frac{\partial L_3}{\partial \beta_{1,2}} = {C_2}^T \cdot z - q_2 - e_{1,2} = 0, \frac{\partial L_3}{\partial b_1} = -{N1}^T \cdot \alpha_1 = 0,$$

$$\frac{\partial L_3}{\partial z} = -1.0\,\gamma\,z + 1.0\ (I - P_D)^T \cdot (I - P_D) \cdot z + v_1 \cdot \alpha_1 + v_2 \cdot \alpha_2 + C_1 \cdot \beta_{1,1} +$$

$$C_2 \cdot \beta_{1,2} = 0, \ \frac{\partial L_3}{\partial b_2} = -{N1}^T \cdot \alpha_2 = 0.$$

3.3 Procedure *Pro-Dual System*

The procedure *Pro-Dual System*, as its name suggests, will produce the corresponding dual system for the given primal problem. The remaining variables are defined by the user. *Pro-Dual System* has four parameters, Lagrangian, differentiation variables, remaining variables and number of w vectors, respectively. In what follows, we illustrate this procedure by applying it to the example 2 of section 3.1.

For example 2, we have

```
> Pro_Dualsystem(L[2],[w,e,alpha[1],alpha[2],Yhat,b],[alpha[2],b],1);
```

$$G1 \cdot \begin{bmatrix} \alpha_2 \\ b \end{bmatrix} = \begin{bmatrix} 2\,Y\gamma + 2\,\eta\ Ystr \\ 0 \end{bmatrix},$$

$$\texttt{`where G1`} = \begin{bmatrix} -\gamma\,\Omega \cdot I_N - I_N - \eta\,\Omega \cdot I_N & 2\,\gamma\,I_N N1 + 2\,\eta\,I_N N1 \\ {N1}^T \cdot I_N & 0 \end{bmatrix}$$

where Ω=$\Phi\Phi^T$ denotes the $N \times N$ kernel matrix.

3.4 Procedure *Pro_Dual Model*

The last procedure denoted by *Pro_Dual Model*, constructs the dual model representation. The input of this procedure is just the primal model provided by the user. Implementing this procedure for the examples 2 and 3 of section 3.1 will result in the following model expressions.

For example 2,

```
> Pro_DualModel(Phi.w+b*N1);
```

$$\frac{1}{2}\,\Phi\Phi^T \cdot \alpha_2 + bN1$$

For example 3,

```
> Pro_DualModel([Phi[1].w[1]+b[1]*N1,Phi[2].w[2]+b[2]*N1]);
```

$$\frac{\Phi_1{\Phi_1}^T \cdot ({M_1}^{-1} \cdot {v_1}^T \cdot z - {M_1}^{-1} \cdot b_1 N1)}{\gamma} + b_1 N1,\ \frac{\Phi_2{\Phi_2}^T \cdot ({M_2}^{-1} \cdot {v_2}^T \cdot z - {M_2}^{-1} \cdot b_2 N1)}{\gamma} + b_2 N1$$

where

$$M_1 = \Phi_1{\Phi_1}^T + \frac{I}{\eta},\ M_2 = \Phi_2{\Phi_2}^T + \frac{I}{\eta}.$$

4 Conclusion and future work

A symbolic solver written in Maple is developed for LS-SVM models. The Maplet of our code is also provided as an alternative to the worksheet. The application of the solver is illustrated on three examples. Currently the LS-SVM models that can be handled in our symbolic solver include equality constraints only. Dealing with additional inequality constraints is a further challenge for future work.

References

[1] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA (2002).

[2] V. Vapnik, *Statistical learning theory*, New York: Wiley (1998).

[3] J. A. K. Suykens, & J. Vandewalle, Least squares support vector machine classifiers. *Neural Processing Letters*, 9 (3), 293-300 (1999).

[4] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific Pub. Co., Singapore, 2002.

[5] J.A.K. Suykens, C. Alzate, K. Pelckmans, Primal and dual model representations in kernel-based learning, *Statistics Surveys*, DOI: 10.1214/09-SS052, vol. 4, Aug. 2010, pp. 148-183.

[6] J.A.K. Suykens, Data Visualization and Dimensionality Reduction using Kernel Maps with a Reference Point, *IEEE Transactions on Neural Networks*, vol. 19, no. 9, Sep. 2008, pp. 1501-1517.

[7] K.B. Petersen, M.S. Pedersen , *The Matrix Cookbook* , [http://matrixcookbook.com]